The Thesis committee for Adam Jess Kirk

certifies that this is the approved version of the following thesis:

Collapse Investigation of the TU Delft Faculty of Architecture Building:

Preliminary Evaluation of Member Capacities

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor: _____

Michael Engelhardt

_____

Wassim Ghannoum

Collapse Investigation of the TU Delft Faculty of Architecture Building: Preliminary Evaluation of Member Capacities

by

Adam Jess Kirk, B.A.E.

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

August 2010

This thesis is dedicated to my parents, sister, and the love of my life, Andrea. It would not have been possible without your unconditional love and support.

Acknowledgements

First and foremost, I would like to thank Dr. Engelhardt for allowing me the opportunity to participate in this investigation. It has been a truly rewarding experience. Your guidance, wisdom, and most of all, patience, has been greatly appreciated.

Thanks are also due to Dalin Mo, our undergraduate assistant whose hours spent pouring through the thousands of photographs taken of the Faculty of Architecture building were not enviable, but very much appreciated.

To the staff of the Ferguson Lab and faculty of the University of Texas Structural Engineering Department, thank you so much for your commitment to me and my fellow students. I will do my best to represent you well and carry the knowledge you provided with me as I leave the academic world and begin my professional engineering career.

Finally, to my friends and family, your support and encouragement were invaluable in keeping me motivated through this process. You are the best. Thanks again and I love you all.

August 2010

Collapse Investigation of the TU Delft Faculty of Architecture: Preliminary Evaluation of Member Capacities

Adam Jess Kirk, M.S.E.

The University of Texas at Austin, 2010

Supervisor: Michael Engelhardt

Co-Supervisor: Wassim Ghannoum

On May 13, 2008, the Faculty of Architecture Building, or 'Bouwkunde', at the Delft University of Technology, Netherlands suffered a fire that resulted in the collapse of the northwest wing of the 13 story building. No one was injured but the building was a complete loss.

Collapse of concrete buildings in fire is rare; this report aims to provide a preliminary evaluation of the structure and point to key areas that may be of interest to future analyses and investigations. To this end, a large database of information was collected, including original and renovation construction documents, original structural calculations, and over 3000 photographs of the structure during and after the fire. This data has been organized and reviewed to provide a clearer understanding of the building and fire.

Preliminary models of the fire are developed and applied to selected structural elements in the FOA to the temperature distributions within the members. Also provided is an overview of available methods for calculating the ultimate strength of reinforced concrete members at elevated temperatures and a computer application, UT Fire: Reinforced Concrete Analysis, which can be used to estimate member capacities through a given fire event, based on their internal temperature distributions.

Table of Contents

List of Tables

List of Figures

CHAPTER 1
Introduction

1.1 OVERVIEW

There is a growing interest in the US and worldwide in transforming the design of buildings for structural fire safety from a prescriptive to a performance based approach. Developing performance based fire safety design approaches requires an understanding of how fires move through large buildings and the behavior of structures at elevated temperatures. This project is a case study on the behavior of a reinforced concrete building in fire intended to contribute to a better understanding of the response of real buildings to real fires and thereby contribute to the advancement of performance based structural fire safety.

On May 13, 2008, a fire occurred at the Faculty of Architecture Building (FOA), or 'Bouwkunde' in Dutch, at the Delft University of Technology (TUD), Netherlands. Figure 1.1 is a photograph of the FOA shortly after completion of construction in about 1970. The fire started in a coffee vending machine on the 6^{th} floor of this 13 story reinforced concrete building, spread rapidly, and ultimately led to collapse of a major portion of the building. No one was injured but the building was a complete loss.

Collapse of reinforced concrete buildings in fire is rare. This event therefore represents a unique opportunity to study a fire induced structural collapse in a major reinforced concrete building. There are important lessons to be learned from this fire in the area of structural-fire safety. It is important to understand the factors that led to the collapse, and the implications these factors have on our understanding and practices for assessing the structural fire resistance of reinforced concrete structures. Although this building was located in the Netherlands, the design features of this structure are typical of reinforced concrete construction throughout the world, including the US. Ultimately, studying this fire event and the causes of structural collapse will lead to an improved

understanding of the performance of structures in major fire events, and will contribute to improved methods of analysis and design for structural-fire safety.

This thesis aims to provide a preliminary evaluation of the structure and its response to the fire. This thesis also identifies key areas that may be of interest to future analyses and investigations of this fire event. To this end, a large database of information was collected, including original and renovation construction documents, original structural calculations, and over 4000 photographs of the structure before, during, and after the fire. This data has been organized and reviewed to provide an understanding of the building and fire. Also provided is an overview of available methods for calculating the ultimate strength of reinforced concrete members at elevated temperatures and a computer application, UT Fire: Reinforced Concrete Analysis, which can be used to estimate member capacities through a given fire event.

1.2 IMPORTANCE OF RESEARCH

Structural fire engineering is a developing field. Much research has been conducted on the behavior of structural members at elevated temperatures. However, less study has been done to provide for the development of relatively accurate design fires based on the building occupancy, layout, and size. Modern buildings are built with increasingly open floor plans. This is especially true of institutional buildings which have large occupancies. Historically, design fires have been based on well defined compartments of single rooms or series of rooms bound by firewalls. When the space becomes much larger, the behavior of the fire can be much different than would be predicted by these compartmentalized models. Computational Fluid Dynamics (CFD) models are available using software such as Fire Dynamics Simulator, FDS (McGrattan, McDermott, Hostikka and Floyd, 2010), but these models are very time consuming and are generally not suitable for routine design.

The fire that will be the focus of this study was very well documented photographically. This allowed for an examination of the movement of the fire through

the building (Chapter 3). It is the goal of this portion of the study to further the knowledge of how fires move through large buildings with complex floor plans and what generalizations can be made that may lead to improved models for design fires in modern buildings.

Figure 1.1 The FOA and Surroundings Shortly After Construction (TU Delft 2008)

In addition, it has long been the assumption that structures constructed of reinforced concrete are intrinsically safe against collapse in fires. Large deformations, spalling, and localized failures could be expected, but a progressive collapse initiated by a fire is a rare event. The FOA was by all accounts a well designed reinforced concrete building. It was designed according to the governing code standards of the time and stood for 38 years with no major incident prior to the fire of May 13, 2008. This study began with a cold analysis of the structure, under loads to be expected at the time of the fire. It was found that the structure was conservatively designed and had what appears a considerable amount of reserve strength to withstand the effects of thermally induced

stresses and strains. However, it must also be understood that structural members are rarely isolated from each other and are meant to act in accord with other elements in the system. It is this system level approach that is neglected in current prescriptive design codes for structural fire safety.

It would be overly ambitious to hope for one case study to shed light on all the questions that arise when considering the response of a major structure to fire. However the amount of information available on the FOA and on the fire of May 13, 2008 is substantial, and gives this case study the potential to provide a great deal of information useful for the advancement of structural fire engineering.

1.3 GOALS OF INVESTIGATION AND OF THESIS

This thesis represents part of a larger effort to collect data and to study the fire at the FOA. Shortly after the fire on May 13, 2008, an international team was assembled to collect data and conduct preliminary analysis of the fire event. The team included researchers from both the Netherlands and from the US. Participating organizations from the Netherlands were TNO, which is the Netherlands Organization for Applied Scientific Research; Efectis, a consulting firm specializing in fire related problems, and the TUD Faculty of Architecture. On the US side, the participating institutions included Worcester Polytechnic Institute, Michigan State University, and the University of Texas at Austin. Further information on this research team and activities on this fire investigation are reported by Meacham et al (2010). Issues of interest to the research team and to the overall investigation were broad, and included not only studies of the structural collapse, but also included studies on the initiation and initial growth of the fire, and spread of the fire beyond the compartment of origin to other portions of the same floor and other floors. Also of interest were performance of fire rated barriers in the building, performance of fire alarm systems, behavior of building occupants during the fire, and fire department response.

The work reported in this thesis is part of the overall fire investigation by the team described above, with a focus on the performance of the structure. The ultimate goal of this portion of the overall investigation is to understand the causes of the structural collapse. The overall objective of this thesis is to conduct preliminary observations and analysis to contribute to subsequent detailed studies of the structural collapse. It is important to note that it is not the objective of this thesis to conduct a detailed analysis of the collapse, but rather to contribute to more detailed and more comprehensive subsequent studies of the collapse. More specifically, items that will be addressed in this thesis are as follows:

1) Compile and organize all available resources describing the building and fire.
2) Conduct preliminary analysis of structural system and individual structural members.
3) Conduct a preliminary study of photographic evidence for information pertinent to structural performance.
4) Develop a timeline of the fire event and collapse sequence.
5) Identify the most critical structural members for further study.
6) Estimate the thermal environment to which the structure was exposed during the fire.
7) Conduct heat transfer analysis for critical members.
8) Develop tools for evaluating member capacities at elevated temperatures.
9) Utilize these tools to estimate critical member capacities.
10) Identify potential contributing factors to the structural collapse and suggest topics for future detailed studies of this collapse.

1.4 DESCRIPTION OF THE BUILDING

The architectural design of the FOA was done by the firm of Van den Broek & Bakema located in Rotterdam, Netherlands. The structural design was headed by Bakker

& Dicke Civil Ingenieurs, Amsterdam. The design documents recovered are dated from the mid 1960's and construction was completed in 1970. A rendering of the orginal design by the architect is shown in Figure 1.2. Some of the history of the building, along with many personal observations of the fire of May 13, 2008 is documented in the publication: "Bouwkunde – Portrait of the Faculty of Architecture of Delft University of Technology 1970-2008" (TU Delft 2008), which was published shortly after the fire as a remembrance of the building and the fire. Figures 1.1 and 1.2 are taken from this publication.

Figure 1.2 Drawing of the Bouwkunde by Van den Broek & Bakema (TU Delft 2008)

The FOA was primarily a reinforced concrete building, consisting of a 13 story tower, which sat on top of a series of six semi-independent three-story structures. This base structure effectively served as a podium for the tower above. The first floor of the podium was below ground with the second and third floors above grade. It housed public use spaces, such as the library, an auditorium, exhibition halls, and a cafeteria. The fire of May 13 did very little damage to these lower floors.

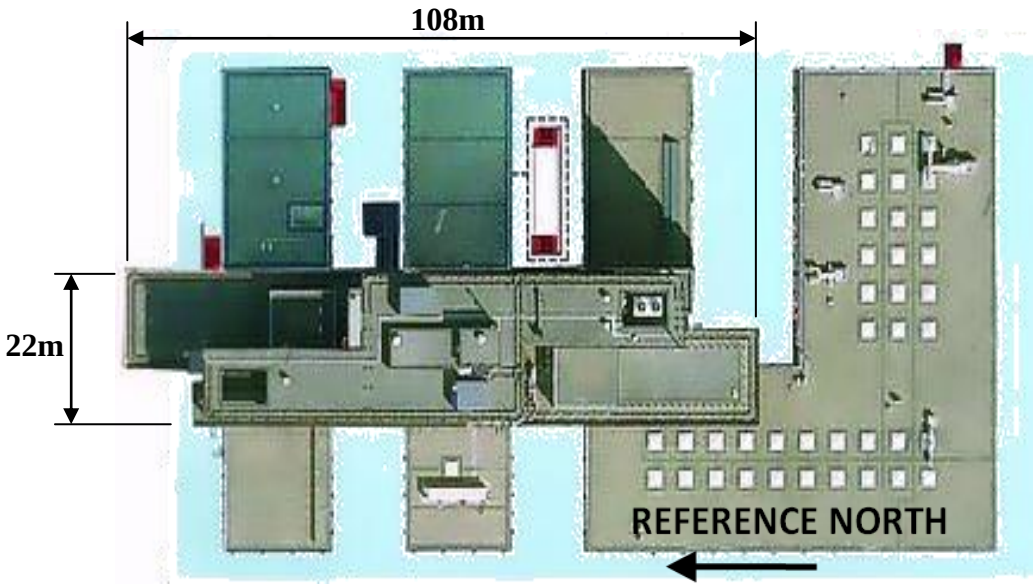


Figure 1.3 Roof Plan Rendering of the FOA (TU Delft 2008)

The focus of this investigation will be on the tower portion, as this is where the fire and collapse both initiated. The tower was characterized by its long, narrow floor plan and large windows that allowed natural lighting into the building and dominated its facade's appearance.

The tower was nominally 108m long by 22m wide and can be looked at as four interconnected wings. The occupancies of the wings are shown in Figure 1.4. The southeast and northwest wings housed the studio spaces and were identical with double story heights. The double height studios were located on even floors and a mezzanine supported by separate structural system hung over the studio spaces on the odd floors. The mezzanines were wooden structures hung by steel supports from the main concrete structure and provided additional space for offices and lecture halls. They can be seen in context with the studio spaces in Figure 1.5. The southeast and northwest wings housed the faculty offices and were single height stories at every floor.

Each wing was separated by a series of corridors which housed the elevators, stairwells and other service spaces. The stairwells were constructed of concrete and were enclosed by glass. This is illustrated in Figure 1.6.

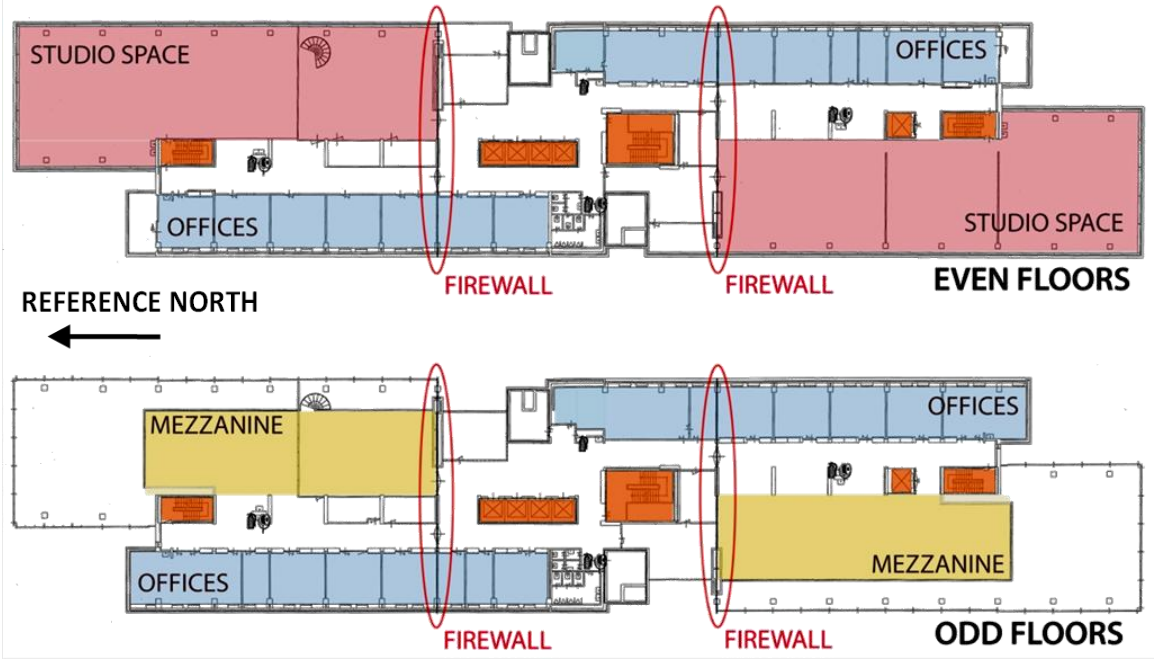


Figure 1.4 Tower Portion of the FOA by Occupancy

As can be seen in Figure 1.4, the north and south wings of the building were reversed mirror images of each other. They were identical to the point that the same drawings were used to detail much of each side. This raises the question of why it was only the northwest wing that collapsed.

Figure 1.5 Studio Space Under Mezzanine

Figure 1.6 Stairwell

The structure of the FOA generally consisted of floor joists and girders supported by perimeter columns and interior walls. The continuous bands of windows on the facade were possible due to the placement of the exterior columns just inside the outer curtain wall. The walls ran down the center spine of the plan and were integrated into the elevator shafts and stairwells where these service spaces were needed.

The studio wings of the FOA (NE and SW) were 12 stories tall, while the office wings (NW and SE) continued to 14 stories. Mechanical equipment rooms were located on both the lower roofs of the studio wings and the upper roofs of the office wings. The first floor of the tower has been designated 'Floor 2' in this thesis. The full floor numbering scheme can be seen in Figures 1.8-10. Figure 1.10 also shows the typical floor to floor heights, though the actual useable heights were slightly less due to the suspended ceiling system. This left the studio spaces with a height of 5.85m and the offices 2.70m. The interstitial space above the suspended ceilings hid the majority of the structural and mechanical systems.

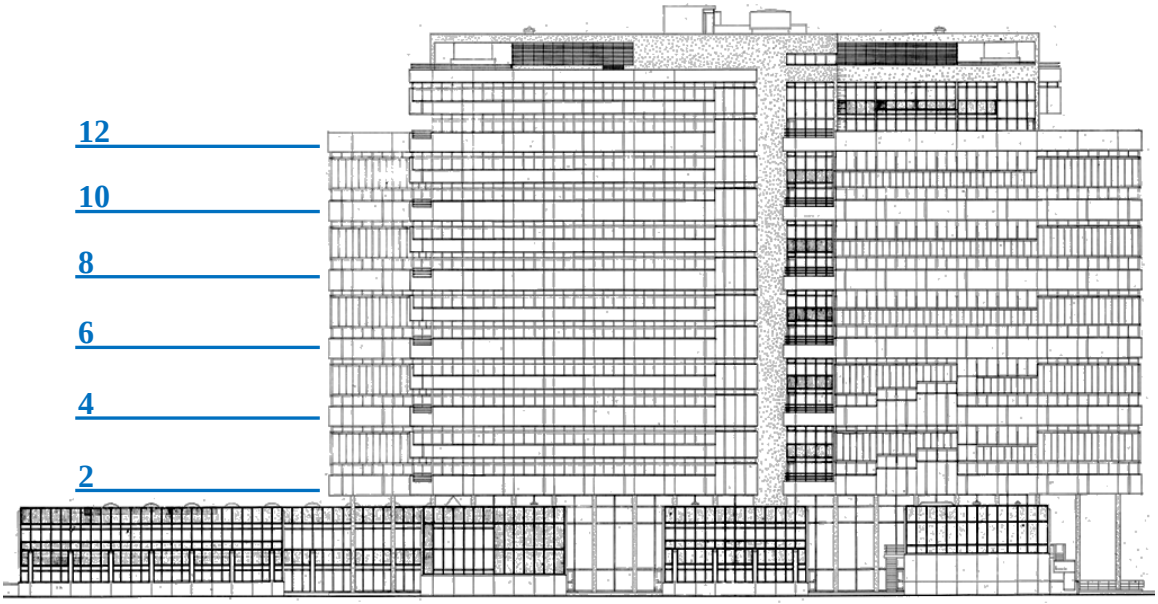


Figure 1.7 East Elevation

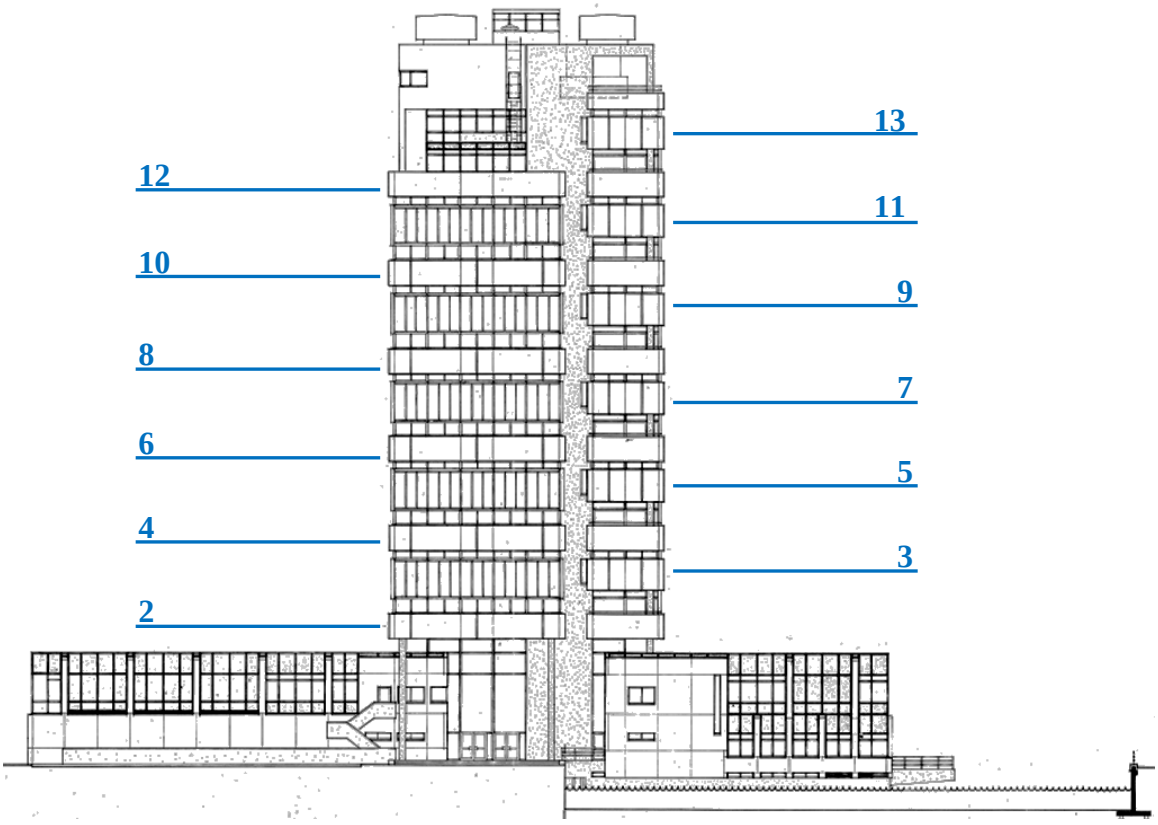


Figure 1.8 North Elevation

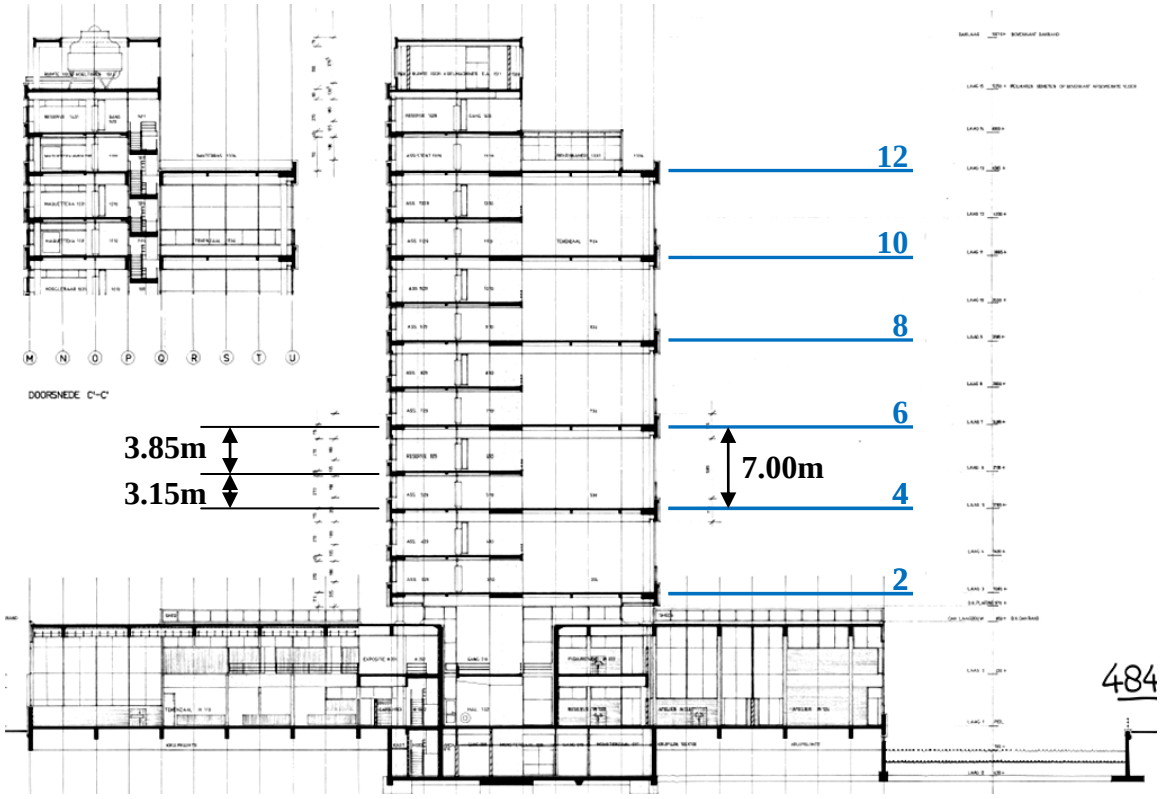


Figure 1.9 FOA Cross-Section

1.5 DESCRIPTION OF THE FIRE

The fire on May 13, 2008 began at approximately 9:00am when a water leak caused a coffee vending machine to spark, smoke, then finally flame. The coffee machine was located on the southwest wing of the 6th floor. The fire quickly spread and the building was evacuated. The FOA did not have fire sprinklers, but there were passive fire protection elements in place. Firewalls (shown in Figure 1.4 in red) divided the tower into three separate compartments, but these were not very effective in confining the fire to the compartment of origin. The fire first spread across the entire 6th floor, then upwards and throughout the entire building. The northwest office wing collapsed at 4:40 pm, approximately 7.5 hours after the first flames were observed and, interestingly, after the fire in that wing of the building was largely extinguished. Figure 1.10 is a photo of

the fire in the northwest wing of the building, and Figure 1.11 is a photo after the collapse of the northwest wing.

When the fire began, the building was successfully evacuated. No one was seriously injured in the fire or collapse, but the building was a complete loss and was later demolished. A detailed account of the fires based on eye witness testimony, observations from photos, and video evidence was obtained by the research team (Meacham et al 2010).

Figure 1.10 NW Wing During the Fire (©Rob Jastrebski)

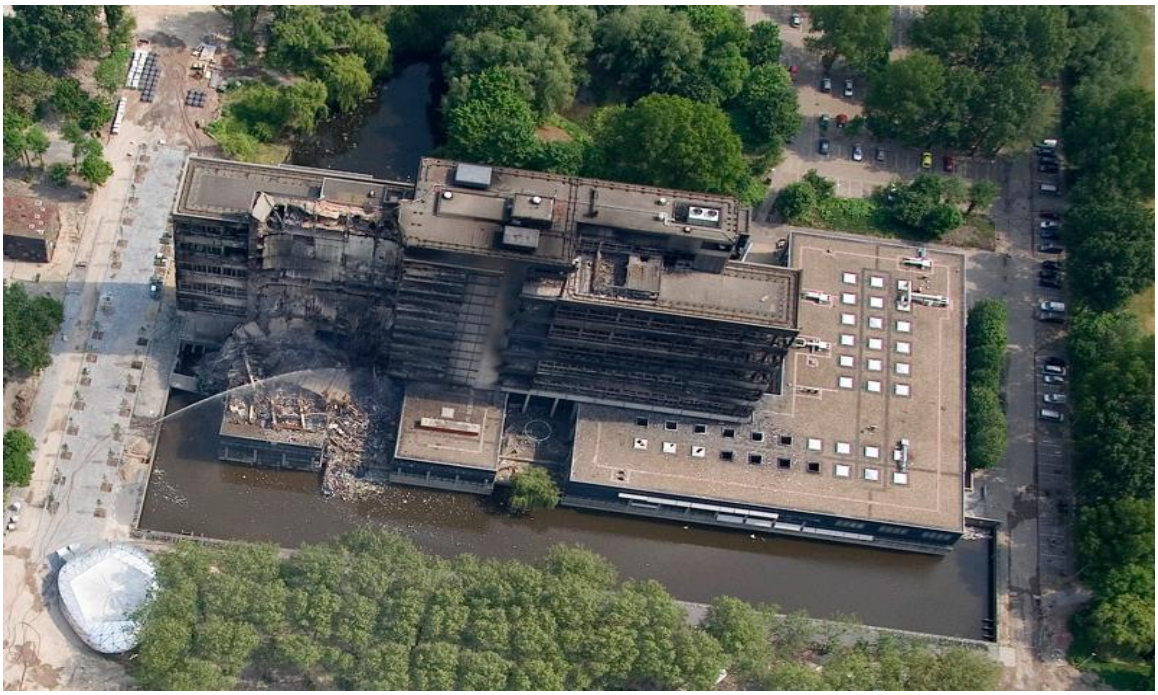


Figure 1.11 Aerial View of the Bouwkunde After the Collapse of the NW Wing

1.6 SCOPE & OUTLINE OF THESIS

This thesis has two major objectives. The first objective is the collection and organization of data on the Faculty of Architecture fire and building itself. The second objective is to conduct a preliminary thermal structural analysis of a few key reinforced concrete members located in the structure and provide some preliminary observations of possible causes of the collapse. Both objectives are intended to support subsequent more detailed studies of the fire and structural collapse.

In evaluating possible causes of collapse, several contributing factors must be examined. These include the effect of thermally induced forces and deformations of the structural system, the effect of material strength and member capacity degradation at elevated temperature, and the effects of spalling. As part of this thesis, several preliminary analyses have been conducted to gain insight into the impact of the fire on the capacity of key structural members. This includes a simplified compartment fire

analysis, heat transfer analysis for selected structural members, and cross-section capacity calculations for the heated members.

The results and detailed descriptions of these analyses will be presented in the same order as they were conducted. Chapter 2 provides a more detailed description of the structural system of the FOA. Chapter 3 examines the photos taken of the FOA before, during, and after the fire. In Chapter 4, an analytical model of the fire is developed and Chapter 5 uses this model to estimate the temperature distribution within a few key members through the fire. Chapter 6 discusses the development of a computer program, UT Fire: Reinforced Concrete Analysis, that uses the temperature distribution of a reinforced concrete section to calculate its reduced ultimate strength. In Chapter 7 this software is used to analyze the key members of the FOA through the fire modeled in Chapter 4. The final chapter will summarize the results of these studies and point to key areas that should be examined in future investigations. In the next chapter, we will take a close look at the structural system of the Faculty of Architecture Building at the Delft University of Technology.

CHAPTER 2
Description of the Structural System

2.1 OVERVIEW

This chapter provides a description of the structural system and key structural elements of the Faculty of Architecture Building (FOA). The focus of this discussion is on the tower portion of the building, where the fire and collapse occurred. This chapter begins with a description of the sources of information available to the research team on the structural system and the structural design. This is followed by a general description of the overall structural system of the FOA tower. Further details are then provided for key elements of the tower structure, including the columns, walls, girders and floor joists. A brief comparison is then provided between the structural details of the FOA and current US requirements for reinforced concrete structures as provided in ACI 318-08 (ACI 2008).

2.2 SOURCES OF INFORMATION

The Facilities Management Department at the Delft University of Technology (TUD) provided the research team with 485 original construction documents. These were a nearly complete set and included architectural, civil, structural, mechanical, and plumbing drawings. Also provided were the original structural design calculations. An initial step in this investigation was organizing these drawings and calculation sheets by type and studying them to ensure a complete understanding of the building. A list of the available original design drawings on record is provided in Appendix A.

The TUD Facilities Management Department also provided a series of floor plans for the tower that was prepared in 2006 for a fire safety retrofit of the tower. The fire safety retrofit included the addition of 30-minute rated fire partitions at the locations

shown in Figure 1.4. The retrofit also included installation of fire hoses and fire extinguishers.

The information on the structural system presented in this chapter comes from a review of the drawings and design documents noted above. Further information and understanding of the structural system was also developed by studying photos of the building before, during and after the fire.

One of the early challenges in the investigation was overcoming the fairly significant language barrier. All of the original drawings and associated calculation sheets are in Dutch and the units used are a version of SI that was common in 1960s Europe. A table of units used and common conversions is provided in Table 2.1. Table 2.2 shows some common reinforcing bar sizes, as they are labeled in the drawings. Common translations that were used in interpreting the drawings and calculations are provided in Appendix B.

Table 2.1 Common Unit Conversions

	Bouwkunde	**SI**	**US**
Length	1 mm	1 mm	0.0394 in
	1 m	1 m	3.281 ft
Area	1 mm^2	1 mm^2	0.0016 in^2
	1 m^2	1 m^2	10.76 ft^2
Force	1 kg	9.807N	2.205 lbs
	102.0 kg	1 kN	0.2248 kip
Stress	1 kg/mm^2	9.807 MPa	1.4224 ksi
	1 kg/cm^2	9.807 kPa	1.4224 psi
	1 kg/m^2	9.807 Pa	0.2048 psf

Table 2.2 SI Reinforcing Bar Sizes and their US equivalent

Designation	Area (mm^2)	Area (in^2)	US Equivalent
Φ5	19.63	0.030	-
Φ8	50.27	0.078	-
Φ10	78.54	0.122	#3
Φ12	113.1	0.175	-
Φ14	153.9	0.239	-
Φ16	201.1	0.312	#5
Φ20	314.2	0.487	-
Φ22	380.1	0.589	#7
Φ24	452.4	0.701	-
Φ26	530.9	0.823	-
Φ32	804.2	1.247	#10

Throughout this chapter, a number of drawings of the structure and structural elements are presented. A number of these drawings were re-drafted from the original design drawings for clarity. For figures in this chapter that show original design drawings or calculations, the figure title will be appended with "(TUD FMD)" to indicate the materials were obtained from the TUD Facilities Management Department.

2.3 General Description of the Structural System

The focus of this investigation was on the tower portion of the Faculty of Architecture Building. Further, the study of the drawings was focused on those that depicted the structural system of the tower alone. The tower was long and narrow with 2 floor plate layouts that alternated on odd and even floors (recall the first floor of the tower is referred to herein as floor 2). The even stories provided the floors for the studio spaces. A typical plan for an even numbered floor of the tower is shown in Figure 2.1.

The odd floor plans are similar, with openings at the studio locations to provide for the double height spaces and mezzanine additions. Figure 2.1 also shows the approximate location of the coffee vending machine where the fire started on the 6th floor. Also shown is the portion of the northwest wing that collapsed in the fire.

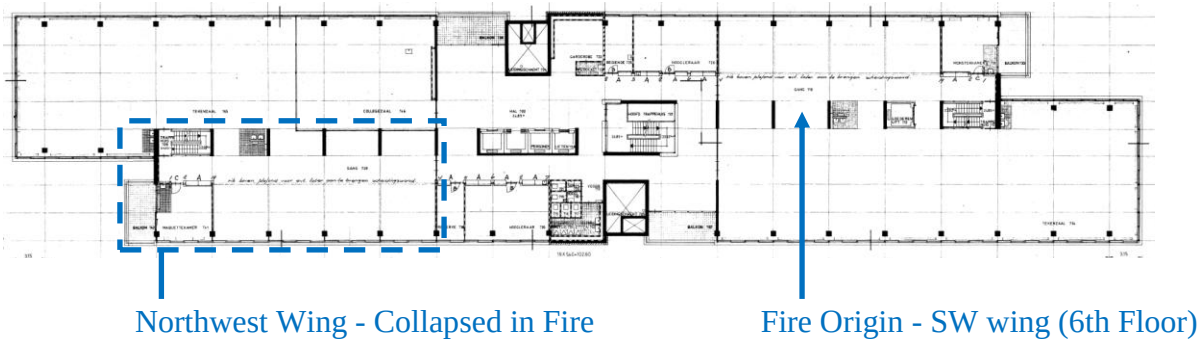


Figure 2.1 Tower Plan for Even Floors (TUD FMD)

The major elements of the tower's structural system included: columns, walls, girders, and joists. Insights into the design, dimensions, and potential weak points are discussed in this chapter. The major components of the structural system are illustrated in Figure 2.2. This plan was redrafted from the original drawings. It shows only the north end of the tower portion of the building (on even floors).

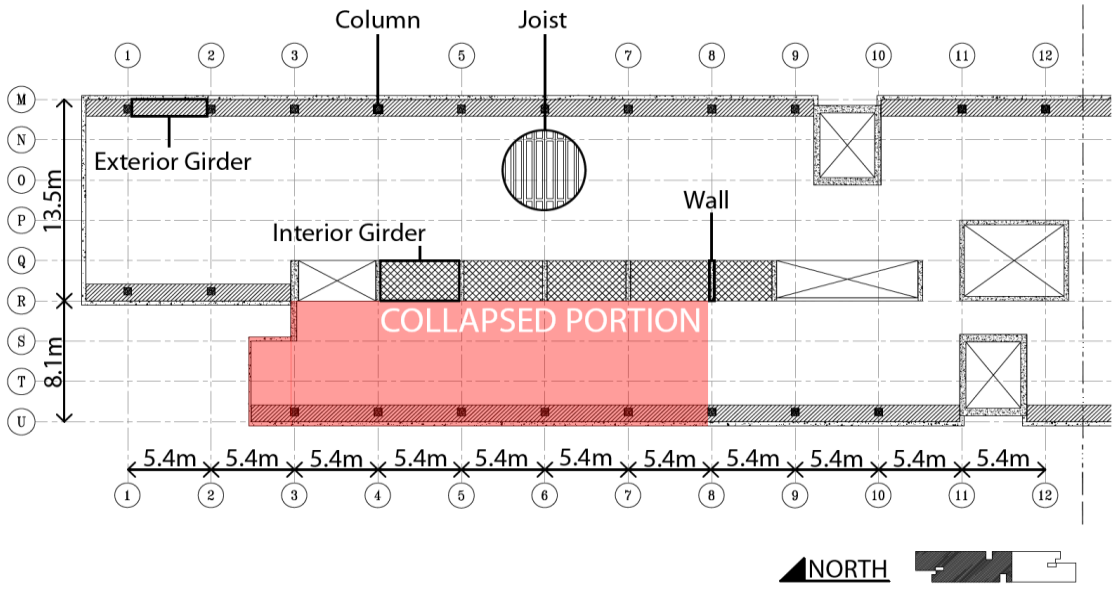


Figure 2.2 Major components of the structural system

The structural system of the tower was composed of 50 cm square columns placed along the perimeter of the building, typically at 5.4 meter intervals. Girders spanned the columns in the north-south direction, providing lateral support for the columns as well as serving as an element of the floor system. Thick walls supported another north-south girder that ran down the center of the structure. This wide, shallow girder more resembled a thickened slab than a beam, but will be referred to as the 'interior girder' in this report.

The floor system was composed of a joist/slab system spanning the east-west direction and connected at its ends to the girders running between the vertical elements. Part of this floor joist system contained precast inverted U-shapes and the rest of the slab was cast-in-place. There were two main joist types used throughout the building; with many slight variations in reinforcement and connections into the girders. The first type was mainly present on the shorter spans of the office wings (NW and SE). These were 250 mm deep while the joists supporting the double height studio spaces were 400 mm deep. The girders themselves were typically wide, shallow sections. A vertical parapet was cast integrally with the girder and ran around the entire perimeter of the structure. This exterior parapet/girder supported the non-structural precast concrete finish and glazing system.

From reviewing the construction documents and structural calculation sheets, it was possible to infer which elements the structural engineers of the FOA intended to carry the lateral forces imposed on the structure. In the North-South direction, the columns formed a moment resisting frame with the exterior girders. The building is narrow (approximately 21.6m wide) in this direction and the forces generated by wind were smaller than in the wider East-West direction. In the East-West direction, the entire lateral force is transmitted to the shear walls running down the center of the structure.

The full photo database will be discussed in Chapter 3 of this report, but a few key photos are presented in this section to illustrate portions of the structural system. Figures 2.3 to 2.6 show photos of the building structure during the original construction in the late 1960s.

Figure 2.3 Walls and Joists During Construction

Figure 2.4 Stairwell and Wall During Construction

Figure 2.5 Columns and exterior girders during construction

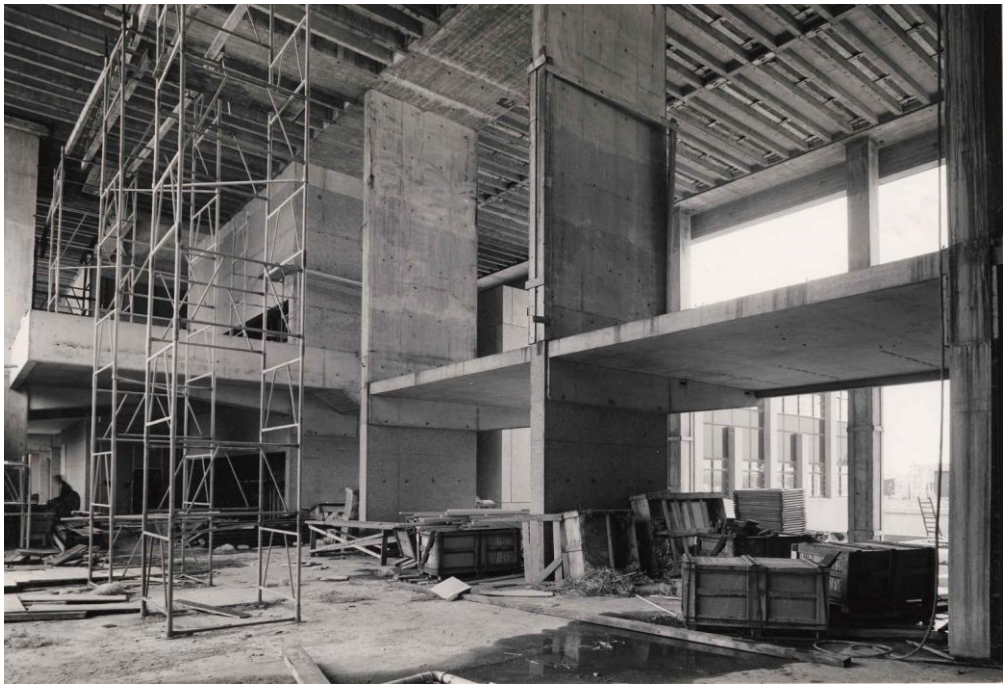


Figure 2.6 Shear walls and interior girders during construction

2.4 DESCRIPTION OF INDIVIDUAL STRUCTURAL ELEMENTS

This section provides more detailed descriptions of elements of the structure. The structure can be broken up into four major components: columns, walls, girders and joists. In the following sections each component is examined individually. Also provided is a description of the manner in which the individual elements were connected to each other.

2.4.1 Columns

The columns of the FOA were designed to resist both gravity and lateral loads. The columns in the podium structure were 52cm square, while those in the tower portion were all 50cm square over the full height of the building. Typical cross sections of the podium and tower columns are shown in Figure 2.7 and 2.8, respectively. The single story portion of the tower had column heights of 3.15m on even numbered floors and 3.85m on odd numbered floors (top of slab to top of slab). The studio spaces were double height and thus had 7.00m columns. Taking into account the thickness of the girders the columns framed into and assuming rigid connections gives slenderness ratios (kL/r) of about 22 for single story even floors, 27 for single story odd floors, and 49 for double height columns. This is a rather high range for concrete columns and led to highly reinforced sections, especially toward the base of the tower. The largest reinforcement ratio used was 7.14%. Though this is a heavily reinforced section, it is less than the maximum allowable reinforcement ratio (8%) from ACI 318-08 (10.9.1). Table 2.3 provides additional information on columns reinforcement on selected levels of the tower.

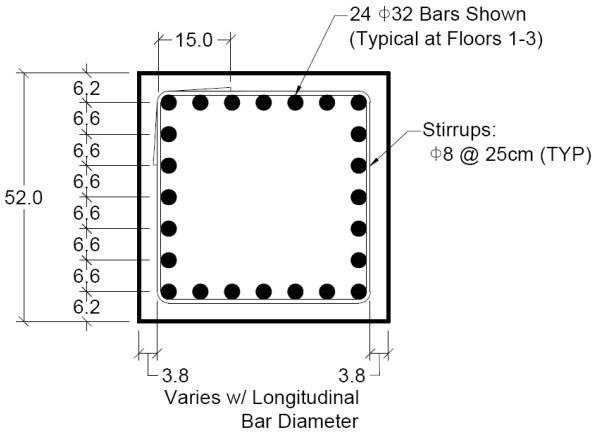


Figure 2.7 Typical Podium Column Cross-Section (cm)

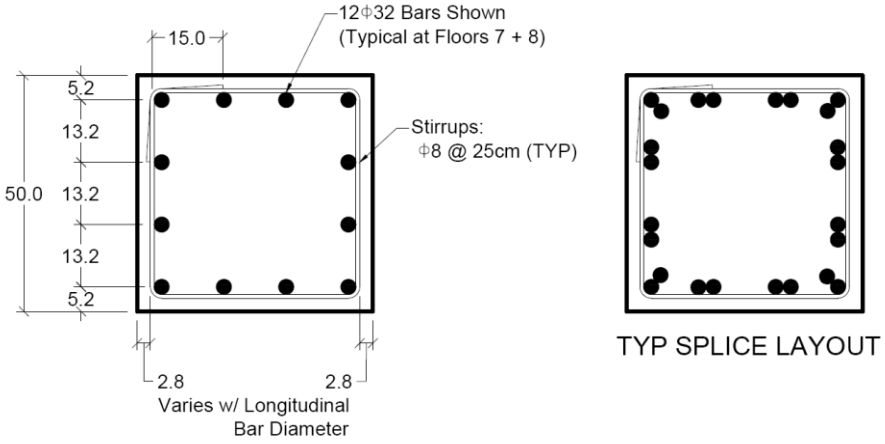


Figure 2.8 Typical Tower Column Cross-Sections (cm)

Table 2.3 Column Reinforcement at Critical Floors

Column Label	Floor	Reinforcement	Transverse
U3	6	16Φ32	Φ8 - 25mm
	7/8	12Φ32	Φ8 - 25mm
	9/10	12Φ26	Φ8 - 25mm
U4-9	6/7	12Φ32	Φ8 - 25mm
	8/9	4Φ24, 8Φ22	Φ8 - 25mm
	10	12Φ16	Φ8 - 25mm

As would be expected, the amount of longitudinal reinforcement decreased up the height of the tower. The podium structure columns typically had 24 bars. This decreased to 12 bars towards the top of the tower. The largest bar diameter used was 32mm (roughly equivalent to a US #10). The smallest longitudinal bar used in the columns was 16mm (#5). Typically half of the longitudinal bars were spliced above every other floor, or at the base of the double height floors. These splice lengths varied based on the diameter bars used. Typical splice lengths were in the range of 145cm for 32mm bars to 75cm for 16mm bars. These splice lengths would meet the requirements specified in 12.16 of ACI 318-08 for splices of deformed bars in compression. In addition to lap splices, some of the longitudinal bars were spliced at midfloor heights using a detail referred to in the construction documents as a "G-Lock Splice". This was most likely a threaded sleeve, but details on this splice type were not found. A more detailed description of the splices in the FOA can be found in Section 2.6.2 of this report.

The construction drawing notes specify a minimum clear cover of 2.5cm for all interior columns. However, the actual cover shown on the drawings was typically greater than this minimum. The detailing of the columns called for 5.2cm (6.2cm for 52cm columns) from the outer face to the center of reinforcement. This allowed for 2.8-3.8cm of concrete clear cover, depending on the diameter of the longitudinal bars. In any case, most of the covers specified were less than the ACI 318-08 requirements for interior columns. Section 7.7.1 of ACI 318-08 specifies a minimum of 1.5in (3.8cm). As will be discussed further in this report, an intense fire reaching high temperatures could potentially have allowed enough heat to transfer through the concrete to the reinforcement to cause a significant loss of strength. This would be especially problematic in highly reinforced sections that depend largely on steel contribution to strength. A more detailed description of the covers in the FOA can be found in Section 2.6.1 of this report.

Lateral ties on all columns were 8mm reinforcing bars. The ties were closed loop with 15cm overlap and were typically spaced at 25cm on center. This spacing was changed to 22cm above the 8^{th} floor. One deviation with requirements of ACI 318-08

was found in the 32 bar columns, located primarily in the podium portion of the structure. In some of these columns the longitudinal reinforcement is inadequately braced according to ACI. In most of the columns, many consecutive unbraced longitudinal bars were in use. To conform to ACI 318-08 (7.10.5.3), every other bar and any bar more than 6in (15cm) from the nearest bar must be braced by the corner of a tie. The additional bars that would be required for ACI 318-08 are illustrated in Figure 2.8.

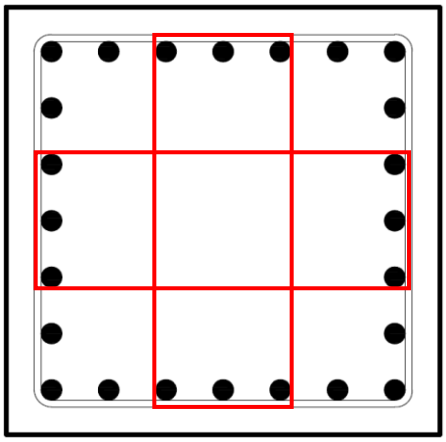


Figure 2.9 Typical Podium Column Cross-Section with ACI Compliant Ties (in red)

2.4.1.1 Corner Columns

The one major deviation from the typical column details discussed above was at the corner columns, U3 and M18. These were constructed as composite members, with 22mm thick steel plates welded into box columns 38cm by 41cm and imbedded in the 50 cm concrete columns. These plates were present through the connections with the girders at floors levels, but did not continue through the entire height of the columns. Continuity of the system was maintained by sliding reinforcement from the girders through holes drilled in the plates. Details are shown in Figures 2.10 and 2.11. It was not determined why these composite columns were necessary at only 2 corners of the building, but U3 was one of the columns that failed in the collapse of the northwest wing of the FOA.

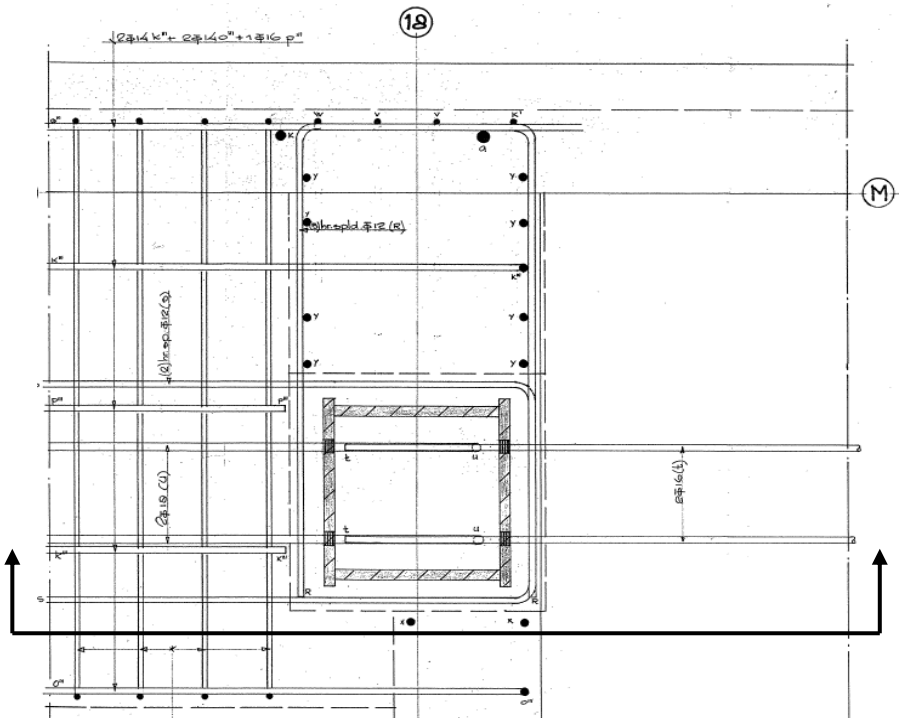


Figure 2.10 Typical Corner Column Plan (TUD FMD)

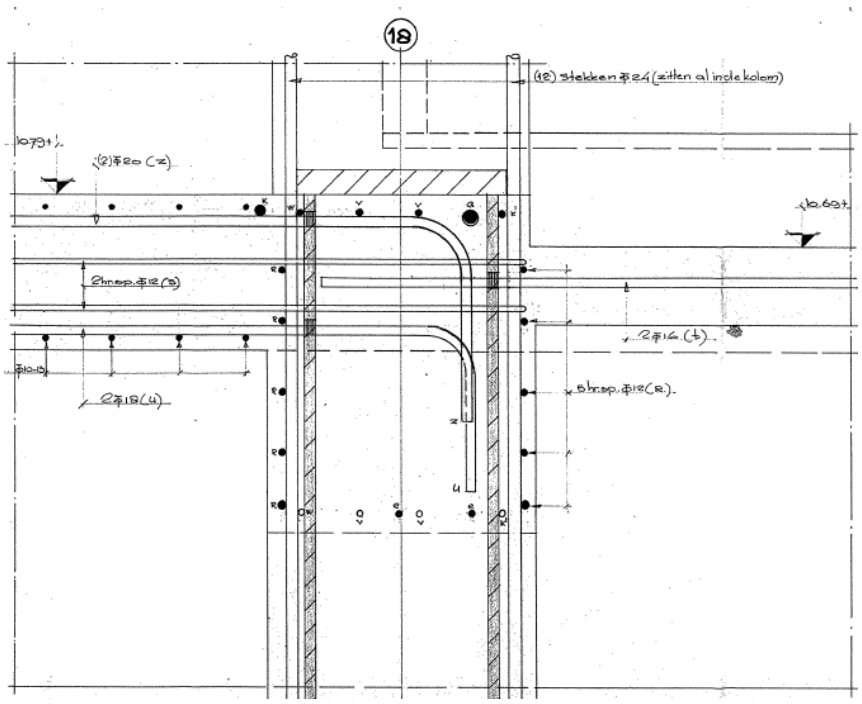


Figure 2.11 Typical Corner Column Section (TUD FMD)

2.4.2 Walls

The walls, like the columns, were designed to carry both gravity loads and lateral loads. In the east-west direction, the walls were designed as shear walls, and were the only elements designed to withstand lateral forces in that direction. Though the walls were not designed as the primary means of resisting lateral loads in the north-south direction, they did form a moment frame with the interior girders and would have contributed some lateral force resistance.

The walls were 30cm by 270cm throughout the tower, with very few openings. The longitudinal reinforcement was distributed so that the ends of the walls typically used larger bars to carry the bending moments introduced by the lateral load. They were detailed in a manner similar to that of the columns, with straight longitudinal reinforcement enclosed by closed loop ties. The ties used in the walls were 10mm in diameter (slightly larger than those used in the columns) and were typically spaced at 25cm. The bar diameters decreased up the height of the walls and were spliced above every other floor. The splices in the walls were not staggered as they were in the columns. Typically, all of the longitudinal bars were spliced at the same elevation. The longitudinal bars were also smaller than those used in the columns. The largest bar used in a wall in the tower was 24mm and decreased to 12mm at the top. Figure 2.11 shows the typical wall reinforcement layout.

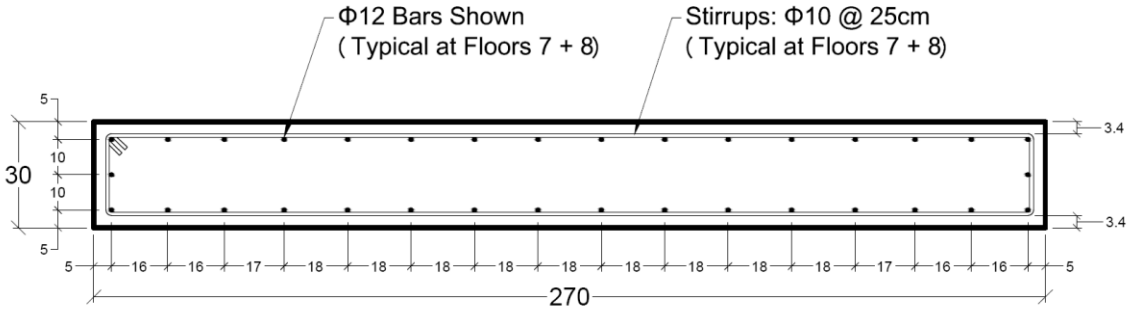


Figure 2.12 Typical Wall Cross-Section

The splice lengths in the walls also varied based on the diameter of the bar being spliced. Splice lengths in the walls were in the range of 75cm for 25mm bars, decreasing to 55cm for 12mm bars. No G-Lock splices were found to be used in the walls.

It should also be noted that the collapse of the northwest wing stopped at the west edge of the shear wall grid suggesting a change in resistance or other characteristic. None of the walls collapsed as a result of the fire.

2.4.3 Girders

The joists of the FOA floors transmitted gravity loads to girders that spanned between the walls and between the columns. These girders were typically wide, shallow sections resembling thickened slabs more than beams.

There were two main types of girders used in the FOA. The interior girders running between the walls were 270cm wide and 40cm deep throughout the entire building. A wide variety of reinforcement was used, and the general layout of this reinforcement is illustrated in Figure 2.12. The significant width of these girders allowed for large areas of concrete and a large number of bars.

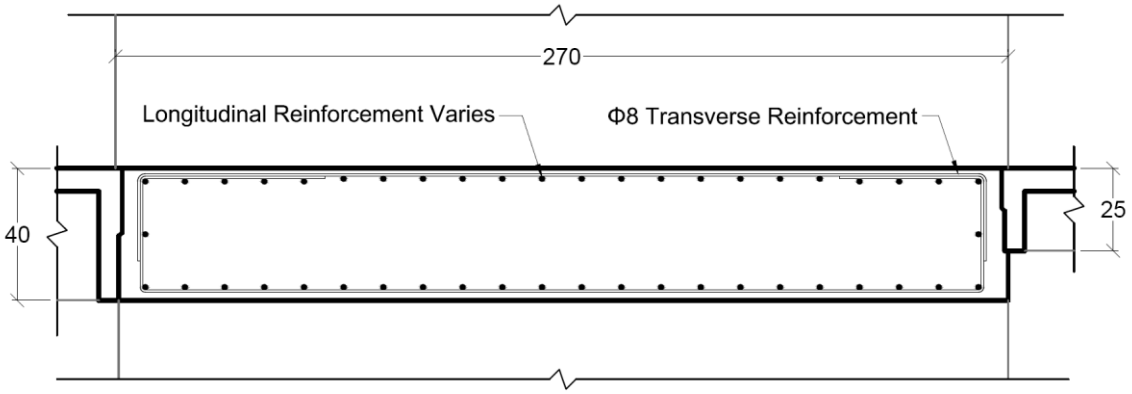


Figure 2.13 Typical Interior Girder Cross-Section

The exterior girders were similar to the interior girders in dimension, but were cast integrally with parapets that varied in height around the facade of the FOA. The girder dimensions and reinforcement were also symmetric about the building's central axis, to the point that the plans use the same drawings to detail each side. These exterior

girders were significantly wider than the columns they framed in to. This and the full girder-to-column joint detailing layout are illustrated in Figures 2.20 to 2.22.

2.4.4 Joists

The floor joists in the FOA were constructed of precast U-beams, connected by a cast-in-place slab. This is illustrated in Figure 2.14. The joists were supported at the interior by the girders spanning between the walls and the exterior by the girders spanning between the columns.

The wooden formwork for the cast in-place portion of the floor between the precast joists was reportedly left in for the life of the building. This formwork was hidden by the suspended ceiling system, but can be seen in the construction photo in Figure 2.4 (Kees van Weeren, 2008).

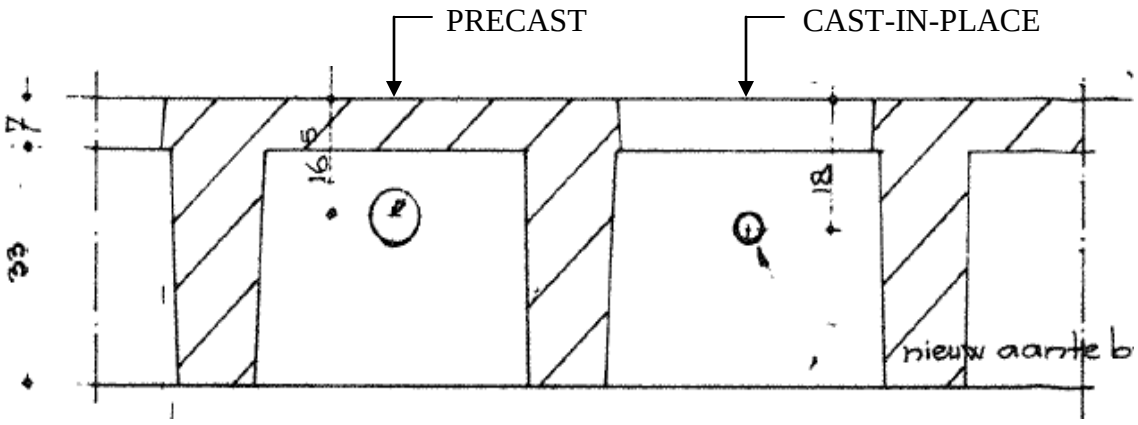


Figure 2.14 Joist Construction Sequence (TUD FMD)

There were two main types of joists in the FOA. The 250mm deep section (Figure 2.15) was used in the shorter spans of the office portion of the tower. The 400mm section (Figure 2.16) was typically used in the longer spans of the studio spaces. None of the portions of the structure containing the 400mm joists collapsed during the fire. A full map of the typical joist layout is shown in Figure 2.17.

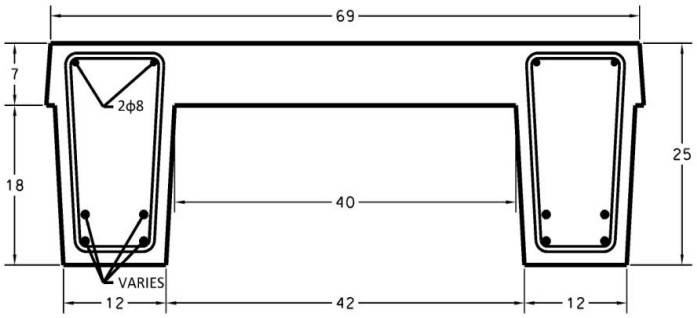


Figure 2.15 Typical Joist Section (250mm)

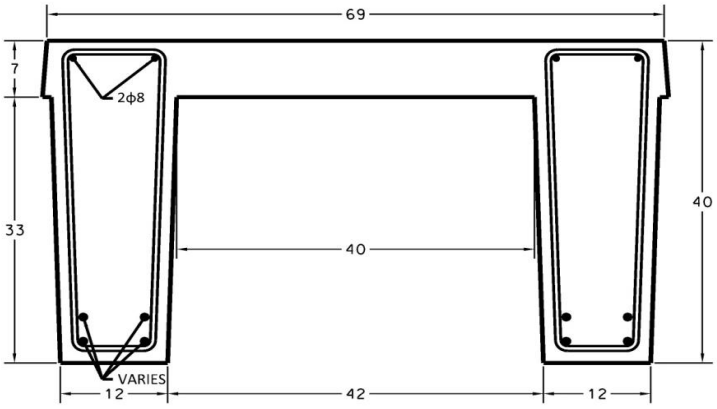


Figure 2.16 Typical Joist Section (400mm)

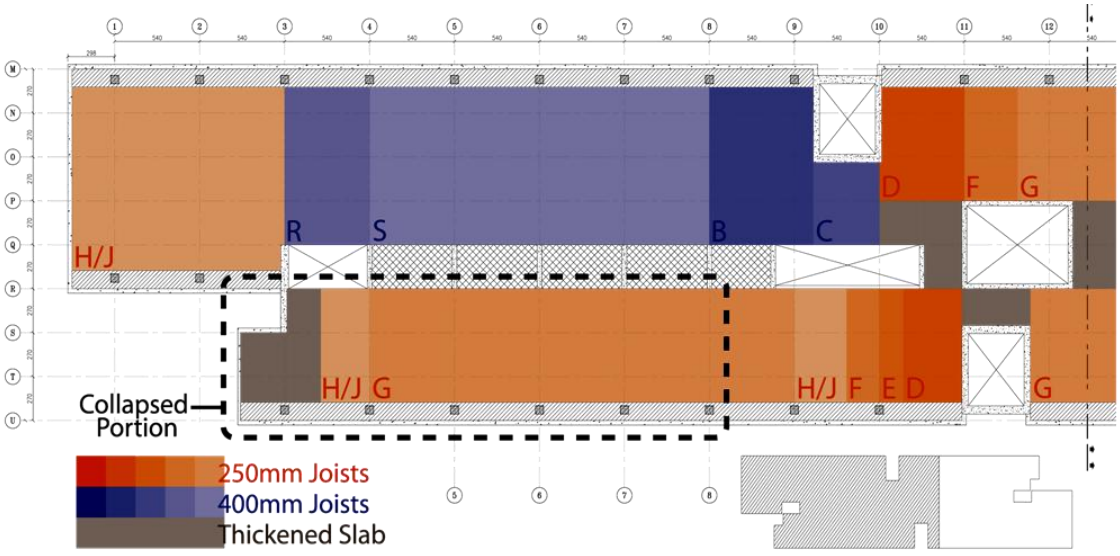


Figure 2.17 Floor 8 Joist Type Map

Table 2.4 Joist Midspan Reinforcement Schedule

Joist Label	Type	Top Steel	Bottom Steel	Transverse
B	400mm	2Φ8	2Φ12, 2Φ14	Φ5 - 25mm
C	400mm	2Φ8	2Φ10	Φ5 - 25mm
D	250mm	2Φ8	2Φ14, 2Φ16	Φ5 - 25mm
E	250mm	2Φ8	2Φ16, 2Φ18	Φ5 - 25mm
F	250mm	2Φ8	4Φ16	Φ5 - 25mm
G	250mm	2Φ8	4Φ12	Φ5 - 25mm
H/J	250mm	2Φ8	4Φ14	Φ5 - 25mm
R	400mm	2Φ8	2Φ14, 2Φ16	Φ5 - 25mm
S	400mm	2Φ8	4Φ14	Φ5 - 25mm

The reinforcement shown in Table 2.4 is taken from the section at mid span of the joist, and typically consisted of 4 bars in the bottom and 2 in the top. Two bars were continuous in the top and bottom with two additional bars at mid span in the bottom and two additional bars in the top at the ends. This is illustrated in Figure 2.18.

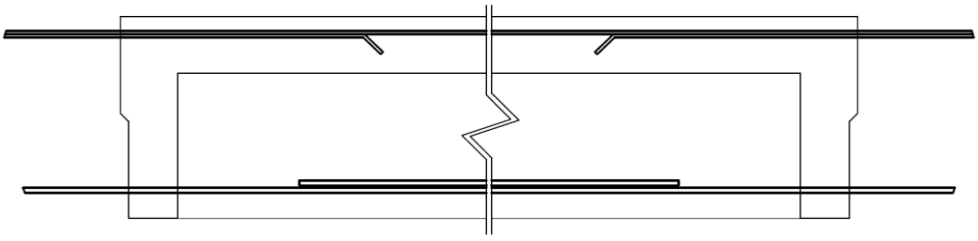


Figure 2.18 Typical Joist Reinforcement Layout

2.4.5 Joist-Girder Connection

The details and load transfer mechanisms at connection between the joists and girders were not completely clear from the available information. Recall that the joists were constructed of both precast and cast-in-place components. The joists were capped

at the ends with a 7 cm wide cast-in-place 'joist cap'. This apparently served as an intermediary piece during the construction process of connecting the precast joists to the cast-in-place girders, though the details of this sequence could not be fully determined from the available documents. This joist cap was present at both the connection to the interior and exterior girders. Figure 2.19 illustrates how this connection was designed (and presumably constructed).

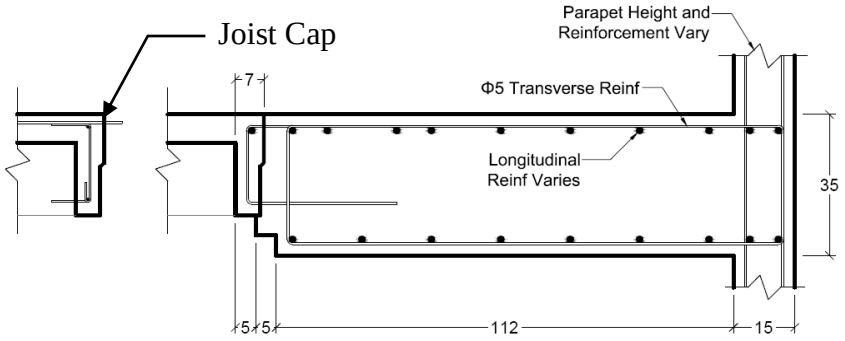


Figure 2.19 Typical Exterior Girder Section at Joist Joints

At some column lines the precast joists were replaced by a cast-in-place concrete beam that formed a joint with the girder and column at the exterior of the building. This presumably provided a load path for lateral forces in the East-West to be transmitted the shear walls. The details of these beams and the girder column joint can be seen in Figures 2.20, 2.21, & 2.22.

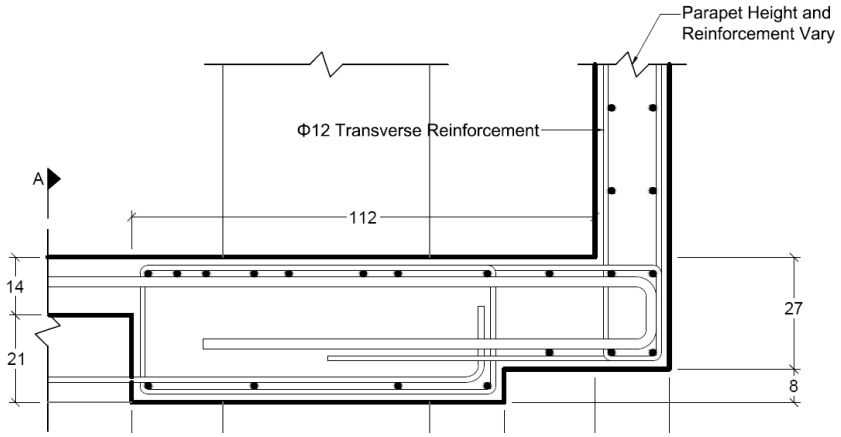


Figure 2.20 Typical Exterior Girder Section at Column Joint

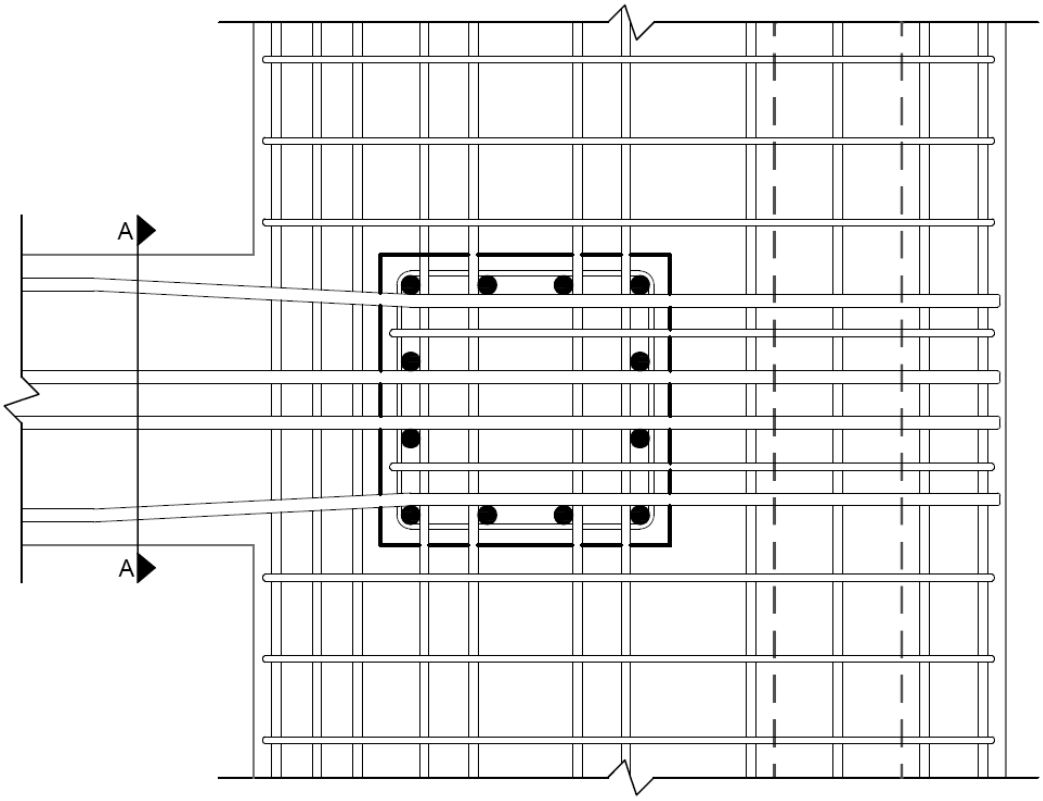


Figure 2.21 Typical Column Plan at Joist/Exterior Girder Connection

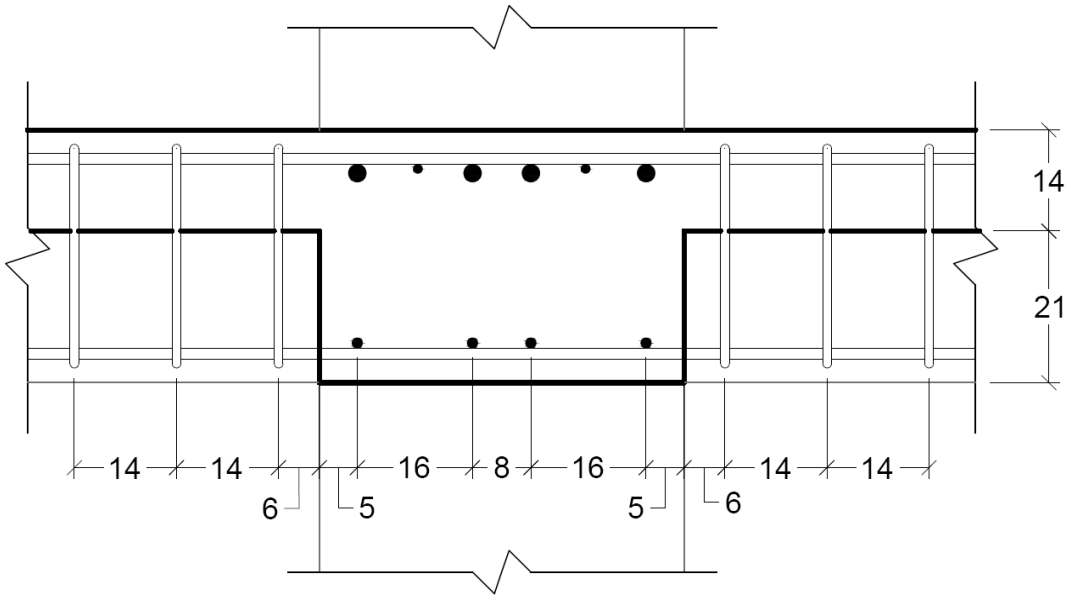


Figure 2.22 Typical Joist Section at Column Joint (Section A-A)

2.4.6 Materials

The columns and walls were constructed of primarily 350 kg/cm^2 (34.32 MPa, 4978 psi) concrete, while the precast joists, slabs, and girders used 300 kg/cm^2 (29.42 MPa, 4267 psi) concrete. The cement used was a Type A blast furnace cement at a density of 375 kg/m^3 concrete. The columns and walls utilized an admixture of 0.2% plasticizer (0.75 kg/m^3 concrete). Some key pieces of information that could not be determined were the type of aggregate and the mechanical properties of the reinforcement. However, the Dutch members of the research team have stated that the aggregate used was most likely river gravel (siliceous aggregate) and a study of the calculation sheets provided with the construction documents showed that the design engineers used a maximum allowable stress of 2110 kg/cm^2 (207 MPa, 30 ksi) in the reinforcement. This, along with a historical study of available grades of reinforcement in the Netherlands in the late 1960s and early 1970s, suggests that the reinforcement was most likely the equivalent of a grade 40 steel. The yield strength of reinforcement used in the calculations of this report was 276 MPa (40 ksi).

2.4.7 Hanging Mezzanine Floors

An additional portion of the structure that has not yet been discussed is the mezzanine space. The mezzanines were constructed by hanging a wood and steel structure from the existing concrete joists within the double story studio spaces. A portion of a mezzanine is visible in the photograph in Figure 1.5. The full set of drawings containing the structural details of the mezzanine structure were not found. Only two sheets with details of this structure were found. Figure 2.23 shows how the steel frame was hung from the joists. Calculation sheets were also found that show the engineers accounted for the additional loads from this additional structure. There were no notes found to specify any fire protection to these steel members and they can be seen to be exposed from images such as Figure 1.5. This space can also be seen in Figure 2.24, taken during a remodeling of the studios.

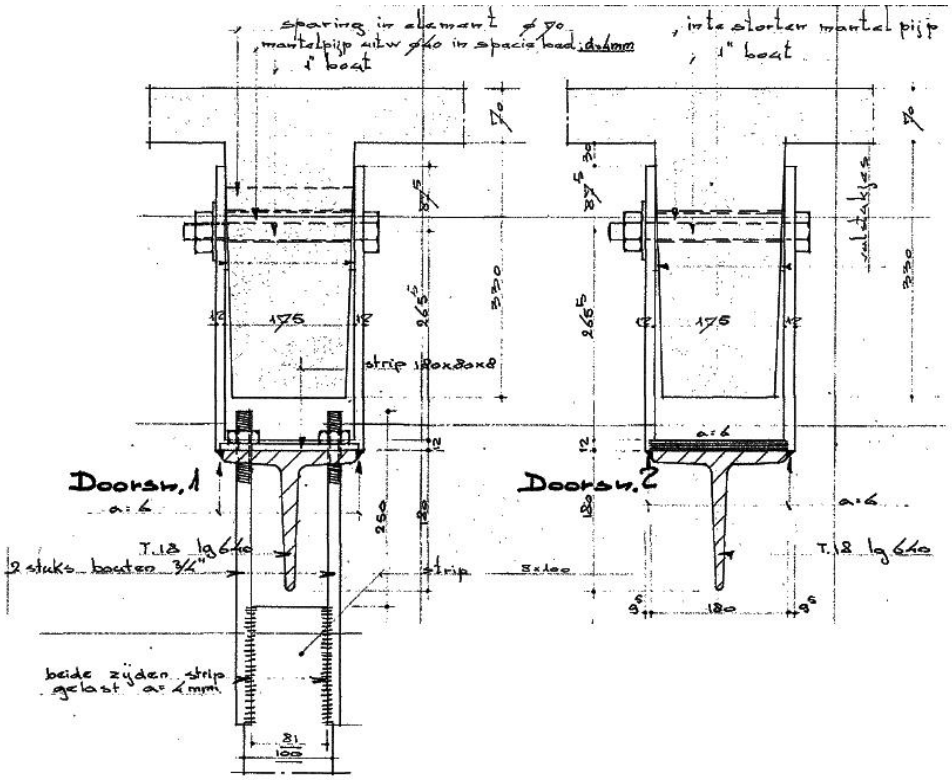


Figure 2.23 Joist-Mezzanine Hanger Connection (TUD FMD)

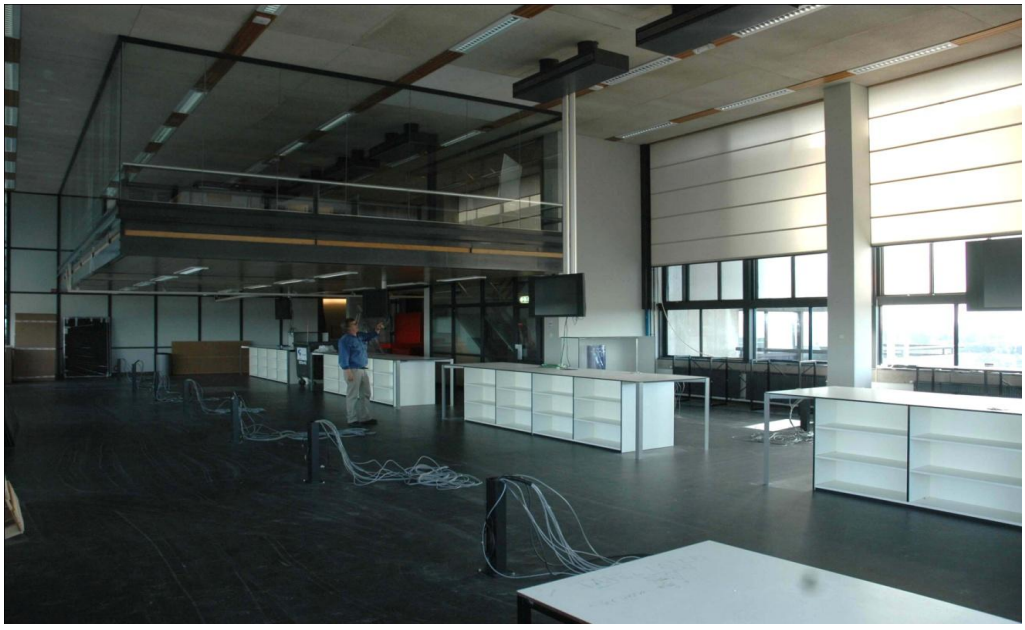


Figure 2.24 Mezzanine overhanging the studio space during renovation

2.4.8 Structural Calculation Sheets

A substantial number of the original structural design calculation sheets were available to the research team. A brief review of these calculations was conducted to obtain further insights into the structure. The calculations were type written in Dutch, and followed an allowable stress design approach. The calculations were quite difficult to follow since they were in Dutch and because the researchers were not familiar with building code requirements in use in the Netherlands at the time of the original design. Nonetheless, several useful insights were gained from the calculations. First was the allowable stress of reinforcement used in the design. This was the only place any reference was found to the grade of reinforcement specified, which was necessary to carry out the rest of the post fire analysis. This value of limiting stress was found to be 2110 kg/cm^2 (207 MPa, 30 ksi). Secondly, from diagrams like Figure 2.25, which shows the design of a wall for wind, the intended lateral force resisting systems of the structure was determined.

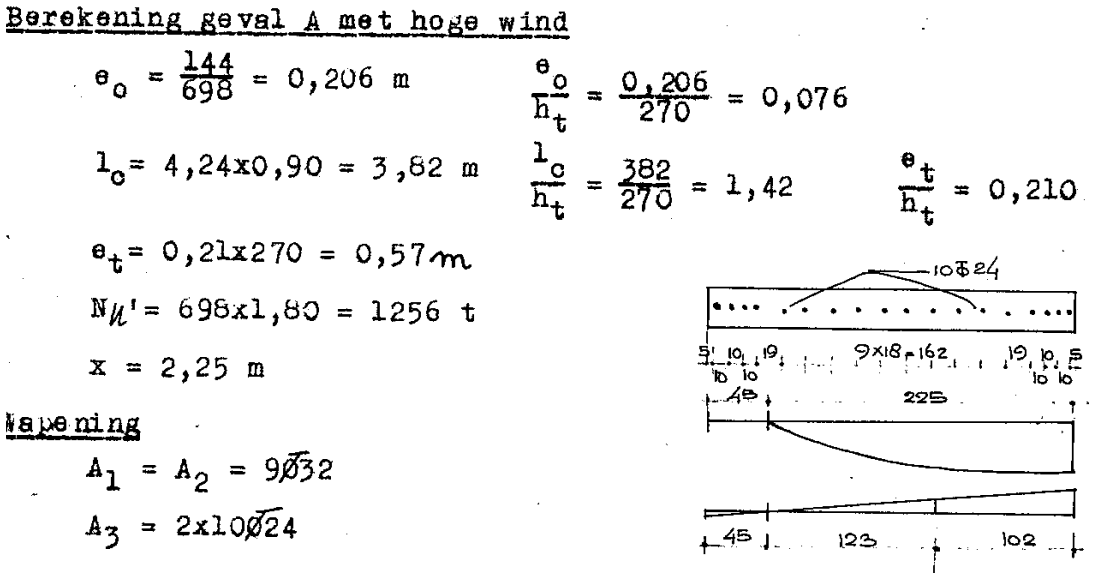


Figure 2.25 Example design calculation of a wall for wind (TUD FMD)

Needless to say, this design was completed before the advent of computer analysis and design. The design was completed by isolating small portions of the structure to analyze and design by hand methods. For example, Figure 2.26 shows 2

exterior columns, 1 girder and 1 floor joist, isolated from the rest of the structure for design.

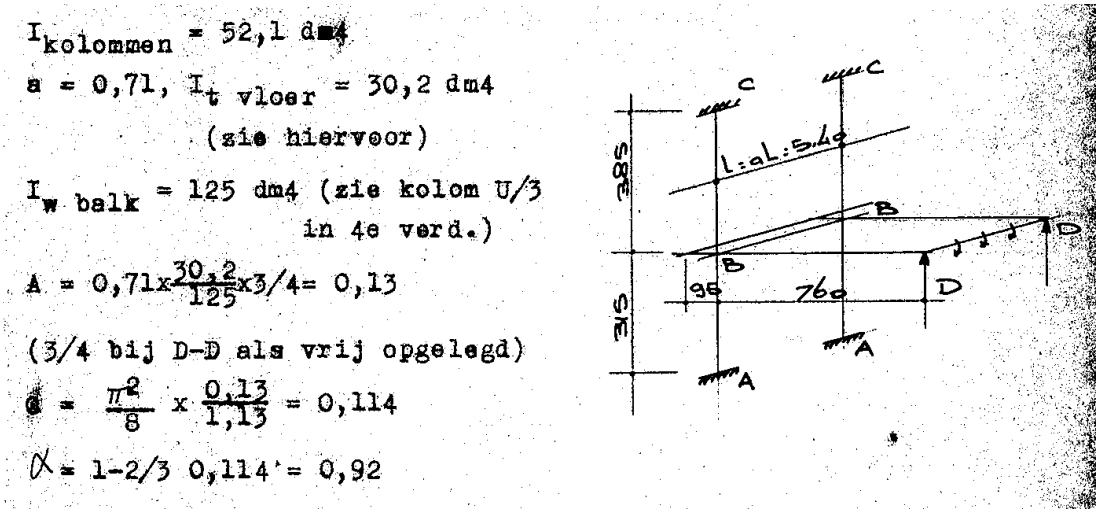


Figure 2.26 Example design calculation of a structural subsystem (TUD FMD)

The calculations were also examined to find information on design loads used for the structural design. They were highly variable and the language barrier prevented any conclusive information from being determined, but a representative sample was examined to find a range of the loads used for the structural design of the FOA. These are shown in Table 2.5.

Table 2.5 Design Loads

Load	kN/m^2	psf
Floor weight (Eigen gewicht vloer)	2.87 - 7.18	60 - 150
Ceiling (Plafond)	0.19 - 0.38	4 - 8
Finishes (Afwerking)	1.01	21
Wall (Muur)	7.18	150
Masonry (Metselwork)	3.93 - 5.99	82 - 125
Partitions (Separaties)	1.01	21
Effective? Load (Nuttige Belasting)	1.00 - 2.97	21 - 62

2.5 UNKNOWN/UNCLEAR INFORMATION

Though the records kept for the FOA design and construction were extensive, there was still some information on the structure that was not found or was unclear. Following is a list of items for which additional information or confirmation of assumed information would be desirable for a more complete analysis of the structural response to the fire.

1) Mechanical properties of steel reinforcement including both specified grades and properties and an estimate of actual expected properties.
2) Mechanical properties of concrete including specified strength and aggregate types and an estimate of actual expected properties.
3) Joist construction methods.
4) Joist-Girder connection details
5) Details of the mezzanine structures.
6) Design codes used.
7) Design loads used.

2.6 ACI 318-08 COMPARISONS

In this section of the report, selected detailing provisions of ACI 318-08 (ACI 2008) are compared with the detailing provided in the FOA. The purpose of this comparison is to identify detailing features of the FOA that deviate from current US building code requirements and to identify any potential weaknesses in the structural system that may have contributed to the fire induced collapse and that merit more detailed investigation.

ACI 318-08 does not explicitly address fire resistance, though a few provisions, most notably cover requirements, implicitly provide fire resistance. Fire resistance of reinforced concrete structures is addressed by ACI Committee 216: Fire Resistance and Fire Protection of Structures, publication ACI 216.1-07 (ACI, 2007). The provisions in this document focus primarily on concrete cover of reinforcement, as well as minimum member widths.

2.6.1 Reinforcement Cover

Table 2.6 lists the covers provided and specified by the construction documents for the FOA. The *specified* values are minimum values of cover listed in the drawing notes. The *provided* values are the cover values shown on the drawings, and are typically greater than the specified values. Also shown are the covers required by ACI 318-08 (7.7.1). The specific members in the table are typical of the 6-8th floors, where the collapse appears to have initiated. Also, keep in mind that ACI requirements vary slightly with respect to bar size. The requirements for bars in the range of sizes used are shown.

Table 2.6 Reinforcement Cover

	Label/Bar Size	Specified Cover, mm	Provided Cover, mm	ACI 318-08 (7.7.1) Req'd, in	Req'd, mm	OKAY for ACI?
Floors	JoistG (top)	10.0	16.0	0.75	19.1	No
	JoistS (top)		16.0			No
Beams	JoistG (bot)	20.0	20.0	1.50	38.1	No
	JoistG (side)		16.0			No
	JoistS (bot)		19.0			No
	JoistS (side)		15.0			No
Columns	Φ32	25.0	36.0	1.50	38.1	No
	Φ28		38.0			No
	Φ26		39.0			Yes
	Φ22		41.0			Yes

In most cases, the concrete cover of reinforcement in the FOA was less than required in ACI 318-08. Since concrete cover provides insulation for the reinforcing bars, this could have lessened the ability of the structure to withstand exposure to fire.

2.6.2 Shear Reinforcement

The joists and girders in the FOA were designed for shear, but the code used in the design may have led to less shear strength than ACI 318-08 would allow. More specifically, it appears that ACI 318-08 would require about twice as much transverse reinforcement as was provided in the joists and girders. This is based on the requirement from Section 11.5.5 of ACI 318-08 that, if shear reinforcing if required, the maximum spacing between bars is the minimum of (d/2, 24in (61cm)). The transverse reinforcement specified for the joists was typically ϕ5 bars, spaced at 25cm (regardless of section height). This is twice the ACI maximum for the 25cm joists and 1.25 times the maximum for the 40cm joists.

Further analysis of the shear resistance of the members of this structure is not provided in this report. However, shear strength of the members, and particularly the joists, merit further investigation.

2.6.3 Column Compression Splices

The required compression lap splice lengths are calculated from the requirements of ACI 318-8, Sections 12.3, 12.16, and 12.17. A summary of common column splice lengths provided in the FOA columns and comparisons with the corresponding ACI requirements are shown in Table 2.7. As can be seen in this table, the majority of the splices conformed to the current code standards and would have been able to fully develop their yield strength.

Table 2.7 Compression Lap Splice Lengths (Columns & Walls)

Bar Size	Compression Splice Length, mm	ACI 318-08 (Ch 12) Required, mm	OKAY for ACI?
Φ16	750	284	Yes
Φ19	1000	337	Yes
Φ22	1000	390	Yes
Φ25	1350	444	Yes
Φ29	1450	515	Yes
Φ32	1450	568	Yes

2.6.4 Joist Development Lengths

As mentioned previously, the connection between joists and girders appears to be largely dependent on the ability of the reinforcement to develop bond with the concrete in the girder. These embedded bars are almost all straight and the embedment length into the girder seems to be determined based on bar diameter. Each joist detail drawing shows the length of this embedment into the girder supporting it. Figure 2.27 shows an example of how this detail was shown on the drawings.

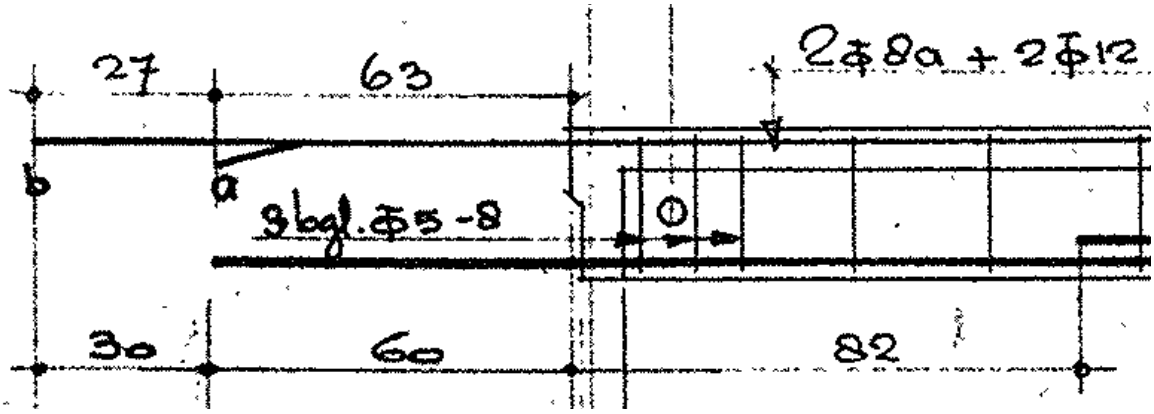


Figure 2.27 Embedded reinforcement detail between Joist D and exterior girder

These lengths are relatively consistent between girders, with the length of embedment proportional to the diameter of the bar. However, some bars of the same diameter have different embedment lengths for reasons not entirely clear. Table 2.8 shows the most common embedment lengths, based on bar size. These lengths are compared with the required development lengths for deformed bars in tension from ACI 318-8 Section 12.2. These required development lengths assume that the spacing between the bars being developed is greater than d_b, the clear cover is greater than d_b and the stirrup spacing over the development length is less than the code minimum. With these assumptions in mind, the required development length (for #6 bars (ϕ18) and smaller) is given by ACI 318-08 Section 12.2.2.

Table 2.8 Joist Longitudinal Reinforcement Embedment Lengths Into Girders

Bar Size	Top Bar Embedment Length, mm	Bottom Bar Embedment Length, mm	ACI 318-08 (12.2.2) Req'd Top	Req'd Bot	OKAY for ACI?
Φ8	630	n/a	255	196	Yes
Φ10	750	n/a	319	245	Yes
Φ12	900	600	382	294	Yes
Φ14	n/a	630	446	343	Yes
Φ16	n/a	680	510	392	Yes
Φ18	n/a	730	573	441	Yes

This table shows that the embedment lengths were adequate to fully develop the joist bars within the girder width. It should be noted that the focus of this investigation is a sectional analysis of members and this connection was not studied in detail. Future studies should look at the possibility of bond degradation at elevated temperatures which could have led to bar pullout as the joists underwent thermal expansion and contraction.

2.7 OBSERVATIONS

After a review of the original construction documents provided by the Delft University of Technology, a few possible issues that merit further investigation in the system were identified. These issues include: cover less than the current code requirements; joist to girder connections vulnerable to significant loss of strength during heating; and some elements with less than minimum shear reinforcement. These pieces of the structural system had the potential to either make the structure more vulnerable to fire or reduce its ability to withstand progressive collapse.

2.8 SUMMARY

In this chapter, a description of the structural system and descriptions of the individual components that form it were provided. The design was compared with the current code requirements of ACI 318-08 and a few deviations were identified. In the next chapter we will look at the visual evidence of the collapse of the FOA by studying photographs and video taken the during the fire.

CHAPTER 3
Photo Database & Preliminary Fire Timeline

3.1 OVERVIEW

As part of the overall investigation of the fire and collapse at the FOA, photographs and videos were collected that documented the building before, during and after the fire of May 13, 2008. This photographic evidence is a very useful aid in understanding the layout and contents of the building before the fire, the movement of the fire through the FOA on May 13, 2008, and the condition of the structure before, during and after the collapse of the northwest wing.

This chapter starts with a preliminary timeline of key events in the May 13, 2008 fire. This is followed by a description of a database of photos that was developed for this project. The chapter then provides some preliminary analysis and observations from the photos that are pertinent to the understanding and analysis of the structural collapse.

3.2 PRELIMINARY TIMELINE OF EVENTS

To provide an overall view of the events at the FOA on May 13, 2008, a preliminary timeline of events was developed by the research team, and is reported in Meacham et al (2010). This preliminary timeline was based on information obtained from photos and videos; interviews of eyewitnesses, fire fighters, and other individuals familiar with the FOA; and other sources of information. Table 3.1 provides the resulting timeline. The fire started in a coffee vending machine located on the south portion of the 6th floor. It appears that flaming combustion started somewhere around 9am and then spread quickly, both horizontally on the floor of origin and vertically to other floors above. Fire fighters first arrived at the building at about 9:30am and initially attempted to fight the fire from inside of the building. Fire fighters evacuated the building by about noon, as the fire continued to move both horizontally and vertically through the building.

The northwest wing the building collapsed at about 4:40pm. Thus, the duration of time from the start of flaming combustion to the collapse was approximately 8 hours.

Table 3.1 Preliminary Timeline of Events at FOA on May 13, 2008 (Meacham et al 2010)

Time	Reported Event
8:15am - 8:30am	People recognize smell from coffee vending machine. Plugs are pulled out of sockets on the fifth and sixth floor. The coffee machine is not touched. No visible smoke.
8:55am	Smoke coming out of the coffee vending machine. Smoke seems to come out of a gap between the top of the machine and the red pantry unit. The smoke is light-grey.
9:00am	White smoke coming out of a slot above the coffee vending machine. A BHV employee gets a call about the fire and is asked to take a look at the sixth floor.
9:10am	Smoke coming out of the machine 'in a stripe' (from a small slot at the top front). The smoke is described as 'thick and black'. 'Fire balls' (likely burning cups) are observed coming out of the machine.
9:15am	Occupants are extinguishing a starting fire using a hand-extinguisher. Smell of an 'unpleasant' fire stink. Big flames begin to come out of the coffee machine.
9:16am	The Fire Department gets a call for help from a TU Delft Project Manager. Fire alarm is activated.
9:18am	The Regional Alarm Centre gets an automatic fire alarm from the Faculty of Architecture.
9:27am	The first fire brigade arrives at the building and notices smoke on the outside, but no fire. BHV-workers have evacuated the people in the building.
9:32am	The second fire officer reports a "medium fire". The first fire officer sends four people with "flatkratjes" to the sixth floor. The people on the sixth floor are trying to extinguish the fire by using the fire hose. There is almost no pressure, so it is useless.

9:33am	The second fire officer reports a "large fire". The heat is intense and a thick layer of smoke fills the space. Soon flames are starting on the ceiling and they retreat. Just before they want to leave through the door. the is a flashover and which puts the whole fire area on fire.
9:50am	Conflagration on the south ends of the 6th and 7th floors and extending very quickly via the facade.
11:30am	Floors 6-11 are on fire (south section).
12:15pm	All fire fighters are now outside of the building. Fire spreads to the center section.
12:55pm	Fire spreads to the north section on the 10th floor.
1:39pm	Fire observed on the north section, floors 7-11.
4:40pm	A portion of the north section collapses.

This timeline gives an overall view of the events of the day, but from a structural perspective, a more detailed view of the behavior of the fire in the areas that eventually collapsed is needed. This chapter will review the extensive photographic/video evidence and provide a preliminary assessment of the fire movement through the building. The results of this assessment are summarized and compiled into a visual timeline of the fire by diagramming the portions of the FOA where flame and smoke are visible in windows, based on photos with accurate timestamps. The primary purpose of this preliminary assessment is to estimate the duration of burning in various areas of the northwest wing prior to collapse. This estimated duration of burning will then be compared to the duration of burning predicted by the simplified fire models described in Chapter 4.

3.3 PHOTO DATABASE

Over 4000 photos were compiled by the research team from a number of different sources. This included photos from the time the FOA was constructed up through its demolition. To facilitate the use of these photos, they organized into a searchable database. The photos were first compiled and sorted into the general times of: before, during, or after the fire. After this, the photos taken during the fire were further

categorized and 'tagged' with information such as: estimated time taken, which floors have visible fire or smoke, and the side(s) of the building the photo shows. A short Microsoft Excel VBA macro was written which allows the user to sort the photos by any of the characteristics the photos are 'tagged' with.

3.4 PRELIMINARY FIRE TIMELINE

By using the photo database described in the previous section, it was possible to construct a preliminary 'visual timeline' that graphically shows the movement of the fire on May 13, 2008, as viewed from the east or west side of the FOA. An example of one page from the west timeline is shown in Figure 3.1. The full timelines of the east and west elevations are provided in Appendix C. Each graphic for the timeline is associated with a specific time. For example, the graphic shown in Figure 3.1 is for 2:35 pm. Observations from photos are then documented in terms of fire and/or smoke visible at the windows. As indicated in the legend in Figure 3.1, three levels of flame observations are recorded: “flames visible,” “flames out window,” and “flames to next floor height.” The first designation, “flames visible,” indicates that flames can be seen through the window but are not outside of the window. The next designation, “flames out window,” indicates flames are visible outside of the window, but are not extending to the floor above. The final designation, “flames to next floor height,” indicates large flames were visible outside of the window and extend to the floor above. These three designations are intended to describe visible flames of increasing intensity. Also indicated on each graphic are windows where dense smoke is visible. Finally, the location of broken windows is also shown. Graphics marked as “West Elevation” show the west side of the building, and consequently the view is looking towards the east. Similarly, the graphics marked as “East Elevation” show the east side of the building and consequently show the view looking west. The collapsed portion of the building is visible in the “West Elevation” graphics. A total of 24 West Elevation graphics and 24 East Elevation graphics were prepared, and are complied in Appendix C. It should be noted that the West Elevation

graphics do not show flames, smoke or window breakage on the south side of the tower, as this portion of the West Elevation was not clearly visible in the available photographs.

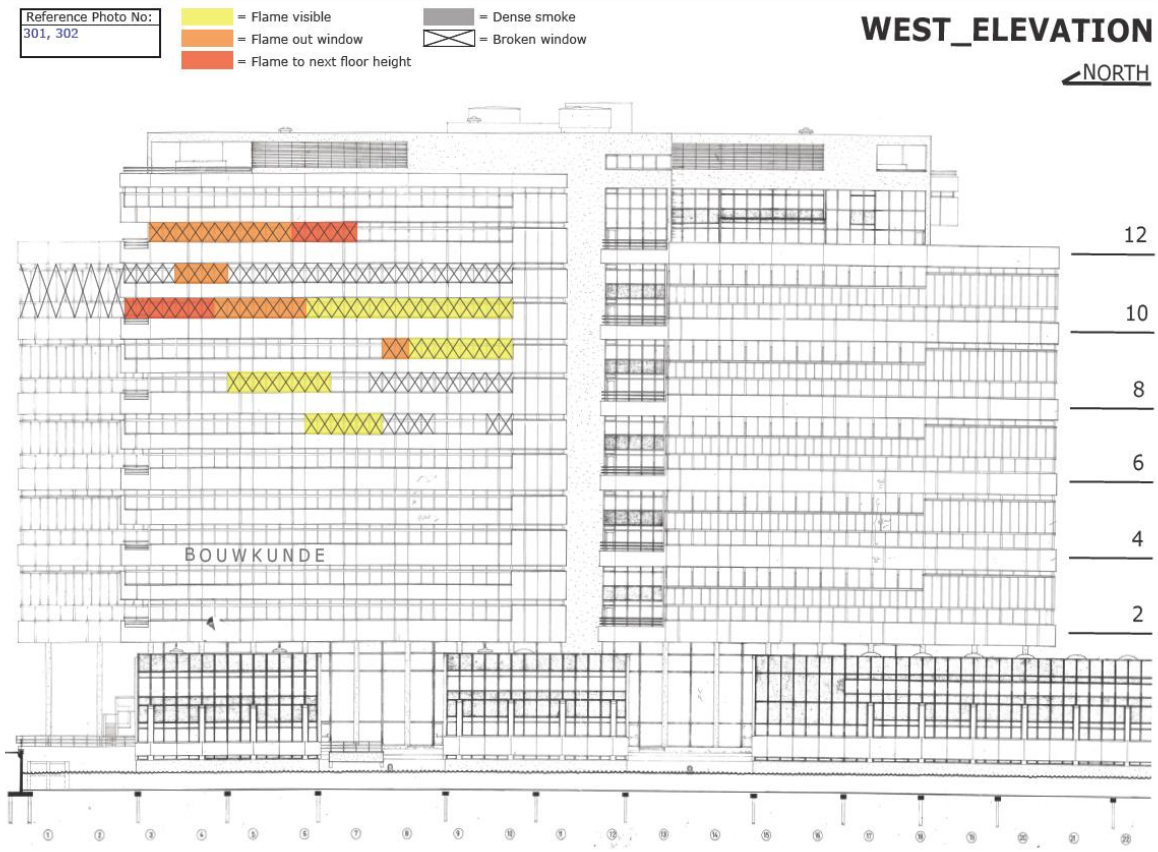


Figure 3.1 West Elevation at 2:35 pm

3.4.1 Observations

By studying the fire timeline graphics described above, several observations can be made. The fire originated at the south end of the 6th floor with flaming combustion starting around 9am. Recall that the collapsed wing was separated from the fire origin by 2 firewalls (illustrated in Figure 1.4). For the first 4 hours of the fire, these seemed to keep the fire out of the north end of the tower. The first visible flames from windows on the north end were seen in photos at 12:55pm on the 10th floor. It was difficult to determine whether the spread of flames was horizontally through the firewalls or around

the firewalls through the broken windows. In either case, after the firewalls were eventually breached, the fire quickly spread from the middle compartments to the north.

As would be expected, the vertical spread of flames was generally directed up. Flames were observed on the west elevation on every floor from the 7th floor upwards. By the time of the collapse (approximately 4:40pm) the fire had generally burnt out the entire north wing above the 6th floor, except for a small flame still visible on the 7th floor. Flames were not visible on the west elevations on any of the floors from the 6th and down until after the collapse.

Recall that the northwest wing of the FOA was comprised of an interior corridor linking offices with large windows on the west exterior. This portion of the structure burned from approximately 12:55 until the entire wing collapsed at approximately 4:40 pm. By examining the photos taken of the west side of the building during this time period, it is possible to estimate the duration of visible flames in various portions of the northwest wing. A few representative samples of approximate times of visible flames from various areas of Floors 7 to 9 are shown in Table 3.2.

Table 3.1 Observed Fire Duration Times on the West Side of the FOA

Floor	Grid Line	Times w/ Visible Flames	Approx. Fire Duration
7	3 - 4	3:05 - 3:36	0:31
	4 - 5	2:53 - 3:36	0:43
	5 - 6	2:46 - 3:11	0:25
	6 - 7	1:39 - 3:05	1:26
	7 - 8	1:39 - 2:46	1:07
	8 - 9	1:37 - 2:46	1:09
8	3 - 4	3:01 - 3:11	0:10
	4 - 5	2:35 - 3:11	0:36
	5 - 6	2:35 - 3:11	0:36

	6 - 7	2:35 - 3:11	0:36
	7 - 8	1:56 - 3:11	1:15
	8 - 9	1:37 - 3:01	1:24
9	3 - 4	3:01 - 3:11	0:10
	4 - 5	2:53 - 3:11	0:18
	5 - 6	2:53 - 3:11	0:18
	6 - 7	2:46 - 3:11	0:25
	7 - 8	2:20 - 3:05	0:45
	8 - 9	2:09 - 3:01	0:52
		AVERAGE	**0:42:33**

Taking an average of the length of time a given office is seen to burn, a preliminary estimate of the time the fire took to completely burn out a single office was developed. Considering only the offices on the floors where collapse appeared to initiate, this average time was calculated to be approximately 42 minutes . This will be used to validate the preliminary fire models that will be presented in the following chapter.

3.4.2 Key Events

Of particular interest in the photo database was photos taken at the time of the structural collapse.. Figure 3.2 is an example of a photo taken near the beginning of the collapse. In this photo, it appears that the columns near the center of the northwest wing were crushed during the collapse. These columns are on the 7th floor and are highlighted in Figure 3.2 What appears to be puffs of smoke in the highlighted area of the photo correspond to the exterior column locations. It is possible these “puffs of smoke” are in fact debris and dust from the crushed columns. Also note that the uppermost roof of the west wing was sagging quite a bit as the 7th floor columns appear to be failing.

Figure 3.2 NW Wing at the Onset of Collapse (©Rob Jastrebski)

From this image alone, it is difficult to tell whether the collapse initiated at the 7th floor or at higher levels. However, this was not the only image that captured the early stages of the collapse. Still shots taken from an aerial video of the collapse are shown in Figure 3.3. These are low resolution images, but give another insight into the collapse mechanism. The uppermost portion of the west wing is seen rotating about the spine of the building, possibly before the puffs of smoke signaling column failure appear. Whether it was the 7th floor columns or the roof that initiated the progressive collapse is still unclear. Regardless, the entire wing did collapse and the focus of this structural investigation described later in this thesis will be on floors 6-8.

Figure 3.3 Photos of Collapse (note the rotation of the upper NW wing)

3.5 POST FIRE PHOTOGRAPHS

The photos taken after the fire was extinguished also provide insight into how the structure may have actually failed. A few of these photos and their implications are discussed in this section of the report. Figure 3.4 shows an overall view of the entire west wing, shortly after the northern portion collapsed.

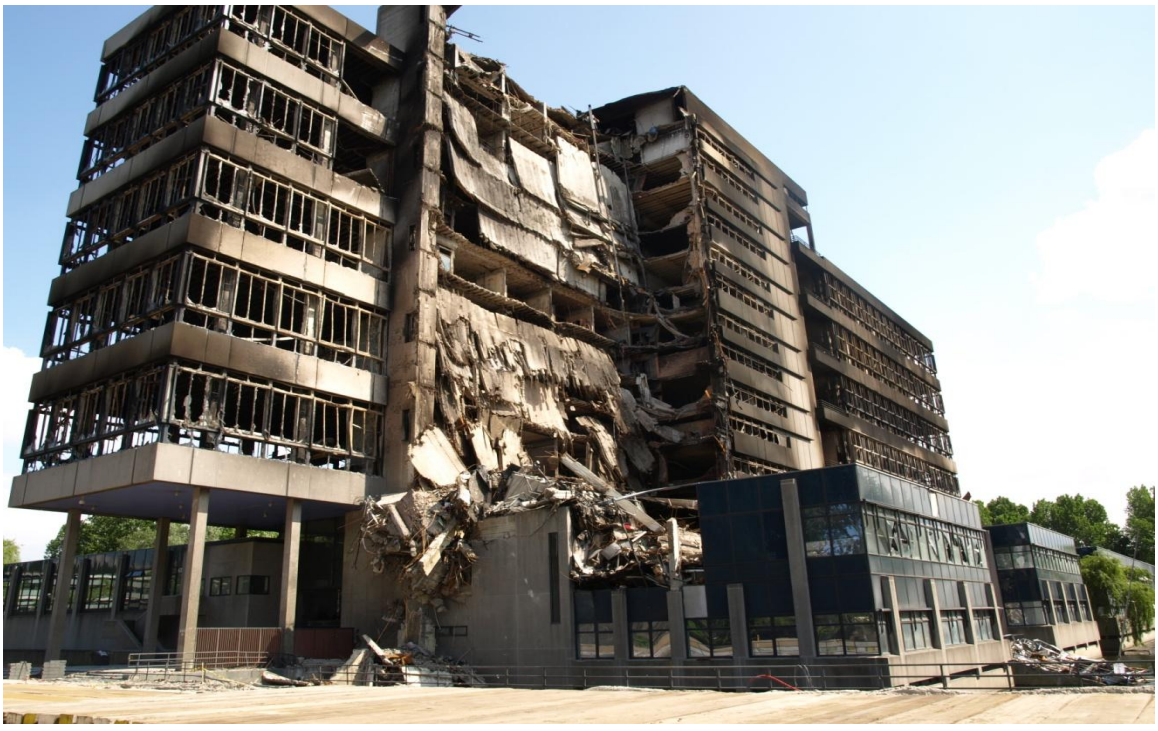


Figure 3.4 View From the Northwest Shortly After Collapse (©Rob Jastrebski)

3.5.1 Evidence of Spalling

One factor that may have possibly played a role in the collapse is spalling. Spalling occurs when the pressure of evaporating water within the concrete exceeds the tensile strength of the concrete. Spalling of cover leaves the reinforcement directly exposed to the fire and can significantly reduce the capacity of the section. This type of spalling is referred to as 'explosive spalling' and typically occurs in the early stages of heating. Spalling can also occur after prolonged exposure due to loss of bond with the reinforcement (Purkiss 2007).

As mentioned in the previous chapter, the type of aggregate used at the FOA was mostly likely river gravel, which is a siliceous aggregate. Research suggests that concrete made with siliceous aggregates are more prone to spalling than those made with calcareous aggregates. (Purkiss, 2007).

Some evidence of spalling was found in the post-fire photos. The photos in Figures 3.5 and 3.6 were taken from a crane during the demolition process and clearly show spalling. Figure 3.5 shows exterior columns with a majority of the cover of the corner bars spalled. Figure 3.6 shows joists with a similar pattern of spalling. With their covers removed, the bars would become directly exposed to the fire. Once directly exposed, the reinforcement's contribution to the member's strength would have decreased much more rapidly than when it was insulated by concrete. Note that the steel pieces shown in Figure 3.6 are the hangers that were used to support the mezzanine spaces. Despite the evidence of spalling seen in these photos, the contribution of spalling to the collapse is not known, and is a topic that merits further investigation in the future.

Figure 3.5 Exterior Columns After the Fire with Spalled Concrete Cover

Figure 3.6 Floor Joist After the Fire with Spalled Concrete Cover

3.5.2 Other Observations

Another photo of the northwest wing after the collapse is shown in Figure 3.7. This photo shows that each floor failed on a line at approximately the same place on each floor plate. In addition, the same photo shows that parts of the floor slab and joist system for each floor are still hanging, strung from the still-standing portion by continuous reinforcement. The one notable exception is seen at floor 7, where no hanging slab is visible. This could indicate that the floor joist system on that floor had failed prior to the failure of the columns. In this scenario, the missing joists at floor 7 would have left the columns unbraced between the 6th and 8th floors and more vulnerable to buckling. Detailed analysis of this scenario is beyond the scope of this thesis, but this scenario merits further investigation in the future.

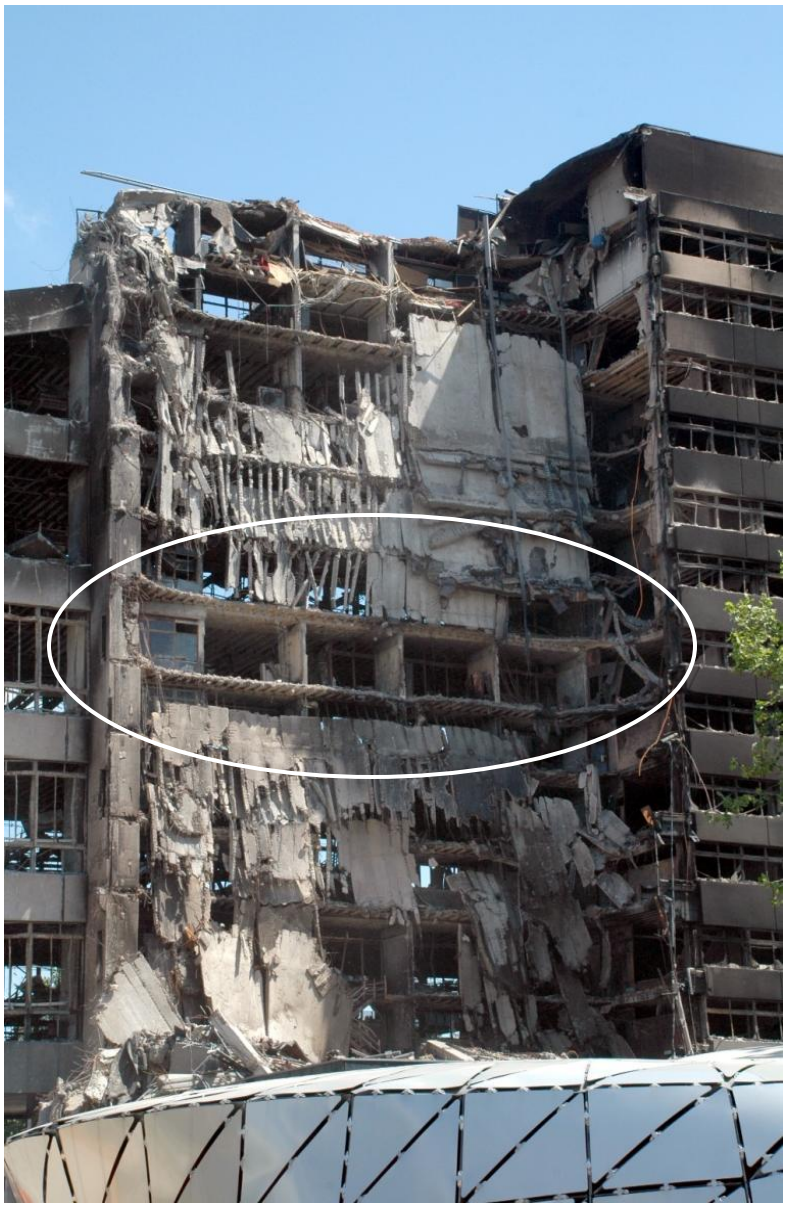


Figure 3.7 View of Northwest Wing After the Collapse Showing Hanging Floor Joists (©Rob Jastrebski)

Another important observation from the post-fire photos was the extent of window breakage. From photos like Figure 3.8, it can be seen that nearly every window in the tower was broken by the time the fire was extinguished. The broken windows provide a path for movement of the fire to higher floors and also affect the ventilation available to the fire, which is pertinent to the simplified fire modeling described in Chapter 4.

