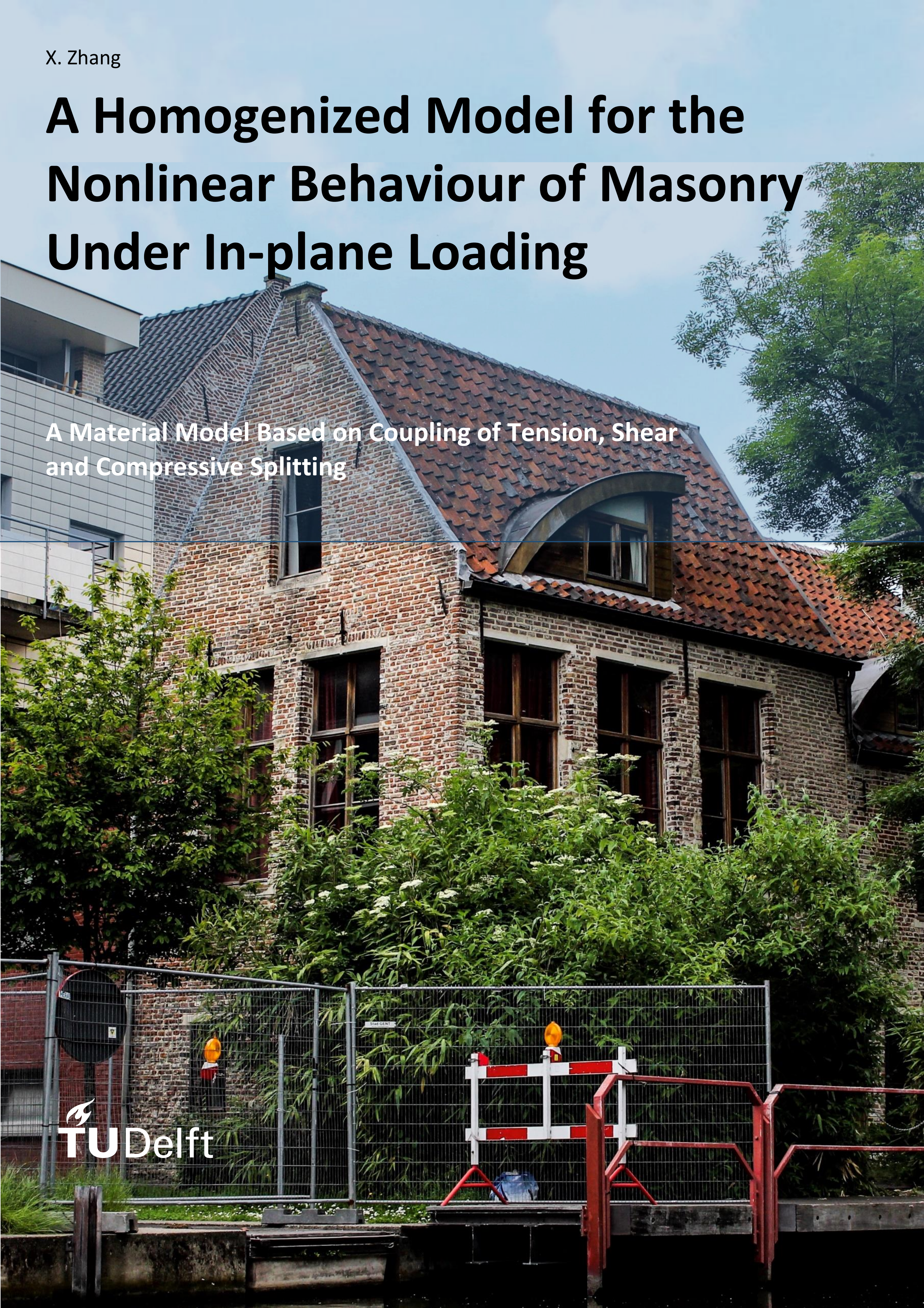
X. Zhang

A Homogenized Model for the Nonlinear Behaviour of Masonry Under In-plane Loading

A Material Model Based on Coupling of Tension, Shear and Compressive Splitting



A Homogenized Model for the Nonlinear Behaviour of Masonry Under In-plane Loading

A Material Model Based on Coupling of Tension, Shear and Compressive Splitting

By

Xinrui Zhang

in partial fulfilment of the requirements for the degree of

Master of Science
in Civil Engineering

at the Delft University of Technology,
to be defended publicly on Thursday 29th July 2021, at 14.00

Student number:	4915879	
Kick-off date:	24th September 2020	
Final defense:	29th July 2021	
Thesis committee:	Prof.dr.ir. J.G. Rots,	TU Delft
	Dr. F. Messali,	TU Delft
	M. Sousamli, MSc,	TU Delft
	Dr.ir. G.J.P. Ravenshorst	TU Delft

TUDelft

Preface

I take this opportunity to express my appreciation to all people who have participated in the successful completion of this thesis.

First of all, I am deeply grateful to all of my committee members for guiding me to move on and finish profound research. Their academic thinking inspires me to have new ideas on improving the concepts of works, and their support helps me better understand how to become a good researcher. Prof. Jan Rots, the Chair of my committee assessment group, always encourages me when I am confused about my works. His enormous knowledge brings me to see a new “masonry world”, and his enthusiasm will always excite me to go further on computational modelling in my future career. I also want to express my gratitude to my supervisor Dr. Francesco Messali and daily supervisor Marianthi Sousamli. They give me excellent guidance throughout the whole thesis period, and they give me adequate freedom on my thesis. The lockdown during the Covid-19 Pandemic period brought inconvenience to communicate with each other face to face. Francesco and Marianthi tried their best to support me as much as possible whenever I need help on my thesis in such a difficult period. It is my first time finishing such a large project in English, so I wrote my thesis in extreme disorder at the beginning. However, they showed their most patience and faith to me and guided me to arrange the thesis academically. My language ability and critical thinking cannot be improved such much without their help. I also want to thank Dr. ir. Geert Ravenshorst, one of my committee members from the section of Biobased Structures and Materials. His deep thinking on the research results inspires me to focus on the validations and applications of models, not just on the theory.

Furthermore, this thesis is impossible to be finished without Zucchini and Lourenço’s works on deriving such excellent constitutive models. The validations cannot be assessed without the experimental results tested by Dr. Rita Esposito, Dr. Francesco Messali, Prof. Jan Rots, and all other staff at Delft University of Technology. Very thanks for their fantastic works.

Finally, I want to give my sincere thanks to my parents and all of my friends. My parents gave me great financial support and accompanied me when I was depressed and fight against my psychological problems. They encouraged me and gave me much confidence, even if they did not understand what I am doing in the thesis. My friends, especially those who also studied mechanics, shared their experience and knowledge without hesitation when I needed help. Thank all the teachers in my life. They taught me how to learn and built me up the most enthusiasm for studying.

X.Zhang
Delft, July 2021

Abstract

Masonry is one of the most common building materials globally due to its ease of construction, price, durability and fire resistance. Its heterogeneity and orthotropic nature make its mechanical behaviour rather complex, highlighting the importance of an appropriate constitutive model to describe such behaviour accurately. In literature, most of the existing constitutive models for masonry fall in one of the two following categories: macro-models or micro-models. The micro-models (also called block-based models), which explicitly describe masonry's geometrical and material heterogeneities, exhibit higher accuracy but require higher numerical efforts when simulating masonry's mechanical behaviour. The macro-models (also called continuum models), which model masonry as homogeneous material and smear out the damage over the continua, are less accurate but offer a good compromise between accuracy and numerical efficiency. They are therefore used more often to simulate masonry structures. However, most of the macro-models are not capturing well the damage localization occurring along the mortar joints and the energy dissipation in the bricks and mortars. To increase the applicability of continuum models, the components' damage localization and energy dissipation should be improved. This thesis presents a homogenized constitutive model for applications on masonry structures under in-plane loading.

The developed homogenized material model for masonry includes the description of shearing, tensile cracking, crushing and splitting. Inspired by the micro-mechanical models of Zucchini and Lourenço, four models were developed in this thesis to describe the different types of in-plane failure of masonry, naming: tensile failure of bed joints, horizontal shear sliding of bed joints, tensile cracking of unit, diagonal tensile cracking failure, masonry crushing failure. The first model is derived for the masonry's pure shear behaviour, where the shear sliding failure is introduced. The second one is derived for the horizontal tensile behaviour, where the vertical tensile cracking of brick units and the vertical joints is proposed. The third one is derived for the vertical compression behaviour, where masonry crushing failure is adopted. The fourth one is coupling the failure mechanisms described in the previous three models with a novel algorithm. Additionally, the diagonal tension cracking failure and the horizontal tensile cracking of horizontal joints are incorporated in the fourth and final material model.

To derive these models, first, a representative volume element (RVE) was selected for running bond wall, where the bricks are staggered by half-length of brick from the adjoining courses above and below. Each RVE consists of two-quarters of bricks connected through a bed joint, and each one is connected to a head joint on one side. For each of the models, the active internal stresses of each component (brick, bed, head or cross joint) are calculated through the compatibility and equilibrium equations resulting from the assumed deformed mechanisms. Different damage state variables are introduced for every component in the damage model, where exponential softening is assumed for tension and shear. Additionally, a Ducker-Prager yield criterion with bi-parabolic hardening is used in combination with an explicit Euler-forward algorithm to describe the elastoplastic behaviour of the material in compression. As a result, the material model's constitutive law is obtained with the homogenization procedures after coupling the damage and plastic model together by a specific algorithm originally introduced by Zucchini and Lourenço.

The constitutive equations for model 1 (shear), model 2 (tension), model 3 (compression) and model 4 (coupled) were coded successfully in MATLAB. The components' shearing, tensile cracking, crushing and splitting failures are correctly modelled analytically with an algorithm, which can be applied to simplify the micro-mechanical model. Besides, constitutive laws of models 1 and 2 were also implemented successfully in the finite element software DIANA version 10.4 (check). The ability of the model to capture the failure of the components in shear, tensile cracking or crushing was examined through simple analytical applications. Moreover, the model was compared against experimental results from tests performed on masonry wallets under compressive loading; the model was able to predict the strength of the specimens satisfactorily.

Therefore, this alternative constitutive material model can adequately simulate masonry's behaviours with a simpler algorithm than the previous model. Meanwhile, the component's elastic and elastoplastic behaviours can be simulated in more detail in this model, as the case that the horizontal joint is first damaged in shear and compressive splitting effects are additionally included.

Contents

List of figures

List of tables

1. Introduction

Masonry is one of the most common materials used for structures in the Netherlands and worldwide. Masonry structures have several advantages. For instance, masonry structures' fire resistance and durability are better than other types of structures as masonry's major components -the bricks- are usually made by stable substances, such as stone or clay, which are fire-resistant materials and are hard to have chemical reactions with water and air.

Masonry is an inhomogeneous material consisting of units (brick or stone block) and joints (dry or mortar) [1]. Although some variability in the mechanical and geometrical properties of the constituents is possible, especially for clay bricks and lime mortar, the construction phase introduces the biggest variability in the construction. The mortar joints are more dependent on manual action [2]. For example, variability in the thickness of the mortar joints or the bond between mortar and bricks can lead to layers with different mechanical properties. Therefore, the mechanical behaviour of masonry is rather complex due to its composite and orthotropic nature. That is why the investigation of constitutive models that can reproduce such behaviour is of relevance.

1.1. Background

Masonry is a distinctive quasi-brittle material since its elastic behaviours and strengths are different in different directions [3]. The different properties of its constituents make masonry an anisotropic material with weak layers, where usually damage concentrates. Based on the experimental results shown in [4, 5], five failure mechanisms have been identified in (unreinforced) masonry [6, 7, 8].

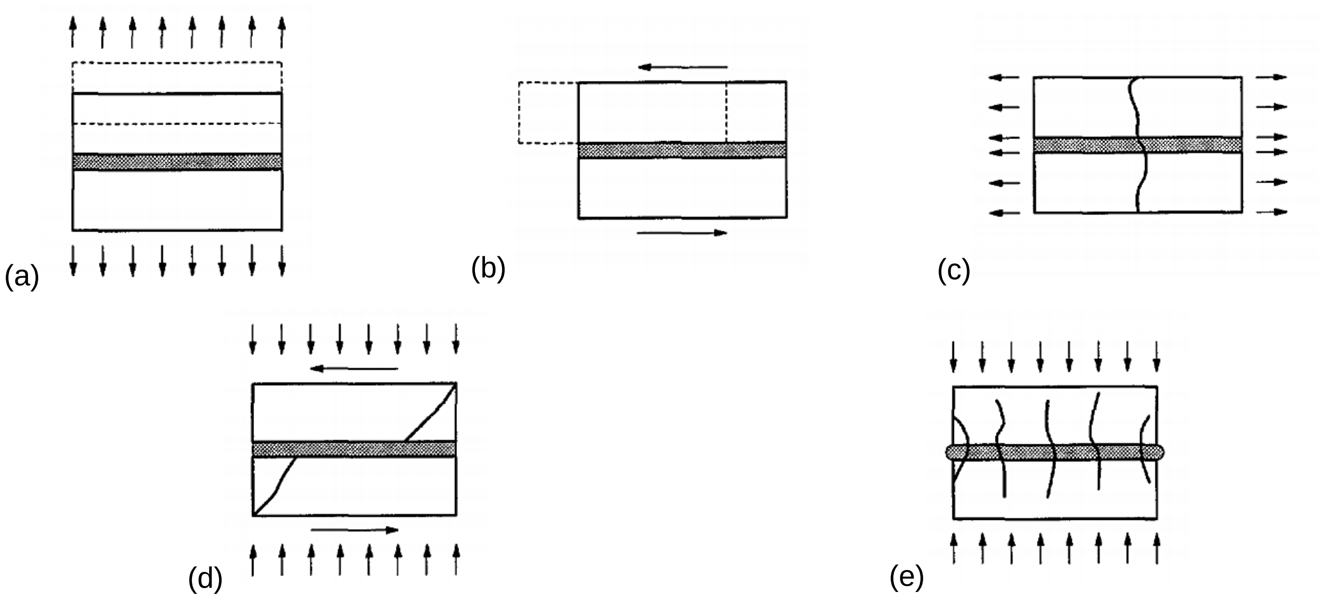


Figure 1-1 Five failure mechanisms [8]

(a) Tensile failure of bed joints, which indicates the potential horizontal cracks generated in joints in tension;
(b) Horizontal shear sliding of bed joints, caused by shear cracks generated in joints;
(c) Tensile cracking of unit, where vertical cracks occur in the unit;
(d) Diagonal tensile cracking failure, caused by shear behaviour at the brick-mortar interface and vertical cracks passing either through the head joints or the brick units;
(e) Masonry crushing failure, produced by micro-cracks generated in units.

The failure mechanisms from these failure modes could be incorporated in constitutive models mathematically by using specific modelling strategies. In 2019, D'Altri et al. classified these strategies into four categories [1]: block-based models, continuum models, macroelement models and geometry-based models.

(1) Block-based models: the blocks in the masonry modelled by the block-based strategy are assumed to be rigid or deformable. Page first introduced this approach by considering masonry as a so-called "texture continuum" [6]. In this assemblage, the elastic brick elements act in conjunction with linkage elements simulating the mortar joints. Based on Page's work, many block-based models have been introduced and developed, such as interface element-based approaches achieved by introducing zero-thickness interface elements [9, 10, 11, 12], contact-based approaches based on contact mechanics [13, 14], textured continuum-based approaches by introducing bricks and mortars separately in FEM framework [15, 16].

(2) Continuum models: the masonry is assumed to be a continuum deformable-body in continuum approaches. Mesh discretization of this type of strategy does not have to describe the geometrical heterogeneities of masonry [1]. Therefore, the computational cost of continuum models is generally lower than for block-based models. However, the definition of suitable constitutive laws for masonry is a challenging task due to its complex and orthotropic mechanical properties. There are two main approaches to derive these homogeneous constitutive laws categorized in [1]: The direct approach, where the mechanical properties could be calibrated through experimental data or other data without resorting to the Representative volume elements [1] -based homogenization procedures [17, 18, 19]; The homogenization procedures and multi-scale approaches, where the homogeneous constitutive laws of structural-scale models can be proposed from homogenization techniques (typically based on RVEs) [20, 21, 22, 23].

(3) Macroelement models: the structure is idealized into plane-scale structural components based on phenomenological or mechanical-based nonlinear response [1]. These models are mainly used to analyze the global seismic response of masonry buildings, and some examples can be found in [24, 25].

(4) Geometry-based models: The structure is considered to be a rigid body in geometry-based models. Therefore, the geometry of the structure is the only input of these approaches beyond certain loading conditions. For this reason, geometry-based modelling strategies are typically used to investigate structural stability and/or collapse through limit analysis-based solutions, and some examples can be found in [26, 27].

