FINITE ELEMENT ANALYSIS OF UNREINFORCED CONCRETE BLOCK WALLS SUBJECT TO OUT-OF-PLANE LOADING

FINITE ELEMENT ANALYSIS OF UNREINFORCED CONCRETE BLOCK WALLS SUBJECT TO OUT-OF-PLANE LOADING

By

ZHONG HE, B.Eng., B.Sc.

A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree
Master of Applied Science

McMaster University

MASTER OF APPLIED SCIENCE (2005) McMaster University
(Civil Engineering) Hamilton, Ontario

TITLE: Finite Element Analysis of Unreinforced Concrete Block Walls Subject to Out-of-Plane Loading

AUTHOR: Zhong He, B.Eng. (Wuhan University of Technology),
B.Sc. (McMaster University)

SUPERVISORS: Professor Samir E. Chidiac
Professor Robert G. Drysdale

NUMBER OF PAGES: xiv, 366

ABSTRACT

Finite element modeling of the structural response of hollow concrete block walls subject to out-of-plane loading has become more common given the availability of computers and general-purpose finite element software packages. In order to develop appropriate models of full-scale walls with and without openings, a parametric study was conducted on simple wall elements to assess different modeling techniques. Two approaches were employed in the study, homogeneous models and heterogeneous models. The linear elastic analysis was carried out to quantify the effects of the modeling techniques for hollow blocks on the structural response of the assembly, specifically for out-of-plane bending. Three structural elements with varying span/thickness ratios were considered, a horizontal spanning strip, a vertical spanning strip and a rectangular wall panel supported on four edges. The values computed using homogeneous and heterogeneous finite element models were found to differ significantly depending on the configuration and span/thickness ratio of the wall.

Further study was carried out through discrete modeling approach to generate a three-dimensional heterogeneous model to investigate nonlinear behaviour of full-scale walls under out-of-plane loading. The Composite Interface Model, established based on multi-surface plasticity, which is capable of describing both tension and shear failure mechanisms, has been incorporated into the analysis to capture adequately the inelastic behaviour of unit-mortar interface.

An effective solution procedure was achieved by implementing the Newton-Raphson method, constrained with the arc-length control method and enhanced by line search algorithm. The proposed model was evaluated using experimental results for ten full-size walls reported in the literature. The comparative analysis has indicated very good agreement between the numerical and experimental results in predicting the cracking and ultimate load values as well as the corresponding crack pattern.

ACKNOWLEDGEMENTS

The author would like to express his sincere and deepest gratitude to Dr. Samir E. Chidiac and Dr. Robert G. Drysdale for their consistent guidance and valuable advice throughout the research project, and their encouragement and support during the completion of the thesis.

The financial assistance of The Natural Science and Engineering Research Council of Canada, McMaster University's Center for Effective Design of Structures and Ontario Concrete Block Association is deeply appreciated.

Finally, a special heartfelt thank is due to my wife, Ying, for her deep understanding and continuous encouragement towards the completion of this study.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURES PAGE

LIST OF TABLES

TABLES PAGE

CHAPTER 1

INTRODUCTION

1.1 Background

Most early analytical masonry research was based on methods employing simplifying assumptions such as isotropic material and linear elastic properties. In reality, masonry is a very complex material that has distinct planes of weakness corresponding to the composite nature of mortar joints combined with masonry units. In addition, the geometric shape of the masonry units and, sometimes, the properties of the material, are orthotropic in nature. As a result, conventional elastic plate types of analyses can, at best, only approximate the actual behaviour even when macroscopic properties are introduced to attempt to account for the above inconsistencies in assumptions. The research reported in this thesis is the development and testing of a numerical analysis technique that eliminates most of the above inconsistencies.

During the past three decades, more rational analytical approaches have been attempted by many researchers. With the availability of computers and general-purpose analysis packages, numerical methods, and specifically finite element methods, have been shown to be more efficient in application to masonry engineering problems. Finite element techniques make it possible to study both linear elastic and nonlinear inelastic behaviour of masonry structures. Through modeling of the material properties at different loading stages, the critical deflection and stress and their locations can be determined and cracking of sections can be simulated. Furthermore, finite element approaches appear to be more economic compared to experimental alternatives as the former requires less cost and time. However, to ensure the success of finite element techniques, attention needs to be paid to developing a well-defined and well-validated constitutive model for the material involved.

Among numerous research topics regarding structural behaviour of masonry, the out-of-plane flexural behaviour is always of great interest. Masonry walls under the action of lateral loads from wind, earthquake or earth pressures and eccentric load may bend out-of-plane. Significant flexural tensile stress can be developed in those walls with larger span (height or length), such as the walls in industrial buildings. In recent years, the flexural strength of unreinforced masonry walls under out-of-plane loading has been of interest to many researchers. They realized that unreinforced masonry can only rely on its own tensile strength and in-plane compressive force, unlike reinforced masonry which can mostly depend on the steel to resist the tension. Unreinforced masonry is fairly sensitive to tension instead of compression and the tensile stresses distributed within the masonry structures may often dominate the design capacity.

The following literature review provides an introduction to previous research on masonry walls subjected to out-of-plane bending, including both experimental study and theoretical analysis. The finite element approach is then discussed in more details. At the end of this chapter, the objectives of this study are laid out.

1.2 Literature Review

Several investigators have dealt with the structural behaviour of masonry under out-of-plane bending over the years. Commonly, a comprehensive review of previous work has been included to summarize the available outcomes related to the flexural behaviour of masonry. Some of these review are well-classified and systematic (Baker 1981, Lawrence 1983, Essawy 1986). For the purpose of the present study, the following review focuses on the research of single-wythe unreinforced masonry walls without superimposed axial load. For concrete block masonry, researches on ungrouted walls are mainly introduced. The applications of finite element analysis in masonry structure are discussed in the last section, without limiting the topic to out-of-plane bending.

1.2.1 Experimental Investigations

1.2.1.1 Flexural Behaviour of Masonry Assemblages

An unreinforced masonry assemblage is composed of two major constituent materials, brick/block units and mortar. The distinct properties of the individual materials and their interactions are known to be important factors affecting the characteristic behaviour of masonry assemblages. In order to achieve the expected properties of the composite masonry, it is necessary to ensure that physical and mechanical properties of the component materials are properly modeled. As a crucial property controlling the flexural behaviour of masonry assemblages, flexural tensile strength is used to determine the resistance of walls to out-of-plane loadings. Assuming linear elastic behaviour for the masonry walls in one-way bending, the flexural tensile strength can be calculated based on the minimum effective area of ungrouted walls and on the gross area for grouted walls.

As masonry walls may span horizontally, vertically or in both directions depending upon the support conditions at the ends or top of the wall, tensile stresses can develop from different bending directions. Flexural tensile strength is therefore referred to as the strength normal to the mortar bed joints or parallel to bed joints. A good understanding of the corresponding flexural behaviour is essential for rational design.

Vertical Flexure

When walls are laterally supported along the top and bottom edges, tensile stresses are developed normal to the bed joints and result in debonding failure between mortar and units along the bed joint. Based on the tests for horizontally spanning wall series with two blocks long (790 mm) by eight blocks high (1590 mm), Hamid and Drysdale (1988) reported that the tensile crack formed along the minimum contact area between the bottom of the block and the mortar for ungrouted walls.

The flexural bond strength has been shown to be affected by various properties of the constituent materials and by construction conditions. Hamid and Drysdale (1988) found that compressive strength of blocks and mortar cubes, block size and bond pattern do not have a high correlation with the bond strength. Along with other investigators' experimental results, Drysdale et al. (1999) summarized that, mortar type, initial rate of absorption, moisture content and temperature of the unit, mortar strength, flow and water retentivity, workmanship and surface conditions of units and curing, can have different degrees of effect on the bond strength. They also pointed out that the relative effect of any one factor can be significantly influenced by others. However, consideration of the usually high coefficients of variation makes it fairly difficult to distinguish between the effects of various factors with a high level of confidence.

Horizontal Flexure

If walls are only simply supported along the two vertical ends, the flexural tensile stresses develop in the direction parallel to bed joints. These support conditions, would be representative of a case when material such as flashing is placed at the base. Combined with leaving the wall free at the top, this support condition would allow the wall to bend horizontally. The behaviour of masonry in horizontal flexure is more complicated than in vertical flexure. Typically, cracks initiate in the head joints, which have lower bond strength than that of the bed joints. Thus the applied moment can only be transferred between units by torsion in the bed joints. If the flexural strength of the units is not strong enough to resist the extra bending moment, failure can occur through the units along a path coinciding with the cracked head joints and this results in a line type of failure. Otherwise, the failure can be delayed until a toothed or stepped failure pattern is formed along a combination of head and bed mortar joints. For ungrouted hollow walls, the latter failure predominated although occasionally there was a mixed mode of failure. This has been demonstrated by Hamid and Drysdale (1988) in the tests of wallettes constructed in four blocks long and four blocks high.

Hamid and Drysdale (1988) also studied the factors affecting the flexural tensile strength parallel to the bed joints. They found that block strength, percent solid of block, block size, bond pattern, and type of mortar exhibit different degrees of effects. This is summarized as follows:

- For grouted block walls, the presence of grout strengthens the bed joints and thereby forces the failure to pass through the units at alternative courses. It could be expected that the tensile strength of the units have a major effect on wall strength, however, the lack of influence of block strength was in fact observed from the test results. A possible explanation has been presented by the authors, indicating that the continuity provided by the grout in the head joints also plays an important role in addition to the unit and mortar strength.

- The tensile strength parallel to the bed joints increases as the percent solid of the ungrouted walls increases. Moreover, for splitting failure mode, the increased capacity is larger than that resulted from the increase of contact area along the mortar joints. Whereas for grouted walls, the tensile strength decreases as the percent of solid increases due to the fact that the benefit of the continuity provided by the grout decreases with decreased volume of grout.

- The increase in block size can result in decreased tensile strength parallel to the bed joint for ungrouted walls when stepped or toothed failure mode is considered. Since the small increase in face shell thickness for larger blocks is much less than the increase in section modulus, the contribution of the bed joints to resist torsional shear stresses does not compensate the decreased tensile stresses at head joints resulting from the increased section modulus.
- The wall constructed with stack pattern yielded lower tensile strength in comparison to the wall with running bond. This can be expected as the failure path can be developed entirely along the continuous head joints in the walls with stack pattern. Lower mortar strength leads to lower tensile strength parallel to the bed joints but the difference is relatively small since tensile debonding on head joints is only part of the failure mechanism.

Based on the test results, Hamid and Drysdale (1988) concluded that the flexural tensile strength parallel to the bed joints tends to be several times higher than the corresponding tensile strength normal to the bed joints.

Biaxial Flexure

In practice, masonry walls are usually supported on more than two sides and bend in both horizontal and vertical directions. The interaction of the flexural tensile stresses in two directions results in more complicated biaxial flexural behaviour. As reported by investigators (Lawrence 1983, Essawy 1986), the load carrying capacity in biaxial bending can be much greater compared to one-way bending.

For walls simply supported on its two vertical sides and its base, the first crack initiates if the principal stresses at some location exceed the tensile strength at that location. As recounted by Drysdale et al. (1999), Baker and Lawrence both reported that the failure mechanism is formed right after this first crack occurs as the wall has only a small residual strength because of the stabilizing effect of self-weight.

Different height-to-length ratios (h/l) result in different failure patterns (Figure 1.1) for walls simply supported on three sides. For walls with lower h/l ratios, two diagonal stepped or splitting cracks are found to start from the top at two separated positions and extend to the left bottom and right bottom corners. In walls with higher h/l ratios, a vertical splitting or toothed crack is initiated at mid-length followed by cracks starting from the vertical crack and extend to the two bottom corners.

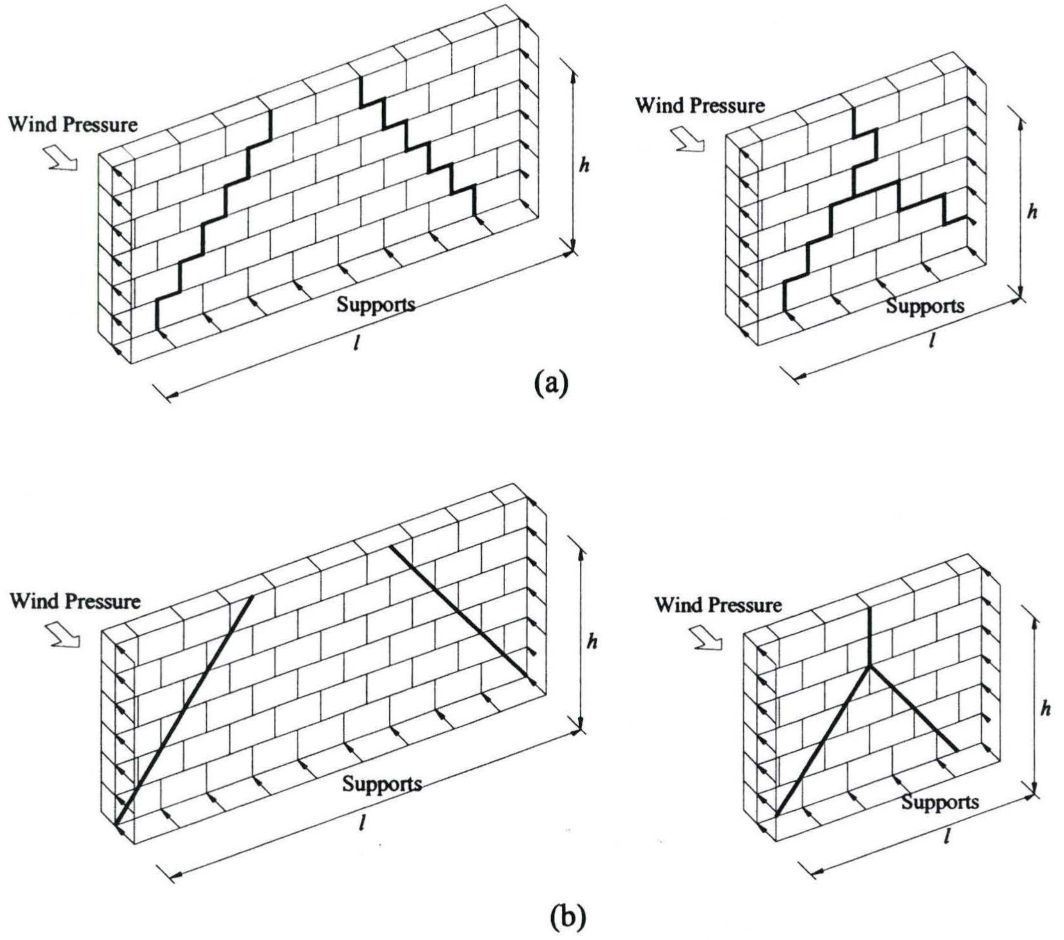


Figure 1.1 Crack patterns at failure for walls supported on three edges: (a) stepped or toothed cracks; (b) splitting cracks.

For a wall supported on all four sides and having a h/l ratio less than one, the load-deflection behaviour is linear until a horizontal crack initiates at about mid-height of the wall, when the flexural capacity of the bed joints has been reached. As mentioned above, most masonry walls have flexural tensile strength parallel to bed joints greater than that normal to bed joints. Hence this horizontal crack propagates along the bed joint under constant load and cracking remains at this state until the horizontal flexural capacity is reached. The wall is then split to two sub-panels, each of which is simply supported on three sides and free on the cracked side and behaves as described previously. A failure mechanism is formed when the cracks extend from the horizontal crack to the four corners. This typical failure phenomenon has been confirmed in many experimental programs (Baker 1981, Lawrence 1983, Essawy 1986, Chen 2002).

Orthogonal Strength Ratio and Stiffness Ratio

As a measure of the degree of orthotropy of the masonry, orthogonal strength ratio (R) is defined to be the ratio of flexural tensile strength parallel to bed joints relative to flexural tensile strength normal to the bed joints (f_{tp} / f_{tn}). The Canadian masonry code (CSA S304.4, 2004) specifies a value of 2 for this ratio. However, the ratios reported in masonry literature seem to be generally greater than the code value. Hamid and Drysdale (1988) reported an average orthogonal ratio of 3.3 from a range between 1.9 to 4.7 for 8 series of hollow block walls with mortar proportions of 1:0.5:4 (Portland cement : lime : sand).

For unreinforced masonry, unit strength, percent solid of the unit and aspect ratio of units (length vs. height) are known to be the major factors affecting the orthogonal strength ratio (Drysdale et al. 1999). When the failure occurs through the units in alternate courses, the flexural strength parallel to the bed joints will be affected by the unit strength. An increased percent solid of the unit results in increased unit tensile capacity. Higher aspect ratio of units indicates increased length of the failure path for the toothed failure mode under flexural tensile stresses parallel to bed joints; as such the tensile strength parallel to bed joints will increase as the orthogonal strength ratio increases.

Another measurement of the degree of orthotropy is the orthogonal stiffness ratio, which is defined as the ratio of modulus of elasticity parallel to the bed joint to modulus of elasticity normal to the bed joint. MSJC/ACI/ASCE/TMS (1999) requires the modulus of elasticity be defined as the chord modulus for a line drawn down from the stress-strain curve at 5% of the maximum compressive stress to 33% of the maximum compressive stress; but traditionally the modulus of elasticity is calculated using the equation $E_m = kf_m^{'}$, where k=750 to 1000 and $f_m^{'}$ is the specified compressive strength (Drysdale et al. 1999). Due to the variations in test methods, the orthogonal stiffness ratios reported in the literature are significantly scattered. As recounted by Essawy (1986), Hendry and Sinha reported that the values ranged from 0.6 to 1.4 for both brick and block masonry.

1.2.1.2 Masonry Walls Subjected to Two-way Bending

In practice, most masonry walls are supported on three or four sides and, as a result, the flexural behaviour of these walls has been of particular interest all through the history of out-of-plane loading research. Some researchers have concentrated on predicting first cracking load and some on ultimate load, but most have studied both the pre-crack and post-crack behaviour.

The intrinsic anisotropy, heterogeneous nature and high variability of material properties have caused difficulties in direct theoretical approaches. And, therefore experimental studies have been relied on to provide adequate evidence for proposing appropriate design methods.

In 1983, noticing the inconsistency of British, Australian, and North American design procedures and code provisions, Essawy and Drysdale (1983) conducted out-of-plane bending tests on 6.0m long by 2.8m high concrete block walls and compared the results with predicted capacities and design loads using existing design methods. They discovered that the elastic plate analysis is more accurate than the crossed strip method, Baker's principle stress method (Baker 1981) and Lawrence's moment coefficient method (Lawrence 1980) in predicted cracking load. However, the yield line approaches predicted capacities to within 10% or 20% of the experimental values depending on whether or not the moment capacity across an early to develop or pre-existing crack was taken into account. Irrespective of the analytical approach, it was doubted whether the design stresses specified in the North American codes that time provided adequate safety against first cracking. They also concluded that it is difficult to ignore the potential of using a yield line approach because of its good estimation of the capacity even though it was objected to by many designers (Hendry and Kheir 1976, Cajdert 1980). It is because of lack of availability of a design method that both rational and accurate investigations on the out-of-plane bending of block walls are necessary.

Sinha and Ng (1994) presented test results on one-way bending wallettes and two-way bending walls using half-scale bricks. The moment-curvature relationships of the wallets bending in two orthogonal directions were found to be linear up to failure and the load dropped immediately to zero once the ultimate tensile strength had been reached. The average reactions of the wall panels were also measured versus applied load and the load distribution indicated that the load from vertical bending does not drop to zero upon cracking. It is shed to the stronger horizontal direction. In addition, the distribution of the strains measured in the two directions at the center of the wall confirms the load shedding behaviour after cracking. All of these results indicate that the brickwork behaves as a brittle material and once cracked, the cracked section cannot support the applied moment. They concluded that brickwork cannot be idealized as the rigid-plastic material on which the yield line method is based.

1.2.2 Analysis and Design

A number of effective methods for the analysis and design of laterally loaded masonry walls have been summarized in the literature. It has been agreed that no one method has been able to provide close prediction of the cracking loads or ultimate strengths of all the cases tested. In this regard, none of these methods is considered to be universally accepted. Nevertheless, understanding of the inherent principles of the available analytical methods is not only instructive but also necessary for mastering rational design concepts.

Elastic Plate Method

Masonry walls have been shown to behave elastically prior to the first cracking. Elastic plate analysis is an obvious option for predicting the first cracking loads or the failure loads for those walls having no reserved strength after the initial crack occurs.

The elastic plate solutions can be based upon the methods given by Timoshenko and Woinowsky-Krieger (1959) for analyzing the behaviour of isotropic plates. Measurements of the modulii of elasticity for horizontal and vertical brick beams (Lawrence 1983) have shown that the relative small value of the stiffness ratio (between 1 and 2) do not significantly influence the plate bending moment distribution. Therefore, the isotropic elastic plate theory is accepted for studying of masonry panels. The solutions may be expressed in the form of trigonometric series that can be optimized for computation in most personal computers. This indicates the efficiency of the elastic plate method for analyzing simple plates.

In practice, the analysis is carried out by employing bending moment coefficients in the two orthogonal directions. Failure is defined when the flexural tensile stress due to the maximum bending moment in either of the two directions exceeds the specified flexural tensile strength in that direction. As a result, the failure always occurs at the mid-span of the panel. However, as stated by Drysdale et al. (1999), the maximum bending moments from these coefficients may not be the critical case because of the orthotropic nature of masonry strength.

The considerable extent of anisotropy was reported by Seward (1982), as he proposed an orthotropic plate solution. This method excels over the standard isotropic solutions as both the stiffness orthotropy and strength orthotropy are taken into account to capture the true nature and behaviour of brickwork. In addition, principal moments that may not correspond to the orthogonal axes are considered such that the critical bending moment can be identified and progressive cracking can be predicted.

However, it should be pointed out that due to inconsistency in the reports of the degree of orthotropy (stiffness ratio) (Hendry 1973, Sinha 1978, Hamid et al. 1988), uncertainties still exist in the application of elastic plate method. On the other hand, random variation of material properties has been considered as an important issue when applying this method. Lawrence and Cao (1988) presented an analytical method based on elastic plate theory to predict the first cracking load of nonloadbearing walls. They adopted a Monte Carlo simulation approach, where bending and twisting moments at the established grid points are compared with the random flexural tensile strengths assigned to these points. A particular failure criterion was used to determine the critical point. Good prediction was found from comparisons with the results of 32 full-scale tests on clay brick walls. The random variability of flexural strength was shown to affect the prediction, with a drop in load capacity of the order of 50% corresponding to a coefficient of variation of 0.05 to 0.40. The observed higher strengths of blockwork compared to brickwork are believed to be partially due to the random strength effect and the relative size of units. Other considerations such as lack of knowledge about the biaxial failure criterion for masonry walls, the difficulty of dealing with irregular geometric configuration of the walls, as recounted by Hendry (1973), and the unreliability of predicting the cracking strength of panels likely to develop high torsional moments, as reported by Fried et al. (1988), have all limited the use of the elastic plate method.

Yield Line / Fracture Line Method

The yield line method was originally applied to predict the failure load for the under-reinforced concrete slabs (Johansen 1972). It assumes that the bending moments after reaching the moment at yield of reinforcement can remain constant along the yield lines (cracks where the moments reach the yield values) as further curvature occurs along these lines. It is understandable that there should exist some reserve of strength in the material as cracks develop in order for the yield line theory to be applicable. There is no doubt about this for reinforced concrete slab because the ductile behaviour of the material has been shown to be due to the presence of reinforcement. However it is not theoretically justifiable for masonry panel composed of brittle material, even though good agreement of using yield line analysis with the test results has been reported in many previous works (Hendry 1973, Haseltine et al. 1977, Cajdert 1980). As pointed out by Essawy (1986), the good predictions for yield line method in some cases were mainly due to the fact that for favorable loading and support conditions, both the theory of elasticity and the theory of plasticity may produce practically identical results. Other arguments (West et al. 1977, Haseltine et al. 1977, Anderson and Bright 1976, Sinha and Ng 1994) such as rotational restraint at the support and the effect of self-weight having been ignored in the analysis may also explain the good predictions. Despite the fact that the brittle nature of masonry panels is not reflected in a typical yield line analysis, the ability to incorporate different strengths in orthogonal directions makes it simple to apply the method in the design.

A rational modification of the yield line method which is called fracture line method was proposed and verified by Sinha (1978, 1980). The method differs from the yield line approach by including the anisotropy of the materials and not allowing maintenance of constant moment after cracking. It is known that the initial cracks develop long before the ultimate strength is reached, and that these cracks can be regarded as hinges incapable of resisting any moment. Drysdale et al. (1999) proposed the similar approach and presented a detailed solution for analyzing two-way bending panel using a failure line method. The energy approach of equating internal work (moment × rotation) and external work (load × displacement) was used to determine the minimum load, which corresponds to the ultimate strength, and the failure pattern. The failure line method yields more conservative results than does the yield line method, since existing cracks are not considered able to resist moment. This method has been incorporated into the current Canadian masonry design code (CSA S304.4, 2004).

Empirical Strip method

A traditional empirical strip method applied for panels simply supported on four sides is known as the Crossed Strips Method. The capacity of the panel is considered to be the sum of the capacities of a vertically spanning strip and a horizontally spanning strip. The proportion of the lateral load in the two directions is normally determined by accounting for the compatibility of deflection at the intersection of the strips. However, a modified method was proposed by Baker (1980) based on the traditional approach without regard to compatibility of deflections. Even though it lacked some rationality, the method yielded reasonable agreement with tests of brickwork panels.

Another strip method was proposed by Hendry (1973) to derive empirical values for effective bending moment coefficients for situations when no fully satisfactory theory for the calculation of the flexural strength of brickwork panels was available. A strip of unit width, parallel to bed joints, on the top edge for a wall supported on three sides or at mid-height for a wall supported on four sides, was considered. By equating the resisting moment and external bending moment based on experimental results for walls with different aspect ratios, Hendry calculated the effective bending moment coefficients and produced curves, which can be used to derive design moments according to aspect ratios. However, the limitation that the approach is based on the moment of resistance in the direction of the bed joints rather than the interaction of moments in two directions, has led to prudent application of the method.

Essawy (1986) evaluated the capacities for 14 walls with aspect ratios ranging from 0.36 to 4.21 and concluded that predictions using the crossed strip method are more conservative than the yield line approach. It was noticed that strips spanning in both directions are not strictly supported only on two sides. Higher predicted capacity of the panel could be expected by taking into account the support provided by strips closer to the edges of the wall panel. However, for larger walls the capacity could be slightly overestimated. This is attributed to the high random variation in material strengths, which causes weak elements to be subjected to high bending moments and produces lower cracking loads (Baker et al. 1985).

A drawback of the empirical strip method is that it is incapable of dealing with walls with openings. A useful extension to this approach has been proposed by Drysdale et al. (1999) wherein a gridwork of strips could be used.

Principal Stress Method

A rational approach presented by Baker (1981) was based on elastic plate theory. He proposed a principal stress failure criterion and a method for the determination of the flexural strength with respect to different orientations of bending. In his analysis, the principal moments at various points were calculated and Monte Carlo simulation techniques were used to account for random variation in properties. The developed principal stress theory was used to predict first cracking and panel capacity and resulted in good predictions for model and full-scale brickwork panels. Comparisons also showed that this method produced much closer and conservative predictions than the unconservative ones from the yield line method (Baker et al. 1985). Similar analysis was reported by Seward (1982) as reviewed before, but did not consider the random variation of material properties.

Normal Moment Method

Fried et al. (1988) applied another method, which is called the Normal Moment Method, along with the above-mentioned methods to study the effects of different analytical techniques on predicting lateral strength of masonry walls. This method, as recounted by the authors, was proposed by Kemp (1965) for the analysis of reinforced concrete slabs. The criteria for failure is described as failure occurs along a plane inclined at an angle θ to the vertical direction in an element when the normal applied moment M_n is equal to the normal resisting moment M_n^*. As the moments are assumed to be directly proportional to the extreme fiber stresses, the lowest ratio of M_n^*/ M_n, and thus σ_n^*/ σ_n, corresponds to the failure load.

Predictions for panels simply supported on three sides or four sides show that the results obtained using the normal moment method were always less than the yield line results at all aspect ratios and either less than or equal to those from elastic plate analysis. For panels simply supported along three edges but free along the top, unlike the elastic plate method that predicts failure at the geometric center of the panel, the normal moment method predicts failure near the bottom corners where high torsional stresses exist.

In view of the available analysis and design methods, it has been suggested by most of the researchers that the yield line method and the empirical strip method be used for prediction of panel capacity whilst the elastic plate method, principal stress method and normal moment method can be used for prediction of the first cracking load.

1.2.3 Applications of Finite Element Modeling

With the development of rational design methods, finite element methods have been extensively used in the analysis of masonry structures. As reviewed by Tzamtzis and Asteris (2003a), two-dimensional plane stress formulations, isotropic elastic material assumptions and macroscopic modeling approaches have been mainly adopted in the early stage of analysis. Some analytical procedures, which account for the nonlinear behaviour of masonry using three-dimensional microscopic modeling approaches, have also been developed. Bull (2001) conducted an elaborate review regarding homogeneous and heterogeneous models for the analysis of masonry structures ranging from simple to complex. The majority of these models includes constitutive models for the analysis of masonry before and after cracking occurs.

As masonry is a composite of block or brick units with interposed mortar joints, it behaves as a heterogeneous material with properties that depend on the block/brick, mortar and their interaction at the interface. The corresponding finite element model, in general, is formulated either by discretizing the individual blocks and mortar joints referred to as a heterogeneous model, or by developing a macroscopic formulation that accounts for the anisotropic properties of the combined assemblage. In the second approach, an equivalent homogeneous material is derived instead of using individual properties of the masonry component materials. These two methods can also be categorized as discrete modeling and composite modeling. Both modeling techniques provide advantages and disadvantages. These considerations will be discussed in what follows along with a brief review of several previous modeling approaches.

1.2.3.1 Heterogeneous Models

Heterogeneous models can be actually developed through a microscopic approach. From previous research literature, two approaches have been employed. First, block/brick units and mortar joints are separately modeled using continuum finite elements. This requires a large computational effort to analyze masonry structures but can be relied on to determine accurate stress distribution in both materials. In the second approach, an interface element is used to model the behaviour of mortar joints so that the interaction between two adjacent units can be reproduced. Likewise, this also results in considerable computational complexity for real structures.

Page (1978) proposed a heterogeneous model for clay masonry walls subjected to in-plane loading, where masonry was considered as a continuum consisting of isotropic elastic bricks and inelastic mortar joints possessing restricted mechanical properties. The properties of joint element were varied depending on the degree of compression present. Failure was identified when joint tensile or shear bond strength was reached. The model incorporated nonlinear joint properties that can be easily derived from uniaxial tests. The assumption of elastic characteristics of bricks resulted in limited computational effort. However, the ultimate load could not be predicted due to the lack of failure criterion as the result of the complex triaxial stress produced in mortar-brick interaction and local effects of the bonding pattern.

Gbosh et al. (1994) developed a two-phase material model for the analysis of unreinforced brick walls. Both brick and mortar were modeled using 2-D continuum elements and the interface between the brick and the mortar was modeled to be subjected to bond failure and friction. They studied the elastoplastic constitutive law and adopted the smeared crack approach to simulate cracking of the brick and the mortar. An elastic-softening behaviour was adopted to model the bond failure of the interface in tension.

A three-dimensional nonlinear microscopic model was presented by Tzamtzis and Asteris (2003b) for static and dynamic analysis of masonry structures. The model was applied to brick walls subjected to in-plane bending. Brick units and mortar joints are treated separately to accommodate nonlinear deformation characteristics and progressive local failure of materials. To account for the influence of mortar joints, interface elements are used to simulate time-dependent sliding and debonding failure. They concluded that the model produced a reasonable degree of accuracy in analysis compared to available analytical and experimental solutions.

1.2.3.2 Homogeneous Models

In homogeneous models, block/brick units and mortar joints are not modeled individually. This approach is appealing, as it demands less effort for discretizing and analyzing the structure. However, the macro models typically do not adequately account for the anisotropic behaviour of unreinforced masonry, specifically the presence of head and bed joints. Hence, the influence of the mortar joints acting as planes of weakness is difficult to address. The homogeneous models are based on the definition of an equivalent continuum, the properties of which can usually be derived from the properties of individual constituents through appropriate homogenization techniques.

Essawy (1986) proposed a nonlinear finite element model to predict the flexural behaviour of hollow block walls. The model was discretized without particular regard to the position of the mortar joint planes. He applied a layered plate approach and included orthotropic properties for each layer, trying to improve the adequacy of the macroscopic approach. The nonlinearity due to cracking and shear deformation caused by the discontinuity of the block webs were taken into account. It has been shown that replacing the block webs by an equivalent lamina tends to underestimate the transverse shear effects, especially for the short-span walls (Essawy 1986).

Some two-dimensional models were also proposed in later research. As recounted by Pande et al. (1994b), Chong et al. adopted a four-node flat shell element and used smeared material properties to calculate failure pressures closer to the experimental results in comparison to yield line theory; Lawrence and Lu performed a parametric analysis of masonry panels by employing a plate element and changing material properties randomly.

A homogenization technique was adopted by Pande et al. (1994b) to investigate the elastic-brittle behaviour of brick walls subject to incrementally increasing lateral load. Firstly, masonry units were homogenized with respect to head joints; the resulting material was then homogenized with respect to bed joints to obtain equivalent orthotropic material properties (Pande et al. 1989, 1994a). Following this process, the cracks were homogenized with the adjacent equivalent material to trace the crack propagation. The stepwise homogenization technique has been shown to minimize the numerical effort required for the analysis.

Based on Pande et al.'s equivalent material approach, Saliba et al. (1992, 1996) developed a 2-D "Equivalent-Material Model" to study the behaviour of brick walls under compression. In addition, they developed a "Two-Materials Model". The predicted stress distributions in the masonry panel using the two models were compared to test results. Although, the two models yielded similar results, they have their own merits and drawbacks. The Equivalent-Material Model is simpler and more economical to use, but it is best suited for linear analysis. The derived average stresses and strains need to be converted to the constituent stresses and strains using a structural relationship matrix. The Two-Materials Model, which can be used for nonlinear analysis, requires however large computer storage and run times.

Chen (2002) developed an elastic 3-D finite element model to predict the first cracking load of concrete block walls with or without openings under lateral loading. The model was simplified using a macroscopic approach without discretizing the blocks and mortar joints. The geometric effects of hollow blocks were incorporated instead of using a single- or multi-layer model, but no quantified comparative analysis was offered.

1.3 Objectives and Scope

The review of previous work shows that many experimental programs have been conducted to investigate the flexural behaviour of masonry walls subjected to out-of-plane loading. Rational design concepts have been proposed and gradually enriched. From the research on brickwork in the early stage and later on blockwork (Hendry 1973, Haseltine 1977, West 1977, Sinha (1978, 1980), Baker 1981, Lawrence (1980, 1983), Seward 1982, Anderson 1976, Cajdert 1980, Essawy 1986, Gazzola 1986), including recent investigations on walls with openings (May and Ma 1988, Middleton and Drysdale 1995, Chen 2002), there is now a significant body of research information. However, success in developing rational design methods is likely to be limited unless progress is made to achieve a good understanding of the true behaviour of masonry construction.

From the above review, it can be seen that insufficient work has been conducted on the finite element modeling of concrete block walls with openings under out-of-plane loadings. In particular, little quantified analyses of the effects of different modeling techniques on the structural response of masonry exist. Attempts to model the nonlinear behaviour of concrete block walls using finite element model are still quite limited. Consequently, an extensive investigation to develop a representative model capable of predicting both the pre-crack and post-crack behaviour of unreinforced concrete block walls subject to out-of-plane bending is instructive and necessary.

The predicted results obtained from the application of the proposed model are to be verified with the experimental data provided by Essawy (1986) and Chen (2002), who conducted the research on block masonry walls with and without openings under out-of-plane loading at McMaster University. It is expected that the proposed model will be used to predict the initiation of the first crack and the development of progressive cracking leading to formation of failure mechanism. Meanwhile, it is intended to propose an appropriate design recommendation for the Canadian masonry code regarding the flexural strength of concrete block walls with openings.

1.4 Thesis Outline

This study is organized in five major stages and, correspondingly, the thesis is organized to comprise five chapters. In the first chapter, a comprehensive literature review has been briefly introduced. The relevant topics include important mechanical properties such as flexural tensile strengths and their affecting factors, test methods, failure mechanisms for flexure of concrete masonry, orthogonal strength ratio and stiffness ratio, and common finite element approaches and their traits.

Chapter 2 contains a detailed evaluation of the effects of distinct modeling techniques on the structural response via linear elastic analysis. A numerical experiment for comparing common finite element modeling approaches of concrete block walls is introduced. Different modeling techniques are studied by taking into account the material constants, mortar property sensitivity and mesh size sensitivity. Based on the comparative analysis, recommended modeling methods are proposed for further study.

Chapter 3 is devoted to the development of a more representative model capable of describing the interaction between block units and mortar by introducing interface elements in the discrete modeling approach. The constitutive model, known as the Composite Interface Model, which can capture both tension and shear failure mechanisms, is introduced through formulation based on multi-surface plasticity. It is incorporated into a nonlinear analysis procedure to predict the inelastic behaviour at the unit-mortar interface.

In Chapter 4, the proposed discrete model for concrete block walls subjected to out-of-plane bending is evaluated by comparison with available test results from full-scale wall tests with different opening sizes and locations. Finally, the overall summary and conclusions are presented in Chapter 5. As an efficient preprocessing tool, a block masonry finite element mesh generator program BMFEM has been developed and it is used to generate all the wall models discussed in Chapter 4. A detailed introduction to the program is presented in Appendix A.

CHAPTER 2

FINITE ELEMENT MODELING OF HOLLOW BLOCK MASONRY

2.1 Introduction

Different finite element models for masonry structures have been introduced in Chapter 1. Do these modeling methods affect the predicted structural response? What are the implications of geometric configuration, material properties and mesh size sensitivity on the predicted structural response? These questions have not been extensively explored in the literature. To this end, and as part of the development of representative finite element model, a numerical experiment is conducted to quantify the effects of modeling techniques on the structural response of hollow block walls subject to out-of-plane loading. For this analysis, the focus is on the adequacy of the models in predicting the structural deformation and stress state of the wall prior to cracking. Accordingly, a linear elastic analysis is adopted for calculating the response on the basis of the finite element method.

Three simple wall geometries are considered, namely: horizontal spanning strip, vertical spanning strip, and two-way spanning wall. For the three wall geometries, four different modeling techniques are assessed; homogeneous solid model, homogeneous detailed model, heterogeneous stack pattern model and heterogeneous running bond model.

This chapter begins with a review of the geometry of the concrete masonry constituents and the common construction types of concrete block walls. The development of the finite element method to model masonry wall as a homogeneous or heterogeneous assembly is then introduced. As a critical factor governing the analytical results, material properties adopted in the modeling of masonry is discussed in detail. The finite element results, which are reproduced from the linear elastic analysis, namely stress and displacement data are presented followed by a detailed discussion through comparative analyses. Consequently, representative modeling methods are proposed for the nonlinear analysis.

2.2 Modeling Consideration for Concrete Masonry Unit and Wall

Concrete blocks with a large variety of shapes and dimensions have been produced for use in masonry construction. Depending on the net solid horizontal cross-sectional area, concrete blocks are classified as solid or hollow. Hollow blocks, defined with less than 75% solid horizontal cross-sectional area, have been widely used because of their reduced weight, ease of construction and overall economy. Typically, the percent solid ranges from 50% to 60%. The standard 20cm block having a size of 190×190×390 mm (height × thickness × length) is the most common unit in concrete masonry construction, as shown in Figure 2.1. Since most available experiment programs were conducted using these standard block units, they are used in this modeling study to compare the predicted behaviour to the experimentally measured one. Consisting of three webs joining the two face shells, a standard block has two cells that are tapered to facilitate manufacturing and construction. Conforming to the code requirement, the minimum thickness of face shells and webs of a standard unit is 32 mm and 25 mm, respectively (A.S.T.M. 1996). The ends of units, indented over the full height and known as frogs, are mainly incorporated due to rain penetration (Drysdale et al. 1999).

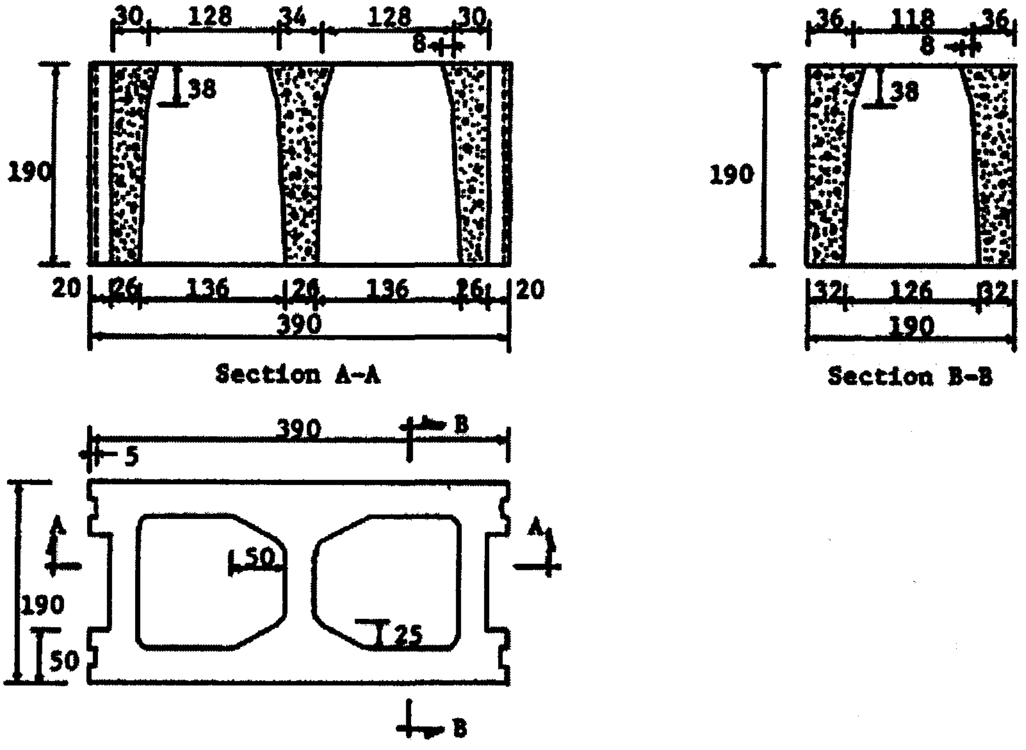


Figure 2.1 Standard block dimensions, Essawy (1986)

In most conventional constructions with concrete blocks, running bond pattern is adopted, where the head joint is positioned at the middle of the units below and above (Figure 2.2). Other patterns or bonds are also used, however it is mandated by masonry design codes that a minimum of one-fourth overlap be designated as running bond, as a structural requirement (Drysdale et al. 1999). The use of this arrangement results in the misalignment of webs of the units with the upper and lower courses. Thus, mortar can only be spread out on the two face shells to produce partial bedding rather than full bedding. As a result, a void is created under the webs, and for ungrouted masonry, loading is transferred only through the face shells (Figure 2.3). Moreover, due to the presence of frogged ends, mortar can also only be placed over the full height of the face shell to form head joints.

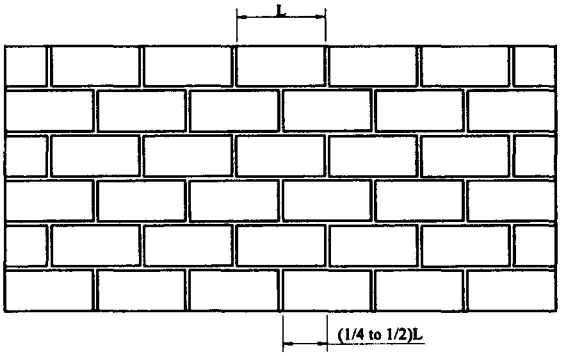


Figure 2.2 Running bond pattern

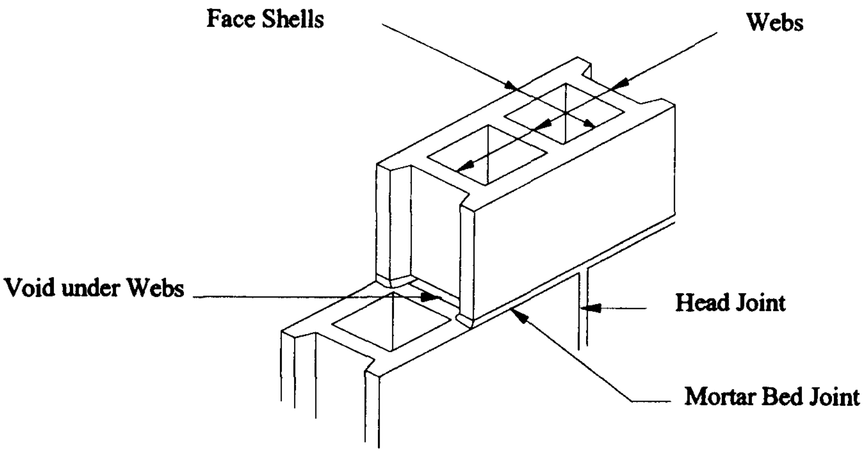


Figure 2.3 Detail showing mortar joint location

Another pattern often used for laying masonry is stack pattern, where block units are simply stacked over each other without any overlapping, forming continuous vertical mortar joints along the head joints of successive courses (Figure 2.4). Although, walls of stack pattern can be designed for both loadbearing and nonloadbearing applications, it is not recommended for the former as the lack of overlap of the units can lead to fairly low tensile strength perpendicular to the head joints. This can lead to the development of continuous crack along the vertical joints. As such, stack pattern is commonly adopted for decorative purpose. In this study, the two types of construction, running bond and stack pattern, are investigated.

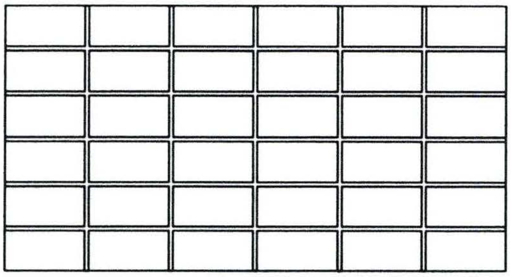
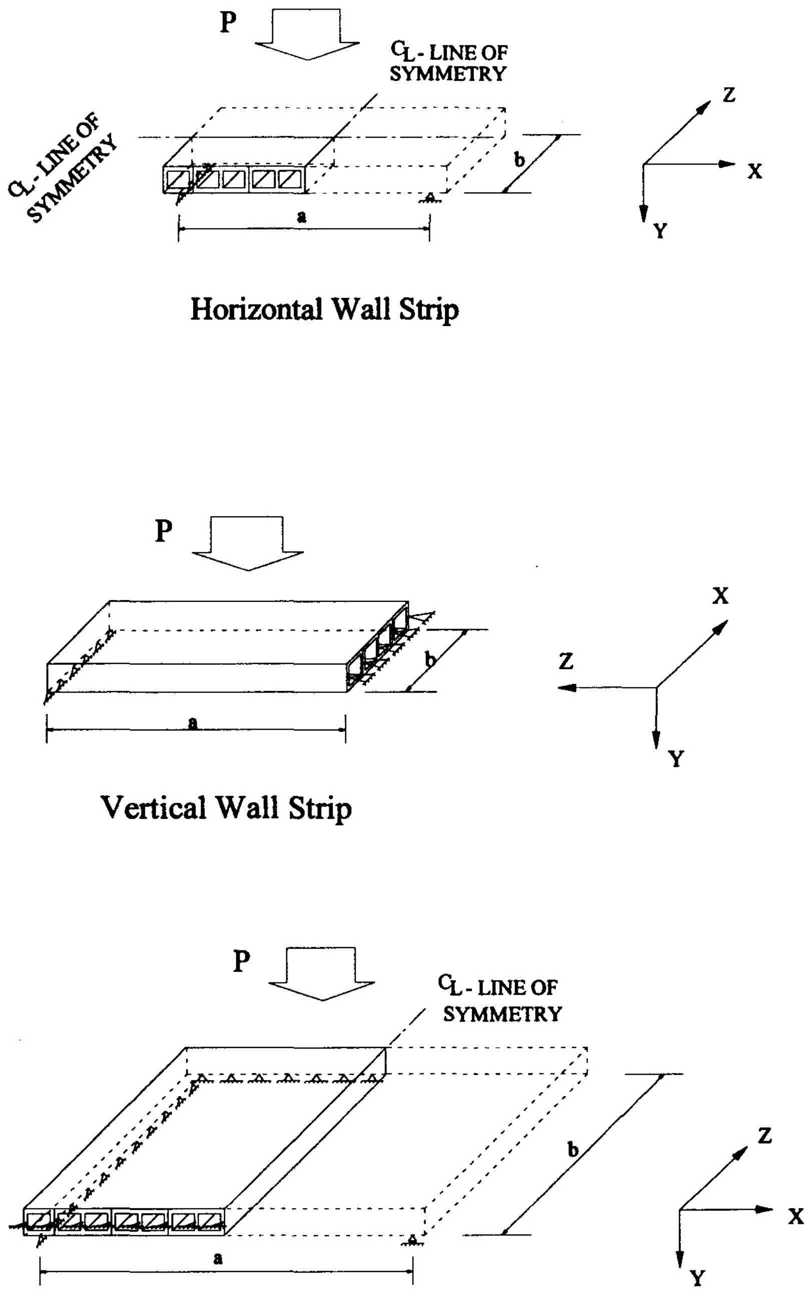


Figure 2.4 Stack pattern

Masonry walls may bend in one or two directions when subjected to out-of-plane loading, depending on their aspect ratio and support conditions. In this study, unreinforced single-wythe walls that are spanning in the horizontal direction, vertical direction or both directions are examined (Chidiac et al. 2004). These three cases are schematically shown in Figure 2.5. The wall ends are either free or simply supported. Depending on the symmetric nature of the wall strip, boundary conditions and loading conditions, the size of the representative model has been reduced accordingly. For the horizontal strip, a quarter-model is used, whereas a full model is used for the vertical strip due to the lack of symmetry in the top and bottom boundary conditions. For the two-way spanning panel, a half model is adopted. The flexural behaviour of these three concrete block wall is studied to determine the effect of wall geometry on the structural response. For this parametric study, the four span/thickness ratios for the two wall strips and the two aspect ratios (horizontal span/vertical span) for the two-way wall panels that are investigated are listed in Table 2.1.



Horizontal Wall Strip

Vertical Wall Strip

Two-way Bending Wall Panel

Figure 2.5 Schematic representations of one-way and two-way spanning walls

Table 2.1 Geometric configuration of the masonry walls

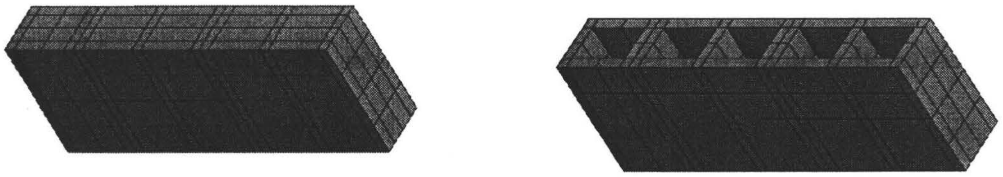
Configuration	Panel dimension a × b (mm)	Span/Thickness ratio
Horizontal strip	1000 × 800	5.26
	1800 × 800	9.47
	3400 × 800	17.89
	5800 × 800	30.52
Vertical strip	800 × 828	4.21
	1600 × 828	8.42
	2800 × 828	14.74
	3200 × 828	16.84
		Aspect ratio
Two-way panel	2600 × 2800	0.93
	5800 × 2800	2.07

2.3 Modeling Description

Two modeling approaches were considered in discretizing the three wall geometries; heterogeneous and homogeneous. For the homogeneous model, which encompasses homogenization techniques for determining material properties, the block unit is represented by either an equivalent solid unit or physical geometry of the hollow block; whereas for the heterogeneous models, the physical geometry of the hollow blocks and mortar joints are represented.

2.3.1 Homogeneous Models

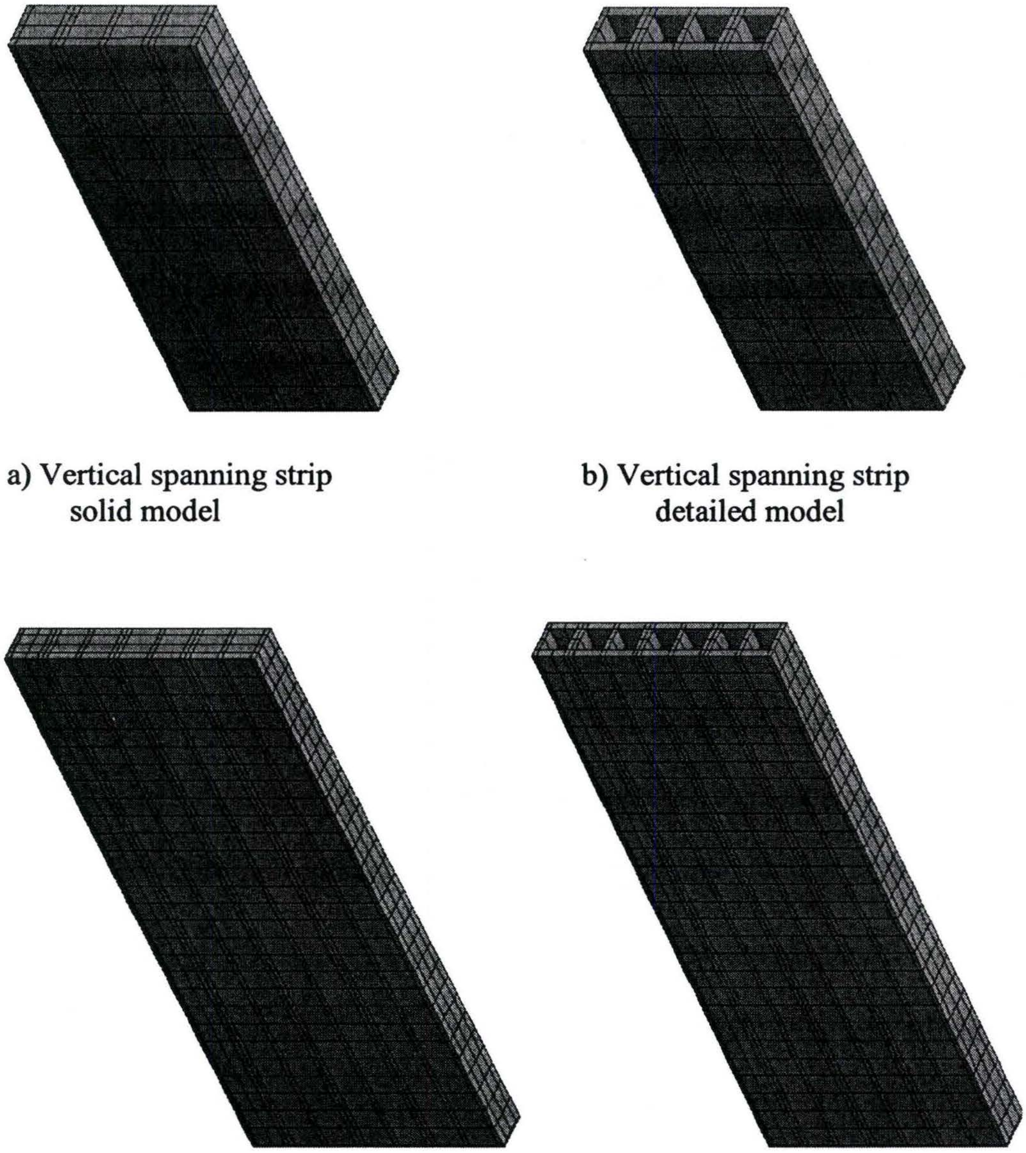
Two homogeneous models are evaluated representing the two extreme conditions, solid model and detailed model. Due to the difficulty in modeling the block geometry, analysts have in the past represented the block with a solid unit of equivalent structural properties. This approach permits the determination of the thickness of the solid block with the premise that the two sections have the same structural properties, namely the cross section area, A, section modulus, S, and second moment of inertia, I. However, only one of the three properties can be met due to the differences in the geometric properties. To derive an equivalent solid section of equal elastic bending stiffness, the second moment of inertia becomes the only mechanical property to be calibrated against. This results in a reduced block thickness of 172.5 mm and 176.9 mm, instead of 190 mm, for horizontally spanning walls and vertically spanning walls, respectively. For the two-way spanning walls, an equivalent block thickness of 176.9 mm is adopted derived from the knowledge that the walls' rigidity will be mostly influenced by the aspect ratio and boundary conditions. The corresponding models for the three wall geometries are shown in Figure 2.6a, 2.7a and 2.7c.



a) Solid model

b) Detailed model

Figure 2.6 Finite element mesh of a horizontal spanning concrete block wall according to the two homogeneous models



a) Vertical spanning strip
solid model

b) Vertical spanning strip
detailed model

c) Two-way spanning panel
solid model

d) Two-way spanning panel
detailed model

Figure 2.7 Finite element mesh of vertical and two-way spanning concrete block walls according to the two homogeneous models

On the other end of the spectrum, referred to as detailed model, the physical geometry of the block is accurately represented as shown in Figure 2.6b, 2.7b and 2.7d. Visual inspection of the model reveals that it appears to be more representative as it attempts to model the pear-shaped block cells in the wall. The discretization of the block unit using average dimensions follows the method proposed by Chen (2002), and reproduced in Figure 2.8. Accordingly, the thickness of the face shells and webs is assumed uniform and equal to 35 mm and 28 mm, respectively. This approach ignores the influence of the flare and taper effect. The influence of the flare and taper in the block face shell on the prism strength was studied by Guo (1991). According to the finite element results, he concluded that the influence is insignificant. Moreover, replacement of the block's frogged ends with flat ends is found to have no significant effects on the structural behaviour and is adopted because it greatly simplifies the model.

Standard block is discretized into three Element Groups (EG) as shown in Figure 2.8. This permits the representation of the face shells (EG3), webs (EG2) and their intersections (EG1). A gap of 1mm was incorporated into the block between two adjacent webs to reflect the actual block configuration of three webs per block. End of the block is left open to ensure the alternative occurrences of the middle webs and end webs when the representative block is repeatedly extended in the horizontal direction. To discretize the wall, the block was first duplicated horizontally then vertically. A half block unit, 214 mm long, or a single web, 28 mm thick, was added to the right end to complete the wall model. The selection

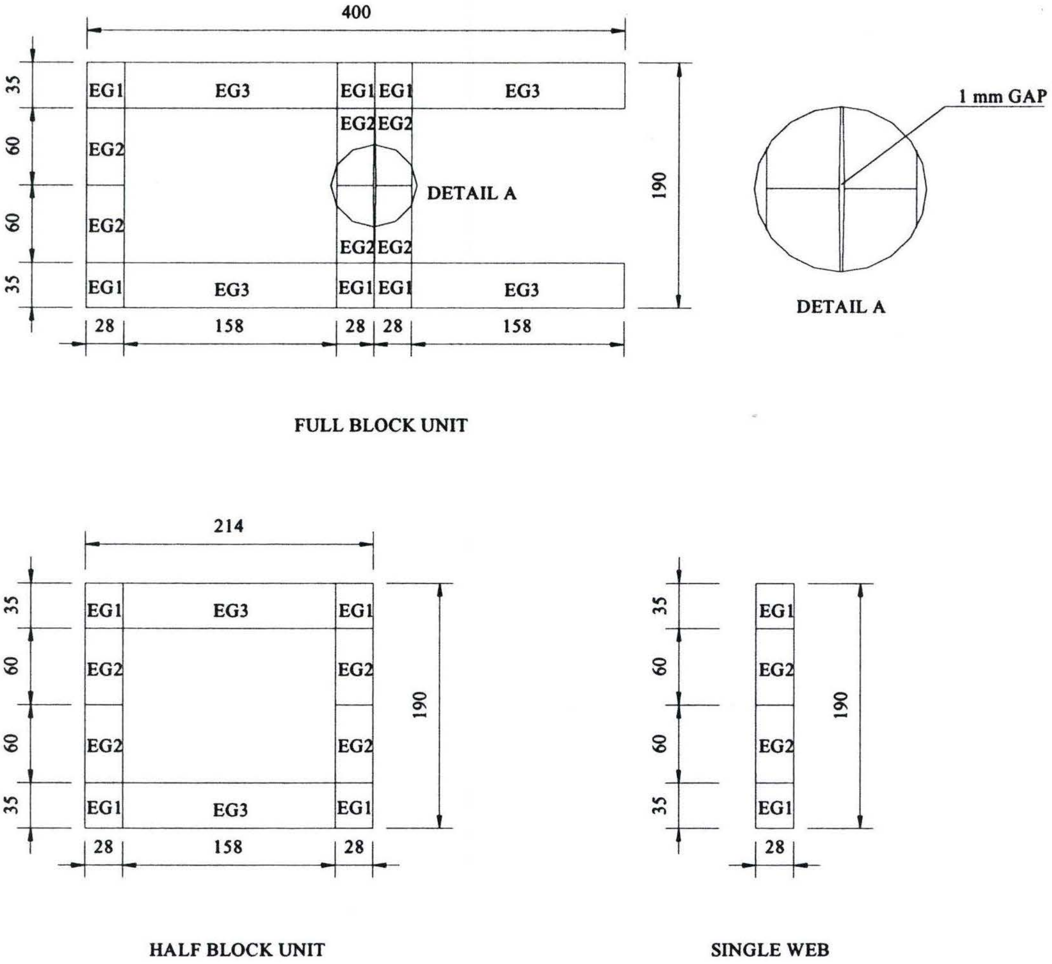


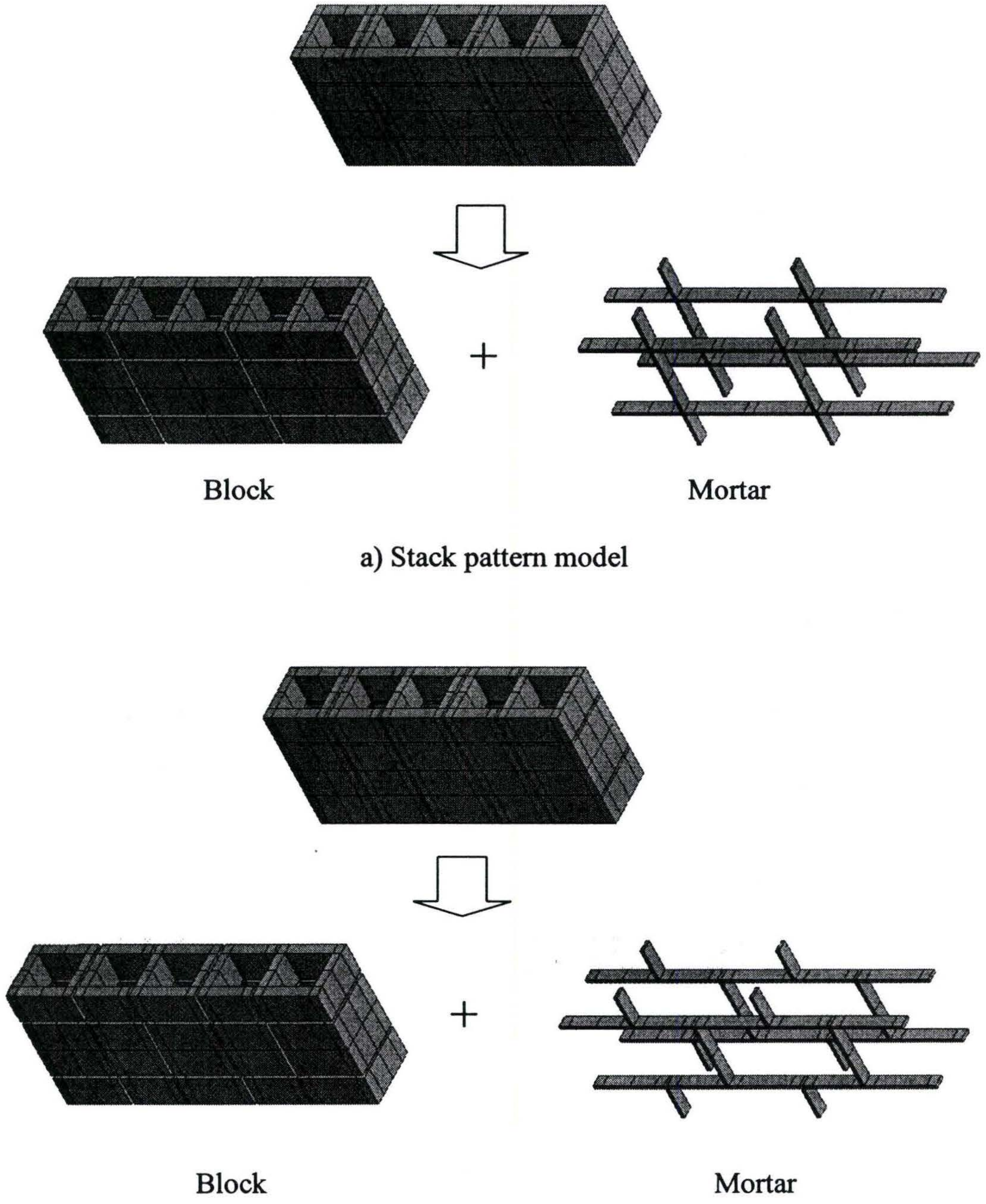
Figure 2.8 Representative standard and half block unit in homogeneous detailed model

depends on the length of the wall. This approach has led to a wall length that is either 14 mm or 28 mm longer than the actual wall length. It should be noted that the location of the wall's boundaries are the same as the actual wall. This has led to subdividing two parts, by adding 4 nodes, to accommodate positioning of the left vertical support condition in horizontal wall strip and two-way spanning panel.

The solid homogeneous model and the detailed homogeneous model employ the three-dimensional solid brick element with 8 nodes to discretize the structure. As observed from Figure 2.6 and 2.7, the same level of mesh refinement is used for both models when comparing the results.

2.3.2 Heterogeneous Models

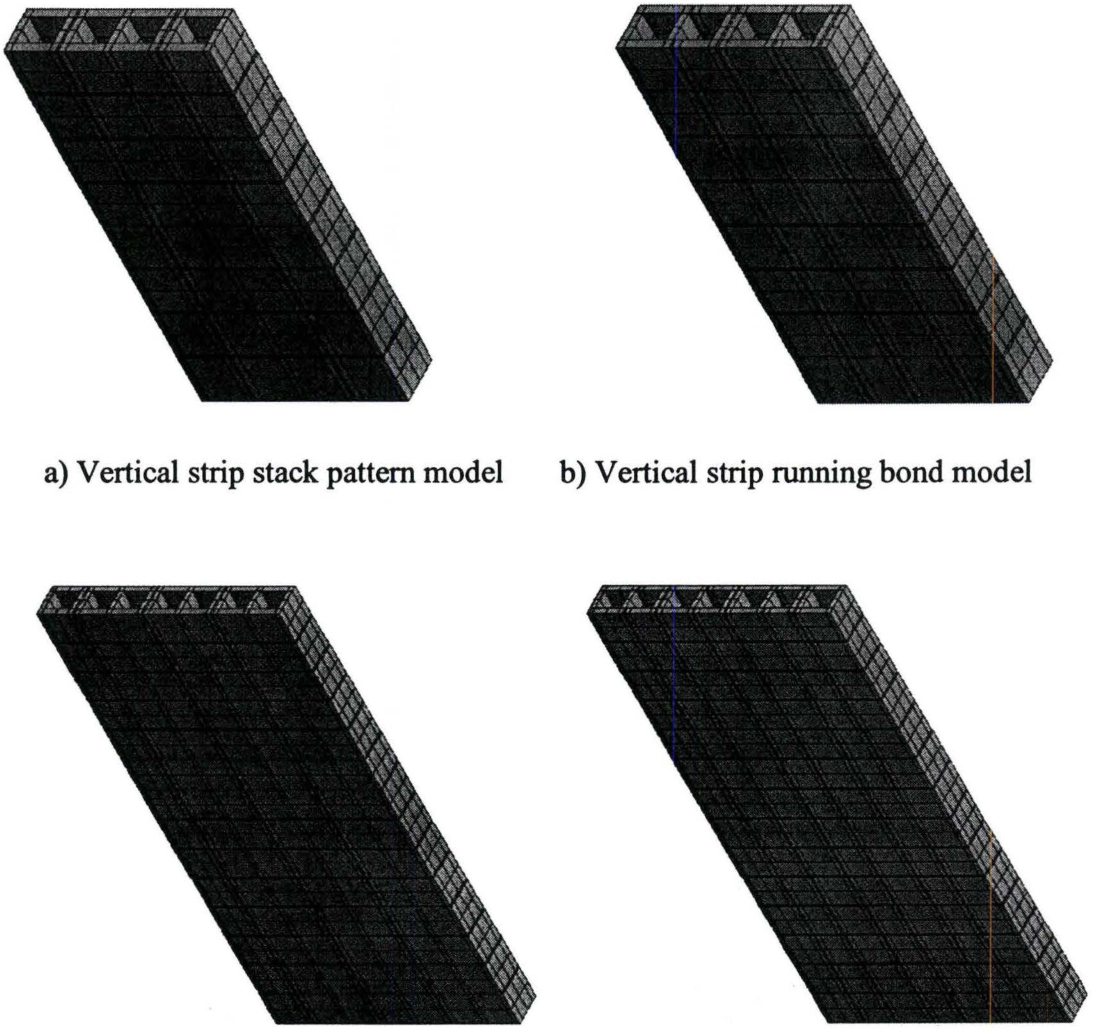
Two heterogeneous models are used to represent two construction patterns, stack pattern and running bond. In each model, the block units, mortar head joints and bed joints are modeled separately using the 8-node three dimensional brick elements. The material properties of the block and the mortar were distinctively defined. Figure 2.9 and 2.10 show the finite element meshes corresponding to three wall geometries. As can be seen from the horizontal spanning strip models, Figure 2.9a&b, the mortar joints are continuous in both directions for the stack pattern model, and staggered for the running bond model. Same observations apply to vertical strip and two-way spanning wall.



a) Stack pattern model

b) Running bond model

Figure 2.9 Finite element mesh of a horizontal concrete block strip according to the two heterogeneous models



a) Vertical strip stack pattern model

b) Vertical strip running bond model

c) Two-way spanning panel stack pattern model

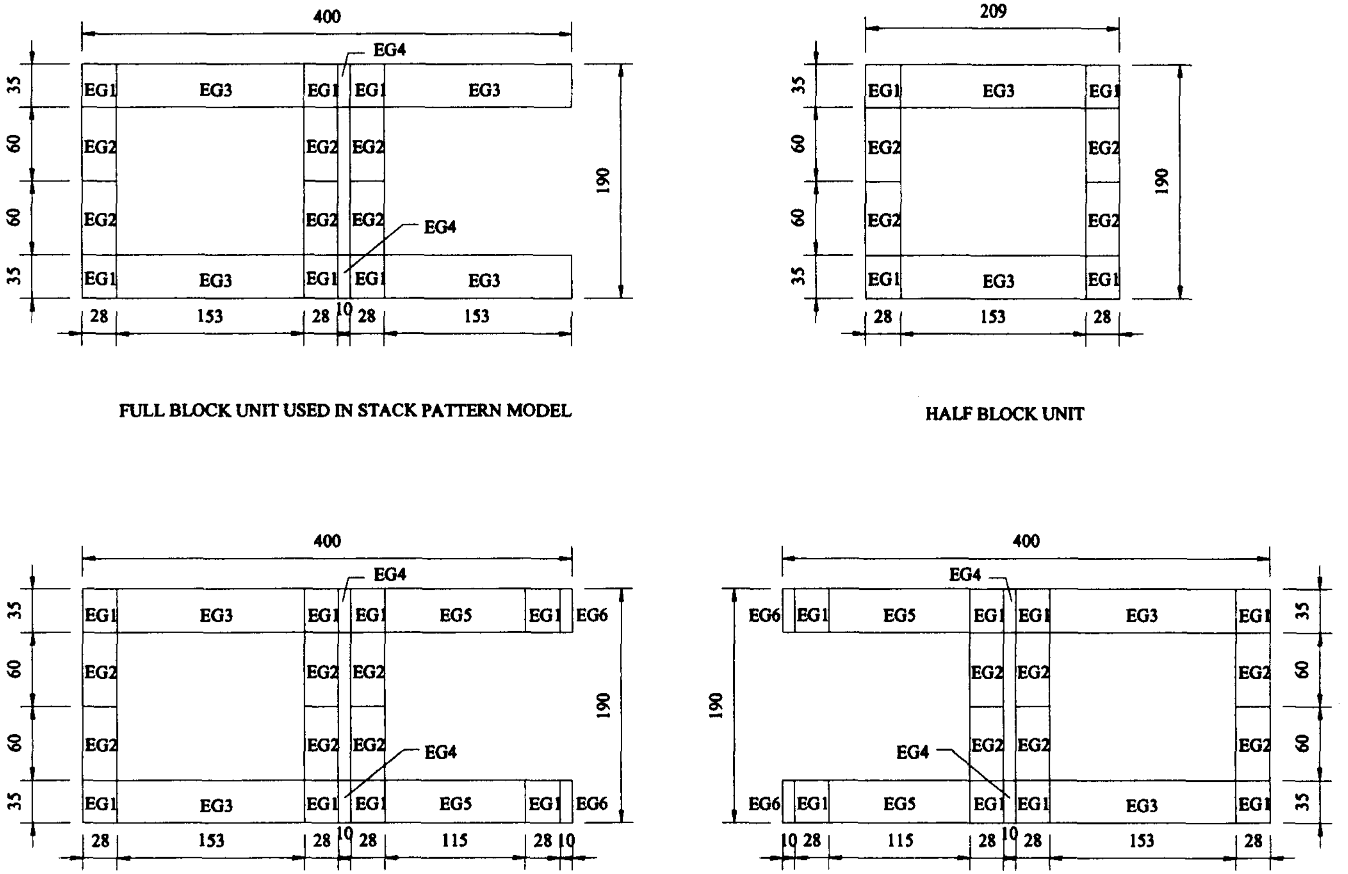
d) Two-way spanning panel running bond model

Figure 2.10 Finite element mesh of vertical and two-way spanning concrete block walls according to the two heterogeneous models

For the heterogeneous models, a rectangular gap between two webs is added to adequately model the mortar head joints as shown in Figure 2.11. To facilitate the generation of running bond model, two representative block units with opposite opening orientations were designed for odd and even courses of the wall. The models created using these units unavoidably lead to the vertical alignment of partial webs. Nonetheless, this method is chosen as it saves considerable computational effort in comparison to models generated with misaligned webs in two adjacent courses. It should be noted that the face shells, for the open end of the unit, are subdivided into three element groups (EG1, EG4, and EG5) to ensure compatible connectivity between two adjacent courses. Odd and even courses are constructed using different representative units.

2.4 Material Properties

The structural response of masonry is anisotropic due to the geometry of masonry unit and the presence of horizontal and vertical mortar joints. To account for the anisotropic characteristics, different strategies have been proposed for different modeling approaches. For the homogeneous model, the material properties of the masonry are assumed to be orthotropic. Accordingly, the properties are average values for the different material directions. Whereas, for heterogeneous model, isotropic material is assumed for the block units and mortar joints, and the anisotropic behaviour arises due to the geometric configuration of the masonry wall (Essawy 1986).



FULL BLOCK UNIT USED IN STACK PATTERN MODEL

HALF BLOCK UNIT

FULL BLOCK UNIT USED IN RUNNING BOND MODEL (ODD COURSES)

FULL BLOCK UNIT USED IN RUNNING BOND MODEL (EVEN COURSES)

Figure 2.11 Representative standard and half block unit in heterogeneous models

One of the objectives of this investigation goes beyond investigating homogeneous and heterogeneous models, it evaluates the implications of not adequately modeling the actual geometry of the blockwork on the overall computed response of hollow block walls under out-of-plane bending. In this regard, the representative properties for the two models are derived following Essawy (1986) proposed formulation.

For the derivation of the averaged properties, Essawy measured the modulus of elasticity and Poisson's ratio by testing 200 mm standard blocks in compression. The corresponding values are 19,660 MPa and 0.3, respectively. For the mortar joints, he adopted Hamid's values of 1,190 MPa and 488 MPa, for the modulus of elasticity and shear modulus (Hamid 1978). In developing a homogenization procedure, Essawy assumed that the local and global state of stress for masonry wallette, consisting of block units and mortar joints, meet the equilibrium state requirements and that the deformations are compatible with the strains. By taking into consideration the material properties and geometry of the masonry constituents, he proposed equivalent material properties for the modulus of elasticity in the two principal material directions and the corresponding shear modulii as reproduced below.

$$E_n = \frac{1}{\frac{h_b}{h_t}\frac{t_m}{t_b} + \frac{E_b}{E_m}\frac{h_m}{h_t}} E_b \qquad (2.1)$$

$$E_p = \left(\frac{1}{\frac{l_b}{l_t}\frac{t_m}{t_b} + \frac{h_m}{l_t}\frac{E_b}{E_m}\frac{1+2\frac{G_m}{E_b}}{1+\frac{G_m}{E_m}+\frac{G_m}{E_b}}}\right)E_b \tag{2.2}$$

$$G_{np} = \frac{1}{2}\left(\frac{1}{\frac{h_b}{h_t}\frac{t_m}{t_b} + \frac{G_b}{G_m}\frac{h_m}{h_t}} + \frac{1}{\frac{l_b}{l_t}\frac{t_m}{t_b} + \frac{h_m}{l_t}\frac{G_b}{G_m}\frac{1+2\frac{E_m}{G_b}}{1+\frac{E_m}{G_m}+\frac{E_m}{G_b}}}\right)G_b \tag{2.3}$$

$$G_{nz} = \frac{1}{\frac{h_b}{h_t}\frac{t_m}{t_b} + \frac{G_b}{G_m}\frac{h_m}{h_t}}G_b \tag{2.4}$$

$$G_{pz} = G_{np} \tag{2.5}$$

where,

E_b, E_m = Block and mortar's elastic modulus, respectively

E_p = Modulus of elasticity parallel to the bed joints

E_n = Modulus of elasticity normal to the bed joints

G_b, G_m = Block and mortar's shear modulus, respectively

G_{np} = In-Plane shear modulus

G_{nz} , G_{pz} = Out-of-plane shear modulii

h_b, h_m, h_t = Block unit height, mortar joint thickness and total height, respectively

l_b, l_t = Block unit length and total length of unit and mortar

t_b, t_m = Average unit and joint thickness perpendicular to the plane of the assemblage

The out-of-plane shear modulus along the head joints, G_{pz}, has been suggested to be equal to the in-plane shear modulus G_{np} by assuming similar stress distribution in the staggered head joints. For 200 mm standard block unit and 10 mm mortar joint, and by substituting for the component material properties given previously, the values for the equivalent elastic constants for masonry become:

$E_n = 0.59E_b = 11{,}600$ MPa

$E_p = 0.84E_b = 16{,}510$ MPa

$G_{np} = 0.79G_b = 5{,}959$ MPa

$G_{nz} = 0.61G_b = 4{,}613$ MPa

$G_{pz} = G_{np} = 5{,}959$ MPa

For determining the equivalent value for the Poisson's ratio, Essawy conducted a numerical experiment on a 400 mm × 400 mm masonry assemblage element subjected to uniform tension normal to the bed joints. From the results, a value of 0.2 was proposed for ν_{np} (Essawy 1986, Appendix C). The subscripts p and n denote the directions parallel and normal to the bed joints, respectively.

Table 2.2 summarizes the material properties adopted for the out-of-plane linear elastic analysis of hollow block masonry walls.

Table 2.2 Material properties employed in the FEA of hollow block walls

Model types	Material	E_x	E_y	E_z	ν_{xy}	ν_{xz}	ν_{yz}	G_{xy}	G_{xz}	G_{yz}
		(MPa)	(MPa)	(MPa)				(MPa)	(MPa)	(MPa)
Homo-geneous	Masonry	16,510 (=E_p)	16,510 (=E_p)	11,600 (=E_n)	0.200 (=ν_{np})	0.200 (=ν_{np})	0.200 (=ν_{np})	5,959 (=G_{pz})	5,959 (=G_{np})	4,613 (=G_{nz})
Hetero-geneous	Block	19,660	19,660	19,660	0.300	0.300	0.300	7,562	7,562	7,562
	Mortar	1,190	1,190	1,190	0.219	0.219	0.219	488	488	488

The elastic modulus in the wall thickness direction (Y direction) is assigned the same value as the modulus in the direction parallel to the bed joints. This stems from the consideration that the properties in the wall thickness direction have little effect on the structural behaviour of the wall since the thickness is relatively small compared to the other two dimensions. In addition, the stresses produced in the wall thickness direction under lateral loading are much lower in comparison to the stress component in the X and Z direction. Same assumption was deduced by Chen (2002). Due to the lack of available values for the Poisson's ratios in the three principal directions, a value of 0.2 was adopted for all three ratios. As noted in Table 2.2, the properties for the heterogeneous models are derived based on the assumption of isotropy in each constituent material. The corresponding shear modulus was calculated using

$$G = \frac{E}{2(1+\nu)} \tag{2.6}$$

where, G, E, and v represent shear modulus, elastic modulus and Poisson's ratio, respectively.

2.5 Parametric Study

A linear elastic analysis was carried out to study the static response of concrete block walls that are subjected to 1 kPa uniformly distributed load. A general-purpose finite element package, AFEMS^{TM} (V7.5), was used for this study. The computed results obtained from the four finite element models for the three geometric configurations are tabulated separately for comparative analysis. These data are presented in the form of maximum and relative values for the out-of-plane displacement, bending stress, first principal stress and Von Mises stress (effective stress).

2.5.1 Horizontal Strip Wall

The computed finite element results for the horizontal spanning strip wall are summarized in Table 2.3. For the purpose of comparing the results, all the values are normalized with respect to those obtained from the homogeneous solid model and reproduced in Table 2.4.

2.5.1.1 Out-of-plane Displacement

From Table 2.4, one observes that the use of an equivalent solid model to simulate the structural response of hollow block wall yields a stiffer response in comparison with homogeneous detailed model and heterogeneous models. For span/thickness ratio of 5.26, the displacement is found less than half that of the values of the other three models. As the span/thickness ratio increases to 30.52, the difference between the two homogeneous models decreases to about 15%, and, is about 40% less in comparison to the two heterogeneous models. Horizontally spanning wall, built with hollow blocks, behaves in the same manner as Vierendeel truss, where the face shells act as upper and lower chords connected with the parallel web chords (Figure 2.12). The bending and shear deformations in these comparatively flexible webs reduce the coupling effects between the face shells. On the other hand, since the webs are perpendicular rather than parallel to the bending direction, their contribution to the effective cross-sectional area is a small fraction of the span. Furthermore, the softer mortar in the head joints of the heterogeneous model, which links the two face shells, adds to the reduced shear and flexural stiffness. Therefore, the larger displacement predicted by the three nonsolid models compared to the solid model is understandable.

Examination of the results of the two heterogeneous models indicates that there are small differences between the two models. At the ratio of 5.26, they differed by about 13% and the difference became very small (less than 1%) at span/thickness ratio of 30.52. The displacements in stack pattern models are

Table 2.3 Computed max values for the horizontal strip block wall

Model type		Span/Thickness ratio	Principal stress (MPa)	Bending stress (MPa)	Displacement (mm)	Von Mises stress (MPa)
Homogeneous	Solid	5.26	0.0271	0.0269	0.0015	0.0269
	Detailed		0.0540	0.0528	0.0034	0.0524
Heterogeneous	Stack pattern		0.0591	0.0581	0.0033	0.0462
	Running bond		0.0524	0.0515	0.0037	0.0440
Homogeneous	Solid	9.47	0.0903	0.0899	0.0151	0.0896
	Detailed		0.1345	0.1326	0.0233	0.1309
Heterogeneous	Stack pattern		0.1489	0.1468	0.0247	0.1196
	Running bond		0.1360	0.1340	0.0265	0.1220
Homogeneous	Solid	17.89	0.3252	0.3238	0.1907	0.3227
	Detailed		0.4295	0.4246	0.2393	0.4186
Heterogeneous	Stack pattern		0.4866	0.4803	0.2773	0.3964
	Running bond		0.4460	0.4400	0.2831	0.4090
Homogeneous	Solid	30.52	0.9699	0.9657	1.6520	0.9625
	Detailed		1.2120	1.1990	1.8950	1.1820
Heterogeneous	Stack pattern		1.3800	1.3630	2.2880	1.1290
	Running bond		1.2700	1.2500	2.2930	1.1700

Table 2.4 Normalized values to homogeneous-solid for the horizontal strip wall

Model type		Span/Thickness ratio	Ratio of homogeneous-solid			
			Principal stress	Bending stress	Displacement	Von Mises stress
Homogeneous	Solid	5.26	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.99	1.96	2.31	1.95
Heterogeneous	Stack pattern		2.18	2.16	2.24	1.72
	Running bond		1.94	1.91	2.53	1.64
Homogeneous	Solid	9.47	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.49	1.47	1.55	1.46
Heterogeneous	Stack pattern		1.65	1.63	1.64	1.33
	Running bond		1.51	1.49	1.76	1.36
Homogeneous	Solid	17.89	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.32	1.31	1.25	1.30
Heterogeneous	Stack pattern		1.50	1.48	1.45	1.23
	Running bond		1.37	1.36	1.48	1.27
Homogeneous	Solid	30.52	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.25	1.24	1.15	1.23
Heterogeneous	Stack pattern		1.42	1.41	1.38	1.17
	Running bond		1.31	1.29	1.39	1.22

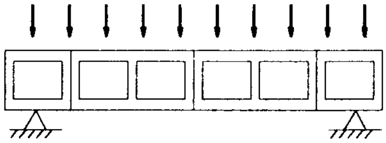


Figure 2.12 Horizontal spanning wall with Vierendeer truss type of construction

observed to be slightly less than those in running bond pattern. This may be attributed to the fact that the alignment of the webs in the stack pattern model contributes more to the effective cross section area in comparison to the webs in running bond model, and hence yields higher stiffness for lower values of span/thickness ratio.

The percent difference between the detailed homogeneous model and the heterogeneous model corresponding to stack pattern is found to increase as the span/thickness ratio increases. The difference varies from 3% to 20% as the ratio varies from 5.26 to 30.52. Although it is known that the same geometric configuration is modeled in the two models, the presence of softer mortar joints in heterogeneous stack pattern model appears to have major effects on the displacement. As the wall span increases, the contribution to the displacement field due to flexure becomes predominant. This results in a lower out-of-plane stiffness.

Similarly, the difference between the detailed homogeneous model and the heterogeneous model corresponding to running bond varies from 10% to 21% as the ratio increases from 5.26 to 30.52. This is understandable as the different geometric configuration (web alignment) and the influence of mortar joints both take effects.

2.5.1.2 Stress

The computed stress values, as given in Table 2.4, exhibit the same trends observed for the out-of-plane displacement. The bending stress values computed using the detailed homogeneous model and the two heterogeneous models, which are referred to as hollow models, are about twice the values of those obtained using homogeneous solid model for a span/thickness ratio of 5.26. As the span/thickness ratio increases to 30.52, the difference decreases to 24%, 41% and 29%, respectively, for the detailed homogeneous, heterogeneous stack pattern and heterogeneous running bond model. These results show that the solid model significantly underestimates the bending stress in comparison to the hollow models. One should recall that the equivalent thickness derived for the homogeneous solid model is obtained by equating moment of inertia of solid section to moment of inertia of the face shells based on the plane-section assumption. This approach, which employs the moment of inertia instead of section modulus, is not adequate in predicting the stresses. This is demonstrated through the results of span/thickness ratio of 30.52. One can observe that the difference in the out-of-plane displacements of the equivalent solid and those predicted by the homogeneous detailed model are 15%; however, the differences in the bending stresses are 25%. Moreover, this approach does not account for the shear deformation. As recounted in Chen (2002), the bending of the flexible web members combined with some shear distortion results in lack of plane section behaviour and, for high rates of change of bending moment results from high shear forces, the effects of the deformations are significant.

Relatively good agreement has been shown between the bending stress values from the homogeneous detailed model and the heterogeneous running bond model for all four span/thickness ratios. This behaviour is anticipated as the effect of the mortar joint is less pronounced here compared to the effect of unit webs. In comparison, the bending stress values from the stack pattern model are found to be somewhat greater than those of the detailed model and the running bond model. A reason for the difference is the loss of benefit due to the overlapping of units in the stack pattern models. The principal stress and effective stress followed patterns similar to the bending stress. The only noticeable difference is the values of the Von Mises stress computed using the heterogeneous models. There, one observes that the normalized values are significantly smaller in comparison to the bending stress and principal stress values.

2.5.2 Vertical Strip Wall

The computed finite element results for the vertical strip wall are summarized in Table 2.5. Again, the computed values are normalized with respect to those obtained from the homogeneous solid model to facilitate the comparison. These normalized data are listed in Table 2.6.

2.5.2.1 Out-of-plane Displacement

The computed results reveal a similar trend to the one noted for the horizontal strip wall but with a much smaller difference when compared to the results of homogeneous solid model. For the lowest span/thickness ratio, the difference between the two homogeneous models is down to 40% and the results using the equivalent solid model tend to be more comparable to the homogeneous detailed model at the higher ratios. Shear deformation again plays a role in the effects leading to this result. At low span/thickness ratios, the comparatively larger shear forces caused large shear stresses in the webs of the block connecting the face shells. Displacements resulting from this type of deformation are underestimated in the equivalent solid model. As the span/thickness ratio increases, the effects of the smaller cross-sectional area at the mortar bed joints and the lower modulus of elasticity of the mortar, which are not modeled in the homogeneous solid model, do not diminish. Comparing the displacement of the solid model to those of the detailed heterogeneous models, one observes that the solid model seems to underestimate the effects of the mortar as it uses an averaged modulus of elasticity.

Table 2.5 Computed max values for the vertical strip wall

Model type		Span/Thickness ratio	Principal stress (MPa)	Bending stress (MPa)	Displacement (mm)	Von Mises stress (MPa)
Homogeneous	Solid	4.21	0.0136	0.0136	0.0010	0.0136
	Detailed		0.0159	0.0159	0.0013	0.0163
Heterogeneous	Stack pattern		0.0176	0.0175	0.0014	0.0149
	Running bond		0.0180	0.0178	0.0015	0.0160
Homogeneous	Solid	8.42	0.0551	0.0551	0.0139	0.0551
	Detailed		0.0623	0.0623	0.0156	0.0623
Heterogeneous	Stack pattern		0.0689	0.0687	0.0180	0.0588
	Running bond		0.0700	0.0690	0.0182	0.0592
Homogeneous	Solid	14.74	0.1692	0.1692	0.1278	0.1692
	Detailed		0.1898	0.1898	0.1355	0.1902
Heterogeneous	Stack pattern		0.2097	0.2089	0.1569	0.1886
	Running bond		0.2132	0.2102	0.1578	0.1929
Homogeneous	Solid	16.84	0.2211	0.2211	0.2174	0.2211
	Detailed		0.2477	0.2477	0.2293	0.2482
Heterogeneous	Stack pattern		0.2736	0.2726	0.2652	0.2463
	Running bond		0.2783	0.2744	0.2667	0.2521

Table 2.6 Normalized values to homogeneous-solid for the vertical strip wall

Model type		Span/Thickness ratio	Ratio of Homogeneous-Solid			
			Principal stress	Bending stress	Displacement	Von Mises stress
Homogeneous	Solid	4.21	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.17	1.17	1.40	1.20
Heterogeneous	Stack pattern		1.30	1.29	1.48	1.10
	Running bond		1.32	1.31	1.51	1.18
Homogeneous	Solid	8.42	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.13	1.13	1.12	1.13
Heterogeneous	Stack pattern		1.25	1.25	1.29	1.07
	Running bond		1.27	1.25	1.31	1.07
Homogeneous	Solid	14.74	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.12	1.12	1.06	1.12
Heterogeneous	Stack pattern		1.24	1.23	1.23	1.11
	Running bond		1.26	1.24	1.23	1.14
Homogeneous	Solid	16.84	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.12	1.12	1.05	1.12
Heterogeneous	Stack pattern		1.24	1.23	1.22	1.11
	Running bond		1.26	1.24	1.23	1.14

It is important to note that, unlike horizontally spanning wall where the block webs provide limited contribution to the effective cross-sectional area, the webs connecting the face shells in a vertically spanning wall work as box-section beam in a full span. The hollow models in vertically spanning wall generally yield stiffer response than those of horizontally spanning wall. The less difference between the solid model and the hollow models in vertical strip wall compared to horizontal strip wall might be partially due to this reason.