Figure 3.8 North-East Wing After the Fire (©Rob Jastrebski)

3.6 SUMMARY

In this chapter, the photo database has been discussed and a preliminary timeline of the fire has been established. Insights into the behavior of the fire and its effect on the structure have been described. In the next chapter, this information will be used to construct and validate preliminary fire models for the fire of May 13, 2008 at the Faculty of Architecture Building. These fire models will give an estimate of the temperature of the hot gases present in the structure through the fire event.

CHAPTER 4
Preliminary Fire Models

4.1 OVERVIEW

As part of the preliminary analysis of the structural collapse of the northwest wing of the FOA in the fire of May 13, 2008, an estimate is needed of the thermal environment to which the structural members were exposed. For the purposes of structural-fire analysis, the thermal environment is commonly characterized in the form of a gas temperature versus time curve (Buchanan 2002). In this chapter, a number of preliminary fire models will be examined with the objective of developing a gas temperature versus time curve that approximates the fire environment in the office spaces located in the northwest wing of the FOA. Gas temperature-time relationships will be developed using several different models and using several different assumptions on the fuel, ventilation and compartment characteristics for a typical office space in the FOA. These models are then refined and a final representative model of gas temperature versus time will be selected for use in the heat transfer analysis of the structural members described in the next chapter.

4.2 APPROACHES FOR DEVELOPMENT OF GAS TEMPERATURE VS. TIME CURVES

There are several approaches for estimating the gas temperature versus time curve for a fire in a building. The first approach is to simply assume a standard fire curve specified in building codes. Standard fire curves are typically used by building codes to establish a prescriptive hourly rating to a structural member. Normally, a single curve is used for all buildings, regardless of fuel load, ventilation characteristics and other building features that can affect the intensity and duration of a fire. These curves are therefore not 'case specific' and often do not provide a realistic representation of the

thermal environment of actual building fires (Wang 2002). Commonly used standard fire curves include those specified in ASTM E119 (ASTM 2008) and ISO 834 (ISO 1999).

A second approach for estimating gas temperature-time relationships is a compartment fire analysis. Compartment fire behavior has been extensively studied and there are many tools available for designers to quickly arrive at a solution. The tools available include computer software and published curves developed by analytical and empirical methods. Both types of methods will be discussed and applied to one office in the FOA.

A third approach for characterizing the thermal environment of fire is the use of computational fluid dynamics (CFD) models. This is done using one of a number of available computer programs, such as Fire Dynamics Simulator, FDS (McGrattan et al 2010). CFD analysis can provide the most realistic and accurate representation of the fire environment (Buchanan 2002, Purkiss 2002). However, CFD modeling is complex, requires highly specialized knowledge and experience to implement correctly, and is beyond the scope of this preliminary study. However, more detailed studies of the FOA fire in the future would likely benefit greatly by the use of CFD models to provide the most realistic representation of the fire environment.

4.3 STANDARD FIRE CURVES

As described earlier, standard gas temperature-time fire curves, such as those specified in ASTM E119 and ISO 834, are commonly used in building standards to characterize the fire exposure for structural members. These two curves are plotted in Figure 4.1. As can be seen from this figure, the ISO and ASTM standard fire curves are nearly identical. They both give a rapid rise in temperature, which begins to slow at around 15 minutes (or 700°C), then continue to heat at a slower rate. The curves reach a temperature of 1000°C at about 1.5 hours. While these curves are widely used for establishing structural-fire safety requirements in building codes, they do not provide a realistic representation of actual building fires. The intensity and duration of gas temperatures can vary widely depending on fuel, ventilation and other building

characteristics, and can differ significantly from the standard fire curves (Phan et al 2009). Further, the standard fire curves do not include a cool down phase of the fire, which can significantly affect structural response. Consequently, to estimate the gas temperature-time relationship for the FOA fire, compartment fire analysis will be conducted, and is described in the following section.

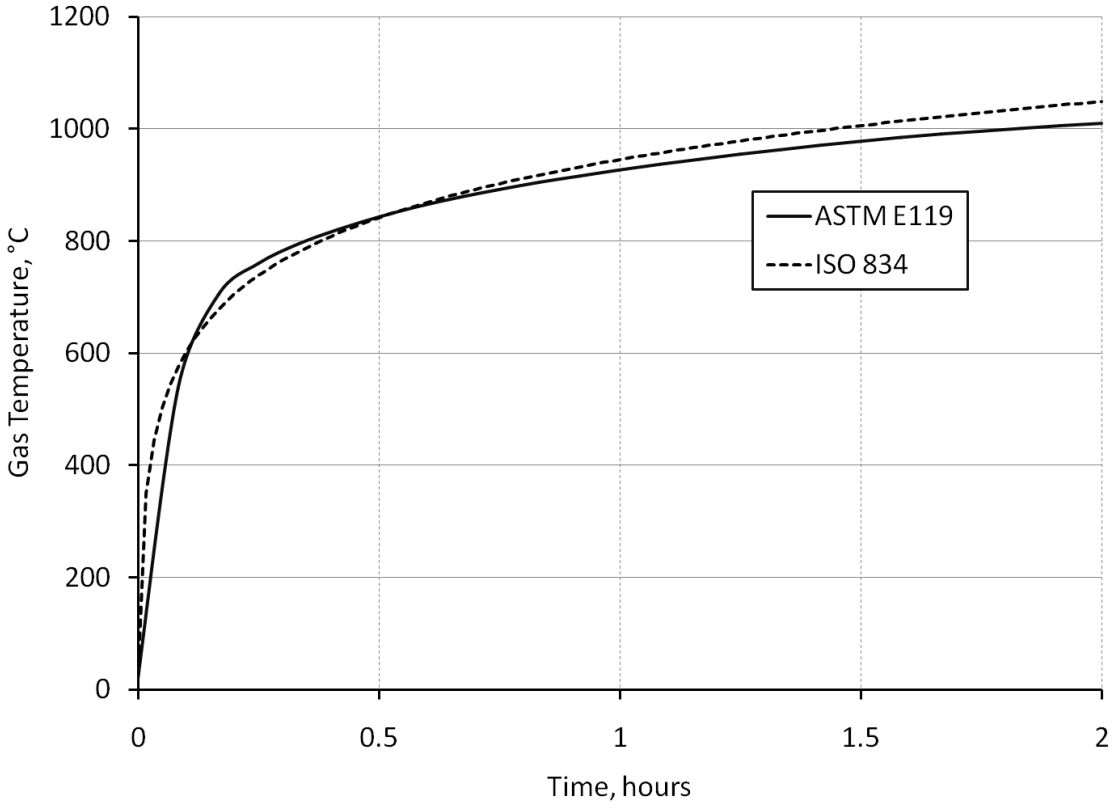


Figure 4.1 Standard Time-Temperature Curves

4.4 COMPARTMENT FIRE MODELS

A compartment fire model is a case specific fire model that can estimate the gas temperature time relationship for a compartment of given dimensions, fuel load, ventilation characteristics and thermal properties of the walls, floor and ceiling. To study the FOA fire, a typical office space will be treated as a compartment for a compartment fire analysis. This represents a simplification, since the photographic evidence suggests

that the office walls did not contain the fire within a single office compartment. That is, it appears that the fire was able to move from office to office by breaching the walls, which were not fire rated. Thus, it is not clear if an office can be treated as a single compartment for the purposes of a compartment fire analysis. Nonetheless, it is believed that a compartment fire analysis will provide a better estimate of the fire environment than the use of a standard fire curve and is the best estimate possible with the information available at this time and within the scope of this preliminary investigation.

The development of compartment fire models is described in numerous publications, including Drysdale (1998) Karlsson et al (2000) and Buchanan (2002). For purposes of structural-fire engineering analysis, one-zone compartment fire models are most commonly used to estimate gas temperature-time curves. One-zone models assume that the gas temperatures are the same throughout the entire compartment and can provide a reasonable representation of the conditions in a post-flashover fire (Drysdale 1998). All of the compartment fire models considered in this study are one-zone models.

4.5 BASELINE COMPARTMENT ANALYSIS

For the purposes of the following studies, one office of the FOA was chosen to represent a fire compartment. A number of gas temperature-time curves were developed using one zone compartment fire models for the compartment geometry, window layout, and boundary materials of a single office. Figures 4.5 and 4.6 show the drawings from which the dimensions and materials of the compartment were defined.

4.5.1 Approaches

There are two approaches for developing gas temperature-time curves based on one zone compartment fire analysis. The first is to use curves from published literature. These provide envelopes for compartments of any size or fuel load. They also account for the openings in the compartment by using a ventilation factor.

$$F_v = \frac{A_o\sqrt{H_o}}{A_t} \qquad (4.1)$$

Where F_v is the ventilation factor, A_o is the area of the opening, H_o is the height of the opening and A_t is the total area of the compartment (including walls and ceiling). It should be noted that some methods define the total area as the floor area while some methods are based on the total area bounding the compartment, including all 4 walls, floor, and ceiling. In case of an office at the FOA, the opening factor based on total area was computed using Equation 4.1 as follows:

$$F_v = \frac{4(1.1m * 1.8m)\sqrt{1.8m}}{134.8m^2} = 0.079\, m^{1/2}$$

The second approach is to utilize compartment fire computer programs. There is currently no single standard for these types of analyses and the results from each program can vary, sometimes dramatically. Some examples of compartment fire software include OZone (Cadorin et al 2001) and BRANZFire (Wade 2004).

4.5.2 Compartment Description

Figures 4.7 and 4.8 show the drawings for a typical office in the FOA. These were used to define the compartment for the preliminary fire models. The variables that typically need to be known about the compartment include: dimensions of compartment, dimensions and locations of openings, compartment boundary conditions (materials & thicknesses), and combustible objects in the compartment (or evenly spread fuel load).

The floor area of an office was 35.2m^2 and it was 2.7 meters from the top of the slab to the bottom of the suspended ceiling system. All of the offices were on the exterior of the building, and all had four windows to the exterior. The walls separating the offices were not meant to prevent the spread of fire. They did not extend beyond the top of the suspended ceiling, but it can be assumed that they still had some resistance to flame spread. These partition walls were stud walls, constructed of wood studs covered by drywall. The interior walls separating the offices from the hallway were a wood book shelving system, with a clerestory opening, covered in glass. This interior wall and

window can be seen in Figure 4.2, a photo taken from the corridor that connected the offices. Also note Figures 4.3 and 4.4, which show the position of an office in the tower and a view of one of the offices after collapse.

Figure 4.2 Interior Office Corridor

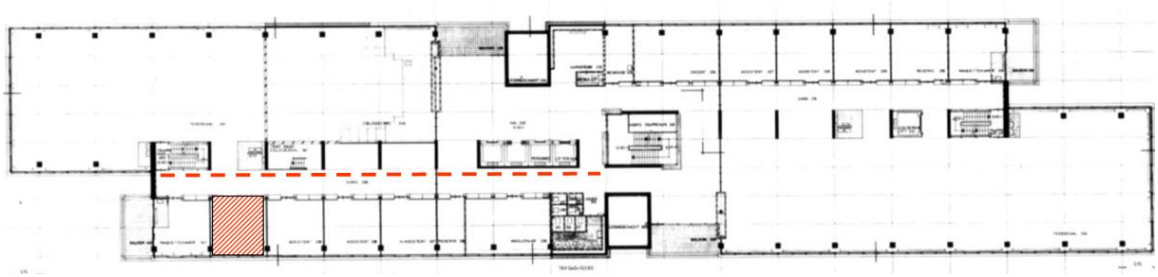


Figure 4.3 Plan View of One Office in Context with the Rest of the Floor

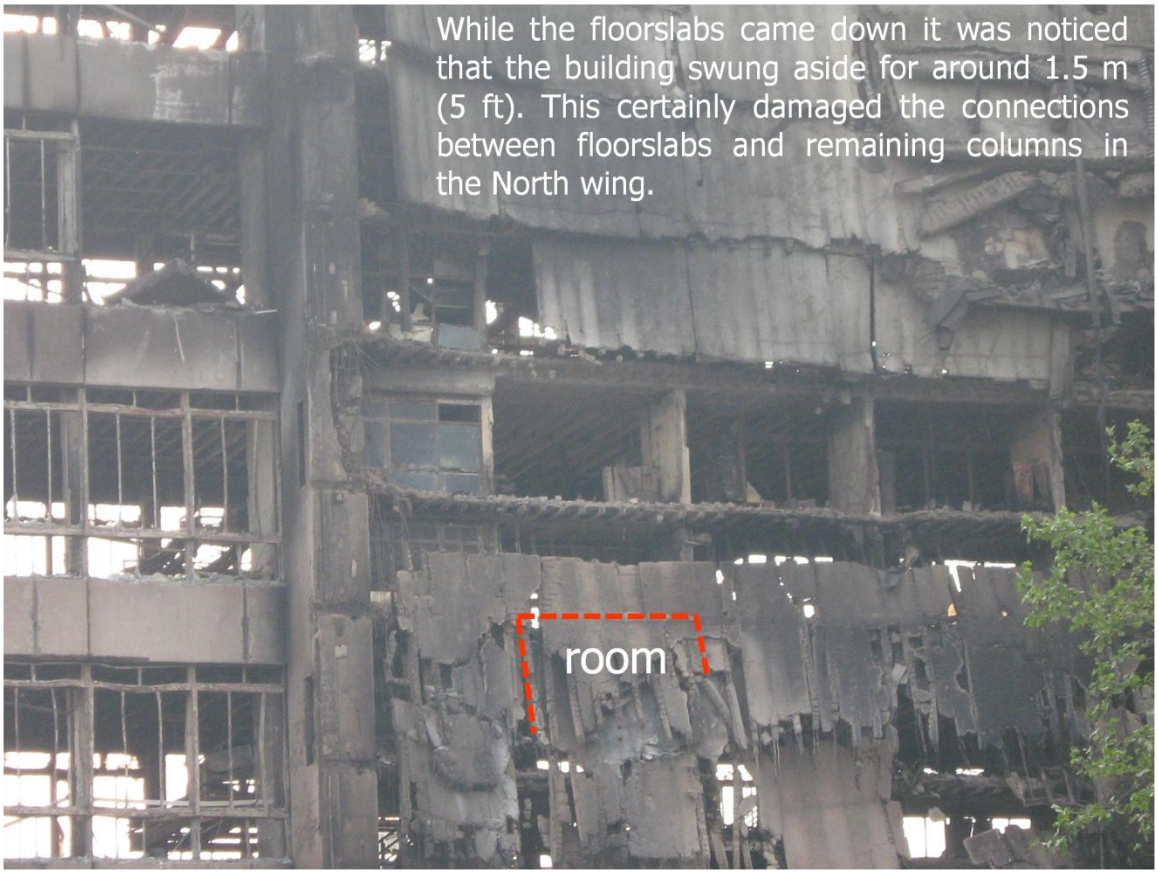


Figure 4.4 Office After Collapse (Van Weeren, 2008)

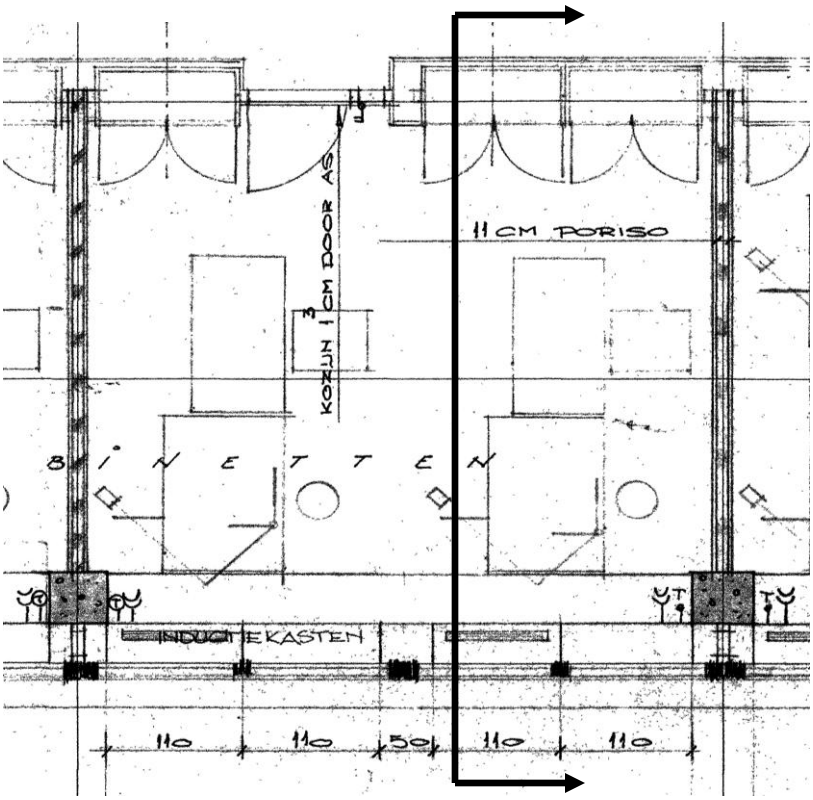


Figure 4.5 Typical Office Architectural Plan

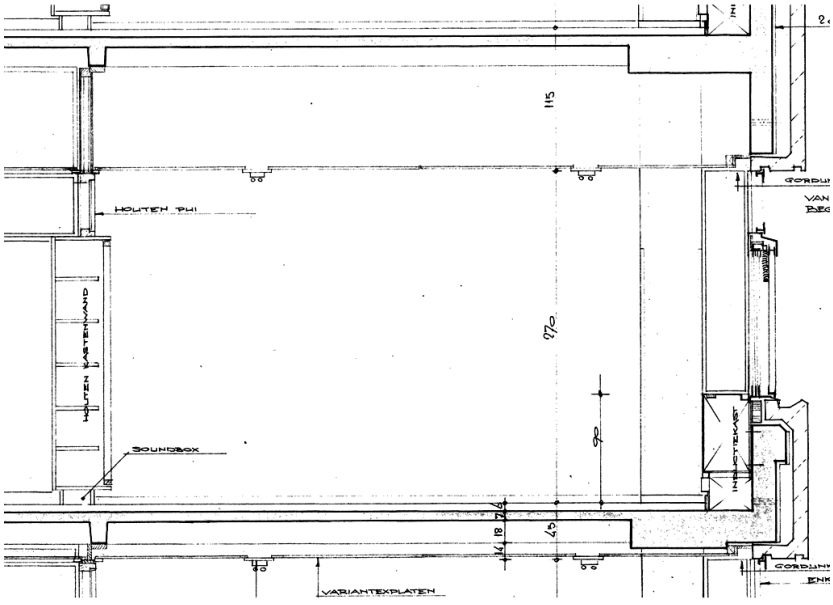


Figure 4.6 Typical Office Architectural Cross-Section

4.5.3 Analysis Results

The gas temperature-time curves developed for the baseline compartment using different methods is presented in Figure 4.7. A description of each of the methods is provided in the following sections.

4.5.3.1 Swedish Curves

Most commonly referred to as the 'Swedish Curves', these are the work of Pettersson, Magnusson and Thor and represent the results of years of study of compartments fires for the Swedish Institute of Steel Construction (Buchanan 2002). The Swedish Curves are based on a few simply calculated variables, such as the total area ventilation factor and fuel load. Tables exist to provide the estimates time vs. temperature curve given a particular total area ventilation factor and fuel load. These equations can also be modified for different boundary materials.

4.5.3.2 Eurocode Parametric Fire

Parametric fires give an equation for the time vs. temperature curve, based on variables that change according to the parametric method used. The Eurocode Parametric Fire Method is presented in EN1992 (Eurocode 2, 2004). The primary equation used to define this fire is:

$$T(°C) = 1325\left(1 - 0.324e^{-0.2t^*} - 0.204e^{-1.7t^*} - 0.472e^{-19t^*}\right) \qquad (4.2)$$

$Where$: $Ficticous\ Time$: $t^* = \gamma t = 1.454t$

$$Ficticous\ Time\ Factor:\ \gamma = \frac{(F_v/0.04)^2}{(b/1160)^2} = 1.454$$

$$\sqrt{Boundary\ thermal\ inertia}:\ b = \sqrt{k\rho c_p} = 1900\frac{\sqrt{W}}{m^2K}\ (From\ Eurocode)$$

$$Duration\ of\ fire:\ t_d = \frac{0.00013e_t}{F_v} = 0.429\ \text{hours}$$

$$Fire\ Load\ Density:\ e_t = \frac{1000\text{MJ} \cdot 35.2\text{m}^2}{134.8\text{m}^2} = 261\ MJ/m^2$$

$$Slope\ of\ Decay\ Phase:\ \frac{\text{dT}}{\text{dt}} = 625° \frac{\text{C}}{\text{hour}}\ (\text{since duration}\ \leq 30\ \text{min})$$

$$Time\ to\ Extinguishment:\ t = \frac{T_{max} - 20°\text{C}}{625°\text{C}/hr} + 0.429hr = 1.763\ hours$$

4.5.3.3 Lie's Parametric Fire

Another parametric fire reported in the literature was developed by Lie (1995). The equations look similar to the Eurocode Parametric equations, though they each produce different results. The main equation used to define Lie's Parametric Fire is:

$$T(°C) = 250(10F_v)^{0.1/F_v^{0.3}} * e^{-F_v^2 t} * [3(1 - e^{-0.6t}) - (1 - e^{-3t}) + 4(1 - e^{-12t})] \quad (4.3)$$

$Where$: $Duration\ of\ fire:\ \tau = \frac{Q_t A_v}{R} = 0.539\ hours$

$Equivalent\ wood\ fire\ load\ density:\ Q_t = 261 \frac{MJ}{m^2}\left(\frac{1}{18.6}\right) = 14.0 \frac{kg}{m^2}$

$Rate\ of\ Burning:\ R = 330 A_o \sqrt{H_o} = 2506\ kg/hr$

$Decay\ phase:\ T(°C) = T_{max} - 600\left(\frac{t}{\tau} - 1\right)$

4.5.3.4 OZone

OZone (Cadorin et al 2001) is a one zone compartment fire computer program, developed at the University of Liege, Belgium. The geometry, openings, and boundary materials were input into the model and the default fuel load for an office occupancy (511 MJ/m^2) was used. Figure 4.7 shows a comparison of the results of each of these compartment fire modeling methods.

4.5.3.5 Observations

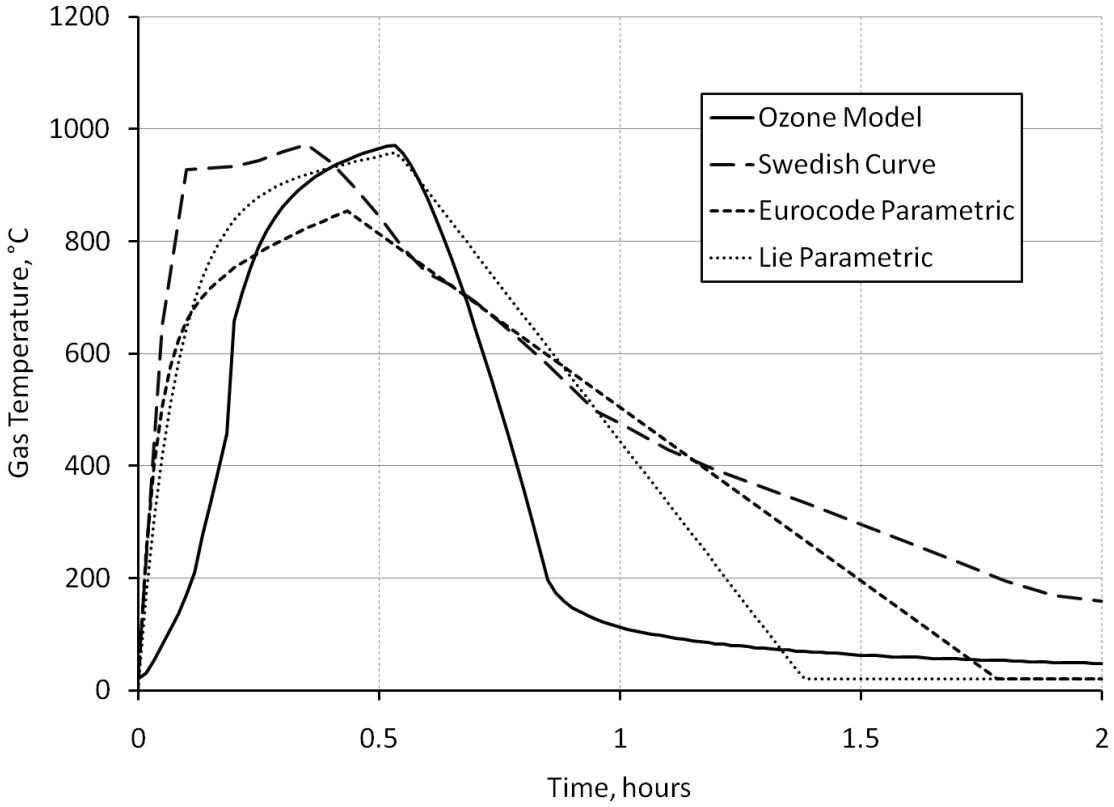


Figure 4.7 Compartment Fire Results

Table 4.1 Compartment Results Comparison

Method	Max Temperature (°C)	Duration of Intense Burning (hours)
OZone	971	0.62
Swedish Curves	972	1.15
Eurocode Parametric	854	1.14
Lie Parametric	958	0.99

Table 4.1 summarizes the maximum gas temperature and duration of intense burning predicted by each of the four methods described above. The duration of intense

burning, for purposes of this study, was defined as the duration time for which the temperature exceeded 400°C. Although somewhat arbitrary, this definition was chosen because the strength of steel and concrete are generally not significantly reduced until temperature exceeds about 400°C. Note that the four methods considered give reasonably similar gas temperature-time curves. The maximum gas temperature predicted by the four methods ranged from approximately 850°C to 970°C. The predicted duration of intense ranged approximately 0.6 hours to 1.15 hours. Recall that the average length of burning in the offices observed from photographs, as discussed in Chapter 3, was 0.7 hours which is in reasonable agreement with the model predictions The closest fire model to the average observed time was the OZone model. OZone will be used for the remainder of the analysis.

4.6 COMPARTMENT SENSITIVITY STUDIES USING OZONE

The gas temperature vs. time curves computed using Ozone and the other methods described above are influenced by a number of variables that are uncertain or unclear in the case of the FOA offices. With this in mind, sensitivity studies were conducted using the OZone compartment computer software to help identify the most influential variables on the calculated gas temperature vs. time curves. In each study, one variable of the baseline compartment used in the previous section was changed while holding the remainder constant. Note that the comparison gas temperature vs. time curves provided in this section show the baseline compartment curve as a solid line. The maximum temperature reached in the baseline compartment analysis using Ozone was 971°C.

4.6.1 Area Sensitivity

The first sensitivity study focused on the area to be defined as a single compartment. The offices cannot be immediately assumed to burn independently because the partitions separating them were not fire rated and did not extend to the bottom of the floor slab above.

An assumption of one zone compartment fire analysis is that the compartment can be well-defined. In general, the accuracy of results will decrease as the area of the compartment is increased. If the area is sufficiently large, the behavior of fire development and spread is not like that of a single compartment with uniform temperature throughout, but rather like the progressive burning of a deep room illustrated in Figure 4.10. A typical limiting floor area value for which one zone compartment analysis provides a reasonable representation for a post-flashover fire environment is is around 100m^2 (Buchanan 2002). The floor area for each office of the FOA was around 35 m^2. Figure 4.10 shows how different portions of a sufficiently large compartment may not burn at the same rate. Figure 4.11 shows the results from OZone, obtained by changing the number of offices assumed to burn as a single compartment. In general, as the number of offices assumed to burning as a single compartment increases, the maximum temperature and duration of intense burning tend to decrease.

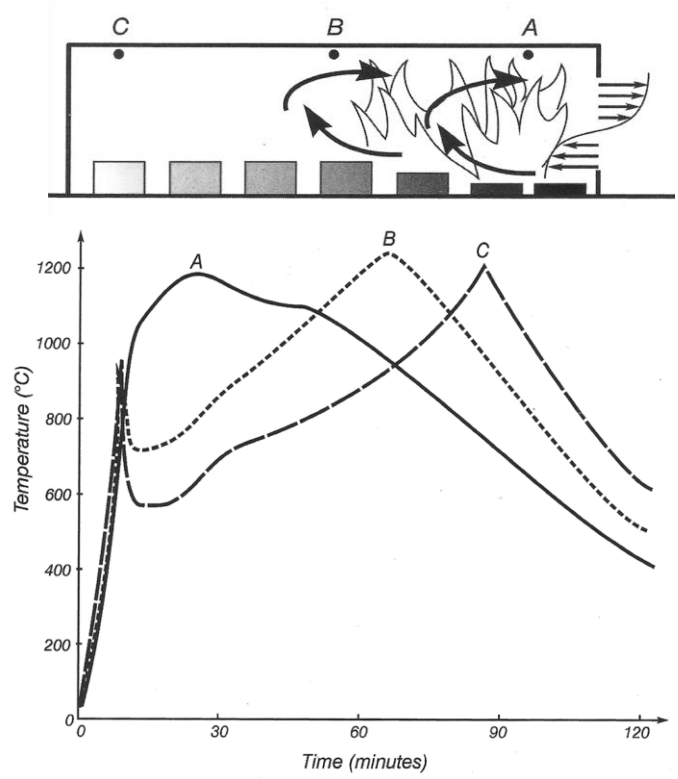


Figure 4.8 Progressive Burning in a Deep Room (Buchanan)

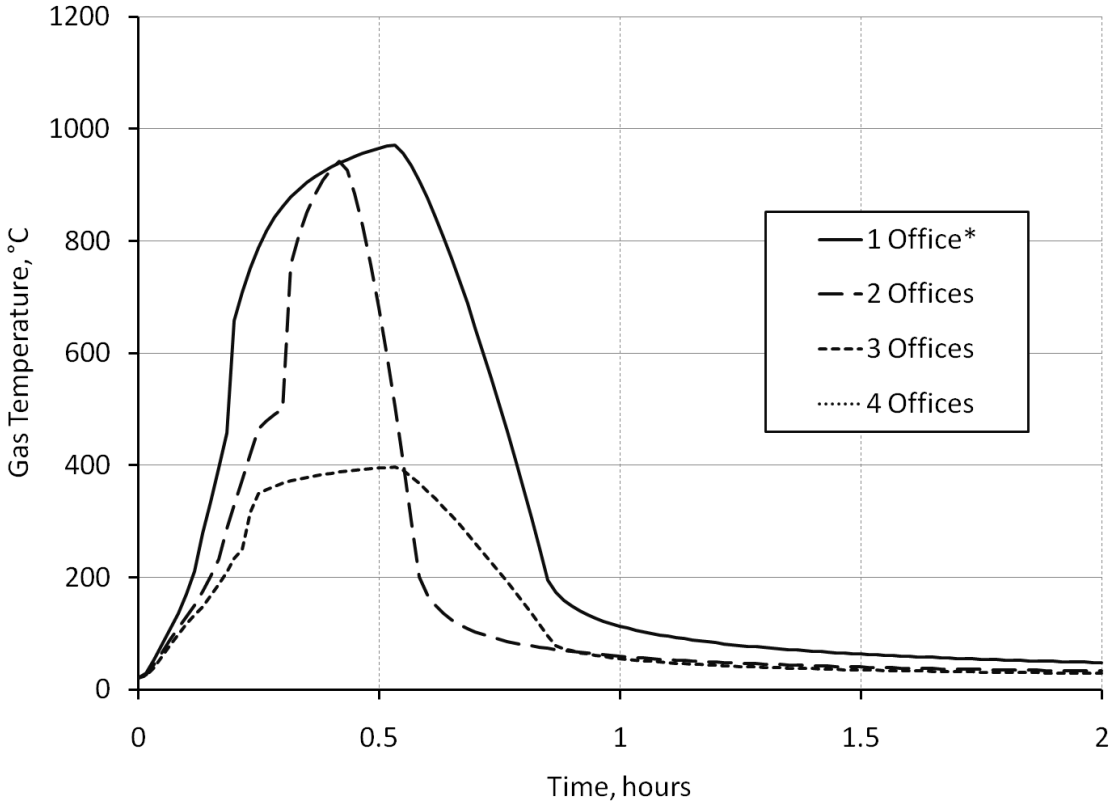


Figure 4.9 Compartment Area Sensitivity Study Results

4.6.2 Height Sensitivity

The ceiling height varied throughout the FOA. The office wings, where the fire and collapse appear to have initiated, varied in height from floor to floor. The odd floors, starting at the tower portion of the structure, were 3.15 meters floor to floor. The even floors were slightly taller, at 3.85 meters. In addition, the in-use building featured suspended ceilings of unknown composition. These ceilings were constant at a height of 2.70 meters, which is the value that was used for the baseline compartment.

In the studio spaces the floor to floor height of the rooms was 7 meters. From photos of the building before the fire and architectural drawings, it can be seen that these spaces also featured the suspended ceilings present in the office portion of the building.

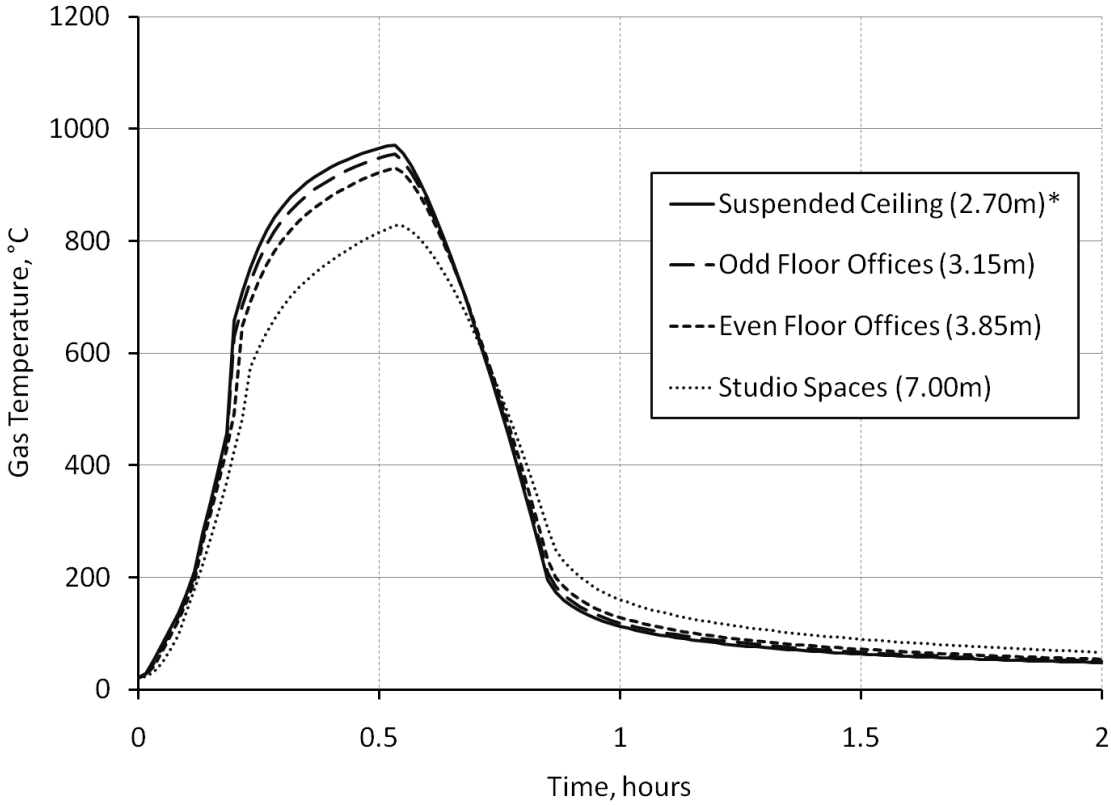


Figure 4.10 Compartment Height Sensitivity Study Results

Figure 4.10 shows that, while holding everything else constant, the height of the compartment does not have a very large effect on the predicted gas temperature-time curve.

4.6.3 Opening Sensitivity

Each office in the FOA had four windows above a sill at 0.90 meters above the finished floor. Each window was 1.10 meters wide by 1.80 meters tall. These windows could be opened manually and many were broken during the course of the fire. As noted in Chapter 3 of this paper, the windows in the portions of the building where the fire burned most intensely were almost all completely broken. The baseline model assumed all the windows were initially broken.

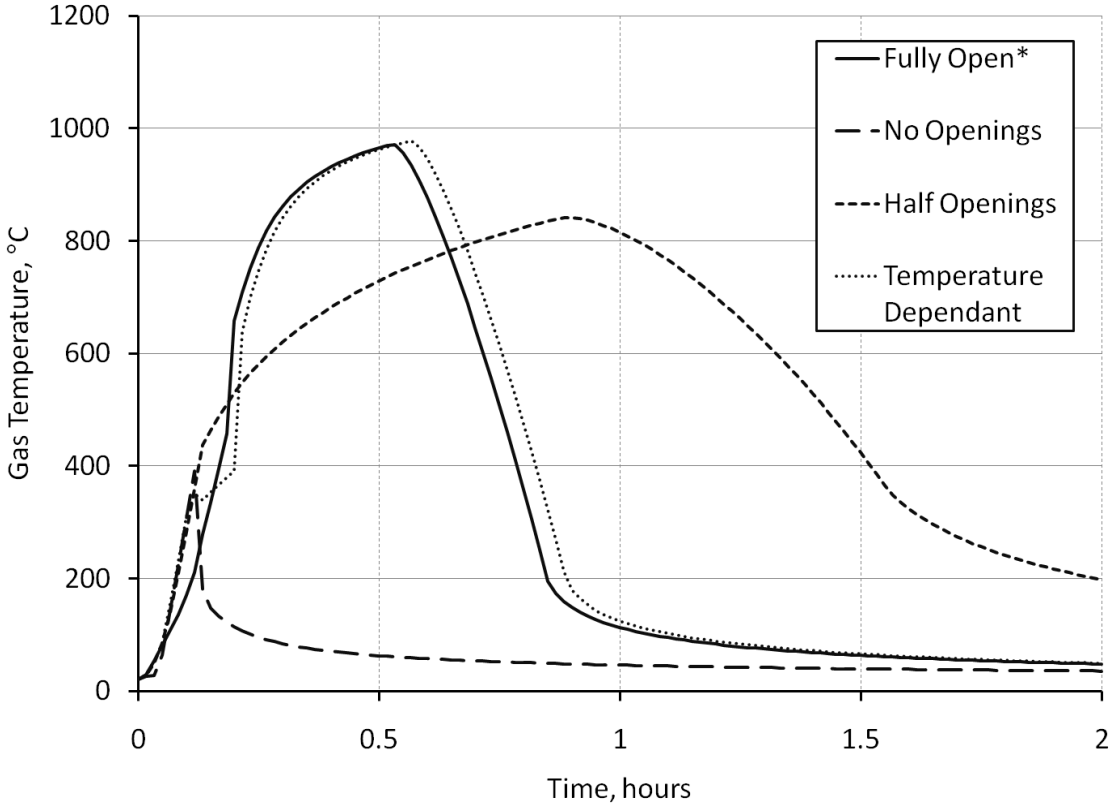


Figure 4.11 Compartment Opening Sensitivity Study Results

As can be seen in Figure 4.11, the thermal environment is very sensitive to the openings of the compartment. When no openings are present (the windows start closed and do not break during the fire) the fire quickly consumes the available oxygen and dies out before reaching flashover. In this case the peak temperature is only 391°C. When only half the openings are present (2 of the 4 are assumed open or broken) then length of burning is dramatically increased and the peak temperature is reduced to 841°C. The curve labeled "Temperature Dependent" refers to a compartment with closed windows that break when the glass reaches a specified temperature. In this case a temperature of 300°C was assumed to be the threshold. This is the default value provided by OZone. This scenario is probably the most realistic, as the observations showed windows breaking as the fire intensity increased. As can be seen in Figure 4.11, the curve of the temperature dependent openings transitions as expected from the 'No Openings' curve to

the 'Fully Open' curve at around 0.2 hours (or when the gas temperature reaches around 350°C).

4.6.4 Boundary Material Sensitivity

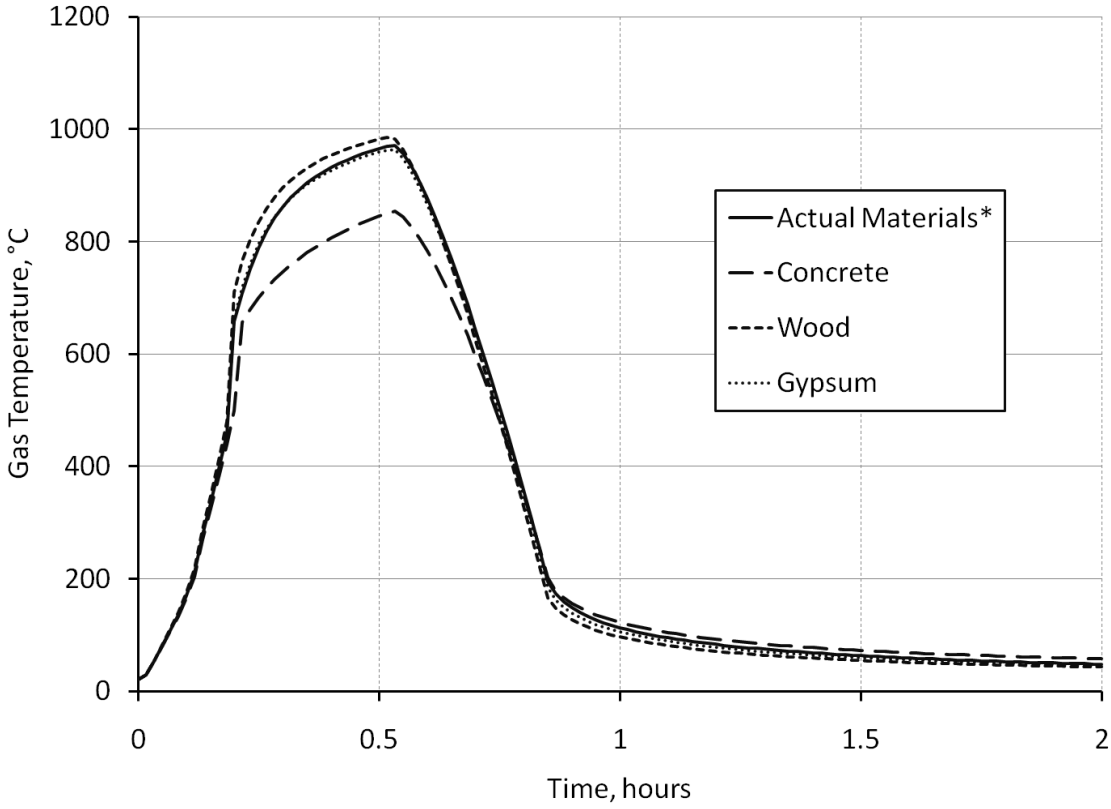


Figure 4.12 Compartment Boundary Material Sensitivity Study Results

One of the factors considered in a compartment fire analysis is the thickness and thermal properties of the walls, floor and ceiling bounding the compartment. These affect heat transfer from the compartment to the surroundings. The OZone analysis was repeated using several different assumptions regarding the boundary materials. The results plotted in Figure 4.12 show that the boundary materials have little effect on the predicted gas temperature-time curves.

4.6.5 Fuel Load Sensitivity

For the baseline compartment analysis, a value of a fuel load density equal to 511 MJ/m^2 was used in the OZone model. This is the default value recommended in OZone for an office occupancy. It would be difficult to accurately estimate the true fuel load and it can be assumed the fuel load density varied through the building spaces and even between offices. The OZone analysis was repeated for several other fuel load densities in the range of 400-800 MJ/m^2 and the results are plotted in Figure 4.13. The fuel load densities recommended for an office occupancy from several sources are compared in Table 4.2. It should be noted that some sources define fuel load based on the floor area and some use the total area bounding the compartment, including all 4 walls, floor, and ceiling. Only fuel load densities based on floor area are shown.

Table 4.2 Fuel Load Density Recommendations

Source	Occupancy	Fuel Load Density (MJ/m^2)
OZone	Office	511, floor area
CIB, 1986	Office, business	800, floor area
CIB, 1986	Office, engineering	600, floor area
CIB, 1986	School	300, floor area
Eurocode	Office	420, floor area
NFPA	Office	1000, floor area

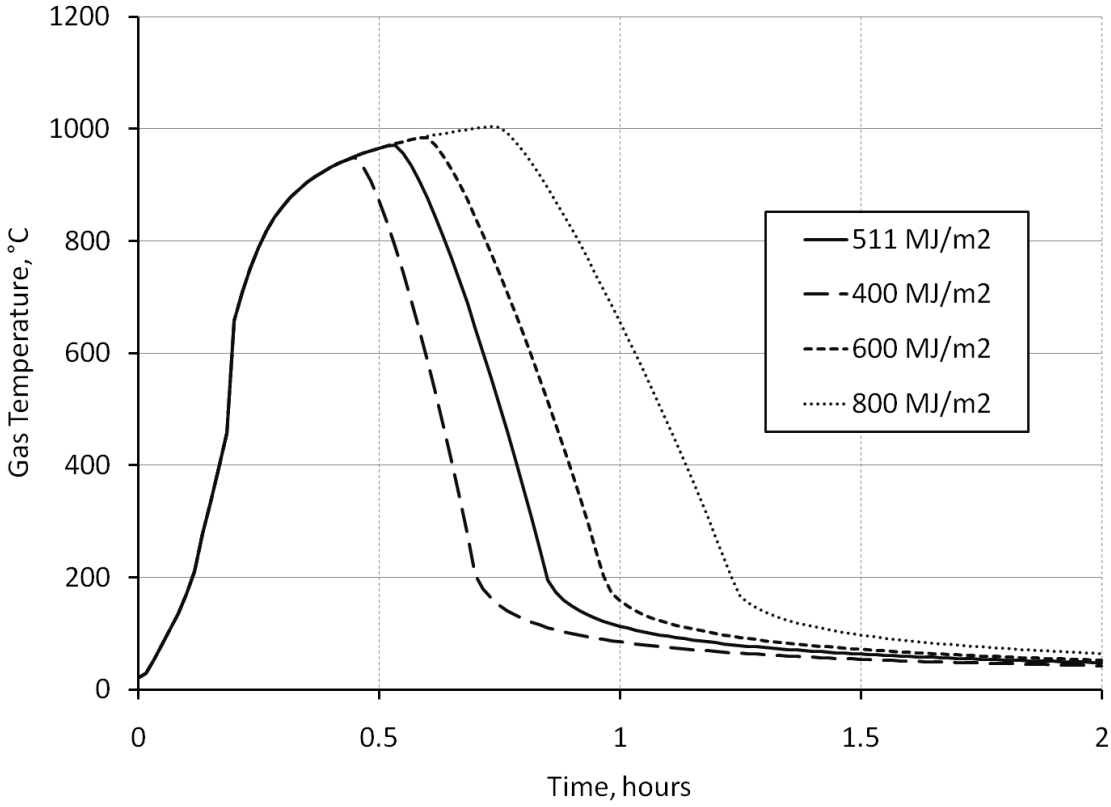


Figure 4.13 Compartment Fuel Load Sensitivity Study Results

As can be seen in Figure 4.13, as the fuel load density increases, both the maximum temperature and duration of intense burning increase.

4.7 COMPARISON WITH OBSERVATIONS

As indicated by the sensitivity studies described in the previous section, several factors have a significant influence on the predicted gas temperature-time curve. More specifically, the three factors that had the largest influence on the gas temperature-time curve was the compartment area (the number of offices assumed to form a single compartment), the window openings and the fuel load density. The values for these variables that most realistically represent the conditions in the FOA on May 13, 2008 are uncertain. Consequently, the most realistic gas temperature-time curve is also uncertain. Observations from photographs and other information available on the building and its

contents can be used to a limited extent to judge the validity of various model predictions. As described in Chapter 3, the photographic evidence suggests that the duration of burning in any given office, based on observations of visible flames, averaged about 0.7 hours. Photos also provide some information on window breakage. This limited observational evidence tends to support the gas temperature-time curve developed for the baseline compartment, or reasonably close variations of this curve. On the other hand, the maximum temperatures predicted by the various models and sensitivity studies can vary widely for similar durations. Consequently, while the photographic evidence provides some validation of the duration of burning predicted by the baseline compartment model, there is considerably less confidence in the predicted maximum temperature.

Figure 4.14 shows each of the preliminary fire models developed in this chapter. As can be seen in the plot, the ASTM E119 standard curve, which is used for the determination of a reinforced concrete member's prescriptive hourly rating in the US, varies significantly from the more realistic gas temperature-time curves generated by compartment fire analysis. Although the different compartment fire models did not produce identical results, observations can still be made about the trends of the predicted gas temperature-time curves. Each method estimates that the highest temperatures occur at around half an hour into the fire. Also, each case shows the majority of the cooling to be complete after one hour. If we define the period of intense burning to be while gas temperatures are above 400°C in the compartment, then most models predict a period of intense burning in the range of 0.6-1.2 hours. This is reasonably consistent with observations from the photo timelines presented in Chapter 3, where the average office burned for approximately 0.7 hours.

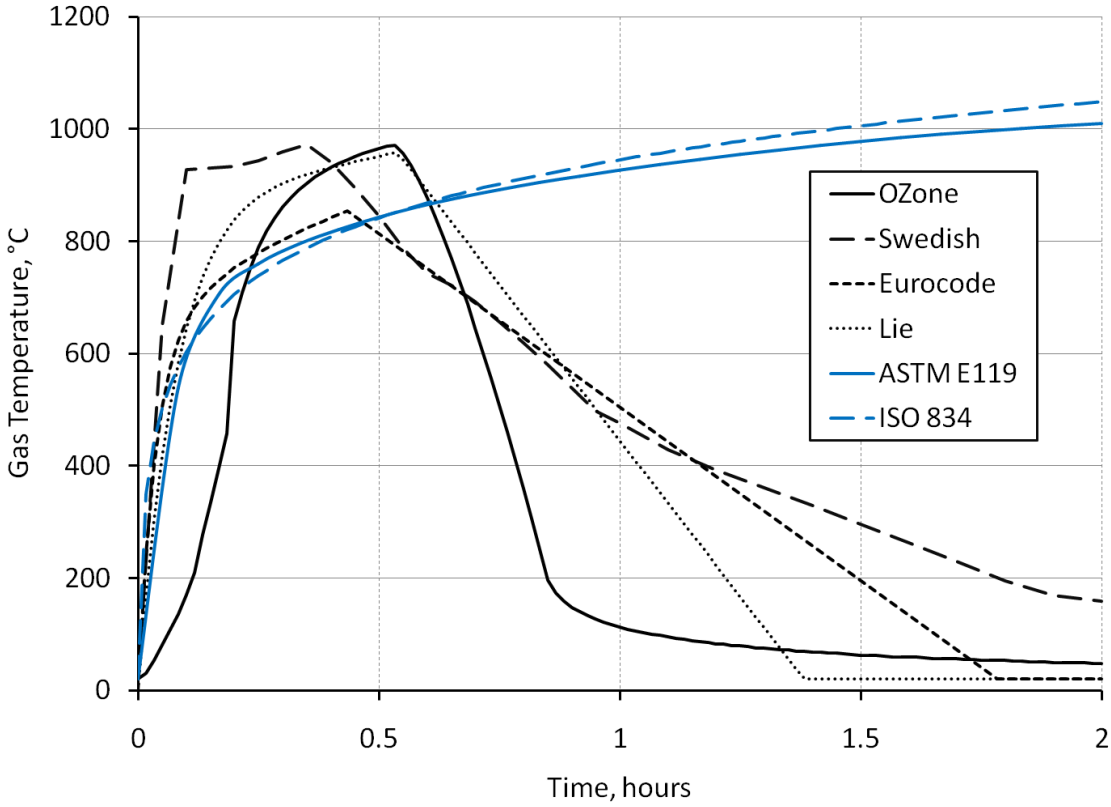


Figure 4.14 Preliminary Fire Analysis Results

Based on the available information, it is not possible to assess the accuracy of the predicted curves relative to one another. Further analysis of heat transfer and structural response in this preliminary study, as described in the following chapters, will be based on the gas temperature-time relationship developed by OZone for the baseline compartment. However, the uncertainty in this preliminary fire analysis should be kept in mind when interpreting the results of the subsequent analysis. For further detailed investigations of the FOA fire, more detailed analysis of the fire environment would be highly beneficial.

CHAPTER 5
Heat Transfer Analysis

5.1 OVERVIEW

In the previous chapter, simplified compartment fire models were evaluated to estimate the gas temperature-time relationship for a typical office in the FOA for the May 13, 2008 fire. The gas temperature-time curve computed using OZone for the baseline compartment was selected as representative of the fire environment, and will be used for further analysis. In this chapter, heat transfer analysis will be conducted for selected structural members of the FOA to determine the temperature distribution within the member cross-section as a function of time during the fire. These computed temperature distributions will then be used in subsequent chapters to estimate the loss of cross-section capacity as a function of time during the fire.

Heat transfer analysis of structural members during a fire is a complex subject and a discussion of the underlying theory is beyond the scope of this thesis. In short, however, the typical approach involves first estimating the radiative and convective heat transfer from the fire environment to the surface of the structural member, based on an assumed gas temperature-time curve. This is followed by a heat conduction analysis of the member to determine temperatures at various locations within the member as a function of time. More detailed discussions of heat transfer analysis for structural-fire engineering problems can be found in several references, including Buchanan (2002), Purkiss (2007), Wang (2002) and SFPE (2002).

For the purposes of this thesis, two-dimensional heat transfer analysis will be conducted for structural members. This analysis will provide the distribution of temperature throughout the member cross-section as a function of time. However, variations of temperature along the length of the member cannot be computed from this approach. Temperature variation along the length of a member requires a three-

dimensional heat transfer analysis, which is beyond the scope of this thesis. Note that a three-dimensional heat transfer analysis also requires more detailed information on the variation of gas temperatures along the length of the member. This information is not available from the one zone compartment fire model being used in this study.

Heat transfer analysis in this thesis will be conducted using the computer program, SAFIR2007 (Franssen, 2007). SAFIR is a heat transfer and structural analyses program developed at the University of Liege, Belgium. For this study, only the heat transfer analysis capabilities of SAFIR will be used. The use of SAFIR is facilitated through the pre and post processors, which are briefly described later in this chapter.

Heat transfer analysis will be conducted for the columns and joists of the 6th to8th floors of the FOA, where the collapse potentially initiated. The columns and joists in other parts of the FOA were similar to those that will be analyzed. Consequently, the heat transfer analysis of the selected members should provide considerable insights into other portions of the FOA.

5.2 SAFIR MODEL

SAFIR 2007 is a finite element computer program capable of both heat transfer analysis and structural analysis for structural-fire engineering applications. As noted above, only the heat transfer capabilities of SAFIR 2007 were used in this study. A SAFIR2007 heat transfer analysis requires a text input file that contains the information about the section and fire being considered. This text file can be written manually or created using pre-processing software. The preprocessor created with SAFIR at the University of Liege is called Wizard (Franssen, 2007). This program works well for standard steel sections, but lacks the capacity to easily create reinforced concrete sections for heat transfer analyses. This limitation prompted the development of a new pre-processing program that allows the user to quickly create the text input file for any structural element. The following section will briefly describe the input file setup program developed for and used by this study, UT Fire (Jennings, 2009).

5.2.1 UT Fire

A companion project to this study was the development of a preprocessor , UT Fire, that is compatible with the heat transfer software SAFIR2007. Using UT Fire, models were constructed to estimate the temperature profiles through various members in the FOA at the different times during the fire event. Three member types were selected as the representative samples for heat transfer analysis using UT Fire and SAFIR 2007. These were: a 500 mm square column, a 250 mm deep floor joist and a 400 mm joist. The required input for UT Fire will be discussed in the following sections.

5.2.1.1 Thermal Properties of Materials

UT Fire requires a several thermal properties of the material(s) of which the section is comprised. These parameters vary, but those required for siliceous concrete are listed in Table 5.1. The values used for this analysis come from the recommendations from Eurocode 2.

Table 5.1 Thermal Properties of Siliceous Concrete (as recommended by Eurocode)

Parameter	
Aggregates	Siliceous
Moisture Content	33 kg/m^3
Convection Coefficient on Hot Surfaces	25 W/m^{2}K
Convection Coefficient on Hot Surfaces	10 W/m^{2}K
Relative Emissivity	0.5

5.2.1.2 Section Finite Element Meshes

UT Fire generates a finite element mesh that is based on the geometry of the section and maximum mesh size specified by the user. In general, as the mesh is made finer (smaller elements), the results of the analysis become more accurate. However,

when the mesh becomes very fine, the analysis can take a long time to run. 10mm square elements were used in the columns and 5mm square elements were used in the joists. This was deemed adequate for the purposes of this study. Also, some simplifications were used in the models. For example, the slight taper of the joist webs were ignored and the sections created were homogenous and completely composed of concrete. The small amount of reinforcement (when compared with the volume of concrete) would make very little difference in the final heat transfer results. However, the temperature of the reinforcement can still be determined from this analysis, as long as the location of the reinforcement within the cross-section is known. Screen shots of the finite element meshes created with UT Fire for the three types of elements studied are shown in Figures 5.1 to 5.3.

5.2.1.3 Input Fire

UT Fire also allows the user to specify a gas temperature-time curve that defines the fire exposure for the member being analyzed. The user can specify a standard curve, such as ASTM E119, or can input any desired gas temperature-time curve. As noted earlier, the gas temperature-time curve generated by OZone for the baseline compartment was used for this analysis, and therefore was provided as input to UT Fire. UT Fire also allows the user to specify which faces of the structural member are exposed to the specified gas temperature-time curve. For the analysis conducted for this study, it was assumed that the column section was exposed on all four sides. For the joists, it was assumed that only the bottom and sides of the stem and the bottom of the slab were exposed to fire. It was assumed that the top of the slab was exposed only to room temperature. The sides of the floor slab were modeled with boundary conditions that replicated the continuity in the floor system.

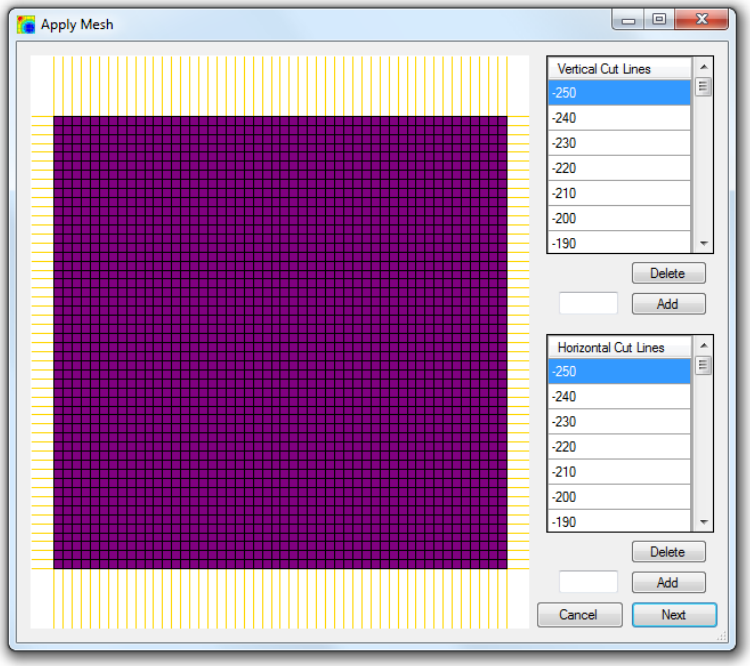


Figure 5.1 Column (500mm) Finite Element Mesh

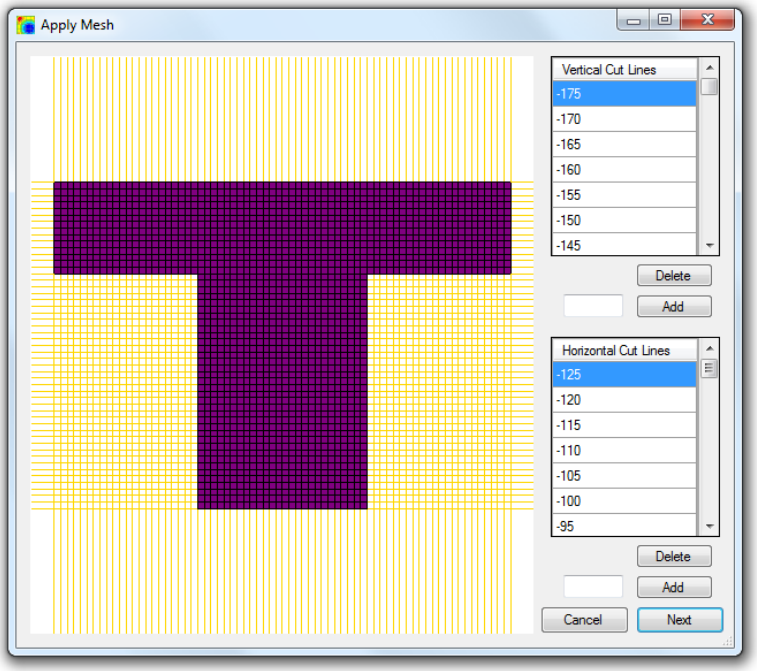


Figure 5.2 Joist (250mm) Finite Element Mesh

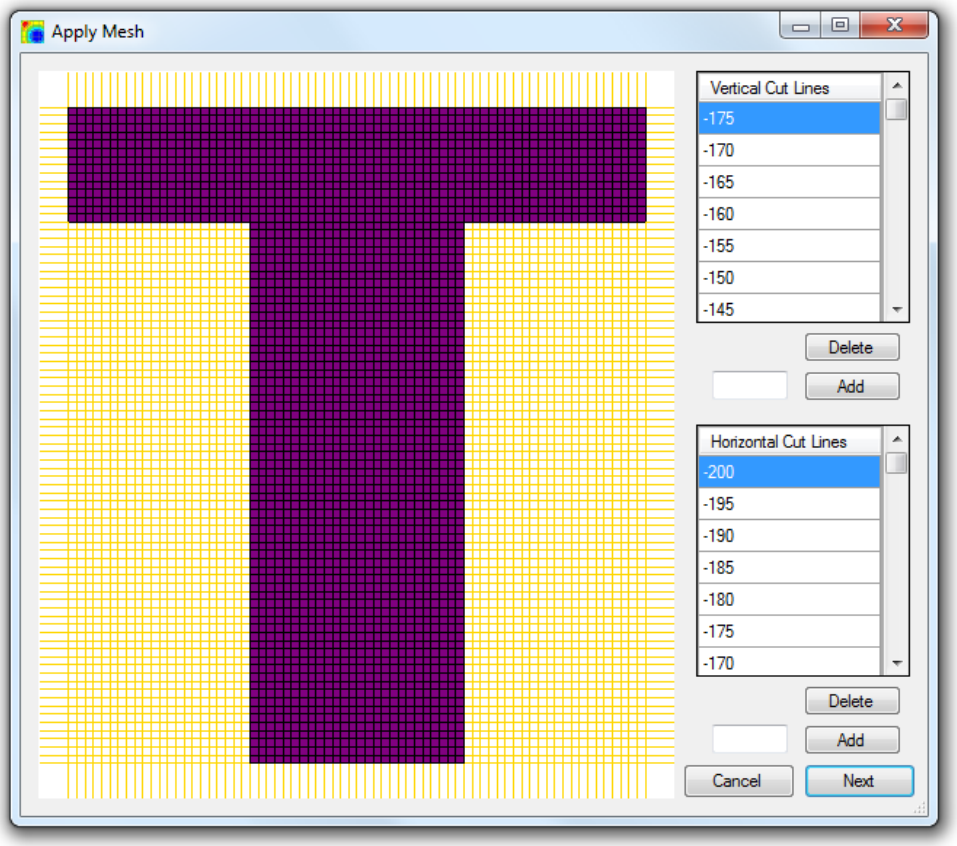


Figure 5.3 Joist (400mm) Finite Element Mesh

5.3 SAFIR RESULTS

The results of the SAFIR heat transfer analysis will be presented in this section. The images shown were produced using a post-processor developed for SAIFR 2007, Diamond (Franssen, 2007). Results were generated for all times throughout the OZone gas temperature-time curve. As an example of the results, Figures 5.4 to 5.6 show the temperature distribution in each of the elements at 40 minutes into the OZone fire. This was an arbitrary time, selected because it is approximately where the OZone fire reaches peak temperature.

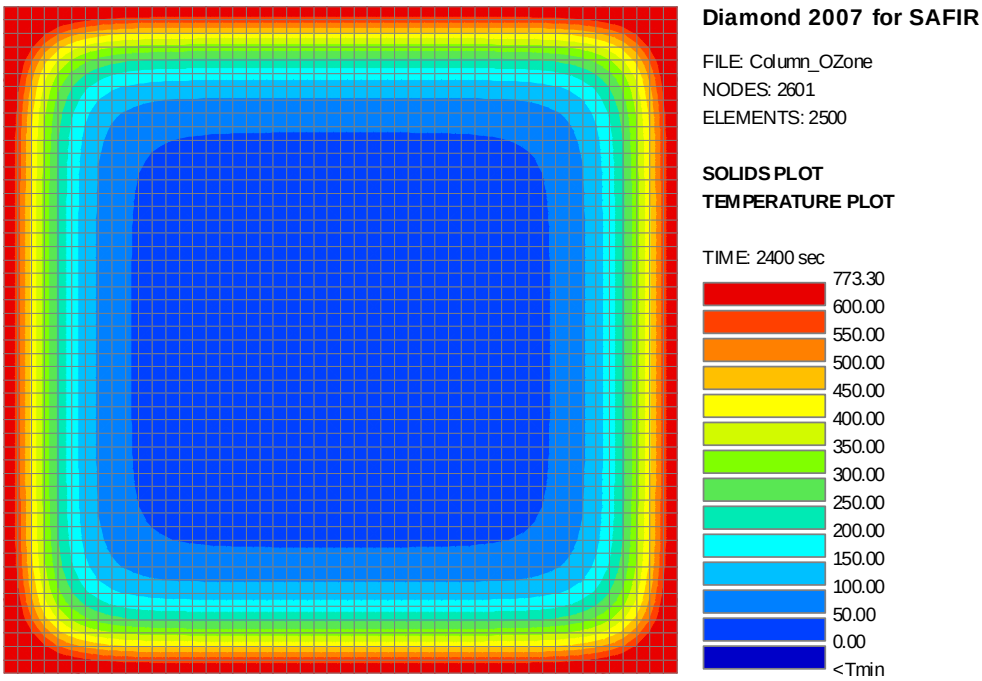


Figure 5.4 SAFIR Column Heat Transfer Results at Time = 40 minutes

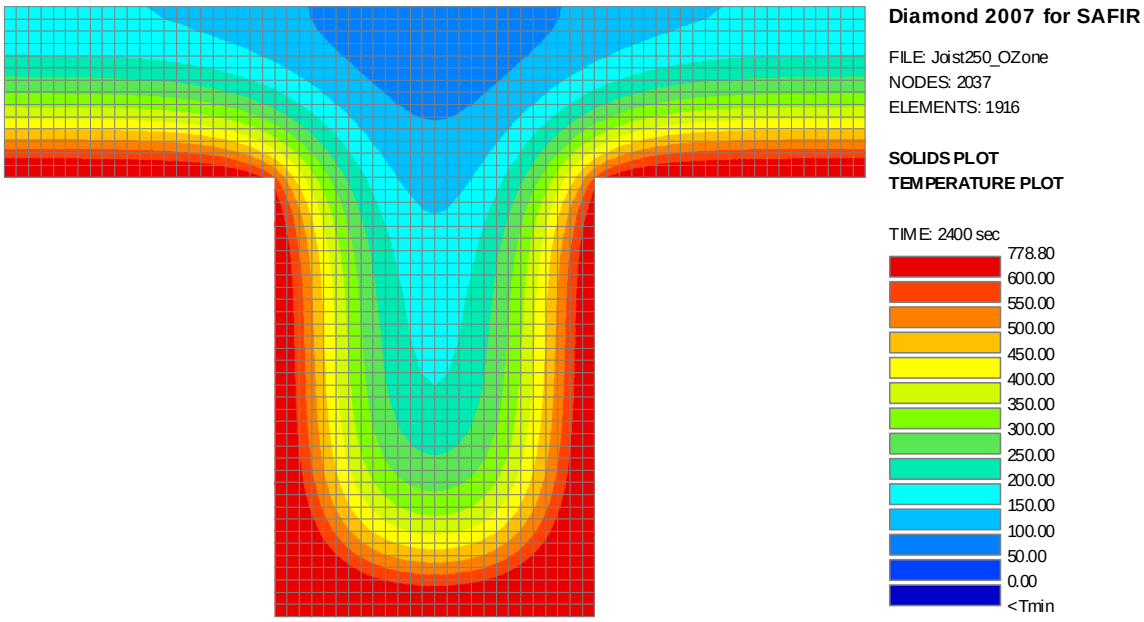


Figure 5.5 SAFIR Joist (250mm) Heat Transfer Results at Time = 40 minutes

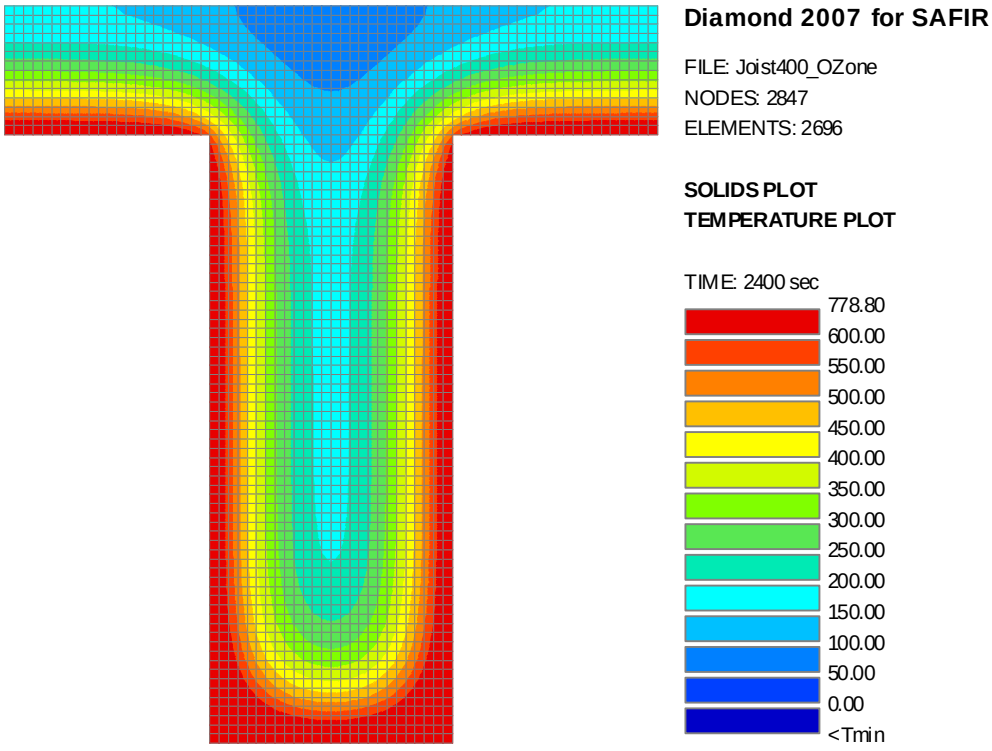


Figure 5.6 SAFIR Joist (400mm) Heat Transfer Results at Time = 40 minutes

These images are just a snapshot of the temperature distribution in the section at a single point in time. The full temperature history through the fire can be viewed at individual nodes. Figure 5.7 shows the temperature through the fire at 4 nodes on the 250mm joist. The first point is located on the surface directly exposed to the fire. As expected, the point nearly follows the air temperature, lagging slightly behind during heating and retaining some heat as the air cools. The other points of interest were chosen as the locations of the section's longitudinal reinforcement.

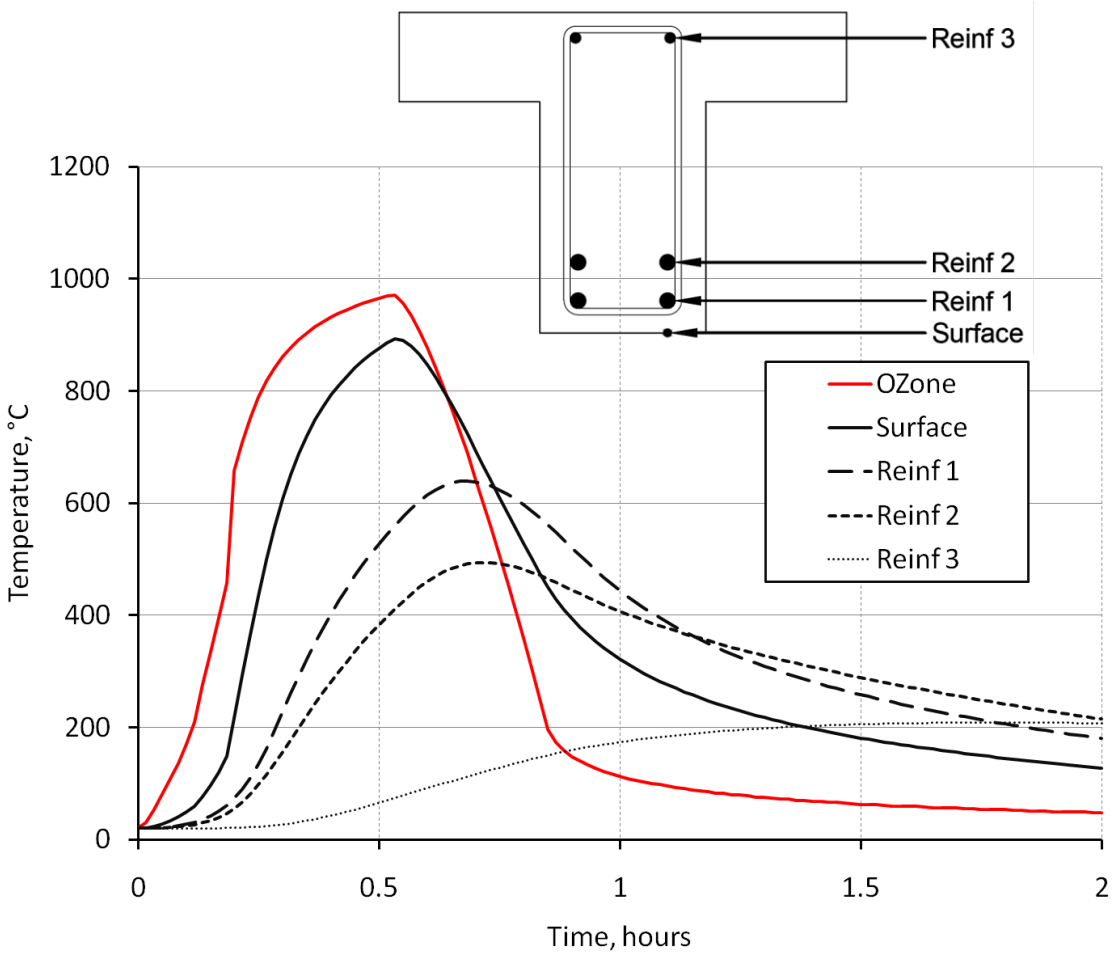


Figure 5.7 Nodal Heat Transfer Results (Joist G)

It is interesting to note that the interior of the section continues to heat, even after the fire has died out. This means that the section will actually reach its peak temperature at some time that does not necessarily correspond to the peak hot gas temperature. This phenomenon is also illustrated in Figure 5.8, where the average concrete and reinforcement temperatures are plotted along with the OZone temperatures of the hot gases surrounding the member. It can again be seen that the time the section reaches its maximum average temperature does not correspond to the time the fire reaches its peak temperature.

Recall from Chapter 3 that, at the time of collapse, the fire was almost completely extinguished in the areas that collapsed. At first glance, this would seem to indicate that

the failure occurred as the members were cooling and contracting. This may not have been the case if the interior of the members were in fact still heating, even though the fire was in, or near the completion of its cool down phase.

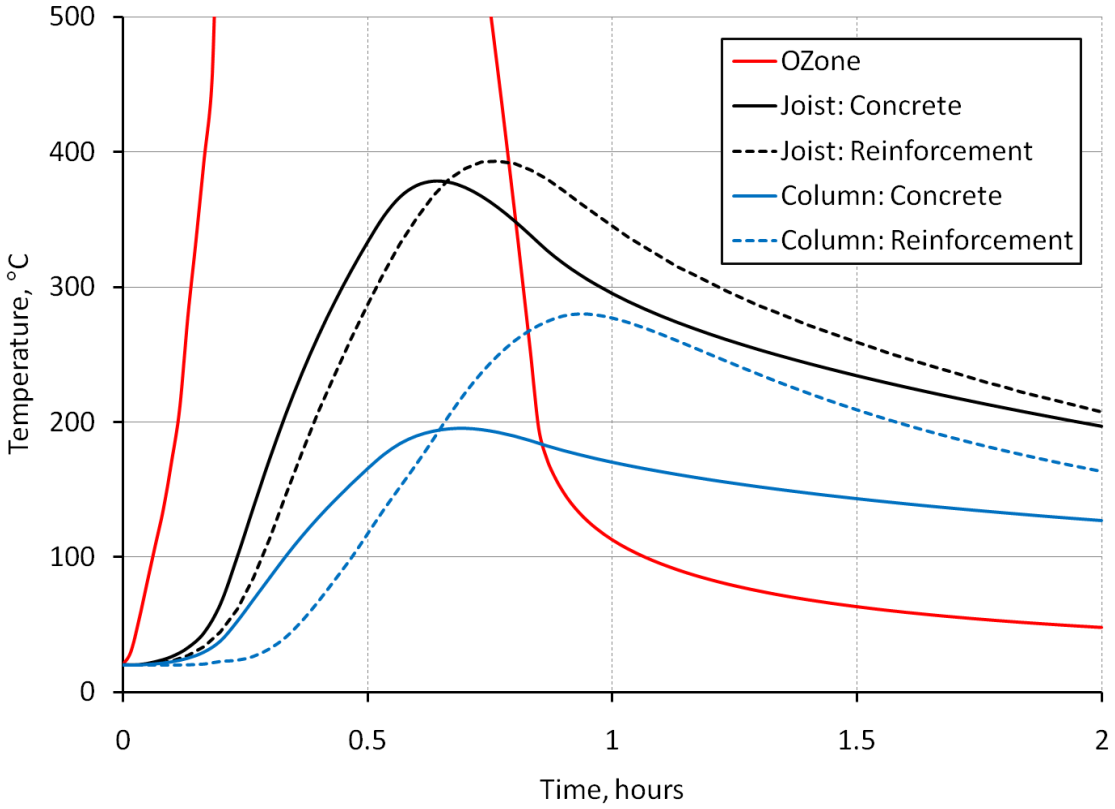


Figure 5.8 Average Section Temperatures

5.4 SUMMARY

In this chapter, finite element models of three selected member types from the FOA were created using UT Fire. A heat transfer analysis was then conducted for each using SAFIR, under the OZone fire model created in Chapter 4 of this report. Diamond was used to present the results. In the following chapters, the distribution of temperature through the sections will be used to estimate the reduced strength of the members at discrete times during the fire.

CHAPTER 6
Development of Tools for Reduced Capacity Analysis

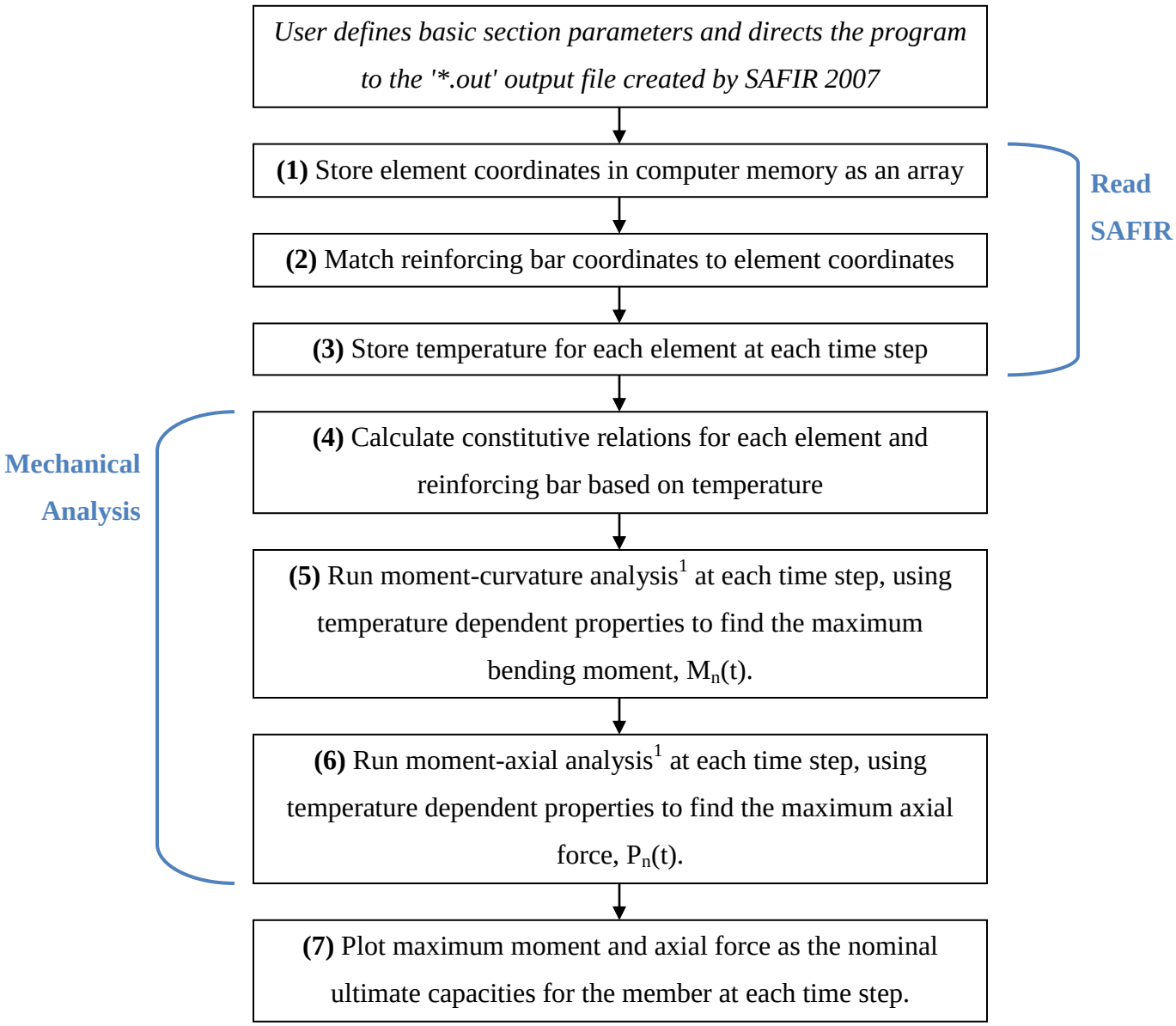
6.1 OVERVIEW

The previous chapter described heat transfer analysis of selected members of the FOA structure. The next step in the preliminary analysis of the response of the FOA structure to the fire of May 13, 2008 is to compute the loss of cross-sectional strength of these members resulting from exposure to the fire. To permit rapid calculation of the cross-sectional strength of heated reinforced concrete members, a computer program referred to as "UT Fire: Reinforced Concrete Analysis" (UTF:R/C) was developed. UTF:R/C reads in the output file from the SAFIR heat transfer analysis, and then computes cross-sectional strength at various times during the fire exposure. This chapter describes the development of UTF:R/C.

The constitutive relationships (stress-strain behavior) of concrete and reinforcement at elevated temperatures will be discussed and incorporated into the program. Also, an example problem will be presented that will show the use and capabilities of the program Finally, the results will be validated against existing software and simplified hand calculations. In the next chapter, UTF:R/C will be used to examine the loss of capacity for a number of the structural members in the FOA

6.2 LOGIC OF PROGRAM DEVELOPMENT

UT Fire: R/C Analysis began as a MS Excel spreadsheet macro that was written to help interpret the output of the SAFIR heat transfer analysis. This eventually developed into a stand-alone Windows application, written using MS Visual Basic 2008. The purpose of the program is to allow the user to estimate member capacities of reinforced concrete sections through the duration of a given fire. The overall flow of the program is summarized as the major steps shown in Figure 6.1.



[1]These steps will be further outlined and explained in Sections 6.3 and 6.4

Figure 6.1 UT Fire: R/C Analysis Flowchart

For SAFIR to complete its heat transfer analysis, the section is broken up into a mesh of two-dimensional finite elements. The temperature of each element in the mesh is recorded at each time step during the fire in a text file, with the extension "*.out". UTF:R/C uses this information to estimate capacities of a reinforced concrete member at

discrete time steps through the given fire. The program works by assigning each element in the mesh its own stress-strain curve based on its temperature at that time. Moment-curvature and moment-axial interaction plots are then generated to find the maximum moment and axial capacities based on the given internal temperature distribution.

6.3 MATERIAL MODELS

After the heat transfer data for the section computed by SAFIR is read and imported into UTF:R/C, the structural analyses begins. The first step in the thermal structural analysis (noted as step 4 in the overall flow chart) requires the program to calculate the temperature dependent constitutive relationships for each element and reinforcing bar. Elevated temperature stress-strain relationships specified by Eurocode 2 were used for this purpose. More specifically, the material models used in UTF:R/C come from EN1992-1-2. It should be noted that some of the notation used in this thesis varies somewhat from that used in EN 1992-1-2. In this thesis, the subscript T will denote a temperature dependent property.

6.3.1 Concrete in Compression

The Eurocode 2 recommendations for concrete in compression decrease the peak stress and increase the peak strain an element can withstand as the temperature is increased. The Eurocode values for the stress-strain behavior of concrete in compression at elevated temperatures is proved in Table 6.1. The resulting stress strain curves are plotted in Figure 6.2. From this plot, it can be seen that by 500°C, siliceous concrete has lost almost 60% of its original strength, but can withstand over 150% of its original peak strain. In UTF:R/C, interpolation is used for intermediate temperatures.

Table 6.1 EN 1992-1-2:2004 (E) - Table 3.1

	Siliceous Aggregates			Calcareous Aggregates		
T (°C)	f'_{cT}/f'_c	ε_{oT}	ε_{cuT}	f'_{cT}/f'_c	ε_{oT}	ε_{cuT}
20	1.00	0.0025	0.0200	1.00	0.0025	0.0200
100	1.00	0.0040	0.0225	1.00	0.0040	0.0025
200	0.95	0.0055	0.0250	0.97	0.0055	0.0250
300	0.85	0.0070	0.0275	0.91	0.0070	0.0275
400	0.75	0.0100	0.0300	0.85	0.0100	0.0300
500	0.60	0.0150	0.0325	0.74	0.0150	0.0325
600	0.45	0.0250	0.0350	0.60	0.0250	0.0350
700	0.30	0.0250	0.0375	0.43	0.0250	0.0375
800	0.15	0.0250	0.0400	0.27	0.0250	0.0400
900	0.08	0.0250	0.0425	0.15	0.0250	0.0425
1000	0.04	0.0250	0.0450	0.06	0.0250	0.0450
1100	0.01	0.0250	0.0475	0.02	0.0250	0.0475
1200	0.00	-	-	0.00	-	-

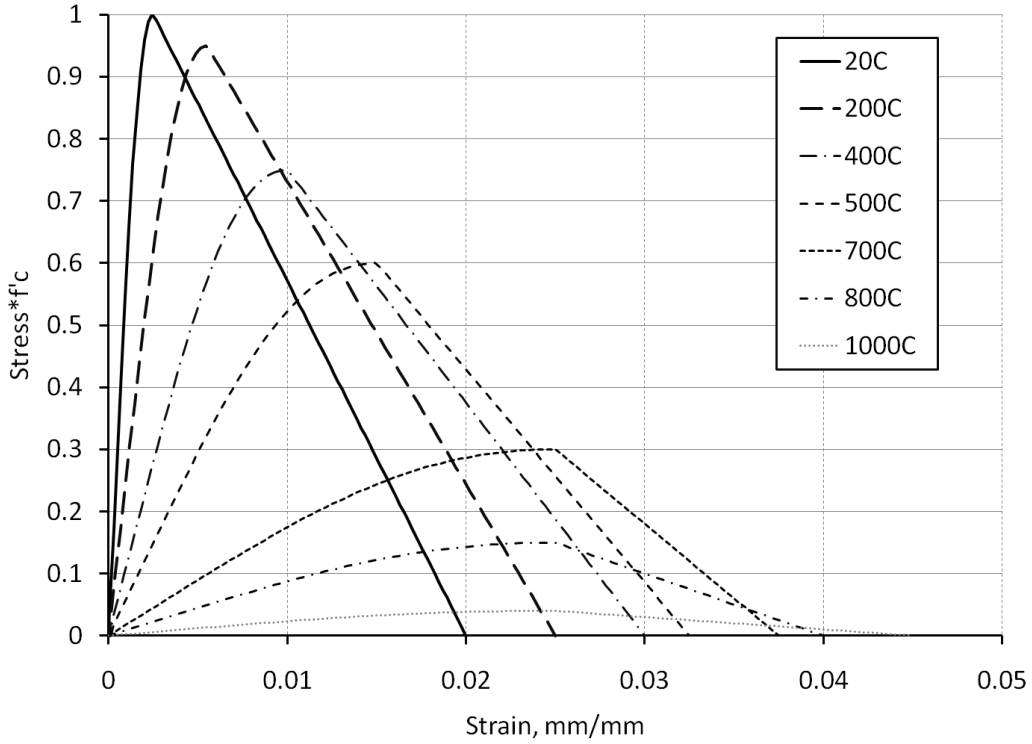


Figure 6.2 Stress-Stain Relationships for Siliceous Concrete in Compression

Table 6.2 Eurocode Equations for σ-ε Behavior of Concrete in Compression at Elevated Temperatures

Strain in Element, ε_c	Stress in Element, f_{cT}
$\mathbf{0} < \boldsymbol{\varepsilon_c} < \boldsymbol{\varepsilon_{oT}}$	$\dfrac{3\varepsilon_c f'_{cT}}{\varepsilon_{oT}\left[2+\left(\dfrac{\varepsilon_c}{\varepsilon_{oT}}\right)^3\right]}$
$\boldsymbol{\varepsilon_{oT}} < \boldsymbol{\varepsilon_c} < \boldsymbol{\varepsilon_{cuT}}$	$f'_{cT}\left[1-\left(\dfrac{\varepsilon_c-\varepsilon_{oT}}{\varepsilon_{cuT}-\varepsilon_{oT}}\right)\right]$
$\boldsymbol{\varepsilon_{cuT}} < \boldsymbol{\varepsilon_c}$	0

6.3.2 Concrete in Tension

Eurocode 2 also allows the use of concrete's tensile strength in analysis . This tensile strength decreases as the concrete is heated and drops to zero once the concrete is above 600°C. The decrease is strength, k_{tT}, is a linear function of the temperature.

For use in UTF:R/C, the maximum concrete tensile stress is taken as the ACI recommended value of: $f'_{tT} = 0.6228\sqrt{f'_{cT}}$, , where f'$_{cT}$ is in units of MPa (this is approximately equal to $f'_{tT} = 7.5\sqrt{f'_c}$, where f'c is in psi). Eurocode 2 does not give recommendations for the tension stiffening effects of concrete, so a bilinear model has been used with the ascending branch having a slope equal to the initial modulus of elasticity of the concrete in compression. The strain at maximum tensile stress is then: $\varepsilon'_{tT} = f'_{tT}/E_c$. The maximum tensile strain is taken as: $\varepsilon_{tuT} = 6\,f'_{tT}/E_c$. For use in UTF:R/C, the initial modulus of elasticity of the concrete in compression is taken as: $E_c = 4733\sqrt{f'_c}$, where f'$_c$ is in MPa (equal to the ACI recommended value of $E_c = 57000\sqrt{f'_c}$, where f'$_c$ is in psi). The full set of equations used for concrete in tension in the program is shown in Table 6.3. The stress strain curves resulting from this table and set of equations are plotted in Figure 6.3.

Table 6.3 Concrete Stress-Stain Relationship in Tension at Elevated Temperatures

Strain in Element, ε_c	Stress in Element, f_{cT}
$\boldsymbol{\varepsilon'_{tT} < \varepsilon_c < 0}$	$\varepsilon_c E_c$
$\boldsymbol{\varepsilon_{tuT} < \varepsilon_c < \varepsilon'_{tT}}$	$f'_{cT}\left[1 - \left(\frac{\varepsilon_c - \varepsilon'_{tT}}{\varepsilon_{tuT} - \varepsilon'_{tT}}\right)\right]$
$\boldsymbol{\varepsilon_c < \varepsilon_{tuT}}$	0

Where: $f'_{tT} = k_{tT} f'_t$ $\quad 20℃ \le T \le 100℃ \quad \therefore k_{tT} = 1.0$

$\quad 100℃ < T \le 600℃ \quad \therefore k_{tT} = 1 - \left(\frac{T-100}{500}\right)$

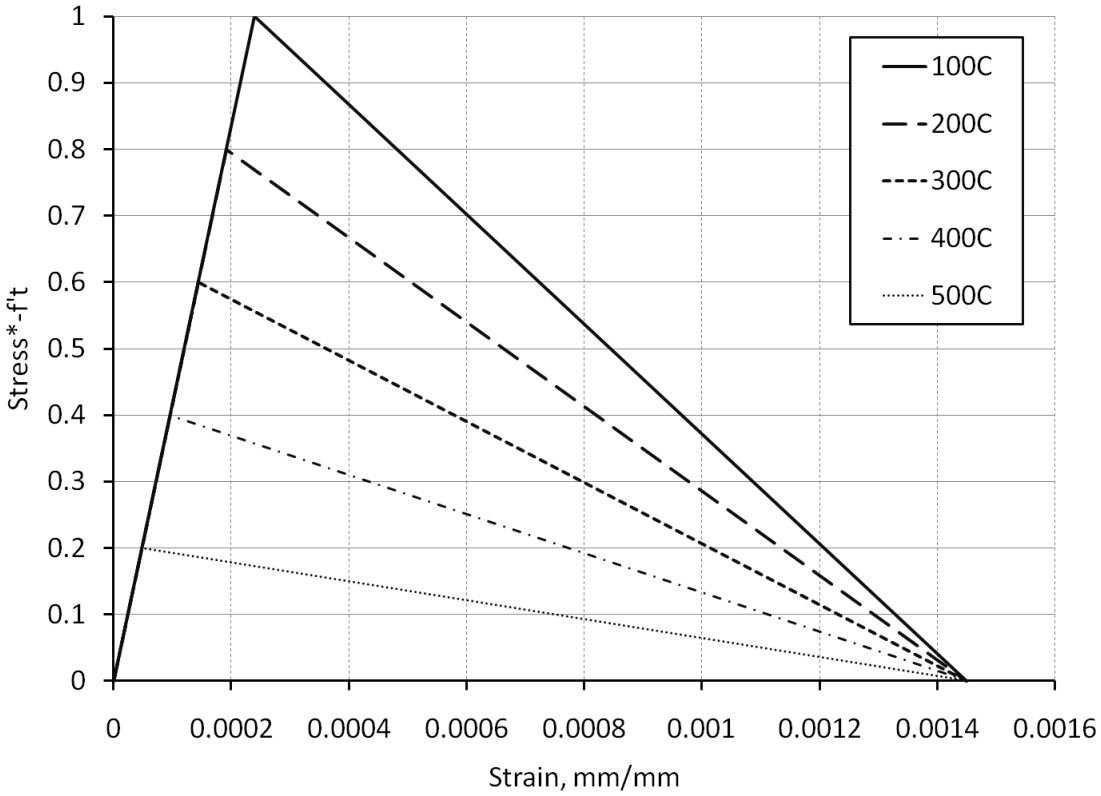


Figure 6.3 Concrete Stress-Stain Relationship in Tension

6.3.3 Reinforcing Steel

The stress-strain relationship for reinforcing steel at elevated temperature in Eurocode 2 is defined by the following parameters: proportional stress limit (f_{pT}), maximum stress level (f_{yT}), and the slope of the linear elastic range (E_{sT}). Like concrete, these defining values decrease with an increase in temperature. The relationships between temperature, strain and stress are defined by the equations provided in Table 6.4 and are plotted in Figure 6.4. The curves produced are essentially linear to the proportional limit, then decreases in stiffness to zero as they approach the maximum stress level. This stress level is constant until the steel reaches 15% strain. Unlike concrete at elevated temperatures, the maximum strain available in the steel does not

increase with an increase in temperature. The steel unloads from 15% to 20% strain, linearly to zero stress. Like concrete, steel is not expected to provide any load carrying capability once it reaches 1200°C.

Table 6.4 EN 1992-1-2:2004 (E) - Table 3.2a

T (°C)	f_{yT}/f_y	f_{pT}/f_p	E_{sT}/E_s
20	1.00	1.00	1.00
100	1.00	1.00	1.00
200	1.00	0.81	0.90
300	1.00	0.61	0.80
400	1.00	0.42	0.70
500	0.78	0.36	0.60
600	0.47	0.18	0.31
700	0.23	0.07	0.13
800	0.11	0.05	0.09
900	0.06	0.04	0.07
1000	0.04	0.02	0.04
1100	0.02	0.01	0.02
1200	0.00	0.00	0.00

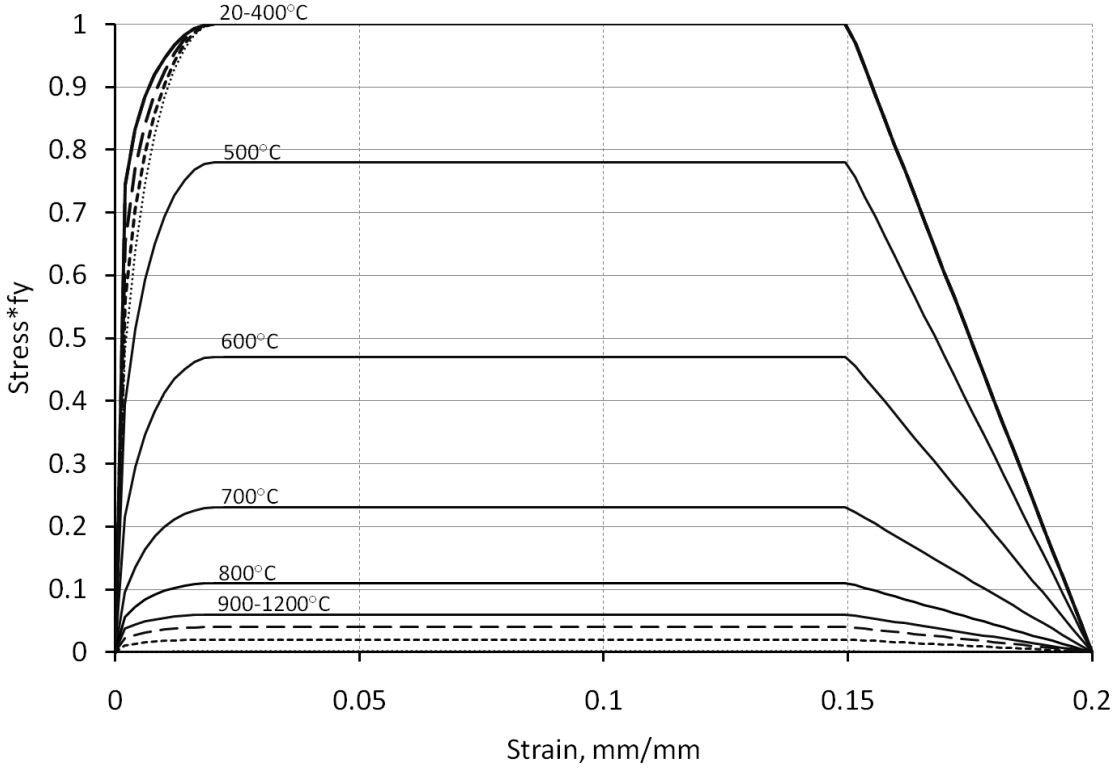


Figure 6.4 Steel Stress-Strain Relationship

Table 6.5 Eurocode Equations for σ-ε Behavior of Steel at Elevated Temperatures

Strain in Steel, ε_s	Stress in Steel, f_{sT}
$\boldsymbol{0 \le \varepsilon_s \le \varepsilon_{pT}}$	$\varepsilon_s E_{sT}$
$\boldsymbol{\varepsilon_{pT} \le \varepsilon_s \le \varepsilon_{yT}}$	$f_{pt} - c + \left(\frac{b}{a}\right)\sqrt{a^2 - \left(\varepsilon_{yT} - \varepsilon_s\right)^2}$
$\boldsymbol{\varepsilon_{yT} \le \varepsilon_s \le \varepsilon_{tT}}$	f_{yt}
$\boldsymbol{\varepsilon_{tT} \le \varepsilon_s \le \varepsilon_{uT}}$	$f_{yt}\left[1 - \left(\frac{\varepsilon_s - \varepsilon_{tT}}{\varepsilon_{uT} - \varepsilon_{tT}}\right)\right]$
$\boldsymbol{\varepsilon_{uT} \le \varepsilon_s}$	0

Where: $\varepsilon_{pT} = f_{pT}/E_{sT}$

$\varepsilon_{yT} = 0.02$ $\qquad a = \sqrt{(\varepsilon_{yT} - \varepsilon_{pT})(\varepsilon_{yT} - \varepsilon_{pT} + c/E_{sT})}$

$\varepsilon_{tT} = 0.15$ $\qquad b = \sqrt{c(\varepsilon_{yT} - \varepsilon_{pT})E_{sT} + c^2}$

$\varepsilon_{uT} = 0.20$ $\qquad c = \frac{(f_{yT} - f_{pT})^2}{(\varepsilon_{yT} - \varepsilon_{pT})E_{sT} - 2(f_{yT} - f_{pT})}$

6.4 MOMENT-CURVATURE ANALYSIS

The ultimate moment capacity for a cross-section at each time step is calculated as the maximum moment found from a moment-curvature analysis for the given internal temperature distribution. UTF:R/C achieves this through a series of nested loops. In the first loop, the strain at the top of the section is varied from -0.2 to 0.2 at an interval of 0.01%. This range captures the full moment-curvature response, from its minimum to maximum moment, for any given section using the Eurocode 2 material models, where the maximum possible material strain is fixed at 0.2 (steel).

Inside of the first loop, for each value of top strain, a guess is made on the location of the neutral axis. With each value of top strain and neutral axis depth, the strain in each individual element is calculated based on the assumption of plane sections. This strain and the temperature in the element at the given time step are used to determine the stress in that element. The force is then calculated as the stress multiplied by the element area. This process is repeated for each element in the section and the resultant axial force on the section is calculated as the sum of the forces in each individual element:

$$\boldsymbol{P_n} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{\#Elements}} \boldsymbol{F_{ci}} + \sum_{\boldsymbol{i=1}}^{\boldsymbol{\#Bars}} \boldsymbol{F_{si}} \qquad (6.1)$$

If the resultant axial force does not equal zero, then the neutral axis location is changed and the process is repeated. The program will continue to iterate the neutral axis depth until the section is in equilibrium with $P_n(t) = 0$. Convergence is sped up by looking at the problem as a function of two variables: the neutral axis depth, c, and the resulting axial force, P. The method of false position is used to find the roots of the function and convergence can typically be obtained in 3-5 iterations. This process is outlined in Figure 6.5. Once $P_n(t) = 0$, the corresponding curvature and moment are calculated for that plot point as:

$$\phi(t, \varepsilon_{top}) = \frac{\varepsilon_{top}}{c} \tag{6.2}$$

$$M_n(t, \varepsilon_{top}) = \sum_{i=1}^{\#Elements} F_{ci}(\bar{y} - y_{ci}) + \sum_{i=1}^{\#Bars} F_{si}(\bar{y} - y_{si}) \tag{6.3}$$

In this equation, y_{bar} is the neutral axis location, measured from the top of the section and y_i is the distance to the centroid of the concrete element or reinforcing bar in question.

After the M-ϕ point has been determined for this value of top strain, the inner loop is complete and the next top strain step is used to find the next M-ϕ point on the curve. This entire process is outlined in the flowchart shown in Figure 6.5.

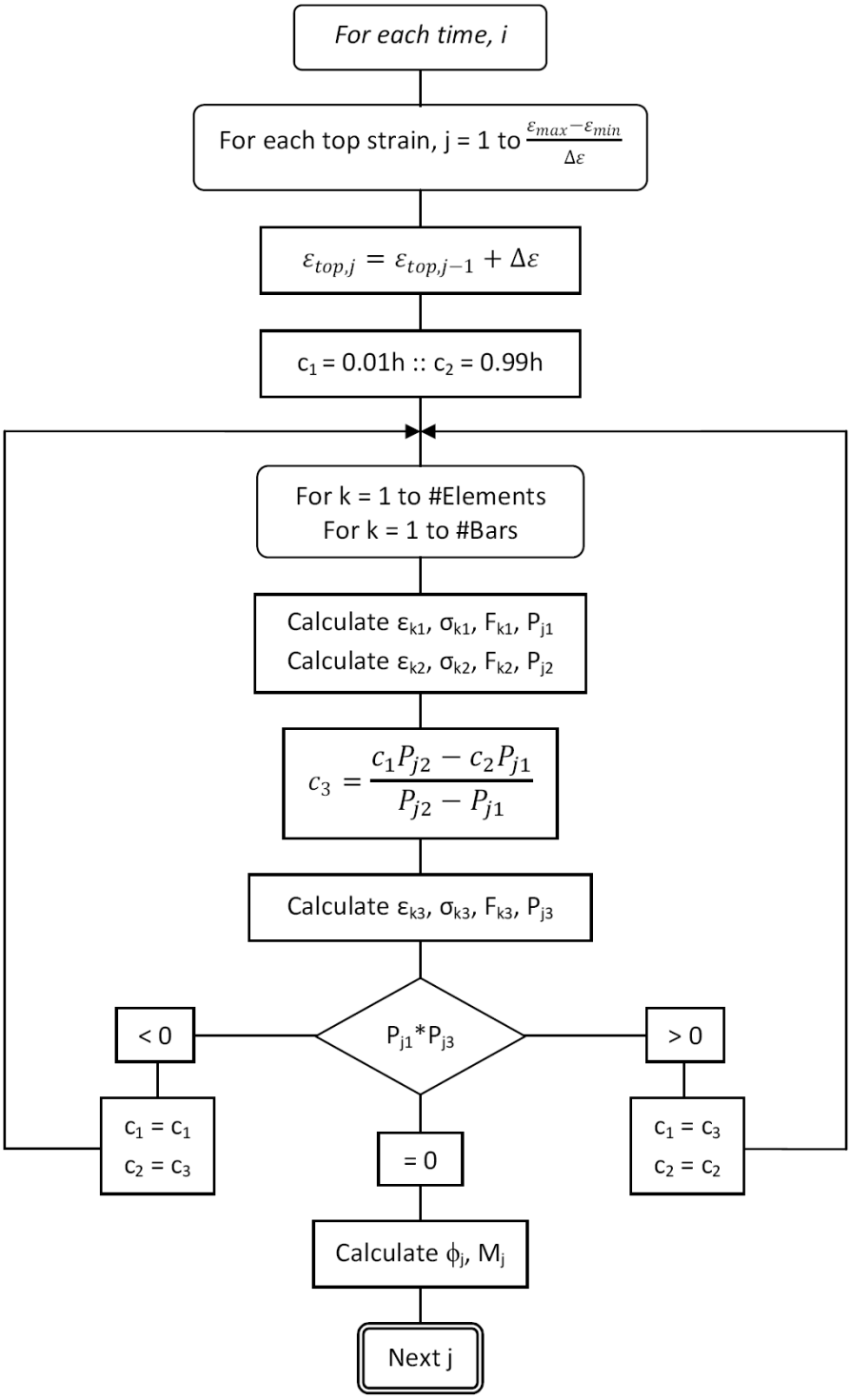


Figure 6.5 Moment-Curvature Analysis Flowchart

6.5 MOMENT-AXIAL FORCE INTERACTION

The moment-axial (M-P) force interaction diagrams at each time step are calculated in a similar manner as the moment-curvature diagrams. The major difference between the two sets of calculations is that there is no need to iterate to find a value of zero axial force in the moment-axial analysis. Any linear strain gradient will give values of axial force and moment that represent a valid point on the interaction surface.

For a member at normal temperatures, an interaction diagram can be constructed relatively easily. At normal temperatures, the ultimate concrete strain is constant through the section. The diagram is then constructed by setting the top strain to the ultimate strain of the concrete and varying the neutral axis location from infinity to 0. The resulting moment and axial force are calculated for each neutral axis depth and the pair is plotted. This simple procedure is outlined in Figure 6.6.

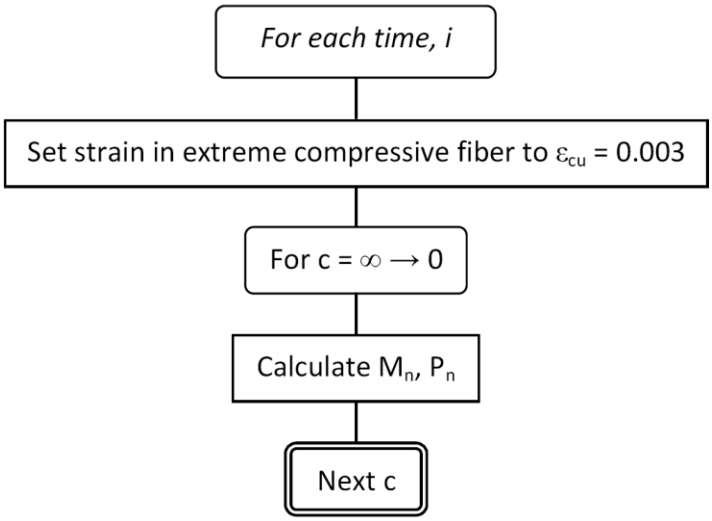


Figure 6.6 'Cold' Moment-Axial Interaction Analysis Flowchart

The problem that arises when attempting to construct an interaction diagram at elevated temperature is that, as the section is heated, the ultimate concrete strain changes for each element at a different rate and each element can crush at a different ultimate strain. This means that the maximum uniform strain does not necessarily represent the top strain that corresponds to the maximum M-P point on the interaction diagram. In

other words, an element in the center of the section may crush before a heated element (with a greater ultimate strain) at the extreme compressive fiber. To solve this problem, an additional loop must be nested in the calculation process. This loop finds the ultimate strain specific to the section with that temperature distribution. The full process is described in the following paragraphs and illustrated in the flow chart of Figure 6.7.

First, the ultimate pure compressive capacity is found by applying a range of uniform strains on the section. Each resulting axial force is calculated. The uniform strain is increased until the axial force begins to drop (as concrete begins to crush). This peak is then the ultimate axial capacity of the section at that time step and the first point on the moment-axial interaction diagram.

The rest of the interaction points are found by a second subroutine. Each point on the diagram is defined by a specific neutral axis depth, ranging from infinity (pure compression) to zero (pure tension). For each neutral axis location, the curvature is increased from 0 to 0.05. This maximum curvature, 0.05, was found to be adequate for capturing all of the interaction points for most sections. By plotting the resultant M-P pairs at each curvature, curves such as those shown in red in Figure 6.9 are produced. In this figure, each solid red line represents a single neutral axis depth and each M-P point on that line represents the equilibrium state for a given curvature. To define a point on the M-P interaction curve, the moment-axial pair at the furthest normalized radial distance from the origin is calculated as:

$$\bar{r} = \sqrt{\left(\frac{P}{P_{max}}\right)^2 + \left(\frac{M}{M_{max}}\right)^2} \qquad (6.4)$$

The dashed red lines in Figure 6.8 indicate this distance for two arbitrary neutral axis depths. By combining the plots of these maximum radial points at each neutral axis depth, an envelope can be constructed that represents the maximum moment-axial interaction surface for the section (shown as the dashed black line on the plot).