The block-based models generally have higher accuracy when simulating masonry's mechanical behaviours. However, their application at structural level is practically inconvenient due to the large size and the complexity of the masonry buildings, so the simulations would take a long time to run with considerable numerical efforts. Therefore, the continuum approaches are selected in this thesis to model masonry structures since they offer a good compromise between accuracy and efficiency.

In 1999, J. Lopez et al. presented a micro-mechanical material model based on a representative cell's compatibility and equilibrium equations based on the multi-scale approach [28]. The bricks and mortars' deformations and micro-constitutive laws in this material model were introduced separately. However, the deformations micro-constitutive laws were introduced as a whole in the macro-models proposed by P.B. Lourenço and Anthoine et al. based on the composite theory [8, 29]. In the second approach, the micro-mechanical material model can have higher accuracy, especially when the differences between the stiffness of the bricks and mortars are large.

In 2002, A. Zucchini and P.B. Lourenço focused on the typically periodic masonry structures with the staggered brick alignment and developed a novel material model based on Lopez's multi-scale approach. They defined the overlaps of horizontal and vertical joints as cross joints and derived correlated elastic properties of a representative volume element in the normal and shear direction according to their assumed deformed mechanisms [30]. After that, they introduced potential tensile (or shear) cracks, which should be vertically (or horizontally) located at components, by a non-linear homogenization procedure [20]. In this procedure, they first introduced several damage state variables for the bricks and mortars to represent the damage status of components. Then, the exponential law of the damage state variables and the stresses proposed in [31] were implemented in the elastic properties of RVE derived in [30]. Finally, the damage status of the components was incorporated into RVE by compatibility and equilibrium equations. In 2007, Zucchini and Lourenço incorporated the elastoplastic phase of their model under pure compressive loading in the

[1] Abbreviation: RVE. In theory of composite material, the representative volume element (also called the representative elementary volume (REV) or the unit cell) is the smallest volume whose properties could be representative of the whole. (From Wikipedia)
https://en.wikipedia.org/wiki/Representative_elementary_volume

vertical direction [32]. They implemented Drucker-Prager yield criteria in every component with an implicit Euler-backward algorithm proposed in [8]. In 2009, they introduced an extended material model based on their previous work [30, 20, 32] by extending their quarter basic cell to a larger RVE to propose coupled behaviour in the normal and shear direction [21]. They considered the shear behaviour of vertical joints and assumed deformations in RVE to be antisymmetrical. In this extended model, they used Mohr-coulomb yield surface for bricks and mortars, rather than the Drucker-Prager one, to avoid the apex problem. Furthermore, Zucchini and Lourenço applied their extended material model on a shear wall with good results.

Zucchini and Lourenço's micro-mechanical model incorporated many failure modes in a representative cell and most localized damages in components. It could still have a high accuracy even when the brick's and mortar's stiffness ratio is large. Therefore, this material model will be studied and further refined in the thesis due to its versatility.

The development of Zucchini and Lourenço's micro-mechanical model could be concluded as four phases:

(1) Micro-mechanical model in 2002: the model was derived from the actual deformations of a basic cell and included additional internal deformation modes [30].
(2) Coupled homogenization-damage model in 2004: the formulation and implementation of this model were proposed by coupling the micro-mechanical model described in (1) and an isotropic damage model for the bricks and mortars. This model was specific for the normal tension behaviour of masonry parallel to bed joints [20].
(3) Developed homogenization-damage model in 2007: the model was developed by implementing a plastic model into the coupled homogenization-damage model described in (2) by a novel homogenization tool. The plastic model incorporated Drucker-Prager yield criteria for every component, and the plastic deformations were computed according to the implicit Euler backward method with bi-parabolic hardening diagram, where the compressive splitting effects were taken into account. This model was derived for the compressive behaviour of masonry perpendicular to bed joints [32].
(4) Extended micro-mechanical model in 2009: in this model, the extended basic cell's normal and shear behaviours were simulated up to complete failure with suitably selected deformed mechanisms and coupled damage and plastic model described in (1)-(3) [21].

1.2. Research gaps

The work of Zucchini and Lourenço is characterized by an acceptable trade-off between accuracy and computational efforts. However, there are still some points that require attention and potential improvements. These gaps could be described from three aspects:

(1) Simplifications of deformed mechanisms to reduce the computational efforts.

Firstly, Zucchini and Lourenço considered nearly all possible deformations of bricks and mortars when they derived the final model for coupled normal and shear behaviours of masonry, such as the vertical shear deformation of the head joint and antisymmetrical deformations of the bed joint. These assumptions lead to accurate simulations of components' deformations but bring outstanding numerical efforts as well. Therefore, the simplifications could be done by making several reasonable assumptions on deformation mechanisms to save computational costs. For example, we could neglect head joints' shear deformations as the head joints generally are damaged in tension. As a result, the bed joint's deformations should be symmetrical, and we could consider fewer interfaces with a smaller unit cell.

(2) An improved assumption for the combination of shear and compressive behaviours.

Secondly, Zucchini and Lourenço used damage variables (damage factors) to define the components' stress status. In other words, they assumed that all of the given component's internal stresses would drop down to zero as the initial failure in the given component occurred. This initial failure was caused by cracks generated in each component, and different types of cracks are assumed in each component. For instance, the bed joint fails due to a single shear crack horizontally located at the middle of thickness direction if the bed joint is assumed to be damaged in “shear sliding failure mode”. In the extended micro-mechanical model proposed by Zucchini and Lourenço in 2009, the bed joint’s stiffness is assumed to be zero when the compressive energy in the y-direction[2] is consumed due to the micro-cracks. Alternatively, it can be said that the bed joint would be damaged in “crushing failure mode” in their assumption. However, their consideration ignores the case when the bed joint fails first due to shear. The horizontal shear crack may appear before the dissipation of compressive energy in the bed joint, leading to different failure mechanisms of the material model. Therefore, this consideration should be taken into account and studied.

(3) The hardening/softening and the splitting effects under compression should be included.

Thirdly, Zucchini and Lourenço abandoned the hardening phase and splitting effects in the plastic model when they introduced the final model. To simplify the formulations and avoid apex problems, they introduced the Mohr-coulomb yield surface in the σ-τ plane[3] with an exponential degraded law of cohesion instead of the Drucker-Prager yield surface in the 3-dimension plane with a bi-parabolic hardening law of the material compressive strength. However, the hardening phase and compressive splitting effects may be worth to be considered, so we should consider the relevant questions.

1.3. Research question

The gaps mentioned above lead to the research question:

Is it possible to define a homogenized constitutive model for masonry structures under in-plane loading that will consider shearing, tensile cracking, crushing and splitting failures based on a micro-mechanical approach with as little computational effort as possible?

To be more specific, this research question can be described in three sub-questions:

(1) How to simplify the deformed mechanisms of Zucchini and Lourenço’s model when coupled behaviours in all directions are considered.
(2) How to consider the phenomena that the horizontal shear crack potentially generated in bed joint may appear before the dissipation of compressive energy.
(3) How to implement the new plastic model, in which the hardening phase and compressive splitting effects are considered with less computational costs.

In conclusion, this thesis will propose an alternative constitutive model based on Zucchini and Lourenço’s research in [30, 20, 32, 21] with several new assumptions on deformed mechanisms, components’ failure modes and elastoplastic behaviours, which includes the five failure mechanisms introduced in section 1.1 (see figure 1-1). This alternative constitutive model should maintain as high accuracy as Zucchini and Lourenço’s micro-material model with fewer computational efforts.

1.4. Basic assumptions

In this thesis, masonry's normal and shear behaviours will be only considered in a 2D plane to simplify the problems. Then, we can study masonry’s behaviours from four aspects to answer the research questions:

(1) The representative homogenized cell is set by considering the symmetry in the geometrical pattern of masonry and is used to study all behaviours.

The representative plane element is quite significant to derive the homogenized constitutive laws for the micro-mechanical model since the compatibility and equilibrium equations are formulated based on the

[2] The y-direction is specific to the direction perpendicular to the bed joint in xy-plane.

[3] σ is referred as the normal stress tensor, while τ is referred as the shear stress tensor.

deformed mechanism of the representative cell. In this thesis, the deformations of the bed joint are assumed to be the same everywhere, and the head joint's shear stress is neglected. Therefore, a quarter basic cell is used to study masonry's pure normal or shear behaviour and the coupled behaviours rather than the extended cell applied by Zucchini and Lourenço in 2009, see figure 1-2.

In this way, several compatibility equations are not necessary to be derived at interfaces between bricks and mortars anymore as there are fewer interfaces in the quarter basic cell than the extended one.

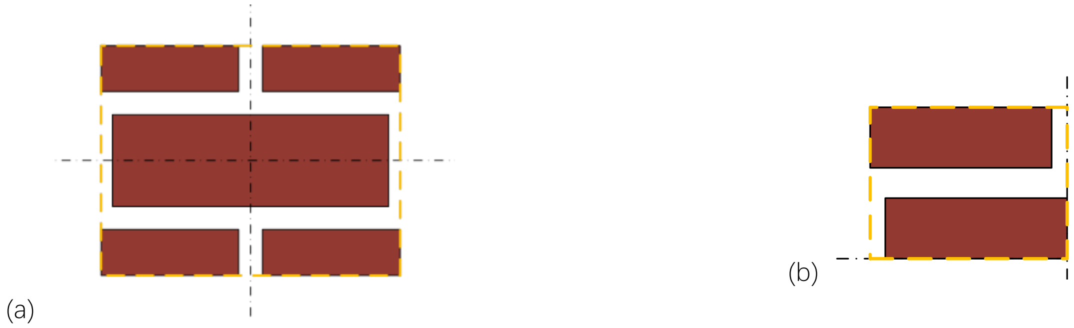


Figure 1-2 The representative cell in the XY plane: (a) extended basic cell defined in [21]; (b) quarter basic cell

Based on this assumption, we could first study the masonry's shear behaviour, normal behaviour in the x-direction and y-direction separately and then suitably incorporate them into one model by distinguishing different boundary conditions on the quarter cell. As a result, the deformed mechanisms of the final model could be simplified.

(2) Deformations of the cross joint are assumed to be different when different behaviours of masonry are studied.

The shear deformation of the cross joint is assumed to be the same as that of the bed joint when the masonry's shear behaviour is studied. Furthermore, the cross joint's horizontal deformation is assumed to be the tension when the masonry's normal tension behaviour in the x-direction is studied, while the compressive one when the normal (tension and compression) behaviour in the y-direction is studied. Meanwhile, the cross joint's vertical deformation is assumed to be the same as the bed joint's vertical deformation when the normal tension behaviour in the x-direction is studied, while not equal to the bed joint's vertical deformation when the normal (tension and compression) behaviour in the y-direction is studied.

(3) The bed joint is assumed to be damaged in shear before its compressive energy is dissipated and will move together with the brick units as a whole after the shear damage occurred.

According to this assumption, the normal and shear stresses of the bed joint should be equal to zero after a horizontal shear crack occurs. If there is an external shear loading, the force will then transform to the shear sliding stress at the interface of this shear crack, producing additional shear deformation. This shear deformation should be restrained by the head joint (vertical joint). As a result, the sliding stress should switch from the static friction stress to the dynamic one once the cohesion of the head joint is consumed. Therefore, the homogenized cell should fail in shear as the Mohr-coulomb yield criterion is met.

(4) The Drucker-Prager yield criteria with bi-parabolic hardening diagram are still implemented in the plastic model when the final model is proposed. However, the explicit Euler-forward algorithm is applied to save the computational costs.

The explicit Euler-forward approach has a cheaper computational cost than the implicit Euler-backward one as a particular equation could compute the plastic multiplier straightforward based on Prager's consistency equation in the explicit one. However, the robustness and accuracy of the explicit approach would be reduced if the load step is large, as Prager's consistency equation is derived based on small deformation hypothesis. Therefore, we could use this cheaper method to implement Drucker-Prager yield criteria and a hardening diagram and set small load steps. In this way, the compressive splitting effects could still be included with fewer efforts in the final model, which has complex deformations of components.

1.5. Goals & objectives

Particularly, the thesis has the following objectives:

(1) To define the failure modes necessary to be included.
(2) To derive the constitutive equations of the homogenized cell.
(3) To validate the model in MATLAB and Diana FEA.

The components' deformations and failure modes correlated to the homogenized cell' failure modes should be defined first.

The components' failure modes could be summarized as:

(i) Tensile failure modes: the given component fails once a vertical (or horizontal) tensile crack appears. In other words, the stresses of the given component will drop to zero once their tension stress exceeds the material strength.
(ii) Shear failure modes: according to the assumptions described in section 1.4, the shear failure mode will only occur in the bed joint. The bed joint will fail once the shear stress exceeds the material strength (computed based on the friction criterion), and the horizontal shear crack is then generated.
(iii) Equivalent splitting failure modes: the given component will fail once its total tension stress reaches the material tensile strength. This total tension stress should incorporate the splitting tension stress produced by the component's compressive stress.
(iv) Crushing failure modes: the given component fails when its compressive energy is consumed. This failure mode is different from but associated with "equivalent splitting failure mode". The components' stiffness is changed when loaded by axial compressive loading as the micro-cracks are generated, producing lateral splitting tensile stress. If the given component's splitting tensile stress reached the material strength, then the splitting failure will occur. If the given component's compressive energy is consumed before tensile stress reached the material strength, then the crushing failure will occur. Moreover, only a single vertical crack is generated in the given component if the component fails in splitting tension, while there will be lots of micro-cracks in the component if the component fails in crushing.

The macro constitutive laws should be derived based on the damage and plastic models by homogenization procedures and a multi-scale approach. The damage model should be proposed according to the chosen deformed mechanisms and components' failure modes, and the plastic model should incorporate the Drucker-Prager yield surface and a bi-parabolic hardening diagram.

Finally, the new constitutive models should be implemented in MATLAB to validate if the assumed failure modes of components and the homogenized cell are introduced successfully by the analytical solutions. Furthermore, these models should also be implemented in Diana FEA by user-supplied subroutines to assess if they could be numerically introduced.

1.6. Methodology

The following steps can be arranged in this thesis to achieve the goals and objectives mentioned in section 1.5:

(1) The representative 2D plane is extracted as the homogenized unit cell based on the symmetrical properties of 1/2 running bond masonry structures;

(2) The deformed cells of the unit cell under shear, horizontal tensile, vertical compressive loading are drawn based on the assumptions on the unit cells' deformed mechanisms in models 1 to 3, respectively;

(3) In models 1 to 3, the external strains and stresses are suitably selected and set to be homogenized (macro) strains and stresses. Meanwhile, the internal stresses are chosen, which compute the brick units and mortar joints' damage state variables by exponential laws according to the components' assumed failure mechanisms.

(4) In models 1 to 3, the relations of the homogenized strains and the internal stresses of the three deformed cells are derived based on several equations. These equations are: the micro-constitutive laws (the constitutive laws of bricks and mortars), the kinematic relations of the unit cells and the equilibrium equations of the systems;

(5) The damage models for three deformed cells in models 1 to 3 are derived by implementing the damage state variables into the compatibility equations of the systems derived in step (4). The components' undamaged internal stresses are then obtained;

(6) For the deformed cell under vertical compressive loading in model 3, three plastic models for the brick unit, head joint and bed joint are proposed by adopting the Drucker-Prager yield criterion and bi-parabolic hardening laws. In these plastic models, the components' plastic strain tensors are computed by the plastic multipliers and the directions of the potential energy at the critical stress points by applying an explicit return-mapping algorithm, the Euler-forwards method;

(7) The homogenized stresses of models 1 to 3 are computed by the damaged internal stresses based on the assumed boundary conditions of the unit cells. The damaged internal stresses are obtained based on the components' damage state variables and their undamaged internal stresses. The undamaged internal stresses are computed by combining the damage and plastic models for model 3;

(8) The homogenized stresses for coupled behaviours are proposed by coupling three deformed cells in models 1 to 3 in model 4;

(9) The constitutive models 1 to 4 are Implemented in MATLAB to find the analytical solutions. The deformed cells' macro stress-strain curves, the components' damaged factors and other relative curves in models 1 to 4 are recorded. Models 1 to 2 are Implemented in DIANA FEA user-supplied subroutines by Fortran 77 to assess if the models can be introduced numerically;

(10) The experimental data for the masonry under compressive loading is adopted in model 4. The comparison between the experimental and analytical results is used to assess the model's vertical compressive behaviour.

1.7. Structures of thesis

The structures of the thesis can be found in table 1-1 according to the methodology introduced in section 1.6.