As was the case with horizontal spanning walls, the heterogeneous models predicted larger displacement than the homogeneous detailed models due to the presence of mortar joints with much lower elastic modulus than that of equivalent materials in homogeneous models. The two heterogeneous models were found to predict same values for the out-of-plane displacement, particularly at higher span/thickness ratios.

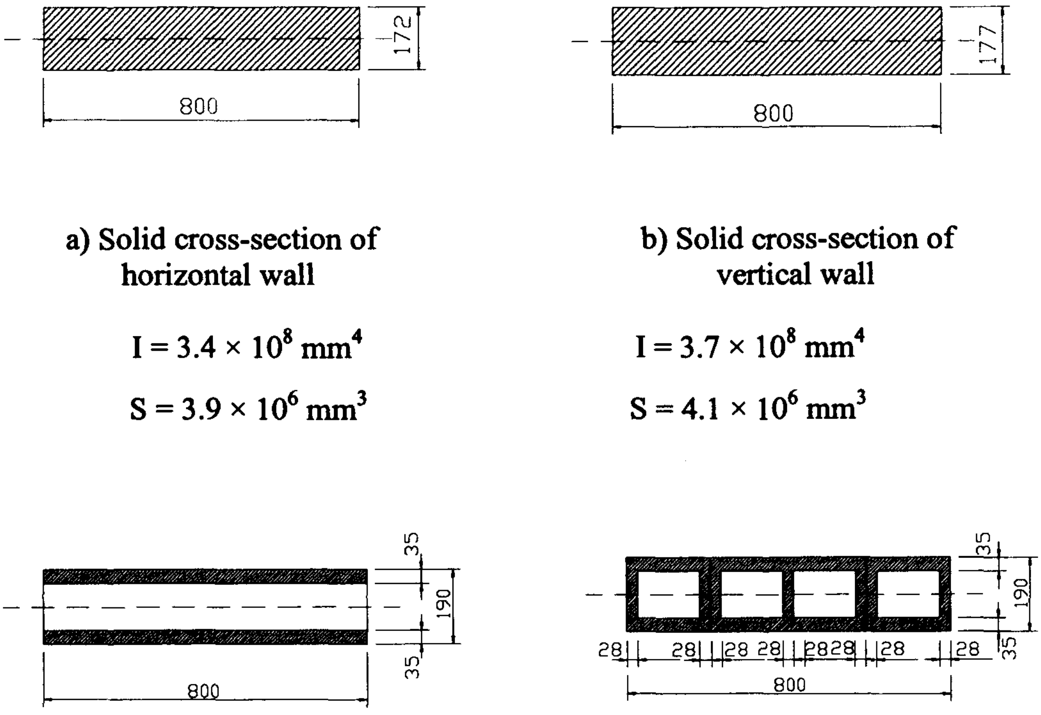
2.5.2.2 Stress

The computed maximum bending stress values from the homogeneous detailed model are on average 14% larger than those of the solid model for the four ratios. Whereas the stresses using the heterogeneous models were on average about 25% higher than those obtained from the solid model but somewhat higher at the lowest span/thickness ratio (Table 2.6). The larger bending stresses for the hollow models are predictable because the equivalent solid wall was based on moment of inertia rather than section modulus. The greater difference at lower span/thickness ratios may reflect the effects of shear deformation in the block webs.

Compared to the differences between the stresses of solid model and hollow models in horizontally spanning walls, the differences in vertically spanning walls appear to be relatively small. Examination of the representative cross-sections shown in Figure 2.13 indicates that different moments of inertia of the hollow block in the two directions yield different equivalent thickness of the solid wall for horizontal and vertical bending. To simplify the calculation, mortar joints are excluded. The percent difference between the section modulus of hollow section and solid section in vertical wall is 17%, which is less than 26% obtained in horizontal wall. This explains the comparative differences in the stress values of solid and hollow models between horizontal spanning walls and vertical spanning walls.

In accordance with displacement predictions, the detailed homogeneous model predicted lower bending stresses than the two heterogeneous models. This can be attributed to reduced local effects of the mortar joints and also to the effects of using an overall lower modulus of elasticity for the equivalent material.

The two heterogeneous models generated fairly similar results but with the running bond model values slightly higher than the stack pattern values. This is the reverse of the situation for the horizontally spanning walls. Similar observations can be made for both the principal stress and the effective stress. Again, the Von Mises stress values for the two heterogeneous models are found to



a) Solid cross-section of horizontal wall

$I = 3.4 \times 10^8 \text{ mm}^4$

$S = 3.9 \times 10^6 \text{ mm}^3$

b) Solid cross-section of vertical wall

$I = 3.7 \times 10^8 \text{ mm}^4$

$S = 4.1 \times 10^6 \text{ mm}^3$

c) Hollow cross-section of horizontal wall

$I = 3.4 \times 10^8 \text{ mm}^4$

$S = 2.9 \times 10^6 \text{ mm}^3$

d) Hollow cross-section of vertical wall

$I = 3.7 \times 10^8 \text{ mm}^4$

$S = 3.4 \times 10^6 \text{ mm}^3$

Figure 2.13 Representative cross-sections of horizontal and vertical spanning walls

be smaller than those of the principal stress and bending stress. In contrast, the detailed homogeneous model yielded same values for all stress fields.

2.5.3 Two-way Spanning Wall

For the two-way spanning masonry wall, the computed results are given in Table 2.7, where the bending stresses in the two orthogonal directions are listed separately. For the panel with aspect ratio of 0.93, the horizontal (X-direction) bending stress values are greater than the vertical (Z-direction) bending stress values. This behaviour indicates that the flexural stiffness of wall is greater along the horizontal direction; whereas, for the panel with aspect ratio of 2.07, the wall flexural stiffness is greater in the vertical direction in comparison to the horizontal direction. This behaviour is expected as it depends on the wall aspect ratio. .

The normalized data with respect to the homogeneous solid model, shown in Table 2.8, also exhibit discrepancies between the equivalent solid model and the other hollow models. At the aspect ratio of 0.93, the homogeneous detailed model predicts a 41% higher displacement value in comparison to the homogeneous solid model and this difference reduces to 15% for the panel with a ratio of 2.07. Again these results correlate with those of the horizontal spanning strips. The two wall panels have the same height but different lengths; the higher aspect ratio corresponds to higher span/thickness ratio in the horizontal direction leading to reduced shear effect. The two heterogeneous models yield slightly softer response than the homogeneous detailed models and it is also believed to be due to the presence of mortar with relatively lower elastic modulus compared to the equivalent modulus.

Examination of the controlling bending stresses in the two panels shows that the differences between the solid model and the three hollow models also decrease as the aspect ratio increases. The reduced local effects of mortar joints and effects of using lower modulus of elasticity for the equivalent materials again confirm the observation that the homogeneous detailed model predicts lower bending stresses than the heterogeneous models. The difference is about 20% in the small panel and 14% in the large panel for the controlling bending stresses. Interestingly, the two heterogeneous models predict almost the same results of the controlling bending stresses and the principal stresses as well as the Von Mises stresses. The discontinuity of the mortar joints in the heterogeneous models seems to have minor effects on the interaction of horizontal and vertical bending.

Table 2.7 Computed max values for the two-way spanning masonry wall

Model type		Aspect ratio	Principal stress (MPa)	X-Bending stress (MPa)	Z-Bending stress (MPa)	Displacement (mm)	Von Mises stress (MPa)
Homogeneous	Solid	0.93	0.0919	0.0915	0.0375	0.0318	0.0789
	Detailed		0.1220	0.1202	0.0544	0.0446	0.1000
Heterogeneous	Stack pattern		0.1504	0.1483	0.0950	0.0458	0.0940
	Running bond		0.1500	0.1480	0.0948	0.0468	0.0941
Homogeneous	Solid	2.07	0.1397	0.0479	0.1397	0.1066	0.1220
	Detailed		0.1680	0.0504	0.1679	0.1224	0.1470
Heterogeneous	Stack pattern		0.1940	0.0757	0.1920	0.1305	0.1410
	Running bond		0.1950	0.0734	0.1928	0.1312	0.1430

Table 2.8 Normalized values to homogeneous-solid for the two-way spanning masonry wall

Model type		Aspect ratio	Ratio of Homogeneous-Solid				
			Principal stress	X-Bending stress	Z-Bending stress	Displacement	Von Mises stress
Homogeneous	Solid	0.93	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.33	1.31	1.45	1.41	1.27
Heterogeneous	Stack pattern		1.64	1.62	2.53	1.44	1.19
	Running bond		1.63	1.62	2.53	1.47	1.19
Homogeneous	Solid	2.07	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.20	1.05	1.20	1.15	1.20
Heterogeneous	Stack pattern		1.39	1.58	1.37	1.22	1.16
	Running bond		1.40	1.53	1.38	1.23	1.17

2.5.4 Mortar Mechanical Property

In the previous section, a fairly low modulus of elasticity for the mortar (E_m = 1,190 MPa) was used in the analysis. The results have shown some considerable differences between the homogeneous models with equivalent materials and the heterogeneous models with block units and mortar joints modeled separately. It is prudent, therefore, to evaluate the effects of the mortar properties on the structural response. Accordingly, the mortar's elastic modulus was increased from 1,190 MPa to 3,000 MPa for the two heterogeneous models. The equivalent material properties for the homogeneous models were also calculated as per section 2.4. The revised elastic constants for the two types of models are given in Table 2.9.

Table 2.9 Modified material properties for the FE models

Model types	Material	E_x	E_y	E_z	ν_{xy}	ν_{xz}	ν_{yz}	G_{xy}	G_{xz}	G_{yz}
		(MPa)	(MPa)	(MPa)				(MPa)	(MPa)	(MPa)
Homo-geneous	Masonry	19,463	19,463	16,514	0.200	0.200	0.200	7,184	7,184	6,428
Hetero-geneous	Block	19,660	19,660	19,660	0.300	0.300	0.300	7,562	7,562	7,562
	Mortar	3,000	3,000	3,000	0.219	0.219	0.219	1,231	1,231	1,231

The computed results are normalized with respect to the homogeneous solid model and are shown in Table 2.10 to Table 2.12. It can be seen that the data follow the same pattern as before except for certain changes of the differences between the predicted values using different models. For example, for horizontal and vertical strip walls, the difference between the displacements predicted using homogeneous model and heterogeneous model is increased about 7~9% at the lowest span/thickness ratio and 3~5% at the highest ratio. Slight increment of the differences between the bending stress of the homogeneous models and the heterogeneous running bond models are also observed in horizontal strip walls. In vertical strip walls, the differences between the bending stresses of the homogeneous models and the two heterogeneous models are both slightly increased. Same observation applied to the principal stress values but the reduced differences are found for effective stress values in horizontal strip walls. In two-way spanning walls, the principal changes reflect in the reduced differences between the homogeneous solid model and the other three hollow models for both displacement and stress values. In brief, the changes are not significant.

Table 2.10 Normalized values to homogeneous solid model for the horizontal strip wall (E_{mortar} = 3000 MPa)

Model type		Span/Thickness ratio	Ratio of homogeneous-solid			
			Principal stress	Bending stress	Displacement	Von Mises stress
Homogeneous	Solid	5.26	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.99	1.96	2.31	1.95
Heterogeneous	Stack pattern		2.18	2.15	2.57	1.69
	Running bond		2.02	1.99	2.82	1.52
Homogeneous	Solid	9.47	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.49	1.47	1.55	1.46
Heterogeneous	Stack pattern		1.65	1.63	1.79	1.32
	Running bond		1.56	1.55	1.89	1.23
Homogeneous	Solid	17.89	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.32	1.31	1.26	1.30
Heterogeneous	Stack pattern		1.49	1.48	1.52	1.22
	Running bond		1.42	1.41	1.55	1.20
Homogeneous	Solid	30.52	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.25	1.24	1.15	1.23
Heterogeneous	Stack pattern		1.42	1.41	1.41	1.17
	Running bond		1.35	1.35	1.43	1.15

Table 2.11 Normalized values to homogeneous solid model for the vertical strip wall (E_{mortar} = 3000 MPa)

Model type		Span/Thickness ratio	Ratio of homogeneous-solid			
			Principal stress	Bending stress	Displacement	Von Mises stress
Homogeneous	Solid	4.21	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.17	1.17	1.43	2.52
Heterogeneous	Stack pattern		1.34	1.33	1.62	2.26
	Running bond		1.37	1.36	1.65	2.37
Homogeneous	Solid	8.42	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.13	1.13	1.12	1.13
Heterogeneous	Stack pattern		1.28	1.27	1.36	1.10
	Running bond		1.30	1.29	1.38	1.11
Homogeneous	Solid	14.74	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.12	1.12	1.06	1.12
Heterogeneous	Stack pattern		1.26	1.26	1.28	1.13
	Running bond		1.28	1.27	1.29	1.14
Homogeneous	Solid	16.84	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.12	1.12	1.05	1.12
Heterogeneous	Stack pattern		1.26	1.26	1.27	1.12
	Running bond		1.28	1.27	1.28	1.14

Table 2.12 Normalized values to homogeneous solid model for the two-way spanning wall (E_{mortar} = 3000 MPa)

Model type		Aspect ratio	Ratio of homogeneous-solid				
			Principal stress	X-Bending stress	Z-Bending stress	Displacement	Von Mises stress
Homogeneous	Solid	0.93	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.30	1.29	1.43	1.39	1.25
Heterogeneous	Stack pattern		1.54	1.53	2.26	1.40	1.13
	Running bond		1.48	1.47	2.24	1.43	1.09
Homogeneous	Solid	2.07	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.19	1.02	1.19	1.13	1.20
Heterogeneous	Stack pattern		1.34	1.90	1.32	1.20	1.11
	Running bond		1.34	1.87	1.33	1.21	1.12

Further investigation on the comparative results given in Table 2.13 to Table 2.15 indicates that much stiffer responses were yielded in the models with improved mortar's elastic modulus. For horizontal strip wall, the displacement obtained using homogeneous model is reduced 16% irrespective of the span/thickness ratios; whereas the displacement predicted using heterogeneous model decreased from 4% to 14% as the span/thickness ratio increases from 5.26 to 30.52. Only the stresses produced in the heterogeneous running bond model increase around 4%. For the vertically spanning walls, a 30% decrease in displacements of the homogeneous models was observed, whereas the heterogeneous models yielded a decrease of 24~28%. These results indicate that the mortar stiffness has a significant effect on the out-of-plane displacement of masonry walls under vertical bending. No notable changes can be observed in the stress values predicted in homogeneous models. For the heterogeneous models, slightly higher stresses are noted when the stiffness of mortar joint is increased from 1,190 MPa to 3,000 MPa. Significant reductions of displacement can also be noted in the two-way spanning walls, particularly for the wall with larger aspect ratio as more mortar joints are presented. Furthermore, small reduction in stresses was observed for the panel with aspect ratio of 0.93 and a slight increase for the panel with a ratio of 2.07.

Thus it can be summarized that, changing the mortar's modulus of elasticity from 1,190 MPa to 3,000 MPa has resulted a significant decrease in displacements, particularly for vertical strip wall and two-way spanning wall

Table 2.13 Normalized values to computed results with E_{mortar} = 1,190 MPa for the horizontal strip wall

Model type		Span/Thickness ratio	Principal stress		Bending stress		Displacement		Von Mises stress	
			Mortar 1	Mortar 2	Mortar 1	Mortar 2	Mortar 1	Mortar 2	Mortar 1	Mortar 2
Homogeneous	Solid	5.26	**1.00**	1.00	**1.00**	1.00	**1.00**	0.84	**1.00**	1.00
	Detailed		**1.00**	1.00	**1.00**	1.00	**1.00**	0.84	**1.00**	1.00
Heterogeneous	Stack pattern		**1.00**	1.00	**1.00**	1.00	**1.00**	0.96	**1.00**	0.99
	Running bond		**1.00**	1.04	**1.00**	1.04	**1.00**	0.93	**1.00**	0.93
Homogeneous	Solid	9.47	**1.00**	1.00	**1.00**	1.00	**1.00**	0.84	**1.00**	1.00
	Detailed		**1.00**	1.00	**1.00**	1.00	**1.00**	0.84	**1.00**	1.00
Heterogeneous	Stack pattern		**1.00**	1.00	**1.00**	1.00	**1.00**	0.91	**1.00**	0.99
	Running bond		**1.00**	1.04	**1.00**	1.04	**1.00**	0.90	**1.00**	0.90
Homogeneous	Solid	17.89	**1.00**	1.00	**1.00**	1.00	**1.00**	0.84	**1.00**	1.00
	Detailed		**1.00**	1.00	**1.00**	1.00	**1.00**	0.84	**1.00**	1.00
Heterogeneous	Stack pattern		**1.00**	1.00	**1.00**	1.00	**1.00**	0.87	**1.00**	0.99
	Running bond		**1.00**	1.04	**1.00**	1.04	**1.00**	0.88	**1.00**	0.95
Homogeneous	Solid	30.52	**1.00**	1.00	**1.00**	1.00	**1.00**	0.84	**1.00**	1.00
	Detailed		**1.00**	1.00	**1.00**	1.00	**1.00**	0.84	**1.00**	1.00
Heterogeneous	Stack pattern		**1.00**	1.00	**1.00**	1.00	**1.00**	0.85	**1.00**	0.99
	Running bond		**1.00**	1.03	**1.00**	1.04	**1.00**	0.86	**1.00**	0.95

Note: Mortar 1 represents mortar with E=1,190 MPa; Mortar 2 represents mortar with E=3,000 MPa.

Table 2.14 Normalized values to computed results with E_{mortar} = 1,190 MPa for the vertical strip wall

Model type		Span/Thickness ratio	Principal stress		Bending stress		Displacement		Von Mises stress	
			Mortar 1	Mortar 2	Mortar 1	Mortar 2	Mortar 1	Mortar 2	Mortar 1	Mortar 2
Homogeneous	Solid	4.21	**1.00**	1.00	**1.00**	1.00	**1.00**	0.70	**1.00**	1.00
	Detailed		**1.00**	1.00	**1.00**	1.00	**1.00**	0.71	**1.00**	0.99
Heterogeneous	Stack pattern		**1.00**	1.04	**1.00**	1.04	**1.00**	0.76	**1.00**	1.01
	Running bond		**1.00**	1.04	**1.00**	1.05	**1.00**	0.76	**1.00**	1.02
Homogeneous	Solid	8.42	**1.00**	1.00	**1.00**	1.00	**1.00**	0.69	**1.00**	1.00
	Detailed		**1.00**	1.00	**1.00**	1.00	**1.00**	0.70	**1.00**	1.00
Heterogeneous	Stack pattern		**1.00**	1.03	**1.00**	1.03	**1.00**	0.73	**1.00**	1.04
	Running bond		**1.00**	1.02	**1.00**	1.03	**1.00**	0.73	**1.00**	1.04
Homogeneous	Solid	14.74	**1.00**	1.00	**1.00**	1.00	**1.00**	0.69	**1.00**	1.00
	Detailed		**1.00**	1.00	**1.00**	1.00	**1.00**	0.69	**1.00**	1.00
Heterogeneous	Stack pattern		**1.00**	1.02	**1.00**	1.02	**1.00**	0.73	**1.00**	1.01
	Running bond		**1.00**	1.02	**1.00**	1.03	**1.00**	0.73	**1.00**	1.01
Homogeneous	Solid	16.84	**1.00**	1.00	**1.00**	1.00	**1.00**	0.69	**1.00**	1.00
	Detailed		**1.00**	1.00	**1.00**	1.00	**1.00**	0.69	**1.00**	1.00
Heterogeneous	Stack pattern		**1.00**	1.02	**1.00**	1.02	**1.00**	0.72	**1.00**	1.01
	Running bond		**1.00**	1.02	**1.00**	1.03	**1.00**	0.73	**1.00**	1.00

Note: Mortar 1 represents mortar with E=1,190 MPa; Mortar 2 represents mortar with E=3,000 MPa.

Table 2.15 Normalized values to computed results with E_{mortar} = 1,190 MPa for the two-way spanning wall

Model type		Aspect ratio	Principal stress		X-Bending Stress		Z-Bending stress		Displacement		Von Mises stress	
			Mortar 1	Mortar 2	Mortar 1	Mortar 2	Mortar 1	Mortar 2	Mortar 1	Mortar 2	Mortar 1	Mortar 2
Homogeneous	Solid	0.93	**1.00**	0.96	**1.00**	0.95	**1.00**	1.18	**1.00**	0.80	**1.00**	0.95
	Detailed		**1.00**	0.93	**1.00**	0.94	**1.00**	1.17	**1.00**	0.79	**1.00**	0.94
Heterogeneous	Stack pattern		**1.00**	0.90	**1.00**	0.90	**1.00**	1.05	**1.00**	0.78	**1.00**	0.90
	Running bond		**1.00**	0.86	**1.00**	0.86	**1.00**	1.04	**1.00**	0.78	**1.00**	0.87
Homogeneous	Solid	2.07	**1.00**	1.06	**1.00**	0.83	**1.00**	1.06	**1.00**	0.73	**1.00**	1.08
	Detailed		**1.00**	1.04	**1.00**	0.81	**1.00**	1.05	**1.00**	0.72	**1.00**	1.07
Heterogeneous	Stack pattern		**1.00**	1.02	**1.00**	1.00	**1.00**	1.02	**1.00**	0.72	**1.00**	1.04
	Running bond		**1.00**	1.02	**1.00**	1.02	**1.00**	1.02	**1.00**	0.72	**1.00**	1.03

Note: Mortar 1 represents mortar with E=1,190 MPa; Mortar 2 represents mortar with E=3,000 MPa.

using either homogeneous or heterogeneous model. For horizontal and vertical strip walls, the predictions of displacement using equivalent homogeneous models are slightly more sensitive to the material properties than using heterogeneous models.

2.1.1 Mesh Size

A source of discretization error in the finite element analysis is mesh size. In this study, the error due to mesh size is evaluated by refining the mesh and comparing the results. Accordingly, the number of elements employed to discretize the face shells and unit webs were doubled in the horizontal (X) direction and in the wall thickness (Y) direction.

Tables 2.16 to 2.18 show the normalized results with respect to the homogeneous solid model. The data display the same trend as observed for the model with coarser mesh. The differences between the predicted values using different models have changed. For horizontal strip walls with lower span/thickness ratios (5.26 and 9.47), the difference between the predicted displacements using solid models and three hollow models have increased. In contrast, these differences are found to decrease slightly for higher ratio walls. Similar observations apply to the stress values. For the vertical strip walls, the differences for the stress and displacement values between solid and hollow models are found to increase for all span/thickness ratios, but not as significant as those observed for the horizontal strip walls. For the two-way spanning walls, the changes observed in the stress values are negligible, and a small difference is noted when the displacement of the solid model is compared to those of the hollow models.

Table 2.16 Normalized values to homogeneous solid model for the horizontal strip wall (refined mesh)

Model type		Span/Thickness ratio	Ratio of homogeneous-solid			
			Principal stress	Bending stress	Displacement	Von Mises stress
Homogeneous	Solid	5.26	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		2.13	2.12	2.96	2.11
Heterogeneous	Stack pattern		2.20	2.20	2.78	1.82
	Running bond		2.09	2.09	3.03	1.74
Homogeneous	Solid	9.47	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.46	1.46	1.70	1.45
Heterogeneous	Stack pattern		1.59	1.59	1.75	1.39
	Running bond		1.49	1.47	1.84	1.41
Homogeneous	Solid	17.89	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.29	1.28	1.22	1.28
Heterogeneous	Stack pattern		1.43	1.42	1.39	1.27
	Running bond		1.48	1.48	1.41	1.32
Homogeneous	Solid	30.52	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.24	1.23	1.09	1.23
Heterogeneous	Stack pattern		1.38	1.38	1.31	1.23
	Running bond		1.44	1.44	1.30	1.29

Table 2.17 Normalized values to homogeneous solid model for the vertical strip wall (refined mesh)

Model type		Span/Thickness ratio	Ratio of homogeneous-solid			
			Principal stress	Bending stress	Displacement	Von Mises stress
Homogeneous	Solid	4.21	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.20	1.20	1.49	1.22
Heterogeneous	Stack pattern		1.32	1.32	1.59	1.13
	Running bond		1.37	1.36	1.62	1.21
Homogeneous	Solid	8.42	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.14	1.14	1.14	1.14
Heterogeneous	Stack pattern		1.26	1.26	1.33	1.09
	Running bond		1.30	1.29	1.34	1.14
Homogeneous	Solid	14.74	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.12	1.12	1.07	1.13
Heterogeneous	Stack pattern		1.25	1.24	1.25	1.14
	Running bond		1.28	1.27	1.26	1.16
Homogeneous	Solid	16.84	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.12	1.12	1.06	1.13
Heterogeneous	Stack pattern		1.25	1.24	1.24	1.14
	Running bond		1.27	1.26	1.25	1.16

Table 2.18 Normalized values to homogeneous solid model for the two-way spanning wall (refined mesh)

Model type		Aspect ratio	Ratio of homogeneous-solid				
			Principal stress	X-Bending stress	Z-Bending stress	Displacement	Von Mises stress
Homogeneous	Solid	0.93	1.00	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.31	1.30	1.58	1.53	1.31
Heterogeneous	Stack pattern		1.69	1.68	2.23	1.53	1.48
	Running bond		1.69	1.69	2.25	1.56	1.48
Homogeneous	Solid	2.07	1.00	**1.00**	**1.00**	**1.00**	**1.00**
	Detailed		1.24	1.05	1.24	1.18	1.26
Heterogeneous	Stack pattern		1.36	1.65	1.36	1.26	1.18
	Running bond		1.37	1.70	1.36	1.27	1.18

The computed results are then normalized with respect to the data predicted using coarse mesh and are listed in Table 2.19 to Table 2.21. It is evident that the refined models yield larger displacement values for all the walls. For the horizontal strip walls, the homogeneous solid model produces 12% to 16% increase in displacement when using finer mesh, as the span/thickness ratio varies from 30.52 to 5.26. The homogeneous detailed model predicted a much higher increase in displacement, 48% at the lowest ratio and 6% at the highest ratio. The two heterogeneous models gave similar trend as detailed model but slightly lower increase in displacement values. The displacements seem to be quite comparable with the unrefined-mesh results for the three hollow models at higher span/thickness ratios, where the increases are within 10%. As for the stress values, less than 10% increase or decrease is found when comparing the results using finer mesh.

Relatively close predictions using two mesh sizes can be seen from the results in vertical strip walls, where most data generated in the refined models have less than 4% increment compared to those of coarse models. This shows that the mesh refinement is not necessary for the walls under vertical bending.

For two-way spanning walls, the results of Table 2.21 show a 6% increase in the displacement for the solid model at the ratio 0.93, and essentially the same value as in the coarser model at ratio of 2.07. There are 12~15% increase in the displacement for the three hollow models when using a finer mesh for the small wall, but in the larger wall only a 4% increase is noted in the displacement of the

Table 2.19 Normalized values to the FE model with unrefined mesh for the horizontal strip wall

Model type		Span/Thickness ratio	Principal stress		Bending stress		Displacement		Von Mises stress	
			Mesh 1	Mesh 2	Mesh 1	Mesh 2	Mesh 1	Mesh 2	Mesh 1	Mesh 2
Homogeneous	Solid	5.26	**1.00**	0.95	**1.00**	0.95	**1.00**	1.16	**1.00**	0.95
	Detailed		**1.00**	1.01	**1.00**	1.03	**1.00**	1.48	**1.00**	1.03
Heterogeneous	Stack pattern		**1.00**	0.96	**1.00**	0.97	**1.00**	1.43	**1.00**	1.01
	Running bond		**1.00**	1.02	**1.00**	1.04	**1.00**	1.39	**1.00**	1.01
Homogeneous	Solid	9.47	**1.00**	0.95	**1.00**	0.95	**1.00**	1.15	**1.00**	0.95
	Detailed		**1.00**	0.93	**1.00**	0.94	**1.00**	1.26	**1.00**	0.95
Heterogeneous	Stack pattern		**1.00**	0.91	**1.00**	0.92	**1.00**	1.22	**1.00**	0.99
	Running bond		**1.00**	0.93	**1.00**	0.94	**1.00**	1.20	**1.00**	0.98
Homogeneous	Solid	17.89	**1.00**	0.93	**1.00**	0.94	**1.00**	1.14	**1.00**	0.94
	Detailed		**1.00**	0.91	**1.00**	0.92	**1.00**	1.11	**1.00**	0.93
Heterogeneous	Stack pattern		**1.00**	0.89	**1.00**	0.90	**1.00**	1.09	**1.00**	0.97
	Running bond		**1.00**	1.01	**1.00**	1.02	**1.00**	1.09	**1.00**	0.98
Homogeneous	Solid	30.52	**1.00**	0.91	**1.00**	0.92	**1.00**	1.12	**1.00**	0.92
	Detailed		**1.00**	0.91	**1.00**	0.91	**1.00**	1.06	**1.00**	0.92
Heterogeneous	Stack pattern		**1.00**	0.89	**1.00**	0.90	**1.00**	1.05	**1.00**	0.97
	Running bond		**1.00**	1.01	**1.00**	1.02	**1.00**	1.05	**1.00**	0.97

Note: Mesh 1 represents the unrefined (coarse) mesh; Mesh 2 represents the finer mesh.

Table 2.20 Normalized values to the FE model with unrefined mesh for the vertical strip wall

Model type		Span/Thickness ratio	Principal stress		Bending stress		Displacement		Von Mises stress	
			Mesh 1	Mesh 2	Mesh 1	Mesh 2	Mesh 1	Mesh 2	Mesh 1	Mesh 2
Homogeneous	Solid	4.21	**1.00**	1.00	**1.00**	1.00	**1.00**	1.00	**1.00**	1.00
	Detailed		**1.00**	1.03	**1.00**	1.03	**1.00**	1.07	**1.00**	1.02
Heterogeneous	Stack pattern		**1.00**	1.02	**1.00**	1.03	**1.00**	1.08	**1.00**	1.03
	Running bond		**1.00**	1.04	**1.00**	1.04	**1.00**	1.08	**1.00**	1.03
Homogeneous	Solid	8.42	**1.00**	1.00	**1.00**	1.00	**1.00**	1.00	**1.00**	1.00
	Detailed		**1.00**	1.01	**1.00**	1.01	**1.00**	1.02	**1.00**	1.01
Heterogeneous	Stack pattern		**1.00**	1.01	**1.00**	1.01	**1.00**	1.03	**1.00**	1.02
	Running bond		**1.00**	1.02	**1.00**	1.03	**1.00**	1.03	**1.00**	1.06
Homogeneous	Solid	14.74	**1.00**	1.00	**1.00**	1.00	**1.00**	1.00	**1.00**	1.00
	Detailed		**1.00**	1.00	**1.00**	1.00	**1.00**	1.01	**1.00**	1.01
Heterogeneous	Stack pattern		**1.00**	1.01	**1.00**	1.01	**1.00**	1.02	**1.00**	1.02
	Running bond		**1.00**	1.01	**1.00**	1.02	**1.00**	1.02	**1.00**	1.02
Homogeneous	Solid	16.84	**1.00**	1.00	**1.00**	1.00	**1.00**	1.00	**1.00**	1.00
	Detailed		**1.00**	1.00	**1.00**	1.00	**1.00**	1.00	**1.00**	1.01
Heterogeneous	Stack pattern		**1.00**	1.01	**1.00**	1.01	**1.00**	1.02	**1.00**	1.02
	Running bond		**1.00**	1.01	**1.00**	1.02	**1.00**	1.02	**1.00**	1.02

Note: Mesh 1 represents the unrefined (coarse) mesh; Mesh 2 represents the finer mesh.

Table 2.21 Normalized values to the FE model with unrefined mesh for the two-way spanning wall

Model type		Aspect ratio	Principal stress		X-Bending Stress		Z-Bending stress		Displacement		Von Mises stress	
			Mesh 1	Mesh 2	Mesh 1	Mesh 2	Mesh 1	Mesh 2	Mesh 1	Mesh 2	Mesh 1	Mesh 2
Homogeneous	Solid	0.93	**1.00**	0.86	**1.00**	0.87	**1.00**	1.06	**1.00**	1.06	**1.00**	0.87
	Detailed		**1.00**	0.85	**1.00**	0.86	**1.00**	1.16	**1.00**	1.15	**1.00**	0.89
Heterogeneous	Stack pattern		**1.00**	0.89	**1.00**	0.90	**1.00**	0.93	**1.00**	1.12	**1.00**	1.07
	Running bond		**1.00**	0.89	**1.00**	0.91	**1.00**	0.94	**1.00**	1.12	**1.00**	1.07
Homogeneous	Solid	2.07	**1.00**	1.00	**1.00**	0.81	**1.00**	1.01	**1.00**	1.01	**1.00**	1.02
	Detailed		**1.00**	1.04	**1.00**	0.81	**1.00**	1.04	**1.00**	1.04	**1.00**	1.07
Heterogeneous	Stack pattern		**1.00**	0.98	**1.00**	0.85	**1.00**	0.99	**1.00**	1.04	**1.00**	1.05
	Running bond		**1.00**	0.98	**1.00**	0.89	**1.00**	0.99	**1.00**	1.04	**1.00**	1.03

Note: Mesh 1 represents the unrefined (coarse) mesh; Mesh 2 represents the finer mesh.

three hollow models. The controlling bending stresses (X-bending stresses) in the small panel were found to decrease by 9~14% when using the refined model; whereas the controlling stresses (Z-bending stresses) in the large panel do not show a noticeable difference. Consequently, it may be concluded that the modeling of concrete block walls with larger aspect ratio is not very sensitive to the mesh size.

2.6 Conclusions

In general, the four models produced similar trends for the three wall configurations. Based on the analytical results, the following conclusions can be made:

1. A proper geometric representation of the hollow block is essential in analyzing the structural response of concrete block walls subject to out-of-plane bending.
2. Applying a solid model to represent hollow concrete masonry can result in erroneous computed values for both the displacements and stresses. The magnitude of the error can be as high as 150% and as low as 20% depending on the wall configuration and its span/thickness ratio or aspect ratio.
3. Use of solid models overestimates the structural behaviour of flexural walls by producing much stiffer responses in displacements and considerable lower stresses, particularly in the walls with lower span/thickness ratio.

4. The homogeneous detailed model generally predicts lower displacements and stresses in comparison to the two heterogeneous models except for horizontal strip walls with very low span/thickness ratio.
5. The two heterogeneous models yield close predictions of both displacements and stresses for vertically spanning walls and two-way spanning walls.
6. Effect of shear deformation was the important factor leading to the discrepancy of the results predicted using different modeling approaches. Shear effect is less significant in vertical spanning walls than in horizontal spanning walls. Plane section assumption is not valid for short spans and the non-planar behaviour is reduced with the increase of the span/thickness ratio.
7. Changing the mortar's modulus of elasticity has significant effect on the predicted displacement for all the models. Stiffer mortar resulted in an increase in the differences between the displacements predicted using the equivalent solid model and the hollow models for horizontal and vertical walls.
8. The predictions obtained using all the models for vertically spanning walls and two-way spanning walls with large aspect ratios are not sensitive to the mesh size.
9. In view of these results, the homogeneous detailed model and heterogeneous running bond model can be selected for the nonlinear analyses of the hollow block masonry walls.

CHAPTER 3

MODELING NONLINEAR STRUCTURAL RESPONSE OF CONCRETE BLOCK MASONRY

3.1 Introduction

Effects of different modeling techniques on the structural response of concrete block walls with lateral loading were presented in Chapter 2. The representation of the actual geometry of concrete block units has a notable influence on the analytical results. Three-dimensional heterogeneous model constructed by means of detailed modeling approach is shown to be the representative model capable of depicting the anisotropic behaviour of the masonry. However, linear analysis used in the previous study is not sufficient as it cannot capture the inherent inelastic characteristics of the material. Therefore, nonlinear analysis is the next step in the evaluation of the modeling techniques. Of particular interest, is modeling the influence of mortar joints, which act as planes of weakness, by way of a Composite Interface Model. This chapter, which is extracted from the work of Lourençо (1996), provides a brief description of the theory that led to the development of the composite interface element, as well as the solution procedure adopted to solve nonlinear response.

3.2 Modeling Strategies

In the last decade, the attempts to employ discrete behaviour in modeling the response of masonry structure have been focused on brick masonry. A comprehensive state of knowledge report on analytical, experimental and numerical research program has been compiled by CUR (1994) and critiqued by Lourenço (1996).

In contrast to continuum models, a discrete model needs to be capable of producing all the envisaged mechanisms of failure shown in Figure 3.1. These mechanisms include tensile cracking of the joints or units, sliding along the bed joints, and crushing of the masonry. To incorporate all these failure phenomena in a mathematical model, appropriate criterion needs to be developed for each of the potential crack planes. One rational approach is to concentrate all the damages in the relatively weak mortar joints and in the middle of each unit for the vertical cracks. For the diagonal cracking of the units and masonry crushing, given the limited combinations of compressive and shear stresses, it is most likely that the majority of the cracks will occur in the potential horizontal and vertical weak plane.

A viable approach that has been put forward to model crack, slip or crushing planes, is by incorporating interface elements, as they allow discontinuities in the displacement field and can relate the forces acting on the interface to the relative displacement of the two sides of the interface (Lourenço

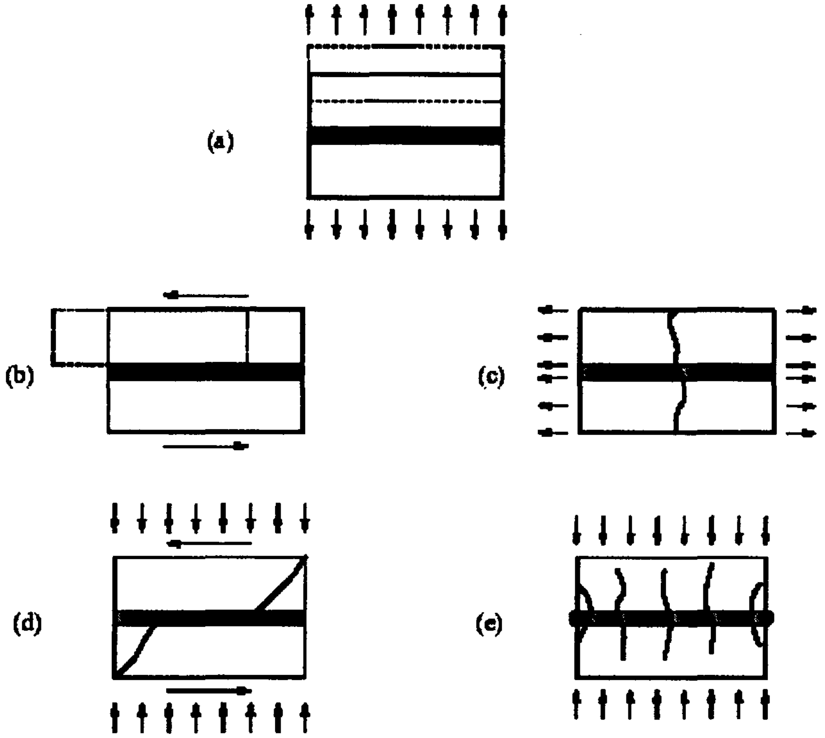


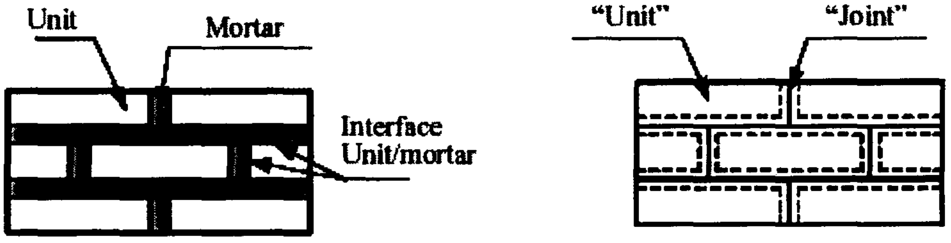
Figure 3.1 Masonry failure mechanisms, Lourenço (1996):
(a) joint tensile cracking; (b) joint slipping; (c) unit direct tensile cracking;
(d) unit diagonal tensile cracking; (e) masonry crushing.

1996). Depending on the level of refinement, two discrete models have been proposed:

1) Detailed discrete modeling – Units and mortar joints are modeled using either 2D or 3D elements. They are assumed to behave either elastically, plastically or viscoplastically. Interface elements are used to model the physical interface between units and mortar (Figure 3.2a). Although damage can occur in

the unit and the joint using the smeared crack approach, discontinuities can only occur at the interface.

2) Simplified discrete modeling – In contrast to method 1), only units are modeled using 2D or 3D elements. The mortar joint is modeled using interface elements. A schematic representation is shown in Figure 3.2b. Again, damage can occur in the unit; however, discontinuities can only occur in the mortar joint represented by interface elements.



(a) detailed discrete modeling (b) simplified discrete modeling

Figure 3.2 Discrete modeling strategies, Lourenço (1996)

Although the detailed discrete modeling provides a better representation of the assembly, it requires more effort to generate the finite element model, more elements to discretize the assembly, and knowledge of the interface's mechanical properties including the bonding strengths. The first two requirements imply more time to generate the model and to run the analysis. However, meeting the third requirement is very difficult as the interface properties are difficult to generalize because they depend on the properties of the unit and mortar, and on the moisture condition and initial surface absorption of the unit. Another complexity exists in measuring the interface properties. As a result, the simplified modeling strategy, depicted in Figure 3.2b, is selected for the current research.

While generating a three dimensional model of the assembly, attention should be given to the orientation of the interface elements that are located at the intersections between the bed joints and the head joints. A three dimensional interface element is defined by two four-node quadrilateral elements as shown in Figure 3.3. To allow the cracks to propagate along the bed joints, the orientation of the interface elements located at the intersection must be the same as those of the bed joint interface elements (Figure 3.3). And for cracks to propagate along the head joints, the orientation of the interface elements at the intersection between the head joint and bed joint must be the same as those of the head joint interface elements. This constraint could be problematic as some knowledge of the crack location and propagation needs to be a priori for the discrete model to work adequately. Fortunately, as the cracks are developed, one can modify the model so that the initiated cracks can continue to propagate.

3.3 Material Constitutive Model

A proper mathematical description of the constitutive relation for masonry work needs to be able to simulate the complete structural behaviour of masonry,

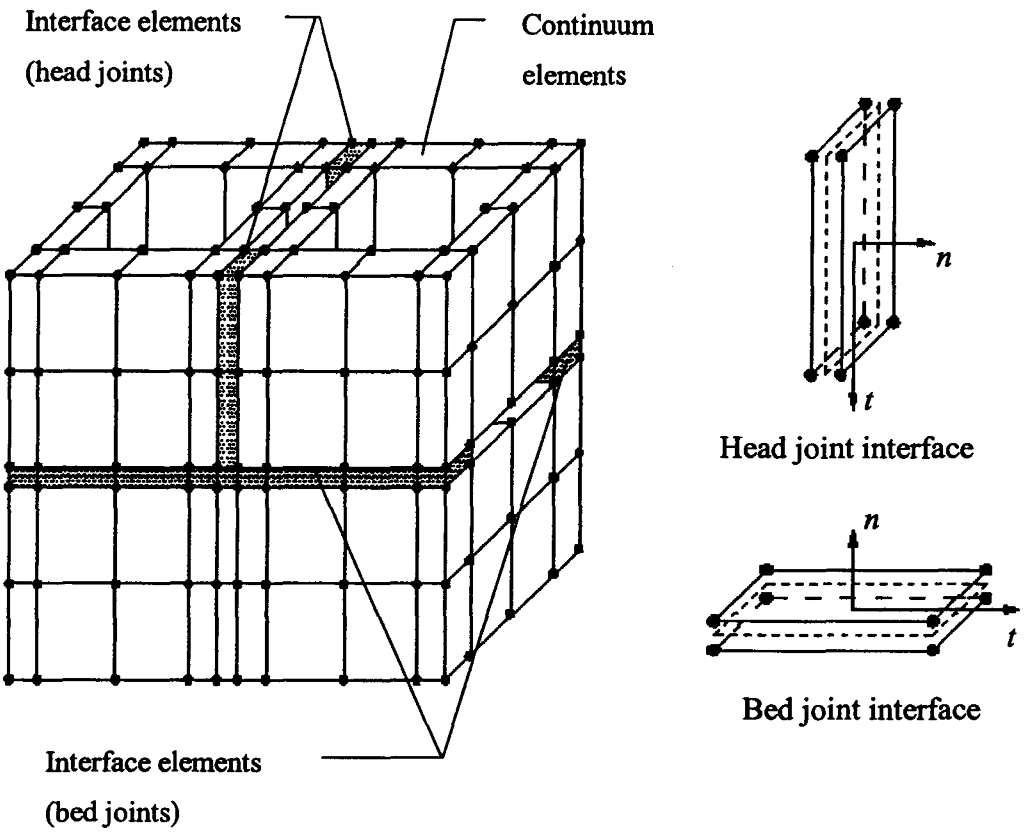


Figure 3.3 Suggested modeling strategy for 3D model

starting from the linear elastic stage, through cracking and degradation of stiffness until the total loss of strength. Such model for two-dimensional plane stress idealization was first formulated by Lourenço and Rots (1997) on the basis of multi-surface rate independent plasticity. It was later enhanced by Van Zijl (1999) and further extended to include three-dimensional modeling. The model is known as Composite Interface Model, which is defined by a convex composite yield criterion with a tension cut-off, a Coulomb friction model and an elliptical compression cap. The model has been shown to adequately simulate fracture, frictional slip and crushing along unit-mortar interface for brick masonry subjected to in-plane loading (Lourenço 1994, 1996). The Composite Interface Model is also referred to as Combined Cracking-Shearing-Crushing Model. Prior to using the Composite Interface Model, various concepts, failure criterion and theories employed for its development are first reviewed. This includes a review of the concepts of softening behaviour and the associated failure modes, and the general formulation of the theory of multi-surface plasticity. For a complete description of the mathematical model including its implementation, one should refer to Lourenço (1996).

3.3.1 Softening Behaviour and Failure Modes of Interface

Masonry is a quasi-brittle material that possesses softening behaviour with the progressive development of cracks. Softening indicates a gradual decrease in mechanical resistance with a continuous increase in deformation when a material specimen is under loading. Initially, micro cracks occur due to the presence of shrinkage and propagate with the degradation of internal stiffness. A relatively stable process of the crack development can be observed until the formation of macro cracks around the peak load, at which time the cracks become unstable. Softening has to take place with the decrease of the load to avoid an uncontrolled growth of the cracks.

Typical stress-displacement relations for quasi-brittle materials under uniaxial loading (Figure 3.4) and shear (Figure 3.5) show that softening occurs for all three failure mechanisms, tensile, compressive and shear failure. The inelastic behaviour in each mode can be described by the integral of the stress-displacement diagram, which represents the fracture energy. More precisely, in uniaxial tension, the fracture energy G_f is defined as the amount of work needed to create a stress-free crack and is equal to the area under the $\sigma - \delta$ diagram. The lower the fracture energy the more brittle the behaviour would be. Similar description applies to compressive fracture energy G_c. In masonry assemblage, two types of failure mechanism occur in the unit-mortar interface, tensile failure (type I) and shear failure (type II). For slip failure along the unit-mortar interface, the type II fracture energy G_f^{II} is defined as the area under the $\tau - \delta$ diagram in the absence of normal confining load.

According to Lourenço (1996), exponential softening curves can be derived from the experimental work of Van der Pluijm (1992, 1993) for both tensile test and shear test, as shown in Figure 3.6. In the following, the issue of the net bond surface is addressed. The bond area could be smaller than the cross sectional area of the specimen due to mortar shrinkage and construction method. It was observed that the tensile bond strength and fracture energy based on the net bound surface are larger than those respectively based on the specimen's cross-sectional area. However, test results (Figure 3.7) show that the fracture energy is

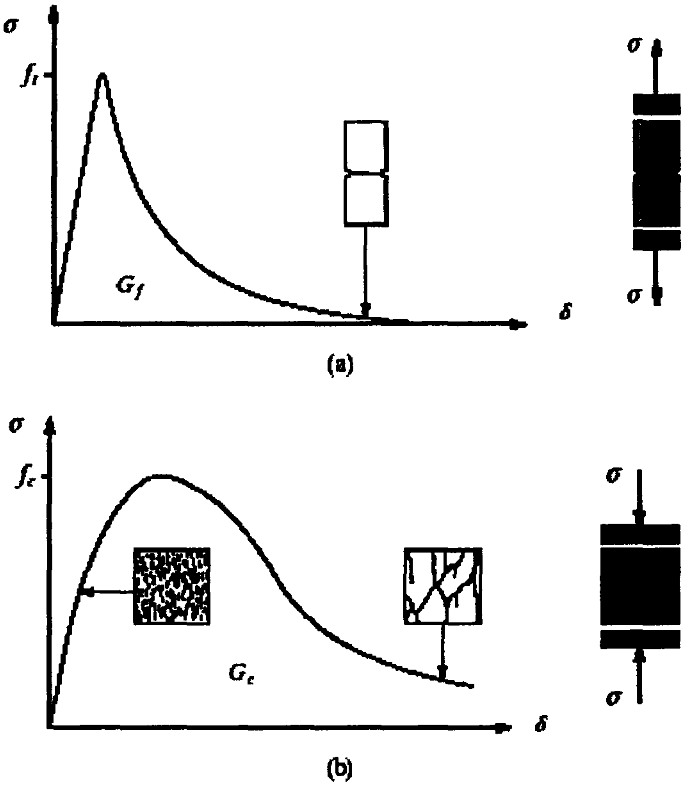


Figure 3.4 Typical behaviour of quasi-brittle materials under uniaxial loading and definition of fracture energy, Lourenço (1996): (a) tensile loading; (b) compressive loading.

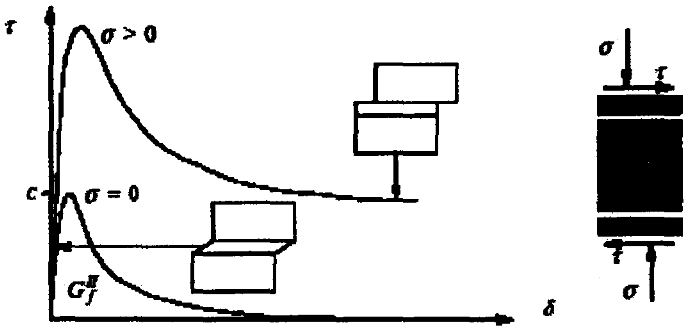


Figure 3.5 Behaviour of masonry under shear and definition of type II fracture energy, Lourenço (1996)

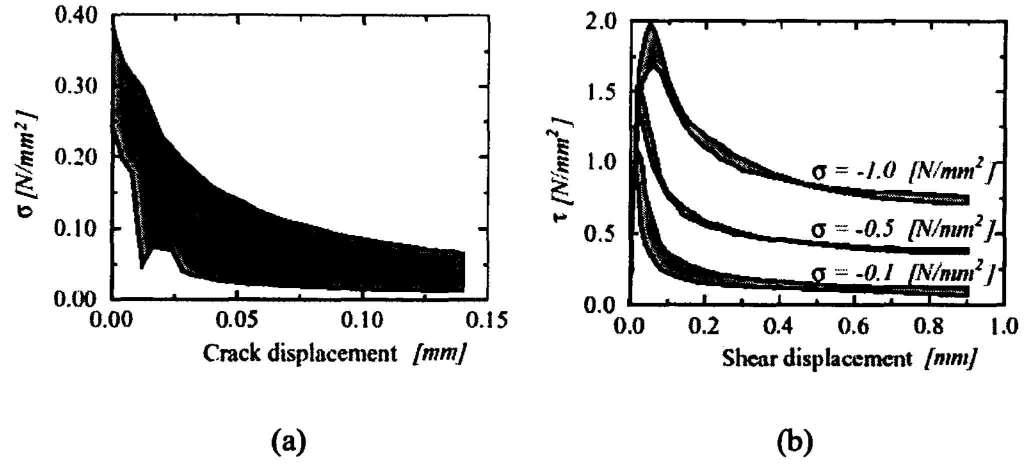
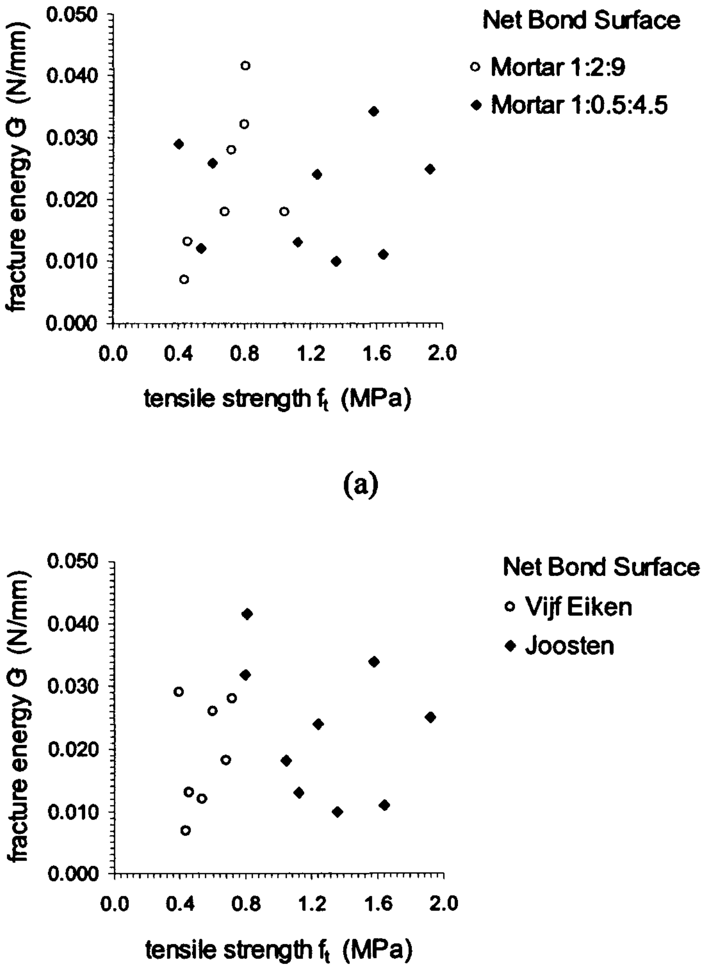


Figure 3.6 Tensile/shear bond behaviour for clay brick masonry:
(a) stress-crack displacement diagram, Van der Pluijm (1992);
(b) stress-shear displacement diagram, Van der Pluijm (1993). (the shaded area represents the envelope for three tests)

independent of the tensile strength. It is noteworthy, when the net bond surface is taken into account, the difference in tensile bond strengths or fracture energy for different unit-mortar combinations is reduced. This observation gives support to the adoption of tensile bond strength and fracture energy for characterizing the properties at the interface when few experimental data is available. The scatter in the tensile bond strength and fracture energy is found small despite the use of different mortar strengths (Van der Pluijm 1992).

Type II fracture energy was found to depend on the initial cohesion c values and the level of confining stress. In that regard, Van der Pluijm (1993) have carried the shear bond test to quantify all the material parameters needed for the Coulomb friction model. This includes initial internal friction angleϕ_0, residual internal friction angle ϕ_r (Figure 3.8a) and the dilatancy angle ψ (Figure 3.8b). The dilatancy angle describes the relation between the lateral displacement and the shear displacement. As the normal confining stress increases, the value of $\tan\psi$ decreases to zero as shown in Figure 3.9. This is due to the fact that the



(a)

(b)

Figure 3.7 Fracture energy of joints versus tensile bond strength, Van de Pluijm (1992): (a) arranged by mortar; (b) arranged by unit-type.

crack surface tends to be smooth during the shear movement as the normal stress increases.

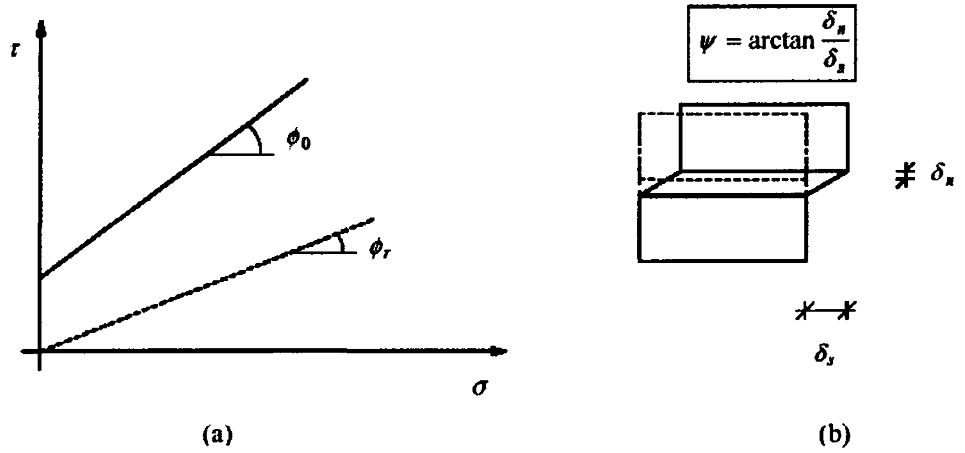


Figure 3.8 Definition of friction and dilatancy angles, Lourenço (1996):
(a) Coulomb friction law, with initial and residual friction angle;
(b) dilatancy angle as the uplift of neighboring units upon shearing.

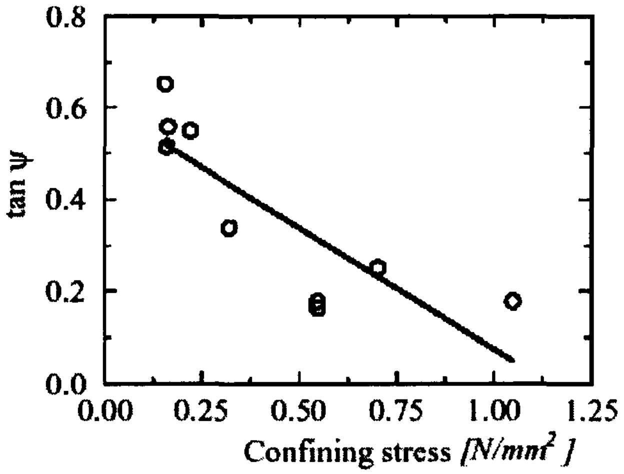


Figure 3.9 Tangent of the dilatancy angle ψ as a function of the normal stress level, Van der Pluijm (1993)

3.3.2 General Formulation of Multi-surface Plasticity

Plastic material behaviour differs from elastic behaviour as a non-unique stress-strain relationship exists during the deformation of the structure under loading. No permanent deformations occur in the elastic range whereas permanent or irreversible deformations can be observed in the plastic range. In the context of small strains, the strain rate vector $\dot{\varepsilon}$ is usually decomposed into an elastic, reversible part, $\dot{\varepsilon}^e$, and an irreversible, or plastic part, $\dot{\varepsilon}^p$:

$$\dot{\varepsilon} = \dot{\varepsilon}^e + \dot{\varepsilon}^p \tag{3.1}$$

The stress and strain history can be described based on the flow theory of plasticity by taking into account some fundamental assumptions. Firstly, applying the elastic stress-strain relation, the elastic strain rate can be related to the total stress rate by

$$\dot{\sigma} = \mathrm{D}\dot{\varepsilon}^e \tag{3.2}$$

with D is the elastic (constitutive) stiffness matrix.

Another basic assumption in plasticity, the non-associated flow rule, which specifies the inelastic or plastic strain rate vector as a function of the state of stress, yields

$$\dot{\varepsilon}^p = \dot{\lambda}\frac{\partial g(\sigma,\kappa)}{\partial\sigma} \tag{3.3}$$

where $\dot{\lambda}$ is the plastic multiplier rate; and g the plastic potential function. The latter is a function of the stress vector σ and the internal state parameter κ. The internal parameter κ is governed by a specific evolution law, which describes the amount of hardening or softening, also named as hardening parameter. The hardening law specifies the evolution of κ as a function of the stress vector and the plastic strain rate vector, i.e., $\dot{\kappa} = h(\sigma, \dot{\varepsilon}^p)$. In the case of strain hardening (or softening), it gives

$$\dot{\kappa} = \sqrt{(\dot{\varepsilon}^p)^T \dot{\varepsilon}^p} \tag{3.4}$$

and for work hardening (or softening) per unit of volume,

$$\dot{\kappa} = \dot{W}^p = \sigma^T \dot{\varepsilon}^p \tag{3.5}$$

The yield condition describes the situation at the boundary of elastic domain and inelastic domain. It specifies the state of stress at which the plastic flow is initiated. Yielding occurs if the stresses σ reach the values such that

$$f(\sigma, \kappa) = 0 \tag{3.6}$$

A negative value indicates an elastic state whereas a positive value is not admissible for rate-independent plasticity.

Equation (3.3) describes the flow rule for single surface plasticity. For multi-surface plasticity, according to Koiter's rule (1953), the plastic strain rate vector should be given by

$$\dot{\varepsilon}^p = \sum_{j=1}^{n} \dot{\lambda}_j \frac{\partial g_j}{\partial \sigma} \tag{3.7}$$

in which the plastic multipliers $\dot{\lambda}_j$ are restricted by the standard Kuhn-Tucker conditions

$$\begin{cases} f_j \le 0 \\ \dot{\lambda}_j \ge 0 \\ \dot{\lambda}_j f_j = 0 \end{cases} \tag{3.8}$$

Clearly, if no plastic flow occurs, $\dot{\lambda}_j$ is equal to zero. For composite yield surface defined by two yield surfaces, the plastic strain rate will be

$$\dot{\varepsilon}^p = \dot{\varepsilon}_1^p + \dot{\varepsilon}_2^p = \dot{\lambda}_1 \frac{\partial g_1}{\partial \sigma} + \dot{\lambda}_2 \frac{\partial g_2}{\partial \sigma} \tag{3.9}$$

The composite hardening scalar rates $\dot{\kappa}_1^c$ and $\dot{\kappa}_2^c$ can also be introduced to couple the respective hardening rate for the two yield surfaces:

$$\dot{\kappa}_1^c = \dot{\kappa}_1^c(\dot{\kappa}_1, \dot{\kappa}_2) \qquad \dot{\kappa}_2^c = \dot{\kappa}_2^c(\dot{\kappa}_1, \dot{\kappa}_2) \tag{3.10}$$

where $\dot{\kappa}_1 = \dot{\kappa}_1(\dot{\varepsilon}_1^p)$ and $\dot{\kappa}_2 = \dot{\kappa}_2(\dot{\varepsilon}_2^p)$ as defined in from equations (3.4) or (3.5).

3.3.2.1 Formation of Nonlinear Equations

The derivation of the nonlinear equations can be established from the additive decomposition of the strain vector. Without loss of generality, at a stage "n", the elastic stress vector is related to the strain vector by

$$\sigma_n = \mathrm{D}\varepsilon_n^e = \mathrm{D}(\varepsilon_n - \varepsilon_n^p) \tag{3.11}$$

At this stage the total strain vector ε_n, the plastic strain vector, ε_n^p, and the internal state parameter(s), $\kappa_{j,n}$, for different yield surfaces are known. In addition, the incremental strain vector, $\Delta\varepsilon_{n+1}$, follows from the loading regime and the total strain can be updated by

$$\varepsilon_{n+1} = \varepsilon_n + \Delta\varepsilon_{n+1} \tag{3.12}$$

To compute the stress vector at stage n+1, the plastic strain vector and the internal state parameters also need to be updated in a consistent manner. This is achieved by integration of the rate equations defined in the flow rules and hardening laws. As recounted by Lourenço (1996), by applying the fully implicit Euler Backward algorithm, the problem is transformed into a constrained optimization problem governed by discrete Kuhn-Tucker conditions as shown by Simo et al. (1988). For single surface plasticity, this algorithm results in a discrete set of equations:

$$\begin{cases} \sigma_{n+1} = \sigma^{trial} - \mathrm{D}\Delta\varepsilon_{n+1}^p \\ \Delta\varepsilon_{n+1}^p = \Delta\lambda_{n+1} \dfrac{\partial g}{\partial \sigma} |_{n+1} \\ \Delta\kappa_{n+1} = \Delta\kappa_{n+1}(\sigma_{n+1}, \Delta\varepsilon_{n+1}^p) \\ f_{n+1}(\sigma_{n+1}, \kappa_{n+1}) = 0 \end{cases} \tag{3.13}$$

where $\Delta\kappa_{n+1}$ is taken from the integration of equation (3.4) or (3.5). Since the algorithm is considered within an elastic predictor-plastic corrector algorithm, the elastic trial stress

$$\sigma^{trial} = \sigma_n + \mathrm{D}\Delta\varepsilon_{n+1} \tag{3.14}$$

is introduced and, if it is outside the yield surface the stress update should be corrected by the method of return mapping.

Simplified algorithm can be obtained if the plastic potential, g, has separate variables. In this case, the function can be written as $g(\sigma,\kappa) = \Phi(\sigma) + \Omega(\kappa)$. After some manipulation, equation (3.13) can be transformed to a set of equations with the updated plastic multiplier $\Delta\lambda_{n+1}$ as the single variable:

$$\begin{cases} \sigma_{n+1} = \sigma_{n+1}(\Delta\lambda_{n+1}) \\ \Delta\kappa_{n+1} = \Delta\kappa_{n+1}(\sigma_{n+1}, \Delta\lambda_{n+1}) \\ f_{n+1}(\Delta\lambda_{n+1}) = 0 \end{cases} \tag{3.15}$$

Equation (3.15)$_3$ is solved with regular Newton-Raphson method. The necessary derivative of $f_{n+1}(\Delta\lambda_{n+1})$ for this procedure reads

$$\frac{\partial f}{\partial \Delta\lambda}|_{n+1} = \gamma^T \frac{\partial \sigma}{\partial \Delta\lambda} - h \tag{3.16}$$

with the modified yield surface gradient γ and the hardening modulus h given by

$$\gamma = \frac{\partial f}{\partial \sigma} + \frac{\partial f}{\partial \kappa}\frac{\partial \kappa}{\partial \sigma}|_{n+1} \qquad h = -\frac{\partial f}{\partial \kappa}\frac{\partial \kappa}{\partial \Delta\lambda}|_{n+1} \tag{3.17}$$

For multi-surface plasticity, similar approach can be applied to derive the following nonlinear equations:

$$\begin{cases} \sigma_{n+1} = \sigma_{n+1}(\Delta\lambda_{1,n+1}, \Delta\lambda_{2,n+1}) \\ \Delta\kappa^c_{1,n+1} = \Delta\kappa^c_{1,n+1}(\sigma_{n+1}, \Delta\lambda_{1,n+1}, \Delta\lambda_{2,n+1}) \\ \Delta\kappa^c_{2,n+1} = \Delta\kappa^c_{2,n+1}(\sigma_{n+1}, \Delta\lambda_{1,n+1}, \Delta\lambda_{2,n+1}) \\ f_{1,n+1}(\Delta\lambda_{1,n+1}, \Delta\lambda_{2,n+1}) = 0 \\ f_{2,n+1}(\Delta\lambda_{1,n+1}, \Delta\lambda_{2,n+1}) = 0 \end{cases} \tag{3.18}$$

The only unknowns in this system are the two scalars, $\Delta\lambda_{1,n+1}$ and $\Delta\lambda_{2,n+1}$. Again, equations $(3.18)_{4,5}$ are solved by applying Newton-Raphson iterative method. The Jacobian needed in this procedure reads

$$\mathbf{J}_{n+1} = \begin{bmatrix} \gamma_1^T \dfrac{\partial\sigma}{\partial\Delta\lambda_1} - h_1 & \gamma_1^T \dfrac{\partial\sigma}{\partial\Delta\lambda_2} + \dfrac{\partial f_1}{\partial\kappa_1^c}\dfrac{\partial\kappa_1^c}{\partial\Delta\lambda_2} \\ \gamma_2^T \dfrac{\partial\sigma}{\partial\Delta\lambda_1} + \dfrac{\partial f_2}{\partial\kappa_2^c}\dfrac{\partial\kappa_2^c}{\partial\Delta\lambda_1} & \gamma_2^T \dfrac{\partial\sigma}{\partial\Delta\lambda_2} - h_2 \end{bmatrix} \tag{3.19}$$

where

$$\gamma_j = \frac{\partial f_j}{\partial\sigma} + \frac{\partial f_j}{\partial\kappa_j^c}\frac{\partial\kappa_j^c}{\partial\sigma}\Big|_{n+1} \qquad h_j = -\frac{\partial f_j}{\partial\kappa_j^c}\frac{\partial\kappa_j^c}{\partial\Delta\lambda_j}\Big|_{n+1} \tag{3.20}$$

with j = 1, 2.

An important issue that arises in the implementation of multi-surface plasticity is how to determine the number of active yield surfaces. This problem exists because the location of the intersection between the two yield surfaces, defined as the corner, is unknown at the beginning of the load step, and because there is no sufficient criterion that can be provided during the trial and error procedures to determine which yield surface is active at the end of the load step.

Based on the work of Simo et al. (1988), Pramono and Willam (1989) and Feenstra (1993), Lourenço (1996) suggested that a trial and error procedure be followed according to Pramono and Willam (1989). The set of initial active yield functions is taken from Simo et al. (1988) ($f_j^{trial} \geq 0$). Upon the return mapping is completed, as long as any $\Delta\lambda_{j,n+1} < 0$ or $f_{j,n+1} > 0$ exists, the number of active yield surface should be adjusted and the return mapping restarted.

3.3.2.2 Derivation of the Tangent Stiffness Matrix

The consistent linearization of the nonlinear equations can be used to derive the tangent stiffness matrix, which plays a crucial role in the performance and robustness of the iterative Newton-Raphson method. It has been pointed out by Simo and Taylor (1985), that the tangent stiffness matrix must be obtained by consistent linearization of the updated stress resulting from the return-mapping algorithm at the end of iteration n+1 (Figure 3.10).

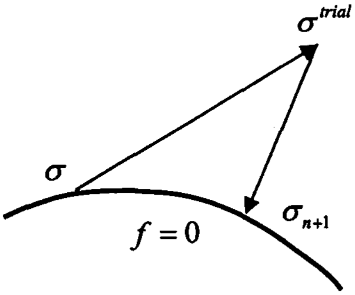


Figure 3.10 Return-mapping algorithm

For single surface plasticity, under the assumption that the plastic potential function g has separate variables, a standard expression for the consistent tangent stiffness matrix D^{ep} was derived by Lourenço (1996) based on equation (3.13) and the Sherman-Morrison formula:

$$\mathrm{D}^{ep} = \frac{d\sigma}{d\varepsilon}|_{n+1} = \mathrm{H} - \frac{\mathrm{H}\frac{\partial \mathrm{g}}{\partial \sigma}\gamma^T \mathrm{H}}{h + \gamma^T \mathrm{H}\frac{\partial \mathrm{g}}{\partial \sigma}} \tag{3.21}$$

with the modified stiffness matrix H defined as

$$\mathrm{H} = \left[\mathrm{D}^{-1} + \Delta\lambda_{n+1}\frac{\partial^2 g}{\partial \sigma^2}\right]^{-1} \tag{3.22}$$

Similarly, for multi-surface plasticity, the tangent stiffness matrix was obtained by applying Riggs and Powel (1990) formulation:

$$\mathrm{D}^{ep} = \frac{d\sigma}{d\varepsilon}|_{n+1} = \mathrm{H} - \mathrm{HU}(\mathrm{E}+\mathrm{V}^T\mathrm{HU})^{-1}\mathrm{V}^T\mathrm{H} \tag{3.23}$$

where the modified stiffness matrix H reads

$$\mathrm{H} = \left[\mathrm{D}^{-1} + \Delta\lambda_{1,n+1}\frac{\partial^2 g_1}{\partial \sigma^2} + \Delta\lambda_{2,n+1}\frac{\partial^2 g_2}{\partial \sigma^2}\right]^{-1} \tag{3.24}$$

the gradient matrices U and V read

$$\mathrm{U} = \left[\frac{\partial \mathrm{g}_1}{\partial \sigma} \quad \frac{\partial \mathrm{g}_2}{\partial \sigma}\right] \qquad \mathrm{V} = [\gamma_1 \quad \gamma_2] \tag{3.25}$$

and the hardening matrix E reads

$$\mathrm{E}=\begin{bmatrix} -h_1 & \dfrac{\partial f_1}{\partial \kappa_1^c}\dfrac{\partial \kappa_1^c}{\partial \Delta\lambda_2} \\ \dfrac{\partial f_2}{\partial \kappa_2^c}\dfrac{\partial \kappa_2^c}{\partial \Delta\lambda_1} & -h_2 \end{bmatrix} \tag{3.26}$$

in which the modified yield surface gradient γ_j and the hardening modulus h_j were defined in equation (3.20).

3.3.3 Failure Criterion of Composite Interface Model

Given the general formulation of a plasticity-based constitutive model, the yield criterion of the three components of the Composite Interface Model, and the composite criterion for two yield surfaces, can be defined accordingly. It is necessary to give a brief introduction of the two-dimensional interface model before the three-dimensional model is presented.

For the two-dimensional configuration, the interface model is described in terms of the generalized stress and strain vectors:

$$\sigma=\begin{Bmatrix}\sigma\\ \tau\end{Bmatrix} \qquad \varepsilon=\begin{Bmatrix}\Delta u_n\\ \Delta u_s\end{Bmatrix} \tag{3.27}$$

with σ and Δu_n, the stress and relative displacement, respectively, in the interface normal direction; and τ and Δu_s, the shear stress and relative displacement, respectively. In the elastic regime the constitutive behaviour can be described as

$$\sigma=\mathrm{D}\varepsilon \tag{3.28}$$

where the elastic stiffness matrix

$$D=\begin{bmatrix} k_n & 0 \\ 0 & k_s \end{bmatrix} \tag{3.29}$$

can be determined from the properties of the masonry constituents, unit and mortar, and the joint thickness. The expressions of k_n and k_s, under the assumption of stack bond and uniform stress distributions, both in the unit and mortar, were first reported in CUR (1994):

$$k_n = \frac{E_u E_m}{h_m(E_u - E_m)} \qquad k_s = \frac{G_u G_m}{h_m(G_u - G_m)} \tag{3.30}$$

where E_u and E_m are the Young's modulii, and G_u and G_m the shear modulii. h_m is the actual thickness of the joint. The subscript *u* and *m* denote the unit and mortar, respectively. The effect of running bond connection of the components, i.e., the overlap of neighboring units subjected to compression, is not taken into account in these formulae but is "intrinsic to the interface elements formulation and is independent of the values of normal stiffness (Lourenço 1996)". The accuracy of the formulae given by equation (3.30) was proven by the comparison of the analytical results for a 2D shear wall model using "continuum model" and "interface model" (Lourenço 1996).

The Composite Interface Model, as shown in Figure 3.11, consists of a tension cut-off model for type I failure, a Coulomb friction model for type II failure and a cap model for compressive failure, each of which is defined by a

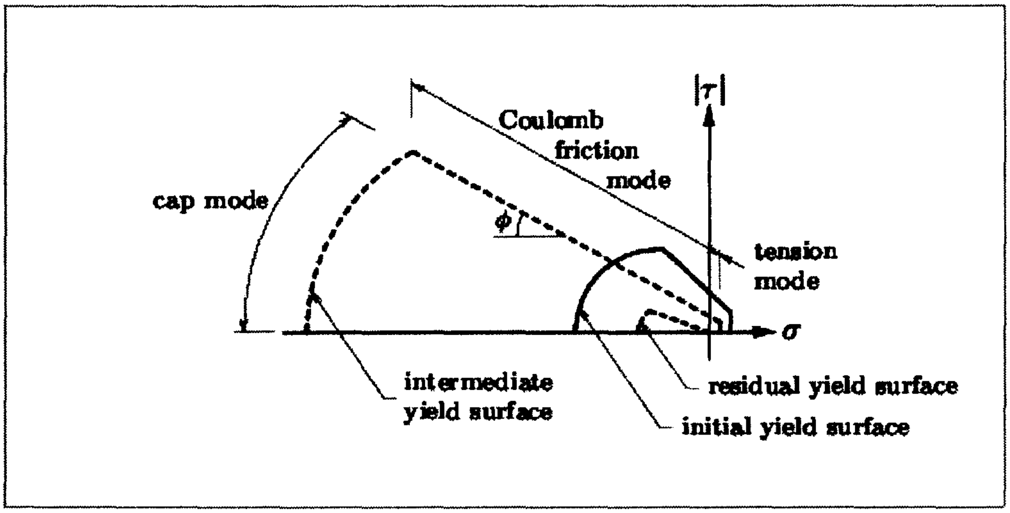


Figure 3.11 Two-dimensional interface model, TNO DIANA BV (2003)

yield criterion as a function of the stress vector σ and the corresponding internal state parameter κ. The first two failure modes are presented below based on the previous discussion. The formulation of compression cap model is not considered in this study because it has not yet been extended to a three-dimensional analysis.

3.3.3.1 The Tension Cut-off Criterion

The yield function for the tension cut-off is defined as

$$f_1(\sigma,\kappa_1)=\sigma-\bar{\sigma}_1(\kappa_1) \tag{3.31}$$

where the yield value $\bar{\sigma}_1$ indicates the tensile strength of the interface, which is assumed to soften exponentially:

$$\bar{\sigma}_1(\kappa_1) = f_t e^{-\frac{f_t}{G_f^{\mathrm{I}}}\kappa_1} \tag{3.32}$$

with f_t the bond strength of the mortar joint and G_f^{I} the Mode-I fracture energy. The softening is governed by a strain-softening hypothesis, and if the normal plastic relative displacement is assumed to control the softening behaviour, equation (3.4) becomes:

$$\dot{\kappa}_1 = \left|\Delta \dot{u}_n^p\right| \tag{3.33}$$

which, upon the consideration of an associated flow rule

$$\dot{\varepsilon}^p = \dot{\lambda}_1 \frac{\partial f_1}{\partial \sigma} \tag{3.34}$$

reduces to

$$\dot{\kappa}_1 = \dot{\lambda}_1 \tag{3.35}$$

Therefore, the update of normal stress and shear stress can be obtained from equation $(3.13)_1$:

$$\begin{cases} \sigma_{n+1} = \sigma^{trial} - \Delta\lambda_{1,n+1} k_n \\ \tau_{n+1} = \tau^{trial} \end{cases} \tag{3.36}$$

and the derivative necessary for the iterative local Newton-Raphson method, from equation (3.16), yields

$$\frac{\partial f_1}{\partial \Delta\lambda_1}|_{n+1} = -k_n - h_1 \tag{3.37}$$

where

$$h_1 = \frac{\partial \bar{\sigma}_1}{\partial \kappa_1} \tag{3.38}$$

The corresponding consistent tangent stiffness matrix, from equation (3.21), reads

$$\mathrm{D}_1^{ep} = \frac{\partial \sigma}{\partial \varepsilon}|_{n+1} = \begin{bmatrix} \dfrac{h_1 k_n}{h_1 + k_n} & 0 \\ 0 & k_s \end{bmatrix} \tag{3.39}$$

3.3.3.2 The Coulomb Friction Criterion

The shear-slipping is described by a Coulomb friction yield criterion

$$f_2(\sigma, \kappa_2) = |\tau| + \sigma \tan\phi(\kappa_2) - \bar{\sigma}_2(\kappa_2) \tag{3.40}$$

where the yield value, $\bar{\sigma}_2$, describes the adhesion softening and is defined as

$$\bar{\sigma}_2(\kappa_2) = c e^{-\frac{c}{G_f^{\mathrm{II}}}\kappa_2} \tag{3.41}$$

ϕ, the friction angle, indicates friction softening that is coupled with adhesion softening according to

$$\tan\phi(\kappa_2) = \tan\phi_0 + (\tan\phi_r - \tan\phi_0)\frac{c - \bar{\sigma}_2(\kappa_2)}{c} \tag{3.42}$$

In the above, c is the adhesion of the unit-mortar interface, G_f^{II} the shear-slip or mode-II fracture energy, ϕ_0 the initial friction angle, and ϕ_r the residual friction angle. These parameters can be determined from the shear-bond test noted in section 3.3.1. The experimentally observed linear relation between the fracture energy and the normal confining stress is captured by letting

$$G_f^{\mathrm{II}} = \begin{cases} a\sigma + b & \text{if } \sigma < 0 \\ b & \text{if } \sigma \geq 0 \end{cases} \tag{3.43}$$

in which the constants a and b are determined from experimental data using linear regression. In the case of significant contribution of the friction softening, a and b need to be adjusted to avoid a too-high energy dissipation at high compressive stress.

A non-associated flow rule

$$\dot{\varepsilon}^p = \begin{Bmatrix} \dot{u}_n^p \\ \dot{u}_s^p \end{Bmatrix} = \dot{\lambda}\frac{\partial g}{\partial \sigma} \tag{3.44}$$

is necessary in this case as a suitable potential function g may be chosen to describe the dilatancy ψ, which measures the normal uplift induced by shear-slip:

$$g(\sigma, \kappa_2) = |\tau| + \sigma \tan\psi(\sigma, \kappa_2) - c \tag{3.45}$$

Note that the dilatancy angle depends on the confining stress and the plastic relative shear displacement. When the strain-softening hypothesis is employed, where the softening behaviour is governed by shear-slipping, equation (3.4) yields

$$\dot{\kappa}_2 = \left|\Delta \dot{u}_s^p\right| = \dot{\lambda}_2 \tag{3.46}$$

upon substitution of equation (3.44) and (3.45).

The stress update can be obtained from equation (3.13)₁:

$$\begin{cases} \sigma_{n+1} = \sigma^{trial} - \Delta\lambda_{2,n+1} k_n \tan\psi \\ \tau_{n+1} = \tau^{trial} - \Delta\lambda_{2,n+1} k_s \operatorname{sign}(\tau^{\mathrm{trial}}) \end{cases} \tag{3.47}$$

in which τ_{n+1} and τ^{trial} should have the same sign. The derivative necessary for the iterative local Newton-Raphson scheme, from equation (3.16), reads

$$\frac{\partial f}{\partial \Delta\lambda_2}|_{n+1} = -k_n \tan\phi \tan\psi - k_s - h_2 \tag{3.48}$$

where

$$h_2 = \left[\sigma_{n+1}\left(\frac{\tan\phi_r - \tan\phi_0}{c}\right) + 1\right]\frac{\partial \bar{\sigma}_2}{\partial \kappa_2} \tag{3.49}$$

The corresponding consistent tangent stiffness matrix is obtained from equation (3.21):

$$\mathrm{D}_2^{ep} = \frac{\partial\sigma}{\partial\varepsilon}|_{n+1} = \frac{1}{h_2 + k_n \tan\phi \tan\psi + k_s} \cdot \begin{bmatrix} k_n(h_2 + k_s) & -k_n k_s \tan\psi \operatorname{sign}(\tau_{n+1}) \\ -k_n k_s \tan\phi \operatorname{sign}(\tau_{n+1}) & (h_2 + k_n \tan\phi \tan\psi) k_s \end{bmatrix} \tag{3.50}$$

3.3.3.3 The Composite Yield Criterion of Tension and Shear

Tensile failure (type I) and shear failure (type II) are the two major mechanisms in the unit-mortar interface. The coupling of the tension and shear softening results in a composite yield surface, which can be defined by an appropriate combination of two single yield surfaces. A natural consideration is the assumption of isotropic softening, that the amounts of the tensile and shear strength degradation are equal. This seems reasonable since both softening behaviour are associated with the breakage of the bond or adhesion through the same carrier, the unit-mortar interface.

To define the isotropic softening at the tension/shear corner, the composite softening scalar rates $\dot{\kappa}_1^c$ and $\dot{\kappa}_2^c$ must be derived. As the softening scalar rate for tension or shear yield surface is already obtained via equations (3.35) and (3.46), the remaining work is to determine the update of the softening scalar of the corresponding mode due to the effect of the other mode. Firstly, assuming only the tension mode is active ($\Delta\lambda_{1,n+1} > 0$, $\Delta\lambda_{2,n+1} = 0$), the yield value at the end of the iterative procedure, from equation (3.32), will be

$$\bar{\sigma}_{1,n+1} = f_t e^{-\frac{f_t}{G_f^I}\kappa_{1,n+1}^c} = \bar{\sigma}_{1,n} e^{-\frac{f_t}{G_f^I}\Delta\kappa_{1,n+1}^c} \tag{3.51}$$

with $\Delta\kappa_{1,n+1}^c = \Delta\lambda_{1,n+1}$. If the cohesion softening is coupled with tension softening, from equation (3.41), the yield value $\bar{\sigma}_{2,n+1}$ reads

$$\bar{\sigma}_{2,n+1} = ce^{-\frac{c}{G_f^{II}}\kappa_{2,n+1}^c} = \bar{\sigma}_{2,n}e^{-\frac{c}{G_f^{II}}\Delta\kappa_{2,n+1}^c} \tag{3.52}$$

In this case, $\Delta\kappa_{2,n+1}^c$ must be calculated based on the isotropic softening assumption. Therefore, the following equality should hold:

$$\frac{\bar{\sigma}_{1,n+1}}{\bar{\sigma}_{1,n}} = \frac{\bar{\sigma}_{2,n+1}}{\bar{\sigma}_{2,n}} \tag{3.53}$$

and it yields

$$\Delta\kappa_{2,n+1}^c = \frac{G_f^{II}}{G_f^I}\frac{f_t}{c}\Delta\lambda_{1,n+1} \tag{3.54}$$

Similar approach can be carried out if only the shear mode is active and yields

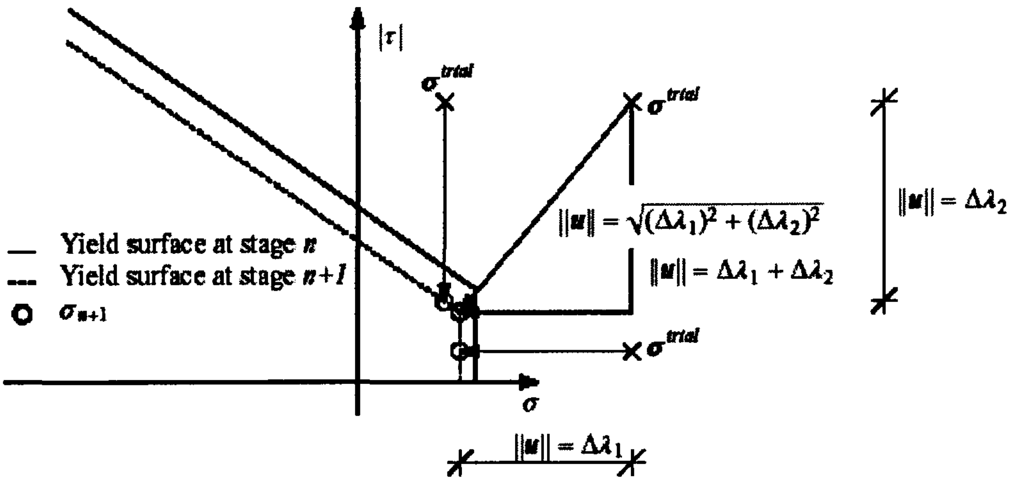
$$\Delta\kappa_{1,n+1}^c = \frac{G_f^I}{G_f^{II}}\frac{c}{f_t}\Delta\lambda_{2,n+1} \tag{3.55}$$

Then, the softening scalar rates for the composite yield surface are defined according to equations (3.10), (3.35) and (3.46):

$$\dot{\kappa}_1^c = \dot{\lambda}_1 + \frac{G_f^I}{G_f{}^{II}}\frac{c}{f_t}\dot{\lambda}_2 \qquad \dot{\kappa}_2^c = \frac{G_f^{II}}{G_f^I}\frac{f_t}{c}\dot{\lambda}_1 + \dot{\lambda}_2 \tag{3.56}$$

However, these equations overestimate the softening at the tension/shear corner, which can be illustrated by Figure 3.12. As recounted in Lourenço (1996), by the plasticity standard definition (Chen and Han 1988), the yield surface shrinks in the stress space but keeps the origin. The amount of softening is measured by the distance between the trial stress state and the stress state at stage n+1, i.e., $\Delta\lambda_{j,n+1} = \left|\sigma^{trial} - \sigma_{n+1}\right|$. In Figure 3.12, the norm of vector u controls the amount of softening for each case. The measurement in the corner of tension and shear yield surface, which is a singular point, should be determined by a quadratic combination, rather than a simple addition, of the softening of two single yield surfaces:

$$\dot{\kappa}_1^c = \sqrt{(\dot{\lambda}_1)^2 + (\frac{G_f^I}{G_f^{II}}\frac{c}{f_t}\dot{\lambda}_2)^2} \qquad \dot{\kappa}_2^c = \sqrt{(\frac{G_f^{II}}{G_f^I}\frac{f_t}{c}\dot{\lambda}_1)^2 + (\dot{\lambda}_2)^2} \qquad (3.57)$$



$|\tau|$
σ^{trial}
σ^{trial}
$\|u\| = \sqrt{(\Delta\lambda_1)^2 + (\Delta\lambda_2)^2}$
$\|u\| = \Delta\lambda_1 + \Delta\lambda_2$
$\|u\| = \Delta\lambda_2$
— Yield surface at stage n
--- Yield surface at stage $n+1$
○ σ_{n+1}
σ^{trial}
σ
$\|u\| = \Delta\lambda_1$

Figure 3.12 Definition of the amount of softening for the tension/shear corner, Lourenço (1996)

The stress update equations can be obtained by following the procedure introduced in section 3.3.2:

$$\begin{cases} \sigma_{n+1} = \sigma^{trial} - \Delta\lambda_{1,n+1} k_n - \Delta\lambda_{2,n+1} k_n \tan\psi \\ \tau_{n+1} = \tau^{trial} - \Delta\lambda_{2,n+1} k_s \mathrm{sign}(\tau^{trial}) \end{cases} \tag{3.58}$$

The Jacobian necessary for the iterative local Newton-Raphson method can be manipulated from equation (3.19):

$$\mathbf{J}_{n+1} = \begin{bmatrix} j_{11} & j_{12} \\ j_{21} & j_{22} \end{bmatrix} \tag{3.59}$$

with

$$\begin{aligned} j_{11} &= -k_n - \frac{\partial \bar{\sigma}_1}{\partial \kappa_1^c} \frac{\Delta\lambda_{1,n+1}}{\Delta\kappa_{1,n+1}^c} \\ j_{12} &= -k_n \tan\psi - \frac{\partial \bar{\sigma}_1}{\partial \kappa_1^c} \left(\frac{G_f^I}{G_f^{II}} \frac{c}{f_t} \right)^2 \frac{\Delta\lambda_{2,n+1}}{\Delta\kappa_{1,n+1}^c} \\ j_{21} &= -k_n \tan\phi - \left[\sigma_{n+1} \left(\frac{\tan\phi_r - \tan\phi_0}{c} \right) + 1 \right] \frac{\partial \bar{\sigma}_2}{\partial \kappa_2^c} \left(\frac{G_f^{II}}{G_f^I} \frac{f_t}{c} \right)^2 \frac{\Delta\lambda_{1,n+1}}{\Delta\kappa_{2,n+1}^c} \\ j_{22} &= -k_n \tan\phi \tan\psi - k_s - \left[\sigma_{n+1} \left(\frac{\tan\phi_r - \tan\phi_0}{c} \right) + 1 \right] \frac{\partial \bar{\sigma}_2}{\partial \kappa_2^c} \frac{\Delta\lambda_{2,n+1}}{\Delta\kappa_{2,n+1}^c} \end{aligned} \tag{3.60}$$

The corresponding consistent tangent stiffness matrix is evaluated from equation (3.23).

3.3.3.4 Three-dimensional Interface Model

The two-dimensional interface model was extended to a three-dimensional interface model by Van Zijl (1999). This development enables the description of delamination (tension cut-off) and relative shear-slipping of two planes (Coulomb friction). The generalized stress and strain vectors for a 3D configuration are:

$$\sigma = \begin{Bmatrix} \sigma \\ \tau_s \\ \tau_t \end{Bmatrix} \qquad \varepsilon = \begin{Bmatrix} \Delta u_n \\ \Delta u_s \\ \Delta u_t \end{Bmatrix} \tag{3.61}$$

where σ and Δu_n are the stress and relative displacement, respectively, normal to the interface plane; τ_s, τ_t are the shear stress acting in the interface plane; and Δu_s, Δu_t are the relative shearing displacements. The stiffness matrix is defined as

$$\mathrm{D} = \begin{bmatrix} k_n & 0 & 0 \\ 0 & k_s & 0 \\ 0 & 0 & k_t \end{bmatrix} \tag{3.62}$$

The three-dimensional tension cut-off and Coulomb friction yield surface are described in Figure 3.13.