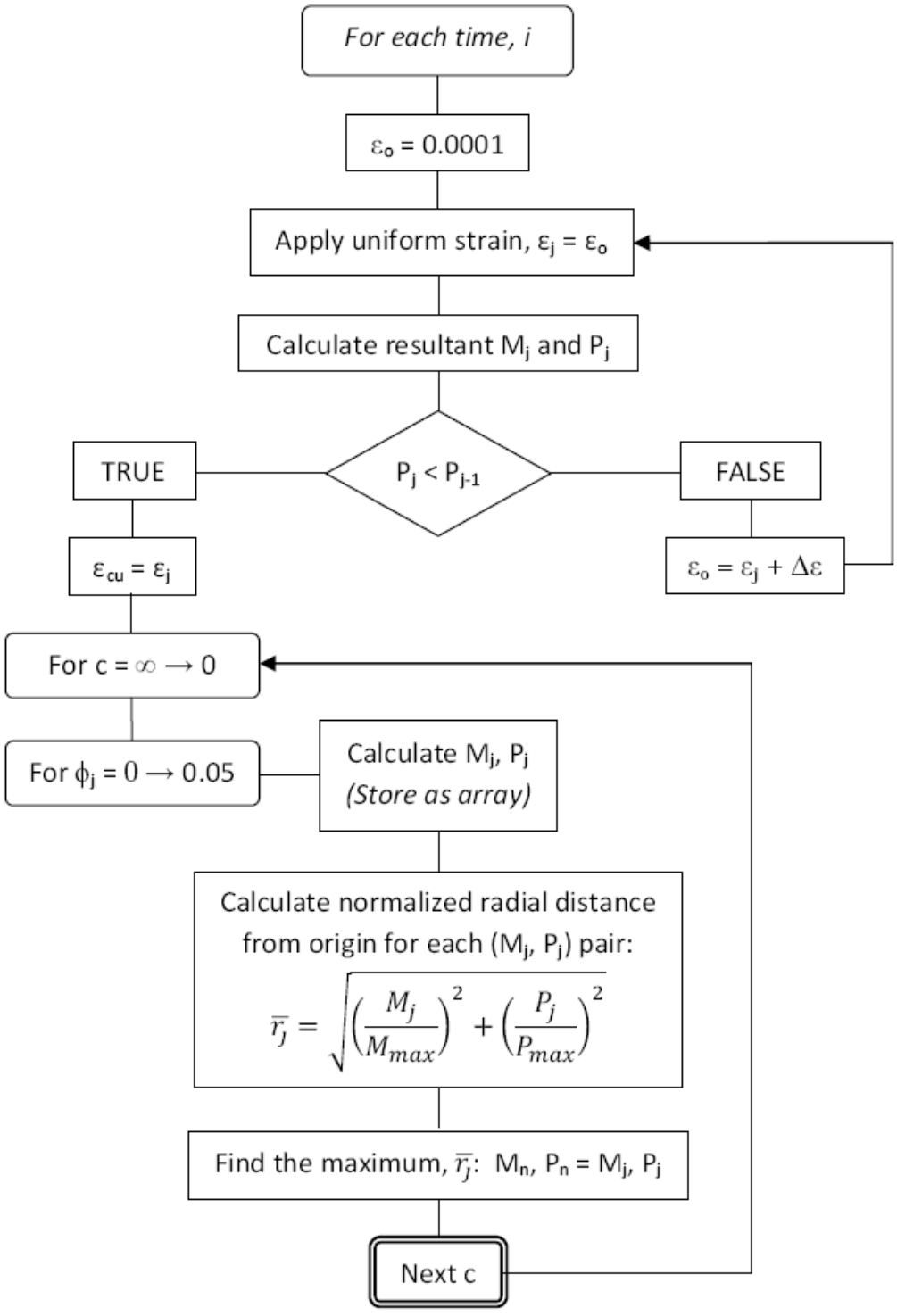


Figure 6.7 Moment-Axial Interaction at Elevated Temperatures Flowchart

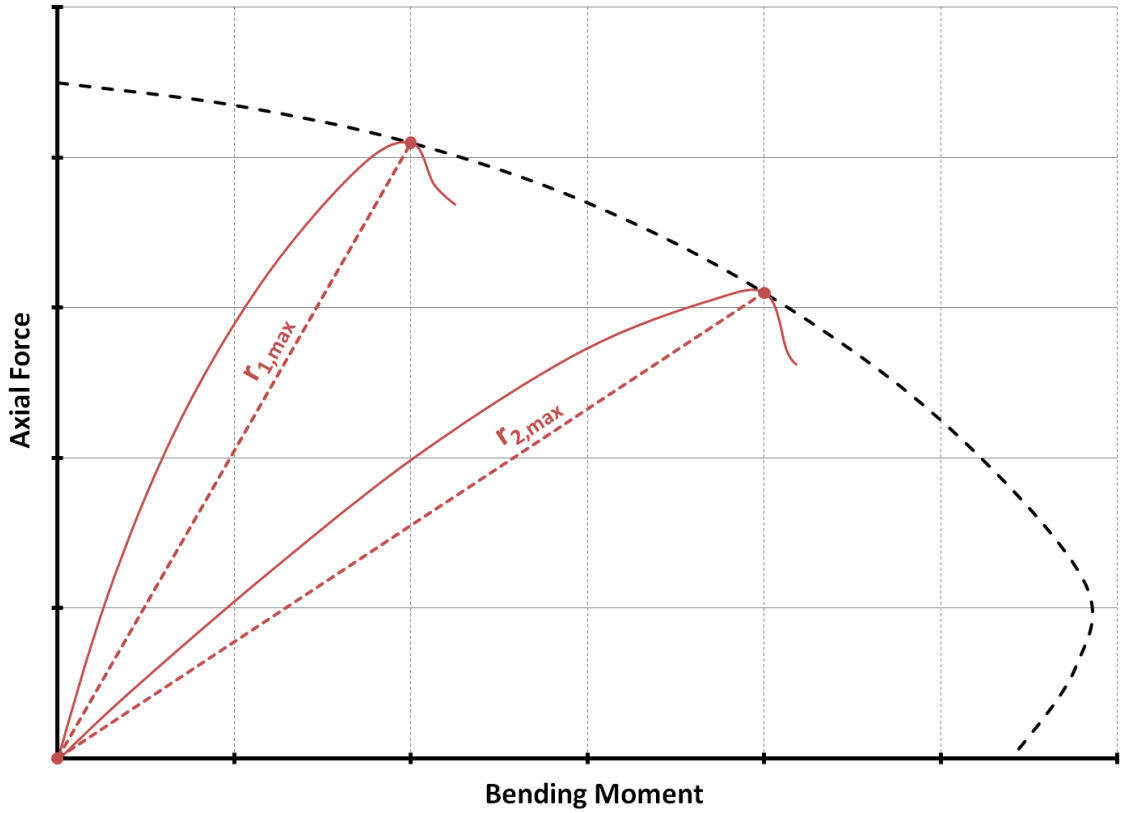


Figure 6.8 Moment-Axial Interaction Construction

6.6 UT FIRE: REINFORCED CONCRETE ANALYSIS - EXAMPLE PROBLEM

To help explain how UTF:R/C can actually be utilized and to demonstrate the capabilities of this program, an example analysis of a generic reinforced concrete member will be presented in this section. Screen shots of the graphical user interface will be shown to illustrate the relatively simple steps required to set up the analysis input file. It should be noted that the output created by the program is an MS Excel spreadsheet, located in the folder specified by the user at start-up.

The example section is a 250mm wide by 500mm tall beam-column, which is doubly reinforced with 4ϕ20 in the top and bottom. The concrete has maximum stress of 30 MPa and is comprised of siliceous aggregates. The proportional stress limit of the steel is 275 MPa and the maximum stress is 400 MPa. The section is illustrated in Figure

6.9. This could be a column or beam and is exposed to the ASTM E119 gas temperature-time curve on all four sides. The heat transfer using SAFIR has been completed and the resulting temperature distribution is illustrated in Figure 6.10 (after 2 hours of exposure to the fire).

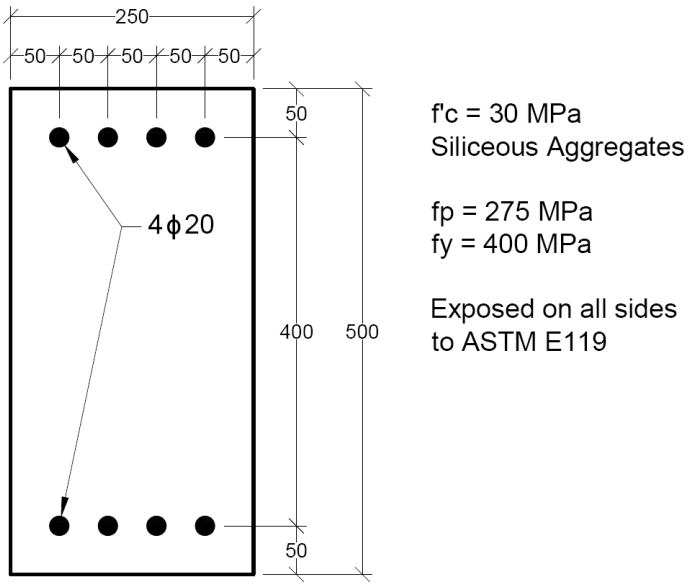


Figure 6.9 Example Section

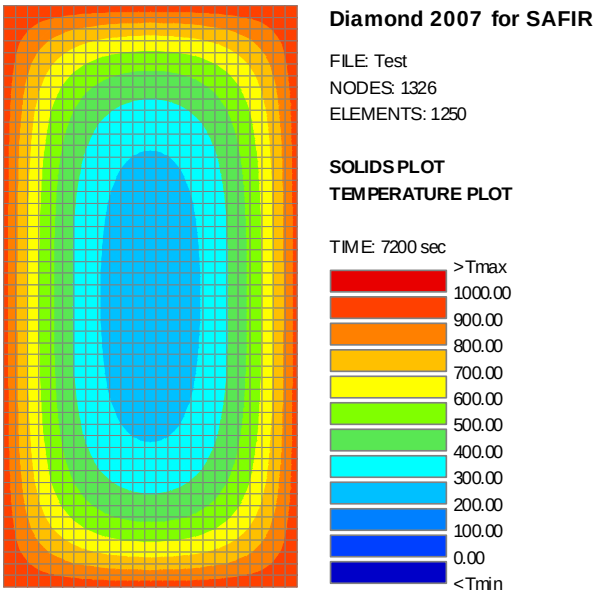


Figure 6.10 Example Section Temperature Contours at 120 Minutes

6.6.1 Getting Started With UT Fire: R/C Analysis

Prior to opening UTF:R/C, an input file for the section using Wizard (Franssen, 2007), UT Fire (Jennings, 2009), or a text editor is created. The heat transfer is completed by running this input file in SAFIR2007 (Franssen, 2007). The '*.out' created by SAFIR contains the data that describes the temperature of each element at each time step and is now ready to be used for post processing in UT Fire: R/C Analysis.

The start-up dialog form that will be displayed when UTF:R/C is opened is shown in Figure 6.11. The user should begin by reviewing the instructions located under the 'Getting Started' tab, but a few of the most important things to keep in mind before beginning the analysis are: the mesh should have less than 5000 nodes, the fire plot file ('*.fct' file used by SAFIR) should be in the same folder as the '*.out' file, and UTF:R/C will automatically close MS Excel if it is running when the program is started

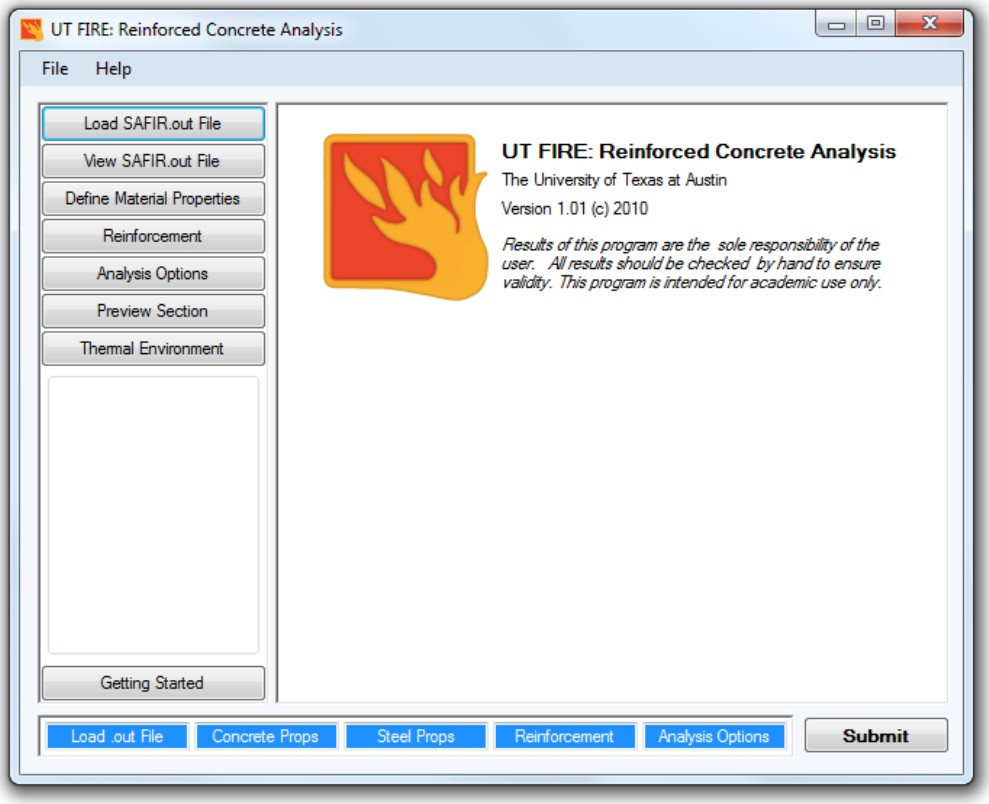


Figure 6.11 UT Fire Concrete Start-up Dialog

6.6.2 Load SAFIR .out File

Now that the program is open, the first step is to load the '*.out' file created by SAFIR when the heat transfer is completed. This is done by directing UTF:R/C to the file under the 'Load SAFIR.out File' tab. Once the file is successfully loaded, a short summary of the section will be displayed on the form. The '*.out' file can be viewed in its entirety under the 'View SAFIR.out File' tab. The form at this point in the process is illustrated in the screen shot of Figure 6.12.

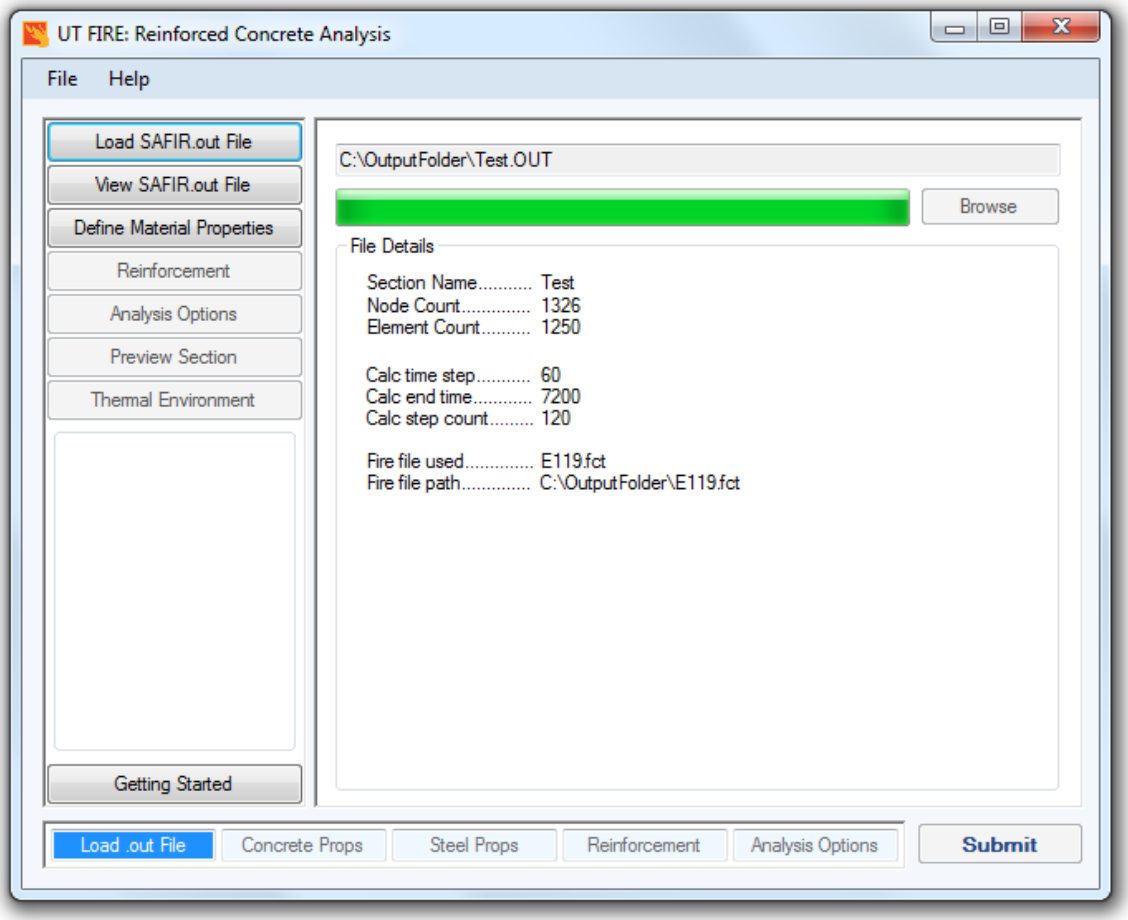


Figure 6.12 Example Section Load SAFIR .out File Dialog

6.6.3 Define Material Properties

As described earlier in this chapter, the material models at elevated temperatures used by UTF:R/C come from EN 1992-1-2:2004. These models have set values for key strain points so the user is only required to input limiting values of stress for each material. These values should be input to the program under the 'Define Material Properties' tab. Once the limiting values of stress are saved to the input file, the user may view the resulting stress-strain plots for each material at any temperature by toggling the boxes at the bottom right of the form. The form at this point in the example is shown in Figure 6.13.

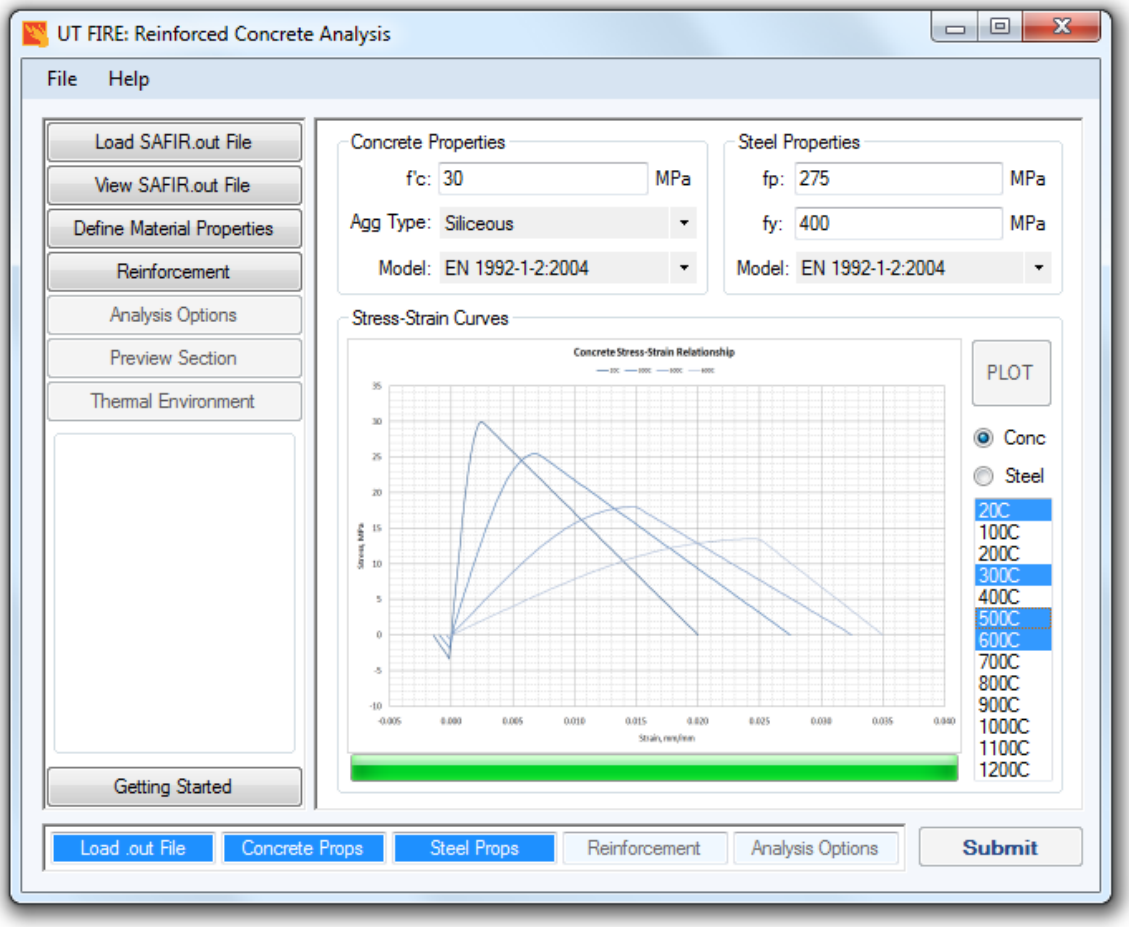


Figure 6.13 Example Section Material Property Input Dialog

6.6.4 Define Reinforcement

The next step required to set up the analysis is to define the size and location of each reinforcing bar in the section. This is done under the 'Reinforcement' tab. The dialog at this point is shown in Figure 6.14.

X and Y coordinates are measured from the origin at the absolute minimum bottom left of the section. Figure 6.15 shows two examples of how UTF:R/C defines this origin. Also, it should be noted that the reinforcement coordinates must match a nodal coordinate of the SAFIR model. This is because UTF:R/C uses this corresponding node to find the temperature of the reinforcement through the fire.

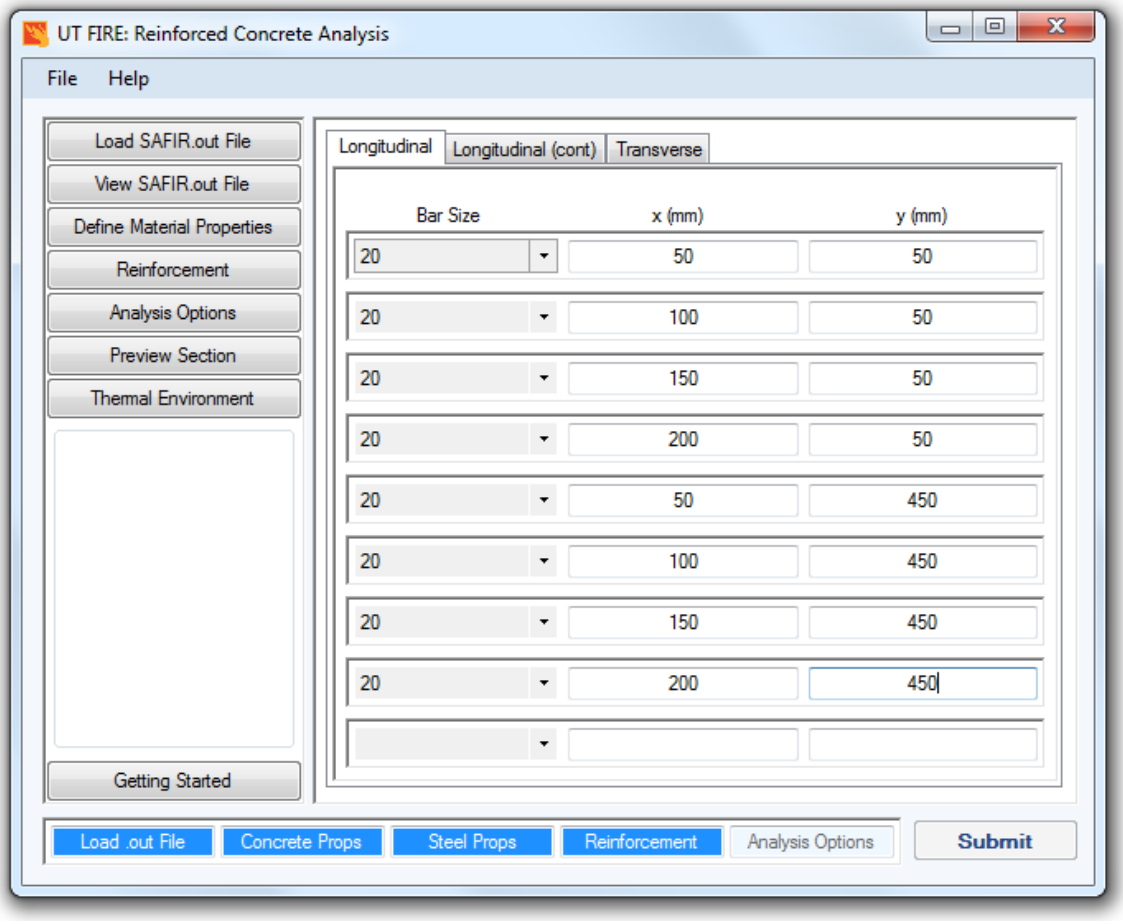


Figure 6.14 Example Section Reinforcement Input Dialog

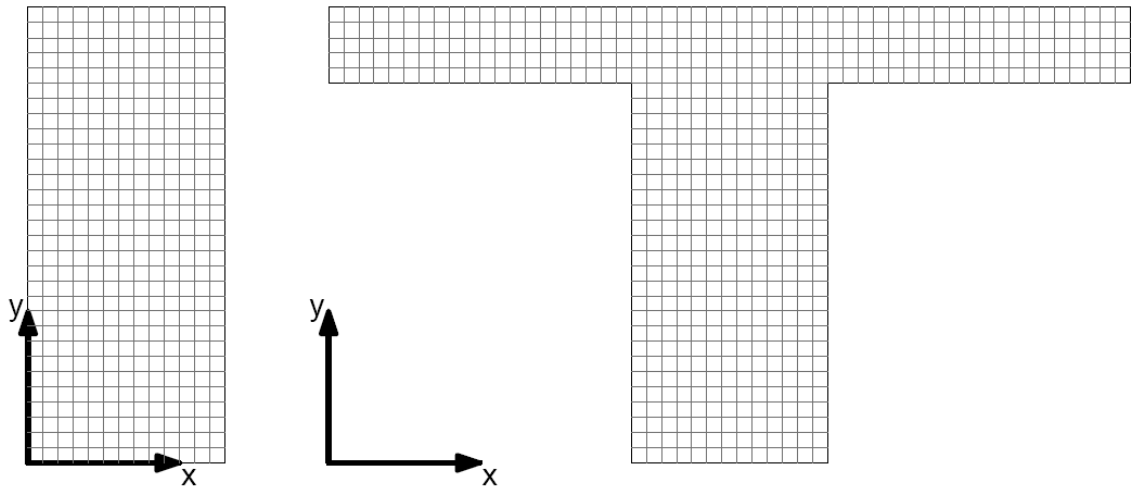


Figure 6.15 Coordinate Origin Illustrations

6.6.5 Analysis Options

The final step in setting up the input is to define a few parameters that will control the analysis. Currently the only analysis option that can be manually controlled by the user is the desired axial force to iterate for during the moment-curvature analysis. For flexural members, this will typically be left as default value of zero. For columns, the user may wish to input a non-zero value to examine the section's moment-curvature behavior under axial load. This value is set equal to zero for the purposes of the example problem.

The set up for the analysis is now complete. At this point, the input file should be saved and the 'SUBMIT' button at the bottom right may be clicked. Calculation times will vary based on number of elements that comprise the section, number of time steps to be evaluated and the speed of the computer the program is being run on, but the user should be prepared to wait up to 3 hours for the analysis to finish. The completed form is illustrated in Figure 6.16.

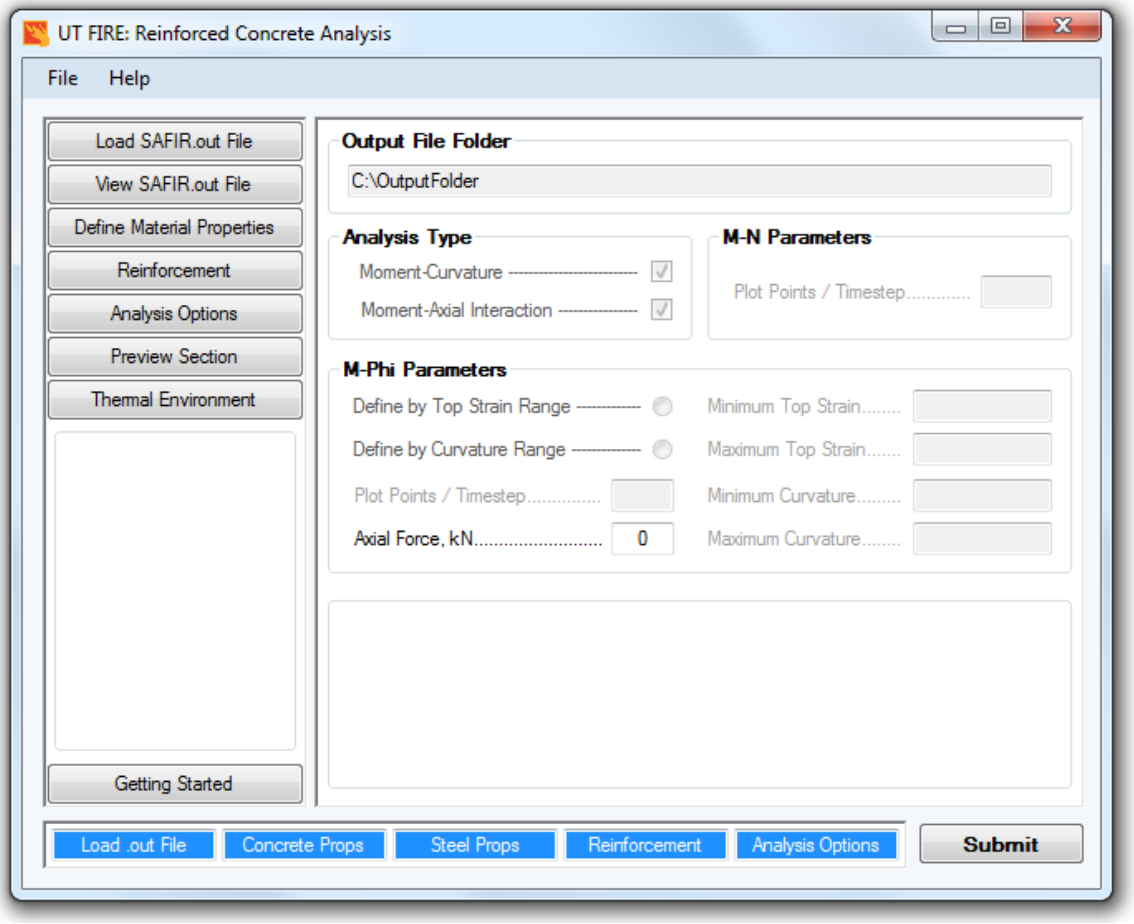


Figure 6.16 Example Section Analysis Options Dialog

6.6.6 Output

Once the calculation loops have completed, a new form will appear that contains the results of the analysis. Each tab contains plots created by the program . Figure 6.18 shows the results of the moment-curvature analysis. The moment-curvature relationship at any time can be seen by selecting the desired time step from the list to the left of the plot. In Figure 6.17, every time step is selected. Figure 6.18 shows the moment-axial interaction results, with every time step selected. All of this information is also automatically saved in a MS Excel file with the filename and directory as specified by the user when the program is closed.

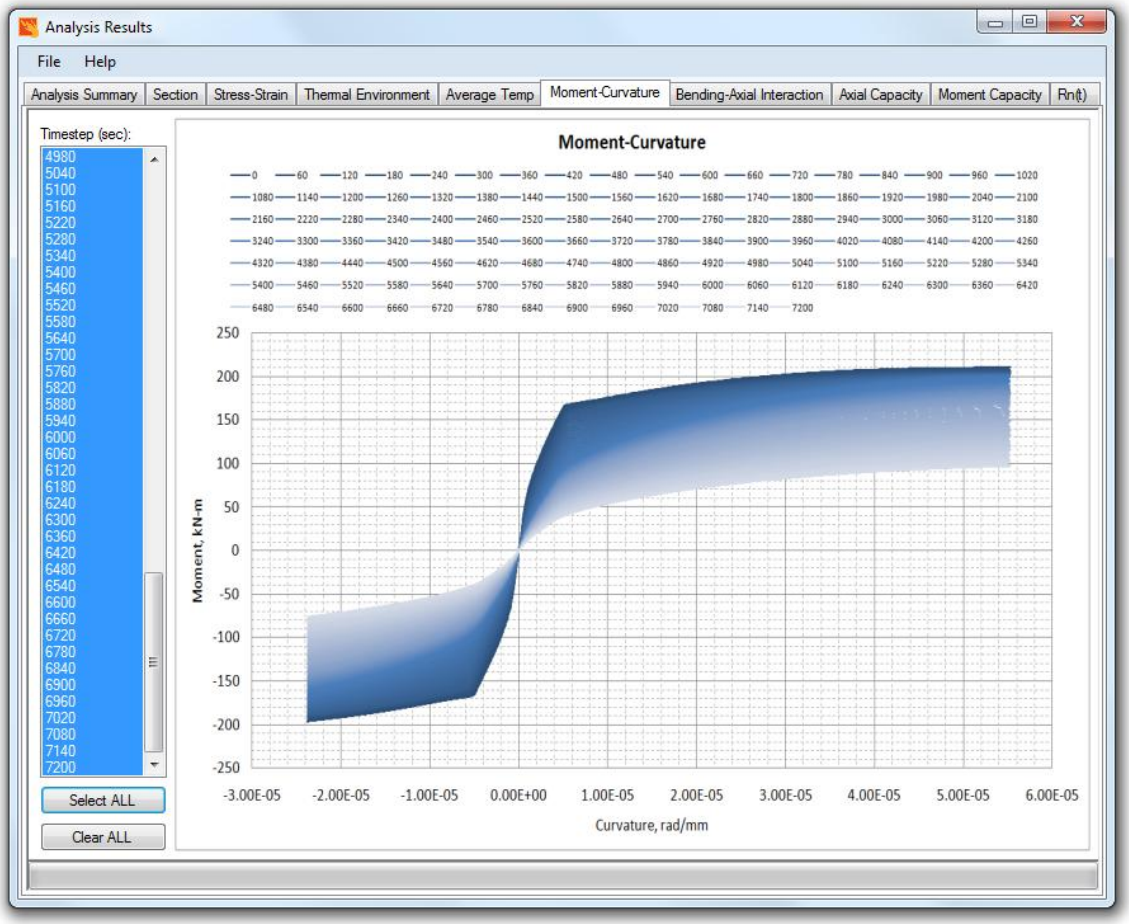


Figure 6.17 Example Section Moment-Curvature Output

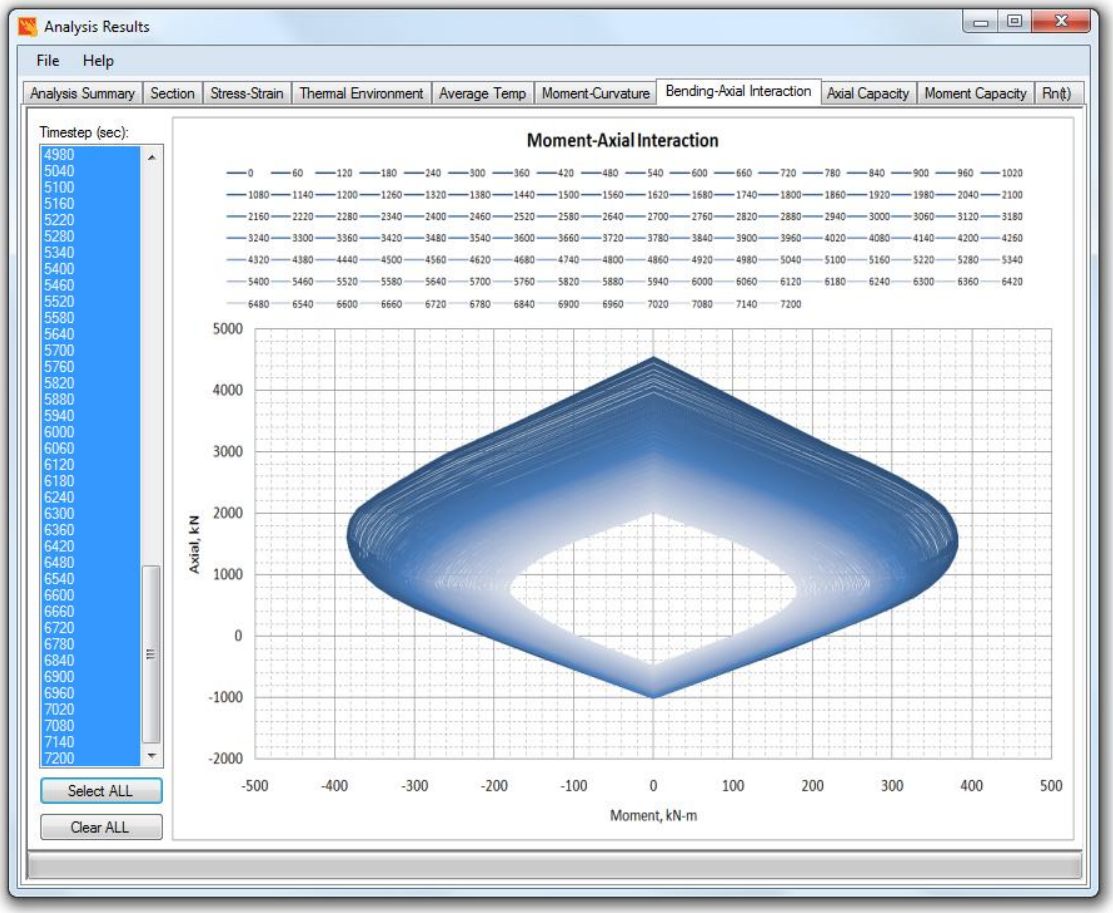


Figure 6.18 Example Section Moment-Axial Interaction Output

Figure 6.19 shows the Results form with the normalized results ($R_n(t)$) tab of the analysis selected. This can be looked at as a summary of all the calculations completed. This figure plots cross-section capacity as a function of time for four cases: flexural capacity for positive bending (tension on bottom), flexural capacity for negative bending (tension on top), axial capacity in compression, and axial capacity in tension. The gas temperature-time curve that was used for the analysis is also plotted. All of these quantities are normalized to their peak values. Gas temperatures are normalized to the peak gas temperature and member capacities are normalized to their capacities at 20°C. For example, a normalized capacity of 0.80 indicates that the cross-section retains 80% of its room temperature capacity.

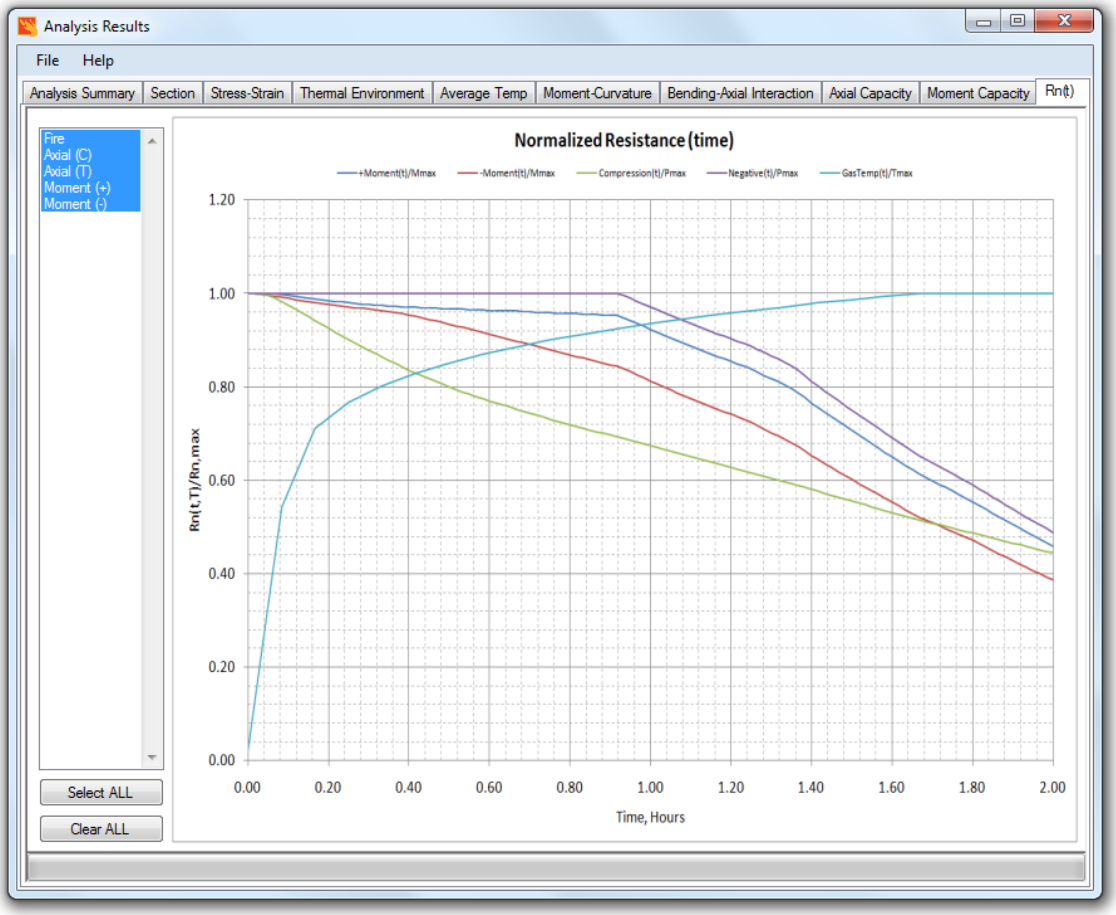


Figure 6.19 Example Section Normalized Capacity Output

6.6.7 Assumptions and Limitations of UT Fire: R/C Analysis

Though UT Fire: R/C Analysis is a useful tool for analyzing structures exposed to fire, there are some limitations that the user should be aware of. A full list can be found under 'Help' in the program, and a few of the most pertinent are listed here. These assumptions and limitations include:

1) The section must be composed of only rectilinear elements. UTF:R/C calculates the area and centroid of each element based on this assumption.
2) The section must be composed of less than 5000 nodes.
3) The analysis assumes that the section will regain its capacity once cool.
4) No spalling is present.

6.7 VALIDATION OF ROOM TEMPERATURE ANALYSIS

To provide some limited validation of UT Fire: Reinforced Concrete Analysis, comparisons were made with RECONASANCE, a reinforced concrete analysis tool developed at the University of Texas by Professor Richard Klingner, PhD. The comparisons are made at room temperature (t = 0), since RECONASANCE does not provide capabilities for analysis at elevated temperature.

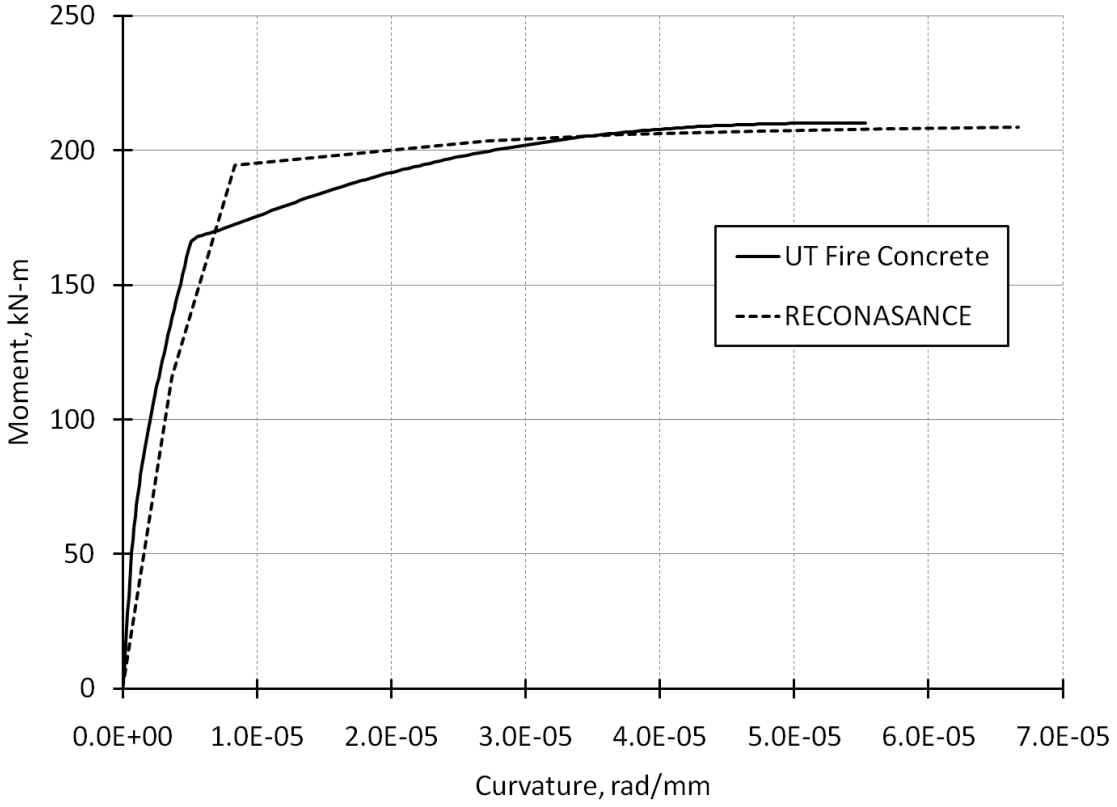


Figure 6.20 Example Section Moment-Curvature

As can be seen in Figures 6.20, the programs give similar moment-curvature results. The differences come from somewhat different constitutive relationship equations. Also, UTF:R/C stops calculating when moment begins to drop because there are no strain hardening effects considered in the Eurocode stress-strain equations for reinforcing steel.

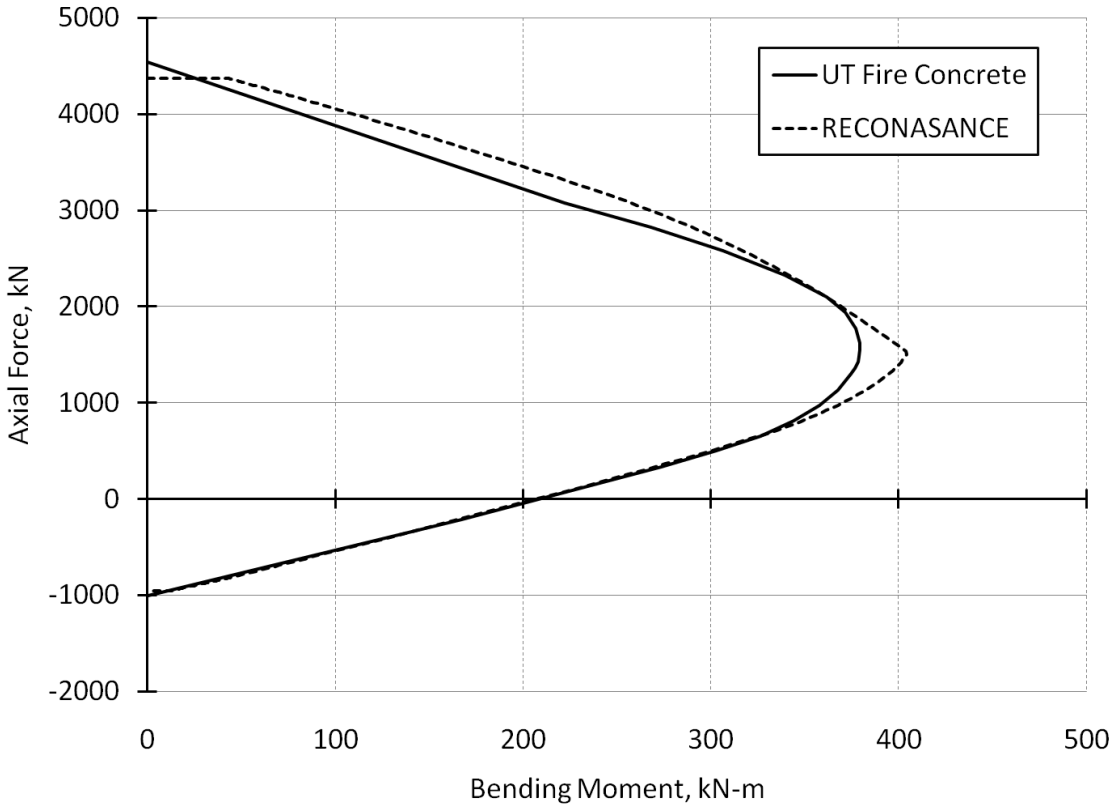


Figure 6.21 Example Section Moment-Axial Interaction

Figure 6.21 shows the a comparison between the moment-axial interaction plots calculated by UTF:R/C and RECON. Again, the plots are very similar. The differences are primarily a result of the RECON assumption that the maximum concrete strain is 0.003. UT Fire Concrete cannot make this assumption because each fiber has a different maximum strain once the section becomes heated.

6.8 COMPARISON WITH 500° ISOTHERM METHOD

There are several simplified methods reported in the literature for computing the capacity of reinforced concrete cross-sections at elevated temperature. One of the more widely cited methods is the 500°C Isotherm Method. This method is recommended by Eurocode 2 (2004), and is described in detail by Purkiss (2007). It was previously mentioned that some simplified methods exist to calculate section capacities by hand. In

this section, the 500°C Isotherm Method will be explained and the results compared with the results from UTF:R/C.

The 500°C Isotherm Method simplifies the analysis of R/C elements at elevated temperatures by eliminating the need to find temperature dependent constituitive relationships for each individual element in the section. This is done by assuming any concrete at a temperature less than 500°C has the same mechanical properties as that of concrete at room temperature, and ignoring any concrete on portions of the cross-section where the temperature exceeds 500°C. Note that the 500°C Isotherm method still requires that a heat transfer analysis be conducted to determine the location of the 500°C Isotherm. An "Isotherm" is a line of constant temperature. The 500°C Isotherm is therefore the line on the cross-section where the temperature is 500°C.

With this type of simplified analysis, the only information needed is the depth of the 500°C isotherm and the temperature of the reinforcing steel in the section. This isotherm can be found by equations such as those derived by (Purkiss 2007) or by published isotherm maps for generic sections (also provided in Purkiss 2007). These equations and published isotherms predict the amount of concrete above 500°C based on the length of time exposed to the fire and the dimensions of the section, but are only valid for standard fire curves, such as ASTM E119 or ISO 834. The equations can also be used to predict the temperature of the reinforcement based on the amount of concrete cover.

Another option for determining the depth of the 500°C isotherm is to conduct a SAFIR heat transfer analysis. For this example, the SAFIR model has already been run and the results from it will be used. Figure 6.22 shows the position of the 500°C isotherm on the example section analyzed in this chapter at two hours with the ASTM E119 fire exposed to all sides. The rounded edges of the isotherm have been ignored and the dimensions of the reduced section are measured at the center lines of the section. The actual temperature of the reinforcement has been used to determine its reduced maximum stress level.

The normalized compressive and bending moment capacities calculated by the 500°C Isotherm Method and UTF:R/C are compared in Figure 6.23. This plot shows that the simplified hand calculations can predict reduced ultimate capacities that are very similar to those calculated by the more complex analysis procedure of UTF:R/C.

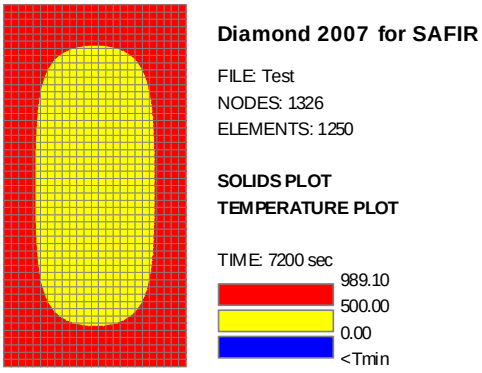


Figure 6.22 Example Section 500°C Isotherm at 120 minutes

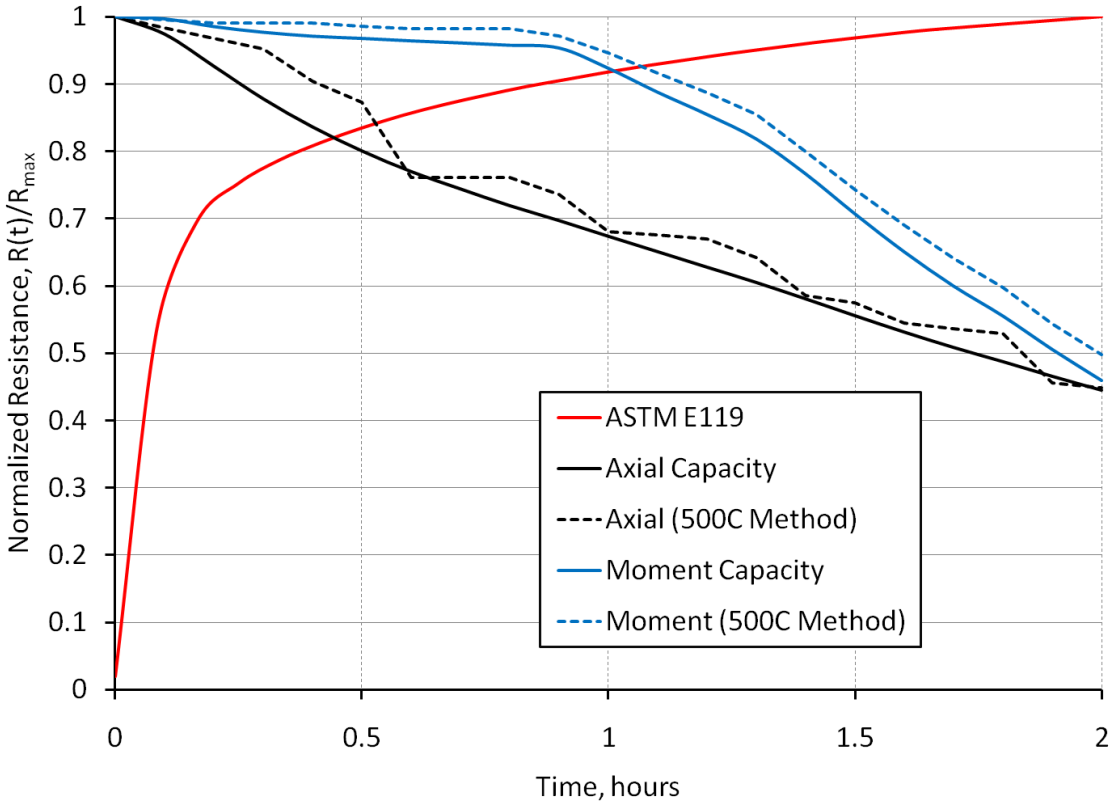


Figure 6.23 Normalized Resistance as a Function of Time

Though Figure 6.23 shows that the 500°C Isotherm Method produces results similar to those obtained from UTF:R/C and requires considerably less complex calculations, it is still not necessarily the most appropriate method for evaluating the load carrying capability of an entire structure during a fire. Published isotherms currently only exist for standard fire curves, and are not available for case specific gas temperature-time curves, such as those generated by OZone for analysis of the FOA structure. Also, in the case of the FOA, over 40 different members and reinforcement layouts needed to be analyzed. Consequently, computer generated heat transfer and cross-section analysis, such as provided by SAFIR and UT Fire: R/C Analysis offers greater flexibility in choice of fire exposure and greater speed in analyzing numerous cross-sections.

6.9 SUMMARY

In this chapter, a computer program that provides an estimate of the reduced cross-section capacity of reinforced concrete members exposed to fire, UT Fire: R/C Analysis, was developed and described. While discussing the development of this program, the mechanical properties of concrete and steel at elevated temperatures were discussed. An example problem was presented and taken step-by-step through the program to further explain its usage. The program was verified by comparing the results at room temperature with the computer program RECONASANCE. The results were also compared with simplified hand calculation methods. UTF:R/C will be used in the following chapter to estimate the reduced capacities of a sample of members from the FOA from the results of the SAFIR heat transfer analyses presented in Chapter 5.

CHAPTER 7
Reduced Capacity Analysis for FOA Members

7.1 OVERVIEW

The last chapter discussed the development of tools for reduced capacity analysis of structural members exposed to fire. The computer program that was developed (UT Fire: Reinforced Concrete Analysis, UTF:R/C) will be utilized in this chapter to study the effects the fire of May 13, 2008 had on a sample of the members at the TU Delft Faculty of Architecture Building (FOA). The members selected for study in this chapter were those present on the 6^{th}-8^{th} floors of the northwest wing of the FOA, the portion of the structural system where the collapse may have initiated. The analysis of these members is meant to provide give a preliminary estimate of the amount of load carrying capacity that these members likely lost during the fire. As is the case for this entire thesis, the objective of the analysis is not to develop conclusions on the cause of the collapse, but to provide data to guide future detailed investigations of the collapse.

7.2 MEMBERS STUDIED

Recall that the heat transfer analysis from Chapter 5 was conducted for three structural element types in the FOA. These were the 500mm columns, 250mm joists, and 400mm joists. Though these three member sizes were used in a large portion of the structural system, the reinforcing of each member type was highly variable throughout the FOA tower. To determine if a particular joist or column may have been more vulnerable to strength loss due to the fire and thus merit more in depth study in the future, all of the reinforcement patterns for these members used on the 6^{th}-8^{th} floors of the northwest wing have been analyzed. The joist and column labels used in this chapter are the same as those presented in Chapter 2. The reader should refer to the tables and

drawings from that chapter for information on the section reinforcements that were input into UT Fire: R/C Analysis.

7.3 RESULTS

For this preliminary analysis to be of the most use to future investigations, the results will be presented in a number of ways. Recall that the primary function of UTF:R/C is to conduct moment-curvature and moment-axial analyses at discrete time steps through the fire. For the flexural members (joists) the results will be first presented as a series of moment-curvature plots. The results for the columns will focus on the moment-axial interaction plots. From these plots, the ultimate axial and bending moment capacities of the sections are extracted and presented as functions of time, thereby providing an indication of the relative strength loss through the fire. All of the analysis results presented in this chapter are based on the gas temperature-time curve defined by the OZone baseline compartment analysis described in Chapter 4.

7.3.1 Joists

Figure 7.1 shows typical room temperature moment-curvature relationships for the two types of joists (250mm and 400mm) present in the FOA. The moment-curvature relationships vary from that shown in Figure 7.1 depending on the amount of reinforcement in the joist. However, the relationships shown in Figure 7.1 provide a general idea of the differences in behavior expected at room temperatures for the two major types of floor joists present in FOA. Note that within the portion of the northwest wing of the FOA that collapsed, only 250mm joists were used.

As described in Chapter 6, UTF:R/C ends its moment-curvature calculations at the point when the bending moment drops with an increase in curvature. There are no strain hardening effects considered in the reinforcement models recommended by Eurocode and used in UTF:R/C. This means that the section cannot withstand a second increase in moment once concrete begins to crush. It is at this point that failure is defined for the purposes of this study. With this definition of failure in mind, note that Figure 7.1

shows that, though they were the only joists in the collapsed portion of the building, the 250mm joists could withstand almost twice the curvature deformation as the taller 400mm joists.

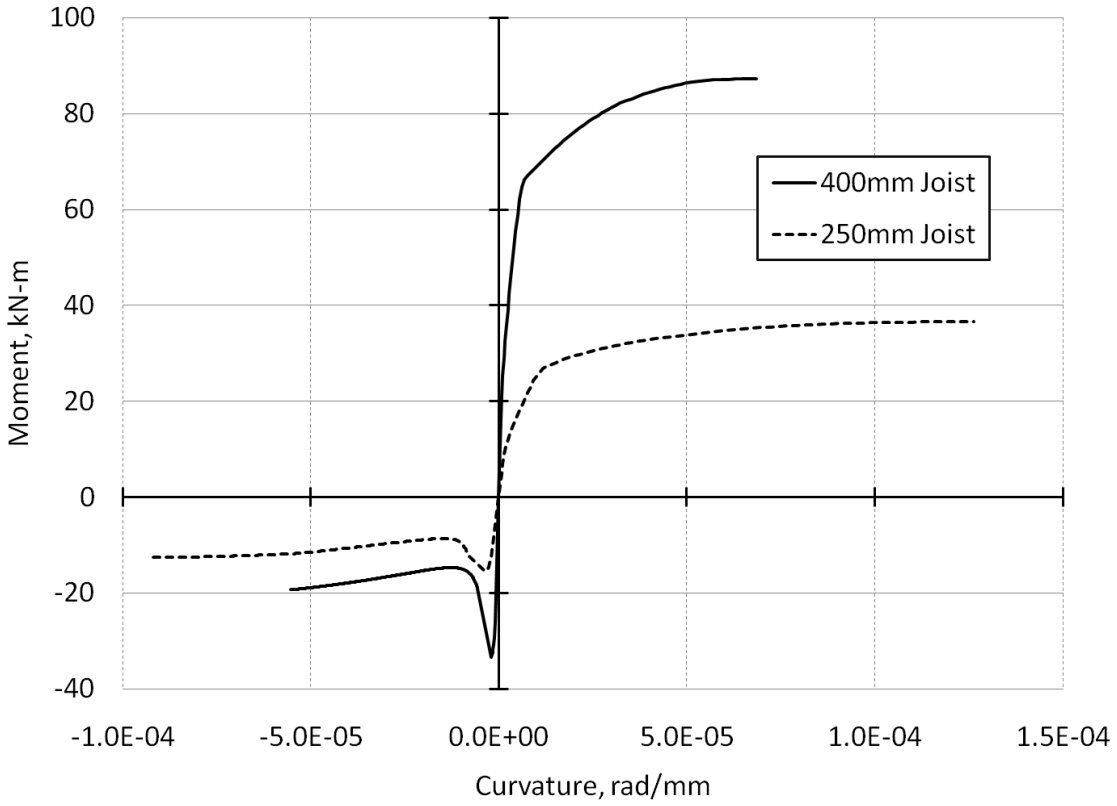


Figure 7.1 Joist Moment-Curvature Relationships at Room Temperature

Figures 7.2 and 7.3 show the change in moment-curvature behavior that two joist types could be expected to undergo during the fire. Each line represents a moment-curvature relationship for the section at a given time step, with a known temperature distribution. As would be expected, the maximum moment decreases as the fire and internal sectional temperatures increase. Also, the initial slopes of the of the moment-curvature relationships decrease with an increase in temperature, indicating a reduction in bending stiffness of the joists. The 250mm and 400mm joists exhibit very similar patterns of change in moment-curvature relationship through the fire.

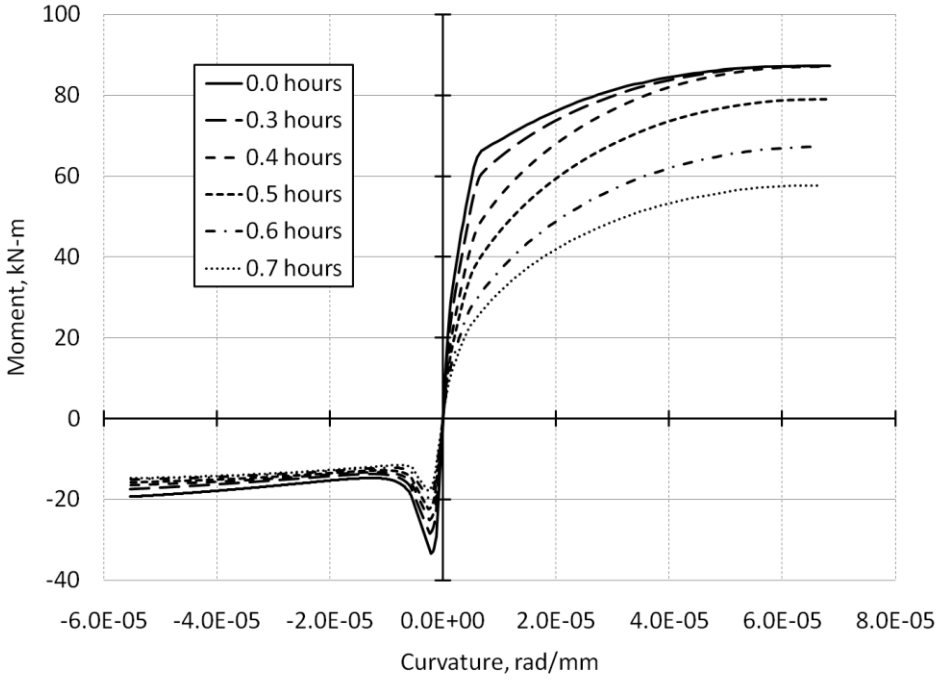


Figure 7.2 Joist S (400mm) Moment-Curvature Relationships at Various Times

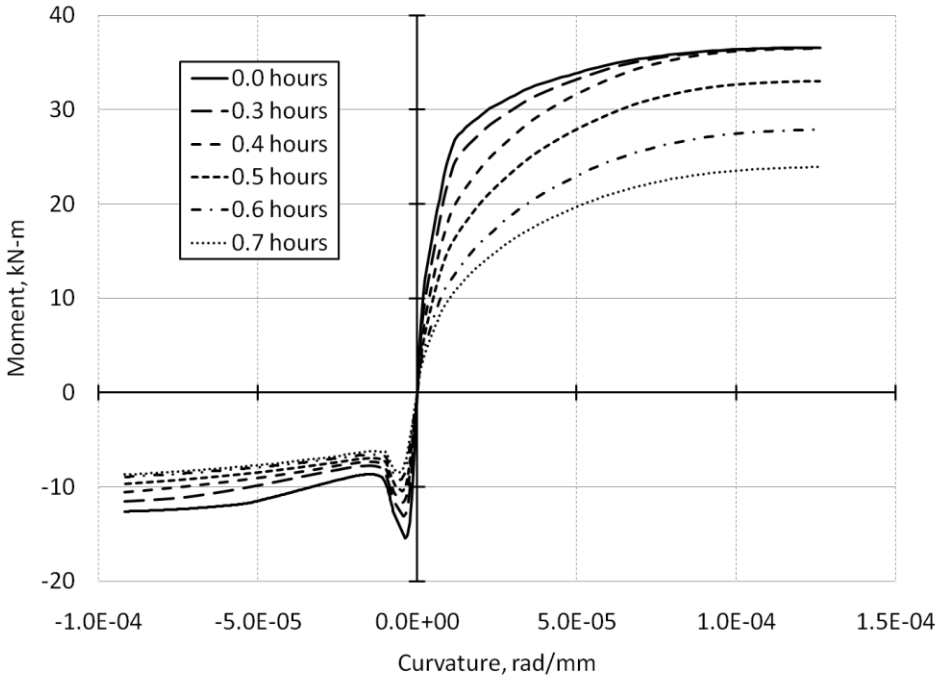


Figure 7.3 Joist G (250mm) Moment-Curvature Relationships at Various Times

7.3.2 Columns

Figure 7.4 shows room temperature moment-axial interaction plots for two columns present in the FOA. These were the columns on the 8th floor of the northwest wing. Column U3 was a corner column that supported a cantilevered balcony and was more heavily reinforced that the rest of the columns in that wing (Columns U4-9), which all had the same reinforcing. The figure shows that the corner column has significantly higher strength than the remainder of the columns.

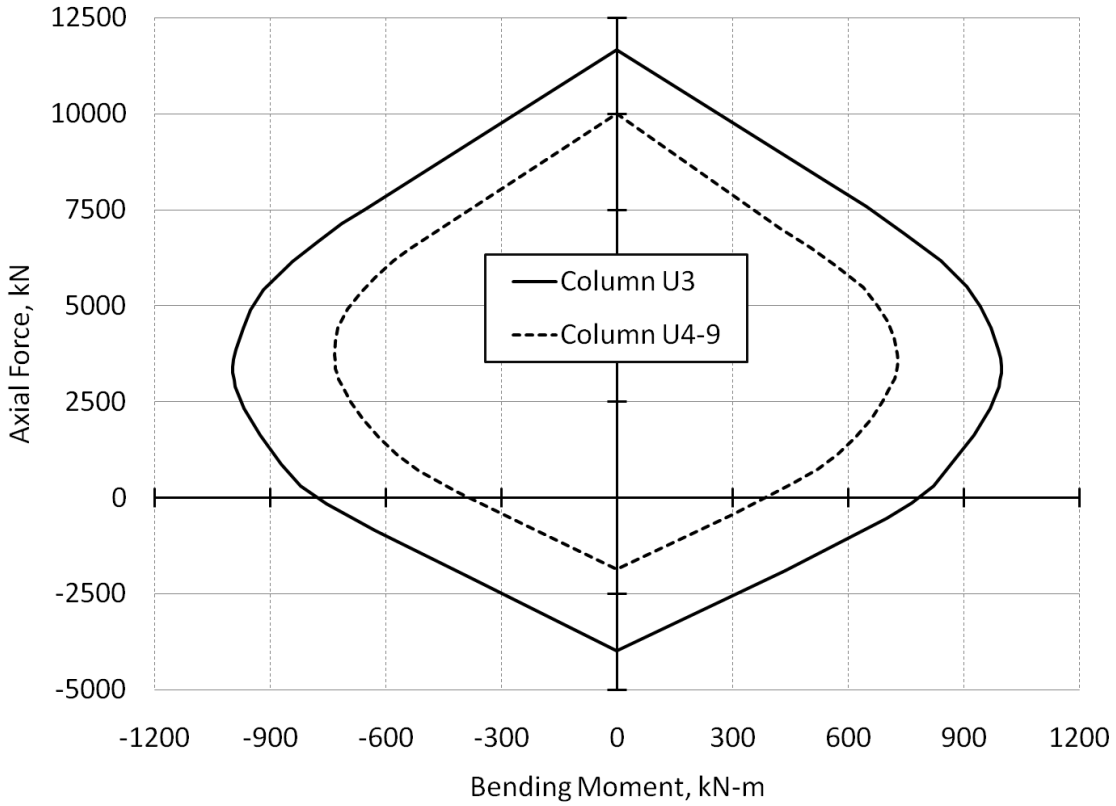


Figure 7.4 Room Temperature Moment-Axial Interaction for Columns at 8th Floor

Figures 7.5 and 7.6 show the change in moment-axial behavior that the two 8th floor column types could be expected to undergo during the fire. Each line represents a moment-axial relationship for the section at a given time step, with a known internal

temperature distribution. As would be expected, the maximum compressive axial forces and associated bending moments both decrease as the fire and sectional temperatures increase. However, note that, though these columns were not in tension, the axial tension capacity of the sections did not decrease with increasing temperature.. Since the sections depend fully on the reinforcement for the little tensile capacity they possessed, this indicates that the reinforcement never reached a temperature in excess of 400°C. According to the Eurocode 2 model for elevated temperature properties of reinforcing steel, which is used in UT Fire: R/C, no loss of strength occurs until the temperature exceeds 400°C. Columns U3 and U4-9 exhibit very similar patterns of change in moment-axial relationship through the fire.

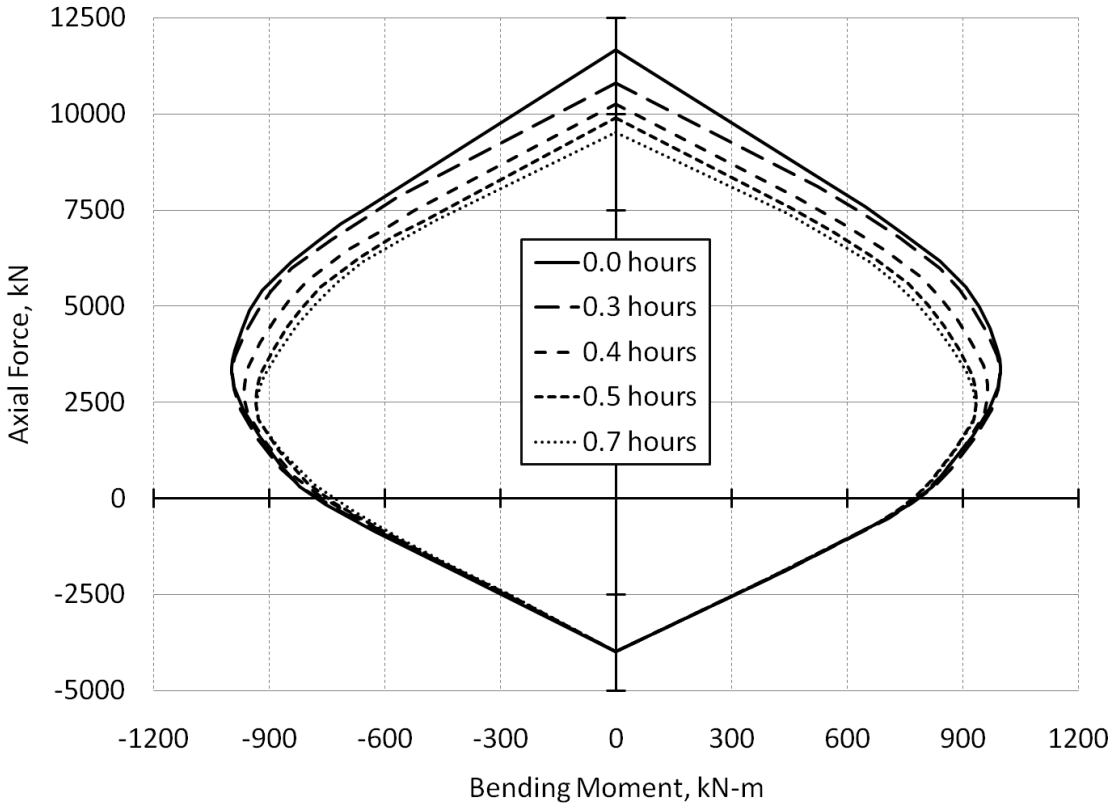


Figure 7.5 Column U3 (8th Floor)Moment-Axial Interaction Plots at Various Times

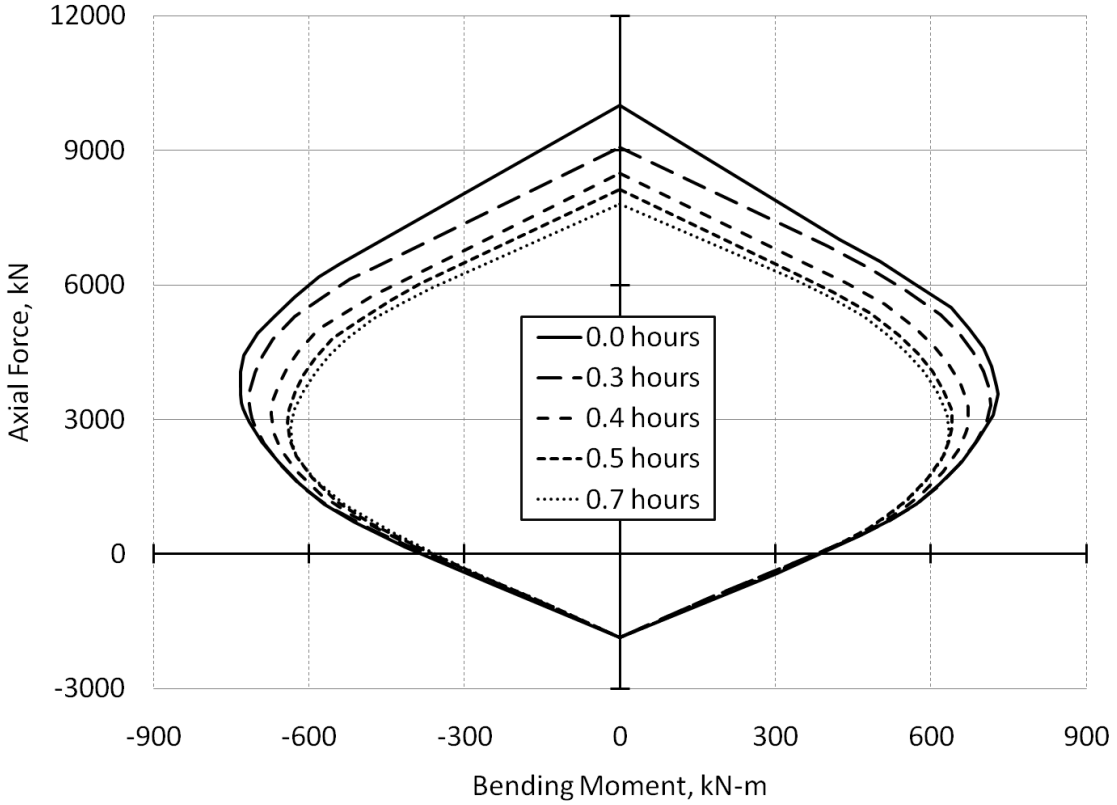


Figure 7.6 Column U4-9 (8th Floor) Moment-Axial Interaction Plots at Various Times

7.4 NORMALIZED RESULTS

Recall from Chapter 6 that a normalized curve refers to a plot where all of the quantities are normalized to their peak values. Gas temperatures are normalized to the peak gas temperature and member capacities are normalized to their capacities at 20°C. This allows the capacities as a function of time for many different members to be compared on the same plot and shows which members were most affected by the fire.

The plots of normalized capacity as a function of time are provided in Figure 7.7 for a typical joist (Joist G, 250mm) and a typical column (Column U3, 8th Floor). It can be seen in this figure that as gas temperature increases, member capacity decreases. Similarly, as the gas temperature decreases, the members begin to regain capacity. The analysis predicts that the members reach their minimum capacity about 45 minutes into the fire. At this point, the gas temperature is well past its peak (which occurs at

approximately 30 minutes) and has already cooled substantially. This occurs because, as demonstrated from the heat transfer analysis in Chapter 5, the temperatures in the interior portion of the member continue to increase for some time after the gas temperature begins to decrease. Thus, the members are at their weakest condition well into the cooling phase of the fire. This plot also shows that the most significant loss of capacity occurs for flexure in the joists. More specifically, negative bending shows the most capacity degradation. At their respective minimums, the negative moment capacity is reduced to roughly 55% of its normal capacity, while the positive moment capacity of this joist is reduced to 65% of its normal value. This significant loss of capacity may be the result, in part, to the rather small concrete cover in these members. Note that the loss of axial capacity in the columns is rather small. The column shown retained more than 80% of its normal axial capacity throughout the analysis.

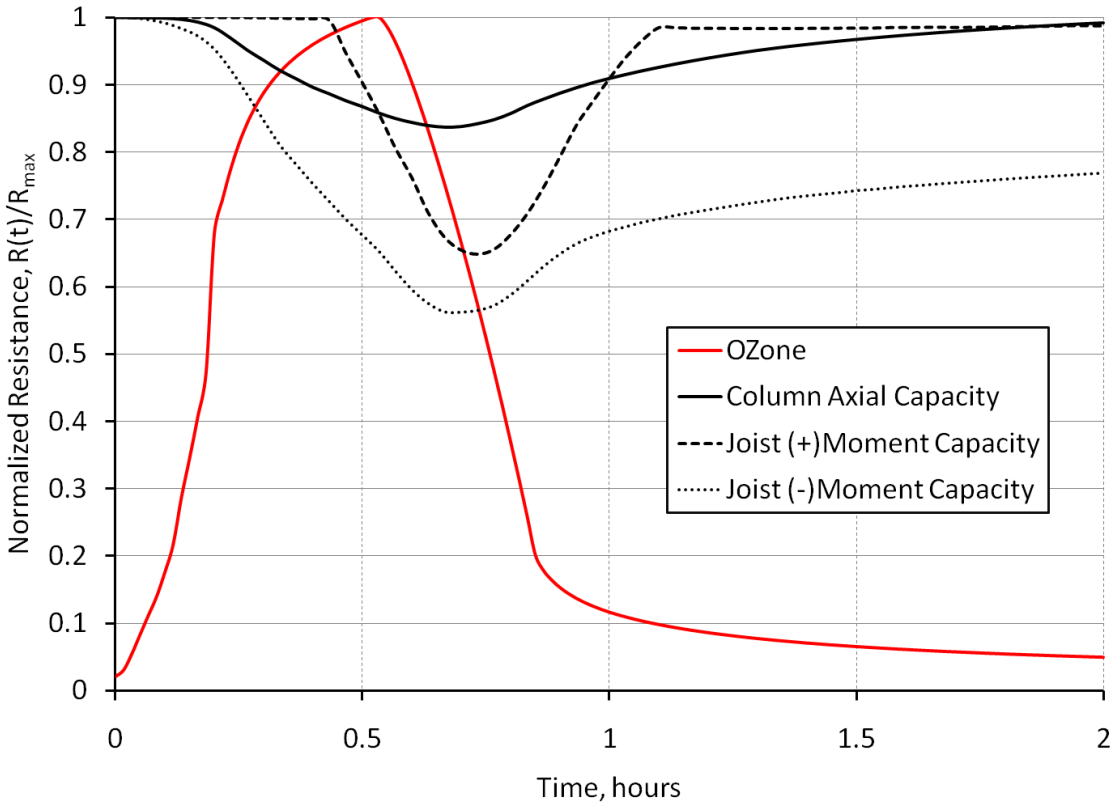


Figure 7.7 Typical Normalized Member Resistance as a Function of Time. Joist G and Column U3 are shown.

Recall that the each member type (500mm columns, 250mm joists, and 450mm joists) had a wide variety of reinforcement layouts that varied throughout the building. In Figures 7.8, 7.9, and 7.10 a closer look at the normalized results for each member type is provided. Each member type is broken up into the various reinforcement layouts that existed on the northwest wing of the 6th to 8th floors and the capacity variations as functions of time are compared. For the details of the reinforcement for each joist and column, please refer Tables 2.3 and 2.4 of this report.

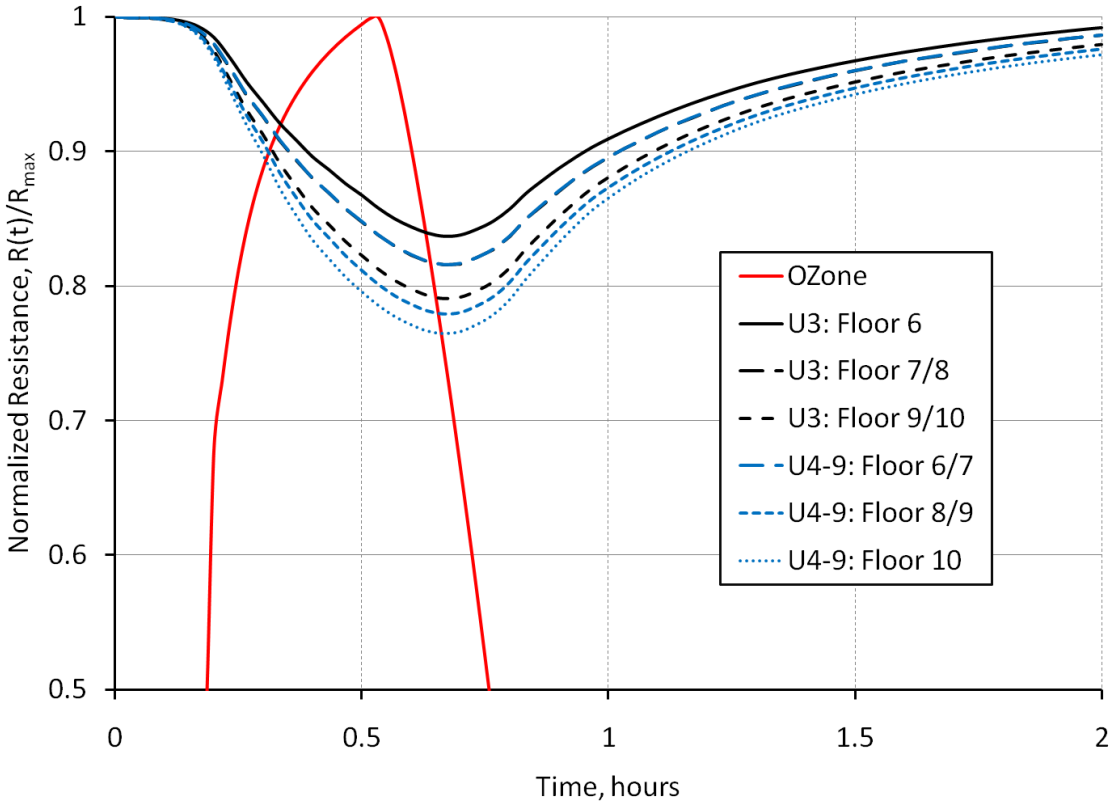


Figure 7.8 Normalized Column Axial Compression Capacity as a Function of Time

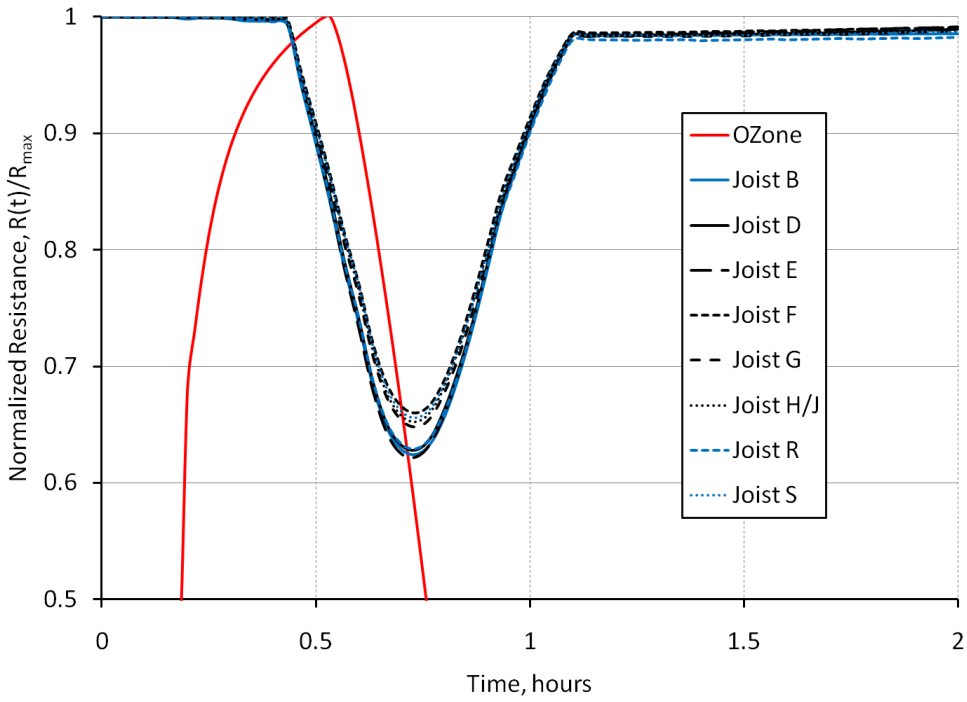


Figure 7.9 Normalized Joist Positive Moment Capacity as a Function of Time

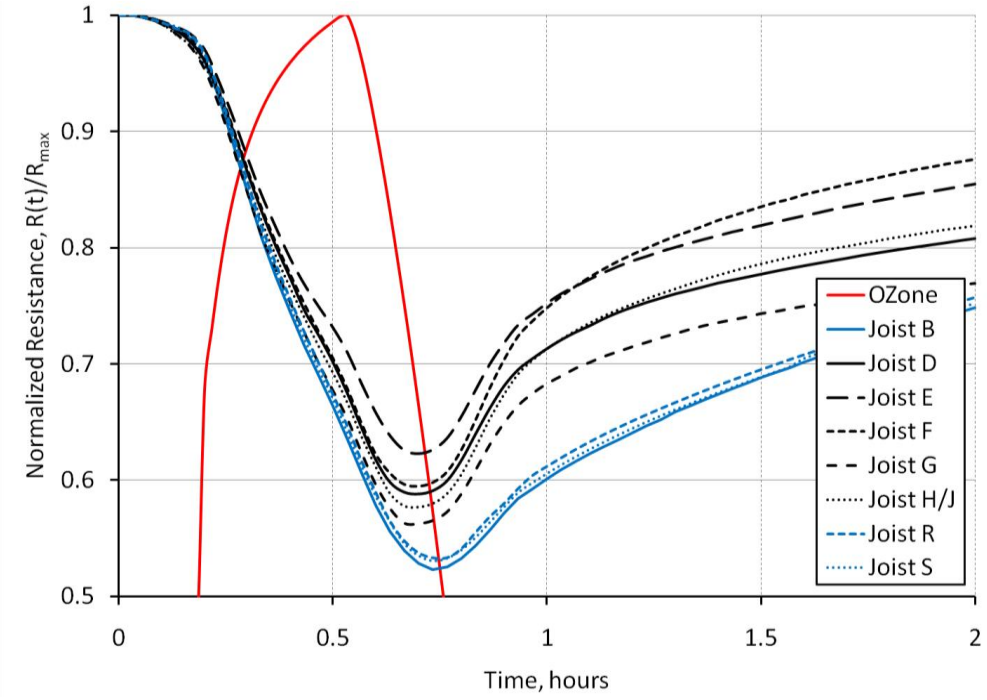


Figure 7.10 Normalized Joist Negative Moment Capacity as a Function of Time

Figure 7.11 shows a plot that was constructed to compare normalized member capacities as a function of gross reinforcement area for each member that was analyzed. This was done to establish whether the loss of capacity could be correlated with the amount of reinforcement provided in the section.

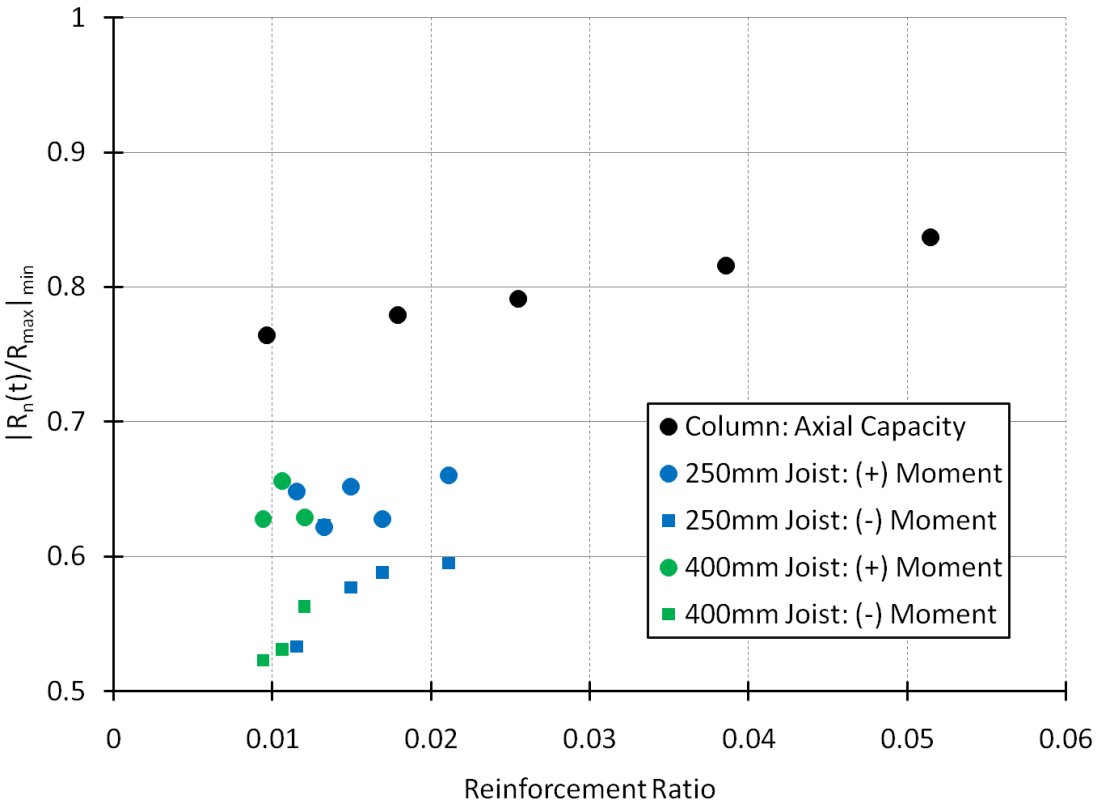


Figure 7.11 Normalized Capacity as a Function of Gross Reinforcement Ratio

In general, it can be seen that an increase in the area of reinforcement decreases the amount of strength the same section can be expected to lose during the fire. For the column axial capacities, the relationship between reinforcement ratio and loss of strength shows a nearly linear correlation. The joist bending moment capacities are less clearly correlated, though they do trend in the same direction as the columns.

7.5 SUMMARY

The program developed in Chapter 6 was utilized in this chapter to analyze a sample of the critical members in the FOA. It was shown that the joists lost a considerably greater percentage of their 20°C capacities than the columns. The analyses also showed that the minimum cross-sectional strength for each type of member studied occurred well into the cooling phase of the fire. In the next chapter the information and analysis results presented in this preliminary study will be summarized and a few general conclusions that have been drawn will be provided.

CHAPTER 8
Summary and Direction for Future Studies

8.1 SUMMARY

A preliminary study of the fire and collapse of the Faculty of Architecture Building (FOA) at the Delft University of Technology on May 13, 2008 has been presented in this report. Collapse of a major reinforced concrete structure in fire, as occurred with this building, is a rare event. Consequently, it is important to understand the factors that led to the collapse, and the implications these factors have on our understanding and practices for assessing the structural fire resistance of reinforced concrete structures.

The overall objective of this thesis was to conduct preliminary analysis to contribute to subsequent detailed studies of this structural collapse. It was not the objective of this thesis to conduct a detailed analysis of the collapse, but rather to contribute to more detailed and more comprehensive subsequent studies of the collapse.

As a first step in this study, it was necessary to obtain a clear description and understanding of the structural system of the building. Excellent records were kept from the original design and construction, and a nearly complete set of structural design drawings and calculations were available. Based on the available information, a thorough description of the structural system was presented in Chapter 2 of this report. This was followed by the assembly of an extensive database of photos of the FOA. This database included photos of the building before, during and after the fire of May 13, 2008. The photo database was used to assemble a preliminary timeline of the progress of the fire through the building through observations of flame and smoke visible in various areas of the building throughout the fire. Keys photos were also identified that provided insights into the structural collapse.

As a next step, to estimate temperatures to which the structural members were exposed, simplified compartment fire analyses were conducted for a typical office in the northwest wing of the building. A one-zone approximation was used to characterize the post-flashover conditions. A series of analyses were conducted using the computer program OZone. Published compartment time-temperature curves were also evaluated. Parametric studies were conducted to consider a reasonable range of fire loads, ventilation conditions, and compartment boundary thermal properties. A single representative gas temperature versus time curve, generated by OZone, was then selected for subsequent heat transfer analysis of the structural members. For this curve, the predicted peak temperature was on the order of 950°C, and the predicted duration of high temperature, defined as temperatures greater than 400°C was on the order of 40. The time frame for which severe burning was observed in any given location on the northwest wing from the photo database presented in Chapter 3 seems to suggest the predicted duration was reasonable.

Results of the compartment fire analysis were then used to conduct two-dimensional heat transfer analyses for selected column and joist cross-sections. The heat transfer analyses provided data throughout the cross-sections as a function of time. The heat transfer analyses were conducted using the computer program SAFIR2007 together with the preprocessor UT Fire. Thermal material properties were taken from Eurocode 2. Finally, using the cross-section temperature distributions predicted from the heat transfer analysis, member cross-section flexural and axial capacities were computer as a function of time. Elevated temperature mechanical properties for concrete and reinforcing steel were also based on Eurocode 2. Member cross-section capacities were computed using a computer program (UTF:R/C) that was custom developed for this project. Analysis of member cross-section strength was conducted for typical columns and floor joists that were used in the northwest wing of the building. When subject to the OZone gas temperature versus time curve, the analysis showed that during the fire, the columns

retained approximately 80-percent of their room temperature capacity. The floor joists, on the other hand suffered greater strength loss. The analysis showed that, during the fire, the flexural capacity of a typical joist under positive bending (tension on bottom) reduced to approximately 65-percent of its room temperature value. A factor that may have contributed to the significant strength loss in the joists was the rather small concrete cover over the reinforcement, as described in Chapter 2.

8.2 AREAS FOR FURTHER INVESTIGATION OF FOA COLLAPSE

An objective of this study was to identify possible areas that merit closer study in future detailed investigations of the structural collapse of the FOA during the fire of May 13, 2008. Although all analyses and observations made in this thesis are preliminary in nature, the results of this study suggest a possible collapse scenario that should be considered in future investigations. This collapse scenario is illustrated in Figure 8.1.

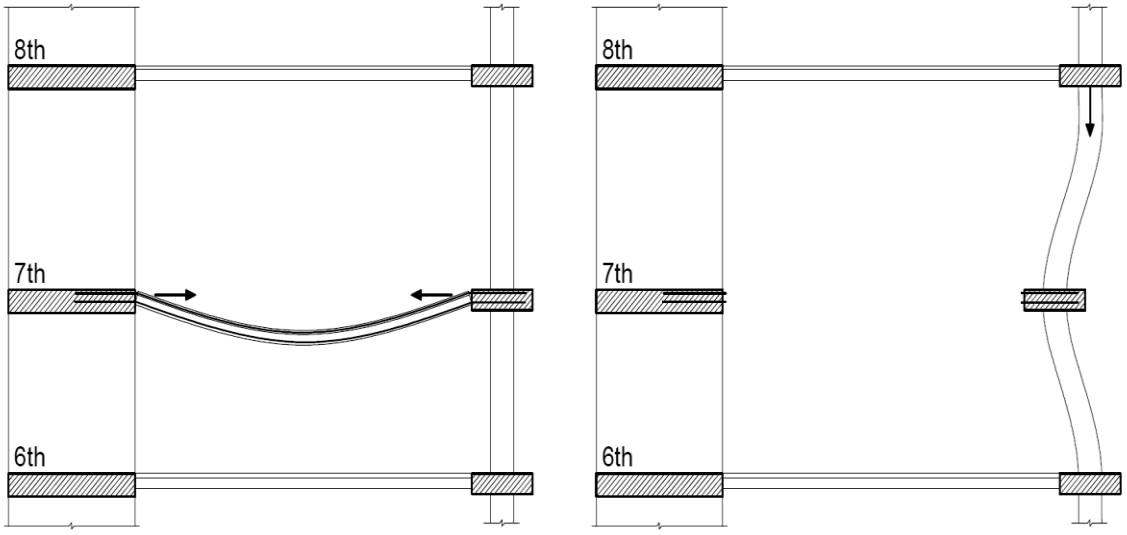


Figure 8.1 Possible Collapse Scenario

In this collapse scenario, large vertical deflections are developed in the floor joists as a result of a significant loss of flexural capacity, as suggested by the analysis in Chapter 7. As deflections increase, the joists begin to develop catenary action, and

therefore begin to develop axial forces that pull on the adjoining girders. As discussed in Chapter 2, the available drawings suggest that the connection between the floor joists and the girders may have been incapable of transmitting significant axial tension. Failure of the joist to girder connections could then have precipitated failure of the columns, which are now unsupported over two stories and have already been pulled inward by the sagging joists. Further, it is hypothesized that the failure of the joist to girder connections may have occurred on the 7th floor, based on the photos of the building after the collapse (Figure 3.7). Failure of the columns near the 7th floor as a key event triggering the collapse is also supported by photographic evidence (Figure 3.2).

The collapse scenario suggested above is speculative, but is supported by the photographic evidence and the preliminary structural analysis presented herein. It is recommended that future investigations evaluate this proposed scenario through more detailed analysis.

When considering the collapse scenario suggested above, as well as other collapse scenarios, there are a number of areas where future detailed investigations of this event would be highly beneficial, as follows:

- Improved Fire Modeling.

  Accurately charactering the fire environment to which the structure was exposed is a key requirement for predicting structural response to the fire. In this study, only very simple fire modeling was conducted. A better approximation of the fire environment is possible using computational fluid dynamics models such as the NIST Fire Dynamics Simulator (FDS). Development of FDS models for this fire would require additional information on the characteristics and distribution of fuel items in various building spaces, and additional information on building layout and the characteristics of compartment boundaries.

- Improved Structural Analysis.

  This study only considered preliminary analysis of member cross-section strength. To obtain a more accurate picture of the response of the FOA structure to the fire, more comprehensive and detailed models are needed. More specifically, three dimensional finite element models are needed for a better assessment of structural response. Such models not only consider strength loss at elevated temperature, but also consider thermally induced forces and deformations and the interaction of structural elements.

- Experimental Evaluation of Critical Members and Details.

  Experiments on selected members and details would be highly desirable to provide validation of computational predictions and data on details that are very difficult to model. Elevated temperature experiments on joists, girders and columns that closely replicate the members in the FOA would be beneficial to validate model predictions. Of particular interest in these tests would be information on the impact of shear on the elevated temperature performance of the joists and girders. As described in Chapter 2, the amount of shear reinforcement provided in these members was about one-half of what would be required by current US codes. It would also be desirable to test the joist to girder connections. As discussed above, these connections may have played an important role in the collapse, and predicting the behavior of these connections by computational models would be very difficult.

- Studies on Spalling.

  Spalling, which is the breaking off of pieces of concrete cover when a member is exposed to fire, can cause rapid loss of strength of a reinforced concrete member. The role that spalling may have played in the FOA collapse was not considered in this study, but should be evaluated as part of future studies. As

described in Chapter 3, there is photographic evidence of spalling in columns and in floor joists of the FOA in portions of the building that did not collapse. Further insights into the role of spalling in this collapse could be developed through carefully designed experiments on materials and members that replicate, as closely as possible, the aggregates and concrete mix design used in the original construction of the FOA. The potential for spalling can also be explored using limited available computational models for spalling.

APPENDIX A
Drawing List

This appendix provides a reference index for the drawings provided by the TU Delft Facilities Management Department. The drawings are labeled with a general discipline: Architectural, Structural, Civil, MEP, or Non-Tower Drawings. The drawing type refers to the type of view presented in the drawing, including: Sections, Elevations, Plans, or Details. The structural drawings were further labeled with the component that is detailed, such as: Columns, Joists, Girders, etc. The description column shows notes that further explain what is shown in the drawing.

DRAWING LABEL				DISCIPLINE	DWG. TYPE	STR. TYPE	DESCRIPTION
24	00	06	0484	Architectural	Section		C-C
24	00	06	0486	Architectural	Section		G-G
24	00	12	0905	Architectural	Elevation		Southwest
24	00	12	0906	Architectural	Elevation		Northeast
24	00	12	0907	Architectural	Elevation		Northwest
24	00	12	0908	Architectural	Elevation		Southeast
24	00	12	0909	Architectural	Elevation		Lower Levels
24	00	12	0910	Architectural	Elevation		Lower Levels
24	00	12	0911	Architectural	Section		A-A
24	00	12	0912	Architectural	Section		B-B
24	00	13	0974		Plan		Basement
24	00	13	0974a		Plan		
24	00	13	0975		Plan		
24	00	13	0976		Plan		
24	00	13	0977		Plan		
24	00	13	0978		Plan		
24	00	13	0979		Plan		
24	00	13	0980		Plan		
24	00	13	0981		Plan		
24	00	13	0982		Plan		

24	00	13	0983		Plan		
24	00	14	0984		Plan		
24	00	14	0994	Architectural	Plan		0th Floor (Basement)
24	00	14	0995	Architectural	Plan		1st Floor
24	00	14	0996	Architectural	Plan		2nd Floor
24	00	14	0998	Architectural	Plan		3rd + 4th
24	00	14	1000	Architectural	Plan		5th + 6th
24	00	14	1001	Architectural	Plan		7th + 8th
24	00	14	1002	Architectural	Plan		9th + 10th
24	00	14	1003	Architectural	Plan		11th + 12th
24	00	14	1004	Architectural	Plan		13th + 14th
24	00	14	1006	Architectural	Plan		15th + Roof
24	00	14	1088	Architectural	Plan		Basement?
24	00	14	1123-1	MEP			
24	00	14	1123-2	MEP			
24	00	17	1124	MEP			
24	00	17	1125	MEP			
24	00	17	1126	MEP			
24	00	17	1127	MEP			
24	00	17	1128	MEP			
24	00	17	1130	MEP			
24	00	17	1131-1	MEP			
24	00	17	1131-2	MEP			
24	00	17	1132	MEP			
24	00	17	1144	MEP			
24	00	17	1145-1	MEP			
24	00	17	1145-2	MEP			
24	00	17	1146	MEP			
24	00	17	1147-1	MEP			
24	00	17	1147-2	MEP			
24	00	17	1148	MEP			
24	00	17	1149	MEP			
24	00	17	1150	MEP			
24	00	17	1151	MEP			
24	00	17	1152	MEP			
24	00	17	1153	MEP			
24	00	17	1154	MEP			
24	00	17	1155	MEP			
24	00	17	1156	MEP			

24	00	17	1157	MEP			
24	00	17	1158	MEP			
24	00	17	1159	MEP			
24	00	17	1160-1	MEP			
24	00	17	1160-2	MEP			
24	00	17	1161	MEP			
24	00	17	1162-1	MEP			
24	00	17	1162-2	MEP			
24	00	17	1163	MEP			
24	00	17	1164	MEP			
24	00	17	1165	MEP			
24	00	17	1166	MEP			
24	00	17	1167-1	MEP			
24	00	17	1167-2	MEP			
24	00	17	1168	MEP			
24	00	17	1169	MEP			
24	00	17	1170	MEP			
24	00	17	1171	MEP			
24	00	17	1172	MEP			
24	00	17	1173	MEP			
24	00	17	1174	MEP			
24	00	17	1175	MEP			
24	00	17	1177	MEP			
24	00	17	1178	MEP			
24	00	17	1179	MEP			
24	00	17	1180	MEP			
24	00	17	1181	MEP			
24	00	17	1182	MEP			
24	00	17	1183	MEP			
24	00	17	1184	MEP			
24	00	20	1265	Structural	Detail	Other	?????
24	00	21	1266	Architectural	Plan		Basement
24	00	21	1267	Architectural	Plan		Basement
24	00	21	1268	Architectural	Plan		1st Floor
24	00	21	1269	Architectural	Plan		1st Floor
24	00	21	1270	Architectural	Plan		2nd Floor
24	00	21	1271	Architectural	Plan		2nd Floor
24	00	22	1313	Architectural	Plan		
24	00	22	1314	Architectural	Plan		

24	00	22	1315	Architectural	Plan		
24	00	22	1316	Architectural	Plan		
24	00	22	1317	Architectural	Plan		
24	00	22	1318	Architectural	Plan		
24	00	22	1346	Architectural	Section		
24	00	22	1354	Structural	Detail	Other	Top High Parapet
24	00	22	1355	Structural	Detail	Other	Under High Parapet
24	00	22	1356	Structural	Detail	Other	Under High Parapet
24	00	22	1357	Structural	Detail	Other	High Eaves Detail
24	00	22	1358	Structural	Detail	Other	13th Floor Terrace Parapet
24	00	24	1458	Architectural	Elevation		
24	00	25	1506	Architectural	Elevation		
24	00	27	1635	Structural	Plan	Plans	Foundation (Piles + Caps)
24	00	27	1658	Civil	Plan		
24	00	27	1659	Civil	Plan		Site Plan
24	00	27	1662	Structural	Detail	Girders	Grade Beams
24	00	27	1663	Structural	Detail	Girders	Grade Beams
24	00	27	1664	Structural	Detail	Girders	Grade Beams
24	00	27	1665	Structural	Detail	Girders	Grade Beams
24	00	27	1666	Structural	Detail	Girders	Grade Beams
24	00	27	1667	Structural	Detail	Girders	Grade Beams
24	00	27	1668	Structural	Detail	Girders	Grade Beams
24	00	27	1669	Structural	Detail	Girders	Grade Beams
24	00	27	1670	Structural	Detail	Girders	Grade Beams
24	00	27	1671	Structural	Detail	Girders	Grade Beams
24	00	27	1672	Structural	Plan	Other	?????
24	00	27	1673	Structural	Plan	Other	?????
24	00	27	1674	Structural	Plan	Other	?????
24	00	27	1675	Structural	Plan	Other	?????
24	00	27	1676	Structural	Section	Sections	Basement Section
24	00	27	1677	Structural	Section	Sections	Basement Section
24	00	27	1678	Structural	Section	Sections	Basement Section
24	00	27	1679	Structural	Detail		
24	00	27	1680	Structural	Detail		
24	00	27	1681	Structural	Detail		
24	00	27	1683	Structural	Section		
24	00	27	1684	Structural	Section		Stairs
24	00	27	1685	Structural	Section		
24	00	27	1686	Structural	Plan	Plans	Ground Floor of Tower (N)

24	00	27	1687	Structural	Plan	Plans	Ground Floor of Tower (S)
24	00	27	1688	Structural	Detail	Girders	
24	00	27	1689	Structural	Detail	Girders	
24	00	27	1690	Structural	Detail		
24	00	27	1691	Structural	Section	Other	Stairs
24	00	27	1692	Structural	Detail	Girders	
24	00	27	1693	Structural	Detail	Columns	Middle
24	00	27	1694	Structural	Detail	Columns	Middle
24	00	28	1695	Structural	Detail	Columns	Lower Floors (U4-U9)
24	00	28	1697	Structural	Section	Shear Walls	
24	00	28	1698	Structural	Section	Sections	
24	00	28	1699	Structural	Section	Sections	
24	00	28	1700	Structural	Section	Sections	
24	00	28	1703	Structural	Plan	Plans	1st Floor of Tower (N)
24	00	28	1705	Structural	Detail		
24	00	28	1706	Structural	Detail		
24	00	28	1707	Structural	Section		
24	00	28	1708	Structural	Plan		
24	00	28	1709	Structural	Detail	Columns	Lower Floors (U3)
24	00	28	1710	Structural	Plan	Plans	3rd Floor (N)
24	00	28	1711	Structural	Plan	Plans	3rd Floor (S)
24	00	28	1712	Structural	Detail	Other	Plate Column Detail
24	00	28	1713	Structural	Detail	Other	Plate Column Detail
24	00	28	1714	Structural	Detail	Girders	3rd Floor
24	00	28	1715	Structural	Detail	Girders	3rd Floor
24	00	28	1716	Structural	Detail	Joists	3rd Floor
24	00	28	1717	Structural	Plan	Plans	4th Floor (N)
24	00	28	1718	Structural	Plan	Plans	4th Floor (S)
24	00	28	1719	Structural	Detail	Girders	
24	00	28	1720	Structural	Detail	Girders	
24	00	28	1721	Structural	Detail	Joists	4th Floor
24	00	28	1722	Structural	Plan	Plans	5th Floor (N)
24	00	28	1723	Structural	Plan	Plans	5th Floor (S)
24	00	28	1724	Structural	Detail	Girders	
24	00	28	1725	Structural	Detail	Girders	
24	00	28	1726	Structural	Detail	Girders	
24	00	28	1727	Structural	Detail	Joists	5th Floor
24	00	28	1728	Structural	Detail	Shear Walls	

24	00	28	1729	Structural	Section	Sections	
24	00	28	1730	Structural	Section	Sections	
24	00	28	1731	Structural	Section	Sections	
24	00	28	1735	Structural	Section	Sections	
24	00	28	1736	Structural	Detail	Shear Walls	
24	00	28	1737	Structural	Section	Sections	
24	00	28	1739	Structural	Detail	Other	
24	00	28	1740	Structural	Section	Other	Stairs
24	00	28	1741	Structural	Section	Other	Stairs
24	00	28	1742	Structural	Detail	Columns	U4-U9
24	00	28	1743	Structural	Detail	Columns	U3
24	00	28	1744	Structural	Detail	Columns	
24	00	28	1745	Structural	Plan	Plans	6th Floor (N)
24	00	28	1746	Structural	Plan	Plans	6th Floor (S)
24	00	29	1747	Structural	Detail		
24	00	29	1748	Structural	Detail		
24	00	29	1749	Structural	Section		
24	00	29	1750	Structural	Section		
24	00	29	1751	Structural	Section		
24	00	29	1752	Structural	Plan	Plans	7th Floor (N)
24	00	29	1753	Structural	Plan	Plans	7th Floor (S)
24	00	29	1754	Structural	Detail	Girders	7th Floor
24	00	29	1755	Structural	Detail	Girders	7th Floor
24	00	29	1756	Structural	Detail	Girders	7th Floor
24	00	29	1757	Structural	Plan	Plans	8th Floor (N)
24	00	29	1758	Structural	Plan	Plans	8th Floor (S)
24	00	29	1759	Structural	Detail	Girders	8th Floor
24	00	29	1760	Structural	Detail	Girders	8th Floor
24	00	29	1761	Architectural		Plans	
24	00	29	1762	Architectural		Plans	
24	00	29	1763	Structural	Section	Sections	
24	00	29	1764	Structural	Section	Shear Walls	
24	00	29	1765	Structural	Section	Sections	
24	00	29	1766	Structural	Section	Sections	
24	00	29	1767	Structural	Plan	Plans	9th Floor (N)
24	00	29	1768	Structural	Plan	Plans	9th Floor (S)
24	00	29	1769	Structural	Detail	Girders	9th Floor
24	00	29	1769a	Structural	Detail	Girders	9th Floor

24	00	29	1770	Structural	Detail	Girders	9th Floor
24	00	29	1771	Structural	Plan	Plans	10th Floor (N)
24	00	29	1772	Structural	Plan	Plans	10th Floor (S)
24	00	29	1773	Structural	Detail	Girders	10th Floor
24	00	29	1774	Structural	Detail	Girders	10th Floor
24	00	29	1776	Structural	Plan	Plans	11th Floor (S)
24	00	29	1777	Structural	Detail	Girders	11th Floor
24	00	29	1778	Structural	Detail	Girders	11th Floor
24	00	29	1779	Structural	Detail	Girders	11th Floor
24	00	29	1781	Structural	Plan	Plans	13th Floor (N)
24	00	29	1782	Structural	Detail		
24	00	29	1783	Structural	Plan	Plans	13th Floor (S)
24	00	29	1784	Structural	Detail		
24	00	29	1785	Structural	Detail		
24	00	29	1786	Structural	Detail		
24	00	29	1787	Structural	Detail	Joists	13th Floor
24	00	29	1788	Structural	Detail	Joists	13th Floor
24	00	29	1789	Structural	Plan	Plans	14th Floor (N)
24	00	29	1791	Structural	Detail		
24	00	29	1792	Structural	Detail		
24	00	29	1794	Structural	Plan	Plans	15th Floor (N)
24	00	29	1795	Structural	Plan	Plans	15th Floor (S)
24	00	29	1796	Structural	Section		
24	00	29	1797	Structural	Section		
24	00	30	1798	Structural	Detail	Shear Walls	
24	00	30	1799	Structural	Section	Sections	
24	00	30	1800	Structural	Plan	Plans	12th Floor (N)
24	00	30	1801	Structural	Plan	Plans	12th Floor (S)
24	00	30	1802	Structural	Detail		
24	00	30	1803	Structural	Detail		
24	00	30	1804	Architectural	Plan		Roof
24	00	30	1805	Structural	Detail		
24	00	30	1806	Structural	Detail	Columns	Upper Floors
24	00	30	1807	Structural	Detail	Columns	Upper Floors
24	00	30	1808	Structural	Detail	Shear Walls	
24	00	30	1809	Structural	Section	Sections	
24	00	30	1810	Structural	Section	Sections	
24	00	30	1811	Structural	Section	Sections	

24	00	30	1812	Structural	Detail	Girders	
24	00	30	1813	Structural	Detail	Girders	
24	00	30	1814	Structural	Section	Sections	
24	00	30	1815	Structural	Detail		
24	00	30	1816	Structural	Detail		
24	00	30	1817	Structural	Detail		
24	00	30	1818	Structural	Detail		
24	00	30	1819	Structural	Detail		
24	00	30	1820	Structural	Detail		
24	00	30	1821	Architectural	Section		
24	00	30	1822	Structural	Detail		
24	00	30	1823	Structural	Detail		
24	00	34	2341	Architectural	Section		
24	00	34	2355	Structural	Detail		
24	00	34	2357	Structural	Detail		
24	00	34	2361	Structural	Detail		
24	00	34	2362				
24	00	34	2363				
24	00	35	2373				
24	00	35	2374				
24	00	35	2375	Non-Tower			
24	00	35	2377	Non-Tower			
24	00	35	2378	Non-Tower			
24	00	35	2379				
24	00	35	2380				
24	00	35	2382				
24	00	35	2383				
24	00	35	2385				
24	00	35	2386				
24	00	35	2388				
24	00	35	2390				
24	00	50	2146				
24	00	50	2147				
24	00	55	2461				
24	00	55	2464				
24	00	59	2480				
24	00	59	2500				
24	00	59	2501				
24	00	59	2508				

24	00	60	2511				
24	00	60	2527				
24	00	60	2534				
24	00	60	2539				
24	00	60	2542				
24	00	61	2558				
24	00	61	2559				
24	00	61	2560				
24	00	61	2564	Structural	Plan	Other	Hanging Floor
24	00	62	2600	Structural	Plan	Plans	11th Floor (N)
24	00	63	2612	Structural	Plan	Plans	14th Floor (S)
24	00	64	2637	Non-Tower			
24	00	68	2724	Non-Tower			
24	00	68	2725	Non-Tower			
24	00	68	2726	Non-Tower			
24	00	68	2727	Non-Tower			
24	00	68	2728				
24	00	68	2729				
24	00	68	2730				
24	00	68	2731				
24	00	69	2739	Calcs			foundations
24	00	69	2740	Calcs			retaining walls
24	00	69	2741	Calcs			column loads
24	00	70	2765	Calcs			
24	00	70	2755	Calcs			
24	00	70	2756	Calcs			
24	00	70	2757	Calcs			
24	00	70	2758	Calcs			
24	00	70	2759	Calcs			
24	00	71	2760	Calcs			
24	00	71	2761	Calcs			column design
24	00	71	2762	Calcs			
24	00	71	2763	Calcs			
24	00	71	2764	Calcs			
24	00	71	2765	Calcs			
24	00	71	2766	Calcs			
24	00	71	2767	Calcs			
24	00	71	2768	Calcs			
24	00	71	2769	Calcs			

24	00	71	2770	Calcs			
24	00	71	2771	Calcs			
24	00	71	2772	Calcs			
24	00	71	2773	Calcs			
24	00	72	2774	Calcs			
24	00	72	2775	Calcs			
24	00	72	2776	Calcs			
24	00	72	2777	Calcs			
24	00	72	2778	Calcs			
24	00	72	2779	Calcs			
24	00	72	2780	Calcs			
24	00	72	2781	Calcs			
24	00	72	2782	Calcs			
24	00	72	2783	Calcs			
24	00	72	2784	Calcs			
24	00	72	2785	Calcs			
24	00	72	2786	Calcs			
24	00	72	2787	Calcs			
24	00	72	2788	Calcs			
24	00	72	2789	Calcs			
24	00	72	2790	Calcs			
24	00	72	2792	Calcs			
24	00	72	2793	Calcs			
24	00	73	2794	Calcs			
24	00	73	2795	Calcs			
24	00	73	2798	Calcs			
24	00	90	007	Calcs			
24	00	90	022	Calcs			
24	BH	73	2799	Calcs			
24	CA	69	2745	Calcs			
24	HW	73	2802	Calcs			
24	HW	73	2803	Calcs			
24	KA	69	2747	Calcs			
24	KA	70	2753	Calcs			
24	LA	56	2400				
24	LA	56	2401				
24	LA	56	2406				
24	LA	56	2407				
24	LA	56	2408				

24	LA	56	2409				
24	LA	56	2410				
24	LA	56	2411				
24	LA	56	2412				
24	LA	66	2666				
24	LA	66	2667				
24	LA	66	2668				
24	LA	66	2669				
24	LA	66	2670				
24	LA	66	2671				
24	LA	66	2674				
24	LA	66	2675				
24	LA	66	2676				
24	LA	66	2677				
24	LA	66	2685				
24	LA	66	2686				
24	LA	69	2742	Calcs			
24	LA	69	2742a	Calcs			
24	LB	34	2347				
24	LB	34	2348				
24	LB	34	2350				
24	LB	34	2360				
24	LB	34	2364				
24	LB	34	2365				
24	LB	34	2366				
24	LB	34	2367				
24	LB	34	2368				
24	LB	64	2638				
24	LB	69	2737	Calcs			
24	LB	69	2738	Calcs			
24	LB	70	2750	Calcs			
24	LB	73	2797	Calcs			
24	LB	73	2804	Calcs			
24	LC	34	2342				
24	LC	34	2345	Structural	Detail		
24	LC	34	2346	Structural	Detail		
24	LC	34	2349	Structural	Detail	Other	Stairs
24	LC	34	2351	Structural	Detail	Other	Stairs
24	LC	34	2352	Structural	Detail		

24	LC	34	2353	Structural	Detail	Other	Stairs
24	LC	34	2354	Structural	Detail	Columns	Upper Floors
24	LC	34	2356	Structural	Detail		
24	LC	64	2652	Structural	Detail		
24	LC	64	2653	Structural	Detail		
24	LC	69	2744	Calcs			
24	LC	69	2746	Calcs			
24	LC	70	2749	Calcs			
24	LC	70	2752	Calcs			
24	LC	72	2791	Calcs			
24	LC	73	2796	Calcs			
24	LC	73	2801	Calcs			
24	LH	35	2369	Structural	Detail		
24	LH	35	2370	Structural	Detail		
24	LH	35	2371	Structural	Detail		
24	LH	35	2372	Non-Tower	Plan		
24	LH	35	2376	Structural	Detail		
24	LH	35	2391	Structural	Detail		
24	LH	35	2392	Structural	Detail		
24	LH	35	2393	Structural	Detail		
24	LH	48	2109				
24	LH	65	2662				
24	LW	53	2257				
24	LW	53	2258				
24	LW	53	2259-1				
24	LW	53	2259-2				
24	LW	53	2260				
24	LW	53	2261-1				
24	LW	53	2261-2				
24	LW	53	2262				
24	LW	53	2274				
24	LW	53	2275				
24	LW	53	2276				
24	LW	53	2277				
24	WP	58	2418				
24	WP	58	2419				
24	WP	58	2420				
24	WP	58	2421				
24	WP	58	2422				

24	WP	58	2423				
24	WP	58	2427				
24	WP	58	2430				
24	WP	58	2431				
24	WP	58	2432				
24	WP	58	2445				
24	WP	58	2447				
24	WP	58	2448				
24	WP	58	2451				
24	WP	67	2687				
24	WP	67	2688				
24	WP	67	2696				
24	WP	67	2697				
24	WP	67	2698				
24	WP	67	2700				
24	WP	67	2701				
24	WP	67	2705				
24	WP	67	2706				
24	WP	67	2707				
24	WP	67	2708				
24	WP	67	2709				
24	WP	67	2710				
24	WP	67	2711				
24	WP	67	2712				
24	WP	67	2713				
24	WP	67	2714				
24	WP	67	2715				
24	WP	67	2716				
24	WP	67	2718				
24	WP	67	2720	Structural	Detail		
24	WP	67	2721	Structural	Detail		
24	WP	67	2722				
24	WP	69	2743				
24	WP	70	2748				
24	WP	70	2751				
24	WP	73	2800				

APPENDIX B

Common Dutch-English Translations

DUTCH	ENGLISH
Aanvangemomenten eigen gewicht	Fixed end moments
Afwerking	Finish
Balk	Beam
Belasting	Load
Beugels	Stirrup
Eigen gewicht (belasting)	Self (weight)
Geval	Case
Is net good	OKAY
Kolom	Column
Nuttige belasting	Live load
Puntlast	Point load
Seperaties	Partitions
Staalspanning	Tension steel
Stijfgeden	Stiffness
Tekening	Drawing
Traagheidmometen	Moment of inertia
Veld	Segment
Verdelingcoeff.	Distribution coefficient
Vloer	Floor
Wand	Wall
Wapening	Reinforcement

APPENDIX C
Graphical Fire Timelines

This appendix provides the full graphical fire timelines described in Chapter 3. The timelines of the fire at the Faculty of Architecture building were constructed by studying the photo database and are presented as a series of elevation diagrams with the locations where fire was observed marked with a color overlay. The severities of the observed flames are denoted by the color of the overlay. The colors and there corresponding meanings are as follows:

Overlay Color	Observed Fire Behavior
Grey	Dense smoke
Yellow	Flame visible
Orange	Flame out of window
Red	Flame to the next floor level
Black X	Broken Window

The time that is represented is shown to the bottom left of each diagram and the photos that were used to construct the diagram and cited in the upper left hand corner of each page. These numbers correspond to the label of the photo in the database.