	Titles	Overviews of the concepts
Chapter 1	Introduction	The research background is introduced first, and then the research gaps are pointed out for further study. After that, the goals of the thesis are stated. Finally, the methodology used to achieve the goal is introduced
Chapter 2	Literature review	The previous research is introduced and studied.
Chapter 3	Theory and assumptions	The theory and assumptions used to introduce the damage model and plastic model are stated in this chapter, such as selecting the internal stresses, introducing a hardening diagram for the plastic model.
Chapter 4	Model 1: shear behaviour	The material model is derived when only the shear behaviour is taken into account. The works can be summarized as deriving the kinematic relations, micro constitutive relations and equilibrium equations of the system and implementing damage variables into the components
Chapter 5	Model 2: horizontal tension behaviour	The material model is derived when only the horizontal tension behaviour is taken into account. The same works are done.
Chapter 6	Model 3: vertical compression behaviour	The material model is derived when only the vertical compressive behaviour is taken into account. Apart from the same works done

		in chapters 4 and 5, the components' plastic models with the Drucker-Prager yield criterion are derived by applying the explicit Euler-forward algorithm.
Chapter 7	Model 4: coupled behaviour	The models derived in chapters 4 to 6 are coupled as one model in this chapter by incorporating the transverse strains into the axial ones and introducing the Mohr-Coulomb friction criterion for shear behaviour.
Chapter 8	Implementations and comparisons	The models derived in chapters 4 to 7 are implemented in MATLAB to judge if the assumed failure mode and algorithm are introduced successfully. The models proposed in chapters 4 and 5 are introduced in Diana FEA by a user-supplied subroutine to assess if the material models can be implemented in a nonlinear finite element program. At the end of this chapter, a comparison of the analytical results solved by MATLAB code and experimental results obtained from the tests is stated to provide the material model that can be used to simulate the behaviour of masonry with reasonable accuracy.
Chapter 9	Conclusions and recommendations	The differences between the works done in this thesis and the previous research are stated. The limitations of the material model proposed in this thesis are explained. The future works that may be worthy of being studied are stated.

2. Literature review

According to the reviews of analytical methods of masonry structures from Dimitris Theodossopoulos and Braj Sinha in 2013 [33], a better understanding of failure patterns to estimate the strengths of a masonry structure is significant to establish the constitutive model of the composite structure numerically. Therefore, failure mechanisms of masonry structures are firstly introduced in this chapter.

Based on the given failure patterns, several techniques on numerically introducing the element cell, such as simplified micro-modelling approach as Paulo B. Lourenço and Jan G. Rots shown in 2004 [10] together with homogenization techniques introduced by Paulo B. Lourenço in 1996 [8], should be suitably selected to show those mechanisms. Therefore, numerical approaches on presenting failure modes are then introduced briefly in this chapter.

After that, Zucchini's damage model proposed in 2004 [20] is briefly introduced, and the formulations of introducing elastoplastic behaviours of the bricks and mortars, proposed by Prof. Borst and Prof. Sluys in [34], is described in this section.

2.1. Failure mechanisms of masonry

An accurate material model should include all basic types of failure mechanisms that characterize masonry, which are the following: (a) tensile cracking of the joints, (b) sliding along the bed joints, (c) cracking of the units in tension, (d) diagonal tensile cracking in the brick units and (e) masonry crushing, shown in figure 2-1 [8].

Failure patterns (a) and (b) were firstly identified by Page in 1978 [6], P.B. Lourenço and J. Rots in 1994 [7], indicating shear and tensile failure of horizontal joints. In 1996, P.B. Lourenço presented a novel compressive cap as (d) or (e) shown by limiting combinations of shear and compressive stresses [8].

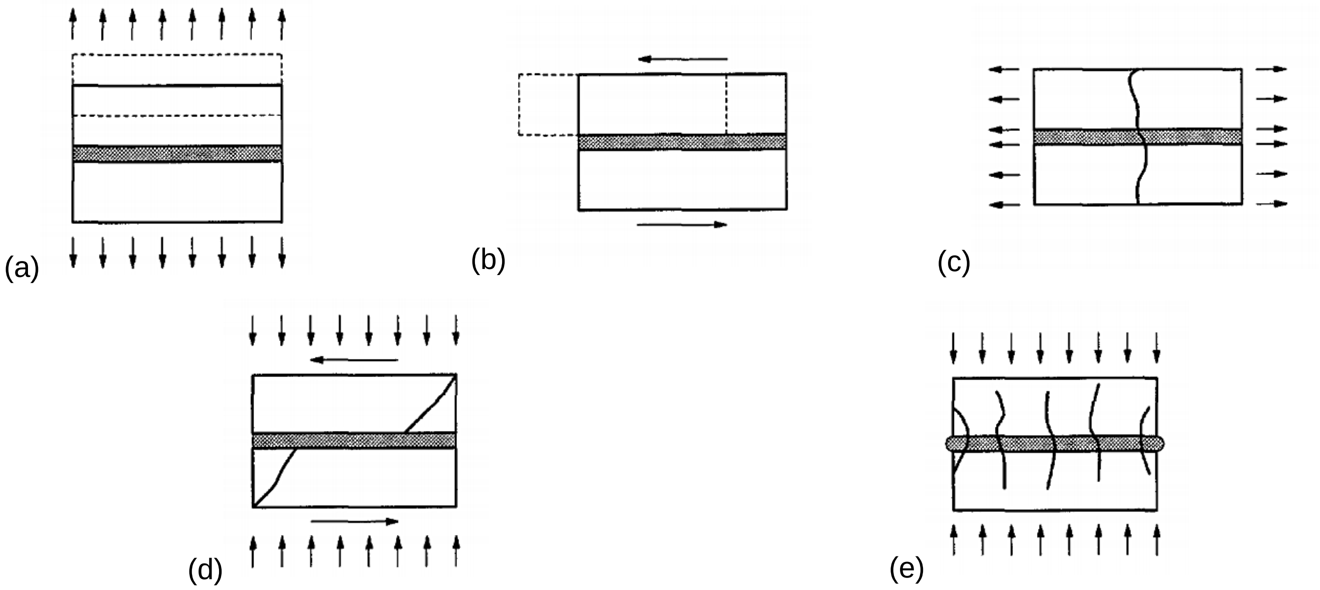


Figure 2-1 Masonry failure mechanisms: (a) joint tensile cracking; (b) joint sliding; (c) unit direct tensile cracking; (d) unit diagonal tensile cracking; (e) masonry crushing. [8]

It can be seen from phenomena presented in figure 2-1 that (a) (b) are joint failure mechanisms, (c) is the unit one, (d) and (e) are failure patterns that include both brick units and mortar joints' failure. The question remains of how to consider all of these phenomena in a homogenized model [8].

Shear failure pattern (b) occurs when masonry structures are loaded by pure shear loading, while this failure pattern would change to failure pattern (d) when structures are loaded both in precompression and shear. When masonry structures are under horizontal loading, they fail as the unit tensile cracking pattern (c). Masonry structures fail once horizontal tensile cracks occur in the horizontal joint if they are loaded by pure vertically tensile stress, see figure 2-1 (a), while they fail as masonry crushing pattern as (e) shown if they are loaded by compressive loading.

2.1.1. Pure shear behaviour

According to figure 2-2 (b), shear damage occurs at the interface of the brick unit and mortar joint when masonry is loaded by pure shear loading. That means masonry would be damaged once the shear stress between the brick unit and the horizontal joint, commonly referred to as the bed joint, reached its material tensile strength.

The shear behaviour of masonry is relative to model II fracture energy G_f^{II}, see figure 2-2.

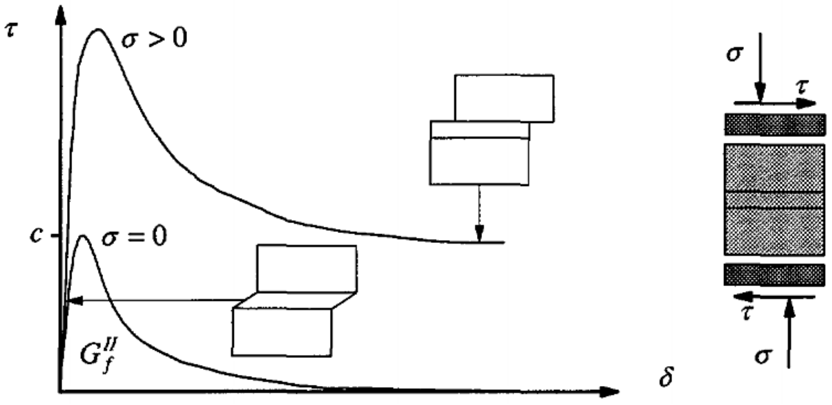


Figure 2-2 the masonry under shear and definition of II fracture energy: shear stress τ versus shear strain δ [8]

Note that from figure 2-1 (d) and 2-2, the representative element for stacked bond masonry loaded by vertical compressive and laterally shear loading could fail once the diagonal cracking occurs. The cracks in the brick units are driven by shear sliding stress at the horizontal interface of the brick and the mortar. Therefore, the vertical loading condition should be concluded when the shear behaviour of the unit cell is studied.

When the masonry is loaded simultaneously by shear and vertical precompression, the relation of the shear strength of masonry and the vertical precompression could be described as equation (2.1).

$$\tau_{max} = c + \sigma \cdot \tan(\phi) \tag{2.1}$$

Where τ_{max} is the shear strength of masonry, and cohesion c is typically the cohesion of the mortar joints. σ is the precompression loading in the vertical direction, and ϕ is the friction angle of the mortar joints.

2.1.2. Combined mechanisms

According to A. Zucchini's deformed cell assumption [30], different components have unequal deformations, leading to different internal stresses and strains of the brick units and mortar joints due to their different stiffness (see figure 2-3. The shear deformation occurs at the interface of the brick unit and the bed joint, see figure 2-3 (b). As a result, the masonry is damaged once vertical cracks occur in brick units, see figure 2-1 (c).

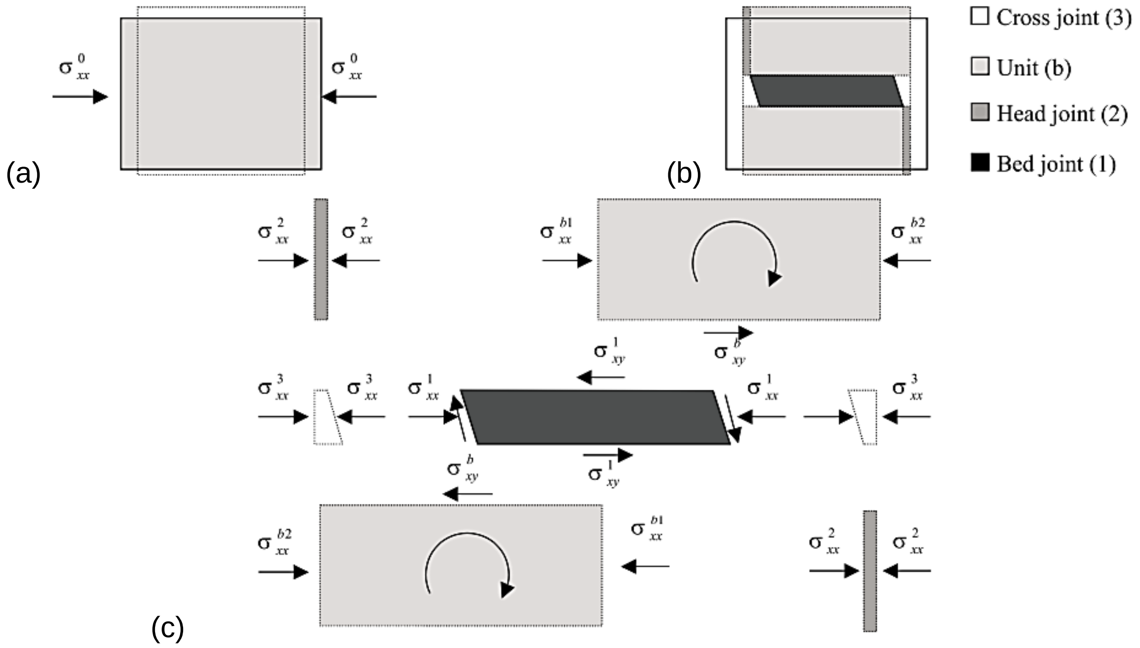


Figure 2-3 Normal stress loading parallel to x-axis: (a) equivalent homogenized cell; (b) assumed deformation behaviour; (c) assumed involved stress components. [30]

2.1.3. Masonry cell under vertical compression

Temporarily, the compressive failure in masonry is mainly governed by the interaction between brick and mortar. The different stiffness of the brick and mortar leads the mortar joints being more deformable than the brick units under uniaxial compression load with the assumptions of compatible strains between each component (see figure 2-4 [35].

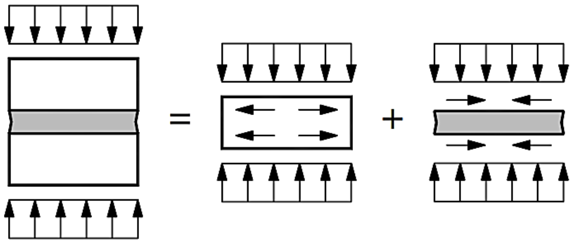


Figure 2-4 uniaxial behaviour of masonry: schematic plane representation of stresses in masonry component [35]

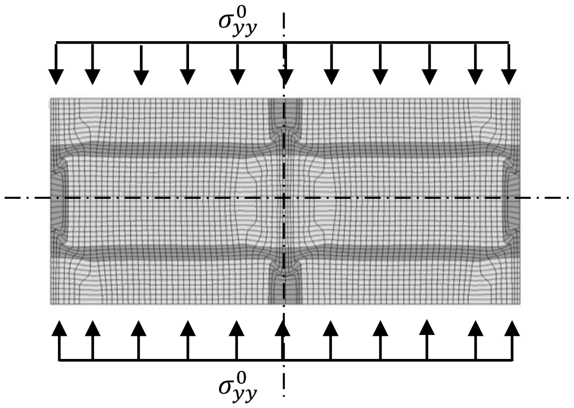


Figure 2-5 Deformation of basic cell from numerical results [35]

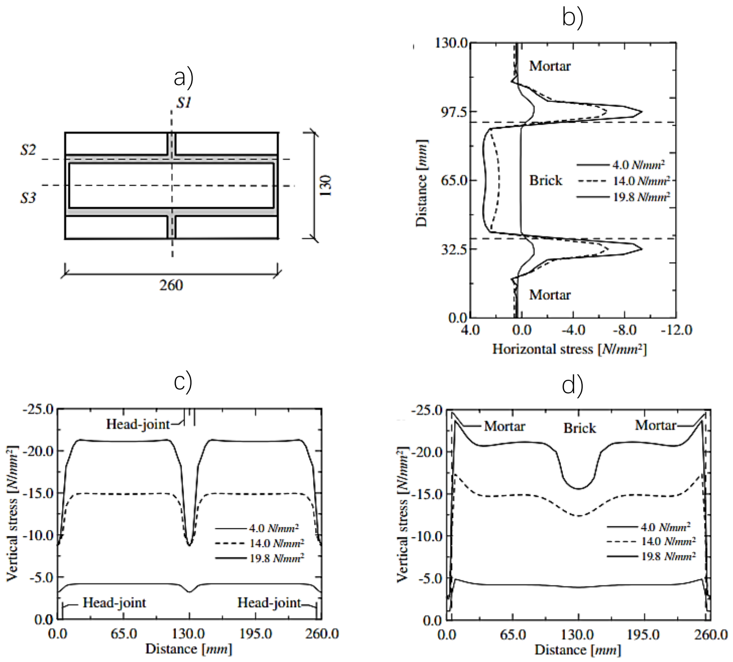


Figure 2-6 stress diagram at increasing load level for different components of cell: a) basic unit cell and definition of S1, S2 and S3; b) horizontal stress distribution at S1; c) vertical stress distribution S2; d) vertical stress distribution in S3 [35]

Numerical results from Lourenço's research in 2006 [35] displayed stress distribution in each section of the representative plane, shown in figure 2-6 (a), under vertical compressive loading, see figures 2-5 and 2-6 shown.

As shown in diagrams in figures 2-4 to 2-6, horizontal tension stresses of the head joints and bricks are distributed uniformly in the vertical direction, and horizontal compression stress linearly increases at the overlap of the bed and head joint.

This result corresponds to the uniaxial behaviour of masonry introduced in figure 1.3-1 and indicates shear stresses at the interface of brick and vertical mortar. The vertical stresses are distributed almost uniformly in the brick and the bed joints but linearly in the head joint, indicating shear stress at the brick and head joint interface.

According to figure 2-1 (d), the basic cell of the masonry would be damaged once the brick units are crushed caused by the uniformly distributed stress in the vertical direction of the brick units.

2.2. Numerical methods to simulate masonry structures

Masonry is a composite material, which consists of brick units and mortar joints. Modelling for this type of structure varies by different assumptions of components and brick-mortar interface.