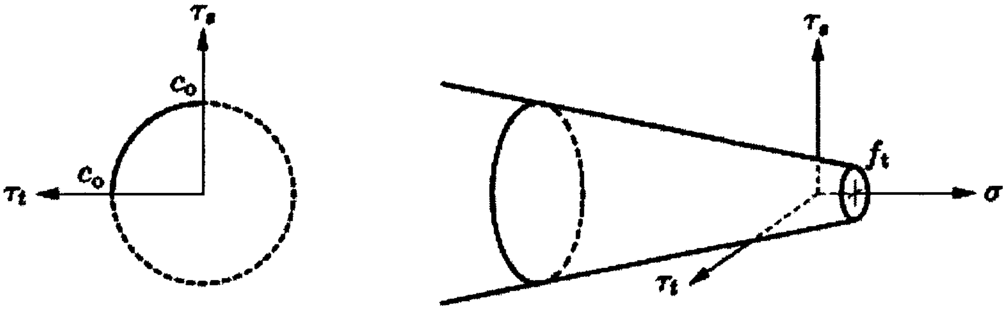


Figure 3.13 Three-dimensional interface model, TNO DIANA BV (2003)

In the three-dimensional interface model, tension cut-off yield criterion remains the same as that described in (3.31) for two-dimensional case. Because of the additional shear stress and strain components, the Coulomb friction yield function was modified to

$$f_2(\sigma,\kappa_2)=\sqrt{\tau_s^2+\tau_t^2}+\sigma\tan\phi(\kappa_2)-\bar{\sigma}_2(\kappa_2) \tag{3.63}$$

Similar to the two-dimensional case, the adhesion softening and friction softening are both modeled as described by equation (3.41) and (3.42). By choice of a suitable potential function, a non-associated flow rule is given as

$$\dot{\varepsilon}^p=\dot{\lambda}\frac{\partial g}{\partial\sigma}=\dot{\lambda}\begin{Bmatrix}\Psi\\ \dfrac{\tau_s}{\sqrt{\tau_s^2+\tau_t^2}}\\ \dfrac{\tau_t}{\sqrt{\tau_s^2+\tau_t^2}}\end{Bmatrix}=\begin{Bmatrix}\Delta\dot{u}_n^p\\ \Delta\dot{u}_s^p\\ \Delta\dot{u}_t^p\end{Bmatrix} \tag{3.64}$$

where $\Psi=\tan\psi(\sigma,\kappa_2)$, the mobilized dilatancy coefficient, is a function of the confining stress and the shear-slip. Assuming that the strain softening is controlled by the equivalent relative shear displacement, the softening scalar rate reads

$$\dot{\kappa}_2=\sqrt{\left(\Delta\dot{u}_s^p\right)^2+\left(\Delta\dot{u}_t^p\right)^2}=\dot{\lambda}_2 \tag{3.65}$$

3.4 Nonlinear Finite Element Analysis

To simulate the plastic material behaviour of masonry structure, nonlinear finite element analysis must be conducted in the current study. Similar to the linear analysis, the relations between the nodal forces and displacements in the elemental domains are first created then assembled in the presence of external loads and boundary conditions to obtain the system of equations. However, since the relations between the force vectors and displacement vectors are nonlinear and the displacements usually depend upon the displacements at earlier stages, the solution procedure in nonlinear case will be more complicated. The purpose is to calculate the displacement vectors that equilibrate the internal and external forces and solve the stress and strain vectors based on the derived nodal displacements.

This section presents a brief description of element types and incremental-iterative solution procedure adopted in the analysis. The analysis is carried out using DIANA finite element program, in which the Composite Interface Model has been implemented. More detailed information can be found in TNO DIANA BV (2003). A complete derivation of the nonlinear finite element method can be obtained from Zienkiewicz and Taylor (2000) and Bathe (1996).

3.4.1 Finite Elements

In the foregoing discussion, continuum elements and interface elements were proposed to represent masonry units and mortar joints in simplified discrete modeling. For three-dimensional model, the lower order continuum elements such as 8-node brick elements are widely used because of the small bandwidths of the resulting global system of equations. However, one should note that linearly interpolated isoparametric elements have intrinsic shortcomings, like parasitic shear and volumetric locking, which can not be easily dealt with in nonlinear analysis (TNO DIANA BV 2003, Vol. Analysis Procedure). Nonetheless, due to the limitation of computational facilities, the 8-node brick element is chosen to model block units in this study to reduce the structure size of the full-scale wall model. To ensure the correct element connectivity, 4+4-node plane quadrilateral interface elements are used for mortar joints. Figure 3.14 illustrates the two finite element types and the corresponding axis and variable definitions.

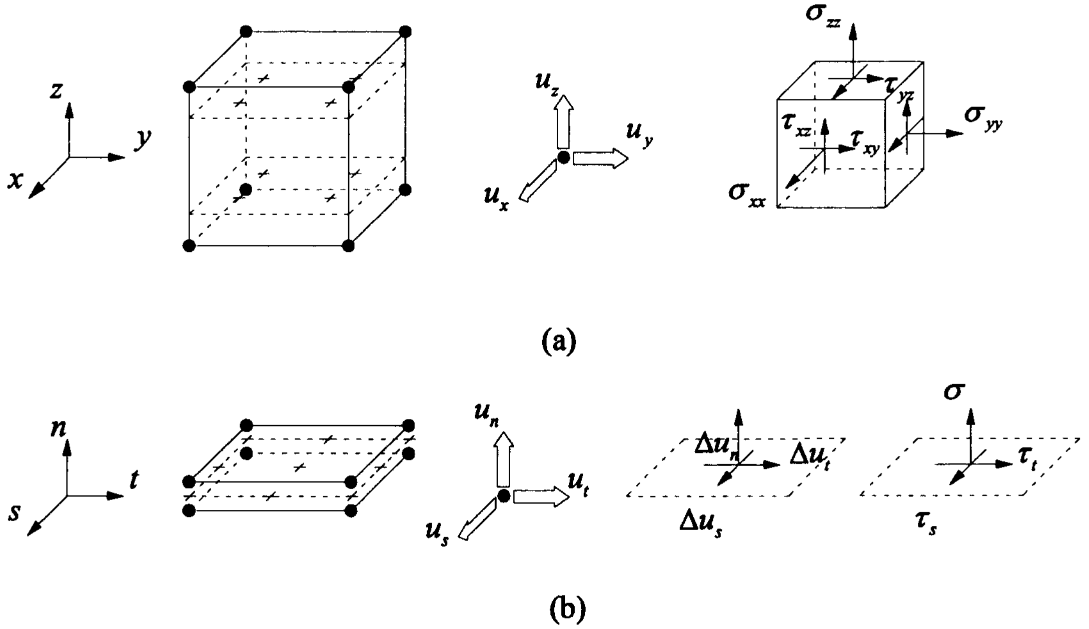


Figure 3.14 Element types adopted in this study:
(a) eight-node brick element with Gauss integration; (b) 4+4-node plane quadrilateral interface element with Newton-Cotes integration.

Usually, the local *x*, *y* and *z* axes in the brick elements are set up parallel to the global *X*, *Y* and *Z* axes, respectively. The basic variables in the nodes of brick elements are the translations u_x, u_y and u_z in the local element directions. The stress and strain tensors can be described as

$$\begin{aligned} \sigma &= \left\{\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \tau_{xy}, \tau_{yz}, \tau_{xz}\right\}^T \\ \varepsilon &= \left\{\varepsilon_{xx}, \varepsilon_{yy}, \varepsilon_{zz}, \gamma_{xy}, \gamma_{yz}, \gamma_{xz}\right\}^T \end{aligned} \tag{3.66}$$

Brick elements are normally integrated with a Gauss integration scheme. In this study, a 2×2×2 integration scheme is considered. For interface elements, variables are oriented in the local *nst* axes. The constitutive relation is established between the traction t and the relative displacements Δu across the interface, which are defined by

$$\begin{aligned} \mathrm{t} &= \left\{\sigma, \tau_s, \tau_t\right\}^T \\ \Delta\mathrm{u} &= \left\{\Delta u_n, \Delta u_s, \Delta u_t\right\}^T \end{aligned} \tag{3.67}$$

An appropriate integration scheme must be chosen for interface elements. As pointed out by Rots (1988), Schellekens (1992) and Lourenço (1996), the exact Gauss integration may lead to erroneous oscillations of stresses under certain conditions. In this study a 3×3 Newton-Cotes integration scheme is used.

3.4.2 Solution Procedures for Nonlinear Systems

The solution of the nonlinear problem can be obtained by solving the displacement vector from the equilibrium between the internal and external force, with the boundary conditions satisfied:

$$\mathbf{f}_{\text{int}} = \mathbf{f}_{\text{ext}} \tag{3.68}$$

For nonlinear plasticity, the internal force vector is usually path dependent, i.e., it depends nonlinearly on the displacements and the displacements' history. Therefore, to determine the state of equilibrium, it is necessary to have the system discretized not only in space (with finite elements), but also in time (with increments). Here the 'time' often indicates a sequence of situations. For a time increment between stage *n* (or time *t*) and stage *n*+1 (or time $t+\Delta t$), the approximated solution $\mathbf{u}_n$ (displacement history) and the time increment are known. Hence the internal force only depends on the displacement increment $\Delta\mathbf{u}$. The external forces only depend on the current geometry. The nonlinear problem become finding $\Delta\mathbf{u}$ such that

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \Delta\mathbf{u} \tag{3.69}$$

and, **g,** the residual force vector (or unbalanced forces),

$$\mathbf{g}(\Delta\mathbf{u}) = \mathbf{f}_{\text{ext}}(\Delta\mathbf{u}) - \mathbf{f}_{\text{int}}(\Delta\mathbf{u}) = 0 \tag{3.70}$$

However, it is hardly possible to achieve equilibrium at the end of the increment within one step for the problems showing strong nonlinearity. Moreover, a purely incremental method can lead to inaccurate solutions. An iterative process is therefore required to achieve convergence and reduce the errors that occur in the analysis. Consequently, the solution procedure of a nonlinear system is a combination of incremental and iterative procedure.

3.4.2.1 Iterative Procedures

The typical iterative process during one time increment is given in Figure 3.15. In all iteration processes, the total displacement increment $\Delta\mathbf{u}$ is updated iteratively by iterative increment $\delta\mathbf{u}$ until equilibrium is reached, up to a prescribed tolerance. At iteration i+1, the incremental displacements are calculated from

$$\Delta\mathbf{u}_{i+1} = \Delta\mathbf{u}_{i} + \delta\mathbf{u}_{i+1} \tag{3.71}$$

where, the iterative increments can be determined from the linearized form of the relation between the force vector and displacement vector, $\mathbf{g}_\mathbf{i} = \mathbf{K}_\mathbf{i}\delta\mathbf{u}_\mathbf{i}$, which yields

$$\delta\mathbf{u}_\mathbf{i} = \mathbf{K}_\mathbf{i}^{-1}\mathbf{g}_\mathbf{i} \tag{3.72}$$

with $\mathbf{g}_\mathbf{i}$ the residual vector and $\mathbf{K}_\mathbf{i}$ the stiffness matrix at the start of iteration *i*.

The most frequently used iteration schemes for the solution of nonlinear finite element equations are based on the classical Newton-Raphson technique. Generally, two types of Newton-Raphson methods are widely used: the Regular Newton-Raphson and the Modified Newton-Raphson method. In both methods, the iterative increment of the displacement vector is determined from equation (3.72), where the stiffness matrix $\mathbf{K}_\mathbf{i}$ represents the tangential stiffness of

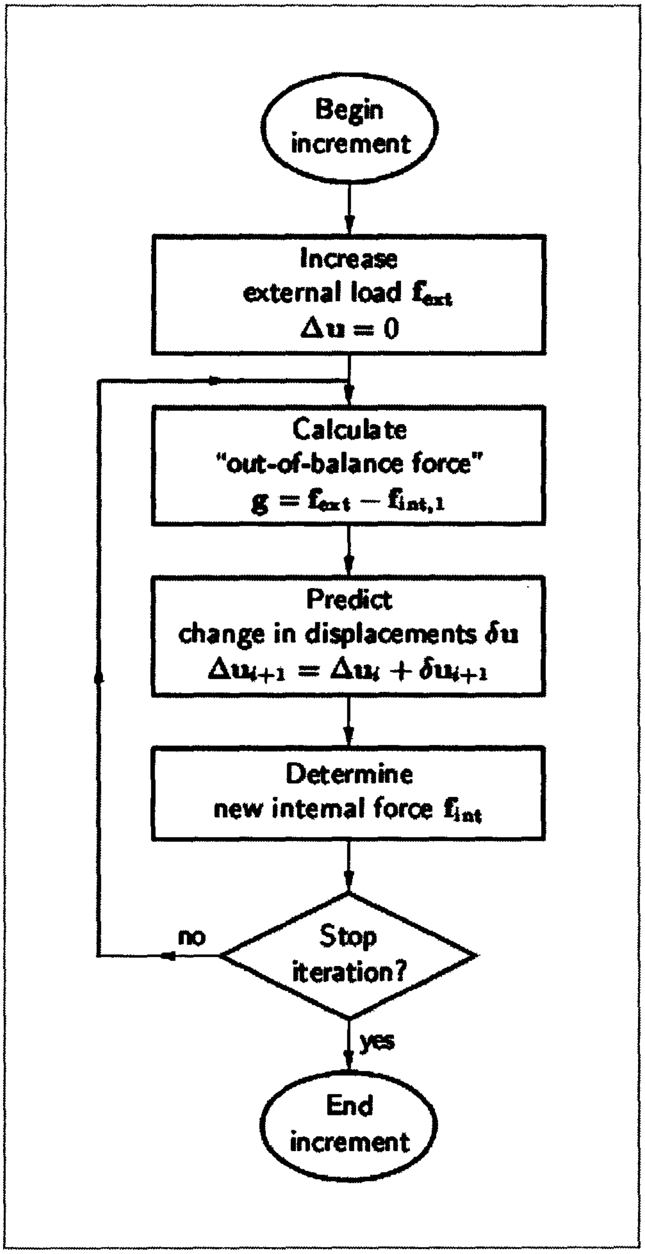

Begin increment
Increase external load $\mathbf{f}_{ext}$ $\Delta\mathbf{u} = 0$
Calculate "out-of-balance force" $\mathbf{g} = \mathbf{f}_{ext} - \mathbf{f}_{int,1}$
Predict change in displacements $\delta\mathbf{u}$ $\Delta\mathbf{u}_{i+1} = \Delta\mathbf{u}_i + \delta\mathbf{u}_{i+1}$
Determine new internal force $\mathbf{f}_{int}$
Stop iteration?
no
yes
End increment

Figure 3.15 Iteration process, TNO DIANA BV (2003)

the structure:

$$\mathbf{K}_i = \frac{\partial \mathbf{g}}{\partial \Delta \mathbf{u}} \tag{3.73}$$

A characteristic of the Regular Newton-Raphson iteration is that a new tangent stiffness matrix is evaluated in each iteration (see Figure 3.16), which is why it is also referred to as the Full Newton-Raphson method. This approach indicates that tangent stiffness matrix is always based on the last predicted result, which might not be in an equilibrium state. Although, the Regular Newton-Raphson enables a quadratic convergence, it requires the calculation and factorization of the stiffness matrix at each iteration, resulting in a large computational cost. Moreover, the quadratic convergence is only guaranteed in the sense that a correct evaluation of stiffness matrix is performed, and the prediction is already close to the final solution. Otherwise, the method will easily diverge (TNO DIANA BV 2003).

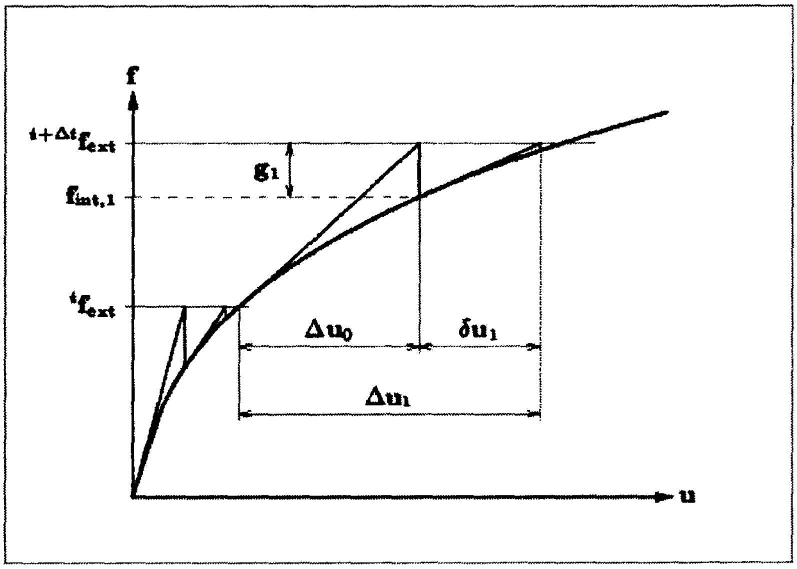


Figure 3.16 Regular Newton-Raphson iteration, TNO DIANA BV (2003)

A severe limitation of the Newton-Raphson method was noted by Lourenço (1996). He observed that the method is not globally convergent, i.e., convergent to some solution from almost any starting point. This stems from the fact that masonry structures show strong nonlinearity in the presence of cracking. In this case, the initial prediction is usually too far from equilibrium and the iteration process will easily diverge. The preferred strategy for dealing with this problem is the technique of Line Search (Crisfield 1991), which can be used to increase the convergence rate.

The essential part of the Line Search algorithm is to search for an optimal magnification factor η that can be used to scale the iterative displacement increment $\delta\mathbf{u}$ such that the energy potential Π can be minimized. This is understandable as the local minimum of the energy potential represents the equilibrium; and the minimum in the line search direction can be regarded as the best solution in the predicted direction. The approach is initiated by writing

$$\Delta\mathbf{u}_{\mathbf{i+1}} = \Delta\mathbf{u}_{\mathbf{i}} + \eta\delta\mathbf{u}_{\mathbf{i+1}} \tag{3.74}$$

A minimum of Π in the line search direction indicates that the derivative of Π to η must be zero:

$$s(\eta) = \frac{\partial\Pi}{\partial\eta} = \frac{\partial\Pi}{\partial\mathbf{u}}\frac{\partial\mathbf{u}}{\partial\eta} = \mathbf{g}(\eta)^T\delta\mathbf{u} = 0 \tag{3.75}$$

Therefore the line search factor η is determined such that the projection of the residual force $\mathbf{g}$ in the search direction $\delta\mathbf{u}_{\mathbf{i+1}}$ equals to zero. Recognizing that

Line Search method is used to accelerate the convergence rate and not to seek the exact solution, it is not really necessary to find η such that $s(\eta) = 0$ holds. A common practice to terminate the line search requires that

$$|s(\eta)| \leq \Psi |s(0)| \tag{3.76}$$

where $s(0)$ is evaluated by equation (3.75) at the start of the iteration, see Figure 3.17. Here Ψ is a tolerance factor and is usually taken as 0.8, which is sufficient to ensure the global convergence for Newton-Raphson method. The upper and lower bounds of η are specified to avoid unrealistic values.

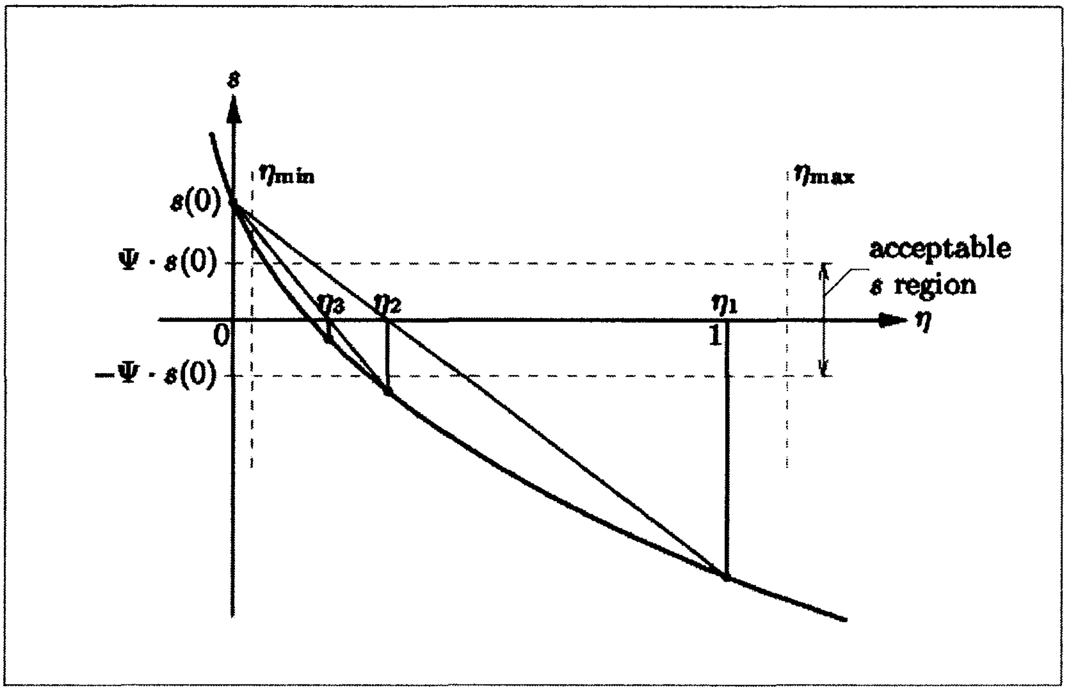


Figure 3.17 Line Search iteration, TNO DIANA BV (2003)

3.4.2.2 Convergence Criteria

The iteration process can be set to terminate either when it converges within a preset tolerance, or when the specified maximum number of iterations has been reached, implying a non-convergent solution. An appropriate convergence tolerance needs to balance the solution accuracy and the computational effort. Three convergence norms are used to check the solution at the end of iteration: displacement norm, force norm and energy norm. The detection of divergence is based on the same norms as the detection of convergence.

The displacement norm is the Euclidian norm of the iterative displacement increment $\delta \mathbf{u}_{\mathbf{i}}$. It is checked against the norm of the displacement increments in the first prediction of the increment, $\Delta \mathbf{u}_{\mathbf{0}}$:

$$\frac{\sqrt{\delta \mathbf{u}_{\mathbf{i}}^{\mathbf{T}} \delta \mathbf{u}_{\mathbf{i}}}}{\sqrt{\Delta \mathbf{u}_{\mathbf{0}}^{\mathbf{T}} \Delta \mathbf{u}_{\mathbf{0}}}} \leq \varepsilon_D \tag{3.77}$$

The force norm is the Euclidian norm of the out-of-balance force vector **g**. It is checked against the norm of the initial unbalanced force $\mathbf{g}_{\mathbf{0}}$:

$$\frac{\sqrt{\mathbf{g}_{\mathbf{i}}^{\mathbf{T}} \mathbf{g}_{\mathbf{i}}}}{\sqrt{\mathbf{g}_{\mathbf{0}}^{\mathbf{T}} \mathbf{g}_{\mathbf{0}}}} \leq \varepsilon_F \tag{3.78}$$

The energy norm is composed of internal forces and relative displacements. The convergence is checked by measuring the increment in internal energy during each iteration, $\delta \mathbf{E}_i$, against the initial internal energy increment, $\Delta \mathbf{E}_0$:

$$\frac{\delta \mathbf{E}_i}{\Delta \mathbf{E}_0} = \left| \frac{\delta \mathbf{u}_i^T (\mathbf{f}_{int,i+1} + \mathbf{f}_{int,i})}{\Delta \mathbf{u}_0^T (\mathbf{f}_{int,1} + \mathbf{f}_{int,0})} \right| \leq \varepsilon_E \tag{3.79}$$

In the above, ε_D, ε_F and ε_E are the convergence tolerances of displacement, force and energy, respectively. Since the energy convergence criterion contains measurements of both the displacements and forces, it is believed to be more stable in comparison to the other two criteria. This criterion is used throughout the analyses in this study. The convergence tolerance, ε_E, is set to 0.0001.

3.4.2.3 Incremental Procedures

In general, two types of increment controls can be employed to calculate the structural response: load control and displacement control. The most natural approach, called 'load control', is to increase the external load vector $\mathbf{f}_{ext}$ directly at the start of the increment. As the load increases, the structural response becomes increasingly nonlinear while approaching the peak load. With the load increment drastically reduced, it is still difficult to establish post peak equilibrium because the slope of the load-displacement curve approaches zero and the stiffness matrix becomes singular. In this situation, a special solution procedure that allows for a decrease in load and an increase in displacement must be used to predict the subsequent response. This problem can be avoided if the external load is put on the structure in the form of a prescribed displacement. This method is called 'displacement control', in which the external force vector is not increased directly but increment is made to the displacement vector. However, as summarized by Lourenço (1996), the direct displacement control "is unable to handle strong localizations and suffers from lack of generality since only a limited number of engineering problems can be modeled in such a way". For the walls subjected to uniformly distributed out-of-plane loading, as in this study, it is not possible to substitute the external load by prescribed displacements.

To overcome the deficiencies arising from the above two methods, an arc-length control method can be used, as in essence proposed by Riks (1979) and developed in various efficient schemes by Crisfield (1981), Ramm (1981) and Bathe and Dvorkin (1983). The arc-length method is capable of capturing both the snap-through and snap-back behaviour, where load control and displacement control fail, respectively. A basic assumption in this approach is that the load vector varies proportionally during the response calculation, i.e., for each iteration, the increment of the external force vector can be written as $\Delta\lambda_i\hat{\mathbf{f}}$, with $\Delta\lambda_i$ the load factor, and $\hat{\mathbf{f}}$ the unit load. The total iterative increment can be derived from equation (3.72) and split in two parts:

$$\begin{aligned}\delta\mathbf{u}_i &= \mathbf{K}_i^{-1}({}^t\mathbf{f}_{ext} + \Delta\lambda_i\hat{\mathbf{f}} - \mathbf{f}_{int,i}) \\ &= \mathbf{K}_i^{-1}({}^t\mathbf{f}_{ext} - \mathbf{f}_{int,i}) + \Delta\lambda_i\mathbf{K}_i^{-1}\hat{\mathbf{f}} \\ &= \delta\mathbf{u}_i^{\mathrm{I}} + \Delta\lambda_i\delta\mathbf{u}_i^{\mathrm{II}}\end{aligned} \tag{3.80}$$

The load factor $\Delta\lambda_i$ now acts as a constraint on the incremental displacement and can be obtained by substituting (3.80)₃ in the constraint equation. The constraint equation is defined by assigning the norm of the incremental displacements to a prescribed value, written as

$$\Delta\mathbf{u}_i^{\mathrm{T}}\Delta\mathbf{u}_i = \Delta l^2 \tag{3.81}$$

where Δl is the required arc length. For a linearized constraint approach, see Ramm (1981), Δl is measured by $\Delta\mathbf{u}_{i-1}$, then the constraint equation for $\Delta\mathbf{u}_i = \Delta\mathbf{u}_{i-1} + \delta\mathbf{u}_i$ becomes

$$(\Delta\mathbf{u}_{i-1})^T\delta\mathbf{u}_i = 0 \tag{3.82}$$

with the quadratic term in $\delta\mathbf{u}_i$ ignored. Substitution of (3.80)₃ in (3.82) yields the expression of load factor as

$$\Delta\lambda_i = -\frac{(\Delta\mathbf{u}_{i-1})^T\delta\mathbf{u}_i^{\mathrm{I}}}{(\Delta\mathbf{u}_{i-1})^T\delta\mathbf{u}_i^{\mathrm{II}}} \tag{3.83}$$

This constraint indicates that the solution of iterative increment is projected on the plane normal to the previous solution; therefore it is referred to as the Updated Normal Plane method.

As mentioned before, attention is given to the global nonlinear behaviour of the concrete block wall subject to out-of-plane loading in this study. In this regard, the previous discussion of the constraint equations in arc-length control is adequate since all displacements (or degrees-of-freedom) were considered. If the local collapse mechanism is of particular interest, as reported by De Borst (1986), it will be necessary to confine the number of degrees-of-freedom in the constraint equations. The vectors $\delta\mathbf{u}$ and $\Delta\mathbf{u}$ will be replaced by $\delta\mathbf{v}$ and $\Delta\mathbf{v}$, respectively, with $\mathbf{v}$ defined by

$$\mathbf{v} = \begin{Bmatrix} \alpha_1 u_1 \\ \alpha_2 u_2 \\ \cdots \\ \alpha_n u_n \end{Bmatrix} \tag{3.84}$$

In the extreme case where only one item in $\mathbf{v}$ is non-zero, the arc length is defined as the displacement of the corresponding degree of freedom. This type of control is called as Indirect Displacement Control because the loading is defined as an external force. Another type of the Indirect Displacement Control is Crack Mouth Opening Displacement (CMOD) control, where the vector used in the constraint equation is formed with new 'degrees of freedom' that provided by two displacements on opposite nodes on a crack plane. This method has been attempted in the study of masonry shear wall by Lourenço (1996, 2002) but will not be resorted to in current study.

It has been shown that arc-length method can adapt the loading during iterations inside a load increment, with the initial step size is still fixed. The choice of step size depends on the physical behaviour of the structure and the convergence characteristics of the selected iteration process. Small step sizes are particularly required for nonlinear structure response, where the relation between stresses and strains is path dependent. However, the optimum step size cannot be determined at the beginning of the analysis and must be adjusted with the progress of incremental procedure. To allow for result dependent increment sizes, the self-adaptive loading is usually considered. An energy based adaptive loading is used in the current study. This method calculates a load increment such that the energy increment in the first prediction of current step, ${}^{t+\Delta t}\tilde{\mathbf{W}}$, equals the final energy increment of the previous step, ${}^{t}\mathbf{W}$(Figure 3.18). Herein, energy increment is calculated as the vector product of the load increment and the displacement increment. The new loading factor is derived from

$$
{}^{t+\Delta t}\Delta\lambda_0 = \sqrt{\frac{\left|{}^{t}\Delta\lambda_n ({}^{t}\Delta\mathbf{u}_n)^{\mathrm{T}}\hat{\mathbf{f}}\right|}{\left|\delta\mathbf{u}_0^{\mathrm{T}}\hat{\mathbf{f}}\right|}} \tag{3.85}
$$

where index n indicates the final iteration of the previous increment. The initial value of $\Delta\lambda$ should be determined from equation (3.83), where the vectors $\delta\mathbf{u}_0^{\mathrm{I}}$ and $\delta\mathbf{u}_0^{\mathrm{II}}$ must be calculated according to equation (3.80).

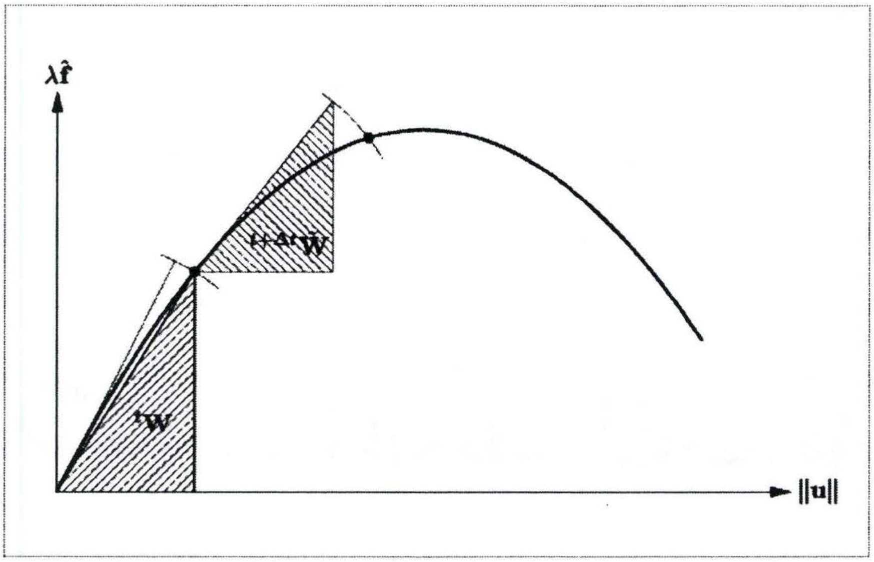


Figure 3.18 Energy increment, TNO DIANA BV (2003)

The remaining question is whether at a certain load level, the load must increase further or decrease. The question usually arises in case of snap-through where the structure is in an unstable state. The change of loading and unloading indicates the change of the sign of the load factor. An effective method that can only be used in combination with the arc-length methods, proposed by Crisfield (1981), requiring that the angle between the current prediction and the previous increment be acute. Therefore, the new loading factor is given by

$$
{}^{t+\Delta t}\Delta\lambda = \begin{cases} +\left|{}^{t+\Delta t}\Delta\lambda\right| & \text{if} \quad {}^{t}\Delta\mathbf{u}_n^T \delta\mathbf{u}_0^{II} \geq 0 \\ -\left|{}^{t+\Delta t}\Delta\lambda\right| & \text{if} \quad {}^{t}\Delta\mathbf{u}_n^T \delta\mathbf{u}_0^{II} < 0 \end{cases} \tag{3.86}
$$

In an overview, the solution procedure for a nonlinear problem is a complex process in which the iterative schemes, increment techniques and convergence criteria should be employed effectively. When self-adaptive loading is taken into account, the initial choice of the step size for every increment is a crucial factor that might affect the convergence of the iteration. Usually the analysis cannot be completed in a single run, it is important to save the last converged step, adjust the initial load step, change the iterative process and control parameters and restart the analysis. As it is not part of the scope of current study, the comparison of different solution techniques will not be presented.

3.5 Summary

A nonlinear finite element model for concrete block masonry based on discrete modeling approach has been presented. In this model, block units and mortar joints are discretized separately with three-dimensional continuum elements and interface elements, respectively. This modeling strategy allows the two major failure mechanisms of the block masonry, tensile cracking and sliding along the mortar joints, to be incorporated in the model.

The constitutive model that describes the relation between stress and strain tensor at a point of the material body is based on the formulation developed by Lourenço (1996). This Composite Interface Model, including softening for tension, shear and compression was originally implemented according to the theory of multi-surface plasticity. In combination with the structural behaviour of the block masonry in current study, the general formulation is discussed for the composite yield surface containing two major failure modes, tension and shear.

The solution procedure for the nonlinear problem was reviewed in the relevant aspects. The Regular (Full) Newton-Raphson method was used to solve the system of nonlinear equations resulting from the finite element discretization. To ensure that global convergence of the iteration process is met, the Line Search technique was employed in combination with the Newton-Raphson scheme. The convergence criterion was set up based on the variation of internal stain energy. Arc-length control along with the Update Normal Plane method uses load factor as the constraint on the incremental displacement such that the 'snap-through' and 'snap-back' behaviour can be easily captured. With Indirect Displacement Control, in which only the displacements of partial degrees of freedom are considered, the local collapse mechanism can be evaluated. Energy-based adaptive loading will be considered to enhance the efficiency of arc-length method. Finally, a method of determining loading and unloading based on the change of the sign of the load factor was introduced.

CHAPTER 4

MODEL EVALUATION VIA NONLINEAR ANALYSIS

4.1 Introduction

The discrete modeling approach along with the well-defined constitutive model described in the last chapter has been shown to closely predict the structural behaviour of brick masonry subject to in-plane loading (see Lourenço 1996). The adequacy of this method in modeling the out-of-plane bending of hollow concrete block masonry is the objective of this study. The nonlinear analysis is carried out using the multi-purpose finite element program DIANA™ (2003) of TNO DIANA BV in the Netherlands. The program is selected because it possesses all the capabilities identified in the previous chapter to carry this analysis.

The material properties used for the analysis are first presented, followed by a description of the finite element model of the wall and a mesh sensitivity analysis. The adequacy of the compiled model is then evaluated using solid walls with different boundary conditions and walls with openings of different dimensions, locations and number. A total of 10 walls are used to compare the computed responses to the experimentally measured ones.

4.2 Material Properties

The experimental results used for comparisons are taken from Essawy (1986) and Chen (2002), neither of which provides complete parameters necessary to characterize the material models in this study, especially the inelastic properties of the mortar joints. The elastic properties for the block units and mortar joints are determined based on the values adopted by Essawy (1986) (Table 2.2, heterogeneous models). For block units modeled with continuum elements, isotropic property is assumed and elastic modulii for the three orthotropic directions are taken equally as 19,660 MPa. The joint interface is described by linear normal stiffness, k_n, and tangential stiffness, k_s, which can be calculated from equation (3.30). These properties are listed in Table 4.1.

Table 4.1 Elastic properties for the blocks and joints

Block unit		Mortar joint	
E (N/mm^2)	ν	k_n (N/mm^3)	k_s (N/mm^3)
19,660	0.3	127	52

The combined cracking-shearing-crushing model requires 4 strength parameters (f_t, c, $\tan\phi$ and $\tan\phi_r$) and 3 inelastic parameters (G_f^I, G_f^{II} and $\tan\psi$). Unfortunately, most of the abovementioned parameters are not available for concrete block masonry in the literatures except the joint bond strength f_t.

The latter property was obtained from Essawy (1986), where f_t was given by the tensile test of undamaged parts (5-block high prisms) of the tested walls. The other inelastic properties for joint interface were adopted from the test of brick masonry (Van der Pruijm 1992, 1993). Table 4.2 provides a summary of the joints' material properties that were used for the analyses of the walls.

Table 4.2 Inelastic properties for the joints

Tension	f_t	(N/mm^2)		0.37
	G_f^I	(Nmm/mm^2)		0.012
Shear	c	(N/mm^2)		0.518
	$\tan\phi$			0.75
	$\tan\psi$			0.6
	$\tan\phi_r$			0.75
	$G_f^{II} = a\sigma + b$	(Nmm/mm^2)	a	0
			b	0.05

Van der Pluijm (1993) reported a range of 0.15 MPa to 1.85 MPa for the cohesion depending on the unit-type and the mortar. It was noted that strong mortar always yields a higher value for cohesion. The test results also showed that the ratio between the cohesion and the tensile bond strength c/f_t was never less than 1.3, and that high ratios were found when the tensile bond strength was low (Van der Pluijm 1993). A ratio of 1.4 was assumed in this study as 0.37 MPa is not considered to be a low tensile strength value. The angle of internal friction, measured by $\tan\phi$, is found to be almost independent of the used material combinations and it is assumed to be a constant, i.e., $\tan\phi = \tan\phi_0 = \tan\phi_r = 0.75$. As discussed in Section 3.3.1, the dilatancy angle, measured by $\tan\psi$, decreases when the normal confining stress becomes greater. Since no confining stress was presented, the value of $\tan\psi$ was taken as 0.6 according to the relation described in Figure 3.9. In Van der Pluijm's shear tests, the value for the mode II fracture energy G_f^{II} was observed to depend on the unit type but not the mortar. In addition, it also depends on the level of the confining stress. With zero normal confining stress, the value varies between 0.009 and 0.058 Nmm/mm^2. A constant value of 0.05 Nmm/mm^2 is adopted for this analysis.

4.3 Description of the Wall Models

All the walls used for evaluation are full-sized concrete block walls tested by Essawy (1986) and Chen (2002) at the Applied Dynamic Laboratory of McMaster University. Essawy's two wall series were both solid walls without openings, denoted as WIII and WF. Both series contain three specimens; whereas Chen's walls, denoted as BC, consist of one solid wall without openings and seven walls with opening(s) at different locations. Except for WF, which is 5.2 m long by 2.8 m high and supported on three sides with a free top, all other walls are 6 m long by 2.8 m high and have the four-sides supported. The left and right supports are located at the leeward face, 100 mm from each end of the wall and symmetrical about the mid-length. The bottom supports are located at the centerline of the wall bottom; and the top supports are at the top-back edge of the walls. All the walls were laid up in running bond. Brief descriptions of these walls are given in Table 4.3.

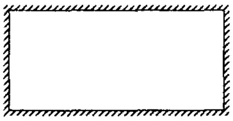
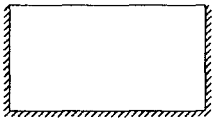
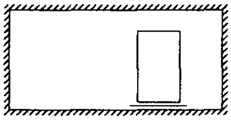
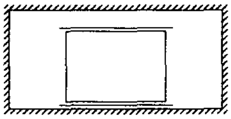
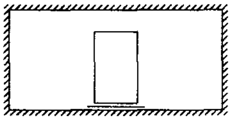
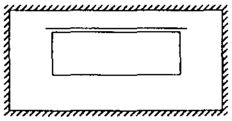
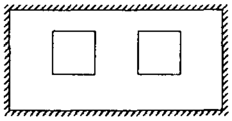
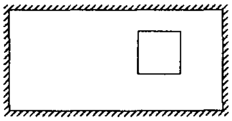
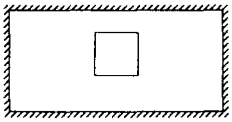
Grouted bond beams with #30 steel reinforcing bars were incorporated in walls BC2, BC3, BC4 and BC5, as noted in Table 4.3, for safely transporting the walls for testing and to act as a support for panels with large openings. Each bond beam was 400-mm longer than the opening width to allow 200-mm development length for the reinforcements on each side beyond opening edges.

4.3.1 Modeling of the Wall Geometry

Walls WIII, WF, BC1, BC3, BC4, BC5, BC6 and BC8 have a single vertical axis of symmetry about the mid-length of the wall. Therefore, a half wall model may be used to reduce the input effort. Walls BC2 and BC7 have no symmetry due to asymmetric opening positions hence a full model has to be used.

The representative full block, half block units and single webs described in Chapter 2 were used as the basic units to create the wall model with running bond pattern. For those walls with bond beams, 3-D brick elements were added in the hollow part of the units representing the grouted cells. The reinforcing bars in the bond beams were not modeled. The effects of the reinforcement on the wall response were investigated using a linear analysis. The results have shown that this presence has a negligible effect.

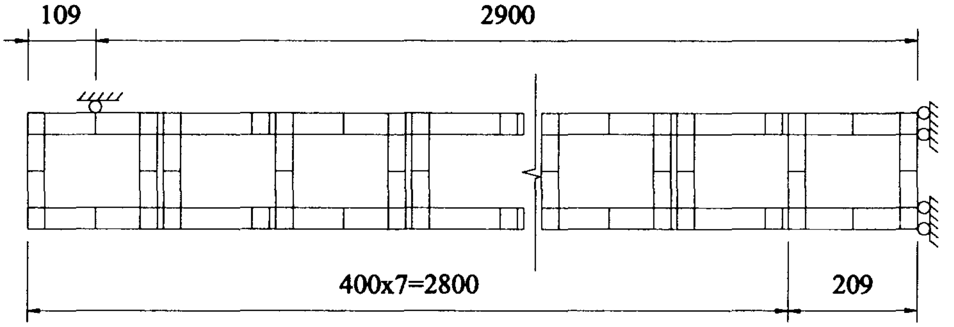
Table 4.3 Wall descriptions

Wall	Wall size (m)	Clear span (m)	Opening size (m)	Opening location	Wall outline
WIII BC1	6.0 × 2.8	5.8 × 2.8	N/A	N/A	
WF	5.2 × 2.8	5.0 × 2.8	N/A	N/A	
BC2	6.0 × 2.8	5.8 × 2.8	1.2 × 2.0	1.2m off center	
BC3	6.0 × 2.8	5.8 × 2.8	2.8 × 2.0	Center	
BC4	6.0 × 2.8	5.8 × 2.8	1.2 × 2.0	Center	
BC5	6.0 × 2.8	5.8 × 2.8	3.6 × 1.2	Center	
BC6	6.0 × 2.8	5.8 × 2.8	Two 1.2 × 2.0	1.2m off center	
BC7	6.0 × 2.8	5.8 × 2.8	1.2 × 2.0	1.2m off center	
BC8	6.0 × 2.8	5.8 × 2.8	1.2 × 2.0	Center	

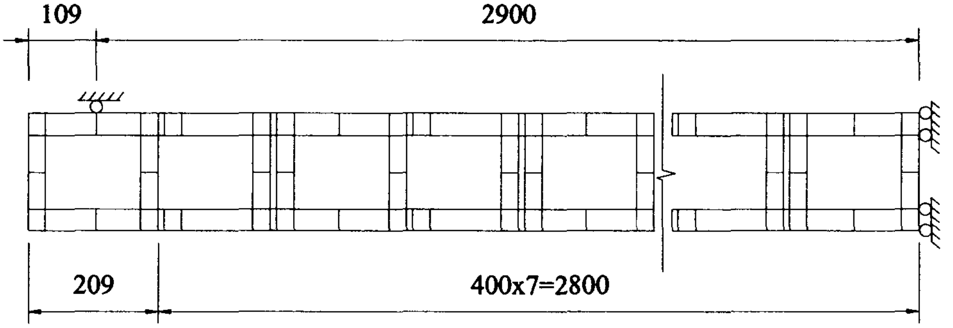
Note: The horizontal lines above or below the opening indicate the bond beams.

To facilitate the construction of the finite element model, odd courses and even courses were distinguished. The full-block and half-block units discussed below should be referred to an "approximate" full-block and half-block units. The details of the half and full models of the solid wall are shown in Figure 4.1 to Figure 4.3. In the half model, seven full-block units and one half-block unit were connected horizontally to produce 3009-mm long model in one course. As the left support line is located at 100 mm from the left edge of the wall in the test, and to ensure the half span is 2900 mm, the nodes at the left support position were adjusted to produce 109-mm distance between the left edge and the left support line in the model. The 9-mm difference, between the actual and modeled overhang length beyond the support, was considered to be negligible.

For the full model, two half units were placed at the two ends of the fourteen consecutive full-block units in an odd course, and a single web is appended at the end of the fifteen full-block units to complete the model in an even course. This configuration results in a wall length of 6028 mm. Again, the nodes at the left and right support lines should be adjusted such that the overhang length beyond the support at each side is 109 mm and 119mm, respectively. By this approach the actual horizontal span of 5800 mm is maintained.

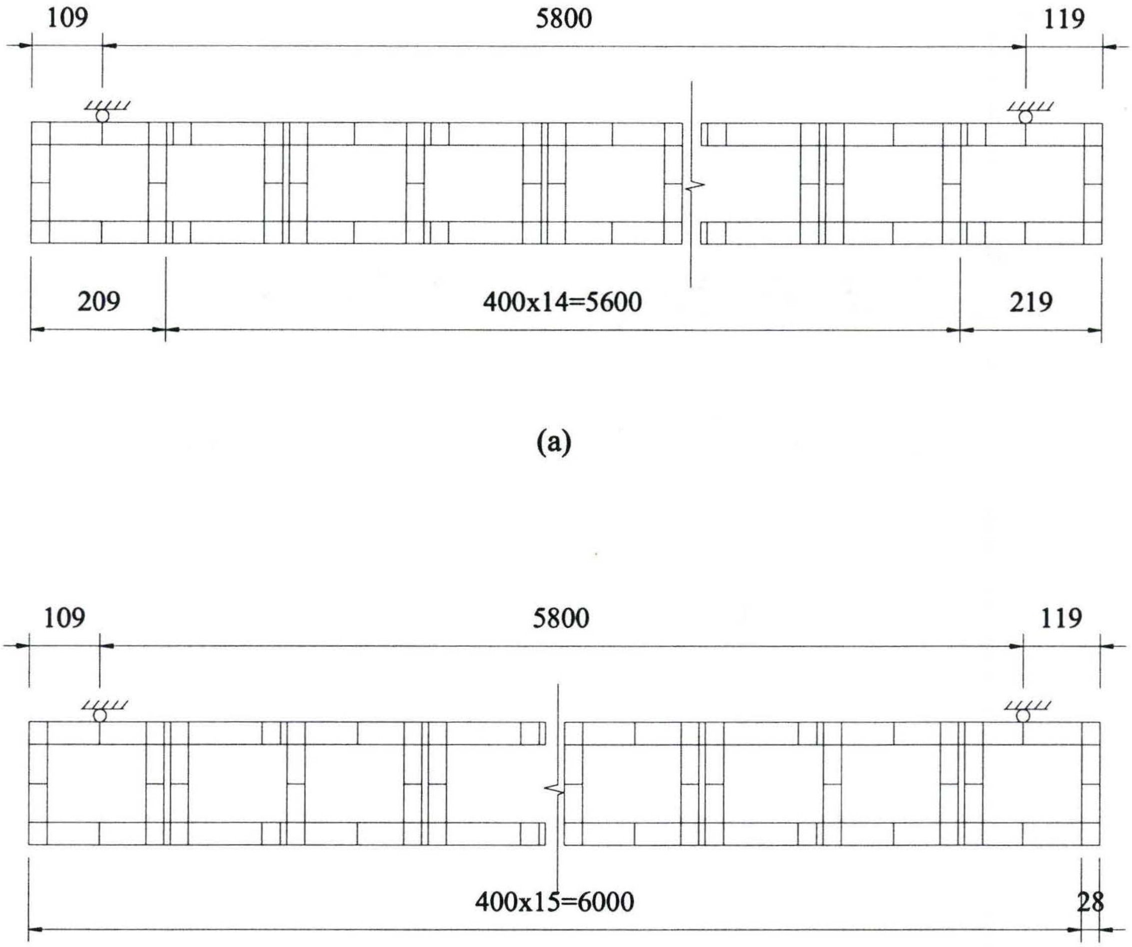


(a)



(b)

Figure 4.1 Details of half model configurations (horizontal cross section): (a) odd course, (b) even course.



(a)

(b)

Figure 4.2 Details of full model configurations (horizontal cross section): (a) odd course, (b) even course.

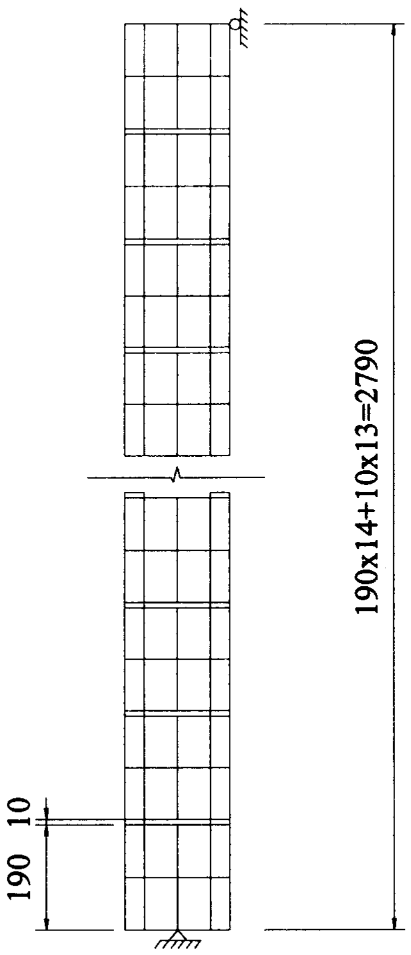


Figure 4.3 Details of wall model configuration (vertical cross section)

4.3.2 Boundary and Load Conditions

The 8-node brick element has three degrees of freedom per node, displacement along X (wall length), Y (wall thickness) and Z (wall height) directions. Boundary conditions are implemented by defining the appropriate constraints along these directions. To simulate the actual support conditions, the vertical and top supports were modeled as rollers by constraining the out-of-plane (Y direction) displacement of the nodes on the support line; the bottom support was modeled as pins along the center of the web bottoms in the bottom course to allow rotation but restrict both the in-plane and out-of-plane movements. When the half model was considered, the boundary at the axis of symmetry was modeled as rollers with the nodes constrained in the X direction. In addition, the left-top and left-bottom corners of the model were constrained in the out-of-plane (Y) direction to prevent the corner displacements.

The out-of-plane loading was modeled as uniformly distributed load applied normal to the front surface of the wall model pointing inward. For the models with opening(s), the pressure load was first transferred to uniformly distributed line loads along the edges of the opening(s), and then converted to the nodal forces according to the tributary distance for each node on the opening edges.

4.4 Mesh Size Sensitivity Analysis

Before proceeding with the analysis of these full-sized walls, a limited mesh size sensitivity analysis was carried out to assess the suitability of the proposed finite element mesh. The elements in a discretized model are connected through nodal points at the inter-element boundaries. The unknown displacements in each element are actually approximated by continuous functions, the so-called interpolation functions, expressed in terms of nodal variables. It can be expected that finer discretization of the model with increased nodal points should achieve better results. However, apparently, this leads to considerable computing time. In this regard, the determination of a proper mesh size is needed to provide both acceptable accuracy and efficiency.

Lourenço (1996) has reported that the composite interface model is insensitive to the mesh size based on the analysis of a 990 ×1000 mm brick shear wall using two-dimensional discrete modeling. In his model, the unit was modeled with 4 ×2 quadratic plane stress elements and the mortar joint was modeled with quadratic line interface elements. In the model with refined mesh, the unit was modeled with 8 ×4 elements, which leaded to 4 times as many continuum elements and 2 times as many interface elements. Comparison of the two results showed that there are no significant differences between the two meshes (Figure 4.4).

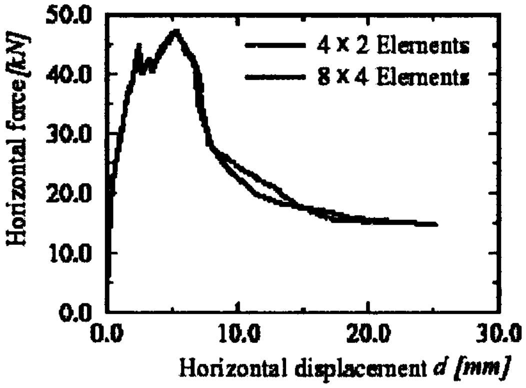


Figure 4.4 Result of mesh sensitivity study for 2D composite interface model, Lourenço (1996)

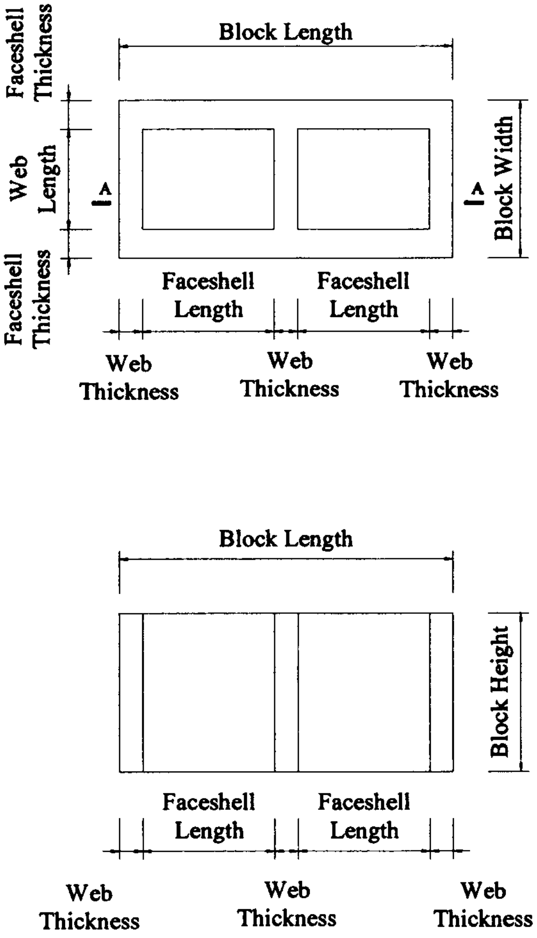
The relevance of Lourenço's observation needs to be investigated for three-dimensional composite interface model. As described before, approximated block units were subdivided into different element groups, which were further into 3-D 8-node brick elements. The mesh size should be consistent between two adjacent groups to ensure the correct connectivity. Discretization of the face shells needs further consideration since the joint interface located between these units, which depend on the tensile stress, control crack initiation and propagation. Also, the webs, which take the role of transferring shear deformation between the face shells, their mesh size needs to be studied.

In addition to the above consideration, the ratio of the smallest dimension to the largest dimension in each element in units was set not exceed 1/3. The mesh of the joint interface follows that of the units in the joint length (X-direction for bed joints or Z-direction for head joints) and width (Y) direction. No further discretization of joint thickness is allowed according to the definition of interface element. Two cases of mesh sizes were investigated in this study where the finer mesh has resulted in 8 times as many continuum elements and 4 times as many interface elements (see Table 4.4). The corresponding definitions of the unit geometry are shown in Figure 4.5.

Wall WIII was selected for the nonlinear analysis. The comparison between the load-displacement diagrams for the two meshes is shown in Figure 4.6. The results agree fairly well up to the initiation of the first mechanism, which corresponds to the first peak points on the curves. A slight difference, within 5%,

Table 4.4 Summary of the mesh sizes

Case	Number of Divisions in					Number of Nodes	Number of Elements
	X-Dimension		Y-Dimension		Z-Dimension		
	Web Thickness	Face Shell Length	Face Shell Thickness	Web Length	Block Height		
1	1	2	1	2	2	14532	7356
2	2	4	2	4	4	68250	45356



DETAIL A-A

Figure 4.5 Definitions of the block unit dimensions

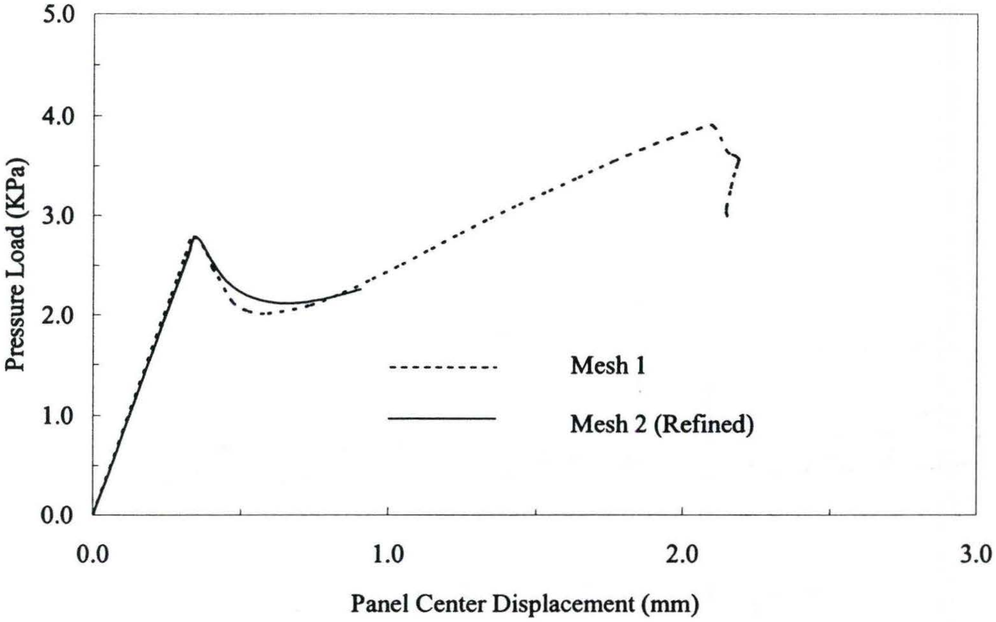


Figure 4.6 Comparison between two mesh discretizations

is found thereafter. During the nonlinear softening process, less energy dissipation is observed in the model with refined mesh. Even though the analysis was not completed due to the considerable time required, the two meshes predict the same results. Comparing the time required to carry these two analyses, using a 2.4-gigabyte Pentium® 4 personal computer with 768-Megabyte memory, the execution time for refined mesh was more than 300% in comparison to the proposed mesh. As a result, the regular mesh (case 1) is adopted for all the analyses through out the current study.

4.5 Analytical Results of the Full-sized Walls

In this section, a detailed discussion of the analytical results for the ten full-sized walls is presented. The behaviour of each wall is characterized via the predicted load-deflection relationship and the failure pattern. Two kinds of failures associated with the experimental observations, first cracking and ultimate failure, are of major interest in the study. The first cracking indicates a substantial change in the wall behaviour as it represents a serviceability limit for design. Accordingly, as pointed out by Baker et al. (1985) and Lawrence (1983), the prediction of this load should be regarded as the first-line analysis consideration. Ultimate failure signifies the safety warning of the structure and should be given more concern. For the sake of computing time, the complete analysis including the full post-failure was not pursued here.

The definition of the two experimentally observed failures follows those given by Chen (2002). The first crack is defined to be a horizontal (except for wall WF) continuous crack forming a part of a propagating crack leading to a failure mechanism. And, the ultimate failure is defined as the development of new crack(s) in addition to continuous cracks previously formed, which act together to form a failure mechanism. It should be pointed out that, due to shrinkage cracks and scatter of material properties and construction technique that are inherited in the test specimen, one does observe some localized and discontinuous cracks that initiate first but are not referred to as the first cracking as they do not lead to a failure mechanism. However, these anomalies are not included in the finite element model, and as such one will not observe many of these localized and discontinuous cracks. In most cases, the initial crack forms a part of failure mechanism.

4.5.1 Walls Simply Supported on Four Sides (Wall WIII/BC1)

For concrete block walls that are simply supported on four sides, experimental results showed that there exist three distinct stages in the behaviour of walls; the occurrence of the first crack, the formation of the full crack pattern where the diagonal cracks occur, and the panel failure identified as the maximum load that the panel can withstand. The experimental crack patterns for the wall WIII-1 and BC1, as shown in Figures 4.7 and 4.8, are also consistent with the yield line theory (Essawy 1986). Both walls show similar behaviour because the two walls have the same dimensions and similar materials, and were subjected to similar boundary and loading conditions. In both walls, the horizontal crack initiated from the mid-height of the wall, followed by diagonal stepped cracks leading to collapse.

The computed and experimentally measured load-displacement diagrams for the displacement at the panel center of walls WIII are given in Figure 4.9. The general behaviour of the structure is well reflected from the predicted curve. One observes that the initial linear elastic stiffness of the model is much stiffer in comparison to the measured one. At the same time, one observes that the difference in the initial stiffness between the computed and measured ones fall

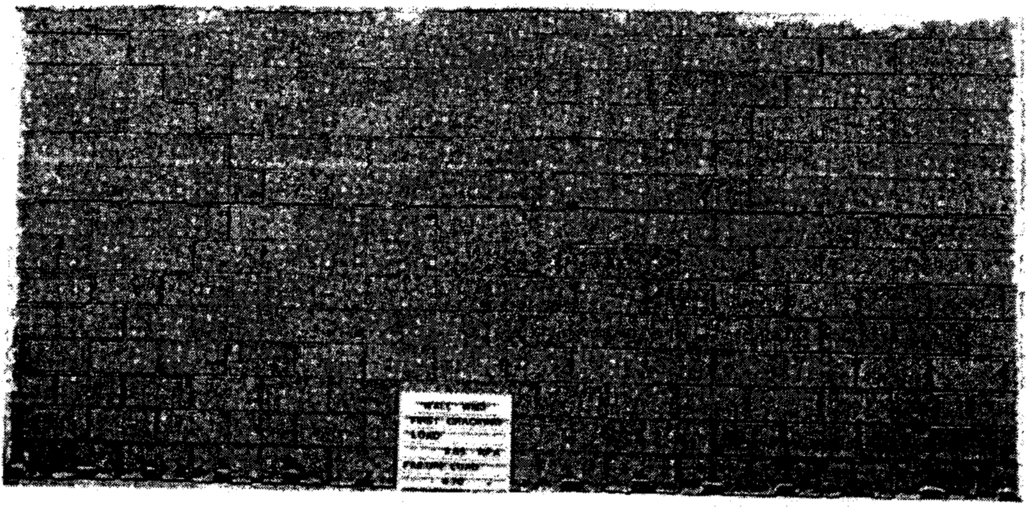


Figure 4.7 Experimental crack pattern for wall WIII-1, Essawy (1986)

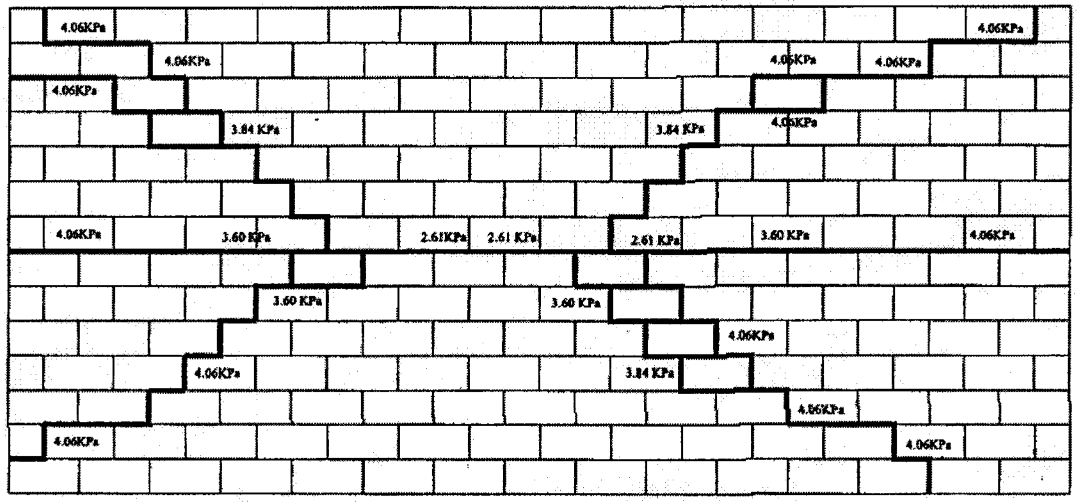


Figure 4.8 Experimental crack patterns for wall BC1, Chen (2002)

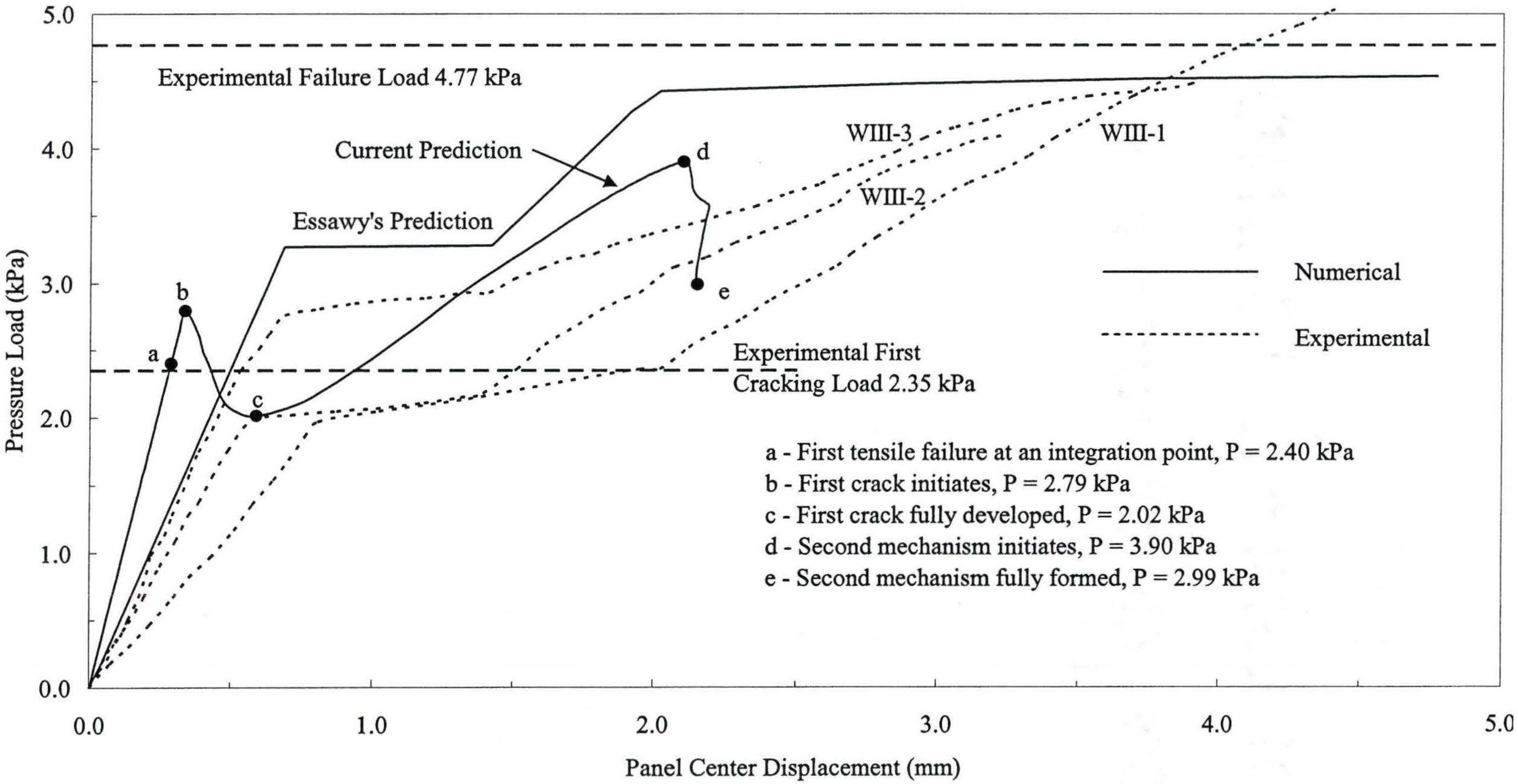


Figure 4.9 Wall WIII load-displacement diagrams

within the measured variability of the three tested walls. Many explanations can be given to why the initial elastic stiffness varies, starting with drying shrinkage of the mortar which affects the bond and contact area between the mortar and the units. Variability of the material properties is another cause as well as the actual construction of the walls.

Essawy observed that the first cracking occurs when the load reaches 2.35 kPa. The computed results show that the crack initiates at the bed-joint interface elements near mid-height on the leeward face of the wall at a comparable load of 2.40 kPa (Figure 4.9, point a). It should be pointed out that a load increment of 0.2 kPa was considered in the linear regime, which could result in a rough estimate of the first tensile crack. In most cases, one would expect the load at which the predicted first crack initiates to be lower than the measured one as it is difficult to visualize the onset of the cracks in an experimental set-up.

As the load continues to increase, the deflection keeps increasing with no noticeable changes to the wall stiffness until the first peak load of 2.79 kPa is reached (point b). During this period, integration points for different elements reach their tensile strength limit and continue until a continuous crack is formed across one full block (point b). Suddenly, the load drops. Therefore, this first peak can be regarded as the initiation of the first mechanism.

As the crack continues to propagate across more elements along the mid-height bed joint, the model exhibits softening behaviour with a decrease in load resistance and an increase in deflection. The strain energy previously absorbed during the linear range is now partially released. During the softening stage, an increase in the value of shear stress in the vicinity of the crack is observed leading to the development of the composite yield surface of tension and shear when the load reaches to 2.13 kPa. Consequently, the out-of-plane shear sliding mechanism is initiated at the bed joint. This mechanism was detected by comparing shear traction to the yield value. Afterwards, as the load decreases to 2.02 kPa, the crack becomes stable. This corresponds to the saddle point c in the curve. Up to this point, the predicted first crack reaches nearly the full length of the wall panel.

The predicted softening behaviour after reaching the first peak load was not captured in either of the three wall panels of WIII in the experiment. The difference in the two behaviours is due to the difference in how the load was applied. In Essawy's experimental program, the tests were load-controlled which is not capable in capturing the softening response. Instead, after the first crack initiates, one observes a significant drop in the stiffness of the wall. Subsequently, the panel continues to deform with almost zero increase in the applied load until a new equilibrium is reached. In comparison to the computed results, this stage is indicative of the development of the first crack.

The crack which is formed along the full length of the bed joint at mid-height structurally separates the wall panel into two sub-panels that are simply supported along three sides and free (contact boundary) along the cracked bed joint. As the new structural system is formed, the numerical load-displacement curve now shows another hardening regime but with a reduced stiffness for the panel. At the second peak load of 3.90 kPa, the second mechanism initiates, as the additional diagonal cracks extending from the horizontal crack toward the corners of the panel are observed (point d). This load is found to be within 20% range of the tested failure load. Subsequently, one observes a decrease in the load without any increase in the deflection. At this stage, the mechanism is fully formed (point e).

Compared with the predicted results of Essawy (1986), where an elasto-plastic finite element model with a layered material model was considered, the current prediction is more conservative. Both numerical results generally show less deformation than in comparison to the experimental ones. The reasons are similar to the ones identified earlier, namely the bond strength and the contact area between the units and the mortar joint are smaller due to shrinkage. Moreover, the presence of cracks softens the behaviour.

Similar comparison was carried out between the numerical and experimental load-displacement diagrams for wall BC1 (Figure 4.10). The measured first cracking load of 2.61 kPa lies between the predicted onset of the first cracking load of 2.40 kPa and the predicted first peak load of 2.79 kPa, the latter of which represents the initiation of the first mechanism. The predicted second failure of 3.90 kPa is within 4% of the experimental failure load. It is noted that after the first crack is fully developed (after point c), the reduced stiffness of the panel agrees well with that observed experimentally. The load-displacement curves still show that larger deflections are measured in comparison

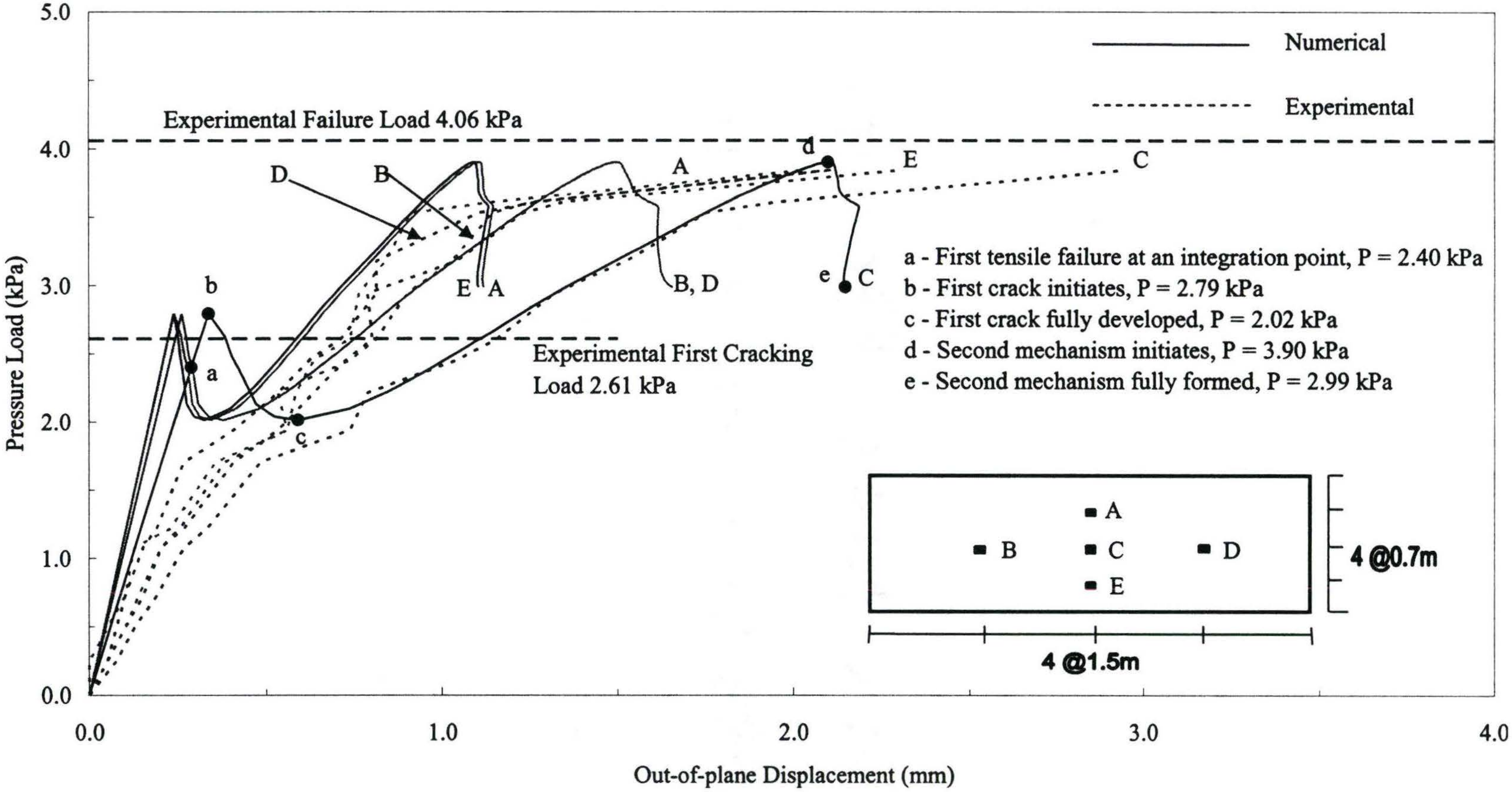


Figure 4.10 Wall BC1 load-displacement diagrams

to the predicted ones for the five locations, A, B, C, D and E. The crack pattern and evolution is the same as wall WIII. In general, one can state that the predicted result for BC1 agree very well with those measured ones.

The predicted crack development is shown in Figures 4.11 to 4.13. This confirms the foregoing discovery. From the profile of the model deformation, the relative movement of the units in out-of-plane direction can be observed, which verifies the phenomenon of shear sliding. Figure 4.14 presents the final crack pattern, which is similar to the experimental finding. Figure 4.15 illustrates the normal strain distribution in the joint interface, demonstrating that the mechanism starts at the peak load and is fully formed after the peak.

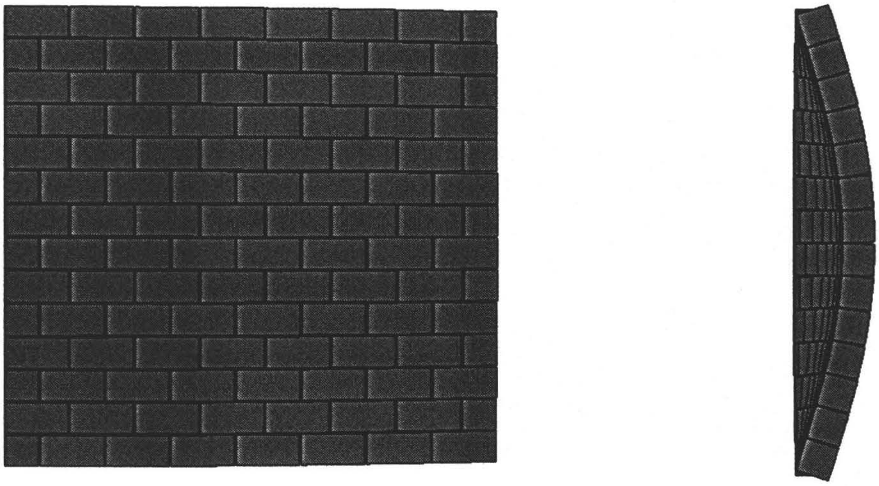


Figure 4.11 Crack development for wall WIII/BC1: the first mechanism initiates at the load of 2.79 kPa;

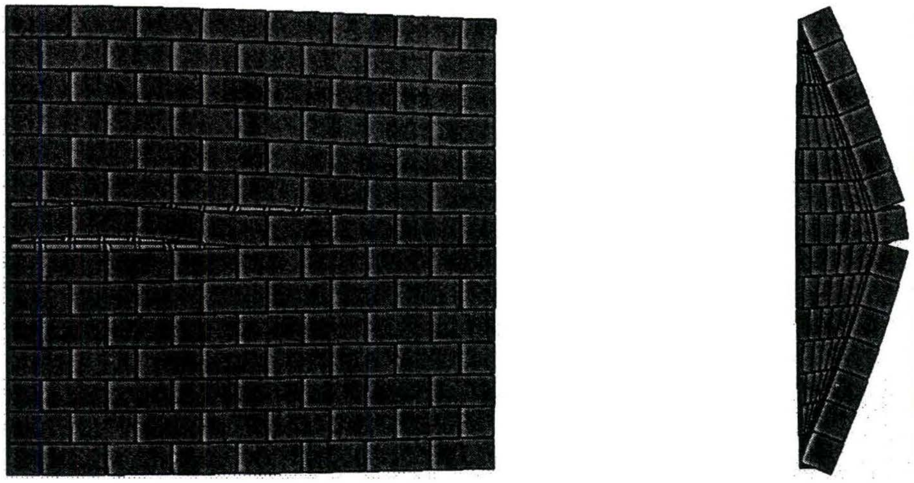


Figure 4.12 Crack development for wall WIII/BC1: the first mechanism is fully formed at the load of 2.02 kPa.

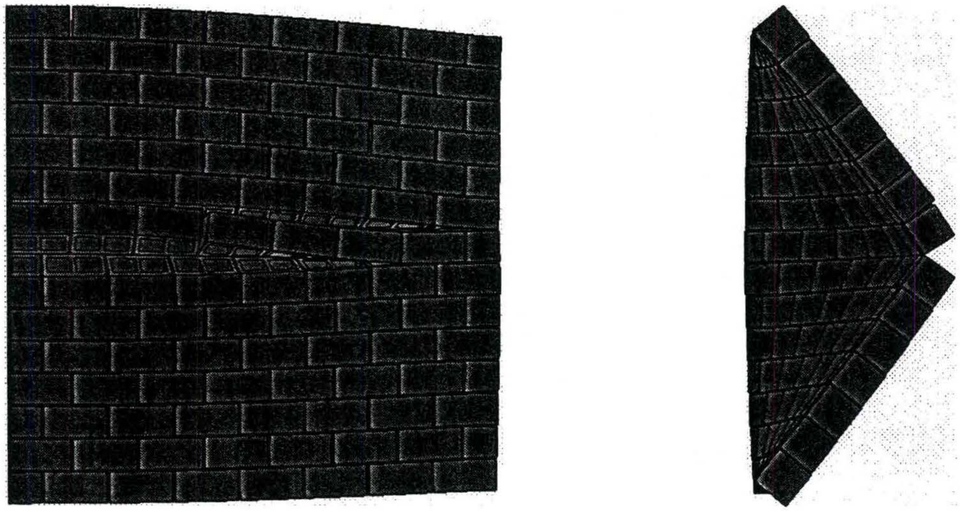


Figure 4.13 Crack development for wall WIII/BC1: the second mechanism initiates at the load of 3.90 kPa.

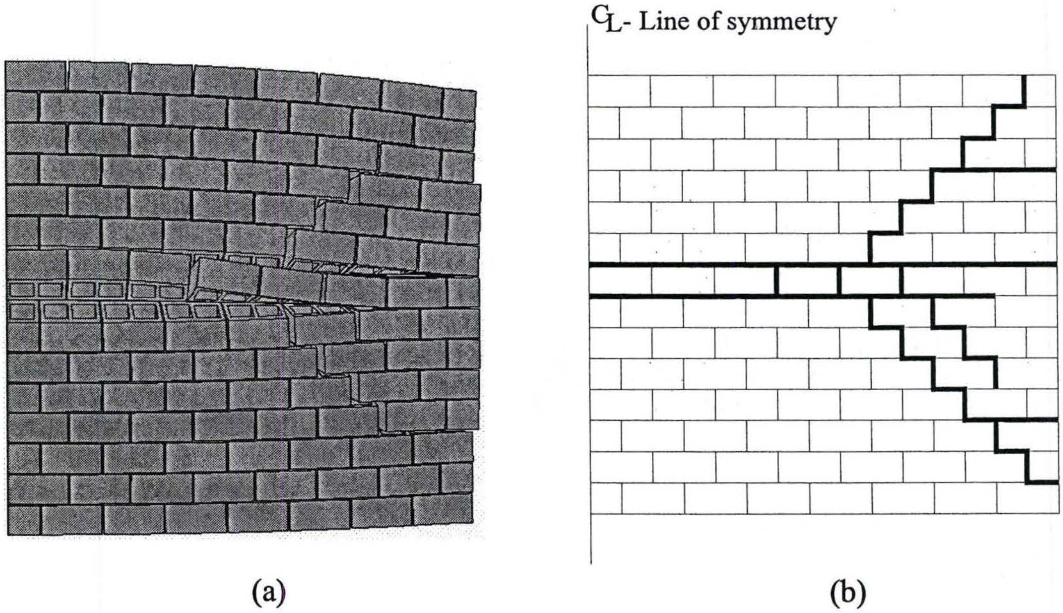


Figure 4.14 Results of the analysis for wall WIII/BC1 at the load of 2.99 kPa: (a) deformed mesh; (b) crack pattern.

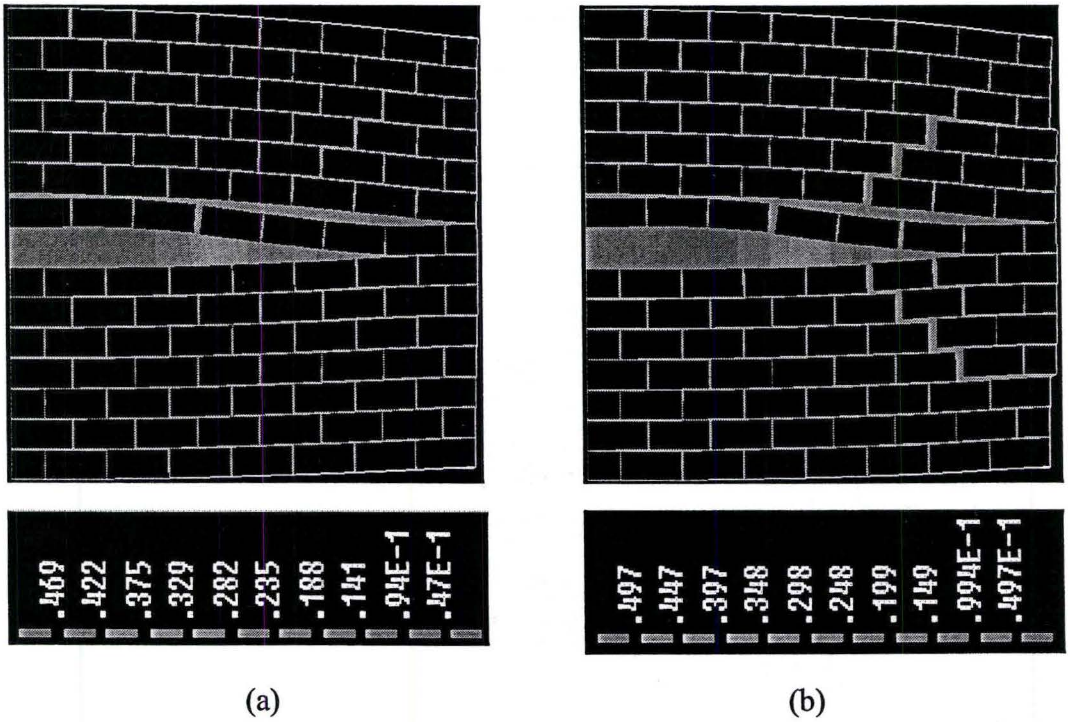


Figure 4.15 Traction deformation (normal strain) in the interface at a load of: (a) 3.90 kPa (peak); (b) 2.99 kPa (%77 post-peak).

4.5.2 Wall Simply Supported on Three Sides (Wall WF)

Series WF represent three walls that are simply supported along the two vertical sides and the bottom, with the top free. These walls have spans of 5.0 m between two vertical supports and heights of 2.8 m. With the ratio of height-to-length of 0.56, the crack pattern is expected to be vertical crack initiating from the center of top edge, splitting into two diagonal cracks to the bottom corners to form a mechanism. Figure 4.16 presents the experimental crack pattern for wall WF-3. It can be seen that a vertical crack passes through the units and head joints at the mid-length of the panel, joined by the stepped diagonal cracks and ended near the bottom corners. Since the potential cracks inside the units are not considered in the current study, one recognizes that the vertical splitting crack for wall WF could not be reproduced using the proposed model. Nonetheless, a full wall model was analyzed to assess the model prediction given the above-mentioned limitations.

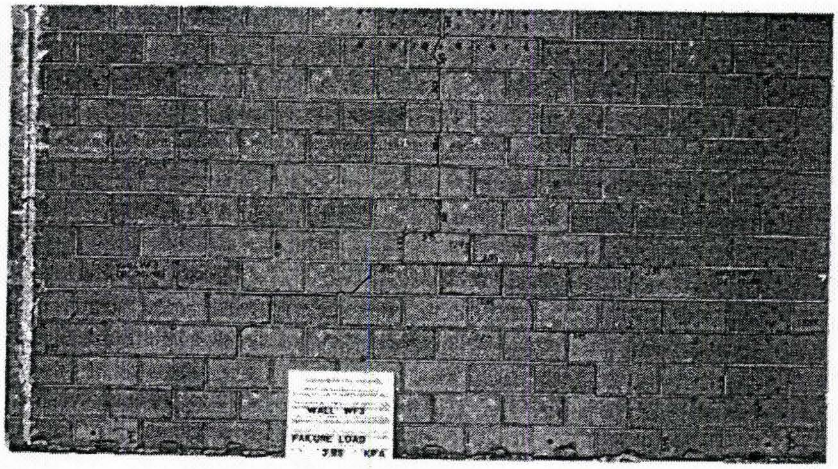
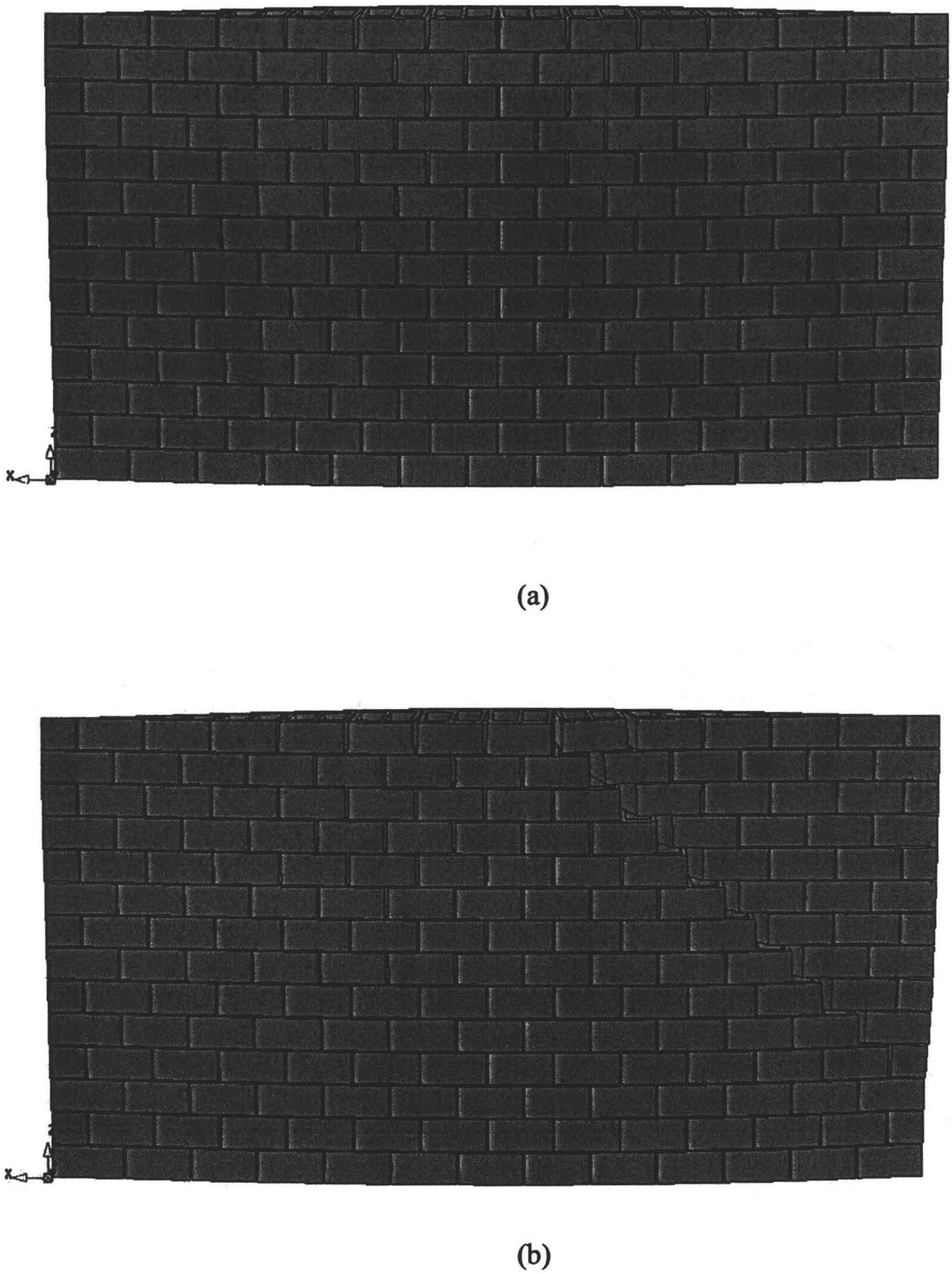


Figure 4.16 Crack pattern for wall WF-3, Essawy (1986)

As reproduced in Figure 4.17(a), one observes that the initial cracks form in the head joints at mid-height of the top free end of the wall. This is similar to the experimentally observed pattern. However, the crack is not permitted to propagate through the units, thus forcing it to follow a stepped-type pattern. Because of the odd and even pattern the walls are numerically assembled, this has created a bias towards the right side of the wall. As a result, the model has predicted that the crack propagates in a stepped-type pattern going diagonally until it reaches the supported end on the right side. The developed crack is schematically shown in Figure 4.18.