Reference Photo No:
162

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

WEST_ELEVATION

NORTH

t = 255 min (12:55 pm)

Reference Photo No:
163, 164, 165

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

WEST_ELEVATION

NORTH

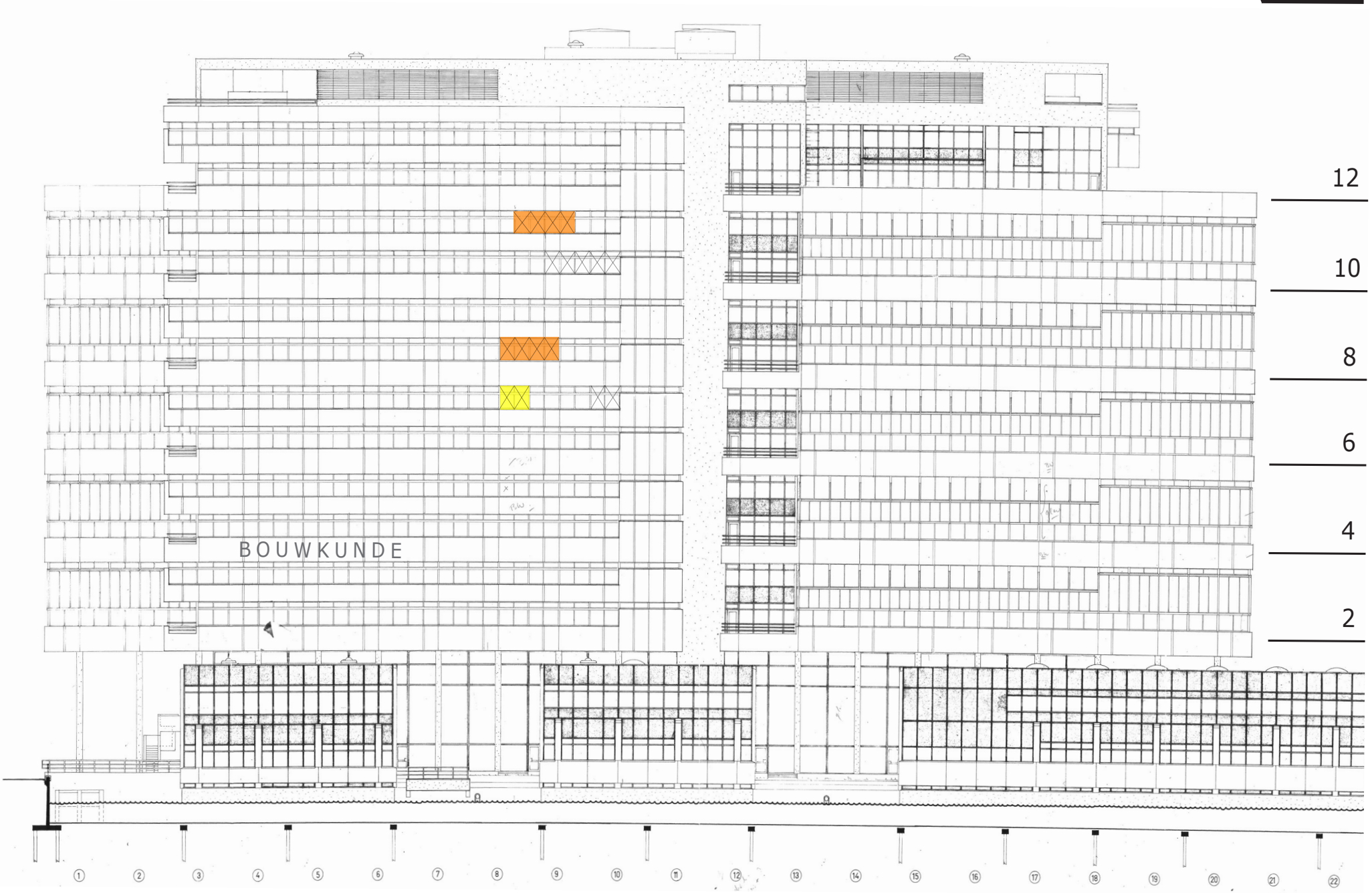
t = 257 min (12:57 pm)

Reference Photo No:
173, 175

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

WEST_ELEVATION

NORTH



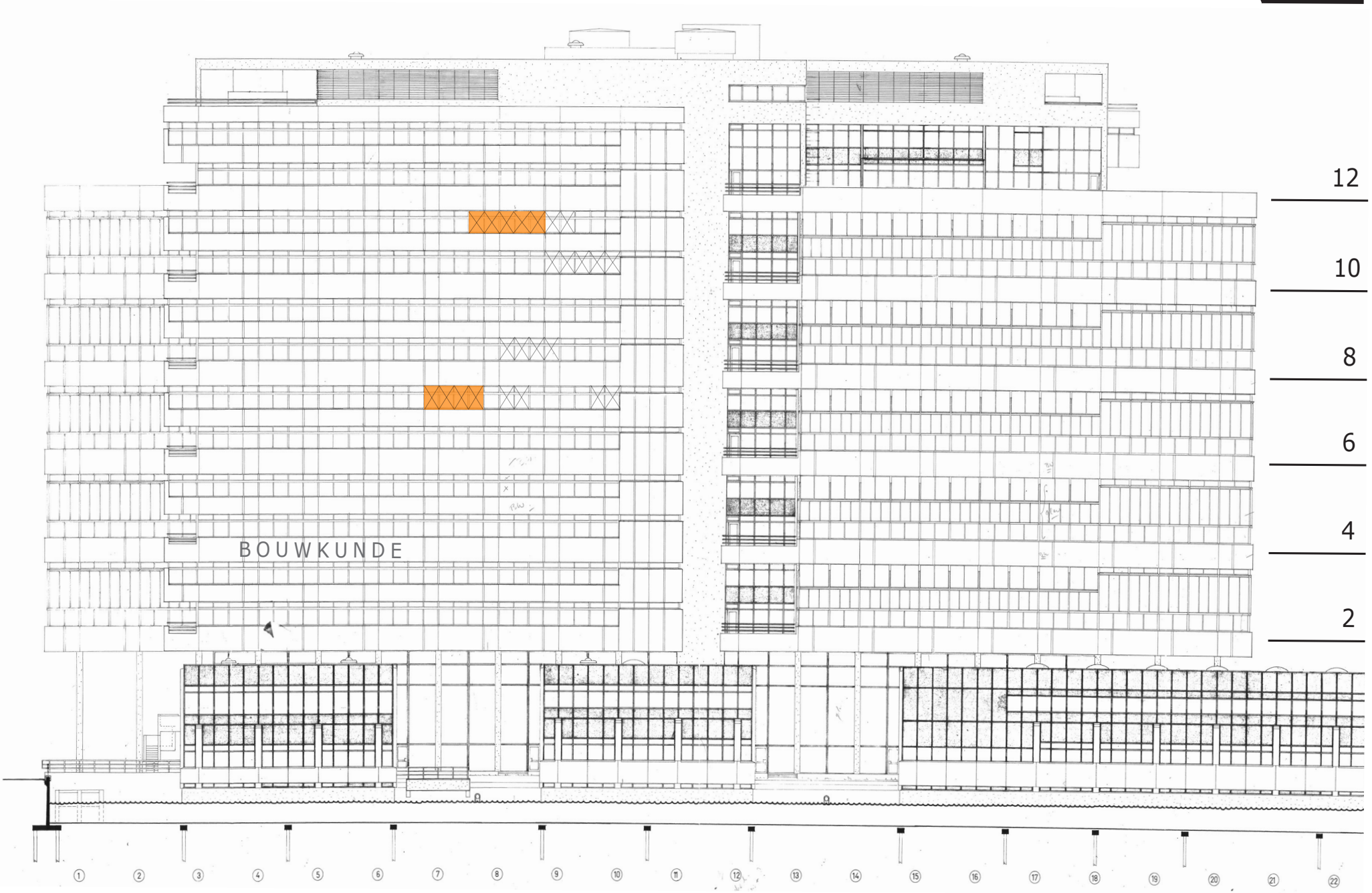
t = 297 min (1:37 pm)

Reference Photo No:
178, 179

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

WEST_ELEVATION

NORTH



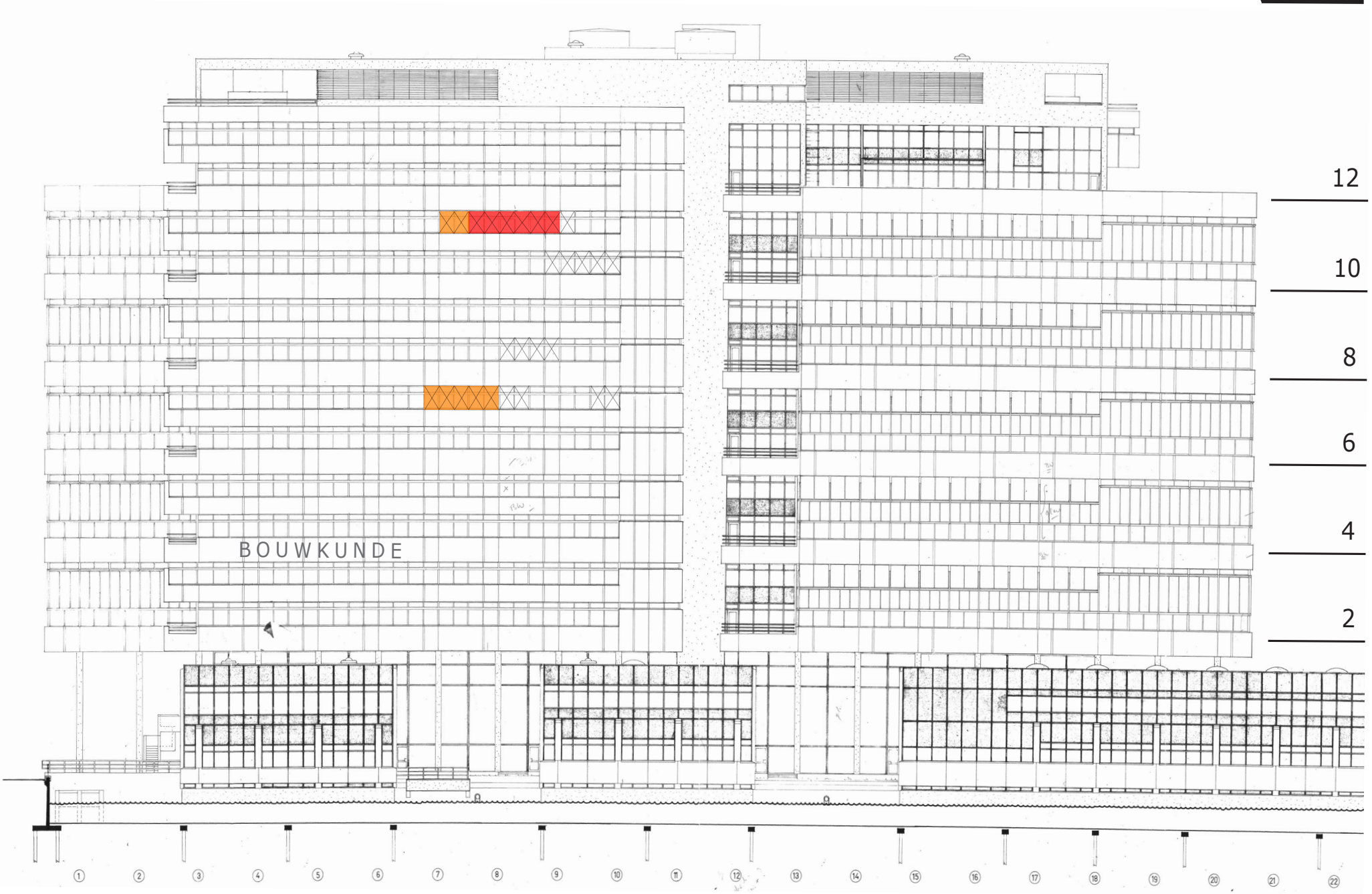
t = 299 min (1:39 pm)

Reference Photo No:
183, 184, 185, 186, 187

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

WEST_ELEVATION

NORTH



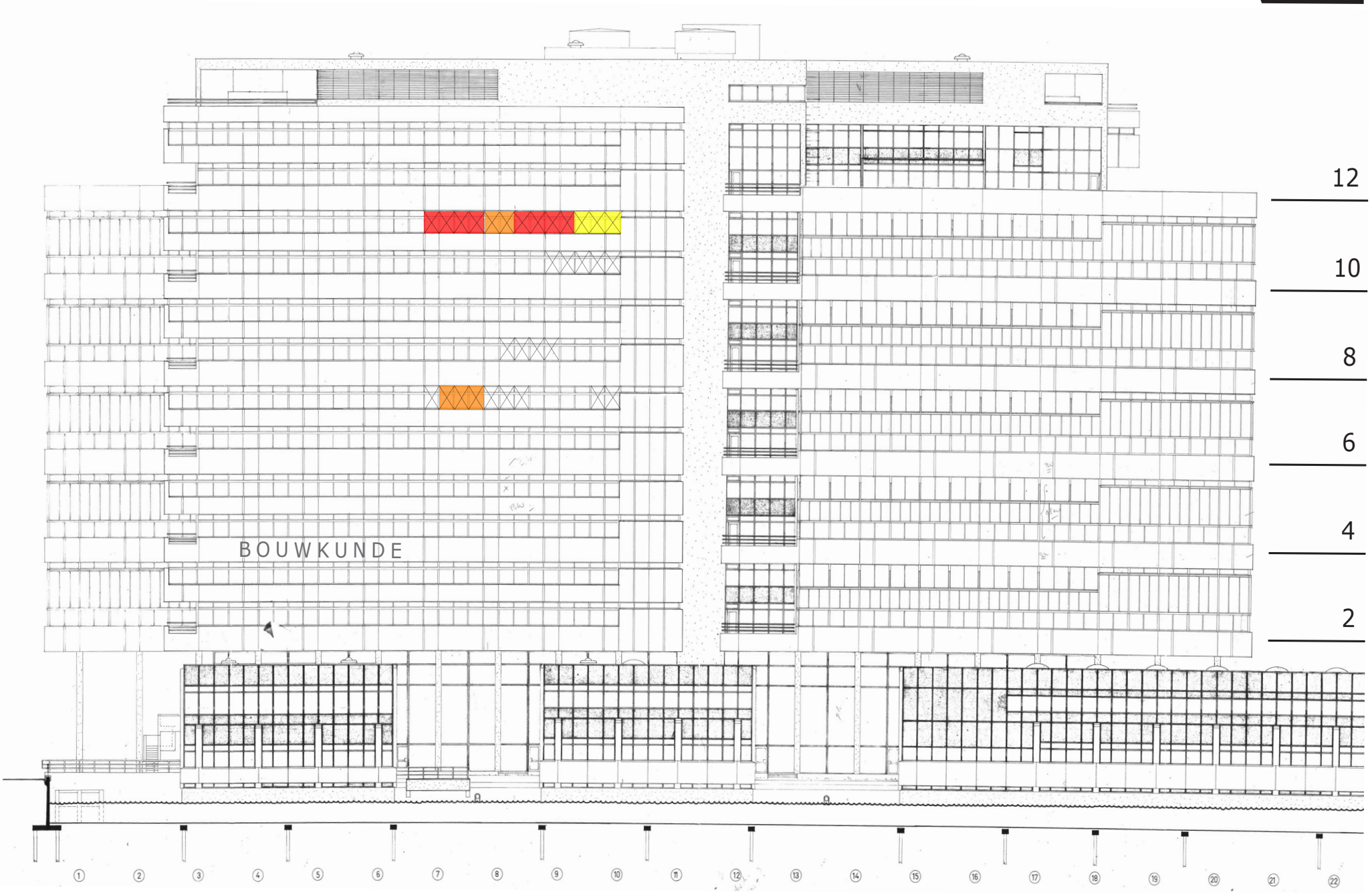
t = 302 min (1:42 pm)

Reference Photo No:
188, 189, 191, 192

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

WEST_ELEVATION

NORTH



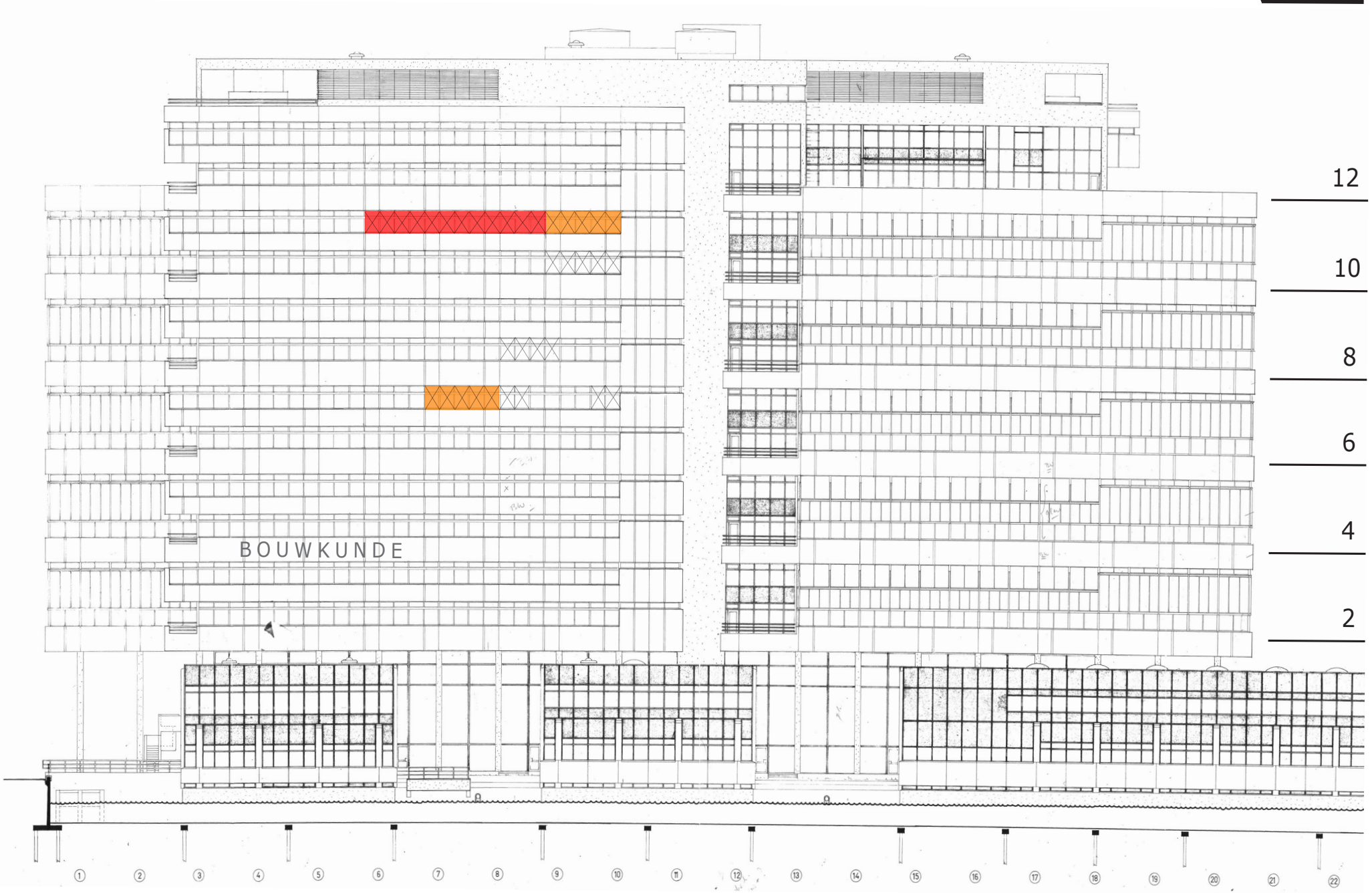
t = 303 min (1:43 pm)

Reference Photo No:
208, 209, 210

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

WEST_ELEVATION

NORTH



t = 308 min (1:48 pm)

Reference Photo No:
213, 214, 215

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

WEST_ELEVATION

NORTH

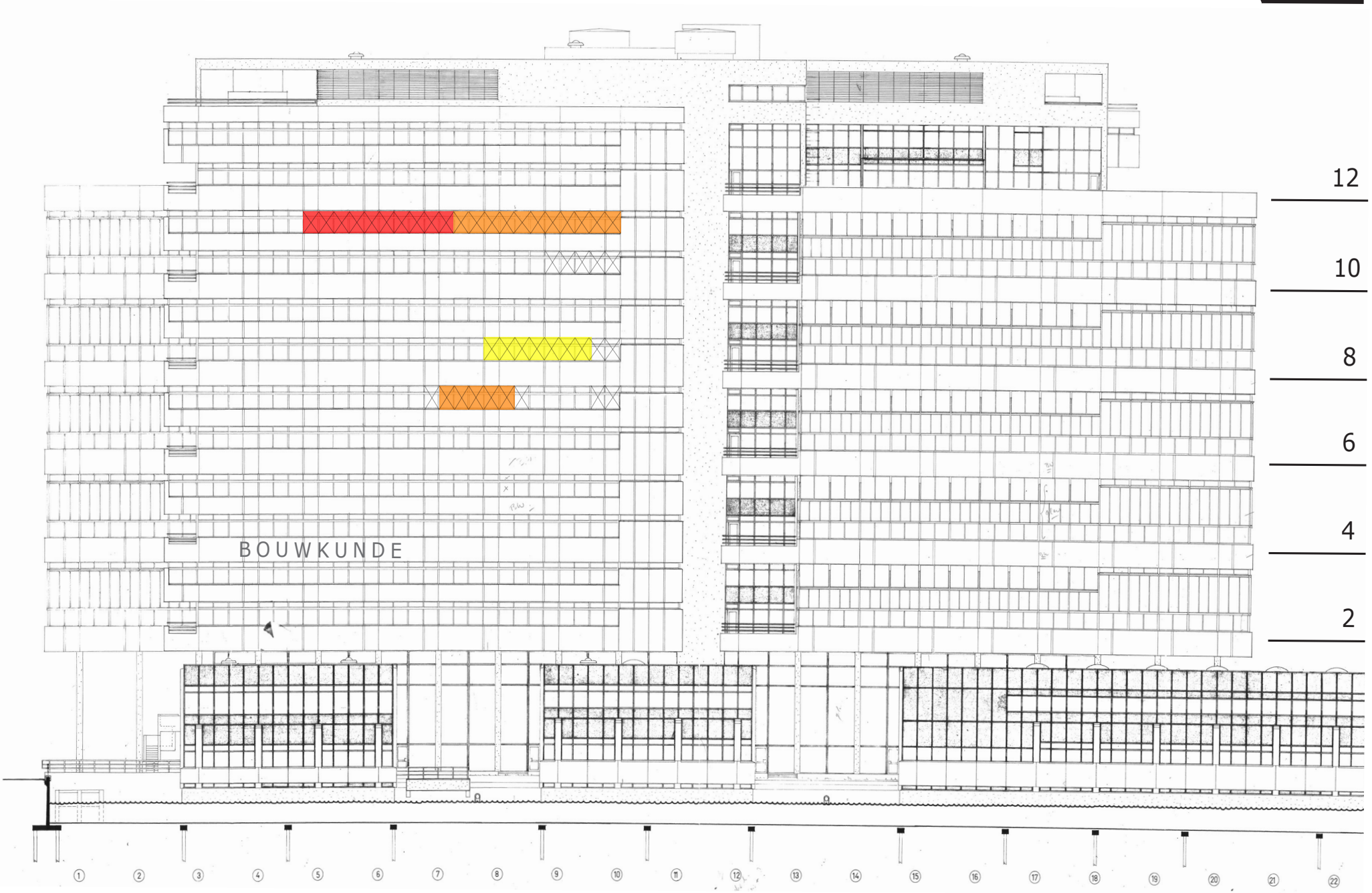
t = 316 min (1:56 pm)

Reference Photo No:
223, 224, 225

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

WEST_ELEVATION

NORTH



t = 325 min (2:05 pm)

Reference Photo No:
240, 241, 242, 243, 244, 245

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

WEST_ELEVATION

NORTH

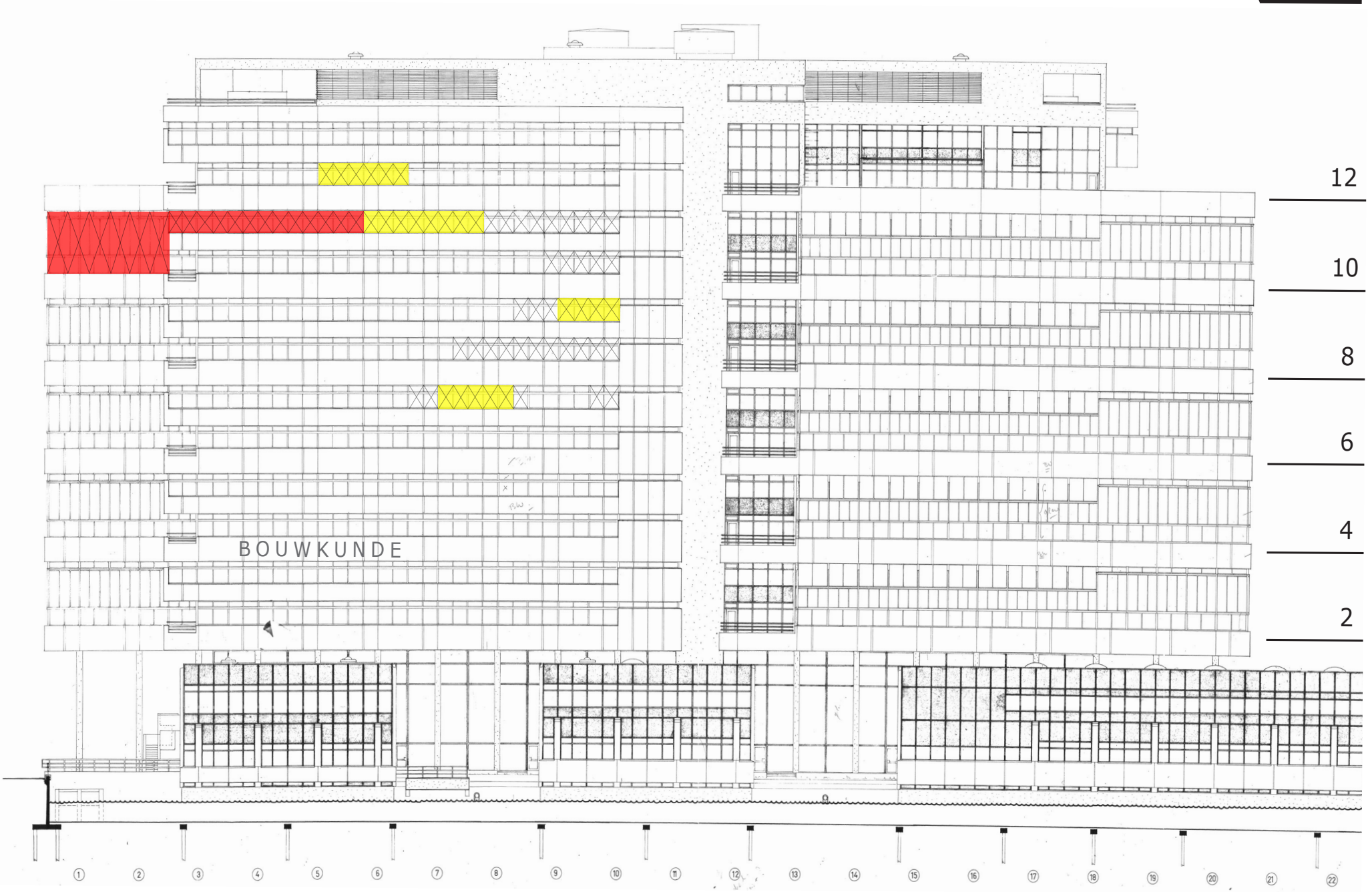
t = 329 min (2:09 pm)

Reference Photo No:
269, 270, 271, 272

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

WEST_ELEVATION

NORTH



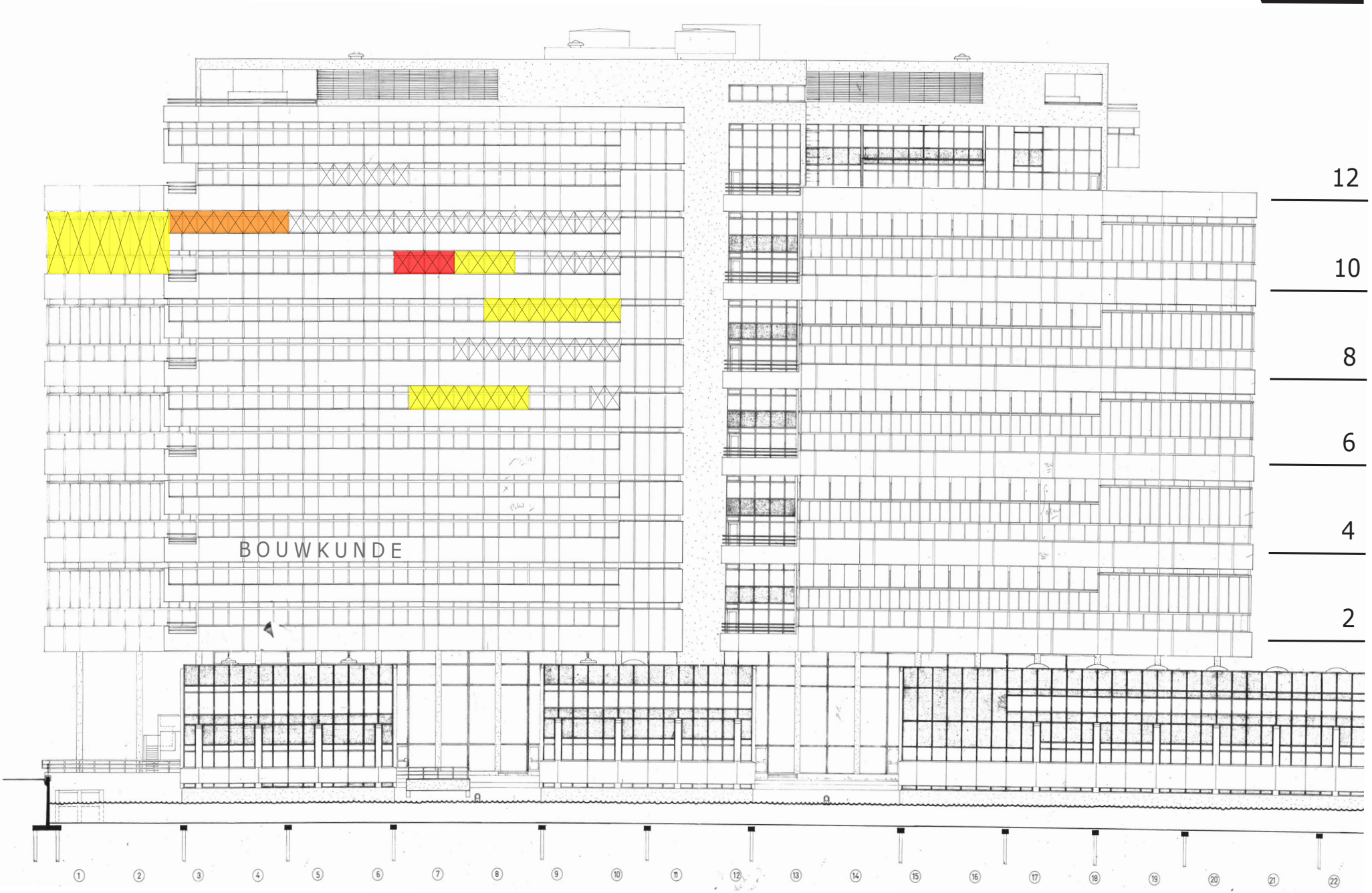
t = 333 min (2:13 pm)

Reference Photo No:
275

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

WEST_ELEVATION

NORTH



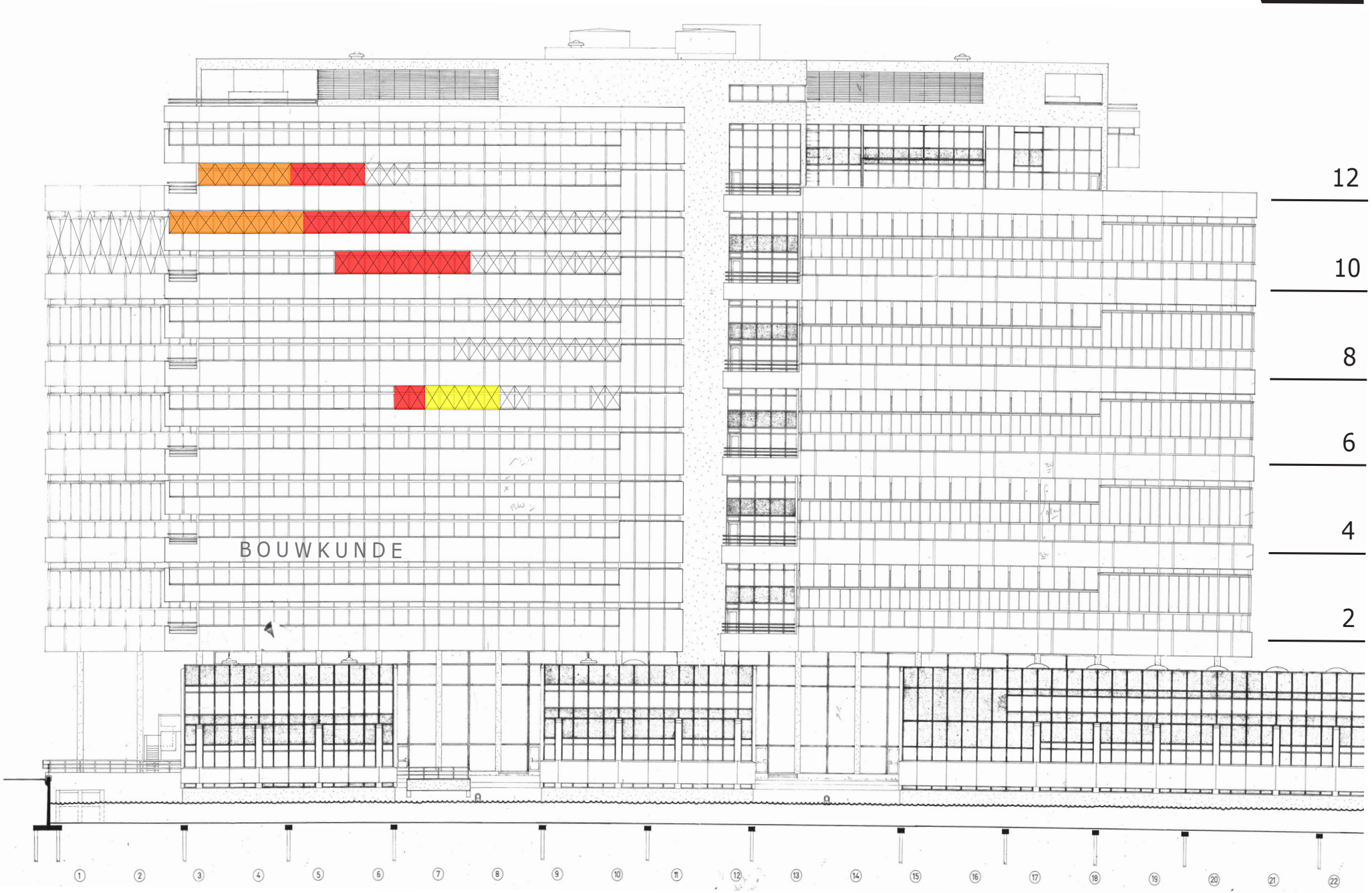
t = 340 min (2:20 pm)

Reference Photo No:
285, 286

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

WEST_ELEVATION

NORTH



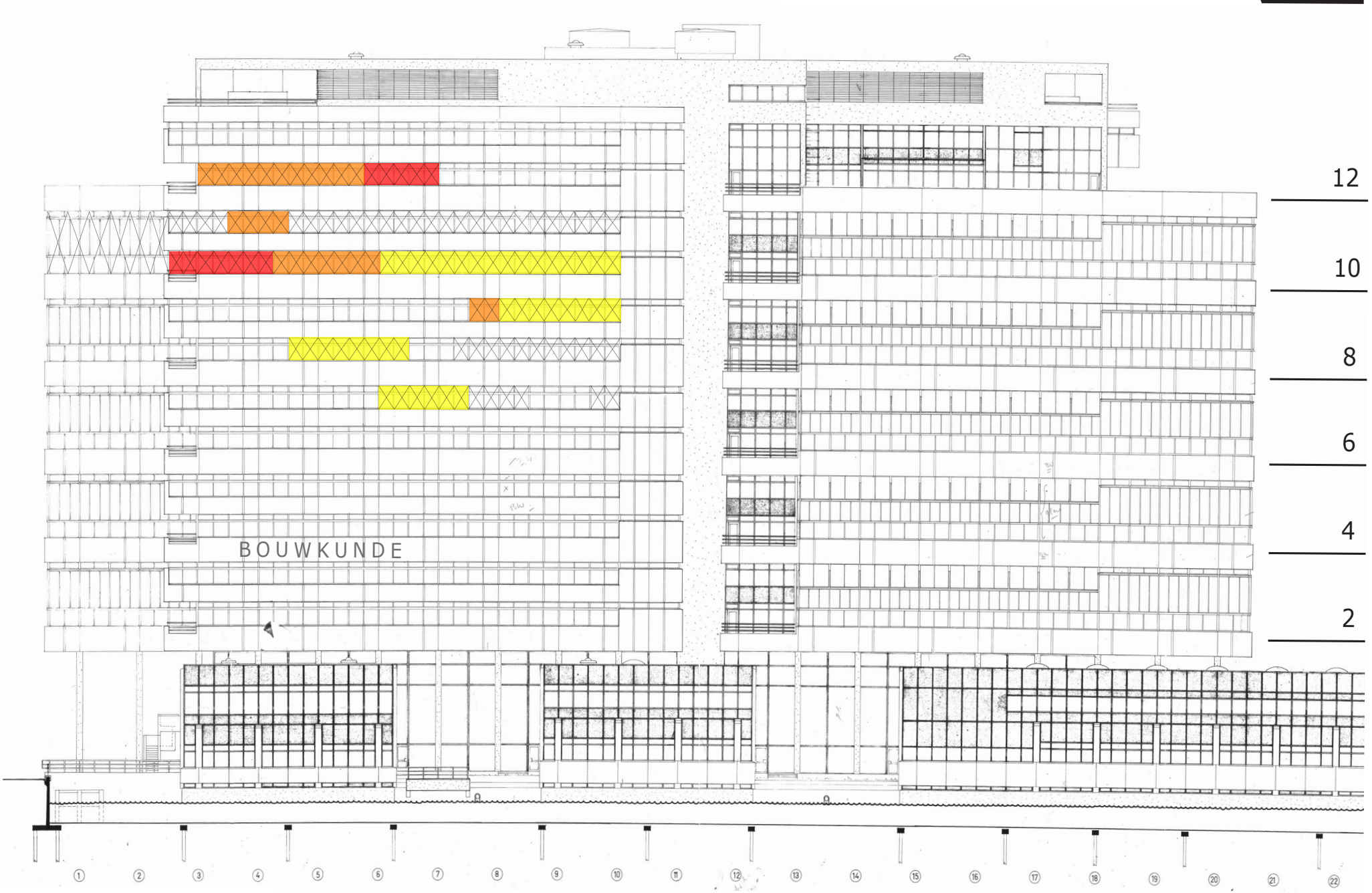
t = 347 min (2:27 pm)

Reference Photo No:
301, 302

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

WEST_ELEVATION

NORTH



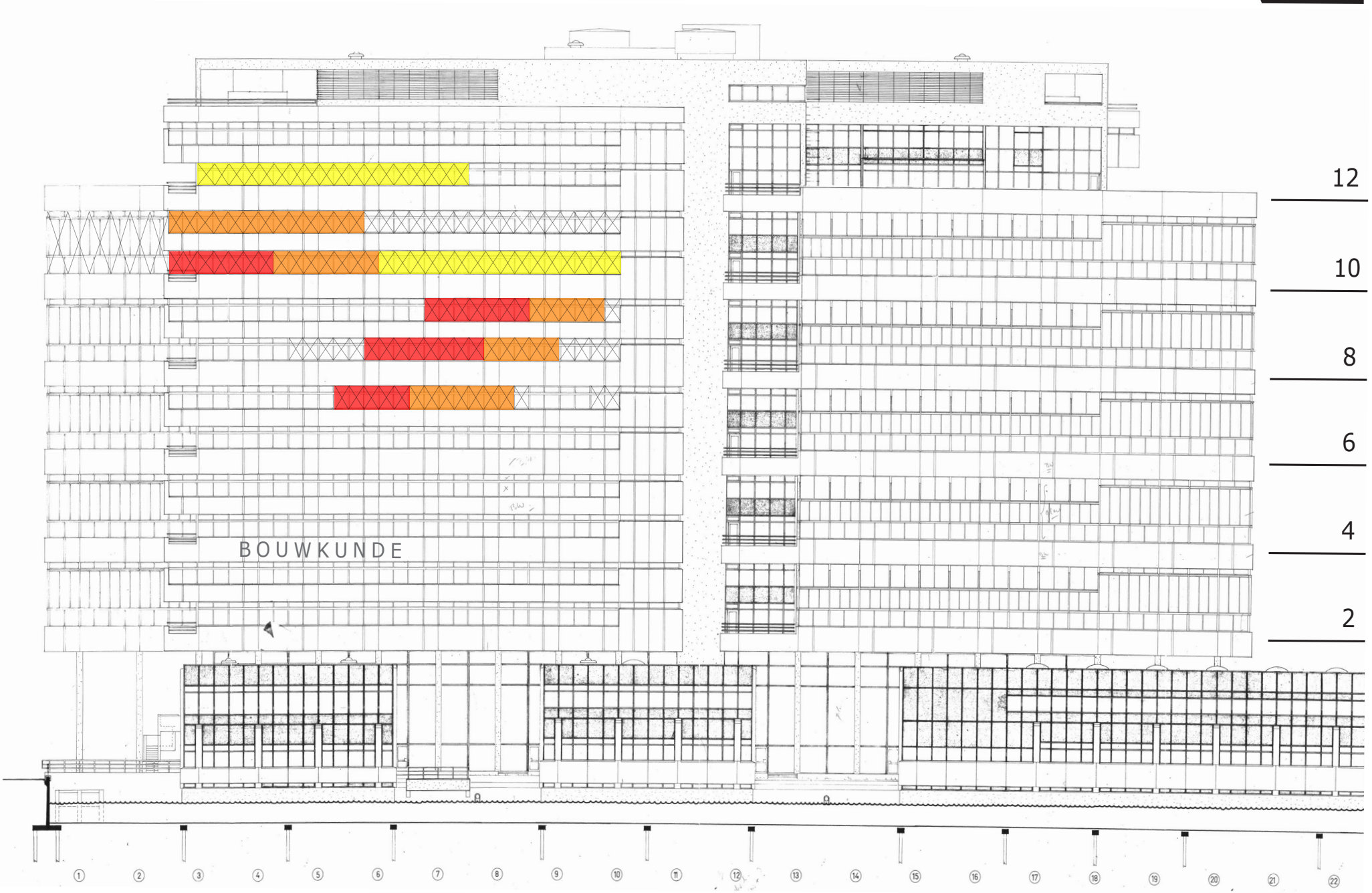
t = 355 min (2:35 pm)

Reference Photo No:
308, 309, 310

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

WEST_ELEVATION

NORTH



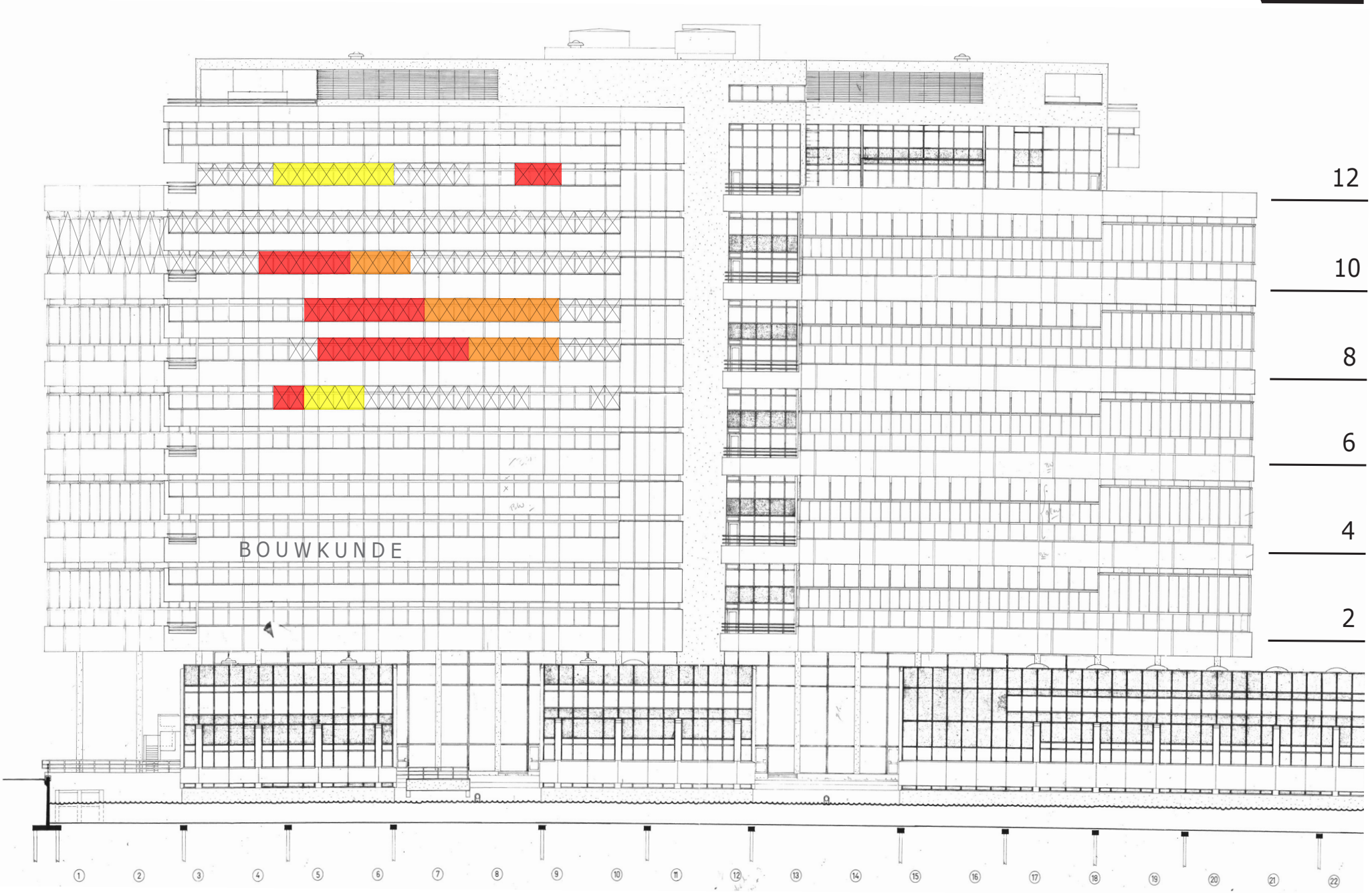
t = 366 min (2:46 pm)

Reference Photo No:
324, 325

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

WEST_ELEVATION

NORTH



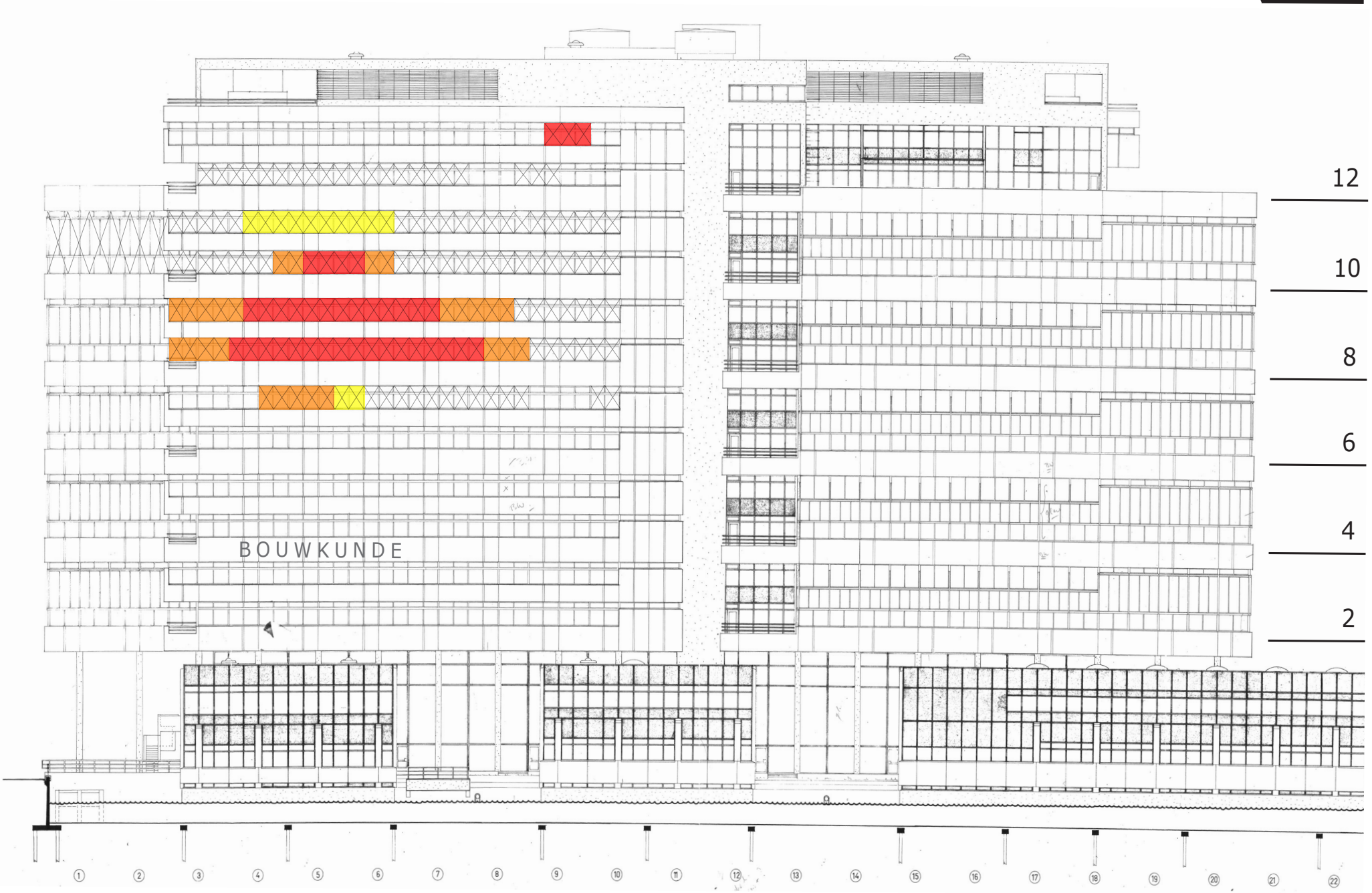
t = 373 min (2:53 pm)

Reference Photo No:
342, 343, 344, 345, 346

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

WEST_ELEVATION

NORTH



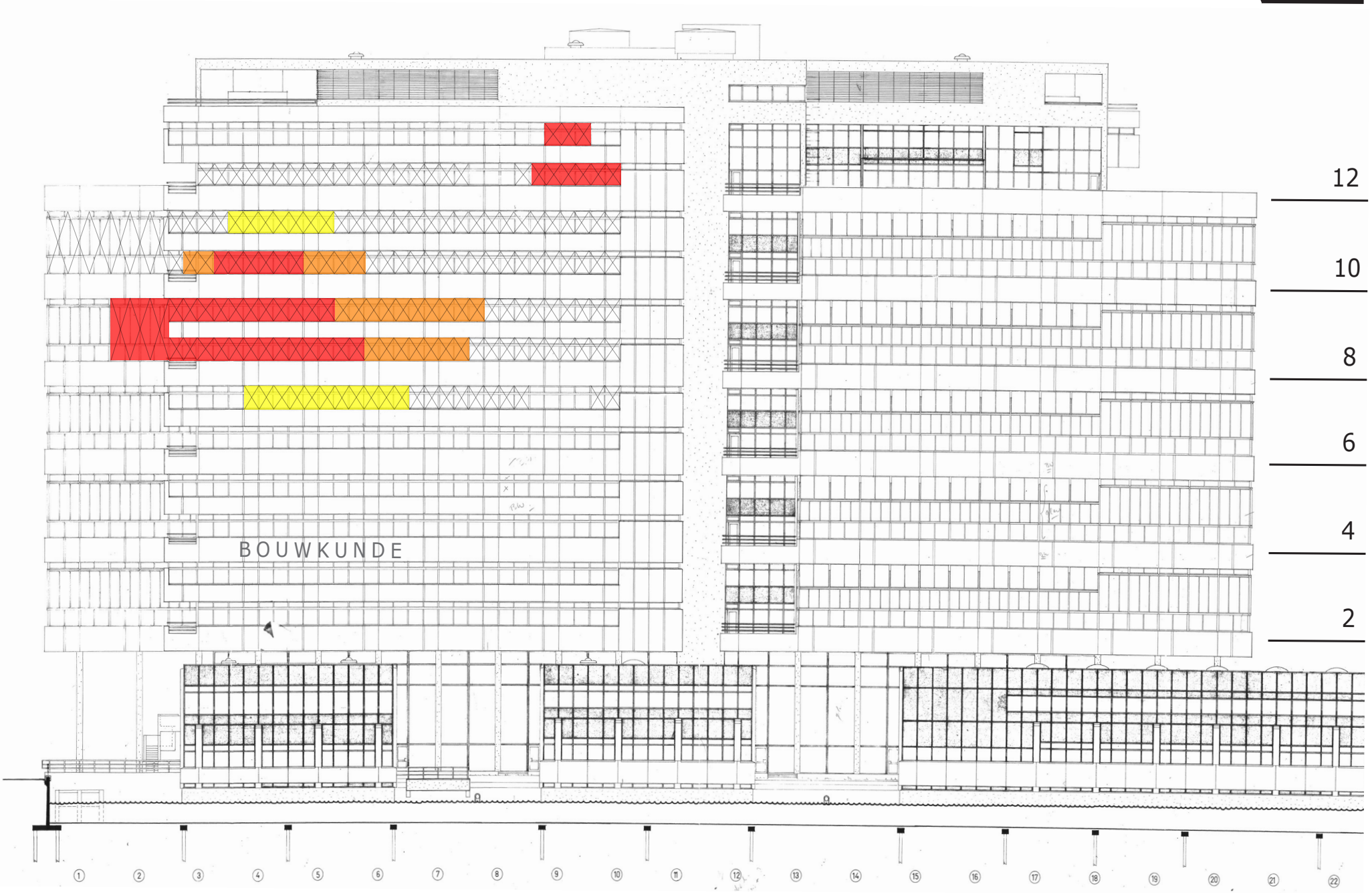
t = 381 min (3:01 pm)

Reference Photo No:
363, 364, 365, 366

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

WEST_ELEVATION

NORTH



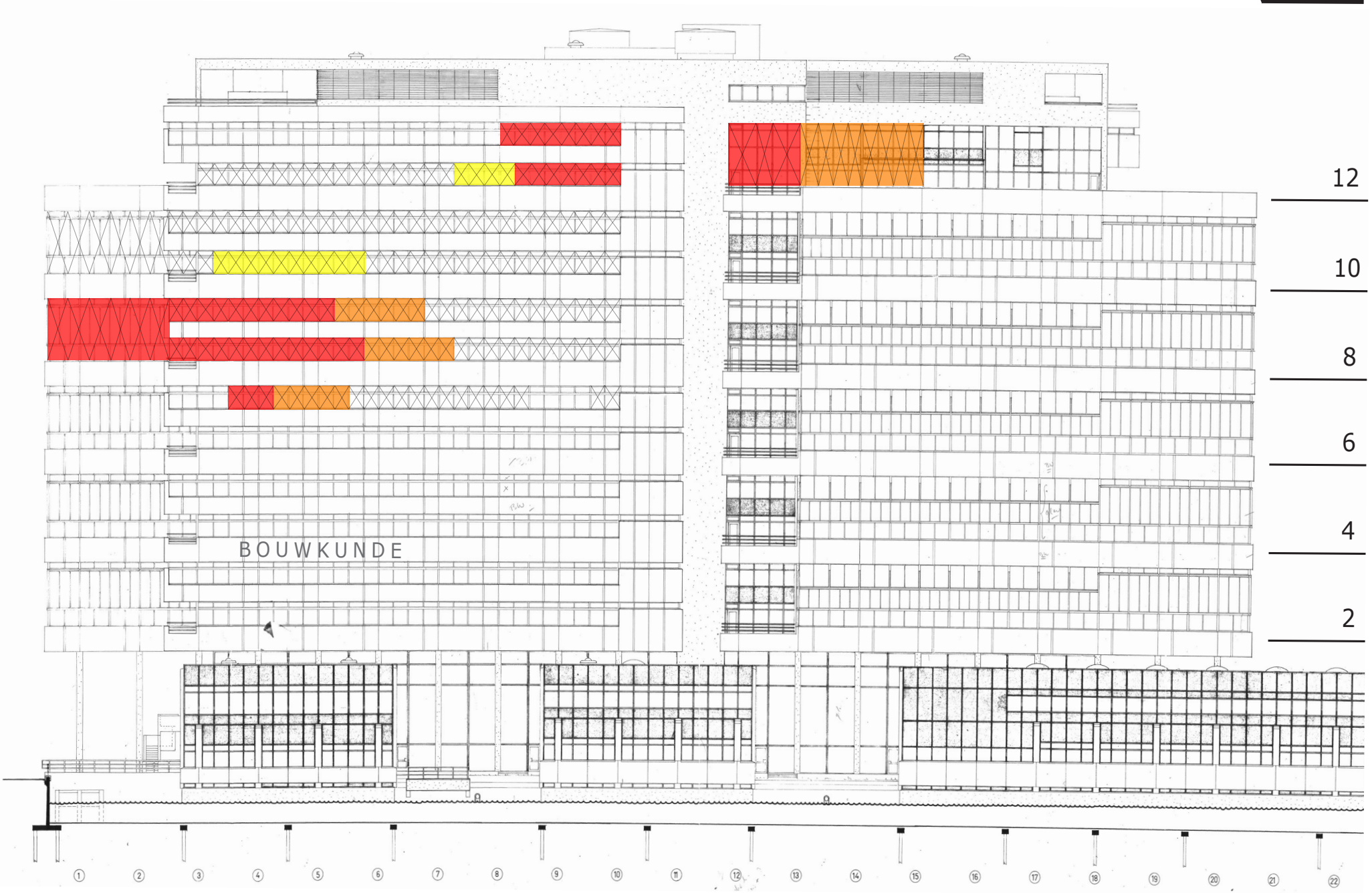
t = 385 min (3:05 pm)

Reference Photo No:
381, 382

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

WEST_ELEVATION

NORTH



t = 391 min (3:11 pm)

Reference Photo No:
391, 392

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

WEST_ELEVATION

NORTH

Collapse at t = 403 min (3:43pm)

t = 416 min (3:36 pm)

Reference Photo No:
425

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window
= Collapsed

WEST_ELEVATION

NORTH

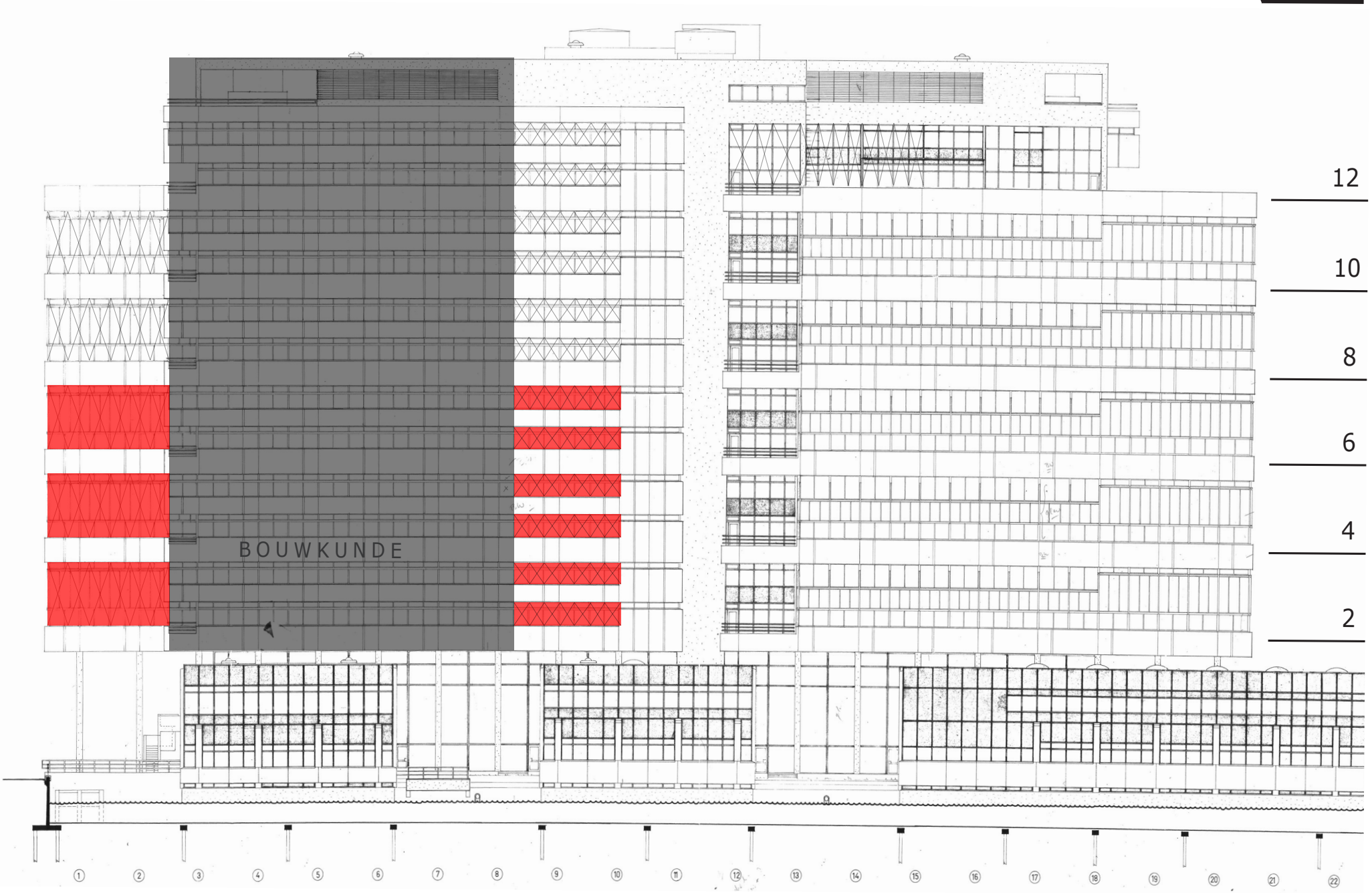
t = 430 min (3:50 pm)

Reference Photo No:
457, 458, 459, 460, 461

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window
= Collapsed

WEST_ELEVATION

NORTH



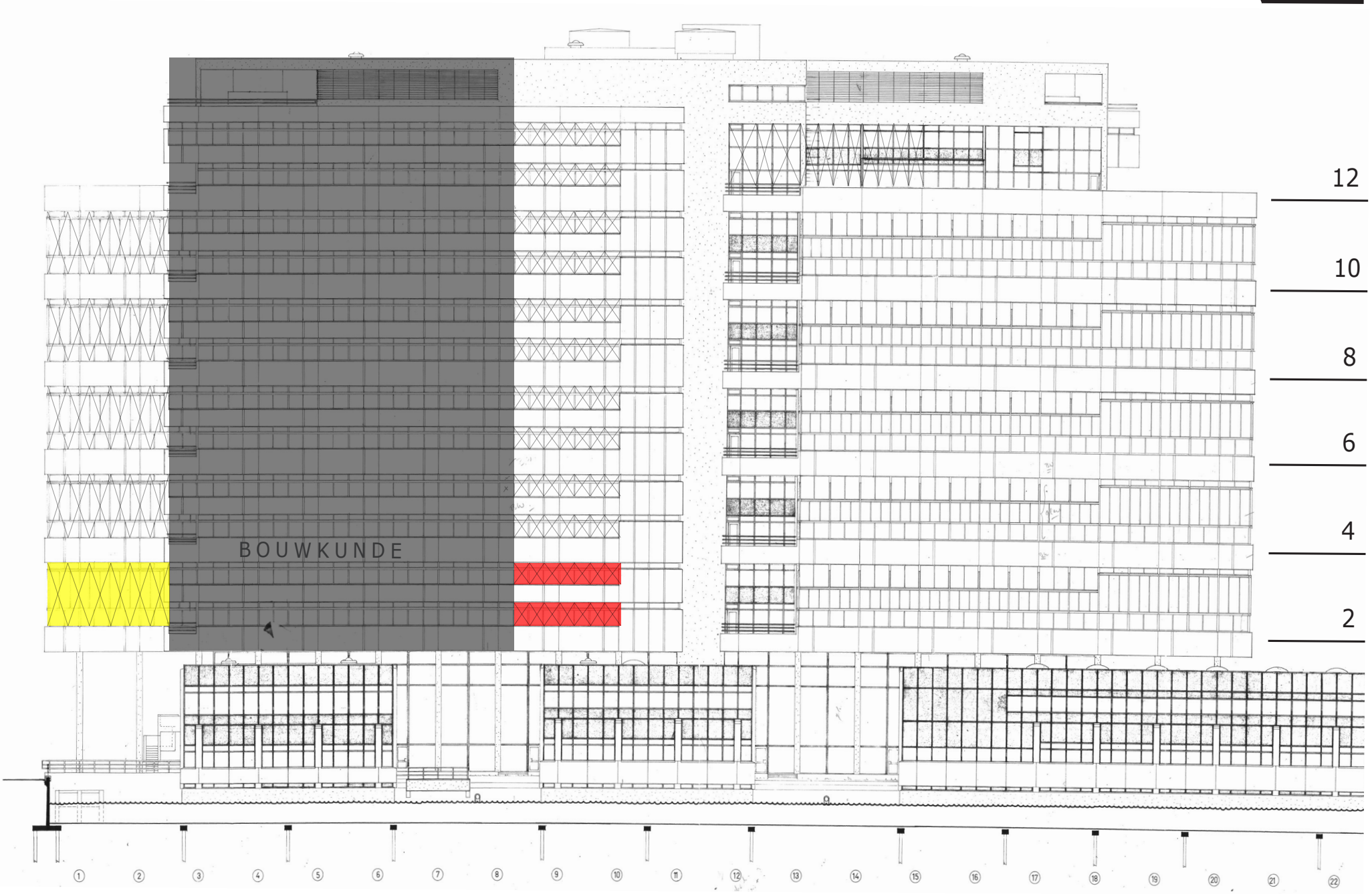
t = 438 min (3:58 pm)

Reference Photo No:
494, 495, 496

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window
= Collapsed

WEST_ELEVATION

NORTH



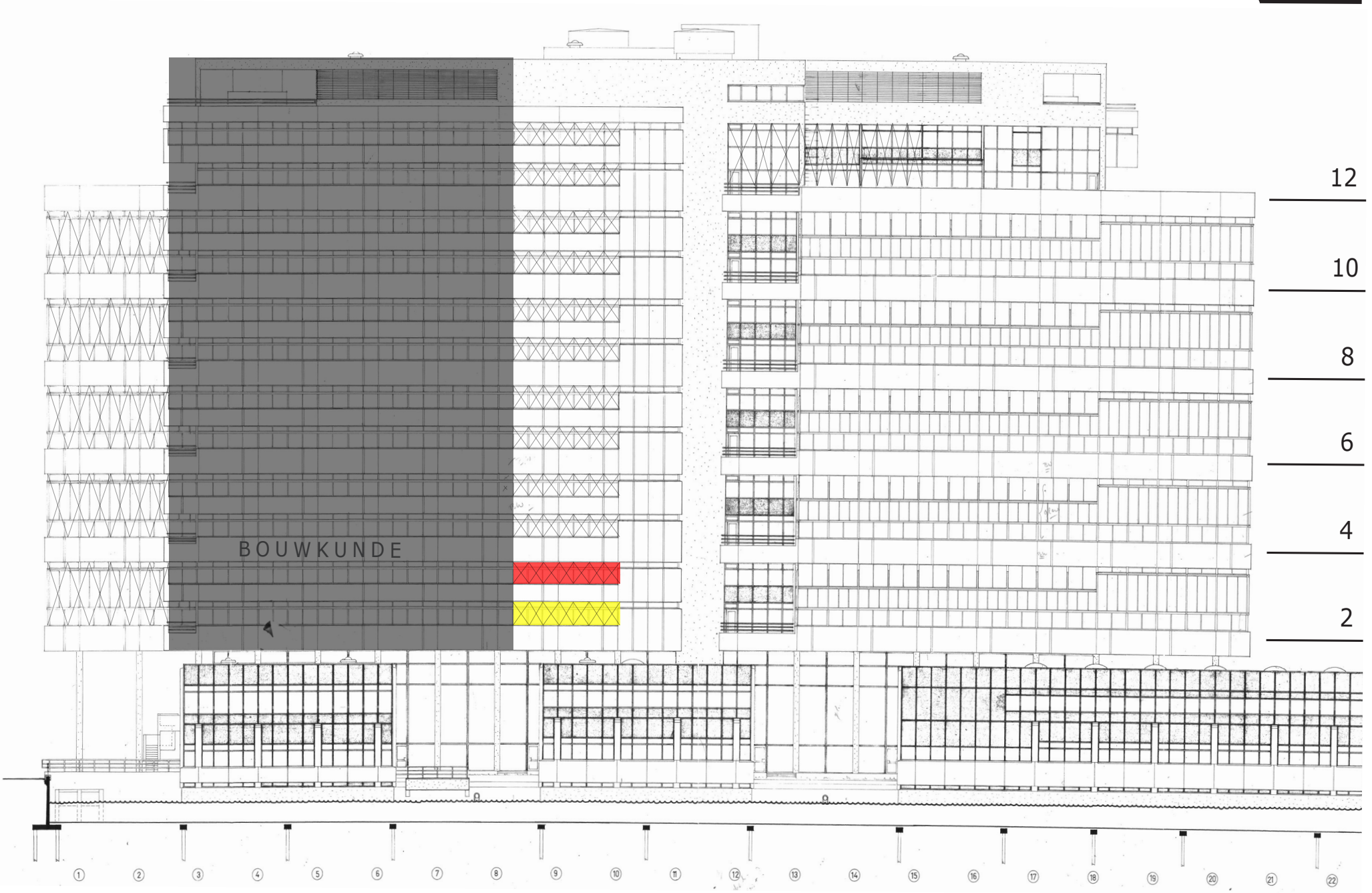
t = 454 min (4:14 pm)

Reference Photo No:
517

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window
= Collapsed

WEST_ELEVATION

NORTH



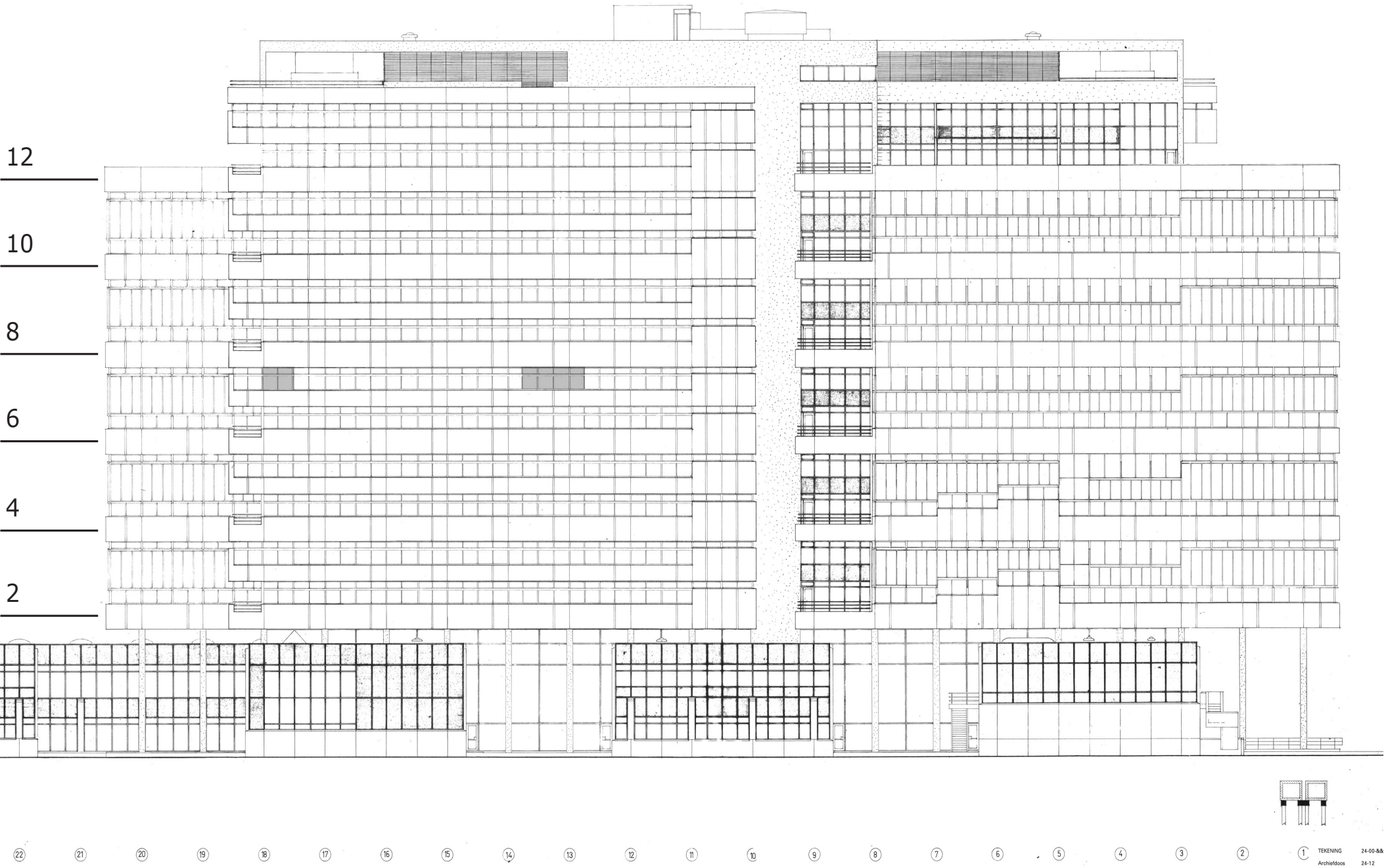
t = 478 min (4:38 pm)

Reference Photo No:
55, 56

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

EAST_ELEVATION

NORTH



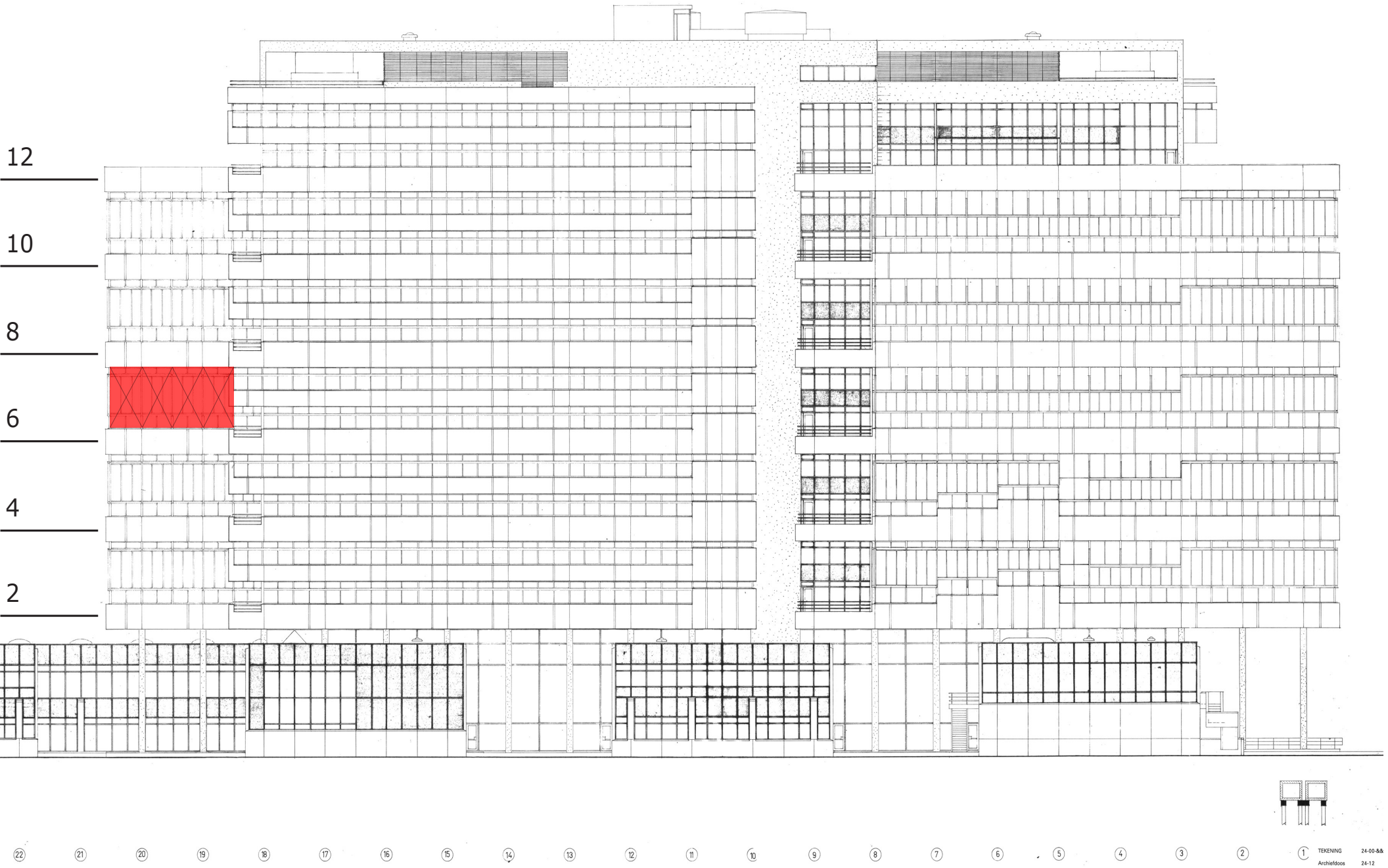
t = 004 min (8:44 am)

Reference Photo No:
70, 71, 72

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

EAST_ELEVATION

NORTH



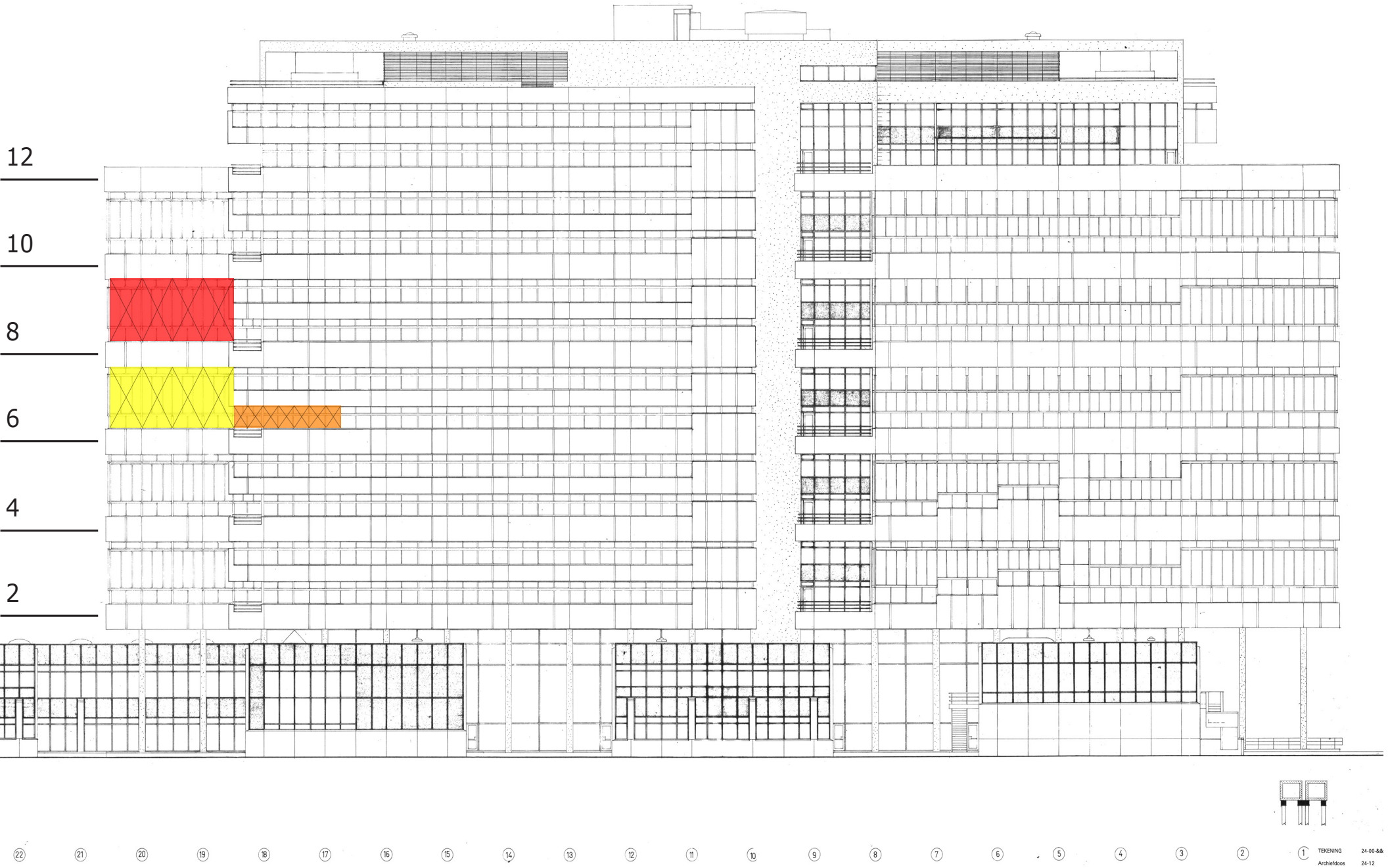
t = 021 min (9:01 am)

Reference Photo No:
77, 78, 79

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

EAST_ELEVATION

NORTH



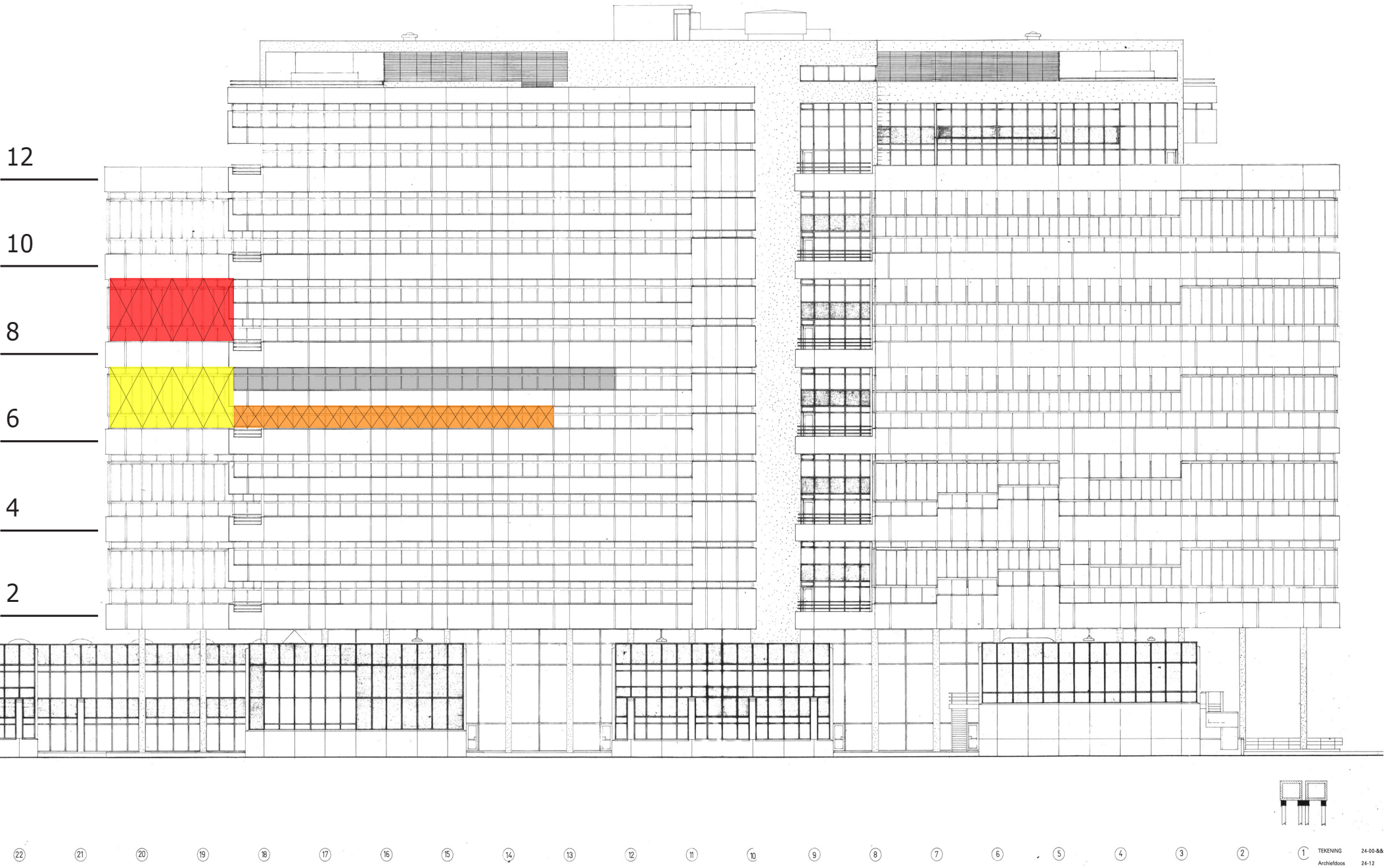
t = 033 min (9:13 am)

Reference Photo No:
84, 85

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

EAST_ELEVATION

NORTH



t = 038 min (9:18 am)

Reference Photo No:
89

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

EAST_ELEVATION

NORTH

t = 043 min (9:23 am)

Reference Photo No:
91, 92

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

EAST_ELEVATION

NORTH

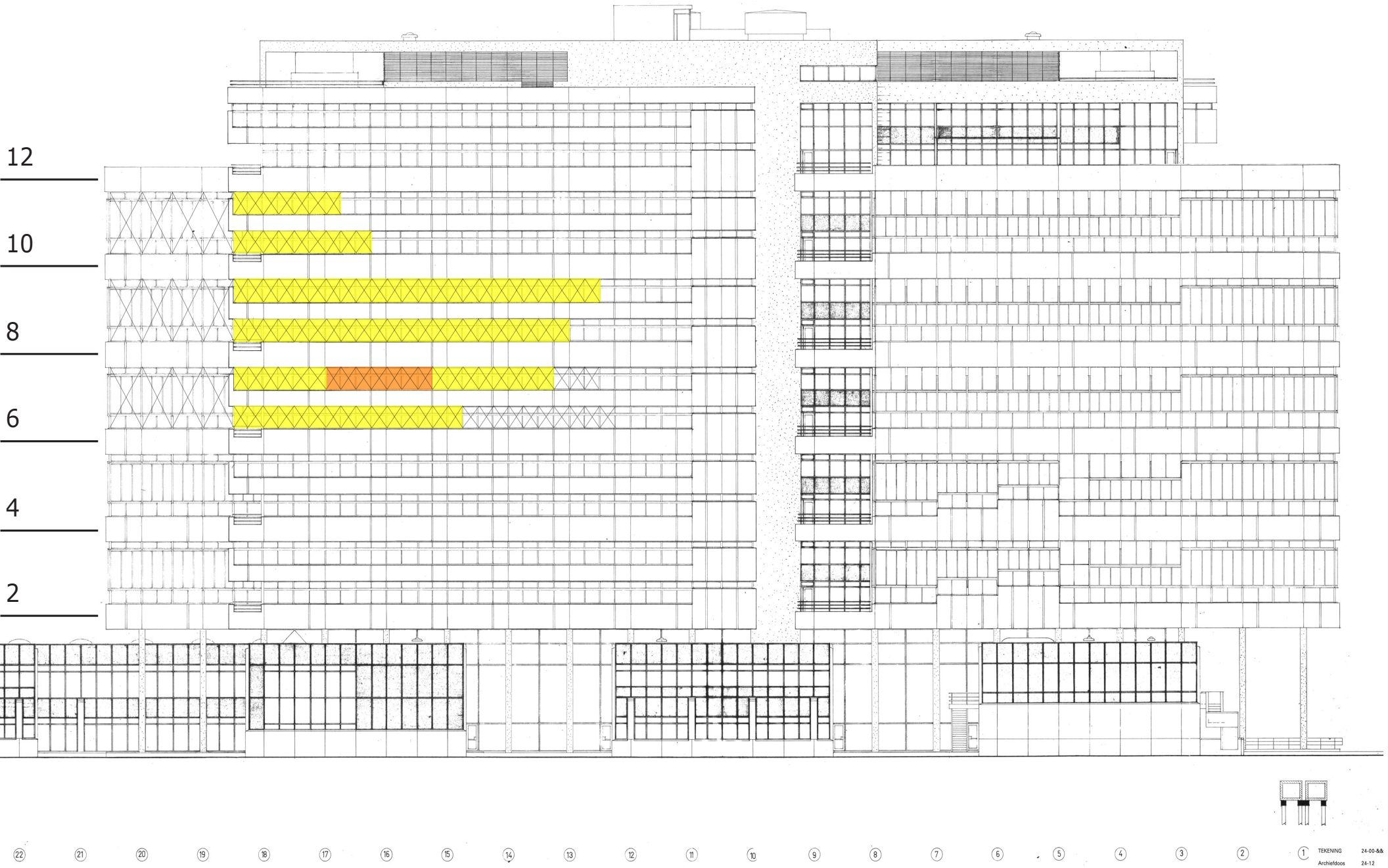
t = 046 min (9:26 am)

Reference Photo No:
97

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

EAST_ELEVATION

NORTH



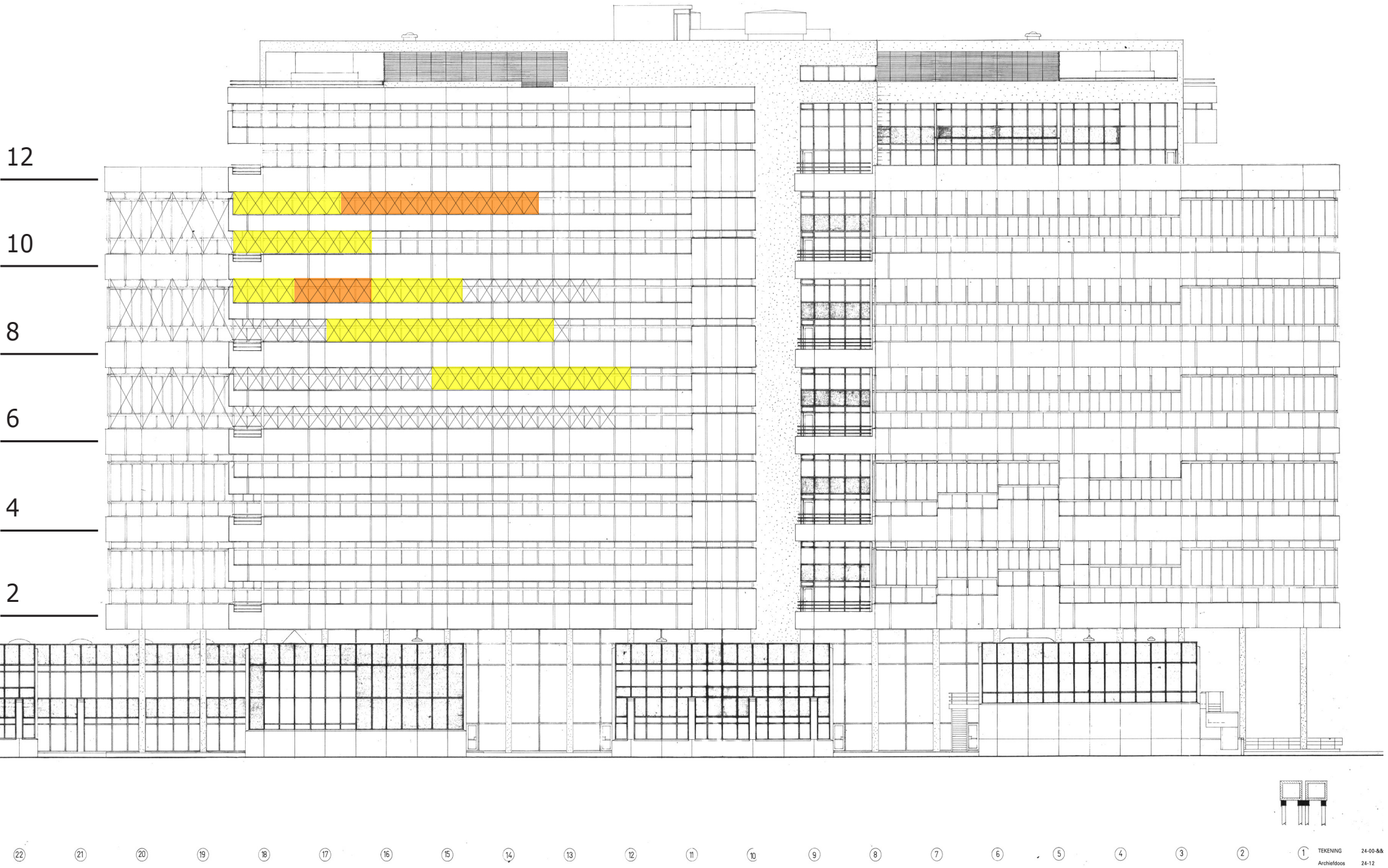
t = 085 min (10:05 am)

Reference Photo No:
98, 99

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

EAST_ELEVATION

NORTH



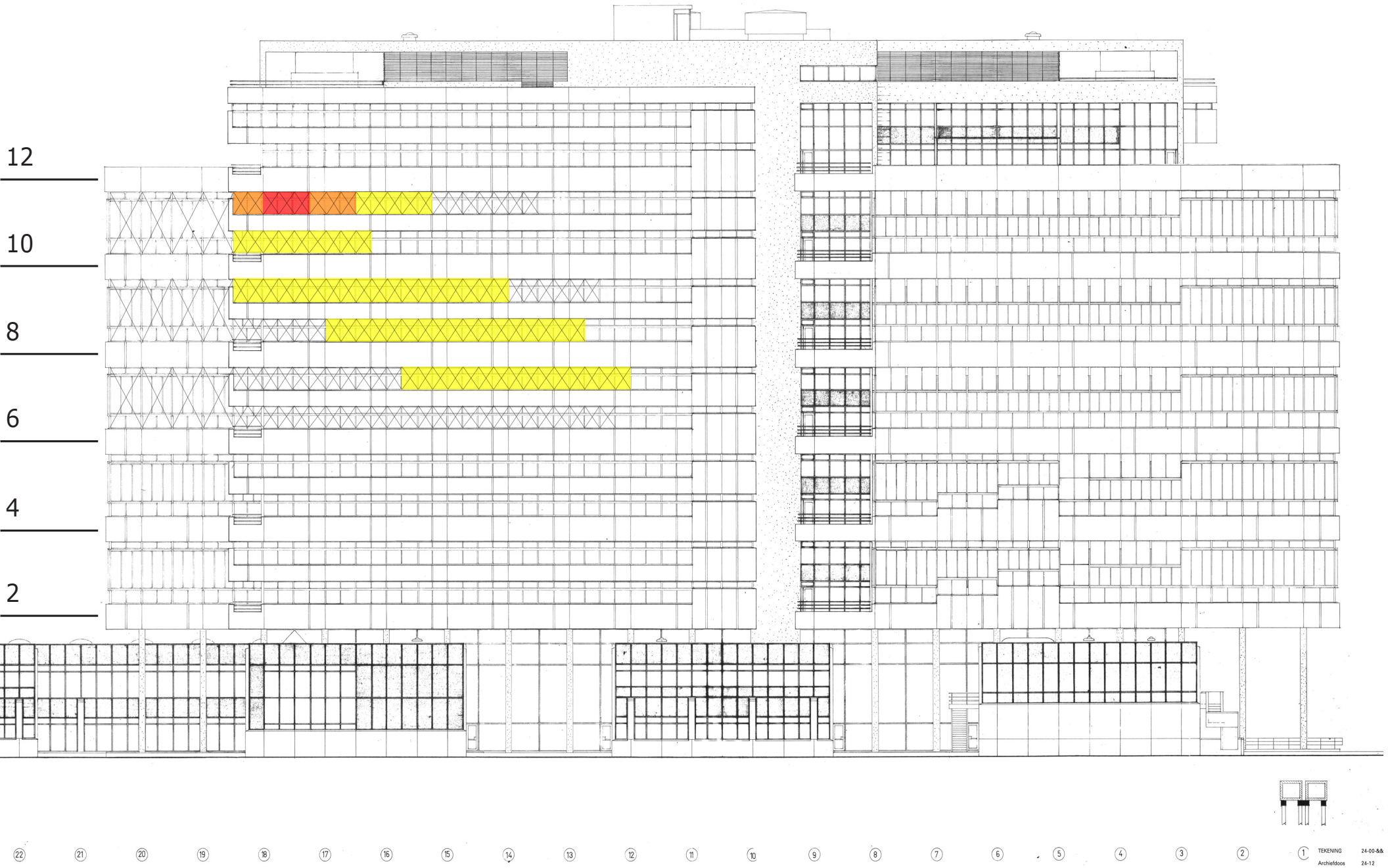
t = 132 min (10:52 am)

Reference Photo No:
100, 101, 102, 103

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

EAST_ELEVATION

NORTH



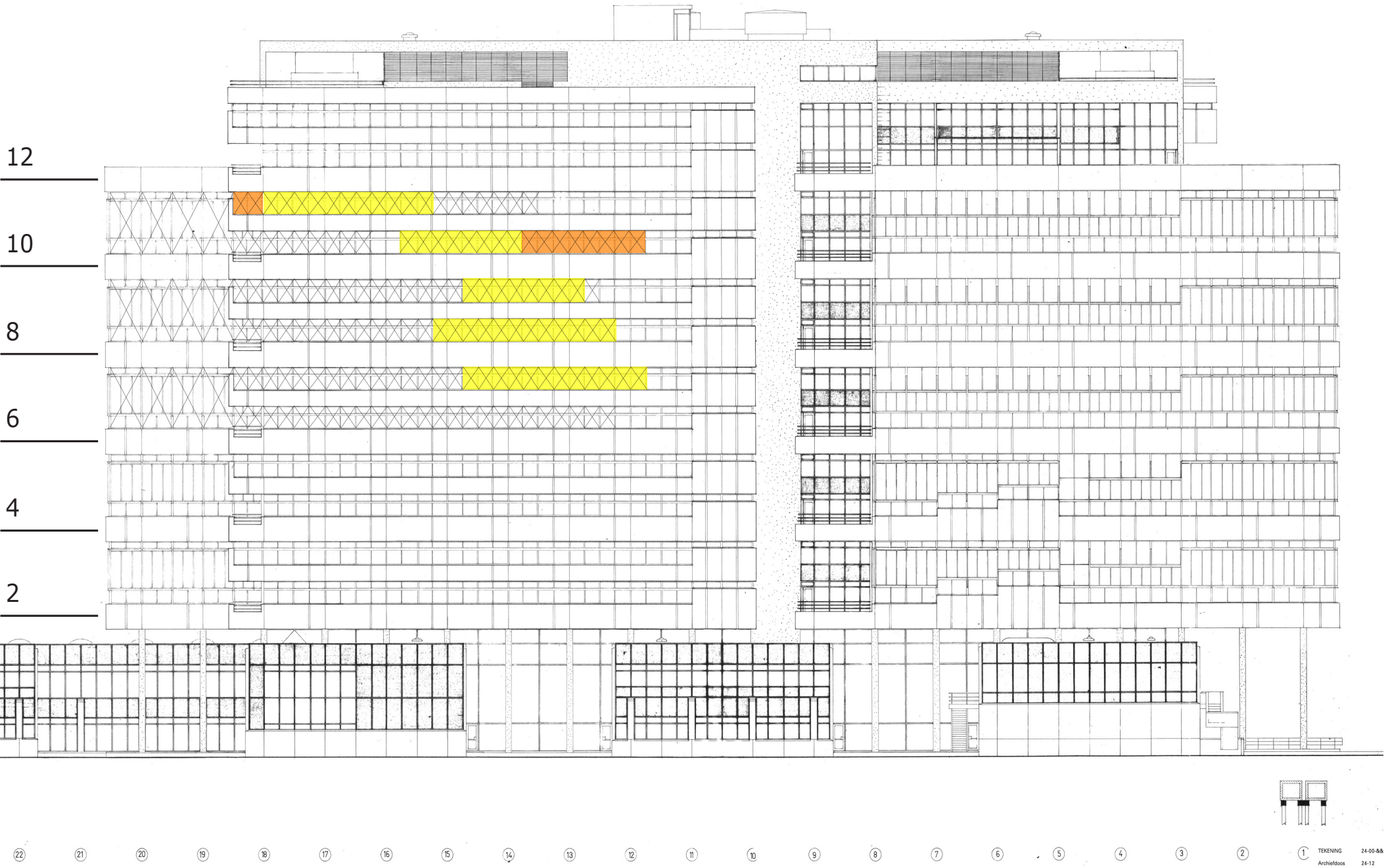
t = 147 min (11:07 am)

Reference Photo No:
113, 114

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

EAST_ELEVATION

NORTH



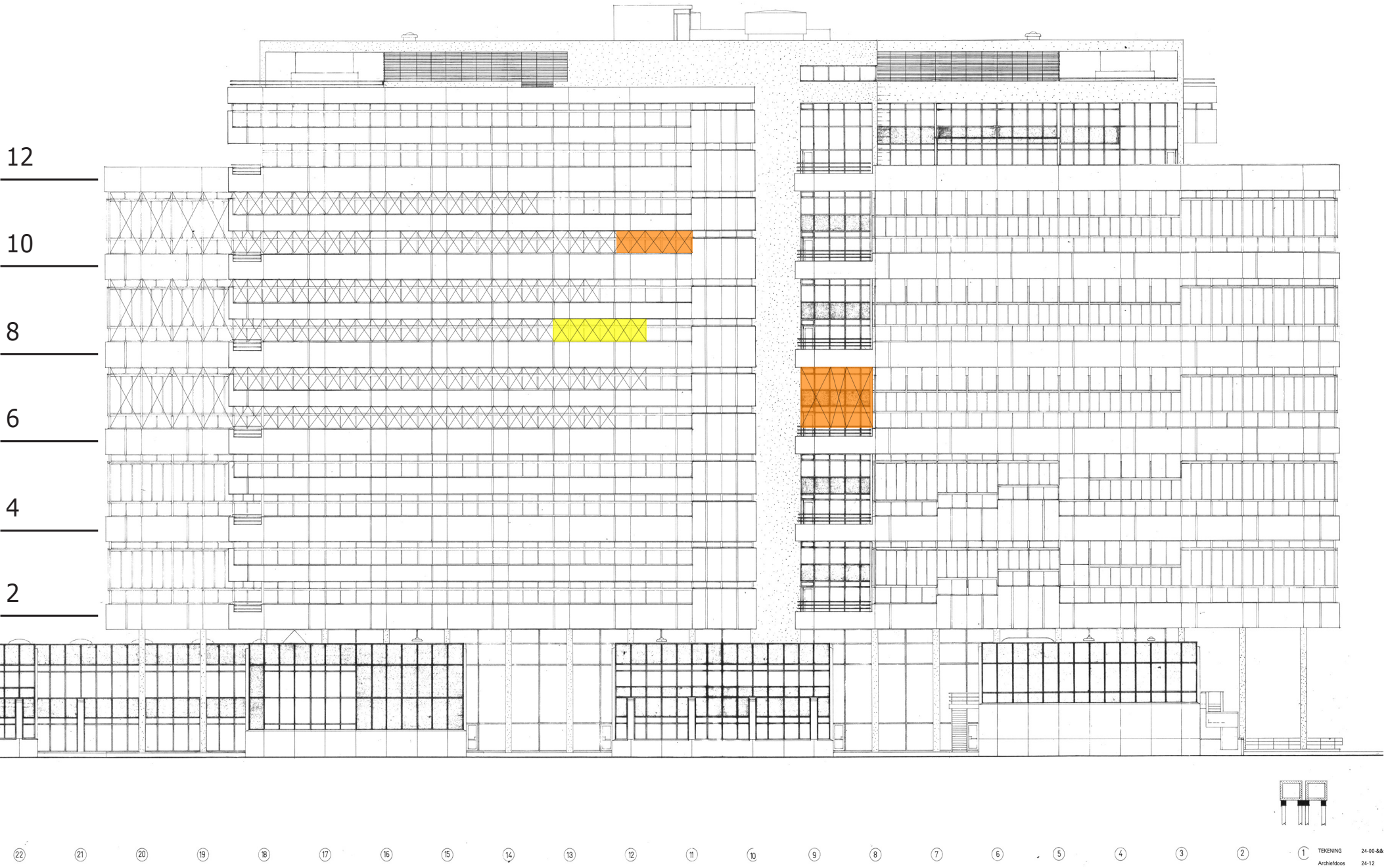
t = 163 min (11:23 am)

Reference Photo No:
119, 120, 121

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

EAST_ELEVATION

NORTH



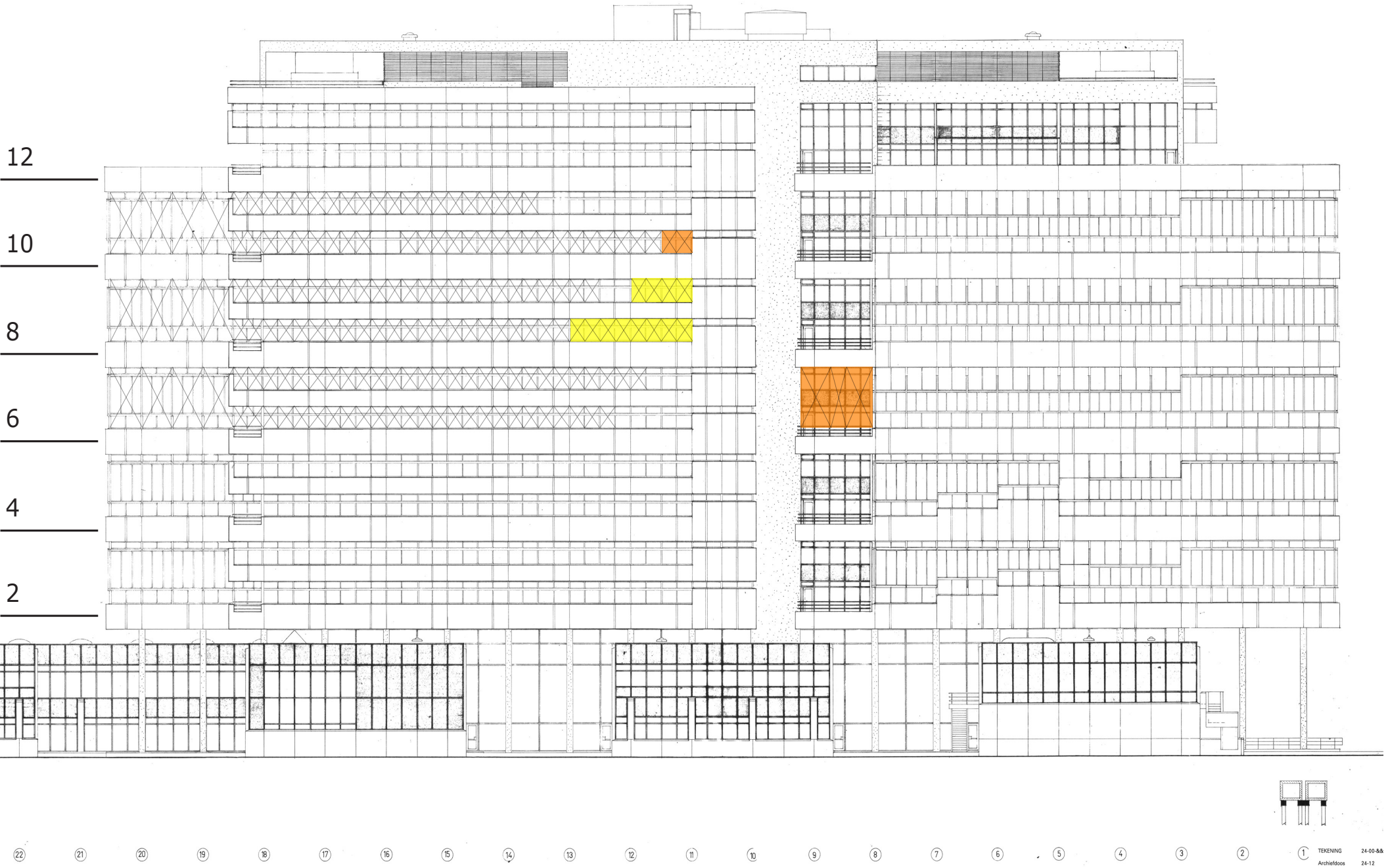
t = 182 min (11:42 am)

Reference Photo No:
145

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

EAST_ELEVATION

NORTH



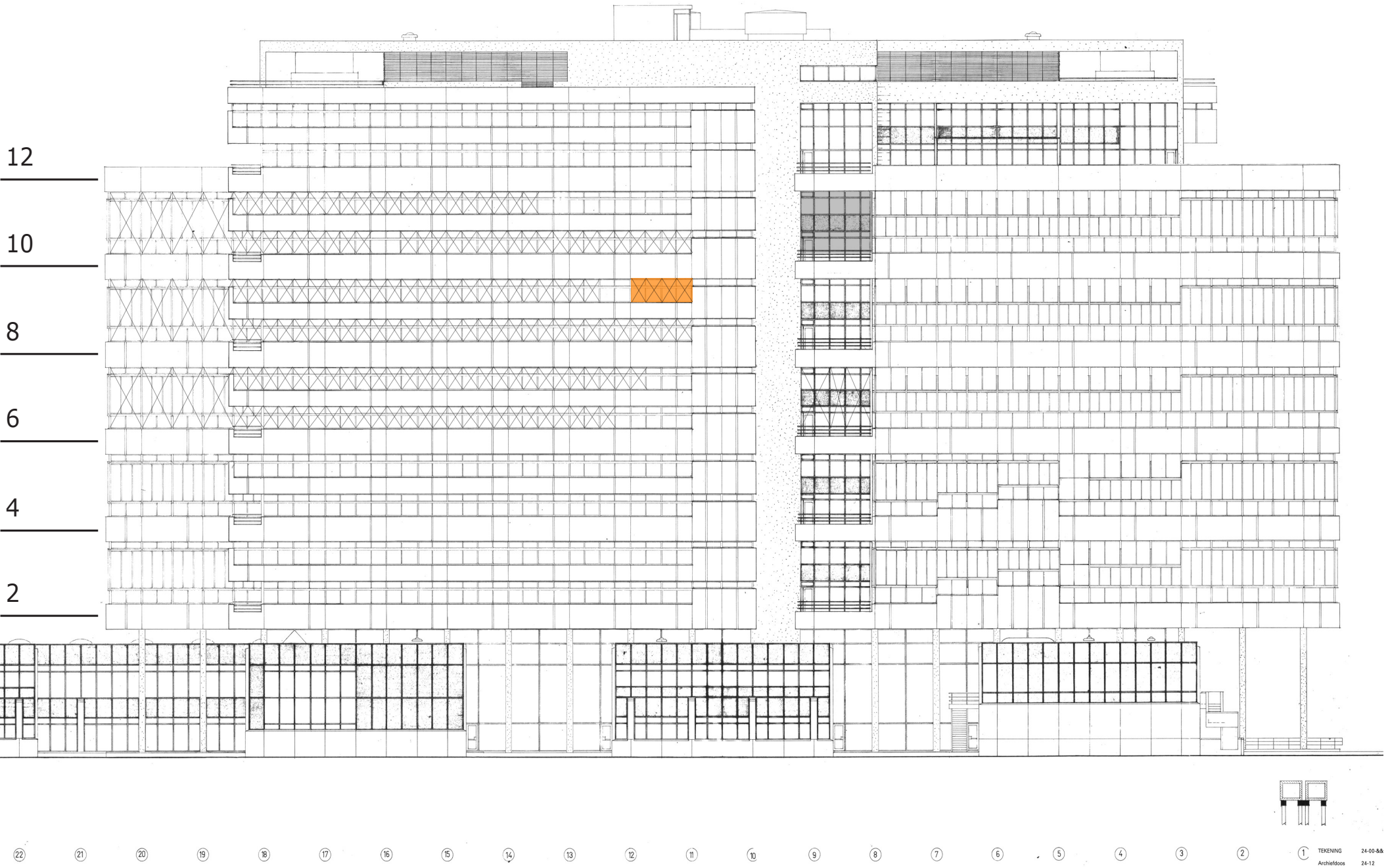
t = 206 min (12:06 pm)