As figure 2-7 shown, modelling strategies can be recognized by the following three when modelling with FEM three main modelling strategies [8]:

(1) Detailed micro-modelling: Units and mortar in the joints are represented by continuum elements, whereas discontinuous elements represent the unit-mortar interface.
(2) Simplified micro-modelling: brick units are represented by expanded continuum elements, whereas the mortar joints' behaviour and unit-mortar interface are lumped in discontinuous elements.
(3) Macro-modelling: Units, mortar and unit-mortar interfaces are smeared out in a homogeneous continuum.

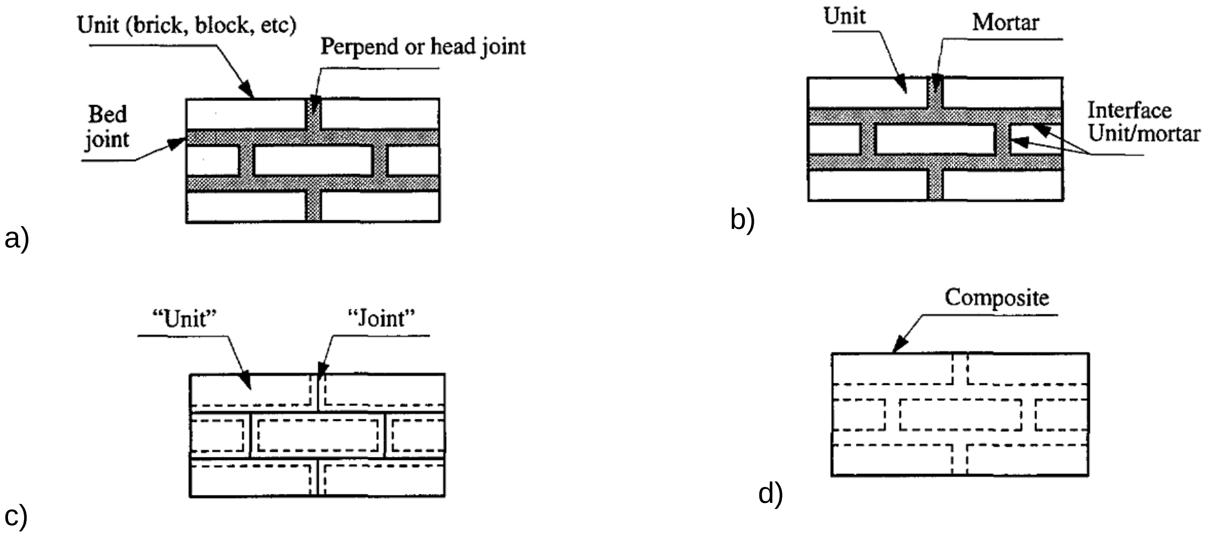


Figure 2-7 modelling strategies of masonry structures: a) masonry sample; b) detailed micro-modelling; c) simplified micro-modelling; d) Macro-modelling [8]

2.2.1. Simplified micro-modelling

The micro-modelling approach concentrated all damages in weak joints and, if necessary, in potential horizontal cracks in brick units vertically placed in the middle of each unit, as figure 2-8 shows [10]. The brick-mortar interfaces are included in the detailed micro-models, while these interfaces are neglected when the micro-models are simplified.

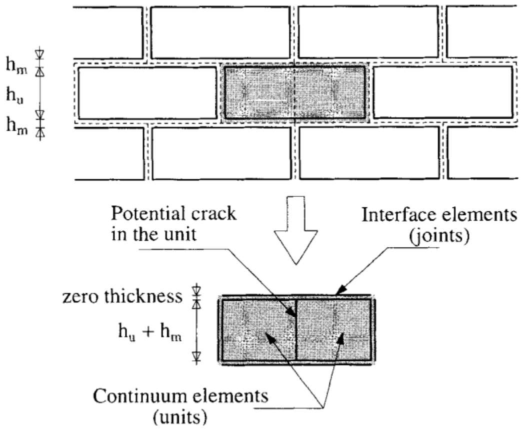

h_m
h_u
h_m
Potential crack in the unit
Interface elements (joints)
zero thickness
$h_u + h_m$
Continuum elements (units)

Figure 2-8 suggested modelling strategy: unit (u) and mortar joint (m) and potential cracks in units [10]

Brick units are typically modelled with continuum elements. Mortar joints and potential tensile cracks vertically placed in the middle of the brick unit are modelled by zero-thickness interface elements [10].

A zero-thickness interface element could allow discontinuities in the displacement field of the mortar joints, and relations could describe the potential tensile cracks and the behaviour of the interface elements in terms of traction $\boldsymbol{t}$ along joint thickness direction and $\boldsymbol{u}$ cross interface.

2.2.2. Macro-modelling

From section 2.2.1, it is noted that the behaviours of masonry structure could be numerically reproduced by applying the material properties and the actual geometry of the brick and the mortar. However, this approach became impractical in the case of many masonry structures consisting of many units. Therefore, masonry is usually homogeneous, although it is a composite material [8].

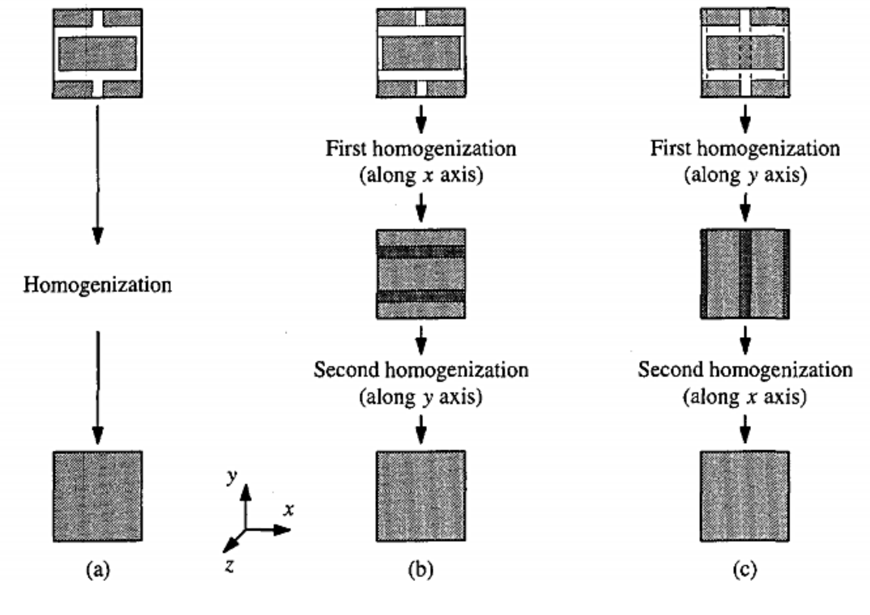

Homogenization
First homogenization (along x axis)
Second homogenization (along y axis)
First homogenization (along y axis)
Second homogenization (along x axis)
y
x
z
(a)
(b)
(c)

Figure 2-9 two-step homogenization procedure for masonry structure: (a) objective of homogenization; (b) homogenization xy; (c) homogenization yx. [8]

There are two approaches for macro-modelling: modelling based on the composite theory; modelling based on the compatibility and equilibrium equations of the unit cell. By applying homogenization techniques, composite behaviour is described in macro or average stresses and strains, derived from micro-constitutive law and geometrical properties.

Firstly, building up the basic cell by using an approximate approach based on two-step homogenization procedures under the assumptions of layered material, see figure 2-9.

Secondly, implement micro-constitutive laws, such as yield surface, constitutive law, and homogenized ones, by applying macro-parameters proposed in step 2.

2.3. Homogenization approaches

In the following sections, 2D plane stress elements with homogenization techniques are selected for numerical analysis. The failure mechanisms of shear, horizontally tensile and vertically compressive behaviour of masonry should be considered.

2.3.1. Existing models

Masonry is assumed to be a continuum deformable-body in continuum models. The existing procedures to derive homogenized constitutive laws of this type of model commonly could be proposed based on composite theory, see Lourenço's research in [8], or compatibility and equilibrium equations of the deformed cell, see A. Zucchini et al. [30].

Macro-model based on the composite theory

Lourenço proposed a homogenized approach in 1996 based on the theory of layered materials. [8] He introduced a novel formulation based on the approach proposed by Salamon in 1968 [36], which originated in the field of rock mechanics to handle inelastic material behaviour [8].

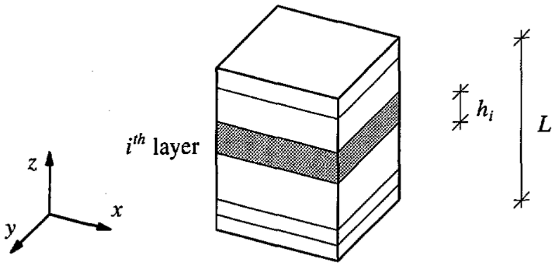


Figure 2-10 Representative volume regime of the periodic system of parallel layers [8]

Figure 2-10 displayed a layered material built from a periodic system of parallel layers, and each layer is assumed to be an isotropic elastic material. The system of layers is assumed to still be continuous after deformation, and it is assumed that there is no relative displacement at the interface between each layer.

This representative volume prism is then assumed to be subjected to homogeneously distributed stresses and strains. That means the volume of the composite material should be small enough to make the representative volume regime negligible when considering variants of stresses and strains across the regime's thickness direction [8].

The objective was to obtain macro constitutive relation between homogenized stresses $\boldsymbol{\sigma}$ and strains $\boldsymbol{\varepsilon}$, see equation (2.2)

$$\boldsymbol{\sigma} = \boldsymbol{D}^h \boldsymbol{\varepsilon} \tag{2.2}$$

$$\boldsymbol{\sigma} = \left\{ \ \sigma_x \ \ \sigma_y \ \ \sigma_z \ \ \tau_{xy} \ \ \tau_{yz} \ \ \tau_{xz} \ \right\}^T \tag{2.3}$$

$$\boldsymbol{\varepsilon} = \left\{ \ \varepsilon_x \ \ \varepsilon_y \ \ \varepsilon_z \ \ \gamma_{xy} \ \ \gamma_{yz} \ \ \gamma_{xz} \ \right\}^T \tag{2.4}$$

Where $\boldsymbol{D}^h$ is homogenized stiffness matrix which was obtained from micro-constitutive relations, given by Lourenço in 1996 [8] as:

$$\boldsymbol{D}^h = \left[\sum_i p_i \left(\boldsymbol{P}_t - \boldsymbol{D}_i \boldsymbol{P}_e \right)^{-1} \right]^{-1} \sum_i p_i \left(\boldsymbol{P}_t - \boldsymbol{D}_i \boldsymbol{P}_e \right)^{-1} \boldsymbol{D}_i \tag{2.5}$$

Where $\boldsymbol{P}_t$ and $\boldsymbol{P}_e$ are the projection matrices into stress and strain space, respectively, $\boldsymbol{D}_i$ is stiffness matrix of i^{th} layer. Normalized thickness p_i could be defined by the thickness (h_i) of the (i^{th}) layer, and the length L of representative volume in z direction shown in figure 2-9. Once the averaged stresses and strains were known, stresses and strains in i^{th} could be calculated as:

$$\boldsymbol{\sigma_i} = \boldsymbol{T}_{ti} \, \boldsymbol{\sigma} \quad with \, \boldsymbol{T}_{ti} = \boldsymbol{I} + \boldsymbol{P}_t \left(\boldsymbol{P}_t - \boldsymbol{D}_i \boldsymbol{P}_e \right)^{-1} \left(\boldsymbol{D}_i \left(\boldsymbol{D}^h \right)^{-1} - \boldsymbol{I} \right) \tag{2.7}$$

$$\boldsymbol{\sigma_i} = \boldsymbol{T}_{ei} \, \boldsymbol{\sigma} \quad with \, \boldsymbol{T}_{ei} = \boldsymbol{I} + \boldsymbol{P}_e \left(\boldsymbol{P}_t - \boldsymbol{D}_i \boldsymbol{P}_e \right)^{-1} \left(\boldsymbol{D}_i - \boldsymbol{D}^h \right) \tag{2.8}$$

This result could be extended to the inelastic formulation by a certain algorithm, see Lourenço's research in 1996 [8].

Micro-mechanical model

According to A. Zucchini's deformed cell assumption, see chapter 2 section 2.1.2, different components have unequal deformations, leading to different internal stresses and strains of the brick and mortar joints due to their different stiffness.

As a result, the individual (internal) stresses and strains of the units and the joints were derived from the average (external) stress and strain of the composite cell [20].

In reverse, we can use relationships between internal strains and stresses of the inner components to derive a stress-strain curve of the whole homogenized cell with certain equilibrium equations of the system.

Relations of the internal and external stresses could be built up based on boundary conditions from the assumptions of loading conditions, and the other relative equations for solving strains and stresses of the bricks and the mortars could be derived from the kinematic relations, the constitutive law of each component as well as the equilibrium equations at the interface between components.

2.3.2. Gaps in existing models

The homogenized material model proposed by P. B. Lourenço 1996 based on composite theory [8] could have the strength cap in shear, horizontal and vertical direction, but the accuracy of this model will be reduced when the stiffness ratio between the mortars and the joints is large. The micro-mechanical model proposed by A. Zucchini and P. B. Lourenço could solve this problem [20, 21, 30, 32], but the coupled behaviour of the material model are incorporated into the models with a complex algorithm. Furthermore, the shear sliding cracking of the bed joint was not considered in their material model.

Therefore, a homogenized material including all failure modes (see figure 2-1) with higher accuracy but fewer computational efforts is worthy of being introduced and studied in the thesis.

2.4. Micro-mechanical material models

The micro-mechanical material models can be proposed by coupling the damage and plastic models. The damage models, considering failure mechanisms of brick units and mortar joints separately, proposed in [20] [2] are referred to in this study. To introduce elastoplastic behaviours of the components, the theories introduced by R.de Borst and Sluys are used here [34].

2.4.1. Damage model

In 2004, Zucchini et al. proposed a novel damage model by applying his detailed micro-mechanical model introduced in section 2.1.2 and 2.3.1 under the basic cell's pure external horizontal loading condition.

24 equations were formulated to find 24 unknown variables, including 23 internal stresses and strains of components and 1 external stress of homogenized cell, which were firstly derived from deformed mechanics. Those equations consist of the micro-constitutive relations of the components, the kinematic relations of the components together with the homogenized cell based on deformed cell assumptions, the equilibrium equations of the system at boundaries and interfaces between components.

And then, 4 damage variables were set as representative coefficients to show the damage status of the components, including brick, head joint, bed joint and cross joint shown in figure 2-3. Those variables could be computed according to the exponential laws of each constituent’s internal stresses and the damage coefficients. And then, the damage factors were implemented into the 24 equations by a particular algorithm introduced by Zucchini et al. [20].

The final 24 equations were called equilibrium “damage” equations. 24 unknown variables, including the external stress of the homogenized cell, could be solved by knowing the external strain caused by the external load and the geometry of the basic cell. As a result, macro constitutive relation between external strain and stress can be obtained.

2.4.2. Elastoplastic theory

Materials, especially brittle ones, are generally damaged inelastic under compressive loading as fissures grow at the micro-level due to the dissipative process. Therefore, in-elastic strains resulting from this process should be considered for masonry under compression load. One of the most developed theories using for describing material nonlinearity is the mathematical theories of plasticity. In a sense, its development goes back to Coulomb, who postulated the dependency of sliding resistance on a plane between two bodies to be a function of cohesion and the frictional properties [34]. Similarly, as a non-smooth Tresca yield surface has been approximated to Von Mises’ yield contour, Drucker and Prager introduced their yield function, called Drucker-Prager criteria, as an approximation Mohr-Coulomb yield contour by a circular cone.

Drucker Prager yield criteria

Figure 2-11 shows Mohr-Coulomb yield surface, with dashed lines of angle shape, and Drucker-Prager yield surface in π-plane, replacing Mohr-Coulomb yield surface to a circle. Drucker-Prager’s yield contour maintains the linear dependency on hydrostatic stress level as the circle passes through the corner, usually the three outermost ones considered safety design, of the dashed line. The Drucker-Prager yield function in the 3D principal coordinate system could be defined as [34]:

$$f(\sigma) = \sqrt{\frac{1}{2}[(\sigma_1-\sigma_2)^2+(\sigma_2-\sigma_3)^2+(\sigma_3-\sigma_1)^2} + \frac{1}{3}\alpha(\sigma_1+\sigma_2+\sigma_3) - k \qquad (2.9)$$

Where $\sigma_i, i = 1,2,3$ are principal stresses, α and k are material constants related to physical parameters. The material cohesion c and friction angle ϕ are defined as:

$$\alpha = \frac{6\ \sin\phi}{3-\sin\phi},\ \ k = \frac{6\ c\ \sin\phi}{3-\sin\phi} \qquad (2.10a)$$

When the yield function meets the condition of $f(\sigma) < 0$ in all stress states, elastic deformations occur. Otherwise, plastic deformations occur. In this research, 2D models with plane stress elements will be considered. Therefore, eq. (2.9) and $(2.10a)$ should be rewritten, which will be presented in detail in chapter 6 as a similar shape of Drucker-Prager yield contour in 2D principal space shown in figure 2-12.