Although the model fails to predict the crack pattern, a comparison of the load-displacement diagrams shown in Figure 4.19, is carried out. The linear elastic behaviour in the early loading stage shows good comparison between the two results. As the load reaches 1.20 kPa, the onset of the first tensile failure occurs in the head joint at the top edge center of the panel (Figure 4.19, point a). The initiation of the first tensile failure occurs at the integration point of the joint interface element. As the lateral load increased, the panel shows a slight but continuous decrease of the stiffness. During this stage of loading, the interface is experiencing a combination of tensile and shear traction. The first shear sliding is found to occur at load of 1.90 kPa and the composite tension and shear yield surface is formed right before the first peak load, 2.38 kPa, is achieved (point b). Up to this point, the load-displacement curves for the analysis and experiment agree fairly well. The steep degradation after the peak indicates a rapid



(a)

(b)

Figure 4.17 Crack development for wall WF:

(a) The failure mechanism initiates at the load of 2.38 kPa;

(b) The failure mechanism is fully formed at the load of 2.00 kPa.

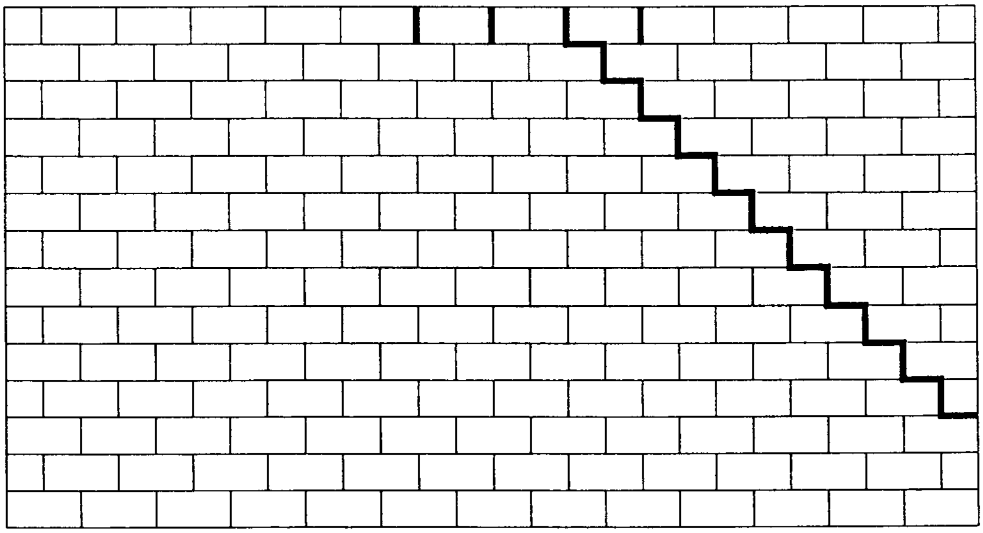


Figure 4.18 Crack pattern for wall WF

development of the crack. The crack propagates diagonally and extends to the bottom corner until a stable state is reached at the load of 2.00 kPa (point c). The failure mechanism is formed thereafter as the wall lost all its structural capacity.

The strain distribution in the interface shown in Figure 4.20 demonstrates that the failure mechanism initiates at the peak load but is fully formed afterwards. In comparison to the experimental results, this demonstrates the model predicted well the experimental results until it reached the stage where cracking through the units is required. As expected, the two results diverged. Therefore, for the description of the wall behaviour to be complete, splitting of the units needs to be incorporated for the future development of the model.

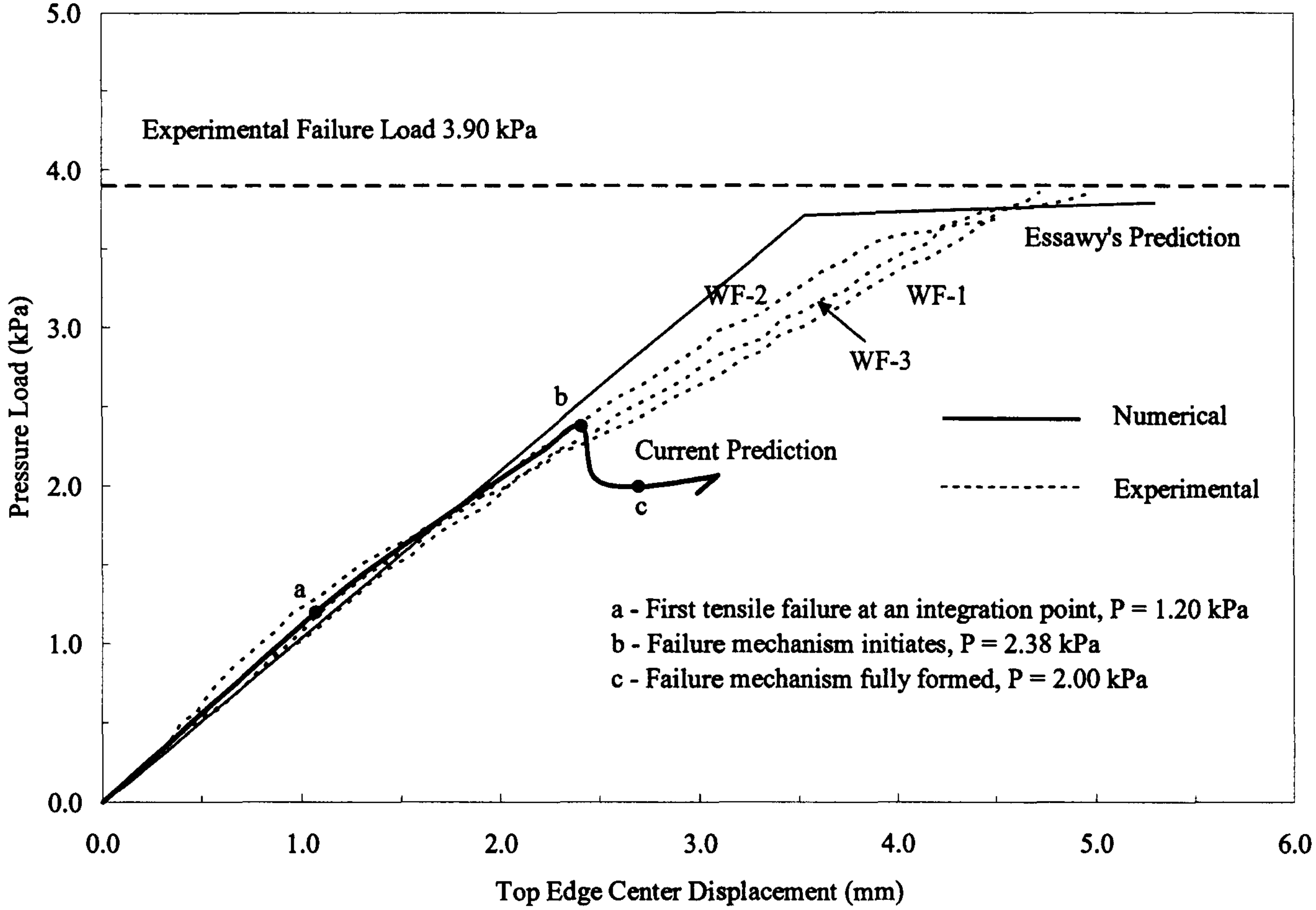
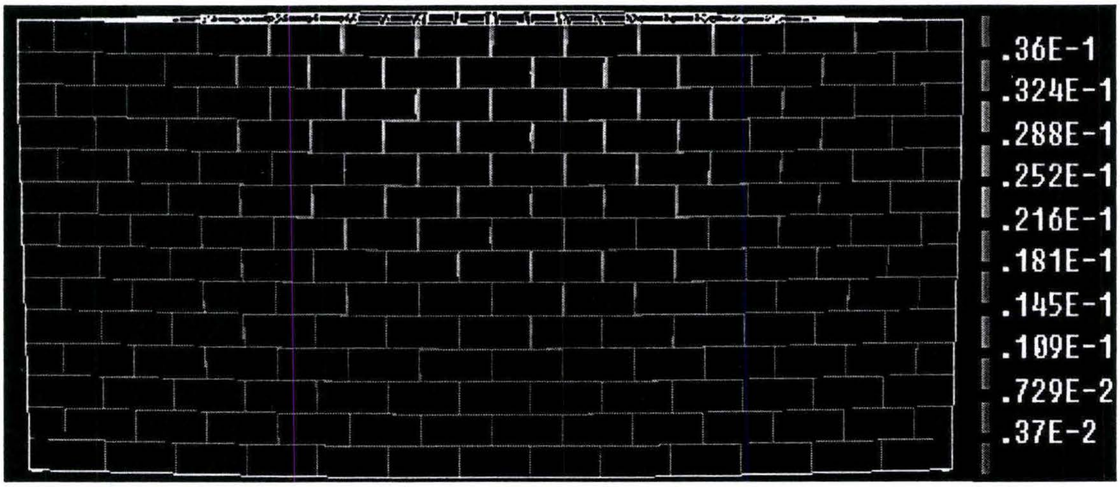
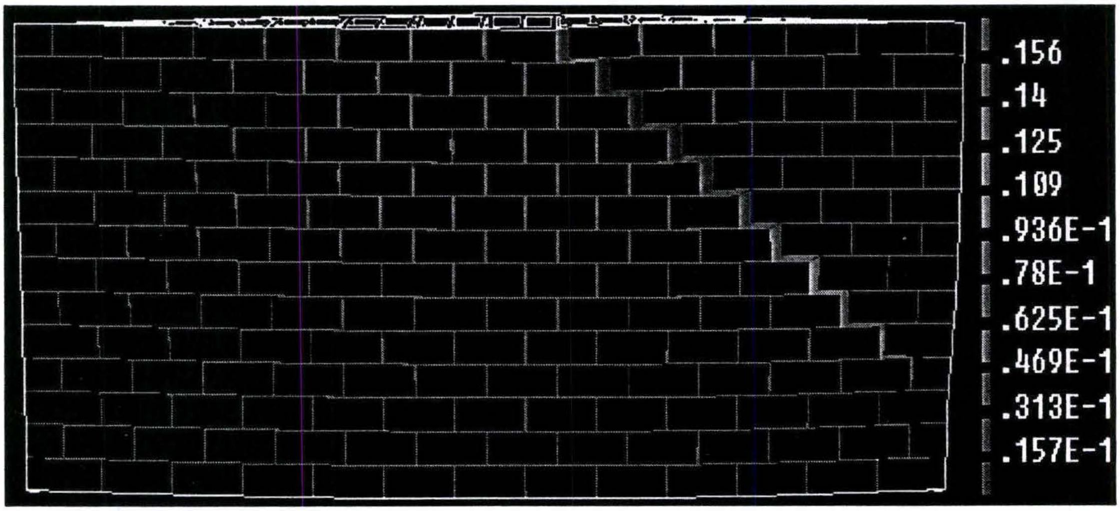


Figure 4.19 Wall WF load-displacement diagrams



(a)



(b)

Figure 4.20 Traction deformation (strain) in the interface at a load of (a) 2.38 kPa (peak); (b) 2.00 kPa (84% post-peak).

4.5.3 Wall BC2

Wall BC2 contained an opening located 1.2 m off the mid-length, representing a door access within a masonry wall. A reinforced bond beam was used at the bottom of the opening to facilitate safe transporting of the wall to the experiment set-up. The crack development and the corresponding cracking loads are shown in Figure 4.21. The first crack initiated at both top corners of the opening at the loading of 1.73 kPa. Another horizontal crack was then observed one course below the mid-height of the wall at the left of the opening, while the previous first crack continued to develop towards the left. Two horizontal cracks at the bottom of the opening were also formed. As the loading increased to 2.94 kPa, diagonal cracks originated from the previously formed horizontal cracks and they continued reaching the wall corners at the loading of 3.65 kPa, which was identified as the ultimate capacity. Note that at failure, a horizontal crack did form close to the bottom of the right pier of the panel.

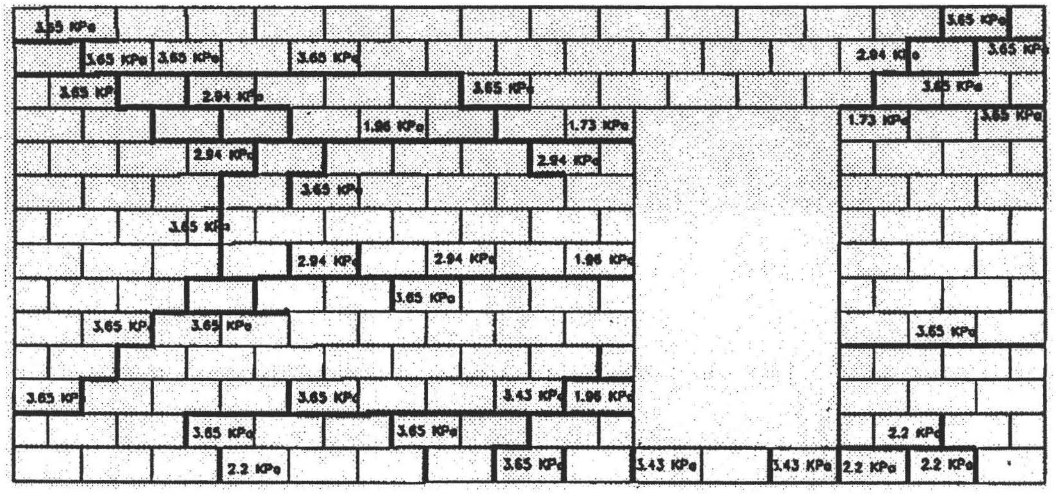


Figure 4.21 Crack pattern for wall BC2, Chen (2002)

The numerical and experimental load-displacement data are both plotted in Figure 4.22 for six locations on the wall. Similar to wall WIII, model of the wall BC2 first behaves linearly but stiffer in comparison to the tested specimen. When the load reaches 1.20 kPa, the onset of the first tensile failure occurs at the top left corner of the opening (Figure 4.22, point a). However, the actual first mechanism is considered to begin at the load of 1.94 kPa (point b) where a crack becomes continuous. The tested cracking load of 1.73 kPa lies in this range. With the increased loading, cracks at the top right corner, bottom corners and mid-height of the left of the opening develop. A linear elastic response can be observed up to the first peak load, meaning that the initiation of the cracks at some integration points has not significantly reduced the wall stiffness. This behaviour changes after a sudden drop from the peak load followed with a nonlinear softening phenomenon. The energy stored in the wall panel is partially released until the saddle point on the load-displacement curve is reached (point c), where the load becomes 1.28 kPa and at which time the horizontal crack at the mid-height of the left of the opening is almost fully developed as shown in Figure 4.23.

Thereafter, the wall exhibits a second hardening behaviour with a reduced stiffness similar to the measured response. The first shear sliding is observed when the load increases to 1.51 kPa, and the composite tension and shear yield surface is formed at load level of 1.77 kPa. Both failures occur at the interface at the top right corner of the opening. The convergence is found to be sensitive to the magnitude of load increments for this wall and achieving

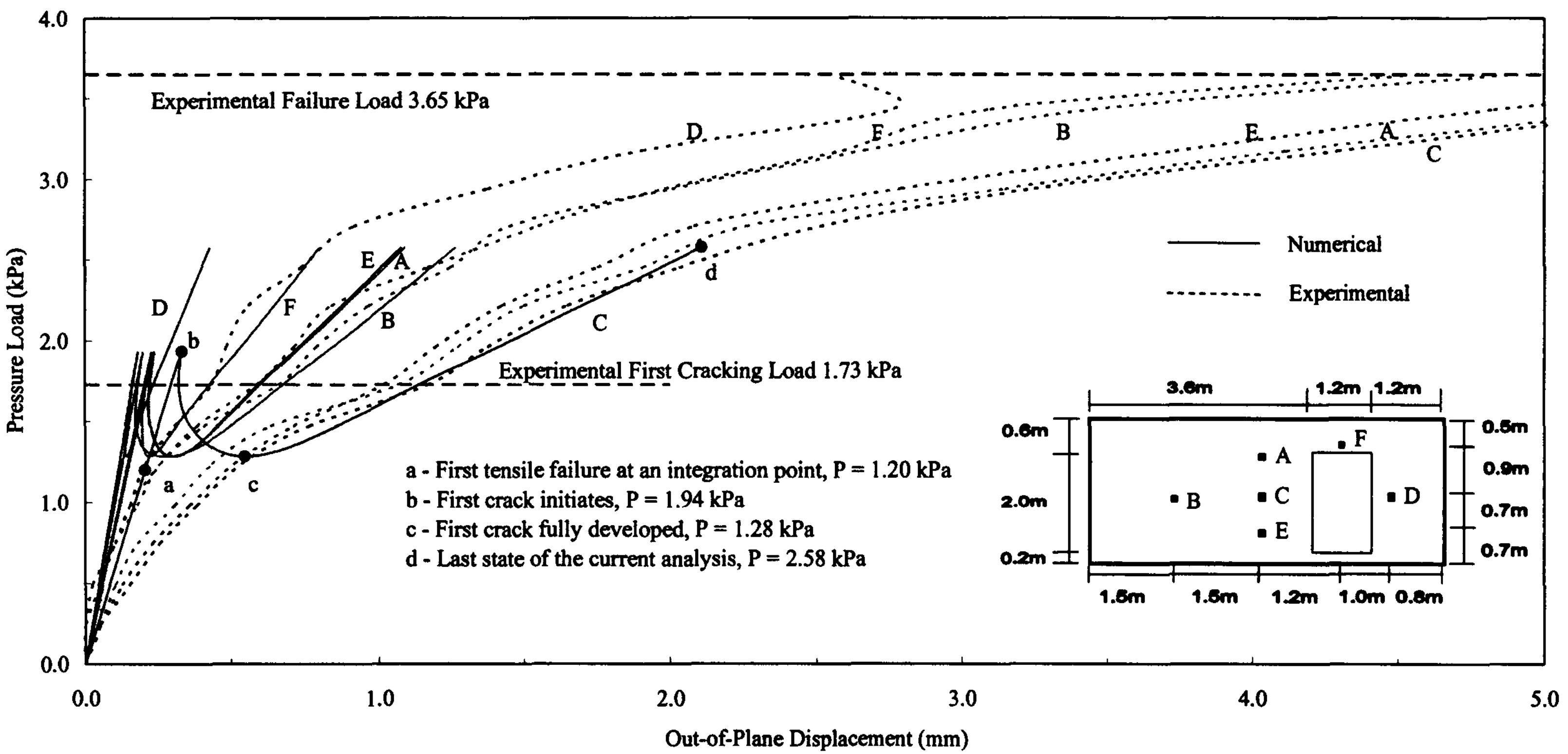


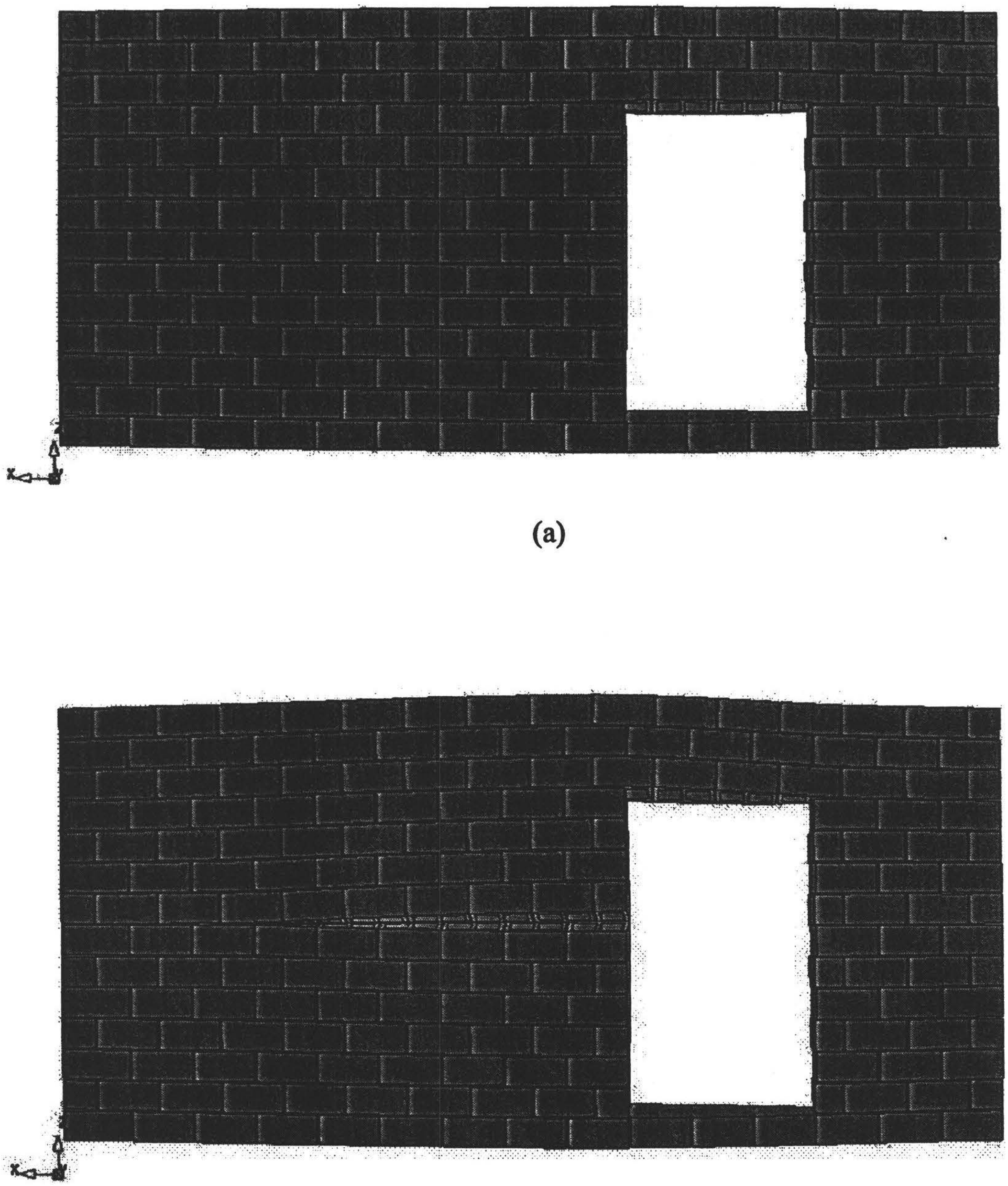
Figure 4.22 Wall BC2 load-displacement diagrams

convergence beyond point d was difficult. At the last load increment of the load-displacement curve (point d), the horizontal cracks located at the left and the top right corner of the opening are fully developed as shown in Figure 4.24. The second mechanism indicated by the diagonal cracks has not clearly formed. Nonetheless, the predicted response generally follows the same trend as reported in the test. The center point of the wall (point C) and the point D at the right pier were measured to be the points with the largest and least deflection, respectively; but larger deflection is observed in point B than in point A and E, which is the expected behaviour but the opposite is observed in the measured response. .

A comparison of the two crack patterns, one observes that the limitations of the current model have affected its ability in predicting the total response, particularly the development of the vertical cracks through the masonry unit. Moreover, the experimentally measured response of the wall, particularly at location B in comparison to locations A and E, indicates that there may be some flexibility at the top and bottom supports of the wall. The combination of the two variations is believed to be why the model did not capture the second mechanism. However, the predicted part of the failure pattern shown in Figure 4.23 and Figure 4.24 basically agrees with that obtained in the test.

4.1.1 Wall BC3

Wall BC3 was designed to include a large central opening with reinforced bond beams placed at the top and bottom of the opening. The development of cracks



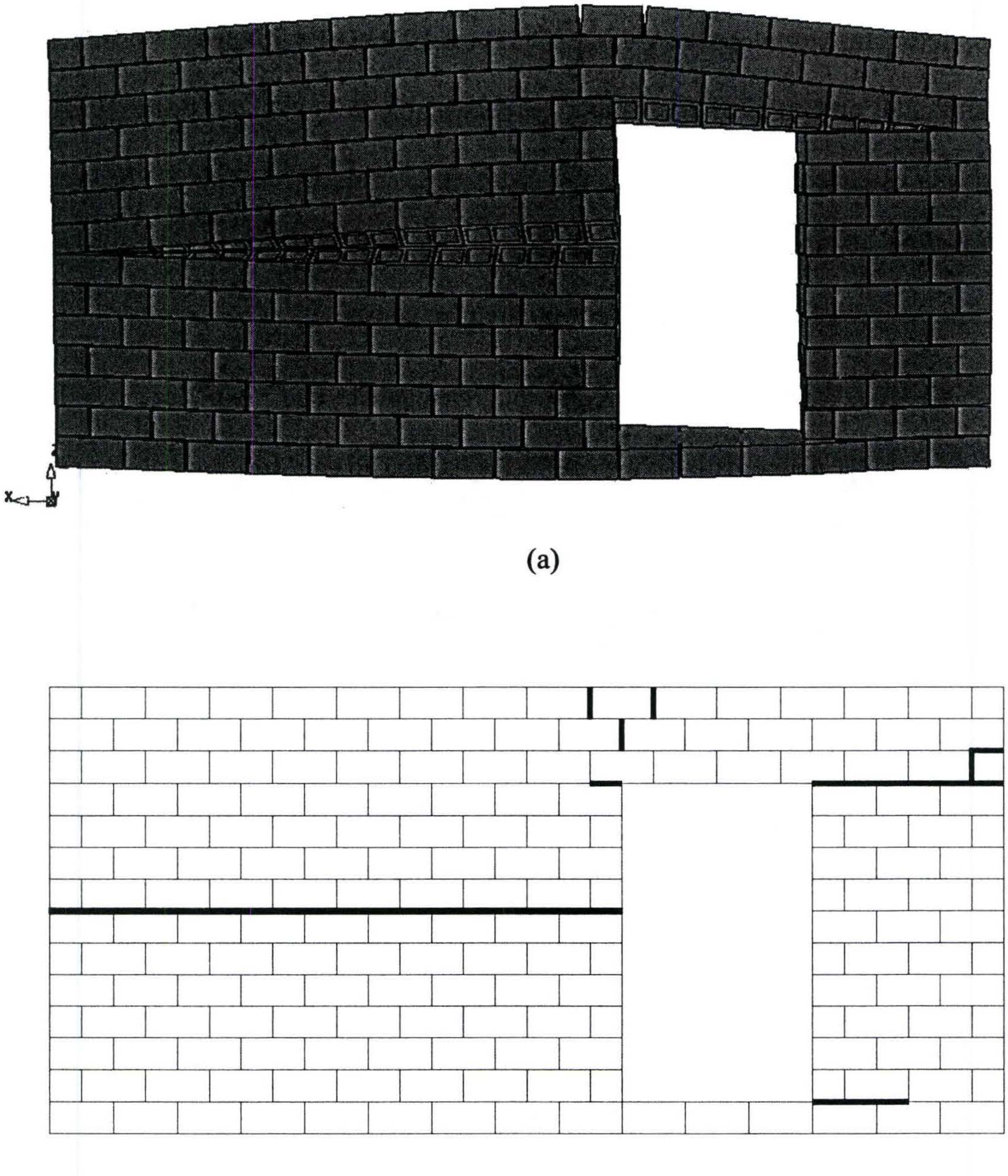
(a)

(b)

Figure 4.23 Crack development for wall BC2:

(a) the first mechanism initiates at the load of 1.94 kPa;

(b) the first mechanism is fully formed at the load of 1.28 kPa.



(a)

(b)

Figure 4.24 Results of analysis for wall BC2 at the load of 2.58 kPa:
(a) deformed mesh; (b) crack pattern.

and the corresponding loads are shown in Figure 4.25. The crack initiated near the top left corner of the opening at the load level of 1.35 kPa, followed by some new horizontal cracks forming at other corners or the edges of the opening. These cracks extended to the edges of the wall with increased loading. At the load of 3.18 kPa, the cracks at the four corners propagated diagonally toward the panel corners, indicating the formation of failure mechanism.

The comparison between the calculated and experimental load-displacement diagrams is shown in Figure 4.26. The data for three points on the wall are presented. For point B and C, the responses predicted numerically are stiffer than the experimental observations. As has been pointed out by Chen (2002), the tested result for point B might be erroneous because the dial gauge was not set up properly or functioning properly. Similar problem might also apply to point C. The result for point A agrees reasonably well up to the formation of the second failure mechanism. The first failure, as shown in Figure 4.27, begins at the right bottom corner of the opening at the load of 1.00 kPa (point a). The experimental first cracking load of 1.35 kPa is again found to lie between this load and the first peak load of 1.58 kPa (point b). Compared to wall BC1 and BC2, a shorter nonlinear softening regime is observed for wall BC3 with only 0.08 kPa decrease in the load before the first crack is fully developed (point c). This could be due to the length of the horizontal cracks given the opening width for wall BC3 is larger. The first shear sliding occurs when the load again increases to 1.49 kPa and the composite tension and shear yielding surface forms immediately. During the second hardening regime, some other horizontal cracks develop at the bottom corners of the opening but are less evident. Mechanism is observed at the second peak load of 2.93 kPa (point d), indicated by the initiation of stepped cracks from the horizontal bed joint at the top corners of the opening. This collapse load is within a 10% difference compared to the experimental data. Afterwards the panel is unable to withstand further loading and the strength is gradually lost.

The predicted crack developments are presented in Figures 4.27 and 4.28. As can be seen in Figure 4.28, the top parts of the predicted failure pattern generally match the experimental observations. The cracks at the bottom corners seem to be less obvious. Figure 4.29, which shows the traction deformation (normal strain) for the peak load, confirms the formation of the failure mechanism.

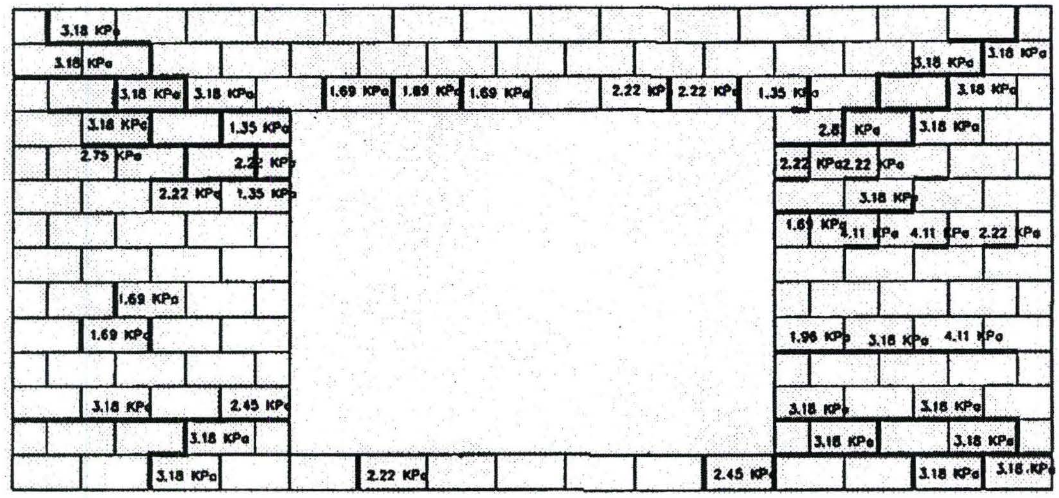


Figure 4.25 Crack pattern for wall BC3, Chen (2002)

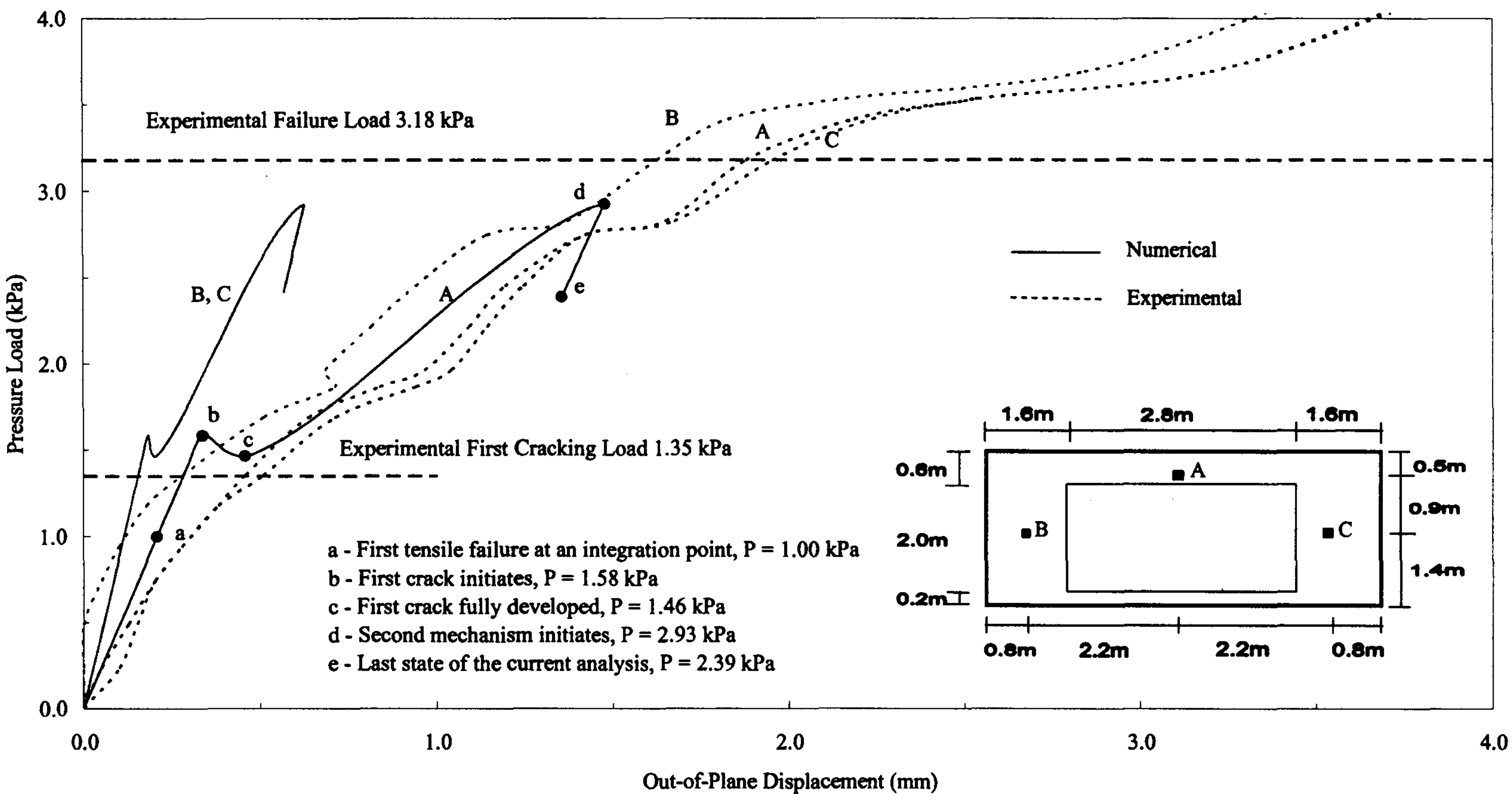
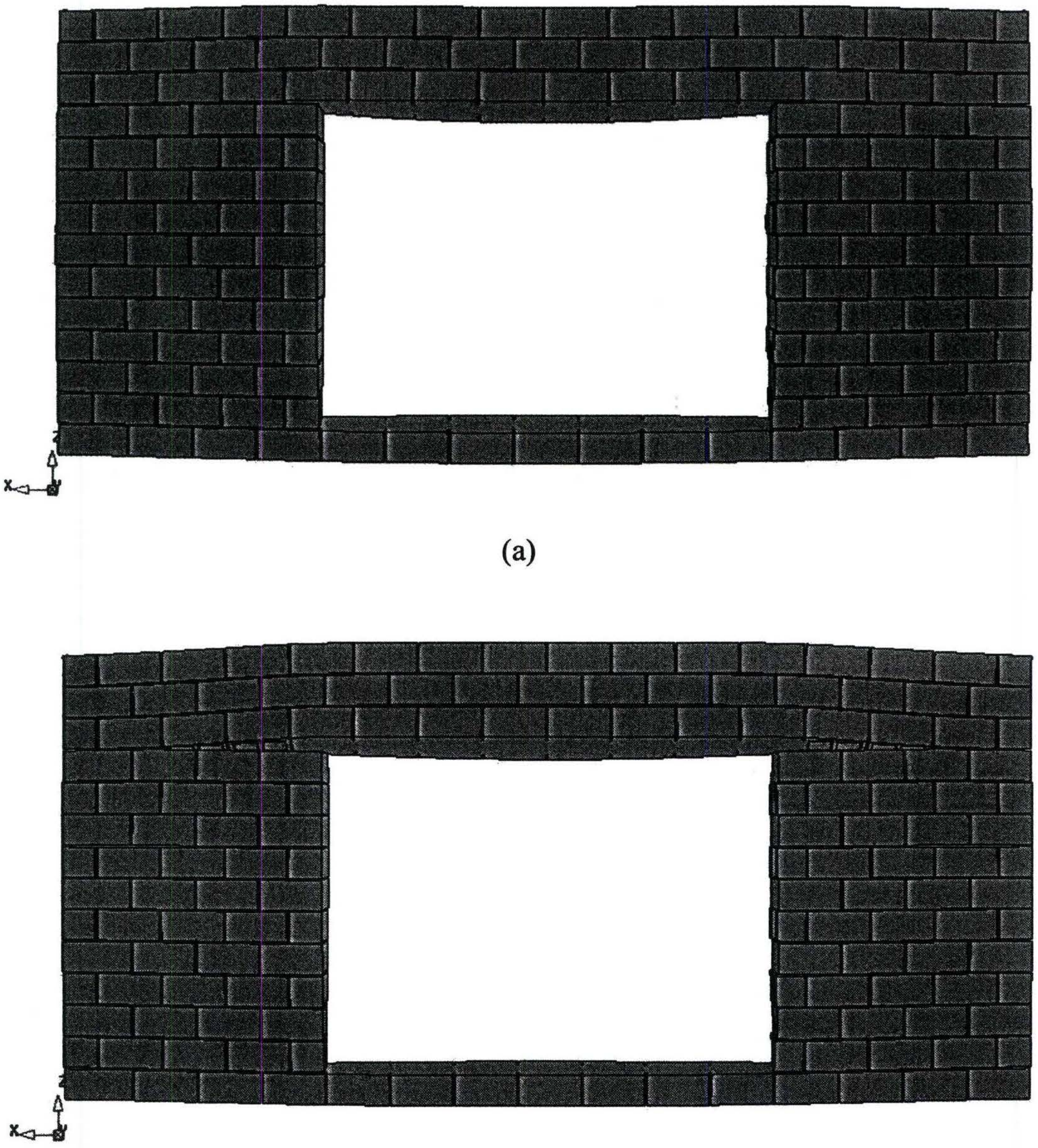


Figure 4.26 Wall BC3 load-displacement diagrams



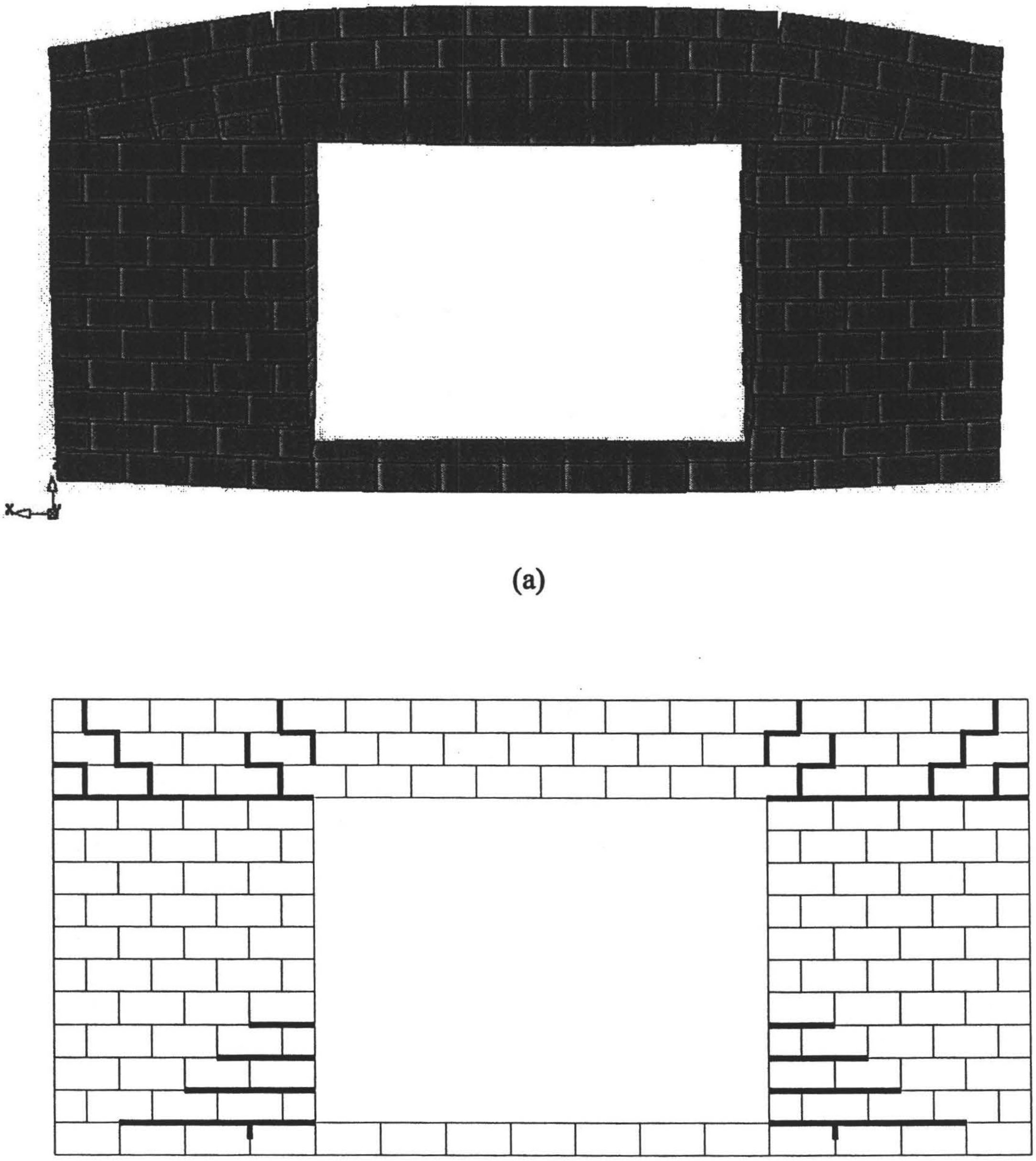
(a)

(b)

Figure 4.27 Crack development for wall BC3:

(a) the first mechanism initiates at the load of 1.58 kPa;

(b) the first mechanism is fully developed at the load of 1.46 kPa.



(a)

(b)

Figure 4.28 Crack development for wall BC3:

(a) the second mechanism initiates at the load of 2.93 kPa; (b) crack pattern.

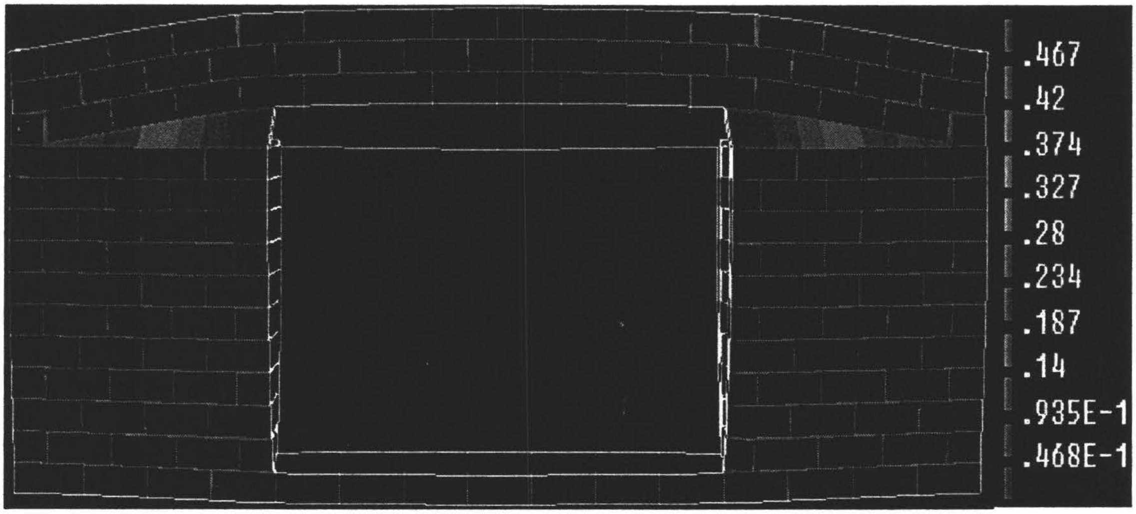


Figure 4.29 Traction deformation in the interface for wall BC3 at the load of 2.93 kPa (peak)

4.5.5 Wall BC4

Wall BC4 had the same opening as wall BC2 but it was located at the center of the wall. The crack development and the corresponding loads are shown in Figure 4.30. The observed first crack was formed at the loading of 1.73 kPa and located near the center of the left edge of the opening. However it did not progress leading to a failure mechanism and thus it was disregarded as the first cracking. The designated first cracks originated at the mid-height of the wall on both sides of the opening when the load reached 1.96 kPa. Another two horizontal cracks were later observed, one at two courses below the mid-height at the left side of the opening and another close to the top right corner of the opening. As the loading increased, diagonal stepped cracks arose from the two top corners and the bottom right corner of the opening. The diagonal crack did not develop at the bottom left corner of the opening, instead, a horizontal crack propagated towards the left. At the load of 3.61 kPa, failure mechanism was formed as these diagonal cracks reached the corner of the wall. An interesting observation is the inclined crack connecting the two horizontal cracks at the bottom left corner of the panel. The scatter of the diagonal cracks might be attributed to the variable tensile strengths of the masonry constituents or the support conditions of the wall.

The comparison between the predicted and experimental load-displacement data are displayed in Figure 4.31 for three points on the wall. Linear behaviour was observed at the early loading stage. Note that in this case the slope of the linear part of the predicted load-displacement curve seems to agree well with that of experimental curve, unlike previous walls where the analytical responses are all stiffer than the tested responses. This might partially explain the scatter of material properties for different wall specimens in the experiment. The first tensile failure begins at the load of 1.06 kPa (point a) but the first mechanism forms at the first peak load of 2.09 kPa (point b). These results are considered good as the experimental first cracking load of 1.96 kPa fits within this range. The first crack is fully developed when the load decreases to 1.46 kPa through a nonlinear softening regime. The load drop of 0.63 kPa from the first peak is quite close to that observed for wall BC2, this is expected since the two walls have the same opening size and with equal length for the first crack. The composite tension and shear yield surface is formed at the interface of the top left corner of the opening when the load decreases to 1.68 kPa. Immediately after the first crack is fully developed, the first shear sliding occurs, where the load is 1.51 kPa.

Under the increased loading, the panel undergoes a second hardening. The numerical response is found stiffer than the experimental response. The sensitivity to the load increment also results in convergence difficulty for this wall. Up to the last point of the load-displacement curve (point d), only the diagonal cracks in the upper panel corner are slightly formed. Nevertheless, the behaviour is believed to follow the same trends as reported in the test. Figure 4.32 and Figure 4.33 show the crack development recorded in the analysis.

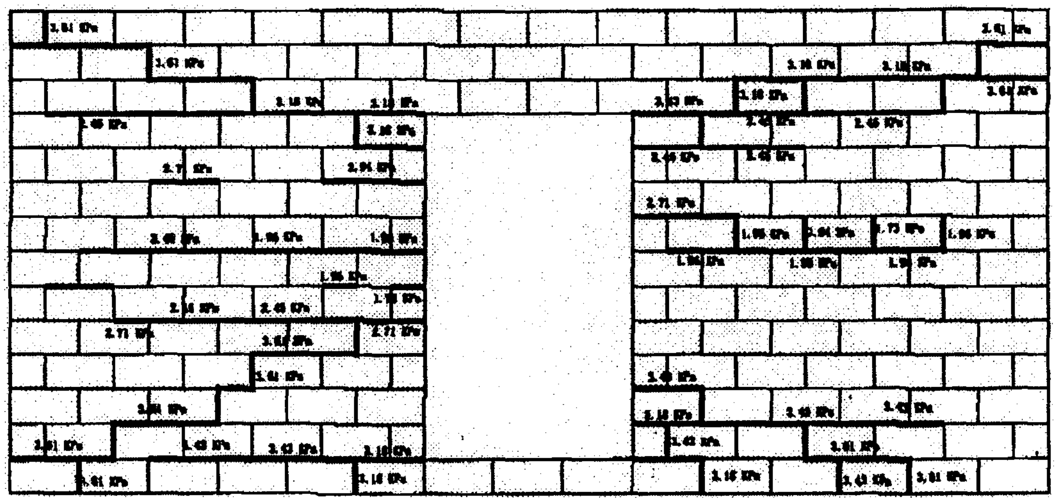


Figure 4.30 Crack pattern for wall BC4, Chen (2002)

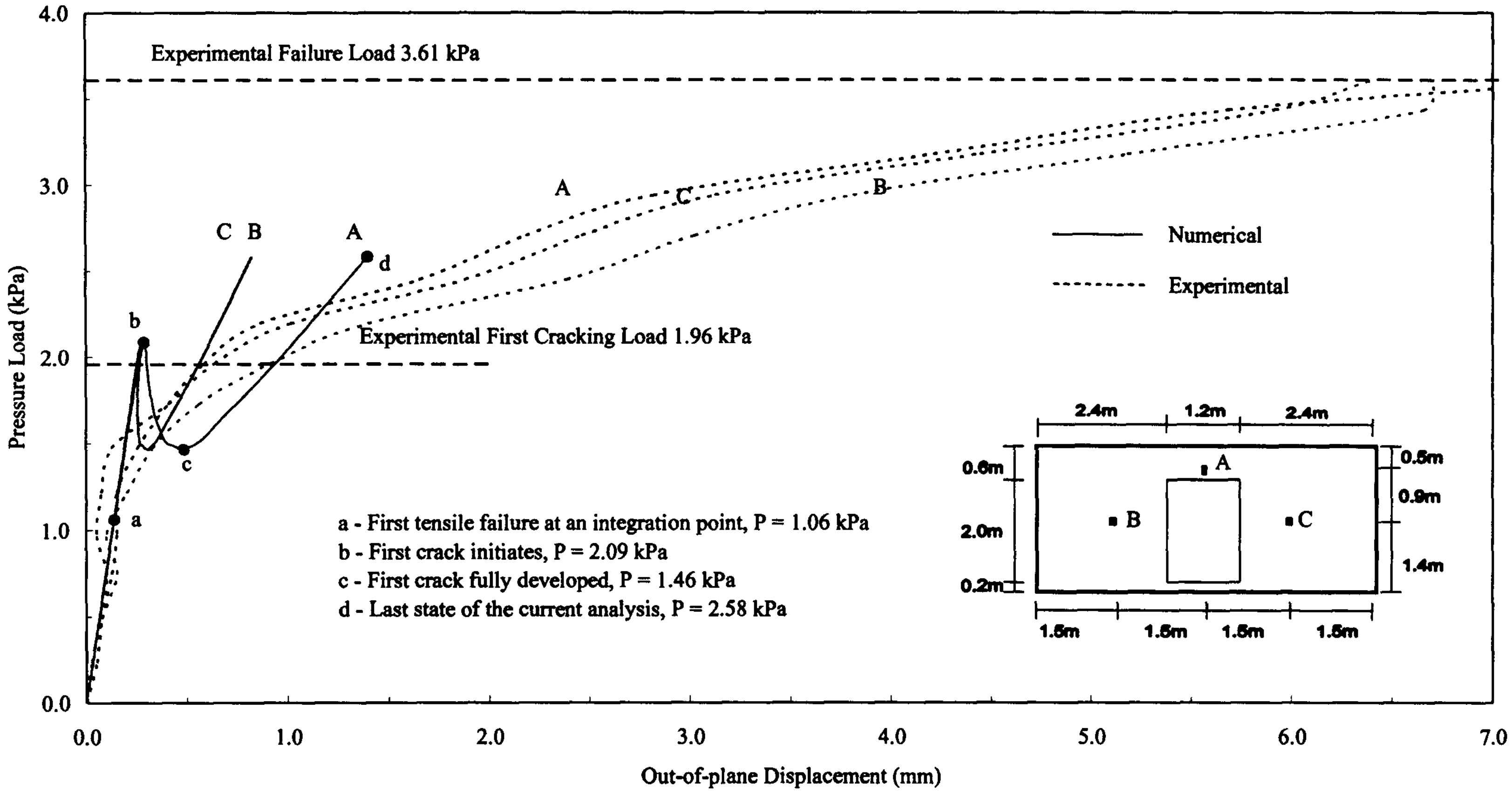
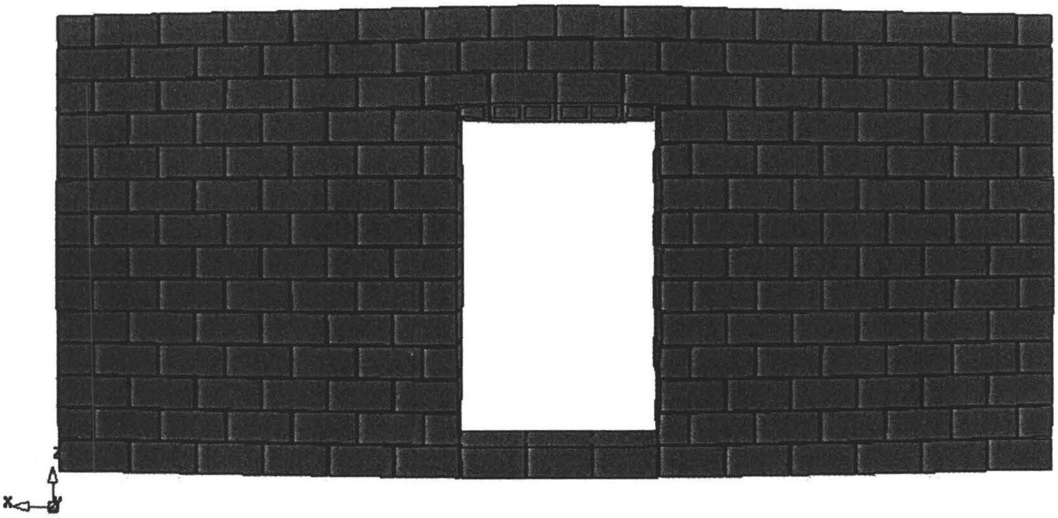
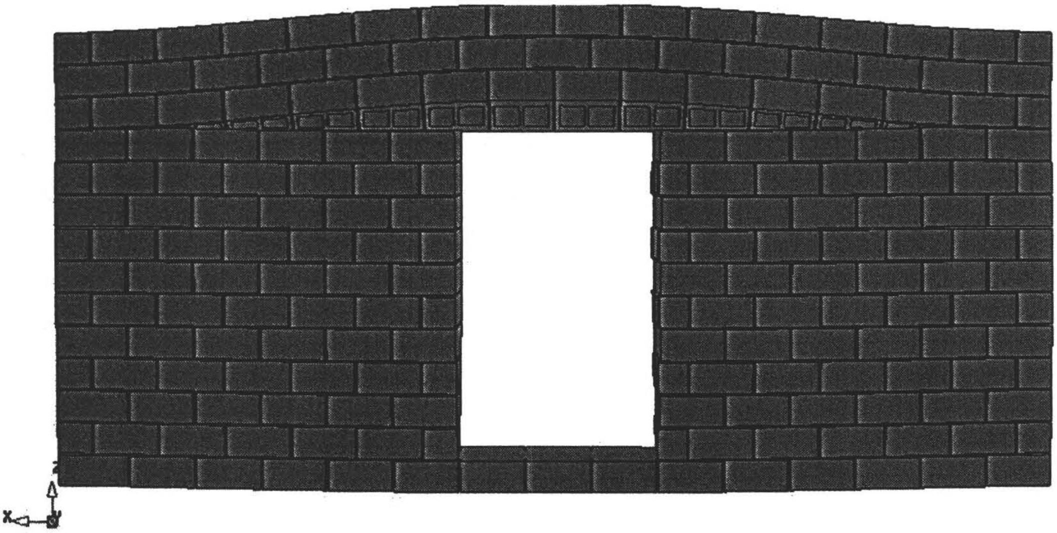


Figure 4.31 Wall BC4 load-displacement diagrams



(a)

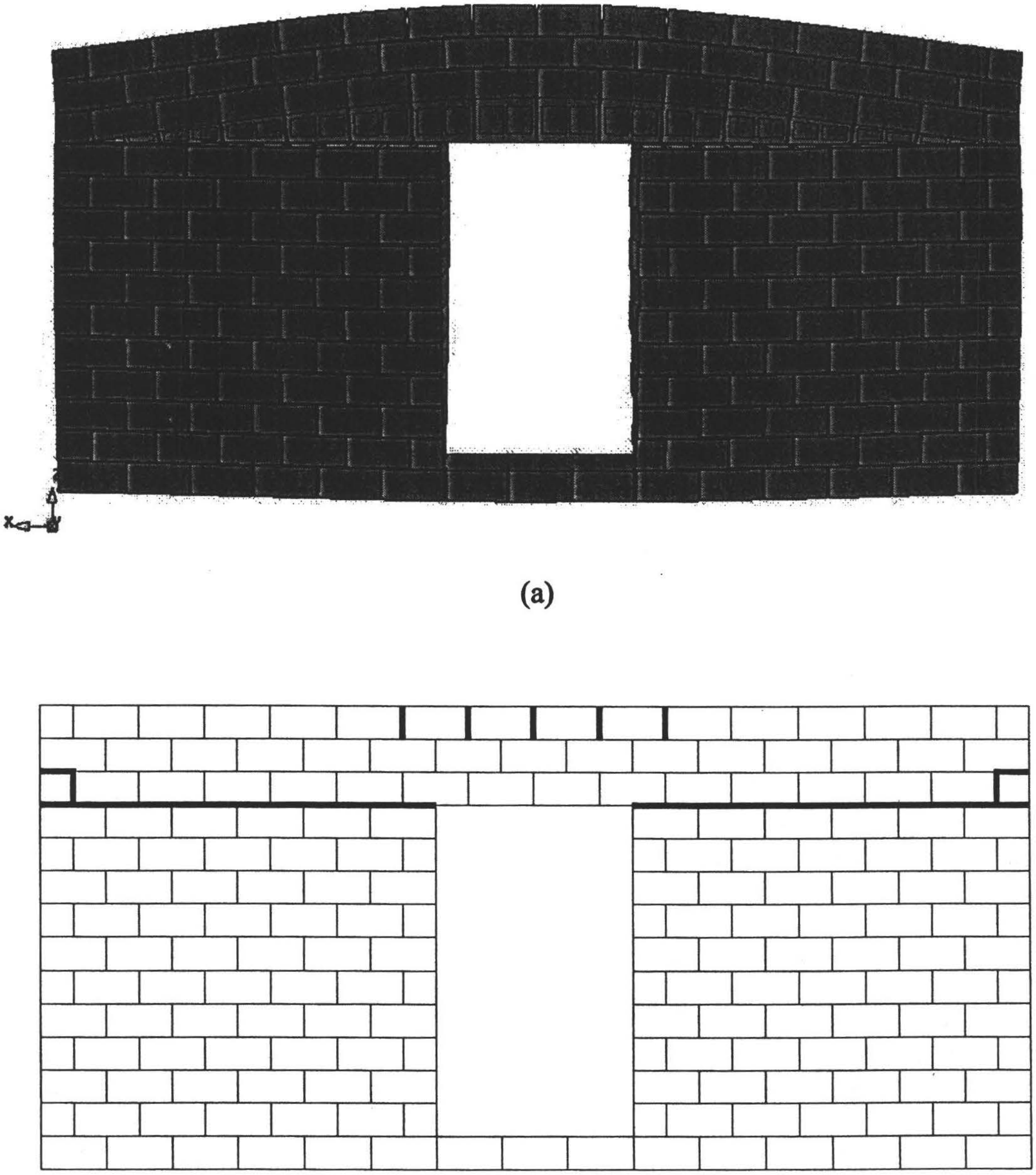


(b)

Figure 4.32 Crack development for wall BC4:

(a) the first mechanism initiates at the load of 2.09 kPa;

(b) the first mechanism is fully formed at the load of 1.46 kPa.



(a)

(b)

Figure 4.33 Results of analysis for wall BC4 at the load of 2.58 kPa: (a) deformed mesh; (b) crack pattern.

Closer examination of the two results indicates that the experimental results, particularly displacements of point B and C should be similar and different from point A. The numerical model predicted equal displacements at B and C, and lower in comparison to A; whereas, the experimental results show that A and C are similar, and lower in comparison to B. Although it is difficult to provide a rational explanation for this behaviour, one suspects the support conditions of the wall. Also, given the initiation of cracks all over the face of the wall, one suspects that curing of the mortar may have been problematic leading to excessive shrinkage. Given the uncertainties, the predicted response of wall BC4 lies within the experimentally measured load range. Predictions of the displacement continue to be problematic.

4.5.6 Wall BC5

Wall BC5 represents a masonry wall with a longitudinal window located at the mid-length of the wall. A reinforced bond beam was placed at the top of the opening. The experimental crack pattern and the corresponding cracking loads are shown in Figure 4.34. The first crack was observed at the bottom left corner of the opening, accompanied with several vertical cracks forming along the bottom of the opening. As the loading increased, the first crack extended to almost full length of the left pier and some localized cracks occur around the other edges and corners of the opening. The mechanism formed when several diagonal cracks developed from the perimeter of the opening and propagated toward the edges and corners of the wall. The first cracking load and failure load were recorded as 1.47 kPa and 2.90 kPa, respectively.

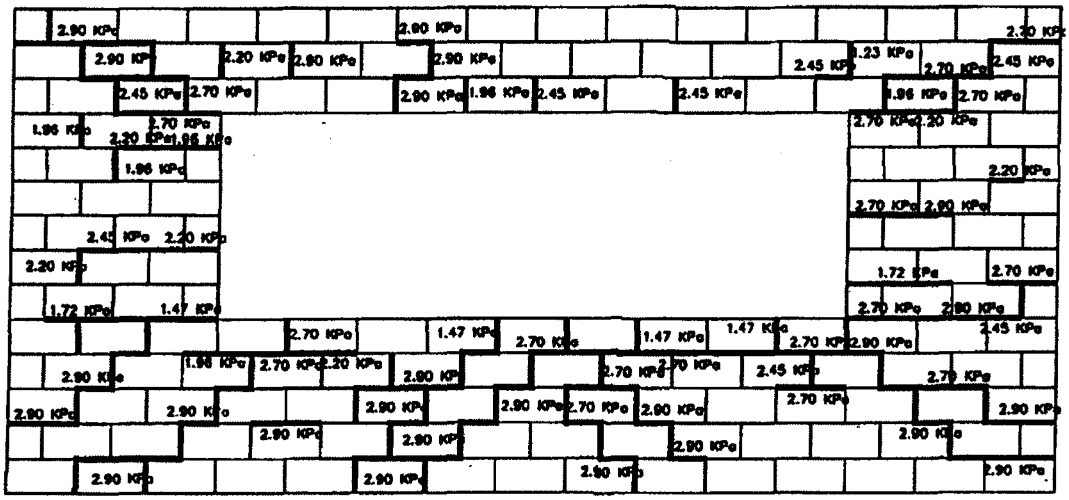


Figure 4.34 Crack pattern for wall BC5, Chen (2002)

The calculated and experimental load-displacement diagrams are given in Figure 4.35 for four locations on the wall. Globally, the trend of the structural response obtained numerically agrees with the experimental results, but the predicted post-cracking behaviour appears to be stiffer than the experimental observations. Another difference is the calculated displacement at point A which is greater than the one calculated at point B and C. This is opposite to the experimentally reported values. The predicted first tensile failure occurs at the load of 0.60 kPa, which is much lower than the experimental first cracking load of 1.47 kPa. However, the predicted actual first mechanism seems to begin at the

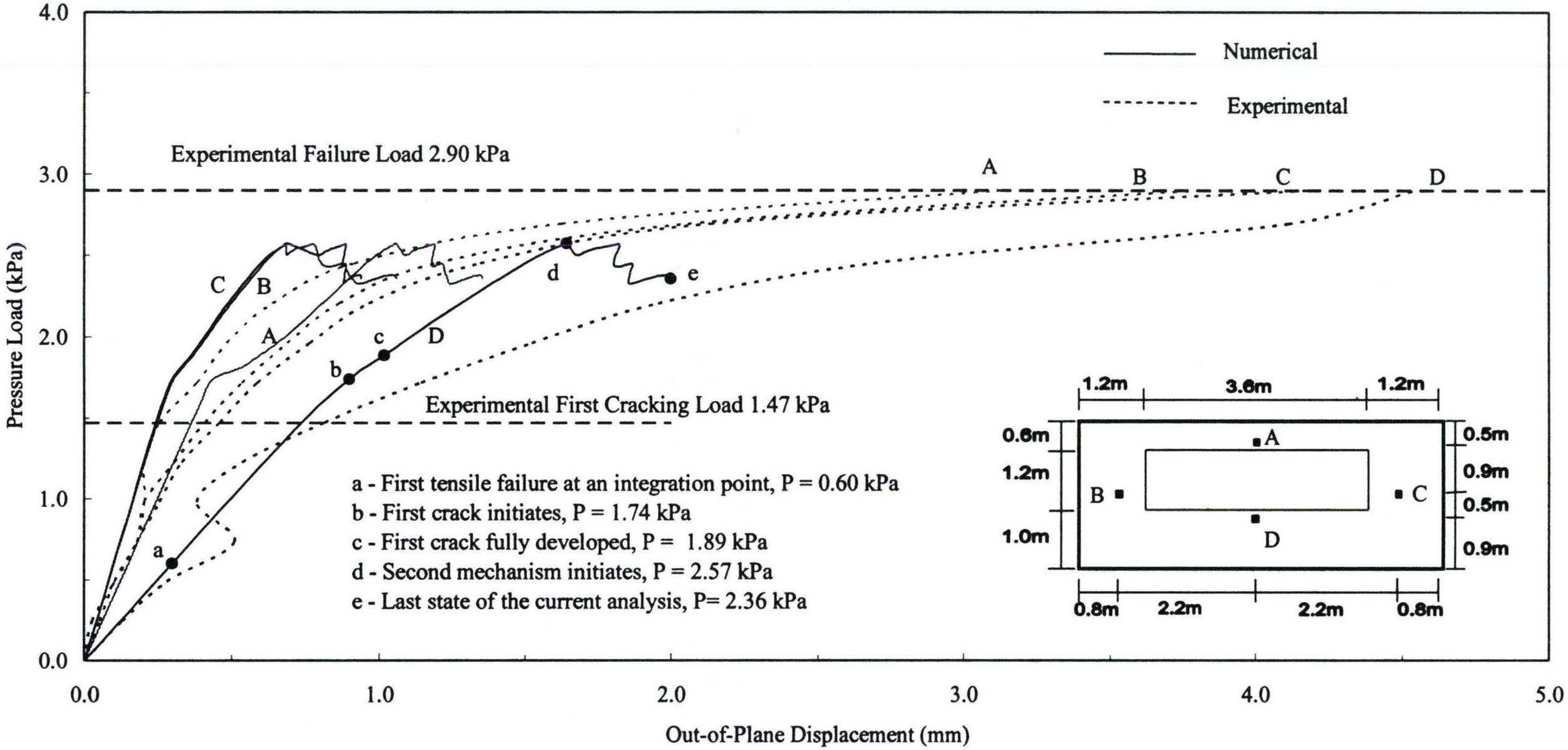
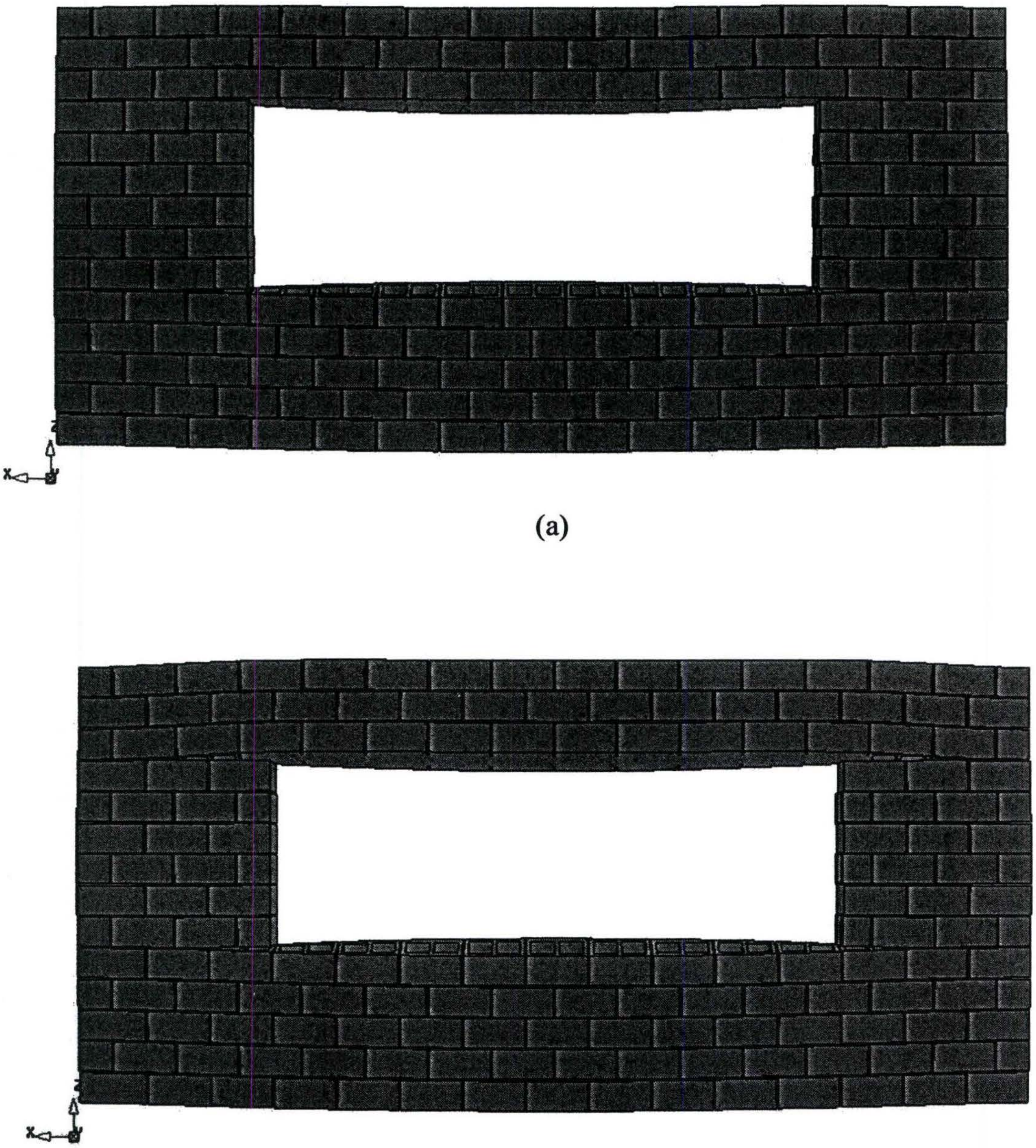


Figure 4.35 Wall BC5 load-displacement diagrams

load of 1.74 kPa, where an obvious reduction of stiffness is be observed thereafter.

The shear sliding occurs at the loading of 0.8 kPa and the composite yield surface of tension and shear forms at the loading of 1.58 kPa. Both phenomena arise prior to the complete formation of the first mechanism, which is achieved when the load reaches 1.89 kPa. In addition, the nonlinear softening after the initiation of the first mechanism is not observed in this wall. This response is different in comparison to the other walls with shorter opening width and longer remaining wall length. The second mechanism is observed at the peak load of 2.57 kPa, where the diagonal stepped cracks initiate. This load is within a 12% of the experimental value. After the peak load, the load-displacement relationship exhibits sort of oscillation, which might be attributed to the opening of some new cracks and closing of some original cracks due to deformation.

The experimentally observed failure pattern is well reproduced except for some localized cracks, as illustrated in Figure 4.36 to Figure 4.38. Initially, the horizontal cracks start to develop from the bottom corners of the opening. Then, other horizontal cracks arise at the top corners of the opening. Note that both the panels above and below the opening are supported on three sides and have a short portion of the fourth side connected to the two piers via bed joints. The cracks initiate first between the piers and the bottom panel, because the bottom panel has larger vertical span than the top panel and thus attracts a larger portion of the load via the bed joint. Under increasing load, the diagonal cracks arise from these



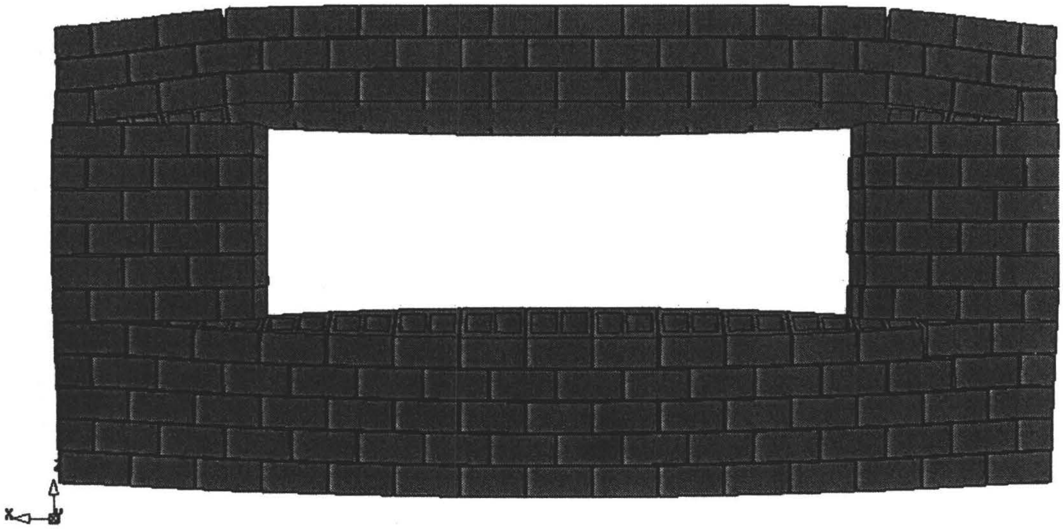
(a)

(b)

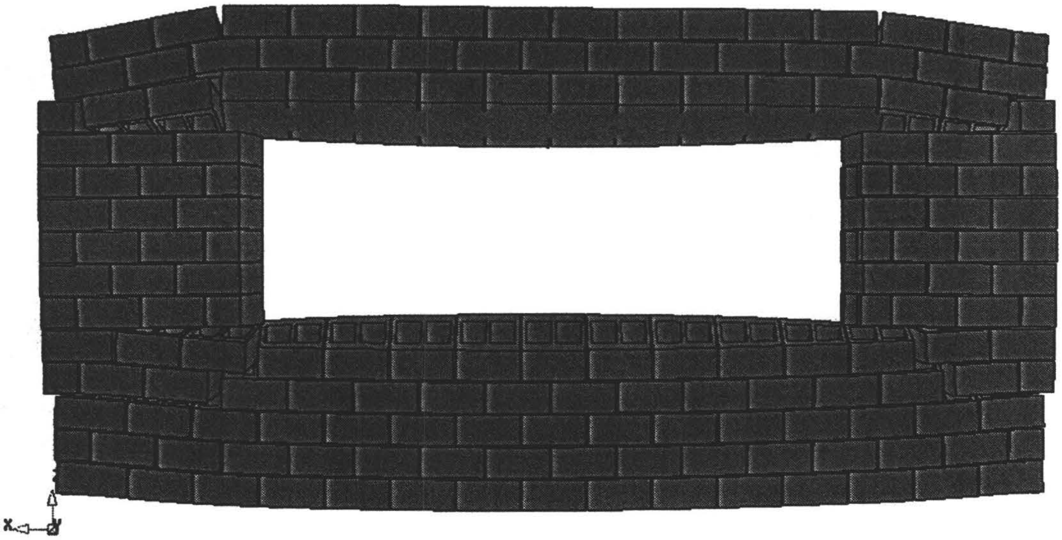
Figure 4.36 Crack development for wall BC5:

(a) the first mechanism initiates at the load of 1.74 kPa;

(b) the first mechanism is fully formed at the load of 1.89 kPa.



(a)



(b)

Figure 4.37 Crack development for wall BC5:
(a) the second mechanism initiates at the load of 2.57 kPa;
(b) the second mechanism is fully formed at the load of 2.36 kPa.

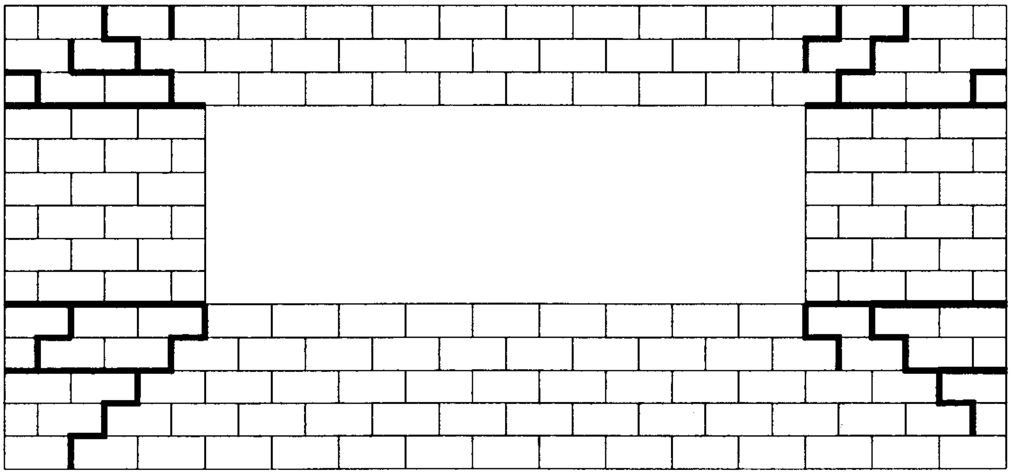
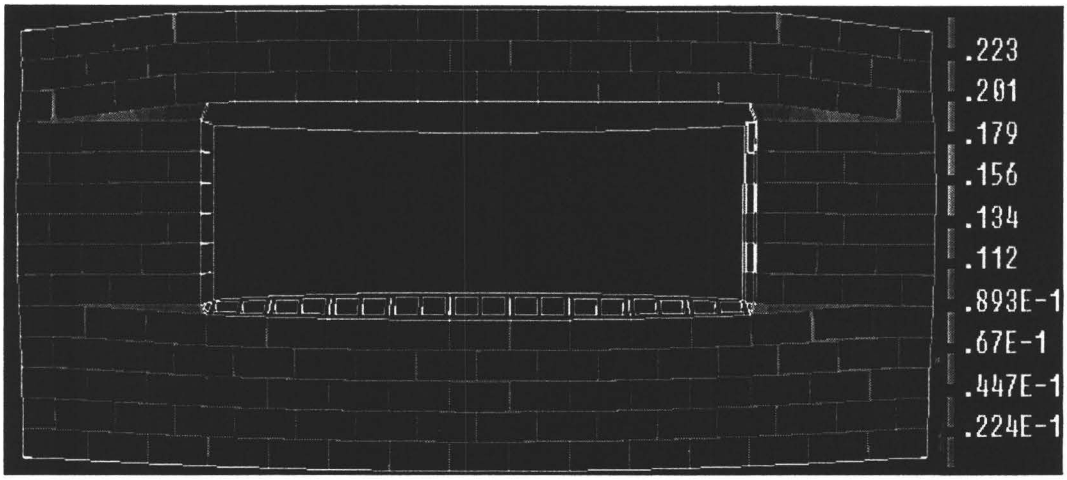


Figure 4.38 Crack pattern for wall BC5

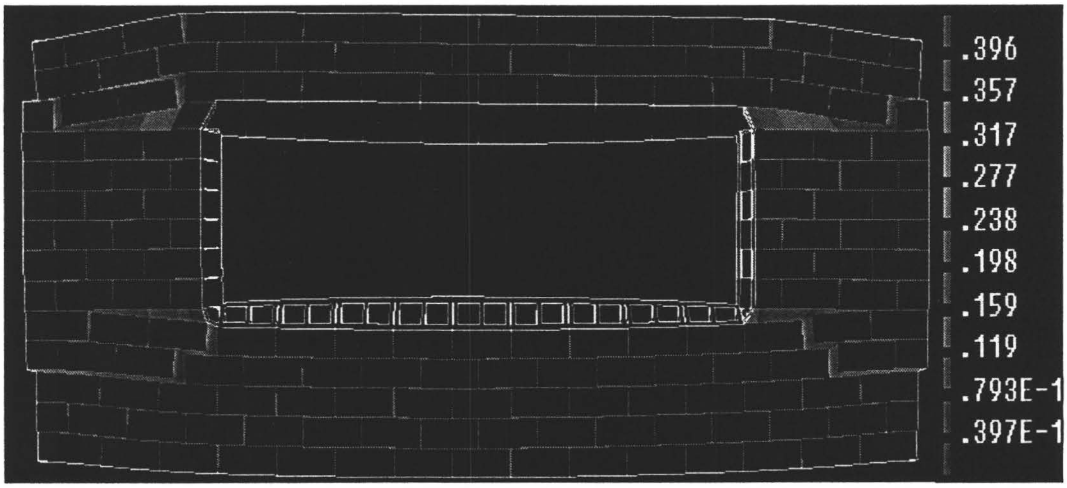
horizontal cracks and propagate to the four corners of the wall, leading to the failure. The formation of failure mechanism can be further confirmed by the observation on strain distribution as shown in Figure 4.39.

4.5.7 Wall BC6

Wall BC6 has two equal sized small openings with each located 1.2 m off the center. No bond beam was cast at the openings. The crack pattern and the corresponding cracking loads obtained in test are shown in Figure 4.40. Noteworthy, vertical cracks were first observed at the bottom of the openings, this might be due to the local horizontal bending between the left, central and right piers, which served as elastic supports for the panels under the openings. At a loading of 1.47 kPa, the designated first crack formed as a horizontal crack at the mid-height of the central pier between the two openings. Under increasing



(a)



(b)

Figure 4.39 Traction deformation (strain) in the interface for wall BC5 at a load of: (a) 2.57 kPa (peak); (b) 2.36 kPa (92% post-peak).

load, two horizontal cracks initiated from the bottom outer corners of the two openings and propagated to the wall supports. When the loading increased to 2.79 kPa, diagonal cracks originated from the top outer corners and bottoms of the openings and extended to the wall corners, indicating the formation of failure mechanism. The bottom diagonal cracks did not form at the bottom corners but near the bottom centers of the openings. This is believed to be due to the larger horizontal bending moments at the bottom of the openings.

Figure 4.41 shows the calculated and experimental load-displacement diagrams. The data for five points on the wall were plotted. Similar to previous walls, the load-displacement curve begins with a linear elastic relationship, followed by a nonlinear softening behaviour before entering to the second hardening regime with a reduced stiffness. Note that in general the panel center point C yields larger displacement in comparison with experimental result, unlike

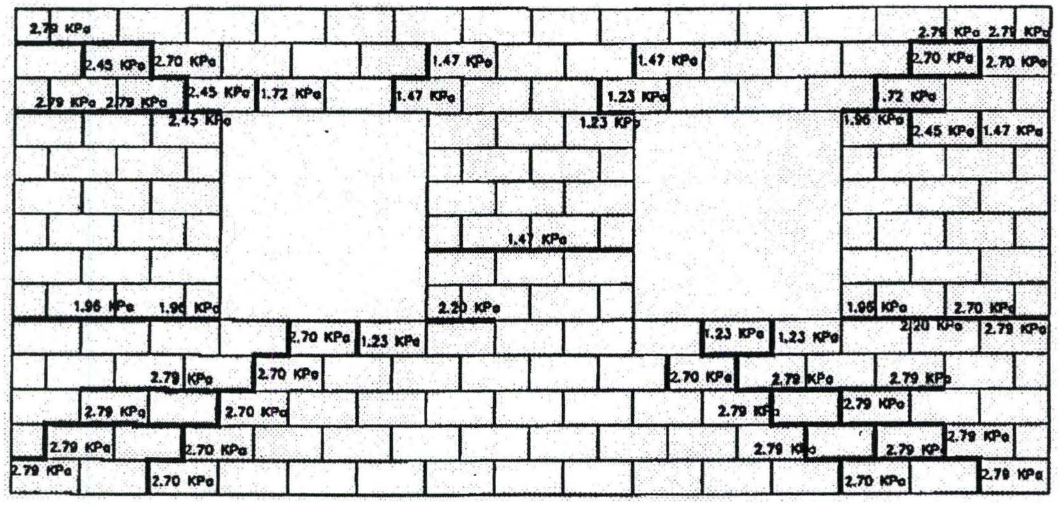


Figure 4.40 Crack pattern for wall BC6, Chen (2002)

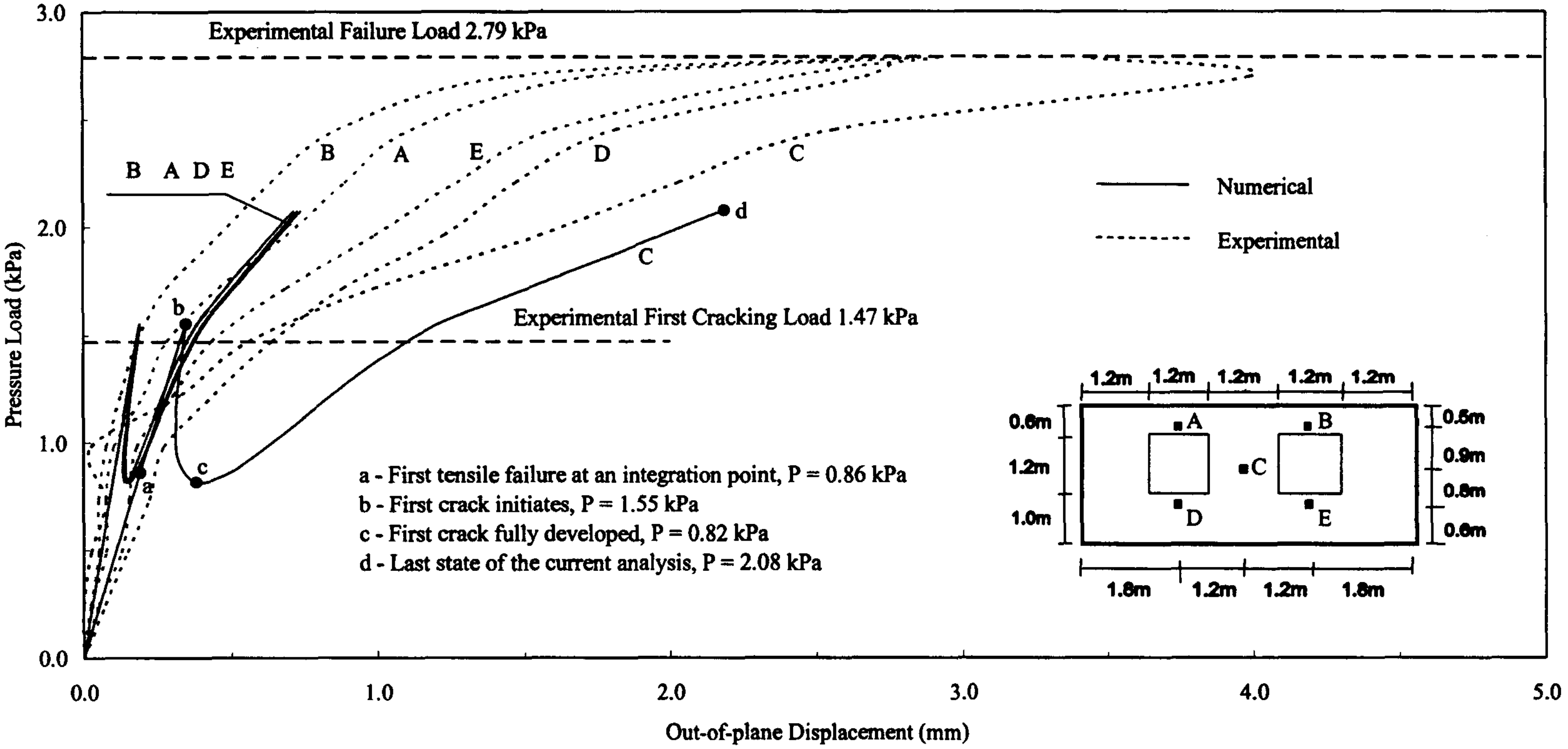
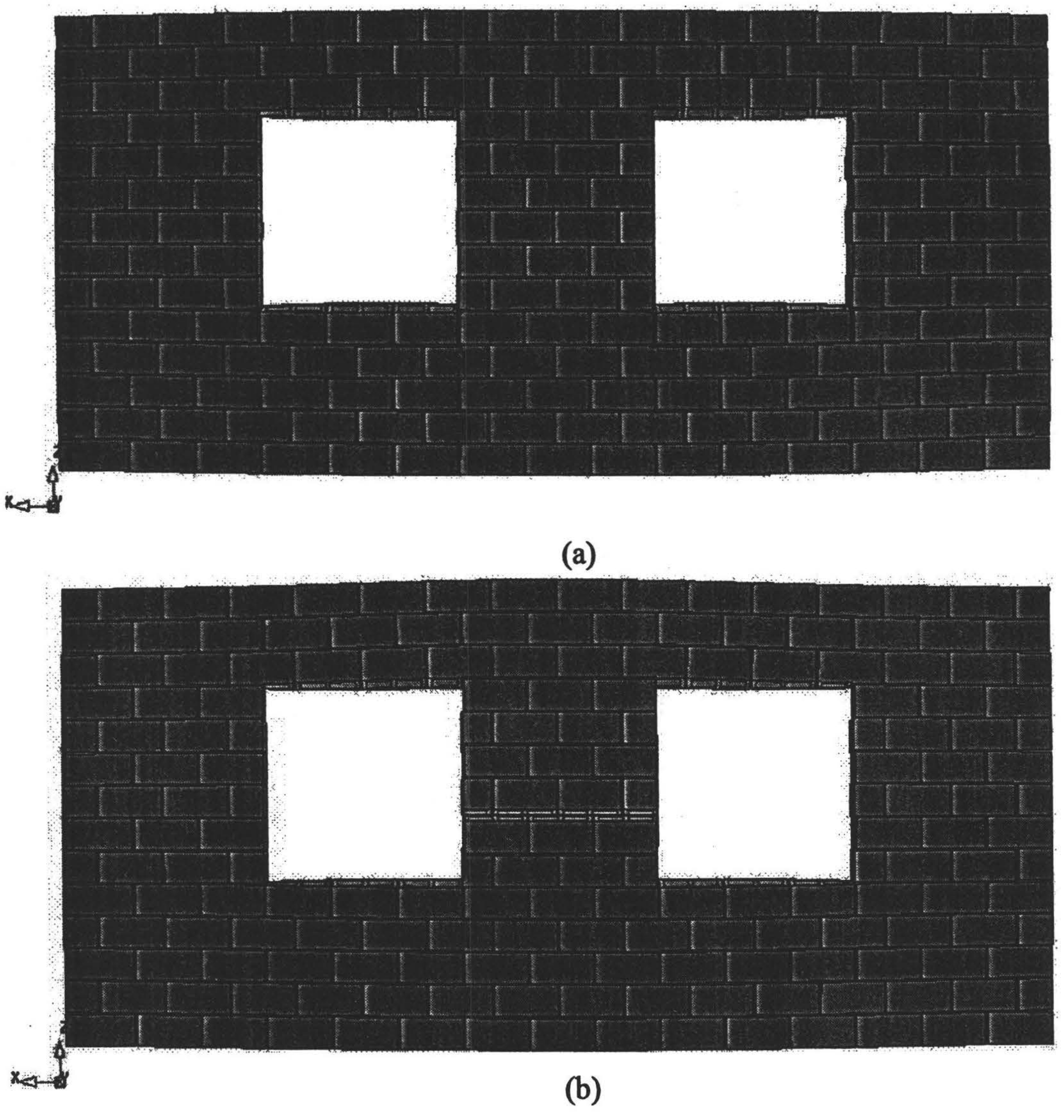


Figure 4.41 Wall BC6 load-displacement diagrams

most of the previous findings where the predicted response is always stiffer than the experimental observations. Is this due to variations in the material properties, shrinkage cracks or flexible support condition? It is probably a combination of all three. Close examination of all the other points, namely A, B, D and E, the predicted responses seem to have similar displacements for the full loading history. This is understandable as point A and D are symmetric to point B and E, respectively; and point A (or B) is at same proximity to the support as point D (or E). In contrast, experimental values seem to yield significant differences of the recorded displacements at these four points.