Reference Photo No:
160

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

EAST_ELEVATION

NORTH



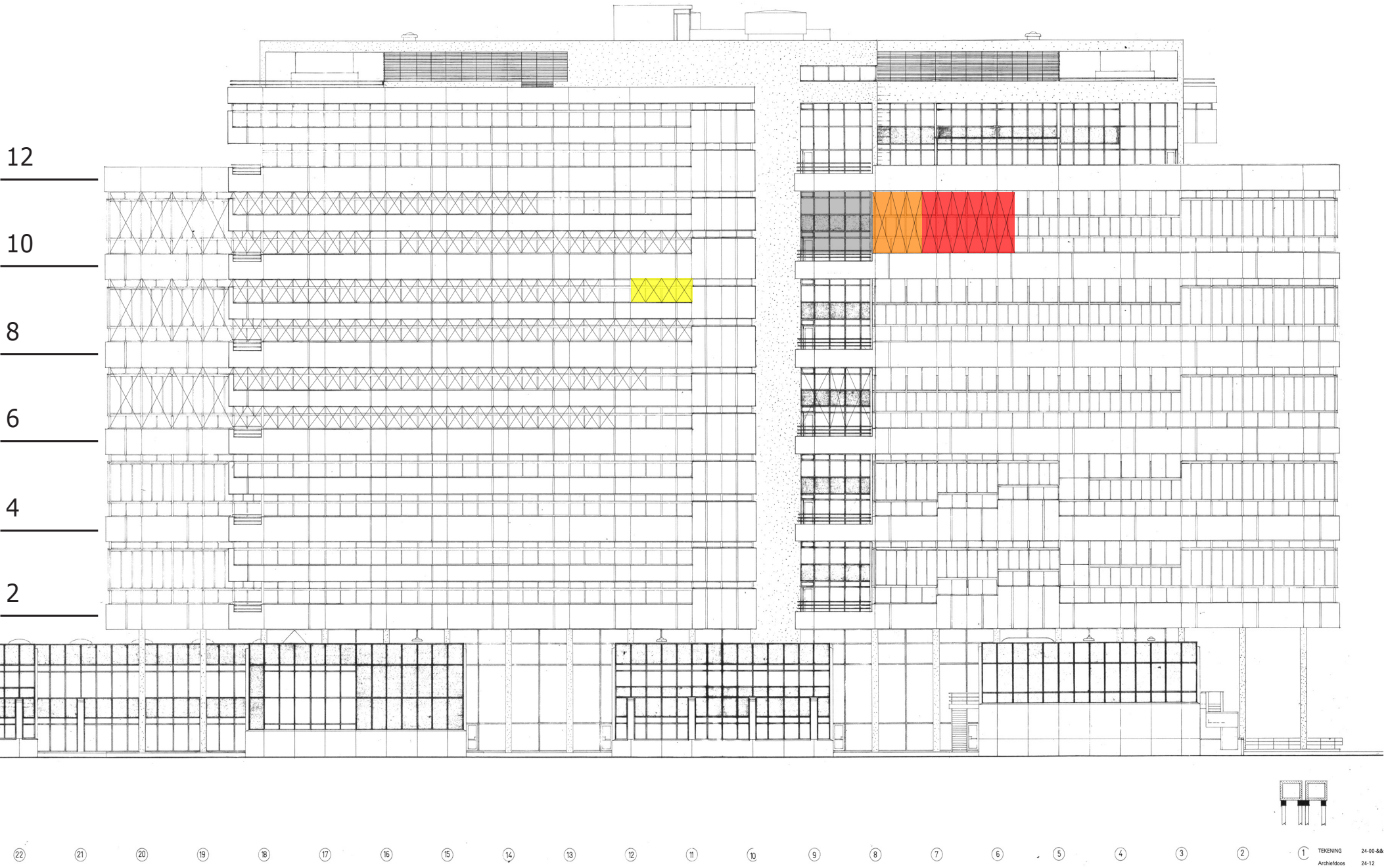
t = 250 min (12:50 pm)

Reference Photo No:
167, 168

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

EAST_ELEVATION

NORTH



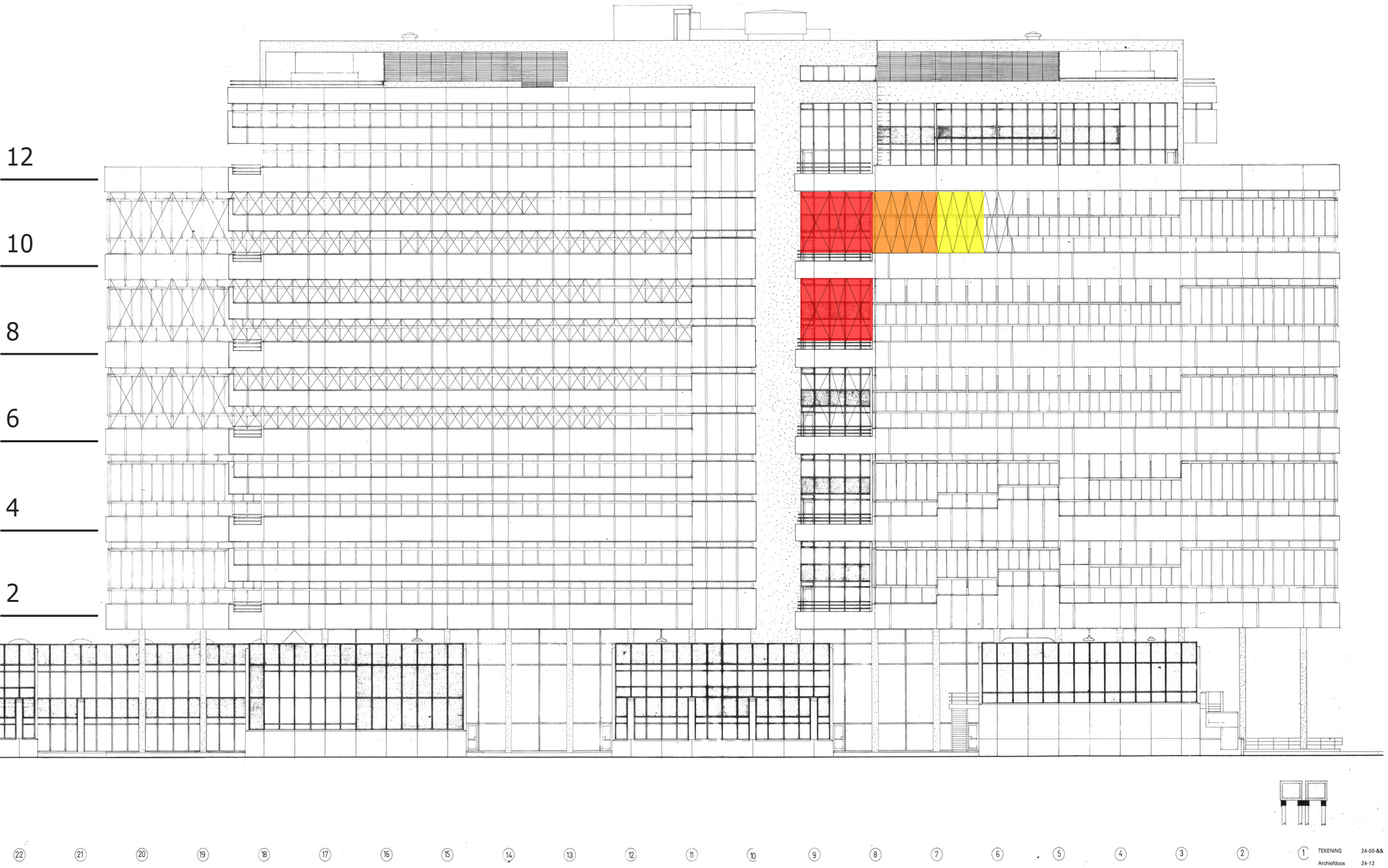
t = 279 min (1:19 pm)

Reference Photo No:
169, 170

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

EAST_ELEVATION

NORTH



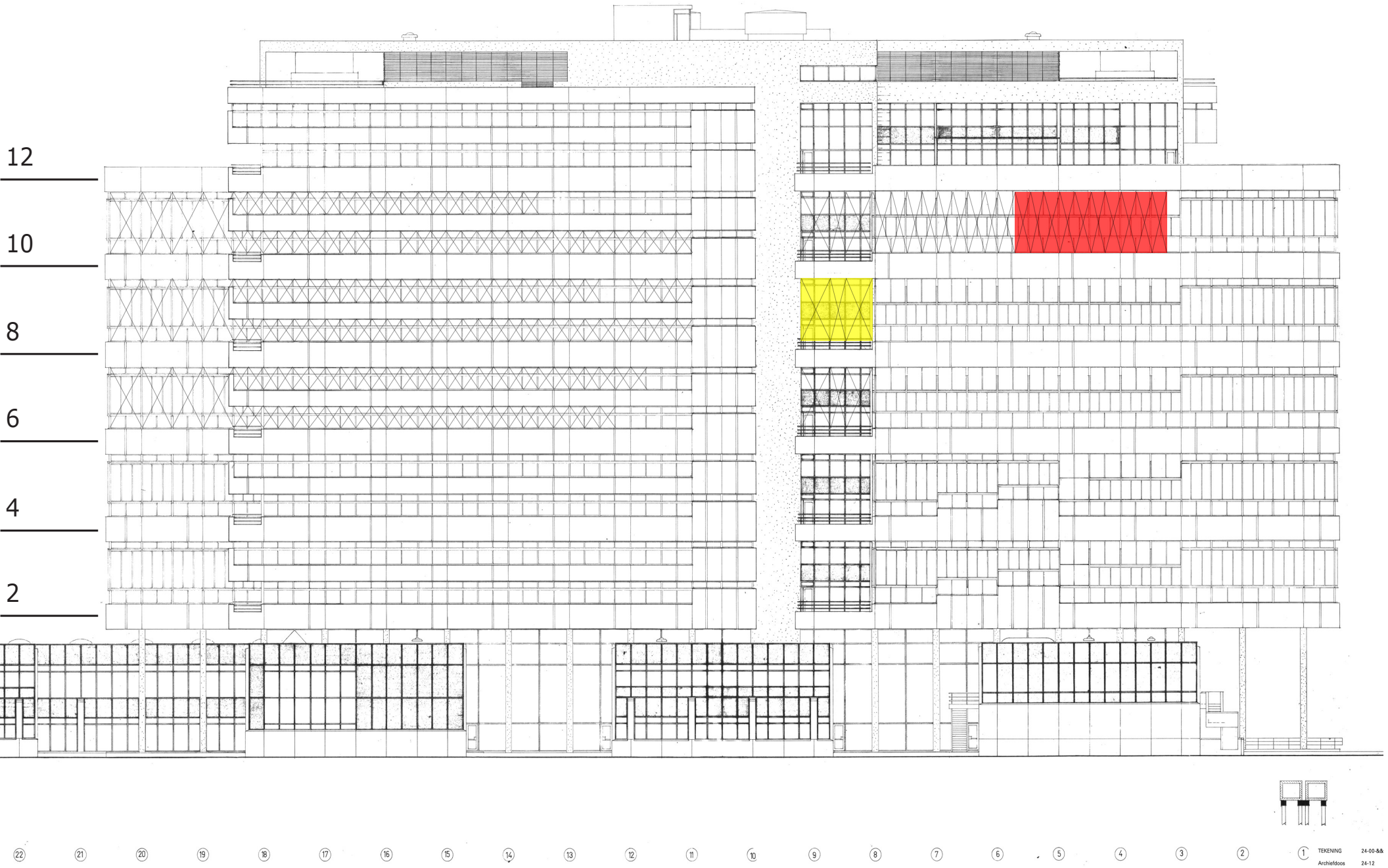
t = 289 min (1:29 pm)

Reference Photo No:
219, 220

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

EAST_ELEVATION

NORTH



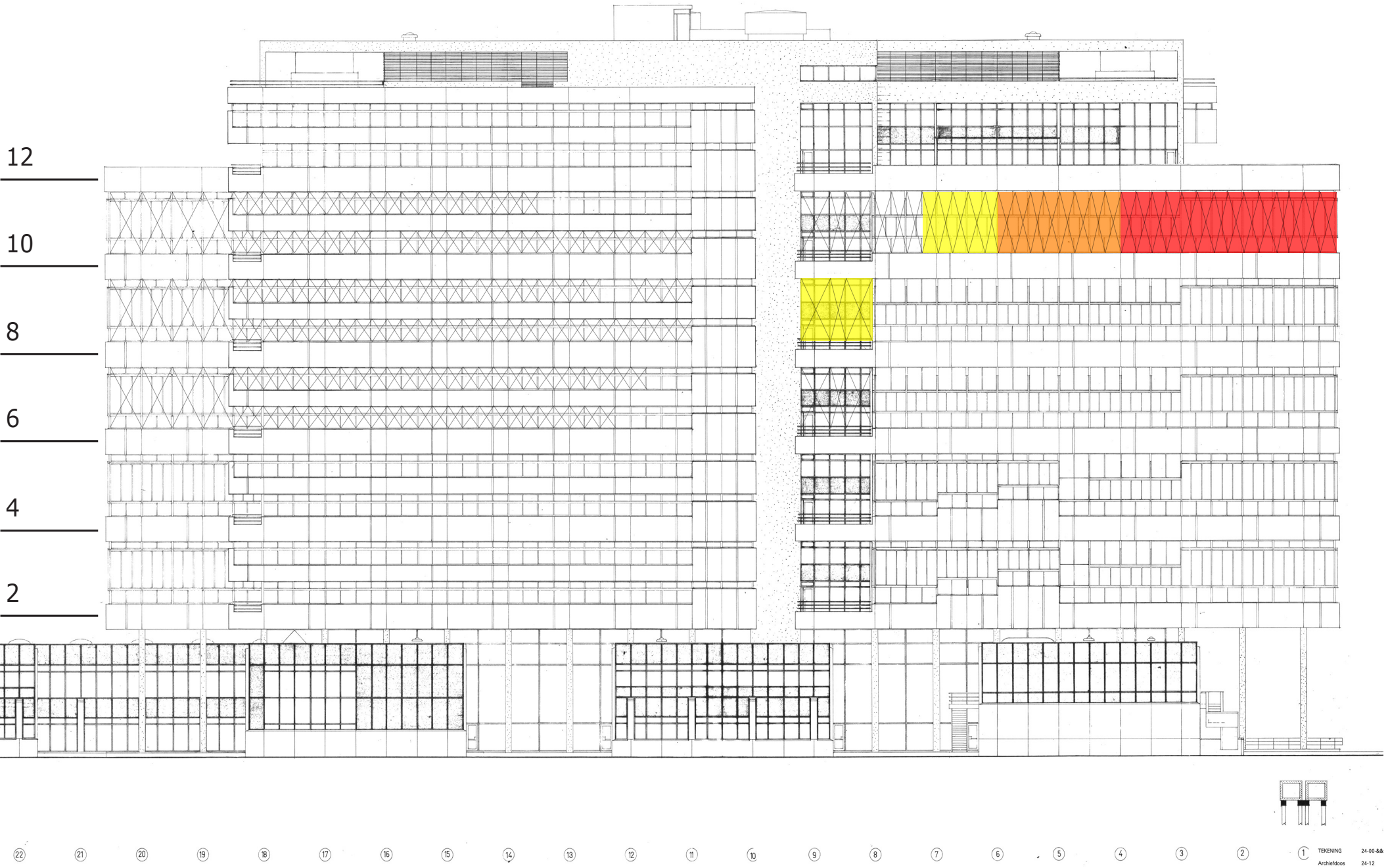
t = 323 min (2:03 pm)

Reference Photo No:
246, 247, 248, 249, 250

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

EAST_ELEVATION

NORTH



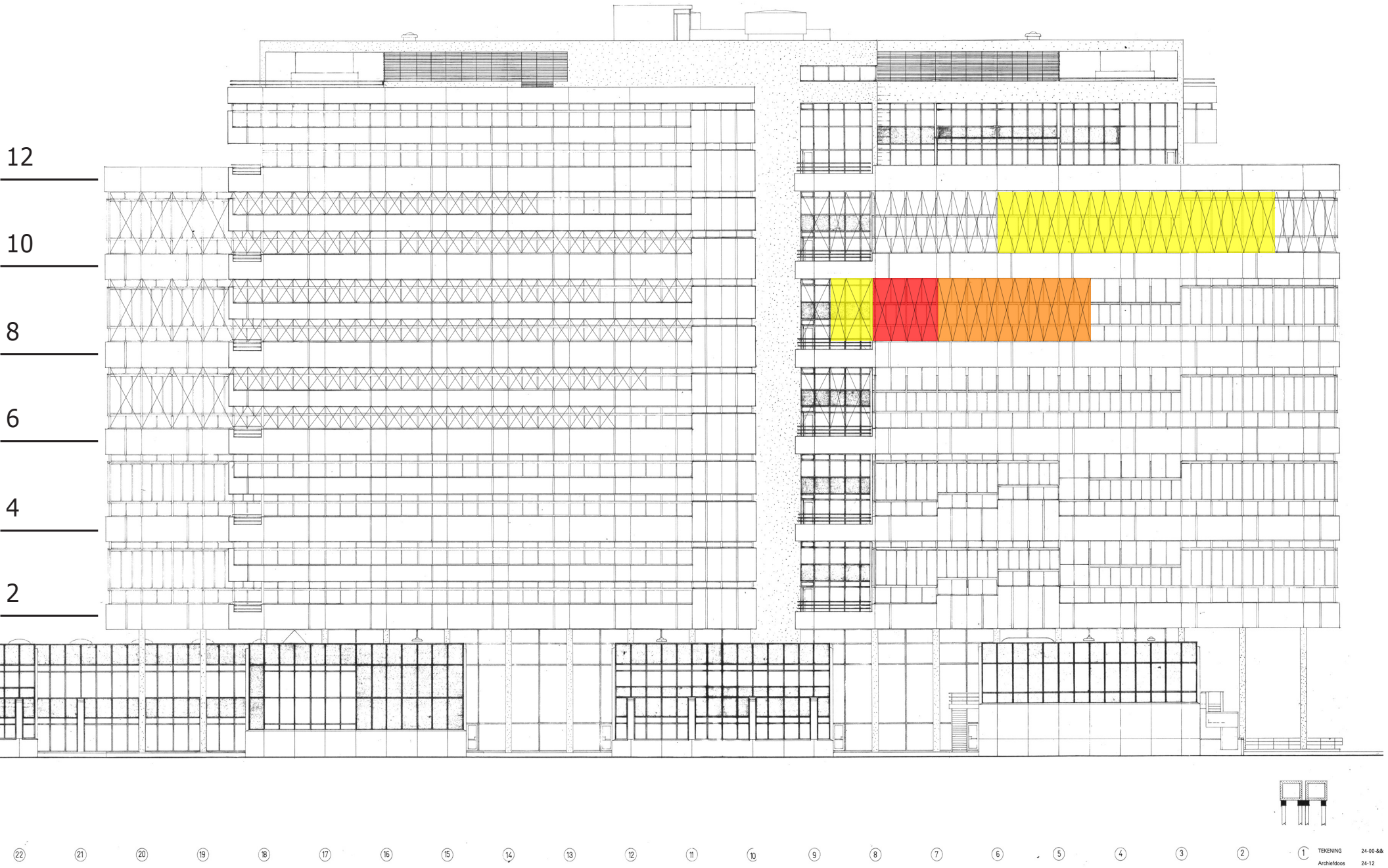
t = 330 min (2:10 pm)

Reference Photo No:
264

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

EAST_ELEVATION

NORTH



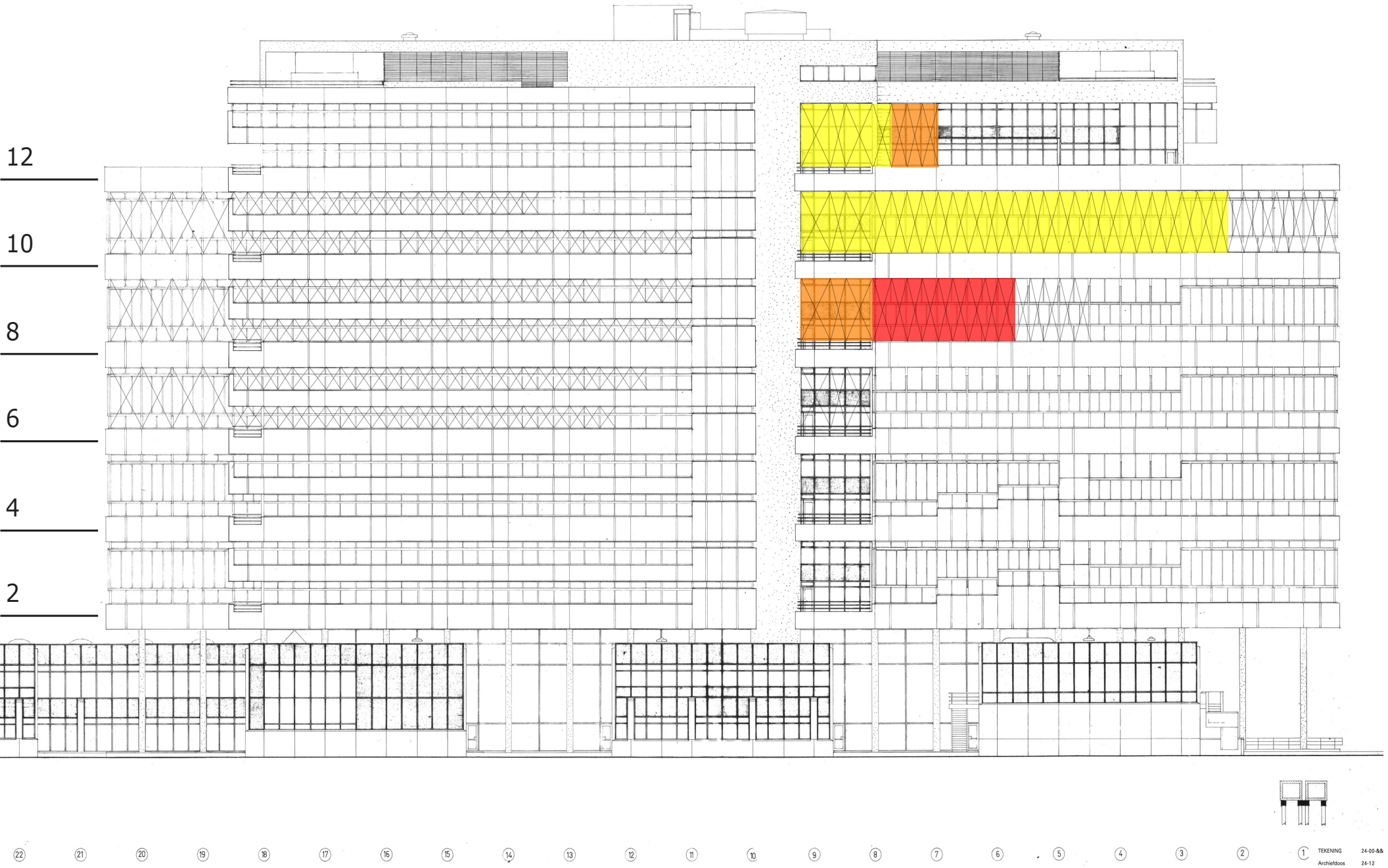
t = 332 min (2:12 pm)

Reference Photo No:
276, 277, 278, 279

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

EAST_ELEVATION

NORTH



t = 343 min (2:23 pm)

Reference Photo No:
377, 378, 379, 380

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

EAST_ELEVATION

NORTH

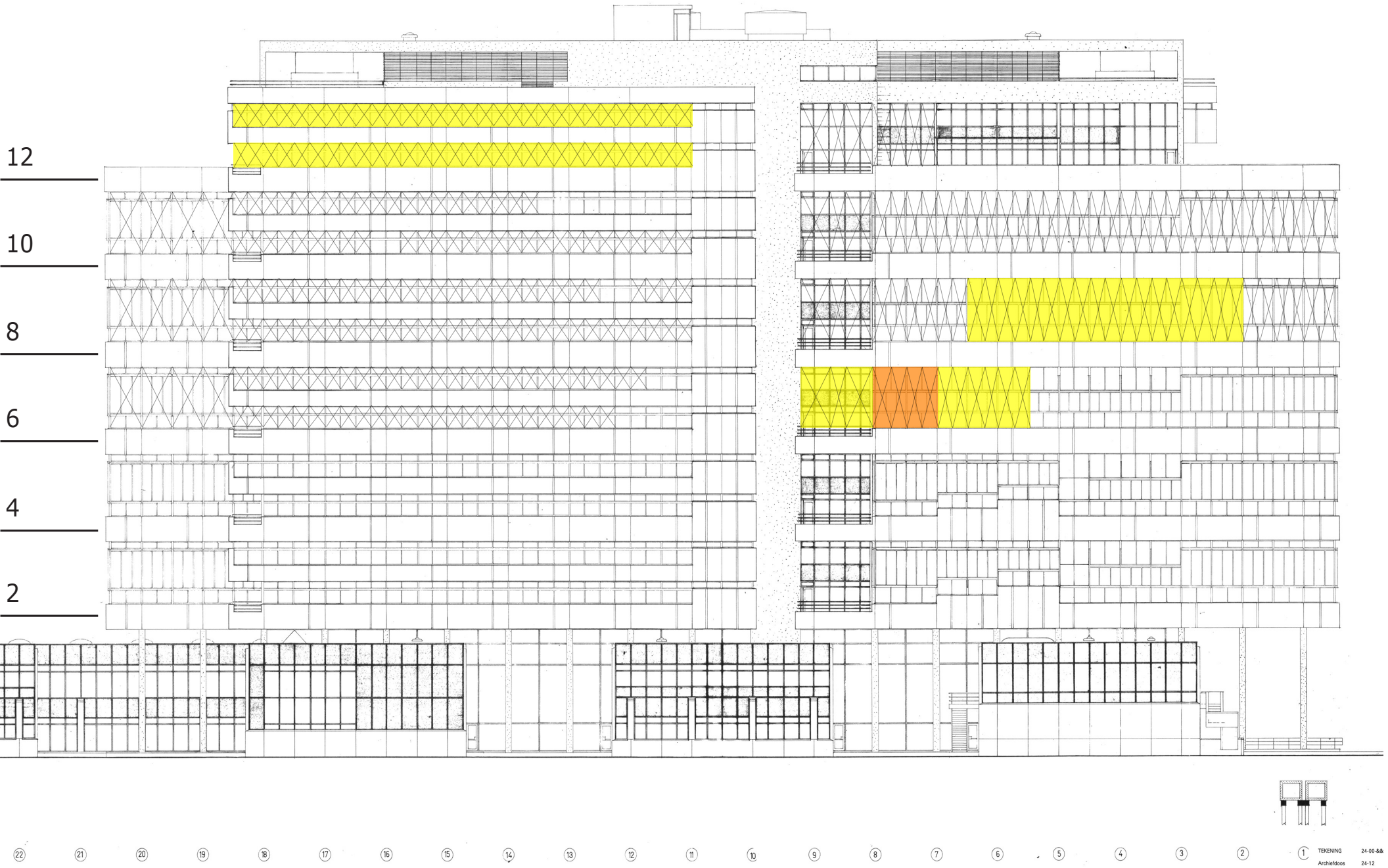
t = 391 min (3:11 pm)

Reference Photo No:
386, 387

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

EAST_ELEVATION

NORTH



Collapse at t = 403 min (3:43pm) 197

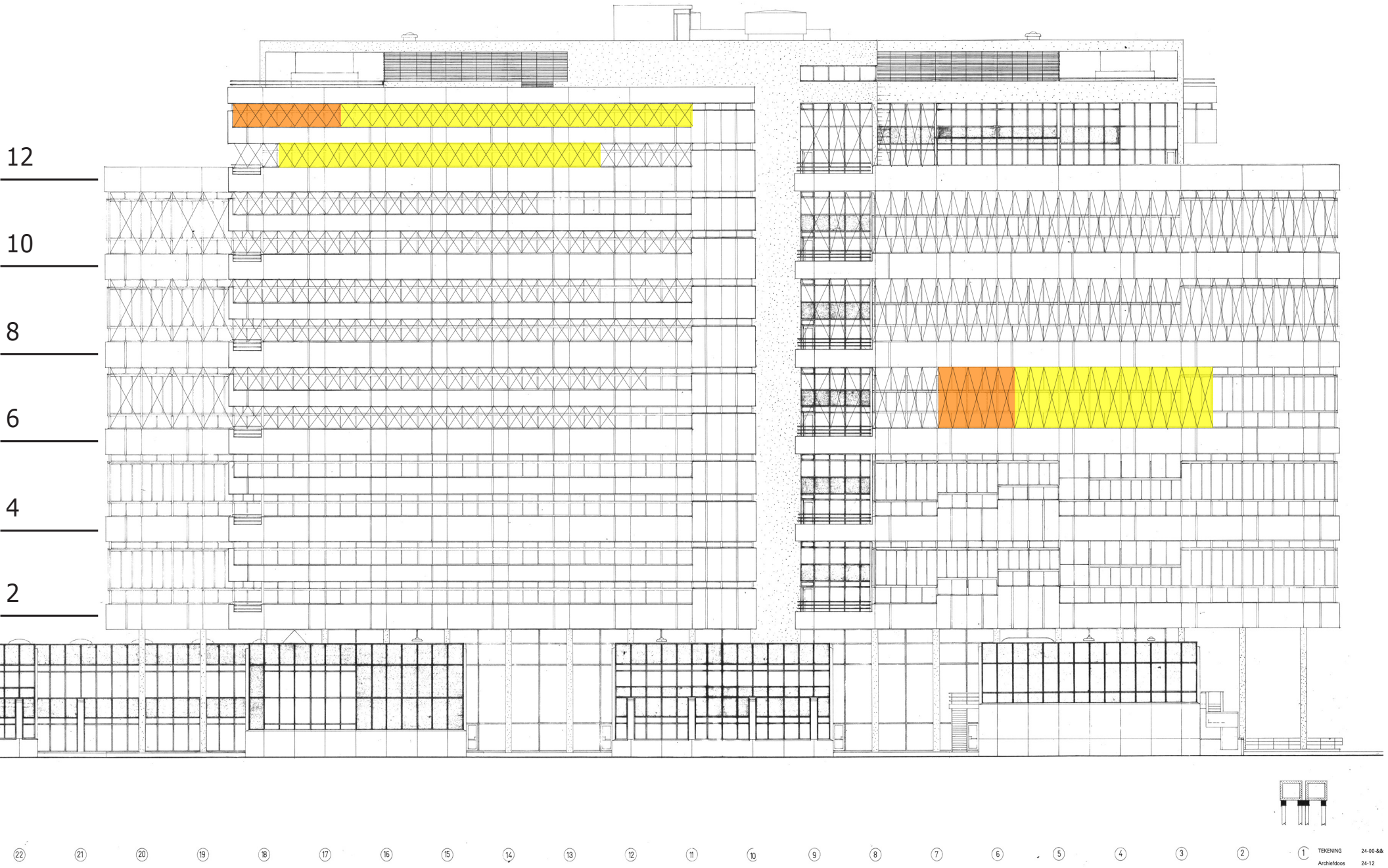
t = 407 min (3:27 pm)

Reference Photo No:
416, 417

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

EAST_ELEVATION

NORTH



t = 425 min (3:45 pm)

Reference Photo No:
426

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

EAST_ELEVATION

NORTH

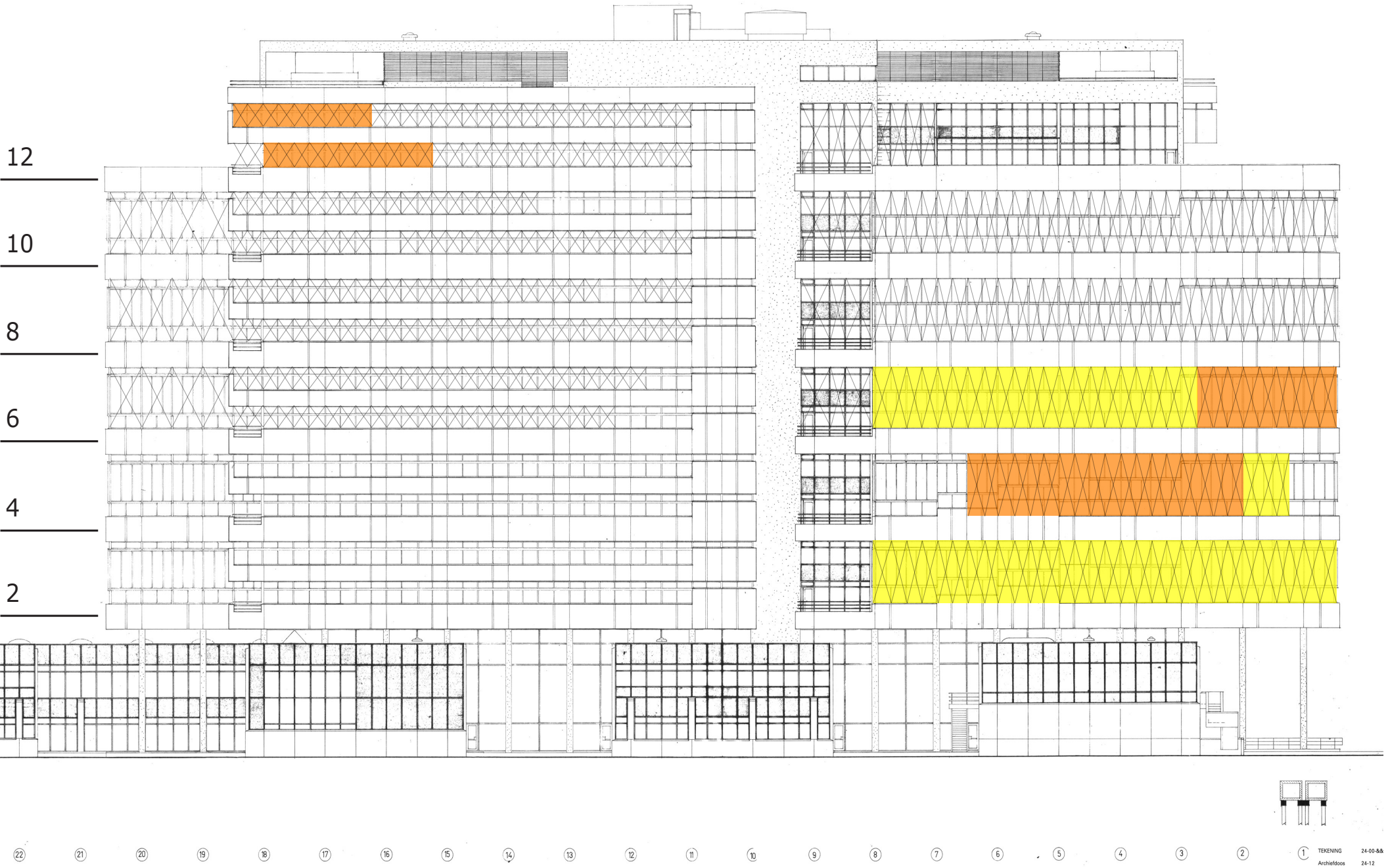
t = 431 min (3:51 pm)

Reference Photo No:
454, 455, 456

= Flame visible
= Flame out window
= Flame to next floor height
= Dense smoke
= Broken window

EAST_ELEVATION

NORTH



t = 438 min (3:58 pm)