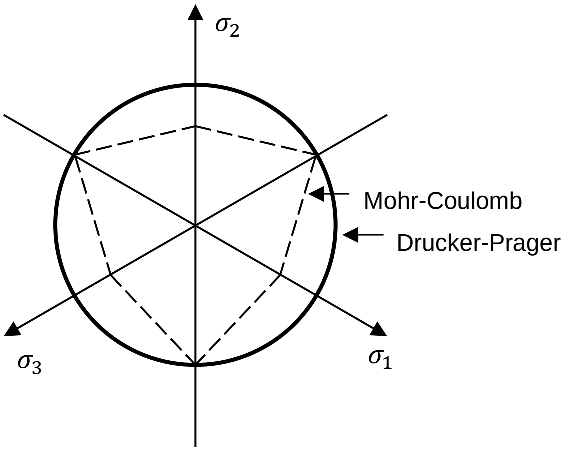


Figure 2-11 Representation of the Mohr-Coulomb and Drucker-Prager yield contour in π-plane [34]

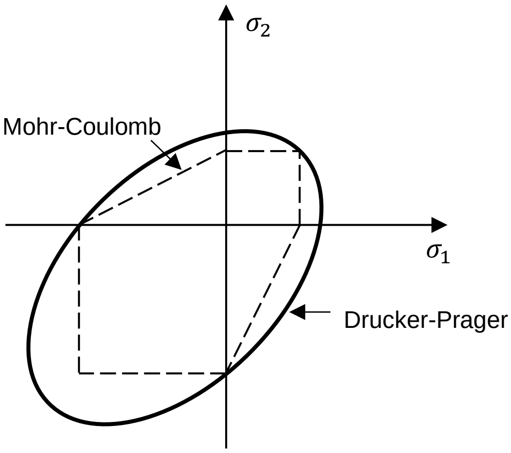


Figure 2-12 Representation of the Mohr-Coulomb and Drucker-Prager yield contour for plane stress condition [34]

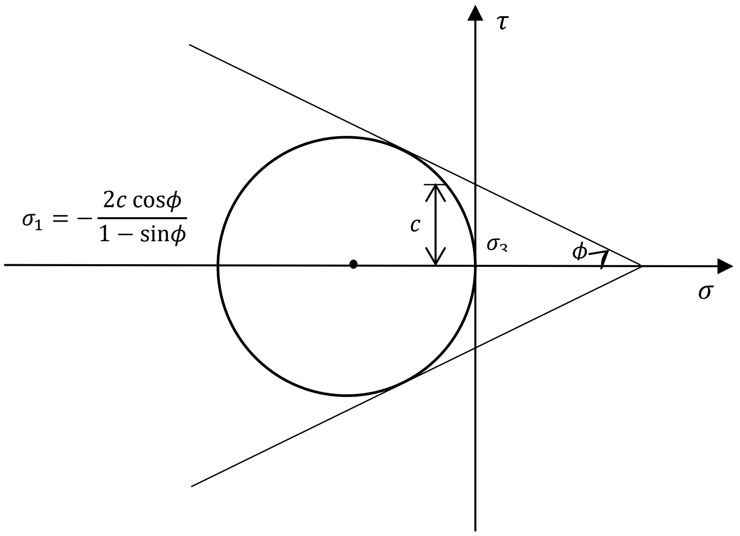


Figure 2-13 Mohr's stress circle for uniaxial compression and envelopes that bound all possible stress states for the Mohr-Coulomb yield criteria [34]

This model was also pretty suitable for describing sand's strength characteristics, drained clays, rocks and concrete. [34] From equations (2.9) and $(2.10a)$, we could obtain that the yield function of the Drucker-Prager model could be described by material parameters including friction angle ϕ and cohesion c. That means, we are able to derive a relationship between the cohesion c and the principal stresses $\sigma_1, \sigma_2, \sigma_3$ once the value of friction angle is known. As can be seen from figure 2-13, let us assume $\sigma_1 = 0$ for 2D condition and $\sigma_3 = 0$ to make $\sigma_2 = -f_c$, where f_c is compressive strength. Then we can use compressive strength (characteristic of material) to describe cohesion c as:

$$c = \frac{1 - \sin\phi}{2\cos\phi} f_c \tag{2.10b}$$

Value of compressive strength f_c could be correlated to the hardening/softening parameter if we consider the hardening/softening behaviour of the materials, cohesion c could be relative to the hardening/softening parameter. As a result, the yield surface described in figure 2-12 eq. (2.9) would shrink or expand as the softening or hardening process occurs.

Flow rule

Yield function is introduced as the contour that defines a spherical surface in 3-dimensional stress space, distinguishing permissible from non-permissible stress states. If the stresses are inside the surface ($f(\sigma) < 0$) defined by yield function, then the deformations are pure elastic, while the plastic deformations can occur if and only if the stress points are on the surface [34]. Stress points stated outside the yield surface are not permitted here.

To obtain plastic deformation, the stress point must be on the yield contour and remain there for a "short period". If the stress point touched the yield surface and it immediately pointed inward or outward the contour, plastic deformation may not happen [34]. In other words, plastic deformations occur when the following two conditions are met:

$$f(\sigma) = 0 \tag{2.11}$$

$$f\dot{(}\sigma) = 0 \tag{2.12}$$

Equation (2.12) is usually called Prager's consistency equation and ensures that the yield condition must be fulfilled for at least a small-time increment so that plastic flow can occur [34].

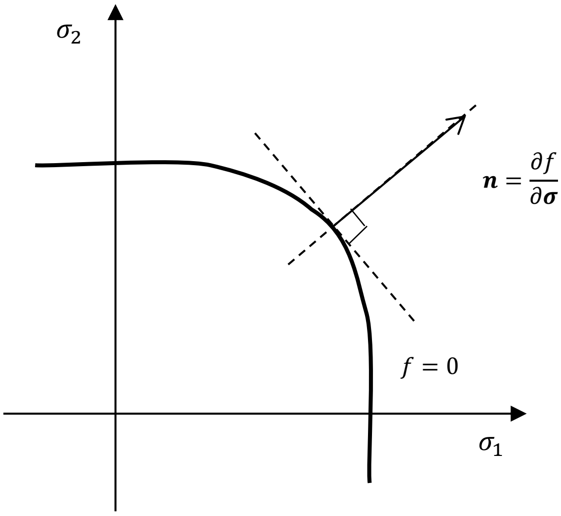

σ_2
$\boldsymbol{n} = \frac{\partial f}{\partial \boldsymbol{\sigma}}$
$f = 0$
σ_1

Figure 2-14 congeniality of the gradient vector $\boldsymbol{n}$ to the yield surface [34]

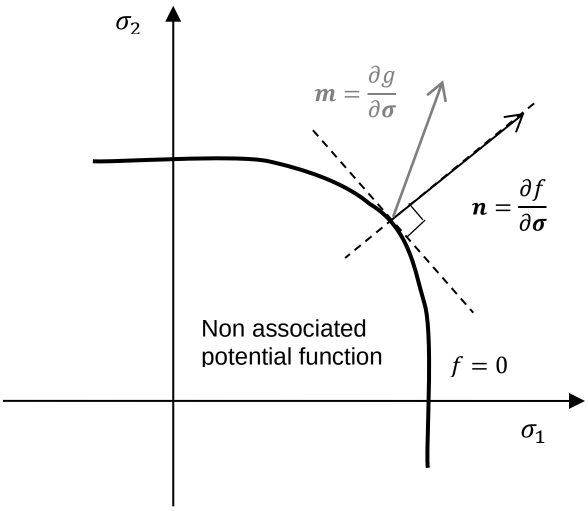

σ_2
$\boldsymbol{m} = \frac{\partial g}{\partial \boldsymbol{\sigma}}$
$\boldsymbol{n} = \frac{\partial f}{\partial \boldsymbol{\sigma}}$
Non associated potential function
$f = 0$
σ_1

Figure 2-15 Orthogonality of the gradient vector $\boldsymbol{m}$ to the non-associated potential function g

While the stress point is inside the yield contour, the elastic deformation is dominating. Where the constitutive relation should be [34]:

$$\boldsymbol{\sigma} = \boldsymbol{D}_e \boldsymbol{\varepsilon} \tag{2.13}$$

Where $\boldsymbol{D_e}$ is the (continuum) elastic stiffness matrix, setting the relationship of stress tensor $\boldsymbol{\sigma}$ and strain tensor $\boldsymbol{\varepsilon}$. This relationship can only be established when the elastic deformation occurred. Therefore, we are able to rewrite Eq. (2.13) and express the stress tensor $\boldsymbol{\sigma}$ by the elastic strain vector $\boldsymbol{\varepsilon_e}$ as:

$$\boldsymbol{\sigma} = \boldsymbol{D}_e \boldsymbol{\varepsilon}_e \tag{2.14}$$

When the plastic strains occur, the remaining part of the strains should be plastic and be obtained from abstracting the elastic contribution $\boldsymbol{\varepsilon_e}$ from the total strain $\boldsymbol{\varepsilon}$: [34]

$$\boldsymbol{\varepsilon}_p = \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_e \tag{2.15}$$

Combining equations (2.6) and (2.7), the expression of stress tensor could be obtained as:

$$\boldsymbol{\sigma} = \boldsymbol{D}_e\left(\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_p\right) \tag{2.16}$$

Since the yield function is assumed to be a sole function of stress tensor as $f = f(\boldsymbol{\sigma})$, the consistency condition could be elaborated as:

$$\frac{\partial f}{\partial t} = \frac{\partial f}{\partial \boldsymbol{\sigma}} \cdot \frac{\partial \boldsymbol{\sigma}}{\partial t} = \boldsymbol{n}^T \dot{\boldsymbol{\sigma}} = 0 \tag{2.17}$$

Where $\boldsymbol{n}$ is the gradient vector of yield function shown in figure 2-14 and $\dot{\boldsymbol{\sigma}}$ is a short time stress increment. Substituting eq. (2.16) for (2.17) results in:

$$\boldsymbol{n}^T \boldsymbol{D}_e\left(\dot{\boldsymbol{\varepsilon}} - \dot{\boldsymbol{\varepsilon}_p}\right) = 0 \tag{2.18}$$

Plastic strain increment for (small) time step is able to be relative to the plastic multiplier $\dot{\lambda}$ and a vector $\boldsymbol{m}$ as:

$$\dot{\boldsymbol{\varepsilon}_p} = \dot{\lambda}\, \boldsymbol{m} \tag{2.19}$$

With $\dot{\lambda}$ determining the magnitude of plastic flow and vector $\boldsymbol{m}$ describing the direction of flow [34].

Combining equations (2.18) and (2.19), the value of the plastic multiplier $\dot{\lambda}$ could be calculated as:

$$\dot{\lambda} = \frac{\boldsymbol{n}^T \boldsymbol{D}_e \dot{\boldsymbol{\varepsilon}}}{\boldsymbol{n}^T \boldsymbol{D}_e \boldsymbol{m}} \tag{2.20}$$

Finally, the function of the stress vector rate $\dot{\boldsymbol{\sigma}}$ can be proposed as:

$$\dot{\boldsymbol{\sigma}} = \boldsymbol{D}_e\left(\dot{\boldsymbol{\varepsilon}} - \frac{\boldsymbol{n}^T \boldsymbol{D}_e \dot{\boldsymbol{\varepsilon}}}{\boldsymbol{n}^T \boldsymbol{D}_e \boldsymbol{m}} \boldsymbol{m}\right) \tag{2.21}$$

Typically, vector $\boldsymbol{m}$ was the gradient of the plastic potential function g obtained from experimental data to indicate plastic volume change. For masonry material, which is sensitive to dilatant behaviour, the non-associated flow rule generally has a different direction compared with the direction of the yield surface, gives a better predictor of volume change by a proper value of dilatancy angle ψ rather than friction angle ϕ. Its function could be described as:

$$g = \sqrt{\frac{1}{2}[(\sigma_1 - \sigma_2)^2 + (\sigma_2 - \sigma_3)^2 + (\sigma_3 - \sigma_1)^2} + \frac{1}{3}\alpha_\psi(\sigma_1 + \sigma_2 + \sigma_3) - k_\psi \tag{2.22}$$

With α_ψ and k_ψ are material constants for Drucker-Prager like potential function.

$$\alpha_\psi = \frac{6\ \sin\psi}{3 - \sin\psi}, \quad k_\psi = \frac{6\ c\ \sin\psi}{3 - \sin\psi} \tag{2.23}$$

Let us suppose that the external loading would increase linearly by time increment, then the stress vector rate $\dot{\boldsymbol{\sigma}}$ should be stress increment at each load step. Therefore, function (2.21) can be used to find out stress tensor considering plastic deformation without hardening or softening behaviour.

Hardening and softening behaviour

So far, it was assumed in the ideal plastic model introduced above that it did not take the hardening or softening behaviour into account, which means the value of the yield function proposed above only depends on the stress tensor. This assumption, however, is less reliable when we consider the compressive loading since cohesion c of brick and mortar will be computed as a constant along with the loading history in this way. Fissures at the micro-level generated in components influence the stress distribution of each component of masonry structure in reverse, especially when the structures are under compressive loading conditions.

For brittle material, the block may not reach its ultimate limit state once the first cracking occurs. Instead, it can still undertake loading with consistently reduced tangent stiffness until it is damaged. For masonry material, differences in the stiffness of the bricks and the mortars lead to the energy at interfaces of different materials being dissipated, influencing components' ultimate limit state strength, which depends on components' strain condition.

Therefore, the dissipative process needs to be considered as introducing the dependence of the plastic strain tensor into yield function as:

$$f = f(\sigma, \kappa) \tag{2.24}$$

Where hardening parameter κ is scalar-valued, and it depends on strain history through plastic tensor $\boldsymbol{\varepsilon_p}$, defined as equation (2.25) with the strain-hardening hypothesis: [34]

$$\dot{\kappa} = \sqrt{2/3\left(\dot{\boldsymbol{\varepsilon}_p}\right)^T \boldsymbol{Q}\dot{\boldsymbol{\varepsilon}_p}} \tag{2.25}$$

Shift matrix $\boldsymbol{Q}$ is a diagonal one, as $\boldsymbol{Q} = diag[1, 1,1,\frac{1}{2},\frac{1}{2},\frac{1}{2}]$, which considers the effect of the shear tensor incorporated in the vector format $\boldsymbol{\varepsilon}$ of engineering shear strains notion.

Now equation (2.9) can be derived as:

$$f(\sigma, \kappa) = \frac{\partial f}{\partial \boldsymbol{\sigma}} \cdot \frac{\partial \boldsymbol{\sigma}}{\partial t} + \frac{\partial f}{\partial \kappa} \cdot \frac{\partial k}{\partial t} = \boldsymbol{n}^T \dot{\boldsymbol{\sigma}} + \frac{\partial f}{\partial \kappa}\dot{\kappa} = 0 \tag{2.26}$$

The plastic multiplier $\dot{\lambda}$ should always be positive [34]. Therefore equation (2.12) is able to be rederived as:

$$\boldsymbol{n}^T \dot{\boldsymbol{\sigma}} - h\dot{\lambda} = 0 \tag{2.27}$$

Where h is so-called the hardening modulus with function show as:

$$h = -\frac{1}{\dot{\lambda}}\frac{\partial f}{\partial \kappa}\dot{\kappa} \tag{2.28}$$

Combining equation (2.16) and flow rule (2.19) with consistency condition for hardening/ softening plasticity (2.28), we are able to get: [34]

$$\dot{\boldsymbol{\varepsilon}} = \left[(\boldsymbol{D}_e)^{-1} + \frac{1}{h}\boldsymbol{m}\boldsymbol{n}^T\right]\dot{\boldsymbol{\sigma}} \tag{2.29}$$

With the new explicit function of plastic multiplier $\dot{\lambda}$ shown as:

$$\dot{\lambda} = \frac{\boldsymbol{n}^T\boldsymbol{D}_e\dot{\boldsymbol{\varepsilon}}}{h + \boldsymbol{n}^T\boldsymbol{D}_e\boldsymbol{m}} \tag{2.30}$$

Therefore, the function of the stress vector rate $\dot{\boldsymbol{\sigma}}$ should be redefined as:

$$\dot{\boldsymbol{\sigma}} = \boldsymbol{D}_e\left(\dot{\boldsymbol{\varepsilon}} - \frac{\boldsymbol{n}^T\boldsymbol{D}_e\dot{\boldsymbol{\varepsilon}}}{h + \boldsymbol{n}^T\boldsymbol{D}_e\boldsymbol{m}}\boldsymbol{m}\right) \tag{2.31}$$

As a result, if we have the hardening parameter h, then the hardening/ softening behaviour will be considered in elastoplastic behaviour. While if $h = 0$, then we will return to ideal plasticity.