The first tensile failure initiates at the load of 0.86 kPa (point a) but the first mechanism actually forms when the load increases to 1.55 kPa (point b). The experimental first cracking load of 1.47 kPa is within this range. The load in the analysis then decreases as the first crack develops. The composite tension and shear yield surface is formed at the interface of the bottom outer corners of the opening when the load decreases to 1.37 kPa. At the same loading level, the first shear sliding occurs at these locations. The first crack is fully developed when the load decreases to 0.82 kPa (point c). A load decrease of 0.73 kPa from the first peak (point a) results in a significant lost of strength due to the complete formation of the first crack located at the central pier. The analysis of wall BC6 also shows convergence difficulty during the second hardening range. At the last point of the load-displacement curve (point d), the diagonal cracks in the upper panel corner are slightly formed. The predicted behaviour is however anticipated to follow the same trend as the experimentally observed one. The predicted crack developments are shown in Figures 4.42 and 4.43. They illustrate the same pattern as the experimentally observed one.

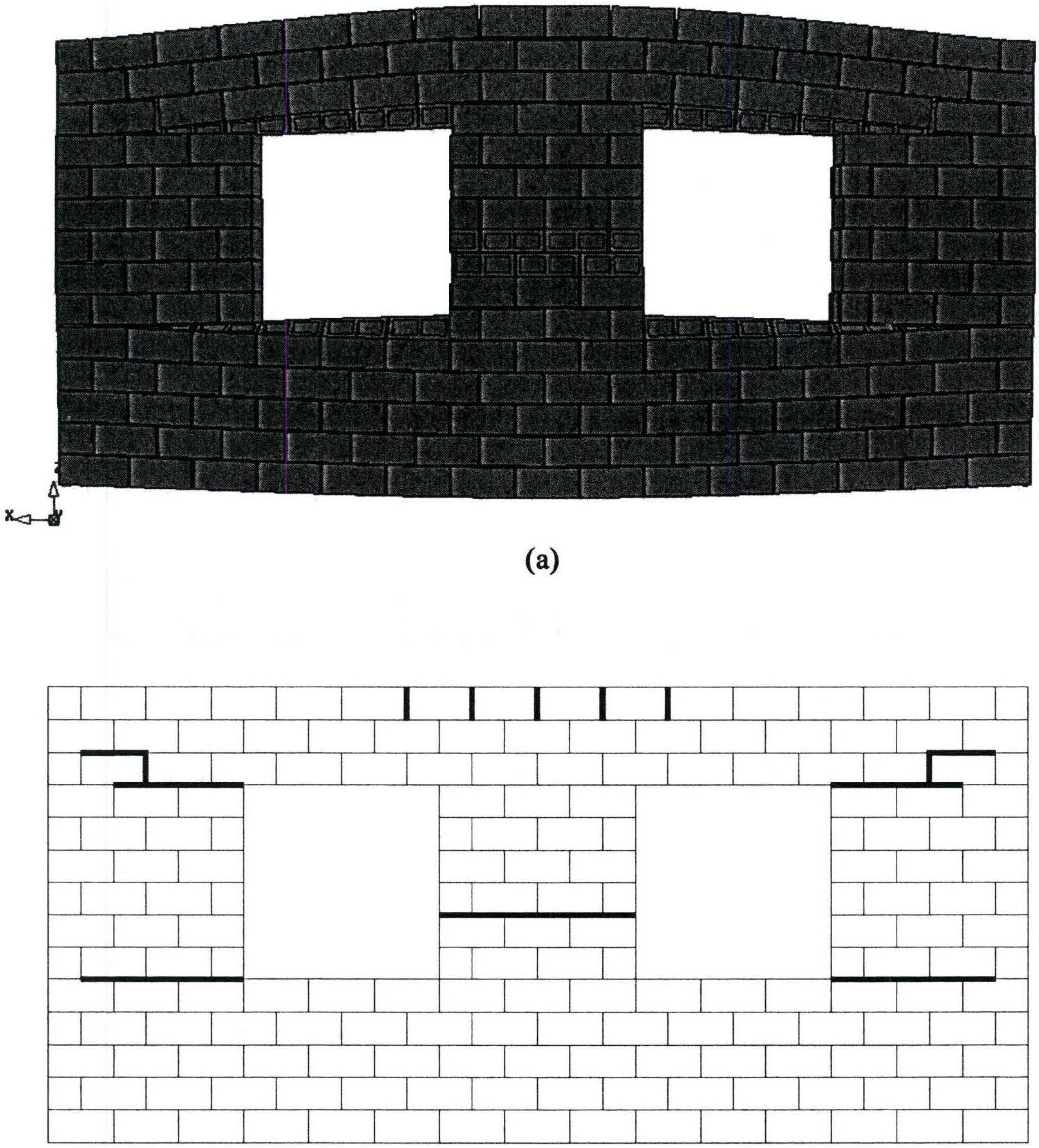

(a)

(b)

Figure 4.42 Crack development for wall BC6:

(a) the first mechanism initiates at the load of 1.55 kPa;

(b) the first mechanism is fully formed at the load of 0.82 kPa.



(a)

(b)

Figure 4.43 Results of analysis for wall BC6 at the load of 2.08 kPa: (a) deformed mesh; (b) crack pattern.

4.5.8 Wall BC7

Wall BC7 has only one opening with the same size and location as the right opening in the wall BC6. No bond beam is placed at either the top or the bottom of the opening. As shown in Figure 4.44, the defined first crack originated one course above the bottom left of the opening. At about twice the load of the first cracking load, another horizontal crack developed at one course below the top left of the opening; meanwhile, a crack at the top right corner of the opening was also formed, followed immediately by a diagonal crack propagating to the panel corner. The full mechanism was identified by the formation of diagonal cracks at the bottom corners of the opening and the left two corners of the wall panel. In addition, an inclined crack developed between the two horizontal cracks at the left part of the wall panel. The first cracking load and failure load were recorded as 1.72 kPa and 3.22 kPa, respectively.

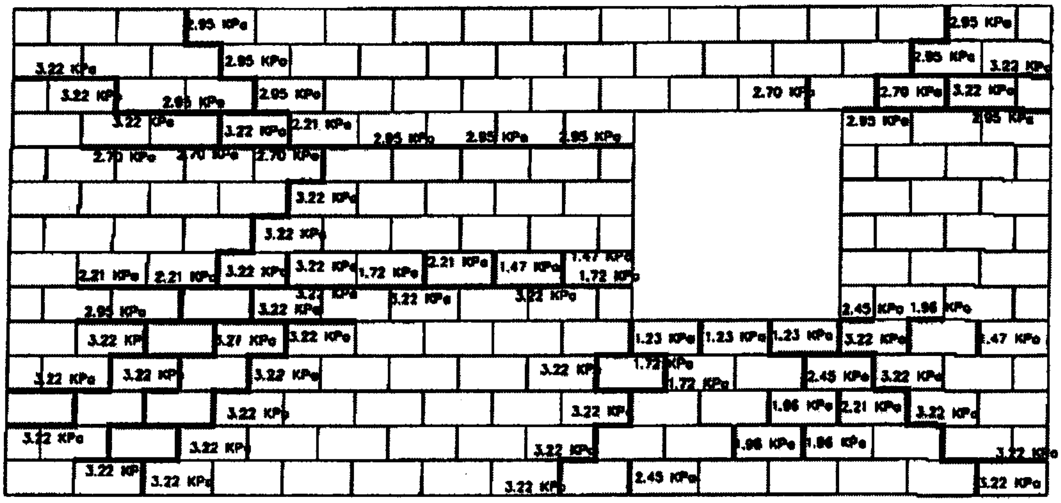


Figure 4.44 Crack pattern for wall BC7

The analysis of wall BC7 was carried out up to the load level of 3.10 kPa and was stopped due to the difficulties in achieving convergence regardless of the size of the load increment. Nonetheless, in general the behaviour is well captured by the model and reasonable agreement can be found in comparison with the experiment, see Figure 4.45. The structural response obtained numerically is stiffer than the experimental results. The predicted first tensile failure occurs near the bottom right corner of the opening at the load of 1.0 kPa (point a) but the continuous crack is observed after reaching the first peak load of 1.99 kPa (point b). This crack initiates at the bottom left of the opening and propagates towards the left support of the panel. Thereafter, the deflection continues to increase with a decreasing load.

The first shear sliding occurs at the mid-length of the crack when the load drops to 1.80 kPa. At the same load, the composite yield surface of tension and shear forms. As the load decreases to 1.60 kPa, the first mechanism is fully formed. Thereafter, the curve exhibits strong nonlinear hardening behaviour. In this range, the second horizontal crack is developed, located at the top right corner of the opening. The diagonal crack starts to develop soon at this location. The crack developments are shown in Figure 4.46 and Figure 4.47. The predicted crack pattern is again similar to the experimental one; however, it is not complete.

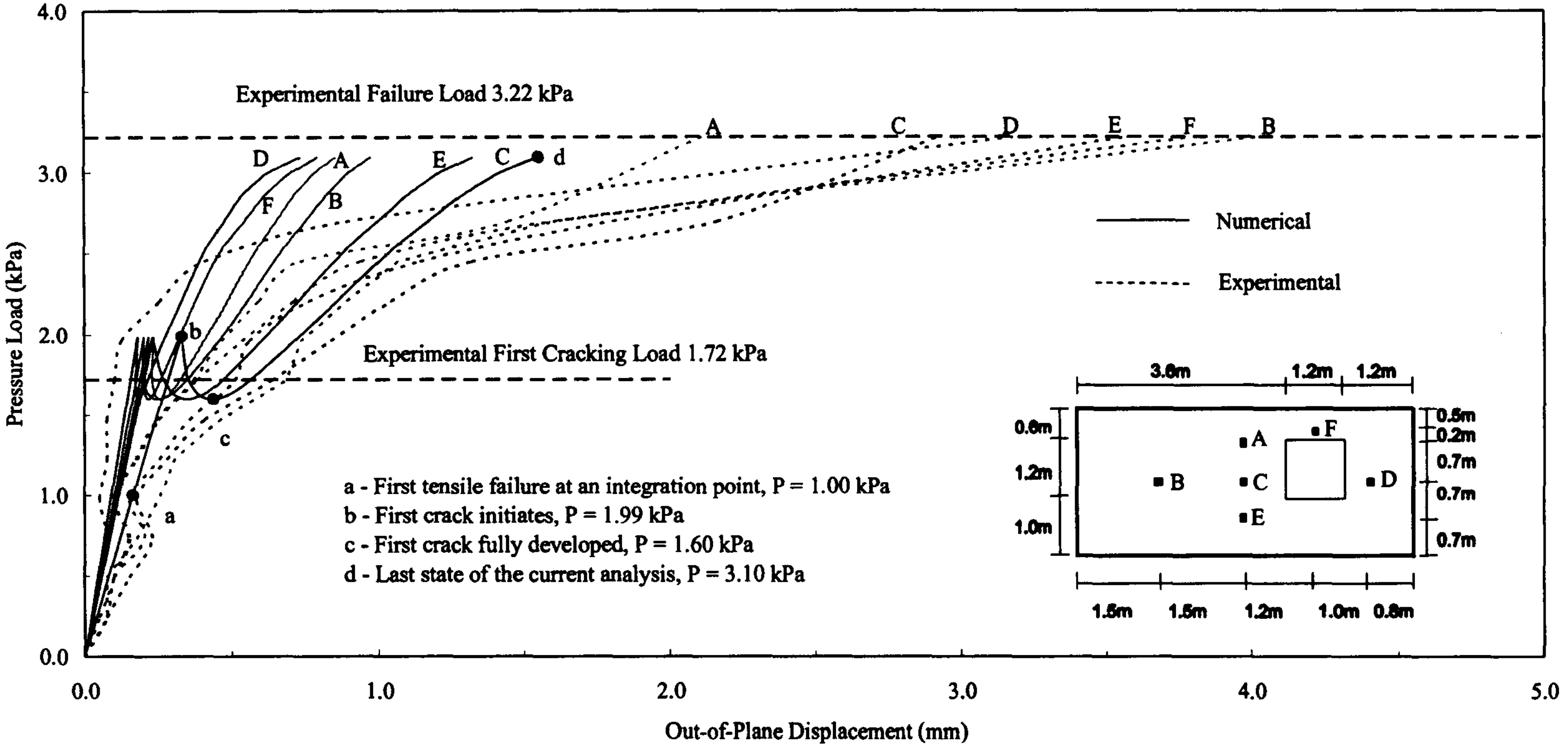
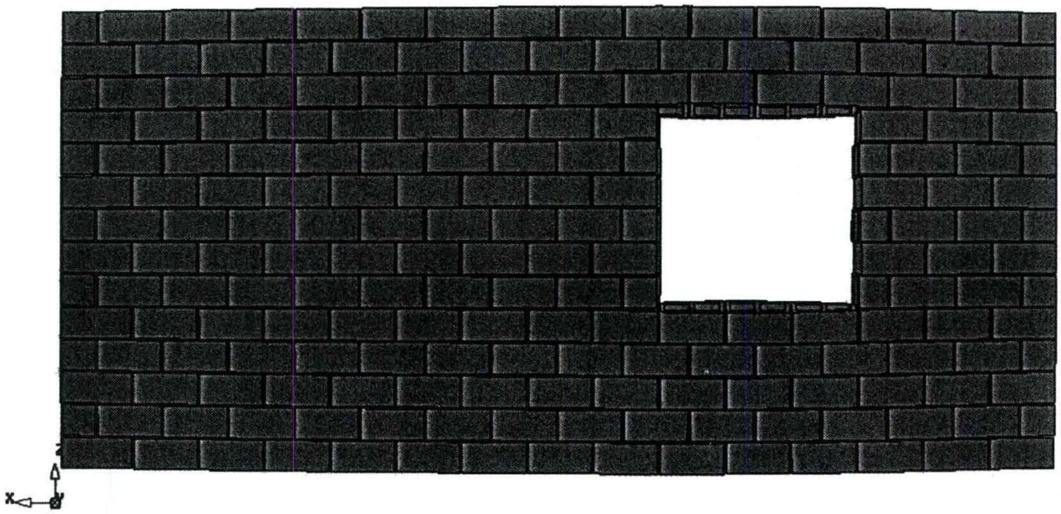
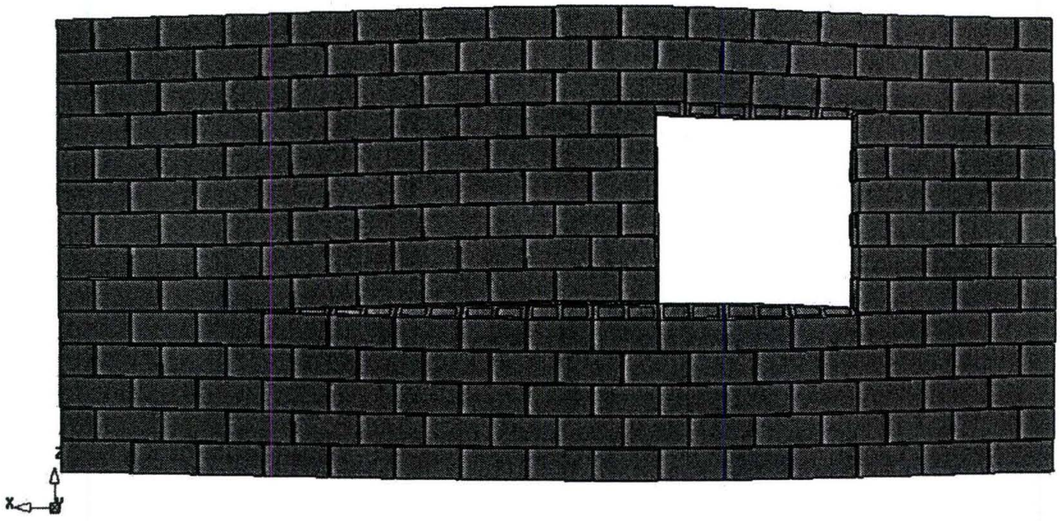


Figure 4.45 Wall BC7 load-displacement diagrams

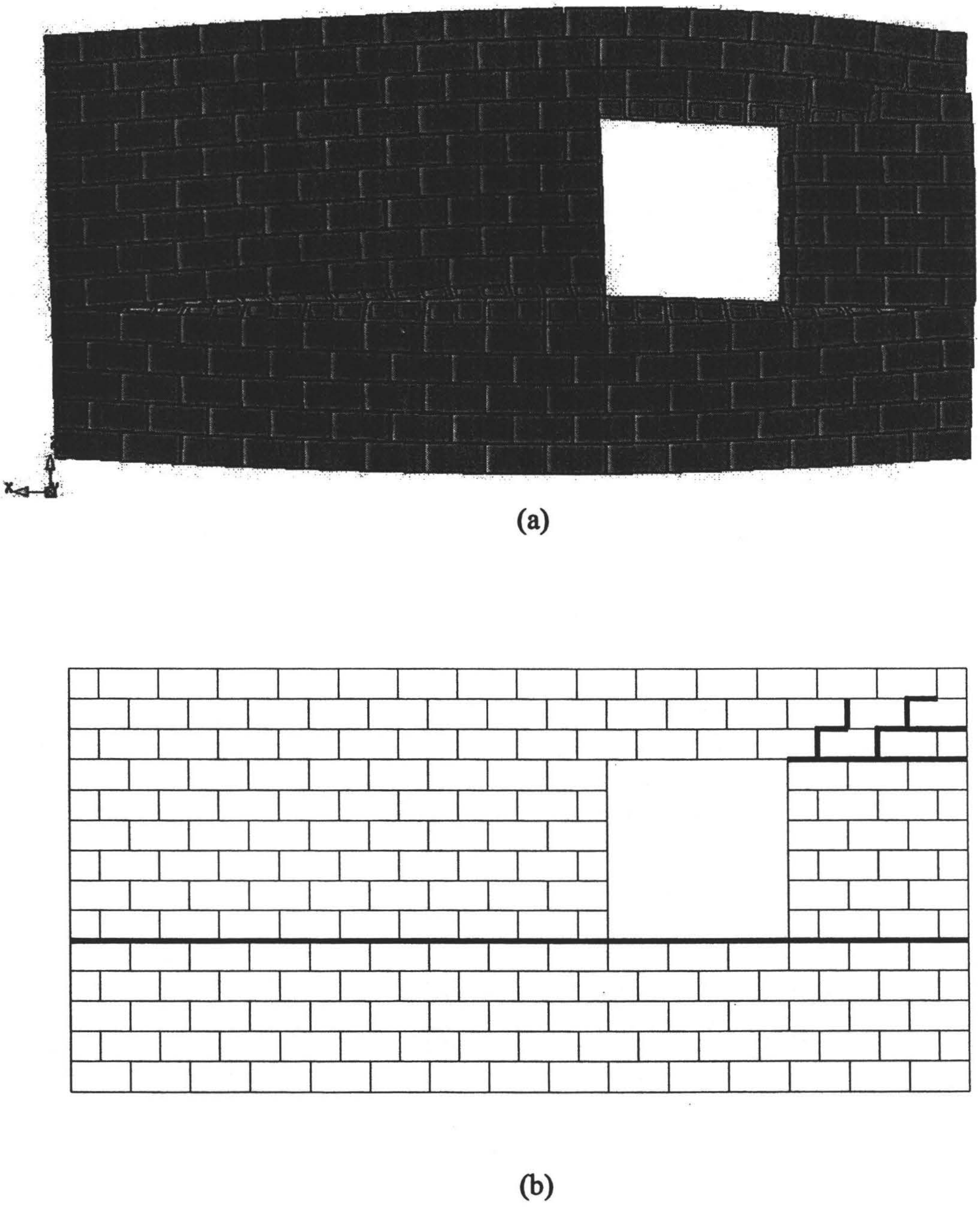


(a)



(b)

Figure 4.46 Crack development for wall BC7:
(a) the first mechanism initiates at the load of 1.00 kPa;
(b) the first mechanism is fully formed at the load of 1.99 kPa.



(a)

(b)

Figure 4.47 Results of analysis for wall BC7 at the load of 3.10 kPa:
(a) deformed mesh; (b) crack pattern.

4.5.9 Wall BC8

Wall BC8 has an opening with the same size as that of wall BC7 but located in the center of the wall. No bond beam was included. The experimental crack development and the corresponding loads are given in Figure 4.48. At the loading of 1.23 kPa, the first crack started at the bed joint located two courses below the top right corner of the opening, and developed towards the right support of the wall. As the load increased, the second horizontal crack occurred at the bottom left corner of the opening; then other cracks were formed near the bottom right corner and at the top corners of the opening, one after the other. With further loading, diagonal cracks arose from the four corners of the opening and extended to the wall corners. The loading corresponding to the formation of this mechanism was 3.57 kPa.

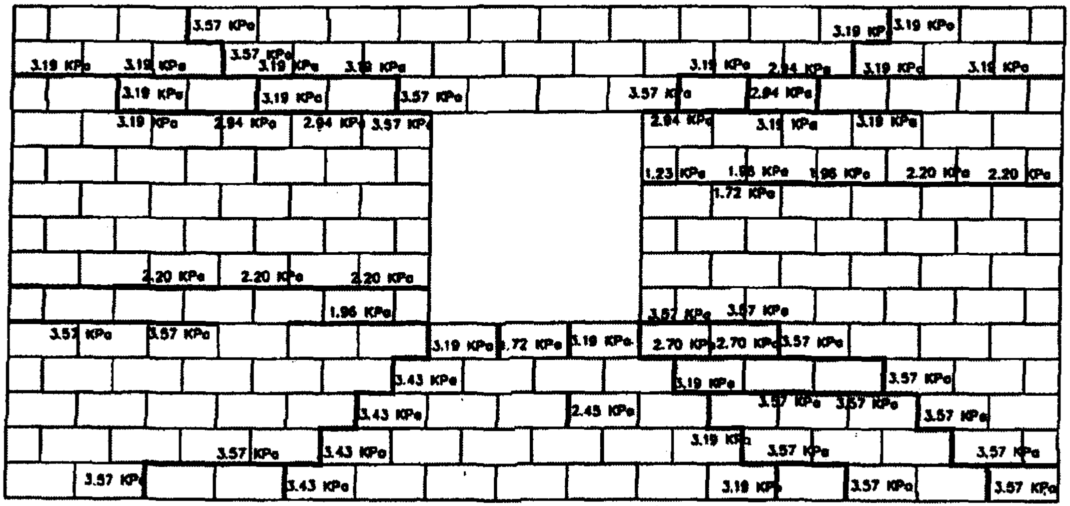


Figure 4.48 Crack pattern for wall BC8, Chen (2002)

The analysis of wall BC8 was carried out up to the loading of 2.68 kPa which is greater than the load required to full form the first mechanism (horizontal cracks). The analytical model depicts well the general behaviour of the wall. The numerical load-displacement diagrams along with the curves obtained experimentally are shown in Figure 4.49. Starting with the linear elastic behaviour, the load-displacement curves go through two nonlinear hardening-softening processes and enter another hardening stage before the analysis is stopped. This indicates two continuous horizontal cracks are developed before the formation of diagonal cracks, which is similar to the experimental observation. Stiffer response is again the simulated response of the wall. Again, one notes that the experimental data for the symmetric point B and C are significantly different. The reasons are believed to be the same as noted for the other wall test results. The measured displacement at point A was the smallest and that of point D was the largest for the whole loading history. However, the analysis shows that the displacement at point A becomes the largest after the second horizontal crack is fully formed.

For this wall, the predicted load of 1.00 kPa (point a) corresponding to the first tensile failure is within 20% of the measured cracking load. This indicates that shortly thereafter the first crack forming in the interface will become visible. The actual formation of the first horizontal crack occurs at the load of 1.96 kPa (point b) and fully developed when the load decreases to 1.78 kPa (point c). This crack initiates from the two bottom corners simultaneously and propagates to the

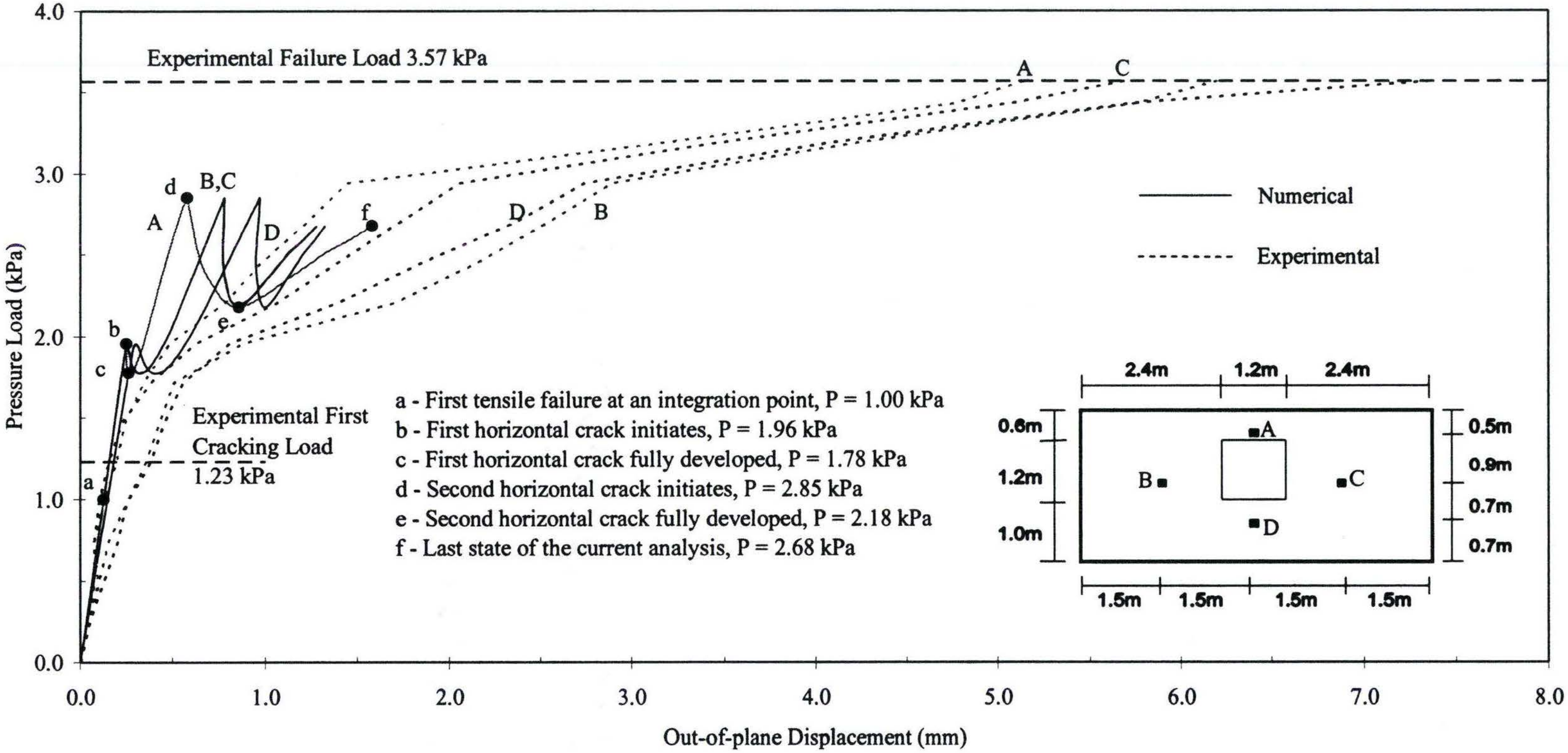
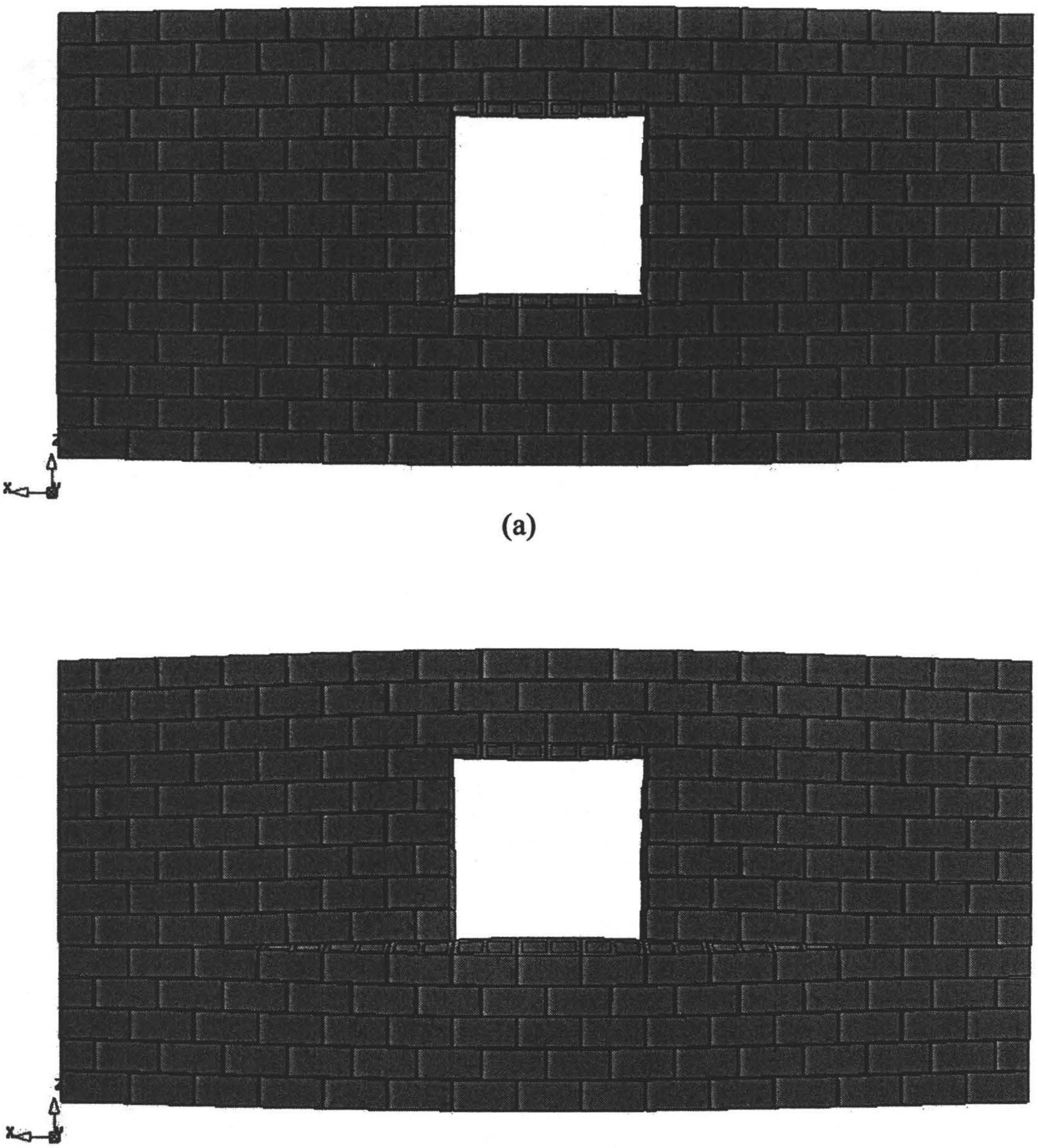


Figure 4.49 Wall BC8 load-displacement diagrams

two vertical supports. The first shear yielding occurs at the top corners of the opening at the load of 1.83 kPa, i.e., after the first horizontal crack is fully formed. Later, when the loading again increases to 2.16 kPa, the composite tension and shear yield surface is formed at the same location. The second horizontal crack originates at the top of the opening when the load reaches 2.85 kPa (point d) and is fully developed while the load decreases to 2.18 kPa (point e).

Figures 4.50 to 4.52 show the predicted crack developments. It is also observed that before the third peak point is reached, the diagonal cracks originates at the top corners of the opening when the load reaches 2.68 kPa, which corresponds to the last point in the load-displacement curves.



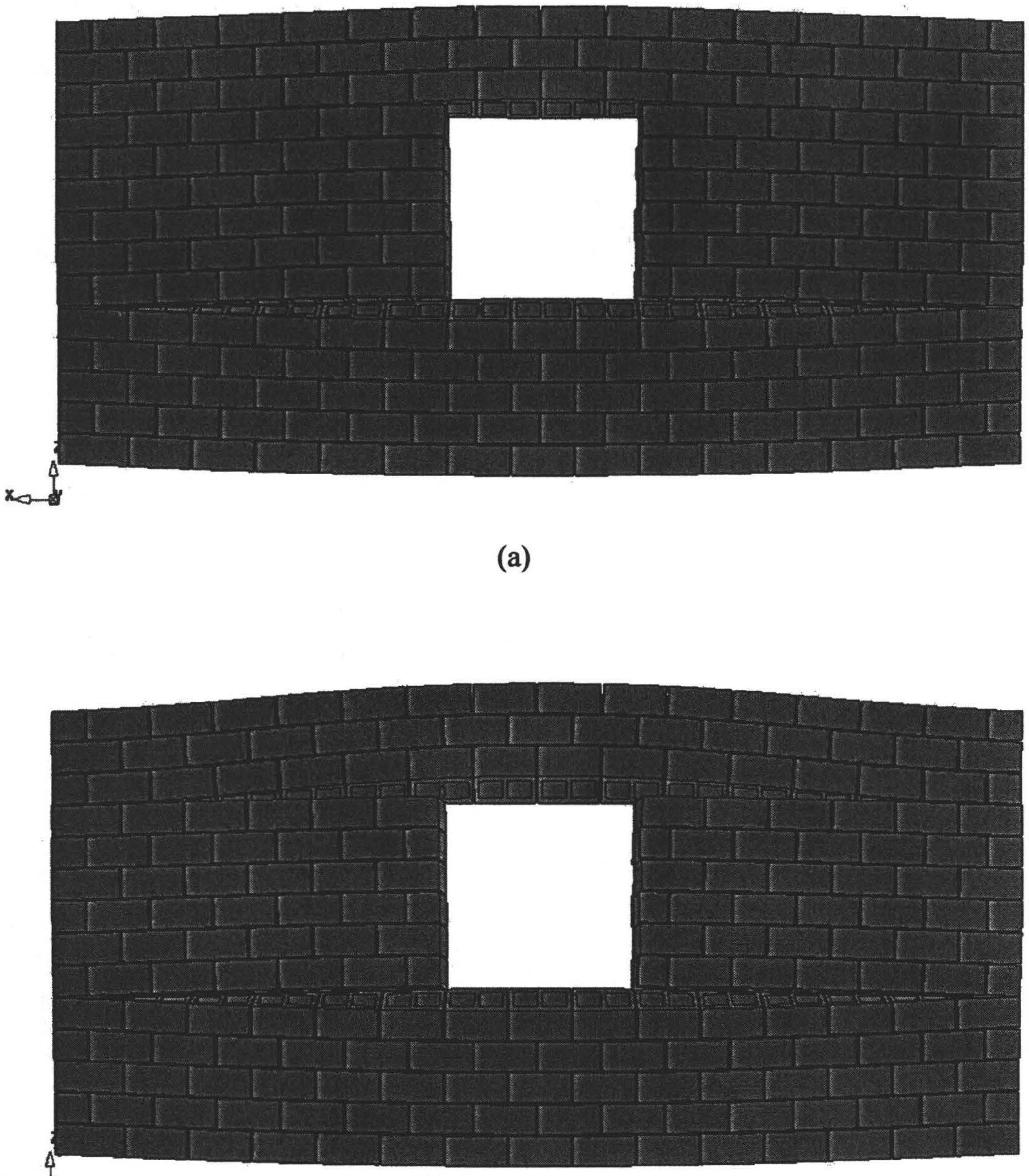
(a)

(b)

Figure 4.50 Crack development for wall BC8:

(a) the first horizontal crack initiates at the load of 1.96 kPa;

(b) the first horizontal crack is fully developed at the load of 1.78 kPa.



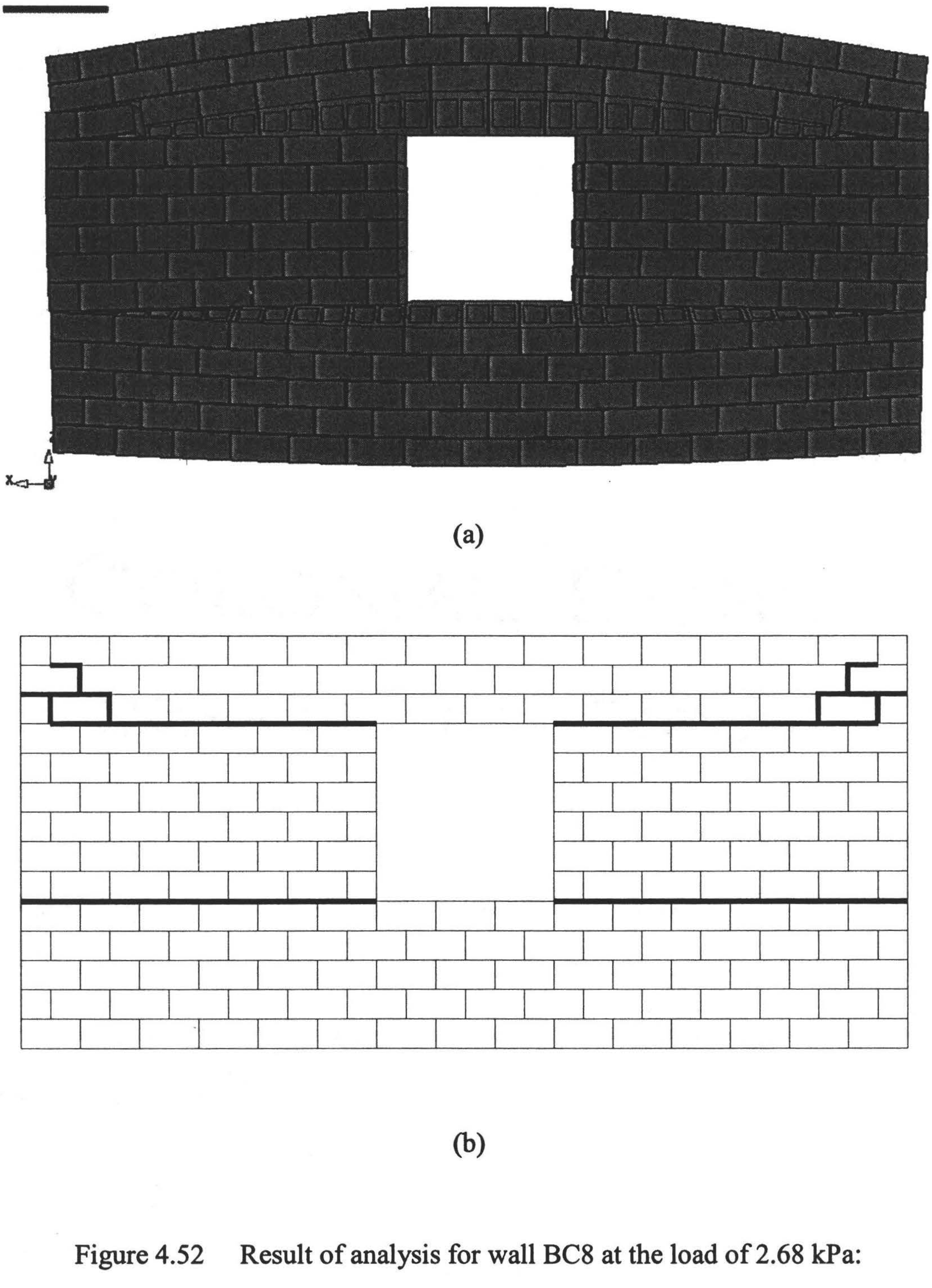
(a)

(b)

Figure 4.51 Crack development for wall BC8:

(a) the second horizontal crack initiates at the load of 2.85 kPa;

(b) the second horizontal crack is fully developed at the load of 2.18 kPa.



(a)

(b)

Figure 4.52 Result of analysis for wall BC8 at the load of 2.68 kPa:
(a) deformed mesh; (b) crack pattern.

4.6 Summary of the Analytical Results

As observed experimentally and numerically, masonry walls experience different failure mechanisms depending on the support conditions, size, location and number of openings, and on the magnitude of the load. It is also noted that the first mechanism represents the serviceability limit state, implying a visible crack has formed however the stability of the wall has not been undermined. The initiation of the second failure mechanism, which is identified for most of the walls as the onset of diagonal cracks, represents the ultimate limit state for the wall. Therefore to compile a complete summary, the load level at which the first and second failure mechanisms occur are reproduced in Table 4.5. For comparison, the experimental data and the analytical data of Essawy (1986) and Chen (2002) are also presented.

Essawy's analytical model was described earlier. In modeling the structural response of the walls, Chen (2002) constructed a 3-D homogeneous model which incorporates the actual unit geometry but did not discriminate between the block units and mortar joint. Orthotropic properties were considered for the homogenized material. The model was used to predict the onset of the first cracking load. He assumed that there is no interaction between the tensile strengths in the principal orthogonal directions for biaxial tension cases. To predict the ultimate strength, Chen employed both the yield line and the failure line approaches.

Table 4.5 Comparisons of analytical and experimental results

Wall	Predicted first cracking load (kPa)				Experimental first cracking load (kPa)	Predicted failure load (kPa)			Experimental failure load (kPa)
	Current model		Essawy's model	Chen's model		Current model	Essawy's model	Yield line/ Failure line method	
	1st tensile failure at integration point(s)	Initiation of the 1st mechanism				Initiation of the 2nd mechanism			
WIII	2.40	2.79	3.27		2.35	3.90	4.59		4.77
WF	1.20	2.38	1.55			2.38	3.79		3.90
BC1	2.40	2.79		1.70	2.61	3.90		3.55/3.10	4.06
BC2	1.20	1.94		1.55	1.73	2.58+		2.98/2.55	3.65
BC3	1.00	1.58		1.33	1.35	2.93		2.34/2.34	3.18
BC4	1.06	2.09		1.45	1.96	2.58+		3.15/3.15	3.61
BC5	0.60	1.74		0.91	1.47	2.57		2.03/2.03	2.90
BC6	0.86	1.55		1.02	1.47	2.08+		2.33/1.92	2.79
BC7	1.00	1.99		1.07	1.72	3.10+		2.95/2.56	3.22
BC8	1.00	1.96(2.85*)		1.05	1.23	2.68+		2.85/2.85	3.57

* Data for the second horizontal crack.

+ Analysis was stopped not because of observed failure mechanism but because of convergence problem. Although the load required to initiate the second mechanism is not known, it is greater than the one noted in the table.

It can be seen from Table 4.5, that the previously predicted cracking and failure loads are smaller than the experimental results except for the first cracking load of wall WIII predicted using Essawy's model. The predicted first cracking loads using the current model, representing the first tensile failure at an integration point, are mostly more conservative than those obtained by Essawy and Chen's models. This prediction represents the initiation of the micro-cracks that are not yet connected. Subsequently, the current model provides another critical value, namely the first peak load or the load corresponding to the initiation of an unstable crack. Without any increase in the load, the first mechanism will develop forming a continuous and visible crack. It can be noted that, the experimental first cracking load for all the walls lie between the load of first tensile failure and the initiation of the first mechanism. It is important to note that the first cracking loads recorded in experiments were based on the first visible cracks, which is expected to be greater than the first prediction and smaller than the second prediction. Therefore, the current model provides a very good prediction of the range of load when the first mechanism is expected to occur.

Discrete modeling of full-sized walls demands considerable effort and analysis time. As a result some of the walls are not completely analyzed in the current attempt. The post-collapse behaviour was only obtained for walls WIII, WF, BC1, BC3 and BC5. Wall BC7 is believed to be close to the collapse (peak load) since the second mechanism is clearly formed. With the exception of the results for wall WF where the current model predicted a lower failure load, for all other walls mentioned above good agreements can be found between the predicted results with the current model and the experimental results. The difference ranges from 4% to 18 %. For wall BC2, BC4, BC6 and BC8, even the failure loads have not been reached, the differences between the last loading of the current analysis and the experimental data are within 30%. It is believed that reasonable agreements will be achieved if the analyses were to continue. For those walls with relatively complete analysis, comparison between the prediction using current model and the yield line and failure line methods shows that both the yield line and failure line approaches underestimate the failure load. The above findings suggest that the proposed discrete finite element model yields safe and more accurate results.

CHAPTER 5

SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

5.1 Summary

Literature review of masonry walls subjected to out-of-plane loading revealed that there are no rational analysis and design methods that have been identified for this type of structure, especially for the hollow concrete block masonry with openings. Among many analysis tools, finite element method is acknowledged to be a reliable approach capable of capturing the structural response of masonry structure under loadings. Two common modeling techniques are used in the masonry communities, discrete modeling and composite modeling. The former is carried out by discretizing the individual brick/block units and mortar joints producing a heterogeneous model; and the latter involves the modeling of the whole structure as a homogeneous model with equivalent anisotropic properties. The available literature showed that little quantified analyses about the effects of different modeling techniques on the structural response of block masonry have been attempted. For laterally loaded hollow block masonry, the complex geometric configuration can produce significant effects on the predicted behaviour; therefore the current study was initiated to investigate the appropriateness of the finite element modeling technique with the aim to identify the representative model for studying the flexural behaviour of hollow block masonry.

The effects of modeling techniques on the structural response were evaluated via linear elastic analysis for three simple wall geometries: horizontal spanning strip, vertical spanning strip and two-way spanning panel. The standard 190 mm concrete block unit and 10-mm thick mortar joint were used and modeled with three-dimensional continuum elements. For each geometric configuration, two different modeling techniques along with the options of using actual hollow units and equivalent solid units as well as construction patterns were considered. This resulted in four different models, homogeneous solid model, homogeneous detailed model, heterogeneous stack pattern model and heterogeneous running bond model. In addition, as the wall span plays an important role in the flexural behaviour, four span/thickness ratios for the two wall strips and two aspect ratios for the two-way bending panel were investigated. Further studies on the effects of mortar property and mesh size provided additional evidence of the differences between several modeling approaches.

The comparative analyses showed that detailed heterogeneous model yielded more representative results, in comparison to the homogeneous solid model and homogeneous detailed model. However, to capture interface between units and mortar, which serves as a plane of weakness and the main cause for the inelastic behaviour, it is felt necessary to further refine the model by incorporating interface elements. Two possible modeling strategies were discussed. A detailed discrete modeling requires units, mortar joint and interface be modeled separately. Undoubtedly, such an approach makes the analysis of large structures almost unpractical due to computational inefficiency. The simplified strategy is thus considered, where block units were modeled by continuum elements and mortar joints were modeled by interface elements.

The constitutive model used to describe the inelastic behaviour of joint interface, also named as Composite Interface Model, was introduced based on the formulation developed by Lourenço (1996). This plasticity model includes three distinct failure mechanisms, namely, a straight tension cut-off for type I (tensile) failure, a Coulomb friction model for type II (shear) failure and an elliptical cap model for compressive failure. It is also known as Combined Cracking-Shearing-Crushing Model with the capability of reproducing all inelastic damage phenomena. The material model, implemented according to the theory of multi-surface plasticity in modern concepts which include multi-surface yield criteria, Euler backward return mappings and consistent tangent operators (tangential stiffness matrix), etc., was described concisely. The internal state parameters, which are related to the fracture energies associated with different failure mechanisms, were used to describe the internal damages corresponding to three failure modes respectively. In current study, however the compressive failure was not included as has not been implemented in DIANA for three-dimensional analysis.

A reliable numerical tool must consist of not only a well-defined material model but also a robust solution procedure. The finite element discretization leads to a set of nonlinear equilibrium equations. The updates of the state variables, as the unknowns of these equations, are to be solved with a regular Newton-Raphson method through incremental-iterative procedures. In order to capture the snap-through and snap-back behaviour, arc-length control with the Update Normal Plane method is employed to constraint the incremental displacement. Line search technique is also considered to improve the convergence of iteration. Energy based adaptive loading was used to adjust the loading increment to enhance the efficiency of arc-length method. A method for distinguishing the sign of the load factor so as to determine the loading and unloading for a particular increment was also introduced.

The proposed discrete finite element model for concrete block masonry was evaluated against experimental results available in the literature. A mesh generator program BMFEM, detailed in Appendix A, which was developed specifically for the current study, was used to create the nonlinear models for laterally loaded hollow block walls with or without openings. The commercially available finite element package, DIANA™ (2003), was used for analysis because the Composite Interface Model is implemented in the program. The predicted results were compared to the test results in terms of load-displacement diagrams and crack development patterns, for characterizing the behaviour of a masonry structure. The effects of mesh size upon the predicted structural response were also investigated.

5.2 Conclusions

The current study focuses on the investigation of an appropriate three-dimensional finite element model capable of simulating the structural behaviour of unreinforced concrete block masonry under out-of-plane loading. Specific conclusions concerning modeling techniques, geometry and material properties of masonry constituents, discrete modeling approach and effects of mesh discretization have been presented at the end of Chapters 2 and 4. To provide an overview of the findings in this research work, the general conclusions are presented as follows:

1. Geometric representation of hollow concrete masonry plays an important role in modeling the structural response of flexural concrete block walls. Conventional approach, such as using equivalent solid units, does lead to erroneous prediction of the structural response. Results have revealed that when using equivalent solid unit, this approach yields stiffer responses in displacements and considerable lower stresses, particularly in the walls with lower span/thickness or aspect ratios.
2. The homogeneous detailed model generally predicts lower displacements and stresses than the stack pattern and running bond heterogeneous models except for very low span/thickness ratios in horizontal strip walls. The differences in the predicted results become more apparent as the span increases.
3. The heterogeneous stack pattern model and running bond model yield closer predictions of both displacements and stresses for vertically spanning walls

and two-way spanning walls but not for horizontally spanning walls. It shows that the structural response is sensitive to construction patterns for horizontally spanning wall, where the heterogeneous running bond model appears to be softer with respect to displacement and produces less stresses in comparison with stack pattern model.

4. Effect of shear deformation is the important factor leading to the discrepancy in the results predicted using different modeling approaches. The displacements due to shear deformation are exclusively underestimated in the equivalent solid models for three geometric configurations. Shear effect is less significant in vertical spanning walls than that in horizontal spanning walls. It has been concluded that with the presence of shear deformation, plane section assumption is not valid for short spans. Moreover, as the span/thickness ratio increases the non-planar behaviour is reduced.
5. Mortar's property has significant effect on the predicted displacement for all the models. Stiffer mortar, i.e., higher elastic modulus, yields smaller displacements. This difference tends to increase when comparing the displacements predicted in the equivalent solid model and hollow models for horizontal spanning strips and vertical spanning walls.
6. According to the results of linear elastic analysis, the predictions obtained by using all the models for vertically spanning walls and two-way spanning walls with large aspect ratios are not sensitive to the mesh size. In contrast, refined mesh results produce notable increases in the difference between the

predicted displacements using solid models and hollow models for horizontal spanning strips with lower span/thickness ratios.

7. The adopted discrete modeling strategy with Composite Interface Model incorporated has proven to be adequate for tracing the entire loading path ranging from the elastic regime, pre- and post-crack inelastic regime, pre- and post-collapse inelastic regime until total degradation of strength. The two failure mechanisms, tension and shear, were observed for all the analyses. In general, the predicted failure patterns are found to agree reasonably well with experimental observations. This demonstrates the stability and suitability of the proposed model for predicting the structural behaviour of concrete block walls subjected to out-of-plane loading.
8. Proposed discrete modeling approach is found to underestimate the first cracking load and failure load. The first cracking load is generally lower than those obtained by Essawy (1996) and Chen (2002). This load provides the range for the serviceability limit state. The fact that the experimental first cracking load for all the walls lies between the initiation of the crack and the initiation of the first mechanism indicates a good prediction for the range of the first cracking load. On the other hand, the proposed model yields a lower failure load in comparison with yield line or failure line approach, but an adequately safe prediction when compared to the experimental measurement.
9. Mortar shrinkage especially in the head joint produces non-uniform bond strength between units and mortar. This phenomenon has produced more

cracks in the tested wall panels in comparison to the model prediction. However, these cracks do not seem to affect the failure pattern. The results have shown that once a crack initiates at one location its propagation is controlled by the state of stress and the material properties. As a result, the model predictions are found to concur with the observed ones.

10. The predicted responses are generally stiffer than the experimental observations. The difference has been attributed to combined effects of the variation in mortar's material properties, shrinkage in the mortar joints, and plastic deformations in the compressive zone, that the mathematical model does not account for. The analyses also show that the second mechanism starts at the peak load and is fully formed at post-peak regime with load value less than the peak load. The observed experimental results show the load corresponding to the full formation of the second mechanism is usually greater than the load at which the second mechanism starts. The post-peak behaviour was not captured in the experiments because the test was carried out under load control conditions, a test procedure that is not capable to observe softening effects observed in the predicted results.
11. The Composite Interface Model used in three-dimensional discrete modeling is not sensitive to the mesh size. This suggests that the discretization with a single element in the web thickness and face-shell thickness and, two elements in the web length, face-shell length and block height can generate adequate accuracy of the predicted results.

12. The finite element analysis, which employs the discrete modeling approach and the Composite Interface Model, is found to be an effective design tool for predicting the structural response of concrete masonry walls with openings under out-of-plane loadings. This method is particularly suitable for analysis of structural components such as walls.

5.3 Recommendations

This research has provided insight into the capabilities of numerical modeling of concrete block masonry. Although, the proposed model was shown to be a useful tool for analyzing concrete block masonry subjected to out-of-plane loads, it has also revealed limitations of the proposed model. Accordingly, the following issues are raised to be addressed in future research projects:

1. Our research has revealed that the material properties needed for the Composite Interface Model is very limited, in particular the data for fracture energy. It is recommended that additional research be carried out to generate more data for different mortar mix, and to determine the variability in the measured data.
2. Generation of cracks in the middle of the unit needs to be added. The current model permits the occurrence of damage only in the joint. This limitation has provided restrictions in the development of cracks and may have contributed to a stiffer response. For some cases, such as the wall supported on three sides, this limitation has led to erroneous crack pattern.

3. Current Composite Interface Model needs to be expanded to include the crushing model for 3-D analysis. This limitation is believed to have contributed to a stiffer response.
4. The proposed discrete modeling method could be used to develop design tables for concrete block walls with different aspect ratios, opening locations and support conditions. These tables, when integrated in the Canadian masonry code, will provide a comprehensive tool for the analysis and design of masonry concrete blocks subjected to out-of-plane loading.

REFERENCES

AFEMS^{TM}, Finite Element Analysis Program (V7.5), FEM Engineering Corporation, California, USA, 2003.

Anderson, C. and Bright, N. J., "Behaviour of Non-Load Bearing Block Walls Under Wind Loading", Concrete, 110(9), pp. 27-30, 1976.

A.S.T.M., "Standard Specifications for Loadbearing Concrete Masonry Units", ASTM C90-96a, ASTM, West Conshohocken, 1996.

Baker, L. R., "Lateral Loading of Masonry Panels", Structural Design of Masonry (Clay and Concrete), 1980.

Baker, L. R., "The Flexural Action of Masonry Structure Under Lateral Load", Ph.D. Thesis, Deakin University, 1981.

Baker, L. R., Gairns, D. A., Lawrence, S. J. and Scrivener, J. C., "Flexural Behaviour of Masonry Panels – A State of the Art", Proc. of the 7th Intern. Brick/Block Masonry Conf., Melbourne, Australia, pp. 27-55, 1985.

Bathe, K. J. and Dvorkin, E. N., "On the Automatic Solution of Nonlinear Finite Element Equations", Computers & Structures, Vol. 17, pp. 871-879, 1983.

Bathe, K.-J., "Finite Element Procedures", Prentice-Hall, New Jersey, USA, 1996.

Bull, J. W., "Computational Modeling of Masonry, Brickwork, and Blockwork Structures", Saxe-Coburg Publications, Dun Eaglais, Station Brae, Kippen Stirling, UK, 2001.

Cajdert, A., "Laterally Loaded Masonry Walls", Division of Concrete Structures, (80:5), 1980.

Canadian Standard Assiciation, CSA S304.4-04, "Design of Masonry Structures", CSA, Mississauga, Ontario, 2004.

Chen, B., "Masonry Walls with Openings under Out-of-Plane Loading", M.A.Sc. Thesis, McMaster University, Hamilton, Canada, 2002

Chen, W. F. and Han, D. J., "Plasticity for Structural Engineers", Springler-Verlag, New York, New York, USA, 1988.

Chidiac, S. E., He, Z. and Drysdale, R. G., "Finite Element Model of Unreinforced Concrete Block Walls Subject to Out-of-Plane Loading – A Parametric Study", Proc. of the 13th Intern. Brick/Block Masonry Conf., Amsterdam, pp. 439-448, 2004.

Crisfield, M. A., "A Fast Incremental/Iterative Solution Procedure That Handles Snap-Through", Computers & Structures, Vol. 13, pp. 55-62, 1981.

Crisfield, M. A., "Non-linear Finite Element Analysis of Solids and Structures: Vol. 1, Essentials", John Wiley and Sons, 1991.

CUR, "Structural Masonry: A Experimental/Numerical Basis for Practical Design Rules", Report 171, CUR, Gouda, The Netherlands, 1994.

De Borst, R., "Non-linear Analysis of Frictional Mateials", Ph.D. Thesis, Delft University of Technology, Delft, The Netherlands, 1986.

DIANA™, Finite Element Analysis Program (Release 8.1), TNO DIANA BV, Delft, The Netherlands, 2003.

Drysdale, R. G., Hamid, A. A. and Baker, L. R., "Masonry Structures – Behaviour and Design", The Masonry Society, Boulder Colorado, USA, 1999.

Essawy, A. S., "Strength of Hollow Concrete Block Masonry Walls Subject to Lateral (Out-of-Plane) Loading", Ph.D. Thesis, McMaster University, Hamilton, Canada, 1986.

Essway, A. S. and Drysdale, R. G., "Capacity of Block Masonry Under Uniformly Distributed Loading Normal to the Surface of the Wall", Proc. of the 3rd Canadian Masonry Symposium, Edmonton, Canada, 39-1-39-16, 1983.

Feenstra, P. H., "Computational Aspects of Biaxial Stress in Plain and Reinforced Concrete", Ph.D. Thesis, Delft University of Technology, Delft, The Netherlands, 1993.

Fried, A., Anderson, C. and Smith, D., "Predicting the Transverse Lateral Strength of Masonry Walls", Proc. of the 8th Intern. Brick/Bloack Masonry Conf., Dublin, Republic of Ireland, pp. 1171-1183, 1988.

Gazzola, E. A., “Macroscopic and Microscopic Failure Criteria for Concrete Block Masonry Subject to General Biaxial Bending”, M.Eng. Thesis, McMaster University, Hamilton, Canada, 1986.

Gere, J. M. and Timoshenko, S. P., “Mechanics of Materials”, Wadsworth Inc., 1984.

Ghosh, A. K., Made, A. M. and Colville, J., “Finite Element Modeling of Unreinforced Masonry”, Proc. of the 10th Intern. Brick/Block Masonry Conf., Calgary, pp. 61-69, 1994.

Guo, P., “Investigation and Modeling of the Mechanical Properties of Masonry”, Ph.D. Thesis, McMaster University, Hamilton, Canada, 1991.

Hamid, A. A., “Behaviour Characteristics of Concrete Masonry”, Ph.D. Thesis, McMaster University, Hamiton, Canada, 1978.

Hamid, A. A. and Drysdale, R. G., “Flexural Tensile Strength of Concrete Block Masonry”, Journal of Structural Engineering, Vol. 114, No. 1, pp. 50-66, 1988.

Hamid, A. A., Ziab, G. and Nawawy, O. E., “Modulus of Elasticity of Concrete Block Masonry”, Proc. of the 4th North American Masonry Conf., 1988.

Haseltine, B. A., West, H. W. H. and Tutt, J. N., “The Resistance of Brickwork to Lateral Loading: Part 2 – Desing of Walls to Resist Lateral Loads”, The Structural Engineer, Vol. 55, No. 10, pp. 422-430, 1977.

Hendry, A. W., “The Lateral Strength of Unreinforced Brickwork”, The Structural Engineer, Vol. 2, No. 51, pp. 43-50, 1973.

Hendry, A. W. and Kheir, A. M. A., “The Lateral Strength of Certain Brickwork Panels”, Proc. of the 4th Intern. Brick Masonry Conference, Brugge, 1976.

Johansen, K. W., “Yield Line Formulae for Slabs”, Cement and Concrete Association, London, 1972.

Kemp, K. Q., “The Yield Criterion for Orthotropically Reinforced Concrete Slabs”, Int. Jnl. Mech. Sci., 7, pp. 737-746, 1965.

Koiter, W. T., “Stress-strain Relations, Uniqueness and Variational Theorems for Elastic-plastic Materials with A Singular Yield Surface”, Q. Appl. Math., 11(3), pp. 350-354, 1953.

Lawrence, S. J., "Design of Masonry Panels for Laterals Loading – Some Interim Recommendations", Rep. No. Technical Record 460, Exp. Building Station, Chatswood, N.S.W., Australia, 1980.

Lawrence, S. J., "Behaviour of Brick Masonry Walls Under Lateral Loading", Ph.D. Thesis, University of New South Wales, 1983.

Lawrence, S. J. and Cao, H. T., "Cracking of Non-loadbearing Masonry Walls under Lateral Forces", Proc. of the 8th Intern. Brick/Block Masonry Conf., Dublin, Republic of Ireland, pp. 1184-1194, 1988.

Lee, J. S., Pande, G. H., Middleton, J. and Kralj, B., "Analysis of Tensile Strength of Masonry", Proc. of the 10th Intern. Brick/Block Masonry Conf., Calgary, pp. 21-29, 1994.

Lourenço, P. B., Palacio, K. and Prieto, F., "Implementation of A Constitutive Model for Masonry Shells as a Stand-alone Subroutine", Report 02-DEC/E-13, University of Minho, Guimarães, Protugal, Nov. 2002.

Lourenço, P. B. and ROTS, J. G., "A Multi-surface Interface Model for the Analysis of Masonry Structures", Journal of Structure Engineer, ASCE, Vol. 123, No. 7, pp. 660-668, 1997.

Lourenço, P. B., "Analysis of Masonry Structures with Interface Elements: Theory and Applications", Report 03-21-22-0-01, Delft University of Technology, Delft, The Netherlands, 1994.

Lourenço, P. B., "Computational Strategies for Masonry Structures", Ph.D. Thesis, Delft University of Technology, Delft, The Netherlands, 1996.

Lu, M., Schulz, A. E. and Stolarski, H. K., "Analysis of the Influence of Tensile Strength on the Stability of Eccentrically Compressed Slender Unreinforced Masonry Walls Under Lateral Loads", Journal of Structural Engineering, Vol. 130, No. 6, pp.921-933, 2004.

May, I. M. and Ma, S. Y. A., "The Analysis and Design of Masonry Panels with Openings", Proc. of the 8th Int. Brick/Block Masonry Conf., pp. 1467-1475, 1988.

Middleton, A. C. and Drysdale, R. G., "Flexural Capacity on Concrete Block Walls with Openings", Proc. of the 7th Canadian Masonry Symposium, pp. 537-546, 1995.

MSJC, ACI, ASCE and TMS, "Building Code Requirements for Masonry Structures", ACI 530/ASCE 5/TMS 402-99, 1999.

Page, A. W., "Finite Element Model for Masonry", Journal of Structural Division, 104(ST8), pp. 1267-1285, 1978.

Pande, G. N., Kralj, B. and Middleton, J., "Analysis of the Compressive Strength of Masonry Given by the Equation $f_k = K(f'^b)^\alpha (f_m)^\beta$", The Structural Engineer, 71, pp. 7-12, 1994a.

Pande, G. N., Kralj, B. and Middleton, J., "Numerical Simulation of Cracking and Collapse of Masonry Panels Subjected to Lateral Loading", Proc. of the 10th Int. Brick/Block Masonry Conf., Calgary, pp. 107-115, 1994b.

Pande, G. N., Liang, J. X. and Middleton, J., "Equivalent Elastic Moduli for Brick Masonry", Computers and Geotechnics, 8, pp. 243-265, 1989.

Pramono, E. and Willam, K., "Implicit Integration of Composite Yield Surfaces with Corners", Engrg. Comput., 6, pp. 186-197, 1989.

Ramm, E., "Strategies for Tracing Nonlineae Response Near Limit Points", Nonlinear Finite Element Analysis in Structural Mechanics (W. Wunderlich, E. Stein, and K. J. Bathe, eds.), pp. 63-89, Springer-Verlag, New York, 1981.

Riggs, H. R. and Powell, G. H., "Tangent Constitutive Matrices for Inelastic Finite Element Analysis", Int. J. Numer. Methods Engrg., 29, pp. 1193-1203, 1990.

Riks, E., "An Incremental Approach to the Solution of Snapping and Buckling Problems", International Journal of Solid and Structures, Vol. 15, pp. 529-551, 1979.

Rots, J. G., "Computational Modeling of Concrete Fracture", Ph.D. Thesis, Delft University of Technology, Delft, The Netherlands, 1988.

Saliba, J. E., Al-Akkad, R. S. and Sawaya, G. E., "Use of Anisotropic Behaviour in Masonry", Proc. of the 6th Canadian Masonry Symposium, Saskatoon, Saskatchewan, pp. 687-694, 1992.

Schellekens, J. C. J., "Computational Strategies for Composite Structures", Ph.D. Thesis, Delft University of Technology, Delft, The Netherlands, 1992.

Seward, D. W., "A Developed Elastic Analysis of Lightly Loaded Brickwork Walls with Lateral Loading", International Journal of Masonry Construction, Vol. 2, No. 3, pp. 129-134, 1982.

Simo, J. C. and Taylor, R. L., "Consistent Tangent Operators for Rate-independent Elastoplasticity", Comp. Meth. Appl. Mech. Engrg., 48, pp. 101-118, 1985.

Simo, J. C., Kennedy, J. G. and Govindjee, S., "Non-smooth Multisurface Plasticity and Viscoplasticity. Loading/unloading Conditions and Numerical Algorithms", Int. J. Numer. Methods Engrg., 26, pp. 2161-2185, 1988.

Sinha, B. P., "A Simplified Ultimate Load Analysis of Laterally Loaded Model Orthotropic Brickwork Panels of Low Tensile Strength", The Structural Engineer, 4(56B), pp. 81-84, 1978.

Sinha, B. P., "An Ultimate Load Analysis of Laterally Loaded Brickwork Panels", International Journal of Masonry Construction, Vol. 1, No. 2, pp. 57-60, 1980.

Sinha, B. P. and Ng, C. L., "Behaviour of Brickwork Panels Under Lateral Pressure", Proc. of the 10th Intern. Brick/Block Masonry Conf., Calgary, Canada, pp. 649-658, 1994.

Timoshenko, S. P. and Woinowsky-Krieger, S., "Theory of Plates and Shells", McGraw-Hill, 1959.

TNO DIANA BV, "DIANA Finite Element Analysis User's Manual (Release 8.1)", Edited by Frits C. de Witte and Gerd-Jan Schreppers, Delft, The Netherlands, 2003.

Tzamtzis, A. D. and Asteris, P. G., "Finite Element Analysis of Masonry Structures: Part 1 – Review of Precious Work", Proc. of the 9th North American Masonry Conf., Clemson, South Carolina, pp. 101-111, 2003a.

Tzamtzis, A. D. and Asteris, P. G., "Finite Element Analysis of Masonry Structures: Part 2 – Proposed 3-D Nonlinear Microscopic Model", Proc. of the 9th North American Masonry Conf., Clemson, South Carolina, pp. 146-155, 2003b.

Van der Pluijm, R., "Material Properties of Masonry and Its Components under Tension and Shear", Proc. of the 6th Canadian Masonry Symposium, Saskatoon, Saskatchewan, Canada, pp. 675-686, 1992.

Van der Pluijm, R., "Shear Behaviour of Bed Joints", Proc. of the 6th North American Masonry Conf., Philadelphia, Pennsylvania, USA, pp. 125-136, 1993.

Van Zijl, G. P. A. G., "Computational Modelling of Masonry Creep and Shrinkage", Ph.D. Thesis, Delft University of Technology, Delft, The Netherlands, 2000.

West, H. W. H., Hodgkinson, H. R. and Haseltine, B. A., "The Resistance of Brickwork to Lateral Loading – Part 1", The Structural Engineer, 55(10), pp. 411-421, 1977.

Zienkiewicz, O. C. and Taylor, R. L., "The Finite Element Method (5th ed.): Volume 1, The Basis", Butterworth-Heinemann, Oxford, UK, 2000.

APPENDIX A

BLOCK MASONRY MESH GENERATOR PROGRAM - BMFEM

A.1 Introduction

Many general-purpose finite element packages have been extensively used to analyze masonry structures. However, the lack of efficiency in creating the block masonry model using the available software usually results in considerable input effort. Even though some other tools such as CAD can be used to generate the input data file for the model, the tedious input work is not only time-consuming but also likely to cause errors. To this point, it is necessary to develop a generic finite element mesh generator for block masonry prior to the application of a specific finite element analysis program. The advantages of the mesh generator become more obvious when the comparisons of block masonry structures with different sizes, openings locations and boundary conditions are required in the research.

A block masonry mesh generator program BMFEM has been developed to generate homogeneous and heterogeneous model for unreinforced single-wythe masonry walls or beams constructed using standard block units. The capability of modeling multiple openings with arbitrary dimensions and locations has also been incorporated in the program. Currently, the loading and boundary conditions are mainly considered for the structures under out-of-plane loading. Nonetheless, since the information of global nodal coordinates, elements and constraint points are all available in the generated input data file, it is convenient to modify the data to create desired models with other common loadings and boundary conditions.

A.2 Modular Structure of BMFEM

BMFEM is developed using MATLAB® V6.5. The main program BMFEM.m is a script file consisting of many subroutines, each of which has a filename with the extension .m and also contain its own subroutines. These subroutines work as modules to perform individual tasks during the construction of the model. The organization of BMFEM routines can be shown in Figures A.1 to A.5.

The definitions of the subroutines are list as follows:

adjustGE	Adjusts grids (nodes) and elements on the opening
bbfblkGE	Defines grids and elements in a full block in a bond beam
bbhblkGE	Defines grids and elements in a half block in a bond beam
blayerElem	Defines elements in the block layer in an odd course
blayerElem2	Defines elements in the block layer in an even course
blayerGrid	Defines grids in the block layer in an odd course
blayerGrid2	Defines grids in the block layer in an even course

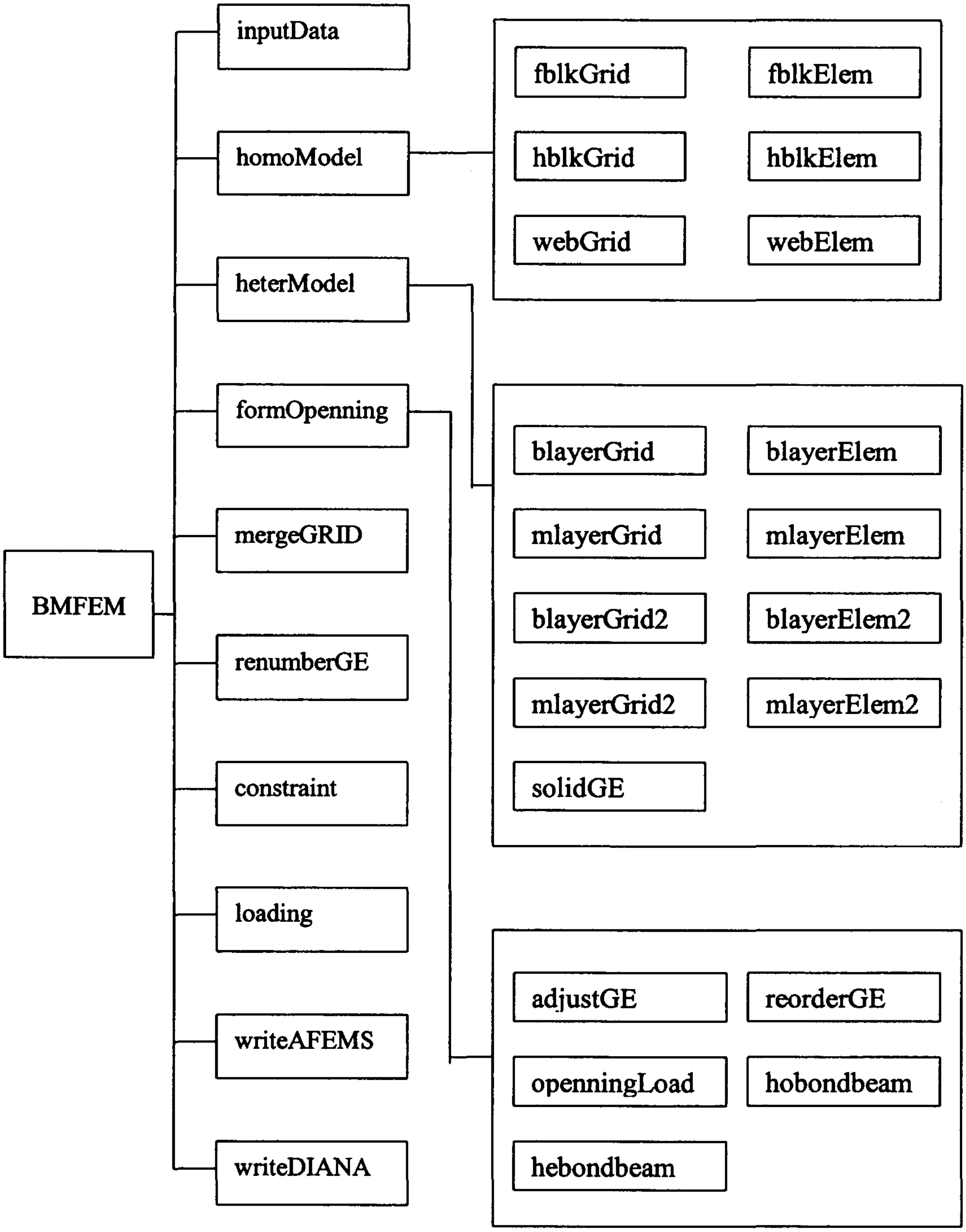


Figure A.1 BMFEM Modular structure

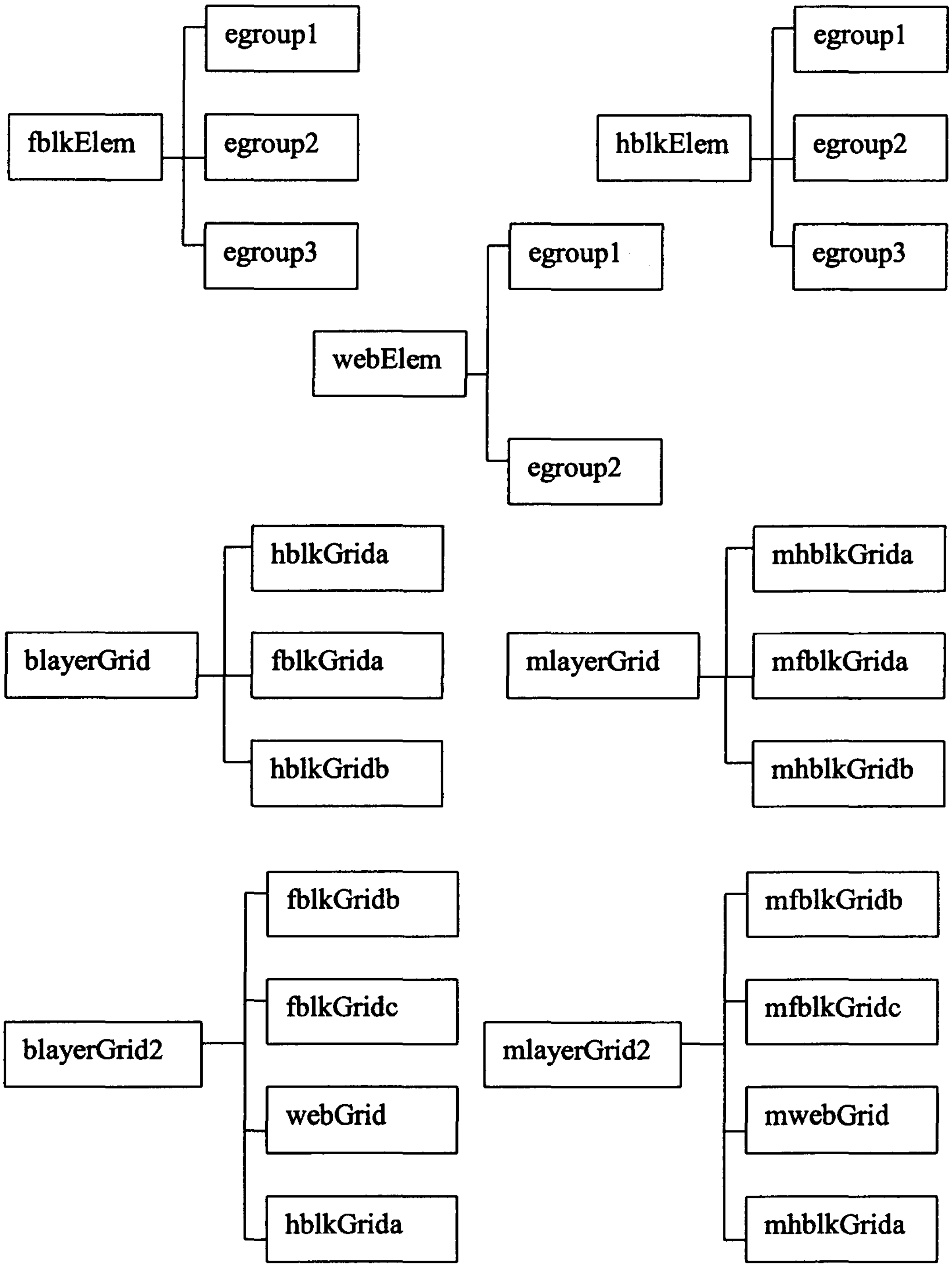


Figure A.2 BMFEM Modular structure (continued)

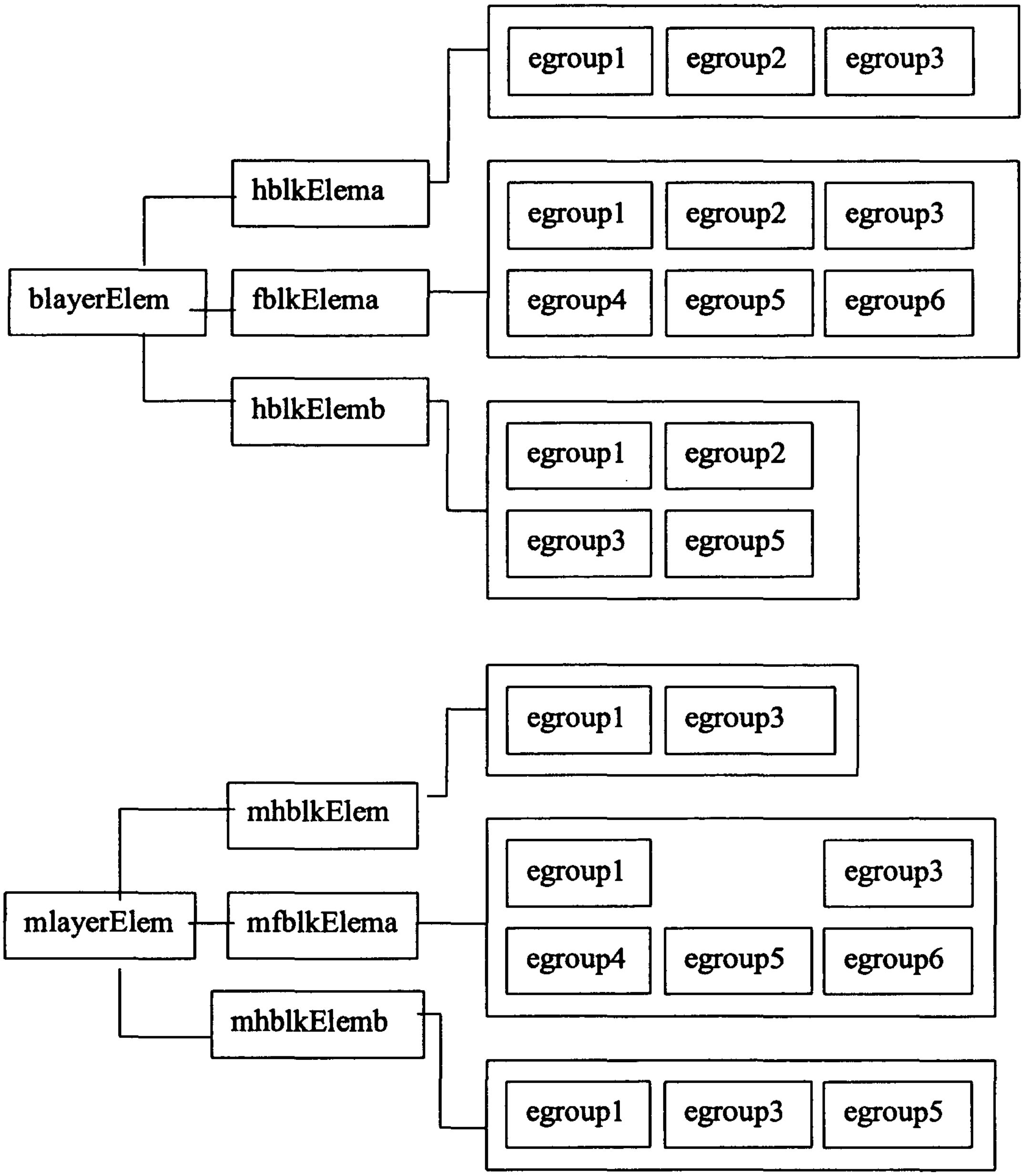


Figure A.3 BMFEM Modular structure (continued)

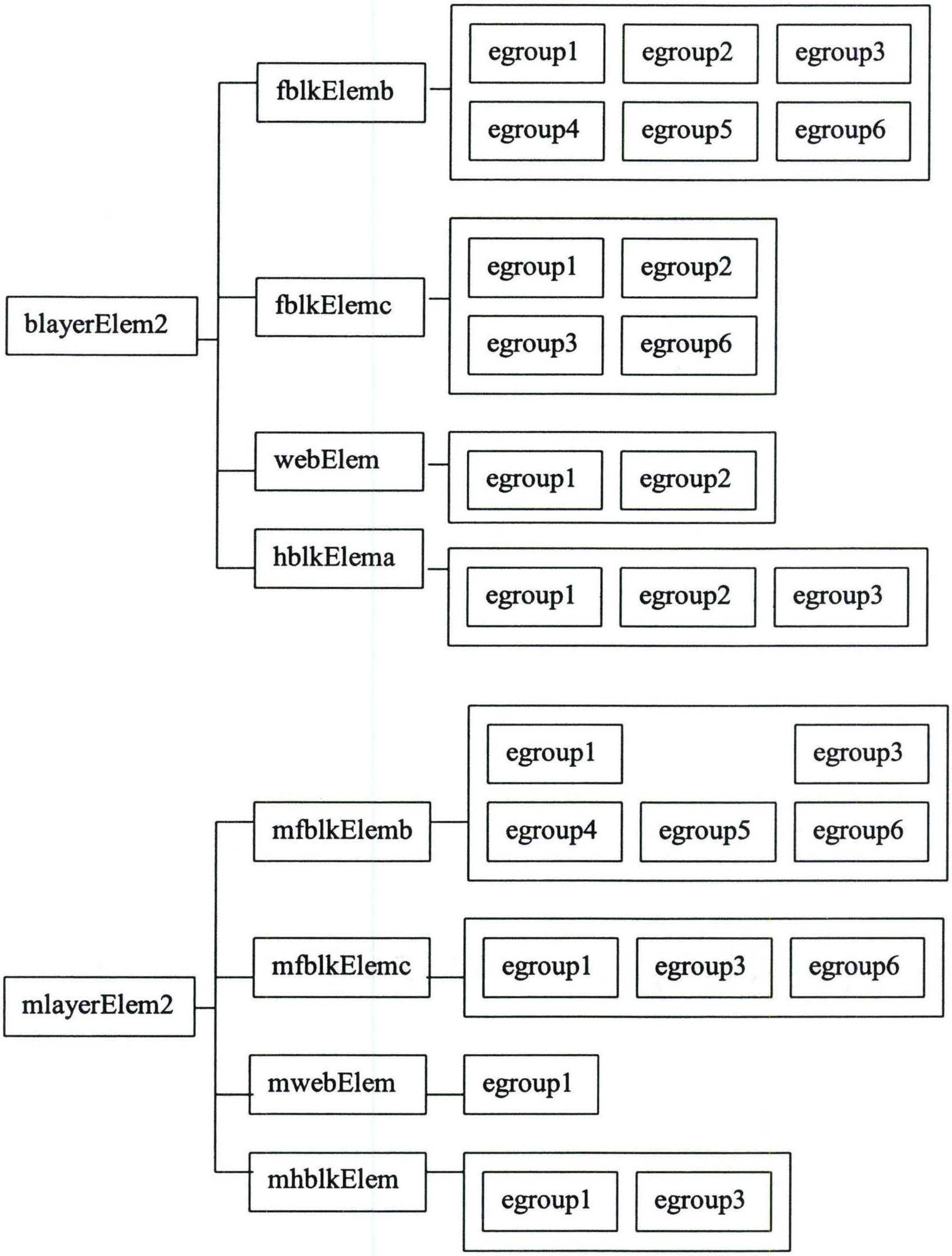


Figure A.4 BMFEM Modular structure (continued)

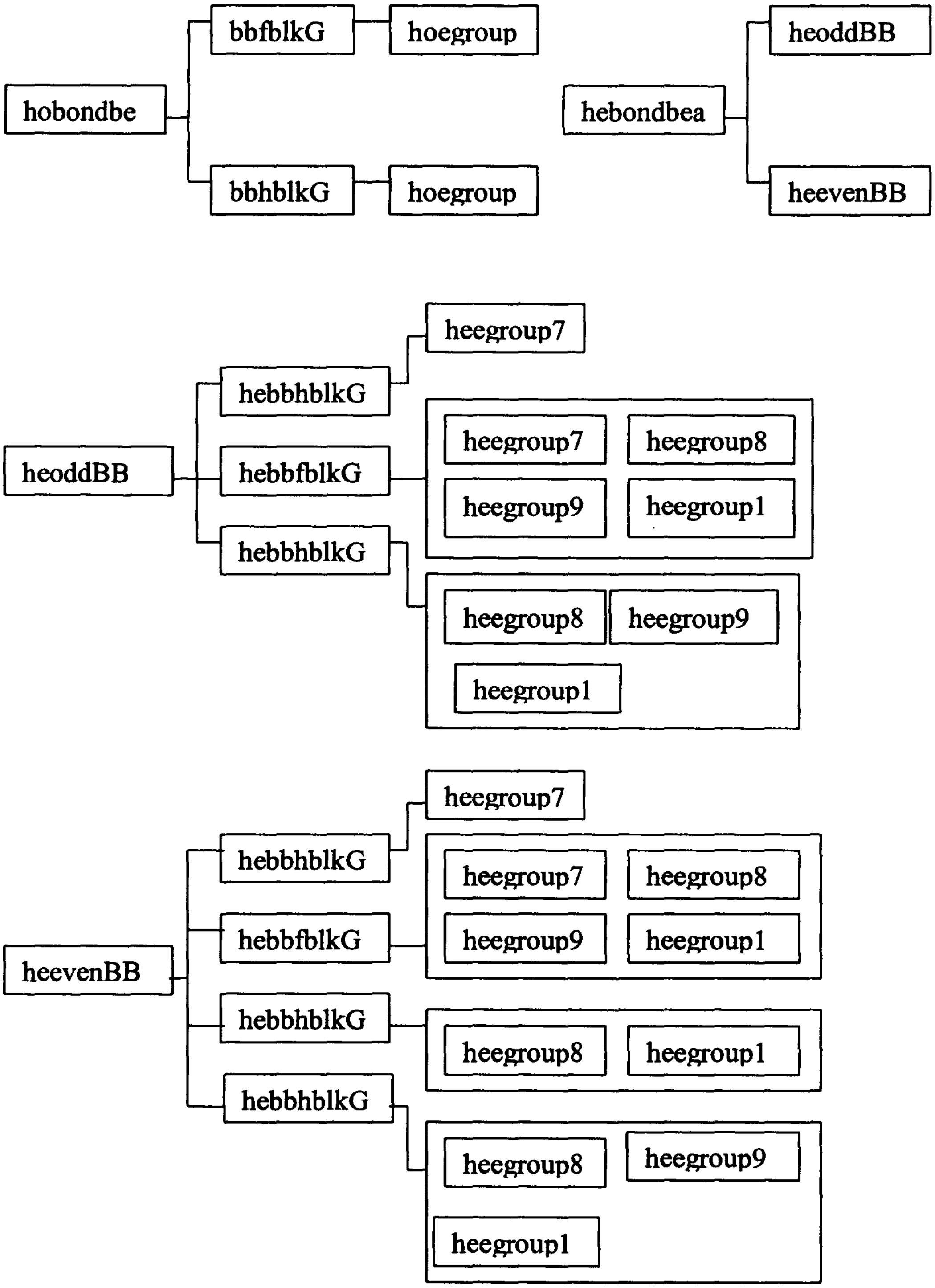


Figure A.5 BMFEM Modular structure (continued)

constraint Generates constraint information

egroup1~6 Defines the element group 1~6

fblkElem Defines elements in a full block unit for homogeneous model

fblkElema Defines elements in a full block unit in an odd course for heterogeneous model

fblkElemb Defines elements in a full block unit in an even course for heterogeneous model

fblkElemc Defines elements in the last full block in even layer for heterogeneous model

fblkGrid Defines grids in a full block unit for homogeneous model

fblkGrida Defines grids in a full block unit in an odd course for heterogeneous model

fblkGridb Defines grids in a full block unit in an even course for heterogeneous model

fblkGridc Defines grids in the last full block in even layer for heterogeneous model

formOpenning Processes grids and elements and loading information in the opening area

hblkElem Defines elements in a half block unit for homogeneous model

hblkElema	Defines elements in the first half block unit for heterogeneous model
hblkElemb	Defines elements in the last half block unit for heterogeneous model
hblkGrid	Defines grids in a half block unit for homogeneous model
hblkGrida	Defines grids in the first half block unit for heterogeneous model
hblkGridb	Defines grids in the last half block unit for heterogeneous model
hebbfblkGEa	Defines grids/elements in a full-block bond beam in odd course (heterogeneous)
hebbfblkGEb	Defines grids/elements in a full-block bond beam in even course (heterogeneous)
hebbhblkGEa	Defines grids/elements in the first half-block bond beam
hebbhblkGEb	Defines grids/elements in the last half-block bond beam in odd course
hebbhblkGEc	Defines grids/elements in the last half-block bond beam in even course (case 1)
hebbhblkGEd	Defines grids/elements in the last half-block bond beam in even course (case 2)
hebondbeam	Models bond beams in heterogeneous model

heegroup7~10	Defines element group 7~10 as bond beam in heterogeneous models
heevenBB	Models bond beams in an even course in heterogeneous models
heoddBB	Models bond beams in an odd course in heterogeneous models
heterModel	Forms heterogeneous model
hobondbeam	Models bond beams in homogeneous model
hoegroup4	Defines element group 4 as bond beam in homogeneous models
homoModel	Form homogeneous model
inputData	Requires interactive inputs of data from users
loading	Generates loading information
mergeGRID	Merges the duplicated grids so that the each grid number is unique
mergeRows	Removes the duplicated rows of a matrix
mfblkElema	Defines elements in a full-block mortar joint in an odd course (heterogeneous)
mfblkElemb	Defines elements in a full-block mortar joint in an even course (heterogeneous)
mfblkElemc	Defines elements in the last full-block mortar in an even course (heterogeneous)

mfblkGrida	Defines grids in a full-block mortar joint in an odd course (heterogeneous)
mfblkGridb	Defines grids in a full-block mortar joint in an even course (heterogeneous)
mfblkGridc	Defines grids in the last full-block mortar in an even course (heterogeneous)
mhblkElema	Defines elements in the first half-block mortar joint for heterogeneous model
mhblkElemb	Defines elements in the last half-block mortar joint for heterogeneous model
mhblkGrida	Defines grids in the first half-block mortar joint for heterogeneous model
mhblkGridb	Defines grids in the last half-block mortar joint for heterogeneous model
mlayerElem	Defines elements in the mortar layer in an odd course (heterogeneous)
mlayerElem2	Defines elements in the mortar layer in an even course (heterogeneous)
mlayerGrid	Defines grids in the mortar layer in an odd course (heterogeneous)

mlayerGrid2 Defines grids in the mortar layer in an even course (heterogeneous)

mwebElem Defines elements in the last single-web mortar joint (heterogeneous)

mwebGrid Defines grids in the last single-web mortar joint (heterogeneous)

openningLoad Calculates the loads applied on the edges of the opennings

renumberGE Renumbers the grids and elements in a continue order

reorderGE Rearrange the order of nodes in an element for all 6-node wedge elements

solidGE Forms grids and elements in the whole solid wall

webElem Defines elements in the last single-web block

webGrid Defines grids in the last single-web block

writeAFEMS Write all model information to the input data file of AFEMS

writeDIANA Write all model information to the input data file of DIANA

A.3 Important Variables

1. Wb, tm, Lb, Hb, tLb, tHb, tf, tw, Lw, Lf (Refer to Figure A.6, A.7)

Wb = block unit width

tm = mortar joint thickness

Lb = block unit length

Hb = block unit height

tLb = total length of a block = block unit length + mortar head joint thickness

tHb = total height of a block = block unit height + mortar bed joint thickness

tf = faceshell thickness

tw = web thickness

Lw = web length

Lf = face shell length

2. Wlength, Wheight

Wlength = wall model length

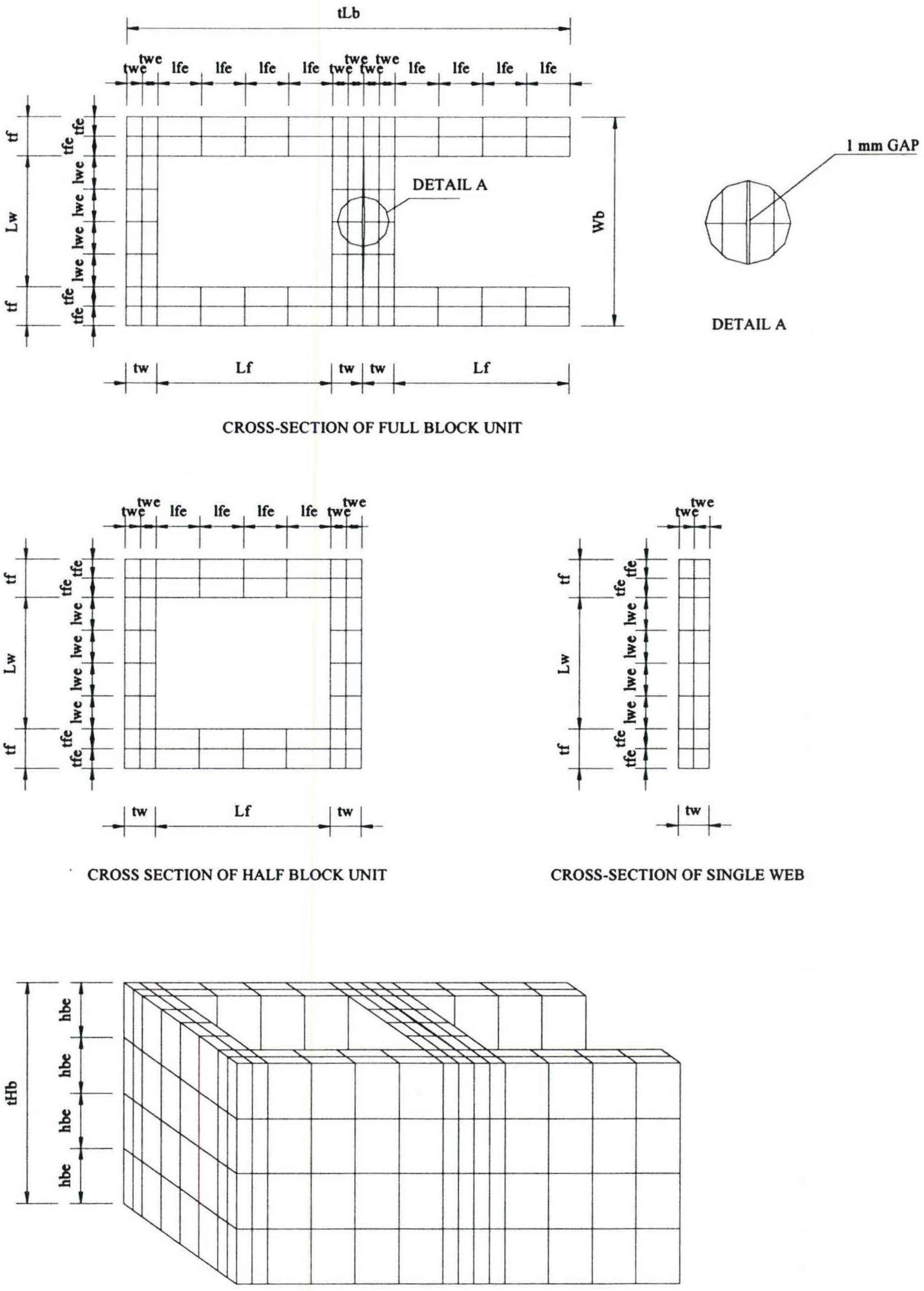
Wheight = wall model height

3. opening, oi, xopen(i), zopen(i), wopen(i), hopen(i), bbond(i), tbond(i)

opening = Boolean variable indicating whether openings exist

oi = number of openings

xopen(i) = distance between the left edge of the wall and the left edge of the i-th opening