APPENDIX D

UT Fire: Reinforced Concrete Analysis - Source Code

UT Fire: Reinforced Concrete Analysis was written using Microsoft Visual Basic 2008 Express Edition. The program consists of two user graphical user interface forms. The first is the input form, which is used to load the SAFIR .out file, define mechanical properties of materials, and define the section reinforcing. The output form shows results of the analysis. These results are also available in the Microsoft Excel file that is created when the program is run. This appendix provides the source code for the program, broken into the input and output form codes.

```
Imports System.IO
Imports Microsoft.VisualBasic
Imports MSChart20Lib
Imports Excel = Microsoft.Office.Interop.Excel
Imports System.Math

Public Class Input

    Dim OUTPUTfolder, FILEname As String
    Dim f_new, cSS_new, sSS_new, doop, soop As Boolean
    Dim teek As Integer

    Dim xlApp As Excel.Application
    Dim xlUTFire As Excel.Workbook
    Dim misValue As Object = System.Reflection.Missing.Value
    Dim xlOUT, xlREOUT, xlFire, xlSection, xlConcrete, xlSteel, xlMPhi, xlScratch, xlMN, ↙
    xlDOOP As Excel.Worksheet

    Dim analysis_MPhi, analysis_MN As Integer
    Dim output_N, output_M, FFF As Integer

    Dim nodeA(0 To 5000), nodeB(0 To 5000), nodeC(0 To 5000), nodeD(0 To 5000) As Single ↙
    'array(element_label)
    Dim nodeX(0 To 5000), nodeY(0 To 5000) As Single 'array(node_label)
    Dim nodeTEMP(0 To 5000) As Single 'array(node_label)
    Dim elementAREA(0 To 5000) As Single 'array(element_label)
    Dim element_Y(0 To 5000) As Single 'array(element_label)

    Dim Start, Nodes, Elements, XY As Integer
    Dim TimeStep, EndTime, NumSteps As Integer
    Dim DUM, REINFcount As Integer
    Dim REINFsize(0 To 20) As Integer 'array(rebar_label)
    Dim reinf_X(0 To 20), reinf_Y(0 To 20), reinf_AREA(0 To 20) As Single 'array ↙
    (rebar_label)

    Dim s_height, s_width As Single
    Dim Xaxis_max, Yaxis_max, axis_max As Single
    Dim Xaxis_min, Yaxis_min, axis_min As Single
    Dim square_scale As Single

    'initial conditions
    Dim fprimec, eo, ecu, fct, ect, Ecc, ectu As Single
    Dim aggregates As String
    Dim fp, fy, Est As Single
    Dim ep, ey, et, eu As Single
    Dim a, b, c As Single

    'for stress-strain plots
    Dim temp_TEMP(0 To 13) As Integer 'array(temp_range)
    Dim ec_TEMP(0 To 13, 0 To 1000), fc_TEMP(0 To 13, 0 To 1000) As Single 'array ↙
    (temp_range, plot_point)
    Dim es_TEMP(0 To 13, 0 To 1000), fs_TEMP(0 To 13, 0 To 1000) As Single 'array ↙
    (temp_range, plot_point)

    Dim fprimec_TEMP(0 To 13), eo_TEMP(0 To 13), ecu_TEMP(0 To 13) As Single 'array ↙
    (temp_range)
    Dim fct_TEMP(0 To 13), ect_TEMP(0 To 13), Ecc_TEMP(0 To 13), ectu_TEMP(0 To 13) As ↙
    Single 'array(temp_range)
    Dim fp_TEMP(0 To 13), fy_TEMP(0 To 13), Est_TEMP(0 To 13) As Single 'array ↙
    (temp_range)
    Dim ep_TEMP(0 To 13), ey_TEMP(0 To 13), et_TEMP(0 To 13), eu_TEMP(0 To 13) As Single ↙
    'array(temp_range)
    Dim a_TEMP(0 To 13), b_TEMP(0 To 13), c_TEMP(0 To 13) As Single 'array(temp_range)

    Dim tempCOUNT, INITIAL, plotCOUNT As Integer
```

```
    Dim x_col(0 To 13), y_col(0 To 13) As String 'array(temp_range)

    'for calculations
    Dim fprimec_t(0 To 5000, 0 To 1000), eo_t(0 To 10000, 0 To 1000), ecu_t(0 To 5000, 0 ↙
    To 1000) As Single 'array(element_label, time_step)
    Dim fct_t(0 To 5000, 0 To 1000), ect_t(0 To 5000, 0 To 1000), Ecc_t(0 To 5000, 0 To  ↙
    1000), ectu_t(0 To 5000, 0 To 1000) As Single 'array(element_label, time_step)
    Dim fp_t(0 To 20, 0 To 1000), fy_t(0 To 20, 0 To 1000), Est_t(0 To 20, 0 To 1000) As ↙
    Single 'array(rebar_label, time_step)
    Dim ep_t(0 To 20, 0 To 1000), ey_t(0 To 20, 0 To 1000), et_t(0 To 20, 0 To 1000),    ↙
    eu_t(0 To 20, 0 To 1000) As Single 'array(rebar_label, time_step)
    Dim a_t(0 To 20, 0 To 1000), b_t(0 To 20, 0 To 1000), c_t(0 To 20, 0 To 1000) As     ↙
    Single 'array(rebar_label, time_step)

    Dim time(0 To 1000), outputRow(0 To 1000), tDUM As Integer 'array(time_step)
    Dim sREFnode(0 To 18) As Integer 'array(rebar_label)
    Dim cTEMP(0 To 5000, 0 To 1000), sTEMP(0 To 18, 0 To 1000) As Single 'array(element/ ↙
    rebar_label, time_step)
    Dim ec(0 To 5000, 0 To 1000), es(0 To 18, 0 To 1000) As Single 'array(element/       ↙
    rebar_label, time_step)
    Dim fc(0 To 5000, 0 To 1000), fs(0 To 18, 0 To 1000) As Single 'array(element/       ↙
    rebar_label, time_step)
    Dim FORc(0 To 5000, 0 To 1000), FORs(0 To 18, 0 To 1000) As Single 'array(element/   ↙
    rebar_label, time_step)
    Dim MOMc(0 To 5000, 0 To 1000), MOMs(0 To 18, 0 To 1000) As Single 'array(element/   ↙
    rebar_label, time_step)

    Dim Tc_avg(0 To 1000), Ts_avg(0 To 1000), MMMM(0 To 10000) As Single 'array          ↙
    (time_step)
    Dim MOMmax(0 To 1000), CURVmax(0 To 1000), MOMmin(0 To 1000), CURVmin(0 To 1000) As  ↙
    Single 'array(time_step)
    Dim maxM, minM, maxF As Single

    'moment-curvature definitions
    Dim eTOP(0 To 1000, 0 To 1000) As Single '(mphi_plotPoint, time_step)
    Dim c_1, c_2, c_3, N_1, N_2, N_3, M_1, M_2, M_3, phi_1, phi_2, phi_3, c_previous As  ↙
    Single
    Dim yCprime(0 To 5000, 0 To 1000), ySprime(0 To 20, 0 To 1000) As Single '(element/  ↙
    rebar_label, mphi_plotPoint)

    Dim Mom(0 To 1000, 0 To 1000), phi(0 To 1000, 0 To 1000) As Single
    Dim mphiCOUNT, DUMMy As Integer
    Dim eTOP_min, eTOP_max, phi_min, phi_max, Accuracy, eTOP_step As Single
    Dim sumC, sumS, sumMC, sumMS As Single

    'moment-axial definitions
    Dim mnCOUNT, itt As Integer
    Dim estep, emax, cstep, c_max, c_min, c_, phi_, Mm, Nn, ee, yprime As Single
    Dim eTest(0 To 1000), eULT(0 To 1000), Po(0 To 1000), Mo(0 To 1000), P_dum(0 To 5000)↙
     As Single 'array(timestep)
    Dim Pon(0 To 1000), Mon(0 To 1000) As Single 'array(timestep)

    Dim koop As Boolean



    'RUN_ANALYSIS
    Private Sub btnSubmit_Click(ByVal sender As System.Object, ByVal e As System.       ↙
    EventArgs) Handles btnSubmit.Click

        koop = True
        soop = False

        prgMAIN.Visible = True
        prgMAIN.Value = 0
```

```
        xlScratch.Cells(1, 1).value = 5

        Me.tabctrlMain.SelectTab(4)

        If xlScratch.Cells(1, 6).value = "DONE" Then GoTo 99999

        txtRUN.Visible = True
        txtRUN.Text = "Finding Element Temps(t)"
        txtRUN1.Visible = True

        timer_i.Visible = True
        timer_j.Visible = True
        timer_iteration.Visible = True
        timer_c.Visible = True
        timer_N.Visible = True
        timer_M.Visible = True
        timer_phi.Visible = True
        lbl_i.Visible = True
        lbl_j.Visible = True
        lbl_iteration.Visible = True
        lbl_c.Visible = True
        lbl_N.Visible = True
        lbl_M.Visible = True
        lbl_phi.Visible = True

        'Redefine concrete properties
        fprimec = txtConcreteStrength.Text
        aggregates = cmbAggregate.Text
        eo = 0.0025
        ecu = 0.02
        fct = -0.6228 * (fprimec) ^ 0.5 '= 7.5*(f'c)^0.5 in psi
        Ecc = 4732.98 * (fprimec) ^ 0.5 '= 57000*(f'c)^0.5 in psi
        ect = fct / Ecc
        ectu = 11 * fct / Ecc

        'Redefine steel properties
        fp = txtSteelProportional.Text
        fy = txtSteelYield.Text
        Est = 199948
        ep = fp / Est
        ey = 0.02
        et = 0.15
        eu = 0.2
        c = ((fy - fp) ^ 2) / ((ey - ep) * Est - 2 * (fy - fp))
        a = ((ey - ep) * (ey - ep + c / Est)) ^ 0.5
        b = (c * (ey - ep) * Est + c ^ 2) ^ 0.5

        'Make sure all bars all written to Excel
        xlREOUT.Cells(2, 38).value = cmbBarSize1.Text
        xlREOUT.Cells(3, 38).value = cmbBarSize2.Text
        xlREOUT.Cells(4, 38).value = cmbBarSize3.Text
        xlREOUT.Cells(5, 38).value = cmbBarSize4.Text
        xlREOUT.Cells(6, 38).value = cmbBarSize5.Text
        xlREOUT.Cells(7, 38).value = cmbBarSize6.Text
        xlREOUT.Cells(8, 38).value = cmbBarSize7.Text
        xlREOUT.Cells(9, 38).value = cmbBarSize8.Text
        xlREOUT.Cells(10, 38).value = cmbBarSize9.Text
        xlREOUT.Cells(11, 38).value = cmbBarSize10.Text
        xlREOUT.Cells(12, 38).value = cmbBarSize11.Text
        xlREOUT.Cells(13, 38).value = cmbBarSize12.Text
        xlREOUT.Cells(14, 38).value = cmbBarSize13.Text
        xlREOUT.Cells(15, 38).value = cmbBarSize14.Text
        xlREOUT.Cells(16, 38).value = cmbBarSize15.Text
        xlREOUT.Cells(17, 38).value = cmbBarSize16.Text
        xlREOUT.Cells(18, 38).value = cmbBarSize17.Text
        xlREOUT.Cells(19, 38).value = cmbBarSize18.Text
```

```
        xlREOUT.Cells(2, 39).value = txtReinfX1.Text
        xlREOUT.Cells(3, 39).value = txtReinfX2.Text
        xlREOUT.Cells(4, 39).value = txtReinfX3.Text
        xlREOUT.Cells(5, 39).value = txtReinfX4.Text
        xlREOUT.Cells(6, 39).value = txtReinfX5.Text
        xlREOUT.Cells(7, 39).value = txtReinfX6.Text
        xlREOUT.Cells(8, 39).value = txtReinfX7.Text
        xlREOUT.Cells(9, 39).value = txtReinfX8.Text
        xlREOUT.Cells(10, 39).value = txtReinfX9.Text
        xlREOUT.Cells(11, 39).value = txtReinfX10.Text
        xlREOUT.Cells(12, 39).value = txtReinfX11.Text
        xlREOUT.Cells(13, 39).value = txtReinfX12.Text
        xlREOUT.Cells(14, 39).value = txtReinfX13.Text
        xlREOUT.Cells(15, 39).value = txtReinfX14.Text
        xlREOUT.Cells(16, 39).value = txtReinfX15.Text
        xlREOUT.Cells(17, 39).value = txtReinfX16.Text
        xlREOUT.Cells(18, 39).value = txtReinfX17.Text
        xlREOUT.Cells(19, 39).value = txtReinfX18.Text
        xlREOUT.Cells(2, 40).value = txtReinfY1.Text
        xlREOUT.Cells(3, 40).value = txtReinfY2.Text
        xlREOUT.Cells(4, 40).value = txtReinfY3.Text
        xlREOUT.Cells(5, 40).value = txtReinfY4.Text
        xlREOUT.Cells(6, 40).value = txtReinfY5.Text
        xlREOUT.Cells(7, 40).value = txtReinfY6.Text
        xlREOUT.Cells(8, 40).value = txtReinfY7.Text
        xlREOUT.Cells(9, 40).value = txtReinfY8.Text
        xlREOUT.Cells(10, 40).value = txtReinfY9.Text
        xlREOUT.Cells(11, 40).value = txtReinfY10.Text
        xlREOUT.Cells(12, 40).value = txtReinfY11.Text
        xlREOUT.Cells(13, 40).value = txtReinfY12.Text
        xlREOUT.Cells(14, 40).value = txtReinfY13.Text
        xlREOUT.Cells(15, 40).value = txtReinfY14.Text
        xlREOUT.Cells(16, 40).value = txtReinfY15.Text
        xlREOUT.Cells(17, 40).value = txtReinfY16.Text
        xlREOUT.Cells(18, 40).value = txtReinfY17.Text
        xlREOUT.Cells(19, 40).value = txtReinfY18.Text

        'Count reinforcing bars
        For i = 1 To 18
            If xlREOUT.Cells(i + 1, 38).value = 0 Or xlREOUT.Cells(i + 1, 39).value = 0 ↙
    Or xlREOUT.Cells(i + 1, 40).value = 0 Then
                REINFcount = i - 1
                GoTo 1
            End If
        Next i
        REINFcount = 18

1:      xlREOUT.Cells(23, 2).value = REINFcount
        For i = REINFcount + 1 To 19
            xlREOUT.Cells(i + 1, 38).value = ""
            xlREOUT.Cells(i + 1, 39).value = ""
            xlREOUT.Cells(i + 1, 40).value = ""
        Next i

        'Convert input to SAFIR coordinate system
        For i = 1 To REINFcount
            reinf_X(i) = -s_width / 2 + xlREOUT.Cells(i + 1, 39).value
            xlSection.Cells(i, 4).value = reinf_X(i)
            xlREOUT.Cells(i + 1, 41).value = reinf_X(i)
            reinf_Y(i) = -s_height / 2 + xlREOUT.Cells(i + 1, 40).value
            xlSection.Cells(i, 5).value = reinf_Y(i)
            xlREOUT.Cells(i + 1, 42).value = reinf_Y(i)
            For j = 1 To Nodes
                If nodeX(j) = reinf_X(i) Then GoTo 11
                GoTo 12
11:             If nodeY(j) = reinf_Y(i) Then
```

```
                    sREFnode(i) = j
                    GoTo 13
                End If
12:         Next j
            MessageBox.Show("Reinforcement coordinates must match a SAFIR.out nodal ↙
    coordinate")
            End
13:         xlREOUT.Cells(i + 1, 43).value = sREFnode(i)
            reinf_AREA(i) = PI * (xlREOUT.Cells(i + 1, 38).value / 2) ^ 2
            xlREOUT.Cells(i + 1, 52).value = reinf_AREA(i)
        Next i

        'Find SAFIR.out temperature rows
        tDUM = 1
        With xlOUT
            .Range("A25:B1025").Clear()
            For i = 1 To 100000
                If IsNumeric(.Cells(i, 2).value) = True Then GoTo 2
                If .Cells(i, 2).value = "TOTAL" Then
                    outputRow(tDUM) = i + 3
                    tDUM = tDUM + 1
                    prgMAIN.Value = Round(1000 * tDUM / (4 * (NumSteps + 1)))
                    If tDUM = NumSteps + 1 Then GoTo 3
                End If
2:          Next i
        End With
3:      With xlREOUT
            .Cells(25, 1).value = "TimeStep"
            .Cells(25, 1).HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter
            .Cells(25, 1).font.bold = True
            .Cells(25, 2).value = "TempRow"
            .Cells(25, 2).HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter
            .Cells(25, 2).font.bold = True
            time(0) = 0
            outputRow(0) = outputRow(1)
            .Cells(26, 1).value = time(0)
            .Cells(26, 2).value = outputRow(0)
            .Cells(26, 1).HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter
            .Cells(26, 2).HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter
            For i = 1 To NumSteps
                time(i) = .Cells(i + 25, 1).value + TimeStep
                .Cells(i + 26, 1).value = time(i)
                .Cells(i + 26, 2).value = outputRow(i)
                .Cells(i + 26, 1).HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter
                .Cells(i + 26, 2).HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter
            Next i
        End With

        prgMAIN.Value = 1000 / 4

        'BIG_LOOP&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&↙
    &&&
        For i = 0 To NumSteps
            prgMAIN.Value = 1000 / 4 + Round(3000 * i / (4 * NumSteps))
            timer_i.Text = i
            MOMmax(i) = 0
            MOMmin(i) = 0
            CURVmax(i) = 0
            CURVmin(i) = 0

            txtRUN.Text = "Mechanical Props" & " at t = " & i * TimeStep

            'Find nodal temperatures
            With xlOUT
                For j = 1 To Nodes
                    For k = Floor(j / 5.1) To Floor(Nodes / 5)
```

```
                        If .Cells(outputRow(i) + k, 2).value = j Then
                            nodeTEMP(j) = .Cells(outputRow(i) + k, 3).value
                            GoTo 4
                        End If
                        If .Cells(outputRow(i) + k, 4).value = j Then
                            nodeTEMP(j) = .Cells(outputRow(i) + k, 5).value
                            GoTo 4
                        End If
                        If .Cells(outputRow(i) + k, 6).value = j Then
                            nodeTEMP(j) = .Cells(outputRow(i) + k, 7).value
                            GoTo 4
                        End If
                        If .Cells(outputRow(i) + k, 8).value = j Then
                            nodeTEMP(j) = .Cells(outputRow(i) + k, 9).value
                            GoTo 4
                        End If
                        If .Cells(outputRow(i) + k, 10).value = j Then
                            nodeTEMP(j) = .Cells(outputRow(i) + k, 11).value
                            GoTo 4
                        End If
                    Next k
4:              Next j
            End With

            With xlREOUT
                For j = 1 To Nodes
                    .Cells(j + 1, 7).value = nodeTEMP(j)
                Next j
                For j = 1 To Elements
                    .Cells(j + 1, 12).value = nodeTEMP(nodeA(j))
                    .Cells(j + 1, 15).value = nodeTEMP(nodeB(j))
                    .Cells(j + 1, 18).value = nodeTEMP(nodeC(j))
                    .Cells(j + 1, 21).value = nodeTEMP(nodeD(j))
                    cTEMP(j, i) = (nodeTEMP(nodeA(j)) + nodeTEMP(nodeB(j)) + nodeTEMP     ↙
    (nodeC(j)) + nodeTEMP(nodeD(j))) / 4
                    .Cells(j + 1, 25).value = cTEMP(j, i)
                Next j

                'Find Reinforcement Temperatures
                For j = 1 To REINFcount
                    sTEMP(j, i) = nodeTEMP(sREFnode(j))
                    .Cells(j + 1, 44).value = sTEMP(j, i)
                Next j

                'Concrete Reduced Mechanical Properties
                If aggregates = "Siliceous" Then GoTo 5
                If aggregates = "Calcareous" Then GoTo 6
                MessageBox.Show("Concrete aggregates not specified")
                End

                'siliceous aggs
5:              For j = 1 To Elements
                    If cTEMP(j, i) <= 100 Then GoTo 1001
                    If 100 < cTEMP(j, i) And cTEMP(j, i) <= 200 Then GoTo 2001
                    If 200 < cTEMP(j, i) And cTEMP(j, i) <= 300 Then GoTo 3001
                    If 300 < cTEMP(j, i) And cTEMP(j, i) <= 400 Then GoTo 4001
                    If 400 < cTEMP(j, i) And cTEMP(j, i) <= 500 Then GoTo 5001
                    If 500 < cTEMP(j, i) And cTEMP(j, i) <= 600 Then GoTo 6001
                    If 600 < cTEMP(j, i) And cTEMP(j, i) <= 700 Then GoTo 7001
                    If 700 < cTEMP(j, i) And cTEMP(j, i) <= 800 Then GoTo 8001
                    If 800 < cTEMP(j, i) And cTEMP(j, i) <= 900 Then GoTo 9001
                    If 900 < cTEMP(j, i) And cTEMP(j, i) <= 1000 Then GoTo 10001
                    If 1000 < cTEMP(j, i) And cTEMP(j, i) <= 1100 Then GoTo 11001
                    If 1100 < cTEMP(j, i) And cTEMP(j, i) <= 1200 Then GoTo 12001
                    If 1200 < cTEMP(j, i) Then GoTo 13001
```

```
1001:               fprimec_t(j, i) = fprimec * (1.0 + (cTEMP(j, i) - 20) * ((1.0 - 1.0) ↙
    / (100 - 20)))
                    eo_t(j, i) = (0.0025 + (cTEMP(j, i) - 20) * ((0.004 - 0.0025) / (100 ↙
    - 20)))
                    ecu_t(j, i) = (0.02 + (cTEMP(j, i) - 20) * ((0.0225 - 0.02) / (100 - ↙
    20)))
                    fct_t(j, i) = 1.0 * fct

                    .Cells(j + 1, 26).value = fprimec_t(j, i)
                    .Cells(j + 1, 27).value = eo_t(j, i)
                    .Cells(j + 1, 28).value = ecu_t(j, i)
                    GoTo 99991

2001:               fprimec_t(j, i) = fprimec * (1.0 + (cTEMP(j, i) - 100) * ((0.95 - 1. ↙
    0) / (200 - 100)))
                    eo_t(j, i) = (0.004 + (cTEMP(j, i) - 100) * ((0.0055 - 0.004) / (200 ↙
    - 100)))
                    ecu_t(j, i) = (0.0225 + (cTEMP(j, i) - 100) * ((0.025 - 0.0225) /    ↙
    (200 - 100)))
                    fct_t(j, i) = (1.0 - 1.0 * (cTEMP(j, i) - 100) / 500) * fct

                    .Cells(j + 1, 26).value = fprimec_t(j, i)
                    .Cells(j + 1, 27).value = eo_t(j, i)
                    .Cells(j + 1, 28).value = ecu_t(j, i)
                    GoTo 99991

3001:               fprimec_t(j, i) = fprimec * (0.95 + (cTEMP(j, i) - 200) * ((0.85 - 0.↙
    95) / (300 - 200)))
                    eo_t(j, i) = (0.0055 + (cTEMP(j, i) - 200) * ((0.007 - 0.0055) / (300↙
     - 200)))
                    ecu_t(j, i) = (0.025 + (cTEMP(j, i) - 200) * ((0.0275 - 0.025) / (300↙
     - 200)))
                    fct_t(j, i) = (1.0 - 1.0 * (cTEMP(j, i) - 100) / 500) * fct

                    .Cells(j + 1, 26).value = fprimec_t(j, i)
                    .Cells(j + 1, 27).value = eo_t(j, i)
                    .Cells(j + 1, 28).value = ecu_t(j, i)
                    GoTo 99991

4001:               fprimec_t(j, i) = fprimec * (0.85 + (cTEMP(j, i) - 300) * ((0.75 - 0.↙
    85) / (400 - 300)))
                    eo_t(j, i) = (0.007 + (cTEMP(j, i) - 300) * ((0.01 - 0.007) / (400 - ↙
    300)))
                    ecu_t(j, i) = (0.0275 + (cTEMP(j, i) - 300) * ((0.03 - 0.0275) / (400↙
     - 300)))
                    fct_t(j, i) = (1.0 - 1.0 * (cTEMP(j, i) - 100) / 500) * fct

                    .Cells(j + 1, 26).value = fprimec_t(j, i)
                    .Cells(j + 1, 27).value = eo_t(j, i)
                    .Cells(j + 1, 28).value = ecu_t(j, i)
                    GoTo 99991

5001:               fprimec_t(j, i) = fprimec * (0.75 + (cTEMP(j, i) - 400) * ((0.6 - 0. ↙
    75) / (500 - 400)))
                    eo_t(j, i) = (0.01 + (cTEMP(j, i) - 400) * ((0.015 - 0.01) / (500 -  ↙
    400)))
                    ecu_t(j, i) = (0.03 + (cTEMP(j, i) - 400) * ((0.0325 - 0.03) / (500 -↙
     400)))
                    fct_t(j, i) = (1.0 - 1.0 * (cTEMP(j, i) - 100) / 500) * fct

                    .Cells(j + 1, 26).value = fprimec_t(j, i)
                    .Cells(j + 1, 27).value = eo_t(j, i)
                    .Cells(j + 1, 28).value = ecu_t(j, i)
                    GoTo 99991

6001:               fprimec_t(j, i) = fprimec * (0.6 + (cTEMP(j, i) - 500) * ((0.45 - 0. ↙
```

```
    6) / (600 - 500)))
                    eo_t(j, i) = (0.015 + (cTEMP(j, i) - 500) * ((0.025 - 0.015) / (600 -↙
     500)))
                    ecu_t(j, i) = (0.0325 + (cTEMP(j, i) - 500) * ((0.035 - 0.0325) /    ↙
    (600 - 500)))
                    fct_t(j, i) = (1.0 - 1.0 * (cTEMP(j, i) - 100) / 500) * fct

                    .Cells(j + 1, 26).value = fprimec_t(j, i)
                    .Cells(j + 1, 27).value = eo_t(j, i)
                    .Cells(j + 1, 28).value = ecu_t(j, i)
                    GoTo 99991

7001:               fprimec_t(j, i) = fprimec * (0.45 + (cTEMP(j, i) - 600) * ((0.3 - 0. ↙
    45) / (700 - 600)))
                    eo_t(j, i) = (0.025 + (cTEMP(j, i) - 600) * ((0.025 - 0.025) / (700 -↙
     600)))
                    ecu_t(j, i) = (0.035 + (cTEMP(j, i) - 600) * ((0.0375 - 0.035) / (700↙
     - 600)))
                    fct_t(j, i) = 0

                    .Cells(j + 1, 26).value = fprimec_t(j, i)
                    .Cells(j + 1, 27).value = eo_t(j, i)
                    .Cells(j + 1, 28).value = ecu_t(j, i)
                    GoTo 99991

8001:               fprimec_t(j, i) = fprimec * (0.3 + (cTEMP(j, i) - 700) * ((0.15 - 0. ↙
    3) / (800 - 700)))
                    eo_t(j, i) = (0.025 + (cTEMP(j, i) - 700) * ((0.025 - 0.025) / (800 -↙
     700)))
                    ecu_t(j, i) = (0.0375 + (cTEMP(j, i) - 700) * ((0.04 - 0.0375) / (800↙
     - 700)))
                    fct_t(j, i) = 0

                    .Cells(j + 1, 26).value = fprimec_t(j, i)
                    .Cells(j + 1, 27).value = eo_t(j, i)
                    .Cells(j + 1, 28).value = ecu_t(j, i)
                    GoTo 99991

9001:               fprimec_t(j, i) = fprimec * (0.15 + (cTEMP(j, i) - 800) * ((0.08 - 0.↙
    15) / (900 - 800)))
                    eo_t(j, i) = (0.025 + (cTEMP(j, i) - 800) * ((0.025 - 0.025) / (900 -↙
     800)))
                    ecu_t(j, i) = (0.04 + (cTEMP(j, i) - 800) * ((0.0425 - 0.04) / (900 -↙
     800)))
                    fct_t(j, i) = 0

                    .Cells(j + 1, 26).value = fprimec_t(j, i)
                    .Cells(j + 1, 27).value = eo_t(j, i)
                    .Cells(j + 1, 28).value = ecu_t(j, i)
                    GoTo 99991

10001:              fprimec_t(j, i) = fprimec * (0.08 + (cTEMP(j, i) - 900) * ((0.04 - 0.↙
    08) / (1000 - 900)))
                    eo_t(j, i) = (0.025 + (cTEMP(j, i) - 900) * ((0.025 - 0.025) / (1000 ↙
    - 900)))
                    ecu_t(j, i) = (0.0425 + (cTEMP(j, i) - 900) * ((0.045 - 0.0425) /    ↙
    (1000 - 900)))
                    fct_t(j, i) = 0

                    .Cells(j + 1, 26).value = fprimec_t(j, i)
                    .Cells(j + 1, 27).value = eo_t(j, i)
                    .Cells(j + 1, 28).value = ecu_t(j, i)
                    GoTo 99991

11001:              fprimec_t(j, i) = fprimec * (0.04 + (cTEMP(j, i) - 1000) * ((0.01 - 0↙
    .04) / (1100 - 1000)))
```

```
                    eo_t(j, i) = (0.025 + (cTEMP(j, i) - 1000) * ((0.025 - 0.025) / (1100 - 1000)))
                    ecu_t(j, i) = (0.045 + (cTEMP(j, i) - 1000) * ((0.0475 - 0.045) / (1100 - 1000)))
                    fct_t(j, i) = 0

                    .Cells(j + 1, 26).value = fprimec_t(j, i)
                    .Cells(j + 1, 27).value = eo_t(j, i)
                    .Cells(j + 1, 28).value = ecu_t(j, i)
                    GoTo 99991

12001:              fprimec_t(j, i) = fprimec * (0.01 + (cTEMP(j, i) - 1100) * ((0.001 - 0.01) / (1200 - 1100)))
                    eo_t(j, i) = (0.025 + (cTEMP(j, i) - 1100) * ((0.025 - 0.025) / (1200 - 1100)))
                    ecu_t(j, i) = (0.0475 + (cTEMP(j, i) - 1100) * ((0.05 - 0.0475) / (1200 - 1100)))
                    fct_t(j, i) = 0

                    .Cells(j + 1, 26).value = fprimec_t(j, i)
                    .Cells(j + 1, 27).value = eo_t(j, i)
                    .Cells(j + 1, 28).value = ecu_t(j, i)
                    GoTo 99991

13001:              fprimec_t(j, i) = fprimec * 0.001
                    eo_t(j, i) = 0.025
                    ecu_t(j, i) = 0.05
                    fct_t(j, i) = 0

                    .Cells(j + 1, 26).value = fprimec_t(j, i)
                    .Cells(j + 1, 27).value = eo_t(j, i)
                    .Cells(j + 1, 28).value = ecu_t(j, i)
                    GoTo 99991

99991:              If fct_t(j, i) = 0 Then
                        Ecc_t(j, i) = 4732.98 * (fprimec_t(j, i)) ^ 0.5
                        ect_t(j, i) = 0
                        ectu_t(j, i) = 0
                        GoTo 999911
                    End If
                    Ecc_t(j, i) = 4732.98 * (fprimec_t(j, i)) ^ 0.5
                    ect_t(j, i) = fct_t(j, i) / Ecc_t(j, i)
                    ectu_t(j, i) = 11 * fct_t(j, i) / Ecc_t(j, i)
999911:         Next j
                GoTo 7

                'calcareous aggs
6:              For j = 1 To Elements
                    If cTEMP(j, i) <= 100 Then GoTo 1002
                    If 100 < cTEMP(j, i) And cTEMP(j, i) <= 200 Then GoTo 2002
                    If 200 < cTEMP(j, i) And cTEMP(j, i) <= 300 Then GoTo 3002
                    If 300 < cTEMP(j, i) And cTEMP(j, i) <= 400 Then GoTo 4002
                    If 400 < cTEMP(j, i) And cTEMP(j, i) <= 500 Then GoTo 5002
                    If 500 < cTEMP(j, i) And cTEMP(j, i) <= 600 Then GoTo 6002
                    If 600 < cTEMP(j, i) And cTEMP(j, i) <= 700 Then GoTo 7002
                    If 700 < cTEMP(j, i) And cTEMP(j, i) <= 800 Then GoTo 8002
                    If 800 < cTEMP(j, i) And cTEMP(j, i) <= 900 Then GoTo 9002
                    If 900 < cTEMP(j, i) And cTEMP(j, i) <= 1000 Then GoTo 10002
                    If 1000 < cTEMP(j, i) And cTEMP(j, i) <= 1100 Then GoTo 11002
                    If 1100 < cTEMP(j, i) And cTEMP(j, i) <= 1200 Then GoTo 12002
                    If 1200 < cTEMP(j, i) Then GoTo 13002

1002:               fprimec_t(j, i) = fprimec * (1.0 + (cTEMP(j, i) - 20) * ((1.0 - 1.0) / (100 - 20)))
                    eo_t(j, i) = (0.0025 + (cTEMP(j, i) - 20) * ((0.004 - 0.0025) / (100 - 20)))
```

```
                    ecu_t(j, i) = (0.02 + (cTEMP(j, i) - 20) * ((0.0225 - 0.02) / (100 - 20)))
                    fct_t(j, i) = 1.0 * fct

                    .Cells(j + 1, 26).value = fprimec_t(j, i)
                    .Cells(j + 1, 27).value = eo_t(j, i)
                    .Cells(j + 1, 28).value = ecu_t(j, i)
                    GoTo 99992

2002:               fprimec_t(j, i) = fprimec * (1.0 + (cTEMP(j, i) - 100) * ((0.97 - 1.0) / (200 - 100)))
                    eo_t(j, i) = (0.004 + (cTEMP(j, i) - 100) * ((0.0055 - 0.004) / (200 - 100)))
                    ecu_t(j, i) = (0.0225 + (cTEMP(j, i) - 100) * ((0.025 - 0.0225) / (200 - 100)))
                    fct_t(j, i) = (1.0 - 1.0 * (cTEMP(j, i) - 100) / 500) * fct

                    .Cells(j + 1, 26).value = fprimec_t(j, i)
                    .Cells(j + 1, 27).value = eo_t(j, i)
                    .Cells(j + 1, 28).value = ecu_t(j, i)
                    GoTo 99992

3002:               fprimec_t(j, i) = fprimec * (0.97 + (cTEMP(j, i) - 200) * ((0.91 - 0.97) / (300 - 200)))
                    eo_t(j, i) = (0.0055 + (cTEMP(j, i) - 200) * ((0.007 - 0.0055) / (300 - 200)))
                    ecu_t(j, i) = (0.025 + (cTEMP(j, i) - 200) * ((0.0275 - 0.025) / (300 - 200)))
                    fct_t(j, i) = (1.0 - 1.0 * (cTEMP(j, i) - 100) / 500) * fct

                    .Cells(j + 1, 26).value = fprimec_t(j, i)
                    .Cells(j + 1, 27).value = eo_t(j, i)
                    .Cells(j + 1, 28).value = ecu_t(j, i)
                    GoTo 99992

4002:               fprimec_t(j, i) = fprimec * (0.91 + (cTEMP(j, i) - 300) * ((0.85 - 0.91) / (400 - 300)))
                    eo_t(j, i) = (0.007 + (cTEMP(j, i) - 300) * ((0.01 - 0.007) / (400 - 300)))
                    ecu_t(j, i) = (0.0275 + (cTEMP(j, i) - 300) * ((0.03 - 0.0275) / (400 - 300)))
                    fct_t(j, i) = (1.0 - 1.0 * (cTEMP(j, i) - 100) / 500) * fct

                    .Cells(j + 1, 26).value = fprimec_t(j, i)
                    .Cells(j + 1, 27).value = eo_t(j, i)
                    .Cells(j + 1, 28).value = ecu_t(j, i)
                    GoTo 99992

5002:               fprimec_t(j, i) = fprimec * (0.85 + (cTEMP(j, i) - 400) * ((0.74 - 0.85) / (500 - 400)))
                    eo_t(j, i) = (0.01 + (cTEMP(j, i) - 400) * ((0.015 - 0.01) / (500 - 400)))
                    ecu_t(j, i) = (0.03 + (cTEMP(j, i) - 400) * ((0.0325 - 0.03) / (500 - 400)))
                    fct_t(j, i) = (1.0 - 1.0 * (cTEMP(j, i) - 100) / 500) * fct

                    .Cells(j + 1, 26).value = fprimec_t(j, i)
                    .Cells(j + 1, 27).value = eo_t(j, i)
                    .Cells(j + 1, 28).value = ecu_t(j, i)
                    GoTo 99992

6002:               fprimec_t(j, i) = fprimec * (0.74 + (cTEMP(j, i) - 500) * ((0.6 - 0.74) / (600 - 500)))
                    eo_t(j, i) = (0.015 + (cTEMP(j, i) - 500) * ((0.025 - 0.015) / (600 - 500)))
                    ecu_t(j, i) = (0.0325 + (cTEMP(j, i) - 500) * ((0.035 - 0.0325) /
```

```
    (600 - 500)))
                    fct_t(j, i) = (1.0 - 1.0 * (cTEMP(j, i) - 100) / 500) * fct

                    .Cells(j + 1, 26).value = fprimec_t(j, i)
                    .Cells(j + 1, 27).value = eo_t(j, i)
                    .Cells(j + 1, 28).value = ecu_t(j, i)
                    GoTo 99992

7002:               fprimec_t(j, i) = fprimec * (0.6 + (cTEMP(j, i) - 600) * ((0.43 - 0. ↙
    6) / (700 - 600)))
                    eo_t(j, i) = (0.025 + (cTEMP(j, i) - 600) * ((0.025 - 0.025) / (700 -↙
     600)))
                    ecu_t(j, i) = (0.035 + (cTEMP(j, i) - 600) * ((0.0375 - 0.035) / (700↙
     - 600)))
                    fct_t(j, i) = 0

                    .Cells(j + 1, 26).value = fprimec_t(j, i)
                    .Cells(j + 1, 27).value = eo_t(j, i)
                    .Cells(j + 1, 28).value = ecu_t(j, i)
                    GoTo 99992

8002:               fprimec_t(j, i) = fprimec * (0.43 + (cTEMP(j, i) - 700) * ((0.27 - 0.↙
    43) / (800 - 700)))
                    eo_t(j, i) = (0.025 + (cTEMP(j, i) - 700) * ((0.025 - 0.025) / (800 -↙
     700)))
                    ecu_t(j, i) = (0.0375 + (cTEMP(j, i) - 700) * ((0.04 - 0.0375) / (800↙
     - 700)))
                    fct_t(j, i) = 0

                    .Cells(j + 1, 26).value = fprimec_t(j, i)
                    .Cells(j + 1, 27).value = eo_t(j, i)
                    .Cells(j + 1, 28).value = ecu_t(j, i)
                    GoTo 99992

9002:               fprimec_t(j, i) = fprimec * (0.27 + (cTEMP(j, i) - 800) * ((0.15 - 0.↙
    27) / (900 - 800)))
                    eo_t(j, i) = (0.025 + (cTEMP(j, i) - 800) * ((0.025 - 0.025) / (900 -↙
     800)))
                    ecu_t(j, i) = (0.04 + (cTEMP(j, i) - 800) * ((0.0425 - 0.04) / (900 -↙
     800)))
                    fct_t(j, i) = 0

                    .Cells(j + 1, 26).value = fprimec_t(j, i)
                    .Cells(j + 1, 27).value = eo_t(j, i)
                    .Cells(j + 1, 28).value = ecu_t(j, i)
                    GoTo 99992

10002:              fprimec_t(j, i) = fprimec * (0.15 + (cTEMP(j, i) - 900) * ((0.06 - 0.↙
    15) / (1000 - 900)))
                    eo_t(j, i) = (0.025 + (cTEMP(j, i) - 900) * ((0.025 - 0.025) / (1000 ↙
    - 900)))
                    ecu_t(j, i) = (0.0425 + (cTEMP(j, i) - 900) * ((0.045 - 0.0425) /    ↙
    (1000 - 900)))
                    fct_t(j, i) = 0

                    .Cells(j + 1, 26).value = fprimec_t(j, i)
                    .Cells(j + 1, 27).value = eo_t(j, i)
                    .Cells(j + 1, 28).value = ecu_t(j, i)
                    GoTo 99992

11002:              fprimec_t(j, i) = fprimec * (0.06 + (cTEMP(j, i) - 1000) * ((0.02 - 0↙
    .06) / (1100 - 1000)))
                    eo_t(j, i) = (0.025 + (cTEMP(j, i) - 1000) * ((0.025 - 0.025) / (1100↙
     - 1000)))
                    ecu_t(j, i) = (0.045 + (cTEMP(j, i) - 1000) * ((0.0475 - 0.045) /    ↙
    (1100 - 1000)))
```

```
                    fct_t(j, i) = 0

                    .Cells(j + 1, 26).value = fprimec_t(j, i)
                    .Cells(j + 1, 27).value = eo_t(j, i)
                    .Cells(j + 1, 28).value = ecu_t(j, i)
                    GoTo 99992

12002:              fprimec_t(j, i) = fprimec * (0.02 + (cTEMP(j, i) - 1100) * ((0.001 - 0.02) / (1200 - 1100)))
                    eo_t(j, i) = (0.025 + (cTEMP(j, i) - 1100) * ((0.025 - 0.025) / (1200 - 1100)))
                    ecu_t(j, i) = (0.0475 + (cTEMP(j, i) - 1100) * ((0.05 - 0.0475) / (1200 - 1100)))
                    fct_t(j, i) = 0

                    .Cells(j + 1, 26).value = fprimec_t(j, i)
                    .Cells(j + 1, 27).value = eo_t(j, i)
                    .Cells(j + 1, 28).value = ecu_t(j, i)
                    GoTo 99992

13002:              fprimec_t(j, i) = fprimec * 0.001
                    eo_t(j, i) = 0.025
                    ecu_t(j, i) = 0.05
                    fct_t(j, i) = 0

                    .Cells(j + 1, 26).value = fprimec_t(j, i)
                    .Cells(j + 1, 27).value = eo_t(j, i)
                    .Cells(j + 1, 28).value = ecu_t(j, i)
                    GoTo 99992

99992:              If fct_t(j, i) = 0 Then
                        Ecc_t(j, i) = 0
                        ect_t(j, i) = 0
                        ectu_t(j, i) = 0
                        GoTo 999921
                    End If
                    Ecc_t(j, i) = 4732.98 * (fprimec_t(j, i)) ^ 0.5
                    ect_t(j, i) = fct_t(j, i) / Ecc_t(j, i)
                    ectu_t(j, i) = 11 * fct_t(j, i) / Ecc_t(j, i)
999921:         Next j

                'Steel Reduced Mechanical Properties
7:              For j = 1 To REINFcount
                    If sTEMP(j, i) <= 100 Then GoTo 1003
                    If 100 < sTEMP(j, i) And sTEMP(j, i) <= 200 Then GoTo 2003
                    If 200 < sTEMP(j, i) And sTEMP(j, i) <= 300 Then GoTo 3003
                    If 300 < sTEMP(j, i) And sTEMP(j, i) <= 400 Then GoTo 4003
                    If 400 < sTEMP(j, i) And sTEMP(j, i) <= 500 Then GoTo 5003
                    If 500 < sTEMP(j, i) And sTEMP(j, i) <= 600 Then GoTo 6003
                    If 600 < sTEMP(j, i) And sTEMP(j, i) <= 700 Then GoTo 7003
                    If 700 < sTEMP(j, i) And sTEMP(j, i) <= 800 Then GoTo 8003
                    If 800 < sTEMP(j, i) And sTEMP(j, i) <= 900 Then GoTo 9003
                    If 900 < sTEMP(j, i) And sTEMP(j, i) <= 1000 Then GoTo 10003
                    If 1000 < sTEMP(j, i) And sTEMP(j, i) <= 1100 Then GoTo 11003
                    If 1100 < sTEMP(j, i) And sTEMP(j, i) <= 1200 Then GoTo 12003
                    If 1200 < sTEMP(j, i) Then GoTo 13003

1003:               fp_t(j, i) = fp * (1.0 + (sTEMP(j, i) - 20) * ((1.0 - 1.0) / (100 - 20)))
                    fy_t(j, i) = fy * (1.0 + (sTEMP(j, i) - 20) * ((1.0 - 1.0) / (100 - 20)))
                    Est_t(j, i) = Est * (1.0 + (sTEMP(j, i) - 20) * ((1.0 - 1.0) / (100 - 20)))

                    ep_t(j, i) = fp_t(j, i) / Est_t(j, i)
                    ey_t(j, i) = 0.02
```

```
                    et_t(j, i) = 0.15
                    eu_t(j, i) = 0.2
                    c_t(j, i) = ((fy_t(j, i) - fp_t(j, i)) ^ 2) / ((ey_t(j, i) - ep_t(j, ↙
    i)) * Est_t(j, i) - 2 * (fy_t(j, i) - fp_t(j, i)))
                    a_t(j, i) = ((ey_t(j, i) - ep_t(j, i)) * (ey_t(j, i) - ep_t(j, i) +  ↙
    c_t(j, i) / Est_t(j, i))) ^ 0.5
                    b_t(j, i) = (c_t(j, i) * (ey_t(j, i) - ep_t(j, i)) * Est_t(j, i) +   ↙
    c_t(j, i) ^ 2) ^ 0.5

                    .Cells(j + 1, 45).value = fp_t(j, i)
                    .Cells(j + 1, 46).value = fy_t(j, i)
                    .Cells(j + 1, 47).value = Est_t(j, i)
                    .Cells(j + 1, 48).value = ep_t(j, i)
                    .Cells(j + 1, 49).value = ey_t(j, i)
                    .Cells(j + 1, 50).value = et_t(j, i)
                    .Cells(j + 1, 51).value = eu_t(j, i)
                    GoTo 99993

2003:               fp_t(j, i) = fp * (1.0 + (sTEMP(j, i) - 100) * ((0.81 - 1.0) / (200 -↙
     100)))
                    fy_t(j, i) = fy * (1.0 + (sTEMP(j, i) - 100) * ((1.0 - 1.0) / (200 - ↙
    100)))
                    Est_t(j, i) = Est * (1.0 + (sTEMP(j, i) - 100) * ((0.9 - 1.0) / (200 ↙
    - 100)))

                    ep_t(j, i) = fp_t(j, i) / Est_t(j, i)
                    ey_t(j, i) = 0.02
                    et_t(j, i) = 0.15
                    eu_t(j, i) = 0.2
                    c_t(j, i) = ((fy_t(j, i) - fp_t(j, i)) ^ 2) / ((ey_t(j, i) - ep_t(j, ↙
    i)) * Est_t(j, i) - 2 * (fy_t(j, i) - fp_t(j, i)))
                    a_t(j, i) = ((ey_t(j, i) - ep_t(j, i)) * (ey_t(j, i) - ep_t(j, i) +  ↙
    c_t(j, i) / Est_t(j, i))) ^ 0.5
                    b_t(j, i) = (c_t(j, i) * (ey_t(j, i) - ep_t(j, i)) * Est_t(j, i) +   ↙
    c_t(j, i) ^ 2) ^ 0.5

                    .Cells(j + 1, 45).value = fp_t(j, i)
                    .Cells(j + 1, 46).value = fy_t(j, i)
                    .Cells(j + 1, 47).value = Est_t(j, i)
                    .Cells(j + 1, 48).value = ep_t(j, i)
                    .Cells(j + 1, 49).value = ey_t(j, i)
                    .Cells(j + 1, 50).value = et_t(j, i)
                    .Cells(j + 1, 51).value = eu_t(j, i)
                    GoTo 99993

3003:               fp_t(j, i) = fp * (0.81 + (sTEMP(j, i) - 200) * ((0.61 - 0.81) / (300↙
     - 200)))
                    fy_t(j, i) = fy * (1.0 + (sTEMP(j, i) - 200) * ((1.0 - 1.0) / (300 - ↙
    200)))
                    Est_t(j, i) = Est * (0.9 + (sTEMP(j, i) - 200) * ((0.8 - 0.9) / (300 ↙
    - 200)))

                    ep_t(j, i) = fp_t(j, i) / Est_t(j, i)
                    ey_t(j, i) = 0.02
                    et_t(j, i) = 0.15
                    eu_t(j, i) = 0.2
                    c_t(j, i) = ((fy_t(j, i) - fp_t(j, i)) ^ 2) / ((ey_t(j, i) - ep_t(j, ↙
    i)) * Est_t(j, i) - 2 * (fy_t(j, i) - fp_t(j, i)))
                    a_t(j, i) = ((ey_t(j, i) - ep_t(j, i)) * (ey_t(j, i) - ep_t(j, i) +  ↙
    c_t(j, i) / Est_t(j, i))) ^ 0.5
                    b_t(j, i) = (c_t(j, i) * (ey_t(j, i) - ep_t(j, i)) * Est_t(j, i) +   ↙
    c_t(j, i) ^ 2) ^ 0.5

                    .Cells(j + 1, 45).value = fp_t(j, i)
                    .Cells(j + 1, 46).value = fy_t(j, i)
                    .Cells(j + 1, 47).value = Est_t(j, i)
```

```
                    .Cells(j + 1, 48).value = ep_t(j, i)
                    .Cells(j + 1, 49).value = ey_t(j, i)
                    .Cells(j + 1, 50).value = et_t(j, i)
                    .Cells(j + 1, 51).value = eu_t(j, i)
                    GoTo 99993

4003:               fp_t(j, i) = fp * (0.61 + (sTEMP(j, i) - 300) * ((0.42 - 0.61) / (400↙
     - 300)))
                    fy_t(j, i) = fy * (1.0 + (sTEMP(j, i) - 300) * ((1.0 - 1.0) / (400 - ↙
    300)))
                    Est_t(j, i) = Est * (0.8 + (sTEMP(j, i) - 300) * ((0.7 - 0.8) / (400 ↙
    - 300)))

                    ep_t(j, i) = fp_t(j, i) / Est_t(j, i)
                    ey_t(j, i) = 0.02
                    et_t(j, i) = 0.15
                    eu_t(j, i) = 0.2
                    c_t(j, i) = ((fy_t(j, i) - fp_t(j, i)) ^ 2) / ((ey_t(j, i) - ep_t(j, ↙
    i)) * Est_t(j, i) - 2 * (fy_t(j, i) - fp_t(j, i)))
                    a_t(j, i) = ((ey_t(j, i) - ep_t(j, i)) * (ey_t(j, i) - ep_t(j, i) +  ↙
    c_t(j, i) / Est_t(j, i))) ^ 0.5
                    b_t(j, i) = (c_t(j, i) * (ey_t(j, i) - ep_t(j, i)) * Est_t(j, i) +   ↙
    c_t(j, i) ^ 2) ^ 0.5

                    .Cells(j + 1, 45).value = fp_t(j, i)
                    .Cells(j + 1, 46).value = fy_t(j, i)
                    .Cells(j + 1, 47).value = Est_t(j, i)
                    .Cells(j + 1, 48).value = ep_t(j, i)
                    .Cells(j + 1, 49).value = ey_t(j, i)
                    .Cells(j + 1, 50).value = et_t(j, i)
                    .Cells(j + 1, 51).value = eu_t(j, i)
                    GoTo 99993

5003:               fp_t(j, i) = fp * (0.42 + (sTEMP(j, i) - 400) * ((0.36 - 0.42) / (500↙
     - 400)))
                    fy_t(j, i) = fy * (1.0 + (sTEMP(j, i) - 400) * ((0.78 - 1.0) / (500 -↙
     400)))
                    Est_t(j, i) = Est * (0.7 + (sTEMP(j, i) - 400) * ((0.6 - 0.7) / (500 ↙
    - 400)))

                    ep_t(j, i) = fp_t(j, i) / Est_t(j, i)
                    ey_t(j, i) = 0.02
                    et_t(j, i) = 0.15
                    eu_t(j, i) = 0.2
                    c_t(j, i) = ((fy_t(j, i) - fp_t(j, i)) ^ 2) / ((ey_t(j, i) - ep_t(j, ↙
    i)) * Est_t(j, i) - 2 * (fy_t(j, i) - fp_t(j, i)))
                    a_t(j, i) = ((ey_t(j, i) - ep_t(j, i)) * (ey_t(j, i) - ep_t(j, i) +  ↙
    c_t(j, i) / Est_t(j, i))) ^ 0.5
                    b_t(j, i) = (c_t(j, i) * (ey_t(j, i) - ep_t(j, i)) * Est_t(j, i) +   ↙
    c_t(j, i) ^ 2) ^ 0.5

                    .Cells(j + 1, 45).value = fp_t(j, i)
                    .Cells(j + 1, 46).value = fy_t(j, i)
                    .Cells(j + 1, 47).value = Est_t(j, i)
                    .Cells(j + 1, 48).value = ep_t(j, i)
                    .Cells(j + 1, 49).value = ey_t(j, i)
                    .Cells(j + 1, 50).value = et_t(j, i)
                    .Cells(j + 1, 51).value = eu_t(j, i)
                    GoTo 99993

6003:               fp_t(j, i) = fp * (0.36 + (sTEMP(j, i) - 500) * ((0.18 - 0.36) / (600↙
     - 500)))
                    fy_t(j, i) = fy * (0.78 + (sTEMP(j, i) - 500) * ((0.47 - 0.78) / (600↙
     - 500)))
                    Est_t(j, i) = Est * (0.6 + (sTEMP(j, i) - 500) * ((0.31 - 0.6) / (600↙
     - 500)))
```

```
                    ep_t(j, i) = fp_t(j, i) / Est_t(j, i)
                    ey_t(j, i) = 0.02
                    et_t(j, i) = 0.15
                    eu_t(j, i) = 0.2
                    c_t(j, i) = ((fy_t(j, i) - fp_t(j, i)) ^ 2) / ((ey_t(j, i) - ep_t(j, ↙
    i)) * Est_t(j, i) - 2 * (fy_t(j, i) - fp_t(j, i)))
                    a_t(j, i) = ((ey_t(j, i) - ep_t(j, i)) * (ey_t(j, i) - ep_t(j, i) +  ↙
    c_t(j, i) / Est_t(j, i))) ^ 0.5
                    b_t(j, i) = (c_t(j, i) * (ey_t(j, i) - ep_t(j, i)) * Est_t(j, i) +  ↙
    c_t(j, i) ^ 2) ^ 0.5

                    .Cells(j + 1, 45).value = fp_t(j, i)
                    .Cells(j + 1, 46).value = fy_t(j, i)
                    .Cells(j + 1, 47).value = Est_t(j, i)
                    .Cells(j + 1, 48).value = ep_t(j, i)
                    .Cells(j + 1, 49).value = ey_t(j, i)
                    .Cells(j + 1, 50).value = et_t(j, i)
                    .Cells(j + 1, 51).value = eu_t(j, i)
                    GoTo 99993

7003:               fp_t(j, i) = fp * (0.18 + (sTEMP(j, i) - 600) * ((0.07 - 0.18) / (700↙
     - 600)))
                    fy_t(j, i) = fy * (0.47 + (sTEMP(j, i) - 600) * ((0.23 - 0.47) / (700↙
     - 600)))
                    Est_t(j, i) = Est * (0.31 + (sTEMP(j, i) - 600) * ((0.13 - 0.31) /  ↙
    (700 - 600)))

                    ep_t(j, i) = fp_t(j, i) / Est_t(j, i)
                    ey_t(j, i) = 0.02
                    et_t(j, i) = 0.15
                    eu_t(j, i) = 0.2
                    c_t(j, i) = ((fy_t(j, i) - fp_t(j, i)) ^ 2) / ((ey_t(j, i) - ep_t(j, ↙
    i)) * Est_t(j, i) - 2 * (fy_t(j, i) - fp_t(j, i)))
                    a_t(j, i) = ((ey_t(j, i) - ep_t(j, i)) * (ey_t(j, i) - ep_t(j, i) +  ↙
    c_t(j, i) / Est_t(j, i))) ^ 0.5
                    b_t(j, i) = (c_t(j, i) * (ey_t(j, i) - ep_t(j, i)) * Est_t(j, i) +  ↙
    c_t(j, i) ^ 2) ^ 0.5

                    .Cells(j + 1, 45).value = fp_t(j, i)
                    .Cells(j + 1, 46).value = fy_t(j, i)
                    .Cells(j + 1, 47).value = Est_t(j, i)
                    .Cells(j + 1, 48).value = ep_t(j, i)
                    .Cells(j + 1, 49).value = ey_t(j, i)
                    .Cells(j + 1, 50).value = et_t(j, i)
                    .Cells(j + 1, 51).value = eu_t(j, i)
                    GoTo 99993

8003:               fp_t(j, i) = fp * (0.07 + (sTEMP(j, i) - 700) * ((0.05 - 0.07) / (800↙
     - 700)))
                    fy_t(j, i) = fy * (0.23 + (sTEMP(j, i) - 700) * ((0.11 - 0.23) / (800↙
     - 700)))
                    Est_t(j, i) = Est * (0.13 + (sTEMP(j, i) - 700) * ((0.09 - 0.13) /  ↙
    (800 - 700)))

                    ep_t(j, i) = fp_t(j, i) / Est_t(j, i)
                    ey_t(j, i) = 0.02
                    et_t(j, i) = 0.15
                    eu_t(j, i) = 0.2
                    c_t(j, i) = ((fy_t(j, i) - fp_t(j, i)) ^ 2) / ((ey_t(j, i) - ep_t(j, ↙
    i)) * Est_t(j, i) - 2 * (fy_t(j, i) - fp_t(j, i)))
                    a_t(j, i) = ((ey_t(j, i) - ep_t(j, i)) * (ey_t(j, i) - ep_t(j, i) +  ↙
    c_t(j, i) / Est_t(j, i))) ^ 0.5
                    b_t(j, i) = (c_t(j, i) * (ey_t(j, i) - ep_t(j, i)) * Est_t(j, i) +  ↙
    c_t(j, i) ^ 2) ^ 0.5
```

```
                    .Cells(j + 1, 45).value = fp_t(j, i)
                    .Cells(j + 1, 46).value = fy_t(j, i)
                    .Cells(j + 1, 47).value = Est_t(j, i)
                    .Cells(j + 1, 48).value = ep_t(j, i)
                    .Cells(j + 1, 49).value = ey_t(j, i)
                    .Cells(j + 1, 50).value = et_t(j, i)
                    .Cells(j + 1, 51).value = eu_t(j, i)
                    GoTo 99993

9003:               fp_t(j, i) = fp * (0.05 + (sTEMP(j, i) - 800) * ((0.04 - 0.05) / (900↙
     - 800)))
                    fy_t(j, i) = fy * (0.11 + (sTEMP(j, i) - 800) * ((0.06 - 0.11) / (900↙
     - 800)))
                    Est_t(j, i) = Est * (0.09 + (sTEMP(j, i) - 800) * ((0.07 - 0.09) /   ↙
    (900 - 800)))

                    ep_t(j, i) = fp_t(j, i) / Est_t(j, i)
                    ey_t(j, i) = 0.02
                    et_t(j, i) = 0.15
                    eu_t(j, i) = 0.2
                    c_t(j, i) = ((fy_t(j, i) - fp_t(j, i)) ^ 2) / ((ey_t(j, i) - ep_t(j, ↙
    i)) * Est_t(j, i) - 2 * (fy_t(j, i) - fp_t(j, i)))
                    a_t(j, i) = ((ey_t(j, i) - ep_t(j, i)) * (ey_t(j, i) - ep_t(j, i) +  ↙
    c_t(j, i) / Est_t(j, i))) ^ 0.5
                    b_t(j, i) = (c_t(j, i) * (ey_t(j, i) - ep_t(j, i)) * Est_t(j, i) +   ↙
    c_t(j, i) ^ 2) ^ 0.5

                    .Cells(j + 1, 45).value = fp_t(j, i)
                    .Cells(j + 1, 46).value = fy_t(j, i)
                    .Cells(j + 1, 47).value = Est_t(j, i)
                    .Cells(j + 1, 48).value = ep_t(j, i)
                    .Cells(j + 1, 49).value = ey_t(j, i)
                    .Cells(j + 1, 50).value = et_t(j, i)
                    .Cells(j + 1, 51).value = eu_t(j, i)
                    GoTo 99993

10003:              fp_t(j, i) = fp * (0.04 + (sTEMP(j, i) - 900) * ((0.02 - 0.04) /     ↙
    (1000 - 900)))
                    fy_t(j, i) = fy * (0.06 + (sTEMP(j, i) - 900) * ((0.04 - 0.06) /     ↙
    (1000 - 900)))
                    Est_t(j, i) = Est * (0.07 + (sTEMP(j, i) - 900) * ((0.04 - 0.07) /   ↙
    (1000 - 900)))

                    ep_t(j, i) = fp_t(j, i) / Est_t(j, i)
                    ey_t(j, i) = 0.02
                    et_t(j, i) = 0.15
                    eu_t(j, i) = 0.2
                    c_t(j, i) = ((fy_t(j, i) - fp_t(j, i)) ^ 2) / ((ey_t(j, i) - ep_t(j, ↙
    i)) * Est_t(j, i) - 2 * (fy_t(j, i) - fp_t(j, i)))
                    a_t(j, i) = ((ey_t(j, i) - ep_t(j, i)) * (ey_t(j, i) - ep_t(j, i) +  ↙
    c_t(j, i) / Est_t(j, i))) ^ 0.5
                    b_t(j, i) = (c_t(j, i) * (ey_t(j, i) - ep_t(j, i)) * Est_t(j, i) +   ↙
    c_t(j, i) ^ 2) ^ 0.5

                    .Cells(j + 1, 45).value = fp_t(j, i)
                    .Cells(j + 1, 46).value = fy_t(j, i)
                    .Cells(j + 1, 47).value = Est_t(j, i)
                    .Cells(j + 1, 48).value = ep_t(j, i)
                    .Cells(j + 1, 49).value = ey_t(j, i)
                    .Cells(j + 1, 50).value = et_t(j, i)
                    .Cells(j + 1, 51).value = eu_t(j, i)
                    GoTo 99993

11003:              fp_t(j, i) = fp * (0.02 + (sTEMP(j, i) - 1000) * ((0.01 - 0.02) /    ↙
    (1100 - 1000)))
                    fy_t(j, i) = fy * (0.04 + (sTEMP(j, i) - 1000) * ((0.02 - 0.04) /    ↙
```

```
    (1100 - 1000)))
                    Est_t(j, i) = Est * (0.04 + (sTEMP(j, i) - 1000) * ((0.02 - 0.04) /
    (1100 - 1000)))

                    ep_t(j, i) = fp_t(j, i) / Est_t(j, i)
                    ey_t(j, i) = 0.02
                    et_t(j, i) = 0.15
                    eu_t(j, i) = 0.2
                    c_t(j, i) = ((fy_t(j, i) - fp_t(j, i)) ^ 2) / ((ey_t(j, i) - ep_t(j,
    i)) * Est_t(j, i) - 2 * (fy_t(j, i) - fp_t(j, i)))
                    a_t(j, i) = ((ey_t(j, i) - ep_t(j, i)) * (ey_t(j, i) - ep_t(j, i) +
    c_t(j, i) / Est_t(j, i))) ^ 0.5
                    b_t(j, i) = (c_t(j, i) * (ey_t(j, i) - ep_t(j, i)) * Est_t(j, i) +
    c_t(j, i) ^ 2) ^ 0.5

                    .Cells(j + 1, 45).value = fp_t(j, i)
                    .Cells(j + 1, 46).value = fy_t(j, i)
                    .Cells(j + 1, 47).value = Est_t(j, i)
                    .Cells(j + 1, 48).value = ep_t(j, i)
                    .Cells(j + 1, 49).value = ey_t(j, i)
                    .Cells(j + 1, 50).value = et_t(j, i)
                    .Cells(j + 1, 51).value = eu_t(j, i)
                    GoTo 99993

12003:              fp_t(j, i) = fp * (0.01 + (sTEMP(j, i) - 1100) * ((0.001 - 0.01) /
    (1200 - 1100)))
                    fy_t(j, i) = fy * (0.02 + (sTEMP(j, i) - 1100) * ((0.001 - 0.02) /
    (1200 - 1100)))
                    Est_t(j, i) = Est * (0.02 + (sTEMP(j, i) - 1100) * ((0.001 - 0.02) /
    (1200 - 1100)))

                    ep_t(j, i) = fp_t(j, i) / Est_t(j, i)
                    ey_t(j, i) = 0.02
                    et_t(j, i) = 0.15
                    eu_t(j, i) = 0.2
                    c_t(j, i) = ((fy_t(j, i) - fp_t(j, i)) ^ 2) / ((ey_t(j, i) - ep_t(j,
    i)) * Est_t(j, i) - 2 * (fy_t(j, i) - fp_t(j, i)))
                    a_t(j, i) = ((ey_t(j, i) - ep_t(j, i)) * (ey_t(j, i) - ep_t(j, i) +
    c_t(j, i) / Est_t(j, i))) ^ 0.5
                    b_t(j, i) = (c_t(j, i) * (ey_t(j, i) - ep_t(j, i)) * Est_t(j, i) +
    c_t(j, i) ^ 2) ^ 0.5

                    .Cells(j + 1, 45).value = fp_t(j, i)
                    .Cells(j + 1, 46).value = fy_t(j, i)
                    .Cells(j + 1, 47).value = Est_t(j, i)
                    .Cells(j + 1, 48).value = ep_t(j, i)
                    .Cells(j + 1, 49).value = ey_t(j, i)
                    .Cells(j + 1, 50).value = et_t(j, i)
                    .Cells(j + 1, 51).value = eu_t(j, i)
                    GoTo 99993

13003:              fp_t(j, i) = fp * 0.001
                    fy_t(j, i) = fy * 0.001
                    Est_t(j, i) = Est * 0.001

                    ep_t(j, i) = fp_t(j, i) / Est_t(j, i)
                    ey_t(j, i) = 0.02
                    et_t(j, i) = 0.15
                    eu_t(j, i) = 0.2
                    c_t(j, i) = ((fy_t(j, i) - fp_t(j, i)) ^ 2) / ((ey_t(j, i) - ep_t(j,
    i)) * Est_t(j, i) - 2 * (fy_t(j, i) - fp_t(j, i)))
                    a_t(j, i) = ((ey_t(j, i) - ep_t(j, i)) * (ey_t(j, i) - ep_t(j, i) +
    c_t(j, i) / Est_t(j, i))) ^ 0.5
                    b_t(j, i) = (c_t(j, i) * (ey_t(j, i) - ep_t(j, i)) * Est_t(j, i) +
    c_t(j, i) ^ 2) ^ 0.5
```

```
                    .Cells(j + 1, 45).value = fp_t(j, i)
                    .Cells(j + 1, 46).value = fy_t(j, i)
                    .Cells(j + 1, 47).value = Est_t(j, i)
                    .Cells(j + 1, 48).value = ep_t(j, i)
                    .Cells(j + 1, 49).value = ey_t(j, i)
                    .Cells(j + 1, 50).value = et_t(j, i)
                    .Cells(j + 1, 51).value = eu_t(j, i)
                    GoTo 99993
99993:          Next j
            End With

            If xlREOUT.Cells(20, 2).value = 12 Then GoTo 1993

            'MOMENT-CURVATURE
            txtRUN.Text = "Moment-Curvature" & " at t = " & i * TimeStep
            With xlMPhi
                .Cells.HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter

                Accuracy = .Cells(4, 1).value

                eTOP_min = -0.01
                eTOP_max = 0.01

                eTOP_step = (eTOP_max - eTOP_min) / mphiCOUNT
                eTOP(0, i) = eTOP_min - eTOP_step

                c_1 = 0.2 * s_height
                c_2 = 0.8 * s_height
                c_previous = 0.5 * s_height 'in case first iteration goes past 100000 (or↙
     whatever limit is set at)


                For j = 0 To mphiCOUNT + 1
                    If j = 0 Then j = 1
                    timer_j.Text = j
                    eTOP(j, i) = eTOP(j - 1, i) + eTOP_step
                    If Abs(eTOP(j, i)) < 0.000001 Then eTOP(j, i) = 0.000001
                    If Accuracy > 1 And Abs(eTOP(j, i)) < 0.0005 Then eTOP(j, i) = 0.0005

                    If j = 1 Then GoTo 99
                    c_1 = 0.8 * c_previous
                    c_2 = 1.2 * c_previous

99:                 DUMMy = 0
                    sumC = 0
                    sumS = 0
                    sumMC = 0
                    sumMS = 0

                    'guess c until convergence
100:                DUMMy = DUMMy + 1
                    timer_iteration.Text = DUMMy

                    'c_1_____________________________________________________________
                    phi_1 = eTOP(j, i) / c_1
                    For k = 1 To Elements
                        yCprime(k, j) = element_Y(k) - (Yaxis_max - c_1)
                        ec(k, j) = phi_1 * yCprime(k, j)
                        If ec(k, j) > 0 Then GoTo 101
                        If ec(k, j) < ectu_t(k, i) Then fc(k, j) = 0
                        If ectu_t(k, i) < ec(k, j) And ec(k, j) < ect_t(k, i) Then fc(k, ↙
    j) = Ecc_t(k, i) * ectu_t(k, i) / 10 - Ecc_t(k, i) * ec(k, j) / 10
                        If ect_t(k, i) < ec(k, j) And ec(k, j) < 0 Then fc(k, j) = Ecc_t ↙
    (k, i) * ec(k, j)
                        GoTo 102
```

```
101:                    If ec(k, j) <= eo_t(k, i) Then fc(k, j) = 3 * ec(k, j) *
    fprimec_t(k, i) / (eo_t(k, i) * (2 + (ec(k, j) / eo_t(k, i)) ^ 3))
                        If eo_t(k, i) < ec(k, j) And ec(k, j) <= ecu_t(k, i) Then fc(k,
    j) = fprimec_t(k, i) - (-fprimec_t(k, i) / (ecu_t(k, i) - eo_t(k, i))) * eo_t(k, i) +
     (-fprimec_t(k, i) / (ecu_t(k, i) - eo_t(k, i))) * ec(k, j)
                        If ec(k, j) > ecu_t(k, i) Then fc(k, j) = 0

102:                    FORc(k, j) = fc(k, j) * elementAREA(k) / 1000
                        sumC = sumC + FORc(k, j)
                        MOMc(k, j) = FORc(k, j) * element_Y(k) / 1000
                        sumMC = sumMC + MOMc(k, j)
                    Next k
                    For k = 1 To REINFcount
                        ySprime(k, j) = reinf_Y(k) - (Yaxis_max - c_1)
                        es(k, j) = phi_1 * ySprime(k, j)
                        If Abs(es(k, j)) < ep_t(k, i) Then fs(k, j) = es(k, j) * Est_t(k,
     i)
                        If ep_t(k, i) <= Abs(es(k, j)) And Abs(es(k, j)) <= ey_t(k, i)
    Then fs(k, j) = (fp_t(k, i) - c_t(k, i) + (b_t(k, i) / a_t(k, i)) * (a_t(k, i) ^ 2 -
    (ey_t(k, i) - Abs(es(k, j))) ^ 2) ^ 0.5) * es(k, j) / Abs(es(k, j))
                        If ey_t(k, i) <= Abs(es(k, j)) And Abs(es(k, j)) <= et_t(k, i)
    Then fs(k, j) = (fy_t(k, i)) * es(k, j) / Abs(es(k, j))
                        If et_t(k, i) <= Abs(es(k, j)) And Abs(es(k, j)) <= eu_t(k, i)
    Then fs(k, j) = (fy_t(k, i) * (1 - (Abs(es(k, j)) - et_t(k, i)) / (eu_t(k, i) - et_t
    (k, i)))) * es(k, j) / Abs(es(k, j))
                        If Abs(es(k, j)) > eu_t(k, i) Then fs(k, j) = 0
                        FORs(k, j) = fs(k, j) * reinf_AREA(k) / 1000
                        sumS = sumS + FORs(k, j)
                        MOMs(k, j) = FORs(k, j) * reinf_Y(k) / 1000
                        sumMS = sumMS + MOMs(k, j)
                    Next k
                    N_1 = sumC + sumS
                    M_1 = sumMC + sumMS

                    sumC = 0
                    sumS = 0
                    sumMC = 0
                    sumMS = 0

                    'c_2_____________________________________________________________
                    phi_2 = eTOP(j, i) / c_2
                    For k = 1 To Elements
                        yCprime(k, j) = element_Y(k) - (Yaxis_max - c_2)
                        ec(k, j) = phi_2 * yCprime(k, j)
                        If ec(k, j) > 0 Then GoTo 111
                        If ec(k, j) < ectu_t(k, i) Then fc(k, j) = 0
                        If ectu_t(k, i) < ec(k, j) And ec(k, j) < ect_t(k, i) Then fc(k,
    j) = Ecc_t(k, i) * ectu_t(k, i) / 10 - Ecc_t(k, i) * ec(k, j) / 10
                        If ect_t(k, i) < ec(k, j) And ec(k, j) < 0 Then fc(k, j) = Ecc_t
    (k, i) * ec(k, j)
                        GoTo 112

111:                    If ec(k, j) <= eo_t(k, i) Then fc(k, j) = 3 * ec(k, j) *
    fprimec_t(k, i) / (eo_t(k, i) * (2 + (ec(k, j) / eo_t(k, i)) ^ 3))
                        If eo_t(k, i) < ec(k, j) And ec(k, j) <= ecu_t(k, i) Then fc(k,
    j) = fprimec_t(k, i) - (-fprimec_t(k, i) / (ecu_t(k, i) - eo_t(k, i))) * eo_t(k, i) +
     (-fprimec_t(k, i) / (ecu_t(k, i) - eo_t(k, i))) * ec(k, j)
                        If ec(k, j) > ecu_t(k, i) Then fc(k, j) = 0

112:                    FORc(k, j) = fc(k, j) * elementAREA(k) / 1000
                        sumC = sumC + FORc(k, j)
                        MOMc(k, j) = FORc(k, j) * element_Y(k) / 1000
                        sumMC = sumMC + MOMc(k, j)
                    Next k
                    For k = 1 To REINFcount
                        ySprime(k, j) = reinf_Y(k) - (Yaxis_max - c_2)
```

```
                    es(k, j) = phi_2 * ySprime(k, j)
                    If Abs(es(k, j)) < ep_t(k, i) Then fs(k, j) = es(k, j) * Est_t(k, i)
                    If ep_t(k, i) <= Abs(es(k, j)) And Abs(es(k, j)) <= ey_t(k, i) Then fs(k, j) = (fp_t(k, i) - c_t(k, i) + (b_t(k, i) / a_t(k, i)) * (a_t(k, i) ^ 2 - (ey_t(k, i) - Abs(es(k, j))) ^ 2) ^ 0.5) * es(k, j) / Abs(es(k, j))
                    If ey_t(k, i) <= Abs(es(k, j)) And Abs(es(k, j)) <= et_t(k, i) Then fs(k, j) = (fy_t(k, i)) * es(k, j) / Abs(es(k, j))
                    If et_t(k, i) <= Abs(es(k, j)) And Abs(es(k, j)) <= eu_t(k, i) Then fs(k, j) = (fy_t(k, i) * (1 - (Abs(es(k, j)) - et_t(k, i)) / (eu_t(k, i) - et_t(k, i)))) * es(k, j) / Abs(es(k, j))
                    If Abs(es(k, j)) > eu_t(k, i) Then fs(k, j) = 0
                    FORs(k, j) = fs(k, j) * reinf_AREA(k) / 1000
                    sumS = sumS + FORs(k, j)
                    MOMs(k, j) = FORs(k, j) * reinf_Y(k) / 1000
                    sumMS = sumMS + MOMs(k, j)
                Next k
                N_2 = sumC + sumS
                M_2 = sumMC + sumMS

                sumC = 0
                sumS = 0
                sumMC = 0
                sumMS = 0

                c_3 = c_1 + (Accuracy - N_1) * (c_2 - c_1) / (N_2 - N_1)

                'c_3_____________________________________________________________
                phi_3 = eTOP(j, i) / c_3
                For k = 1 To Elements
                    yCprime(k, j) = element_Y(k) - (Yaxis_max - c_3)
                    ec(k, j) = phi_3 * yCprime(k, j)
                    If ec(k, j) > 0 Then GoTo 121
                    If ec(k, j) < ectu_t(k, i) Then fc(k, j) = 0
                    If ectu_t(k, i) < ec(k, j) And ec(k, j) < ect_t(k, i) Then fc(k, j) = Ecc_t(k, i) * ectu_t(k, i) / 10 - Ecc_t(k, i) * ec(k, j) / 10
                    If ect_t(k, i) < ec(k, j) And ec(k, j) < 0 Then fc(k, j) = Ecc_t(k, i) * ec(k, j)
                    GoTo 122

121:                If ec(k, j) <= eo_t(k, i) Then fc(k, j) = 3 * ec(k, j) * fprimec_t(k, i) / (eo_t(k, i) * (2 + (ec(k, j) / eo_t(k, i)) ^ 3))
                    If eo_t(k, i) < ec(k, j) And ec(k, j) <= ecu_t(k, i) Then fc(k, j) = fprimec_t(k, i) - (-fprimec_t(k, i) / (ecu_t(k, i) - eo_t(k, i))) * eo_t(k, i) + (-fprimec_t(k, i) / (ecu_t(k, i) - eo_t(k, i))) * ec(k, j)
                    If ec(k, j) > ecu_t(k, i) Then fc(k, j) = 0

122:                FORc(k, j) = fc(k, j) * elementAREA(k) / 1000
                    sumC = sumC + FORc(k, j)
                    MOMc(k, j) = FORc(k, j) * element_Y(k) / 1000
                    sumMC = sumMC + MOMc(k, j)
                Next k
                For k = 1 To REINFcount
                    ySprime(k, j) = reinf_Y(k) - (Yaxis_max - c_3)
                    es(k, j) = phi_3 * ySprime(k, j)
                    If Abs(es(k, j)) < ep_t(k, i) Then fs(k, j) = es(k, j) * Est_t(k, i)
                    If ep_t(k, i) <= Abs(es(k, j)) And Abs(es(k, j)) <= ey_t(k, i) Then fs(k, j) = (fp_t(k, i) - c_t(k, i) + (b_t(k, i) / a_t(k, i)) * (a_t(k, i) ^ 2 - (ey_t(k, i) - Abs(es(k, j))) ^ 2) ^ 0.5) * es(k, j) / Abs(es(k, j))
                    If ey_t(k, i) <= Abs(es(k, j)) And Abs(es(k, j)) <= et_t(k, i) Then fs(k, j) = (fy_t(k, i)) * es(k, j) / Abs(es(k, j))
                    If et_t(k, i) <= Abs(es(k, j)) And Abs(es(k, j)) <= eu_t(k, i) Then fs(k, j) = (fy_t(k, i) * (1 - (Abs(es(k, j)) - et_t(k, i)) / (eu_t(k, i) - et_t(k, i)))) * es(k, j) / Abs(es(k, j))
                    If Abs(es(k, j)) > eu_t(k, i) Then fs(k, j) = 0
```

```
                    FORs(k, j) = fs(k, j) * reinf_AREA(k) / 1000
                    sumS = sumS + FORs(k, j)
                    MOMs(k, j) = FORs(k, j) * reinf_Y(k) / 1000
                    sumMS = sumMS + MOMs(k, j)
                Next k
                N_3 = sumC + sumS
                M_3 = sumMC + sumMS

                sumC = 0
                sumS = 0
                sumMC = 0
                sumMS = 0

                timer_c.Text = Round(c_3, 2)
                timer_N.Text = Round(N_3, 2)
                timer_M.Text = Round(M_3, 2)
                timer_phi.Text = Round(phi_3, 7)


                If N_3 - 0.1 < Accuracy And N_3 + 0.1 > Accuracy Then GoTo 199

                If DUMMy >= 50 Then
                    eTOP(j, i) = eTOP(j, i) + eTOP_step
                    GoTo 100
                End If


                If N_1 * N_3 < 0 Then
                    c_1 = c_1
                    c_2 = c_3
                    GoTo 100
                End If

                If N_1 * N_3 >= 0 Then
                    c_1 = c_3
                    c_2 = c_2
                    GoTo 100
                End If

                'PrintPlotPoint_____________________________________________
199:            .Cells(1, 4 * i + 3).value = i * TimeStep
                .Cells(1, 4 * i + 4).value = "Phi"
                .Columns(4 * i + 4).NumberFormat = "0.00E+00"
                .Columns(4 * i + 4).ColumnWidth = 12
                .Cells(1, 4 * i + 5).value = "Mom"
                .Columns(4 * i + 5).NumberFormat = "0"
                .Columns(4 * i + 6).ColumnWidth = 6

                c_previous = c_3

                If eTOP(j, i) > eTOP_max Then
                    If i > 0 Then
                        eTOP(j, i) = eTOP(j, i - 1) / 2
                    End If
                End If

                If M_3 = 0 Then
                    j = j - 1
                    eTOP(j, i) = eTOP(j, i) + eTOP_step
                    GoTo 1991
                End If
                If Abs(phi_3) > 0.1 Then
                    j = j - 1
                    eTOP(j, i) = eTOP(j, i) + eTOP_step
                    GoTo 1991
                End If
```

```
                    If M_3 < 0 Then
                        If M_3 > .Cells(2, 4 * i + 3).value Then
                            If soop = False Then
                                soop = True
                                GoTo 1990
                            End If
                            If soop = True Then
                                GoTo 1990
                            End If
                        End If
                        .Cells(2, 4 * i + 3).value = M_3
                        j = j - 1
                        eTOP(j, i) = eTOP(j, i) + eTOP_step
                        GoTo 1991
                    End If

1990:               If M_3 > 0 Then
                        If j > 1 Then
                            If M_3 < .Cells(j, 4 * i + 5).value Then
                                GoTo 1993
                            End If
                        End If
                    End If

                    If i = 0 Then
                        If phi_3 < phi_min Then
                            phi_min = phi_3
                        End If
                        If phi_3 > phi_max Then
                            phi_max = phi_3
                        End If
                    End If
                    If i > 0 Then
                        If phi_3 < phi_min Then
                            j = j - 1
                            eTOP(j, i) = eTOP(j, i) + eTOP_step
                            GoTo 1991
                        End If
                        If phi_3 > phi_max Then
                            GoTo 1993
                        End If
                    End If

                    .Cells(j + 1, 4 * i + 4).value = phi_3
                    .Cells(j + 1, 4 * i + 5).value = M_3

                    If M_3 > MOMmax(i) Then
                        MOMmax(i) = M_3
                    End If
                    If M_3 < MOMmin(i) Then
                        MOMmin(i) = M_3
                    End If
1991:           Next j
            End With
1993:   Next i
        'END_BIG_LOOP&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
    &&&&&&&


        Dim xlCharts As Excel.ChartObjects
        Dim myChart As Excel.ChartObject

        If xlREOUT.Cells(20, 2).value = 12 Then GoTo 909

        'Plot Moment Curvature
```

```
        Dim MPHIplot As Excel.Chart
        Dim MPHI_SerCol As Excel.SeriesCollection
        Dim MPhi0_Series As Excel.Series
        Dim MPHI_AxisCategory, MPHI_AxisValue As Excel.Axes

        xlCharts = xlMPhi.ChartObjects
        myChart = xlCharts.Add(306, 100, 900, 600)
        MPHIplot = myChart.Chart
        MPHIplot.ChartType = Excel.XlChartType.xlXYScatterLinesNoMarkers
        MPHIplot.ChartStyle = 3
        MPHI_SerCol = MPHIplot.SeriesCollection

        MPhi0_Series = MPHI_SerCol.NewSeries
        MPhi0_Series.Name = xlMPhi.Cells(1, 3).value
        MPhi0_Series.XValues = xlMPhi.Range("D2:D" & mphiCOUNT + 2)
        MPhi0_Series.Values = xlMPhi.Range("E2:E" & mphiCOUNT + 2)
        MPhi0_Series.Format.Line.Weight = 1.5

        With MPHIplot
            .Legend.Position = Excel.XlLegendPosition.xlLegendPositionTop
            .HasTitle = True
            .ChartTitle.Text = "Moment-Curvature"

            MPHI_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
            MPHI_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = True
            MPHI_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Text↙
    () = "Curvature, rad/mm"
            MPHI_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Font↙
    .Bold = False
            MPHI_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Font↙
    .Size = 14
            MPHI_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines = True
            MPHI_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines = True
            MPHI_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format.   ↙
    Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            MPHI_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScaleIsAuto = True
            MPHI_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition = Excel↙
    .XlTickLabelPosition.xlTickLabelPositionLow
            MPHI_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold =   ↙
    False
            MPHI_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size = 14

            MPHI_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
            MPHI_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = True
            MPHI_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Text() =  ↙
    "Moment, kN-m"
            MPHI_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Bold ↙
    = False
            MPHI_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Size ↙
    = 14
            MPHI_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
            MPHI_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
            MPHI_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line.    ↙
    DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            MPHI_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScaleIsAuto = True
            MPHI_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.     ↙
    XlTickLabelPosition.xlTickLabelPositionLow
            MPHI_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold = False
            MPHI_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
        End With
        MPHIplot.Refresh()


        'Plot Max Moment Envelope Curve
        With xlScratch
```

```
            .Range("AH1:AJ1000").Clear()
            .Columns(34).NumberFormat = "0.00"
            .Columns(35).NumberFormat = "0"
            .Columns(36).NumberFormat = "0"

            For i = 0 To NumSteps
                .Cells(i + 1, 34).value = time(i) / 3600
                .Cells(i + 1, 35).value = MOMmax(i)
                .Cells(i + 1, 36).value = Abs(MOMmin(i))
            Next i
        End With

        Dim MOMplot As Excel.Chart
        Dim MOM_SerCol As Excel.SeriesCollection
        Dim MOM_Series, NEG_Series As Excel.Series
        Dim MOM_AxisCategory, MOM_AxisValue As Excel.Axes

        xlCharts = xlScratch.ChartObjects
        myChart = xlCharts.Add(1754, 10, 900, 600)
        MOMplot = myChart.Chart
        MOMplot.ChartType = Excel.XlChartType.xlXYScatterLinesNoMarkers
        MOMplot.ChartStyle = 2
        MOM_SerCol = MOMplot.SeriesCollection

        MOM_Series = MOM_SerCol.NewSeries
        MOM_Series.Name = "(+)Moment"
        MOM_Series.XValues = xlScratch.Range("AH1:AH" & NumSteps + 1)
        MOM_Series.Values = xlScratch.Range("AI1:AI" & NumSteps + 1)
        MOM_Series.Format.Line.Weight = 1.5

        If MOMmin(1) = 0 Then GoTo 200
        NEG_Series = MOM_SerCol.NewSeries
        NEG_Series.Name = "(-)Moment"
        NEG_Series.XValues = xlScratch.Range("AH1:AH" & NumSteps + 1)
        NEG_Series.Values = xlScratch.Range("AJ1:AJ" & NumSteps + 1)
        NEG_Series.Format.Line.Weight = 1.5

200:    With MOMplot
            .Legend.Position = Excel.XlLegendPosition.xlLegendPositionTop
            .HasTitle = True
            .ChartTitle.Text = "Maximum Moment Capacity (time)"

            MOM_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
            MOM_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = True
            MOM_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Text ↙
    () = "Time, Hours"
            MOM_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Font.↙
    Bold = False
            MOM_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Font.↙
    Size = 14
            MOM_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines = True
            MOM_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines = True
            MOM_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format.Line↙
    .DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            MOM_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScale = 0
            MOM_AxisCategory.Item(Excel.XlAxisType.xlCategory).MaximumScale = EndTime /  ↙
    3600
            MOM_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition = Excel.↙
    XlTickLabelPosition.xlTickLabelPositionLow
            MOM_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold =    ↙
    False
            MOM_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size = 14

            MOM_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
            MOM_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = True
            MOM_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Text() =   ↙
```

```
    "Maximum Moment, kN-m"
            MOM_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Bold =
     False
            MOM_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Size =
     14
            MOM_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
            MOM_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
            MOM_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line.
    DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            MOM_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScale = 0
            MOM_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.
    XlTickLabelPosition.xlTickLabelPositionLow
            MOM_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold = False
            MOM_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
        End With
        MOMplot.Refresh()



        If xlREOUT.Cells(20, 2).value = 21 Then GoTo 201

        'MOMENT-
    AXIAL_INTERACTION
    ____________________________________________________________________________
    _________________
        '============================================================================
    ================================================
909:    prgMAIN.Value = 1000

        With xlMN
            .Cells.Clear()
            .Cells.HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter
            .Columns(2).HorizontalAlignment = Excel.XlHAlign.xlHAlignRight
            .Cells(1, 2).value = "TIME:"
            .Cells(2, 2).value = "eULT:"
            .Cells(3, 2).value = "c_FromTop:"
            .Rows(1).Font.Bold = True
            .Columns(2).Font.Bold = True
            .Columns(2).ColumnWidth = 12
            .Rows(1).Borders(Excel.XlBordersIndex.xlEdgeBottom).LineStyle = Excel.
    XlLineStyle.xlDouble
            .Rows(1).Borders(Excel.XlBordersIndex.xlEdgeBottom).Weight = Excel.
    XlBorderWeight.xlThick

            'Find uniform strain corresponding to maximum axial force
            estep = 0.0001
            emax = 0.1

            mnCOUNT = 100
            .Cells(1, 1).value = mnCOUNT

            For i = 0 To NumSteps
                .Cells(1, 4 * i + 3).value = i * TimeStep
                .Cells(1, 4 * i + 3).HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter
                .Cells(1, 4 * i + 3).Font.size = 14

                timer_i.Text = ""
                timer_j.Text = ""
                timer_iteration.Text = ""
                timer_c.Text = ""
                timer_N.Text = ""
                timer_M.Text = ""
                timer_phi.Text = ""
                txtRUN.Text = "Moment-Axial"
```

```
                itt = 0
                prgMAIN.Value = 1000 + Round(1000 * (i / NumSteps))

                Po(i) = 0
                If i = 0 Then GoTo 910
                eTest(i) = eTest(i - 1) / 2

910:            eTest(i) = eTest(i) + estep
                If eTest(i) > emax Then GoTo 999

                itt = itt + 1
                timer_iteration.Text = itt

                sumC = 0
                sumMC = 0
                sumS = 0
                sumMS = 0

                For k = 1 To Elements
                    If eTest(i) > 0 Then GoTo 921
                    If eTest(i) < ectu_t(k, i) Then fc(k, i) = 0
                    If ectu_t(k, i) < eTest(i) And eTest(i) < ect_t(k, i) Then fc(k, i) =↙
     Ecc_t(k, i) * ectu_t(k, i) / 10 - Ecc_t(k, i) * eTest(i) / 10
                    If ect_t(k, i) < eTest(i) And eTest(i) < 0 Then fc(k, i) = Ecc_t(k,  ↙
    i) * eTest(i)
                    GoTo 922

921:                If eTest(i) <= eo_t(k, i) Then fc(k, i) = 3 * eTest(i) * fprimec_t(k,↙
     i) / (eo_t(k, i) * (2 + (eTest(i) / eo_t(k, i)) ^ 3))
                    If eo_t(k, i) < eTest(i) And eTest(i) <= ecu_t(k, i) Then fc(k, i) = ↙
    fprimec_t(k, i) - (-fprimec_t(k, i) / (ecu_t(k, i) - eo_t(k, i))) * eo_t(k, i) + (-  ↙
    fprimec_t(k, i) / (ecu_t(k, i) - eo_t(k, i))) * eTest(i)
                    If eTest(i) > ecu_t(k, i) Then fc(k, i) = 0

922:                FORc(k, i) = fc(k, i) * elementAREA(k) / 1000
                    sumC = sumC + FORc(k, i)
                    MOMc(k, i) = FORc(k, i) * element_Y(k) / 1000
                    sumMC = sumMC + MOMc(k, i)
                Next k

                For k = 1 To REINFcount
                    If Abs(eTest(i)) < ep_t(k, i) Then fs(k, i) = eTest(i) * Est_t(k, i)
                    If ep_t(k, i) <= Abs(eTest(i)) And Abs(eTest(i)) <= ey_t(k, i) Then  ↙
    fs(k, i) = (fp_t(k, i) - c_t(k, i) + (b_t(k, i) / a_t(k, i)) * (a_t(k, i) ^ 2 - (ey_t↙
    (k, i) - Abs(eTest(i))) ^ 2) ^ 0.5) * eTest(i) / Abs(eTest(i))
                    If ey_t(k, i) <= Abs(eTest(i)) And Abs(eTest(i)) <= et_t(k, i) Then  ↙
    fs(k, i) = (fy_t(k, i)) * eTest(i) / Abs(eTest(i))
                    If et_t(k, i) <= Abs(eTest(i)) And Abs(eTest(i)) <= eu_t(k, i) Then  ↙
    fs(k, i) = (fy_t(k, i) * (1 - (Abs(eTest(i)) - et_t(k, i)) / (eu_t(k, i) - et_t(k,  ↙
    i)))) * eTest(i) / Abs(eTest(i))
                    If Abs(eTest(i)) > eu_t(k, i) Then fs(k, i) = 0

                    FORs(k, i) = fs(k, i) * reinf_AREA(k) / 1000
                    sumS = sumS + FORs(k, i)
                    MOMs(k, i) = FORs(k, i) * reinf_Y(k) / 1000
                    sumMS = sumMS + MOMs(k, i)
                Next k

                P_dum(itt) = sumC + sumS
                timer_N.Text = P_dum(itt)
                If P_dum(itt) > Po(i) Then
                    Po(i) = P_dum(itt)
                    Mo(i) = sumMC + sumMS
                    GoTo 998
                Else
                    If itt < 5 Then GoTo 910
```

```
                    If P_dum(itt) < P_dum(itt - 1) And P_dum(itt) < P_dum(itt - 2) And
    P_dum(itt) < P_dum(itt - 3) Then GoTo 999
                    GoTo 910
                End If

998:            'Print
                .Cells(2, 4 * i + 3).value = eTest(i)
                .Cells(2, 4 * i + 3).NumberFormat = "0.00000"

                .Cells(1, 4 * i + 4).value = "M (kN-m)"
                .Cells(2, 4 * i + 4).value = Mo(i)
                .Columns(4 * i + 4).NumberFormat = "0"
                .Columns(4 * i + 4).ColumnWidth = 12
                .Cells(1, 4 * i + 5).value = "P (kN)"
                .Cells(2, 4 * i + 5).value = Po(i)
                .Columns(4 * i + 5).NumberFormat = "0"
                .Columns(4 * i + 6).ColumnWidth = 4
                GoTo 910

999:            eULT(i) = .Cells(2, 4 * i + 3).value
            Next i


            'Find uniform strain corresponding to minimum axial force
            estep = 0.0001
            emax = -0.2


            For i = 0 To NumSteps
                .Cells(1, 4 * i + 3).value = i * TimeStep
                .Cells(1, 4 * i + 3).HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter
                .Cells(1, 4 * i + 3).Font.size = 14

                timer_i.Text = ""
                timer_j.Text = ""
                timer_iteration.Text = ""
                timer_c.Text = ""
                timer_N.Text = ""
                timer_M.Text = ""
                timer_phi.Text = ""
                txtRUN.Text = "Moment-Axial"

                itt = 0
                prgMAIN.Value = 1000 + Round(1000 * (i / NumSteps))

                Pon(i) = 0
                If i = 0 Then GoTo 610
                eTest(i) = eTest(i - 1) / 2

610:            eTest(i) = eTest(i) - estep
                If eTest(i) < emax Then GoTo 699

                itt = itt + 1
                timer_iteration.Text = itt

                sumC = 0
                sumMC = 0
                sumS = 0
                sumMS = 0

                For k = 1 To Elements
                    If eTest(i) > 0 Then GoTo 621
                    If eTest(i) < ectu_t(k, i) Then fc(k, i) = 0
                    If ectu_t(k, i) < eTest(i) And eTest(i) < ect_t(k, i) Then fc(k, i) =
     Ecc_t(k, i) * ectu_t(k, i) / 10 - Ecc_t(k, i) * eTest(i) / 10
                    If ect_t(k, i) < eTest(i) And eTest(i) < 0 Then fc(k, i) = Ecc_t(k,
```

```
    i) * eTest(i)
                    GoTo 622

621:                If eTest(i) <= eo_t(k, i) Then fc(k, i) = 3 * eTest(i) * fprimec_t(k, ↙
     i) / (eo_t(k, i) * (2 + (eTest(i) / eo_t(k, i)) ^ 3))
                    If eo_t(k, i) < eTest(i) And eTest(i) <= ecu_t(k, i) Then fc(k, i) = ↙
    fprimec_t(k, i) - (-fprimec_t(k, i) / (ecu_t(k, i) - eo_t(k, i))) * eo_t(k, i) + (- ↙
    fprimec_t(k, i) / (ecu_t(k, i) - eo_t(k, i))) * eTest(i)
                    If eTest(i) > ecu_t(k, i) Then fc(k, i) = 0

622:                FORc(k, i) = fc(k, i) * elementAREA(k) / 1000
                    sumC = sumC + FORc(k, i)
                    MOMc(k, i) = FORc(k, i) * element_Y(k) / 1000
                    sumMC = sumMC + MOMc(k, i)
                Next k

                For k = 1 To REINFcount
                    If Abs(eTest(i)) < ep_t(k, i) Then fs(k, i) = eTest(i) * Est_t(k, i)
                    If ep_t(k, i) <= Abs(eTest(i)) And Abs(eTest(i)) <= ey_t(k, i) Then ↙
    fs(k, i) = (fp_t(k, i) - c_t(k, i) + (b_t(k, i) / a_t(k, i)) * (a_t(k, i) ^ 2 - (ey_t↙
    (k, i) - Abs(eTest(i))) ^ 2) ^ 0.5) * eTest(i) / Abs(eTest(i))
                    If ey_t(k, i) <= Abs(eTest(i)) And Abs(eTest(i)) <= et_t(k, i) Then ↙
    fs(k, i) = (fy_t(k, i)) * eTest(i) / Abs(eTest(i))
                    If et_t(k, i) <= Abs(eTest(i)) And Abs(eTest(i)) <= eu_t(k, i) Then ↙
    fs(k, i) = (fy_t(k, i) * (1 - (Abs(eTest(i)) - et_t(k, i)) / (eu_t(k, i) - et_t(k, ↙
    i)))) * eTest(i) / Abs(eTest(i))
                    If Abs(eTest(i)) > eu_t(k, i) Then fs(k, i) = 0

                    FORs(k, i) = fs(k, i) * reinf_AREA(k) / 1000
                    sumS = sumS + FORs(k, i)
                    MOMs(k, i) = FORs(k, i) * reinf_Y(k) / 1000
                    sumMS = sumMS + MOMs(k, i)
                Next k

                P_dum(itt) = sumC + sumS
                timer_N.Text = P_dum(itt)
                If P_dum(itt) < Pon(i) Then
                    Pon(i) = P_dum(itt)
                    Mon(i) = sumMC + sumMS
                    GoTo 698
                Else
                    If itt < 5 Then GoTo 610
                    If P_dum(itt) > P_dum(itt - 1) And P_dum(itt) > P_dum(itt - 2) And ↙
    P_dum(itt) > P_dum(itt - 3) Then GoTo 699
                    GoTo 610
                End If

698:            'Print
                .Cells(2, 4 * i + 4).value = Mo(i)
                .Cells(2, 4 * i + 5).value = Po(i)
                .Cells(23, 4 * i + 4).value = Mon(i)
                .Cells(23, 4 * i + 5).value = Pon(i)
                .Cells(44, 4 * i + 4).value = Mo(i)
                .Cells(44, 4 * i + 5).value = Po(i)
                GoTo 610

699:        Next i
        End With


        'DOOP/                                                                   ↙
    MN_____________________________________________________________________________
        Dim cc(0 To 1000) As Single
        Dim eTOOP As Single
        Dim goop As Integer
        Dim goopMAX As Integer
```

```
        Dim maxx As Single
        Dim maxxM, maxxN As Single
        Dim Mmm(0 To 240, 0 To 150, 0 To 7000), Nnn(0 To 240, 0 To 150, 0 To 7000) As ↙
    Single

        cc(1) = 0.9 * s_height
        cc(2) = 0.85 * s_height
        cc(3) = 0.8 * s_height
        cc(4) = 0.75 * s_height
        cc(5) = 0.7 * s_height
        cc(6) = 0.65 * s_height
        cc(7) = 0.6 * s_height
        cc(8) = 0.55 * s_height
        cc(9) = 0.53 * s_height
        cc(10) = 0.51 * s_height
        cc(11) = 0.49 * s_height
        cc(12) = 0.47 * s_height
        cc(13) = 0.45 * s_height
        cc(14) = 0.4 * s_height
        cc(15) = 0.35 * s_height
        cc(16) = 0.3 * s_height
        cc(17) = 0.25 * s_height
        cc(18) = 0.2 * s_height
        cc(19) = 0.15 * s_height
        cc(20) = 0.1 * s_height


        With xlDOOP
            For i = 0 To NumSteps
                prgMAIN.Value = 2000 + Round(1000 * i / NumSteps)
                timer_i.Text = i
                timer_iteration.Text = "N/A"
                timer_phi.Text = "N/A"

                maxxM = 0
                maxxN = 0

                For j = 1 To 20

                    timer_c.Text = cc(j)
                    timer_j.Text = j
                    goop = 0
                    eTOOP = 0
                    maxxM = 0
                    maxxN = 0

300:                goop = goop + 1
                    eTOOP = eTOOP + 0.0001
                    phi_ = eTOOP / cc(j)
                    If goop > goopMAX Then goopMAX = goop

                    If eTOOP > 0.02 Then GoTo 399

                    sumC = 0
                    sumMC = 0
                    sumS = 0
                    sumMS = 0

                    For k = 1 To Elements
                        yprime = element_Y(k) - (Yaxis_max - cc(j))
                        ee = phi_ * yprime

                        If ee > 0 Then GoTo 321
                        If ee < ectu_t(k, i) Then fc(k, j) = 0
                        If ectu_t(k, i) < ee And ee < ect_t(k, i) Then fc(k, j) = Ecc_t(k↙
    , i) * ectu_t(k, i) / 10 - Ecc_t(k, i) * ee / 10
```

```
                    If ect_t(k, i) < ee And ee < 0 Then fc(k, j) = Ecc_t(k, i) * ee
                    GoTo 322

321:                If ee <= eo_t(k, i) Then fc(k, j) = 3 * ee * fprimec_t(k, i) /
    (eo_t(k, i) * (2 + (ee / eo_t(k, i)) ^ 3))
                    If eo_t(k, i) < ee And ee <= ecu_t(k, i) Then fc(k, j) =
    fprimec_t(k, i) - (-fprimec_t(k, i) / (ecu_t(k, i) - eo_t(k, i))) * eo_t(k, i) + (-
    fprimec_t(k, i) / (ecu_t(k, i) - eo_t(k, i))) * ee
                    If ee > ecu_t(k, i) Then fc(k, j) = 0

322:                FORc(k, j) = fc(k, j) * elementAREA(k) / 1000
                    sumC = sumC + FORc(k, j)
                    MOMc(k, j) = FORc(k, j) * element_Y(k) / 1000
                    sumMC = sumMC + MOMc(k, j)
                Next k

                For k = 1 To REINFcount
                    yprime = reinf_Y(k) - (Yaxis_max - cc(j))
                    ee = phi_ * yprime

                    If Abs(ee) < ep_t(k, i) Then fs(k, j) = ee * Est_t(k, i)
                    If ep_t(k, i) <= Abs(ee) And Abs(ee) <= ey_t(k, i) Then fs(k, j)
    = (fp_t(k, i) - c_t(k, i) + (b_t(k, i) / a_t(k, i)) * (a_t(k, i) ^ 2 - (ey_t(k, i) -
    Abs(ee)) ^ 2) ^ 0.5) * ee / Abs(ee)
                    If ey_t(k, i) <= Abs(ee) And Abs(ee) <= et_t(k, i) Then fs(k, j)
    = (fy_t(k, i)) * ee / Abs(ee)
                    If et_t(k, i) <= Abs(ee) And Abs(ee) <= eu_t(k, i) Then fs(k, j)
    = (fy_t(k, i) * (1 - (Abs(ee) - et_t(k, i)) / (eu_t(k, i) - et_t(k, i)))) * ee / Abs
    (ee)
                    If Abs(ee) > eu_t(k, i) Then fs(k, j) = 0

                    FORs(k, j) = fs(k, j) * reinf_AREA(k) / 1000
                    sumS = sumS + FORs(k, j)
                    MOMs(k, j) = FORs(k, j) * reinf_Y(k) / 1000
                    sumMS = sumMS + MOMs(k, j)
                Next k

                Nnn(i, j, goop) = sumC + sumS
                timer_N.Text = Nnn(i, j, goop)
                Mmm(i, j, goop) = sumMC + sumMS
                timer_M.Text = Mmm(i, j, goop)

                GoTo 300

399:            For k = 1 To goop
                    If Abs(Mmm(i, j, k)) > maxxM Then maxxM = Abs(Mmm(i, j, k))
                    If Abs(Nnn(i, j, k)) > maxxN Then maxxN = Abs(Nnn(i, j, k))
                Next k
            Next j

            For j = 1 To 20
                maxx = 0
                For k = 1 To goop
                    If Sqrt(Abs(Mmm(i, j, k) / maxxM) ^ 2 + Abs(Nnn(i, j, k) / maxxN)
     ^ 2) > maxx Then
                        maxx = Sqrt(Abs(Mmm(i, j, k) / maxxM) ^ 2 + Abs(Nnn(i, j, k)
    / maxxN) ^ 2)
                        xlMN.Cells(2 + j, 4 * i + 4).value = Mmm(i, j, k)
                        xlMN.Cells(2 + j, 4 * i + 5).value = Nnn(i, j, k)
                    End If
                Next k
            Next j
398:    Next i
    End With

    '_____________________________________________________________________________
```

```
    _____


        'NEGATIVE DOOP/MN*****************************************************************
    *********
        With xlDOOP
            For i = 0 To NumSteps
                prgMAIN.Value = 2000 + Round(1000 * i / NumSteps)
                timer_i.Text = i
                timer_iteration.Text = "N/A"
                timer_phi.Text = "N/A"

                maxxM = 0
                maxxN = 0

                For j = 1 To 20

                    timer_c.Text = cc(j)
                    timer_j.Text = j
                    goop = 0
                    eTOOP = 0
                    maxxM = 0
                    maxxN = 0

400:                goop = goop + 1
                    eTOOP = eTOOP - 0.0001
                    phi_ = eTOOP / cc(j)
                    If goop > goopMAX Then goopMAX = goop

                    If eTOOP < -0.02 Then GoTo 499

                    sumC = 0
                    sumMC = 0
                    sumS = 0
                    sumMS = 0

                    For k = 1 To Elements
                        yprime = element_Y(k) - (Yaxis_max - cc(j))
                        ee = phi_ * yprime

                        If ee > 0 Then GoTo 421
                        If ee < ectu_t(k, i) Then fc(k, j) = 0
                        If ectu_t(k, i) < ee And ee < ect_t(k, i) Then fc(k, j) = Ecc_t(k
    , i) * ectu_t(k, i) / 10 - Ecc_t(k, i) * ee / 10
                        If ect_t(k, i) < ee And ee < 0 Then fc(k, j) = Ecc_t(k, i) * ee
                        GoTo 422

421:                    If ee <= eo_t(k, i) Then fc(k, j) = 3 * ee * fprimec_t(k, i) /
    (eo_t(k, i) * (2 + (ee / eo_t(k, i)) ^ 3))
                        If eo_t(k, i) < ee And ee <= ecu_t(k, i) Then fc(k, j) =
    fprimec_t(k, i) - (-fprimec_t(k, i) / (ecu_t(k, i) - eo_t(k, i))) * eo_t(k, i) + (-
    fprimec_t(k, i) / (ecu_t(k, i) - eo_t(k, i))) * ee
                        If ee > ecu_t(k, i) Then fc(k, j) = 0

422:                    FORc(k, j) = fc(k, j) * elementAREA(k) / 1000
                        sumC = sumC + FORc(k, j)
                        MOMc(k, j) = FORc(k, j) * element_Y(k) / 1000
                        sumMC = sumMC + MOMc(k, j)
                    Next k

                    For k = 1 To REINFcount
                        yprime = reinf_Y(k) - (Yaxis_max - cc(j))
                        ee = phi_ * yprime

                        If Abs(ee) < ep_t(k, i) Then fs(k, j) = ee * Est_t(k, i)
```

```
                        If ep_t(k, i) <= Abs(ee) And Abs(ee) <= ey_t(k, i) Then fs(k, j) ↙
    = (fp_t(k, i) - c_t(k, i) + (b_t(k, i) / a_t(k, i)) * (a_t(k, i) ^ 2 - (ey_t(k, i) - ↙
    Abs(ee)) ^ 2) ^ 0.5) * ee / Abs(ee)
                        If ey_t(k, i) <= Abs(ee) And Abs(ee) <= et_t(k, i) Then fs(k, j) ↙
    = (fy_t(k, i)) * ee / Abs(ee)
                        If et_t(k, i) <= Abs(ee) And Abs(ee) <= eu_t(k, i) Then fs(k, j) ↙
    = (fy_t(k, i) * (1 - (Abs(ee) - et_t(k, i)) / (eu_t(k, i) - et_t(k, i)))) * ee / Abs ↙
    (ee)
                        If Abs(ee) > eu_t(k, i) Then fs(k, j) = 0

                        FORs(k, j) = fs(k, j) * reinf_AREA(k) / 1000
                        sumS = sumS + FORs(k, j)
                        MOMs(k, j) = FORs(k, j) * reinf_Y(k) / 1000
                        sumMS = sumMS + MOMs(k, j)
                    Next k

                    Nnn(i, j, goop) = sumC + sumS
                    timer_N.Text = Nnn(i, j, goop)
                    Mmm(i, j, goop) = sumMC + sumMS
                    timer_M.Text = Mmm(i, j, goop)

                    GoTo 400

499:                For k = 1 To goop
                        If Abs(Mmm(i, j, k)) > maxxM Then maxxM = Abs(Mmm(i, j, k))
                        If Abs(Nnn(i, j, k)) > maxxN Then maxxN = Abs(Nnn(i, j, k))
                    Next k
                Next j

                For j = 1 To 20
                    maxx = 0
                    For k = 1 To goop
                        If Sqrt(Abs(Mmm(i, j, k) / maxxM) ^ 2 + Abs(Nnn(i, j, k) / maxxN)↙
     ^ 2) > maxx Then
                            maxx = Sqrt(Abs(Mmm(i, j, k) / maxxM) ^ 2 + Abs(Nnn(i, j, k) ↙
    / maxxN) ^ 2)
                            xlMN.Cells(23 + j, 4 * i + 4).value = Mmm(i, j, k)
                            xlMN.Cells(23 + j, 4 * i + 5).value = Nnn(i, j, k)
                        End If
                    Next k
                Next j

498:        Next i
        End With
                                                                                    ↙
    '_____________________________________________________________________________↙
    _____


888:    'Plot Moment-Axial Interaction
        Dim MNplot As Excel.Chart
        Dim MN_SerCol As Excel.SeriesCollection
        Dim MN0_Series As Excel.Series
        Dim MN_AxisCategory, MN_AxisValue As Excel.Axes

        xlCharts = xlMN.ChartObjects
        myChart = xlCharts.Add(270, 100, 900, 600)
        MNplot = myChart.Chart
        MNplot.ChartType = Excel.XlChartType.xlXYScatterLines
        MNplot.ChartStyle = 3
        MN_SerCol = MNplot.SeriesCollection

        MN0_Series = MN_SerCol.NewSeries
        MN0_Series.Name = xlMN.Cells(1, 3).value
        MN0_Series.XValues = xlMN.Range("D2:D44")
        MN0_Series.Values = xlMN.Range("E2:E44")
```

```
        MN0_Series.MarkerSize = 6
        MN0_Series.MarkerStyle = 8
        MN0_Series.Format.Line.ForeColor.RGB = 1

        With MNplot
            .Legend.Position = Excel.XlLegendPosition.xlLegendPositionTop
            .HasTitle = True
            .ChartTitle.Text = "Moment-Axial Interaction"

            MN_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
            MN_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = True
            MN_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Text() ↙
     = "Moment, kN-m"
            MN_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Font. ↙
    Bold = False
            MN_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Font. ↙
    Size = 14
            MN_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines = True
            MN_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines = True
            MN_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format.Line. ↙
    DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            MN_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScaleIsAuto = True
            MN_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition = Excel. ↙
    XlTickLabelPosition.xlTickLabelPositionLow
            MN_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold =     ↙
    False
            MN_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size = 14

            MN_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
            MN_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = True
            MN_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Text() =    ↙
    "Axial Load, kN"
            MN_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Bold = ↙
    False
            MN_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Size = ↙
    14
            MN_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
            MN_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
            MN_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line.      ↙
    DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            MN_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScaleIsAuto = True
            MN_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.       ↙
    XlTickLabelPosition.xlTickLabelPositionLow
            MN_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold = False
            MN_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
        End With
        MNplot.Refresh()



        'Plot Max Axial Envelope Curve
        With xlScratch

            .Range("DA1:DC1000").Clear()
            .Columns(105).NumberFormat = "0.00"
            .Columns(106).NumberFormat = "0"
            .Columns(107).NumberFormat = "0"

            For i = 0 To NumSteps
                .Cells(i + 1, 105).value = time(i) / 3600
                .Cells(i + 1, 106).value = Po(i)
                .Cells(i + 1, 107).value = Abs(Pon(i))
            Next i
        End With

        Dim AXIALplot As Excel.Chart
```

```
        Dim AXIAL_SerCol As Excel.SeriesCollection
        Dim COMP_Series, TEN_Series As Excel.Series
        Dim AXIAL_AxisCategory, AXIAL_AxisValue As Excel.Axes

        xlCharts = xlScratch.ChartObjects
        myChart = xlCharts.Add(5164, 10, 900, 600)
        AXIALplot = myChart.Chart
        AXIALplot.ChartType = Excel.XlChartType.xlXYScatterLinesNoMarkers
        AXIALplot.ChartStyle = 2
        AXIAL_SerCol = AXIALplot.SeriesCollection

        COMP_Series = AXIAL_SerCol.NewSeries
        COMP_Series.Name = "Compression"
        COMP_Series.XValues = xlScratch.Range("DA1:DA" & NumSteps + 1)
        COMP_Series.Values = xlScratch.Range("DB1:DB" & NumSteps + 1)
        COMP_Series.Format.Line.Weight = 1.5

        TEN_Series = AXIAL_SerCol.NewSeries
        TEN_Series.Name = "Tension"
        TEN_Series.XValues = xlScratch.Range("DA1:DA" & NumSteps + 1)
        TEN_Series.Values = xlScratch.Range("DC1:DC" & NumSteps + 1)
        TEN_Series.Format.Line.Weight = 1.5

        With AXIALplot
            .Legend.Position = Excel.XlLegendPosition.xlLegendPositionTop
            .HasTitle = True
            .ChartTitle.Text = "Maximum Axial Capacity (time)"

            AXIAL_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
            AXIAL_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = True
            AXIAL_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.
Text() = "Time, Hours"
            AXIAL_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.
Font.Bold = False
            AXIAL_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.
Font.Size = 14
            AXIAL_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines = True
            AXIAL_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines = True
            AXIAL_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format.
Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            AXIAL_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScale = 0
            AXIAL_AxisCategory.Item(Excel.XlAxisType.xlCategory).MaximumScale = EndTime /
 3600
            AXIAL_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition =
Excel.XlTickLabelPosition.xlTickLabelPositionLow
            AXIAL_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold =
False
            AXIAL_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size =
14

            AXIAL_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
            AXIAL_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = True
            AXIAL_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Text() =
"Maximum Axial Force, kN"
            AXIAL_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Bold
 = False
            AXIAL_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Size
 = 14
            AXIAL_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
            AXIAL_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
            AXIAL_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line.
DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            AXIAL_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScale = 0
            AXIAL_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.
XlTickLabelPosition.xlTickLabelPositionLow
            AXIAL_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold = False
```

```
            AXIAL_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
        End With
        AXIALplot.Refresh()

        '=============================================================================================
    ==============================================
        '=============================================================================================
    ==============================================



        'Plot Axial/MaxAxial, Moment/MaxMoment, and Fire/MaxFire
201:    With xlScratch
            maxM = 0
            For i = 0 To NumSteps
                If .Cells(i + 1, 35).value > maxM Then
                    maxM = .Cells(i + 1, 35).value
                End If
            Next i
            .Cells(10, 1).value = maxM
            minM = 0
            For i = 0 To NumSteps
                If .Cells(i + 1, 36).value > minM Then
                    minM = .Cells(i + 1, 36).value
                End If
            Next i
            .Cells(11, 1).value = minM
            maxF = 0
            For i = 1 To .Cells(9, 1).value
                If xlFire.Cells(i, 1).value > EndTime Then
                    FFF = i
                    GoTo 204
                End If
                If xlFire.Cells(i, 2).value > maxF Then
                    maxF = xlFire.Cells(i, 2).value
                End If
            Next i
204:        .Range("CA1:CG1000").Clear()
            .Columns(79).NumberFormat = "0.00"
            .Columns(80).NumberFormat = "0.00"
            .Columns(81).NumberFormat = "0.00"
            .Columns(82).NumberFormat = "0.00"
            .Columns(83).NumberFormat = "0.00"
            .Columns(84).NumberFormat = "0.00"
            .Columns(85).NumberFormat = "0.00"
            For i = 0 To NumSteps
                .Cells(i + 1, 79).value = time(i) / 3600
                .Cells(i + 1, 80).value = MOMmax(i) / maxM
                .Cells(i + 1, 81).value = Abs(MOMmin(i)) / Abs(minM)
                .Cells(i + 1, 84).value = Po(i) / Po(0)
                .Cells(i + 1, 85).value = Abs(Pon(i)) / Abs(Pon(0))
            Next i
            For i = 1 To FFF
                .Cells(i, 82).value = xlFire.Cells(i, 3).value
                .Cells(i, 83).value = xlFire.Cells(i, 2).value / maxF
            Next i
        End With

        Dim RNplot As Excel.Chart
        Dim RN_SerCol As Excel.SeriesCollection
        Dim MOMpos_Series, MOMneg_Series, FIRE_Series, COMPrn_Series, NEGPrn_Series As
    Excel.Series
        Dim RN_AxisCategory, RN_AxisValue As Excel.Axes

        xlCharts = xlScratch.ChartObjects
        myChart = xlCharts.Add(4102, 10, 900, 600)
```

```
        RNplot = myChart.Chart
        RNplot.ChartType = Excel.XlChartType.xlXYScatterLinesNoMarkers
        RNplot.ChartStyle = 2
        RN_SerCol = RNplot.SeriesCollection

        If xlREOUT.Cells(20, 2).value = 12 Then GoTo 202
        MOMpos_Series = RN_SerCol.NewSeries
        MOMpos_Series.Name = "+Moment(t)/Mmax"
        MOMpos_Series.XValues = xlScratch.Range("CA1:CA" & NumSteps + 1)
        MOMpos_Series.Values = xlScratch.Range("CB1:CB" & NumSteps + 1)
        MOMpos_Series.Format.Line.Weight = 1.5

        MOMneg_Series = RN_SerCol.NewSeries
        MOMneg_Series.Name = "-Moment(t)/Mmax"
        MOMneg_Series.XValues = xlScratch.Range("CA1:CA" & NumSteps + 1)
        MOMneg_Series.Values = xlScratch.Range("CC1:CC" & NumSteps + 1)
        MOMneg_Series.Format.Line.Weight = 1.5

202:    If xlREOUT.Cells(20, 2).value = 21 Then GoTo 203
        COMPrn_Series = RN_SerCol.NewSeries
        COMPrn_Series.Name = "Compression(t)/Pmax"
        COMPrn_Series.XValues = xlScratch.Range("CA1:CA" & NumSteps + 1)
        COMPrn_Series.Values = xlScratch.Range("CF1:CF" & NumSteps + 1)
        COMPrn_Series.Format.Line.Weight = 1.5

        NEGPrn_Series = RN_SerCol.NewSeries
        NEGPrn_Series.Name = "Negative(t)/Pmax"
        NEGPrn_Series.XValues = xlScratch.Range("CA1:CA" & NumSteps + 1)
        NEGPrn_Series.Values = xlScratch.Range("CG1:CG" & NumSteps + 1)
        NEGPrn_Series.Format.Line.Weight = 1.5

203:    FIRE_Series = RN_SerCol.NewSeries
        FIRE_Series.Name = "GasTemp(t)/Tmax"
        FIRE_Series.XValues = xlScratch.Range("CD1:CD" & xlScratch.Cells(9, 1).value)
        FIRE_Series.Values = xlScratch.Range("CE1:CE" & xlScratch.Cells(9, 1).value)
        FIRE_Series.Format.Line.Weight = 1.5


        With RNplot
            .Legend.Position = Excel.XlLegendPosition.xlLegendPositionTop
            .HasTitle = True
            .ChartTitle.Text = "Normalized Resistance (time)"

            RN_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
            RN_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = True
            RN_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Text()↙
     = "Time, Hours"
            RN_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Font. ↙
    Bold = False
            RN_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Font. ↙
    Size = 14
            RN_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines = True
            RN_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines = True
            RN_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format.Line.↙
    DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            RN_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScale = 0
            RN_AxisCategory.Item(Excel.XlAxisType.xlCategory).MaximumScale = EndTime /    ↙
    3600
            RN_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition = Excel. ↙
    XlTickLabelPosition.xlTickLabelPositionLow
            RN_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold =      ↙
    False
            RN_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size = 14

            RN_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
            RN_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = True
```

```
            RN_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Text() = "Rn
    (t,T)/Rn,max"
            RN_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Bold =
    False
            RN_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Size =
    14
            RN_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
            RN_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
            RN_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line.
    DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            RN_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScale = 0
            RN_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.
    XlTickLabelPosition.xlTickLabelPositionLow
            RN_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold = False
            RN_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
        End With
        RNplot.Refresh()




        'Format re.out
        With xlREOUT
            .Columns(5).NumberFormat = "0.0"
            .Columns(6).NumberFormat = "0.0"
            .Columns(7).NumberFormat = "0.0"
            .Columns(11).NumberFormat = "0.0"
            .Columns(12).NumberFormat = "0.0"
            .Columns(14).NumberFormat = "0.0"
            .Columns(15).NumberFormat = "0.0"
            .Columns(17).NumberFormat = "0.0"
            .Columns(18).NumberFormat = "0.0"
            .Columns(20).NumberFormat = "0.0"
            .Columns(21).NumberFormat = "0.0"
            .Columns(24).NumberFormat = "0.0"
            .Columns(25).NumberFormat = "0.0"
            .Columns(26).NumberFormat = "0.0"
            .Columns(27).NumberFormat = "0.000000"
            .Columns(27).ColumnWidth = 12
            .Columns(28).NumberFormat = "0.000000"
            .Columns(28).ColumnWidth = 12
            .Columns(29).NumberFormat = "0.0"
            .Columns(30).NumberFormat = "0.0"
            .Columns(31).NumberFormat = "0.000000"
            .Columns(31).ColumnWidth = 12
            .Columns(33).NumberFormat = "0.0"
            .Columns(34).NumberFormat = "0.0"
            .Columns(35).NumberFormat = "0.0"
            .Columns(36).NumberFormat = "0.0"
            .Columns(39).NumberFormat = "0.0"
            .Columns(40).NumberFormat = "0.0"
            .Columns(41).NumberFormat = "0.0"
            .Columns(42).NumberFormat = "0.0"
            .Columns(44).NumberFormat = "0.0"
            .Columns(45).NumberFormat = "0.0"
            .Columns(46).NumberFormat = "0.0"
            .Columns(47).NumberFormat = "0"
            .Columns(48).NumberFormat = "0.000000"
            .Columns(48).ColumnWidth = 12
            .Columns(49).NumberFormat = "0.00"
            .Columns(50).NumberFormat = "0.00"
            .Columns(51).NumberFormat = "0.00"
```

```
            .Columns(52).NumberFormat = "0.0"
            .Columns(53).NumberFormat = "0.0"
            .Columns(54).NumberFormat = "0.000000"
            .Columns(54).ColumnWidth = 12
            .Columns(56).NumberFormat = "0.0"
            .Columns(57).NumberFormat = "0.0"
            .Columns(58).NumberFormat = "0.0"
            .Columns(59).NumberFormat = "0.0"
        End With


        'replot section (refresh to match submitted data)
        xlSection.ChartObjects(1).Delete()

        Dim sectionplot As Excel.Chart
        Dim section_SerCol As Excel.SeriesCollection
        Dim section_Series, reinf_Series As Excel.Series
        Dim section_AxisCategory, section_AxisValue As Excel.Axes

        xlCharts = xlSection.ChartObjects
        myChart = xlCharts.Add(260, 10, 900, 600)
        sectionplot = myChart.Chart
        sectionplot.ChartType = Excel.XlChartType.xlXYScatter
        section_SerCol = sectionplot.SeriesCollection

        section_Series = section_SerCol.NewSeries
        section_Series.Name = "section"
        section_Series.XValues = xlSection.Range("A1:A" & Nodes)
        section_Series.Values = xlSection.Range("B1:B" & Nodes)
        section_Series.MarkerSize = 5
        section_Series.MarkerStyle = Excel.XlMarkerStyle.xlMarkerStyleSquare

        reinf_Series = section_SerCol.NewSeries
        reinf_Series.Name = "reinf"
        reinf_Series.XValues = xlSection.Range("D1:D" & REINFcount)
        reinf_Series.Values = xlSection.Range("E1:E" & REINFcount)
        reinf_Series.MarkerSize = 12
        reinf_Series.MarkerStyle = Excel.XlMarkerStyle.xlMarkerStyleCircle

        'max,min X & Y values
        Xaxis_max = 0
        Yaxis_max = 0
        Xaxis_min = 0
        Yaxis_min = 0

        For i = 1 To Nodes
            If nodeX(i) > Xaxis_max Then Xaxis_max = nodeX(i)
            If nodeY(i) > Yaxis_max Then Yaxis_max = nodeY(i)
        Next i
        If Xaxis_max > Yaxis_max Then
            axis_max = Xaxis_max
        Else
            axis_max = Yaxis_max
        End If

        For i = 1 To Nodes
            If nodeX(i) < Xaxis_min Then Xaxis_min = nodeX(i)
            If nodeY(i) < Yaxis_min Then Yaxis_min = nodeY(i)
        Next i
        If Xaxis_min < Yaxis_min Then
            axis_min = Xaxis_min
        Else
            axis_min = Yaxis_min
        End If

        If axis_max > Abs(axis_min) Then
```

```
            square_scale = axis_max
        Else
            square_scale = Abs(axis_min)
        End If

        s_width = Abs(Xaxis_min) + Abs(Xaxis_max)
        s_height = Abs(Yaxis_min) + Abs(Yaxis_max)
        xlSection.Cells(1, 3).value = s_width
        xlSection.Cells(2, 3).value = s_height

        With sectionplot
            .Legend.Delete()
            .HasTitle = True
            .ChartTitle.Text = lblSection.Text

            section_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = False
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines =
    True
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format.
    Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines =
    True
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScale = -
    (square_scale + square_scale / 10)
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).MaximumScale =
    (square_scale + square_scale / 10)
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).MajorUnit = Round(
    (square_scale + square_scale / 10) * 2 / 5, 1)
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition =
    Excel.XlTickLabelPosition.xlTickLabelPositionLow
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).CrossesAt = 0
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold =
     False
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size =
     14

            section_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
            section_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = False
            section_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
            section_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
            section_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line.
    DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            section_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScale = -
    (square_scale + square_scale / 10)
            section_AxisValue.Item(Excel.XlAxisType.xlValue).MaximumScale = (square_scale
     + square_scale / 10)
            section_AxisValue.Item(Excel.XlAxisType.xlValue).MajorUnit = Round(
    (square_scale + square_scale / 10) * 2 / 5, 1)
            section_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.
    XlTickLabelPosition.xlTickLabelPositionLow
            section_AxisValue.Item(Excel.XlAxisType.xlValue).CrossesAt = 0
            section_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold = False
            section_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
        End With
        sectionplot.Refresh()
        prgMAIN.Value = 3000

        'Average Temperatures
        With xlScratch

            .Range("J1:L1000").Clear()
            .Columns(10).NumberFormat = "0.000"
            .Columns(11).NumberFormat = "0"
            .Columns(12).NumberFormat = "0"
```

```
            For i = 0 To NumSteps
                .Cells(i + 1, 10).value = time(i) / 3600

                Tc_avg(i) = 0
                For j = 1 To Elements
                    Tc_avg(i) = Tc_avg(i) + cTEMP(j, i)
                Next j
                Tc_avg(i) = Tc_avg(i) / Elements
                .Cells(i + 1, 11).value = Tc_avg(i)

                Ts_avg(i) = 0
                For j = 1 To REINFcount
                    Ts_avg(i) = Ts_avg(i) + sTEMP(j, i)
                Next j
                Ts_avg(i) = Ts_avg(i) / REINFcount
                .Cells(i + 1, 12).value = Ts_avg(i)
            Next i
        End With

        Dim tempplot As Excel.Chart
        Dim temp_SerCol As Excel.SeriesCollection
        Dim tempC_Series, tempS_Series As Excel.Series
        Dim temp_AxisCategory, temp_AxisValue As Excel.Axes

        xlCharts = xlScratch.ChartObjects
        myChart = xlCharts.Add(612, 10, 900, 600)
        tempplot = myChart.Chart
        tempplot.ChartType = Excel.XlChartType.xlXYScatterLinesNoMarkers
        tempplot.ChartStyle = 2
        temp_SerCol = tempplot.SeriesCollection

        tempC_Series = temp_SerCol.NewSeries
        tempC_Series.Name = "Concrete"
        tempC_Series.XValues = xlScratch.Range("J1:J" & NumSteps + 1)
        tempC_Series.Values = xlScratch.Range("K1:K" & NumSteps + 1)
        tempC_Series.Format.Line.Weight = 1.5

        tempS_Series = temp_SerCol.NewSeries
        tempS_Series.Name = "Reinforcement"
        tempS_Series.XValues = xlScratch.Range("J1:J" & NumSteps + 1)
        tempS_Series.Values = xlScratch.Range("L1:L" & NumSteps + 1)
        tempS_Series.Format.Line.Weight = 1.5

        With tempplot
            .Legend.Position = Excel.XlLegendPosition.xlLegendPositionTop
            .HasTitle = True
            .ChartTitle.Text = "Average Section Temperature(time)"

            temp_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
            temp_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = True
            temp_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Text↙
    () = "Time, Hours"
            temp_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Font↙
    .Bold = False
            temp_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Font↙
    .Size = 14
            temp_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines = True
            temp_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines = True
            temp_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format.   ↙
    Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            temp_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScale = 0
            temp_AxisCategory.Item(Excel.XlAxisType.xlCategory).MaximumScale = EndTime / ↙
    3600
            temp_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition = Excel↙
    .XlTickLabelPosition.xlTickLabelPositionLow
            temp_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold =   ↙
```

```
    False
            temp_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size = 14

            temp_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
            temp_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = True
            temp_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Text() =
    "Temperature, C"
            temp_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Bold
    = False
            temp_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Size
    = 14
            temp_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
            temp_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
            temp_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line.
    DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            temp_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScale = 0
            temp_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.
    XlTickLabelPosition.xlTickLabelPositionLow
            temp_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold = False
            temp_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
        End With
        tempplot.Refresh()

        xlScratch.Cells(1, 6).value = "DONE"
        GoTo 99999

        'Save and close Excel file
99999:  txtRUN.Visible = False
        txtRUN1.Visible = False
        prgMAIN.Visible = False
        xlScratch.Cells(1, 1).value = 5
        xlUTFire.Close(True, OUTPUTfolder & "\temp\temp.xlsx")
        xlApp.Quit()

        releaseObject(xlApp)
        releaseObject(xlUTFire)
        releaseObject(xlOUT)
        releaseObject(xlREOUT)
        releaseObject(xlFire)
        releaseObject(xlSection)
        releaseObject(xlSteel)
        releaseObject(xlConcrete)
        releaseObject(xlMPhi)
        releaseObject(xlScratch)
        releaseObject(doop)

        Dim AllProcesses() As Process = Process.GetProcessesByName("excel")
        Dim ExcelProcess As Process
        For Each ExcelProcess In AllProcesses
            ExcelProcess.Kill()
        Next
        ExcelProcess = Nothing
        AllProcesses = Nothing

        Me.Visible = False

        'Show results form
        Dim Results As Results = New Results(txtFullPath_Section.Text, txtOutputFolder.
    Text)
        Results.Refresh()
        MessageBox.Show("Mission Accomplished!", "", MessageBoxButtons.OK)
        doop = True
        Me.Close()

    End Sub
```

```
    'INPUT_EVENTS
    Private Sub Input_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
    Me.Load
        Me.tabctrlMain.SelectTab(0)
        Me.tabctrlMain.Top = -22

        Dim AllProcesses() As Process = Process.GetProcessesByName("excel")
        Dim ExcelProcess As Process
        For Each ExcelProcess In AllProcesses
            ExcelProcess.Kill()
        Next
        ExcelProcess = Nothing
        AllProcesses = Nothing

        DUM = 0
        analysis_MPhi = 1
        analysis_MN = 1
        output_N = 1
        output_M = 1
        doop = False
        koop = False
        mphiCOUNT = 1000
        mnCOUNT = 50

        Me.Activate()
        Me.Visible = True
        If MessageBox.Show("Create new input file?", "NEW ANALYSIS / OPEN EXISTING",
    MessageBoxButtons.YesNo) = Windows.Forms.DialogResult.Yes Then
            f_new = True
            cSS_new = True
            sSS_new = True
            load1.Visible = False
            load2.Visible = False
            GoTo 100
        Else
            f_new = False
            GoTo 103
        End If

100:    'NEW FILE
        'Determine Output Folder and Filename
        Dim theFolderBrowser As New FolderBrowserDialog
101:    theFolderBrowser.Description = "Please select a folder for the output file."
        theFolderBrowser.ShowNewFolderButton = False
        theFolderBrowser.RootFolder = System.Environment.SpecialFolder.MyComputer
        theFolderBrowser.SelectedPath = My.Computer.FileSystem.SpecialDirectories.
    MyDocuments
        If theFolderBrowser.ShowDialog = Windows.Forms.DialogResult.OK Then
            txtOutputFolder.Text = theFolderBrowser.SelectedPath
            OUTPUTfolder = txtOutputFolder.Text
        Else
            Me.Activate()
            Me.Visible = True
            If MessageBox.Show("Nothing Selected", "", MessageBoxButtons.RetryCancel) =
    Windows.Forms.DialogResult.Retry Then
                GoTo 101
            Else
                End
            End If
        End If

102:    Me.Activate()
        Me.Visible = True
```

```
        Me.FILEname = InputBox("New project filename", "Create New Project", "UTFire")
        If FILEname = "" Then GoTo 102

        If System.IO.File.Exists(OUTPUTfolder & "\" & FILEname & ".xlsx") = False Then
    GoTo 1021
        If MessageBox.Show("File exists, please choose another name.", "",
    MessageBoxButtons.RetryCancel) = Windows.Forms.DialogResult.Retry Then
            GoTo 102
        Else
            End
        End If

1021:   'Define Worksheets (create new then rename sheets)
        xlApp = New Excel.ApplicationClass
        xlUTFire = xlApp.Workbooks.Add(misValue)
        xlOUT = xlUTFire.Sheets(1)
        xlOUT.Name = ".out"
        xlREOUT = xlUTFire.Sheets(2)
        xlREOUT.Name = "re.out"
        xlFire = xlUTFire.Sheets(3)
        xlFire.Name = "fire"
        xlUTFire.Sheets.Add(After:=xlFire)
        xlSection = xlUTFire.Sheets(4)
        xlSection.Name = "section"
        xlUTFire.Sheets.Add(After:=xlSection)
        xlConcrete = xlUTFire.Sheets(5)
        xlConcrete.Name = "concrete"
        xlUTFire.Sheets.Add(After:=xlConcrete)
        xlSteel = xlUTFire.Sheets(6)
        xlSteel.Name = "steel"
        xlUTFire.Sheets.Add(After:=xlSteel)
        xlMPhi = xlUTFire.Sheets(7)
        xlMPhi.Name = "MPhi"
        xlUTFire.Sheets.Add(After:=xlMPhi)
        xlScratch = xlUTFire.Sheets(8)
        xlScratch.Name = "scratch"
        xlUTFire.Sheets.Add(After:=xlScratch)
        xlMN = xlUTFire.Sheets(9)
        xlMN.Name = "MN"
        xlUTFire.Sheets.Add(After:=xlMN)
        xlDOOP = xlUTFire.Sheets(10)
        xlDOOP.Name = "doop"


        txtFullPath_Section.Text = OUTPUTfolder & "\" & FILEname & ".xlsx"
        txtFullPath_Fire.Text = OUTPUTfolder & "\" & FILEname & ".xlsx"
        xlScratch.Cells(6, 1).value = "TRUE"
        xlScratch.Cells(7, 1).value = "TRUE"
        xlScratch.Cells(8, 1).value = "TRUE"

        xlUTFire.SaveAs(OUTPUTfolder & "\" & FILEname & ".xlsx")

        xlREOUT.Cells(20, 2).value = 22
        xlMPhi.Cells(3, 1).value = mphiCOUNT
        xlMN.Cells(1, 1).value = mnCOUNT

        If My.Computer.FileSystem.DirectoryExists(OUTPUTfolder & "\temp") = True Then
    GoTo 9999
        My.Computer.FileSystem.CreateDirectory(OUTPUTfolder & "\temp")
        GoTo 9999



103:    'OPEN EXISTING FILE
        Me.Activate()
        Me.Visible = True
```

```
        load1.Visible = True
        load2.Visible = True

        Dim full_existing As String

        With opnExisting
            .Filter = "Excel07 Files (*.xlsx)|*.xlsx"
            .InitialDirectory = "C:\"
            .Title = "Select Existing File to Load"
            If .ShowDialog = Windows.Forms.DialogResult.OK Then
                full_existing = .FileName
            Else
                GoTo 9998
            End If
        End With

        'Convert.ToString()
        OUTPUTfolder = GetDirectoryName(FullPath:=full_existing)
        txtOutputFolder.Text = OUTPUTfolder

        load1.Text = "Loading..."
        FILEname = GetFileNameWithoutExtension(FileOnly:=full_existing)
        txtFullPath_Section.Text = OUTPUTfolder & "\" & FILEname & ".xlsx"
        txtFullPath_Fire.Text = OUTPUTfolder & "\" & FILEname & ".xlsx"

        'Define Worksheets
        load1.Text = "Loading...."
        xlApp = New Excel.ApplicationClass
        xlUTFire = xlApp.Workbooks.Add(OUTPUTfolder & "\" & FILEname & ".xlsx")
        xlOUT = xlUTFire.Sheets(".out")
        xlREOUT = xlUTFire.Sheets("re.out")
        xlFire = xlUTFire.Sheets("fire")
        xlSection = xlUTFire.Sheets("section")
        xlConcrete = xlUTFire.Sheets("concrete")
        xlSteel = xlUTFire.Sheets("steel")
        xlMPhi = xlUTFire.Sheets("MPhi")
        xlScratch = xlUTFire.Sheets("scratch")
        xlMN = xlUTFire.Sheets("MN")
        xlDOOP = xlUTFire.Sheets("doop")
        load1.Text = "Loading....."

        If My.Computer.FileSystem.DirectoryExists(OUTPUTfolder & "\temp") = True Then
    GoTo 1031
        My.Computer.FileSystem.CreateDirectory(OUTPUTfolder & "\temp")

1031:   If xlScratch.Cells(1, 1).value > 0 Then GoTo 104
        xlScratch.Cells(6, 1).value = "TRUE"
        f_new = True
        cSS_new = True
        sSS_new = True
        xlREOUT.Cells(20, 2).value = 22
        xlMPhi.Cells(3, 1).value = mphiCOUNT
        xlMN.Cells(1, 1).value = mnCOUNT
        GoTo 9999

        'LOAD INPUT DATA FROM EXISTING WORKBOOK=========================================
    ===
104:    xlScratch.Cells(6, 1).value = "FALSE"
        f_new = False
        load1.Text = "Loading......"
        If xlScratch.Cells(7, 1).value = "FALSE" Then cSS_new = False
        If xlScratch.Cells(7, 1).value = "TRUE" Then cSS_new = True
        If xlScratch.Cells(8, 1).value = "FALSE" Then cSS_new = False
        If xlScratch.Cells(8, 1).value = "TRUE" Then cSS_new = True
```

```
        If xlScratch.Cells(1, 1).value = 1 Then GoTo 1000
        If xlScratch.Cells(1, 1).value = 2 Then GoTo 3000
        If xlScratch.Cells(1, 1).value = 3 Then GoTo 3000
        If xlScratch.Cells(1, 1).value = 4 Then GoTo 4000
        If xlScratch.Cells(1, 1).value = 5 Then GoTo 5000

5000:   'Analysis Options Completed
        txtProgress5.BackColor = Color.DodgerBlue
        txtProgress5.ForeColor = Color.White
        txtProgress4.BackColor = Color.DodgerBlue
        txtProgress4.ForeColor = Color.White
        txtProgress3.BackColor = Color.DodgerBlue
        txtProgress3.ForeColor = Color.White
        txtProgress2.BackColor = Color.DodgerBlue
        txtProgress2.ForeColor = Color.White
        txtProgress1.BackColor = Color.DodgerBlue
        txtProgress1.ForeColor = Color.White

        load1.Text = "Loading......."

        txtAccuracy.Text = xlMPhi.Cells(4, 1).value
        mphiCOUNT = 1000
        mnCOUNT = 50


4000:   'Reinforcement Completed
        txtProgress4.BackColor = Color.DodgerBlue
        txtProgress4.ForeColor = Color.White
        txtProgress3.BackColor = Color.DodgerBlue
        txtProgress3.ForeColor = Color.White
        txtProgress2.BackColor = Color.DodgerBlue
        txtProgress2.ForeColor = Color.White
        txtProgress1.BackColor = Color.DodgerBlue
        txtProgress1.ForeColor = Color.White

        load1.Text = "Loading........"

        REINFcount = xlREOUT.Cells(23, 2).Value
        If REINFcount = 18 Then GoTo 4018
        If REINFcount = 17 Then GoTo 4017
        If REINFcount = 16 Then GoTo 4016
        If REINFcount = 15 Then GoTo 4015
        If REINFcount = 14 Then GoTo 4014
        If REINFcount = 13 Then GoTo 4013
        If REINFcount = 12 Then GoTo 4012
        If REINFcount = 11 Then GoTo 4011
        If REINFcount = 10 Then GoTo 4010
        If REINFcount = 9 Then GoTo 4009
        If REINFcount = 8 Then GoTo 4008
        If REINFcount = 7 Then GoTo 4007
        If REINFcount = 6 Then GoTo 4006
        If REINFcount = 5 Then GoTo 4005
        If REINFcount = 4 Then GoTo 4004
        If REINFcount = 3 Then GoTo 4003
        If REINFcount = 2 Then GoTo 4002
        If REINFcount = 1 Then GoTo 4001

4018:   cmbBarSize18.Text = xlREOUT.Cells(19, 38).value
        txtReinfX18.Text = xlREOUT.Cells(19, 39).value
        txtReinfY18.Text = xlREOUT.Cells(19, 40).value
4017:   cmbBarSize17.Text = xlREOUT.Cells(18, 38).value
        txtReinfX17.Text = xlREOUT.Cells(18, 39).value
        txtReinfY17.Text = xlREOUT.Cells(18, 40).value
4016:   cmbBarSize16.Text = xlREOUT.Cells(17, 38).value
        txtReinfX16.Text = xlREOUT.Cells(17, 39).value
        txtReinfY16.Text = xlREOUT.Cells(17, 40).value
```

```
4015:   cmbBarSize15.Text = xlREOUT.Cells(16, 38).value
        txtReinfX15.Text = xlREOUT.Cells(16, 39).value
        txtReinfY15.Text = xlREOUT.Cells(16, 40).value
4014:   cmbBarSize14.Text = xlREOUT.Cells(15, 38).value
        txtReinfX14.Text = xlREOUT.Cells(15, 39).value
        txtReinfY14.Text = xlREOUT.Cells(15, 40).value
4013:   cmbBarSize13.Text = xlREOUT.Cells(14, 38).value
        txtReinfX13.Text = xlREOUT.Cells(14, 39).value
        txtReinfY13.Text = xlREOUT.Cells(14, 40).value
4012:   cmbBarSize12.Text = xlREOUT.Cells(13, 38).value
        txtReinfX12.Text = xlREOUT.Cells(13, 39).value
        txtReinfY12.Text = xlREOUT.Cells(13, 40).value
4011:   cmbBarSize11.Text = xlREOUT.Cells(12, 38).value
        txtReinfX11.Text = xlREOUT.Cells(12, 39).value
        txtReinfY11.Text = xlREOUT.Cells(12, 40).value
4010:   cmbBarSize10.Text = xlREOUT.Cells(11, 38).value
        txtReinfX10.Text = xlREOUT.Cells(11, 39).value
        txtReinfY10.Text = xlREOUT.Cells(11, 40).value
4009:   cmbBarSize9.Text = xlREOUT.Cells(10, 38).value
        txtReinfX9.Text = xlREOUT.Cells(10, 39).value
        txtReinfY9.Text = xlREOUT.Cells(10, 40).value
4008:   cmbBarSize8.Text = xlREOUT.Cells(9, 38).value
        txtReinfX8.Text = xlREOUT.Cells(9, 39).value
        txtReinfY8.Text = xlREOUT.Cells(9, 40).value
4007:   cmbBarSize7.Text = xlREOUT.Cells(8, 38).value
        txtReinfX7.Text = xlREOUT.Cells(8, 39).value
        txtReinfY7.Text = xlREOUT.Cells(8, 40).value
4006:   cmbBarSize6.Text = xlREOUT.Cells(7, 38).value
        txtReinfX6.Text = xlREOUT.Cells(7, 39).value
        txtReinfY6.Text = xlREOUT.Cells(7, 40).value
4005:   cmbBarSize5.Text = xlREOUT.Cells(6, 38).value
        txtReinfX5.Text = xlREOUT.Cells(6, 39).value
        txtReinfY5.Text = xlREOUT.Cells(6, 40).value
4004:   cmbBarSize4.Text = xlREOUT.Cells(5, 38).value
        txtReinfX4.Text = xlREOUT.Cells(5, 39).value
        txtReinfY4.Text = xlREOUT.Cells(5, 40).value
4003:   cmbBarSize3.Text = xlREOUT.Cells(4, 38).value
        txtReinfX3.Text = xlREOUT.Cells(4, 39).value
        txtReinfY3.Text = xlREOUT.Cells(4, 40).value
4002:   cmbBarSize2.Text = xlREOUT.Cells(3, 38).value
        txtReinfX2.Text = xlREOUT.Cells(3, 39).value
        txtReinfY2.Text = xlREOUT.Cells(3, 40).value
4001:   cmbBarSize1.Text = xlREOUT.Cells(2, 38).value
        txtReinfX1.Text = xlREOUT.Cells(2, 39).value
        txtReinfY1.Text = xlREOUT.Cells(2, 40).value



3000:   'Material Properties Completed
        txtProgress3.BackColor = Color.DodgerBlue
        txtProgress3.ForeColor = Color.White
        txtProgress2.BackColor = Color.DodgerBlue
        txtProgress2.ForeColor = Color.White
        txtProgress1.BackColor = Color.DodgerBlue
        txtProgress1.ForeColor = Color.White

        load1.Text = "Loading........."

        txtConcreteStrength.Text = xlREOUT.Cells(11, 2).value
        fprimec = xlREOUT.Cells(11, 2).value
        cmbAggregate.Text = xlREOUT.Cells(12, 2).value
        aggregates = xlREOUT.Cells(12, 2).value
        cmbConcreteModel.Text = xlREOUT.Cells(13, 2).value

        txtSteelProportional.Text = xlREOUT.Cells(15, 2).value
        fp = xlREOUT.Cells(15, 2).value
```

```
        txtSteelYield.Text = xlREOUT.Cells(16, 2).value
        fy = xlREOUT.Cells(16, 2).value
        cmbSteelModel.Text = xlREOUT.Cells(17, 2).value


1000:   'Load SAFIR.out File Completed
        txtProgress1.BackColor = Color.DodgerBlue
        txtProgress1.ForeColor = Color.White

        load1.Text = "Loading..........."

        txtFilePath.Text = xlScratch.Cells(2, 1).value
        lblSection.Text = xlScratch.Cells(3, 1).value
        lblNodes.Text = xlREOUT.Cells(3, 2).value
        Nodes = lblNodes.Text
        lblElements.Text = xlREOUT.Cells(4, 2).value
        Elements = lblElements.Text
        lblTimeStep.Text = xlREOUT.Cells(7, 2).value
        TimeStep = lblTimeStep.Text
        lblEndTime.Text = xlREOUT.Cells(8, 2).value
        EndTime = lblEndTime.Text
        lblNumSteps.Text = xlREOUT.Cells(9, 2).value
        NumSteps = lblNumSteps.Text
        lblFire.Text = xlScratch.Cells(4, 1).value
        lblFirePath.Text = xlScratch.Cells(5, 1).value

        If xlScratch.Cells(6, 1).value = "TRUE" Then f_new = True
        If xlScratch.Cells(6, 1).value = "FALSE" Then f_new = False
        If xlScratch.Cells(7, 1).value = "TRUE" Then cSS_new = True
        If xlScratch.Cells(7, 1).value = "FALSE" Then cSS_new = False
        If xlScratch.Cells(8, 1).value = "TRUE" Then sSS_new = True
        If xlScratch.Cells(8, 1).value = "FALSE" Then sSS_new = False

        For i = 1 To Nodes
            nodeX(i) = xlREOUT.Cells(i + 1, 5).value
            nodeY(i) = xlREOUT.Cells(i + 1, 6).value
        Next i

        For i = 1 To Elements
            nodeA(i) = xlREOUT.Cells(i + 1, 10).value
            nodeB(i) = xlREOUT.Cells(i + 1, 13).value
            nodeC(i) = xlREOUT.Cells(i + 1, 16).value
            nodeD(i) = xlREOUT.Cells(i + 1, 19).value
            elementAREA(i) = xlREOUT.Cells(i + 1, 29).value
            element_Y(i) = xlREOUT.Cells(i + 1, 24).value
        Next i

        'PLOT FIRE FILE
        load1.Text = "Loading............."
        If f_new = True Then GoTo 1001
        xlFire.ChartObjects(1).Delete()

1001:   Dim xlCharts As Excel.ChartObjects
        Dim myChart As Excel.ChartObject
        Dim fireplot As Excel.Chart
        Dim fire_SerCol As Excel.SeriesCollection
        Dim fire_Series As Excel.Series
        Dim fire_AxisCategory, fire_AxisValue As Excel.Axes

        xlCharts = xlFire.ChartObjects
        myChart = xlCharts.Add(120, 10, 900, 600)
        fireplot = myChart.Chart
        fireplot.ChartType = Excel.XlChartType.xlXYScatterLinesNoMarkers
        fireplot.ChartStyle = 2
        fire_SerCol = fireplot.SeriesCollection
```

```
        fire_Series = fire_SerCol.NewSeries
        fire_Series.Name = "fire"
        fire_Series.XValues = xlFire.Range("C1:C" & NumSteps)
        fire_Series.Values = xlFire.Range("B1:B" & NumSteps)
        fire_Series.Format.Line.Weight = 1.5

        With fireplot
            .Legend.Delete()
            .HasTitle = True
            .ChartTitle.Text = lblFire.Text

            fire_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
            fire_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = True
            fire_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Text
    () = "Time, Hours"
            fire_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Font
    .Bold = False
            fire_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Font
    .Size = 14
            fire_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines = True
            fire_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines = True
            fire_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format.
    Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            fire_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScale = 0
            fire_AxisCategory.Item(Excel.XlAxisType.xlCategory).MaximumScale = EndTime /
    3600
            fire_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition = Excel
    .XlTickLabelPosition.xlTickLabelPositionLow
            fire_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold =
    False
            fire_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size = 14

            fire_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
            fire_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = True
            fire_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Text() =
    "Temperature, C"
            fire_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Bold
    = False
            fire_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Size
    = 14
            fire_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
            fire_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
            fire_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line.
    DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            fire_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScale = 0
            fire_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.
    XlTickLabelPosition.xlTickLabelPositionLow
            fire_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold = False
            fire_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
        End With
        fireplot.Refresh()

        xlFire.ChartObjects(1).chart.Export(FILEname:=OUTPUTfolder & "\temp" & "\fireplot
    .bmp", FilterName:="BMP")
        plotFire.BackgroundImage = New System.Drawing.Bitmap(OUTPUTfolder & "\temp" & "\
    fireplot.bmp")
        plotFire.BackgroundImageLayout = ImageLayout.Stretch


        'Plot Section
        load1.Text = "Loading................."
        If f_new = True Then GoTo 1002
        xlSection.ChartObjects(1).Delete()

1002:   With xlSection
            For i = 1 To Nodes
```

```
            .Cells(i, 1) = nodeX(i)
            .Cells(i, 2) = nodeY(i)
        Next i
    End With

    Dim sectionplot As Excel.Chart
    Dim section_SerCol As Excel.SeriesCollection
    Dim section_Series, reinf_Series As Excel.Series
    Dim section_AxisCategory, section_AxisValue As Excel.Axes

    xlCharts = xlSection.ChartObjects
    myChart = xlCharts.Add(260, 10, 900, 600)
    sectionplot = myChart.Chart
    sectionplot.ChartType = Excel.XlChartType.xlXYScatter
    section_SerCol = sectionplot.SeriesCollection

    section_Series = section_SerCol.NewSeries
    section_Series.Name = "section"
    section_Series.XValues = xlSection.Range("A1:A" & Nodes)
    section_Series.Values = xlSection.Range("B1:B" & Nodes)
    section_Series.MarkerSize = 5
    section_Series.MarkerStyle = Excel.XlMarkerStyle.xlMarkerStyleSquare

    reinf_Series = section_SerCol.NewSeries
    reinf_Series.Name = "reinf"
    reinf_Series.XValues = xlSection.Range("D1:D" & REINFcount)
    reinf_Series.Values = xlSection.Range("E1:E" & REINFcount)
    reinf_Series.MarkerSize = 12
    reinf_Series.MarkerStyle = Excel.XlMarkerStyle.xlMarkerStyleCircle

    Xaxis_max = 0
    Yaxis_max = 0
    Xaxis_min = 0
    Yaxis_min = 0

    For i = 1 To Nodes
        If nodeX(i) > Xaxis_max Then Xaxis_max = nodeX(i)
        If nodeY(i) > Yaxis_max Then Yaxis_max = nodeY(i)
    Next i
    If Xaxis_max > Yaxis_max Then
        axis_max = Xaxis_max
    Else
        axis_max = Yaxis_max
    End If

    load1.Text = "Loading..............."

    For i = 1 To Nodes
        If nodeX(i) < Xaxis_min Then Xaxis_min = nodeX(i)
        If nodeY(i) < Yaxis_min Then Yaxis_min = nodeY(i)
    Next i
    If Xaxis_min < Yaxis_min Then
        axis_min = Xaxis_min
    Else
        axis_min = Yaxis_min
    End If

    If axis_max > Abs(axis_min) Then
        square_scale = axis_max
    Else
        square_scale = Abs(axis_min)
    End If

    s_width = Abs(Xaxis_min) + Abs(Xaxis_max)
    s_height = Abs(Yaxis_min) + Abs(Yaxis_max)
    xlSection.Cells(1, 3).value = s_width
```

```
        xlSection.Cells(2, 3).value = s_height

        With sectionplot
            .Legend.Delete()
            .HasTitle = True
            .ChartTitle.Text = lblSection.Text

            section_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = False
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines =
    True
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format.
    Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines =
    True
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScale = -
    (square_scale + square_scale / 10)
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).MaximumScale =
    (square_scale + square_scale / 10)
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).MajorUnit = Round(
    (square_scale + square_scale / 10) * 2 / 5, 1)
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition =
    Excel.XlTickLabelPosition.xlTickLabelPositionLow
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).CrossesAt = 0
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold =
     False
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size =
     14

            section_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
            section_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = False
            section_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
            section_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
            section_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line.
    DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            section_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScale = -
    (square_scale + square_scale / 10)
            section_AxisValue.Item(Excel.XlAxisType.xlValue).MaximumScale = (square_scale
     + square_scale / 10)
            section_AxisValue.Item(Excel.XlAxisType.xlValue).MajorUnit = Round(
    (square_scale + square_scale / 10) * 2 / 5, 1)
            section_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.
    XlTickLabelPosition.xlTickLabelPositionLow
            section_AxisValue.Item(Excel.XlAxisType.xlValue).CrossesAt = 0
            section_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold = False
            section_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
        End With
        sectionplot.Refresh()

        xlSection.ChartObjects(1).chart.Export(FILEname:=OUTPUTfolder & "\temp" & "\
    sectionplot.bmp", FilterName:="BMP")
        plotSection.BackgroundImage = New System.Drawing.Bitmap(OUTPUTfolder & "\temp" &
    "\sectionplot.bmp")
        plotSection.BackgroundImageLayout = ImageLayout.Stretch


        GoTo 9999
9998:   End
9999:   imgStart.Visible = True
        start1.Visible = True
        start2.Visible = True
        start3.Visible = True
        start4.Visible = True
        Me.load1.Visible = False
        Me.load2.Visible = False
```

```
    End Sub
    Private Sub Input_FormClosing(ByVal sender As Object, ByVal e As System.Windows.Forms↙
    .FormClosingEventArgs) Handles Me.FormClosing

        If koop = True Then GoTo 999

        Me.tabctrlMain.SelectTab(5)

        'Refresh count
        DUM = DUM + 1

        'Make sure all bars all written to Excel
        xlREOUT.Cells(2, 38).value = cmbBarSize1.Text
        xlREOUT.Cells(3, 38).value = cmbBarSize2.Text
        xlREOUT.Cells(4, 38).value = cmbBarSize3.Text
        xlREOUT.Cells(5, 38).value = cmbBarSize4.Text
        xlREOUT.Cells(6, 38).value = cmbBarSize5.Text
        xlREOUT.Cells(7, 38).value = cmbBarSize6.Text
        xlREOUT.Cells(8, 38).value = cmbBarSize7.Text
        xlREOUT.Cells(9, 38).value = cmbBarSize8.Text
        xlREOUT.Cells(10, 38).value = cmbBarSize9.Text
        xlREOUT.Cells(11, 38).value = cmbBarSize10.Text
        xlREOUT.Cells(12, 38).value = cmbBarSize11.Text
        xlREOUT.Cells(13, 38).value = cmbBarSize12.Text
        xlREOUT.Cells(14, 38).value = cmbBarSize13.Text
        xlREOUT.Cells(15, 38).value = cmbBarSize14.Text
        xlREOUT.Cells(16, 38).value = cmbBarSize15.Text
        xlREOUT.Cells(17, 38).value = cmbBarSize16.Text
        xlREOUT.Cells(18, 38).value = cmbBarSize17.Text
        xlREOUT.Cells(19, 38).value = cmbBarSize18.Text
        xlREOUT.Cells(2, 39).value = txtReinfX1.Text
        xlREOUT.Cells(3, 39).value = txtReinfX2.Text
        xlREOUT.Cells(4, 39).value = txtReinfX3.Text
        xlREOUT.Cells(5, 39).value = txtReinfX4.Text
        xlREOUT.Cells(6, 39).value = txtReinfX5.Text
        xlREOUT.Cells(7, 39).value = txtReinfX6.Text
        xlREOUT.Cells(8, 39).value = txtReinfX7.Text
        xlREOUT.Cells(9, 39).value = txtReinfX8.Text
        xlREOUT.Cells(10, 39).value = txtReinfX9.Text
        xlREOUT.Cells(11, 39).value = txtReinfX10.Text
        xlREOUT.Cells(12, 39).value = txtReinfX11.Text
        xlREOUT.Cells(13, 39).value = txtReinfX12.Text
        xlREOUT.Cells(14, 39).value = txtReinfX13.Text
        xlREOUT.Cells(15, 39).value = txtReinfX14.Text
        xlREOUT.Cells(16, 39).value = txtReinfX15.Text
        xlREOUT.Cells(17, 39).value = txtReinfX16.Text
        xlREOUT.Cells(18, 39).value = txtReinfX17.Text
        xlREOUT.Cells(19, 39).value = txtReinfX18.Text
        xlREOUT.Cells(2, 40).value = txtReinfY1.Text
        xlREOUT.Cells(3, 40).value = txtReinfY2.Text
        xlREOUT.Cells(4, 40).value = txtReinfY3.Text
        xlREOUT.Cells(5, 40).value = txtReinfY4.Text
        xlREOUT.Cells(6, 40).value = txtReinfY5.Text
        xlREOUT.Cells(7, 40).value = txtReinfY6.Text
        xlREOUT.Cells(8, 40).value = txtReinfY7.Text
        xlREOUT.Cells(9, 40).value = txtReinfY8.Text
        xlREOUT.Cells(10, 40).value = txtReinfY9.Text
        xlREOUT.Cells(11, 40).value = txtReinfY10.Text
        xlREOUT.Cells(12, 40).value = txtReinfY11.Text
        xlREOUT.Cells(13, 40).value = txtReinfY12.Text
        xlREOUT.Cells(14, 40).value = txtReinfY13.Text
        xlREOUT.Cells(15, 40).value = txtReinfY14.Text
        xlREOUT.Cells(16, 40).value = txtReinfY15.Text
        xlREOUT.Cells(17, 40).value = txtReinfY16.Text
        xlREOUT.Cells(18, 40).value = txtReinfY17.Text
        xlREOUT.Cells(19, 40).value = txtReinfY18.Text
```

```
        'Count reinforcing bars
        For i = 1 To 18
            If xlREOUT.Cells(i + 1, 38).value = 0 Or xlREOUT.Cells(i + 1, 39).value = 0
    Or xlREOUT.Cells(i + 1, 40).value = 0 Then
                REINFcount = i - 1
                GoTo 200
            End If
        Next i
        REINFcount = 18

200:    xlREOUT.Cells(23, 2).value = REINFcount

        For i = REINFcount + 1 To 19
            xlREOUT.Cells(i + 1, 38).value = ""
            xlREOUT.Cells(i + 1, 39).value = ""
            xlREOUT.Cells(i + 1, 40).value = ""
        Next i

        'Convert input to SAFIR coordinate system
        For i = 1 To REINFcount
            xlSection.Cells(i, 4).value = -s_width / 2 + xlREOUT.Cells(i + 1, 39).value
            xlREOUT.Cells(i + 1, 41).value = -s_width / 2 + xlREOUT.Cells(i + 1, 39).
    value
            xlSection.Cells(i, 5).value = -s_height / 2 + xlREOUT.Cells(i + 1, 40).value
            xlREOUT.Cells(i + 1, 42).value = -s_height / 2 + xlREOUT.Cells(i + 1, 40).
    value
        Next i

        If txtProgress1.BackColor = Color.DodgerBlue Then
            xlScratch.Cells(1, 1).value = 1
        Else
            GoTo 100
        End If
        If txtProgress2.BackColor = Color.DodgerBlue And txtProgress3.BackColor = Color.
    DodgerBlue Then
            xlScratch.Cells(1, 1).value = 3
        Else
            GoTo 100
        End If
        If txtProgress4.BackColor = Color.DodgerBlue Then
            xlScratch.Cells(1, 1).value = 4
        Else
            GoTo 100
        End If
        If txtProgress5.BackColor = Color.DodgerBlue Then
            xlScratch.Cells(1, 1).value = 5
        Else
            GoTo 100
        End If

100:    MessageBox.Show("Saving input as.... " & OUTPUTfolder & "\" & FILEname & ".xlsx",
     "Closing...", MessageBoxButtons.OK)

        xlUTFire.Close(True, OUTPUTfolder & "\" & FILEname & ".xlsx")
        xlApp.Quit()

        releaseObject(xlApp)
        releaseObject(xlUTFire)
        releaseObject(xlOUT)
        releaseObject(xlREOUT)
        releaseObject(xlFire)
        releaseObject(xlSection)
        releaseObject(xlSteel)
        releaseObject(xlConcrete)
        releaseObject(xlMPhi)
```

```
        releaseObject(xlScratch)
        releaseObject(xlMN)
        releaseObject(xlDOOP)

        Dim AllProcesses() As Process = Process.GetProcessesByName("excel")
        Dim ExcelProcess As Process
        For Each ExcelProcess In AllProcesses
            ExcelProcess.Kill()
        Next
        ExcelProcess = Nothing
        AllProcesses = Nothing

999:
    End Sub
    Private Sub Input_FormClosed(ByVal sender As Object, ByVal e As System.Windows.Forms.↙
    FormClosedEventArgs) Handles Me.FormClosed

        plotSS.BackgroundImage.Dispose()
        plotSection.BackgroundImage.Dispose()
        plotFire.BackgroundImage.Dispose()

        If doop = True Then GoTo 999

        Dim s As String
        For Each s In System.IO.Directory.GetFiles(OUTPUTfolder & "\temp")
            System.IO.File.Delete(s)
        Next s

999:
    End Sub



    'PROGRESS_BAR
    Private Sub txtProgress1_BackColorChanged(ByVal sender As Object, ByVal e As System.↙
    EventArgs) Handles txtProgress1.BackColorChanged

        If txtProgress1.BackColor = Color.DodgerBlue Then
            btnView.Enabled = True
            btnMaterials.Enabled = True
        End If

        If txtProgress1.BackColor = Color.DodgerBlue And txtProgress2.BackColor = Color.↙
    DodgerBlue _
        And txtProgress3.BackColor = Color.DodgerBlue And txtProgress4.BackColor = Color.↙
    DodgerBlue _
        And txtProgress5.BackColor = Color.DodgerBlue Then
            btnSubmit.Enabled = True
        Else
            btnSubmit.Enabled = False
        End If

    End Sub
    Private Sub txtProgress2_BackColorChanged(ByVal sender As Object, ByVal e As System.↙
    EventArgs) Handles txtProgress2.BackColorChanged

        If txtProgress2.BackColor = Color.DodgerBlue And txtProgress3.BackColor = Color.↙
    DodgerBlue Then
            btnReinforcement.Enabled = True
        End If

        If txtProgress1.BackColor = Color.DodgerBlue And txtProgress2.BackColor = Color.↙
    DodgerBlue _
        And txtProgress3.BackColor = Color.DodgerBlue And txtProgress4.BackColor = Color.↙
    DodgerBlue _
```

```
        And txtProgress5.BackColor = Color.DodgerBlue Then
            btnSubmit.Enabled = True
        Else
            btnSubmit.Enabled = False
        End If

    End Sub
    Private Sub txtProgress3_BackColorChanged(ByVal sender As Object, ByVal e As System. ↙
    EventArgs) Handles txtProgress3.BackColorChanged

        If txtProgress2.BackColor = Color.DodgerBlue And txtProgress3.BackColor = Color. ↙
    DodgerBlue Then
            btnReinforcement.Enabled = True
        End If

        If txtProgress1.BackColor = Color.DodgerBlue And txtProgress2.BackColor = Color. ↙
    DodgerBlue _
        And txtProgress3.BackColor = Color.DodgerBlue And txtProgress4.BackColor = Color.↙
    DodgerBlue _
        And txtProgress5.BackColor = Color.DodgerBlue Then
            btnSubmit.Enabled = True
        Else
            btnSubmit.Enabled = False
        End If

    End Sub
    Private Sub txtProgress4_BackColorChanged(ByVal sender As Object, ByVal e As System. ↙
    EventArgs) Handles txtProgress4.BackColorChanged

        If txtProgress4.BackColor = Color.DodgerBlue Then
            btnAnalysis.Enabled = True
            btnPreviewSection.Enabled = True
            btnThermalEnvironment.Enabled = True
        End If

        If txtProgress1.BackColor = Color.DodgerBlue And txtProgress2.BackColor = Color. ↙
    DodgerBlue _
        And txtProgress3.BackColor = Color.DodgerBlue And txtProgress4.BackColor = Color.↙
    DodgerBlue _
        And txtProgress5.BackColor = Color.DodgerBlue Then
            btnSubmit.Enabled = True
        Else
            btnSubmit.Enabled = False
        End If

    End Sub
    Private Sub txtProgress5_BackColorChanged(ByVal sender As Object, ByVal e As System. ↙
    EventArgs) Handles txtProgress5.BackColorChanged

        If txtProgress1.BackColor = Color.DodgerBlue And txtProgress2.BackColor = Color. ↙
    DodgerBlue _
        And txtProgress3.BackColor = Color.DodgerBlue And txtProgress4.BackColor = Color.↙
    DodgerBlue _
        And txtProgress5.BackColor = Color.DodgerBlue Then
            btnSubmit.Enabled = True
        Else
            btnSubmit.Enabled = False
        End If

    End Sub



    'MENU_STRIP
    Private Sub NewAnalysisToolStripMenuItem1_Click(ByVal sender As System.Object, ByVal ↙
```

```
    e As System.EventArgs) Handles NewAnalysisToolStripMenuItem1.Click

    End Sub
    Private Sub OpenExistingToolStripMenuItem_Click(ByVal sender As System.Object, ByVal ↙
    e As System.EventArgs) Handles OpenExistingToolStripMenuItem.Click

    End Sub
    Private Sub SaveCurrentToolStripMenuItem1_Click(ByVal sender As System.Object, ByVal ↙
    e As System.EventArgs) Handles SaveCurrentToolStripMenuItem1.Click

        If txtProgress1.BackColor = Color.DodgerBlue Then
            xlScratch.Cells(1, 1).value = 1
        Else
            GoTo 100
        End If
        If txtProgress2.BackColor = Color.DodgerBlue And txtProgress3.BackColor = Color. ↙
    DodgerBlue Then
            xlScratch.Cells(1, 1).value = 3
        Else
            GoTo 100
        End If
        If txtProgress4.BackColor = Color.DodgerBlue Then
            xlScratch.Cells(1, 1).value = 4
        Else
            GoTo 100
        End If
        If txtProgress5.BackColor = Color.DodgerBlue Then
            xlScratch.Cells(1, 1).value = 5
        Else
            GoTo 100
        End If

100:    xlUTFire.SaveAs(OUTPUTfolder & "\" & FILEname & ".xlsx")

    End Sub
    Private Sub SaveAsToolStripMenuItem1_Click(ByVal sender As System.Object, ByVal e As ↙
    System.EventArgs) Handles SaveAsToolStripMenuItem1.Click

        If txtProgress1.BackColor = Color.DodgerBlue Then
            xlScratch.Cells(1, 1).value = 1
        Else
            GoTo 100
        End If
        If txtProgress2.BackColor = Color.DodgerBlue And txtProgress3.BackColor = Color. ↙
    DodgerBlue Then
            xlScratch.Cells(1, 1).value = 3
        Else
            GoTo 100
        End If
        If txtProgress4.BackColor = Color.DodgerBlue Then
            xlScratch.Cells(1, 1).value = 4
        Else
            GoTo 100
        End If
        If txtProgress5.BackColor = Color.DodgerBlue Then
            xlScratch.Cells(1, 1).value = 5
        Else
            GoTo 100
        End If

100:    xlUTFire.SaveAs(OUTPUTfolder & "\" & FILEname & ".xlsx")

    End Sub
    Public Sub ExitToolStripMenuItem1_Click(ByVal sender As System.Object, ByVal e As    ↙
    System.EventArgs) Handles ExitToolStripMenuItem1.Click
```

```
        Me.Close()

    End Sub
    Private Sub HelpToolStripMenuItem3_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles HelpToolStripMenuItem3.Click
        Dim Help As New Help
        Help.ShowDialog()
        Help.Refresh()
    End Sub
    Private Sub AboutToolStripMenuItem1_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles AboutToolStripMenuItem1.Click
        Dim About As New About
        About.ShowDialog()
        About.Refresh()
    End Sub



    '(0)START_
    Private Sub imgStart_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles imgStart.Click
        Dim About As New About
        About.ShowDialog()
        About.Refresh()
    End Sub
    Private Sub btnStart_GettingStarted_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles btnStart_GettingStarted.Click
        Dim Help As New Help
        Help.ShowDialog()
        Help.Refresh()
    End Sub



    '(1)LOAD_SAFIR.out_FILE
    Private Sub btnLoad_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
     Handles btnLoad.Click
        Me.tabctrlMain.SelectTab(1)
    End Sub
    Private Sub btnBrowse_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles btnBrowse.Click

        With opnSAFIRout
            .Filter = "SAFIR Output Files (*.out)|*.out" '|All Files (*.*)|*.*
            .InitialDirectory = OUTPUTfolder
            .Title = "Select SAFIR.out File to Load"
            If .ShowDialog = Windows.Forms.DialogResult.OK Then
                Me.txtFilePath.Text = .FileName
            Else
                GoTo 998
            End If
        End With

        xlScratch.Cells(2, 1).Value = txtFilePath.Text
        xlScratch.Cells(6, 1).value = "FALSE"
        btnBrowse.Enabled = False

        'Progress Bar
        prgLoadOut.Value = 0
        prgLoadOut.Maximum = 100
        prgLoadOut.Minimum = 0

        For i = 1 To 10
            prgLoadOut.Value = i
```

```
        Next i

        'Import .out File into UTFire.xlsx=============================================↙
    ============
        xlREOUT.Cells.Delete()
        xlOUT.Cells.Delete()
        With xlOUT.QueryTables.Add(Connection:="TEXT;" & txtFilePath.Text, Destination:= ↙
    xlOUT.Range("$A$1"))
            .FieldNames = True
            .RowNumbers = False
            .FillAdjacentFormulas = False
            .PreserveFormatting = True
            .RefreshOnFileOpen = False
            .SavePassword = False
            .SaveData = True
            .AdjustColumnWidth = True
            .RefreshPeriod = 0
            .TextFilePromptOnRefresh = False
            .TextFilePlatform = 65001
            .TextFileStartRow = 1
            .TextFileConsecutiveDelimiter = True
            .TextFileTabDelimiter = False
            .TextFileSemicolonDelimiter = False
            .TextFileCommaDelimiter = False
            .TextFileSpaceDelimiter = True
            .TextFileTrailingMinusNumbers = True
            .Refresh(BackgroundQuery:=False)
        End With
        '=============================================================================↙
    =============

        For i = 11 To 20
            prgLoadOut.Value = i
        Next i

        'Import .fct File to UTFire.xlsx=============================================↙
    =============
        Dim ddd As Integer
        Dim eee As Integer
        Dim FireSteps As Integer
        Dim FireMax As Integer

        For i = 1 To 20000
            If xlOUT.Cells(i, 2).Value = "FRONTIER" Then
                ddd = i + 1
                GoTo 100
            Else
            End If
        Next i

100:    For j = 4 To 7
            If xlOUT.Cells(ddd, j).Value = "NO" Then
                GoTo 101
            Else
                eee = j
                GoTo 102
            End If
101:    Next j
102:    lblFire.Text = xlOUT.Cells(ddd, eee).Value
        xlScratch.Cells(4, 1).Value = lblFire.Text

        lblFirePath.Text = GetDirectoryName(FullPath:=txtFilePath.Text) & "\" & lblFire. ↙
    Text
        xlScratch.Cells(5, 1).Value = lblFirePath.Text
        xlFire.Cells.Delete()
```

```
        With xlFire.QueryTables.Add(Connection:="TEXT;" & lblFirePath.Text, Destination:=↙
    xlFire.Range("$A$1"))
            .Name = lblFire.Text
            .FieldNames = True
            .RowNumbers = False
            .FillAdjacentFormulas = False
            .PreserveFormatting = True
            .RefreshOnFileOpen = False
            .RefreshStyle = Excel.XlCellInsertionMode.xlInsertDeleteCells
            .SavePassword = False
            .SaveData = True
            .AdjustColumnWidth = True
            .RefreshPeriod = 0
            .TextFilePromptOnRefresh = False
            .TextFilePlatform = 437
            .TextFileStartRow = 1
            .TextFileParseType = Excel.XlTextParsingType.xlDelimited
            .TextFileTextQualifier = Excel.XlTextQualifier.xlTextQualifierDoubleQuote
            .TextFileConsecutiveDelimiter = False
            .TextFileTabDelimiter = True
            .TextFileSemicolonDelimiter = False
            .TextFileCommaDelimiter = False
            .TextFileSpaceDelimiter = False
            .TextFileTrailingMinusNumbers = True
            .Refresh(BackgroundQuery:=False)
        End With

        For i = 21 To 30
            prgLoadOut.Value = i
        Next i

        For i = 2 To 1000
            If xlFire.Cells(i, 1).value = 0 Then
                FireSteps = i - 1
                GoTo 104
            Else
                GoTo 103
            End If
103:    Next i

104:    FireMax = xlFire.Cells(FireSteps, 1).value
        xlScratch.Cells(9, 1).value = FireSteps

        For i = 1 To FireSteps
            xlFire.Cells(i, 3).value = xlFire.Cells(i, 1).value / 3600
        Next i

        'PLOT FIRE FILE
        If f_new = True Then GoTo 1041
        xlFire.ChartObjects(1).Delete()

1041:   Dim fireplot As Excel.Chart
        Dim xlCharts As Excel.ChartObjects
        Dim myChart As Excel.ChartObject
        Dim fire_SerCol As Excel.SeriesCollection
        Dim fire_Series As Excel.Series
        Dim fire_AxisCategory, fire_AxisValue As Excel.Axes

        xlCharts = xlFire.ChartObjects
        myChart = xlCharts.Add(120, 10, 900, 600)
        fireplot = myChart.Chart
        fireplot.ChartType = Excel.XlChartType.xlXYScatterLinesNoMarkers
        fireplot.ChartStyle = 2
        fire_SerCol = fireplot.SeriesCollection

        fire_Series = fire_SerCol.NewSeries
```

```
        fire_Series.Name = "fire"
        fire_Series.XValues = xlFire.Range("C1:C" & FireSteps)
        fire_Series.Values = xlFire.Range("B1:B" & FireSteps)
        fire_Series.Format.Line.Weight = 1.5

        With fireplot
            .Legend.Delete()
            .HasTitle = True
            .ChartTitle.Text = lblFire.Text

            fire_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
            fire_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = True
            fire_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Text↙
    () = "Time, Hours"
            fire_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Font↙
    .Bold = False
            fire_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Font↙
    .Size = 14
            fire_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines = True
            fire_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines = True
            fire_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format.   ↙
    Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            fire_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScale = 0
            fire_AxisCategory.Item(Excel.XlAxisType.xlCategory).MaximumScale = FireMax / ↙
    3600
            fire_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition = Excel↙
    .XlTickLabelPosition.xlTickLabelPositionLow
            fire_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold =   ↙
    False
            fire_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size = 14

            fire_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
            fire_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = True
            fire_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Text() =  ↙
    "Temperature, C"
            fire_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Bold ↙
    = False
            fire_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Size ↙
    = 14
            fire_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
            fire_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
            fire_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line.    ↙
    DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            fire_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScale = 0
            fire_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.     ↙
    XlTickLabelPosition.xlTickLabelPositionLow
            fire_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold = False
            fire_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
        End With

        fireplot.Refresh()

        'exporting chart as picture file
        xlFire.ChartObjects(1).chart.Export(FILEname:=OUTPUTfolder & "\temp" & "\fireplot↙
    .bmp", FilterName:="BMP")

        'load the picture into the picture box
        plotFire.BackgroundImage = New System.Drawing.Bitmap(OUTPUTfolder & "\temp" & "\ ↙
    fireplot.bmp")
        plotFire.BackgroundImageLayout = ImageLayout.Stretch
        '=============================================================================↙
    =============

        For i = 31 To 40
            prgLoadOut.Value = i
        Next i
```

```
        'RE.OUT=============================================================================↙
    =============

        'Setup Sheet re.out
        With xlREOUT
            .Cells.Delete()
            .Cells.HorizontalAlignment = Excel.XlVAlign.xlVAlignCenter

            'Scratch_data
            .Columns(1).ColumnWidth = 12
            .Columns(1).HorizontalAlignment = Excel.XlHAlign.xlHAlignRight
            .Cells(2, 1).value = "Start:"
            .Cells(3, 1).value = "Nodes:"
            .Cells(4, 1).value = "Elements:"
            .Cells(5, 1).value = "XY:"
            .Cells(7, 1).value = "TimeStep:"
            .Cells(8, 1).value = "EndTime:"
            .Cells(9, 1).value = "NumSteps:"
            .Cells(11, 1).value = "f'c:"
            .Cells(12, 1).value = "Aggs:"
            .Cells(13, 1).Value = "C_Model:"
            .Cells(15, 1).value = "fp:"
            .Cells(16, 1).value = "fy:"
            .Cells(17, 1).Value = "S_Model:"
            .Cells(20, 1).value = "Analysis:"
            .Cells(23, 1).value = "#Reinf:"
            .Columns(2).ColumnWidth = 16
            .Columns(2).HorizontalAlignment = Excel.XlHAlign.xlHAlignLeft
            .Columns(3).ColumnWidth = 3
            .Columns(3).Interior.Colorindex = 1
            .Columns(3).Interior.Pattern = 1
            .Columns(3).Interior.PatternColorIndex = -4105

            'Node_Data
            .Cells(1, 4).value = "NODE"
            .Cells(1, 5).value = "x"
            .Cells(1, 6).value = "y"
            .Cells(1, 7).value = "Temp"
            .Columns(8).ColumnWidth = 3
            .Columns(8).Interior.Colorindex = 1
            .Columns(8).Interior.Pattern = 1
            .Columns(8).Interior.PatternColorIndex = -4105

            'Element_Data
            .Cells(1, 9).value = "ELEMENT"
            .Cells(1, 10).value = "nodeA"
            .Cells(1, 11).value = "yA"
            .Cells(1, 12).value = "tempA"
            .Cells(1, 13).value = "nodeB"
            .Cells(1, 14).value = "yB"
            .Cells(1, 15).value = "tempB"
            .Cells(1, 16).value = "nodeC"
            .Cells(1, 17).value = "yC"
            .Cells(1, 18).value = "tempC"
            .Cells(1, 19).value = "nodeD"
            .Cells(1, 20).value = "yD"
            .Cells(1, 21).value = "tempD"
            .Columns(22).ColumnWidth = 3
            .Columns(22).Interior.Colorindex = 1
            .Columns(22).Interior.Pattern = 1
            .Columns(22).Interior.PatternColorIndex = -4105

            'Concrete_Data
            .Cells(1, 23).value = "ELEMENT"
            .Cells(1, 24).value = "yBar"
```

```
            .Cells(1, 25).value = "tempBar"
            .Cells(1, 26).value = "fcT"
            .Cells(1, 27).value = "eoT"
            .Cells(1, 28).value = "ecuT"
            .Cells(1, 29).value = "Ac"
            .Cells(1, 30).value = "y'"
            .Cells(1, 31).value = "ec"
            .Cells(1, 32).value = "range"
            .Cells(1, 33).value = "fc"
            .Cells(1, 34).value = "Fc"
            .Cells(1, 35).value = "sum"
            .Cells(1, 36).value = "Mc"
            .Columns(37).ColumnWidth = 3
            .Columns(37).Interior.Colorindex = 1
            .Columns(37).Interior.Pattern = 1
            .Columns(37).Interior.PatternColorIndex = -4105

            'Reinf_Data
            .Cells(1, 38).value = "REINF"
            .Cells(1, 39).value = "x_x"
            .Cells(1, 40).value = "y_y"
            .Cells(1, 41).value = "x"
            .Cells(1, 42).value = "y"
            .Cells(1, 43).value = "refNode"
            .Cells(1, 44).value = "refTemp"
            .Cells(1, 45).value = "fpT"
            .Cells(1, 46).value = "fyT"
            .Cells(1, 47).value = "EsT"
            .Cells(1, 48).value = "epT"
            .Cells(1, 49).value = "eyT"
            .Cells(1, 50).value = "etT"
            .Cells(1, 51).value = "euT"
            .Cells(1, 52).value = "As"
            .Cells(1, 53).value = "y'"
            .Cells(1, 54).value = "es"
            .Cells(1, 55).value = "range"
            .Cells(1, 56).value = "fs"
            .Cells(1, 57).value = "Fs"
            .Cells(1, 58).value = "sum"
            .Cells(1, 59).value = "Ms"
            .Columns(60).ColumnWidth = 3
            .Columns(60).Interior.Colorindex = 1
            .Columns(60).Interior.Pattern = 1
            .Columns(60).Interior.PatternColorIndex = -4105

            .Rows(1).Font.Bold = True
            .Rows(1).Borders(Excel.XlBordersIndex.xlEdgeBottom).LineStyle = Excel.
    XlLineStyle.xlDouble
            .Rows(1).Borders(Excel.XlBordersIndex.xlEdgeBottom).Weight = Excel.
    XlBorderWeight.xlThick
        End With

        'Read .out for Nodes
        With xlOUT
            For i = 1 To 10000
                If IsNumeric(.Cells(i, 1).value) = True Then GoTo 2001
                If IsNumeric(.Cells(i, 2).value) = True Then GoTo 2001
                If .Cells(i, 1).value = "File" Then
                    lblSection.Text = .Cells(i - 1, 2).value
                    xlScratch.Cells(3, 1).Value = lblSection.Text
                    For j = 46 To 50
                        prgLoadOut.Value = j
                    Next j
                End If
                If .Cells(i, 2).value = "NNODE" Then
                    Nodes = .Cells(i, 3).value
```

```
                    lblNodes.Text = Nodes
                    For j = 51 To 55
                        prgLoadOut.Value = j
                    Next j
                End If
                If .Cells(i, 2).value = "SOLID" Then
                    Elements = .Cells(i, 3).value
                    lblElements.Text = Elements
                    For j = 56 To 60
                        prgLoadOut.Value = j
                    Next j
                End If
                If .Cells(i, 2).value = "END_ELEM" Then
                    XY = i + 1
                    For j = 61 To 66
                        prgLoadOut.Value = j
                    Next j
                End If
                If .Cells(i, 1).value = "NODOFSOLID" Then
                    Start = i
                    For j = 67 To 73
                        prgLoadOut.Value = j
                    Next j
                End If
                If .Cells(i, 2).value = "TIMEPRINT" Then
                    TimeStep = .Cells(i + 1, 2).value
                    EndTime = .Cells(i + 1, 3).value
                    NumSteps = EndTime / TimeStep
                    lblTimeStep.Text = TimeStep
                    lblEndTime.Text = EndTime
                    lblNumSteps.Text = NumSteps
                    For j = 74 To 80
                        prgLoadOut.Value = j
                    Next j
                    GoTo 200
                End If
2001:       Next i
        End With

200:    With xlREOUT
            .Cells(2, 2).value = Start
            .Cells(3, 2).value = Nodes
            .Cells(4, 2).value = Elements
            .Cells(5, 2).value = XY
            .Cells(7, 2).value = TimeStep
            .Cells(8, 2).value = EndTime
            .Cells(9, 2).value = NumSteps
        End With

        'Print Nodes
        With xlOUT
            For i = 1 To Nodes
                nodeY(i) = .Cells(XY + i, 4).value * 1000
                nodeX(i) = .Cells(XY + i, 5).value * 1000
            Next i
        End With
        With xlREOUT
            For i = 1 To Nodes
                .Cells(i + 1, 4) = i
                .Cells(i + 1, 5) = nodeX(i)
                .Cells(i + 1, 6) = nodeY(i)
            Next i
        End With

        'Print Elements
        With xlOUT
```

```
            For i = 1 To Elements
                nodeA(i) = .Cells(Start + i, 4).value
                nodeB(i) = .Cells(Start + i, 5).value
                nodeC(i) = .Cells(Start + i, 6).value
                nodeD(i) = .Cells(Start + i, 7).value
            Next i
        End With

        For i = 1 To Elements
            elementAREA(i) = (Max(nodeX(nodeA(i)), Max(nodeX(nodeB(i)), Max(nodeX(nodeC
    (i)), nodeX(nodeD(i))))) - Min(nodeX(nodeA(i)), Min(nodeX(nodeB(i)), Min(nodeX(nodeC
    (i)), nodeX(nodeD(i)))))) * (Max(nodeY(nodeA(i)), Max(nodeY(nodeB(i)), Max(nodeY
    (nodeC(i)), nodeY(nodeD(i))))) - Min(nodeY(nodeA(i)), Min(nodeY(nodeB(i)), Min(nodeY
    (nodeC(i)), nodeY(nodeD(i))))))
        Next

        With xlREOUT
            For i = 1 To Elements
                .Cells(i + 1, 9).value = i
                .Cells(i + 1, 10).value = nodeA(i)
                .Cells(i + 1, 11).value = nodeY(nodeA(i))
                .Cells(i + 1, 13).value = nodeB(i)
                .Cells(i + 1, 14).value = nodeY(nodeB(i))
                .Cells(i + 1, 16).value = nodeC(i)
                .Cells(i + 1, 17).value = nodeY(nodeC(i))
                .Cells(i + 1, 19).value = nodeD(i)
                .Cells(i + 1, 20).value = nodeY(nodeD(i))
                .Cells(i + 1, 23).value = i
                element_Y(i) = (nodeY(nodeA(i)) + nodeY(nodeB(i)) + nodeY(nodeC(i)) +
    nodeY(nodeD(i))) / 4
                .Cells(i + 1, 24).value = element_Y(i)
                .Cells(i + 1, 29).value = elementAREA(i)
            Next i
        End With

        '=============================================================================
    ==============

        For i = 81 To 90
            prgLoadOut.Value = i
        Next i

        'Plot Section=================================================================
    ==============
        If f_new = True Then GoTo 201
        xlSection.ChartObjects(1).Delete()

201:    With xlSection
            For i = 1 To Nodes
                .Cells(i, 1).value = nodeX(i)
                .Cells(i, 2).value = nodeY(i)
            Next i
        End With

        Dim sectionplot As Excel.Chart
        Dim section_SerCol As Excel.SeriesCollection
        Dim section_Series As Excel.Series
        Dim section_AxisCategory, section_AxisValue As Excel.Axes

        xlCharts = xlSection.ChartObjects
        myChart = xlCharts.Add(260, 10, 900, 600)
        sectionplot = myChart.Chart
        sectionplot.ChartType = Excel.XlChartType.xlXYScatter
        section_SerCol = sectionplot.SeriesCollection

        section_Series = section_SerCol.NewSeries
```

```
        section_Series.Name = "section"
        section_Series.XValues = xlSection.Range("A1:A" & Nodes)
        section_Series.Values = xlSection.Range("B1:B" & Nodes)

        'max,min X & Y values
        Xaxis_max = 0
        Yaxis_max = 0
        Xaxis_min = 0
        Yaxis_min = 0

        For i = 1 To Nodes
            If nodeX(i) > Xaxis_max Then Xaxis_max = nodeX(i)
            If nodeY(i) > Yaxis_max Then Yaxis_max = nodeY(i)
        Next i
        If Xaxis_max > Yaxis_max Then
            axis_max = Xaxis_max
        Else
            axis_max = Yaxis_max
        End If

        For i = 1 To Nodes
            If nodeX(i) < Xaxis_min Then Xaxis_min = nodeX(i)
            If nodeY(i) < Yaxis_min Then Yaxis_min = nodeY(i)
        Next i
        If Xaxis_min < Yaxis_min Then
            axis_min = Xaxis_min
        Else
            axis_min = Yaxis_min
        End If

        If axis_max > Abs(axis_min) Then
            square_scale = axis_max
        Else
            square_scale = Abs(axis_min)
        End If

        s_width = Abs(Xaxis_min) + Abs(Xaxis_max)
        s_height = Abs(Yaxis_min) + Abs(Yaxis_max)
        xlSection.Cells(1, 3).value = s_width
        xlSection.Cells(2, 3).value = s_height
        xlSection.Cells(4, 3).value = Yaxis_min
        xlSection.Cells(5, 3).value = Yaxis_max



        With sectionplot
            .Legend.Delete()
            .HasTitle = True
            .ChartTitle.Text = lblSection.Text

            section_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = False
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines =
    True
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format.
    Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines =
    True
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScale = -
    (square_scale + square_scale / 10)
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).MaximumScale =
    (square_scale + square_scale / 10)
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).MajorUnit = Round(
    (square_scale + square_scale / 10) * 2 / 5, 1)
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition =
    Excel.XlTickLabelPosition.xlTickLabelPositionLow
```

```
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).CrossesAt = 0
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold =↙
     False
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size =↙
     14

            section_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
            section_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = False
            section_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
            section_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
            section_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line. ↙
    DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            section_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScale = -            ↙
    (square_scale + square_scale / 10)
            section_AxisValue.Item(Excel.XlAxisType.xlValue).MaximumScale = (square_scale↙
     + square_scale / 10)
            section_AxisValue.Item(Excel.XlAxisType.xlValue).MajorUnit = Round(          ↙
    (square_scale + square_scale / 10) * 2 / 5, 1)
            section_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.  ↙
    XlTickLabelPosition.xlTickLabelPositionLow
            section_AxisValue.Item(Excel.XlAxisType.xlValue).CrossesAt = 0
            section_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold = False
            section_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
        End With

        sectionplot.Refresh()

        'exporting chart as picture file
        xlSection.ChartObjects(1).chart.Export(FILEname:=OUTPUTfolder & "\temp" & "\     ↙
    sectionplot.bmp", FilterName:="BMP")

        'load the picture into the picture box
        plotSection.BackgroundImage = New System.Drawing.Bitmap(OUTPUTfolder & "\temp" & ↙
    "\sectionplot.bmp")
        plotSection.BackgroundImageLayout = ImageLayout.Stretch
        '================================================================================↙
    ==============

        For i = 91 To 100
            prgLoadOut.Value = i
        Next i
        GoTo 999

998:    MessageBox.Show("Nothing selected")
999:    prgLoadOut.Enabled = True
    End Sub
    Private Sub prgLoadOut_EnabledChanged(ByVal sender As Object, ByVal e As System.     ↙
    EventArgs) Handles prgLoadOut.EnabledChanged
        If prgLoadOut.Value < prgLoadOut.Maximum Then
            txtProgress1.BackColor = Color.AliceBlue
            txtProgress1.ForeColor = Color.DimGray
        Else
            txtProgress1.BackColor = Color.DodgerBlue
            txtProgress1.ForeColor = Color.White
        End If
    End Sub



    '(2)VIEW_SAIFR.out_FILE
    Private Sub btnView_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)↙
     Handles btnView.Click

        If txtFilePath.Text = "" Then GoTo 100
```

```
        Dim SAFIRoutText As StreamReader
        SAFIRoutText = File.OpenText(txtFilePath.Text)
        txtOutFile.Text = SAFIRoutText.ReadToEnd()
        Me.tabctrlMain.SelectTab(7)
        GoTo 101

100:    MessageBox.Show("SAFIR.out file not loaded")
101:

    End Sub



    '(3)DEFINE_MATERIAL_PROPERTIES
    Private Sub btnMaterials_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles btnMaterials.Click
        Me.tabctrlMain.SelectTab(2)

    End Sub
    Private Sub txtConcreteStrength_TextChanged(ByVal sender As System.Object, ByVal e As
     System.EventArgs) Handles txtConcreteStrength.TextChanged

        'Write to excel
        xlREOUT.Cells(11, 2).value = txtConcreteStrength.Text

        'Completed_progress2
        If txtConcreteStrength.Text = "" Or cmbAggregate.Text = "" Or cmbConcreteModel.
    Text = "" Then
            txtProgress2.BackColor = Color.AliceBlue
            txtProgress2.ForeColor = Color.DimGray
        Else
            txtProgress2.BackColor = Color.DodgerBlue
            txtProgress2.ForeColor = Color.White
        End If

        'Enable Plot
        If txtConcreteStrength.Text = "" Or cmbAggregate.Text = "" Or cmbConcreteModel.
    Text = "" Or txtSteelProportional.Text = "" Or txtSteelYield.Text = "" Or
    cmbSteelModel.Text = "" Then
            btnPLOT.Enabled = False
        Else
            btnPLOT.Enabled = True
        End If

    End Sub
    Private Sub cmbAggregate_SelectedIndexChanged(ByVal sender As System.Object, ByVal e
    As System.EventArgs) Handles cmbAggregate.SelectedIndexChanged

        'Write to excel
        xlREOUT.Cells(12, 2).value = cmbAggregate.Text

        'Completed_progress2
        If txtConcreteStrength.Text = "" Or cmbAggregate.Text = "" Or cmbConcreteModel.
    Text = "" Then
            txtProgress2.BackColor = Color.AliceBlue
            txtProgress2.ForeColor = Color.DimGray
        Else
            txtProgress2.BackColor = Color.DodgerBlue
            txtProgress2.ForeColor = Color.White
        End If

        'Enable Plot
        If txtConcreteStrength.Text = "" Or cmbAggregate.Text = "" Or cmbConcreteModel.
    Text = "" Or txtSteelProportional.Text = "" Or txtSteelYield.Text = "" Or
    cmbSteelModel.Text = "" Then
```

```
            btnPLOT.Enabled = False
        Else
            btnPLOT.Enabled = True
        End If

    End Sub
    Private Sub cmbConcreteModel_SelectedIndexChanged(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles cmbConcreteModel.SelectedIndexChanged

        'Write to excel
        xlREOUT.Cells(13, 2).value = cmbConcreteModel.Text

        'Completed_progress2
        If txtConcreteStrength.Text = "" Or cmbAggregate.Text = "" Or cmbConcreteModel.
    Text = "" Then
            txtProgress2.BackColor = Color.AliceBlue
            txtProgress2.ForeColor = Color.DimGray
        Else
            txtProgress2.BackColor = Color.DodgerBlue
            txtProgress2.ForeColor = Color.White
        End If

        'Enable Plot
        If txtConcreteStrength.Text = "" Or cmbAggregate.Text = "" Or cmbConcreteModel.
    Text = "" Or txtSteelProportional.Text = "" Or txtSteelYield.Text = "" Or
    cmbSteelModel.Text = "" Then
            btnPLOT.Enabled = False
        Else
            btnPLOT.Enabled = True
        End If

    End Sub
    Private Sub txtSteelProportional_TextChanged(ByVal sender As System.Object, ByVal e
    As System.EventArgs) Handles txtSteelProportional.TextChanged

        'Write to excel
        xlREOUT.Cells(15, 2).value = txtSteelProportional.Text

        'Completed_progress3
        If txtSteelProportional.Text = "" Or txtSteelYield.Text = "" Or cmbSteelModel.
    Text = "" Then
            txtProgress3.BackColor = Color.AliceBlue
            txtProgress3.ForeColor = Color.DimGray
        Else
            txtProgress3.BackColor = Color.DodgerBlue
            txtProgress3.ForeColor = Color.White
        End If

        'Enable Plot
        If txtConcreteStrength.Text = "" Or cmbAggregate.Text = "" Or cmbConcreteModel.
    Text = "" Or txtSteelProportional.Text = "" Or txtSteelYield.Text = "" Or
    cmbSteelModel.Text = "" Then
            btnPLOT.Enabled = False
        Else
            btnPLOT.Enabled = True
        End If

    End Sub
    Private Sub txtSteelYield_TextChanged(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles txtSteelYield.TextChanged

        'Write to excel
        xlREOUT.Cells(16, 2).value = txtSteelYield.Text

        'Completed_progress3
        If txtSteelProportional.Text = "" Or txtSteelYield.Text = "" Or cmbSteelModel.
```

```
    Text = "" Then
            txtProgress3.BackColor = Color.AliceBlue
            txtProgress3.ForeColor = Color.DimGray
        Else
            txtProgress3.BackColor = Color.DodgerBlue
            txtProgress3.ForeColor = Color.White
        End If

        'Enable Plot
        If txtConcreteStrength.Text = "" Or cmbAggregate.Text = "" Or cmbConcreteModel.
    Text = "" Or txtSteelProportional.Text = "" Or txtSteelYield.Text = "" Or
    cmbSteelModel.Text = "" Then
            btnPLOT.Enabled = False
        Else
            btnPLOT.Enabled = True
        End If

    End Sub
    Private Sub cmbSteelModel_SelectedIndexChanged(ByVal sender As System.Object, ByVal e
     As System.EventArgs) Handles cmbSteelModel.SelectedIndexChanged

        'Write to excel
        xlREOUT.Cells(17, 2).value = cmbSteelModel.Text

        'Completed_progress3
        If txtSteelProportional.Text = "" Or txtSteelYield.Text = "" Or cmbSteelModel.
    Text = "" Then
            txtProgress3.BackColor = Color.AliceBlue
            txtProgress3.ForeColor = Color.DimGray
        Else
            txtProgress3.BackColor = Color.DodgerBlue
            txtProgress3.ForeColor = Color.White
        End If

        'Enable Plot
        If txtConcreteStrength.Text = "" Or cmbAggregate.Text = "" Or cmbConcreteModel.
    Text = "" Or txtSteelProportional.Text = "" Or txtSteelYield.Text = "" Or
    cmbSteelModel.Text = "" Then
            btnPLOT.Enabled = False
        Else
            btnPLOT.Enabled = True
        End If

    End Sub
    Private Sub btnPLOT_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
     Handles btnPLOT.Click

        INITIAL = 0
        btnPLOT.Enabled = False

        '**CONCRETE*****************************************************************
    ***************
        With xlConcrete
            .Cells.Clear()
            .Cells.HorizontalAlignment = Excel.XlVAlign.xlVAlignCenter
            .Columns(1).ColumnWidth = 12
            .Columns(1).HorizontalAlignment = Excel.XlHAlign.xlHAlignRight
            .Rows(1).Font.Bold = True
        End With

        prgSS.Value = 0

        'Initial
        fprimec = txtConcreteStrength.Text
        aggregates = cmbAggregate.Text
        eo = 0.0025
```

```
        ecu = 0.02
        fct = -0.6228 * (fprimec) ^ 0.5 '= 7.5*(f'c)^0.5 in psi
        Ecc = 4732.98 * (fprimec) ^ 0.5 '= 57000*(f'c)^0.5 in psi
        ect = fct / Ecc
        ectu = 11 * fct / Ecc

        xlConcrete.Cells(1, 1).value = "CONCRETE"
        xlConcrete.Cells(2, 1).value = "f'c:"
        xlConcrete.Cells(3, 1).value = "Aggs:"
        xlConcrete.Cells(4, 1).value = "eo:"
        xlConcrete.Cells(5, 1).value = "ecu:"
        xlConcrete.Cells(6, 1).value = "f'ct:"
        xlConcrete.Cells(7, 1).value = "Ec:"
        xlConcrete.Cells(8, 1).value = "e'ct:"
        xlConcrete.Cells(9, 1).value = "ectu:"

        xlConcrete.Cells(2, 2).value = fprimec
        xlConcrete.Cells(3, 2).value = aggregates
        xlConcrete.Cells(4, 2).value = eo
        xlConcrete.Cells(5, 2).value = ecu
        xlConcrete.Cells(6, 2).value = fct
        xlConcrete.Cells(7, 2).value = Ecc
        xlConcrete.Cells(8, 2).value = ect
        xlConcrete.Cells(9, 2).value = ectu



        'Heated
        temp_TEMP(0) = 20
        temp_TEMP(1) = 100
        temp_TEMP(2) = 200
        temp_TEMP(3) = 300
        temp_TEMP(4) = 400
        temp_TEMP(5) = 500
        temp_TEMP(6) = 600
        temp_TEMP(7) = 700
        temp_TEMP(8) = 800
        temp_TEMP(9) = 900
        temp_TEMP(10) = 1000
        temp_TEMP(11) = 1100
        temp_TEMP(12) = 1200

        x_col(0) = "E"
        y_col(0) = "F"
        x_col(1) = "I"
        y_col(1) = "J"
        x_col(2) = "M"
        y_col(2) = "N"
        x_col(3) = "Q"
        y_col(3) = "R"
        x_col(4) = "U"
        y_col(4) = "V"
        x_col(5) = "Y"
        y_col(5) = "Z"
        x_col(6) = "AC"
        y_col(6) = "AD"
        x_col(7) = "AG"
        y_col(7) = "AH"
        x_col(8) = "AK"
        y_col(8) = "AL"
        x_col(9) = "AO"
        y_col(9) = "AP"
        x_col(10) = "AS"
        y_col(10) = "AT"
        x_col(11) = "AW"
        y_col(11) = "AX"
```

```
        x_col(12) = "BA"
        y_col(12) = "BB"

        If aggregates = "Siliceous" Then GoTo 100
        If aggregates = "Calcareous" Then GoTo 101

100:    For i = 0 To 12

            If temp_TEMP(i) = 20 Then fprimec_TEMP(i) = 1.0 * fprimec
            If temp_TEMP(i) = 20 Then eo_TEMP(i) = 0.0025
            If temp_TEMP(i) = 20 Then ecu_TEMP(i) = 0.02
            If temp_TEMP(i) = 20 Then fct_TEMP(i) = 1.0 * fct

            If temp_TEMP(i) = 100 Then fprimec_TEMP(i) = 1.0 * fprimec
            If temp_TEMP(i) = 100 Then eo_TEMP(i) = 0.004
            If temp_TEMP(i) = 100 Then ecu_TEMP(i) = 0.0225
            If temp_TEMP(i) = 100 Then fct_TEMP(i) = 1.0 * fct

            If temp_TEMP(i) = 200 Then fprimec_TEMP(i) = 0.95 * fprimec
            If temp_TEMP(i) = 200 Then eo_TEMP(i) = 0.0055
            If temp_TEMP(i) = 200 Then ecu_TEMP(i) = 0.025
            If temp_TEMP(i) = 200 Then fct_TEMP(i) = 0.8 * fct

            If temp_TEMP(i) = 300 Then fprimec_TEMP(i) = 0.85 * fprimec
            If temp_TEMP(i) = 300 Then eo_TEMP(i) = 0.007
            If temp_TEMP(i) = 300 Then ecu_TEMP(i) = 0.0275
            If temp_TEMP(i) = 300 Then fct_TEMP(i) = 0.6 * fct

            If temp_TEMP(i) = 400 Then fprimec_TEMP(i) = 0.75 * fprimec
            If temp_TEMP(i) = 400 Then eo_TEMP(i) = 0.01
            If temp_TEMP(i) = 400 Then ecu_TEMP(i) = 0.03
            If temp_TEMP(i) = 400 Then fct_TEMP(i) = 0.4 * fct

            If temp_TEMP(i) = 500 Then fprimec_TEMP(i) = 0.6 * fprimec
            If temp_TEMP(i) = 500 Then eo_TEMP(i) = 0.015
            If temp_TEMP(i) = 500 Then ecu_TEMP(i) = 0.0325
            If temp_TEMP(i) = 500 Then fct_TEMP(i) = 0.2 * fct

            If temp_TEMP(i) = 600 Then fprimec_TEMP(i) = 0.45 * fprimec
            If temp_TEMP(i) = 600 Then eo_TEMP(i) = 0.025
            If temp_TEMP(i) = 600 Then ecu_TEMP(i) = 0.035
            If temp_TEMP(i) = 600 Then fct_TEMP(i) = 0 * fct

            If temp_TEMP(i) = 700 Then fprimec_TEMP(i) = 0.3 * fprimec
            If temp_TEMP(i) = 700 Then eo_TEMP(i) = 0.025
            If temp_TEMP(i) = 700 Then ecu_TEMP(i) = 0.0375
            If temp_TEMP(i) = 700 Then fct_TEMP(i) = 0 * fct

            If temp_TEMP(i) = 800 Then fprimec_TEMP(i) = 0.15 * fprimec
            If temp_TEMP(i) = 800 Then eo_TEMP(i) = 0.025
            If temp_TEMP(i) = 800 Then ecu_TEMP(i) = 0.04
            If temp_TEMP(i) = 800 Then fct_TEMP(i) = 0 * fct

            If temp_TEMP(i) = 900 Then fprimec_TEMP(i) = 0.08 * fprimec
            If temp_TEMP(i) = 900 Then eo_TEMP(i) = 0.025
            If temp_TEMP(i) = 900 Then ecu_TEMP(i) = 0.0425
            If temp_TEMP(i) = 900 Then fct_TEMP(i) = 0 * fct

            If temp_TEMP(i) = 1000 Then fprimec_TEMP(i) = 0.04 * fprimec
            If temp_TEMP(i) = 1000 Then eo_TEMP(i) = 0.025
            If temp_TEMP(i) = 1000 Then ecu_TEMP(i) = 0.045
            If temp_TEMP(i) = 1000 Then fct_TEMP(i) = 0 * fct

            If temp_TEMP(i) = 1100 Then fprimec_TEMP(i) = 0.01 * fprimec
            If temp_TEMP(i) = 1100 Then eo_TEMP(i) = 0.025
            If temp_TEMP(i) = 1100 Then ecu_TEMP(i) = 0.0475
```

```
            If temp_TEMP(i) = 1100 Then fct_TEMP(i) = 0 * fct

            If temp_TEMP(i) = 1200 Then fprimec_TEMP(i) = 0.001 * fprimec
            If temp_TEMP(i) = 1200 Then eo_TEMP(i) = 0.025
            If temp_TEMP(i) = 1200 Then ecu_TEMP(i) = 0.05
            If temp_TEMP(i) = 1200 Then fct_TEMP(i) = 0 * fct

            If fct_Temp(i) = 0 Then
                Ecc_Temp(i) = 0
                ect_Temp(i) = 0
                ectu_Temp(i) = 0
                GoTo 1001
            End If
            Ecc_Temp(i) = 4732.98 * (fprimec_TEMP(i)) ^ 0.5
            ect_Temp(i) = fct_Temp(i) / Ecc_Temp(i)
            ectu_TEMP(i) = 11 * fct_TEMP(i) / Ecc_TEMP(i)

1001:       prgSS.Value = (i + 1) * 7
        Next i
        GoTo 102

101:    For i = 0 To 12

            If temp_TEMP(i) = 20 Then fprimec_TEMP(i) = 1.0 * fprimec
            If temp_TEMP(i) = 20 Then eo_TEMP(i) = 0.0025
            If temp_TEMP(i) = 20 Then ecu_TEMP(i) = 0.02
            If temp_TEMP(i) = 20 Then fct_TEMP(i) = 1.0 * fct

            If temp_TEMP(i) = 100 Then fprimec_TEMP(i) = 1.0 * fprimec
            If temp_TEMP(i) = 100 Then eo_TEMP(i) = 0.004
            If temp_TEMP(i) = 100 Then ecu_TEMP(i) = 0.0225
            If temp_TEMP(i) = 100 Then fct_TEMP(i) = 1.0 * fct

            If temp_TEMP(i) = 200 Then fprimec_TEMP(i) = 0.97 * fprimec
            If temp_TEMP(i) = 200 Then eo_TEMP(i) = 0.0055
            If temp_TEMP(i) = 200 Then ecu_TEMP(i) = 0.025
            If temp_TEMP(i) = 200 Then fct_TEMP(i) = 0.8 * fct

            If temp_TEMP(i) = 300 Then fprimec_TEMP(i) = 0.91 * fprimec
            If temp_TEMP(i) = 300 Then eo_TEMP(i) = 0.007
            If temp_TEMP(i) = 300 Then ecu_TEMP(i) = 0.0275
            If temp_TEMP(i) = 300 Then fct_TEMP(i) = 0.6 * fct

            If temp_TEMP(i) = 400 Then fprimec_TEMP(i) = 0.85 * fprimec
            If temp_TEMP(i) = 400 Then eo_TEMP(i) = 0.01
            If temp_TEMP(i) = 400 Then ecu_TEMP(i) = 0.03
            If temp_TEMP(i) = 400 Then fct_TEMP(i) = 0.4 * fct

            If temp_TEMP(i) = 500 Then fprimec_TEMP(i) = 0.74 * fprimec
            If temp_TEMP(i) = 500 Then eo_TEMP(i) = 0.015
            If temp_TEMP(i) = 500 Then ecu_TEMP(i) = 0.0325
            If temp_TEMP(i) = 500 Then fct_TEMP(i) = 0.2 * fct

            If temp_TEMP(i) = 600 Then fprimec_TEMP(i) = 0.6 * fprimec
            If temp_TEMP(i) = 600 Then eo_TEMP(i) = 0.025
            If temp_TEMP(i) = 600 Then ecu_TEMP(i) = 0.035
            If temp_TEMP(i) = 600 Then fct_TEMP(i) = 0 * fct

            If temp_TEMP(i) = 700 Then fprimec_TEMP(i) = 0.43 * fprimec
            If temp_TEMP(i) = 700 Then eo_TEMP(i) = 0.025
            If temp_TEMP(i) = 700 Then ecu_TEMP(i) = 0.0375
            If temp_TEMP(i) = 700 Then fct_TEMP(i) = 0 * fct

            If temp_TEMP(i) = 800 Then fprimec_TEMP(i) = 0.27 * fprimec
            If temp_TEMP(i) = 800 Then eo_TEMP(i) = 0.025
            If temp_TEMP(i) = 800 Then ecu_TEMP(i) = 0.04
```

```
            If temp_TEMP(i) = 800 Then fct_TEMP(i) = 0 * fct

            If temp_TEMP(i) = 900 Then fprimec_TEMP(i) = 0.15 * fprimec
            If temp_TEMP(i) = 900 Then eo_TEMP(i) = 0.025
            If temp_TEMP(i) = 900 Then ecu_TEMP(i) = 0.0425
            If temp_TEMP(i) = 900 Then fct_TEMP(i) = 0 * fct

            If temp_TEMP(i) = 1000 Then fprimec_TEMP(i) = 0.06 * fprimec
            If temp_TEMP(i) = 1000 Then eo_TEMP(i) = 0.025
            If temp_TEMP(i) = 1000 Then ecu_TEMP(i) = 0.045
            If temp_TEMP(i) = 1000 Then fct_TEMP(i) = 0 * fct

            If temp_TEMP(i) = 1100 Then fprimec_TEMP(i) = 0.02 * fprimec
            If temp_TEMP(i) = 1100 Then eo_TEMP(i) = 0.025
            If temp_TEMP(i) = 1100 Then ecu_TEMP(i) = 0.0475
            If temp_TEMP(i) = 1100 Then fct_TEMP(i) = 0 * fct

            If temp_TEMP(i) = 1200 Then fprimec_TEMP(i) = 0.001 * fprimec
            If temp_TEMP(i) = 1200 Then eo_TEMP(i) = 0.025
            If temp_TEMP(i) = 1200 Then ecu_TEMP(i) = 0.05
            If temp_TEMP(i) = 1200 Then fct_TEMP(i) = 0 * fct

            If fct_Temp(i) = 0 Then
                Ecc_Temp(i) = 0
                ect_Temp(i) = 0
                ectu_Temp(i) = 0
                GoTo 1011
            End If
            ecc_Temp(i) = 4732.98 * (fprimec_TEMP(i)) ^ 0.5
            ect_Temp(i) = fct_Temp(i) / Ecc_Temp(i)
            ectu_TEMP(i) = 11 * fct_TEMP(i) / Ecc_TEMP(i)

1011:       prgSS.Value = (i + 1) * 7
        Next i

102:    With xlConcrete
            For i = 0 To 12

                .Cells(1, 4 * (i + 1)).value = temp_TEMP(i)
                .Cells(1, 4 * (i + 1) + 1).value = "ecT"
                ec_TEMP(i, 1) = ectu_TEMP(i)
                .Cells(2, 4 * (i + 1) + 1).value = ec_TEMP(i, 1)
                .Cells(1, 4 * (i + 1) + 2).value = "fcT"
                fc_TEMP(i, 1) = 0
                .Cells(2, 4 * (i + 1) + 2).value = fc_TEMP(i, 1)

                For j = 2 To 100
                    ec_TEMP(i, j) = ec_TEMP(i, j - 1) + (ecu_TEMP(i) - ectu_TEMP(i)) / 99
                    .Cells(j + 1, 4 * (i + 1) + 1).value = ec_TEMP(i, j)
                    If ec_TEMP(i, j) > 0 Then GoTo 1021
                    If ec_TEMP(i, j) < ectu_TEMP(i) Then fc_TEMP(i, j) = 0
                    If ectu_TEMP(i) < ec_TEMP(i, j) And ec_TEMP(i, j) < ect_TEMP(i) Then ↙
    fc_TEMP(i, j) = Ecc_TEMP(i) * ectu_TEMP(i) / 10 - Ecc_TEMP(i) * ec_TEMP(i, j) / 10
                    If ect_TEMP(i) < ec_TEMP(i, j) And ec_TEMP(i, j) < 0 Then fc_TEMP(i, ↙
    j) = Ecc_TEMP(i) * ec_TEMP(i, j)
                    GoTo 1022
1021:               If ec_TEMP(i, j) < eo_TEMP(i) Then fc_TEMP(i, j) = (3 * ec_TEMP(i, j)↙