Return mapping algorithm

To obtain the strains and stresses in structure in generic loading stage, equation (2.29) must be integrated along the loading path. Here we use the one-point Euler forward integration rule, which is the most straightforward way. Such a scheme is fully explicit that the hardening modulus h and stress increment $\Delta\boldsymbol{\sigma}$ can be evaluated once strain increment $\Delta\boldsymbol{\varepsilon}$ is known. Therefore, the tangential stiffness matrix at the beginning of strain increment can be calculated directly [37].

If the initial stress point $\boldsymbol{\sigma_0}$ is on yield surface, stress increment at the beginning of strain increment can be evaluated from equation (2.32) as [37]

$$\Delta\boldsymbol{\sigma} = \boldsymbol{D}_e\left(\Delta\boldsymbol{\varepsilon} - \frac{\boldsymbol{n}_0^T\boldsymbol{D}_e\Delta\boldsymbol{\varepsilon}}{h_0 + \boldsymbol{n}_0^T\boldsymbol{D}_e\boldsymbol{m}_0}\boldsymbol{m}_0\right) \tag{2.32}$$

Where subscript "0" means flow direction $\boldsymbol{m}$, the direction of the yield surface $\boldsymbol{n}^T$ and the hardening modulus h are computed at the initial stress point. The new stress state $\boldsymbol{\sigma}_n$ at the end of the load step should be:

$$\boldsymbol{\sigma}_n = \boldsymbol{\sigma}_0 + \Delta\boldsymbol{\sigma} \tag{2.33}$$

If the initial stress point is located inside the yield surface, strain increment should be firstly subdivided into the purely elastic part, which is needed to reach the yield contour shown as $\Delta\boldsymbol{\varepsilon}_A$ in figure 2-16, and the part involving elastoplastic straining is shown as $\Delta\boldsymbol{\varepsilon}_B$ in figure 2-16. Now the stress increment is able to be proposed as [37]

$$\Delta\boldsymbol{\sigma} = \boldsymbol{D}_e\,\Delta\boldsymbol{\varepsilon}_A + \boldsymbol{D}_e\left(\Delta\boldsymbol{\varepsilon}_B - \frac{\boldsymbol{n}_c^T\boldsymbol{D}_e\Delta\boldsymbol{\varepsilon}_B}{h_c + \boldsymbol{n}_c^T\boldsymbol{D}_e\boldsymbol{m}_c}\boldsymbol{m}_c\right) \tag{2.34}$$

With subscript "c" means $\boldsymbol{m}$, $\boldsymbol{n}^T$ and h should be calculated at the critical stress point shown in figure 2-16. By substituting this new function of stress increment $\Delta\boldsymbol{\sigma}$ for equation (2.33), the stress point at the end of each step stage can be computed as:

$$\boldsymbol{\sigma}_n = \boldsymbol{\sigma}_0 + \boldsymbol{D}_e\,\Delta\boldsymbol{\varepsilon}_A + \boldsymbol{D}_e\left(\Delta\boldsymbol{\varepsilon}_B - \frac{\boldsymbol{n}_c^T\boldsymbol{D}_e\Delta\boldsymbol{\varepsilon}_B}{h_c + \boldsymbol{n}_c^T\boldsymbol{D}_e\boldsymbol{m}_c}\boldsymbol{m}_c\right) \tag{2.35}$$

This procedure could also be considered in another way as follows. [37] Firstly, the strain increment can be seen as purely “elastic” strains. Then stress increment is able to be computed as:

$$\Delta\boldsymbol{\sigma}_e = \boldsymbol{D}_e\,\Delta\boldsymbol{\varepsilon} \tag{2.36}$$

In this stage, the calculation is irrelevant whether the initial stress point is located inside or on the current yield surface. The strain increment is considered a trial increment based on the assumption of elastic straining during the whole loading increment. Possible plastic straining is not taken into account during this trial stage. [37]

Then, the total stress is set up as a sum of the initial stress at the beginning of the loading increment $\boldsymbol{\sigma}_0$ and the trial strain increment $\Delta\boldsymbol{\sigma}_e$ described as:

$$\boldsymbol{\sigma}_e = \boldsymbol{\sigma}_0 + \boldsymbol{D}_e\,\Delta\boldsymbol{\varepsilon} \tag{2.37}$$

Based on the yield criteria of Drucker-Prager yield surface introduced above, we are able to judge if such stress point is inside the yield surface by evaluating yield function $f(\boldsymbol{\sigma}_e, k_0)$ using equation (2.24) with $\boldsymbol{\sigma}_e$ and initial hardening parameter k_0. If $f(\boldsymbol{\sigma}_e, k_0) > 0$, then the plastic strain tensor, as a corrector, will have a value computed as:

$$\Delta\boldsymbol{\varepsilon}_p = \frac{\boldsymbol{n}_c^T\boldsymbol{D}_e\Delta\boldsymbol{\varepsilon}_B}{h_c + \boldsymbol{n}_c^T\boldsymbol{D}_e\boldsymbol{m}_c}\boldsymbol{m}_c \tag{2.38}$$

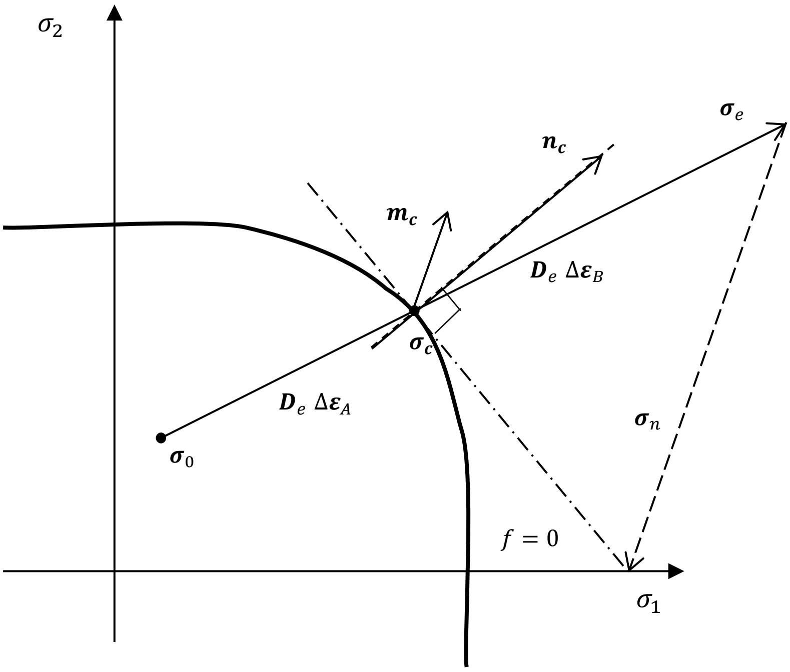


Figure 2-16 Explicit integration scheme: total strain increment should be divided into the purely elastic part and plastic part integrated with one-point Euler forward rule [34]

As a result, the stress state at the end of the loading stage should be:

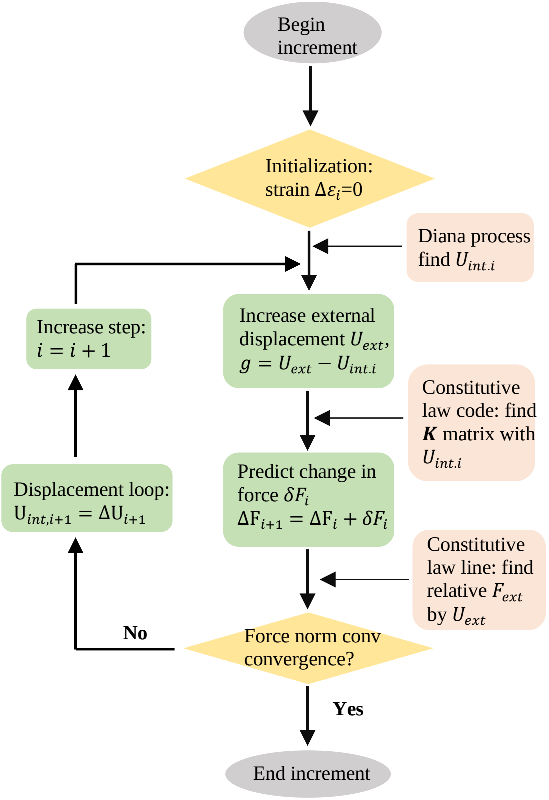
$$\boldsymbol{\sigma}_n = \boldsymbol{\sigma}_e - \Delta\lambda_c\boldsymbol{D}_e\boldsymbol{m}_c \tag{2.39}$$

$$\Delta\lambda_c = \frac{\boldsymbol{n}_c^T\boldsymbol{D}_e\Delta\boldsymbol{\varepsilon}_B}{h_c + \boldsymbol{n}_c^T\boldsymbol{D}_e\boldsymbol{m}_c} \tag{2.40}$$

The stress point $\boldsymbol{\sigma}_e$ is called as “elastic predictor”, while plastic stress point $\boldsymbol{\sigma}_p$ as “plastic corrector”.

This “elastic predictor- plastic corrector” process is a return mapping algorithm.

2.5. Diana FEA iterative process



Begin increment
Initialization: strain $\Delta\varepsilon_i$=0
Diana process find $U_{int.i}$
Increase step: $i = i + 1$
Increase external displacement U_{ext}, $g = U_{ext} - U_{int.i}$
Constitutive law code: find $\boldsymbol{K}$ matrix with $U_{int.i}$
Displacement loop: $U_{int,i+1} = \Delta U_{i+1}$
Predict change in force δF_i $\Delta F_{i+1} = \Delta F_i + \delta F_i$
Constitutive law line: find relative F_{ext} by U_{ext}
No
Force norm conv convergence?
Yes
End increment

Figure 2-17 Algorithm of incremental-iterative solution

The incremental-iterative solution introduced in Diana FEA documentation section 30.1 could be seen from figure 2-17 [38].

Firstly, initializing strain of per element increment at i step $\Delta\varepsilon_i$ being equal to zero. Displacement of model $U_{int.i}$ could be calculated based on the specific integration scheme, which is categorised by types of elements.

Secondly, increasing external displacement by applying displacement control. The increment of the displacement g could be calculated, and this increment could be used for finding the predicted change in force δF_i by introducing stiffness matrix $\boldsymbol{K}$ at displacement point $U_{int.i}$.

Thirdly, the corresponding force F_{ext} is able to be fond according to the external displacement U_{ext} by introducing strain-stress curve from the constitutive law of the material model and the particular integration scheme.

Finally, comparing the predicted force $\Delta F_{i+1} = \Delta F_i + \delta F_i$ and the external force F_{ext}. If the convergency condition is not satisfied, then the incrementing loop would occur, and the process would go back to step 2.

3. Theory and assumptions

The relative assumptions are derived based on the theory from chapter 2 in this part. There are two parts:

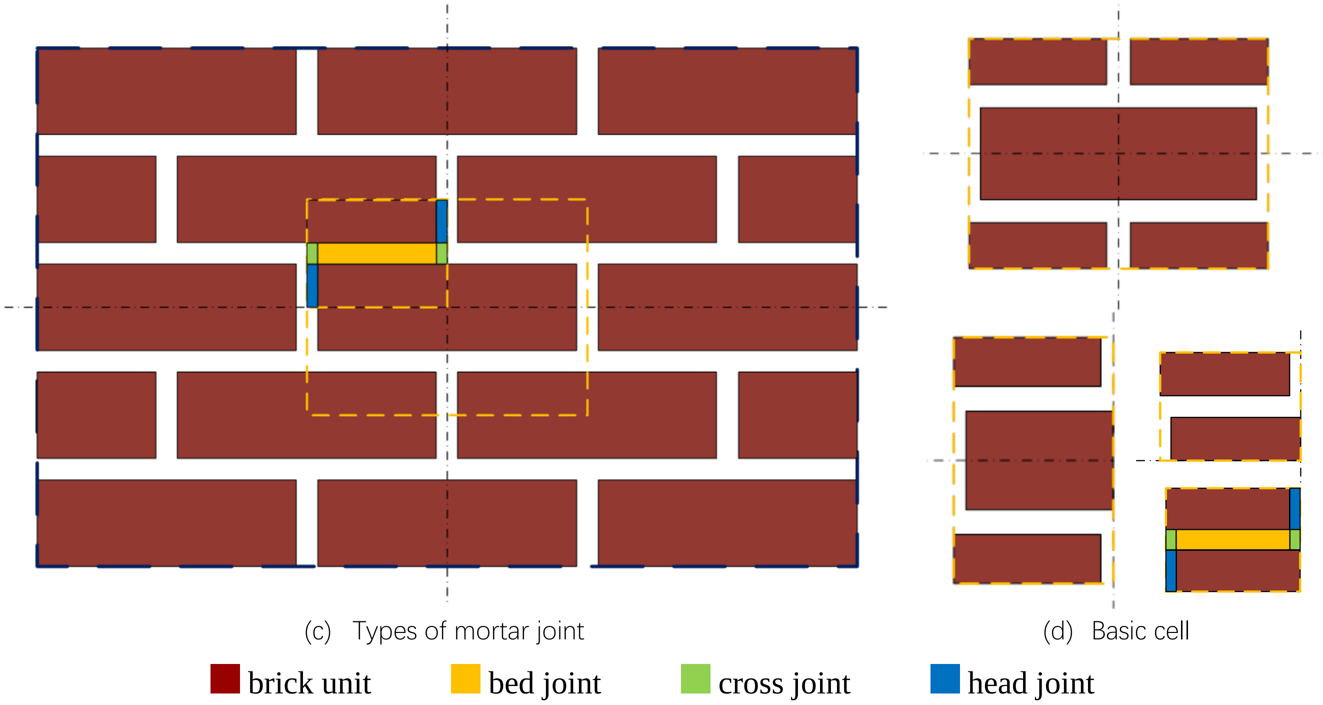
(1) The homogenised cell's definition is based on the one built by Zucchini and Lourenço's in 2002 [30].
(2) 4 main deformed cells are studied for behaviours of masonry structures in shear, horizontal tensile and vertical compressive direction.

These deformed cells are established according to the numerical results from A. Zucchini and P.B. Lourenço's research in 2002 for interaction behaviours of micro-mechanical model [30] and the numerical results of internal stress distribution from P.B. Lourenço and J. Pina-Henriques in 2006 [35]. The stress and strain tensors, which are assumed to be relative to the micro-constitutive model of components in the basic cell under different loading cases, are suitably chosen in this part according to Zucchini's formulation [20], while all the others are neglected by assuming their value to be zero. The derivations of the homogenized constitutive laws, including proposing relative kinematic relations, equilibrium equations of the system, plastic deformations, will be introduced in chapters 4 to 7 in detail.

In this chapter, these deformed cells are firstly drawn based on the assumed failure mechanisms, and then the internal stresses following the deformed cells are selected for all models. Other assumptions on the elastoplastic phase in model 3 and the coupled behaviours under mixed loading in model 4, such as combined shear and vertical compressive loading, are also made based on previous researches introduced in chapter 2.

3.1. Definition of deformed cell

Unreinforced masonry structures normally consist of brick units and mortar joints with different mechanical properties. In this research, masonry with the staggered alignment of brick units is considered. Three types of mortar connections are considered, being categorized by their locations relative to the brick units. These are the vertical head joints, located beside the brick units; the horizontal bed joints, located under or above the brick units; the cross joints, located at the corner of the brick units, where the head and bed joints overlap, which can be seen from figure 3-1(a).



(c) Types of mortar joint

(d) Basic cell

Figure 3-1 Assumption of per basic cell

The representative plane could be extracted from the periodic system of each layer consisting of the brick units and the vertical head joints based on masonry walls' symmetrical geometry, see figure 3-1 (b). This plane is called "basic unit cell", and its geometry should be relative to the geometrical properties of components.

3.2. Model 1: shear behaviour

In this model, the unit cell deforms under pure external shear loading, which means that only the external shear strain is considered to be the homogenized strain. Based on this assumption, the deformed cell of model 1 could be drawn as figure 3-2.