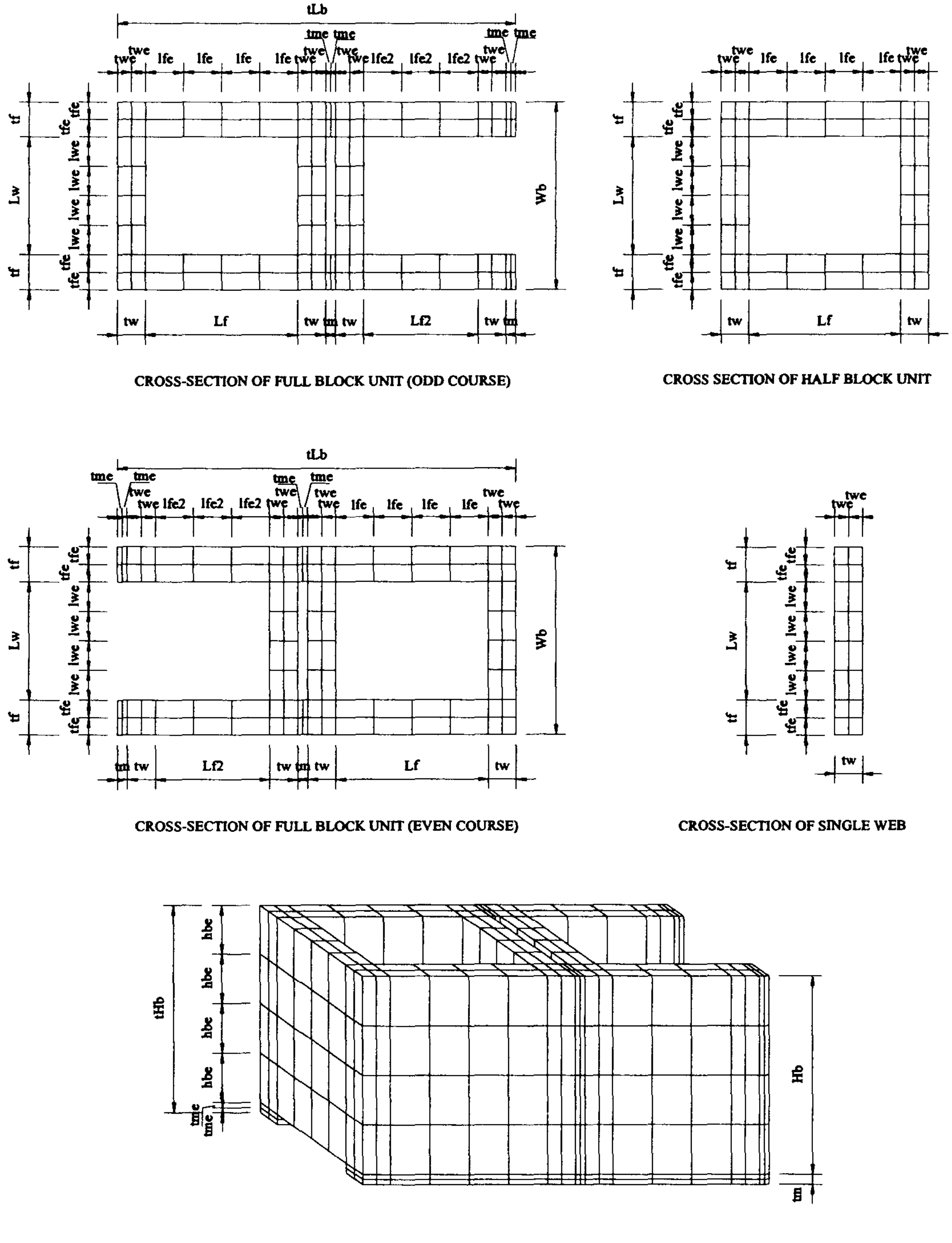
CROSS-SECTION OF FULL BLOCK UNIT

CROSS SECTION OF HALF BLOCK UNIT

CROSS-SECTION OF SINGLE WEB

APPROXIMATE FULL BLOCK UNIT

Figure A.6 Dimension variables of approximate block unit in homogeneous models



CROSS-SECTION OF FULL BLOCK UNIT (ODD COURSE)

CROSS SECTION OF HALF BLOCK UNIT

CROSS-SECTION OF FULL BLOCK UNIT (EVEN COURSE)

CROSS-SECTION OF SINGLE WEB

APPROXIMATE FULL BLOCK UNIT (ODD COURSE)

Figure A.7 Dimension variables of approximate block unit in heterogeneous models

zopen(i) = distance between the bottom edge of the wall and the bottom edge of the i-th opening

wopen(i) = width of the i-th opening

hopen(i) = height of the i-th opening

bbond(i) = Boolean variable indicating whether the i-th opening has a bottom bond beam

tbond(i) = Boolean variable indicating whether the i-th opening has a top bond beam

4. ftdiv, wtdiv, mtdiv, fldiv, fldiv2, wldiv, bhdiv

ftdiv = number of divisions in face shell thickness

wtdiv = number of divisions in web thickness

mtdiv = number of divisions in mortar joint thickness

fldiv = number of divisions in face shell length

fldiv2 = number of divisions in the longer section length of the second face shell (heterogeneous)

wldiv = number of divisions in web length

bhdiv = number of divisions in block height

5. micro

micro = Boolean variable indicating whether homogeneous or heterogeneous modeling is used

6. mvalue

mvalue = matrix holding the values of material properties

7. P, udl, PL

P = pressure load applied to the wall panel

udl = uniformly distributed load applied at the edges of the opening

PL = point loads applied at the grids at the edges of the opening

8. sym, lsym, rsym, bsym, tsym

sym = Boolean variable indicating whether symmetry is considered in the model

lsym = Boolean variable indicating whether the left edge of the wall is on the symmetric axis

rsym = Boolean variable indicating whether the right edge of the wall is on the symmetric axis

bsym = Boolean variable indicating whether the bottom edge of the wall is on the symmetric axis

tsym = Boolean variable indicating whether the top edge of the wall is on the symmetric axis

9. lc, rc, lbc, rbc, bbc, tbc

lc = left constraint location (distance from the left edge of the wall)

rc = right constraint location (distance from the left edge of the wall)

lbc = left boundary condition

rbc = right boundary condition

bbc = bottom boundary condition

tbc = top boundary condition

10. tfe, lwe, twe, lfe, lfe2, tme, hbe (Refer to Figure A.6, A.7)

tfe = face shell element thickness

lwe = web element length

twe = web element thickness

lfe = face shell element length

lfe2 = face shell element length in the longer section of the second face shell in heterogeneous model

tme = mortar element thickness

hbe = block element height

11. fullxnum, fullznum, yGNUM, zGNUM

fullxnum = number of full blocks in x direction fullznum = number of full blocks in z direction

yGNUM = number of grids across the block width with the same x and z coordinates

zGNUM = number of grids along the wall height with the same x and y coordinates

12. WGRID, WELEM, MGRID, mapG, rnGRID, mapRG, triELEM, GRID, ELEM, gnum, enum

WGRID = matrix containing the number and coordinates of the total grids created in the model

WELEM = matrix containing the number, grid connectivity, element group and material group information of the total elements created in the model

MGRID = matrix containing the information of the merged grids

mapG = matrix containing the mapping between the original grids and the merged grids

rnGRID = matrix containing the information of renumbered grids

mapRG = matrix containing the mapping between the grids (after merging) and the renumbered grids

triELEM = matrix containing the information of triangular-section (6-node wedge) elements in the corners of the openings

GRID = matrix containing the final information of the grids (after merging and renumbering)

ELEM = matrix containing the final information of the elements (after connectivity adjusting and renumbering)

gnum = number of grids

enum = number of elements

13. LCGRID, RCGRID, BCGRID, TCGRID, LCON, RCON, BCON, TCON

LCGRID = matrix containing the information of the grids on the left constraints

RCGRID = matrix containing the information of the grids on the right constraints

BCGRID = matrix containing the information of the grids on the bottom constraints

TCGRID = matrix containing the information of the grids on the top constraints

LCON = matrix containing the grid numbers and constraint information of the left constraints

RCON = matrix containing the grid numbers and constraint information of the right constraints

BCON = matrix containing the grid numbers and constraint information of the bottom constraints

TCON = matrix containing the grid numbers and constraint information of the top constraints

14. fGRID, EP1, EP2, EP

fGRID = matrix containing the grids on the wall surface where the pressure load applied

EP1 = matrix containing the element numbers and load-applied face numbers of the prismatic elements

EP2 = matrix containing the element numbers and load-applied face numbers of the triangular-section (6-node wedge) elements

EP = matrix containing the element numbers, load-applied face numbers, increments of element numbers and load values of all the elements under loading

A.4 User's Guide

BMFEM is composed of 1 main program BMFEM.m and 74 subroutines, including script files and functions. As a simple preprocessing tool, BMFEM is designed to provide a friendly text-format interface to allow for interactive data input. Only a few steps are necessary to create a model using BMFEM:

1. Copy all the matlab files with extension .m into the working directory;

2. Enter the MATLAB environment;

3. In the command window, type in 'bmfem' and press the Return key;

4. Follow the prompts to enter the required data for the model to be created.

After all the required data are entered, the program starts processing until all the information for the model, including mesh, material, constraint and loading conditions are written into the file specified by the users. For now, two files with different formats are generated, one for AFEMS with file extension .txt and another for DIANA with extension .dat.

There are 9 groups of input data required to generate the finite element model of block masonry. Currently, only the two-cell standard block unit is considered in this mesh generator program, since it is the most common block unit used in load-bearing masonry structures. For most groups of the data, the typical values or the range of the values are given. A default value will be taken if the user presses the Return key without entering any data. In case when the user gives incorrect inputs, the program will keep prompting the user to correct the value or exit the program. A brief explanation is given as follows for each data group.

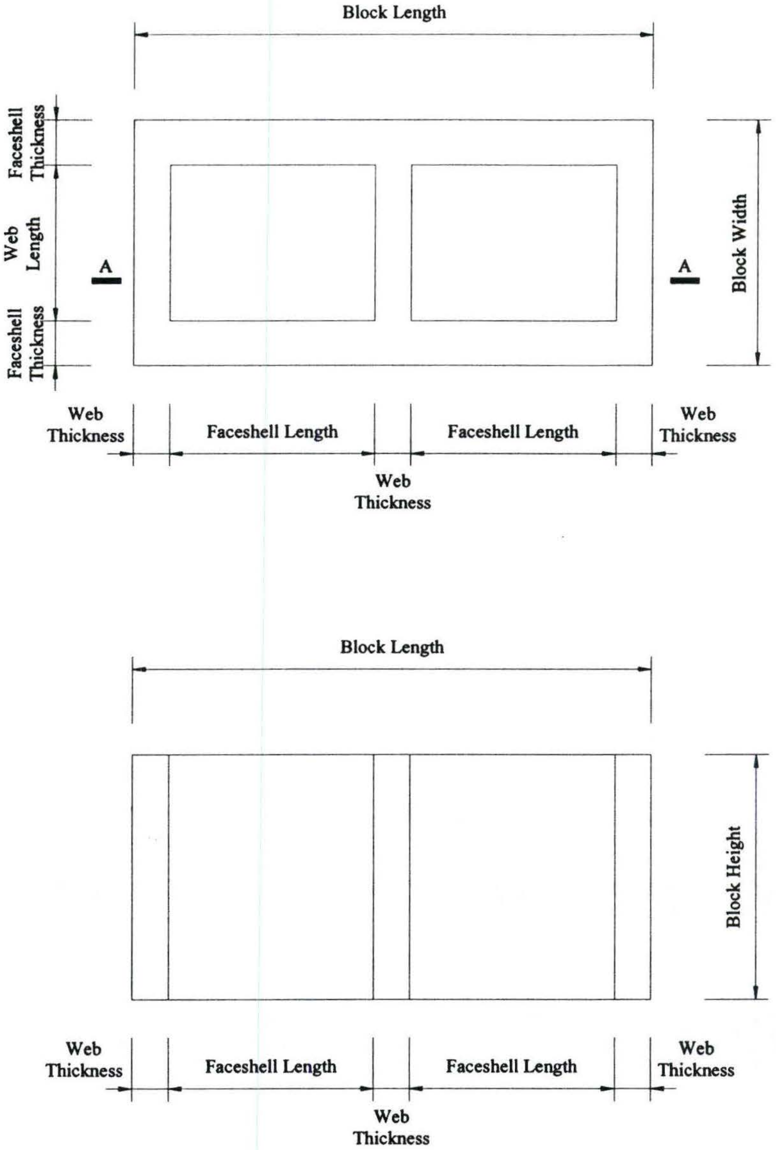
1. Dimension of the masonry components

Block unit width, face shell thickness, web thickness and mortar joint thickness are specified in this group. The available standard concrete block products have 140, 190, 240, and 290 mm widths but their lengths and heights are fixed, which are 390 mm and 190mm, respectively. These default values of the length and height are incorporated in the program. The definitions of all the dimensions of a block unit are illustrated in Figure A.8.

2. Dimension of the wall

In practice, a concrete block wall with running bond is usually constructed using full block units and half block units. The wall thus has modular dimensions with the length being either the multiple of the nominal half block length (e.g. 200mm) or the nominal full block length (e.g. 400mm), and the height being the multiple of the nominal block height (e.g. 200mm). Therefore, only the modular values of the wall length and height are considered to be correct inputs in this program. A wall size down to one block size and up to 8×6m is chosen to be effective size. The upper limit is determined by taking into account the computer performance. Nevertheless, this limitation can be easily modified in the file inputData.m if necessary.

If symmetry is considered, the wall dimensions should be taken as the values of the half or quarter model. For example, one considers modeling a wall with a horizontal span of 6 m and a vertical span of 2.8 m under uniformly distributed out-of-plane load. Given the same support conditions at the left and right side but different at the top and bottom side, the model can be considered to be horizontally symmetry and the wall length of 3 m and the height of 2.8 m should be the input.



DETAIL A-A

Figure A.8 Definitions of block unit dimensions

3. Opening information

BMFEM is capable of dealing with multiple rectangular openings with arbitrary dimensions, i.e., the opening dimensions are not necessarily modular. However, if bond beams above or below the openings need to be included, the input dimensions should be modular due to current limitation of the program. It is also suggested that the widths of the piers between openings not less than half block lengths (e.g. 200mm) to accommodate the extension of the bond beam on each side of the openings. It should be noted that, if symmetry is considered in the model, the width or height of the opening across the symmetric axis should be taken as the half value of the full opening (see example 1).

4. Mesh sizes

Mesh size is defined by giving the number of divisions of the dimensions of each material component. The maximum number of face shell or web thickness divisions is limited to 4 in the program. This is due to consideration that by combining this division limit with appropriate discretization of other parts of the block, the adequate accuracy of the calculated results (within 1% difference between two meshes) for common wall models can be achieved. Same explanation applies to mortar joint divisions. Face shell length, web length and block height have relatively longer dimensions and hence they are given larger division numbers. Note that only even number greater than 2 is valid for web length division because usually the nodes in the middle of the webs need to be presented as constraint or loading points.

5. Modeling method

Two types of model, heterogeneous model and homogeneous model need to be specified, as different mesh generating procedures will be evoked.

6. Material properties

According to the given modeling method, different material property inputs are required. For heterogeneous model, if block and mortar joints are both modeled with continuum elements, the materials are usually assumed to be isotropic therefore only the elastic modulii, Poisson ratios and mass density of block and mortar are needed; whereas if composite interface elements are used to model the mortar joints, parameters including mortar bond strength, tensile fracture energy, cohesion, tangent of friction angle, tangent of dilatancy angle, residual friction coefficient, confining normal stress, exponential degradation coefficient, shear fracture energy coefficients, compressive strength, shear traction control factor, compressive fracture energy and equivalent plastic relative displacement must be input. For homogeneous model, users should give all the nine values of the elastic constants (elastic modulii, Poisson ratios and shear modulii) and mass density due to the typical anisotropic assumption of the combined block masonry material. Default values are provided to facilitate the input (simply press the Return key to accept the value). Users may modify these default values in the array variable named 'mdefault' in the file inputData.m whenever necessary.

7. Pressure load

In current program, pressure load is input as the pressure load applied normal to the surface of the wall panel. By modifying the elements where the loads apply in the subroutine loading.m, the pressure loads acting on other faces of the wall model can also be defined.

8. Symmetric condition

Symmetric condition is considered only when the geometry, constraint and loading are all symmetric for a half or quarter model.

9. Support condition

The program provides three common boundary condition options for each side of the wall, free, fix or simply supported. As mentioned above, for now out-of-plane analysis is of major interest, the simply supported conditions on the left and right side are restricted to the constraints on Y direction (wall thickness direction); and the in-plane simply supported conditions on the bottom and top sides are restricted to the fixed constraints on the middle nodes of the webs. Likewise, users can easily change the constraint information (e.g. 1 indicates fix and 0 indicates free for the constraint points) in the matrix variables LCON, RCON, BCON and TCON, which represents the constraint information of the left, right, bottom and top side of the wall, respectively. On the other hand, taking into account the fact that actual experimental set-up may result in the left and right supports located at 100 mm of distance from each side, the options of the locations of the left and right supports are open for users.

The following two examples show the detailed inputs for two models. A '↵' sign indicates the Return key is pressed.

Example 1 A garage wall 6 m by 2.8 m is constructed using 20-cm standard blocks and 10-mm mortar joints. The door is 2.8 m wide and 2 m high as shown in Figure A.9. Bond beams are placed on top and bottom of the opening. Four sides of the wall are simply supported but the left, right and top constraint points are located at the edges of the back sides of the wall and, the bottom constraints are located at the middle of the bottom side of the wall. The desired mesh is 2 elements within face shell and web thickness, 4 elements within face shell and web length as well as block height. Chosen material properties are: Ex = 7550 MPa, Ey = 7550 MPa, Ez = 5800 MPa, νx = 0.24, νy = 0.24, νz = 0.24, Gx = 3500 MPa, Gy = 3500 MPa, Gz = 2600 MPa and density ρ = 2330 kg/m^3. Wind load of 1 kPa is applied laterally to the panel. Create a homogeneous model.

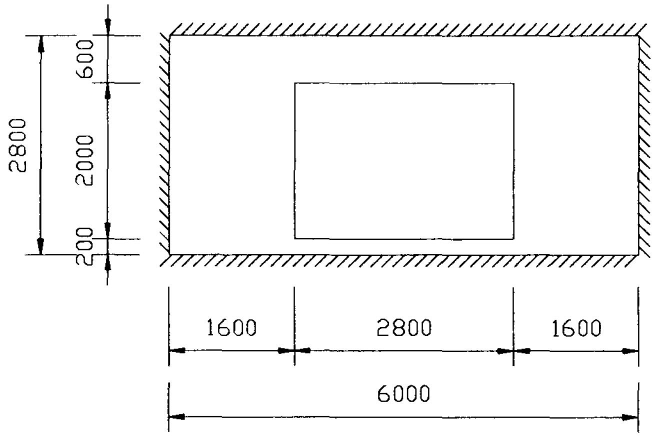


Figure A.9 Wall model of example 1

```
>> bmfem↵

Block Masonry Finite Element Mesh Generator (V1.0  2004)
      22-Dec-2004 15:04:17


Enter a filename for the data file
  to be generated (without extension):  wall1↵

Enter the following 9 groups of the input data:

1. Dimension of the masonry components

       Block unit width (mm) (140, 190<default>, 240 or 290):  ↵

       Mortar joint thickness (mm) (6, 10<default> or 15):  ↵

       Face shell thickness (mm) (28, 35<default>, 38, 40):  ↵

       Web thickness (mm) (28<default>, 30, 34):  ↵

2. Dimension of the wall

       Wall Length (mm) (>= 400 and <= 8000 and be divisible by 200.0):
3000↵

       Wall Height (mm) (>= 200 and <= 6000 and be divisible by 200):
2800↵
```

```
3. Opening information

       With openings? (Please enter 1 (yes) or 0 (no <default>))  1↵

       Number of openings (1<default>, 2, 3, 4, 5, 6, 7, 8):  ↵

       Input data from the leftmost opening to the rightmost opening:

       Opening 1 dimension

         Distance from the left side (mm) (>= 0 and < 3000):  1600↵

         Distance from the bottom side (mm) (>= 0 and < 2800):  200↵

         Opening width (mm) (> 0):  1400↵

         Opening height (mm) (> 0):  2000↵

         With  bottom  bond  beam?  (Please  enter  1  (yes)  or  0  (no
<default>))  1↵

         With top bond beam? (Please enter 1 (yes) or 0 (no <default>))
1↵

4. Mesh sizes

       Number of divisions of face shell thickness (1<default>, 2, 3, 4):
2↵

       Number of divisions of web thickness (1<default>, 2, 3, 4):  2↵

       Number of divisions of mortar joint thickness (1<default>, 2):  ↵

       Number of divisions of face shell length (excluding web thickness)
         (1, 2<default>, 3, 4, 5, 6, 7, 8, 9, 10):  4↵

       Number of divisions of web length (excluding face shell thickness)
         (2<default>, 4, 6, 8, 10):  4↵

       Number of divisions of block height
         (1, 2<default>, 3, 4, 5, 6, 7, 8, 9, 10):  2↵

5. Modeling method (1 (heterogeneous) or 0 (homogeneous <default>)):  ↵

6. Material properties

       Equivalent material properties of masonry:

         EX (MPa) (7550<default>):  ↵

         EY (MPa) (7550<default>):  ↵

         EZ (MPa) (5600<default>):  5800↵

         NUXY (0.213<default>):  0.24↵
         NUXZ (0.213<default>):  0.24↵

         NUYZ (0.213<default>):  0.24↵

         GXY (MPa) (3112<default>):  3500↵

         GXZ (MPa) (3112<default>):  3500↵

         GYZ (MPa) (2800<default>):  2600↵

         MASS (kg/mm^3) (2.33E-006<default>):  ↵

7. Pressure load (KPa) (1<default>):  ↵

8. Symmetric condition

       Is there any symmetric axis (1 (yes) or 0 (no)<default>)?  1↵

       Is the left side on the symmetric axis (1 (yes) or 0
(no)<default>)?  ↵

       Is the right side on the symmetric axis (1 (yes) or 0
(no)<default>)? 1↵

       Is the bottom side on the symmetric axis (1 (yes) or 0
(no)<default>)? ↵

       Is the top side on the symmetric axis (1 (yes) or 0
(no)<default>)? ↵

9. Support condition

       Left support

       Location (1 (left edge of the wall) or
                 2 (100 from the left edge of the wall<default>)):  1↵

       Boundary condition (1(free), 2(fix) or 3(simply supported
<default>)):

       Right support

       Boundary condition: roller

       Bottom support

       Boundary condition (1(free), 2(fix), 3(out-of-plane simply
supported)
         or 4(in-plane simply supported <default>)):  ↵

       Top support

       Boundary condition (1(free), 2(fix), 3(out-of-plane simply
supported <default>)
         or 4(in-plane simply supported)):  ↵
```

The input data is stored in the file 'wall1.txt' for AFEMS and 'wall1.dat' for DIANA.

Example 2 A wall panel 3.8 m by 3 m contains a window and a door with dimension 1.2 m × 1.2 m and 1 m × 2.2 m, respectively (Figure A.10). The standard 30cm blocks are used. Bond beams are placed on top of the openings. The left and right side of the wall are fixed ends; the bottom is in-plane simply supported and the top is free. Create a heterogeneous model using continuum elements for both units and mortar joints. Mesh size: 3 elements within the face shell thickness, 2 elements within the web thickness, 6 elements within face shell length and 4 in web length, and 4 elements within the block height. Chosen material properties are: Block: Ex = 9000 MPa, νx = 0.213 and ρ = 2330 kg/m^3; Mortar: Ex = 1180 MPa, νx = 0.2 and ρ = 1970 kg/m^3. Wind load of 1 kPa is applied laterally to the panel.

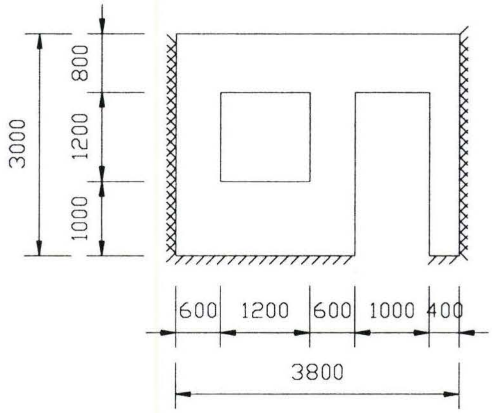


Figure A.10 Wall model of example 2

```
>> bmfem

Block Masonry Finite Element Mesh Generator (V1.0  2004)
      28-Dec-2004 20:10:28


Enter a filename for the data file
  to be generated (without extension):  wall2↵

Enter the following 9 groups of the input data:

1. Dimension of the masonry components

       Block unit width (mm) (140, 190<default>, 240 or 290):  290↵

       Mortar joint thickness (mm) (6, 10<default> or 15): ↵

       Face shell thickness (mm) (28, 35<default>, 38, 40):  40↵

       Web thickness (mm) (28<default>, 30, 34):  34↵

2. Dimension of the wall

       Wall Length (mm) (>= 400 and <= 8000 and be divisible by 200.0):
3800↵

       Wall Height (mm) (>= 200 and <= 6000 and be divisible by 200):
3000↵

3. Opening information

       With openings? (please enter 1 (yes) or 0 (no <default>))  1↵

       Number of openings (1<default>, 2, 3, 4, 5, 6, 7, 8):  2↵

       Input data from the leftmost opening to the rightmost opening:

       Opening 1 dimension

         Distance from the left side (mm) (>= 0 and < 3800):  600↵

         Distance from the bottom side (mm) (>= 0 and < 3000):  1000↵

         Opening width (mm) (> 0):  1200↵

         Opening height (mm) (> 0):  1200↵

         With  bottom  bond  beam?  (Please  enter  1  (yes)  or  0  (no
<default>))  ↵

         With top bond beam? (Please enter 1 (yes) or 0 (no <default>))
1↵

       Opening 2 dimension

         Distance from the left side (mm) (>= 0 and < 3800):  2400↵
         Distance from the bottom side (mm) (>= 0 and < 3000):  0↲

         Opening width (mm) (> 0):  1000↲

         Opening height (mm) (> 0):  2200↲

         With top bond beam? (Please enter 1 (yes) or 0 (no <default>))
1↲

4. Mesh sizes

       Number of divisions of face shell thickness (1<default>, 2, 3, 4):
3↲

       Number of divisions of web thickness (1<default>, 2, 3, 4):  2↲

       Number of divisions of mortar joint thickness (1<default>, 2):  ↲

       Number of divisions of face shell length (excluding web thickness)
         (1, 2<default>, 3, 4, 5, 6, 7, 8, 9, 10):  6↲

       Number of divisions of web length (excluding face shell thickness)
         (2<default>, 4, 6, 8, 10):  4↲

       Number of divisions of block height
         (1, 2<default>, 3, 4, 5, 6, 7, 8, 9, 10):  4↲

5. Modeling method (1 (heterogeneous) or 0 (homogeneous <default>)):  1↲

6. Material properties

       Block:

         EX (MPa) (9000<default>):  ↲

         NUXY (0.213<default>):  ↲

         MASS (kg/mm^3) (2.33E-006<default>):  ↲

       Mortar:

         EX (MPa) (1000<default>):  1180↲

         NUXY (0.200<default>):  ↲

         MASS (kg/mm^3) (1.97E-006<default>):  ↲

7. Pressure load (KPa) (1<default>):  ↲

8. Symmetric condition

       Is there any symmetric axis (1 (yes) or 0 (no)<default>)?  ↲

9. Support condition

       Left support
       Location (1 (left edge of the wall) or
                 2 (100 from the left edge of the wall<default>)):  1↵

       Boundary  condition  (1(free),  2(fix)  or  3(simply  supported
<default>)): 2↵

       Right support

       Location (1 (right edge of the wall) or
                 2 (-100 from the right edge of the wall<default>)):  1↵

       Boundary  condition  (1(free),  2(fix)  or  3(simply  supported
<default>)):  2↵

       Bottom support

       Boundary  condition  (1(free),  2(fix),  3(out-of-plane  simply
supported)
         or 4(in-plane simply supported <default>)):  ↵

       Top support

       Boundary  condition  (1(free),  2(fix),  3(out-of-plane  simply
supported <default>) or 4(in-plane simply supported)):  1↵
```

The generated data files are 'wall2.txt' for AFEMS and 'wall2.dat' for DIANA.

A.5 Source Code

A.5.1 Main Program BMFEM.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The script file BMFEM.m is the main program to create a 3-D finite
% element model of block masonry wall, including the mesh, material,
loading and
% constraint information. It contains the following subroutines:
%      inputData.m
%      homoModel.m
%      heterModel.m
%      formOpenning.m
%      mergeGRID.m
%      renumberGE.m
%      constraint.m
%      loading.m
%      writeAFEMS.m
%      writeDIANA.m
% It can be used to generate data files used as input files for finite
element
```

```
% packages. Currently the generated files *.txt and *.dat are suitable
for AFEMS
% and DIANA finite element analysis packages, respectively.
%
% Author: Zhong He
% Supervised by: Dr. Samir E. Chidiac and Dr. Robert G. Drysdale
% Version of 18-Nov-2004 23:58:30.
% Updated: 08-Dec-2004 20:35:38
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear

disp(sprintf('\nBlock Masonry Finite Element Mesh Generator (V1.0
2004)\n       %s', datestr(clock,0)));
fname = input('\n\nEnter a filename for the data file \n  to be generated
(without extension):  ','s');
if isempty(fname)
    fname='default';
end
file1 = [ fname '.txt'];
file2 = [ fname '.dat'];

%request input data
alldata=0;
inputData;
if ~alldata
    return;
end

t0=clock;
disp(sprintf('\n\nProcessing starts at %s.', datestr(t0,0)));

fullxnum=fix(Wlength/tLb);
fullznum=Wheight/tHb;
tfe=tf/ftdiv;  % thickness of face shell element (y direction)
lwe=Lw/wldiv;  % length of web element (y direction)
twe=tw/wtdiv;  % thickness of web element (x direction)
lfe=Lf/fldiv;  % length of face shell element (x direction)
tme=tm/mtdiv;  % thickness of mortar element (x or z direction)
if micro==1     % length of second face shell element in heterogeneous
model
    if fldiv <= 2
        fldiv2=fldiv;
    else
        fldiv2=fldiv-1;
    end
    lfe2=(Lf-tw-tm)/fldiv2;
    hbe=Hb/bhdiv; % height of block element (z direction)
else
    hbe=tHb/bhdiv;
end
yGNUM=2*ftdiv+wldiv+1;
zGNUM=bhdiv+1;

%create the solid wall meshes
nstart=1;
xstart=0; ystart=0; zstart=0;
if micro==0
    homoModel;
else
```

```
    heterModel;
end
disp(sprintf('\nFull wall mesh completed.'));

%process opennings
if ~openning
    GRID=WGRID;
    ELEM=WELEM;
else
    formOpenning;
end

%merge the duplicated grids
disp(sprintf('\nMerging nodes......'));
mergeGRID;
disp(sprintf('\nNodes merged at %0.1f.', etime(clock,t0)));

%delete the grids not associated with elements
neGi=find(~ismember(GRID(:,1),ELEM(:,2:9)));
GRID(neGi,:)=[];

%renumber the grids and elements
disp(sprintf('\nRenumberring nodes and elements......'));
renumberGE;
disp(sprintf('\nRenumberring completed at %0.1f.', etime(clock,t0)));

%adjust the grids at left and right support
if lc~=0
    lcGrid=[GRID(:,1) GRID(:,2)-lc];
    lcGi=find(abs(lcGrid(:,2))==min(abs(lcGrid(:,2))));
    GRID(lcGi,2)=lc;
end
if rc~=maxx
    rcGrid=[GRID(:,1) GRID(:,2)-rc];
    rcGi=find(abs(rcGrid(:,2))==min(abs(rcGrid(:,2))));
    GRID(rcGi,2)=rc;
end
gnum=size(GRID,1);
enum=size(ELEM,1);

% generate constraint information
constraint;
disp(sprintf('\nConstraint points defined at %0.1f.', etime(clock,t0)));

% generate loading information
loading;
disp(sprintf('\nPressure loads defined at %0.1f.', etime(clock,t0)));

%form input file for FE packages
disp(sprintf('\nWriting data to the file %s for AFEMS......', file1));
writeAFEMS;
disp(sprintf('\nWriting data to the file %s for DIANA......', file2));
writeDIANA;

t1=clock;
disp(sprintf('\nProcessing completed at %s.', datestr(t1,0)));
disp(sprintf('\nTotal time = %0.1f second.', etime(t1,t0)));
```

A.5.2 Subroutine inputData.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The script file inputData.m is a subroutine of BMFEM.m to obtain
% interactive inputs from users.
%
% Author: Zhong He
% Supervised by: Dr. Samir E. Chidiac and Dr. Robert G. Drysdale
% Version of 18-Nov-2004 23:58:30.
% Updated: 08-Dec-2004 20:35:38
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

disp(sprintf('\nEnter the following 9 groups of the input data:'));
disp(sprintf('\n1. Dimension of the masonry components'));

%block unit width
Wb = input('\n        Block unit width (mm) (140, 190<default>, 240 or
290):  ');
if isempty(Wb)
    Wb=190;
else
      while ~ismember(Wb, [140 190 240 290])
       Wb = input('\n               Please try an appropriate value (or
press Return                key to exit):  ');
       if isempty(Wb)
           return;
       end
      end
end

%mortar joint thickness
tm = input('\n       Mortar joint thickness (mm) (6, 10<default> or 15):
');
if isempty(tm)
    tm=10;
else
      while ~ismember(tm, [6 10 15])
       tm = input('\n               Please try an appropriate value (or
press Return key to exit):  ');
       if isempty(tm)
           return;
       end
      end
end

Lb=390;  %block unit length
Hb=190;  %block unit height
tLb=Lb+tm; %total block length
tHb=Hb+tm; %total block height

%faceshell thickness
tf = input('\n       Faceshell thickness (mm) (30, 32, 35<default>, 38,
40):  ');
if isempty(tf)
    tf=35;
else
      while ~ismember(tf, [30 32 35 38 40])
       tf = input('\n               Please try an appropriate value (or
press Return key to exit):  ');
```

```
        if isempty(tf)
            return;
        end
      end
end

%web thickness
tw = input('\n      Web thickness (mm) (26, 28<default>, 30):  ');
if isempty(tw)
    tw=28;
else
      while ~ismember(tw, [26 28 30])
        tw = input('\n               Please try an appropriate value (or
press Return key to exit):  ');
        if isempty(tw)
            return;
        end
      end
end

disp(sprintf('\n2. Dimension of the wall'));
%wall length
Wlength = input(sprintf('\n       Wall Length (mm) (>= %i and <= %i and
be divisible by %0.1f):  ', tLb, 20*tLb, tLb/2));
while    (isempty(Wlength)    |    Wlength<tLb    |    Wlength>8000    |
mod(Wlength,tLb/2)~=0)
    Wlength = input('\n               Please try an appropriate value (or
press Return key to exit):  ');
    if isempty(Wlength)
        return;
    end
end

%wall height
Wheight = input(sprintf('\n       Wall Height (mm) (>= %i and <= %i and
be divisible by %i):  ', tHb, 30*tHb, tHb));
while    (isempty(Wheight)    |    Wheight<tHb    |    Wheight>6000    |
mod(Wheight,tHb)~=0)
    Wheight = input('\n               Please try an appropriate value (or
press Return key to exit):  ');
    if isempty(Wheight)
        return;
    end
end

disp(sprintf('\n3. Openning information'));
%if opennings exit
openning = input('\n      With opennings? (please enter 1 (yes) or 0 (no
<default>))  ');
if isempty(openning)
    openning=0;
else
      while (openning~=0 & openning~=1)
        openning = input('\n             Please try an appropriate answer
(or press Return key to exit):  ');
        if isempty(openning)
            return;
        end
      end
end
```

```
%number of openning
if openning
     oi = input('\n      Number of opennings (1<default>, 2, 3, 4, 5,
6, 7, 8):  ');
     if isempty(oi)
      oi=1;
     else
      while ~ismember(oi, [1 2 3 4 5 6 7 8])
         oi = input('\n          Please try an appropriate value (or
press Return key to exit):  ');
         if isempty(oi)
            return;
         end
          end
     end
     %openning dimension
   disp(sprintf('\n      Input data from the leftmost openning to the
rightmost openning:'));
     for i=1:oi
      if i==1
         disp(sprintf('\n      Openning %i dimension', i));
      else
         disp(sprintf('\n      Openning %i dimension', i));
      end
      xo = input(sprintf('\n        Distance from the left side (mm)
(>= 0 and < %i):  ', Wlength));
          while (isempty(xo) | xo<0 | xo>=Wlength)
         xo = input('\n          Please try an appropriate value (or
press Return key to exit):  ');
         if isempty(xo)
            return;
         end
          end
      xopen(i)=xo;
      zo = input(sprintf('\n        Distance from the the bottom side
(mm) (>= 0 and < %i):  ', Wheight));
          while (isempty(zo) | zo<0 | zo>=Wheight)
         zo = input('\n          Please try an appropriate value (or
press Return key to exit):  ');
         if isempty(zo)
            return;
         end
          end
      zopen(i)=zo;
      wo = input('\n       Openning width (mm) (> 0):  ');
      if wo>Wlength-xopen(i)
         wo=Wlength-xopen(i);
      else
               while (isempty(wo) | wo<=0)
            wo = input('\n         Please try an appropriate value
(or press Return key to exit):  ');
            if isempty(wo)
               return;
            end
               end
      end
      wopen(i)=wo;
      ho = input('\n       Openning height (mm) (> 0):  ');
      if ho>Wheight-zopen(i)
```

```
            ho=Wheight-zopen(i);
        else
                    while (isempty(ho) | ho<=0)
                ho = input('\n         Please try an appropriate value
(or press Return key to exit):  ');
                if isempty(ho)
                    return;
                end
                    end
        end
        hopen(i)=ho;
        %bond beam
        bbond(i)=0;
        if  (xopen(i)>=tLb/2  &  zopen(i)>=tHb  &  mod(xopen(i),tLb/2)==0  &
mod(wopen(i),tLb/2)==0 & mod(zopen(i),tHb)==0)
            bb = input('\n        With bottom bond beam? (please enter 1
(yes) or 0 (no <default>))  ');
                    if isempty(bb)
                bb=0;
                    else
                           while (bb~=0 & bb~=1)
                    bb = input('\n              Please try an appropriate
answer (or press Return key to exit):  ');
                    if isempty(bb)
                        return;
                    end
                           end
                    end
            bbond(i)=bb;
        end
        tbond(i)=0;
        if   (xopen(i)>=tLb/2    &    (zopen(i)+hopen(i)<=Wheight-tHb)    &
mod(xopen(i),tLb/2)==0         &         mod(wopen(i),tLb/2)==0         &
mod(zopen(i)+hopen(i),tHb)==0)
            tb  =  input('\n        With top bond beam? (please enter 1
(yes) or 0 (no <default>))  ');
                    if isempty(tb)
                tb=0;
                    else
                           while (tb~=0 & tb~=1)
                    tb = input('\n              Please try an appropriate
answer (or press Return key to exit):  ');
                    if isempty(tb)
                        return;
                    end
                           end
                    end
            tbond(i)=tb;
        end
       end
end

%mesh sizes
disp(sprintf('\n4. Mesh sizes'));
ftdiv  =  input('\n        Number  of  divisions  of  faceshell  thickness
(1<default>, 2, 3, 4):  ');
if isempty(ftdiv)
    ftdiv=1;
else
      while (~ismember(ftdiv, [1 2 3 4]))
```

```
         ftdiv = input('\n           Please try an appropriate value (or
press Return key to exit):  ');
         if isempty(ftdiv)
             return;
         end
       end
end
wtdiv = input('\n       Number of divisions of web thickness (1<default>,
2, 3, 4):  ');
if isempty(wtdiv)
    wtdiv=1;
else
       while ~ismember(wtdiv, [2 3 4])
        wtdiv = input('\n           Please try an appropriate value (or
press Return key to exit):  ');
        if isempty(wtdiv)
            return;
        end
       end
end
mtdiv = input('\n       Number of divisions of mortar joint thickness
(1<default>, 2):  ');
if isempty(mtdiv)
    mtdiv=1;
else
       while ~ismember(mtdiv, [1 2])
        mtdiv = input('\n           Please try an appropriate value (or
press Return key to exit):  ');
        if isempty(mtdiv)
            return;
        end
       end
end
fldiv = input('\n        Number of divisions of faceshell length
(excluding web thickness)\n       (1, 2<default>, 3, 4, 5, 6, 7, 8, 9,
10):  ');
if isempty(fldiv)
    fldiv=2;
else
       while ~ismember(fldiv, [1 2 3 4 5 6 7 8 9 10])
        fldiv = input('\n           Please try an appropriate value (or
press Return key to exit):  ');
        if isempty(fldiv)
            return;
        end
       end
end
wldiv = input('\n       Number of divisions of web length (excluding
faceshell thickness)\n       (2<default>, 4, 6, 8, 10):  ');
if isempty(wldiv)
    wldiv=2;
else
       while ~ismember(wldiv, [2 4 6 8 10])
        wldiv = input('\n           Please try an appropriate value (or
press Return key to exit):  ');
        if isempty(wldiv)
            return;
        end
       end
end
```

```
bhdiv = input('\n       Number of divisions of block height\n         (1,
2<default>, 3, 4, 5, 6, 7, 8, 9, 10):  ');
if isempty(bhdiv)
    bhdiv=2;
else
       while ~ismember(bhdiv, [1 2 3 4 5 6 7 8 9 10])
        bhdiv = input('\n             Please try an appropriate value (or
press Return key to exit):  ');
        if isempty(bhdiv)
            return;
        end
       end
end

%modeling method
micro = input('\n5. Modeling method (1 (heterogeneous) or 0 (homogeneous
<default>)):  ');
if isempty(micro)
    micro=0;
else
       while (micro~=0 & micro~=1)
        micro = input('\n           Please try an appropriate answer (or
press Return key to exit):  ');
        if isempty(micro)
            return;
        end
       end
end

%material properties
disp(sprintf('\n6. Material properties'));
if micro==0
    disp(sprintf('\n       Equivalent material properties of masonry:'));
    MCON={'EX (MPa)' 'EY (MPa)' 'EZ (MPa)' 'NUXY' 'NUXZ' 'NUYZ' 'GXY
(MPa)' 'GXZ (MPa)' 'GYZ (MPa)' 'MASS (kg/mm^3)'};
    mdefault=[7550 7550 5600 0.213 0.213 0.213 3112 3112 2800 2.33E-06];
else
    MCON={'EX (MPa)' 'NUXY' 'MASS (kg/mm^3)'};
    mdefault=[9000 0.213 2.33E-06; 1000 0.2 1.97E-06];
end
for i=1:size(mdefault,1)
    if size(mdefault,1)==2
        if i==1
            disp(sprintf('\n       Block:'));
        else
            disp(sprintf('\n       Mortar:'));
        end
    end
       for j=1:size(mdefault,2)
        if micro==0
            if ismember(j,[1 2 3 7 8 9])
                mv = input(sprintf('\n            %s (%i<default>):   ',
char(MCON(j)), mdefault(i,j)));
            else if ismember(j,[4 5 6])
                    mv = input(sprintf('\n          %s (%0.3f<default>):
', char(MCON(j)), mdefault(i,j)));
                else
                    mv = input(sprintf('\n          %s (%0.2E<default>):
', char(MCON(j)), mdefault(i,j)));
                end
            end
        else
            if j==1
                mv = input(sprintf('\n            %s  (%i<default>):   ',
char(MCON(j)), mdefault(i,j)));
            else if j==2
                    mv = input(sprintf('\n            %s  (%0.3f<default>):
', char(MCON(j)), mdefault(i,j)));
                else
                    mv = input(sprintf('\n            %s  (%0.2E<default>):
', char(MCON(j)), mdefault(i,j)));
                end
            end
        end
              if isempty(mv)
            mv=mdefault(i,j);
              else
                    while mv<0
                mv = input('\n           Please try an appropriate value
(or press Return key to exit):  ');
                if isempty(mv)
                    return;
                end
                    end
              end
        mvalue(i,j)=mv;
       end
end

%loading
P = input('\n7. Pressure load (KPa) (1<default>):  ');
if isempty(P)
    P=1;
else
    while P==0
        P = input('\n         Please try an appropriate answer (or press
Return key to exit):  ');
        if isempty(micro)
            return;
        end
       end
end

Lw=Wb-2*tf; %web length
if micro==0
    tgap=1;  %gap width in homogeneous model
    Lf=(tLb-3*tw)/2; %face shell length of homogeneous model
else
    Lf=(tLb-3*tw-10)/2; %face shell length of heterogeneous model
end
if mod(Wlength,tLb)==tLb/2
    maxx=Wlength-tLb/2+2*tw+Lf; %maximum x-coordinate value
else if mod(Wlength,tLb)==0
        maxx=Wlength+tw;
    end
end

%symmetric condition
disp(sprintf('\n8. Symmetric condition'));
```

```
sym = input('\n           Is there any symmetric axis (1 (yes) or 0
(no)<default>)?  ');
if isempty(sym)
    sym=0;
else
       while ~ismember(sym, [0 1])
        sym = input('\n           Please try an appropriate value (or
press Return key to exit):  ');
        if isempty(sym)
            return;
        end
       end
end
lsym=0; rsym=0; bsym=0; tsym=0;
if sym==1
    lsym = input('\n         Is the left side on the symmetric axis (1
(yes) or 0 (no)<default>)?  ');
       if isempty(lsym)
        lsym=0;
       else
             while ~ismember(lsym, [0 1])
            lsym = input('\n           Please try an appropriate value
(or press Return key to exit):  ');
            if isempty(lsym)
                return;
            end
             end
       end
    if lsym==0
        rsym = input('\n         Is the right side on the symmetric axis (1
(yes) or 0 (no)<default>)?  ');
             if isempty(rsym)
            rsym=0;
             else
                    while ~ismember(rsym, [0 1])
                rsym = input('\n           Please try an appropriate
value (or press Return key to exit):  ');
                if isempty(rsym)
                    return;
                end
                    end
             end
    end
    bsym = input('\n         Is the bottom side on the symmetric axis (1
(yes) or 0 (no)<default>)?  ');
       if isempty(bsym)
        bsym=0;
       else
             while ~ismember(bsym, [0 1])
            bsym = input('\n           Please try an appropriate value
(or press Return key to exit):  ');
            if isempty(bsym)
                return;
            end
             end
       end
    if bsym==0
        tsym = input('\n         Is the top side on the symmetric axis (1
(yes) or 0 (no)<default>)?  ');
             if isempty(tsym)
            tsym=0;
             else
                    while ~ismember(tsym, [0 1])
                tsym = input('\n              Please try an appropriate
value (or press Return key to exit):  ');
                if isempty(tsym)
                    return;
                end
                    end
             end
    end
end

%support condition
disp(sprintf('\n9. Support condition'));
disp(sprintf('\n       Left support'));
if lsym==0    %left side not a symmetric axis
    %support location
       lc = input(sprintf('\n        Location (1 (left edge of the wall)
or\n                  2 (100 from the left edge of the wall<default>)):
'));
       if isempty(lc)
        lc=Lf+2*tw-100;
       else
             while ~ismember(lc, [1 2])
            lc = input('\n            Please try an appropriate value (or
press Return key to exit):  ');
            if isempty(lc)
                return;
            end
             end
        if lc==1
            lc=0;
        else
            lc=Lf+2*tw-100;
        end
       end
       if (lc~=0 & fldiv<2)
        fldiv = input('\n        The number of divisions of faceshell
length should be at least 2\n         to provide grids at the left
support line. Please redefine the number \n          (2<default>, 3,
4, 5, 6, 7, 8, 9, 10):  ');
             if isempty(fldiv)
            fldiv=2;
             else
                    while ~ismember(fldiv, [2 3 4 5 6 7 8 9 10])
                fldiv = input('\n              Please try an appropriate
value (or press Return key to exit):  ');
                if isempty(fldiv)
                    return;
                end
                    end
             end
       end
else
    lc=0;
end
%left support boundary condition
if lc==0
    if lsym==0
```

```
        lbc = input('\n           Boundary condition (1(free), 2(fix) or
3(simply supported <default>)):  ');
             if isempty(lbc)
            lbc=3;
             else
                    while ~ismember(lbc, [1 2 3])
                lbc = input('\n                Please try an appropriate
value (or press Return key to exit):  ');
                if isempty(lbc)
                    return;
                end
                    end
             end
       else
        disp(sprintf('\n       Boundary condition: roller'));
        lbc=0; %roller support
       end
else
    lbc=3;
end
disp(sprintf('\n       Right support'));
if rsym==0    %right side not a symmetric axis
    %support location
       rc = input(sprintf('\n        Location (1 (right edge of the wall)
or\n                   2 (-100 from the right edge of the wall<default>)):
'));
       if isempty(rc)
        rc=Wlength-200+lc;
       else
             while ~ismember(rc, [1 2])
            rc = input('\n           Please try an appropriate value (or
press Return key to exit):  ');
            if isempty(rc)
                return;
            end
             end
        if rc==1
            rc=maxx;
        else
            rc=Wlength-200+lc;
        end
       end
       if (rc~=maxx & fldiv<2)
        fldiv = input('\n          The number of divisions of faceshell
length should be at least 2\n               to provide grids at the right
support line. Please redefine the number \n               (2<default>, 3,
4, 5, 6, 7, 8, 9, 10):  ');
             if isempty(fldiv)
            fldiv=2;
             else
                    while ~ismember(fldiv, [2 3 4 5 6 7 8 9 10])
                fldiv = input('\n               Please try an appropriate
value (or press Return key to exit):  ');
                if isempty(fldiv)
                    return;
                end
                    end
             end
       end
else
```

```
    rc=maxx;
end
%right support boundary condition
if rc==maxx
    if rsym==0
        rbc = input('\n         Boundary condition (1(free), 2(fix) or
3(simply supported <default>)):  ');
             if isempty(rbc)
            rbc=3;
             else
                    while ~ismember(rbc, [1 2 3])
                rbc = input('\n               Please try an appropriate
value (or press Return key to exit):  ');
                if isempty(rbc)
                    return;
                end
                    end
             end
      else
        disp(sprintf('\n        Boundary condition: roller'));
        rbc=0; %roller support
      end
else
    rbc=3;
end
disp(sprintf('\n       Bottom support'));
%bottom support boundary condition
if bsym==0
    bbc = input('\n       Boundary condition (1(free), 2(fix), 3(out-of-
plane simply supported)\n              or 4(in-plane simply supported
<default>)):  ');
      if isempty(bbc)
        bbc=4;
      else
             while ~ismember(bbc, [1 2 3 4])
            bbc = input('\n              Please try an appropriate value
(or press Return key to exit):  ');
            if isempty(bbc)
                return;
            end
             end
      end
else
    disp(sprintf('\n       Boundary condition: roller'));
    bbc=0; %roller support
end
disp(sprintf('\n       Top support'));
%top support boundary condition
if tsym==0
    tbc = input('\n       Boundary condition (1(free), 2(fix), 3(out-of-
plane simply supported <default>)\n              or 4(in-plane simply
supported)):  ');
      if isempty(tbc)
        tbc=3;
      else
             while ~ismember(tbc, [1 2 3 4])
            tbc = input('\n              Please try an appropriate value
(or press Return key to exit):  ');
            if isempty(tbc)
                return;
            end
              end
        end
else
    disp(sprintf('\n       Boundary condition: roller'));
    tbc=0; %roller support
end
alldata=1;
```

A.5.3 Subroutine homoModel.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The script file homoModel.m is a subroutine of BMFEM.m to create a
homogeneous
% model of block masonry wall, where the block and mortar are not
discretized
% separately and the combined material is used. It contains the following
% subroutines:
%       fblkGrid.m  -- define grids in a full block unit
%       hblkGrid.m  -- define grids in a half block unit
%       webGrid.m   -- define grids in a single web
%       fblkElem.m  -- define elements in a full block unit
%       hblkElem.m  -- define elements in a half block unit
%       webElem.m   -- define elements in a single web
%
% Author: Zhong He
% Supervised by: Dr. Samir E. Chidiac and Dr. Robert G. Drysdale
% Version of 18-Nov-2004 23:58:30.
% Updated: 08-Dec-2004 20:35:38
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%% DEFINE GRIDS %%%%%%%%%%%%%%%%%

%%define grids in a full block
fblkGrid;

%%define grids in one course
blkGnum=size(BGRID,1);
%grids in a course without openning
CGRID=BGRID;
for i=1:fullxnum-1
    CGRID=[CGRID;   BGRID(:,1)+i*blkGnum   BGRID(:,2)+i*tLb   BGRID(:,3)
BGRID(:,4)];
end

%%Last web or half block to complete the course
coubGnum=size(CGRID,1); % number of grids in the created full blocks in
one course
if mod(Wlength,tLb)==0  %last web
    webGrid;
    CGRID=[CGRID;      BGRIDW(:,1)+coubGnum      BGRIDW(:,2)+fullxnum*tLb
BGRIDW(:,3) BGRIDW(:,4)];
else if mod(Wlength,tLb)==tLb/2 %half block
        hblkGrid;
        CGRID=[CGRID;    BGRIDH(:,1)+coubGnum    BGRIDH(:,2)+fullxnum*tLb
BGRIDH(:,3) BGRIDH(:,4)];
     end
```

```
end

%%grids in the whole wall
couGnum=size(CGRID,1); %number of grids in one course
WGRID=CGRID;
for k=1:fullznum-1
    WGRID=[WGRID;     CGRID(:,1)+k*couGnum      CGRID(:,2)      CGRID(:,3)
CGRID(:,4)+k*tHb];
end

%%%%%%%%%%%%%%%% DEFINE ELEMENTS %%%%%%%%%%%%%%%%%

%%define elements in a full block
fblkElem;

%elements in one course of the wall
blkEnum=size(BELEM,1);
%elements in a course without openning
CELEM=BELEM;
for i=1:fullxnum-1
    CELEM=[CELEM;        BELEM(:,1)+i*blkEnum         BELEM(:,2)+i*blkGnum
BELEM(:,3)+i*blkGnum BELEM(:,4)+i*blkGnum......
                  BELEM(:,5)+i*blkGnum                BELEM(:,6)+i*blkGnum
BELEM(:,7)+i*blkGnum BELEM(:,8)+i*blkGnum......
                  BELEM(:,9)+i*blkGnum      BELEM(:,10)       BELEM(:,11)
BELEM(:,12)];
end

%elements in the last web or half block to complete the course
LELEM=[];
ei=0;
if mod(Wlength,tLb)==0  %last web
    webElem;
    CELEM=[CELEM;      BELEMW(:,1)+size(CELEM,1)      BELEMW(:,2)+coubGnum
BELEMW(:,3)+coubGnum BELEMW(:,4)+coubGnum......
                  BELEMW(:,5)+coubGnum                BELEMW(:,6)+coubGnum
BELEMW(:,7)+coubGnum BELEMW(:,8)+coubGnum......
                  BELEMW(:,9)+coubGnum      BELEMW(:,10)      BELEMW(:,11)
BELEMW(:,12)];
else if mod(Wlength,tLb)==200 %half block
        hblkElem;
        CELEM=[CELEM;    BELEMH(:,1)+size(CELEM,1)    BELEMH(:,2)+coubGnum
BELEMH(:,3)+coubGnum BELEMH(:,4)+coubGnum......
                      BELEMH(:,5)+coubGnum            BELEMH(:,6)+coubGnum
BELEMH(:,7)+coubGnum BELEMH(:,8)+coubGnum......
                      BELEMH(:,9)+coubGnum    BELEMH(:,10)    BELEMH(:,11)
BELEMH(:,12)];
     end
end

%%elements in the whole wall
couEnum=size(CELEM,1); %number of elements in one course
WELEM=CELEM;
for k=1:fullznum-1
    WELEM=[WELEM;        CELEM(:,1)+k*couEnum         CELEM(:,2)+k*couGnum
CELEM(:,3)+k*couGnum CELEM(:,4)+k*couGnum......
                  CELEM(:,5)+k*couGnum                CELEM(:,6)+k*couGnum
CELEM(:,7)+k*couGnum CELEM(:,8)+k*couGnum......
                  CELEM(:,9)+k*couGnum      CELEM(:,10)       CELEM(:,11)
CELEM(:,12)];
```

```
end
```

A.5.4 Subroutine heterModel.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The script file heterModel.m is a subroutine of BMFEM.m to create a
heterogeneous
% model of block masonry wall, where the block and mortar are discretized
% separately and the different materials are used. Currently it is
desinged for
% running bond pattern block wall. It contains the following subroutines:
%      blayerGrid.m  -- define grids in the block layer in an odd course
%      mlayerGrid.m  -- define grids in the mortar layer in an odd course
%      blayerGrid2.m -- define grids in the block layer in an even course
%        mlayerGrid2.m -- define grids in the mortar layer in an even
course
%
%       blayerElem.m   -- define elements in the block layer in an odd
course
%       mlayerElem.m   -- define elements in the mortar layer in an odd
course
%       blayerElem2.m -- define elements in the block layer in an even
course
%       mlayerElem2.m -- define elements in the mortar layer in an even
course
%
%      solidGE.m     -- Form grids and elements in the whole wall
%
% Author: Zhong He
% Supervised by: Dr. Samir E. Chidiac and Dr. Robert G. Drysdale
% Version of 18-Nov-2004 23:58:30.
% Updated: 08-Dec-2004 20:35:38
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%% DEFINE GRIDS %%%%%%%%%%%%%%%%%
%%%%%Odd Course%%%%%%
%%block layer
blayerGrid;

%%mortar layer
if fullznum > 1
    mlayerGrid;
end

%%%%Even Courses%%%%%
%block layer
if fullznum > 1
    blayerGrid2;
end

%%mortar layer
if fullznum > 2
    mlayerGrid2;
end

% %%%%%%%%%%%%%%%% DEFINE ELEMENTS %%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%Odd Layer%%%%%%%%%%%%
%%block layer
m=1; %material group
blayerElem;

%%mortar layer
if fullznum > 1
       m=2;
       mlayerElem;
end

%%%%%%%Even Layer%%%%%%%%%%%%
%block layer
if fullznum > 1
       m=1; %material group
       blayerElem2;
end
%%mortar layer
if fullznum > 2
       m=2;
       mlayerElem2;
end

WGRID=CGRIDa;
WELEM=CELEMa;

%Form grids and elements in the whole wall
solidGE;
```

A.5.5 Subroutine formOpenning.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The script file formOpenning.m is a subroutine of BMFEM.m to process
openings,
% including the adjustment of grids and elements, load transfermation and
modeling
% of bond beams. For now, bond beams are considered only for opennings
with
% modular dimensions.
% It contains the following subroutines:
%       adjustGE.m      -- adjust grids and elements on the opennings
%        reorderGE.m     -- modify the order of the grids for 6-node wedge
elements
%       openningLoad.m -- generate point loads transferred to the edge of
the
%                          opennings
%       hobondbeam.m    -- create bond beams in the homogeneous models
%       hebondbeam.m    -- create bond beams in the heterogeneous models
%
% Author: Zhong He
% Supervised by: Dr. Samir E. Chidiac and Dr. Robert G. Drysdale
% Version of 18-Nov-2004 23:58:30.
% Updated: 08-Dec-2004 20:35:38
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

disp(sprintf('\nProcessing opennings at %0.1f', etime(clock,t0)));
```

```
TGRID=WGRID;
TELEM=WELEM;
triELEM=[];
PL=[]; %point load at openning edges
for oi=1:size(xopen,2)
    %if  the  input  openning  dimension  are  modules  (eg.  multiples  of
200(half
    %block  size),  they  are  converted  to  the  corresponding  half  block
model size
    %(209(homogeneous model) or 214(heterogeneous model))
    left=xopen(oi);
    if mod(left,tLb)==tLb/2
        left=left-tLb/2+2*tw+Lf;
    end
    right=xopen(oi)+wopen(oi);
    if mod(right,tLb)==tLb/2
        right=right-tLb/2+2*tw+Lf;
    else if (mod(right,tLb)==0 & right==Wlength)
            right=right+tw;
        end
    end
    bottom=zopen(oi);
    if (micro==1 & mod(bottom,tHb)==0 & bottom~=0)
        bottom=bottom-tm;
    end
    top=zopen(oi)+hopen(oi);
    if (micro==1 & top==Wheight)
        top=top-tm;
    end

    if micro==0
        if (mod(left,tLb)==2*tw+Lf)
            for i=1:size(TGRID,1)
                if TGRID(i,2)==left-tgap/2
                    TGRID(i,2)=left;
                end
            end
            for i=1:size(WGRID,1)
                if WGRID(i,2)==left-tgap/2
                    WGRID(i,2)=left;
                end
            end
        end
        if (mod(right,tLb)==2*tw+Lf)
            for i=1:size(TGRID,1)
                if TGRID(i,2)==right+tgap/2
                    TGRID(i,2)=right;
                end
            end
            for i=1:size(WGRID,1)
                if WGRID(i,2)==right+tgap/2
                    WGRID(i,2)=right;
                end
            end
        end
    end

    if oi>1
        TGRID=GRID;
        TELEM=ELEM;
    end
    disp(sprintf('\nOpenning location identified at %0.1f.',
etime(clock,t0)));
    GRID=[]; ELEM=[];
    INGRID=[]; LEGRID=[]; REGRID=[]; BEGRID=[]; TEGRID=[];
    %determine the grids inside and at the edges of the openning
    inGi=find((TGRID(:,2)>left) & (TGRID(:,2)<right) &
(TGRID(:,4)>bottom) & (TGRID(:,4)<top));
    INGRID=TGRID(inGi,:);
    leGi=find((TGRID(:,2)==left) & (TGRID(:,4)>=bottom) &
(TGRID(:,4)<=top));
    LEGRID=TGRID(leGi,:);
    reGi=find((TGRID(:,2)==right) & (TGRID(:,4)>=bottom) &
(TGRID(:,4)<=top));
    REGRID=TGRID(reGi,:);
    beGi=find((TGRID(:,4)==bottom) & (TGRID(:,2)>=left) &
(TGRID(:,2)<=right));
    BEGRID=TGRID(beGi,:);
    teGi=find((TGRID(:,4)==top) & (TGRID(:,2)>=left) &
(TGRID(:,2)<=right));
    TEGRID=TGRID(teGi,:);
    disp(sprintf('\nGrids inside or at the openning edges identified at
%0.1f.', etime(clock,t0)));

    %determine the elements containing some or all the grids determined
above
    OGRID=[INGRID; LEGRID; REGRID; BEGRID; TEGRID];
    if isempty(LEGRID)
        LEGRID=zeros(1,4);
    end
    if isempty(REGRID)
        REGRID=zeros(1,4);
    end
    if isempty(BEGRID)
        BEGRID=zeros(1,4);
    end
    if isempty(TEGRID)
        TEGRID=zeros(1,4);
    end

    eagi=find(ismember(TELEM(:,2),OGRID(:,1)) |
ismember(TELEM(:,3),OGRID(:,1)) |......
              ismember(TELEM(:,4),OGRID(:,1)) |
ismember(TELEM(:,5),OGRID(:,1)) |......
              ismember(TELEM(:,6),OGRID(:,1)) |
ismember(TELEM(:,7),OGRID(:,1)) |......
              ismember(TELEM(:,8),OGRID(:,1)) |
ismember(TELEM(:,9),OGRID(:,1)) );
    OELEM=TELEM(eagi,:);
    disp(sprintf('\nElements associated with the grids identified at
%0.1f.', etime(clock,t0)));

    %identify the elements inside and on the edges of the openning
    inEi=find(ismember(OELEM(:,2),OGRID(:,1)) &
ismember(OELEM(:,3),OGRID(:,1)) &......
              ismember(OELEM(:,6),OGRID(:,1)) &
ismember(OELEM(:,7),OGRID(:,1)) );
    INELEM=OELEM(inEi,:);
    onEi=find(~ismember(find(OELEM(:,1)),inEi));
    ONELEM=OELEM(onEi,:);
    disp(sprintf('\nElements inside and on the openning identified at
%0.1f.', etime(clock,t0)));

    %delete all the grids inside the openning
       if ~isempty(INGRID)
        dgin=find(ismember(TGRID(:,1),INGRID(:,1)));
        TGRID(dgin,:)=[];
        GRID=TGRID;
       end

    %delete all the elements inside the openning
       if ~isempty(INELEM)
        dein=find(ismember(TELEM(:,1),INELEM(:,1)));
        TELEM(dein,:)=[];
        ELEM=TELEM;
       end
    disp(sprintf('\nGrids and elements inside the openning deleted at
%0.1f.', etime(clock,t0)));
    %adjust the nodes and elements at the edges of the openning (8 cases)
    NEWGRID=[];
    NEWELEM=[];
    MDFELEM=ONELEM; %modified elements at the edges

    adjustGE;
    disp(sprintf('\nGrids and elements on the opening edges adjusted at
%0.1f.', etime(clock,t0)));

    %form grid matrix
    GRID=[GRID; NEWGRID];
    %rearrange the order of grids in the modified and new-added
triangular-type elements
    [triMDE, MDFELEM]=reorderGE(GRID, MDFELEM);
    [triNEWE, NEWELEM]=reorderGE(GRID, NEWELEM);
    triELEM=[triELEM; triMDE; triNEWE];
    disp(sprintf('\nGrid order in triangular elements rearranged at
%0.1f.', etime(clock,t0)));
    %form element matrix
    for i=1:size(ELEM,1)
        for j=1:size(MDFELEM,1)
            if ELEM(i,1)==MDFELEM(j,1)
                ELEM(i,:)=MDFELEM(j,:);
            end
        end
    end
    ELEM=[ELEM; NEWELEM];

    %update total grids and elements (including the grids inside the
openning
    WGRID=[WGRID; NEWGRID];
    WELEM=[WELEM; NEWELEM];

    %calculate point loads applied at the edge of the openning
    openningLoad;
    disp(sprintf('\nPoint loads on the openning edges calculated at
%0.1f.', etime(clock,t0)));

    %define grids and elements in bond beam
    if (bbond(oi)==1 | tbond(oi)==1)
        if micro==0
            hobondbeam;
        else
            hebondbeam;
        end
        CBBGRID(:,1)=CBBGRID(:,1)+size(WGRID,1);
        CBBELEM(:,1)=CBBELEM(:,1)+size(WELEM,1);
        CBBELEM(:,2:9)=CBBELEM(:,2:9)+size(WGRID,1);
        GRID=[GRID; CBBGRID];
        ELEM=[ELEM; CBBELEM];
        WGRID=[WGRID; CBBGRID];
        WELEM=[WELEM; CBBELEM];
    end
end

%accumulate point loads at the duplicated grids and delete the duplicated
loading
dub=[];
for i=1:size(PL-1,1)
    if ~ismember(i,dub)
        for j=i+1:size(PL,1)
            if PL(j,1)==PL(i,1)
                PL(i,3)=PL(i,3)+PL(j,3);
                dub=[dub j];
            end
        end
    end
end
[PLG,pli]=mergeRows(PL(:,1));
PL=PL(pli,:);
disp(sprintf('\nDuplicated    point    loads    deleted    at    %0.1f.',
etime(clock,t0)));
```

A.5.6 Subroutine mergeGRID.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The script file mergeGRID.m is a subroutine of BMFEM.m to do merging of
% the grids and modify the grids to keep the same connectivity of
elements
%
% Author: Zhong He
% Supervised by: Dr. Samir E. Chidiac and Dr. Robert G. Drysdale
% Version of 18-Nov-2004 23:58:30.
% Updated: 08-Dec-2004 20:35:38
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[uG, uGi, inuG]=mergeRows(GRID(:,2:4));
MGRID=GRID(uGi,:); %merged grids
mapG=[GRID(:,1) MGRID(inuG,1)]; %create mapping between original grids
and the merged grids
GRID=MGRID;
%change the duplicated grids to the unique grids
for j=2:9
    %find grids to be modified
    chGEi=find(~ismember(ELEM(:,j), MGRID(:,1)));
    chEG=ELEM(chGEi,j);
    %find the corresponding unique grids
    [sortchEG,si]=sort(chEG);
    mi=find(ismember(mapG(:,1),chEG)); %find the index of grids to be
modified in the mapping matrix
    chEG=mapG(mi,2);
    [sortsi,sii]=sort(si);
    chEG=chEG(sii);
    %change the duplicated grids to the unique grids
    ELEM(chGEi,j)=chEG;
end

if (openning & ~isempty(triELEM))
    for j=2:9
        tchGEi=find(~ismember(triELEM(:,j), MGRID(:,1))); %find grids to
be modified
        tchEG=triELEM(tchGEi,j);
        %find the corresponding unique grids
        [sorttchEG,tsi]=sort(tchEG);
        tmi=find(ismember(mapG(:,1),tchEG)); %find the index of grids to
be modified in the mapping matrix
        tchEG=mapG(tmi,2);
        [sorttsi,tsii]=sort(tsi);
        tchEG=tchEG(tsii);
        %change the duplicated grids to the unique grids
        triELEM(tchGEi,j)=tchEG;
    end
end
```

A.5.7 Subroutine renumberGE.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The script file renumberGE.m is a subroutine of BMFEM.m to do
renumberring of
% the grids and elements so that they are odered consecutively
%
% Author: Zhong He
% Supervised by: Dr. Samir E. Chidiac and Dr. Robert G. Drysdale
% Version of 18-Nov-2004 23:58:30.
% Updated: 08-Dec-2004 20:35:38
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[rnGRID,rni]=sortrows(GRID(:,2:4));
rnGRID=[find(GRID(:,1)) rnGRID];
%create mapping between original(merged) grids and the renumberred grids
[sortrni,rnii]=sort(rni);
mapRG=[GRID(:,1) rnii];

%change the grids in element matrix to the renumberred grids
rei=find(ELEM(:,1));
for j=2:9
    rEG=ELEM(:,j); %grids need to be changed
    rEGi=rei;
    for k=1:2
        if (openning & ~isempty(triELEM))
            if k==1
                [rEG,rEGi]=mergeRows(rEG);
            else
                rEGi=find(~ismember(rei(:,1),rEGi));
                rEG=ELEM(rEGi,j);
            end
        end
        %find the corresponding renumberred grids
        [sortrEG,ri]=sort(rEG);
        mi=find(ismember(mapRG(:,1),rEG)); %find the index of grids to be
modified in the mapping matrix
        rEG=mapRG(mi,2);
        [sortri,rii]=sort(ri);
        rEG=rEG(rii);
        %change the grids to the renumerred grids
        ELEM(rEGi,j)=rEG;
        if ~openning | isempty(triELEM)
            break;
        end
    end
end
mapRE=[ELEM(:,1)     find(ELEM(:,1))];     %create     mapping     between
original(merged) elements and the renumberred elements

if (openning & ~isempty(triELEM))
    for j=2:9
        trEG=triELEM(:,j);
        %find the corresponding renumberred grids
        [sorttrEG,tri]=sort(trEG);
        tmi=find(ismember(mapRG(:,1),trEG)); %find the index of grids to
be modified in the mapping matrix
        trEG=mapRG(tmi,2);
        [sorttri,trii]=sort(tri);
        trEG=trEG(trii);
        %change the grids to the renumberred grids
        triELEM(:,j)=trEG;
    end

    %renumber the wedge elements
    tei=find(ismember(mapRE(:,1),triELEM(:,1)));
    triELEM(:,1)=mapRE(tei,2);
end

%renumber the grids where point load applied
if openning
       rPL=PL(:,1);
       [sortPL,pli]=sort(rPL);
       pmi=find(ismember(mapRG(:,1),rPL));
       rPL=mapRG(pmi,2);
       [sortpli,plii]=sort(pli);
       rPL=rPL(plii);
       PL(:,1)=rPL;
end

GRID=rnGRID; %renumber the girds
ELEM(:,1)=mapRE(:,2); %renumber the elements
```

A.5.8 Subroutine constraint.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The script file constraint.m is a subroutine of BMFEM.m to generate the
% constraint information of the block wall FE model
%
% Author: Zhong He
% Supervised by: Dr. Samir E. Chidiac and Dr. Robert G. Drysdale
% Version of 18-Nov-2004 23:58:30.
% Updated: 08-Dec-2004 20:35:38
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

tc=max(GRID(:,4));
%identify the constraint points
%left constraint
LCGRID=GRID(find(GRID(:,2)==lc),:);
% [LCGu,lci]=mergeRows(LCGRID(:,2:4));
% LCGRID=LCGRID(lci,:);
if lbc==0 %roller
    LCGRID=LCGRID(find(LCGRID(:,3)<=tf | LCGRID(:,3)>=tf+Lw),:);
    lcri=size(LCGRID,1);
    LCON=[LCGRID(:,1) ones(lcri,1) zeros(lcri,1) zeros(lcri,1)];
else if lbc==1 %free
        LCON=[];
    else if lbc==2 %fix
        lcri=size(LCGRID,1);
        LCON=[LCGRID(:,1) ones(lcri,1) ones(lcri,1) ones(lcri,1)];
    else if lbc==3 %simply supported
              LCGRID=LCGRID(find(abs(LCGRID(:,3)-Wb)<1E-3),:);
              lcri=size(LCGRID,1);
              LCON=[LCGRID(:,1) zeros(lcri,1) ones(lcri,1) zeros(lcri,1)];
          end
    end
    end
end
%right constraint
RCGRID=GRID(find(GRID(:,2)==rc),:);
% [RCGu,rci]=mergeRows(RCGRID(:,2:4));
% RCGRID=RCGRID(rci,:);
if rbc==0 %roller
    RCGRID=RCGRID(find(RCGRID(:,3)<=tf | RCGRID(:,3)>=tf+Lw),:);
    rcri=size(RCGRID,1);
    RCON=[RCGRID(:,1) ones(rcri,1) zeros(rcri,1) zeros(rcri,1)];
else if rbc==1 %free
        RCON=[];
    else if rbc==2 %fix
        rcri=size(RCGRID,1);
        RCON=[RCGRID(:,1) ones(rcri,1) ones(rcri,1) ones(rcri,1)];
    else if rbc==3 %simply supported
              RCGRID=RCGRID(find(abs(RCGRID(:,3)-Wb)<1E-3),:);
              rcri=size(RCGRID,1);
              RCON=[RCGRID(:,1) zeros(rcri,1) ones(rcri,1) zeros(rcri,1)];
          end
    end
    end
end
%bottom constraint
BCGRID=GRID(find(GRID(:,4)==0),:);
% [BCGu,bci]=mergeRows(BCGRID(:,2:4));
% BCGRID=BCGRID(bci,:);
if bbc==0
    bcri=size(BCGRID,1);
    BCON=[BCGRID(:,1) zeros(bcri,1) zeros(bcri,1) ones(bcri,1)];
else if bbc==1 %free
```

```
        BCON=[];
    else if bbc==2 %fix
        bcri=size(BCGRID,1);
        BCON=[BCGRID(:,1) ones(bcri,1) ones(bcri,1) ones(bcri,1)];
    else if bbc==3 %out-of-plane simply supported
             BCGRID=BCGRID(find(abs(BCGRID(:,3)-Wb)<1E-3),:);
             bcri=size(BCGRID,1);
             BCON=[BCGRID(:,1) zeros(bcri,1) ones(bcri,1) zeros(bcri,1)];
         else if bbc==4 %in-plane simply supported
                  BCGRID=BCGRID(find(BCGRID(:,3)==Wb/2),:);
                  bcri=size(BCGRID,1);
                  BCON=[BCGRID(:,1)        ones(bcri,1)        ones(bcri,1)
ones(bcri,1)];
              end
         end
    end
    end
end
%top constraint
TCGRID=GRID(find(GRID(:,4)==tc),:);
% [TCGu,tci]=mergeRows(TCGRID(:,2:4));
% TCGRID=TCGRID(tci,:);
if tbc==0 %roller
    tcri=size(TCGRID,1);
    TCON=[TCGRID(:,1) zeros(tcri,1) zeros(tcri,1) ones(tcri,1)];
else if tbc==1 %free
        TCON=[];
    else if tbc==2 %fix
        tcri=size(TCGRID,1);
        TCON=[TCGRID(:,1) ones(tcri,1) ones(tcri,1) ones(tcri,1)];
    else if tbc==3 %out-of-plane simply supported
             TCGRID=TCGRID(find(abs(TCGRID(:,3)-Wb)<1E-3),:);
             tcri=size(TCGRID,1);
             TCON=[TCGRID(:,1) zeros(tcri,1) ones(tcri,1) zeros(tcri,1)];
         else if tbc==4 %in-plane simply supported
                  TCGRID=TCGRID(find(TCGRID(:,3)==Wb/2),:);
                  tcri=size(TCGRID,1);
                  TCON=[TCGRID(:,1)        ones(tcri,1)        ones(tcri,1)
ones(tcri,1)];
              end
         end
    end
    end
end
%top left corner mount
if (lsym==0 & tsym==0 & ~(lbc==1 & tbc==1))
    %when not both left and top side on symmetric axis, and not both left
    %and top side are free, the top left corner mount is set
    TLCON=[GRID(find(GRID(:,2)==lc & GRID(:,3)==0 & GRID(:,4)==tc),1) 0 1
0];
    if size(TLCON,2)==4
        if ~isempty(LCON)
            [TL,i,j]=intersect(LCON(:,1),TLCON(:,1));
            if ~isempty(TL)
                LCON(i,2:4)=LCON(i,2:4) | TLCON(j,2:4);
            else
                LCON=[LCON; TLCON];
            end
        else
            LCON=TLCON;
        end
    end
end
%bottom left shear key
if (lsym==0 & bsym==0 & ~(lbc==1 & bbc==1))
    %when not both left and bottom side on symmetric axis, and not both
left
    %and bottom side are free, the bottom left corner mount is set
    BLCON=[GRID(find(GRID(:,2)==0 & GRID(:,3)==0 & GRID(:,4)==0),1) 0 1
0];
    if size(BLCON,2)==4
        if ~isempty(LCON)
            [BL,i,j]=intersect(LCON(:,1),BLCON(:,1));
            if ~isempty(BL)
                LCON(i,2:4)=LCON(i,2:4) | BLCON(j,2:4);
            else
                LCON=[LCON; BLCON];
            end
        else
            LCON=BLCON;
        end
    end
end
%top right corner mount
if (rsym==0 & tsym==0 & ~(rbc==1 & tbc==1))
    %when not both right and top side on symmetric axis, and not both
right
    %and top side are free, the top right corner mount is set
    TRCON=[GRID(find(GRID(:,2)==rc & GRID(:,3)==0 & GRID(:,4)==tc),1) 0 1
0];
    if size(TRCON,2)==4
        if ~isempty(RCON)
            [TR,i,j]=intersect(RCON(:,1),TRCON(:,1));
            if ~isempty(TR)
                RCON(i,2:4)=RCON(i,2:4) | TRCON(j,2:4);
            else
                RCON=[RCON; TRCON];
            end
        else
            RCON=TRCON;
        end
    end
end
%bottom right shear key
if (rsym==0 & bsym==0 & ~(rbc==1 & bbc==1))
    %when not both right and bottom side on symmetric axis, and not both
right
    %and bottom side are free, the bottom right corner mount is set
    BRCON=[GRID(find(GRID(:,2)==maxx & GRID(:,3)==0 & GRID(:,4)==0),1) 0
1 0];
    if size(BRCON,2)==4
        if ~isempty(RCON)
            [BR,i,j]=intersect(RCON(:,1),BRCON(:,1));
            if ~isempty(BR)
                RCON(i,2:4)=RCON(i,2:4) | BRCON(j,2:4);
            else
                RCON=[RCON; BRCON];
            end
        else
            RCON=BRCON;
        end
    end
end
%merge the constraint information for the same points
%common points in the left constraint and the top constraint
if ~isempty(LCON) & ~isempty(TCON)
       [LT,i,j]=intersect(LCON(:,1),TCON(:,1),'rows');
       LCON(i,2:4)=LCON(i,2:4) | TCON(j,2:4);
       TCON(j,:)=[];
end
%common points in the left constraint and the bottom constraint
if ~isempty(LCON) & ~isempty(BCON)
       [LB,i,j]=intersect(LCON(:,1),BCON(:,1),'rows');
       LCON(i,2:4)=LCON(i,2:4) | BCON(j,2:4);
       BCON(j,:)=[];
end
%common points in the right constraint and the top constraint
if ~isempty(RCON) & ~isempty(TCON)
       [RT,i,j]=intersect(RCON(:,1),TCON(:,1),'rows');
       RCON(i,2:4)=RCON(i,2:4) | TCON(j,2:4);
       TCON(j,:)=[];
end
%common points in the right constraint and the bottom constraint
if ~isempty(RCON) & ~isempty(BCON)
       [RB,i,j]=intersect(RCON(:,1),BCON(:,1),'rows');
       RCON(i,2:4)=RCON(i,2:4) | BCON(j,2:4);
       BCON(j,:)=[];
end
```

A.5.9 Subroutine loading.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The script file loading.m is a subroutine of BMFEM.m to generate the
%  loading  information  of  the  block  wall  FE  model.  Currently  it  is
designed
% to generate the pressure load applied to the surface of the wall
%
% Author: Zhong He
% Supervised by: Dr. Samir E. Chidiac and Dr. Robert G. Drysdale
% Version of 18-Nov-2004 23:58:30.
% Updated: 08-Dec-2004 20:35:38
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Add pressure load
%identify the grids on the face where pressure loads apply
fGRID=GRID(find(GRID(:,3)==0),1);
%identify the rectangular elements on the face
fEi1=find(ismember(ELEM(:,2),fGRID)    &    ismember(ELEM(:,3),fGRID)    &
ismember(ELEM(:,6),fGRID));
EP1=ELEM(fEi1, 1);
EP1=[EP1 2*ones(size(EP1,1),1)];
%identify the triangular elements on the face
fEi2=find(ismember(ELEM(:,2),fGRID)    &    ismember(ELEM(:,3),fGRID)    &
ismember(ELEM(:,4),fGRID));
EP2=ELEM(fEi2, 1);
EP2=[EP2 ones(size(EP2,1),1)];
```

```
EP=[EP1; EP2];
EP=[(1:size(EP,1))'    EP(:,1)    EP(:,1)    ones(size(EP,1),1)    EP(:,2)
P*ones(size(EP,1),1)];
```

A.5.10 Subroutine writeAFEMS.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The script file writeAFEMS.m is a subroutine of BMFEM.m to generate the
% input data file for AFEMS finite element analysis package
%
% Author: Zhong He
% Supervised by: Dr. Samir E. Chidiac and Dr. Robert G. Drysdale
% Version of 18-Nov-2004 23:58:30.
% Updated: 08-Dec-2004 20:35:38
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fid1=fopen(file1,'w');
if micro==0
    % homogeneous model
       % print material information to the input file
       fprintf(fid1,'%s\n','mg,1,');
    fprintf(fid1,'%s,%i,\n','EX',mvalue(1,1)*1E3); %unit is MPa*10^-3 or
KPa
    fprintf(fid1,'%s,%i,\n','EY',mvalue(1,2)*1E3);
    fprintf(fid1,'%s,%i,\n','EZ',mvalue(1,3)*1E3);
       fprintf(fid1,'%s,%0.3f,\n','NUXY',mvalue(1,4));
    fprintf(fid1,'%s,%0.3f,\n','NUXZ',mvalue(1,5));
    fprintf(fid1,'%s,%0.3f,\n','NUYZ',mvalue(1,6));
    fprintf(fid1,'%s,%i,\n','GXY',mvalue(1,7)*1E3); %unit is MPa*10^-3 or
KPa
    fprintf(fid1,'%s,%i,\n','GXZ',mvalue(1,8)*1E3);
    fprintf(fid1,'%s,%i,\n','GYZ',mvalue(1,9)*1E3);
    leftPt = GRID(find(GRID(:,2)==0&GRID(:,3)==0),1);
    fprintf(fid1,'%s,%i,%s\n','ORTD,1,1,1,2',leftPt(2,1),'1,');
       fprintf(fid1,'%s\n','RG,1,');
       fprintf(fid1,'%s,%0.2E,\n','MASS',mvalue(1,10));
       fprintf(fid1,'%s\n','DEAD_LOAD,0.00E+00,0.00E+00,-9.806E3,');

       % print the grid information
       for i=1:gnum

fprintf(fid1,'%s,%i,%0.1f,%0.1f,%0.1f,','G',GRID(i,1),GRID(i,2),GRID(i,3)
,GRID(i,4));
        fprintf(fid1,'\n');
       end

       % print the masonry element information
       for j=1:max(ELEM(:,10))
        fprintf(fid1,'%s,%i,%s,\n','ELEM_GROUP',j,'3D8');
        fprintf(fid1,'%s\n','EO,ON,FULL,OFF,');
        for i=1:enum
            if ELEM(i,10)==j
                fprintf(fid1,'%s,','E');
                for k=1:9
                    fprintf(fid1,'%i,',ELEM(i,k));
                end
                fprintf(fid1,'\n');
            end
        end
       end
else
    %% heterogeneous model
    % define material axes
    leftPt = GRID(find(GRID(:,2)==0&GRID(:,3)==0),1);
    fprintf(fid1,'%s,%i,%s\n','ORTD,1,1,1,2',leftPt(2,1),'1,');

       % print the grid information
       for i=1:gnum

fprintf(fid1,'%s,%i,%0.1f,%0.1f,%0.1f,','G',GRID(i,1),GRID(i,2),GRID(i,3)
,GRID(i,4));
        fprintf(fid1,'\n');
       end

       % print the element information
       for j=1:max(ELEM(:,10))
        fprintf(fid1,'%s,%i,%s,\n','ELEM_GROUP',j,'3D8');
        fprintf(fid1,'%s\n','EO,ON,FULL,OFF,');
        for m=1:2
            if ~(j==2 & m==2)
                fprintf(fid1,'%s,%i,\n','mg',m);
                fprintf(fid1,'%s,%i,\n','EX',mvalue(m,1)*1E3);  %unit  is
MPa*10^-3 or KPa
                fprintf(fid1,'%s,%0.3f,\n','NUXY',mvalue(m,2));
                fprintf(fid1,'%s,%i,\n','RG',m);
                fprintf(fid1,'%s,%0.2E,\n','MASS',mvalue(m,3));
                fprintf(fid1,'%s\n','DEAD_LOAD,0.00E+00,0.00E+00,-
9.806E3,');
                for i=1:enum
                    if (ELEM(i,10)==j & ELEM(i,11)==m)
                        fprintf(fid1,'%s,','E');
                        for k=1:9
                            fprintf(fid1,'%i,',ELEM(i,k));
                        end
                        fprintf(fid1,'\n');
                    end
                end
            end
        end
       end
end

%write the constraint information to the input file for AFEMS
if ~isempty(LCON)
       fprintf(fid1,'%s\n','CC LEFT');
       for i=1:size(LCON,1)
        fprintf(fid1,'%s,%i,','CONS_POINT',LCON(i,1));
        for j=2:size(LCON,2)
            if LCON(i,j)==0
                fprintf(fid1,'%s,','free');
            else
                fprintf(fid1,'%s,','fix');
            end
        end
        fprintf(fid1,'\n');
       end
```

```
end
if ~isempty(RCON)
        fprintf(fid1,'%s\n','CC RIGHT');
        for i=1:size(RCON,1)
         fprintf(fid1,'%s,%i,','CONS_POINT',RCON(i,1));
         for j=2:size(RCON,2)
             if RCON(i,j)==0
                 fprintf(fid1,'%s,','free');
             else
                 fprintf(fid1,'%s,','fix');
             end
         end
         fprintf(fid1,'\n');
        end
end
if ~isempty(BCON)
        fprintf(fid1,'%s\n','CC BOTTOM');
        for i=1:size(BCON,1)
         fprintf(fid1,'%s,%i,','CONS_POINT',BCON(i,1));
         for j=2:size(BCON,2)
             if BCON(i,j)==0
                 fprintf(fid1,'%s,','free');
             else
                 fprintf(fid1,'%s,','fix');
             end
         end
         fprintf(fid1,'\n');
        end
end
if ~isempty(TCON)
        fprintf(fid1,'%s\n','CC TOP');
        for i=1:size(TCON,1)
         fprintf(fid1,'%s,%i,','CONS_POINT',TCON(i,1));
         for j=2:size(TCON,2)
             if TCON(i,j)==0
                 fprintf(fid1,'%s,','free');
             else
                 fprintf(fid1,'%s,','fix');
             end
         end
         fprintf(fid1,'\n');
        end
end

%write loading data to the file
EP=[EP; size(EP,1)+1 EP(1,2) EP(1,3) EP(1,4) EP(1,5) EP(1,6)];
EP(1,6)=0;
for k=1:size(EP,1)
    fprintf(fid1,'%s,%i,%i,%i,%i,%i,%0.2E,\n','ELEM_PRESS',EP(k,1),
EP(k,2), EP(k,3), EP(k,4), EP(k,5), EP(k,6));
end

% Add point loads at the openings
if openning
        for k=1:size(PL,1)

fprintf(fid1,'%s,%i,%i,%0.2f,%i,%i,%i,%i,\n','LOAD_POINT',PL(k,1),
PL(k,2), PL(k,3), PL(k,4), PL(k,5), PL(k,6), PL(k,7));
        end
end
```

```
fprintf(fid1,'%s\n','ST,0,');
fprintf(fid1,'%s','REN,ON,');
fclose(fid1);
```