     * fprimec_TEMP(i)) / (eo_TEMP(i) * (2 + (ec_TEMP(i, j) / eo_TEMP(i)) ^ 3))
                    If ec_TEMP(i, j) = eo_TEMP(i) Then fc_TEMP(i, j) = fprimec_TEMP(i)
                    If ec_TEMP(i, j) > eo_TEMP(i) Then fc_TEMP(i, j) = fprimec_TEMP(i) - ↙
    ((-fprimec_TEMP(i)) / (ecu_TEMP(i) - eo_TEMP(i))) * eo_TEMP(i) + ((-fprimec_TEMP(i)) ↙
    / (ecu_TEMP(i) - eo_TEMP(i))) * ec_TEMP(i, j)
1022:               .Cells(j + 1, 4 * (i + 1) + 2).value = fc_TEMP(i, j)
                    prgSS.Value = 91 + 32 * (i) + Round(j / 5.27)
                Next j
```

```
                .Columns(4 * (i + 1) + 1).NumberFormat = "0.00000"
                .Columns(4 * (i + 1) + 2).NumberFormat = "0.00"

                .Cells(2, 4 * (i + 1)).value = fprimec_TEMP(i)
                .Cells(3, 4 * (i + 1)).value = eo_TEMP(i)
                .Cells(4, 4 * (i + 1)).value = ecu_TEMP(i)
                .Cells(5, 4 * (i + 1)).value = fct_TEMP(i)
                .Cells(6, 4 * (i + 1)).value = Ecc_TEMP(i)
                .Cells(7, 4 * (i + 1)).value = ect_TEMP(i)
                .Cells(8, 4 * (i + 1)).value = ectu_TEMP(i)

            Next i
        End With

        'Plot concrete stress-strain_____________________________________________
        If cSS_new = False Then GoTo 201
        GoTo 202
201:    xlConcrete.ChartObjects.Delete()

202:    cSS_new = False
        xlScratch.Cells(7, 1).value = "FALSE"
        Dim concrete_plot As Excel.Chart
        Dim xlCharts As Excel.ChartObjects
        Dim myChart As Excel.ChartObject
        Dim concrete_SerCol As Excel.SeriesCollection
        Dim concrete_Series As Excel.Series
        Dim concrete_AxisCategory, concrete_AxisValue As Excel.Axes

        xlCharts = xlConcrete.ChartObjects
        myChart = xlCharts.Add(180, 260, 1000, 600)
        concrete_plot = myChart.Chart
        concrete_plot.ChartType = Excel.XlChartType.xlXYScatterLinesNoMarkers
        concrete_plot.ChartStyle = 3
        concrete_SerCol = concrete_plot.SeriesCollection

        concrete_Series = concrete_SerCol.NewSeries
        concrete_Series.Name = temp_TEMP(0) & "C"
        concrete_Series.XValues = xlConcrete.Range("E2:E101")
        concrete_Series.Values = xlConcrete.Range("F2:F101")
        concrete_Series.Format.Line.Weight = 1.5

        With concrete_plot
            .Legend.Position = Excel.XlLegendPosition.xlLegendPositionTop
            .HasTitle = True
            .ChartTitle.Text = "Concrete Stress-Strain Relationship"

            concrete_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.↙
    Text() = "Strain, mm/mm"
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.↙
    Font.Bold = False
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.↙
    Font.Size = 14
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines =  ↙
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines =  ↙
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format↙
    .Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScaleIsAuto = ↙
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MaximumScaleIsAuto = ↙
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MajorUnitIsAuto =    ↙
    True
```

```
        concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition = Excel.XlTickLabelPosition.xlTickLabelPositionLow
        concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.NumberFormat = "0.000"
        concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold = False
        concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size = 14

        concrete_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
        concrete_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = True
        concrete_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Text() = "Stress, MPa"
        concrete_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Bold = False
        concrete_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Size = 14
        concrete_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
        concrete_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
        concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
        concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScaleIsAuto = True
        concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MaximumScaleIsAuto = True
        concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MajorUnitIsAuto = True
        concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.XlTickLabelPosition.xlTickLabelPositionLow
        concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.NumberFormat = "0"
        concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold = False
        concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
    End With
    concrete_plot.Refresh()



    prgSS.Value = 500
    DUM = DUM + 1

    '**STEEL*****************************************************************************
    With xlSteel
        .Cells.Clear()
        .Cells.HorizontalAlignment = Excel.XlVAlign.xlVAlignCenter
        .Columns(1).ColumnWidth = 12
        .Columns(1).HorizontalAlignment = Excel.XlHAlign.xlHAlignRight
        .Rows(1).Font.Bold = True
    End With

    'Initial
    fp = txtSteelProportional.Text
    fy = txtSteelYield.Text
    Est = 199948
    ep = fp / Est
    ey = 0.02
    et = 0.15
    eu = 0.2
    c = ((fy - fp) ^ 2) / ((ey - ep) * Est - 2 * (fy - fp))
    a = ((ey - ep) * (ey - ep + c / Est)) ^ 0.5
    b = (c * (ey - ep) * Est + c ^ 2) ^ 0.5

    xlSteel.Cells(1, 1).value = "STEEL"
    xlSteel.Cells(2, 1).value = "fp:"
    xlSteel.Cells(3, 1).value = "fy:"
    xlSteel.Cells(4, 1).value = "Est:"
    xlSteel.Cells(5, 1).value = "ep:"
```

```
        xlSteel.Cells(6, 1).value = "ey:"
        xlSteel.Cells(7, 1).value = "et:"
        xlSteel.Cells(8, 1).value = "eu:"
        xlSteel.Cells(9, 1).value = "a:"
        xlSteel.Cells(10, 1).value = "b:"
        xlSteel.Cells(11, 1).value = "c:"

        xlSteel.Cells(2, 2).value = fp
        xlSteel.Cells(3, 2).value = fy
        xlSteel.Cells(4, 2).value = Est
        xlSteel.Cells(5, 2).value = ep
        xlSteel.Cells(6, 2).value = ey
        xlSteel.Cells(7, 2).value = et
        xlSteel.Cells(8, 2).value = eu
        xlSteel.Cells(9, 2).value = a
        xlSteel.Cells(10, 2).value = b
        xlSteel.Cells(11, 2).value = c

        'Heated
        temp_TEMP(0) = 20
        temp_TEMP(1) = 100
        temp_TEMP(2) = 200
        temp_TEMP(3) = 300
        temp_TEMP(4) = 400
        temp_TEMP(5) = 500
        temp_TEMP(6) = 600
        temp_TEMP(7) = 700
        temp_TEMP(8) = 800
        temp_TEMP(9) = 900
        temp_TEMP(10) = 1000
        temp_TEMP(11) = 1100
        temp_TEMP(12) = 1200

        x_col(0) = "E"
        y_col(0) = "F"
        x_col(1) = "I"
        y_col(1) = "J"
        x_col(2) = "M"
        y_col(2) = "N"
        x_col(3) = "Q"
        y_col(3) = "R"
        x_col(4) = "U"
        y_col(4) = "V"
        x_col(5) = "Y"
        y_col(5) = "Z"
        x_col(6) = "AC"
        y_col(6) = "AD"
        x_col(7) = "AG"
        y_col(7) = "AH"
        x_col(8) = "AK"
        y_col(8) = "AL"
        x_col(9) = "AO"
        y_col(9) = "AP"
        x_col(10) = "AS"
        y_col(10) = "AT"
        x_col(11) = "AW"
        y_col(11) = "AX"
        x_col(12) = "BA"
        y_col(12) = "BB"

        For i = 0 To 12

            If temp_TEMP(i) = 20 Then fp_TEMP(i) = 1.0 * fp
            If temp_TEMP(i) = 20 Then fy_TEMP(i) = 1.0 * fy
            If temp_TEMP(i) = 20 Then Est_TEMP(i) = 1.0 * Est
```

```
            If temp_TEMP(i) = 100 Then fp_TEMP(i) = 1.0 * fp
            If temp_TEMP(i) = 100 Then fy_TEMP(i) = 1.0 * fy
            If temp_TEMP(i) = 100 Then Est_TEMP(i) = 1.0 * Est

            If temp_TEMP(i) = 200 Then fp_TEMP(i) = 0.81 * fp
            If temp_TEMP(i) = 200 Then fy_TEMP(i) = 1.0 * fy
            If temp_TEMP(i) = 200 Then Est_TEMP(i) = 0.9 * Est

            If temp_TEMP(i) = 300 Then fp_TEMP(i) = 0.61 * fp
            If temp_TEMP(i) = 300 Then fy_TEMP(i) = 1.0 * fy
            If temp_TEMP(i) = 300 Then Est_TEMP(i) = 0.8 * Est

            If temp_TEMP(i) = 400 Then fp_TEMP(i) = 0.42 * fp
            If temp_TEMP(i) = 400 Then fy_TEMP(i) = 1.0 * fy
            If temp_TEMP(i) = 400 Then Est_TEMP(i) = 0.7 * Est

            If temp_TEMP(i) = 500 Then fp_TEMP(i) = 0.36 * fp
            If temp_TEMP(i) = 500 Then fy_TEMP(i) = 0.78 * fy
            If temp_TEMP(i) = 500 Then Est_TEMP(i) = 0.6 * Est

            If temp_TEMP(i) = 600 Then fp_TEMP(i) = 0.18 * fp
            If temp_TEMP(i) = 600 Then fy_TEMP(i) = 0.47 * fy
            If temp_TEMP(i) = 600 Then Est_TEMP(i) = 0.31 * Est

            If temp_TEMP(i) = 700 Then fp_TEMP(i) = 0.07 * fp
            If temp_TEMP(i) = 700 Then fy_TEMP(i) = 0.23 * fy
            If temp_TEMP(i) = 700 Then Est_TEMP(i) = 0.13 * Est

            If temp_TEMP(i) = 800 Then fp_TEMP(i) = 0.05 * fp
            If temp_TEMP(i) = 800 Then fy_TEMP(i) = 0.11 * fy
            If temp_TEMP(i) = 800 Then Est_TEMP(i) = 0.09 * Est

            If temp_TEMP(i) = 900 Then fp_TEMP(i) = 0.04 * fp
            If temp_TEMP(i) = 900 Then fy_TEMP(i) = 0.06 * fy
            If temp_TEMP(i) = 900 Then Est_TEMP(i) = 0.07 * Est

            If temp_TEMP(i) = 1000 Then fp_TEMP(i) = 0.02 * fp
            If temp_TEMP(i) = 1000 Then fy_TEMP(i) = 0.04 * fy
            If temp_TEMP(i) = 1000 Then Est_TEMP(i) = 0.04 * Est

            If temp_TEMP(i) = 1100 Then fp_TEMP(i) = 0.01 * fp
            If temp_TEMP(i) = 1100 Then fy_TEMP(i) = 0.02 * fy
            If temp_TEMP(i) = 1100 Then Est_TEMP(i) = 0.02 * Est

            If temp_TEMP(i) = 1200 Then fp_TEMP(i) = 0.001 * fp
            If temp_TEMP(i) = 1200 Then fy_TEMP(i) = 0.001 * fy
            If temp_TEMP(i) = 1200 Then Est_TEMP(i) = 0.001 * Est

            ep_TEMP(i) = fp_TEMP(i) / Est_TEMP(i)
            ey_TEMP(i) = 0.02
            et_TEMP(i) = 0.15
            eu_TEMP(i) = 0.2
            c_TEMP(i) = ((fy_TEMP(i) - fp_TEMP(i)) ^ 2) / ((ey_TEMP(i) - ep_TEMP(i)) *    ↙
    Est_TEMP(i) - 2 * (fy_TEMP(i) - fp_TEMP(i)))
            a_TEMP(i) = ((ey_TEMP(i) - ep_TEMP(i)) * (ey_TEMP(i) - ep_TEMP(i) + c_TEMP(i)↙
     / Est_TEMP(i))) ^ 0.5
            b_TEMP(i) = (c_TEMP(i) * (ey_TEMP(i) - ep_TEMP(i)) * Est_TEMP(i) + c_TEMP(i) ↙
    ^ 2) ^ 0.5

            prgSS.Value = 500 + (i + 1) * 7
        Next i

        With xlSteel
            For i = 0 To 12

                .Cells(1, 4 * (i + 1)).value = temp_TEMP(i)
```

```
                .Cells(1, 4 * (i + 1) + 1).value = "esT"
                es_TEMP(i, 1) = 0
                .Cells(2, 4 * (i + 1) + 1).value = es_TEMP(i, 1)
                .Cells(1, 4 * (i + 1) + 2).value = "fsT"
                fs_TEMP(i, 1) = 0
                .Cells(2, 4 * (i + 1) + 2).value = fs_TEMP(i, 1)

                For j = 2 To 100
                    es_TEMP(i, j) = es_TEMP(i, j - 1) + eu_TEMP(i) / 99
                    .Cells(j + 1, 4 * (i + 1) + 1).value = es_TEMP(i, j)
                    If es_TEMP(i, j) < ep_TEMP(i) Then fs_TEMP(i, j) = es_TEMP(i, j) *
    Est_TEMP(i)
                    If ep_TEMP(i) < es_TEMP(i, j) And es_TEMP(i, j) < ey_TEMP(i) Then
    fs_TEMP(i, j) = fp_TEMP(i) - c_TEMP(i) + (b_TEMP(i) / a_TEMP(i)) * ((a_TEMP(i) ^ 2) -
     ((ey_TEMP(i) - es_TEMP(i, j)) ^ 2)) ^ 0.5
                    If ey_TEMP(i) < es_TEMP(i, j) And es_TEMP(i, j) < et_TEMP(i) Then
    fs_TEMP(i, j) = fy_TEMP(i)
                    If et_TEMP(i) < es_TEMP(i, j) And es_TEMP(i, j) < eu_TEMP(i) Then
    fs_TEMP(i, j) = fy_TEMP(i) * (1 - (es_TEMP(i, j) - et_TEMP(i)) / (eu_TEMP(i) -
    et_TEMP(i)))
                    If es_TEMP(i, j) = eu_TEMP(i) Then fs_TEMP(i, j) = 0
                    If es_TEMP(i, j) > eu_TEMP(i) Then fs_TEMP(i, j) = 0
                    .Cells(j + 1, 4 * (i + 1) + 2).value = fs_TEMP(i, j)
                    prgSS.Value = 591 + 32 * (i) + Round(j / 5.27)
                Next j

                .Columns(4 * (i + 1) + 1).NumberFormat = "0.00000"
                .Columns(4 * (i + 1) + 2).NumberFormat = "0.00"

                .Cells(2, 4 * (i + 1)).value = fp_TEMP(i)
                .Cells(3, 4 * (i + 1)).value = fy_TEMP(i)
                .Cells(4, 4 * (i + 1)).value = Est_TEMP(i)
                .Cells(5, 4 * (i + 1)).value = ep_TEMP(i)
                .Cells(6, 4 * (i + 1)).value = ey_TEMP(i)
                .Cells(7, 4 * (i + 1)).value = et_TEMP(i)
                .Cells(8, 4 * (i + 1)).value = eu_TEMP(i)
                .Cells(9, 4 * (i + 1)).value = a_TEMP(i)
                .Cells(10, 4 * (i + 1)).value = b_TEMP(i)
                .Cells(11, 4 * (i + 1)).value = c_TEMP(i)

            Next i
        End With

        'Plot steel stress-strain_____________________________________________
        If sSS_new = False Then GoTo 301
        GoTo 302
301:    xlSteel.ChartObjects.Delete()

302:    sSS_new = False
        xlScratch.Cells(8, 1).value = "FALSE"
        Dim Steel_plot As Excel.Chart
        Dim Steel_SerCol As Excel.SeriesCollection
        Dim Steel_Series As Excel.Series
        Dim Steel_AxisCategory, Steel_AxisValue As Excel.Axes

        xlCharts = xlSteel.ChartObjects
        myChart = xlCharts.Add(180, 260, 1000, 600)
        Steel_plot = myChart.Chart
        Steel_plot.ChartType = Excel.XlChartType.xlXYScatterLinesNoMarkers
        Steel_plot.ChartStyle = 3
        Steel_SerCol = Steel_plot.SeriesCollection

        Steel_Series = Steel_SerCol.NewSeries
        Steel_Series.Name = temp_TEMP(0) & "C"
        Steel_Series.XValues = xlSteel.Range("E2:E101")
        Steel_Series.Values = xlSteel.Range("F2:F101")
```

```
        Steel_Series.Format.Line.Weight = 1.5

        With Steel_plot
            .Legend.Position = Excel.XlLegendPosition.xlLegendPositionTop
            .HasTitle = True
            .ChartTitle.Text = "Steel Stress-Strain Relationship"

            Steel_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = True
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.
    Text() = "Strain, mm/mm"
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.
    Font.Bold = False
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.
    Font.Size = 14
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines = True
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines = True
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format.
    Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScale = 0
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MaximumScale = 0.2
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MajorUnitIsAuto = True
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition =
    Excel.XlTickLabelPosition.xlTickLabelPositionLow
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.NumberFormat
    = "0.000"
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold =
    False
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size =
    14

            Steel_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Text() =
    "Stress, MPa"
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Bold
     = False
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Size
     = 14
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line.
    DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScale = 0
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MaximumScaleIsAuto = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MajorUnitIsAuto = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.
    XlTickLabelPosition.xlTickLabelPositionLow
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.NumberFormat = "0"
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold = False
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
        End With
        Steel_plot.Refresh()

        'Export Concrete at 20C
        radSSConcrete.Checked = True
        lstSSTemp.SelectedItems.Clear()
        lstSSTemp.SelectedItem = "20C"
        xlConcrete.ChartObjects(1).chart.Export(FILEname:=OUTPUTfolder & "\temp" & "\
    concreteplot" & DUM & ".bmp", FilterName:="BMP")
        plotSS.BackgroundImage = New System.Drawing.Bitmap(OUTPUTfolder & "\temp" & "\
    concreteplot" & DUM & ".bmp")
        plotSS.BackgroundImageLayout = ImageLayout.Stretch

999:    prgSS.Value = 1000
        DUM = DUM + 1
```

```
    End Sub
    Private Sub btnPLOT_EnabledChanged(ByVal sender As Object, ByVal e As System.
    EventArgs) Handles btnPLOT.EnabledChanged

        If btnPLOT.Enabled = True Then radSSConcrete.Enabled = False
        If btnPLOT.Enabled = True Then radSSSteel.Enabled = False
        If btnPLOT.Enabled = True Then lstSSTemp.Enabled = False

        If btnPLOT.Enabled = False Then radSSConcrete.Enabled = True
        If btnPLOT.Enabled = False Then radSSSteel.Enabled = True
        If btnPLOT.Enabled = False Then lstSSTemp.Enabled = True

    End Sub
    Private Sub radSSConcrete_CheckedChanged(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles radSSConcrete.CheckedChanged

        INITIAL = INITIAL + 1
        If INITIAL = 1 Then GoTo 999
        If radSSConcrete.Checked = True Then GoTo 1
        If radSSSteel.Checked = True Then GoTo 2
        GoTo 999

        '**CONCRETE*****************************************************************
    ***************
1:      prgSS.Value = 0
        xlConcrete.Range("A12:C26").Clear()
        tempCOUNT = lstSSTemp.SelectedItems.Count
        xlConcrete.Cells(13, 1).value = tempCOUNT
        tempCOUNT = tempCOUNT - 1
        If tempCOUNT = -1 Then GoTo 999

        For i = 0 To tempCOUNT
            xlConcrete.Cells(13 + i, 2).value = lstSSTemp.SelectedItems.Item(i)
            If lstSSTemp.SelectedItems.Item(i) = "20C" Then xlConcrete.Cells(13 + i, 3).
    value = 1
            If lstSSTemp.SelectedItems.Item(i) = "100C" Then xlConcrete.Cells(13 + i, 3).
    value = 2
            If lstSSTemp.SelectedItems.Item(i) = "200C" Then xlConcrete.Cells(13 + i, 3).
    value = 3
            If lstSSTemp.SelectedItems.Item(i) = "300C" Then xlConcrete.Cells(13 + i, 3).
    value = 4
            If lstSSTemp.SelectedItems.Item(i) = "400C" Then xlConcrete.Cells(13 + i, 3).
    value = 5
            If lstSSTemp.SelectedItems.Item(i) = "500C" Then xlConcrete.Cells(13 + i, 3).
    value = 6
            If lstSSTemp.SelectedItems.Item(i) = "600C" Then xlConcrete.Cells(13 + i, 3).
    value = 7
            If lstSSTemp.SelectedItems.Item(i) = "700C" Then xlConcrete.Cells(13 + i, 3).
    value = 8
            If lstSSTemp.SelectedItems.Item(i) = "800C" Then xlConcrete.Cells(13 + i, 3).
    value = 9
            If lstSSTemp.SelectedItems.Item(i) = "900C" Then xlConcrete.Cells(13 + i, 3).
    value = 10
            If lstSSTemp.SelectedItems.Item(i) = "1000C" Then xlConcrete.Cells(13 + i, 3)
    .value = 11
            If lstSSTemp.SelectedItems.Item(i) = "1100C" Then xlConcrete.Cells(13 + i, 3)
    .value = 12
            If lstSSTemp.SelectedItems.Item(i) = "1200C" Then xlConcrete.Cells(13 + i, 3)
    .value = 13
        Next i

        'Plot concrete stress-strain_____________________________________________
        xlConcrete.ChartObjects.Delete()

        Dim concrete_plot As Excel.Chart
        Dim xlCharts As Excel.ChartObjects
```

```
        Dim myChart As Excel.ChartObject
        Dim concrete_SerCol As Excel.SeriesCollection
        Dim concrete_Series(0 To 13) As Excel.Series
        Dim concrete_AxisCategory, concrete_AxisValue As Excel.Axes

        xlCharts = xlConcrete.ChartObjects
        myChart = xlCharts.Add(180, 260, 1000, 600)
        concrete_plot = myChart.Chart
        concrete_plot.ChartType = Excel.XlChartType.xlXYScatterLinesNoMarkers
        concrete_plot.ChartStyle = 3
        concrete_SerCol = concrete_plot.SeriesCollection

        plotCOUNT = -1

        For i = 1 To 13
            For j = 0 To 12
                If xlConcrete.Cells(13 + j, 3).value = i Then GoTo 100
            Next j
            GoTo 101
100:        plotCOUNT = plotCOUNT + 1
            concrete_Series(plotCOUNT) = concrete_SerCol.NewSeries
            concrete_Series(plotCOUNT).Name = temp_TEMP(i - 1) & "C"
            concrete_Series(plotCOUNT).XValues = xlConcrete.Range(x_col(i - 1) & "2:" &  ↙
    x_col(i - 1) & "101")
            concrete_Series(plotCOUNT).Values = xlConcrete.Range(y_col(i - 1) & "2:" &   ↙
    y_col(i - 1) & "101")
            concrete_Series(plotCOUNT).Format.Line.Weight = 1.5
101:    Next i

        With concrete_plot
            .Legend.Position = Excel.XlLegendPosition.xlLegendPositionTop
            .HasTitle = True
            .ChartTitle.Text = "Concrete Stress-Strain Relationship"

            concrete_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.↙
    Text() = "Strain, mm/mm"
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.↙
    Font.Bold = False
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.↙
    Font.Size = 14
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines =  ↙
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines =  ↙
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format↙
    .Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScaleIsAuto = ↙
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MaximumScaleIsAuto = ↙
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MajorUnitIsAuto =    ↙
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition =  ↙
    Excel.XlTickLabelPosition.xlTickLabelPositionLow
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.          ↙
    NumberFormat = "0.000"
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold ↙
    = False
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size ↙
    = 14

            concrete_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Text()↙
```

```
     = "Stress, MPa"
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.
    Bold = False
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.
    Size = 14
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line.
    DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScaleIsAuto = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MaximumScaleIsAuto = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MajorUnitIsAuto = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.
    XlTickLabelPosition.xlTickLabelPositionLow
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.NumberFormat =
    "0"
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold =
    False
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
        End With
        concrete_plot.Refresh()

        xlConcrete.ChartObjects(1).chart.Export(FILEname:=OUTPUTfolder & "\temp" & "\
    concreteplot" & DUM & ".bmp", FilterName:="BMP")
        plotSS.BackgroundImage = New System.Drawing.Bitmap(OUTPUTfolder & "\temp" & "\
    concreteplot" & DUM & ".bmp")
        plotSS.BackgroundImageLayout = ImageLayout.Stretch

        DUM = DUM + 1
        GoTo 999

        '**STEEL*****************************************************************************
    ************
2:      prgSS.Value = 0
        xlSteel.Range("A12:C26").Clear()
        tempCOUNT = lstSSTemp.SelectedItems.Count
        xlSteel.Cells(13, 1).value = tempCOUNT
        tempCOUNT = tempCOUNT - 1
        If tempCOUNT = -1 Then GoTo 999

        For i = 0 To tempCOUNT
            xlSteel.Cells(13 + i, 2).value = lstSSTemp.SelectedItems.Item(i)
            If lstSSTemp.SelectedItems.Item(i) = "20C" Then xlSteel.Cells(13 + i, 3).
    value = 1
            If lstSSTemp.SelectedItems.Item(i) = "100C" Then xlSteel.Cells(13 + i, 3).
    value = 2
            If lstSSTemp.SelectedItems.Item(i) = "200C" Then xlSteel.Cells(13 + i, 3).
    value = 3
            If lstSSTemp.SelectedItems.Item(i) = "300C" Then xlSteel.Cells(13 + i, 3).
    value = 4
            If lstSSTemp.SelectedItems.Item(i) = "400C" Then xlSteel.Cells(13 + i, 3).
    value = 5
            If lstSSTemp.SelectedItems.Item(i) = "500C" Then xlSteel.Cells(13 + i, 3).
    value = 6
            If lstSSTemp.SelectedItems.Item(i) = "600C" Then xlSteel.Cells(13 + i, 3).
    value = 7
            If lstSSTemp.SelectedItems.Item(i) = "700C" Then xlSteel.Cells(13 + i, 3).
    value = 8
            If lstSSTemp.SelectedItems.Item(i) = "800C" Then xlSteel.Cells(13 + i, 3).
    value = 9
            If lstSSTemp.SelectedItems.Item(i) = "900C" Then xlSteel.Cells(13 + i, 3).
    value = 10
            If lstSSTemp.SelectedItems.Item(i) = "1000C" Then xlSteel.Cells(13 + i, 3).
    value = 11
            If lstSSTemp.SelectedItems.Item(i) = "1100C" Then xlSteel.Cells(13 + i, 3).
    value = 12
```

```
            If lstSSTemp.SelectedItems.Item(i) = "1200C" Then xlSteel.Cells(13 + i, 3).
    value = 13
        Next i

        'Plot Steel stress-strain_____________________________________________
        xlSteel.ChartObjects.Delete()

        Dim Steel_plot As Excel.Chart
        Dim Steel_SerCol As Excel.SeriesCollection
        Dim Steel_Series(0 To 13) As Excel.Series
        Dim Steel_AxisCategory, Steel_AxisValue As Excel.Axes

        xlCharts = xlSteel.ChartObjects
        myChart = xlCharts.Add(180, 260, 1000, 600)
        Steel_plot = myChart.Chart
        Steel_plot.ChartType = Excel.XlChartType.xlXYScatterLinesNoMarkers
        Steel_plot.ChartStyle = 3
        Steel_SerCol = Steel_plot.SeriesCollection

        plotCOUNT = -1

        For i = 1 To 13
            For j = 0 To 12
                If xlSteel.Cells(13 + j, 3).value = i Then GoTo 200
            Next j
            GoTo 201
200:        plotCOUNT = plotCOUNT + 1
            Steel_Series(plotCOUNT) = Steel_SerCol.NewSeries
            Steel_Series(plotCOUNT).Name = temp_TEMP(i - 1) & "C"
            Steel_Series(plotCOUNT).XValues = xlSteel.Range(x_col(i - 1) & "2:" & x_col(i
     - 1) & "101")
            Steel_Series(plotCOUNT).Values = xlSteel.Range(y_col(i - 1) & "2:" & y_col(i
    - 1) & "101")
            Steel_Series(plotCOUNT).Format.Line.Weight = 1.5
201:    Next i

        With Steel_plot
            .Legend.Position = Excel.XlLegendPosition.xlLegendPositionTop
            .HasTitle = True
            .ChartTitle.Text = "Steel Stress-Strain Relationship"

            Steel_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = True
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.
    Text() = "Strain, mm/mm"
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.
    Font.Bold = False
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.
    Font.Size = 14
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines = True
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines = True
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format.
    Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScale = 0
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MaximumScale = 0.2
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MajorUnitIsAuto = True
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition =
    Excel.XlTickLabelPosition.xlTickLabelPositionLow
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.NumberFormat
    = "0.000"
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold =
    False
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size =
    14

            Steel_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
```

```
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Text() = ↙
    "Stress, MPa"
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Bold↙
     = False
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Size↙
     = 14
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line.   ↙
    DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScale = 0
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MaximumScaleIsAuto = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MajorUnitIsAuto = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.    ↙
    XlTickLabelPosition.xlTickLabelPositionLow
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.NumberFormat = "0"
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold = False
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
        End With

        Steel_plot.Refresh()

        xlSteel.ChartObjects(1).chart.Export(FILEname:=OUTPUTfolder & "\temp" & "\      ↙
    Steelplot" & DUM & ".bmp", FilterName:="BMP")
        plotSS.BackgroundImage = New System.Drawing.Bitmap(OUTPUTfolder & "\temp" & "\   ↙
    Steelplot" & DUM & ".bmp")
        plotSS.BackgroundImageLayout = ImageLayout.Stretch

        DUM = DUM + 1
        GoTo 999

999:    prgSS.Value = 1000
    End Sub
    Private Sub radSSSteel_CheckedChanged(ByVal sender As System.Object, ByVal e As     ↙
    System.EventArgs) Handles radSSSteel.CheckedChanged

        INITIAL = INITIAL + 1
        If INITIAL = 1 Then GoTo 999
        If radSSConcrete.Checked = True Then GoTo 1
        If radSSSteel.Checked = True Then GoTo 2
        GoTo 999

        '**CONCRETE*****************************************************************↙
    ***************
1:      prgSS.Value = 0
        xlConcrete.Range("A12:C26").Clear()
        tempCOUNT = lstSSTemp.SelectedItems.Count
        xlConcrete.Cells(13, 1).value = tempCOUNT
        tempCOUNT = tempCOUNT - 1
        If tempCOUNT = -1 Then GoTo 999

        For i = 0 To tempCOUNT
            xlConcrete.Cells(13 + i, 2).value = lstSSTemp.SelectedItems.Item(i)
            If lstSSTemp.SelectedItems.Item(i) = "20C" Then xlConcrete.Cells(13 + i, 3). ↙
    value = 1
            If lstSSTemp.SelectedItems.Item(i) = "100C" Then xlConcrete.Cells(13 + i, 3).↙
    value = 2
            If lstSSTemp.SelectedItems.Item(i) = "200C" Then xlConcrete.Cells(13 + i, 3).↙
    value = 3
            If lstSSTemp.SelectedItems.Item(i) = "300C" Then xlConcrete.Cells(13 + i, 3).↙
    value = 4
            If lstSSTemp.SelectedItems.Item(i) = "400C" Then xlConcrete.Cells(13 + i, 3).↙
    value = 5
            If lstSSTemp.SelectedItems.Item(i) = "500C" Then xlConcrete.Cells(13 + i, 3).↙
    value = 6
```

```
            If lstSSTemp.SelectedItems.Item(i) = "600C" Then xlConcrete.Cells(13 + i, 3).↙
    value = 7
            If lstSSTemp.SelectedItems.Item(i) = "700C" Then xlConcrete.Cells(13 + i, 3).↙
    value = 8
            If lstSSTemp.SelectedItems.Item(i) = "800C" Then xlConcrete.Cells(13 + i, 3).↙
    value = 9
            If lstSSTemp.SelectedItems.Item(i) = "900C" Then xlConcrete.Cells(13 + i, 3).↙
    value = 10
            If lstSSTemp.SelectedItems.Item(i) = "1000C" Then xlConcrete.Cells(13 + i, 3)↙
    .value = 11
            If lstSSTemp.SelectedItems.Item(i) = "1100C" Then xlConcrete.Cells(13 + i, 3)↙
    .value = 12
            If lstSSTemp.SelectedItems.Item(i) = "1200C" Then xlConcrete.Cells(13 + i, 3)↙
    .value = 13
        Next i

        'Plot concrete stress-strain_____________________________________________
        xlConcrete.ChartObjects.Delete()

        Dim concrete_plot As Excel.Chart
        Dim xlCharts As Excel.ChartObjects
        Dim myChart As Excel.ChartObject
        Dim concrete_SerCol As Excel.SeriesCollection
        Dim concrete_Series(0 To 13) As Excel.Series
        Dim concrete_AxisCategory, concrete_AxisValue As Excel.Axes

        xlCharts = xlConcrete.ChartObjects
        myChart = xlCharts.Add(180, 260, 1000, 600)
        concrete_plot = myChart.Chart
        concrete_plot.ChartType = Excel.XlChartType.xlXYScatterLinesNoMarkers
        concrete_plot.ChartStyle = 3
        concrete_SerCol = concrete_plot.SeriesCollection

        plotCOUNT = -1

        For i = 1 To 13
            For j = 0 To 12
                If xlConcrete.Cells(13 + j, 3).value = i Then GoTo 100
            Next j
            GoTo 101
100:        plotCOUNT = plotCOUNT + 1
            concrete_Series(plotCOUNT) = concrete_SerCol.NewSeries
            concrete_Series(plotCOUNT).Name = temp_TEMP(i - 1) & "C"
            concrete_Series(plotCOUNT).XValues = xlConcrete.Range(x_col(i - 1) & "2:" &  ↙
    x_col(i - 1) & "101")
            concrete_Series(plotCOUNT).Values = xlConcrete.Range(y_col(i - 1) & "2:" &  ↙
    y_col(i - 1) & "101")
            concrete_Series(plotCOUNT).Format.Line.Weight = 1.5
101:    Next i

        With concrete_plot
            .Legend.Position = Excel.XlLegendPosition.xlLegendPositionTop
            .HasTitle = True
            .ChartTitle.Text = "Concrete Stress-Strain Relationship"

            concrete_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.↙
    Text() = "Strain, mm/mm"
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.↙
    Font.Bold = False
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.↙
    Font.Size = 14
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines =  ↙
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines =  ↙
```

```
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format
    .Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScaleIsAuto =
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MaximumScaleIsAuto =
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MajorUnitIsAuto =
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition =
    Excel.XlTickLabelPosition.xlTickLabelPositionLow
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.
    NumberFormat = "0.000"
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold
    = False
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size
    = 14

            concrete_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Text()
     = "Stress, MPa"
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.
    Bold = False
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.
    Size = 14
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line.
    DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScaleIsAuto = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MaximumScaleIsAuto = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MajorUnitIsAuto = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.
    XlTickLabelPosition.xlTickLabelPositionLow
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.NumberFormat =
    "0"
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold =
    False
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
        End With
        concrete_plot.Refresh()

        xlConcrete.ChartObjects(1).chart.Export(FILEname:=OUTPUTfolder & "\temp" & "\
    concreteplot" & DUM & ".bmp", FilterName:="BMP")
        plotSS.BackgroundImage = New System.Drawing.Bitmap(OUTPUTfolder & "\temp" & "\
    concreteplot" & DUM & ".bmp")
        plotSS.BackgroundImageLayout = ImageLayout.Stretch

        DUM = DUM + 1
        GoTo 999

        '**STEEL*****************************************************************
    ************
2:      prgSS.Value = 0
        xlSteel.Range("A12:C26").Clear()
        tempCOUNT = lstSSTemp.SelectedItems.Count
        xlSteel.Cells(13, 1).value = tempCOUNT
        tempCOUNT = tempCOUNT - 1
        If tempCOUNT = -1 Then GoTo 999

        For i = 0 To tempCOUNT
            xlSteel.Cells(13 + i, 2).value = lstSSTemp.SelectedItems.Item(i)
            If lstSSTemp.SelectedItems.Item(i) = "20C" Then xlSteel.Cells(13 + i, 3).
    value = 1
            If lstSSTemp.SelectedItems.Item(i) = "100C" Then xlSteel.Cells(13 + i, 3).
```

```
    value = 2
            If lstSSTemp.SelectedItems.Item(i) = "200C" Then xlSteel.Cells(13 + i, 3).
    value = 3
            If lstSSTemp.SelectedItems.Item(i) = "300C" Then xlSteel.Cells(13 + i, 3).
    value = 4
            If lstSSTemp.SelectedItems.Item(i) = "400C" Then xlSteel.Cells(13 + i, 3).
    value = 5
            If lstSSTemp.SelectedItems.Item(i) = "500C" Then xlSteel.Cells(13 + i, 3).
    value = 6
            If lstSSTemp.SelectedItems.Item(i) = "600C" Then xlSteel.Cells(13 + i, 3).
    value = 7
            If lstSSTemp.SelectedItems.Item(i) = "700C" Then xlSteel.Cells(13 + i, 3).
    value = 8
            If lstSSTemp.SelectedItems.Item(i) = "800C" Then xlSteel.Cells(13 + i, 3).
    value = 9
            If lstSSTemp.SelectedItems.Item(i) = "900C" Then xlSteel.Cells(13 + i, 3).
    value = 10
            If lstSSTemp.SelectedItems.Item(i) = "1000C" Then xlSteel.Cells(13 + i, 3).
    value = 11
            If lstSSTemp.SelectedItems.Item(i) = "1100C" Then xlSteel.Cells(13 + i, 3).
    value = 12
            If lstSSTemp.SelectedItems.Item(i) = "1200C" Then xlSteel.Cells(13 + i, 3).
    value = 13
        Next i

        'Plot Steel stress-strain_____________________________________________
        xlSteel.ChartObjects.Delete()

        Dim Steel_plot As Excel.Chart
        Dim Steel_SerCol As Excel.SeriesCollection
        Dim Steel_Series(0 To 13) As Excel.Series
        Dim Steel_AxisCategory, Steel_AxisValue As Excel.Axes

        xlCharts = xlSteel.ChartObjects
        myChart = xlCharts.Add(180, 260, 1000, 600)
        Steel_plot = myChart.Chart
        Steel_plot.ChartType = Excel.XlChartType.xlXYScatterLinesNoMarkers
        Steel_plot.ChartStyle = 3
        Steel_SerCol = Steel_plot.SeriesCollection

        plotCOUNT = -1

        For i = 1 To 13
            For j = 0 To 12
                If xlSteel.Cells(13 + j, 3).value = i Then GoTo 200
            Next j
            GoTo 201
200:        plotCOUNT = plotCOUNT + 1
            Steel_Series(plotCOUNT) = Steel_SerCol.NewSeries
            Steel_Series(plotCOUNT).Name = temp_TEMP(i - 1) & "C"
            Steel_Series(plotCOUNT).XValues = xlSteel.Range(x_col(i - 1) & "2:" & x_col(i
     - 1) & "101")
            Steel_Series(plotCOUNT).Values = xlSteel.Range(y_col(i - 1) & "2:" & y_col(i
    - 1) & "101")
            Steel_Series(plotCOUNT).Format.Line.Weight = 1.5
201:    Next i

        With Steel_plot
            .Legend.Position = Excel.XlLegendPosition.xlLegendPositionTop
            .HasTitle = True
            .ChartTitle.Text = "Steel Stress-Strain Relationship"

            Steel_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = True
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.
    Text() = "Strain, mm/mm"
```

```
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.
    Font.Bold = False
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.
    Font.Size = 14
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines = True
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines = True
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format.
    Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScale = 0
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MaximumScale = 0.2
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MajorUnitIsAuto = True
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition =
    Excel.XlTickLabelPosition.xlTickLabelPositionLow
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.NumberFormat
    = "0.000"
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold =
    False
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size =
    14

            Steel_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Text() =
    "Stress, MPa"
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Bold
     = False
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Size
     = 14
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line.
    DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScale = 0
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MaximumScaleIsAuto = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MajorUnitIsAuto = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.
    XlTickLabelPosition.xlTickLabelPositionLow
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.NumberFormat = "0"
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold = False
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
        End With

        Steel_plot.Refresh()

        xlSteel.ChartObjects(1).chart.Export(FILEname:=OUTPUTfolder & "\temp" & "\
    Steelplot" & DUM & ".bmp", FilterName:="BMP")
        plotSS.BackgroundImage = New System.Drawing.Bitmap(OUTPUTfolder & "\temp" & "\
    Steelplot" & DUM & ".bmp")
        plotSS.BackgroundImageLayout = ImageLayout.Stretch

        DUM = DUM + 1
        GoTo 999

999:    prgSS.Value = 1000
    End Sub
    Private Sub lstSSTemp_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles lstSSTemp.SelectedIndexChanged

        INITIAL = INITIAL + 1
        If INITIAL = 1 Then GoTo 999
        If radSSConcrete.Checked = True Then GoTo 1
        If radSSSteel.Checked = True Then GoTo 2
        GoTo 999

        '**CONCRETE*****************************************************************
    ***************
```

```
1:      prgSS.Value = 0
        xlConcrete.Range("A12:C26").Clear()
        tempCOUNT = lstSSTemp.SelectedItems.Count
        xlConcrete.Cells(13, 1).value = tempCOUNT
        tempCOUNT = tempCOUNT - 1
        If tempCOUNT = -1 Then GoTo 999

        For i = 0 To tempCOUNT
            xlConcrete.Cells(13 + i, 2).value = lstSSTemp.SelectedItems.Item(i)
            If lstSSTemp.SelectedItems.Item(i) = "20C" Then xlConcrete.Cells(13 + i, 3). ↙
    value = 1
            If lstSSTemp.SelectedItems.Item(i) = "100C" Then xlConcrete.Cells(13 + i, 3).↙
    value = 2
            If lstSSTemp.SelectedItems.Item(i) = "200C" Then xlConcrete.Cells(13 + i, 3).↙
    value = 3
            If lstSSTemp.SelectedItems.Item(i) = "300C" Then xlConcrete.Cells(13 + i, 3).↙
    value = 4
            If lstSSTemp.SelectedItems.Item(i) = "400C" Then xlConcrete.Cells(13 + i, 3).↙
    value = 5
            If lstSSTemp.SelectedItems.Item(i) = "500C" Then xlConcrete.Cells(13 + i, 3).↙
    value = 6
            If lstSSTemp.SelectedItems.Item(i) = "600C" Then xlConcrete.Cells(13 + i, 3).↙
    value = 7
            If lstSSTemp.SelectedItems.Item(i) = "700C" Then xlConcrete.Cells(13 + i, 3).↙
    value = 8
            If lstSSTemp.SelectedItems.Item(i) = "800C" Then xlConcrete.Cells(13 + i, 3).↙
    value = 9
            If lstSSTemp.SelectedItems.Item(i) = "900C" Then xlConcrete.Cells(13 + i, 3).↙
    value = 10
            If lstSSTemp.SelectedItems.Item(i) = "1000C" Then xlConcrete.Cells(13 + i, 3)↙
    .value = 11
            If lstSSTemp.SelectedItems.Item(i) = "1100C" Then xlConcrete.Cells(13 + i, 3)↙
    .value = 12
            If lstSSTemp.SelectedItems.Item(i) = "1200C" Then xlConcrete.Cells(13 + i, 3)↙
    .value = 13
        Next i

        'Plot concrete stress-strain_____________________________________________
        xlConcrete.ChartObjects.Delete()

        Dim concrete_plot As Excel.Chart
        Dim xlCharts As Excel.ChartObjects
        Dim myChart As Excel.ChartObject
        Dim concrete_SerCol As Excel.SeriesCollection
        Dim concrete_Series(0 To 13) As Excel.Series
        Dim concrete_AxisCategory, concrete_AxisValue As Excel.Axes

        xlCharts = xlConcrete.ChartObjects
        myChart = xlCharts.Add(180, 260, 1000, 600)
        concrete_plot = myChart.Chart
        concrete_plot.ChartType = Excel.XlChartType.xlXYScatterLinesNoMarkers
        concrete_plot.ChartStyle = 3
        concrete_SerCol = concrete_plot.SeriesCollection

        plotCOUNT = -1

        For i = 1 To 13
            For j = 0 To 12
                If xlConcrete.Cells(13 + j, 3).value = i Then GoTo 100
            Next j
            GoTo 101
100:        plotCOUNT = plotCOUNT + 1
            concrete_Series(plotCOUNT) = concrete_SerCol.NewSeries
            concrete_Series(plotCOUNT).Name = temp_TEMP(i - 1) & "C"
            concrete_Series(plotCOUNT).XValues = xlConcrete.Range(x_col(i - 1) & "2:" &  ↙
    x_col(i - 1) & "101")
```

```
            concrete_Series(plotCOUNT).Values = xlConcrete.Range(y_col(i - 1) & "2:" &
    y_col(i - 1) & "101")
            concrete_Series(plotCOUNT).Format.Line.Weight = 1.5
101:    Next i

        With concrete_plot
            .Legend.Position = Excel.XlLegendPosition.xlLegendPositionTop
            .HasTitle = True
            .ChartTitle.Text = "Concrete Stress-Strain Relationship"

            concrete_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.
    Text() = "Strain, mm/mm"
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.
    Font.Bold = False
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.
    Font.Size = 14
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines =
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines =
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format
    .Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScaleIsAuto =
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MaximumScaleIsAuto =
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MajorUnitIsAuto =
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition =
    Excel.XlTickLabelPosition.xlTickLabelPositionLow
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.
    NumberFormat = "0.000"
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold
    = False
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size
    = 14

            concrete_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Text()
     = "Stress, MPa"
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.
    Bold = False
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.
    Size = 14
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line.
    DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScaleIsAuto = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MaximumScaleIsAuto = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MajorUnitIsAuto = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.
    XlTickLabelPosition.xlTickLabelPositionLow
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.NumberFormat =
    "0"
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold =
    False
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
        End With
        concrete_plot.Refresh()

        xlConcrete.ChartObjects(1).chart.Export(FILEname:=OUTPUTfolder & "\temp" & "\
    concreteplot" & DUM & ".bmp", FilterName:="BMP")
```

```
        plotSS.BackgroundImage = New System.Drawing.Bitmap(OUTPUTfolder & "\temp" & "\concreteplot" & DUM & ".bmp")
        plotSS.BackgroundImageLayout = ImageLayout.Stretch

        DUM = DUM + 1
        GoTo 999

        '**STEEL*****************************************************************************
2:      prgSS.Value = 0
        xlSteel.Range("A12:C26").Clear()
        tempCOUNT = lstSSTemp.SelectedItems.Count
        xlSteel.Cells(13, 1).value = tempCOUNT
        tempCOUNT = tempCOUNT - 1
        If tempCOUNT = -1 Then GoTo 999

        For i = 0 To tempCOUNT
            xlSteel.Cells(13 + i, 2).value = lstSSTemp.SelectedItems.Item(i)
            If lstSSTemp.SelectedItems.Item(i) = "20C" Then xlSteel.Cells(13 + i, 3).value = 1
            If lstSSTemp.SelectedItems.Item(i) = "100C" Then xlSteel.Cells(13 + i, 3).value = 2
            If lstSSTemp.SelectedItems.Item(i) = "200C" Then xlSteel.Cells(13 + i, 3).value = 3
            If lstSSTemp.SelectedItems.Item(i) = "300C" Then xlSteel.Cells(13 + i, 3).value = 4
            If lstSSTemp.SelectedItems.Item(i) = "400C" Then xlSteel.Cells(13 + i, 3).value = 5
            If lstSSTemp.SelectedItems.Item(i) = "500C" Then xlSteel.Cells(13 + i, 3).value = 6
            If lstSSTemp.SelectedItems.Item(i) = "600C" Then xlSteel.Cells(13 + i, 3).value = 7
            If lstSSTemp.SelectedItems.Item(i) = "700C" Then xlSteel.Cells(13 + i, 3).value = 8
            If lstSSTemp.SelectedItems.Item(i) = "800C" Then xlSteel.Cells(13 + i, 3).value = 9
            If lstSSTemp.SelectedItems.Item(i) = "900C" Then xlSteel.Cells(13 + i, 3).value = 10
            If lstSSTemp.SelectedItems.Item(i) = "1000C" Then xlSteel.Cells(13 + i, 3).value = 11
            If lstSSTemp.SelectedItems.Item(i) = "1100C" Then xlSteel.Cells(13 + i, 3).value = 12
            If lstSSTemp.SelectedItems.Item(i) = "1200C" Then xlSteel.Cells(13 + i, 3).value = 13
        Next i

        'Plot Steel stress-strain_____________________________________________________
        xlSteel.ChartObjects.Delete()

        Dim Steel_plot As Excel.Chart
        Dim Steel_SerCol As Excel.SeriesCollection
        Dim Steel_Series(0 To 13) As Excel.Series
        Dim Steel_AxisCategory, Steel_AxisValue As Excel.Axes

        xlCharts = xlSteel.ChartObjects
        myChart = xlCharts.Add(180, 260, 1000, 600)
        Steel_plot = myChart.Chart
        Steel_plot.ChartType = Excel.XlChartType.xlXYScatterLinesNoMarkers
        Steel_plot.ChartStyle = 3
        Steel_SerCol = Steel_plot.SeriesCollection

        plotCOUNT = -1

        For i = 1 To 13
            For j = 0 To 12
                If xlSteel.Cells(13 + j, 3).value = i Then GoTo 200
```

```
            Next j
            GoTo 201
200:        plotCOUNT = plotCOUNT + 1
            Steel_Series(plotCOUNT) = Steel_SerCol.NewSeries
            Steel_Series(plotCOUNT).Name = temp_TEMP(i - 1) & "C"
            Steel_Series(plotCOUNT).XValues = xlSteel.Range(x_col(i - 1) & "2:" & x_col(i
     - 1) & "101")
            Steel_Series(plotCOUNT).Values = xlSteel.Range(y_col(i - 1) & "2:" & y_col(i
    - 1) & "101")
            Steel_Series(plotCOUNT).Format.Line.Weight = 1.5
201:    Next i

        With Steel_plot
            .Legend.Position = Excel.XlLegendPosition.xlLegendPositionTop
            .HasTitle = True
            .ChartTitle.Text = "Steel Stress-Strain Relationship"

            Steel_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = True
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.
    Text() = "Strain, mm/mm"
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.
    Font.Bold = False
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.
    Font.Size = 14
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines = True
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines = True
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format.
    Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScale = 0
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MaximumScale = 0.2
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MajorUnitIsAuto = True
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition =
    Excel.XlTickLabelPosition.xlTickLabelPositionLow
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.NumberFormat
    = "0.000"
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold =
    False
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size =
    14

            Steel_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Text() =
    "Stress, MPa"
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Bold
     = False
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Size
     = 14
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line.
    DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScale = 0
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MaximumScaleIsAuto = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MajorUnitIsAuto = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.
    XlTickLabelPosition.xlTickLabelPositionLow
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.NumberFormat = "0"
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold = False
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
        End With

        Steel_plot.Refresh()

        xlSteel.ChartObjects(1).chart.Export(FILEname:=OUTPUTfolder & "\temp" & "\
```

```
    Steelplot" & DUM & ".bmp", FilterName:="BMP")
        plotSS.BackgroundImage = New System.Drawing.Bitmap(OUTPUTfolder & "\temp" & "\
    Steelplot" & DUM & ".bmp")
        plotSS.BackgroundImageLayout = ImageLayout.Stretch

        DUM = DUM + 1
        GoTo 999

999:    prgSS.Value = 1000
    End Sub



    '(4)REINFORCEMENT
    Private Sub btnReinforcement_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles btnReinforcement.Click
        Me.tabctrlMain.SelectTab(3)
    End Sub
    Private Sub cmbBarSize1_SelectedIndexChanged(ByVal sender As System.Object, ByVal e
    As System.EventArgs) Handles cmbBarSize1.SelectedIndexChanged

        'Write to excel
        xlREOUT.Cells(2, 38).value = cmbBarSize1.Text

        'Enable next input panel
        If cmbBarSize1.Text = "" Or txtReinfX1.Text = "" Or txtReinfY1.Text = "" Then
            pnlReinf2.Enabled = False
        Else
            pnlReinf2.Enabled = True
        End If

        'Completed_progress4
        If cmbBarSize1.Text = "" Or txtReinfX1.Text = "" Or txtReinfY1.Text = "" Then
            txtProgress4.BackColor = Color.AliceBlue
            txtProgress4.ForeColor = Color.DimGray
        Else
            txtProgress4.BackColor = Color.DodgerBlue
            txtProgress4.ForeColor = Color.White
        End If

    End Sub
    Private Sub cmbBarSize2_SelectedIndexChanged(ByVal sender As System.Object, ByVal e
    As System.EventArgs) Handles cmbBarSize2.SelectedIndexChanged

        'Write to excel
        xlREOUT.Cells(3, 38).value = cmbBarSize2.Text

        'Enable next input panel
        If cmbBarSize2.Text = "" Or txtReinfX2.Text = "" Or txtReinfY2.Text = "" Then
            pnlReinf3.Enabled = False
        Else
            pnlReinf3.Enabled = True
        End If

    End Sub
    Private Sub cmbBarSize3_SelectedIndexChanged(ByVal sender As System.Object, ByVal e
    As System.EventArgs) Handles cmbBarSize3.SelectedIndexChanged

        'Write to excel
        xlREOUT.Cells(4, 38).value = cmbBarSize3.Text

        'Enable next input panel
        If cmbBarSize3.Text = "" Or txtReinfX3.Text = "" Or txtReinfY3.Text = "" Then
            pnlReinf4.Enabled = False
        Else
```

```
            pnlReinf4.Enabled = True
        End If

    End Sub
    Private Sub cmbBarSize4_SelectedIndexChanged(ByVal sender As System.Object, ByVal e ↙
    As System.EventArgs) Handles cmbBarSize4.SelectedIndexChanged

        'Write to excel
        xlREOUT.Cells(5, 38).value = cmbBarSize4.Text

        'Enable next input panel
        If cmbBarSize4.Text = "" Or txtReinfX4.Text = "" Or txtReinfY4.Text = "" Then
            pnlReinf5.Enabled = False
        Else
            pnlReinf5.Enabled = True
        End If

    End Sub
    Private Sub cmbBarSize5_SelectedIndexChanged(ByVal sender As System.Object, ByVal e ↙
    As System.EventArgs) Handles cmbBarSize5.SelectedIndexChanged

        'Write to excel
        xlREOUT.Cells(6, 38).value = cmbBarSize5.Text

        'Enable next input panel
        If cmbBarSize5.Text = "" Or txtReinfX5.Text = "" Or txtReinfY5.Text = "" Then
            pnlReinf6.Enabled = False
        Else
            pnlReinf6.Enabled = True
        End If

    End Sub
    Private Sub cmbBarSize6_SelectedIndexChanged(ByVal sender As System.Object, ByVal e ↙
    As System.EventArgs) Handles cmbBarSize6.SelectedIndexChanged

        'Write to excel
        xlREOUT.Cells(7, 38).value = cmbBarSize6.Text

        'Enable next input panel
        If cmbBarSize6.Text = "" Or txtReinfX6.Text = "" Or txtReinfY6.Text = "" Then
            pnlReinf7.Enabled = False
        Else
            pnlReinf7.Enabled = True
        End If

    End Sub
    Private Sub cmbBarSize7_SelectedIndexChanged(ByVal sender As System.Object, ByVal e ↙
    As System.EventArgs) Handles cmbBarSize7.SelectedIndexChanged

        'Write to excel
        xlREOUT.Cells(8, 38).value = cmbBarSize7.Text

        'Enable next input panel
        If cmbBarSize7.Text = "" Or txtReinfX7.Text = "" Or txtReinfY7.Text = "" Then
            pnlReinf8.Enabled = False
        Else
            pnlReinf8.Enabled = True
        End If

    End Sub
    Private Sub cmbBarSize8_SelectedIndexChanged(ByVal sender As System.Object, ByVal e ↙
    As System.EventArgs) Handles cmbBarSize8.SelectedIndexChanged

        'Write to excel
        xlREOUT.Cells(9, 38).value = cmbBarSize8.Text
```

```
        'Enable next input panel
        If cmbBarSize8.Text = "" Or txtReinfX8.Text = "" Or txtReinfY8.Text = "" Then
            pnlReinf9.Enabled = False
        Else
            pnlReinf9.Enabled = True
        End If

    End Sub
    Private Sub cmbBarSize9_SelectedIndexChanged(ByVal sender As System.Object, ByVal e ↙
    As System.EventArgs) Handles cmbBarSize9.SelectedIndexChanged

        'Write to excel
        xlREOUT.Cells(10, 38).value = cmbBarSize9.Text

        'Enable next input panel
        If cmbBarSize9.Text = "" Or txtReinfX9.Text = "" Or txtReinfY9.Text = "" Then
            pnlReinf10.Enabled = False
        Else
            pnlReinf10.Enabled = True
        End If

    End Sub
    Private Sub cmbBarSize10_SelectedIndexChanged(ByVal sender As System.Object, ByVal e ↙
    As System.EventArgs) Handles cmbBarSize10.SelectedIndexChanged

        'Write to excel
        xlREOUT.Cells(11, 38).value = cmbBarSize10.Text

        'Enable next input panel
        If cmbBarSize10.Text = "" Or txtReinfX10.Text = "" Or txtReinfY10.Text = "" Then
            pnlReinf11.Enabled = False
        Else
            pnlReinf11.Enabled = True
        End If

    End Sub
    Private Sub cmbBarSize11_SelectedIndexChanged(ByVal sender As System.Object, ByVal e ↙
    As System.EventArgs) Handles cmbBarSize11.SelectedIndexChanged

        'Write to excel
        xlREOUT.Cells(12, 38).value = cmbBarSize11.Text

        'Enable next input panel
        If cmbBarSize11.Text = "" Or txtReinfX11.Text = "" Or txtReinfY11.Text = "" Then
            pnlReinf12.Enabled = False
        Else
            pnlReinf12.Enabled = True
        End If

    End Sub
    Private Sub cmbBarSize12_SelectedIndexChanged(ByVal sender As System.Object, ByVal e ↙
    As System.EventArgs) Handles cmbBarSize12.SelectedIndexChanged

        'Write to excel
        xlREOUT.Cells(13, 38).value = cmbBarSize12.Text

        'Enable next input panel
        If cmbBarSize12.Text = "" Or txtReinfX12.Text = "" Or txtReinfY12.Text = "" Then
            pnlReinf13.Enabled = False
        Else
            pnlReinf13.Enabled = True
        End If

    End Sub
    Private Sub cmbBarSize13_SelectedIndexChanged(ByVal sender As System.Object, ByVal e ↙
    As System.EventArgs) Handles cmbBarSize13.SelectedIndexChanged
```

```
        'Write to excel
        xlREOUT.Cells(14, 38).value = cmbBarSize13.Text

        'Enable next input panel
        If cmbBarSize13.Text = "" Or txtReinfX13.Text = "" Or txtReinfY13.Text = "" Then
            pnlReinf14.Enabled = False
        Else
            pnlReinf14.Enabled = True
        End If

    End Sub
    Private Sub cmbBarSize14_SelectedIndexChanged(ByVal sender As System.Object, ByVal e ↙
    As System.EventArgs) Handles cmbBarSize14.SelectedIndexChanged

        'Write to excel
        xlREOUT.Cells(15, 38).value = cmbBarSize14.Text

        'Enable next input panel
        If cmbBarSize14.Text = "" Or txtReinfX14.Text = "" Or txtReinfY14.Text = "" Then
            pnlReinf15.Enabled = False
        Else
            pnlReinf15.Enabled = True
        End If

    End Sub
    Private Sub cmbBarSize15_SelectedIndexChanged(ByVal sender As System.Object, ByVal e ↙
    As System.EventArgs) Handles cmbBarSize15.SelectedIndexChanged

        'Write to excel
        xlREOUT.Cells(16, 38).value = cmbBarSize15.Text

        'Enable next input panel
        If cmbBarSize15.Text = "" Or txtReinfX15.Text = "" Or txtReinfY15.Text = "" Then
            pnlReinf16.Enabled = False
        Else
            pnlReinf16.Enabled = True
        End If

    End Sub
    Private Sub cmbBarSize16_SelectedIndexChanged(ByVal sender As System.Object, ByVal e ↙
    As System.EventArgs) Handles cmbBarSize16.SelectedIndexChanged

        'Write to excel
        xlREOUT.Cells(17, 38).value = cmbBarSize16.Text

        'Enable next input panel
        If cmbBarSize16.Text = "" Or txtReinfX16.Text = "" Or txtReinfY16.Text = "" Then
            pnlReinf17.Enabled = False
        Else
            pnlReinf17.Enabled = True
        End If

    End Sub
    Private Sub cmbBarSize17_SelectedIndexChanged(ByVal sender As System.Object, ByVal e ↙
    As System.EventArgs) Handles cmbBarSize17.SelectedIndexChanged

        'Write to excel
        xlREOUT.Cells(18, 38).value = cmbBarSize17.Text

        'Enable next input panel
        If cmbBarSize17.Text = "" Or txtReinfX17.Text = "" Or txtReinfY17.Text = "" Then
            pnlReinf18.Enabled = False
        Else
            pnlReinf18.Enabled = True
        End If
```

```
    End Sub
    Private Sub cmbBarSize18_SelectedIndexChanged(ByVal sender As System.Object, ByVal e ↙
    As System.EventArgs) Handles cmbBarSize18.SelectedIndexChanged

        'Write to excel
        xlREOUT.Cells(19, 38).value = cmbBarSize18.Text

    End Sub
    Private Sub txtReinfX1_TextChanged(ByVal sender As System.Object, ByVal e As System. ↙
    EventArgs) Handles txtReinfX1.TextChanged

        'Write to excel
        xlREOUT.Cells(2, 39).value = txtReinfX1.Text

        'Enable next input panel
        If cmbBarSize1.Text = "" Or txtReinfX1.Text = "" Or txtReinfY1.Text = "" Then
            pnlReinf2.Enabled = False
        Else
            pnlReinf2.Enabled = True
        End If

        'Completed_progress4
        If cmbBarSize1.Text = "" Or txtReinfX1.Text = "" Or txtReinfY1.Text = "" Then
            txtProgress4.BackColor = Color.AliceBlue
            txtProgress4.ForeColor = Color.DimGray
        Else
            txtProgress4.BackColor = Color.DodgerBlue
            txtProgress4.ForeColor = Color.White
        End If

    End Sub
    Private Sub txtReinfX2_TextChanged(ByVal sender As System.Object, ByVal e As System. ↙
    EventArgs) Handles txtReinfX2.TextChanged

        'Write to excel
        xlREOUT.Cells(3, 39).value = txtReinfX2.Text

        'Enable next input panel
        If cmbBarSize2.Text = "" Or txtReinfX2.Text = "" Or txtReinfY2.Text = "" Then
            pnlReinf3.Enabled = False
        Else
            pnlReinf3.Enabled = True
        End If

    End Sub
    Private Sub txtReinfX3_TextChanged(ByVal sender As System.Object, ByVal e As System. ↙
    EventArgs) Handles txtReinfX3.TextChanged

        'Write to excel
        xlREOUT.Cells(4, 39).value = txtReinfX3.Text

        'Enable next input panel
        If cmbBarSize3.Text = "" Or txtReinfX3.Text = "" Or txtReinfY3.Text = "" Then
            pnlReinf4.Enabled = False
        Else
            pnlReinf4.Enabled = True
        End If

    End Sub
    Private Sub txtReinfX4_TextChanged(ByVal sender As System.Object, ByVal e As System. ↙
    EventArgs) Handles txtReinfX4.TextChanged

        'Write to excel
        xlREOUT.Cells(5, 39).value = txtReinfX4.Text
```

```
        'Enable next input panel
        If cmbBarSize4.Text = "" Or txtReinfX4.Text = "" Or txtReinfY4.Text = "" Then
            pnlReinf5.Enabled = False
        Else
            pnlReinf5.Enabled = True
        End If

    End Sub
    Private Sub txtReinfX5_TextChanged(ByVal sender As System.Object, ByVal e As System. ↙
    EventArgs) Handles txtReinfX5.TextChanged

        'Write to excel
        xlREOUT.Cells(6, 39).value = txtReinfX5.Text

        'Enable next input panel
        If cmbBarSize5.Text = "" Or txtReinfX5.Text = "" Or txtReinfY5.Text = "" Then
            pnlReinf6.Enabled = False
        Else
            pnlReinf6.Enabled = True
        End If

    End Sub
    Private Sub txtReinfX6_TextChanged(ByVal sender As System.Object, ByVal e As System. ↙
    EventArgs) Handles txtReinfX6.TextChanged

        'Write to excel
        xlREOUT.Cells(7, 39).value = txtReinfX6.Text

        'Enable next input panel
        If cmbBarSize6.Text = "" Or txtReinfX6.Text = "" Or txtReinfY6.Text = "" Then
            pnlReinf7.Enabled = False
        Else
            pnlReinf7.Enabled = True
        End If

    End Sub
    Private Sub txtReinfX7_TextChanged(ByVal sender As System.Object, ByVal e As System. ↙
    EventArgs) Handles txtReinfX7.TextChanged

        'Write to excel
        xlREOUT.Cells(8, 39).value = txtReinfX7.Text

        'Enable next input panel
        If cmbBarSize7.Text = "" Or txtReinfX7.Text = "" Or txtReinfY7.Text = "" Then
            pnlReinf8.Enabled = False
        Else
            pnlReinf8.Enabled = True
        End If

    End Sub
    Private Sub txtReinfX8_TextChanged(ByVal sender As System.Object, ByVal e As System. ↙
    EventArgs) Handles txtReinfX8.TextChanged

        'Write to excel
        xlREOUT.Cells(9, 39).value = txtReinfX8.Text

        'Enable next input panel
        If cmbBarSize8.Text = "" Or txtReinfX8.Text = "" Or txtReinfY8.Text = "" Then
            pnlReinf9.Enabled = False
        Else
            pnlReinf9.Enabled = True
        End If

    End Sub
    Private Sub txtReinfX9_TextChanged(ByVal sender As System.Object, ByVal e As System. ↙
    EventArgs) Handles txtReinfX9.TextChanged
```

```
        'Write to excel
        xlREOUT.Cells(10, 39).value = txtReinfX9.Text

        'Enable next input panel
        If cmbBarSize9.Text = "" Or txtReinfX9.Text = "" Or txtReinfY9.Text = "" Then
            pnlReinf10.Enabled = False
        Else
            pnlReinf10.Enabled = True
        End If

    End Sub
    Private Sub txtReinfX10_TextChanged(ByVal sender As System.Object, ByVal e As System.↙
    EventArgs) Handles txtReinfX10.TextChanged

        'Write to excel
        xlREOUT.Cells(11, 39).value = txtReinfX10.Text

        'Enable next input panel
        If cmbBarSize10.Text = "" Or txtReinfX10.Text = "" Or txtReinfY10.Text = "" Then
            pnlReinf11.Enabled = False
        Else
            pnlReinf11.Enabled = True
        End If

    End Sub
    Private Sub txtReinfX11_TextChanged(ByVal sender As System.Object, ByVal e As System.↙
    EventArgs) Handles txtReinfX11.TextChanged

        'Write to excel
        xlREOUT.Cells(12, 39).value = txtReinfX11.Text

        'Enable next input panel
        If cmbBarSize11.Text = "" Or txtReinfX11.Text = "" Or txtReinfY11.Text = "" Then
            pnlReinf12.Enabled = False
        Else
            pnlReinf12.Enabled = True
        End If

    End Sub
    Private Sub txtReinfX12_TextChanged(ByVal sender As System.Object, ByVal e As System.↙
    EventArgs) Handles txtReinfX12.TextChanged

        'Write to excel
        xlREOUT.Cells(13, 39).value = txtReinfX12.Text

        'Enable next input panel
        If cmbBarSize12.Text = "" Or txtReinfX12.Text = "" Or txtReinfY12.Text = "" Then
            pnlReinf13.Enabled = False
        Else
            pnlReinf13.Enabled = True
        End If

    End Sub
    Private Sub txtReinfX13_TextChanged(ByVal sender As System.Object, ByVal e As System.↙
    EventArgs) Handles txtReinfX13.TextChanged

        'Write to excel
        xlREOUT.Cells(14, 39).value = txtReinfX13.Text

        'Enable next input panel
        If cmbBarSize13.Text = "" Or txtReinfX13.Text = "" Or txtReinfY13.Text = "" Then
            pnlReinf14.Enabled = False
        Else
            pnlReinf14.Enabled = True
        End If
```

```
    End Sub
    Private Sub txtReinfX14_TextChanged(ByVal sender As System.Object, ByVal e As System.↙
    EventArgs) Handles txtReinfX14.TextChanged

        'Write to excel
        xlREOUT.Cells(15, 39).value = txtReinfX14.Text

        'Enable next input panel
        If cmbBarSize14.Text = "" Or txtReinfX14.Text = "" Or txtReinfY14.Text = "" Then
            pnlReinf15.Enabled = False
        Else
            pnlReinf15.Enabled = True
        End If

    End Sub
    Private Sub txtReinfX15_TextChanged(ByVal sender As System.Object, ByVal e As System.↙
    EventArgs) Handles txtReinfX15.TextChanged

        'Write to excel
        xlREOUT.Cells(16, 39).value = txtReinfX15.Text

        'Enable next input panel
        If cmbBarSize15.Text = "" Or txtReinfX15.Text = "" Or txtReinfY15.Text = "" Then
            pnlReinf16.Enabled = False
        Else
            pnlReinf16.Enabled = True
        End If

    End Sub
    Private Sub txtReinfX16_TextChanged(ByVal sender As System.Object, ByVal e As System.↙
    EventArgs) Handles txtReinfX16.TextChanged

        'Write to excel
        xlREOUT.Cells(17, 39).value = txtReinfX16.Text

        'Enable next input panel
        If cmbBarSize16.Text = "" Or txtReinfX16.Text = "" Or txtReinfY16.Text = "" Then
            pnlReinf17.Enabled = False
        Else
            pnlReinf17.Enabled = True
        End If

    End Sub
    Private Sub txtReinfX17_TextChanged(ByVal sender As System.Object, ByVal e As System.↙
    EventArgs) Handles txtReinfX17.TextChanged

        'Write to excel
        xlREOUT.Cells(18, 39).value = txtReinfX17.Text

        'Enable next input panel
        If cmbBarSize17.Text = "" Or txtReinfX17.Text = "" Or txtReinfY17.Text = "" Then
            pnlReinf18.Enabled = False
        Else
            pnlReinf18.Enabled = True
        End If

    End Sub
    Private Sub txtReinfX18_TextChanged(ByVal sender As System.Object, ByVal e As System.↙
    EventArgs) Handles txtReinfX18.TextChanged

        'Write to excel
        xlREOUT.Cells(19, 39).value = txtReinfX18.Text

    End Sub
    Private Sub txtReinfY1_TextChanged(ByVal sender As System.Object, ByVal e As System. ↙
```

```
    EventArgs) Handles txtReinfY1.TextChanged

        'Write to excel
        xlREOUT.Cells(2, 40).value = txtReinfY1.Text

        'Enable next input panel
        If cmbBarSize1.Text = "" Or txtReinfX1.Text = "" Or txtReinfY1.Text = "" Then
            pnlReinf2.Enabled = False
        Else
            pnlReinf2.Enabled = True
        End If

        'Completed_progress4
        If cmbBarSize1.Text = "" Or txtReinfX1.Text = "" Or txtReinfY1.Text = "" Then
            txtProgress4.BackColor = Color.AliceBlue
            txtProgress4.ForeColor = Color.DimGray
        Else
            txtProgress4.BackColor = Color.DodgerBlue
            txtProgress4.ForeColor = Color.White
        End If

    End Sub
    Private Sub txtReinfY2_TextChanged(ByVal sender As System.Object, ByVal e As System. ↙
    EventArgs) Handles txtReinfY2.TextChanged

        'Write to excel
        xlREOUT.Cells(3, 40).value = txtReinfY2.Text

        'Enable next input panel
        If cmbBarSize2.Text = "" Or txtReinfX2.Text = "" Or txtReinfY2.Text = "" Then
            pnlReinf3.Enabled = False
        Else
            pnlReinf3.Enabled = True
        End If

    End Sub
    Private Sub txtReinfY3_TextChanged(ByVal sender As System.Object, ByVal e As System. ↙
    EventArgs) Handles txtReinfY3.TextChanged

        'Write to excel
        xlREOUT.Cells(4, 40).value = txtReinfY3.Text

        'Enable next input panel
        If cmbBarSize3.Text = "" Or txtReinfX3.Text = "" Or txtReinfY3.Text = "" Then
            pnlReinf4.Enabled = False
        Else
            pnlReinf4.Enabled = True
        End If

    End Sub
    Private Sub txtReinfY4_TextChanged(ByVal sender As System.Object, ByVal e As System. ↙
    EventArgs) Handles txtReinfY4.TextChanged

        'Write to excel
        xlREOUT.Cells(5, 40).value = txtReinfY4.Text

        'Enable next input panel
        If cmbBarSize4.Text = "" Or txtReinfX4.Text = "" Or txtReinfY4.Text = "" Then
            pnlReinf5.Enabled = False
        Else
            pnlReinf5.Enabled = True
        End If

    End Sub
    Private Sub txtReinfY5_TextChanged(ByVal sender As System.Object, ByVal e As System. ↙
    EventArgs) Handles txtReinfY5.TextChanged
```

```
        'Write to excel
        xlREOUT.Cells(6, 40).value = txtReinfY5.Text

        'Enable next input panel
        If cmbBarSize5.Text = "" Or txtReinfX5.Text = "" Or txtReinfY5.Text = "" Then
            pnlReinf6.Enabled = False
        Else
            pnlReinf6.Enabled = True
        End If

    End Sub
    Private Sub txtReinfY6_TextChanged(ByVal sender As System.Object, ByVal e As System. ↙
    EventArgs) Handles txtReinfY6.TextChanged

        'Write to excel
        xlREOUT.Cells(7, 40).value = txtReinfY6.Text

        'Enable next input panel
        If cmbBarSize6.Text = "" Or txtReinfX6.Text = "" Or txtReinfY6.Text = "" Then
            pnlReinf7.Enabled = False
        Else
            pnlReinf7.Enabled = True
        End If

    End Sub
    Private Sub txtReinfY7_TextChanged(ByVal sender As System.Object, ByVal e As System. ↙
    EventArgs) Handles txtReinfY7.TextChanged

        'Write to excel
        xlREOUT.Cells(8, 40).value = txtReinfY7.Text

        'Enable next input panel
        If cmbBarSize7.Text = "" Or txtReinfX7.Text = "" Or txtReinfY7.Text = "" Then
            pnlReinf8.Enabled = False
        Else
            pnlReinf8.Enabled = True
        End If

    End Sub
    Private Sub txtReinfY8_TextChanged(ByVal sender As System.Object, ByVal e As System. ↙
    EventArgs) Handles txtReinfY8.TextChanged

        'Write to excel
        xlREOUT.Cells(9, 40).value = txtReinfY8.Text

        'Enable next input panel
        If cmbBarSize8.Text = "" Or txtReinfX8.Text = "" Or txtReinfY8.Text = "" Then
            pnlReinf9.Enabled = False
        Else
            pnlReinf9.Enabled = True
        End If

    End Sub
    Private Sub txtReinfY9_TextChanged(ByVal sender As System.Object, ByVal e As System. ↙
    EventArgs) Handles txtReinfY9.TextChanged

        'Write to excel
        xlREOUT.Cells(10, 40).value = txtReinfY9.Text

        'Enable next input panel
        If cmbBarSize9.Text = "" Or txtReinfX9.Text = "" Or txtReinfY9.Text = "" Then
            pnlReinf10.Enabled = False
        Else
            pnlReinf10.Enabled = True
        End If
```

```
    End Sub
    Private Sub txtReinfY10_TextChanged(ByVal sender As System.Object, ByVal e As System.↙
    EventArgs) Handles txtReinfY10.TextChanged

        'Write to excel
        xlREOUT.Cells(11, 40).value = txtReinfY10.Text

        'Enable next input panel
        If cmbBarSize10.Text = "" Or txtReinfX10.Text = "" Or txtReinfY10.Text = "" Then
            pnlReinf11.Enabled = False
        Else
            pnlReinf11.Enabled = True
        End If

    End Sub
    Private Sub txtReinfY11_TextChanged(ByVal sender As System.Object, ByVal e As System.↙
    EventArgs) Handles txtReinfY11.TextChanged

        'Write to excel
        xlREOUT.Cells(12, 40).value = txtReinfY11.Text

        'Enable next input panel
        If cmbBarSize11.Text = "" Or txtReinfX11.Text = "" Or txtReinfY11.Text = "" Then
            pnlReinf12.Enabled = False
        Else
            pnlReinf12.Enabled = True
        End If

    End Sub
    Private Sub txtReinfY12_TextChanged(ByVal sender As System.Object, ByVal e As System.↙
    EventArgs) Handles txtReinfY12.TextChanged

        'Write to excel
        xlREOUT.Cells(13, 40).value = txtReinfY12.Text

        'Enable next input panel
        If cmbBarSize12.Text = "" Or txtReinfX12.Text = "" Or txtReinfY12.Text = "" Then
            pnlReinf13.Enabled = False
        Else
            pnlReinf13.Enabled = True
        End If

    End Sub
    Private Sub txtReinfY13_TextChanged(ByVal sender As System.Object, ByVal e As System.↙
    EventArgs) Handles txtReinfY13.TextChanged

        'Write to excel
        xlREOUT.Cells(14, 40).value = txtReinfY13.Text

        'Enable next input panel
        If cmbBarSize13.Text = "" Or txtReinfX13.Text = "" Or txtReinfY13.Text = "" Then
            pnlReinf14.Enabled = False
        Else
            pnlReinf14.Enabled = True
        End If

    End Sub
    Private Sub txtReinfY14_TextChanged(ByVal sender As System.Object, ByVal e As System.↙
    EventArgs) Handles txtReinfY14.TextChanged

        'Write to excel
        xlREOUT.Cells(15, 40).value = txtReinfY14.Text

        'Enable next input panel
        If cmbBarSize14.Text = "" Or txtReinfX14.Text = "" Or txtReinfY14.Text = "" Then
```

```
            pnlReinf15.Enabled = False
        Else
            pnlReinf15.Enabled = True
        End If

    End Sub
    Private Sub txtReinfY15_TextChanged(ByVal sender As System.Object, ByVal e As System.↙
    EventArgs) Handles txtReinfY15.TextChanged

        'Write to excel
        xlREOUT.Cells(16, 40).value = txtReinfY15.Text

        'Enable next input panel
        If cmbBarSize15.Text = "" Or txtReinfX15.Text = "" Or txtReinfY15.Text = "" Then
            pnlReinf16.Enabled = False
        Else
            pnlReinf16.Enabled = True
        End If

    End Sub
    Private Sub txtReinfY16_TextChanged(ByVal sender As System.Object, ByVal e As System.↙
    EventArgs) Handles txtReinfY16.TextChanged

        'Write to excel
        xlREOUT.Cells(17, 40).value = txtReinfY16.Text

        'Enable next input panel
        If cmbBarSize16.Text = "" Or txtReinfX16.Text = "" Or txtReinfY16.Text = "" Then
            pnlReinf17.Enabled = False
        Else
            pnlReinf17.Enabled = True
        End If

    End Sub
    Private Sub txtReinfY17_TextChanged(ByVal sender As System.Object, ByVal e As System.↙
    EventArgs) Handles txtReinfY17.TextChanged

        'Write to excel
        xlREOUT.Cells(18, 40).value = txtReinfY17.Text

        'Enable next input panel
        If cmbBarSize17.Text = "" Or txtReinfX17.Text = "" Or txtReinfY17.Text = "" Then
            pnlReinf18.Enabled = False
        Else
            pnlReinf18.Enabled = True
        End If

    End Sub
    Private Sub txtReinfY18_TextChanged(ByVal sender As System.Object, ByVal e As System.↙
    EventArgs) Handles txtReinfY18.TextChanged

        'Write to excel
        xlREOUT.Cells(19, 40).value = txtReinfY18.Text

    End Sub



    '(5)ANALYSIS_OPTIONS
    Private Sub btnAnalysis_Click(ByVal sender As System.Object, ByVal e As System.      ↙
    EventArgs) Handles btnAnalysis.Click
        Me.tabctrlMain.SelectTab(4)
    End Sub
    Private Sub txtAccuracy_TextChanged(ByVal sender As System.Object, ByVal e As System.↙
    EventArgs) Handles txtAccuracy.TextChanged
```

```
        If txtAccuracy.Enabled = True Then
            xlMPhi.Cells(4, 1).value = txtAccuracy.Text
            txtProgress5.BackColor = Color.DodgerBlue
            txtProgress5.ForeColor = Color.White
        Else
            txtProgress5.BackColor = Color.AliceBlue
            txtProgress5.ForeColor = Color.DimGray
        End If
    End Sub


    '(6)PREVIEW_SECTION
    Private Sub btnPreviewSection_Click(ByVal sender As System.Object, ByVal e As System.↙
    EventArgs) Handles btnPreviewSection.Click
        Me.tabctrlMain.SelectTab(5)

        'Refresh count
        DUM = DUM + 1

        'Make sure all bars all written to Excel
        xlREOUT.Cells(2, 38).value = cmbBarSize1.Text
        xlREOUT.Cells(3, 38).value = cmbBarSize2.Text
        xlREOUT.Cells(4, 38).value = cmbBarSize3.Text
        xlREOUT.Cells(5, 38).value = cmbBarSize4.Text
        xlREOUT.Cells(6, 38).value = cmbBarSize5.Text
        xlREOUT.Cells(7, 38).value = cmbBarSize6.Text
        xlREOUT.Cells(8, 38).value = cmbBarSize7.Text
        xlREOUT.Cells(9, 38).value = cmbBarSize8.Text
        xlREOUT.Cells(10, 38).value = cmbBarSize9.Text
        xlREOUT.Cells(11, 38).value = cmbBarSize10.Text
        xlREOUT.Cells(12, 38).value = cmbBarSize11.Text
        xlREOUT.Cells(13, 38).value = cmbBarSize12.Text
        xlREOUT.Cells(14, 38).value = cmbBarSize13.Text
        xlREOUT.Cells(15, 38).value = cmbBarSize14.Text
        xlREOUT.Cells(16, 38).value = cmbBarSize15.Text
        xlREOUT.Cells(17, 38).value = cmbBarSize16.Text
        xlREOUT.Cells(18, 38).value = cmbBarSize17.Text
        xlREOUT.Cells(19, 38).value = cmbBarSize18.Text
        xlREOUT.Cells(2, 39).value = txtReinfX1.Text
        xlREOUT.Cells(3, 39).value = txtReinfX2.Text
        xlREOUT.Cells(4, 39).value = txtReinfX3.Text
        xlREOUT.Cells(5, 39).value = txtReinfX4.Text
        xlREOUT.Cells(6, 39).value = txtReinfX5.Text
        xlREOUT.Cells(7, 39).value = txtReinfX6.Text
        xlREOUT.Cells(8, 39).value = txtReinfX7.Text
        xlREOUT.Cells(9, 39).value = txtReinfX8.Text
        xlREOUT.Cells(10, 39).value = txtReinfX9.Text
        xlREOUT.Cells(11, 39).value = txtReinfX10.Text
        xlREOUT.Cells(12, 39).value = txtReinfX11.Text
        xlREOUT.Cells(13, 39).value = txtReinfX12.Text
        xlREOUT.Cells(14, 39).value = txtReinfX13.Text
        xlREOUT.Cells(15, 39).value = txtReinfX14.Text
        xlREOUT.Cells(16, 39).value = txtReinfX15.Text
        xlREOUT.Cells(17, 39).value = txtReinfX16.Text
        xlREOUT.Cells(18, 39).value = txtReinfX17.Text
        xlREOUT.Cells(19, 39).value = txtReinfX18.Text
        xlREOUT.Cells(2, 40).value = txtReinfY1.Text
        xlREOUT.Cells(3, 40).value = txtReinfY2.Text
        xlREOUT.Cells(4, 40).value = txtReinfY3.Text
        xlREOUT.Cells(5, 40).value = txtReinfY4.Text
        xlREOUT.Cells(6, 40).value = txtReinfY5.Text
        xlREOUT.Cells(7, 40).value = txtReinfY6.Text
        xlREOUT.Cells(8, 40).value = txtReinfY7.Text
        xlREOUT.Cells(9, 40).value = txtReinfY8.Text
        xlREOUT.Cells(10, 40).value = txtReinfY9.Text
```

```
        xlREOUT.Cells(11, 40).value = txtReinfY10.Text
        xlREOUT.Cells(12, 40).value = txtReinfY11.Text
        xlREOUT.Cells(13, 40).value = txtReinfY12.Text
        xlREOUT.Cells(14, 40).value = txtReinfY13.Text
        xlREOUT.Cells(15, 40).value = txtReinfY14.Text
        xlREOUT.Cells(16, 40).value = txtReinfY15.Text
        xlREOUT.Cells(17, 40).value = txtReinfY16.Text
        xlREOUT.Cells(18, 40).value = txtReinfY17.Text
        xlREOUT.Cells(19, 40).value = txtReinfY18.Text

        'Count reinforcing bars
        For i = 1 To 18
            If xlREOUT.Cells(i + 1, 38).value = 0 Or xlREOUT.Cells(i + 1, 39).value = 0 ↙
    Or xlREOUT.Cells(i + 1, 40).value = 0 Then
                REINFcount = i - 1
                GoTo 100
            End If
        Next i
        REINFcount = 18

100:    xlREOUT.Cells(23, 2).value = REINFcount

        For i = REINFcount + 1 To 19
            xlREOUT.Cells(i + 1, 38).value = ""
            xlREOUT.Cells(i + 1, 39).value = ""
            xlREOUT.Cells(i + 1, 40).value = ""
        Next i

        'Convert input to SAFIR coordinate system
        For i = 1 To REINFcount
            xlSection.Cells(i, 4).value = -s_width / 2 + xlREOUT.Cells(i + 1, 39).value
            xlREOUT.Cells(i + 1, 41).value = -s_width / 2 + xlREOUT.Cells(i + 1, 39).    ↙
    value
            xlSection.Cells(i, 5).value = -s_height / 2 + xlREOUT.Cells(i + 1, 40).value
            xlREOUT.Cells(i + 1, 42).value = -s_height / 2 + xlREOUT.Cells(i + 1, 40).   ↙
    value
        Next i

        'PLOT_SECTION=========================================================================↙
    ====================================
        xlSection.ChartObjects(1).Delete()

        Dim xlCharts As Excel.ChartObjects
        Dim myChart As Excel.ChartObject
        Dim sectionplot As Excel.Chart
        Dim section_SerCol As Excel.SeriesCollection
        Dim section_Series, reinf_Series As Excel.Series
        Dim section_AxisCategory, section_AxisValue As Excel.Axes

        xlCharts = xlSection.ChartObjects
        myChart = xlCharts.Add(260, 10, 900, 600)
        sectionplot = myChart.Chart
        sectionplot.ChartType = Excel.XlChartType.xlXYScatter
        section_SerCol = sectionplot.SeriesCollection

        section_Series = section_SerCol.NewSeries
        section_Series.Name = "section"
        section_Series.XValues = xlSection.Range("A1:A" & Nodes)
        section_Series.Values = xlSection.Range("B1:B" & Nodes)
        section_Series.MarkerSize = 5
        section_Series.MarkerStyle = Excel.XlMarkerStyle.xlMarkerStyleSquare

        reinf_Series = section_SerCol.NewSeries
        reinf_Series.Name = "reinf"
        reinf_Series.XValues = xlSection.Range("D1:D" & REINFcount)
        reinf_Series.Values = xlSection.Range("E1:E" & REINFcount)
```

```
        reinf_Series.MarkerSize = 12
        reinf_Series.MarkerStyle = Excel.XlMarkerStyle.xlMarkerStyleCircle

        'max,min X & Y values
        Xaxis_max = 0
        Xaxis_max = 0
        Xaxis_min = 0
        Yaxis_min = 0

        For i = 1 To Nodes
            If nodeX(i) > Xaxis_max Then Xaxis_max = nodeX(i)
            If nodeY(i) > Yaxis_max Then Yaxis_max = nodeY(i)
        Next i
        If Xaxis_max > Yaxis_max Then
            axis_max = Xaxis_max
        Else
            axis_max = Yaxis_max
        End If

        For i = 1 To Nodes
            If nodeX(i) < Xaxis_min Then Xaxis_min = nodeX(i)
            If nodeY(i) < Yaxis_min Then Yaxis_min = nodeY(i)
        Next i
        If Xaxis_min < Yaxis_min Then
            axis_min = Xaxis_min
        Else
            axis_min = Yaxis_min
        End If

        If axis_max > Abs(axis_min) Then
            square_scale = axis_max
        Else
            square_scale = Abs(axis_min)
        End If

        s_width = Abs(Xaxis_min) + Abs(Xaxis_max)
        s_height = Abs(Yaxis_min) + Abs(Yaxis_max)
        xlSection.Cells(1, 3).value = s_width
        xlSection.Cells(2, 3).value = s_height

        With sectionplot
            .Legend.Delete()
            .HasTitle = True
            .ChartTitle.Text = lblSection.Text

            section_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = False
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines =
    True
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format.
    Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines =
    True
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScale = -
    (square_scale + square_scale / 10)
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).MaximumScale =
    (square_scale + square_scale / 10)
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).MajorUnit = Round(
    (square_scale + square_scale / 10) * 2 / 5, 1)
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition =
    Excel.XlTickLabelPosition.xlTickLabelPositionLow
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).CrossesAt = 0
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold =
     False
            section_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size =
     14
```

```
            section_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
            section_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = False
            section_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
            section_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
            section_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line.
    DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            section_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScale = -
    (square_scale + square_scale / 10)
            section_AxisValue.Item(Excel.XlAxisType.xlValue).MaximumScale = (square_scale
     + square_scale / 10)
            section_AxisValue.Item(Excel.XlAxisType.xlValue).MajorUnit = Round(
    (square_scale + square_scale / 10) * 2 / 5, 1)
            section_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.
    XlTickLabelPosition.xlTickLabelPositionLow
            section_AxisValue.Item(Excel.XlAxisType.xlValue).CrossesAt = 0
            section_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold = False
            section_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
        End With

        sectionplot.Refresh()

        'exporting chart as picture file
        xlSection.ChartObjects(1).chart.Export(FILEname:=OUTPUTfolder & "\temp" & "\
    sectionplot" & DUM & ".bmp", FilterName:="BMP")

        'load the picture into the picture box
        plotSection.BackgroundImage = New System.Drawing.Bitmap(OUTPUTfolder & "\temp" &
    "\sectionplot" & DUM & ".bmp")
        plotSection.BackgroundImageLayout = ImageLayout.Stretch


    End Sub



    '(7)PREVIEW_FIRE
    Public Sub btnThermalEnvironment_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles btnThermalEnvironment.Click
        Me.tabctrlMain.SelectTab(6)
    End Sub



    'SYSTEM_SUBROUTINES+FUNCTIONS
    Private Sub releaseObject(ByVal obj As Object)
        Try
            System.Runtime.InteropServices.Marshal.ReleaseComObject(obj)
            obj = Nothing
        Catch ex As Exception
            obj = Nothing
        Finally
            GC.Collect()
        End Try
    End Sub
    Public Function GetDirectoryName(ByVal FullPath As String) As String
        Return System.IO.Path.GetDirectoryName(FullPath)
    End Function
    Public Function GetFileNameWithoutExtension(ByVal FileOnly As String) As String
        Return System.IO.Path.GetFileNameWithoutExtension(FileOnly)
    End Function



End Class
```

```
Imports System.IO
Imports Microsoft.VisualBasic
Imports MSChart20Lib
Imports Excel = Microsoft.Office.Interop.Excel
Imports System.Math

Public Class Results

    Dim FilePath, FolderPath As String
    Dim timesteps, DUM, mphiCOUNT, mphiPLOTpoints As Integer
    Dim Rho, areaS, areaG, areaN As Single

    Dim xlApp As Excel.Application
    Dim xlUTFire As Excel.Workbook
    Dim misValue As Object = System.Reflection.Missing.Value
    Dim xlOUT, xlREOUT, xlFire, xlSection, xlConcrete, xlSteel, xlMPhi, xlScratch, xlMN, ↙
    xlDOOP As Excel.Worksheet
    Dim x_col(0 To 1000), y_col(0 To 1000) As String
    Dim xSS_col(0 To 1000), ySS_col(0 To 1000) As String

    Dim MPhi_Index(0 To 100000), dontRefresh, dontRefreshSS, DUMss, INITIAL, SStempCOUNT,↙
     SSplotCOUNT As Integer
    Dim maxM, minM, maxN, minN, maxM_t, minM_t, maxN_t, minN_t As Single

    Dim MN_Index(0 To 100000), dontRefreshMN, DUMmn, mnPLOTpoints, mnCOUNT As Integer

    Dim inM, inN, inR, inAVG As Integer

    'for stress-strain plots
    Dim fp, fy, Est As Single
    Dim ep, ey, et, eu As Single
    Dim a, b, c As Single
    Dim fprimec, eo, ecu, fct, ect, Ecc, ectu As Single
    Dim aggregates As String
    Dim temp_TEMP(0 To 13) As Integer 'array(temp_range)
    Dim ec_TEMP(0 To 13, 0 To 1000), fc_TEMP(0 To 13, 0 To 1000) As Single 'array        ↙
    (temp_range, plot_point)
    Dim es_TEMP(0 To 13, 0 To 1000), fs_TEMP(0 To 13, 0 To 1000) As Single 'array        ↙
    (temp_range, plot_point)

    Dim fprimec_TEMP(0 To 13), eo_TEMP(0 To 13), ecu_TEMP(0 To 13) As Single 'array      ↙
    (temp_range)
    Dim fct_TEMP(0 To 13), ect_TEMP(0 To 13), Ecc_TEMP(0 To 13), ectu_TEMP(0 To 13) As   ↙
    Single 'array(temp_range)
    Dim fp_TEMP(0 To 13), fy_TEMP(0 To 13), Est_TEMP(0 To 13) As Single 'array           ↙
    (temp_range)
    Dim ep_TEMP(0 To 13), ey_TEMP(0 To 13), et_TEMP(0 To 13), eu_TEMP(0 To 13) As Single ↙
    'array(temp_range)
    Dim a_TEMP(0 To 13), b_TEMP(0 To 13), c_TEMP(0 To 13) As Single 'array(temp_range)


    'LOAD_EVENTS
    Public Sub New(ByVal currentFile_Path As String, ByVal currentFolder_Path As String)
        InitializeComponent()
        txtFilePath.Text = currentFile_Path
        txtOutputFolder.Text = currentFolder_Path
        FilePath = txtFilePath.Text
        FolderPath = txtOutputFolder.Text

        Me.Activate()
        Me.Visible = True

        prgLOAD.Value = 0

        Dim AllProcesses() As Process = Process.GetProcessesByName("excel")
        Dim ExcelProcess As Process
```

```
        For Each ExcelProcess In AllProcesses
            ExcelProcess.Kill()
        Next
        ExcelProcess = Nothing
        AllProcesses = Nothing

        xlApp = New Excel.ApplicationClass
        xlUTFire = xlApp.Workbooks.Add(currentFolder_Path & "\temp\temp.xlsx")
        xlOUT = xlUTFire.Sheets(".out")
        xlREOUT = xlUTFire.Sheets("re.out")
        xlFire = xlUTFire.Sheets("fire")
        xlSection = xlUTFire.Sheets("section")
        xlConcrete = xlUTFire.Sheets("concrete")
        xlSteel = xlUTFire.Sheets("steel")
        xlMPhi = xlUTFire.Sheets("MPhi")
        xlScratch = xlUTFire.Sheets("scratch")
        xlMN = xlUTFire.Sheets("MN")
        xlDOOP = xlUTFire.Sheets("doop")


        prgLOAD.Value = 250



        'Load Analysis Summary Data
        prgLOAD.Value = 500
        timesteps = xlREOUT.Cells(9, 2).value

        For i = 1 To xlREOUT.Cells(23, 2).value
            areaS = areaS + xlREOUT.Cells(i + 1, 52).value
        Next i
        areaS = Round(areaS, 2)
        For i = 1 To xlREOUT.Cells(4, 2).value
            areaG = areaG + xlREOUT.Cells(i + 1, 29).value
        Next i
        areaG = Round(areaG, 2)
        areaN = Round(areaG - areaS, 2)
        Rho = Round(areaS / areaG * 100, 2)

        maxM = 0
        minM = 0
        maxN = 0
        minM = 0

        For i = 1 To timesteps + 1
            If xlScratch.Cells(i, 35).value > maxM Then
                maxM = Round(xlScratch.Cells(i, 35).value, 2)
                maxM_t = Round(xlScratch.Cells(i, 34).value, 2)
            End If
            If xlScratch.Cells(i, 36).value > minM Then
                minM = Round(xlScratch.Cells(i, 36).value, 2)
                minM_t = Round(xlScratch.Cells(i, 34).value, 2)
            End If
        Next i

        maxN = xlScratch.Cells(1, 106).Value
        maxN_t = 0
        minN = xlScratch.Cells(1, 107).Value
        minN_t = 0

        prgLOAD.Value = 750

        fct = Round(0.6228 * (xlREOUT.Cells(11, 2).value) ^ 0.5, 2)

        txtSectionName.Text = xlScratch.Cells(3, 1).value
        txtNodeCount.Text = xlREOUT.Cells(3, 2).value & " nodes"
```

```
        txtElementCount.Text = xlREOUT.Cells(4, 2).value & " elements"
        txtFireFileUsed.Text = xlScratch.Cells(4, 1).value
        txtCalcTimeStep.Text = xlREOUT.Cells(7, 2).value & " seconds"
        txtCalcEndTime.Text = xlREOUT.Cells(8, 2).value & " seconds"
        txtCalcStepCount.Text = xlREOUT.Cells(9, 2).value & " steps"
        txtFireStepCount.Text = xlScratch.Cells(9, 1).value & " steps"
        txtSectionHeightMax.Text = xlSection.Cells(2, 3).value & " mm"
        txtSectionHeightMin.Text = xlSection.Cells(2, 3).value & " mm"
        txtSectionWidthMax.Text = xlSection.Cells(1, 3).value & " mm"
        txtSectionWidthMin.Text = xlSection.Cells(1, 3).value & " mm"
        txtRebarCount.Text = xlREOUT.Cells(23, 2).value & " bars"
        txtTotalSteelArea.Text = areaS & " mm^2"
        txtGrossSectionArea.Text = areaG & " mm^2"
        txtNetSectionArea.Text = areaN & " mm^2"
        txtReinforcementRatio.Text = Rho & "%"
        txtSteelProportionalStressLimit.Text = xlREOUT.Cells(15, 2).value & " MPa"
        txtSteelMaximumStressLevel.Text = xlREOUT.Cells(16, 2).value & " MPa"
        txtConcreteCompressiveStrength.Text = xlREOUT.Cells(11, 2).value & " MPa"
        txtConcreteTensileStrength.Text = fct & " MPa"
        txtConcreteAggregatesUsed.Text = xlREOUT.Cells(12, 2).value & " Aggregates"
        txtMaximumAxialCapacity.Text = maxN & " kN"
        txtMaximumAxialCapacity_t.Text = maxN_t & " hours"
        txtMinimumAxialCapacity.Text = minN & " kN"
        txtMinimumAxialCapacity_t.Text = minN_t & " hours"
        txtMaximumMomentCapacity.Text = maxM & " kN-m"
        txtMaximumMomentCapacity_t.Text = maxM_t & " hours"
        txtMinimumMomentCapacity.Text = minM & " kN-m"
        txtMinimumMomentCapacity_t.Text = minM_t & " hours"



        'Load Section Plot
        xlSection.ChartObjects(1).chart.Export(FILEname:=FolderPath & "\temp" & "\
    results_section.bmp", FilterName:="BMP")
        plotSECTION.BackgroundImage = New System.Drawing.Bitmap(FolderPath & "\temp" & "\
    results_section.bmp")
        plotSECTION.BackgroundImageLayout = ImageLayout.Stretch

        For i = 0 To xlREOUT.Cells(4, 2).value
            lstElement_Sect.Items.Add(i)
            lstElement_Sect.Items.Item(i) = i
        Next i

        For i = 0 To xlREOUT.Cells(23, 2).value
            lstRebar_Sect.Items.Add(i)
            lstRebar_Sect.Items.Item(i) = i
        Next i

        inM = 1
        inN = 2
        inR = 3
        inAVG = 4


        'Load Stress-Strain
        '**CONCRETE**********************************************************************
    ***************
            prgLOAD.Value = 1000
            With xlConcrete
                .Cells.Clear()
                .Cells.HorizontalAlignment = Excel.XlVAlign.xlVAlignCenter
                .Columns(1).ColumnWidth = 12
                .Columns(1).HorizontalAlignment = Excel.XlHAlign.xlHAlignRight
                .Rows(1).Font.Bold = True
            End With
```

```
'Initial
fprimec = xlREOUT.Cells(11, 2).value
aggregates = xlREOUT.Cells(12, 2).value
eo = 0.0025
ecu = 0.02
fct = -0.6228 * (fprimec) ^ 0.5 '= 7.5*(f'c)^0.5 in psi
Ecc = 4732.98 * (fprimec) ^ 0.5 '= 57000*(f'c)^0.5 in psi
ect = fct / Ecc
ectu = 6 * fct / Ecc

xlConcrete.Cells(1, 1).value = "CONCRETE"
xlConcrete.Cells(2, 1).value = "f'c:"
xlConcrete.Cells(3, 1).value = "Aggs:"
xlConcrete.Cells(4, 1).value = "eo:"
xlConcrete.Cells(5, 1).value = "ecu:"
xlConcrete.Cells(6, 1).value = "f'ct:"
xlConcrete.Cells(7, 1).value = "Ec:"
xlConcrete.Cells(8, 1).value = "e'ct:"
xlConcrete.Cells(9, 1).value = "ectu:"

xlConcrete.Cells(2, 2).value = fprimec
xlConcrete.Cells(3, 2).value = aggregates
xlConcrete.Cells(4, 2).value = eo
xlConcrete.Cells(5, 2).value = ecu
xlConcrete.Cells(6, 2).value = fct
xlConcrete.Cells(7, 2).value = Ecc
xlConcrete.Cells(8, 2).value = ect
xlConcrete.Cells(9, 2).value = ectu



'Heated
temp_TEMP(0) = 20
temp_TEMP(1) = 100
temp_TEMP(2) = 200
temp_TEMP(3) = 300
temp_TEMP(4) = 400
temp_TEMP(5) = 500
temp_TEMP(6) = 600
temp_TEMP(7) = 700
temp_TEMP(8) = 800
temp_TEMP(9) = 900
temp_TEMP(10) = 1000
temp_TEMP(11) = 1100
temp_TEMP(12) = 1200

x_col(0) = "E"
y_col(0) = "F"
x_col(1) = "I"
y_col(1) = "J"
x_col(2) = "M"
y_col(2) = "N"
x_col(3) = "Q"
y_col(3) = "R"
x_col(4) = "U"
y_col(4) = "V"
x_col(5) = "Y"
y_col(5) = "Z"
x_col(6) = "AC"
y_col(6) = "AD"
x_col(7) = "AG"
y_col(7) = "AH"
x_col(8) = "AK"
y_col(8) = "AL"
x_col(9) = "AO"
y_col(9) = "AP"
```

```
            x_col(10) = "AS"
            y_col(10) = "AT"
            x_col(11) = "AW"
            y_col(11) = "AX"
            x_col(12) = "BA"
            y_col(12) = "BB"

            If aggregates = "Siliceous" Then GoTo 100
            If aggregates = "Calcareous" Then GoTo 101

100:        For i = 0 To 12

                If temp_TEMP(i) = 20 Then fprimec_TEMP(i) = 1.0 * fprimec
                If temp_TEMP(i) = 20 Then eo_TEMP(i) = 0.0025
                If temp_TEMP(i) = 20 Then ecu_TEMP(i) = 0.02
                If temp_TEMP(i) = 20 Then fct_TEMP(i) = 1.0 * fct

                If temp_TEMP(i) = 100 Then fprimec_TEMP(i) = 1.0 * fprimec
                If temp_TEMP(i) = 100 Then eo_TEMP(i) = 0.004
                If temp_TEMP(i) = 100 Then ecu_TEMP(i) = 0.0225
                If temp_TEMP(i) = 100 Then fct_TEMP(i) = 1.0 * fct

                If temp_TEMP(i) = 200 Then fprimec_TEMP(i) = 0.95 * fprimec
                If temp_TEMP(i) = 200 Then eo_TEMP(i) = 0.0055
                If temp_TEMP(i) = 200 Then ecu_TEMP(i) = 0.025
                If temp_TEMP(i) = 200 Then fct_TEMP(i) = 0.8 * fct

                If temp_TEMP(i) = 300 Then fprimec_TEMP(i) = 0.85 * fprimec
                If temp_TEMP(i) = 300 Then eo_TEMP(i) = 0.007
                If temp_TEMP(i) = 300 Then ecu_TEMP(i) = 0.0275
                If temp_TEMP(i) = 300 Then fct_TEMP(i) = 0.6 * fct

                If temp_TEMP(i) = 400 Then fprimec_TEMP(i) = 0.75 * fprimec
                If temp_TEMP(i) = 400 Then eo_TEMP(i) = 0.01
                If temp_TEMP(i) = 400 Then ecu_TEMP(i) = 0.03
                If temp_TEMP(i) = 400 Then fct_TEMP(i) = 0.4 * fct

                If temp_TEMP(i) = 500 Then fprimec_TEMP(i) = 0.6 * fprimec
                If temp_TEMP(i) = 500 Then eo_TEMP(i) = 0.015
                If temp_TEMP(i) = 500 Then ecu_TEMP(i) = 0.0325
                If temp_TEMP(i) = 500 Then fct_TEMP(i) = 0.2 * fct

                If temp_TEMP(i) = 600 Then fprimec_TEMP(i) = 0.45 * fprimec
                If temp_TEMP(i) = 600 Then eo_TEMP(i) = 0.025
                If temp_TEMP(i) = 600 Then ecu_TEMP(i) = 0.035
                If temp_TEMP(i) = 600 Then fct_TEMP(i) = 0 * fct

                If temp_TEMP(i) = 700 Then fprimec_TEMP(i) = 0.3 * fprimec
                If temp_TEMP(i) = 700 Then eo_TEMP(i) = 0.025
                If temp_TEMP(i) = 700 Then ecu_TEMP(i) = 0.0375
                If temp_TEMP(i) = 700 Then fct_TEMP(i) = 0 * fct

                If temp_TEMP(i) = 800 Then fprimec_TEMP(i) = 0.15 * fprimec
                If temp_TEMP(i) = 800 Then eo_TEMP(i) = 0.025
                If temp_TEMP(i) = 800 Then ecu_TEMP(i) = 0.04
                If temp_TEMP(i) = 800 Then fct_TEMP(i) = 0 * fct

                If temp_TEMP(i) = 900 Then fprimec_TEMP(i) = 0.08 * fprimec
                If temp_TEMP(i) = 900 Then eo_TEMP(i) = 0.025
                If temp_TEMP(i) = 900 Then ecu_TEMP(i) = 0.0425
                If temp_TEMP(i) = 900 Then fct_TEMP(i) = 0 * fct

                If temp_TEMP(i) = 1000 Then fprimec_TEMP(i) = 0.04 * fprimec
                If temp_TEMP(i) = 1000 Then eo_TEMP(i) = 0.025
                If temp_TEMP(i) = 1000 Then ecu_TEMP(i) = 0.045
                If temp_TEMP(i) = 1000 Then fct_TEMP(i) = 0 * fct
```

```
                If temp_TEMP(i) = 1100 Then fprimec_TEMP(i) = 0.01 * fprimec
                If temp_TEMP(i) = 1100 Then eo_TEMP(i) = 0.025
                If temp_TEMP(i) = 1100 Then ecu_TEMP(i) = 0.0475
                If temp_TEMP(i) = 1100 Then fct_TEMP(i) = 0 * fct

                If temp_TEMP(i) = 1200 Then fprimec_TEMP(i) = 0.001 * fprimec
                If temp_TEMP(i) = 1200 Then eo_TEMP(i) = 0.025
                If temp_TEMP(i) = 1200 Then ecu_TEMP(i) = 0.05
                If temp_TEMP(i) = 1200 Then fct_TEMP(i) = 0 * fct

                If fct_TEMP(i) = 0 Then
                    Ecc_TEMP(i) = 0
                    ect_TEMP(i) = 0
                    ectu_TEMP(i) = 0
                    GoTo 1001
                End If
                Ecc_TEMP(i) = 4732.98 * (fprimec_TEMP(i)) ^ 0.5
                ect_TEMP(i) = fct_TEMP(i) / Ecc_TEMP(i)
                ectu_TEMP(i) = 11 * fct_TEMP(i) / Ecc_TEMP(i)

1001:       Next i
            GoTo 102

101:        For i = 0 To 12

                If temp_TEMP(i) = 20 Then fprimec_TEMP(i) = 1.0 * fprimec
                If temp_TEMP(i) = 20 Then eo_TEMP(i) = 0.0025
                If temp_TEMP(i) = 20 Then ecu_TEMP(i) = 0.02
                If temp_TEMP(i) = 20 Then fct_TEMP(i) = 1.0 * fct

                If temp_TEMP(i) = 100 Then fprimec_TEMP(i) = 1.0 * fprimec
                If temp_TEMP(i) = 100 Then eo_TEMP(i) = 0.004
                If temp_TEMP(i) = 100 Then ecu_TEMP(i) = 0.0225
                If temp_TEMP(i) = 100 Then fct_TEMP(i) = 1.0 * fct

                If temp_TEMP(i) = 200 Then fprimec_TEMP(i) = 0.97 * fprimec
                If temp_TEMP(i) = 200 Then eo_TEMP(i) = 0.0055
                If temp_TEMP(i) = 200 Then ecu_TEMP(i) = 0.025
                If temp_TEMP(i) = 200 Then fct_TEMP(i) = 0.8 * fct

                If temp_TEMP(i) = 300 Then fprimec_TEMP(i) = 0.91 * fprimec
                If temp_TEMP(i) = 300 Then eo_TEMP(i) = 0.007
                If temp_TEMP(i) = 300 Then ecu_TEMP(i) = 0.0275
                If temp_TEMP(i) = 300 Then fct_TEMP(i) = 0.6 * fct

                If temp_TEMP(i) = 400 Then fprimec_TEMP(i) = 0.85 * fprimec
                If temp_TEMP(i) = 400 Then eo_TEMP(i) = 0.01
                If temp_TEMP(i) = 400 Then ecu_TEMP(i) = 0.03
                If temp_TEMP(i) = 400 Then fct_TEMP(i) = 0.4 * fct

                If temp_TEMP(i) = 500 Then fprimec_TEMP(i) = 0.74 * fprimec
                If temp_TEMP(i) = 500 Then eo_TEMP(i) = 0.015
                If temp_TEMP(i) = 500 Then ecu_TEMP(i) = 0.0325
                If temp_TEMP(i) = 500 Then fct_TEMP(i) = 0.2 * fct

                If temp_TEMP(i) = 600 Then fprimec_TEMP(i) = 0.6 * fprimec
                If temp_TEMP(i) = 600 Then eo_TEMP(i) = 0.025
                If temp_TEMP(i) = 600 Then ecu_TEMP(i) = 0.035
                If temp_TEMP(i) = 600 Then fct_TEMP(i) = 0 * fct

                If temp_TEMP(i) = 700 Then fprimec_TEMP(i) = 0.43 * fprimec
                If temp_TEMP(i) = 700 Then eo_TEMP(i) = 0.025
                If temp_TEMP(i) = 700 Then ecu_TEMP(i) = 0.0375
                If temp_TEMP(i) = 700 Then fct_TEMP(i) = 0 * fct
```

```
                If temp_TEMP(i) = 800 Then fprimec_TEMP(i) = 0.27 * fprimec
                If temp_TEMP(i) = 800 Then eo_TEMP(i) = 0.025
                If temp_TEMP(i) = 800 Then ecu_TEMP(i) = 0.04
                If temp_TEMP(i) = 800 Then fct_TEMP(i) = 0 * fct

                If temp_TEMP(i) = 900 Then fprimec_TEMP(i) = 0.15 * fprimec
                If temp_TEMP(i) = 900 Then eo_TEMP(i) = 0.025
                If temp_TEMP(i) = 900 Then ecu_TEMP(i) = 0.0425
                If temp_TEMP(i) = 900 Then fct_TEMP(i) = 0 * fct

                If temp_TEMP(i) = 1000 Then fprimec_TEMP(i) = 0.06 * fprimec
                If temp_TEMP(i) = 1000 Then eo_TEMP(i) = 0.025
                If temp_TEMP(i) = 1000 Then ecu_TEMP(i) = 0.045
                If temp_TEMP(i) = 1000 Then fct_TEMP(i) = 0 * fct

                If temp_TEMP(i) = 1100 Then fprimec_TEMP(i) = 0.02 * fprimec
                If temp_TEMP(i) = 1100 Then eo_TEMP(i) = 0.025
                If temp_TEMP(i) = 1100 Then ecu_TEMP(i) = 0.0475
                If temp_TEMP(i) = 1100 Then fct_TEMP(i) = 0 * fct

                If temp_TEMP(i) = 1200 Then fprimec_TEMP(i) = 0.001 * fprimec
                If temp_TEMP(i) = 1200 Then eo_TEMP(i) = 0.025
                If temp_TEMP(i) = 1200 Then ecu_TEMP(i) = 0.05
                If temp_TEMP(i) = 1200 Then fct_TEMP(i) = 0 * fct

                If fct_TEMP(i) = 0 Then
                    Ecc_TEMP(i) = 0
                    ect_TEMP(i) = 0
                    ectu_TEMP(i) = 0
                    GoTo 1011
                End If
                Ecc_TEMP(i) = 4732.98 * (fprimec_TEMP(i)) ^ 0.5
                ect_TEMP(i) = fct_TEMP(i) / Ecc_TEMP(i)
                ectu_TEMP(i) = 11 * fct_TEMP(i) / Ecc_TEMP(i)

1011:       Next i

102:        With xlConcrete
                For i = 0 To 12

                    .Cells(1, 4 * (i + 1)).value = temp_TEMP(i)
                    .Cells(1, 4 * (i + 1) + 1).value = "ecT"
                    ec_TEMP(i, 1) = ectu_TEMP(i)
                    .Cells(2, 4 * (i + 1) + 1).value = ec_TEMP(i, 1)
                    .Cells(1, 4 * (i + 1) + 2).value = "fcT"
                    fc_TEMP(i, 1) = 0
                    .Cells(2, 4 * (i + 1) + 2).value = fc_TEMP(i, 1)

                    For j = 2 To 100
                        ec_TEMP(i, j) = ec_TEMP(i, j - 1) + (ecu_TEMP(i) - ectu_TEMP(i)) ↙
    / 99
                        .Cells(j + 1, 4 * (i + 1) + 1).value = ec_TEMP(i, j)
                        If ec_TEMP(i, j) > 0 Then GoTo 1021
                        If ec_TEMP(i, j) < ectu_TEMP(i) Then fc_TEMP(i, j) = 0
                        If ectu_TEMP(i) < ec_TEMP(i, j) And ec_TEMP(i, j) < ect_TEMP(i)  ↙
    Then fc_TEMP(i, j) = Ecc_TEMP(i) * ectu_TEMP(i) / 10 - Ecc_TEMP(i) * ec_TEMP(i, j) / ↙
    10
                        If ect_TEMP(i) < ec_TEMP(i, j) And ec_TEMP(i, j) < 0 Then fc_TEMP↙
    (i, j) = Ecc_TEMP(i) * ec_TEMP(i, j)
                        GoTo 1022
1021:                   If ec_TEMP(i, j) < eo_TEMP(i) Then fc_TEMP(i, j) = (3 * ec_TEMP(i↙
    , j) * fprimec_TEMP(i)) / (eo_TEMP(i) * (2 + (ec_TEMP(i, j) / eo_TEMP(i)) ^ 3))
                        If ec_TEMP(i, j) = eo_TEMP(i) Then fc_TEMP(i, j) = fprimec_TEMP  ↙
    (i)
                        If ec_TEMP(i, j) > eo_TEMP(i) Then fc_TEMP(i, j) = fprimec_TEMP  ↙
    (i) - ((-fprimec_TEMP(i)) / (ecu_TEMP(i) - eo_TEMP(i))) * eo_TEMP(i) + ((-           ↙
```

```
    fprimec_TEMP(i)) / (ecu_TEMP(i) - eo_TEMP(i))) * ec_TEMP(i, j)
1022:                   .Cells(j + 1, 4 * (i + 1) + 2).value = fc_TEMP(i, j)
                    Next j

                    .Columns(4 * (i + 1) + 1).NumberFormat = "0.00000"
                    .Columns(4 * (i + 1) + 2).NumberFormat = "0.00"

                    .Cells(2, 4 * (i + 1)).value = fprimec_TEMP(i)
                    .Cells(3, 4 * (i + 1)).value = eo_TEMP(i)
                    .Cells(4, 4 * (i + 1)).value = ecu_TEMP(i)
                    .Cells(5, 4 * (i + 1)).value = fct_TEMP(i)
                    .Cells(6, 4 * (i + 1)).value = Ecc_TEMP(i)
                    .Cells(7, 4 * (i + 1)).value = ect_TEMP(i)
                    .Cells(8, 4 * (i + 1)).value = ectu_TEMP(i)

                Next i
            End With

            'Plot concrete stress-strain_____________________________________________
            If xlScratch.Cells(7, 1).value = "FALSE" Then GoTo 201
            GoTo 202
201:        xlConcrete.ChartObjects.Delete()

202:        xlScratch.Cells(7, 1).value = "FALSE"
            Dim concrete_plot As Excel.Chart
            Dim xlCharts As Excel.ChartObjects
            Dim myChart As Excel.ChartObject
            Dim concrete_SerCol As Excel.SeriesCollection
            Dim concrete_Series As Excel.Series
            Dim concrete_AxisCategory, concrete_AxisValue As Excel.Axes

            xlCharts = xlConcrete.ChartObjects
            myChart = xlCharts.Add(180, 260, 1000, 600)
            concrete_plot = myChart.Chart
            concrete_plot.ChartType = Excel.XlChartType.xlXYScatterLinesNoMarkers
            concrete_plot.ChartStyle = 3
            concrete_SerCol = concrete_plot.SeriesCollection

            concrete_Series = concrete_SerCol.NewSeries
            concrete_Series.Name = temp_TEMP(0) & "C"
            concrete_Series.XValues = xlConcrete.Range("E2:E101")
            concrete_Series.Values = xlConcrete.Range("F2:F101")
            concrete_Series.Format.Line.Weight = 1.5

            With concrete_plot
                .Legend.Position = Excel.XlLegendPosition.xlLegendPositionTop
                .HasTitle = True
                .ChartTitle.Text = "Concrete Stress-Strain Relationship"

                concrete_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
                concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = True
                concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.
    Characters.Text() = "Strain, mm/mm"
                concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.
    Characters.Font.Bold = False
                concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.
    Characters.Font.Size = 14
                concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines
     = True
                concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines
     = True
                concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.
    Format.Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
                concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).
    MinimumScaleIsAuto = True
                concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).
```

```
    MaximumScaleIsAuto = True
                concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MajorUnitIsAuto =↙
     True
                concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition↙
     = Excel.XlTickLabelPosition.xlTickLabelPositionLow
                concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.      ↙
    NumberFormat = "0.000"
                concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font. ↙
    Bold = False
                concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font. ↙
    Size = 14

                concrete_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
                concrete_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = True
                concrete_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.  ↙
    Text() = "Stress, MPa"
                concrete_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.  ↙
    Font.Bold = False
                concrete_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.  ↙
    Font.Size = 14
                concrete_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines =    ↙
    True
                concrete_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines =    ↙
    True
                concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format. ↙
    Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
                concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScaleIsAuto =   ↙
    True
                concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MaximumScaleIsAuto =   ↙
    True
                concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MajorUnitIsAuto = True
                concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition =    ↙
    Excel.XlTickLabelPosition.xlTickLabelPositionLow
                concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.NumberFormat↙
     = "0"
                concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold = ↙
    False
                concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = ↙
    14
            End With
            concrete_plot.Refresh()

            '**STEEL*************************************************************************↙
    ***************
            prgLOAD.Value = 5000
            With xlSteel
                .Cells.Clear()
                .Cells.HorizontalAlignment = Excel.XlVAlign.xlVAlignCenter
                .Columns(1).ColumnWidth = 12
                .Columns(1).HorizontalAlignment = Excel.XlHAlign.xlHAlignRight
                .Rows(1).Font.Bold = True
            End With

            'Initial
            fp = xlREOUT.Cells(15, 2).value
            fy = xlREOUT.Cells(16, 2).value
            Est = 199948
            ep = fp / Est
            ey = 0.02
            et = 0.15
            eu = 0.2
            c = ((fy - fp) ^ 2) / ((ey - ep) * Est - 2 * (fy - fp))
            a = ((ey - ep) * (ey - ep + c / Est)) ^ 0.5
            b = (c * (ey - ep) * Est + c ^ 2) ^ 0.5

            xlSteel.Cells(1, 1).value = "STEEL"
```

```
            xlSteel.Cells(2, 1).value = "fp:"
            xlSteel.Cells(3, 1).value = "fy:"
            xlSteel.Cells(4, 1).value = "Est:"
            xlSteel.Cells(5, 1).value = "ep:"
            xlSteel.Cells(6, 1).value = "ey:"
            xlSteel.Cells(7, 1).value = "et:"
            xlSteel.Cells(8, 1).value = "eu:"
            xlSteel.Cells(9, 1).value = "a:"
            xlSteel.Cells(10, 1).value = "b:"
            xlSteel.Cells(11, 1).value = "c:"

            xlSteel.Cells(2, 2).value = fp
            xlSteel.Cells(3, 2).value = fy
            xlSteel.Cells(4, 2).value = Est
            xlSteel.Cells(5, 2).value = ep
            xlSteel.Cells(6, 2).value = ey
            xlSteel.Cells(7, 2).value = et
            xlSteel.Cells(8, 2).value = eu
            xlSteel.Cells(9, 2).value = a
            xlSteel.Cells(10, 2).value = b
            xlSteel.Cells(11, 2).value = c

            'Heated
            temp_TEMP(0) = 20
            temp_TEMP(1) = 100
            temp_TEMP(2) = 200
            temp_TEMP(3) = 300
            temp_TEMP(4) = 400
            temp_TEMP(5) = 500
            temp_TEMP(6) = 600
            temp_TEMP(7) = 700
            temp_TEMP(8) = 800
            temp_TEMP(9) = 900
            temp_TEMP(10) = 1000
            temp_TEMP(11) = 1100
            temp_TEMP(12) = 1200

            xSS_col(0) = "E"
            ySS_col(0) = "F"
            xSS_col(1) = "I"
            ySS_col(1) = "J"
            xSS_col(2) = "M"
            ySS_col(2) = "N"
            xSS_col(3) = "Q"
            ySS_col(3) = "R"
            xSS_col(4) = "U"
            ySS_col(4) = "V"
            xSS_col(5) = "Y"
            ySS_col(5) = "Z"
            xSS_col(6) = "AC"
            ySS_col(6) = "AD"
            xSS_col(7) = "AG"
            ySS_col(7) = "AH"
            xSS_col(8) = "AK"
            ySS_col(8) = "AL"
            xSS_col(9) = "AO"
            ySS_col(9) = "AP"
            xSS_col(10) = "AS"
            ySS_col(10) = "AT"
            xSS_col(11) = "AW"
            ySS_col(11) = "AX"
            xSS_col(12) = "BA"
            ySS_col(12) = "BB"

            For i = 0 To 12
```

```
            prgLOAD.Value = 5000 + Round(i * 1000 / 12, 0)

            If temp_TEMP(i) = 20 Then fp_TEMP(i) = 1.0 * fp
            If temp_TEMP(i) = 20 Then fy_TEMP(i) = 1.0 * fy
            If temp_TEMP(i) = 20 Then Est_TEMP(i) = 1.0 * Est

            If temp_TEMP(i) = 100 Then fp_TEMP(i) = 1.0 * fp
            If temp_TEMP(i) = 100 Then fy_TEMP(i) = 1.0 * fy
            If temp_TEMP(i) = 100 Then Est_TEMP(i) = 1.0 * Est

            If temp_TEMP(i) = 200 Then fp_TEMP(i) = 0.81 * fp
            If temp_TEMP(i) = 200 Then fy_TEMP(i) = 1.0 * fy
            If temp_TEMP(i) = 200 Then Est_TEMP(i) = 0.9 * Est

            If temp_TEMP(i) = 300 Then fp_TEMP(i) = 0.61 * fp
            If temp_TEMP(i) = 300 Then fy_TEMP(i) = 1.0 * fy
            If temp_TEMP(i) = 300 Then Est_TEMP(i) = 0.8 * Est

            If temp_TEMP(i) = 400 Then fp_TEMP(i) = 0.42 * fp
            If temp_TEMP(i) = 400 Then fy_TEMP(i) = 1.0 * fy
            If temp_TEMP(i) = 400 Then Est_TEMP(i) = 0.7 * Est

            If temp_TEMP(i) = 500 Then fp_TEMP(i) = 0.36 * fp
            If temp_TEMP(i) = 500 Then fy_TEMP(i) = 0.78 * fy
            If temp_TEMP(i) = 500 Then Est_TEMP(i) = 0.6 * Est

            If temp_TEMP(i) = 600 Then fp_TEMP(i) = 0.18 * fp
            If temp_TEMP(i) = 600 Then fy_TEMP(i) = 0.47 * fy
            If temp_TEMP(i) = 600 Then Est_TEMP(i) = 0.31 * Est

            If temp_TEMP(i) = 700 Then fp_TEMP(i) = 0.07 * fp
            If temp_TEMP(i) = 700 Then fy_TEMP(i) = 0.23 * fy
            If temp_TEMP(i) = 700 Then Est_TEMP(i) = 0.13 * Est

            If temp_TEMP(i) = 800 Then fp_TEMP(i) = 0.05 * fp
            If temp_TEMP(i) = 800 Then fy_TEMP(i) = 0.11 * fy
            If temp_TEMP(i) = 800 Then Est_TEMP(i) = 0.09 * Est

            If temp_TEMP(i) = 900 Then fp_TEMP(i) = 0.04 * fp
            If temp_TEMP(i) = 900 Then fy_TEMP(i) = 0.06 * fy
            If temp_TEMP(i) = 900 Then Est_TEMP(i) = 0.07 * Est

            If temp_TEMP(i) = 1000 Then fp_TEMP(i) = 0.02 * fp
            If temp_TEMP(i) = 1000 Then fy_TEMP(i) = 0.04 * fy
            If temp_TEMP(i) = 1000 Then Est_TEMP(i) = 0.04 * Est

            If temp_TEMP(i) = 1100 Then fp_TEMP(i) = 0.01 * fp
            If temp_TEMP(i) = 1100 Then fy_TEMP(i) = 0.02 * fy
            If temp_TEMP(i) = 1100 Then Est_TEMP(i) = 0.02 * Est

            If temp_TEMP(i) = 1200 Then fp_TEMP(i) = 0.001 * fp
            If temp_TEMP(i) = 1200 Then fy_TEMP(i) = 0.001 * fy
            If temp_TEMP(i) = 1200 Then Est_TEMP(i) = 0.001 * Est

            ep_TEMP(i) = fp_TEMP(i) / Est_TEMP(i)
            ey_TEMP(i) = 0.02
            et_TEMP(i) = 0.15
            eu_TEMP(i) = 0.2
            c_TEMP(i) = ((fy_TEMP(i) - fp_TEMP(i)) ^ 2) / ((ey_TEMP(i) - ep_TEMP(i)) ↙
* Est_TEMP(i) - 2 * (fy_TEMP(i) - fp_TEMP(i)))
            a_TEMP(i) = ((ey_TEMP(i) - ep_TEMP(i)) * (ey_TEMP(i) - ep_TEMP(i) +      ↙
c_TEMP(i) / Est_TEMP(i))) ^ 0.5
            b_TEMP(i) = (c_TEMP(i) * (ey_TEMP(i) - ep_TEMP(i)) * Est_TEMP(i) + c_TEMP↙
(i) ^ 2) ^ 0.5

        Next i
```

```
            With xlSteel
                For i = 0 To 12

                    prgLOAD.Value = 6000 + Round(i * 2000 / 12, 0)

                    .Cells(1, 4 * (i + 1)).value = temp_TEMP(i)
                    .Cells(1, 4 * (i + 1) + 1).value = "esT"
                    es_TEMP(i, 1) = 0
                    .Cells(2, 4 * (i + 1) + 1).value = es_TEMP(i, 1)
                    .Cells(1, 4 * (i + 1) + 2).value = "fsT"
                    fs_TEMP(i, 1) = 0
                    .Cells(2, 4 * (i + 1) + 2).value = fs_TEMP(i, 1)

                    For j = 2 To 100
                        es_TEMP(i, j) = es_TEMP(i, j - 1) + eu_TEMP(i) / 99
                        .Cells(j + 1, 4 * (i + 1) + 1).value = es_TEMP(i, j)
                        If es_TEMP(i, j) < ep_TEMP(i) Then fs_TEMP(i, j) = es_TEMP(i, j) ↙
    * Est_TEMP(i)
                        If ep_TEMP(i) < es_TEMP(i, j) And es_TEMP(i, j) < ey_TEMP(i) Then↙
     fs_TEMP(i, j) = fp_TEMP(i) - c_TEMP(i) + (b_TEMP(i) / a_TEMP(i)) * ((a_TEMP(i) ^ 2) ↙
    - ((ey_TEMP(i) - es_TEMP(i, j)) ^ 2)) ^ 0.5
                        If ey_TEMP(i) < es_TEMP(i, j) And es_TEMP(i, j) < et_TEMP(i) Then↙
     fs_TEMP(i, j) = fy_TEMP(i)
                        If et_TEMP(i) < es_TEMP(i, j) And es_TEMP(i, j) < eu_TEMP(i) Then↙
     fs_TEMP(i, j) = fy_TEMP(i) * (1 - (es_TEMP(i, j) - et_TEMP(i)) / (eu_TEMP(i) -      ↙
    et_TEMP(i)))
                        If es_TEMP(i, j) = eu_TEMP(i) Then fs_TEMP(i, j) = 0
                        If es_TEMP(i, j) > eu_TEMP(i) Then fs_TEMP(i, j) = 0
                        .Cells(j + 1, 4 * (i + 1) + 2).value = fs_TEMP(i, j)
                    Next j

                    .Columns(4 * (i + 1) + 1).NumberFormat = "0.00000"
                    .Columns(4 * (i + 1) + 2).NumberFormat = "0.00"

                    .Cells(2, 4 * (i + 1)).value = fp_TEMP(i)
                    .Cells(3, 4 * (i + 1)).value = fy_TEMP(i)
                    .Cells(4, 4 * (i + 1)).value = Est_TEMP(i)
                    .Cells(5, 4 * (i + 1)).value = ep_TEMP(i)
                    .Cells(6, 4 * (i + 1)).value = ey_TEMP(i)
                    .Cells(7, 4 * (i + 1)).value = et_TEMP(i)
                    .Cells(8, 4 * (i + 1)).value = eu_TEMP(i)
                    .Cells(9, 4 * (i + 1)).value = a_TEMP(i)
                    .Cells(10, 4 * (i + 1)).value = b_TEMP(i)
                    .Cells(11, 4 * (i + 1)).value = c_TEMP(i)

                Next i
            End With

            'Plot steel stress-strain_____________________________________________
            prgLOAD.Value = 8000
            If xlScratch.Cells(8, 1).value = "FALSE" Then GoTo 301
            GoTo 302
301:        xlSteel.ChartObjects.Delete()

302:        xlScratch.Cells(8, 1).value = "FALSE"
            Dim Steel_plot As Excel.Chart
            Dim Steel_SerCol As Excel.SeriesCollection
            Dim Steel_Series As Excel.Series
            Dim Steel_AxisCategory, Steel_AxisValue As Excel.Axes

            xlCharts = xlSteel.ChartObjects
            myChart = xlCharts.Add(180, 260, 1000, 600)
            Steel_plot = myChart.Chart
            Steel_plot.ChartType = Excel.XlChartType.xlXYScatterLinesNoMarkers
            Steel_plot.ChartStyle = 3
```

```
            Steel_SerCol = Steel_plot.SeriesCollection

            Steel_Series = Steel_SerCol.NewSeries
            Steel_Series.Name = temp_TEMP(0) & "C"
            Steel_Series.XValues = xlSteel.Range("E2:E101")
            Steel_Series.Values = xlSteel.Range("F2:F101")
            Steel_Series.Format.Line.Weight = 1.5

            With Steel_plot
                .Legend.Position = Excel.XlLegendPosition.xlLegendPositionTop
                .HasTitle = True
                .ChartTitle.Text = "Steel Stress-Strain Relationship"

                Steel_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
                Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = True
                Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters↙
    .Text() = "Strain, mm/mm"
                Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters↙
    .Font.Bold = False
                Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters↙
    .Font.Size = 14
                Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines = ↙
    True
                Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines = ↙
    True
                Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.     ↙
    Format.Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
                Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScale = 0
                Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MaximumScale = 0.2
                Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MajorUnitIsAuto =   ↙
    True
                Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition = ↙
    Excel.XlTickLabelPosition.xlTickLabelPositionLow
                Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.         ↙
    NumberFormat = "0.000"
                Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold↙
     = False
                Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size↙
     = 14

                Steel_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
                Steel_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = True
                Steel_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Text ↙
    () = "Stress, MPa"
                Steel_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.↙
    Bold = False
                Steel_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.↙
    Size = 14
                Steel_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
                Steel_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
                Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line↙
    .DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
                Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScale = 0
                Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MaximumScaleIsAuto = True
                Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MajorUnitIsAuto = True
                Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.↙
    XlTickLabelPosition.xlTickLabelPositionLow
                Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.NumberFormat = ↙
    "0"
                Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold =    ↙
    False
                Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
            End With
            Steel_plot.Refresh()

            INITIAL = 0
```

```
            DUMss = 0
            dontRefreshSS = 0

            radConcrete.Checked = True
            lstSS_temps.SelectedItems.Clear()
            lstSS_temps.SelectedItem = "20C"
            xlConcrete.ChartObjects(1).chart.Export(FILEname:=FolderPath & "\temp" & "\
    results_CONC.bmp", FilterName:="BMP")
            plotSS.BackgroundImage = New System.Drawing.Bitmap(FolderPath & "\temp" & "\
    results_CONC.bmp")
            plotSS.BackgroundImageLayout = ImageLayout.Stretch



            'Load Fire Plot
            xlFire.ChartObjects(1).chart.Export(FILEname:=FolderPath & "\temp" & "\
    results_fire.bmp", FilterName:="BMP")
            plotFIRE.BackgroundImage = New System.Drawing.Bitmap(FolderPath & "\temp" &
    "\results_fire.bmp")
            plotFIRE.BackgroundImageLayout = ImageLayout.Stretch

            prgLOAD.Value = 8250

            lstFire.Items.Add(0)
            lstFire.Items.Item(0) = "0"
            For i = 1 To xlScratch.Cells(9, 1).value - 1
                lstFire.Items.Add(i)
                lstFire.Items.Item(i) = xlFire.Cells(i + 1, 1).value
            Next i



            'Load Average Temperature Plot
            xlScratch.ChartObjects(inAVG).chart.Export(FILEname:=FolderPath & "\temp" &
    "\results_avgtemp.bmp", FilterName:="BMP")
            plotTEMP.BackgroundImage = New System.Drawing.Bitmap(FolderPath & "\temp" &
    "\results_avgtemp.bmp")
            plotTEMP.BackgroundImageLayout = ImageLayout.Stretch

            prgLOAD.Value = 8500

            lstElements.Items.Add(0)
            lstElements.Items.Item(0) = "Average"
            For i = 1 To xlREOUT.Cells(4, 2).value
                lstElements.Items.Add(i)
                lstElements.Items.Item(i) = i
            Next i
            lstRebar.Items.Add(0)
            lstRebar.Items.Item(0) = "Average"
            For i = 1 To xlREOUT.Cells(23, 2).value
                lstRebar.Items.Add(i)
                lstRebar.Items.Item(i) = i
            Next i







            If xlREOUT.Cells(20, 2).value = 12 Then GoTo 400

            'Load Moment-Curvature Plots
            'Define Mphi Column References
```

```
            x_col(0) = "D"
            y_col(0) = "E"
            x_col(1) = "H"
            y_col(1) = "I"
            x_col(2) = "L"
            y_col(2) = "M"
            x_col(3) = "P"
            y_col(3) = "Q"
            x_col(4) = "T"
            y_col(4) = "U"
            x_col(5) = "X"
            y_col(5) = "Y"

            x_col(6) = "AB"
            y_col(6) = "AC"
            x_col(7) = "AF"
            y_col(7) = "AG"
            x_col(8) = "AJ"
            y_col(8) = "AK"
            x_col(9) = "AN"
            y_col(9) = "AO"
            x_col(10) = "AR"
            y_col(10) = "AS"
            x_col(11) = "AV"
            y_col(11) = "AW"
            x_col(12) = "AZ"

            y_col(12) = "BA"
            x_col(13) = "BD"
            y_col(13) = "BE"
            x_col(14) = "BH"
            y_col(14) = "BI"
            x_col(15) = "BL"
            y_col(15) = "BM"
            x_col(16) = "BP"
            y_col(16) = "BQ"
            x_col(17) = "BT"
            y_col(17) = "BU"
            x_col(18) = "BX"
            y_col(18) = "BY"

            x_col(19) = "CB"
            y_col(19) = "CC"
            x_col(20) = "CF"
            y_col(20) = "CG"
            x_col(21) = "CJ"
            y_col(21) = "CK"
            x_col(22) = "CN"
            y_col(22) = "CO"
            x_col(23) = "CR"
            y_col(23) = "CS"
            x_col(24) = "CV"
            y_col(24) = "CW"
            x_col(25) = "CZ"

            y_col(25) = "DA"
            x_col(26) = "DD"
            y_col(26) = "DE"
            x_col(27) = "DH"
            y_col(27) = "DI"
            x_col(28) = "DL"
            y_col(28) = "DM"
            x_col(29) = "DP"
            y_col(29) = "DQ"
            x_col(30) = "DT"
            y_col(30) = "DU"
```

```
            x_col(31) = "DX"
            y_col(31) = "DY"

            x_col(32) = "EB"
            y_col(32) = "EC"
            x_col(33) = "EF"
            y_col(33) = "EG"
            x_col(34) = "EJ"
            y_col(34) = "EK"
            x_col(35) = "EN"
            y_col(35) = "EO"
            x_col(36) = "ER"
            y_col(36) = "ES"
            x_col(37) = "EV"
            y_col(37) = "EW"
            x_col(38) = "EZ"

            y_col(38) = "FA"
            x_col(39) = "FD"
            y_col(39) = "FE"
            x_col(40) = "FH"
            y_col(40) = "FI"
            x_col(41) = "FL"
            y_col(41) = "FM"
            x_col(42) = "FP"
            y_col(42) = "FQ"
            x_col(43) = "FT"
            y_col(43) = "FU"
            x_col(44) = "FX"
            y_col(44) = "FY"

            x_col(45) = "GB"
            y_col(45) = "GC"
            x_col(46) = "GF"
            y_col(46) = "GG"
            x_col(47) = "GJ"
            y_col(47) = "GK"
            x_col(48) = "GN"
            y_col(48) = "GO"
            x_col(49) = "GR"
            y_col(49) = "GS"
            x_col(50) = "GV"
            y_col(50) = "GW"
            x_col(51) = "GZ"

            y_col(51) = "HA"
            x_col(52) = "HD"
            y_col(52) = "HE"
            x_col(53) = "HH"
            y_col(53) = "HI"
            x_col(54) = "HL"
            y_col(54) = "HM"
            x_col(55) = "HP"
            y_col(55) = "HQ"
            x_col(56) = "HT"
            y_col(56) = "HU"
            x_col(57) = "HX"
            y_col(57) = "HY"

            x_col(58) = "IB"
            y_col(58) = "IC"
            x_col(59) = "IF"
            y_col(59) = "IG"
            x_col(60) = "IJ"
            y_col(60) = "IK"
            x_col(61) = "IN"
```

```
            y_col(61) = "IO"
            x_col(62) = "IR"
            y_col(62) = "IS"
            x_col(63) = "IV"
            y_col(63) = "IW"
            x_col(64) = "IZ"

            y_col(64) = "JA"
            x_col(65) = "JD"
            y_col(65) = "JE"
            x_col(66) = "JH"
            y_col(66) = "JI"
            x_col(67) = "JL"
            y_col(67) = "JM"
            x_col(68) = "JP"
            y_col(68) = "JQ"
            x_col(69) = "JT"
            y_col(69) = "JU"
            x_col(70) = "JX"
            y_col(70) = "JY"

            x_col(71) = "KB"
            y_col(71) = "KC"
            x_col(72) = "KF"
            y_col(72) = "KG"
            x_col(73) = "KJ"
            y_col(73) = "KK"
            x_col(74) = "KN"
            y_col(74) = "KO"
            x_col(75) = "KR"
            y_col(75) = "KS"
            x_col(76) = "KV"
            y_col(76) = "KW"
            x_col(77) = "KZ"

            y_col(77) = "LA"
            x_col(78) = "LD"
            y_col(78) = "LE"
            x_col(79) = "LH"
            y_col(79) = "LI"
            x_col(80) = "LL"
            y_col(80) = "LM"
            x_col(81) = "LP"
            y_col(81) = "LQ"
            x_col(82) = "LT"
            y_col(82) = "LU"
            x_col(83) = "LX"
            y_col(83) = "LY"

            x_col(84) = "MB"
            y_col(84) = "MC"
            x_col(85) = "MF"
            y_col(85) = "MG"
            x_col(86) = "MJ"
            y_col(86) = "MK"
            x_col(87) = "MN"
            y_col(87) = "MO"
            x_col(88) = "MR"
            y_col(88) = "MS"
            x_col(89) = "MV"
            y_col(89) = "MW"
            x_col(90) = "MZ"

            y_col(90) = "NA"
            x_col(91) = "ND"
            y_col(91) = "NE"
```

```
            x_col(92) = "NH"
            y_col(92) = "NI"
            x_col(93) = "NL"
            y_col(93) = "NM"
            x_col(94) = "NP"
            y_col(94) = "NQ"
            x_col(95) = "NT"
            y_col(95) = "NU"
            x_col(96) = "NX"
            y_col(96) = "NY"

            x_col(97) = "OB"
            y_col(97) = "OC"
            x_col(98) = "OF"
            y_col(98) = "OG"
            x_col(99) = "OJ"
            y_col(99) = "OK"
            x_col(100) = "ON"
            y_col(100) = "OO"
            x_col(101) = "OR"
            y_col(101) = "OS"
            x_col(102) = "OV"
            y_col(102) = "OW"
            x_col(103) = "OZ"

            y_col(103) = "PA"
            x_col(104) = "PD"
            y_col(104) = "PE"
            x_col(105) = "PH"
            y_col(105) = "PI"
            x_col(106) = "PL"
            y_col(106) = "PM"
            x_col(107) = "PP"
            y_col(107) = "PQ"
            x_col(108) = "PT"
            y_col(108) = "PU"
            x_col(109) = "PX"
            y_col(109) = "PY"

            x_col(110) = "QB"
            y_col(110) = "QC"
            x_col(111) = "QF"
            y_col(111) = "QG"
            x_col(112) = "QJ"
            y_col(112) = "QK"
            x_col(113) = "QN"
            y_col(113) = "QO"
            x_col(114) = "QR"
            y_col(114) = "QS"
            x_col(115) = "QV"
            y_col(115) = "QW"
            x_col(116) = "QZ"

            y_col(116) = "RA"
            x_col(117) = "RD"
            y_col(117) = "RE"
            x_col(118) = "RH"
            y_col(118) = "RI"
            x_col(119) = "RL"
            y_col(119) = "RM"
            x_col(120) = "RP"
            y_col(120) = "RQ"
            x_col(121) = "RT"
            y_col(121) = "RU"
            x_col(122) = "RX"
            y_col(122) = "RY"
```

```
            x_col(123) = "SB"
            y_col(123) = "SC"
            x_col(124) = "SF"
            y_col(124) = "SG"
            x_col(125) = "SJ"
            y_col(125) = "SK"
            x_col(126) = "SN"
            y_col(126) = "SO"
            x_col(127) = "SR"
            y_col(127) = "SS"
            x_col(128) = "SV"
            y_col(128) = "SW"
            x_col(129) = "SZ"

            y_col(129) = "TA"
            x_col(130) = "TD"
            y_col(130) = "TE"
            x_col(131) = "TH"
            y_col(131) = "TI"
            x_col(132) = "TL"
            y_col(132) = "TM"
            x_col(133) = "TP"
            y_col(133) = "TQ"
            x_col(134) = "TT"
            y_col(134) = "TU"
            x_col(135) = "TX"
            y_col(135) = "TY"

            x_col(136) = "UB"
            y_col(136) = "UC"
            x_col(137) = "UF"
            y_col(137) = "UG"
            x_col(138) = "UJ"
            y_col(138) = "UK"
            x_col(139) = "UN"
            y_col(139) = "UO"
            x_col(140) = "UR"
            y_col(140) = "US"
            x_col(141) = "UV"
            y_col(141) = "UW"
            x_col(142) = "UZ"

            y_col(142) = "VA"
            x_col(143) = "VD"
            y_col(143) = "VE"
            x_col(144) = "VH"
            y_col(144) = "VI"
            x_col(145) = "VL"
            y_col(145) = "VM"
            x_col(146) = "VP"
            y_col(146) = "VQ"
            x_col(147) = "VT"
            y_col(147) = "VU"
            x_col(148) = "VX"
            y_col(148) = "VY"

            x_col(149) = "WB"
            y_col(149) = "WC"
            x_col(150) = "WF"
            y_col(150) = "WG"
            x_col(151) = "WJ"
            y_col(151) = "WK"
            x_col(152) = "WN"
            y_col(152) = "WO"
            x_col(153) = "WR"
```

```
            y_col(153) = "WS"
            x_col(154) = "WV"
            y_col(154) = "WW"
            x_col(155) = "WZ"

            y_col(155) = "XA"
            x_col(156) = "XD"
            y_col(156) = "XE"
            x_col(157) = "XH"
            y_col(157) = "XI"
            x_col(158) = "XL"
            y_col(158) = "XM"
            x_col(159) = "XP"
            y_col(159) = "XQ"
            x_col(160) = "XT"
            y_col(160) = "XU"
            x_col(161) = "XX"
            y_col(161) = "XY"

            x_col(162) = "YB"
            y_col(162) = "YC"
            x_col(163) = "YF"
            y_col(163) = "YG"
            x_col(164) = "YJ"
            y_col(164) = "YK"
            x_col(165) = "YN"
            y_col(165) = "YO"
            x_col(166) = "YR"
            y_col(166) = "YS"
            x_col(167) = "YV"
            y_col(167) = "YW"
            x_col(168) = "YZ"

            y_col(168) = "ZA"
            x_col(169) = "ZD"
            y_col(169) = "ZE"
            x_col(170) = "ZH"
            y_col(170) = "ZI"
            x_col(171) = "ZL"
            y_col(171) = "ZM"
            x_col(172) = "ZP"
            y_col(172) = "ZQ"
            x_col(173) = "ZT"
            y_col(173) = "ZU"
            x_col(174) = "ZX"
            y_col(174) = "ZY"

            x_col(175) = "AAB"
            y_col(175) = "AAC"
            x_col(176) = "AAF"
            y_col(176) = "AAG"
            x_col(177) = "AAJ"
            y_col(177) = "AAK"
            x_col(178) = "AAN"
            y_col(178) = "AAO"
            x_col(179) = "AAR"
            y_col(179) = "AAS"
            x_col(180) = "AAV"
            y_col(180) = "AAW"
            x_col(181) = "AAZ"

            y_col(181) = "ABA"
            x_col(182) = "ABD"
            y_col(182) = "ABE"
            x_col(183) = "ABH"
            y_col(183) = "ABI"
```

```
            x_col(184) = "ABL"
            y_col(184) = "ABM"
            x_col(185) = "ABP"
            y_col(185) = "ABQ"
            x_col(186) = "ABT"
            y_col(186) = "ABU"
            x_col(187) = "ABX"
            y_col(187) = "ABY"

            x_col(188) = "ACB"
            y_col(188) = "ACC"
            x_col(189) = "ACF"
            y_col(189) = "ACG"
            x_col(190) = "ACJ"
            y_col(190) = "ACK"
            x_col(191) = "ACN"
            y_col(191) = "ACO"
            x_col(192) = "ACR"
            y_col(192) = "ACS"
            x_col(193) = "ACV"
            y_col(193) = "ACW"
            x_col(194) = "ACZ"

            y_col(194) = "ADA"
            x_col(195) = "ADD"
            y_col(195) = "ADE"
            x_col(196) = "ADH"
            y_col(196) = "ADI"
            x_col(197) = "ADL"
            y_col(197) = "ADM"
            x_col(198) = "ADP"
            y_col(198) = "ADQ"
            x_col(199) = "ADT"
            y_col(199) = "ADU"
            x_col(200) = "ADX"
            y_col(200) = "ADY"

            x_col(201) = "AEB"
            y_col(201) = "AEC"
            x_col(202) = "AEF"
            y_col(202) = "AEG"
            x_col(203) = "AEJ"
            y_col(203) = "AEK"
            x_col(204) = "AEN"
            y_col(204) = "AEO"
            x_col(205) = "AER"
            y_col(205) = "AES"
            x_col(206) = "AEV"
            y_col(206) = "AEW"
            x_col(207) = "AEZ"

            y_col(207) = "AFA"
            x_col(208) = "AFD"
            y_col(208) = "AFE"
            x_col(209) = "AFH"
            y_col(209) = "AFI"
            x_col(210) = "AFL"
            y_col(210) = "AFM"
            x_col(211) = "AFP"
            y_col(211) = "AFQ"
            x_col(212) = "AFT"
            y_col(212) = "AFU"
            x_col(213) = "AFX"
            y_col(213) = "AFY"

            x_col(214) = "AGB"
```

```
            y_col(214) = "AGC"
            x_col(215) = "AGF"
            y_col(215) = "AGG"
            x_col(216) = "AGJ"
            y_col(216) = "AGK"
            x_col(217) = "AGN"
            y_col(217) = "AGO"
            x_col(218) = "AGR"
            y_col(218) = "AGS"
            x_col(219) = "AGV"
            y_col(219) = "AGW"
            x_col(220) = "AGZ"

            y_col(220) = "AHA"
            x_col(221) = "AHD"
            y_col(221) = "AHE"
            x_col(222) = "AHH"
            y_col(222) = "AHI"
            x_col(223) = "AHL"
            y_col(223) = "AHM"
            x_col(224) = "AHP"
            y_col(224) = "AHQ"
            x_col(225) = "AHT"
            y_col(225) = "AHU"
            x_col(226) = "AHX"
            y_col(226) = "AHY"

            x_col(227) = "AIB"
            y_col(227) = "AIC"
            x_col(228) = "AIF"
            y_col(228) = "AIG"
            x_col(229) = "AIJ"
            y_col(229) = "AIK"
            x_col(230) = "AIN"
            y_col(230) = "AIO"
            x_col(231) = "AIR"
            y_col(231) = "AIS"
            x_col(232) = "AIV"
            y_col(232) = "AIW"
            x_col(233) = "AIZ"

            y_col(233) = "AJA"
            x_col(234) = "AJD"
            y_col(234) = "AJE"
            x_col(235) = "AJH"
            y_col(235) = "AJI"
            x_col(236) = "AJL"
            y_col(236) = "AJM"
            x_col(237) = "AJP"
            y_col(237) = "AJQ"
            x_col(238) = "AJT"
            y_col(238) = "AJU"
            x_col(239) = "AJX"
            y_col(239) = "AJY"

            x_col(240) = "AKB"
            y_col(240) = "AKC"

            prgLOAD.Value = 9000

            'Load Moment-Curvature Plot
            xlMPhi.ChartObjects(1).chart.Export(FILEname:=FolderPath & "\temp" & "\      ↙
    results_mphi.bmp", FilterName:="BMP")
            plotMPHI.BackgroundImage = New System.Drawing.Bitmap(FolderPath & "\temp" &  ↙
    "\results_mphi.bmp")
            plotMPHI.BackgroundImageLayout = ImageLayout.Stretch
```

```
            DUM = 0
            dontRefresh = 1

            timesteps = xlREOUT.Cells(9, 2).value
            For i = 0 To timesteps
                lstMPHI_step.Items.Add(i)
                lstMPHI_step.Items.Item(i) = xlMPhi.Cells(1, 4 * i + 3).value
                MPhi_Index(lstMPHI_step.Items.Item(i)) = i
            Next i

            lstMPHI_step.SelectedItem = lstMPHI_step.Items.Item(0)
            dontRefresh = 0
            mphiPLOTpoints = xlMPhi.Cells(3, 1).value




400:        prgLOAD.Value = 9500

            If xlREOUT.Cells(20, 2).value = 21 Then GoTo 401
            'Load Moment-Axial Plots
            xlMN.ChartObjects(1).chart.Export(FILEname:=FolderPath & "\temp" & "\ ↙
    results_MN.bmp", FilterName:="BMP")
            plotMN.BackgroundImage = New System.Drawing.Bitmap(FolderPath & "\temp" & "\ ↙
    results_MN.bmp")
            plotMN.BackgroundImageLayout = ImageLayout.Stretch

            DUMmn = 0
            dontRefreshMN = 1

            timesteps = xlREOUT.Cells(9, 2).value
            For i = 0 To timesteps
                lstMN_step.Items.Add(i)
                lstMN_step.Items.Item(i) = xlMN.Cells(1, 4 * i + 3).value
                MN_Index(lstMN_step.Items.Item(i)) = i
            Next i

            lstMN_step.SelectedItem = lstMN_step.Items.Item(0)
            dontRefreshMN = 0
            mnPLOTpoints = xlMN.Cells(1, 1).value


            'Load Max Axial Envelope Curve
            xlScratch.ChartObjects(inN).chart.Export(FILEname:=FolderPath & "\temp" & "\ ↙
    results_axial.bmp", FilterName:="BMP")
            plotAXIAL.BackgroundImage = New System.Drawing.Bitmap(FolderPath & "\temp" & ↙
    "\results_axial.bmp")
            plotAXIAL.BackgroundImageLayout = ImageLayout.Stretch

            prgLOAD.Value = 9750
            For i = 0 To 3
                lstAxial.Items.Add(i)
            Next i

            lstAxial.Items.Item(0) = "Compression"
            lstAxial.Items.Item(1) = "Tension"
            lstAxial.Items.Item(2) = "Fire/+Pmax"
            lstAxial.Items.Item(3) = "Fire/-Pmax"

            lstAxial.SelectedItem = lstAxial.Items.Item(0)
```

```
401:        If xlREOUT.Cells(20, 2).value = 12 Then GoTo 402
            'Load Max Moment Envelope Curve
            prgLOAD.Value = 9850
            xlScratch.ChartObjects(inM).chart.Export(FILEname:=FolderPath & "\temp" & "\ ↙
    results_mom.bmp", FilterName:="BMP")
            plotMOMENT.BackgroundImage = New System.Drawing.Bitmap(FolderPath & "\temp" &↙
     "\results_mom.bmp")
            plotMOMENT.BackgroundImageLayout = ImageLayout.Stretch

            For i = 0 To 3
                lstMoment.Items.Add(i)
            Next i

            lstMoment.Items.Item(0) = "Positive"
            lstMoment.Items.Item(1) = "Negative"
            lstMoment.Items.Item(2) = "Fire/+Mmax"
            lstMoment.Items.Item(3) = "Fire/-Mmax"

            lstMoment.SelectedItem = lstMoment.Items.Item(0)
            lstMoment.SelectedItem = lstMoment.Items.Item(1)



402:        'Load Rn(t) Curves
            xlScratch.ChartObjects(inR).chart.Export(FILEname:=FolderPath & "\temp" & "\ ↙
    results_rn.bmp", FilterName:="BMP")
            plotRN.BackgroundImage = New System.Drawing.Bitmap(FolderPath & "\temp" & "\ ↙
    results_rn.bmp")
            plotRN.BackgroundImageLayout = ImageLayout.Stretch

            For i = 0 To 4
                lstRN.Items.Add(i)
            Next i

            lstRN.Items.Item(0) = "Fire"
            lstRN.Items.Item(1) = "Axial (C)"
            lstRN.Items.Item(2) = "Axial (T)"
            lstRN.Items.Item(3) = "Moment (+)"
            lstRN.Items.Item(4) = "Moment (-)"

            lstRN.SelectedItem = lstRN.Items.Item(0)
            lstRN.SelectedItem = lstRN.Items.Item(1)
            lstRN.SelectedItem = lstRN.Items.Item(2)
            lstRN.SelectedItem = lstRN.Items.Item(3)
            lstRN.SelectedItem = lstRN.Items.Item(4)




            'DONEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
            prgLOAD.Value = 10000
            title1.Visible = True
            title2.Visible = True
            title3.Visible = True
            title4.Visible = True
            imgStart.Visible = True
            title5.Visible = False
            txtFilePath.ForeColor = Color.Black
            txtOutputFolder.ForeColor = Color.Black
            txtSectionName.ForeColor = Color.Black
            txtNodeCount.ForeColor = Color.Black
            txtElementCount.ForeColor = Color.Black
            txtFireFileUsed.ForeColor = Color.Black
```

```
            txtCalcTimeStep.ForeColor = Color.Black
            txtCalcEndTime.ForeColor = Color.Black
            txtCalcStepCount.ForeColor = Color.Black
            txtFireStepCount.ForeColor = Color.Black
            txtSectionHeightMax.ForeColor = Color.Black
            txtSectionHeightMin.ForeColor = Color.Black
            txtSectionWidthMax.ForeColor = Color.Black
            txtSectionWidthMin.ForeColor = Color.Black
            txtRebarCount.ForeColor = Color.Black
            txtTotalSteelArea.ForeColor = Color.Black
            txtGrossSectionArea.ForeColor = Color.Black
            txtNetSectionArea.ForeColor = Color.Black
            txtReinforcementRatio.ForeColor = Color.Black
            txtSteelProportionalStressLimit.ForeColor = Color.Black
            txtSteelMaximumStressLevel.ForeColor = Color.Black
            txtConcreteCompressiveStrength.ForeColor = Color.Black
            txtConcreteTensileStrength.ForeColor = Color.Black
            txtConcreteAggregatesUsed.ForeColor = Color.Black
            txtMaximumAxialCapacity.ForeColor = Color.Black
            txtMaximumAxialCapacity_t.ForeColor = Color.Black
            txtMinimumAxialCapacity.ForeColor = Color.Black
            txtMinimumAxialCapacity_t.ForeColor = Color.Black
            txtMaximumMomentCapacity.ForeColor = Color.Black
            txtMaximumMomentCapacity_t.ForeColor = Color.Black
            txtMinimumMomentCapacity.ForeColor = Color.Black
            txtMinimumMomentCapacity_t.ForeColor = Color.Black
    End Sub
    Private Sub Results_FormClosing(ByVal sender As Object, ByVal e As System.Windows.    ↙
    Forms.FormClosingEventArgs) Handles Me.FormClosing

        MessageBox.Show("Saving output as.... " & FilePath, "Closing...",                 ↙
    MessageBoxButtons.OK)

        xlUTFire.Close(True, FilePath)
        xlApp.Quit()

        releaseObject(xlApp)
        releaseObject(xlUTFire)
        releaseObject(xlOUT)
        releaseObject(xlREOUT)
        releaseObject(xlFire)
        releaseObject(xlSection)
        releaseObject(xlSteel)
        releaseObject(xlConcrete)
        releaseObject(xlMPhi)
        releaseObject(xlScratch)
        releaseObject(xlMN)
        releaseObject(xlDOOP)

        Dim AllProcesses() As Process = Process.GetProcessesByName("excel")
        Dim ExcelProcess As Process
        For Each ExcelProcess In AllProcesses
            ExcelProcess.Kill()
        Next
        ExcelProcess = Nothing
        AllProcesses = Nothing

    End Sub
    Private Sub Results_FormClosed(ByVal sender As Object, ByVal e As System.Windows.     ↙
    Forms.FormClosedEventArgs) Handles Me.FormClosed

        plotSECTION.BackgroundImage.Dispose()
        plotSS.BackgroundImage.Dispose()
        plotFIRE.BackgroundImage.Dispose()
        plotTEMP.BackgroundImage.Dispose()
        plotMPHI.BackgroundImage.Dispose()
```

```
        plotMN.BackgroundImage.Dispose()
        plotAXIAL.BackgroundImage.Dispose()
        plotMOMENT.BackgroundImage.Dispose()
        plotRN.BackgroundImage.Dispose()

        Dim s As String
        For Each s In System.IO.Directory.GetFiles(FolderPath & "\temp")
            System.IO.File.Delete(s)
        Next s

        End

    End Sub



    'ANALYSIS SUMMARY
    Private Sub imgStart_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles imgStart.Click
        Dim About As New About
        About.ShowDialog()
        About.Refresh()
    End Sub



    'STRESS-STRAIN
    Private Sub radConcrete_CheckedChanged(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles radConcrete.CheckedChanged

        If dontRefreshSS = 1 Then GoTo 999
        DUMss = DUMss + 1

        INITIAL = INITIAL + 1
        If INITIAL = 1 Then GoTo 999

        If radConcrete.Checked = True Then GoTo 1
        If radSteel.Checked = True Then GoTo 2
        GoTo 999

        '**CONCRETE*****************************************************************
    ***************
1:      xlConcrete.Range("A12:C26").Clear()
        SStempCOUNT = lstSS_temps.SelectedItems.Count
        xlConcrete.Cells(13, 1).value = SStempCOUNT
        SStempCOUNT = SStempCOUNT - 1
        If SStempCOUNT = -1 Then GoTo 999

        For i = 0 To SStempCOUNT
            xlConcrete.Cells(13 + i, 2).value = lstSS_temps.SelectedItems.Item(i)
            If lstSS_temps.SelectedItems.Item(i) = "20C" Then xlConcrete.Cells(13 + i, 3)
    .value = 1
            If lstSS_temps.SelectedItems.Item(i) = "100C" Then xlConcrete.Cells(13 + i,
    3).value = 2
            If lstSS_temps.SelectedItems.Item(i) = "200C" Then xlConcrete.Cells(13 + i,
    3).value = 3
            If lstSS_temps.SelectedItems.Item(i) = "300C" Then xlConcrete.Cells(13 + i,
    3).value = 4
            If lstSS_temps.SelectedItems.Item(i) = "400C" Then xlConcrete.Cells(13 + i,
    3).value = 5
            If lstSS_temps.SelectedItems.Item(i) = "500C" Then xlConcrete.Cells(13 + i,
    3).value = 6
            If lstSS_temps.SelectedItems.Item(i) = "600C" Then xlConcrete.Cells(13 + i,
    3).value = 7
            If lstSS_temps.SelectedItems.Item(i) = "700C" Then xlConcrete.Cells(13 + i,
    3).value = 8
```

```
            If lstSS_temps.SelectedItems.Item(i) = "800C" Then xlConcrete.Cells(13 + i,
    3).value = 9
            If lstSS_temps.SelectedItems.Item(i) = "900C" Then xlConcrete.Cells(13 + i,
    3).value = 10
            If lstSS_temps.SelectedItems.Item(i) = "1000C" Then xlConcrete.Cells(13 + i,
    3).value = 11
            If lstSS_temps.SelectedItems.Item(i) = "1100C" Then xlConcrete.Cells(13 + i,
    3).value = 12
            If lstSS_temps.SelectedItems.Item(i) = "1200C" Then xlConcrete.Cells(13 + i,
    3).value = 13
        Next i

        'Plot concrete stress-strain_____________________________________________
        xlConcrete.ChartObjects.Delete()

        Dim concrete_plot As Excel.Chart
        Dim xlCharts As Excel.ChartObjects
        Dim myChart As Excel.ChartObject
        Dim concrete_SerCol As Excel.SeriesCollection
        Dim concrete_Series(0 To 13) As Excel.Series
        Dim concrete_AxisCategory, concrete_AxisValue As Excel.Axes

        xlCharts = xlConcrete.ChartObjects
        myChart = xlCharts.Add(180, 260, 1000, 600)
        concrete_plot = myChart.Chart
        concrete_plot.ChartType = Excel.XlChartType.xlXYScatterLinesNoMarkers
        concrete_plot.ChartStyle = 3
        concrete_SerCol = concrete_plot.SeriesCollection

        SSplotCOUNT = -1

        For i = 1 To 13
            For j = 0 To 12
                If xlConcrete.Cells(13 + j, 3).value = i Then GoTo 100
            Next j
            GoTo 101
100:        SSplotCOUNT = SSplotCOUNT + 1
            concrete_Series(SSplotCOUNT) = concrete_SerCol.NewSeries
            concrete_Series(SSplotCOUNT).Name = temp_TEMP(i - 1) & "C"
            concrete_Series(SSplotCOUNT).XValues = xlConcrete.Range(xSS_col(i - 1) & "2:"
     & xSS_col(i - 1) & "101")
            concrete_Series(SSplotCOUNT).Values = xlConcrete.Range(ySS_col(i - 1) & "2:"
    & ySS_col(i - 1) & "101")
            concrete_Series(SSplotCOUNT).Format.Line.Weight = 1.5
101:    Next i

        With concrete_plot
            .Legend.Position = Excel.XlLegendPosition.xlLegendPositionTop
            .HasTitle = True
            .ChartTitle.Text = "Concrete Stress-Strain Relationship"

            concrete_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.
    Text() = "Strain, mm/mm"
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.
    Font.Bold = False
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.
    Font.Size = 14
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines =
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines =
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format
    .Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScaleIsAuto =
```

```
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MaximumScaleIsAuto =
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MajorUnitIsAuto =
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition =
    Excel.XlTickLabelPosition.xlTickLabelPositionLow
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.
    NumberFormat = "0.000"
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold
    = False
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size
    = 14

            concrete_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Text()
     = "Stress, MPa"
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.
    Bold = False
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.
    Size = 14
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line.
    DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScaleIsAuto = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MaximumScaleIsAuto = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MajorUnitIsAuto = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.
    XlTickLabelPosition.xlTickLabelPositionLow
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.NumberFormat =
    "0"
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold =
    False
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
        End With
        concrete_plot.Refresh()

        xlConcrete.ChartObjects(1).chart.Export(FILEname:=FolderPath & "\temp" & "\
    results_CONC" & DUMss & ".bmp", FilterName:="BMP")
        plotSS.BackgroundImage = New System.Drawing.Bitmap(FolderPath & "\temp" & "\
    results_CONC" & DUMss & ".bmp")
        plotSS.BackgroundImageLayout = ImageLayout.Stretch

        GoTo 999

        '**STEEL*****************************************************************
    ************
2:      xlSteel.Range("A12:C26").Clear()
        SStempCOUNT = lstSS_temps.SelectedItems.Count
        xlSteel.Cells(13, 1).value = SStempCOUNT
        SStempCOUNT = SStempCOUNT - 1
        If SStempCOUNT = -1 Then GoTo 999

        For i = 0 To SStempCOUNT
            xlSteel.Cells(13 + i, 2).value = lstSS_temps.SelectedItems.Item(i)
            If lstSS_temps.SelectedItems.Item(i) = "20C" Then xlSteel.Cells(13 + i, 3).
    value = 1
            If lstSS_temps.SelectedItems.Item(i) = "100C" Then xlSteel.Cells(13 + i, 3).
    value = 2
            If lstSS_temps.SelectedItems.Item(i) = "200C" Then xlSteel.Cells(13 + i, 3).
    value = 3
            If lstSS_temps.SelectedItems.Item(i) = "300C" Then xlSteel.Cells(13 + i, 3).
    value = 4
            If lstSS_temps.SelectedItems.Item(i) = "400C" Then xlSteel.Cells(13 + i, 3).
```

```
    value = 5
            If lstSS_temps.SelectedItems.Item(i) = "500C" Then xlSteel.Cells(13 + i, 3). ↙
    value = 6
            If lstSS_temps.SelectedItems.Item(i) = "600C" Then xlSteel.Cells(13 + i, 3). ↙
    value = 7
            If lstSS_temps.SelectedItems.Item(i) = "700C" Then xlSteel.Cells(13 + i, 3). ↙
    value = 8
            If lstSS_temps.SelectedItems.Item(i) = "800C" Then xlSteel.Cells(13 + i, 3). ↙
    value = 9
            If lstSS_temps.SelectedItems.Item(i) = "900C" Then xlSteel.Cells(13 + i, 3). ↙
    value = 10
            If lstSS_temps.SelectedItems.Item(i) = "1000C" Then xlSteel.Cells(13 + i, 3).↙
    value = 11
            If lstSS_temps.SelectedItems.Item(i) = "1100C" Then xlSteel.Cells(13 + i, 3).↙
    value = 12
            If lstSS_temps.SelectedItems.Item(i) = "1200C" Then xlSteel.Cells(13 + i, 3).↙
    value = 13
        Next i

        'Plot Steel stress-strain_____________________________________________
        xlSteel.ChartObjects.Delete()

        Dim Steel_plot As Excel.Chart
        Dim Steel_SerCol As Excel.SeriesCollection
        Dim Steel_Series(0 To 13) As Excel.Series
        Dim Steel_AxisCategory, Steel_AxisValue As Excel.Axes

        xlCharts = xlSteel.ChartObjects
        myChart = xlCharts.Add(180, 260, 1000, 600)
        Steel_plot = myChart.Chart
        Steel_plot.ChartType = Excel.XlChartType.xlXYScatterLinesNoMarkers
        Steel_plot.ChartStyle = 3
        Steel_SerCol = Steel_plot.SeriesCollection

        SSplotCOUNT = -1

        For i = 1 To 13
            For j = 0 To 12
                If xlSteel.Cells(13 + j, 3).value = i Then GoTo 200
            Next j
            GoTo 201
200:        SSplotCOUNT = SSplotCOUNT + 1
            Steel_Series(SSplotCOUNT) = Steel_SerCol.NewSeries
            Steel_Series(SSplotCOUNT).Name = temp_TEMP(i - 1) & "C"
            Steel_Series(SSplotCOUNT).XValues = xlSteel.Range(xSS_col(i - 1) & "2:" &     ↙
    xSS_col(i - 1) & "101")
            Steel_Series(SSplotCOUNT).Values = xlSteel.Range(ySS_col(i - 1) & "2:" &      ↙
    ySS_col(i - 1) & "101")
            Steel_Series(SSplotCOUNT).Format.Line.Weight = 1.5
201:    Next i

        With Steel_plot
            .Legend.Position = Excel.XlLegendPosition.xlLegendPositionTop
            .HasTitle = True
            .ChartTitle.Text = "Steel Stress-Strain Relationship"

            Steel_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = True
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.    ↙
    Text() = "Strain, mm/mm"
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.    ↙
    Font.Bold = False
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.    ↙
    Font.Size = 14
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines = True
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines = True
```

```
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format.
    Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScale = 0
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MaximumScale = 0.2
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MajorUnitIsAuto = True
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition =
    Excel.XlTickLabelPosition.xlTickLabelPositionLow
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.NumberFormat
    = "0.000"
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold =
    False
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size =
    14

            Steel_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Text() =
    "Stress, MPa"
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Bold
     = False
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Size
     = 14
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line.
    DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScale = 0
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MaximumScaleIsAuto = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MajorUnitIsAuto = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.
    XlTickLabelPosition.xlTickLabelPositionLow
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.NumberFormat = "0"
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold = False
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
        End With

        Steel_plot.Refresh()

        xlSteel.ChartObjects(1).chart.Export(FILEname:=FolderPath & "\temp" & "\
    results_REINF" & DUMss & ".bmp", FilterName:="BMP")
        plotSS.BackgroundImage = New System.Drawing.Bitmap(FolderPath & "\temp" & "\
    results_REINF" & DUMss & ".bmp")
        plotSS.BackgroundImageLayout = ImageLayout.Stretch

999:

    End Sub
    Private Sub radSteel_CheckedChanged(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles radSteel.CheckedChanged

        If dontRefreshSS = 1 Then GoTo 999
        DUMss = DUMss + 1

        INITIAL = INITIAL + 1
        If INITIAL = 1 Then GoTo 999

        If radConcrete.Checked = True Then GoTo 1
        If radSteel.Checked = True Then GoTo 2
        GoTo 999

        '**CONCRETE*****************************************************************
    ***************
1:      xlConcrete.Range("A12:C26").Clear()
        SStempCOUNT = lstSS_temps.SelectedItems.Count
        xlConcrete.Cells(13, 1).value = SStempCOUNT
        SStempCOUNT = SStempCOUNT - 1
```

```
        If SStempCOUNT = -1 Then GoTo 999

        For i = 0 To SStempCOUNT
            xlConcrete.Cells(13 + i, 2).value = lstSS_temps.SelectedItems.Item(i)
            If lstSS_temps.SelectedItems.Item(i) = "20C" Then xlConcrete.Cells(13 + i, 3)↙
    .value = 1
            If lstSS_temps.SelectedItems.Item(i) = "100C" Then xlConcrete.Cells(13 + i,  ↙
    3).value = 2
            If lstSS_temps.SelectedItems.Item(i) = "200C" Then xlConcrete.Cells(13 + i,  ↙
    3).value = 3
            If lstSS_temps.SelectedItems.Item(i) = "300C" Then xlConcrete.Cells(13 + i,  ↙
    3).value = 4
            If lstSS_temps.SelectedItems.Item(i) = "400C" Then xlConcrete.Cells(13 + i,  ↙
    3).value = 5
            If lstSS_temps.SelectedItems.Item(i) = "500C" Then xlConcrete.Cells(13 + i,  ↙
    3).value = 6
            If lstSS_temps.SelectedItems.Item(i) = "600C" Then xlConcrete.Cells(13 + i,  ↙
    3).value = 7
            If lstSS_temps.SelectedItems.Item(i) = "700C" Then xlConcrete.Cells(13 + i,  ↙
    3).value = 8
            If lstSS_temps.SelectedItems.Item(i) = "800C" Then xlConcrete.Cells(13 + i,  ↙
    3).value = 9
            If lstSS_temps.SelectedItems.Item(i) = "900C" Then xlConcrete.Cells(13 + i,  ↙
    3).value = 10
            If lstSS_temps.SelectedItems.Item(i) = "1000C" Then xlConcrete.Cells(13 + i, ↙
    3).value = 11
            If lstSS_temps.SelectedItems.Item(i) = "1100C" Then xlConcrete.Cells(13 + i, ↙
    3).value = 12
            If lstSS_temps.SelectedItems.Item(i) = "1200C" Then xlConcrete.Cells(13 + i, ↙
    3).value = 13
        Next i

        'Plot concrete stress-strain_____________________________________________
        xlConcrete.ChartObjects.Delete()

        Dim concrete_plot As Excel.Chart
        Dim xlCharts As Excel.ChartObjects
        Dim myChart As Excel.ChartObject
        Dim concrete_SerCol As Excel.SeriesCollection
        Dim concrete_Series(0 To 13) As Excel.Series
        Dim concrete_AxisCategory, concrete_AxisValue As Excel.Axes

        xlCharts = xlConcrete.ChartObjects
        myChart = xlCharts.Add(180, 260, 1000, 600)
        concrete_plot = myChart.Chart
        concrete_plot.ChartType = Excel.XlChartType.xlXYScatterLinesNoMarkers
        concrete_plot.ChartStyle = 3
        concrete_SerCol = concrete_plot.SeriesCollection

        SSplotCOUNT = -1

        For i = 1 To 13
            For j = 0 To 12
                If xlConcrete.Cells(13 + j, 3).value = i Then GoTo 100
            Next j
            GoTo 101
100:        SSplotCOUNT = SSplotCOUNT + 1
            concrete_Series(SSplotCOUNT) = concrete_SerCol.NewSeries
            concrete_Series(SSplotCOUNT).Name = temp_TEMP(i - 1) & "C"
            concrete_Series(SSplotCOUNT).XValues = xlConcrete.Range(xSS_col(i - 1) & "2:"↙
     & xSS_col(i - 1) & "101")
            concrete_Series(SSplotCOUNT).Values = xlConcrete.Range(ySS_col(i - 1) & "2:" ↙
    & ySS_col(i - 1) & "101")
            concrete_Series(SSplotCOUNT).Format.Line.Weight = 1.5
101:    Next i
```

```
        With concrete_plot
            .Legend.Position = Excel.XlLegendPosition.xlLegendPositionTop
            .HasTitle = True
            .ChartTitle.Text = "Concrete Stress-Strain Relationship"

            concrete_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.↙
    Text() = "Strain, mm/mm"
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.↙
    Font.Bold = False
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.↙
    Font.Size = 14
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines =  ↙
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines =  ↙
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format↙
    .Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScaleIsAuto = ↙
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MaximumScaleIsAuto = ↙
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MajorUnitIsAuto =    ↙
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition =  ↙
    Excel.XlTickLabelPosition.xlTickLabelPositionLow
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.          ↙
    NumberFormat = "0.000"
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold ↙
    = False
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size ↙
    = 14

            concrete_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Text()↙
     = "Stress, MPa"
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font. ↙
    Bold = False
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font. ↙
    Size = 14
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line.↙
    DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScaleIsAuto = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MaximumScaleIsAuto = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MajorUnitIsAuto = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel. ↙
    XlTickLabelPosition.xlTickLabelPositionLow
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.NumberFormat =  ↙
    "0"
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold =     ↙
    False
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
        End With
        concrete_plot.Refresh()

        xlConcrete.ChartObjects(1).chart.Export(FILEname:=FolderPath & "\temp" & "\      ↙
    results_CONC" & DUMss & ".bmp", FilterName:="BMP")
        plotSS.BackgroundImage = New System.Drawing.Bitmap(FolderPath & "\temp" & "\     ↙
    results_CONC" & DUMss & ".bmp")
        plotSS.BackgroundImageLayout = ImageLayout.Stretch

        GoTo 999
```

```
        '**STEEL*****************************************************************
    ***********
2:      xlSteel.Range("A12:C26").Clear()
        SStempCOUNT = lstSS_temps.SelectedItems.Count
        xlSteel.Cells(13, 1).value = SStempCOUNT
        SStempCOUNT = SStempCOUNT - 1
        If SStempCOUNT = -1 Then GoTo 999

        For i = 0 To SStempCOUNT
            xlSteel.Cells(13 + i, 2).value = lstSS_temps.SelectedItems.Item(i)
            If lstSS_temps.SelectedItems.Item(i) = "20C" Then xlSteel.Cells(13 + i, 3).
    value = 1
            If lstSS_temps.SelectedItems.Item(i) = "100C" Then xlSteel.Cells(13 + i, 3).
    value = 2
            If lstSS_temps.SelectedItems.Item(i) = "200C" Then xlSteel.Cells(13 + i, 3).
    value = 3
            If lstSS_temps.SelectedItems.Item(i) = "300C" Then xlSteel.Cells(13 + i, 3).
    value = 4
            If lstSS_temps.SelectedItems.Item(i) = "400C" Then xlSteel.Cells(13 + i, 3).
    value = 5
            If lstSS_temps.SelectedItems.Item(i) = "500C" Then xlSteel.Cells(13 + i, 3).
    value = 6
            If lstSS_temps.SelectedItems.Item(i) = "600C" Then xlSteel.Cells(13 + i, 3).
    value = 7
            If lstSS_temps.SelectedItems.Item(i) = "700C" Then xlSteel.Cells(13 + i, 3).
    value = 8
            If lstSS_temps.SelectedItems.Item(i) = "800C" Then xlSteel.Cells(13 + i, 3).
    value = 9
            If lstSS_temps.SelectedItems.Item(i) = "900C" Then xlSteel.Cells(13 + i, 3).
    value = 10
            If lstSS_temps.SelectedItems.Item(i) = "1000C" Then xlSteel.Cells(13 + i, 3).
    value = 11
            If lstSS_temps.SelectedItems.Item(i) = "1100C" Then xlSteel.Cells(13 + i, 3).
    value = 12
            If lstSS_temps.SelectedItems.Item(i) = "1200C" Then xlSteel.Cells(13 + i, 3).
    value = 13
        Next i

        'Plot Steel stress-strain_____________________________________________
        xlSteel.ChartObjects.Delete()

        Dim Steel_plot As Excel.Chart
        Dim Steel_SerCol As Excel.SeriesCollection
        Dim Steel_Series(0 To 13) As Excel.Series
        Dim Steel_AxisCategory, Steel_AxisValue As Excel.Axes

        xlCharts = xlSteel.ChartObjects
        myChart = xlCharts.Add(180, 260, 1000, 600)
        Steel_plot = myChart.Chart
        Steel_plot.ChartType = Excel.XlChartType.xlXYScatterLinesNoMarkers
        Steel_plot.ChartStyle = 3
        Steel_SerCol = Steel_plot.SeriesCollection

        SSplotCOUNT = -1

        For i = 1 To 13
            For j = 0 To 12
                If xlSteel.Cells(13 + j, 3).value = i Then GoTo 200
            Next j
            GoTo 201
200:        SSplotCOUNT = SSplotCOUNT + 1
            Steel_Series(SSplotCOUNT) = Steel_SerCol.NewSeries
            Steel_Series(SSplotCOUNT).Name = temp_TEMP(i - 1) & "C"
            Steel_Series(SSplotCOUNT).XValues = xlSteel.Range(xSS_col(i - 1) & "2:" &
    xSS_col(i - 1) & "101")
```

```
            Steel_Series(SSplotCOUNT).Values = xlSteel.Range(ySS_col(i - 1) & "2:" & ySS_col(i - 1) & "101")
            Steel_Series(SSplotCOUNT).Format.Line.Weight = 1.5
201:    Next i

        With Steel_plot
            .Legend.Position = Excel.XlLegendPosition.xlLegendPositionTop
            .HasTitle = True
            .ChartTitle.Text = "Steel Stress-Strain Relationship"

            Steel_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = True
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Text() = "Strain, mm/mm"
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Font.Bold = False
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Font.Size = 14
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines = True
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines = True
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format.Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScale = 0
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MaximumScale = 0.2
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MajorUnitIsAuto = True
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition = Excel.XlTickLabelPosition.xlTickLabelPositionLow
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.NumberFormat = "0.000"
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold = False
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size = 14

            Steel_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Text() = "Stress, MPa"
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Bold = False
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Size = 14
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScale = 0
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MaximumScaleIsAuto = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MajorUnitIsAuto = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.XlTickLabelPosition.xlTickLabelPositionLow
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.NumberFormat = "0"
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold = False
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
        End With

        Steel_plot.Refresh()

        xlSteel.ChartObjects(1).chart.Export(FILEname:=FolderPath & "\temp" & "\results_REINF" & DUMss & ".bmp", FilterName:="BMP")
        plotSS.BackgroundImage = New System.Drawing.Bitmap(FolderPath & "\temp" & "\results_REINF" & DUMss & ".bmp")
        plotSS.BackgroundImageLayout = ImageLayout.Stretch

999:
    End Sub
```

```
    Private Sub lstSS_temps_SelectedIndexChanged(ByVal sender As System.Object, ByVal e
    As System.EventArgs) Handles lstSS_temps.SelectedIndexChanged

        If dontRefreshSS = 1 Then GoTo 999
        DUMss = DUMss + 1

        INITIAL = INITIAL + 1
        If INITIAL = 1 Then GoTo 999

        If radConcrete.Checked = True Then GoTo 1
        If radSteel.Checked = True Then GoTo 2
        GoTo 999

        '**CONCRETE*****************************************************************
    ***************
1:      xlConcrete.Range("A12:C26").Clear()
        SStempCOUNT = lstSS_temps.SelectedItems.Count
        xlConcrete.Cells(13, 1).value = SStempCOUNT
        SStempCOUNT = SStempCOUNT - 1
        If SStempCOUNT = -1 Then GoTo 999

        For i = 0 To SStempCOUNT
            xlConcrete.Cells(13 + i, 2).value = lstSS_temps.SelectedItems.Item(i)
            If lstSS_temps.SelectedItems.Item(i) = "20C" Then xlConcrete.Cells(13 + i, 3)
    .value = 1
            If lstSS_temps.SelectedItems.Item(i) = "100C" Then xlConcrete.Cells(13 + i,
    3).value = 2
            If lstSS_temps.SelectedItems.Item(i) = "200C" Then xlConcrete.Cells(13 + i,
    3).value = 3
            If lstSS_temps.SelectedItems.Item(i) = "300C" Then xlConcrete.Cells(13 + i,
    3).value = 4
            If lstSS_temps.SelectedItems.Item(i) = "400C" Then xlConcrete.Cells(13 + i,
    3).value = 5
            If lstSS_temps.SelectedItems.Item(i) = "500C" Then xlConcrete.Cells(13 + i,
    3).value = 6
            If lstSS_temps.SelectedItems.Item(i) = "600C" Then xlConcrete.Cells(13 + i,
    3).value = 7
            If lstSS_temps.SelectedItems.Item(i) = "700C" Then xlConcrete.Cells(13 + i,
    3).value = 8
            If lstSS_temps.SelectedItems.Item(i) = "800C" Then xlConcrete.Cells(13 + i,
    3).value = 9
            If lstSS_temps.SelectedItems.Item(i) = "900C" Then xlConcrete.Cells(13 + i,
    3).value = 10
            If lstSS_temps.SelectedItems.Item(i) = "1000C" Then xlConcrete.Cells(13 + i,
    3).value = 11
            If lstSS_temps.SelectedItems.Item(i) = "1100C" Then xlConcrete.Cells(13 + i,
    3).value = 12
            If lstSS_temps.SelectedItems.Item(i) = "1200C" Then xlConcrete.Cells(13 + i,
    3).value = 13
        Next i

        'Plot concrete stress-strain_____________________________________________
        xlConcrete.ChartObjects.Delete()

        Dim concrete_plot As Excel.Chart
        Dim xlCharts As Excel.ChartObjects
        Dim myChart As Excel.ChartObject
        Dim concrete_SerCol As Excel.SeriesCollection
        Dim concrete_Series(0 To 13) As Excel.Series
        Dim concrete_AxisCategory, concrete_AxisValue As Excel.Axes

        xlCharts = xlConcrete.ChartObjects
        myChart = xlCharts.Add(180, 260, 1000, 600)
        concrete_plot = myChart.Chart
        concrete_plot.ChartType = Excel.XlChartType.xlXYScatterLinesNoMarkers
        concrete_plot.ChartStyle = 3
```

```
        concrete_SerCol = concrete_plot.SeriesCollection

        SSplotCOUNT = -1

        For i = 1 To 13
            For j = 0 To 12
                If xlConcrete.Cells(13 + j, 3).value = i Then GoTo 100
            Next j
            GoTo 101
100:        SSplotCOUNT = SSplotCOUNT + 1
            concrete_Series(SSplotCOUNT) = concrete_SerCol.NewSeries
            concrete_Series(SSplotCOUNT).Name = temp_TEMP(i - 1) & "C"
            concrete_Series(SSplotCOUNT).XValues = xlConcrete.Range(xSS_col(i - 1) & "2:"↙
     & xSS_col(i - 1) & "101")
            concrete_Series(SSplotCOUNT).Values = xlConcrete.Range(ySS_col(i - 1) & "2:" ↙
    & ySS_col(i - 1) & "101")
            concrete_Series(SSplotCOUNT).Format.Line.Weight = 1.5
101:    Next i

        With concrete_plot
            .Legend.Position = Excel.XlLegendPosition.xlLegendPositionTop
            .HasTitle = True
            .ChartTitle.Text = "Concrete Stress-Strain Relationship"

            concrete_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.↙
    Text() = "Strain, mm/mm"
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.↙
    Font.Bold = False
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.↙
    Font.Size = 14
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines =  ↙
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines =  ↙
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format↙
    .Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScaleIsAuto = ↙
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MaximumScaleIsAuto = ↙
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).MajorUnitIsAuto =    ↙
    True
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition =  ↙
    Excel.XlTickLabelPosition.xlTickLabelPositionLow
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.          ↙
    NumberFormat = "0.000"
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold ↙
    = False
            concrete_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size ↙
    = 14

            concrete_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Text()↙
     = "Stress, MPa"
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font. ↙
    Bold = False
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font. ↙
    Size = 14
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line.↙
    DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScaleIsAuto = True
```

```
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MaximumScaleIsAuto = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).MajorUnitIsAuto = True
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.
    XlTickLabelPosition.xlTickLabelPositionLow
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.NumberFormat =
    "0"
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold =
    False
            concrete_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
        End With
        concrete_plot.Refresh()

        xlConcrete.ChartObjects(1).chart.Export(FILEname:=FolderPath & "\temp" & "\
    results_CONC" & DUMss & ".bmp", FilterName:="BMP")
        plotSS.BackgroundImage = New System.Drawing.Bitmap(FolderPath & "\temp" & "\
    results_CONC" & DUMss & ".bmp")
        plotSS.BackgroundImageLayout = ImageLayout.Stretch

        GoTo 999

        '**STEEL*********************************************************************
    ************
2:      xlSteel.Range("A12:C26").Clear()
        SStempCOUNT = lstSS_temps.SelectedItems.Count
        xlSteel.Cells(13, 1).value = SStempCOUNT
        SStempCOUNT = SStempCOUNT - 1
        If SStempCOUNT = -1 Then GoTo 999

        For i = 0 To SStempCOUNT
            xlSteel.Cells(13 + i, 2).value = lstSS_temps.SelectedItems.Item(i)
            If lstSS_temps.SelectedItems.Item(i) = "20C" Then xlSteel.Cells(13 + i, 3).
    value = 1
            If lstSS_temps.SelectedItems.Item(i) = "100C" Then xlSteel.Cells(13 + i, 3).
    value = 2
            If lstSS_temps.SelectedItems.Item(i) = "200C" Then xlSteel.Cells(13 + i, 3).
    value = 3
            If lstSS_temps.SelectedItems.Item(i) = "300C" Then xlSteel.Cells(13 + i, 3).
    value = 4
            If lstSS_temps.SelectedItems.Item(i) = "400C" Then xlSteel.Cells(13 + i, 3).
    value = 5
            If lstSS_temps.SelectedItems.Item(i) = "500C" Then xlSteel.Cells(13 + i, 3).
    value = 6
            If lstSS_temps.SelectedItems.Item(i) = "600C" Then xlSteel.Cells(13 + i, 3).
    value = 7
            If lstSS_temps.SelectedItems.Item(i) = "700C" Then xlSteel.Cells(13 + i, 3).
    value = 8
            If lstSS_temps.SelectedItems.Item(i) = "800C" Then xlSteel.Cells(13 + i, 3).
    value = 9
            If lstSS_temps.SelectedItems.Item(i) = "900C" Then xlSteel.Cells(13 + i, 3).
    value = 10
            If lstSS_temps.SelectedItems.Item(i) = "1000C" Then xlSteel.Cells(13 + i, 3).
    value = 11
            If lstSS_temps.SelectedItems.Item(i) = "1100C" Then xlSteel.Cells(13 + i, 3).
    value = 12
            If lstSS_temps.SelectedItems.Item(i) = "1200C" Then xlSteel.Cells(13 + i, 3).
    value = 13
        Next i

        'Plot Steel stress-strain_____________________________________________
        xlSteel.ChartObjects.Delete()

        Dim Steel_plot As Excel.Chart
        Dim Steel_SerCol As Excel.SeriesCollection
        Dim Steel_Series(0 To 13) As Excel.Series
        Dim Steel_AxisCategory, Steel_AxisValue As Excel.Axes
```

```
        xlCharts = xlSteel.ChartObjects
        myChart = xlCharts.Add(180, 260, 1000, 600)
        Steel_plot = myChart.Chart
        Steel_plot.ChartType = Excel.XlChartType.xlXYScatterLinesNoMarkers
        Steel_plot.ChartStyle = 3
        Steel_SerCol = Steel_plot.SeriesCollection

        SSplotCOUNT = -1

        For i = 1 To 13
            For j = 0 To 12
                If xlSteel.Cells(13 + j, 3).value = i Then GoTo 200
            Next j
            GoTo 201
200:        SSplotCOUNT = SSplotCOUNT + 1
            Steel_Series(SSplotCOUNT) = Steel_SerCol.NewSeries
            Steel_Series(SSplotCOUNT).Name = temp_TEMP(i - 1) & "C"
            Steel_Series(SSplotCOUNT).XValues = xlSteel.Range(xSS_col(i - 1) & "2:" &
    xSS_col(i - 1) & "101")
            Steel_Series(SSplotCOUNT).Values = xlSteel.Range(ySS_col(i - 1) & "2:" &
    ySS_col(i - 1) & "101")
            Steel_Series(SSplotCOUNT).Format.Line.Weight = 1.5
201:    Next i

        With Steel_plot
            .Legend.Position = Excel.XlLegendPosition.xlLegendPositionTop
            .HasTitle = True
            .ChartTitle.Text = "Steel Stress-Strain Relationship"

            Steel_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = True
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.
    Text() = "Strain, mm/mm"
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.
    Font.Bold = False
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.
    Font.Size = 14
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines = True
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines = True
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format.
    Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScale = 0
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MaximumScale = 0.2
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).MajorUnitIsAuto = True
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition =
    Excel.XlTickLabelPosition.xlTickLabelPositionLow
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.NumberFormat
    = "0.000"
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold =
    False
            Steel_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size =
    14

            Steel_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Text() =
    "Stress, MPa"
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Bold
     = False
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Size
     = 14
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line.
    DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScale = 0
```

```
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MaximumScaleIsAuto = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).MajorUnitIsAuto = True
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.
    XlTickLabelPosition.xlTickLabelPositionLow
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.NumberFormat = "0"
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold = False
            Steel_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
        End With

        Steel_plot.Refresh()

        xlSteel.ChartObjects(1).chart.Export(FILEname:=FolderPath & "\temp" & "\
    results_REINF" & DUMss & ".bmp", FilterName:="BMP")
        plotSS.BackgroundImage = New System.Drawing.Bitmap(FolderPath & "\temp" & "\
    results_REINF" & DUMss & ".bmp")
        plotSS.BackgroundImageLayout = ImageLayout.Stretch

999:

    End Sub
    Private Sub btnSelectAll_SS_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles btnSelectAll_SS.Click

        dontRefreshSS = 1
        lstSS_temps.SelectedItems.Clear()
        For i = 0 To 11
            lstSS_temps.SelectedItem = lstSS_temps.Items.Item(i)
        Next i
        dontRefreshSS = 0
        lstSS_temps.SelectedItem = lstSS_temps.Items.Item(12)

    End Sub
    Private Sub btnClearAll_SS_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles btnClearAll_SS.Click

        dontRefreshSS = 1
        lstSS_temps.SelectedItems.Clear()
        dontRefreshSS = 0
        lstSS_temps.SelectedItem = lstSS_temps.Items.Item(0)

    End Sub



    'MOMENT-CURVATURE
    Private Sub lstMPHI_step_SelectedIndexChanged(ByVal sender As System.Object, ByVal e
    As System.EventArgs) Handles lstMPHI_step.SelectedIndexChanged

        If dontRefresh = 1 Then GoTo 999
        DUM = DUM + 1

        xlMPhi.Range("A8:B1008").Clear()
        mphiCOUNT = lstMPHI_step.SelectedItems.Count
        xlMPhi.Cells(8, 1).value = mphiCOUNT
        mphiCOUNT = mphiCOUNT - 1
        If mphiCOUNT = -1 Then GoTo 999

        For i = 0 To mphiCOUNT
            xlMPhi.Cells(9 + i, 1).value = lstMPHI_step.SelectedItems.Item(i)
            xlMPhi.Cells(9 + i, 2).value = MPhi_Index(lstMPHI_step.SelectedItems.Item(i))
        Next i

        'Plot
        xlMPhi.ChartObjects.delete()

        Dim xlCharts As Excel.ChartObjects
```

```
        Dim myChart As Excel.ChartObject
        Dim MPHIplot As Excel.Chart
        Dim MPHI_SerCol As Excel.SeriesCollection
        Dim MPhi_Series(0 To 240) As Excel.Series
        Dim MPHI_AxisCategory, MPHI_AxisValue As Excel.Axes

        xlCharts = xlMPhi.ChartObjects
        myChart = xlCharts.Add(306, 100, 900, 600)
        MPHIplot = myChart.Chart
        MPHIplot.ChartType = Excel.XlChartType.xlXYScatterLinesNoMarkers
        MPHIplot.ChartStyle = 3
        MPHI_SerCol = MPHIplot.SeriesCollection

        For i = 0 To mphiCOUNT
            MPhi_Series(i) = MPHI_SerCol.NewSeries
            MPhi_Series(i).Name = xlMPhi.Cells(1, 3 + 4 * MPhi_Index(lstMPHI_step.
    SelectedItems.Item(i))).value
            MPhi_Series(i).XValues = xlMPhi.Range(x_col(MPhi_Index(lstMPHI_step.
    SelectedItems.Item(i))) & "2:" & x_col(MPhi_Index(lstMPHI_step.SelectedItems.Item
    (i))) & mphiPLOTpoints + 2)
            MPhi_Series(i).Values = xlMPhi.Range(y_col(MPhi_Index(lstMPHI_step.
    SelectedItems.Item(i))) & "2:" & y_col(MPhi_Index(lstMPHI_step.SelectedItems.Item
    (i))) & mphiPLOTpoints + 2)
            MPhi_Series(i).Format.Line.Weight = 1.5
        Next i

        With MPHIplot
            .Legend.Position = Excel.XlLegendPosition.xlLegendPositionTop
            .HasTitle = True
            .ChartTitle.Text = "Moment-Curvature"

            MPHI_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
            MPHI_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = True
            MPHI_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Text
    () = "Curvature, rad/mm"
            MPHI_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Font
    .Bold = False
            MPHI_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Font
    .Size = 14
            MPHI_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines = True
            MPHI_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines = True
            MPHI_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format.
    Line.DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            MPHI_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScaleIsAuto = True
            MPHI_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition = Excel
    .XlTickLabelPosition.xlTickLabelPositionLow
            MPHI_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold =
    False
            MPHI_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size = 14

            MPHI_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
            MPHI_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = True
            MPHI_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Text() =
    "Moment, kN-m"
            MPHI_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Bold
    = False
            MPHI_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Size
    = 14
            MPHI_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
            MPHI_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
            MPHI_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line.
    DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
            MPHI_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScaleIsAuto = True
            MPHI_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.
    XlTickLabelPosition.xlTickLabelPositionLow
            MPHI_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold = False
```

```
            MPHI_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
        End With
        MPHIplot.Refresh()

        xlMPhi.ChartObjects(1).chart.Export(FILEname:=FolderPath & "\temp" & "\
    results_mphi" & DUM & ".bmp", FilterName:="BMP")
        plotMPHI.BackgroundImage = New System.Drawing.Bitmap(FolderPath & "\temp" & "\
    results_mphi" & DUM & ".bmp")
        plotMPHI.BackgroundImageLayout = ImageLayout.Stretch

999:
    End Sub
    Private Sub btnSelectAll_MPhi_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles btnSelectAll_MPhi.Click

        dontRefresh = 1
        lstMPHI_step.SelectedItems.Clear()
        For i = 0 To timesteps - 1
            lstMPHI_step.SelectedItem = lstMPHI_step.Items.Item(i)
        Next i

        dontRefresh = 0
        lstMPHI_step.SelectedItem = lstMPHI_step.Items.Item(timesteps)


    End Sub
    Private Sub btnClearAll_MPhi_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles btnClearAll_MPhi.Click

        dontRefresh = 1
        lstMPHI_step.SelectedItems.Clear()

        dontRefresh = 0
        lstMPHI_step.SelectedItem = lstMPHI_step.Items.Item(0)

    End Sub



    'MOMENT-AXIAL
    Private Sub lstMN_step_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
     System.EventArgs) Handles lstMN_step.SelectedIndexChanged

        If dontRefreshMN = 1 Then GoTo 999
        DUMmn = DUMmn + 1

        xlMN.Range("A8:B1008").Clear()
        mnCOUNT = lstMN_step.SelectedItems.Count
        xlMN.Cells(8, 1).value = mnCOUNT
        mnCOUNT = mnCOUNT - 1
        If mnCOUNT = -1 Then GoTo 999

        For i = 0 To mnCOUNT
            xlMN.Cells(9 + i, 1).value = lstMN_step.SelectedItems.Item(i)
            xlMN.Cells(9 + i, 2).value = MN_Index(lstMN_step.SelectedItems.Item(i))
        Next i

        'Plot
        xlMN.ChartObjects.delete()

        Dim xlCharts As Excel.ChartObjects
        Dim myChart As Excel.ChartObject
        Dim MNplot As Excel.Chart
        Dim MN_SerCol As Excel.SeriesCollection
        Dim MN_Series(0 To 240) As Excel.Series
        Dim MN_AxisCategory, MN_AxisValue As Excel.Axes
```

```
    xlCharts = xlMN.ChartObjects
    myChart = xlCharts.Add(306, 100, 900, 600)
    MNplot = myChart.Chart
    MNplot.ChartType = Excel.XlChartType.xlXYScatterLinesNoMarkers
    MNplot.ChartStyle = 3
    MN_SerCol = MNplot.SeriesCollection

    For i = 0 To mnCOUNT
        MN_Series(i) = MN_SerCol.NewSeries
        MN_Series(i).Name = xlMN.Cells(1, 3 + 4 * MN_Index(lstMN_step.SelectedItems. ↙
Item(i))).value
        MN_Series(i).XValues = xlMN.Range(x_col(MN_Index(lstMN_step.SelectedItems.   ↙
Item(i))) & "2:" & x_col(MN_Index(lstMN_step.SelectedItems.Item(i))) & 44)
        MN_Series(i).Values = xlMN.Range(y_col(MN_Index(lstMN_step.SelectedItems.Item↙
(i))) & "2:" & y_col(MN_Index(lstMN_step.SelectedItems.Item(i))) & 44)
        MN_Series(i).Format.Line.Weight = 1.5
    Next i

    With MNplot
        .Legend.Position = Excel.XlLegendPosition.xlLegendPositionTop
        .HasTitle = True
        .ChartTitle.Text = "Moment-Axial Interaction"

        MN_AxisCategory = .Axes(, Excel.XlAxisGroup.xlPrimary)
        MN_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasTitle = True
        MN_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Text()↙
 = "Moment, kN-m"
        MN_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Font. ↙
Bold = False
        MN_AxisCategory.Item(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Font. ↙
Size = 14
        MN_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMajorGridlines = True
        MN_AxisCategory.Item(Excel.XlAxisType.xlCategory).HasMinorGridlines = True
        MN_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinorGridlines.Format.Line.↙
DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
        MN_AxisCategory.Item(Excel.XlAxisType.xlCategory).MinimumScaleIsAuto = True
        MN_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabelPosition = Excel. ↙
XlTickLabelPosition.xlTickLabelPositionLow
        MN_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Bold =     ↙
False
        MN_AxisCategory.Item(Excel.XlAxisType.xlCategory).TickLabels.Font.Size = 14

        MN_AxisValue = .Axes(, Excel.XlAxisGroup.xlPrimary)
        MN_AxisValue.Item(Excel.XlAxisType.xlValue).HasTitle = True
        MN_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Text() =    ↙
"Axial, kN"
        MN_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Bold = ↙
False
        MN_AxisValue.Item(Excel.XlAxisType.xlValue).AxisTitle.Characters.Font.Size = ↙
14
        MN_AxisValue.Item(Excel.XlAxisType.xlValue).HasMajorGridlines = True
        MN_AxisValue.Item(Excel.XlAxisType.xlValue).HasMinorGridlines = True
        MN_AxisValue.Item(Excel.XlAxisType.xlValue).MinorGridlines.Format.Line.      ↙
DashStyle = Microsoft.Office.Core.MsoLineDashStyle.msoLineDash
        MN_AxisValue.Item(Excel.XlAxisType.xlValue).MinimumScaleIsAuto = True
        MN_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabelPosition = Excel.       ↙
XlTickLabelPosition.xlTickLabelPositionLow
        MN_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Bold = False
        MN_AxisValue.Item(Excel.XlAxisType.xlValue).TickLabels.Font.Size = 14
    End With
    MNplot.Refresh()


    xlMN.ChartObjects(1).chart.Export(FILEname:=FolderPath & "\temp" & "\results_mn" ↙
& DUMmn & ".bmp", FilterName:="BMP")
```

```
        plotMN.BackgroundImage = New System.Drawing.Bitmap(FolderPath & "\temp" & "\results_mn" & DUMmn & ".bmp")
        plotMN.BackgroundImageLayout = ImageLayout.Stretch

999:

    End Sub
    Private Sub btnSelecetAll_MN_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnSelecetAll_MN.Click

        dontRefreshMN = 1
        lstMN_step.SelectedItems.Clear()
        For i = 0 To timesteps - 1
            lstMN_step.SelectedItem = lstMN_step.Items.Item(i)
        Next i

        dontRefreshMN = 0
        lstMN_step.SelectedItem = lstMN_step.Items.Item(timesteps)

    End Sub
    Private Sub btnClearAll_MN_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnClearAll_MN.Click

        dontRefreshMN = 1
        lstMN_step.SelectedItems.Clear()

        dontRefreshMN = 0
        lstMN_step.SelectedItem = lstMN_step.Items.Item(0)

    End Sub



    'MENU_STRIP
    Private Sub NewAnalysisToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles NewAnalysisToolStripMenuItem.Click

    End Sub
    Private Sub SaveResultsToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles SaveResultsToolStripMenuItem.Click

        xlUTFire.SaveAs(FilePath)

    End Sub
    Private Sub SaveAsToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles SaveAsToolStripMenuItem.Click

        xlUTFire.SaveAs(FilePath)

    End Sub
    Private Sub ExitToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ExitToolStripMenuItem.Click

        Me.Close()

    End Sub
    Private Sub AboutToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles AboutToolStripMenuItem.Click
        Dim About As New About
        About.ShowDialog()
        About.Refresh()
    End Sub
    Private Sub HelpToolStripMenuItem1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles HelpToolStripMenuItem1.Click
        Dim Help As New Help
        Help.ShowDialog()
```

```
        Help.Refresh()
    End Sub


    'SYTEM_SUBROUTINES+FUNCTIONS
    Private Sub releaseObject(ByVal obj As Object)
        Try
            System.Runtime.InteropServices.Marshal.ReleaseComObject(obj)
            obj = Nothing
        Catch ex As Exception
            obj = Nothing
        Finally
            GC.Collect()
        End Try
    End Sub




End Class
```