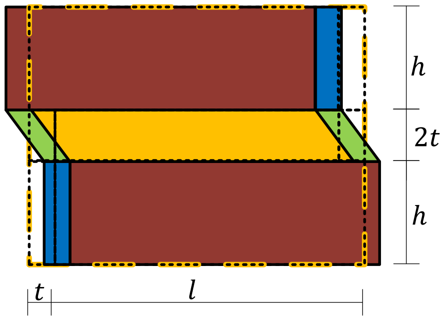


Figure 3-2 Deformed Cell of Model 1

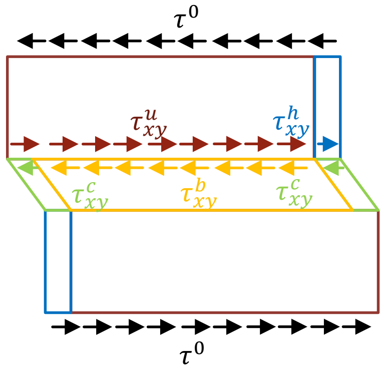


Figure 3-3 Internal stresses Model 1

🟥 brick unit 🟨 bed joint 🟩 cross joint 🟦 head joint

Note: "0" means external force, "u" means brick unit (red), "h" means head joint (blue), "c" means cross joint (green), "b" means bed joint (orange)

Furthermore, the failure pattern of model 1 is assumed to be "shear sliding", the one described in chapter 2 section 2.1 figure 2-1 (b), which means only the in-plane shear stresses of the cell are considered. Components, including the brick unit and the head, the bed, and the cross joint, fail once their shear stresses reach their strengths without hardening. As can be seen in figure 3-2, the considered stresses are:

1. External shear stress τ^0;
2. Internal shear stresses: between the brick unit and bed joint τ^u_{xy}, τ^b_{xy}; between the brick unit and cross joint τ^u_{xy}, τ^{cu}_{xy}; between head joint and cross joint τ^h_{xy}, τ^{ch}_{xy}

τ^0 could be firstly relative to the internal shear stresses τ^u_{xy}, τ^b_{xy}, τ^u_{xy}, τ^{cu}_{xy}, $\tau^h_{xy}, \tau^{ch}_{xy}$ using equilibrium equations of the system. And then, the damage factors, calculated from the internal shear stresses at each interface (brick-head joint interface, brick-bed joint interface, head-cross joint interface), should be attached to the value of τ^0, since the values of τ^u_{xy}, τ^b_{xy}, τ^u_{xy}, τ^{cu}_{xy}, $\tau^h_{xy}, \tau^{ch}_{xy}$ are considered to be the undamaged ones. In this case, the shear cracks may occur in each component at different times, and the basic cell would be fully damaged after all of the components failed in shear.

3.3. Model 2: horizontal tensile behaviour

In this model, the unit cell is assumed to be the only crack in the horizontal tension direction, which means each component could deform horizontally caused by the external loading and vertically considering the positive poison ratio of the unit cell. In other words, the homogenized unit only has lateral tension strain and stress. However, according to Zucchini's research in [30], the basic cell under horizontal loading has displacement caused by the internal shear deformation between the bed joint and the brick unit and the tensile deformations of the components. The deformation of the unit cell in model 2 can be seen from figure 3-4.

This deformed cell, in which the shear, the tensile, and compressive behaviours of each component should be considered, is assumed to be damaged in tensile cracking failure mode, see chapter 2 figure 2-1 (c). The internal stresses should be selected based on the deformation assumption of model 2 shown in figure 3-4, especially the internal shear stresses.

According to those assumptions of model 2, the internal stresses are:

External tension stress σ^0_{xx};

1. Internal horizontal stresses: brick unit σ^u_{xx}, head joint σ^h_{xx}, bed joint σ^b_{xx}, cross joint σ^c_{xx};
2. Internal vertical stresses: brick unit σ^u_{yy}, head joint σ^b_{yy}, bed joint σ^b_{yy}, cross joint σ^c_{yy};
3. Internal shear stresses: between the brick unit and bed joint $\tau^u_{xy} = \tau^b_{xy}$.

The stress distribution of the deformed cell could be seen from figure 3-5. Homogenized stress σ^0_{xx} can be relative to the internal stresses by deriving equilibrium equations at side boundaries. And then, the internal stresses could be substituted for the damaged ones by introducing the damage factors. In this case, homogenously horizontal stresses σ^0_{xx} could be relative to the internal stresses and the damage factors in shear, horizontal and vertical directions of the components. The damage factors depend on the value of the corresponding stresses. For instance, the damage factor of brick in the x-direction d^u_{xx} is computed by the internal horizontal stress σ^u_{xx}. The scalar value of the damage variable d^u_{xx} (damage factor) can represent the damage level of the brick unit in the x-direction, as "0" means the brick is undamaged in the x-direction, while "1" means the brick is damaged in the x-direction.

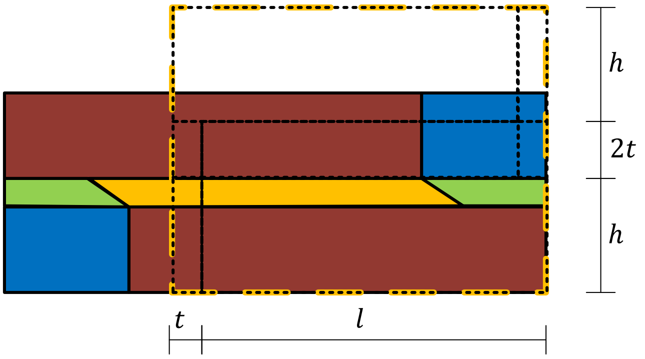


Figure 3-4 Deformed Cell of Model 2

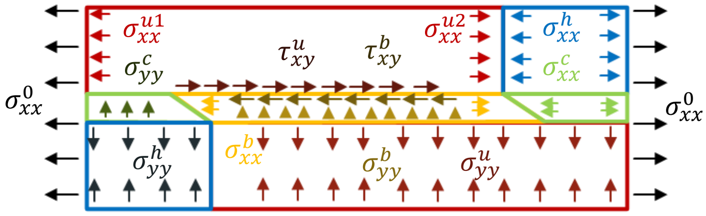


Figure 3-5 Internal stresses Model 2

🟥 brick unit 🟨 bed joint 🟩 cross joint 🟦 head joint

Note: "0" means external force, "u" means brick unit (red), "h" means head joint (blue), "c" means cross joint (green), "b" means bed joint (orange)

As a result, there are 10 internal stresses in model 2. That means the macro stress σ^0_{xx} is relative to 10 damage factors together with 10 internal stresses. There should be a vast computational cost if we derive σ^0_{xx} in this way. Therefore, the damage factors should be selected to save the computational time by making the following assumptions of model 2.

In this project, the damage factors are firstly computed by the value of the internal stresses of components based on the exponential softening process. And then, these damage factors are implemented in the expression of macro stress by substituting the damaged internal stresses for the undamaged ones to couple failure mechanisms of the brick unit and the joints together. Therefore, the derivation of σ^0_{xx} could be simplified by selecting the suitable failure mechanisms of components.

According to A. Zucchini and P.B. Lourenço's research in 2002 [30]:

The failure mode of Model 2 is assumed to be the coupled failure pattern, which means failure mechanisms of shear sliding at the interface between the brick unit and the bed joint and the unit tensile cracking failure patterns are considered. The significant damage occurring in the homogenized cell is caused by cracks generated in the x-direction of the brick unit, the head and the cross joint, and shear failure between the bed joint and the brick unit. As a result, damage factors in x-direction computed by the internal tension stresses of the brick unit, the head and the cross joint, and the damage factor computed by the shear stress between the bed joint and the brick unit are the dominating coefficients selected for calculating homogenized stress.

3.4. Model 3: vertical compression behaviour

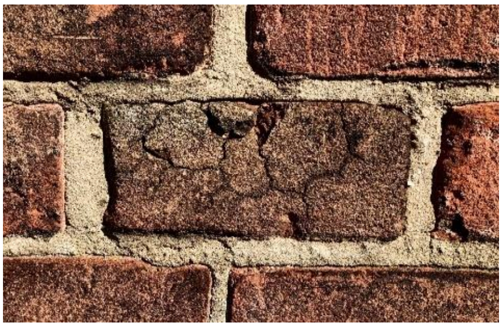


Figure 3-6 localized damage in brick

Crushing in brick occurs typically when the masonry walls are under vertical compression loading, shown in figure 3-6. The experimental results indicated that both local and continuum fracturing processes governing the compressive behaviour of the masonry introduced by Maurizio Angelillo et al. in [39]. Therefore, model 3 is built up based on two main parts:

(1) The damage model for continuum fracturing process;
(2) The plastic deformation is introduced in bricks and mortar for the local fracturing process.

3.4.1. Damage model

The damage model is firstly proposed as models 1 and 2 did. The deformed mechanisms with the elastic properties of the bricks and the mortars are proposed for model 3. That means the internal stresses of components increase linearly as the external loading is gradually imposed. Until the values of the stresses reach their strength, the components' internal stresses drop to zero immediately, following the exponential softening process. The deformations of the basic unit cell loaded by vertical compressive force could be drawn as figure 3-7 shown based on Zucchini's works [30] introduced in chapter 2.

Figure 3-7 demonstrates that the head joint is subjected to the mixed shear and normal stresses while other components are subjected to the normal stresses only when only the vertical compressive load case is taken into account.

In the micro-mechanical model, the loadings effect on the behaviours of the head joints under mixed shear and normal stresses are hardly included. In 1997, the results of Lourenço's research showed that the errors between the homogenized model that included and excluded mixed shear behaviour of the head joint were smaller than 2% recorded by Zucchini et al. in 2002. [30] Therefore, this deformed cell could be simplified by neglecting the shear stresses of the head joints overall the behaviours of the basic cell as figure 3-8 shown. As the dilatant effects being included, the deformed cell of model 3 is assumed as figure 3-9 shown.

As can be seen from figure 3-10, the internal stress of each component from model 3 is similarly distributed as from model 2, the differences displayed in the cross and the bed joint that the horizontal stresses of them shift from in tensile to in compressive direction supported by the research introduced in chapter 2 section 2.1.3. The vertical stress of the basic unit cell considered in this model is σ_{yy}^{0} in compression.

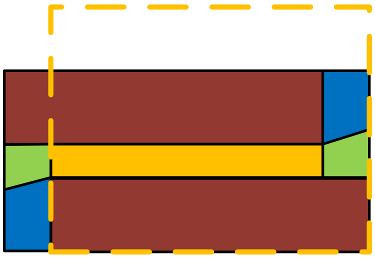


Figure 3-7 deformed cell from Zucchini et al. in 2002 [30]

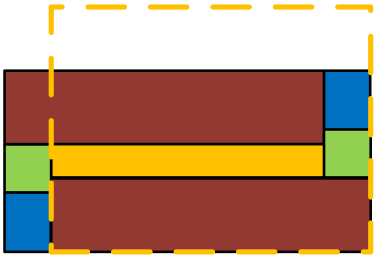


Figure 3-8 simplified deformed cell

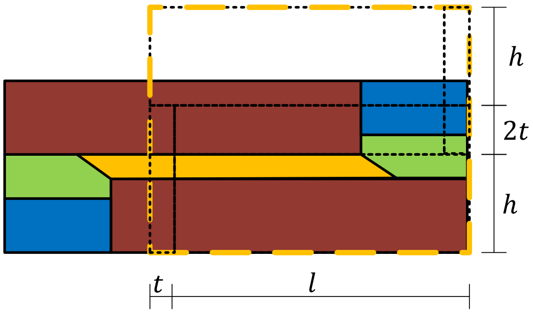


Figure 3-9 deformed cell of model 3 with dilatancy angle

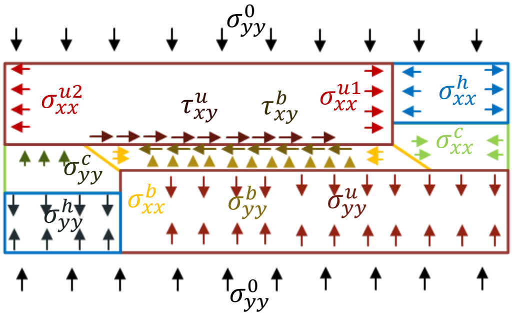


Figure 3-10 internal stresses of components

🟥 brick unit 🟨 bed joint 🟩 cross joint 🟦 head joint

Note: "0" means external force, "u" means brick unit (red), "h" means head joint (blue), "c" means cross joint (green), "b" means bed joint (orange)

3.4.2. Elastoplastic deformation

The plastic strain tensor and the hardening parameters of the components are introduced in this model to represent the inelastic behaviour of the brick unit and the joints under compressive loading, which is considered a localized damage process. As equation (2.38) introduced in chapter 2 section 2.4.2 shown, the plastic strain increment is relative to the value of the hardening modulus, the direction of yield surface and the direction of potential energy at critical stress point on yield surface and the strain increment calculated from load increment at each step together with the geometrical properties of the basic unit cell.

In this model, Drucker-Prager yield criteria and their relevant protentional energy function are introduced based on the theory listed in chapter 2 section 2.4.2. Hardening modulus h_c used for reflecting the hardening and softening process, could be calculated from equation (2.28) introduced in chapter 2 section 2.4.2.

It can be assumed that the strain increment at each load step is the same. Therefore, the function of the hardening modulus can be rewritten as:

$$h_c = -\frac{1}{\Delta\lambda_c}\frac{\partial f}{\partial\kappa}\bigg|_c \Delta\kappa \tag{3.1}$$

Where $\Delta\lambda_c$ is the plastic multiplier, definition of which will be introduced in detail later in chapter 6 and κ is the hardening parameter, typically depends on the strain history through plastic tensor $\boldsymbol{\varepsilon}_p$ gained from the strain-hardening hypothesis.

For 4-node plane element, tensor $\boldsymbol{\varepsilon}_p$ could be displayed in matrix form:

$$\boldsymbol{\varepsilon}_p^T = \begin{bmatrix}\varepsilon_{xx}^p & \varepsilon_{yy}^p & 2\varepsilon_{xy}^p\end{bmatrix}, \qquad \boldsymbol{\varepsilon}_p = \begin{bmatrix}\varepsilon_{xx}^p \\ \varepsilon_{yy}^p \\ 2\varepsilon_{xy}^p\end{bmatrix} \tag{3.2}$$

If shear behaviour was neglected, the function of the hardening parameter κ at load step $t = t$ is:

$$\kappa = \sqrt{\frac{2}{3}\boldsymbol{\varepsilon}_p^T \boldsymbol{Q} \boldsymbol{\varepsilon}_p} = \sqrt{\frac{2}{3}\left[\left(\varepsilon_{xx}^p\right)^2 + \left(\varepsilon_{yy}^p\right)^2\right]} = \sum_{t=0}^{t=t} \Delta\kappa \,, \boldsymbol{Q} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{1}{2} \end{bmatrix} \tag{3.3}$$

Dependence of yield function on loading history only through the scalar-valued hardening parameter κ, which also could be seen as equivalent plastic strain ε_{yy} as figure 3-11 shown. The yield surface could only expand or shrink but could not translate or rotate in stress space. [34]

In this model, the cohesion is supposed to vary during softening or hardening phase assessed by the hardening parameter, while the values of other material parameters, such as friction angle ϕ, are supposed to be constants of the components. The function of cohesion c has been proposed in chapter 2 section 2.4.2.

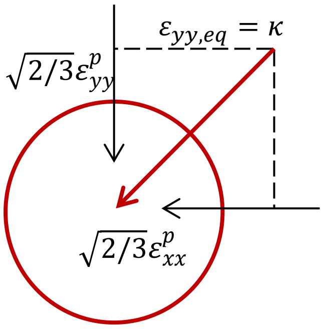

$\varepsilon_{yy,eq} = \kappa$
$\sqrt{2/3}\varepsilon_{yy}^p$
$\sqrt{2/3}\varepsilon_{xx}^p$

Figure 3-11 κ physically defined by plastic strain tensor $\boldsymbol{\varepsilon}_p$

The scalar value of cohesion depends on material constant ϕ and compression strength f_c which is a variable relative to the value of the hardening parameter κ.

In conclusion, the yield surface could expand or shrink following with the hardening or softening phase through cohesion c, which varies by compression strength of the material f_c defined by the hardening parameter κ through the plastic strain tensor $\boldsymbol{\varepsilon}_p$.

The function of cohesion could be rederived in the form of equation (3.4) shown.

$$c = \frac{1 - \sin\phi}{2\cos\phi}\sigma_{yy,eq}(\kappa) = \frac{1 - \sin\phi}{2\cos\phi}\sigma_{yy,eq}\left(\varepsilon_{yy,eq}\right) \tag{3.4}$$

It is noted that this scalar value should be calculated at the critical stress point on the yield surface since the elastoplastic behaviours of the components could and only could exist when stress points are located on the yield criteria. Therefore, the value of compression strength f_c at the end of each load step could be substituted by an equivalent compression stress $\sigma_{yy,eq}$, which depends on an equivalent strain $\varepsilon_{yy,eq}$ through an inelastic law of quasi-brittle material regime under pure compression.

According to the function of Drucker-Prager yield criteria introduced in chapter 2 section 2.4.2 equation (2.9), the derivative of yield function f by the hardening parameter κ at the critical stress point could be expressed by the slope of the inelastic strain-stress curve of the quasi-brittle material regime under pure compression:

$$\left.\frac{\partial f}{\partial \kappa}\right|_c = \frac{\partial f}{\partial c} \cdot \frac{\partial c}{\partial \sigma_{yy,eq}} \cdot \frac{\partial \sigma_{yy,eq}}{\partial \varepsilon_{yy,eq}} \tag{3.5}$$

Bi-parabolic law introduced by Zucchini et al. in 2007 [32] is applied here as figure 3-12 shown.