A.5.11 Subroutine writeDIANA.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The script file writeDIANA.m is a subroutine of BMFEM.m to generate the
% input data file for DIANA finite element analysis package
%
% Author: Zhong He
% Supervised by: Dr. Samir E. Chidiac and Dr. Robert G. Drysdale
% Version of 18-Nov-2004 23:58:30.
% Updated: 08-Dec-2004 20:35:38
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fid2=fopen(file2,'w');
% annotation of the data file
fprintf(fid2,'%-18s: %s\n','FEMGEN MODEL',fname);
fprintf(fid2,'%-18s: %s\n','ANALYSIS TYPE','Structural 3D');
if micro==0
    % homogeneous model
    fprintf(fid2,'%-18s: %s\n','MODEL DESCRIPTION','homogeneous');
else
    % heterogeneous model
    fprintf(fid2,'%-18s: %s\n','MODEL DESCRIPTION','heterogeneous');
end
% units
fprintf(fid2,'%s\n','''UNITS''');
fprintf(fid2,'%-9s%s\n','MASS','KG');
fprintf(fid2,'%-9s%s\n','LENGTH','MM');
fprintf(fid2,'%-9s%s\n','TIME','SEC');
fprintf(fid2,'%-9s%s\n','TEMPER','KELVIN');

% grids
fprintf(fid2,'%s\n','''COORDINATES''');
for i=1:gnum
    fprintf(fid2,'%i                               %0.1f             %0.1f
%0.1f',GRID(i,1),GRID(i,2),GRID(i,3),GRID(i,4));
    fprintf(fid2,'\n');
end

% elements
fprintf(fid2,'%s\n','''ELEMENTS''');
fprintf(fid2,'%s\n','CONNECTIVITY');
priELEM=ELEM; %prism 3D element
if (openning & ~isempty(triELEM))
    %remove the triangular 3D element from the element list
    trii=find(ismember(priELEM(:,1),triELEM(:,1)));
    priELEM(trii,:)=[];
    enum=size(priELEM,1);
    %modify the 8-node triangular 3D element to 6-node wedge element
    trinum=size(triELEM,1);
    wedgeE=[];
    for i=1:trinum
```

```
        wedgeE=[wedgeE; mergeRows(triELEM(i,2:9).').'];
    end
    wedgeE=[triELEM(:,1) wedgeE];
end
for i=1:enum
    fprintf(fid2,'%i     ',priELEM(i,1));
    fprintf(fid2,'%s  ','HX24L');
    for k=2:9
        fprintf(fid2,'%i ',priELEM(i,k));
    end
    fprintf(fid2,'\n');
end
if (openning & ~isempty(triELEM))
       for i=1:trinum
        fprintf(fid2,'%i     ',wedgeE(i,1));
        fprintf(fid2,'%s  ','TP18L');
        for k=2:7
            fprintf(fid2,'%i ',wedgeE(i,k));
        end
        fprintf(fid2,'\n');
       end
end
fprintf(fid2,'%s\n','MATERIALS');
if micro==0
    if ~openning
        fprintf(fid2,'/ 1-%i /  1\n', enum);
    else
        fprintf(fid2,'/ ');
        fprintf(fid2,'%i ',ELEM(1,1));
        for i=2:size(ELEM,1)
            if mod(i,8)==1
                fprintf(fid2,'\n  %i ',ELEM(i,1));
            else
                fprintf(fid2,'%i ',ELEM(i,1));
            end
        end
        fprintf(fid2,'/  1\n');
    end
else
    blkEi=find(ELEM(:,11)==1);
    blkELEM=ELEM(blkEi,1);
    mtEi=find(ELEM(:,11)==2);
    mtELEM=ELEM(mtEi,1);
    fprintf(fid2,'/ ');
    fprintf(fid2,'%i ',blkELEM(1,1));
    for i=2:size(blkELEM,1)
        if mod(i,8)==1
            fprintf(fid2,'\n  %i ',blkELEM(i,1));
        else
            fprintf(fid2,'%i ',blkELEM(i,1));
        end
    end
    fprintf(fid2,'/  1\n');
    fprintf(fid2,'/ ');
    if ~isempty(mtELEM)
        fprintf(fid2,'%i ',mtELEM(1,1));
        for i=2:size(mtELEM,1)
            if mod(i,8)==1
                fprintf(fid2,'\n  %i ',mtELEM(i,1));
            else
```

```
                fprintf(fid2,'%i ',mtELEM(i,1));
            end
        end
        fprintf(fid2,'/  2\n');
    end
end

% materials
fprintf(fid2,'%s\n','''MATERIALS''');
if micro==0
    fprintf(fid2,'%i     %-9s%0.2E\n',1,'DENSIT',mvalue(1,10));
    %For Young's moduli and shear moduli, unit is converted to MPa*10^-
3 or KPa
    fprintf(fid2,'                                        %-9s%i       %i
%i\n','YOUNG',mvalue(1,1)*1E3,mvalue(1,2)*1E3,mvalue(1,3)*1E3);
    fprintf(fid2,'                                   %-9s%0.3f      %0.3f
%0.3f\n','POISON',mvalue(1,4),mvalue(1,5),mvalue(1,6));
    fprintf(fid2,'                                        %-9s%i       %i
%i\n','SHRMOD',mvalue(1,7)*1E3,mvalue(1,8)*1E3,mvalue(1,9)*1E3);
else
    for m=1:2
        fprintf(fid2,'%i     %-9s%0.2E\n',m,'DENSIT',mvalue(m,3));
        fprintf(fid2,'      %-9s%i\n','YOUNG',mvalue(m,1)*1E3);
        fprintf(fid2,'      %-9s%0.3f\n','POISON',mvalue(m,2));
    end
end

% element groups
fprintf(fid2,'%s\n','''GROUPS''');
fprintf(fid2,'%s\n','ELEMEN');
% groups of element entities
for j=1:max(ELEM(:,10))
    fprintf(fid2,'%i     EG_%i /   ',j,j);
    egi=find(ELEM(:,10)==j);
    grELEM=ELEM(egi,1);
    if ~isempty(grELEM)
        fprintf(fid2,'%i ',grELEM(1,1));
        for i=2:size(grELEM,1)
            if mod(i,8)==1
                fprintf(fid2,'\n               %i ',grELEM(i,1));
            else
                fprintf(fid2,'%i ',grELEM(i,1));
            end
        end
        fprintf(fid2,'/\n');
    end
end
% group of elements with face load applied
% 8-node brick elements
egnum=j+1;
fprintf(fid2,'%i     %s /   ',egnum,'FA_L1');
fprintf(fid2,'%i ',EP1(1,1));
for i=2:size(EP1,1)
    if mod(i,8)==1
        fprintf(fid2,'\n               %i ',EP1(i,1));
    else
        fprintf(fid2,'%i ',EP1(i,1));
    end
end
fprintf(fid2,'/\n');
```

```
% 6-node wedge elements
if ~isempty(EP2)
    fprintf(fid2,'%i     %s /  ',egnum+1,'FA_L2');
       fprintf(fid2,'%i ',EP2(1,1));
       for i=2:size(EP2,1)
        if mod(i,8)==1
            fprintf(fid2,'\n               %i ',EP2(i,1));
        else
            fprintf(fid2,'%i ',EP2(i,1));
        end
       end
       fprintf(fid2,'/\n');
end

% boundary condition
fprintf(fid2,'%s\n','''SUPPORTS''');
if ~isempty(LCON)   %left support
    for i=1:3
        conGRID=LCON(find(LCON(:,i+1)==1),1);
        if ~isempty(conGRID)
            fprintf(fid2,'/ ');
            fprintf(fid2,'%i ',conGRID(1,1));
            for j=2:size(conGRID,1)
                if mod(j,8)==1
                    fprintf(fid2,'\n  %i ',conGRID(j,1));
                else
                    fprintf(fid2,'%i ',conGRID(j,1));
                end
            end
            fprintf(fid2,'/ %s %i\n','TR',i);
        end
    end
end
if ~isempty(RCON)   %right support
    for i=1:3
        conGRID=RCON(find(RCON(:,i+1)==1),1);
        if ~isempty(conGRID)
            fprintf(fid2,'/ ');
            fprintf(fid2,'%i ',conGRID(1,1));
            for j=2:size(conGRID,1)
                if mod(j,8)==1
                    fprintf(fid2,'\n  %i ',conGRID(j,1));
                else
                    fprintf(fid2,'%i ',conGRID(j,1));
                end
            end
            fprintf(fid2,'/ %s %i\n','TR',i);
        end
    end
end
if ~isempty(BCON)   %bottom support
    for i=1:3
        conGRID=BCON(find(BCON(:,i+1)==1),1);
        if ~isempty(conGRID)
            fprintf(fid2,'/ ');
            fprintf(fid2,'%i ',conGRID(1,1));
            for j=2:size(conGRID,1)
                if mod(j,8)==1
                    fprintf(fid2,'\n  %i ',conGRID(j,1));
                else
```

```
                    fprintf(fid2,'%i ',conGRID(j,1));
                end
            end
            fprintf(fid2,'/ %s %i\n','TR',i);
        end
    end
end
if ~isempty(TCON)   %top support
    for i=1:3
        conGRID=TCON(find(TCON(:,i+1)==1),1);
        if ~isempty(conGRID)
            fprintf(fid2,'/ ');
            fprintf(fid2,'%i ',conGRID(1,1));
            for j=2:size(conGRID,1)
                if mod(j,8)==1
                    fprintf(fid2,'\n  %i ',conGRID(j,1));
                else
                    fprintf(fid2,'%i ',conGRID(j,1));
                end
            end
            fprintf(fid2,'/ %s %i\n','TR',i);
        end
    end
end

% loading
fprintf(fid2,'%s\n','''LOADS''');
fprintf(fid2,'%s\n','CASE 1');
fprintf(fid2,'%s\n','WEIGHT');
fprintf(fid2,'%i    %0.3E\n',3,-9.806E03);
fprintf(fid2,'%s\n','ELEMEN');
fprintf(fid2,'/ %s / \n','FA_L1');
fprintf(fid2,'      %s %s\n','FACE','ETA1');
fprintf(fid2,'                 %s   %0.2E   %i   %i   %i   %0.2E   %i   %i
%i\n','HYDRO',P,0,0,0,P,0,0,Wheight);
fprintf(fid2,'      %s %i\n','DIRECT',2);
if ~isempty(EP2)
       fprintf(fid2,'/ %s / \n','FA_L2');
       fprintf(fid2,'      %s %s\n','FACE','ZETA1');
       fprintf(fid2,'              %s   %0.2E   %i   %i   %i   %0.2E   %i   %i
%i\n','HYDRO',P,0,0,0,P,0,0,Wheight);
    fprintf(fid2,'      %s %i\n','DIRECT',2);
end
if openning
    fprintf(fid2,'%s\n','NODAL');
    for i=1:size(PL,1)
        fprintf(fid2,'%i     %s %i %0.2f\n', PL(i,1),'FORCE',2,PL(i,3));
    end
end

% directions
fprintf(fid2,'%s\n','''DIRECTIONS''');
fprintf(fid2,'    %i  %0.6E  %0.6E  %0.6E\n',1,1,0,0);
fprintf(fid2,'    %i  %0.6E  %0.6E  %0.6E\n',2,0,1,0);
fprintf(fid2,'    %i  %0.6E  %0.6E  %0.6E\n',3,0,0,1);
fprintf(fid2,'%s\n','''END''');

fclose(fid2);
```

A.5.12 Subroutine fblkGrid.m

```
%%define grids for a full block unit
%%section grids of a block
SGRID=[];
SGRID=[SGRID; nstart xstart ystart zstart];
gi=nstart;  % SGRID index
% grids of the first web
for i=2:ftdiv+1
    gi=gi+1;
    SGRID=[SGRID; gi SGRID(gi-1,2) SGRID(gi-1,3)+tfe SGRID(gi-1,4)];
end
for i=1:wldiv
    gi=gi+1;
    SGRID=[SGRID; gi SGRID(gi-1,2) SGRID(gi-1,3)+lwe SGRID(gi-1,4)];
end
for i=1:ftdiv
    gi=gi+1;
    SGRID=[SGRID; gi SGRID(gi-1,2) SGRID(gi-1,3)+tfe SGRID(gi-1,4)];
end
for j=1:wtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRID=[SGRID;     gi     SGRID(gi-yGNUM,2)+twe     SGRID(gi-yGNUM,3)
SGRID(gi-yGNUM,4)];
    end
end
% grids of the first two face shells
for j=1:fldiv
    for i=1:yGNUM
        gi=gi+1;
        SGRID=[SGRID;     gi     SGRID(gi-yGNUM,2)+lfe     SGRID(gi-yGNUM,3)
SGRID(gi-yGNUM,4)];
    end
end
% grids of the second web
for j=1:wtdiv
    if j*twe <= tw-0.5*tgap
        for i=1:yGNUM
        gi=gi+1;
        SGRID=[SGRID;     gi     SGRID(gi-yGNUM,2)+twe     SGRID(gi-yGNUM,3)
SGRID(gi-yGNUM,4)];
        end
    else
        for i=1:ftdiv+1
            gi=gi+1;
            SGRID=[SGRID;   gi   SGRID(gi-yGNUM,2)+twe   SGRID(gi-yGNUM,3)
SGRID(gi-yGNUM,4)];
        end
        for i=1:wldiv/2
            gi=gi+1;
            x=i*lwe*tgap/Lw;
            SGRID=[SGRID;  gi  SGRID(gi-yGNUM,2)+twe-x  SGRID(gi-yGNUM,3)
SGRID(gi-yGNUM,4)];
        end
        for i=1:wldiv/2
            gi=gi+1;
            x=(wldiv/2-i)*lwe*tgap/Lw;
            SGRID=[SGRID;  gi  SGRID(gi-yGNUM,2)+twe-x  SGRID(gi-yGNUM,3)
SGRID(gi-yGNUM,4)];
        end
        for i=1:ftdiv
            gi=gi+1;
            SGRID=[SGRID;   gi   SGRID(gi-yGNUM,2)+twe   SGRID(gi-yGNUM,3)
SGRID(gi-yGNUM,4)];
        end
    end
end
% grids of the third web
for j=1:wtdiv+1
    if j==1
        for i=1:ftdiv+1
            gi=gi+1;
            SGRID=[SGRID;    gi    SGRID(gi-yGNUM,2)    SGRID(gi-yGNUM,3)
SGRID(gi-yGNUM,4)];
        end
        for i=1:wldiv/2
            gi=gi+1;
            x=i*lwe*tgap/Lw;
            SGRID=[SGRID;  gi  SGRID(gi-yGNUM,2)+2*x  SGRID(gi-yGNUM,3)
SGRID(gi-yGNUM,4)];
        end
        for i=1:wldiv/2
            gi=gi+1;
            x=(wldiv/2-i)*lwe*tgap/Lw;
            SGRID=[SGRID;  gi  SGRID(gi-yGNUM,2)+2*x  SGRID(gi-yGNUM,3)
SGRID(gi-yGNUM,4)];
        end
        for i=1:ftdiv
            gi=gi+1;
            SGRID=[SGRID;    gi    SGRID(gi-yGNUM,2)    SGRID(gi-yGNUM,3)
SGRID(gi-yGNUM,4)];
        end
    else if j==2
            for i=1:yGNUM
                gi=gi+1;
                backnum=(wtdiv+2)*yGNUM;
                SGRID=[SGRID;  gi  SGRID(gi-backnum,2)+tw+twe  SGRID(gi-
backnum,3) SGRID(backnum,4)];
            end
         else
            for i=1:yGNUM
                gi=gi+1;
                SGRID=[SGRID; gi SGRID(gi-yGNUM,2)+twe SGRID(gi-yGNUM,3)
SGRID(gi-yGNUM,4)];
            end
        end
    end
end
%grids of the second two face shells
for j=1:fldiv
    for i=1:yGNUM
        gi=gi+1;
        SGRID=[SGRID;    gi    SGRID(gi-yGNUM,2)+lfe    SGRID(gi-yGNUM,3)
SGRID(gi-yGNUM,4)];
    end
end
```

```
%%define grids in one block volume
bsecGnum=size(SGRID,1);
BGRID=SGRID;
for k=1:bhdiv
    BGRID=[BGRID;    SGRID(1:bsecGnum,1)+k*bsecGnum    SGRID(1:bsecGnum,2)
SGRID(1:bsecGnum,3) SGRID(1:bsecGnum,4)+k*hbe];
end
```

A.5.13 Subroutine fblkElem.m

```
%%define elements for a full block unit
secGnum=bsecGnum;
% element group 1
m=1;
gstart=1; %start grid of the 1st column (intersection of face shells and
webs)
ei=0;
LELEM=[];
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup1;
gstart=(wtdiv+fldiv)*yGNUM+1; %start grid of the 3rd column
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 4th column
egroup1;
gstart=(2*wtdiv+fldiv+1)*yGNUM+1; %start grid of the 5th column
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 6th column
egroup1;

% element group 2
gstart=1+ftdiv; %start grid of the 1st column (leftmost web)
egroup2;
gstart=gstart+(wtdiv+fldiv)*yGNUM;
egroup2;
gstart=gstart+(wtdiv+1)*yGNUM;
egroup2;

% element group3
gstart=1+wtdiv*yGNUM; %start grid of the 1st column (face shell)
egroup3;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup3;
gstart=1+(wtdiv*3+fldiv+1)*yGNUM;
egroup3; %start grid of the 3rd column
gstart=gstart+ftdiv+wldiv; %start grid of the 4th column
egroup3;

%elements in one block volume
layerEnum=size(LELEM,1);
BELEM=LELEM;
for k=1:bhdiv-1
    BELEM=[BELEM;       LELEM(:,1)+k*layerEnum       LELEM(:,2)+k*bsecGnum
LELEM(:,3)+k*bsecGnum LELEM(:,4)+k*bsecGnum......
                 LELEM(:,5)+k*bsecGnum               LELEM(:,6)+k*bsecGnum
LELEM(:,7)+k*bsecGnum LELEM(:,8)+k*bsecGnum......
```

```
                  LELEM(:,9)+k*bsecGnum      LELEM(:,10)      LELEM(:,11)
LELEM(:,12)];
end
```

A.5.14 Subroutine hblkGrid.m

```
%%section grids of a half block
SGRIDH=[];
SGRIDH=[SGRIDH; nstart xstart ystart zstart];
hgi=nstart;  % SGRIDH index
% grids of the first web
for i=2:ftdiv+1
    hgi=hgi+1;
    SGRIDH=[SGRIDH;  hgi  SGRIDH(hgi-1,2)  SGRIDH(hgi-1,3)+tfe  SGRIDH(hgi-
1,4)];
end
for i=1:wldiv
    hgi=hgi+1;
    SGRIDH=[SGRIDH;  hgi  SGRIDH(hgi-1,2)  SGRIDH(hgi-1,3)+lwe  SGRIDH(hgi-
1,4)];
end
for i=1:ftdiv
    hgi=hgi+1;
    SGRIDH=[SGRIDH;  hgi  SGRIDH(hgi-1,2)  SGRIDH(hgi-1,3)+tfe  SGRIDH(hgi-
1,4)];
end
for j=1:wtdiv
    for i=1:yGNUM
        hgi=hgi+1;
        SGRIDH=[SGRIDH;  hgi  SGRIDH(hgi-yGNUM,2)+twe  SGRIDH(hgi-yGNUM,3)
SGRIDH(hgi-yGNUM,4)];
    end
end
% grids of the first face shells
for j=1:fldiv
    for i=1:yGNUM
        hgi=hgi+1;
        SGRIDH=[SGRIDH;  hgi  SGRIDH(hgi-yGNUM,2)+lfe  SGRIDH(hgi-yGNUM,3)
SGRIDH(hgi-yGNUM,4)];
    end
end
% grids of the second web
for j=1:wtdiv
    if j*twe <= tw-0.5*tgap
        for i=1:yGNUM
        hgi=hgi+1;
        SGRIDH=[SGRIDH;  hgi  SGRIDH(hgi-yGNUM,2)+twe  SGRIDH(hgi-yGNUM,3)
SGRIDH(hgi-yGNUM,4)];
        end
    else
        for i=1:ftdiv+1
            hgi=hgi+1;
            SGRIDH=[SGRIDH;   hgi   SGRIDH(hgi-yGNUM,2)+twe   SGRIDH(hgi-
yGNUM,3) SGRIDH(hgi-yGNUM,4)];
        end
        for i=1:wldiv/2
```

```
            hgi=hgi+1;
            x=i*lwe*tgap/Lw;
            SGRIDH=[SGRIDH; hgi SGRIDH(hgi-yGNUM,2)+twe-x SGRIDH(hgi-
yGNUM,3) SGRIDH(hgi-yGNUM,4)];
        end
        for i=1:wldiv/2
            hgi=hgi+1;
            x=(wldiv/2-i)*lwe*tgap/Lw;
            SGRIDH=[SGRIDH; hgi SGRIDH(hgi-yGNUM,2)+twe-x SGRIDH(hgi-
yGNUM,3) SGRIDH(hgi-yGNUM,4)];
        end
        for i=1:ftdiv
            hgi=hgi+1;
            SGRIDH=[SGRIDH; hgi SGRIDH(hgi-yGNUM,2)+twe SGRIDH(hgi-
yGNUM,3) SGRIDH(hgi-yGNUM,4)];
        end
    end
end

%%grids in half block volume
hsecGnum=size(SGRIDH,1);
BGRIDH=SGRIDH;
for k=1:bhdiv
    BGRIDH=[BGRIDH; SGRIDH(1:hsecGnum,1)+k*hsecGnum SGRIDH(1:hsecGnum,2)
SGRIDH(1:hsecGnum,3) SGRIDH(1:hsecGnum,4)+k*hbe];
end
```

A.5.15 Subroutine hblkElem.m

```
%%define elements for a half block unit
secGnum=hsecGnum;
% element group 1
m=1;
gstart=1; %start grid of the 1st column (intersection of face shells and
webs)
ei=0;
LELEM=[];
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup1;
gstart=(wtdiv+fldiv)*yGNUM+1; %start grid of the 3rd column
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 4th column
egroup1;

% element group 2
gstart=1+ftdiv; %start grid of the 1st column (leftmost web)
egroup2;
gstart=gstart+(wtdiv+fldiv)*yGNUM;
egroup2;

% element group3
gstart=1+wtdiv*yGNUM; %start grid of the 1st column (face shell)
egroup3;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup3;
```

```
%elements in one half block volume
layerEnum=size(LELEM,1);
BELEMH=LELEM;
for k=1:bhdiv-1
    BELEMH=[BELEMH;      LELEM(:,1)+k*layerEnum      LELEM(:,2)+k*hsecGnum
LELEM(:,3)+k*hsecGnum LELEM(:,4)+k*hsecGnum......
                  LELEM(:,5)+k*hsecGnum             LELEM(:,6)+k*hsecGnum
LELEM(:,7)+k*hsecGnum LELEM(:,8)+k*hsecGnum......
                  LELEM(:,9)+k*hsecGnum      LELEM(:,10)      LELEM(:,11)
LELEM(:,12)];
end
```

A.5.16 Subroutine webGrid.m

```
%%section grids in a web
SGRIDW=[];
SGRIDW=[SGRIDW; nstart xstart ystart zstart];
wgi=nstart;  % SGRIDW index

for i=2:ftdiv+1
    wgi=wgi+1;
    SGRIDW=[SGRIDW;  wgi  SGRIDW(wgi-1,2)  SGRIDW(wgi-1,3)+tfe  SGRIDW(wgi-
1,4)];
end
for i=1:wldiv
    wgi=wgi+1;
    SGRIDW=[SGRIDW;  wgi  SGRIDW(wgi-1,2)  SGRIDW(wgi-1,3)+lwe  SGRIDW(wgi-
1,4)];
end
for i=1:ftdiv
    wgi=wgi+1;
    SGRIDW=[SGRIDW;  wgi  SGRIDW(wgi-1,2)  SGRIDW(wgi-1,3)+tfe  SGRIDW(wgi-
1,4)];
end
for j=1:wtdiv
    for i=1:yGNUM
        wgi=wgi+1;
        SGRIDW=[SGRIDW;  wgi  SGRIDW(wgi-yGNUM,2)+twe  SGRIDW(wgi-yGNUM,3)
SGRIDW(wgi-yGNUM,4)];
    end
end

%%grids in one web volume
wsecGnum=size(SGRIDW,1);
BGRIDW=SGRIDW;
for k=1:bhdiv
    BGRIDW=[BGRIDW;  SGRIDW(1:wsecGnum,1)+k*wsecGnum  SGRIDW(1:wsecGnum,2)
SGRIDW(1:wsecGnum,3) SGRIDW(1:wsecGnum,4)+k*hbe];
end
```

A.5.17 Subroutine webElem.m

```
%elements in one layer of a web
secGnum=wsecGnum;
% element group 1
m=1; %matrial group number
gstart=1; %start grid of the 1st column
ei=0;
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup1;
% element group 2
gstart=1+ftdiv; %start grid of the 1st column
egroup2;

%elements in one web volume
layerEnum=size(LELEM,1);
BELEMW=LELEM;
for k=1:bhdiv-1
    BELEMW=[BELEMW;      LELEM(:,1)+k*layerEnum      LELEM(:,2)+k*wsecGnum
LELEM(:,3)+k*wsecGnum LELEM(:,4)+k*wsecGnum......
                  LELEM(:,5)+k*wsecGnum              LELEM(:,6)+k*wsecGnum
LELEM(:,7)+k*wsecGnum LELEM(:,8)+k*wsecGnum......
                  LELEM(:,9)+k*wsecGnum      LELEM(:,10)       LELEM(:,11)
LELEM(:,12)];
end
```

A.5.18 Subroutine blayerGrid.m

```
%%define grids for a half block unit
hblkGrida;

%%define grids for a full block unit
fblkGrida;
%%define grids for consecutive full blocks in one course
blkGnuma=size(BGRIDa,1);

CBGRIDa=BGRIDa;
blknum=1; %number of consecutive full blocks
if mod(Wlength,tLb)==0
       for i=1:fullxnum-2
        blknum=blknum+1;
        CBGRIDa=[CBGRIDa;     BGRIDa(:,1)+i*blkGnuma     BGRIDa(:,2)+i*tLb
BGRIDa(:,3) BGRIDa(:,4)];
       end
else
    for i=1:fullxnum-1
        blknum=blknum+1;
        CBGRIDa=[CBGRIDa;     BGRIDa(:,1)+i*blkGnuma     BGRIDa(:,2)+i*tLb
BGRIDa(:,3) BGRIDa(:,4)];
       end
end

hblkGnuma=size(BGRIDHa,1);
```

```
%grids in the first half block and consecutive full blocks in one course
if Wlength==tLb
    CGRIDa=BGRIDHa;
    blknum=0;
else
    CGRIDa=[BGRIDHa;      CBGRIDa(:,1)+hblkGnuma      CBGRIDa(:,2)+2*tw+Lf
CBGRIDa(:,3) CBGRIDa(:,4)];
end
coubGnuma=size(CGRIDa,1); %number of grids in the first half block and
the consecutive full blocks
if mod(Wlength,tLb)==0
       %define grids for last half block
       hblkGridb;
    %define grids in the whole course
       CGRIDa=[CGRIDa;                             BGRIDHb(:,1)+coubGnuma
BGRIDHb(:,2)+(2*tw+Lf)+blknum*tLb BGRIDHb(:,3) BGRIDHb(:,4)];
end
```

A.5.19 Subroutine blayerElem.m

```
%%define elements in the first half block
hblkElema;

%%define elements for a full block unit
fblkElema;

%%define elements for consecutive full blocks in one course
blkEnuma=size(BELEMa,1);
%elements in a course without openning
CBELEMa=BELEMa;
if mod(Wlength,tLb)==0
       for i=1:fullxnum-2
        CBELEMa=[CBELEMa;  BELEMa(:,1)+i*blkEnuma  BELEMa(:,2)+i*blkGnuma
BELEMa(:,3)+i*blkGnuma BELEMa(:,4)+i*blkGnuma......
                       BELEMa(:,5)+i*blkGnuma       BELEMa(:,6)+i*blkGnuma
BELEMa(:,7)+i*blkGnuma BELEMa(:,8)+i*blkGnuma......
                       BELEMa(:,9)+i*blkGnuma  BELEMa(:,10)  BELEMa(:,11)
BELEMa(:,12)];
       end
else
    for i=1:fullxnum-1
        CBELEMa=[CBELEMa;  BELEMa(:,1)+i*blkEnuma  BELEMa(:,2)+i*blkGnuma
BELEMa(:,3)+i*blkGnuma BELEMa(:,4)+i*blkGnuma......
                       BELEMa(:,5)+i*blkGnuma       BELEMa(:,6)+i*blkGnuma
BELEMa(:,7)+i*blkGnuma BELEMa(:,8)+i*blkGnuma......
                       BELEMa(:,9)+i*blkGnuma  BELEMa(:,10)  BELEMa(:,11)
BELEMa(:,12)];
       end
end

%elements in the first half block and consecutive full blocks in one
course
hblkEnuma=size(BELEMHa,1);
if Wlength==tLb
    CELEMa=BELEMHa;
else
```

```
    CELEMa=[BELEMHa;     CBELEMa(:,1)+hblkEnuma     CBELEMa(:,2)+hblkGnuma
CBELEMa(:,3)+hblkGnuma CBELEMa(:,4)+hblkGnuma......
                      CBELEMa(:,5)+hblkGnuma        CBELEMa(:,6)+hblkGnuma
CBELEMa(:,7)+hblkGnuma CBELEMa(:,8)+hblkGnuma......
                      CBELEMa(:,9)+hblkGnuma  CBELEMa(:,10)  CBELEMa(:,11)
CBELEMa(:,12)];
end
coubEnuma=size(CELEMa,1); %number of elements in the first half block and
the consecutive full blocks
if mod(Wlength,tLb)==0
    %define elements for last half block
    hblkElemb;
    %%define elements in the whole course
    CELEMa=[CELEMa;     BELEMHb(:,1)+coubEnuma     BELEMHb(:,2)+coubGnuma
BELEMHb(:,3)+coubGnuma BELEMHb(:,4)+coubGnuma......
                      BELEMHb(:,5)+coubGnuma        BELEMHb(:,6)+coubGnuma
BELEMHb(:,7)+coubGnuma BELEMHb(:,8)+coubGnuma......
                      BELEMHb(:,9)+coubGnuma  BELEMHb(:,10)  BELEMHb(:,11)
BELEMHb(:,12)];
end
```

A.5.20 Subroutine mlayerGrid.m

```
%%define grids for a half block unit
mhblkGrida;

%%define grids for a full block unit
mfblkGrida;
%%define grids for consecutive full blocks in one course
mblkGnuma=size(MBGRIDa,1);
%grids in a course without openning
MCBGRIDa=MBGRIDa;
blknum=1; %number of consecutive full blocks
if mod(Wlength,tLb)==0
       for i=1:fullxnum-2
        blknum=blknum+1;
        MCBGRIDa=[MCBGRIDa;  MBGRIDa(:,1)+i*mblkGnuma  MBGRIDa(:,2)+i*tLb
MBGRIDa(:,3) MBGRIDa(:,4)];
       end
else
    for i=1:fullxnum-1
        blknum=blknum+1;
        MCBGRIDa=[MCBGRIDa;  MBGRIDa(:,1)+i*mblkGnuma  MBGRIDa(:,2)+i*tLb
MBGRIDa(:,3) MBGRIDa(:,4)];
       end
end

%grids in the first half block and consecutive full blocks in one course
mhblkGnuma=size(MBGRIDHa,1);
if Wlength==tLb
    MCGRIDa=MBGRIDHa;
    blknum=0;
else
    MCGRIDa=[MBGRIDHa;   MCBGRIDa(:,1)+mhblkGnuma   MCBGRIDa(:,2)+2*tw+Lf
MCBGRIDa(:,3) MCBGRIDa(:,4)];
end
```

```
mcoubGnuma=size(MCGRIDa,1); %number of grids in the first half block and
the consecutive full blocks
if mod(Wlength,tLb)==0
      %define grids for last half block
      mhblkGridb;
    %define grids in the whole course
      MCGRIDa=[MCGRIDa;                                MBGRIDHb(:,1)+mcoubGnuma
MBGRIDHb(:,2)+(2*tw+Lf)+blknum*tLb MBGRIDHb(:,3) MBGRIDHb(:,4)];
end
```

A.5.21 Subroutine mlayerElem.m

```
%%define elements in the first half block
mhblkElema;

%%define elements for a full block unit
mfblkElema;

%%define elements for consecutive full blocks in one course
mblkEnuma=size(MBELEMa,1);
%elements in a course without openning
MCBELEMa=MBELEMa;
if mod(Wlength,tLb)==0
      for i=1:fullxnum-2
       MCBELEMa=[MCBELEMa;                             MBELEMa(:,1)+i*mblkEnuma
MBELEMa(:,2)+i*mblkGnuma                               MBELEMa(:,3)+i*mblkGnuma
MBELEMa(:,4)+i*mblkGnuma......
                      MBELEMa(:,5)+i*mblkGnuma   MBELEMa(:,6)+i*mblkGnuma
MBELEMa(:,7)+i*mblkGnuma MBELEMa(:,8)+i*mblkGnuma......
                      MBELEMa(:,9)+i*mblkGnuma               MBELEMa(:,10)
MBELEMa(:,11) MBELEMa(:,12)];
      end
else
    for i=1:fullxnum-1
       MCBELEMa=[MCBELEMa;                             MBELEMa(:,1)+i*mblkEnuma
MBELEMa(:,2)+i*mblkGnuma                               MBELEMa(:,3)+i*mblkGnuma
MBELEMa(:,4)+i*mblkGnuma......
                      MBELEMa(:,5)+i*mblkGnuma   MBELEMa(:,6)+i*mblkGnuma
MBELEMa(:,7)+i*mblkGnuma MBELEMa(:,8)+i*mblkGnuma......
                      MBELEMa(:,9)+i*mblkGnuma               MBELEMa(:,10)
MBELEMa(:,11) MBELEMa(:,12)];
      end
end

%elements in the first half block and consecutive full blocks in one
course
mhblkEnuma=size(MBELEMHa,1);
if Wlength==tLb
    MCELEMa=MBELEMHa;
else
    MCELEMa=[MBELEMHa; MCBELEMa(:,1)+mhblkEnuma MCBELEMa(:,2)+mhblkGnuma
MCBELEMa(:,3)+mhblkGnuma MCBELEMa(:,4)+mhblkGnuma......
                      MCBELEMa(:,5)+mhblkGnuma   MCBELEMa(:,6)+mhblkGnuma
MCBELEMa(:,7)+mhblkGnuma MCBELEMa(:,8)+mhblkGnuma......
                      MCBELEMa(:,9)+mhblkGnuma              MCBELEMa(:,10)
MCBELEMa(:,11) MCBELEMa(:,12)];
```

```
end
mcoubEnuma=size(MCELEMa,1); %number of elements in the first half block
and the consecutive full blocks
if mod(Wlength,tLb)==0
    %define elements for last half block
    mhblkElemb;
    %%define elements in the whole course
    MCELEMa=[MCELEMa;  MBELEMHb(:,1)+mcoubEnuma  MBELEMHb(:,2)+mcoubGnuma
MBELEMHb(:,3)+mcoubGnuma MBELEMHb(:,4)+mcoubGnuma......
                      MBELEMHb(:,5)+mcoubGnuma   MBELEMHb(:,6)+mcoubGnuma
MBELEMHb(:,7)+mcoubGnuma MBELEMHb(:,8)+mcoubGnuma......
                      MBELEMHb(:,9)+mcoubGnuma           MBELEMHb(:,10)
MBELEMHb(:,11) MBELEMHb(:,12)];
end
```

A.5.22 Subroutine blayerGrid2.m

```
%%define grids in a full block in even layer
fblkGridb;
%%define grids for consecutive full blocks in one course
blkGnumb=size(BGRIDb,1);
%grids in a course without openning
CBGRIDb=BGRIDb;
blknum=1; %number of consecutive full blocks
if mod(Wlength,tLb)==0
    for i=1:fullxnum-2
        blknum=blknum+1;
        CBGRIDb=[CBGRIDb;    BGRIDb(:,1)+i*blkGnumb    BGRIDb(:,2)+i*tLb
BGRIDb(:,3) BGRIDb(:,4)];
    end
    fblkGridc; %define grids in the last full block in even layer
    if Wlength==tLb
        blknum=0;
        CBGRIDb=BGRIDc;
    else
        cbGnumb=size(CBGRIDb,1);
        CBGRIDb=[CBGRIDb;   BGRIDc(:,1)+cbGnumb   BGRIDc(:,2)+blknum*tLb
BGRIDc(:,3) BGRIDc(:,4)];
    end
else
       for i=1:fullxnum-1
        blknum=blknum+1;
        CBGRIDb=[CBGRIDb;    BGRIDb(:,1)+i*blkGnumb    BGRIDb(:,2)+i*tLb
BGRIDb(:,3) BGRIDb(:,4)];
       end
end
%%Last web or half block to complete the course
coubGnumb=size(CBGRIDb,1); % number of grids in the consecutive full
blocks in one course
if mod(Wlength,tLb)==0  %last web
    webGrid;
    CGRIDb=[CBGRIDb;   BGRIDW(:,1)+coubGnumb   BGRIDW(:,2)+fullxnum*tLb
BGRIDW(:,3) BGRIDW(:,4)];
else if mod(Wlength,tLb)==200 %half block
        hblkGrida;
        CGRIDb=[CBGRIDb; BGRIDHa(:,1)+coubGnumb BGRIDHa(:,2)+fullxnum*tLb
BGRIDHa(:,3) BGRIDHa(:,4)];
     end
end
```

A.5.23 Subroutine blayerElem2.m

```
%%define elements for a full block unit
fblkElemb;

%%define elements for consecutive full blocks in one course
blkEnumb=size(BELEMb,1);
%elements in a course without openning
CBELEMb=BELEMb;
if mod(Wlength,tLb)==0
       for i=1:fullxnum-2
        CBELEMb=[CBELEMb;  BELEMb(:,1)+i*blkEnumb  BELEMb(:,2)+i*blkGnumb
BELEMb(:,3)+i*blkGnumb BELEMb(:,4)+i*blkGnumb......
                    BELEMb(:,5)+i*blkGnumb               BELEMb(:,6)+i*blkGnumb
BELEMb(:,7)+i*blkGnumb BELEMb(:,8)+i*blkGnumb......
                    BELEMb(:,9)+i*blkGnumb     BELEMb(:,10)     BELEMb(:,11)
BELEMb(:,12)];
    end
    fblkElemc; %define elements in the last full block in even layer
    if Wlength==tLb
        CBELEMb=BELEMc;
    else
        cbEnumb=size(CBELEMb,1);
        CBELEMb=[CBELEMb;     BELEMc(:,1)+cbEnumb     BELEMc(:,2)+cbGnumb
BELEMc(:,3)+cbGnumb BELEMc(:,4)+cbGnumb......
                    BELEMc(:,5)+cbGnumb                   BELEMc(:,6)+cbGnumb
BELEMc(:,7)+cbGnumb BELEMc(:,8)+cbGnumb......
                    BELEMc(:,9)+cbGnumb      BELEMc(:,10)      BELEMc(:,11)
BELEMc(:,12)];
    end
else
       for i=1:fullxnum-1
        CBELEMb=[CBELEMb;  BELEMb(:,1)+i*blkEnumb  BELEMb(:,2)+i*blkGnumb
BELEMb(:,3)+i*blkGnumb BELEMb(:,4)+i*blkGnumb......
                        BELEMb(:,5)+i*blkGnumb       BELEMb(:,6)+i*blkGnumb
BELEMb(:,7)+i*blkGnumb BELEMb(:,8)+i*blkGnumb......
                        BELEMb(:,9)+i*blkGnumb   BELEMb(:,10)   BELEMb(:,11)
BELEMb(:,12)];
       end
end
%elements in the last web or half block to complete the course
LELEM=[];
ei=0;
coubEnumb=size(CBELEMb,1);
if mod(Wlength,tLb)==0  %last web
    webElem;
    CELEMb=[CBELEMb;       BELEMW(:,1)+coubEnumb       BELEMW(:,2)+coubGnumb
BELEMW(:,3)+coubGnumb BELEMW(:,4)+coubGnumb......
                    BELEMW(:,5)+coubGnumb              BELEMW(:,6)+coubGnumb
BELEMW(:,7)+coubGnumb BELEMW(:,8)+coubGnumb......
```

```
                  BELEMW(:,9)+coubGnumb     BELEMW(:,10)      BELEMW(:,11)
BELEMW(:,12)];
else if mod(Wlength,tLb)==200 %half block
        hblkElema;
        CELEMb=[CBELEMb;   BELEMHa(:,1)+coubEnumb   BELEMHa(:,2)+coubGnumb
BELEMHa(:,3)+coubGnumb BELEMHa(:,4)+coubGnumb......
                      BELEMHa(:,5)+coubGnumb        BELEMHa(:,6)+coubGnumb
BELEMHa(:,7)+coubGnumb BELEMHa(:,8)+coubGnumb......
                      BELEMHa(:,9)+coubGnumb  BELEMHa(:,10)  BELEMHa(:,11)
BELEMHa(:,12)];
     end
end
```

A.5.24 Subroutine mlayerGrid2.m

```
%%define grids in a full block in even layer
mfblkGridb;
%%define grids for consecutive full blocks in one course
mblkGnumb=size(MBGRIDb,1);
%grids in a course without openning
MCBGRIDb=MBGRIDb;
blknum=1; %number of consecutive full blocks
if mod(Wlength,tLb)==0
    for i=1:fullxnum-2
        blknum=blknum+1;
        MCBGRIDb=[MCBGRIDb;  MBGRIDb(:,1)+i*mblkGnumb  MBGRIDb(:,2)+i*tLb
MBGRIDb(:,3) MBGRIDb(:,4)];
    end
    mfblkGridc; %define grids in the last full block in even layer
    if Wlength==tLb
        blknum=0;
        MCBGRIDb=MBGRIDc;
    else
        mcbGnumb=size(MCBGRIDb,1);
        MCBGRIDb=[MCBGRIDb; MBGRIDc(:,1)+mcbGnumb MBGRIDc(:,2)+blknum*tLb
MBGRIDc(:,3) MBGRIDc(:,4)];
    end
else
       for i=1:fullxnum-1
        blknum=blknum+1;
        MCBGRIDb=[MCBGRIDb;  MBGRIDb(:,1)+i*mblkGnumb  MBGRIDb(:,2)+i*tLb
MBGRIDb(:,3) MBGRIDb(:,4)];
       end
end
%%Last web or half block to complete the course
mcoubGnumb=size(MCBGRIDb,1); % number of grids in the consecutive full
blocks in one course
if mod(Wlength,tLb)==0  %last web
    mwebGrid;
    MCGRIDb=[MCBGRIDb; MBGRIDW(:,1)+mcoubGnumb MBGRIDW(:,2)+fullxnum*tLb
MBGRIDW(:,3) MBGRIDW(:,4)];
else if mod(Wlength,tLb)==200 %half block
        mhblkGrida;
        MCGRIDb=[MCBGRIDb;                         MBGRIDHa(:,1)+mcoubGnumb
MBGRIDHa(:,2)+fullxnum*tLb MBGRIDHa(:,3) MBGRIDHa(:,4)];
     end
```

```
end
```

A.5.25 Subroutine mlayerElem2.m

```
%%define elements for a full block unit
mfblkElemb;

%%define elements for consecutive full blocks in one course
mblkEnumb=size(MBELEMb,1);
%elements in a course without openning
MCBELEMb=MBELEMb;
if mod(Wlength,tLb)==0
       for i=1:fullxnum-2
        MCBELEMb=[MCBELEMb;                              MBELEMb(:,1)+i*mblkEnumb
MBELEMb(:,2)+i*mblkGnumb                                 MBELEMb(:,3)+i*mblkGnumb
MBELEMb(:,4)+i*mblkGnumb......
                   MBELEMb(:,5)+i*mblkGnumb              MBELEMb(:,6)+i*mblkGnumb
MBELEMb(:,7)+i*mblkGnumb MBELEMb(:,8)+i*mblkGnumb......
                   MBELEMb(:,9)+i*mblkGnumb    MBELEMb(:,10)    MBELEMb(:,11)
MBELEMb(:,12)];
    end
    mfblkElemc; %define elements in the last full block in even layer
    if Wlength==tLb
        MCBELEMb=MBELEMc;
    else
        mcbEnumb=size(MCBELEMb,1);
        MCBELEMb=[MCBELEMb;  MBELEMc(:,1)+mcbEnumb   MBELEMc(:,2)+mcbGnumb
MBELEMc(:,3)+mcbGnumb MBELEMc(:,4)+mcbGnumb......
                   MBELEMc(:,5)+mcbGnumb                 MBELEMc(:,6)+mcbGnumb
MBELEMc(:,7)+mcbGnumb MBELEMc(:,8)+mcbGnumb......
                   MBELEMc(:,9)+mcbGnumb     MBELEMc(:,10)     MBELEMc(:,11)
MBELEMc(:,12)];
    end
else
       for i=1:fullxnum-1
        MCBELEMb=[MCBELEMb;                              MBELEMb(:,1)+i*mblkEnumb
MBELEMb(:,2)+i*mblkGnumb                                 MBELEMb(:,3)+i*mblkGnumb
MBELEMb(:,4)+i*mblkGnumb......
                       MBELEMb(:,5)+i*mblkGnumb    MBELEMb(:,6)+i*mblkGnumb
MBELEMb(:,7)+i*mblkGnumb MBELEMb(:,8)+i*mblkGnumb......
                       MBELEMb(:,9)+i*mblkGnumb                MBELEMb(:,10)
MBELEMb(:,11) MBELEMb(:,12)];
       end
end
%elements in the last web or half block to complete the course
LELEM=[];
ei=0;
mcoubEnumb=size(MCBELEMb,1);
if mod(Wlength,tLb)==0  %last web
    mwebElem;
    MCELEMb=[MCBELEMb;   MBELEMW(:,1)+mcoubEnumb   MBELEMW(:,2)+mcoubGnumb
MBELEMW(:,3)+mcoubGnumb MBELEMW(:,4)+mcoubGnumb......
                   MBELEMW(:,5)+mcoubGnumb            MBELEMW(:,6)+mcoubGnumb
MBELEMW(:,7)+mcoubGnumb MBELEMW(:,8)+mcoubGnumb......
                   MBELEMW(:,9)+mcoubGnumb    MBELEMW(:,10)    MBELEMW(:,11)
MBELEMW(:,12)];
```

```
else if mod(Wlength,tLb)==200 %half block
        mhblkElema;
        MCELEMb=[MCBELEMb;                             MBELEMHa(:,1)+mcoubEnumb
MBELEMHa(:,2)+mcoubGnumb                               MBELEMHa(:,3)+mcoubGnumb
MBELEMHa(:,4)+mcoubGnumb......
                      MBELEMHa(:,5)+mcoubGnumb   MBELEMHa(:,6)+mcoubGnumb
MBELEMHa(:,7)+mcoubGnumb MBELEMHa(:,8)+mcoubGnumb......
                      MBELEMHa(:,9)+mcoubGnumb             MBELEMHa(:,10)
MBELEMHa(:,11) MBELEMHa(:,12)];
     end
end
```

A.5.26 Subroutine solidGE.m

```
for k=1:fullznum-1
       if mod(k,2)==1
        pcouGnum=size(WGRID,1); %grid numbers in previous courses
        pcouEnum=size(WELEM,1); %element numbers in previous courses
        WGRID=[WGRID;  MCGRIDa(:,1)+pcouGnum  MCGRIDa(:,2)  MCGRIDa(:,3)
MCGRIDa(:,4)+WGRID(pcouGnum,4)];
        WELEM=[WELEM;    MCELEMa(:,1)+pcouEnum    MCELEMa(:,2)+pcouGnum
MCELEMa(:,3)+pcouGnum MCELEMa(:,4)+pcouGnum......
             MCELEMa(:,5)+pcouGnum                MCELEMa(:,6)+pcouGnum
MCELEMa(:,7)+pcouGnum MCELEMa(:,8)+pcouGnum......
             MCELEMa(:,9)+pcouGnum      MCELEMa(:,10)       MCELEMa(:,11)
MCELEMa(:,12)];

        pcouGnum=size(WGRID,1); %grid numbers in previous courses
        pcouEnum=size(WELEM,1); %element numbers in previous courses
        WGRID=[WGRID;   CGRIDb(:,1)+pcouGnum   CGRIDb(:,2)   CGRIDb(:,3)
CGRIDb(:,4)+WGRID(pcouGnum,4)];
        WELEM=[WELEM;     CELEMb(:,1)+pcouEnum     CELEMb(:,2)+pcouGnum
CELEMb(:,3)+pcouGnum CELEMb(:,4)+pcouGnum......
             CELEMb(:,5)+pcouGnum                  CELEMb(:,6)+pcouGnum
CELEMb(:,7)+pcouGnum CELEMb(:,8)+pcouGnum......
             CELEMb(:,9)+pcouGnum       CELEMb(:,10)        CELEMb(:,11)
CELEMb(:,12)];
       else
        pcouGnum=size(WGRID,1); %grid numbers in previous courses
        pcouEnum=size(WELEM,1); %element numbers in previous courses
        WGRID=[WGRID;  MCGRIDb(:,1)+pcouGnum  MCGRIDb(:,2)  MCGRIDb(:,3)
MCGRIDb(:,4)+WGRID(pcouGnum,4)];
        WELEM=[WELEM;    MCELEMb(:,1)+pcouEnum    MCELEMb(:,2)+pcouGnum
MCELEMb(:,3)+pcouGnum MCELEMb(:,4)+pcouGnum......
             MCELEMb(:,5)+pcouGnum                MCELEMb(:,6)+pcouGnum
MCELEMb(:,7)+pcouGnum MCELEMb(:,8)+pcouGnum......
             MCELEMb(:,9)+pcouGnum      MCELEMb(:,10)       MCELEMb(:,11)
MCELEMb(:,12)];

        pcouGnum=size(WGRID,1); %grid numbers in previous courses
        pcouEnum=size(WELEM,1); %element numbers in previous courses
        WGRID=[WGRID;   CGRIDa(:,1)+pcouGnum   CGRIDa(:,2)   CGRIDa(:,3)
CGRIDa(:,4)+WGRID(pcouGnum,4)];
        WELEM=[WELEM;     CELEMa(:,1)+pcouEnum     CELEMa(:,2)+pcouGnum
CELEMa(:,3)+pcouGnum CELEMa(:,4)+pcouGnum......
```

```
              CELEMa(:,5)+pcouGnum                          CELEMa(:,6)+pcouGnum
CELEMa(:,7)+pcouGnum CELEMa(:,8)+pcouGnum......
              CELEMa(:,9)+pcouGnum         CELEMa(:,10)        CELEMa(:,11)
CELEMa(:,12)];
       end
end
```

A.5.27 Subroutine adjustGE.m

```
for i=1:size(ONELEM,1)
    if (WGRID(ONELEM(i,3),2)~=left | WGRID(ONELEM(i,3),4)~=top) &......
        ~ismember(ONELEM(i,2),OGRID(:,1))                                 &
ismember(ONELEM(i,3),OGRID(:,1)) &......
        ~ismember(ONELEM(i,6),OGRID(:,1))                                 &
~ismember(ONELEM(i,7),OGRID(:,1))
        %left-top corner
        NEWGRID=[NEWGRID;       size(WGRID,1)+size(NEWGRID,1)+1       left
WGRID(ONELEM(i,3),3) WGRID(ONELEM(i,3),4)];
        NEWGRID=[NEWGRID;       size(WGRID,1)+size(NEWGRID,1)+1       left
WGRID(ONELEM(i,4),3) WGRID(ONELEM(i,4),4)];
        NEWGRID=[NEWGRID;       size(WGRID,1)+size(NEWGRID,1)+1       left
WGRID(ONELEM(i,3),3) top];
        NEWGRID=[NEWGRID;       size(WGRID,1)+size(NEWGRID,1)+1       left
WGRID(ONELEM(i,4),3) top];
        NEWGRID=[NEWGRID;                  size(WGRID,1)+size(NEWGRID,1)+1
WGRID(ONELEM(i,3),2) WGRID(ONELEM(i,3),3) top];
        NEWGRID=[NEWGRID;                  size(WGRID,1)+size(NEWGRID,1)+1
WGRID(ONELEM(i,4),2) WGRID(ONELEM(i,4),3) top];

        MDFELEM(i,3)=NEWGRID(size(NEWGRID,1)-5,1);
        MDFELEM(i,4)=NEWGRID(size(NEWGRID,1)-4,1);
        MDFELEM(i,7)=NEWGRID(size(NEWGRID,1)-3,1);
        MDFELEM(i,8)=NEWGRID(size(NEWGRID,1)-2,1);
        NEWELEM=[NEWELEM;                  size(WELEM,1)+size(NEWELEM,1)+1
NEWGRID(size(NEWGRID,1)-3,1) NEWGRID(size(NEWGRID,1)-1,1)......
                 NEWGRID(size(NEWGRID,1),1)    NEWGRID(size(NEWGRID,1)-2,1)
ONELEM(i,6) ONELEM(i,7) ONELEM(i,8)......
                 ONELEM(i,9) ONELEM(i,10) ONELEM(i,11) ONELEM(i,12)];

    else if  (WGRID(ONELEM(i,7),2)~=left  |  WGRID(ONELEM(i,7),4)~=bottom)
&......
        ~ismember(ONELEM(i,2),OGRID(:,1))                                 &
~ismember(ONELEM(i,3),OGRID(:,1)) &......
        ~ismember(ONELEM(i,6),OGRID(:,1))                                 &
ismember(ONELEM(i,7),OGRID(:,1))
        %left-bottom corner
        NEWGRID=[NEWGRID;       size(WGRID,1)+size(NEWGRID,1)+1       left
WGRID(ONELEM(i,7),3) WGRID(ONELEM(i,7),4)];
        NEWGRID=[NEWGRID;       size(WGRID,1)+size(NEWGRID,1)+1       left
WGRID(ONELEM(i,8),3) WGRID(ONELEM(i,8),4)];
        NEWGRID=[NEWGRID;       size(WGRID,1)+size(NEWGRID,1)+1       left
WGRID(ONELEM(i,7),3) bottom];
        NEWGRID=[NEWGRID;       size(WGRID,1)+size(NEWGRID,1)+1       left
WGRID(ONELEM(i,8),3) bottom];
        NEWGRID=[NEWGRID;                  size(WGRID,1)+size(NEWGRID,1)+1
WGRID(ONELEM(i,7),2) WGRID(ONELEM(i,7),3) bottom];
```

```
        NEWGRID=[NEWGRID;                      size(WGRID,1)+size(NEWGRID,1)+1
WGRID(ONELEM(i,8),2) WGRID(ONELEM(i,8),3) bottom];

        MDFELEM(i,7)=NEWGRID(size(NEWGRID,1)-5,1);
        MDFELEM(i,8)=NEWGRID(size(NEWGRID,1)-4,1);
        MDFELEM(i,3)=NEWGRID(size(NEWGRID,1)-3,1);
        MDFELEM(i,4)=NEWGRID(size(NEWGRID,1)-2,1);
        NEWELEM=[NEWELEM;   size(WELEM,1)+size(NEWELEM,1)+1    ONELEM(i,2)
ONELEM(i,3) ONELEM(i,4) ONELEM(i,5)......
                NEWGRID(size(NEWGRID,1)-3,1) NEWGRID(size(NEWGRID,1)-1,1)
NEWGRID(size(NEWGRID,1),1)......
                NEWGRID(size(NEWGRID,1)-2,1)   ONELEM(i,10)   ONELEM(i,11)
ONELEM(i,12)];

    else if (WGRID(ONELEM(i,6),2)~=right | WGRID(ONELEM(i,6),4)~=bottom)
&......
        ~ismember(ONELEM(i,2),OGRID(:,1))                              &
~ismember(ONELEM(i,3),OGRID(:,1)) &......
        ismember(ONELEM(i,6),OGRID(:,1))                               &
~ismember(ONELEM(i,7),OGRID(:,1))
        %right-bottom corner
        NEWGRID=[NEWGRID;      size(WGRID,1)+size(NEWGRID,1)+1      right
WGRID(ONELEM(i,6),3) WGRID(ONELEM(i,6),4)];
        NEWGRID=[NEWGRID;      size(WGRID,1)+size(NEWGRID,1)+1      right
WGRID(ONELEM(i,9),3) WGRID(ONELEM(i,9),4)];
        NEWGRID=[NEWGRID;      size(WGRID,1)+size(NEWGRID,1)+1      right
WGRID(ONELEM(i,6),3) bottom];
        NEWGRID=[NEWGRID;      size(WGRID,1)+size(NEWGRID,1)+1      right
WGRID(ONELEM(i,9),3) bottom];
        NEWGRID=[NEWGRID;                      size(WGRID,1)+size(NEWGRID,1)+1
WGRID(ONELEM(i,6),2) WGRID(ONELEM(i,6),3) bottom];
        NEWGRID=[NEWGRID;                      size(WGRID,1)+size(NEWGRID,1)+1
WGRID(ONELEM(i,9),2) WGRID(ONELEM(i,9),3) bottom];

        MDFELEM(i,6)=NEWGRID(size(NEWGRID,1)-5,1);
        MDFELEM(i,9)=NEWGRID(size(NEWGRID,1)-4,1);
        MDFELEM(i,2)=NEWGRID(size(NEWGRID,1)-3,1);
        MDFELEM(i,5)=NEWGRID(size(NEWGRID,1)-2,1);
        NEWELEM=[NEWELEM;   size(WELEM,1)+size(NEWELEM,1)+1    ONELEM(i,2)
ONELEM(i,3) ONELEM(i,4) ONELEM(i,5)......
                NEWGRID(size(NEWGRID,1)-1,1) NEWGRID(size(NEWGRID,1)-3,1)
NEWGRID(size(NEWGRID,1)-2,1)......
                NEWGRID(size(NEWGRID,1),1)    ONELEM(i,10)    ONELEM(i,11)
ONELEM(i,12)];
    else  if  (WGRID(ONELEM(i,2),2)~=right  |  WGRID(ONELEM(i,2),4)~=top)
&......
        ismember(ONELEM(i,2),OGRID(:,1))                               &
~ismember(ONELEM(i,3),OGRID(:,1)) &......
        ~ismember(ONELEM(i,6),OGRID(:,1))                              &
~ismember(ONELEM(i,7),OGRID(:,1))
        %right-top corner
        NEWGRID=[NEWGRID;      size(WGRID,1)+size(NEWGRID,1)+1      right
WGRID(ONELEM(i,2),3) WGRID(ONELEM(i,2),4)];
        NEWGRID=[NEWGRID;      size(WGRID,1)+size(NEWGRID,1)+1      right
WGRID(ONELEM(i,5),3) WGRID(ONELEM(i,5),4)];
        NEWGRID=[NEWGRID;      size(WGRID,1)+size(NEWGRID,1)+1      right
WGRID(ONELEM(i,2),3) top];
        NEWGRID=[NEWGRID;      size(WGRID,1)+size(NEWGRID,1)+1      right
WGRID(ONELEM(i,5),3) top];
```

```
        NEWGRID=[NEWGRID;                  size(WGRID,1)+size(NEWGRID,1)+1
WGRID(ONELEM(i,2),2) WGRID(ONELEM(i,2),3) top];
        NEWGRID=[NEWGRID;                  size(WGRID,1)+size(NEWGRID,1)+1
WGRID(ONELEM(i,5),2) WGRID(ONELEM(i,5),3) top];

        MDFELEM(i,2)=NEWGRID(size(NEWGRID,1)-5,1);
        MDFELEM(i,5)=NEWGRID(size(NEWGRID,1)-4,1);
        MDFELEM(i,6)=NEWGRID(size(NEWGRID,1)-3,1);
        MDFELEM(i,9)=NEWGRID(size(NEWGRID,1)-2,1);
        NEWELEM=[NEWELEM;                  size(WELEM,1)+size(NEWELEM,1)+1
NEWGRID(size(NEWGRID,1)-1,1) NEWGRID(size(NEWGRID,1)-3,1)......
                NEWGRID(size(NEWGRID,1)-2,1)    NEWGRID(size(NEWGRID,1),1)
ONELEM(i,6) ONELEM(i,7) ONELEM(i,8)......
                ONELEM(i,9) ONELEM(i,10) ONELEM(i,11) ONELEM(i,12)];
    else          if          ~ismember(ONELEM(i,2),OGRID(:,1))          &
ismember(ONELEM(i,3),OGRID(:,1)) &......
        ~ismember(ONELEM(i,3),LEGRID(:,1))                               &
~ismember(ONELEM(i,6),OGRID(:,1)) & ......
        ismember(ONELEM(i,7),OGRID(:,1))                                 &
~ismember(ONELEM(i,7),LEGRID(:,1))
        %left
        NEWGRID=[NEWGRID;        size(WGRID,1)+size(NEWGRID,1)+1       left
WGRID(ONELEM(i,7),3) WGRID(ONELEM(i,7),4)];
        NEWGRID=[NEWGRID;        size(WGRID,1)+size(NEWGRID,1)+1       left
WGRID(ONELEM(i,8),3) WGRID(ONELEM(i,8),4)];
        NEWGRID=[NEWGRID;        size(WGRID,1)+size(NEWGRID,1)+1       left
WGRID(ONELEM(i,3),3) WGRID(ONELEM(i,3),4)];
        NEWGRID=[NEWGRID;        size(WGRID,1)+size(NEWGRID,1)+1       left
WGRID(ONELEM(i,4),3) WGRID(ONELEM(i,4),4)];

        MDFELEM(i,7)=NEWGRID(size(NEWGRID,1)-3,1);
        MDFELEM(i,8)=NEWGRID(size(NEWGRID,1)-2,1);
        MDFELEM(i,3)=NEWGRID(size(NEWGRID,1)-1,1);
        MDFELEM(i,4)=NEWGRID(size(NEWGRID,1),1);
     else          if          ~ismember(ONELEM(i,2),OGRID(:,1))         &
~ismember(ONELEM(i,3),OGRID(:,1)) &......
        ismember(ONELEM(i,6),OGRID(:,1))                                 &
~ismember(ONELEM(i,6),BEGRID(:,1)) & ......
        ismember(ONELEM(i,7),OGRID(:,1))                                 &
~ismember(ONELEM(i,7),BEGRID(:,1))
        %bottom
        NEWGRID=[NEWGRID;                  size(WGRID,1)+size(NEWGRID,1)+1
WGRID(ONELEM(i,6),2) WGRID(ONELEM(i,6),3) bottom];
        NEWGRID=[NEWGRID;                  size(WGRID,1)+size(NEWGRID,1)+1
WGRID(ONELEM(i,9),2) WGRID(ONELEM(i,9),3) bottom];
        NEWGRID=[NEWGRID;                  size(WGRID,1)+size(NEWGRID,1)+1
WGRID(ONELEM(i,7),2) WGRID(ONELEM(i,7),3) bottom];
        NEWGRID=[NEWGRID;                  size(WGRID,1)+size(NEWGRID,1)+1
WGRID(ONELEM(i,8),2) WGRID(ONELEM(i,8),3) bottom];

        MDFELEM(i,6)=NEWGRID(size(NEWGRID,1)-3,1);
        MDFELEM(i,9)=NEWGRID(size(NEWGRID,1)-2,1);
        MDFELEM(i,7)=NEWGRID(size(NEWGRID,1)-1,1);
        MDFELEM(i,8)=NEWGRID(size(NEWGRID,1),1);
     else          if           ismember(ONELEM(i,2),OGRID(:,1))         &
~ismember(ONELEM(i,2),REGRID(:,1)) &......
        ~ismember(ONELEM(i,3),OGRID(:,1))                                &
ismember(ONELEM(i,6),OGRID(:,1)) & ......
        ~ismember(ONELEM(i,6),REGRID(:,1))                               &
~ismember(ONELEM(i,7),OGRID(:,1))
```

```
        %right
        NEWGRID=[NEWGRID;       size(WGRID,1)+size(NEWGRID,1)+1      right
WGRID(ONELEM(i,6),3) WGRID(ONELEM(i,6),4)];
        NEWGRID=[NEWGRID;       size(WGRID,1)+size(NEWGRID,1)+1      right
WGRID(ONELEM(i,9),3) WGRID(ONELEM(i,9),4)];
        NEWGRID=[NEWGRID;       size(WGRID,1)+size(NEWGRID,1)+1      right
WGRID(ONELEM(i,2),3) WGRID(ONELEM(i,2),4)];
        NEWGRID=[NEWGRID;       size(WGRID,1)+size(NEWGRID,1)+1      right
WGRID(ONELEM(i,5),3) WGRID(ONELEM(i,5),4)];

        MDFELEM(i,6)=NEWGRID(size(NEWGRID,1)-3,1);
        MDFELEM(i,9)=NEWGRID(size(NEWGRID,1)-2,1);
        MDFELEM(i,2)=NEWGRID(size(NEWGRID,1)-1,1);
        MDFELEM(i,5)=NEWGRID(size(NEWGRID,1),1);
     else        if        ismember(ONELEM(i,2),OGRID(:,1))             &
~ismember(ONELEM(i,2),TEGRID(:,1)) &......
        ismember(ONELEM(i,3),OGRID(:,1))                                &
~ismember(ONELEM(i,3),TEGRID(:,1)) & ......
        ~ismember(ONELEM(i,6),OGRID(:,1))                               &
~ismember(ONELEM(i,7),OGRID(:,1))
        %top
        NEWGRID=[NEWGRID;                  size(WGRID,1)+size(NEWGRID,1)+1
WGRID(ONELEM(i,2),2) WGRID(ONELEM(i,2),3) top];
        NEWGRID=[NEWGRID;                  size(WGRID,1)+size(NEWGRID,1)+1
WGRID(ONELEM(i,5),2) WGRID(ONELEM(i,5),3) top];
        NEWGRID=[NEWGRID;                  size(WGRID,1)+size(NEWGRID,1)+1
WGRID(ONELEM(i,3),2) WGRID(ONELEM(i,3),3) top];
        NEWGRID=[NEWGRID;                  size(WGRID,1)+size(NEWGRID,1)+1
WGRID(ONELEM(i,4),2) WGRID(ONELEM(i,4),3) top];

        MDFELEM(i,2)=NEWGRID(size(NEWGRID,1)-3,1);
        MDFELEM(i,5)=NEWGRID(size(NEWGRID,1)-2,1);
        MDFELEM(i,3)=NEWGRID(size(NEWGRID,1)-1,1);
        MDFELEM(i,4)=NEWGRID(size(NEWGRID,1),1);
     end
     end
     end
     end
     end
     end
     end
     end
end
```

A.5.28 Subroutine reorderGE.m

```
function [triE,E]=reorderGE(G,E)
   trinum=0;
   triE=[];
   for i=1:size(E,1)
       temp=E(i,:);
       % if the first two grids are identical
        if G(find(G(:,1)==E(i,2)),2) == G(find(G(:,1)==E(i,3)),2) &......
               G(find(G(:,1)==E(i,2)),3)    ==    G(find(G(:,1)==E(i,3)),3)
&......
               G(find(G(:,1)==E(i,2)),4) == G(find(G(:,1)==E(i,3)),4)
```

```
            trinum=trinum+1;
            E(i,3)=temp(1,6);
            E(i,4)=temp(1,7);
            E(i,5)=temp(1,2);
            E(i,6)=temp(1,5);
            E(i,7)=temp(1,9);
            E(i,9)=temp(1,5);
            triE=[triE; E(i,:)];
        else  if  G(find(G(:,1)==E(i,6)),2)  ==  G(find(G(:,1)==E(i,7)),2)
&......
                   G(find(G(:,1)==E(i,6)),3)  ==  G(find(G(:,1)==E(i,7)),3)
&......
                   G(find(G(:,1)==E(i,6)),4)  == G(find(G(:,1)==E(i,7)),4)
                % if the 5th and 6th grids are identical
                trinum=trinum+1;
                E(i,3)=temp(1,6);
                E(i,4)=temp(1,3);
                E(i,5)=temp(1,2);
                E(i,6)=temp(1,5);
                E(i,7)=temp(1,9);
                E(i,8)=temp(1,4);
                E(i,9)=temp(1,5);
                triE=[triE; E(i,:)];
         else  if  G(find(G(:,1)==E(i,2)),2)  ==  G(find(G(:,1)==E(i,6)),2)
&......
                G(find(G(:,1)==E(i,2)),3)   ==   G(find(G(:,1)==E(i,6)),3)
&......
                G(find(G(:,1)==E(i,2)),4) == G(find(G(:,1)==E(i,6)),4)
                % if the 1st and 5th grids are identical
                trinum=trinum+1;
                E(i,3)=temp(1,7);
                E(i,4)=temp(1,3);
                E(i,5)=temp(1,2);
                E(i,6)=temp(1,5);
                E(i,7)=temp(1,8);
                E(i,8)=temp(1,4);
                E(i,9)=temp(1,5);
                triE=[triE; E(i,:)];
          else   if G(find(G(:,1)==E(i,3)),2) == G(find(G(:,1)==E(i,7)),2)
&......
                G(find(G(:,1)==E(i,3)),3)   ==   G(find(G(:,1)==E(i,7)),3)
&......
                G(find(G(:,1)==E(i,3)),4) == G(find(G(:,1)==E(i,7)),4)
                % if the 2nd and 6th grids are identical
                trinum=trinum+1;
                E(i,3)=temp(1,6);
                E(i,4)=temp(1,3);
                E(i,5)=temp(1,2);
                E(i,6)=temp(1,5);
                E(i,7)=temp(1,9);
                E(i,8)=temp(1,4);
                E(i,9)=temp(1,5);
                triE=[triE; E(i,:)];
           end
           end
           end
           end
    end
```

A.5.29 Subroutine openningLoad.m

```
%identify the grids at the edges of the openning
if ismember(0,LEGRID(:,1))
    LEGRID=[];
end
if ismember(0,REGRID(:,1))
    REGRID=[];
end
if ismember(0,BEGRID(:,1))
    BEGRID=[];
end
if ismember(0,TEGRID(:,1))
    TEGRID=[];
end
if ~isempty(NEWGRID)
       LEGRID=[LEGRID; NEWGRID(find(NEWGRID(:,2)==left),:)];
       REGRID=[REGRID; NEWGRID(find(NEWGRID(:,2)==right),:)];
       BEGRID=[BEGRID; NEWGRID(find(NEWGRID(:,4)==bottom),:)];
       TEGRID=[TEGRID; NEWGRID(find(NEWGRID(:,4)==top),:)];
end

%identify the grids on the face edges of the openning
FLEGRID=LEGRID(find(LEGRID(:,3)==0),:);
FREGRID=REGRID(find(REGRID(:,3)==0),:);
FBEGRID=BEGRID(find(BEGRID(:,3)==0),:);
FTEGRID=TEGRID(find(TEGRID(:,3)==0),:);
FLEGRID=sortrows(FLEGRID,4);
FREGRID=sortrows(FREGRID,4);
FBEGRID=sortrows(FBEGRID,2);
FTEGRID=sortrows(FTEGRID,2);

%delete the dublicated grids
[LG,li]=mergeRows(FLEGRID(:,2:4));
[RG,ri]=mergeRows(FREGRID(:,2:4));
[BG,bi]=mergeRows(FBEGRID(:,2:4));
[TG,ti]=mergeRows(FTEGRID(:,2:4));
uniqLG=FLEGRID(li,:);
uniqRG=FREGRID(ri,:);
uniqBG=FBEGRID(bi,:);
uniqTG=FTEGRID(ti,:);

%calculate uniformly distributed load per unit length along the openning
edges
if  (left==0  |  right==Wlength+tw  |  right==Wlength-tLb/2+2*tw+Lf)  &
(bottom~=0 & top~=Wheight & top~=Wheight-tm)
    udl=P*wopen(oi)*hopen(oi)/(2*wopen(oi)+hopen(oi));
else  if  (bottom==0  |  top==Wheight  |  top==Wheight-tm)  &  (left~=0  &
right~=Wlength+tw & right~=Wlength-tLb/2+2*tw+Lf)
        udl=P*wopen(oi)*hopen(oi)/(wopen(oi)+2*hopen(oi));
    else if (left==0 & bottom==0) | (left==0 & top==Wheight) | (left==0 &
top==Wheight-tm) |......
            (right==Wlength+tw    &    bottom==0)    |    (right==Wlength-
tLb/2+2*tw+Lf & bottom==0) |......
            (right==Wlength+tw  &  top==Wheight)  |  (right==Wlength+tw  &
top==Wheight-tm) |......
            (right==Wlength-tLb/2+2*tw+Lf      &      top==Wheight)      |
(right==Wlength-tLb/2+2*tw+Lf & top==Wheight-tm)
            udl=P*wopen(oi)*hopen(oi)/(wopen(oi)+hopen(oi));
```

```
         else
            udl=P*wopen(oi)*hopen(oi)/(2*wopen(oi)+2*hopen(oi));
         end
    end
end
%add point loads at the left edge
if left~=0
    PL=[PL; uniqLG(1,1) 0 udl*0.5*(uniqLG(2,4)-uniqLG(1,4)) 0 0 0 0];
    nLG=size(uniqLG,1);
    for i=2:nLG-1
        PL=[PL; uniqLG(i,1) 0 udl*0.5*(uniqLG(i+1,4)-uniqLG(i-1,4)) 0 0 0
0];
    end
    PL=[PL; uniqLG(nLG,1) 0 udl*0.5*(uniqLG(nLG,4)-uniqLG(nLG-1,4)) 0 0 0
0];
end
%add point loads at the right edge
if (right~=Wlength+tw & right~=Wlength-tLb/2+2*tw+Lf)
    PL=[PL; uniqRG(1,1) 0 udl*0.5*(uniqRG(2,4)-uniqRG(1,4)) 0 0 0 0];
    nRG=size(uniqRG,1);
    for i=2:nRG-1
        PL=[PL; uniqRG(i,1) 0 udl*0.5*(uniqRG(i+1,4)-uniqRG(i-1,4)) 0 0 0
0];
    end
    PL=[PL; uniqRG(nRG,1) 0 udl*0.5*(uniqRG(nRG,4)-uniqRG(nRG-1,4)) 0 0 0
0];
end
%add point loads at the bottom edge
if bottom~=0
    PL=[PL; uniqBG(1,1) 0 udl*0.5*(uniqBG(2,2)-uniqBG(1,2)) 0 0 0 0];
    nBG=size(uniqBG,1);
    for i=2:nBG-1
        PL=[PL; uniqBG(i,1) 0 udl*0.5*(uniqBG(i+1,2)-uniqBG(i-1,2)) 0 0 0
0];
    end
    PL=[PL; uniqBG(nBG,1) 0 udl*0.5*(uniqBG(nBG,2)-uniqBG(nBG-1,2)) 0 0 0
0];
end
%add point loads at the top edge
if (top~=Wheight & top~=Wheight-tm)
    PL=[PL; uniqTG(1,1) 0 udl*0.5*(uniqTG(2,2)-uniqTG(1,2)) 0 0 0 0];
    nTG=size(uniqTG,1);
    for i=2:nTG-1
        PL=[PL; uniqTG(i,1) 0 udl*0.5*(uniqTG(i+1,2)-uniqTG(i-1,2)) 0 0 0
0];
    end
    PL=[PL; uniqTG(nTG,1) 0 udl*0.5*(uniqTG(nTG,2)-uniqTG(nTG-1,2)) 0 0 0
0];
end
```

A.5.30 Subroutine hobondbeam.m

```
%bottom bond beam
if (mod(left,tLb)==2*tw+Lf) %left distance is the multiple of half block
length
    %define grids and elements in the cells in one full block
    xstart=left-Lf-tw;
    bbfblkGE;
    %define grids in the cells in the consecutive full blocks
    obnum=fix(wopen(oi)/tLb);
    bblkGnum=size(BBGRID,1);
    CBBGRID=BBGRID;
    for i=1:obnum
         CBBGRID=[CBBGRID;    BBGRID(:,1)+i*bblkGnum    BBGRID(:,2)+i*tLb
BBGRID(:,3) BBGRID(:,4)];
    end
    %define elements in the cells in the consecutive full blocks
    bblkEnum=size(BBELEM,1);
    CBBELEM=BBELEM;
    for i=1:obnum
         CBBELEM=[CBBELEM; BBELEM(:,1)+i*bblkEnum BBELEM(:,2)+i*bblkGnum
BBELEM(:,3)+i*bblkGnum BBELEM(:,4)+i*bblkGnum......
              BBELEM(:,5)+i*bblkGnum                BBELEM(:,6)+i*bblkGnum
BBELEM(:,7)+i*bblkGnum BBELEM(:,8)+i*bblkGnum......
              BBELEM(:,9)+i*bblkGnum      BBELEM(:,10)      BBELEM(:,11)
BBELEM(:,12)];
    end
    %add grids and elements in the cells in a half block
    if (mod(wopen(oi),tLb)==tLb/2)
        xstart=tw;
        bbhblkGE;
        bblkGnum=size(CBBGRID,1);
        CBBGRID=[CBBGRID;                               BBGRIDH(:,1)+bblkGnum
BBGRIDH(:,2)+max(CBBGRID(:,2)) BBGRIDH(:,3) BBGRIDH(:,4)];
        bblkEnum=size(CBBELEM,1);
        CBBELEM=[CBBELEM;   BBELEMH(:,1)+bblkEnum   BBELEMH(:,2)+bblkGnum
BBELEMH(:,3)+bblkGnum BBELEMH(:,4)+bblkGnum......
              BBELEMH(:,5)+bblkGnum                 BBELEMH(:,6)+bblkGnum
BBELEMH(:,7)+bblkGnum BBELEMH(:,8)+bblkGnum......
              BBELEMH(:,9)+bblkGnum      BBELEMH(:,10)      BBELEMH(:,11)
BBELEMH(:,12)];
    end

else %left distance is the multiple of full block length
    %define grids in the cell in the first half block
    xstart=left-Lf;
    bbhblkGE;
    BBGRIDH1=BBGRIDH;
    BBELEMH1=BBELEMH;
    %define grids and elements in the cells in one full block
    xstart=tw;
    bbfblkGE;
    %define grids in the cells in the consecutive full blocks
    obnum=ceil(wopen(oi)/tLb);
    bblkGnum=size(BBGRID,1);
    CBBGRID=BBGRID;
    for i=1:obnum-1
         CBBGRID=[CBBGRID;    BBGRID(:,1)+i*bblkGnum    BBGRID(:,2)+i*tLb
BBGRID(:,3) BBGRID(:,4)];
    end
    %define elements in the cells in the consecutive full blocks
    bblkEnum=size(BBELEM,1);
    CBBELEM=BBELEM;
    for i=1:obnum-1
         CBBELEM=[CBBELEM; BBELEM(:,1)+i*bblkEnum BBELEM(:,2)+i*bblkGnum
BBELEM(:,3)+i*bblkGnum BBELEM(:,4)+i*bblkGnum......
```

```
              BBELEM(:,5)+i*bblkGnum                BBELEM(:,6)+i*bblkGnum
BBELEM(:,7)+i*bblkGnum BBELEM(:,8)+i*bblkGnum......
              BBELEM(:,9)+i*bblkGnum      BBELEM(:,10)      BBELEM(:,11)
BBELEM(:,12)];
    end
    %add grids and elements in the cells in the last half block
    if (mod(wopen(oi),tLb)==0)
        xstart=tw;
        bbhblkGE;
        bblkGnum=size(CBBGRID,1);
        CBBGRID=[CBBGRID;                                BBGRIDH(:,1)+bblkGnum
BBGRIDH(:,2)+max(CBBGRID(:,2)) BBGRIDH(:,3) BBGRIDH(:,4)];
        bblkEnum=size(CBBELEM,1);
        CBBELEM=[CBBELEM;   BBELEMH(:,1)+bblkEnum   BBELEMH(:,2)+bblkGnum
BBELEMH(:,3)+bblkGnum BBELEMH(:,4)+bblkGnum......
              BBELEMH(:,5)+bblkGnum                 BBELEMH(:,6)+bblkGnum
BBELEMH(:,7)+bblkGnum BBELEMH(:,8)+bblkGnum......
              BBELEMH(:,9)+bblkGnum     BBELEMH(:,10)     BBELEMH(:,11)
BBELEMH(:,12)];
    end
    %add grids and elements in the cells in the first half block
    bblkGnum=size(BBGRIDH1,1);
    CBBGRID=[BBGRIDH1;                                  CBBGRID(:,1)+bblkGnum
CBBGRID(:,2)+max(BBGRIDH1(:,2)) CBBGRID(:,3) CBBGRID(:,4)];
    bblkEnum=size(BBELEMH1,1);
    CBBELEM=[BBELEMH1;    CBBELEM(:,1)+bblkEnum    CBBELEM(:,2)+bblkGnum
CBBELEM(:,3)+bblkGnum CBBELEM(:,4)+bblkGnum......
          CBBELEM(:,5)+bblkGnum                     CBBELEM(:,6)+bblkGnum
CBBELEM(:,7)+bblkGnum CBBELEM(:,8)+bblkGnum......
          CBBELEM(:,9)+bblkGnum       CBBELEM(:,10)       CBBELEM(:,11)
CBBELEM(:,12)];
end
%number of grids and elements in bottom bond beam
bblkGnum=size(CBBGRID,1);
bblkEnum=size(CBBELEM,1);
%delete the grids outside the wall
maxx=max(WGRID(:,2));
dGi=find(CBBGRID(:,2)>maxx);
if ~isempty(dGi)
    CBBGRID(dGi,:)=[];
    dEi=find(~ismember(CBBELEM(:,2),CBBGRID(:,1))                        |
~ismember(CBBELEM(:,3),CBBGRID(:,1)) |......
            ~ismember(CBBELEM(:,4),CBBGRID(:,1))                         |
~ismember(CBBELEM(:,5),CBBGRID(:,1)) |......
            ~ismember(CBBELEM(:,6),CBBGRID(:,1))                         |
~ismember(CBBELEM(:,7),CBBGRID(:,1)) |......
            ~ismember(CBBELEM(:,8),CBBGRID(:,1))                         |
~ismember(CBBELEM(:,9),CBBGRID(:,1)) );
    CBBELEM(dEi,:)=[];
end

%%top bond beam
if bbond(oi)==0
    CBBGRID(:,4)=CBBGRID(:,4)+hopen(oi)+tHb;
else
    if tbond(oi)==1
        CBBGRID=[CBBGRID; CBBGRID(:,1)+bblkGnum CBBGRID(:,2) CBBGRID(:,3)
CBBGRID(:,4)+hopen(oi)+tHb];
        CBBELEM=[CBBELEM;   CBBELEM(:,1)+bblkEnum   CBBELEM(:,2)+bblkGnum
CBBELEM(:,3)+bblkGnum CBBELEM(:,4)+bblkGnum......
```

```
              CBBELEM(:,5)+bblkGnum                    CBBELEM(:,6)+bblkGnum
CBBELEM(:,7)+bblkGnum CBBELEM(:,8)+bblkGnum......
              CBBELEM(:,9)+bblkGnum      CBBELEM(:,10)       CBBELEM(:,11)
CBBELEM(:,12)];
    end
end
```

A.5.31 Subroutine hebondbeam.m

```
%create bond beam model
%bond beam at odd course
heoddBB;
%bond beam at even course
heevenBB;

%%%Define grids and elements in the bottom and top bond beams
if bbond(oi)==0 %no bottom bond beam
    if mod((zopen(oi)+hopen(oi))/tHb, 2)==0 %top bond beam at odd course
        CBBGRID1(:,4)=CBBGRID1(:,4)+hopen(oi)+tHb;
        CBBGRID=CBBGRID1;
        CBBELEM=CBBELEM1;
    else %top bond beam at even course
        CBBGRID2(:,4)=CBBGRID2(:,4)+hopen(oi)+tHb;
        CBBGRID=CBBGRID2;
        CBBELEM=CBBELEM2;
    end
else %bottom bond beam exists
    if mod(zopen(oi)/tHb, 2)==1 %bottom bond beam at odd course
        CBBGRID=CBBGRID1;
        CBBELEM=CBBELEM1;
    else %bottom bond beam at even course
        CBBGRID=CBBGRID2;
        CBBELEM=CBBELEM2;
    end
    %number of grids and elements in bottom bond beam
    bblkGnum=size(CBBGRID,1);
    bblkEnum=size(CBBELEM,1);
    if tbond(oi)==1 %top bond beam exists
        if  mod((zopen(oi)+hopen(oi))/tHb,  2)==0  %top  bond  beam  at  odd
course
            CBBGRID=[CBBGRID;     CBBGRID1(:,1)+bblkGnum     CBBGRID1(:,2)
CBBGRID1(:,3) CBBGRID1(:,4)+hopen(oi)+tHb];
            CBBELEM=[CBBELEM;                          CBBELEM1(:,1)+bblkEnum
CBBELEM1(:,2)+bblkGnum                                 CBBELEM1(:,3)+bblkGnum
CBBELEM1(:,4)+bblkGnum......
                  CBBELEM1(:,5)+bblkGnum           CBBELEM1(:,6)+bblkGnum
CBBELEM1(:,7)+bblkGnum CBBELEM1(:,8)+bblkGnum......
                  CBBELEM1(:,9)+bblkGnum   CBBELEM1(:,10)   CBBELEM1(:,11)
CBBELEM1(:,12)];
        else %top bond beam at even course
            CBBGRID=[CBBGRID;     CBBGRID2(:,1)+bblkGnum     CBBGRID2(:,2)
CBBGRID2(:,3) CBBGRID2(:,4)+hopen(oi)+tHb];
            CBBELEM=[CBBELEM;                          CBBELEM2(:,1)+bblkEnum
CBBELEM2(:,2)+bblkGnum                                 CBBELEM2(:,3)+bblkGnum
CBBELEM2(:,4)+bblkGnum......
```

```
                  CBBELEM2(:,5)+bblkGnum          CBBELEM2(:,6)+bblkGnum
CBBELEM2(:,7)+bblkGnum CBBELEM2(:,8)+bblkGnum......
                  CBBELEM2(:,9)+bblkGnum  CBBELEM2(:,10)   CBBELEM2(:,11)
CBBELEM2(:,12)];
        end
    end
end
```

A.5.32 Subroutine hblkGrida.m

```
%%section grids of the first half block in heterogeneous model
SGRIDHa=[];
SGRIDHa=[SGRIDHa; nstart xstart ystart zstart];
hgi=nstart;  % SGRIDHa index
% grids of the first web
for i=2:ftdiv+1
    hgi=hgi+1;
    SGRIDHa=[SGRIDHa;     hgi     SGRIDHa(hgi-1,2)     SGRIDHa(hgi-1,3)+tfe
SGRIDHa(hgi-1,4)];
end
for i=1:wldiv
    hgi=hgi+1;
    SGRIDHa=[SGRIDHa;     hgi     SGRIDHa(hgi-1,2)     SGRIDHa(hgi-1,3)+lwe
SGRIDHa(hgi-1,4)];
end
for i=1:ftdiv
    hgi=hgi+1;
    SGRIDHa=[SGRIDHa;     hgi     SGRIDHa(hgi-1,2)     SGRIDHa(hgi-1,3)+tfe
SGRIDHa(hgi-1,4)];
end
for j=1:wtdiv
    for i=1:yGNUM
        hgi=hgi+1;
        SGRIDHa=[SGRIDHa;   hgi   SGRIDHa(hgi-yGNUM,2)+twe   SGRIDHa(hgi-
yGNUM,3) SGRIDHa(hgi-yGNUM,4)];
    end
end
% grids of the two face shells
for j=1:fldiv
    for i=1:yGNUM
        hgi=hgi+1;
        SGRIDHa=[SGRIDHa;   hgi   SGRIDHa(hgi-yGNUM,2)+lfe   SGRIDHa(hgi-
yGNUM,3) SGRIDHa(hgi-yGNUM,4)];
    end
end
% grids of the second web
for j=1:wtdiv
    for i=1:yGNUM
        hgi=hgi+1;
        SGRIDHa=[SGRIDHa;   hgi   SGRIDHa(hgi-yGNUM,2)+twe   SGRIDHa(hgi-
yGNUM,3) SGRIDHa(hgi-yGNUM,4)];
    end
end

%%grids in half block volume
hsecGnuma=size(SGRIDHa,1);
```

```
BGRIDHa=SGRIDHa;
for k=1:bhdiv
    BGRIDHa=[BGRIDHa; SGRIDHa(:,1)+k*hsecGnuma SGRIDHa(:,2) SGRIDHa(:,3)
SGRIDHa(:,4)+k*hbe];
end
```

A.5.33 Subroutine fblkGrida.m

```
%%define grids for a full block unit (odd layer) in heterogeneous model
%%section grids of a block
SGRIDa=[];
SGRIDa=[SGRIDa; nstart xstart ystart zstart];
gi=nstart;  % SGRIDa index

%% grids of the first face shells
% grids in mortar thickness zone
for i=2:ftdiv+1
    gi=gi+1;
    SGRIDa=[SGRIDa; gi SGRIDa(gi-1,2) SGRIDa(gi-1,3)+tfe SGRIDa(gi-1,4)];
end
for i=1:wldiv
    gi=gi+1;
    SGRIDa=[SGRIDa; gi SGRIDa(gi-1,2) SGRIDa(gi-1,3)+lwe SGRIDa(gi-1,4)];
end
for i=1:ftdiv
    gi=gi+1;
    SGRIDa=[SGRIDa; gi SGRIDa(gi-1,2) SGRIDa(gi-1,3)+tfe SGRIDa(gi-1,4)];
end
for j=1:mtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDa=[SGRIDa;   gi   SGRIDa(gi-yGNUM,2)+tme   SGRIDa(gi-yGNUM,3)
SGRIDa(gi-yGNUM,4)];
    end
end
% grids in web thickness zone
for j=1:wtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDa=[SGRIDa;   gi   SGRIDa(gi-yGNUM,2)+twe   SGRIDa(gi-yGNUM,3)
SGRIDa(gi-yGNUM,4)];
    end
end
% grids in short face shell length zone
for j=1:fldiv2
    for i=1:yGNUM
        gi=gi+1;
        SGRIDa=[SGRIDa;   gi   SGRIDa(gi-yGNUM,2)+lfe2   SGRIDa(gi-yGNUM,3)
SGRIDa(gi-yGNUM,4)];
    end
end