References Cited

1. ACI (2007). "Code Requirements for Determining Fire Resistance of Concrete and Masonry Construction Assemblies," American Concrete Institute 216.1-07, Detroit, MI.

2. ACI (2008). "Building Code Requirements for Structural Concrete and Commentary," American Concrete Institute 318-08, Detroit, MI.

3. ASTM (2008). "Standard Test Methods for Fire Tests of Building Construction Materials," *Standard No. E119-00a*, American Society for Testing and Materials, West Conshohocken, PA.

4. Buchanan, A.H. (2002). "Structural Design for Fire Safety," John Wiley and Sons, West Sussex, England.

5. Cadorin, J.F., Pintea, D., Franssen, J.M. (2001). "The Design Fire Tool OZone v2.0 - Theoretical Description and Validation on Experimental Fire Tests," University of Liege, Belgium.

6. Drysdale, Douglas (1998). "An Introduction to Fire Dynamics," 2nd Edition, Wiley.

7. Engelhardt, M. (2008). "CE 397 - Structural Fire Engineering Course Notes," The University of Texas at Austin.

8. Eurocode 2 (2004). "Eurocode 2: Design of Concrete Structures - Part 1-2: General Rules - Structural Fire Design." *BS EN 1992-1-2:2004*, British Standards Institute.

9. Franssen, J.M. (2007). "SAFIR2007. A Thermal/Structural Program Modeling Structures Under Fire, Version 2007," University of Liege, Belgium.

10. Franssen, J.M. (2007). "Diamond User's Guide, Version 2007," University of Liege, Belgium.

11. ISO (1975). "Fire Resistance Tests - Elements of Building Construction," *ISO 834-1975*. International Organization for Standardization.

12. Jennings, T. (2009). "UTFIRE, a Preprocessor for SAFIR2007, for Analysis of Heat Transfer for Structural Members Exposed to Fire." *MS Thesis*, Department of Civil, Architectural, and Environmental Engineering, University of Texas at Austin.

13. Karlsson, Bjorn and Quintiere, James (2000). “Enclosure Fire Dynamics,” CRC Press.

14. Lie (1995). "Fire Temperature-Time Relations," Chapter 4-8. *SFPE Handbook of Fire Protection Engineering, Second Edition.* Society of Fire Protection Engineers, USA.

15. McGrattan, K., McDermott, R., Hostikka, S. and Floyd, J. (2010). "Fire Dynamics Simulator (Version 5) User's Guide," *NIST Special Publication 1019-5,* National Institute of Standards and Technology, Washington, D.C.

16. Meacham, B., Engelhardt, M. and Kodur, V. (2009). "Collection of Data on Fire and Collapse, Faculty of Architecture Building, Delft University of Technology," *National Science Foundation, Division on Civil, Mechanical, and Manufacturing Innovation, Research and Innovation Conference 2009*, 22-25 June 2009.

17. Meacham, B. et. al. (2010). "Fire and Collapse, Faculty of Architecture Building, Delft University of Technology: Implications for Performance Based Design," Worchester Polytechnic Institute.

18. Meacham, B., Park, H., Engelhardt, M., Kirk, A., Kodur, V., van Straalen, I., Maljaars, J., van Weeren, K., de Feijter, R. and Both, K. (2010). “Fire and Collapse, Faculty of Architecture Building, Delft University of Technology: Data Collection and Preliminary Analysis.” *Proceedings, 8th International Conference on Performance-Based Codes and Fire Safety Design Methods*. Lund University, Sweden, June 16-18, 2010.

19. Pettersson, O., Magnusson, S.E., and Thor, J. "Fire Engineering Design of Steel Structures." *Swedish Institute of Steel Construction, Publication No. 50*, Stockholm, 1976.

20. Phan, L.T., McAllister, T.P., Gross, J.L. and Hurley, M.J. (2009). “Best Practice Guidelines for Structural Fire Resistance Design of Concrete and Steel Buildings, Report NISTIR 7563 (draft), National Institute of Standards and Technology, Gaithersburg, MD.

21. Purkiss, J.A. (2007). "Fire Safety Engineering: Design of Structures," Elsevier, Oxford, England.

22. SFPE (2002). “The SFPE Handbook of Fire Protection Engineering,” Third Edition, Society of Fire Protection Engineers, Bethesda, Maryland.

23. TU Delft (2008). “Bouwkunde – Portrait of the Faculty of Architecture of Delft University of Technology, 1970-2008).” Delft University of Technology, Delft, Netherlands.

24. Van Weeren, K. (2009). "The fire at Delft University of technology, faculty of Architecture on 13th May 2008," Delft University of Technology, the Netherlands.

25. Wade, C. (2004). "BRANZFIRE Technical Reference Guide," *Branz Study Report No. 92*, Building Research Association, Judgeford, New Zealand.

26. Wang, Y.C. (2002). “Steel and Composite Structures; Behaviour and Design for Fire Safety.” Spon Press, London.

VITA

Adam Jess Kirk was born to Mark and Terri Kirk on October 23, 1984 in Oklahoma City, OK. He grew up in Edmond, OK and graduated from Edmond Memorial High School in 2003 before attending Oklahoma State University. He received a Bachelor of Architecture Engineering degree from OSU in May of 2008. After a summer internship at Frankfurt Short Bruza, an architecture and engineering firm in Oklahoma City, he began his graduate studies in structural engineering at the University of Texas at Austin. During this time, he worked as both a research assistant for Dr. Michael Engelhardt (in the area of structural fire engineering) and a laboratory instructor for Civil Engineering Systems, the introduction to CE course for entering freshmen. After completing his coursework and thesis at UT, he will move to New York City to begin a career at Leslie E. Robertson Associates, a structural consulting firm best known for its design of the original World Trade Center towers.

Permanent Address: *809 Fox Bend Trail*
Edmond, OK 73034
adam.kirk@mail.com

This thesis was typed by the author.