%% grids of the first web
for j=1:wtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDa=[SGRIDa;   gi   SGRIDa(gi-yGNUM,2)+twe   SGRIDa(gi-yGNUM,3)
SGRIDa(gi-yGNUM,4)];
    end
end

%% grids of the mortar head joint
for j=1:mtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDa=[SGRIDa;   gi   SGRIDa(gi-yGNUM,2)+tme   SGRIDa(gi-yGNUM,3)
SGRIDa(gi-yGNUM,4)];
    end
end

%% grids of the second web
for j=1:wtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDa=[SGRIDa;   gi   SGRIDa(gi-yGNUM,2)+twe   SGRIDa(gi-yGNUM,3)
SGRIDa(gi-yGNUM,4)];
    end
end

%grids of the second face shells
for j=1:fldiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDa=[SGRIDa;   gi   SGRIDa(gi-yGNUM,2)+lfe   SGRIDa(gi-yGNUM,3)
SGRIDa(gi-yGNUM,4)];
    end
end

%% grids of the third web
for j=1:wtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDa=[SGRIDa;   gi   SGRIDa(gi-yGNUM,2)+twe   SGRIDa(gi-yGNUM,3)
SGRIDa(gi-yGNUM,4)];
    end
end

%%define grids in one full block volume
bsecGnuma=size(SGRIDa,1);
BGRIDa=SGRIDa;
for k=1:bhdiv
    BGRIDa=[BGRIDa;   SGRIDa(:,1)+k*bsecGnuma   SGRIDa(:,2)   SGRIDa(:,3)
SGRIDa(:,4)+k*hbe];
end
```

A.5.34 Subroutine hblkGridb.m

```
%%section grids of last half block in heterogeneous model
SGRIDHb=[];
SGRIDHb=[SGRIDHb; nstart xstart ystart zstart];
hgi=nstart;  % SGRIDHb index
% grids of the face shells
```

```
for i=2:ftdiv+1
    hgi=hgi+1;
    SGRIDHb=[SGRIDHb;     hgi     SGRIDHb(hgi-1,2)     SGRIDHb(hgi-1,3)+tfe
SGRIDHb(hgi-1,4)];
end
for i=1:wldiv
    hgi=hgi+1;
    SGRIDHb=[SGRIDHb;     hgi     SGRIDHb(hgi-1,2)     SGRIDHb(hgi-1,3)+lwe
SGRIDHb(hgi-1,4)];
end
for i=1:ftdiv
    hgi=hgi+1;
    SGRIDHb=[SGRIDHb;     hgi     SGRIDHb(hgi-1,2)     SGRIDHb(hgi-1,3)+tfe
SGRIDHb(hgi-1,4)];
end
for j=1:mtdiv
    for i=1:yGNUM
        hgi=hgi+1;
        SGRIDHb=[SGRIDHb;    hgi    SGRIDHb(hgi-yGNUM,2)+tme    SGRIDHb(hgi-
yGNUM,3) SGRIDHb(hgi-yGNUM,4)];
    end
end

for j=1:wtdiv
    for i=1:yGNUM
        hgi=hgi+1;
        SGRIDHb=[SGRIDHb;    hgi    SGRIDHb(hgi-yGNUM,2)+twe    SGRIDHb(hgi-
yGNUM,3) SGRIDHb(hgi-yGNUM,4)];
    end
end

for j=1:fldiv
    for i=1:yGNUM
        hgi=hgi+1;
        SGRIDHb=[SGRIDHb;    hgi    SGRIDHb(hgi-yGNUM,2)+lfe    SGRIDHb(hgi-
yGNUM,3) SGRIDHb(hgi-yGNUM,4)];
    end
end

% grids of the web
for j=1:wtdiv
    for i=1:yGNUM
        hgi=hgi+1;
        SGRIDHb=[SGRIDHb;    hgi    SGRIDHb(hgi-yGNUM,2)+twe    SGRIDHb(hgi-
yGNUM,3) SGRIDHb(hgi-yGNUM,4)];
    end
end

%%grids in half block volume
hsecGnumb=size(SGRIDHb,1);
BGRIDHb=SGRIDHb;
for k=1:bhdiv
    BGRIDHb=[BGRIDHb; SGRIDHb(:,1)+k*hsecGnumb SGRIDHb(:,2) SGRIDHb(:,3)
SGRIDHb(:,4)+k*hbe];
end
```

A.5.35 Subroutine mhblkGrida.m

```
%%section grids of the first half block in heterogeneous model
SGRIDHa=[];
SGRIDHa=[SGRIDHa; nstart xstart ystart zstart];
hgi=nstart;  % SGRIDHa index
% grids of the first web
for i=2:ftdiv+1
    hgi=hgi+1;
    SGRIDHa=[SGRIDHa;     hgi     SGRIDHa(hgi-1,2)     SGRIDHa(hgi-1,3)+tfe
SGRIDHa(hgi-1,4)];
end
for i=1:wldiv
    hgi=hgi+1;
    SGRIDHa=[SGRIDHa;     hgi     SGRIDHa(hgi-1,2)     SGRIDHa(hgi-1,3)+lwe
SGRIDHa(hgi-1,4)];
end
for i=1:ftdiv
    hgi=hgi+1;
    SGRIDHa=[SGRIDHa;     hgi     SGRIDHa(hgi-1,2)     SGRIDHa(hgi-1,3)+tfe
SGRIDHa(hgi-1,4)];
end
for j=1:wtdiv
    for i=1:yGNUM
        hgi=hgi+1;
        SGRIDHa=[SGRIDHa;   hgi   SGRIDHa(hgi-yGNUM,2)+twe   SGRIDHa(hgi-
yGNUM,3) SGRIDHa(hgi-yGNUM,4)];
    end
end
% grids of the first face shells
for j=1:fldiv
    for i=1:yGNUM
        hgi=hgi+1;
        SGRIDHa=[SGRIDHa;   hgi   SGRIDHa(hgi-yGNUM,2)+lfe   SGRIDHa(hgi-
yGNUM,3) SGRIDHa(hgi-yGNUM,4)];
    end
end
% grids of the second web
for j=1:wtdiv
    for i=1:yGNUM
        hgi=hgi+1;
        SGRIDHa=[SGRIDHa;   hgi   SGRIDHa(hgi-yGNUM,2)+twe   SGRIDHa(hgi-
yGNUM,3) SGRIDHa(hgi-yGNUM,4)];
    end
end

%%grids in half block volume
hsecGnuma=size(SGRIDHa,1);
MBGRIDHa=SGRIDHa;
for k=1:mtdiv
    MBGRIDHa=[MBGRIDHa;       SGRIDHa(:,1)+k*hsecGnuma       SGRIDHa(:,2)
SGRIDHa(:,3) SGRIDHa(:,4)+k*tme];
end
```

A.5.36 Subroutine mfblkGrida.m

```
%%define grids for a full block unit (odd layer) in heterogeneous model
%%section grids of a block
SGRIDa=[];
SGRIDa=[SGRIDa; nstart xstart ystart zstart];
gi=nstart;  % SGRIDa index

%% grids of the first face shells
% grids in mortar thickness zone
for i=2:ftdiv+1
    gi=gi+1;
    SGRIDa=[SGRIDa; gi SGRIDa(gi-1,2) SGRIDa(gi-1,3)+tfe SGRIDa(gi-1,4)];
end
for i=1:wldiv
    gi=gi+1;
    SGRIDa=[SGRIDa; gi SGRIDa(gi-1,2) SGRIDa(gi-1,3)+lwe SGRIDa(gi-1,4)];
end
for i=1:ftdiv
    gi=gi+1;
    SGRIDa=[SGRIDa; gi SGRIDa(gi-1,2) SGRIDa(gi-1,3)+tfe SGRIDa(gi-1,4)];
end
for j=1:mtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDa=[SGRIDa;   gi   SGRIDa(gi-yGNUM,2)+tme   SGRIDa(gi-yGNUM,3)
SGRIDa(gi-yGNUM,4)];
    end
end
% grids in web thickness zone
for j=1:wtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDa=[SGRIDa;   gi   SGRIDa(gi-yGNUM,2)+twe   SGRIDa(gi-yGNUM,3)
SGRIDa(gi-yGNUM,4)];
    end
end
% grids in short face shell length zone
for j=1:fldiv2
    for i=1:yGNUM
        gi=gi+1;
        SGRIDa=[SGRIDa;   gi   SGRIDa(gi-yGNUM,2)+lfe2   SGRIDa(gi-yGNUM,3)
SGRIDa(gi-yGNUM,4)];
    end
end

%% grids of the first web
for j=1:wtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDa=[SGRIDa;   gi   SGRIDa(gi-yGNUM,2)+twe   SGRIDa(gi-yGNUM,3)
SGRIDa(gi-yGNUM,4)];
    end
end

%% grids of the mortar head joint
for j=1:mtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDa=[SGRIDa;   gi   SGRIDa(gi-yGNUM,2)+tme   SGRIDa(gi-yGNUM,3)
SGRIDa(gi-yGNUM,4)];
    end
end

%% grids of the second web
for j=1:wtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDa=[SGRIDa;   gi   SGRIDa(gi-yGNUM,2)+twe   SGRIDa(gi-yGNUM,3)
SGRIDa(gi-yGNUM,4)];
    end
end

%grids of the second face shells
for j=1:fldiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDa=[SGRIDa;   gi   SGRIDa(gi-yGNUM,2)+lfe   SGRIDa(gi-yGNUM,3)
SGRIDa(gi-yGNUM,4)];
    end
end

%% grids of the third web
for j=1:wtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDa=[SGRIDa;   gi   SGRIDa(gi-yGNUM,2)+twe   SGRIDa(gi-yGNUM,3)
SGRIDa(gi-yGNUM,4)];
    end
end

%%define grids in one full block volume
bsecGnuma=size(SGRIDa,1);
MBGRIDa=SGRIDa;
for k=1:mtdiv
    MBGRIDa=[MBGRIDa;  SGRIDa(:,1)+k*bsecGnuma  SGRIDa(:,2)  SGRIDa(:,3)
SGRIDa(:,4)+k*tme];
end
```

A.5.37 Subroutine mhblkGridb.m

```
%%section grids of last half block in heterogeneous model
SGRIDHb=[];
SGRIDHb=[SGRIDHb; nstart xstart ystart zstart];
hgi=nstart;  % SGRIDHb index
% grids of the face shells
for i=2:ftdiv+1
    hgi=hgi+1;
    SGRIDHb=[SGRIDHb;    hgi    SGRIDHb(hgi-1,2)    SGRIDHb(hgi-1,3)+tfe
SGRIDHb(hgi-1,4)];
end
for i=1:wldiv
    hgi=hgi+1;
    SGRIDHb=[SGRIDHb;    hgi    SGRIDHb(hgi-1,2)    SGRIDHb(hgi-1,3)+lwe
SGRIDHb(hgi-1,4)];
```

```
end
for i=1:ftdiv
    hgi=hgi+1;
    SGRIDHb=[SGRIDHb;     hgi     SGRIDHb(hgi-1,2)     SGRIDHb(hgi-1,3)+tfe
SGRIDHb(hgi-1,4)];
end
for j=1:mtdiv
    for i=1:yGNUM
        hgi=hgi+1;
        SGRIDHb=[SGRIDHb;   hgi   SGRIDHb(hgi-yGNUM,2)+tme   SGRIDHb(hgi-
yGNUM,3) SGRIDHb(hgi-yGNUM,4)];
    end
end

for j=1:wtdiv
    for i=1:yGNUM
        hgi=hgi+1;
        SGRIDHb=[SGRIDHb;   hgi   SGRIDHb(hgi-yGNUM,2)+twe   SGRIDHb(hgi-
yGNUM,3) SGRIDHb(hgi-yGNUM,4)];
    end
end

for j=1:fldiv
    for i=1:yGNUM
        hgi=hgi+1;
        SGRIDHb=[SGRIDHb;   hgi   SGRIDHb(hgi-yGNUM,2)+lfe   SGRIDHb(hgi-
yGNUM,3) SGRIDHb(hgi-yGNUM,4)];
    end
end

% grids of the web
for j=1:wtdiv
    for i=1:yGNUM
        hgi=hgi+1;
        SGRIDHb=[SGRIDHb;   hgi   SGRIDHb(hgi-yGNUM,2)+twe   SGRIDHb(hgi-
yGNUM,3) SGRIDHb(hgi-yGNUM,4)];
    end
end

%%grids in half block volume
hsecGnumb=size(SGRIDHb,1);
MBGRIDHb=SGRIDHb;
for k=1:mtdiv
    MBGRIDHb=[MBGRIDHb;       SGRIDHb(:,1)+k*hsecGnumb       SGRIDHb(:,2)
SGRIDHb(:,3) SGRIDHb(:,4)+k*tme];
end
```

A.5.38 Subroutine fblkGridb.m

```
%%define grids for a full block unit (even layer) in heterogeneous model
%%section grids of a block
SGRIDb=[];
SGRIDb=[SGRIDb; nstart xstart ystart zstart];
gi=nstart;  % SGRIDb index

% grids in the first web
```

```
for i=2:ftdiv+1
    gi=gi+1;
    SGRIDb=[SGRIDb; gi SGRIDb(gi-1,2) SGRIDb(gi-1,3)+tfe SGRIDb(gi-1,4)];
end
for i=1:wldiv
    gi=gi+1;
    SGRIDb=[SGRIDb; gi SGRIDb(gi-1,2) SGRIDb(gi-1,3)+lwe SGRIDb(gi-1,4)];
end
for i=1:ftdiv
    gi=gi+1;
    SGRIDb=[SGRIDb; gi SGRIDb(gi-1,2) SGRIDb(gi-1,3)+tfe SGRIDb(gi-1,4)];
end
for j=1:wtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDb=[SGRIDb;   gi   SGRIDb(gi-yGNUM,2)+twe   SGRIDb(gi-yGNUM,3)
SGRIDb(gi-yGNUM,4)];
    end
end
% grids of the first face shells
for j=1:fldiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDb=[SGRIDb;   gi   SGRIDb(gi-yGNUM,2)+lfe   SGRIDb(gi-yGNUM,3)
SGRIDb(gi-yGNUM,4)];
    end
end
% grids in the second web
for j=1:wtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDb=[SGRIDb;   gi   SGRIDb(gi-yGNUM,2)+twe   SGRIDb(gi-yGNUM,3)
SGRIDb(gi-yGNUM,4)];
    end
end
%% grids of the mortar head joint
for j=1:mtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDb=[SGRIDb;   gi   SGRIDb(gi-yGNUM,2)+tme   SGRIDb(gi-yGNUM,3)
SGRIDb(gi-yGNUM,4)];
    end
end
%% grids of the third web
for j=1:wtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDb=[SGRIDb;   gi   SGRIDb(gi-yGNUM,2)+twe   SGRIDb(gi-yGNUM,3)
SGRIDb(gi-yGNUM,4)];
    end
end
% grids in short face shell length zone
for j=1:fldiv2
    for i=1:yGNUM
        gi=gi+1;
        SGRIDb=[SGRIDb;   gi   SGRIDb(gi-yGNUM,2)+lfe2   SGRIDb(gi-yGNUM,3)
SGRIDb(gi-yGNUM,4)];
    end
end
%grids in web thickness zone in the second face shells
```

```
for j=1:wtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDb=[SGRIDb;   gi   SGRIDb(gi-yGNUM,2)+twe   SGRIDb(gi-yGNUM,3)
SGRIDb(gi-yGNUM,4)];
    end
end
%% grids in mortar thickness zone in the second face shells
for j=1:mtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDb=[SGRIDb;   gi   SGRIDb(gi-yGNUM,2)+tme   SGRIDb(gi-yGNUM,3)
SGRIDb(gi-yGNUM,4)];
    end
end

%%define grids in one full block volume
bsecGnumb=size(SGRIDb,1);
BGRIDb=SGRIDb;
for k=1:bhdiv
    BGRIDb=[BGRIDb;   SGRIDb(:,1)+k*bsecGnumb   SGRIDb(:,2)   SGRIDb(:,3)
SGRIDb(:,4)+k*hbe];
end
```

A.5.39 Subroutine fblkGridc.m

```
%%define grids for a full block unit (even layer) in heterogeneous model
%%section grids of a block
SGRIDc=[];
SGRIDc=[SGRIDc; nstart xstart ystart zstart];
gi=nstart;  % SGRIDc index

% grids in the first web
for i=2:ftdiv+1
    gi=gi+1;
    SGRIDc=[SGRIDc; gi SGRIDc(gi-1,2) SGRIDc(gi-1,3)+tfe SGRIDc(gi-1,4)];
end
for i=1:wldiv
    gi=gi+1;
    SGRIDc=[SGRIDc; gi SGRIDc(gi-1,2) SGRIDc(gi-1,3)+lwe SGRIDc(gi-1,4)];
end
for i=1:ftdiv
    gi=gi+1;
    SGRIDc=[SGRIDc; gi SGRIDc(gi-1,2) SGRIDc(gi-1,3)+tfe SGRIDc(gi-1,4)];
end
for j=1:wtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDc=[SGRIDc;   gi   SGRIDc(gi-yGNUM,2)+twe   SGRIDc(gi-yGNUM,3)
SGRIDc(gi-yGNUM,4)];
    end
end
% grids of the first face shells
for j=1:fldiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDc=[SGRIDc;   gi   SGRIDc(gi-yGNUM,2)+lfe   SGRIDc(gi-yGNUM,3)
SGRIDc(gi-yGNUM,4)];
    end
end
% grids in the second web
for j=1:wtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDc=[SGRIDc;   gi   SGRIDc(gi-yGNUM,2)+twe   SGRIDc(gi-yGNUM,3)
SGRIDc(gi-yGNUM,4)];
    end
end
%% grids of the mortar head joint
for j=1:mtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDc=[SGRIDc;   gi   SGRIDc(gi-yGNUM,2)+tme   SGRIDc(gi-yGNUM,3)
SGRIDc(gi-yGNUM,4)];
    end
end
%% grids of the third web
for j=1:wtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDc=[SGRIDc;   gi   SGRIDc(gi-yGNUM,2)+twe   SGRIDc(gi-yGNUM,3)
SGRIDc(gi-yGNUM,4)];
    end
end
% grids in the second face shells
for j=1:fldiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDc=[SGRIDc;   gi   SGRIDc(gi-yGNUM,2)+lfe   SGRIDc(gi-yGNUM,3)
SGRIDc(gi-yGNUM,4)];
    end
end

%%define grids in one full block volume
bsecGnumc=size(SGRIDc,1);
BGRIDc=SGRIDc;
for k=1:bhdiv
    BGRIDc=[BGRIDc;   SGRIDc(:,1)+k*bsecGnumc   SGRIDc(:,2)   SGRIDc(:,3)
SGRIDc(:,4)+k*hbe];
end
```

A.5.40 Subroutine mfblkGridb.m

```
%%define grids for a full block unit (even layer) in heterogeneous model
%%section grids of a block
SGRIDb=[];
SGRIDb=[SGRIDb; nstart xstart ystart zstart];
gi=nstart;  % SGRIDb index

% grids in the first web
for i=2:ftdiv+1
    gi=gi+1;
    SGRIDb=[SGRIDb; gi SGRIDb(gi-1,2) SGRIDb(gi-1,3)+tfe SGRIDb(gi-1,4)];
end
for i=1:wldiv
    gi=gi+1;
    SGRIDb=[SGRIDb; gi SGRIDb(gi-1,2) SGRIDb(gi-1,3)+lwe SGRIDb(gi-1,4)];
end
for i=1:ftdiv
    gi=gi+1;
    SGRIDb=[SGRIDb; gi SGRIDb(gi-1,2) SGRIDb(gi-1,3)+tfe SGRIDb(gi-1,4)];
end
for j=1:wtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDb=[SGRIDb;   gi   SGRIDb(gi-yGNUM,2)+twe   SGRIDb(gi-yGNUM,3)
SGRIDb(gi-yGNUM,4)];
    end
end
% grids of the first face shells
for j=1:fldiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDb=[SGRIDb;   gi   SGRIDb(gi-yGNUM,2)+lfe   SGRIDb(gi-yGNUM,3)
SGRIDb(gi-yGNUM,4)];
    end
end
% grids in the second web
for j=1:wtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDb=[SGRIDb;   gi   SGRIDb(gi-yGNUM,2)+twe   SGRIDb(gi-yGNUM,3)
SGRIDb(gi-yGNUM,4)];
    end
end
%% grids of the mortar head joint
for j=1:mtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDb=[SGRIDb;   gi   SGRIDb(gi-yGNUM,2)+tme   SGRIDb(gi-yGNUM,3)
SGRIDb(gi-yGNUM,4)];
    end
end
%% grids of the third web
for j=1:wtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDb=[SGRIDb;   gi   SGRIDb(gi-yGNUM,2)+twe   SGRIDb(gi-yGNUM,3)
SGRIDb(gi-yGNUM,4)];
    end
end
% grids in short face shell length zone
for j=1:fldiv2
    for i=1:yGNUM
        gi=gi+1;
        SGRIDb=[SGRIDb;   gi   SGRIDb(gi-yGNUM,2)+lfe2   SGRIDb(gi-yGNUM,3)
SGRIDb(gi-yGNUM,4)];
    end
end
%grids in web thickness zone in the second face shells
for j=1:wtdiv
    for i=1:yGNUM
```

```
        gi=gi+1;
        SGRIDb=[SGRIDb;   gi   SGRIDb(gi-yGNUM,2)+twe   SGRIDb(gi-yGNUM,3)
SGRIDb(gi-yGNUM,4)];
    end
end
%% grids in mortar thickness zone in the second face shells
for j=1:mtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDb=[SGRIDb;   gi   SGRIDb(gi-yGNUM,2)+tme   SGRIDb(gi-yGNUM,3)
SGRIDb(gi-yGNUM,4)];
    end
end

%%define grids in one full block volume
bsecGnumb=size(SGRIDb,1);
MBGRIDb=SGRIDb;
for k=1:mtdiv
    MBGRIDb=[MBGRIDb;   SGRIDb(:,1)+k*bsecGnumb   SGRIDb(:,2)   SGRIDb(:,3)
SGRIDb(:,4)+k*tme];
end
```

A.5.41 Subroutine mfblkGridc.m

```
%%define grids for a full block unit (even layer) in heterogeneous model
%%section grids of a block
SGRIDc=[];
SGRIDc=[SGRIDc; nstart xstart ystart zstart];
gi=nstart;  % SGRIDc index

% grids in the first web
for i=2:ftdiv+1
    gi=gi+1;
    SGRIDc=[SGRIDc; gi SGRIDc(gi-1,2) SGRIDc(gi-1,3)+tfe SGRIDc(gi-1,4)];
end
for i=1:wldiv
    gi=gi+1;
    SGRIDc=[SGRIDc; gi SGRIDc(gi-1,2) SGRIDc(gi-1,3)+lwe SGRIDc(gi-1,4)];
end
for i=1:ftdiv
    gi=gi+1;
    SGRIDc=[SGRIDc; gi SGRIDc(gi-1,2) SGRIDc(gi-1,3)+tfe SGRIDc(gi-1,4)];
end
for j=1:wtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDc=[SGRIDc;   gi   SGRIDc(gi-yGNUM,2)+twe   SGRIDc(gi-yGNUM,3)
SGRIDc(gi-yGNUM,4)];
    end
end
% grids of the first face shells
for j=1:fldiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDc=[SGRIDc;   gi   SGRIDc(gi-yGNUM,2)+lfe   SGRIDc(gi-yGNUM,3)
SGRIDc(gi-yGNUM,4)];
```

```
    end
end
% grids in the second web
for j=1:wtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDc=[SGRIDc;   gi   SGRIDc(gi-yGNUM,2)+twe   SGRIDc(gi-yGNUM,3)
SGRIDc(gi-yGNUM,4)];
    end
end
%% grids of the mortar head joint
for j=1:mtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDc=[SGRIDc;   gi   SGRIDc(gi-yGNUM,2)+tme   SGRIDc(gi-yGNUM,3)
SGRIDc(gi-yGNUM,4)];
    end
end
%% grids of the third web
for j=1:wtdiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDc=[SGRIDc;   gi   SGRIDc(gi-yGNUM,2)+twe   SGRIDc(gi-yGNUM,3)
SGRIDc(gi-yGNUM,4)];
    end
end
% grids in the second face shells
for j=1:fldiv
    for i=1:yGNUM
        gi=gi+1;
        SGRIDc=[SGRIDc;   gi   SGRIDc(gi-yGNUM,2)+lfe   SGRIDc(gi-yGNUM,3)
SGRIDc(gi-yGNUM,4)];
    end
end


%%define grids in one full block volume
bsecGnumc=size(SGRIDc,1);
MBGRIDc=SGRIDc;
for k=1:mtdiv
    MBGRIDc=[MBGRIDc;   SGRIDc(:,1)+k*bsecGnumc   SGRIDc(:,2)   SGRIDc(:,3)
SGRIDc(:,4)+k*tme];
end
```

A.5.42 Subroutine mwebGrid.m

```
%%section grids in a web
SGRIDW=[];
SGRIDW=[SGRIDW; nstart xstart ystart zstart];
wgi=nstart;  % SGRIDW index

for i=2:ftdiv+1
    wgi=wgi+1;
    SGRIDW=[SGRIDW;  wgi  SGRIDW(wgi-1,2)  SGRIDW(wgi-1,3)+tfe  SGRIDW(wgi-
1,4)];
end
```

```
for i=1:wldiv
    wgi=wgi+1;
    SGRIDW=[SGRIDW;  wgi  SGRIDW(wgi-1,2)  SGRIDW(wgi-1,3)+lwe  SGRIDW(wgi-
1,4)];
end
for i=1:ftdiv
    wgi=wgi+1;
    SGRIDW=[SGRIDW;  wgi  SGRIDW(wgi-1,2)  SGRIDW(wgi-1,3)+tfe  SGRIDW(wgi-
1,4)];
end
for j=1:wtdiv
    for i=1:yGNUM
        wgi=wgi+1;
        SGRIDW=[SGRIDW;  wgi  SGRIDW(wgi-yGNUM,2)+twe  SGRIDW(wgi-yGNUM,3)
SGRIDW(wgi-yGNUM,4)];
    end
end

%%grids in one web volume
wsecGnum=size(SGRIDW,1);
MBGRIDW=SGRIDW;
for k=1:mtdiv
    MBGRIDW=[MBGRIDW;                          SGRIDW(1:wsecGnum,1)+k*wsecGnum
SGRIDW(1:wsecGnum,2) SGRIDW(1:wsecGnum,3) SGRIDW(1:wsecGnum,4)+k*tme];
end
```

A.5.43 Subroutine hblkElema.m

```
%%define elements for the first half block unit
secGnum=hsecGnuma;
% element group 1
gstart=1; %start grid of the 1st column
ei=0;
LELEM=[];
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup1;
gstart=(wtdiv+fldiv)*yGNUM+1; %start grid of the 3rd column
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 4th column
egroup1;

% element group 2
gstart=1+ftdiv; %start grid of the 1st column (leftmost web)
egroup2;
gstart=gstart+(wtdiv+fldiv)*yGNUM;
egroup2;

% element group3
gstart=1+wtdiv*yGNUM; %start grid of the 1st column (face shell)
egroup3;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup3;

%elements in the first half block volume
layerEnum=size(LELEM,1);
```

```
BELEMHa=LELEM;
for k=1:bhdiv-1
    BELEMHa=[BELEMHa;    LELEM(:,1)+k*layerEnum    LELEM(:,2)+k*hsecGnuma
LELEM(:,3)+k*hsecGnuma LELEM(:,4)+k*hsecGnuma......
                  LELEM(:,5)+k*hsecGnuma              LELEM(:,6)+k*hsecGnuma
LELEM(:,7)+k*hsecGnuma LELEM(:,8)+k*hsecGnuma......
                  LELEM(:,9)+k*hsecGnuma      LELEM(:,10)      LELEM(:,11)
LELEM(:,12)];
end
```

A.5.44 Subroutine fblkElema.m

```
%%define elements for a full block unit (odd layer) in heterogeneous
model
secGnum=bsecGnuma;
% element group 5
gstart=1; %start grid of the 1st column
ei=0;
LELEM=[];
egroup5;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup5;
% element group 1
gstart=1+mtdiv*yGNUM; %start grid of the 1st column
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup1;
gstart=(mtdiv+wtdiv+fldiv2)*yGNUM+1; %start grid of the 3rd column
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 4th column
egroup1;
gstart=(2*mtdiv+2*wtdiv+fldiv2)*yGNUM+1; %start grid of the 5th column
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 6th column
egroup1;
gstart=(2*mtdiv+3*wtdiv+fldiv2+fldiv)*yGNUM+1;  %start  grid  of  the  7th
column
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 8th column
egroup1;

% element group 2
gstart=(mtdiv+wtdiv+fldiv2)*yGNUM+ftdiv+1; %start grid of the 1st column
(leftmost web)
egroup2;
gstart=gstart+(wtdiv+mtdiv)*yGNUM;
egroup2;
gstart=gstart+(wtdiv+fldiv)*yGNUM;
egroup2;

% element group4
gstart=(mtdiv+wtdiv)*yGNUM+1; %start grid of the 1st column
egroup4;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup4;

% element group3
```

```
gstart=(2*mtdiv+3*wtdiv+fldiv2)*yGNUM+1; %start grid of the 1st column
egroup3;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup3;

% element group6
gstart=(mtdiv+2*wtdiv+fldiv2)*yGNUM+1; %start grid of the 1st column
egroup6;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup6;

%elements in one block volume
layerEnum=size(LELEM,1);
BELEMa=LELEM;
for k=1:bhdiv-1
    BELEMa=[BELEMa;     LELEM(:,1)+k*layerEnum     LELEM(:,2)+k*bsecGnuma
LELEM(:,3)+k*bsecGnuma LELEM(:,4)+k*bsecGnuma......
                  LELEM(:,5)+k*bsecGnuma          LELEM(:,6)+k*bsecGnuma
LELEM(:,7)+k*bsecGnuma LELEM(:,8)+k*bsecGnuma......
                  LELEM(:,9)+k*bsecGnuma     LELEM(:,10)     LELEM(:,11)
LELEM(:,12)];
end
```

A.5.45 Subroutine hblkElemb.m

```
%%define elements for the last half block unit
secGnum=hsecGnumb;
% element group 5
gstart=1; %start grid of the 1st column
ei=0;
LELEM=[];
egroup5;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup5;
% element group 1
gstart=1+mtdiv*yGNUM; %start grid of the 1st column
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup1;
gstart=(mtdiv+wtdiv+fldiv)*yGNUM+1; %start grid of the 3rd column
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 4th column
egroup1;

% element group 2
gstart=(mtdiv+wtdiv+fldiv)*yGNUM+ftdiv+1; %start grid of the 1st column
egroup2;


% element group3
gstart=(mtdiv+wtdiv)*yGNUM+1; %start grid of the 1st column
egroup3;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup3;

%elements in the last half block volume
```

```
layerEnum=size(LELEM,1);
BELEMHb=LELEM;
for k=1:bhdiv-1
    BELEMHb=[BELEMHb;    LELEM(:,1)+k*layerEnum    LELEM(:,2)+k*hsecGnumb
LELEM(:,3)+k*hsecGnumb LELEM(:,4)+k*hsecGnumb......
                  LELEM(:,5)+k*hsecGnumb           LELEM(:,6)+k*hsecGnumb
LELEM(:,7)+k*hsecGnumb LELEM(:,8)+k*hsecGnumb......
                  LELEM(:,9)+k*hsecGnumb     LELEM(:,10)     LELEM(:,11)
LELEM(:,12)];
end
```

A.5.46 Subroutine mhblkElema.m

```
%%define elements for the first half block unit
secGnum=hsecGnuma;
% element group 1
gstart=1; %start grid of the 1st column
ei=0;
LELEM=[];
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup1;
gstart=(wtdiv+fldiv)*yGNUM+1; %start grid of the 3rd column
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 4th column
egroup1;

% element group3
gstart=1+wtdiv*yGNUM; %start grid of the 1st column (face shell)
egroup3;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup3;

%elements in the first half block volume
layerEnum=size(LELEM,1);
MBELEMHa=LELEM;
for k=1:mtdiv-1
    MBELEMHa=[MBELEMHa;   LELEM(:,1)+k*layerEnum   LELEM(:,2)+k*hsecGnuma
LELEM(:,3)+k*hsecGnuma LELEM(:,4)+k*hsecGnuma......
                  LELEM(:,5)+k*hsecGnuma           LELEM(:,6)+k*hsecGnuma
LELEM(:,7)+k*hsecGnuma LELEM(:,8)+k*hsecGnuma......
                  LELEM(:,9)+k*hsecGnuma     LELEM(:,10)     LELEM(:,11)
LELEM(:,12)];
end
```

A.5.47 Subroutine mfblkElema.m

```
%%define elements for a full block unit (odd layer) in heterogeneous
model
secGnum=bsecGnuma;
% element group 5
gstart=1; %start grid of the 1st column
ei=0;
LELEM=[];
```

```
egroup5;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup5;
% element group 1
gstart=1+mtdiv*yGNUM; %start grid of the 1st column
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup1;
gstart=(mtdiv+wtdiv+fldiv2)*yGNUM+1; %start grid of the 3rd column
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 4th column
egroup1;
gstart=(2*mtdiv+2*wtdiv+fldiv2)*yGNUM+1; %start grid of the 5th column
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 6th column
egroup1;
gstart=(2*mtdiv+3*wtdiv+fldiv2+fldiv)*yGNUM+1;  %start  grid  of  the  7th
column
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 8th column
egroup1;

% element group4
gstart=(mtdiv+wtdiv)*yGNUM+1; %start grid of the 1st column
egroup4;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup4;

% element group3
gstart=(2*mtdiv+3*wtdiv+fldiv2)*yGNUM+1; %start grid of the 1st column
egroup3;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup3;

% element group6
gstart=(mtdiv+2*wtdiv+fldiv2)*yGNUM+1; %start grid of the 1st column
egroup6;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup6;

%elements in one block volume
layerEnum=size(LELEM,1);
MBELEMa=LELEM;
for k=1:mtdiv-1
    MBELEMa=[MBELEMa;    LELEM(:,1)+k*layerEnum    LELEM(:,2)+k*bsecGnuma
LELEM(:,3)+k*bsecGnuma LELEM(:,4)+k*bsecGnuma......
                  LELEM(:,5)+k*bsecGnuma           LELEM(:,6)+k*bsecGnuma
LELEM(:,7)+k*bsecGnuma LELEM(:,8)+k*bsecGnuma......
                  LELEM(:,9)+k*bsecGnuma     LELEM(:,10)      LELEM(:,11)
LELEM(:,12)];
end
```

A.5.48 Subroutine mhblkElemb.m

```
%%define elements for the last half block unit
secGnum=hsecGnumb;
```

```
% element group 5
gstart=1; %start grid of the 1st column
ei=0;
LELEM=[];
egroup5;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup5;
% element group 1
gstart=1+mtdiv*yGNUM; %start grid of the 1st column
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup1;
gstart=(mtdiv+wtdiv+fldiv)*yGNUM+1; %start grid of the 3rd column
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 4th column
egroup1;

% element group3
gstart=(mtdiv+wtdiv)*yGNUM+1; %start grid of the 1st column
egroup3;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup3;

%elements in the last half block volume
layerEnum=size(LELEM,1);
MBELEMHb=LELEM;
for k=1:mtdiv-1
    MBELEMHb=[MBELEMHb;   LELEM(:,1)+k*layerEnum   LELEM(:,2)+k*hsecGnumb
LELEM(:,3)+k*hsecGnumb LELEM(:,4)+k*hsecGnumb......
                  LELEM(:,5)+k*hsecGnumb          LELEM(:,6)+k*hsecGnumb
LELEM(:,7)+k*hsecGnumb LELEM(:,8)+k*hsecGnumb......
                  LELEM(:,9)+k*hsecGnumb     LELEM(:,10)      LELEM(:,11)
LELEM(:,12)];
end
```

A.5.49 Subroutine fblkElemb.m

```
%%define elements for a full block unit (even layer) in heterogeneous
model
secGnum=bsecGnumb;

% element group 1
gstart=1; %start grid of the 1st column
ei=0;
LELEM=[];
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup1;
gstart=(wtdiv+fldiv)*yGNUM+1; %start grid of the 3rd column
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 4th column
egroup1;
gstart=(2*wtdiv+fldiv+mtdiv)*yGNUM+1; %start grid of the 5th column
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 6th column
egroup1;
```

```
gstart=(mtdiv+3*wtdiv+fldiv+fldiv2)*yGNUM+1;  %start  grid  of  the  7th
column
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 8th column
egroup1;

% element group 2
gstart=1+ftdiv; %start grid of the 1st column (leftmost web)
egroup2;
gstart=gstart+(wtdiv+fldiv)*yGNUM;
egroup2;
gstart=gstart+(wtdiv+mtdiv)*yGNUM;
egroup2;

% element group3
gstart=1+wtdiv*yGNUM; %start grid of the 1st column (face shell)
egroup3;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup3;

% element group4
gstart=(mtdiv+3*wtdiv+fldiv)*yGNUM+1; %start grid of the 1st column
egroup4;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup4;

% element group 5
gstart=(mtdiv+4*wtdiv+fldiv+fldiv2)*yGNUM+1;
egroup5;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup5;

% element group6
gstart=(2*wtdiv+fldiv)*yGNUM+1; %start grid of the 1st column
egroup6;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup6;

%elements in one block volume
layerEnum=size(LELEM,1);
BELEMb=LELEM;
for k=1:bhdiv-1
    BELEMb=[BELEMb;      LELEM(:,1)+k*layerEnum      LELEM(:,2)+k*bsecGnumb
LELEM(:,3)+k*bsecGnumb LELEM(:,4)+k*bsecGnumb......
                   LELEM(:,5)+k*bsecGnumb            LELEM(:,6)+k*bsecGnumb
LELEM(:,7)+k*bsecGnumb LELEM(:,8)+k*bsecGnumb......
                   LELEM(:,9)+k*bsecGnumb      LELEM(:,10)       LELEM(:,11)
LELEM(:,12)];
end
```

A.5.50 Subroutine fblkElemc.m

```
%%define elements for a full block unit (even layer) in heterogeneous
model
secGnum=bsecGnumc;
```

```
% element group 1
gstart=1; %start grid of the 1st column
ei=0;
LELEM=[];
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup1;
gstart=(wtdiv+fldiv)*yGNUM+1; %start grid of the 3rd column
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 4th column
egroup1;
gstart=(2*wtdiv+fldiv+mtdiv)*yGNUM+1; %start grid of the 5th column
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 6th column
egroup1;

% element group 2
gstart=1+ftdiv; %start grid of the 1st column (leftmost web)
egroup2;
gstart=gstart+(wtdiv+fldiv)*yGNUM;
egroup2;
gstart=gstart+(wtdiv+mtdiv)*yGNUM;
egroup2;

% element group3
gstart=1+wtdiv*yGNUM; %start grid of the 1st column (face shell)
egroup3;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup3;
gstart=1+(wtdiv*3+fldiv+mtdiv)*yGNUM; %start grid of the 3rd column
egroup3;
gstart=gstart+ftdiv+wldiv; %start grid of the 4th column
egroup3;

% element group6
gstart=(2*wtdiv+fldiv)*yGNUM+1; %start grid of the 1st column
egroup6;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup6;

%elements in one block volume
layerEnum=size(LELEM,1);
BELEMc=LELEM;
for k=1:bhdiv-1
    BELEMc=[BELEMc;      LELEM(:,1)+k*layerEnum      LELEM(:,2)+k*bsecGnumc
LELEM(:,3)+k*bsecGnumc LELEM(:,4)+k*bsecGnumc......
                  LELEM(:,5)+k*bsecGnumc           LELEM(:,6)+k*bsecGnumc
LELEM(:,7)+k*bsecGnumc LELEM(:,8)+k*bsecGnumc......
                  LELEM(:,9)+k*bsecGnumc     LELEM(:,10)      LELEM(:,11)
LELEM(:,12)];
end
```

A.5.51 Subroutine mfblkElemb.m

```
%%define elements for a full block unit (even layer) in heterogeneous
model
```

```
secGnum=bsecGnumb;

% element group 1
gstart=1; %start grid of the 1st column
ei=0;
LELEM=[];
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup1;
gstart=(wtdiv+fldiv)*yGNUM+1; %start grid of the 3rd column
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 4th column
egroup1;
gstart=(2*wtdiv+fldiv+mtdiv)*yGNUM+1; %start grid of the 5th column
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 6th column
egroup1;
gstart=(mtdiv+3*wtdiv+fldiv+fldiv2)*yGNUM+1;  %start  grid  of  the  7th
column
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 8th column
egroup1;

% element group3
gstart=1+wtdiv*yGNUM; %start grid of the 1st column (face shell)
egroup3;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup3;

% element group4
gstart=(mtdiv+3*wtdiv+fldiv)*yGNUM+1; %start grid of the 1st column
egroup4;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup4;

% element group 5
gstart=(mtdiv+4*wtdiv+fldiv+fldiv2)*yGNUM+1;
egroup5;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup5;

% element group6
gstart=(2*wtdiv+fldiv)*yGNUM+1; %start grid of the 1st column
egroup6;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup6;

%elements in one block volume
layerEnum=size(LELEM,1);
MBELEMb=LELEM;
for k=1:mtdiv-1
    MBELEMb=[MBELEMb;    LELEM(:,1)+k*layerEnum    LELEM(:,2)+k*bsecGnumb
LELEM(:,3)+k*bsecGnumb LELEM(:,4)+k*bsecGnumb......
                  LELEM(:,5)+k*bsecGnumb              LELEM(:,6)+k*bsecGnumb
LELEM(:,7)+k*bsecGnumb LELEM(:,8)+k*bsecGnumb......
                  LELEM(:,9)+k*bsecGnumb      LELEM(:,10)      LELEM(:,11)
LELEM(:,12)];
end
```

A.5.52 Subroutine mfblkElemc.m

```
%%define elements for a full block unit (even layer) in heterogeneous
model
secGnum=bsecGnumc;

% element group 1
gstart=1; %start grid of the 1st column
ei=0;
LELEM=[];
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup1;
gstart=(wtdiv+fldiv)*yGNUM+1; %start grid of the 3rd column
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 4th column
egroup1;
gstart=(2*wtdiv+fldiv+mtdiv)*yGNUM+1; %start grid of the 5th column
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 6th column
egroup1;

% element group3
gstart=1+wtdiv*yGNUM; %start grid of the 1st column (face shell)
egroup3;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup3;
gstart=1+(wtdiv*3+fldiv+mtdiv)*yGNUM; %start grid of the 3rd column
egroup3;
gstart=gstart+ftdiv+wldiv; %start grid of the 4th column
egroup3;

% element group6
gstart=(2*wtdiv+fldiv)*yGNUM+1; %start grid of the 1st column
egroup6;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup6;

%elements in one block volume
layerEnum=size(LELEM,1);
MBELEMc=LELEM;
for k=1:mtdiv-1
    MBELEMc=[MBELEMc;    LELEM(:,1)+k*layerEnum    LELEM(:,2)+k*bsecGnumc
LELEM(:,3)+k*bsecGnumc LELEM(:,4)+k*bsecGnumc......
                  LELEM(:,5)+k*bsecGnumc          LELEM(:,6)+k*bsecGnumc
LELEM(:,7)+k*bsecGnumc LELEM(:,8)+k*bsecGnumc......
                  LELEM(:,9)+k*bsecGnumc      LELEM(:,10)      LELEM(:,11)
LELEM(:,12)];
end
```

A.5.53 Subroutine mwebElem.m

```
%elements in one layer of a web
secGnum=wsecGnum;
% element group 1
m=1; %matrial group number
```

```
gstart=1; %start grid of the 1st column
ei=0;
egroup1;
gstart=gstart+ftdiv+wldiv; %start grid of the 2nd column
egroup1;

%elements in one web volume
layerEnum=size(LELEM,1);
MBELEMW=LELEM;
for k=1:mtdiv-1
    MBELEMW=[MBELEMW;    LELEM(:,1)+k*layerEnum     LELEM(:,2)+k*wsecGnum
LELEM(:,3)+k*wsecGnum LELEM(:,4)+k*wsecGnum......
              LELEM(:,5)+k*wsecGnum                 LELEM(:,6)+k*wsecGnum
LELEM(:,7)+k*wsecGnum LELEM(:,8)+k*wsecGnum......
              LELEM(:,9)+k*wsecGnum      LELEM(:,10)      LELEM(:,11)
LELEM(:,12)];
end
```

A.5.54 Subroutine bbfblkGE.m

```
%%define grids in the cell regions in a full block
%%section grids
BBSGRID=[];
BBSGRID=[BBSGRID; 1 xstart tf bottom-tHb];
gi=1;  % BBSGRID index
% grids of the first cell
for i=2:wldiv+1
    gi=gi+1;
    BBSGRID=[BBSGRID; gi BBSGRID(gi-1,2) BBSGRID(gi-1,3)+lwe BBSGRID(gi-
1,4)];
end
byGNUM=wldiv+1;
for j=1:fldiv
    for i=1:wldiv+1
        gi=gi+1;
        BBSGRID=[BBSGRID;   gi   BBSGRID(gi-byGNUM,2)+lfe    BBSGRID(gi-
byGNUM,3) BBSGRID(gi-byGNUM,4)];
    end
end
% add grids of the second cell
BBSGRID=[BBSGRID;    BBSGRID(:,1)+size(BBSGRID,1)     BBSGRID(:,2)+Lf+2*tw
BBSGRID(:,3) BBSGRID(:,4)];

%%define grids in the cells in one block volume
bbsecGnum=size(BBSGRID,1);
BBGRID=BBSGRID;
for k=1:bhdiv
    BBGRID=[BBGRID;  BBSGRID(:,1)+k*bbsecGnum  BBSGRID(:,2)  BBSGRID(:,3)
BBSGRID(:,4)+k*hbe];
end

%%define elements in the cell regions in a full block
secGnum=bbsecGnum;
% element group 4
m=1;
gstart=1; %start grid of the 1st cell
```

```
ei=0;
LELEM=[];
hoegroup4;
gstart=gstart+(fldiv+1)*byGNUM; %start grid of the 2nd cell
hoegroup4;

%elements in one block volume
layerEnum=size(LELEM,1);
BBELEM=LELEM;
for k=1:bhdiv-1
    BBELEM=[BBELEM;     LELEM(:,1)+k*layerEnum     LELEM(:,2)+k*bbsecGnum
LELEM(:,3)+k*bbsecGnum LELEM(:,4)+k*bbsecGnum......
                  LELEM(:,5)+k*bbsecGnum           LELEM(:,6)+k*bbsecGnum
LELEM(:,7)+k*bbsecGnum LELEM(:,8)+k*bbsecGnum......
                  LELEM(:,9)+k*bbsecGnum     LELEM(:,10)     LELEM(:,11)
LELEM(:,12)];
end
```

A.5.55 Subroutine bbhblkGE.m

```
%%define grids in the cell regions in a half block
%%section grids
BBSGRIDH=[];
BBSGRIDH=[BBSGRIDH; 1 xstart tf bottom-tHb];
gi=1;  % BBSGRIDH index
% grids of the first cell
for i=2:wldiv+1
    gi=gi+1;
    BBSGRIDH=[BBSGRIDH;    gi    BBSGRIDH(gi-1,2)    BBSGRIDH(gi-1,3)+lwe
BBSGRIDH(gi-1,4)];
end
byGNUM=wldiv+1;
for j=1:fldiv
    for i=1:wldiv+1
        gi=gi+1;
        BBSGRIDH=[BBSGRIDH;  gi  BBSGRIDH(gi-byGNUM,2)+lfe  BBSGRIDH(gi-
byGNUM,3) BBSGRIDH(gi-byGNUM,4)];
    end
end

%%define grids in the cells in half block volume
bbsecGnumh=size(BBSGRIDH,1);
BBGRIDH=BBSGRIDH;
for k=1:bhdiv
    BBGRIDH=[BBGRIDH;      BBSGRIDH(:,1)+k*bbsecGnumh      BBSGRIDH(:,2)
BBSGRIDH(:,3) BBSGRIDH(:,4)+k*hbe];
end

%%define elements in the cell regions in a half block
secGnum=bbsecGnumh;
% element group 4
m=1;
gstart=1; %start grid of the 1st cell
ei=0;
LELEM=[];
hoegroup4;
```

```
%elements in half block volume
layerEnum=size(LELEM,1);
BBELEMH=LELEM;
for k=1:bhdiv-1
    BBELEMH=[BBELEMH;    LELEM(:,1)+k*layerEnum    LELEM(:,2)+k*bbsecGnumh
LELEM(:,3)+k*bbsecGnumh LELEM(:,4)+k*bbsecGnumh......
                  LELEM(:,5)+k*bbsecGnumh          LELEM(:,6)+k*bbsecGnumh
LELEM(:,7)+k*bbsecGnumh LELEM(:,8)+k*bbsecGnumh......
                  LELEM(:,9)+k*bbsecGnumh     LELEM(:,10)     LELEM(:,11)
LELEM(:,12)];
end
```

A.5.56 Subroutine heoddBB.m

```
if (mod(left,tLb)==2*tw+Lf) %left distance is the multiple of half block
length
    %define grids and elements in the cell in the first half block
    xstart=left-Lf-tw;
    hebbhblkGEa;
    BBGRIDH1=BBGRIDH;
    BBELEMH1=BBELEMH;

    if Wlength==tLb
        CBBGRID1=[];
        CBBELEM1=[];
    else
        %define grids and elements in the cells in one full block
        xstart=tw;
        hebbfblkGEa;
        %define grids in the cells in the consecutive full blocks
        if (xopen(oi)+wopen(oi)==Wlength)
            obnum=fix(wopen(oi)/tLb);
        else
            obnum=ceil(wopen(oi)/tLb);
        end
        bblkGnum=size(BBGRID,1);
        CBBGRID1=BBGRID;
        for i=1:obnum-1
             CBBGRID1=[CBBGRID1; BBGRID(:,1)+i*bblkGnum BBGRID(:,2)+i*tLb
BBGRID(:,3) BBGRID(:,4)];
        end
        %define elements in the cells in the consecutive full blocks
        bblkEnum=size(BBELEM,1);
        CBBELEM1=BBELEM;
        for i=1:obnum-1
             CBBELEM1=[CBBELEM1;                       BBELEM(:,1)+i*bblkEnum
BBELEM(:,2)+i*bblkGnum                                 BBELEM(:,3)+i*bblkGnum
BBELEM(:,4)+i*bblkGnum......
                  BBELEM(:,5)+i*bblkGnum         BBELEM(:,6)+i*bblkGnum
BBELEM(:,7)+i*bblkGnum BBELEM(:,8)+i*bblkGnum......
                  BBELEM(:,9)+i*bblkGnum     BBELEM(:,10)     BBELEM(:,11)
BBELEM(:,12)];
        end
    end
    %add grids and elements in the cells in the last half block
    if (mod(wopen(oi),tLb)==0 & xopen(oi)+wopen(oi)~=Wlength) |......
            (xopen(oi)+wopen(oi)==Wlength & mod(Wlength,tLb)==0)
        xstart=tw;
        if              (xopen(oi)+wopen(oi)==Wlength-tLb/2              |
xopen(oi)+wopen(oi)==Wlength)
            temp1=fldiv2;
            fldiv2=fldiv;
            temp2=lfe2;
            lfe2=lfe;
            hebbhblkGEb;
            fldiv2=temp1;
            lfe2=temp2;
        else
            hebbhblkGEb;
        end
        bblkGnum=size(CBBGRID1,1);
        if ~isempty(CBBGRID1)
            bbmax=max(CBBGRID1(:,2));
        else
            bbmax=0;
        end
        CBBGRID1=[CBBGRID1;    BBGRIDH(:,1)+bblkGnum    BBGRIDH(:,2)+bbmax
BBGRIDH(:,3) BBGRIDH(:,4)];
        bblkEnum=size(CBBELEM1,1);
        CBBELEM1=[CBBELEM1;  BBELEMH(:,1)+bblkEnum  BBELEMH(:,2)+bblkGnum
BBELEMH(:,3)+bblkGnum BBELEMH(:,4)+bblkGnum......
              BBELEMH(:,5)+bblkGnum                 BBELEMH(:,6)+bblkGnum
BBELEMH(:,7)+bblkGnum BBELEMH(:,8)+bblkGnum......
              BBELEMH(:,9)+bblkGnum      BBELEMH(:,10)      BBELEMH(:,11)
BBELEMH(:,12)];
    end
    %add grids and elements in the cells in the first half block
    bblkGnum=size(BBGRIDH1,1);
    CBBGRID1=[BBGRIDH1;                              CBBGRID1(:,1)+bblkGnum
CBBGRID1(:,2)+max(BBGRIDH1(:,2)) CBBGRID1(:,3) CBBGRID1(:,4)];
    bblkEnum=size(BBELEMH1,1);
    CBBELEM1=[BBELEMH1;  CBBELEM1(:,1)+bblkEnum   CBBELEM1(:,2)+bblkGnum
CBBELEM1(:,3)+bblkGnum CBBELEM1(:,4)+bblkGnum......
          CBBELEM1(:,5)+bblkGnum                 CBBELEM1(:,6)+bblkGnum
CBBELEM1(:,7)+bblkGnum CBBELEM1(:,8)+bblkGnum......
          CBBELEM1(:,9)+bblkGnum      CBBELEM1(:,10)      CBBELEM1(:,11)
CBBELEM1(:,12)];
else %left distance is the multiple of full block length
    %define grids and elements in the cells in one full block
    xstart=left-Lf-tw-tm;
    hebbfblkGEa;
    %define grids in the cells in the consecutive full blocks
    obnum=fix(wopen(oi)/tLb);
    if (xopen(oi)+wopen(oi)==Wlength & mod(Wlength,tLb)==0)
        obnum=obnum-1;
    end
    bblkGnum=size(BBGRID,1);
    CBBGRID1=BBGRID;
    for i=1:obnum
         CBBGRID1=[CBBGRID1;   BBGRID(:,1)+i*bblkGnum   BBGRID(:,2)+i*tLb
BBGRID(:,3) BBGRID(:,4)];
    end
    %define elements in the cells in the consecutive full blocks
    bblkEnum=size(BBELEM,1);
    CBBELEM1=BBELEM;
    for i=1:obnum
         CBBELEM1=[CBBELEM1;                               BBELEM(:,1)+i*bblkEnum
BBELEM(:,2)+i*bblkGnum                                     BBELEM(:,3)+i*bblkGnum
BBELEM(:,4)+i*bblkGnum......
              BBELEM(:,5)+i*bblkGnum                       BBELEM(:,6)+i*bblkGnum
BBELEM(:,7)+i*bblkGnum BBELEM(:,8)+i*bblkGnum......
              BBELEM(:,9)+i*bblkGnum      BBELEM(:,10)       BBELEM(:,11)
BBELEM(:,12)];
    end
    %add grids and elements in the cells in the last half block
    if (mod(wopen(oi),tLb)==tLb/2 & xopen(oi)+wopen(oi)~=Wlength) |......
            (xopen(oi)+wopen(oi)==Wlength & mod(Wlength,tLb)==0)
        xstart=tw;
        if              (xopen(oi)+wopen(oi)==Wlength-tLb/2              |
xopen(oi)+wopen(oi)==Wlength)
            temp1=fldiv2;
            fldiv2=fldiv;
            temp2=lfe2;
            lfe2=lfe;
            hebbhblkGEb;
            fldiv2=temp1;
            lfe2=temp2;
        else
            hebbhblkGEb;
        end
        bblkGnum=size(CBBGRID1,1);
        bbmax=max(CBBGRID1(:,2));
        CBBGRID1=[CBBGRID1;    BBGRIDH(:,1)+bblkGnum    BBGRIDH(:,2)+bbmax
BBGRIDH(:,3) BBGRIDH(:,4)];
        bblkEnum=size(CBBELEM1,1);
        CBBELEM1=[CBBELEM1;  BBELEMH(:,1)+bblkEnum  BBELEMH(:,2)+bblkGnum
BBELEMH(:,3)+bblkGnum BBELEMH(:,4)+bblkGnum......
              BBELEMH(:,5)+bblkGnum                         BBELEMH(:,6)+bblkGnum
BBELEMH(:,7)+bblkGnum BBELEMH(:,8)+bblkGnum......
              BBELEMH(:,9)+bblkGnum      BBELEMH(:,10)      BBELEMH(:,11)
BBELEMH(:,12)];
    end

end
```

A.5.57 Subroutine heevenBB.m

```
if (mod(left,tLb)==2*tw+Lf) %left distance is the multiple of half block
length
    %define grids in the cell in the first half block
    xstart=left-Lf-tw;
    hebbhblkGEa;
    BBGRIDH1=BBGRIDH;
    BBELEMH1=BBELEMH;

    if Wlength==tLb
        CBBGRID2=[];
        CBBELEM2=[];
    else
        %define grids and elements in the cells in one full block
        xstart=tw;
        hebbfblkGEb;
        %define grids in the cells in the consecutive full blocks
        if (xopen(oi)+wopen(oi)==Wlength)
            obnum=fix(wopen(oi)/tLb);
        else
            obnum=ceil(wopen(oi)/tLb);
        end
        bblkGnum=size(BBGRID,1);
        CBBGRID2=BBGRID;
        for i=1:obnum-1
             CBBGRID2=[CBBGRID2; BBGRID(:,1)+i*bblkGnum BBGRID(:,2)+i*tLb
BBGRID(:,3) BBGRID(:,4)];
        end
        %define elements in the cells in the consecutive full blocks
        bblkEnum=size(BBELEM,1);
        CBBELEM2=BBELEM;
        for i=1:obnum-1
             CBBELEM2=[CBBELEM2;                   BBELEM(:,1)+i*bblkEnum
BBELEM(:,2)+i*bblkGnum                             BBELEM(:,3)+i*bblkGnum
BBELEM(:,4)+i*bblkGnum......
                  BBELEM(:,5)+i*bblkGnum           BBELEM(:,6)+i*bblkGnum
BBELEM(:,7)+i*bblkGnum BBELEM(:,8)+i*bblkGnum......
                  BBELEM(:,9)+i*bblkGnum    BBELEM(:,10)     BBELEM(:,11)
BBELEM(:,12)];
        end
    end
    %add grids and elements in the cells in the last half block
    if (mod(wopen(oi),tLb)==0 & xopen(oi)+wopen(oi)~=Wlength) |......
            (xopen(oi)+wopen(oi)==Wlength & mod(Wlength,tLb)==0)
        xstart=tw;
        if           (xopen(oi)+wopen(oi)==Wlength-tLb/2               |
xopen(oi)+wopen(oi)==Wlength)
            templ=fldiv2;
            fldiv2=fldiv;
            hebbhblkGEc;
            fldiv2=templ;
        else
            hebbhblkGEd;
        end
        bblkGnum=size(CBBGRID2,1);
        if ~isempty(CBBGRID2)
            bbmax=max(CBBGRID2(:,2));
        else
            bbmax=0;
        end
        CBBGRID2=[CBBGRID2;    BBGRIDH(:,1)+bblkGnum    BBGRIDH(:,2)+bbmax
BBGRIDH(:,3) BBGRIDH(:,4)];
        bblkEnum=size(CBBELEM2,1);
        CBBELEM2=[CBBELEM2;  BBELEMH(:,1)+bblkEnum  BBELEMH(:,2)+bblkGnum
BBELEMH(:,3)+bblkGnum BBELEMH(:,4)+bblkGnum......
              BBELEMH(:,5)+bblkGnum                BBELEMH(:,6)+bblkGnum
BBELEMH(:,7)+bblkGnum BBELEMH(:,8)+bblkGnum......
              BBELEMH(:,9)+bblkGnum      BBELEMH(:,10)      BBELEMH(:,11)
BBELEMH(:,12)];
    end
    %add grids and elements in the cells in the first half block
    bblkGnum=size(BBGRIDH1,1);
    CBBGRID2=[BBGRIDH1;                            CBBGRID2(:,1)+bblkGnum
CBBGRID2(:,2)+max(BBGRIDH1(:,2)) CBBGRID2(:,3) CBBGRID2(:,4)];
    bblkEnum=size(BBELEMH1,1);
    CBBELEM2=[BBELEMH1;   CBBELEM2(:,1)+bblkEnum   CBBELEM2(:,2)+bblkGnum
CBBELEM2(:,3)+bblkGnum CBBELEM2(:,4)+bblkGnum......
          CBBELEM2(:,5)+bblkGnum                   CBBELEM2(:,6)+bblkGnum
CBBELEM2(:,7)+bblkGnum CBBELEM2(:,8)+bblkGnum......
          CBBELEM2(:,9)+bblkGnum      CBBELEM2(:,10)      CBBELEM2(:,11)
CBBELEM2(:,12)];
else %left distance is the multiple of full block length
    %define grids and elements in the cells in one full block
    xstart=left-Lf-tw-tm;
    hebbfblkGEb;
    %define grids in the cells in the consecutive full blocks
    obnum=fix(wopen(oi)/tLb);
    if (xopen(oi)+wopen(oi)==Wlength & mod(Wlength,tLb)==0)
        obnum=obnum-1;
    end
    bblkGnum=size(BBGRID,1);
    CBBGRID2=BBGRID;
    for i=1:obnum
         CBBGRID2=[CBBGRID2;   BBGRID(:,1)+i*bblkGnum   BBGRID(:,2)+i*tLb
BBGRID(:,3) BBGRID(:,4)];
    end
    %define elements in the cells in the consecutive full blocks
    bblkEnum=size(BBELEM,1);
    CBBELEM2=BBELEM;
    for i=1:obnum
         CBBELEM2=[CBBELEM2;                        BBELEM(:,1)+i*bblkEnum
BBELEM(:,2)+i*bblkGnum                              BBELEM(:,3)+i*bblkGnum
BBELEM(:,4)+i*bblkGnum......
              BBELEM(:,5)+i*bblkGnum                BBELEM(:,6)+i*bblkGnum
BBELEM(:,7)+i*bblkGnum BBELEM(:,8)+i*bblkGnum......
              BBELEM(:,9)+i*bblkGnum      BBELEM(:,10)      BBELEM(:,11)
BBELEM(:,12)];
    end
    %add grids and elements in the cells in the last half block
    if (mod(wopen(oi),tLb)==tLb/2 & xopen(oi)+wopen(oi)~=Wlength) |......
            (xopen(oi)+wopen(oi)==Wlength & mod(Wlength,tLb)==0)
        xstart=tw;
        if             (xopen(oi)+wopen(oi)==Wlength-tLb/2             |
xopen(oi)+wopen(oi)==Wlength)
            temp1=fldiv2;
            fldiv2=fldiv;
            hebbhblkGEc;
            fldiv2=temp1;
        else
            hebbhblkGEd;
        end
        bblkGnum=size(CBBGRID2,1);
        bbmax=max(CBBGRID2(:,2));
        CBBGRID2=[CBBGRID2;   BBGRIDH(:,1)+bblkGnum   BBGRIDH(:,2)+bbmax
BBGRIDH(:,3) BBGRIDH(:,4)];
        bblkEnum=size(CBBELEM2,1);
        CBBELEM2=[CBBELEM2;  BBELEMH(:,1)+bblkEnum  BBELEMH(:,2)+bblkGnum
BBELEMH(:,3)+bblkGnum BBELEMH(:,4)+bblkGnum......
              BBELEMH(:,5)+bblkGnum                BBELEMH(:,6)+bblkGnum
BBELEMH(:,7)+bblkGnum BBELEMH(:,8)+bblkGnum......
              BBELEMH(:,9)+bblkGnum      BBELEMH(:,10)      BBELEMH(:,11)
BBELEMH(:,12)];
    end

end
```

A.5.58 Subroutine hebbhblkGEa.m

```
%%define grids in the cell regions in a half block
%%section grids
BBSGRIDH=[];
BBSGRIDH=[BBSGRIDH; 1 xstart tf zopen(oi)-tHb];
gi=1;  % BBSGRIDH index
% grids of the first cell
for i=2:wldiv+1
    gi=gi+1;
    BBSGRIDH=[BBSGRIDH;    gi    BBSGRIDH(gi-1,2)    BBSGRIDH(gi-1,3)+lwe
BBSGRIDH(gi-1,4)];
end
byGNUM=wldiv+1;
for j=1:fldiv
    for i=1:wldiv+1
        gi=gi+1;
        BBSGRIDH=[BBSGRIDH;  gi  BBSGRIDH(gi-byGNUM,2)+lfe  BBSGRIDH(gi-
byGNUM,3) BBSGRIDH(gi-byGNUM,4)];
    end
end

%%define grids in the cells in half block volume
bbsecGnumh=size(BBSGRIDH,1);
BBGRIDH=BBSGRIDH;
for k=1:bhdiv
    BBGRIDH=[BBGRIDH;      BBSGRIDH(:,1)+k*bbsecGnumh       BBSGRIDH(:,2)
BBSGRIDH(:,3) BBSGRIDH(:,4)+k*hbe];
end

%%define elements in the cell regions in a half block
secGnum=bbsecGnumh;
% element group 7
m=1;
gstart=1; %start grid of the 1st cell
ei=0;
LELEM=[];
heegroup7;

%elements in half block volume
layerEnum=size(LELEM,1);
BBELEMH=LELEM;
for k=1:bhdiv-1
    BBELEMH=[BBELEMH;   LELEM(:,1)+k*layerEnum   LELEM(:,2)+k*bbsecGnumh
LELEM(:,3)+k*bbsecGnumh LELEM(:,4)+k*bbsecGnumh......
                  LELEM(:,5)+k*bbsecGnumh         LELEM(:,6)+k*bbsecGnumh
LELEM(:,7)+k*bbsecGnumh LELEM(:,8)+k*bbsecGnumh......
                  LELEM(:,9)+k*bbsecGnumh     LELEM(:,10)     LELEM(:,11)
LELEM(:,12)];
end
```

A.5.59 Subroutine hebbfblkGEa.m

```
%%define grids in the cell and mortar head joint regions in a full block
%%section grids
```

```
BBSGRID=[];
BBSGRID=[BBSGRID; 1 xstart tf zopen(oi)-tHb];
gi=1;  % BBSGRID index
% grids of the first cell
for i=2:wldiv+1
    gi=gi+1;
    BBSGRID=[BBSGRID; gi BBSGRID(gi-1,2) BBSGRID(gi-1,3)+lwe BBSGRID(gi-
1,4)];
end
byGNUM=wldiv+1;
for j=1:mtdiv
    for i=1:wldiv+1
        gi=gi+1;
        BBSGRID=[BBSGRID;    gi    BBSGRID(gi-byGNUM,2)+tme    BBSGRID(gi-
byGNUM,3) BBSGRID(gi-byGNUM,4)];
    end
end
for j=1:wtdiv
    for i=1:wldiv+1
        gi=gi+1;
        BBSGRID=[BBSGRID;    gi    BBSGRID(gi-byGNUM,2)+twe    BBSGRID(gi-
byGNUM,3) BBSGRID(gi-byGNUM,4)];
    end
end
for j=1:fldiv2
    for i=1:wldiv+1
        gi=gi+1;
        BBSGRID=[BBSGRID;    gi    BBSGRID(gi-byGNUM,2)+lfe2    BBSGRID(gi-
byGNUM,3) BBSGRID(gi-byGNUM,4)];
    end
end
% add grids of the mortar head joint
for i=1:wldiv+1
    gi=gi+1;
    BBSGRID=[BBSGRID;   gi   BBSGRID(gi-byGNUM,2)+tw   BBSGRID(gi-byGNUM,3)
BBSGRID(gi-byGNUM,4)];
end
for j=1:mtdiv
    for i=1:wldiv+1
        gi=gi+1;
        BBSGRID=[BBSGRID;    gi    BBSGRID(gi-byGNUM,2)+tme    BBSGRID(gi-
byGNUM,3) BBSGRID(gi-byGNUM,4)];
    end
end
% add grids of the second cell
for i=1:wldiv+1
    gi=gi+1;
    BBSGRID=[BBSGRID;   gi   BBSGRID(gi-byGNUM,2)+tw   BBSGRID(gi-byGNUM,3)
BBSGRID(gi-byGNUM,4)];
end
for j=1:fldiv
    for i=1:wldiv+1
        gi=gi+1;
        BBSGRID=[BBSGRID;    gi    BBSGRID(gi-byGNUM,2)+lfe    BBSGRID(gi-
byGNUM,3) BBSGRID(gi-byGNUM,4)];
    end
end

%%define grids in the cells and mortar head joints in one block volume
bbsecGnum=size(BBSGRID,1);
```

```
BBGRID=BBSGRID;
for k=1:bhdiv
    BBGRID=[BBGRID;  BBSGRID(:,1)+k*bbsecGnum  BBSGRID(:,2)  BBSGRID(:,3)
BBSGRID(:,4)+k*hbe];
end

%%define elements in the cell and mortar head joint regions in a full
block
secGnum=bbsecGnum;
% element group 8
m=1;
gstart=1; %start grid of the 1st cell
ei=0;
LELEM=[];
heegroup8;
% element group 9
gstart=gstart+mtdiv*byGNUM;
heegroup9;
% element group 10
gstart=gstart+wtdiv*byGNUM;
heegroup10;
% element group 8
gstart=gstart+(fldiv2+1)*byGNUM; %start grid of the mortar head joint
heegroup8;
% element group 7
gstart=gstart+(mtdiv+1)*byGNUM; %start grid of the second cell
heegroup7;

%elements in one block volume
layerEnum=size(LELEM,1);
BBELEM=LELEM;
for k=1:bhdiv-1
    BBELEM=[BBELEM;     LELEM(:,1)+k*layerEnum     LELEM(:,2)+k*bbsecGnum
LELEM(:,3)+k*bbsecGnum LELEM(:,4)+k*bbsecGnum......
                  LELEM(:,5)+k*bbsecGnum           LELEM(:,6)+k*bbsecGnum
LELEM(:,7)+k*bbsecGnum LELEM(:,8)+k*bbsecGnum......
                  LELEM(:,9)+k*bbsecGnum     LELEM(:,10)      LELEM(:,11)
LELEM(:,12)];
end
```

A.5.60 Subroutine hebbhblkGEb.m

```
%%define grids in the cell and mortar head joint regions in a half block
%%section grids
BBSGRIDH=[];
BBSGRIDH=[BBSGRIDH; 1 xstart tf zopen(oi)-tHb];
gi=1;  % BBSGRIDH index
% grids of the first cell
for i=2:wldiv+1
    gi=gi+1;
    BBSGRIDH=[BBSGRIDH;    gi    BBSGRIDH(gi-1,2)    BBSGRIDH(gi-1,3)+lwe
BBSGRIDH(gi-1,4)];
end
byGNUM=wldiv+1;
for j=1:mtdiv
    for i=1:wldiv+1
```

```
        gi=gi+1;
        BBSGRIDH=[BBSGRIDH; gi BBSGRIDH(gi-byGNUM,2)+tme BBSGRIDH(gi-
byGNUM,3) BBSGRIDH(gi-byGNUM,4)];
    end
end
for j=1:wtdiv
    for i=1:wldiv+1
        gi=gi+1;
        BBSGRIDH=[BBSGRIDH; gi BBSGRIDH(gi-byGNUM,2)+twe BBSGRIDH(gi-
byGNUM,3) BBSGRIDH(gi-byGNUM,4)];
    end
end
for j=1:fldiv2
    for i=1:wldiv+1
        gi=gi+1;
        BBSGRIDH=[BBSGRIDH; gi BBSGRIDH(gi-byGNUM,2)+lfe2 BBSGRIDH(gi-
byGNUM,3) BBSGRIDH(gi-byGNUM,4)];
    end
end

%%define grids in the cells in half block volume
bbsecGnumh=size(BBSGRIDH,1);
BBGRIDH=BBSGRIDH;
for k=1:bhdiv
    BBGRIDH=[BBGRIDH;      BBSGRIDH(:,1)+k*bbsecGnumh      BBSGRIDH(:,2)
BBSGRIDH(:,3) BBSGRIDH(:,4)+k*hbe];
end

%%define elements in the cell in a half block
secGnum=bbsecGnumh;
% element group 8
m=1;
gstart=1; %start grid of the 1st cell
ei=0;
LELEM=[];
heegroup8;
% element group 9
gstart=gstart+mtdiv*byGNUM;
heegroup9;
% element group 10
gstart=gstart+wtdiv*byGNUM;
heegroup10;

%elements in half block volume
layerEnum=size(LELEM,1);
BBELEMH=LELEM;
for k=1:bhdiv-1
    BBELEMH=[BBELEMH;   LELEM(:,1)+k*layerEnum   LELEM(:,2)+k*bbsecGnumh
LELEM(:,3)+k*bbsecGnumh LELEM(:,4)+k*bbsecGnumh......
                  LELEM(:,5)+k*bbsecGnumh        LELEM(:,6)+k*bbsecGnumh
LELEM(:,7)+k*bbsecGnumh LELEM(:,8)+k*bbsecGnumh......
                  LELEM(:,9)+k*bbsecGnumh     LELEM(:,10)     LELEM(:,11)
LELEM(:,12)];
end
```

A.5.61 Subroutine hebbfblkGEb.m

```
%%define grids in the cell and mortar head joint regions in a full block
%%section grids
BBSGRID=[];
BBSGRID=[BBSGRID; 1 xstart tf zopen(oi)-tHb];
gi=1;  % BBSGRID index
% grids of the mortar head joint
for i=2:wldiv+1
    gi=gi+1;
    BBSGRID=[BBSGRID; gi BBSGRID(gi-1,2) BBSGRID(gi-1,3)+lwe BBSGRID(gi-
1,4)];
end
byGNUM=wldiv+1;
for j=1:mtdiv
    for i=1:wldiv+1
        gi=gi+1;
        BBSGRID=[BBSGRID;    gi    BBSGRID(gi-byGNUM,2)+tme    BBSGRID(gi-
byGNUM,3) BBSGRID(gi-byGNUM,4)];
    end
end
% grids of the first cell
for i=1:wldiv+1
    gi=gi+1;
    BBSGRID=[BBSGRID;   gi   BBSGRID(gi-byGNUM,2)+tw   BBSGRID(gi-byGNUM,3)
BBSGRID(gi-byGNUM,4)];
end
for j=1:fldiv2
    for i=1:wldiv+1
        gi=gi+1;
        BBSGRID=[BBSGRID;    gi    BBSGRID(gi-byGNUM,2)+lfe2    BBSGRID(gi-
byGNUM,3) BBSGRID(gi-byGNUM,4)];
    end
end
for j=1:wtdiv
    for i=1:wldiv+1
        gi=gi+1;
        BBSGRID=[BBSGRID;    gi    BBSGRID(gi-byGNUM,2)+twe    BBSGRID(gi-
byGNUM,3) BBSGRID(gi-byGNUM,4)];
    end
end
for j=1:mtdiv
    for i=1:wldiv+1
        gi=gi+1;
        BBSGRID=[BBSGRID;    gi    BBSGRID(gi-byGNUM,2)+tme    BBSGRID(gi-
byGNUM,3) BBSGRID(gi-byGNUM,4)];
    end
end

% add grids of the second cell
for i=1:wldiv+1
    gi=gi+1;
    BBSGRID=[BBSGRID;   gi   BBSGRID(gi-byGNUM,2)+tw   BBSGRID(gi-byGNUM,3)
BBSGRID(gi-byGNUM,4)];
end
for j=1:fldiv
    for i=1:wldiv+1
        gi=gi+1;
        BBSGRID=[BBSGRID;   gi   BBSGRID(gi-byGNUM,2)+lfe   BBSGRID(gi-
byGNUM,3) BBSGRID(gi-byGNUM,4)];
    end
end

%%define grids in the cells and mortar head joints in one block volume
bbsecGnum=size(BBSGRID,1);
BBGRID=BBSGRID;
for k=1:bhdiv
    BBGRID=[BBGRID;  BBSGRID(:,1)+k*bbsecGnum  BBSGRID(:,2)  BBSGRID(:,3)
BBSGRID(:,4)+k*hbe];
end

%%define elements in the cell and mortar head joint regions in a full
block
secGnum=bbsecGnum;
% element group 8
m=1;
gstart=1; %start grid of the 1st cell
ei=0;
LELEM=[];
heegroup8;
% element group 10
gstart=gstart+(mtdiv+1)*byGNUM;
heegroup10;
% element group 9
gstart=gstart+fldiv2*byGNUM;
heegroup9;
% element group 8
gstart=gstart+wtdiv*byGNUM; %start grid of the mortar head joint
heegroup8;
% element group 7
gstart=gstart+(mtdiv+1)*byGNUM; %start grid of the second cell
heegroup7;

%elements in one block volume
layerEnum=size(LELEM,1);
BBELEM=LELEM;
for k=1:bhdiv-1
    BBELEM=[BBELEM;   LELEM(:,1)+k*layerEnum   LELEM(:,2)+k*bbsecGnum
LELEM(:,3)+k*bbsecGnum LELEM(:,4)+k*bbsecGnum......
                  LELEM(:,5)+k*bbsecGnum        LELEM(:,6)+k*bbsecGnum
LELEM(:,7)+k*bbsecGnum LELEM(:,8)+k*bbsecGnum......
                  LELEM(:,9)+k*bbsecGnum    LELEM(:,10)    LELEM(:,11)
LELEM(:,12)];
end
```

A.5.62 Subroutine hebbhblkGEc.m

```
%%define grids in the cell and mortar head joint regions in a half block
%%section grids
BBSGRIDH=[];
BBSGRIDH=[BBSGRIDH; 1 xstart tf zopen(oi)-tHb];
gi=1;  % BBSGRIDH index
% grids of the mortar head joint
for i=2:wldiv+1
    gi=gi+1;
    BBSGRIDH=[BBSGRIDH;    gi    BBSGRIDH(gi-1,2)    BBSGRIDH(gi-1,3)+lwe
BBSGRIDH(gi-1,4)];
end
byGNUM=wldiv+1;
for j=1:mtdiv
    for i=1:wldiv+1
        gi=gi+1;
        BBSGRIDH=[BBSGRIDH;  gi  BBSGRIDH(gi-byGNUM,2)+tme  BBSGRIDH(gi-
byGNUM,3) BBSGRIDH(gi-byGNUM,4)];
    end
end
% grids of the first cell
for i=1:wldiv+1
    gi=gi+1;
    BBSGRIDH=[BBSGRIDH; gi BBSGRIDH(gi-byGNUM,2)+tw BBSGRIDH(gi-byGNUM,3)
BBSGRIDH(gi-byGNUM,4)];
end
for j=1:fldiv
    for i=1:wldiv+1
        gi=gi+1;
        BBSGRIDH=[BBSGRIDH;  gi  BBSGRIDH(gi-byGNUM,2)+lfe  BBSGRIDH(gi-
byGNUM,3) BBSGRIDH(gi-byGNUM,4)];
    end
end

%%define grids in the cells in half block volume
bbsecGnumh=size(BBSGRIDH,1);
BBGRIDH=BBSGRIDH;
for k=1:bhdiv
    BBGRIDH=[BBGRIDH;     BBSGRIDH(:,1)+k*bbsecGnumh      BBSGRIDH(:,2)
BBSGRIDH(:,3) BBSGRIDH(:,4)+k*hbe];
end

%%define elements in the cell in a half block
secGnum=bbsecGnumh;
% element group 8
m=1;
gstart=1; %start grid of the 1st cell
ei=0;
LELEM=[];
heegroup8;
% element group 10
gstart=gstart+(mtdiv+1)*byGNUM;
heegroup10;

%elements in half block volume
layerEnum=size(LELEM,1);
BBELEMH=LELEM;
for k=1:bhdiv-1
    BBELEMH=[BBELEMH;   LELEM(:,1)+k*layerEnum   LELEM(:,2)+k*bbsecGnumh
LELEM(:,3)+k*bbsecGnumh LELEM(:,4)+k*bbsecGnumh.....
                  LELEM(:,5)+k*bbsecGnumh         LELEM(:,6)+k*bbsecGnumh
LELEM(:,7)+k*bbsecGnumh LELEM(:,8)+k*bbsecGnumh.....
                  LELEM(:,9)+k*bbsecGnumh     LELEM(:,10)     LELEM(:,11)
LELEM(:,12)];
end
```

A.5.63 Subroutine hebbhblkGEd.m

```
%%define grids in the cell and mortar head joint regions in a half block
%%section grids
BBSGRIDH=[];
BBSGRIDH=[BBSGRIDH; 1 xstart tf zopen(oi)-tHb];
gi=1;  % BBSGRIDH index
% grids of the mortar head joint
for i=2:wldiv+1
    gi=gi+1;
    BBSGRIDH=[BBSGRIDH;    gi    BBSGRIDH(gi-1,2)    BBSGRIDH(gi-1,3)+lwe
BBSGRIDH(gi-1,4)];
end
byGNUM=wldiv+1;
for j=1:mtdiv
    for i=1:wldiv+1
        gi=gi+1;
        BBSGRIDH=[BBSGRIDH;  gi  BBSGRIDH(gi-byGNUM,2)+tme  BBSGRIDH(gi-
byGNUM,3) BBSGRIDH(gi-byGNUM,4)];
    end
end
% grids of the first cell
for i=1:wldiv+1
    gi=gi+1;
    BBSGRIDH=[BBSGRIDH; gi BBSGRIDH(gi-byGNUM,2)+tw BBSGRIDH(gi-byGNUM,3)
BBSGRIDH(gi-byGNUM,4)];
end
for j=1:fldiv2
    for i=1:wldiv+1
        gi=gi+1;
        BBSGRIDH=[BBSGRIDH;  gi  BBSGRIDH(gi-byGNUM,2)+lfe2  BBSGRIDH(gi-
byGNUM,3) BBSGRIDH(gi-byGNUM,4)];
    end
end
for j=1:wtdiv
    for i=1:wldiv+1
        gi=gi+1;
        BBSGRIDH=[BBSGRIDH;  gi  BBSGRIDH(gi-byGNUM,2)+twe  BBSGRIDH(gi-
byGNUM,3) BBSGRIDH(gi-byGNUM,4)];
    end
end
for j=1:mtdiv
    for i=1:wldiv+1
        gi=gi+1;
        BBSGRIDH=[BBSGRIDH;  gi  BBSGRIDH(gi-byGNUM,2)+tme  BBSGRIDH(gi-
byGNUM,3) BBSGRIDH(gi-byGNUM,4)];
    end
end

%%define grids in the cells in half block volume
bbsecGnumh=size(BBSGRIDH,1);
BBGRIDH=BBSGRIDH;
for k=1:bhdiv
    BBGRIDH=[BBGRIDH;      BBSGRIDH(:,1)+k*bbsecGnumh       BBSGRIDH(:,2)
BBSGRIDH(:,3) BBSGRIDH(:,4)+k*hbe];
end
%%define elements in the cell in a half block
secGnum=bbsecGnumh;
% element group 8
```

```
m=1;
gstart=1; %start grid of the 1st cell
ei=0;
LELEM=[];
heegroup8;
% element group 10
gstart=gstart+(mtdiv+1)*byGNUM;
heegroup10;
% element group 9
gstart=gstart+fldiv2*byGNUM;
heegroup9;
% element group 8
gstart=gstart+wtdiv*byGNUM; %start grid of the mortar head joint
heegroup8;

%elements in half block volume
layerEnum=size(LELEM,1);
BBELEMH=LELEM;
for k=1:bhdiv-1
    BBELEMH=[BBELEMH;    LELEM(:,1)+k*layerEnum    LELEM(:,2)+k*bbsecGnumh
LELEM(:,3)+k*bbsecGnumh LELEM(:,4)+k*bbsecGnumh......
                  LELEM(:,5)+k*bbsecGnumh          LELEM(:,6)+k*bbsecGnumh
LELEM(:,7)+k*bbsecGnumh LELEM(:,8)+k*bbsecGnumh......
                  LELEM(:,9)+k*bbsecGnumh     LELEM(:,10)     LELEM(:,11)
LELEM(:,12)];
end
```

A.5.64 Subroutine egroup1.m

```
for i=1:ftdiv+1
    for j=1:wtdiv+1
        EG1(i,j)=(j-1)*yGNUM+gstart+i-1;
    end
end
for i=1:ftdiv
    for j=1:wtdiv
        ei=ei+1;
        LELEM(ei,:)=[ei   EG1(i,j)   EG1(i,j+1)   EG1(i+1,j+1)   EG1(i+1,j)
EG1(i,j)+secGnum         EG1(i,j+1)+secGnum         EG1(i+1,j+1)+secGnum
EG1(i+1,j)+secGnum 1 m m];
    end
end
```

A.5.65 Subroutine egroup2.m

```
for i=1:wldiv+1
    for j=1:wtdiv+1
        EG2(i,j)=(j-1)*yGNUM+gstart+i-1;
    end
end
for i=1:wldiv
    for j=1:wtdiv
        ei=ei+1;
        LELEM(ei,:)=[ei   EG2(i,j)   EG2(i,j+1)   EG2(i+1,j+1)   EG2(i+1,j)
EG2(i,j)+secGnum          EG2(i,j+1)+secGnum           EG2(i+1,j+1)+secGnum
EG2(i+1,j)+secGnum 2 1 1];
    end
end
```

A.5.66 Subroutine egroup3.m

```
for i=1:ftdiv+1
    for j=1:fldiv+1
        EG3(i,j)=(j-1)*yGNUM+gstart+i-1;
    end
end
for i=1:ftdiv
    for j=1:fldiv
        ei=ei+1;
        LELEM(ei,:)=[ei   EG3(i,j)   EG3(i,j+1)   EG3(i+1,j+1)   EG3(i+1,j)
EG3(i,j)+secGnum          EG3(i,j+1)+secGnum           EG3(i+1,j+1)+secGnum
EG3(i+1,j)+secGnum 3 m m];
    end
end
```

A.5.67 Subroutine egroup4.m

```
for i=1:ftdiv+1
    for j=1:fldiv2+1
        EG4(i,j)=(j-1)*yGNUM+gstart+i-1;
    end
end
for i=1:ftdiv
    for j=1:fldiv2
        ei=ei+1;
        LELEM(ei,:)=[ei   EG4(i,j)   EG4(i,j+1)   EG4(i+1,j+1)   EG4(i+1,j)
EG4(i,j)+secGnum          EG4(i,j+1)+secGnum           EG4(i+1,j+1)+secGnum
EG4(i+1,j)+secGnum 4 m m];
    end
end
```

A.5.68 Subroutine egroup5.m

```
for i=1:ftdiv+1
    for j=1:mtdiv+1
        EG5(i,j)=(j-1)*yGNUM+gstart+i-1;
    end
end

for i=1:ftdiv
    for j=1:mtdiv
        ei=ei+1;
        LELEM(ei,:)=[ei   EG5(i,j)   EG5(i,j+1)   EG5(i+1,j+1)   EG5(i+1,j)
EG5(i,j)+secGnum          EG5(i,j+1)+secGnum           EG5(i+1,j+1)+secGnum
EG5(i+1,j)+secGnum 5 m m];
    end
end
```

A.5.69 Subroutine egroup6.m

```
for i=1:ftdiv+1
    for j=1:mtdiv+1
        EG6(i,j)=(j-1)*yGNUM+gstart+i-1;
    end
end
for i=1:ftdiv
    for j=1:mtdiv
        ei=ei+1;
        LELEM(ei,:)=[ei   EG6(i,j)   EG6(i,j+1)   EG6(i+1,j+1)   EG6(i+1,j)
EG6(i,j)+secGnum          EG6(i,j+1)+secGnum           EG6(i+1,j+1)+secGnum
EG6(i+1,j)+secGnum 6 2 2];
    end
end
```

A.5.70 Subroutine hoegroup4.m

```
for i=1:wldiv+1
    for j=1:fldiv+1
        hoEG4(i,j)=(j-1)*byGNUM+gstart+i-1;
    end
end
for i=1:wldiv
    for j=1:fldiv
        ei=ei+1;
        LELEM(ei,:)=[ei     hoEG4(i,j)     hoEG4(i,j+1)      hoEG4(i+1,j+1)
hoEG4(i+1,j)            hoEG4(i,j)+secGnum            hoEG4(i,j+1)+secGnum
hoEG4(i+1,j+1)+secGnum hoEG4(i+1,j)+secGnum 4 1 1];
    end
end
```

A.5.71 Subroutine heegroup7.m

```
for i=1:wldiv+1
    for j=1:fldiv+1
        heEG7(i,j)=(j-1)*byGNUM+gstart+i-1;
    end
end
for i=1:wldiv
    for j=1:fldiv
        ei=ei+1;
        LELEM(ei,:)=[ei     heEG7(i,j)     heEG7(i,j+1)     heEG7(i+1,j+1)
heEG7(i+1,j)            heEG7(i,j)+secGnum              heEG7(i,j+1)+secGnum
heEG7(i+1,j+1)+secGnum heEG7(i+1,j)+secGnum 7 1 1];
    end
end
```

A.5.72 Subroutine heegroup8.m

```
for i=1:wldiv+1
    for j=1:mtdiv+1
        heEG8(i,j)=(j-1)*byGNUM+gstart+i-1;
    end
end
for i=1:wldiv
    for j=1:mtdiv
        ei=ei+1;
        LELEM(ei,:)=[ei     heEG8(i,j)     heEG8(i,j+1)     heEG8(i+1,j+1)
heEG8(i+1,j)            heEG8(i,j)+secGnum              heEG8(i,j+1)+secGnum
heEG8(i+1,j+1)+secGnum heEG8(i+1,j)+secGnum 8 1 1];
    end
end
```

A.5.73 Subroutine heegroup9.m

```
for i=1:wldiv+1
    for j=1:wtdiv+1
        heEG9(i,j)=(j-1)*byGNUM+gstart+i-1;
    end
end
for i=1:wldiv
    for j=1:wtdiv
        ei=ei+1;
        LELEM(ei,:)=[ei     heEG9(i,j)     heEG9(i,j+1)     heEG9(i+1,j+1)
heEG9(i+1,j)            heEG9(i,j)+secGnum              heEG9(i,j+1)+secGnum
heEG9(i+1,j+1)+secGnum heEG9(i+1,j)+secGnum 9 1 1];
    end
end
```

A.5.74 Subroutine heegroup10.m

```
for i=1:wldiv+1
    for j=1:fldiv2+1
        heEG10(i,j)=(j-1)*byGNUM+gstart+i-1;
    end
end
for i=1:wldiv
    for j=1:fldiv2
        ei=ei+1;
        LELEM(ei,:)=[ei    heEG10(i,j)    heEG10(i,j+1)    heEG10(i+1,j+1)
heEG10(i+1,j)          heEG10(i,j)+secGnum          heEG10(i,j+1)+secGnum
heEG10(i+1,j+1)+secGnum heEG10(i+1,j)+secGnum 10 1 1];
    end
end
```

A.5.75 Subroutine mergeRows.m

```
function [B, rowi, posi] = mergeRows(A)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
% This function removes the duplicated rows in a matrix. The tolerance
% between the two entries is set as 1E-3.
% [B, rowi, posi] = mergeRows(A) returns a matrix B with unique rows and
% the indices rowi and posi such that B = A(rowi, :) and A = B(posi,:)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%

[m, n] = size(A);
rown = m;
if rown == 1
    A = A.';
end
[m, n] = size(A);

newA = A;

rowi = (1:m).';
for j=1:n
%for j = n:-1:1
       [C, i] = sort(A(:, j));
       A = A(i, :);
       rowi = rowi(i);
end

d = diff(A);

if n == 1
       f = find(abs(d) < 1E-3);
else
       f = find(all(abs(d.') < 1E-3).');
end


posi=rowi;
[c,k]=sort(posi);
if any(f)
    rowi(f+1) = [];
       rowi = sort(rowi);
    newA = newA(rowi, :);

    for j=1:size(f)
        posi(f(j)+1)=posi(f(j));
    end
    pi=[];
    for i=1:size(posi,1)
        pi=[pi; find(rowi(:,1)==posi(i,1))];
    end
    posi=pi(k);
end

if rown == 1
    newA = newA.';
end

if nargout > 0
     B = newA;
else
     disp(newA)
end
```