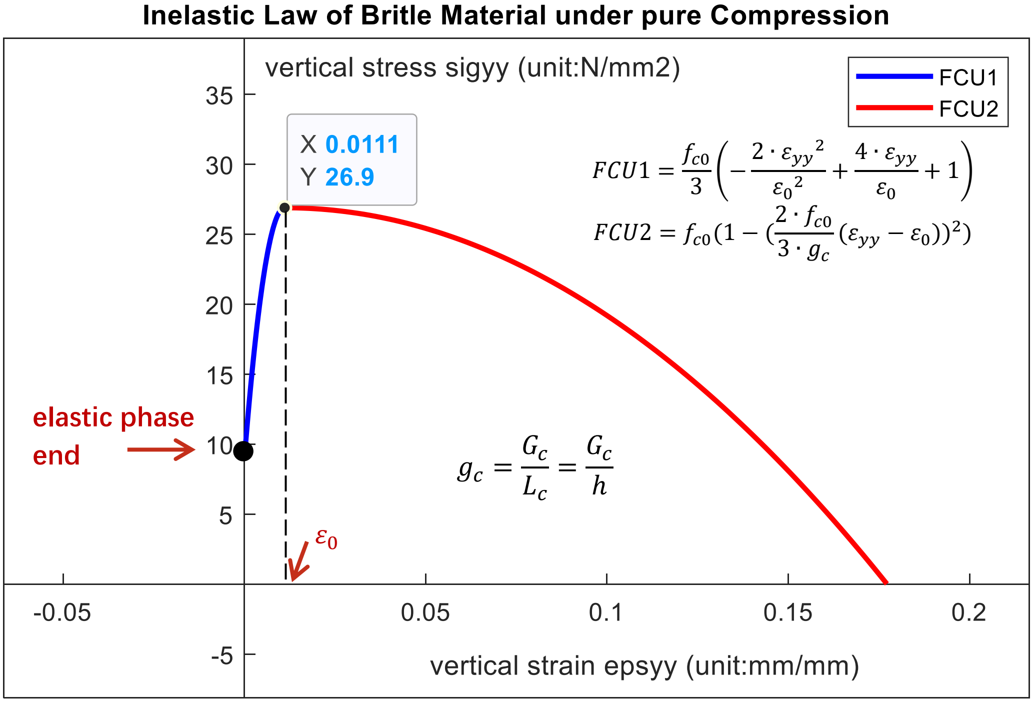


Figure 3-12 Bi-parabolic law of brittle material under pure compression: vertical stress σ_{yy}^0 versus vertical strain ε_{yy}^0

Let us substitute $\sigma_{yy,eq}$ for $FCU1$ and $FCU2$, $\varepsilon_{yy,eq}$ for ε_{yy}:

$$\sigma_{yy,eq} = \begin{cases} \frac{f_{c0}}{3}\left(-\frac{2 \cdot \varepsilon_{yy,eq}{}^2}{\varepsilon_0{}^2} + \frac{4 \cdot \varepsilon_{yy,eq}}{\varepsilon_0} + 1\right), & if\ 0 \leq \varepsilon_{yy,eq} \leq \varepsilon_0 \\ f_{c0}\left(1 - \left(\frac{2 \cdot f_c}{3 \cdot g_c}(\varepsilon_{yy,eq} - \varepsilon_0)\right)^2\right), & if\ \varepsilon_0 < \varepsilon_{yy,eq} \leq \varepsilon_{max} \end{cases} \tag{3.6}$$

Where f_{c0} is peak stress and $\varepsilon_0 = 2f_{c0}/E$ is peak equivalent plastic strain with young's modulus E. g_c is post-specific fracture energy defined by compression fracture energy G_c and characteristic length L_c, with $L_c = h$ for the smeared crack model. h is the element size.

It can be seen from figure 3.12, the compressive equivalent stress should be equal to $f_{c0}/3$, at which point the elastic phase end when $\varepsilon_{yy,eq} = 0$.

3.5. Model 4: coupled behaviour

Behaviours of the homogenized cell under pure shear, horizontal tension and vertical compression loading were introduced above in section 3.2, 3.3 and 3.4 separately. However, the failure patterns may be changed under mixed loading conditions. For instance, the basic cell under mixed shear and vertical compression loading should fail in the diagonal cracking model, see figure 2-1 (d) in chapter 2 section 2.1, rather than sliding one under pure shear loading or crushing one under pure vertical loading. Therefore, combinations of shear, horizontal and vertical behaviours should also be worth to be studied.

Shear behaviour from model 1, horizontal behaviour from model 2 and vertical behaviour from model 3 are combined. Transverse strain in the vertical direction (or horizontal direction) relative to horizontal (or vertical) loading should be considered. Meanwhile, the coupled behaviour of the basic cell under combined shear and vertical loading is studied in this part.

3.5.1. Combination of vertical and horizontal behaviour

Usually, quasi-brittle materials should have positive poison ratios, which means this type of materials should shrink in the vertical direction and expand in the horizontal direction when they are loaded by vertical compressive loading or horizontal tensile loading.

According to this phenomenon, horizontal (or vertical) strain increment at each load step could be assumed as consisting of two main parts: the one from directly horizontal (vertical) loading ε^0_{xx} (or ε^0_{yy}) together with the one from the corresponding deformation caused by vertical (or horizontal) loading $\varepsilon^0_{xx,y}$ (or $\varepsilon^0_{yy,x}$), see figures 3-13 and 3-14.

It can be noted that the deformed cell assumption should be correlated to the horizontal tension or the vertical compression behaviours introduced in section 3.3 or 3.4. For example, it can be seen from figure 3-13 that the deformation of the basic cell should be similar to the one under horizontal tension loading proposed in section 3.3. the main reason is that the output variable, in this case, is horizontal stress σ^0_{xx} which means we focus on the “horizontal tensile cracking” failure pattern, see figure 2-1 (c).

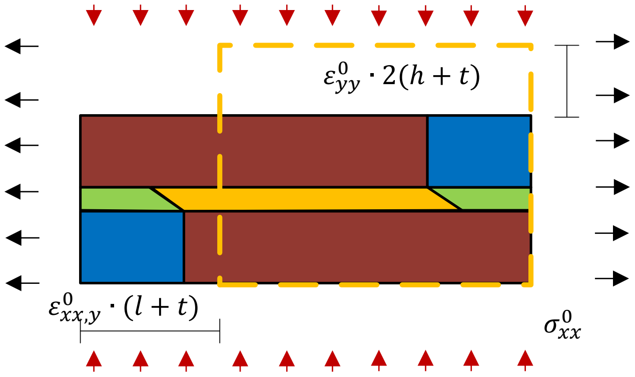

$\varepsilon^0_{yy}\cdot 2(h+t)$
$\varepsilon^0_{xx,y}\cdot(l+t)$
σ^0_{xx}

Figure 3-13 horizontal strain $\varepsilon^0_{xx,y}$ caused by vertical strain increment ε^0_{yy}

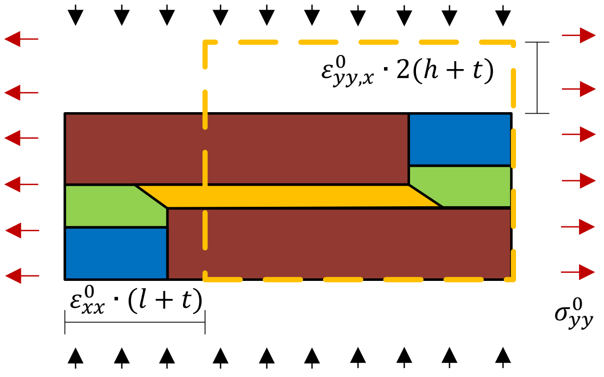

$\varepsilon^0_{yy,x}\cdot 2(h+t)$
$\varepsilon^0_{xx}\cdot(l+t)$
σ^0_{yy}

Figure 3-14 vertical strain $\varepsilon^0_{yy,x}$ caused by horizontal strain increment ε^0_{xx}

Note: direction of input loading is indicated by red arrows, while black arrows indicate output stress direction

As a result, the new input strains $\varepsilon^0_{xx,\,total}$ and $\varepsilon^0_{yy,\,total}$ in both directions could be expressed by ε^0_{xx} and ε^0_{yy}:

$$\varepsilon^0_{xx,\,total} = \varepsilon^0_{xx} + \varepsilon^0_{xx,y} = \varepsilon^0_{xx} + f_{xx0y}\left(\varepsilon^0_{yy}\right) \tag{3.7}$$

$$\varepsilon^0_{yy,\,total} = \varepsilon^0_{xx} + \varepsilon^0_{yy,x} = \varepsilon^0_{yy} + f_{yy0x}(\varepsilon^0_{xx}) \tag{3.8}$$

Coupled behaviour of the homogenized cell under horizontal tensile together with vertical compressive loading could be obtained by substituting the new input strains $\varepsilon^0_{xx,\,total}$ and $\varepsilon^0_{yy,\,total}$ for the original ones. As a result, the relation of macro strain and stress tensor could be derived in the form of:

$$\begin{bmatrix}\sigma^0_{xx}\\ \sigma^0_{yy}\end{bmatrix} = \begin{bmatrix}K_{11} & K_{12}\\ K_{21} & K_{22}\end{bmatrix}\begin{bmatrix}\varepsilon^0_{xx}\\ \varepsilon^0_{yy}\end{bmatrix} \tag{3.9}$$

$$K_{11} = \frac{\partial \sigma_{xx}^0}{\partial \varepsilon_{xx}^0}, K_{22} = \frac{\partial \sigma_{yy}^0}{\partial \varepsilon_{yy}^0} \tag{3.10}$$

$$K_{12} = \frac{\partial \sigma_{xx}^0}{\partial \varepsilon_{xx,y}^0} \cdot \frac{\partial \varepsilon_{xx,y}^0}{\partial \varepsilon_{yy}^0}, K_{21} = \frac{\partial \sigma_{yy}^0}{\partial \varepsilon_{yy,x}^0} \cdot \frac{\partial \varepsilon_{yy,x}^0}{\partial \varepsilon_{xx}^0} \tag{3.11}$$

3.5.2. Combination of shear and vertical behaviour

Diagonal tensile cracking generally occurs in masonry under shear and vertical compressive loading, see figure 2-1 (d) in chapter 2 section 2.1. In this project, the brick units are always supposed to be stiffer than mortar joints, and only the type of masonry structures with a staggered arrangement of brick units and mortar joints are studied. Therefore, we can assume that diagonal tensile cracking could and only could occur in the vertical joints, also called head joints, in this study.

The failure mechanisms could be concluded as following steps:

(1) Firstly, the shear stress between the brick unit and bed joint increases in both of elastic and elastoplastic phase, the micro-fissures are generating in components under vertical compressive loading as model 3 proposed at the same time;
(2) Secondly, the shear stress at the interface of the brick unit and bed joint is larger than the cohesion of the head joint C_H plus the dynamical friction σ_{yy}^0·tan (ϕ);
(3) Then, the horizontal interface starts sliding along the length direction since the resistance of it at side boundary conditions are damaged (residual shear stress is larger than cohesion in head joint);
(4) Finally, the shear sliding maintains at the interface of the brick unit and bed joint under external shear loading.

Additionally, the homogenized cell should still be damaged in pure shear if the vertical loading is tensile. As a result, the maximum value of the homogenized shear stress under shear together with vertical loading is:

$$\begin{cases} \tau_{xy}^0 = \sigma_s^b, & if\ \sigma_{yy}^0 \geq 0 \\ \tau_{xy}^0 = C_H + \sigma_{yy}^0 \cdot \tan(\phi), & if\ \sigma_{yy}^0 < 0 \end{cases} \tag{3.12}$$

Where σ_s^b and ϕ are shear strength and friction angle of bed joint, respectively. C_H is the cohesion of the head joint, which varies following the hardening or softening process and depends on the coupled behaviour of the basic cell under vertical tensile together with vertical compressive loading. σ_{yy}^0 is the homogenized vertical stress.

As can be seen from figure 2-2 in chapter 2 section 2.1.1, the definition of shear fracture energy in coupled behaviour is the same as that in pure shear behaviour. Therefore, consumption of cohesion in the head joint could be considered by a similar approach as model 1 did. Constitutive law of the homogenized cell in this study could be concluded as the following form considering coupled behaviour of all loading conditions in the 2D plane:

$$\begin{bmatrix} \sigma_{xx}^0 \\ \sigma_{yy}^0 \\ \tau_{xy}^0 \end{bmatrix} = \begin{bmatrix} K_{11} & K_{12} & 0 \\ K_{21} & K_{22} & 0 \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \begin{bmatrix} \varepsilon_{xx}^0 \\ \varepsilon_{yy}^0 \\ \varepsilon_{xy}^0 \end{bmatrix} \tag{3.13}$$

$$K_{33} = \frac{\partial \tau_{xy}^0}{\partial \varepsilon_{xy}^0} \tag{3.14}$$

$$K_{31} = \frac{\partial \sigma_{yy}^{0}}{\partial \varepsilon_{yy,x}^{0}} \cdot \frac{\partial \varepsilon_{yy,x}^{0}}{\partial \varepsilon_{xx}^{0}},\ K_{32} = \frac{\partial \sigma_{yy}^{0}}{\partial \varepsilon_{yy}^{0}} \tag{3.15}$$

Where K_{11}, K_{12}, K_{21} and K_{22} could be found in equation (3.11). The homogenized tensile σ_{xx}^{0} and compressive stresses σ_{yy}^{0} can be obtained from models 2 and 3, respectively, by substituting the new external strain $\varepsilon_{xx,total}^{0}$, $\varepsilon_{yy,total}^{0}$ for the original ε_{xx}^{0}, ε_{yy}^{0}, while the shear one could be computed by the value C_H and σ_{yy}^{0} obtained in model 3 in a similar way applied in model 1.

4. Model 1: shear behaviour

In this model, the homogenized unit cell is supposed to be damaged by only shear stresses inside the assumed cell. According to the assumptions made in chapter 3, the unit fails once the shear stress between the brick unit and the bed joint reaches its strength. Based on this idea, damage model 1 can be derived from the following concepts.

4.1. Derive "damage" equations

To derive the "damage" equations of model 1, the damage factors, used as the internal state variables, which could show the situation of micro-cracking generated in the components, should be firstly formulated by the exponential relation of the internal stresses. In this model, cracks are only allowed to appear at the interface of the brick unit and the mortar in shear. Based on zucchini's research in [20], the isotropic damage model with a single damage variable in shear of each component of the basic cell has been adopted as:

(a) Scalar damage model

The undamaged $\boldsymbol{\tau}_{xy}^{i}$ and damaged $\boldsymbol{\tau}_{d}$ shear tensor are correlated according to the theories of continuum damage model stated in Oliver's research in [31], the equation evolved for shear behaviour can be:

$$\boldsymbol{\tau}_d = (1-d)\boldsymbol{D}\boldsymbol{\gamma}_{xy}^{i} = (1-d)\boldsymbol{\tau}_{xy}^{i}, \qquad i = b, u, h, c \tag{4.1}$$

Where d is damage state variable with a scalar value, ranging from 0 to 1, as "0" means undamaged while "1" means damaged state at local system. $\boldsymbol{D}$ is elastic stiffness matrix and $\boldsymbol{\gamma}_{xy}^{i}$ is local shear strain tensor.

(b) Limit damage surface

Damage criterion should be decided by shear strength. The initial threshold values of the shear stresses of the components should be equal to their shear strength.

$$\tau_{max} = \sigma_s \tag{4.2}$$

Where σ_s is the shear strength of the given component and τ_{max} is the maximum value of shear stress of the given cell component.

(c) Equivalent effective stress

A suitable norm, the so-called equivalent effective strain or stress, compares the different states of the deformation [31] and then decide each cell component's damaged state. This norm is the damage threshold at the current time or iterative step shown as [31]:

$$\boldsymbol{\tau}_{xy}^{i} = G_{xy}^{i}\boldsymbol{\gamma}_{i}, \qquad i = b, u, h, c \tag{4.3}$$

$$\tau = \max\{\tau_{xy}^{i}, \sigma_s\}, \qquad i = b, u, h, c \tag{4.4}$$

Where $\boldsymbol{\tau}_{xy}^{i}$ and $\boldsymbol{\gamma}_{i}$ are the equivalent effective stress and strain tensor, τ is the norm of damage threshold at the current time with a scalar value. Subscripts "b, u, h, c" mean the variables of the bed joint, the brick unit, the head joint and the cross joint, respectively.

(d) Damage evaluation law

Shear behaviours of the components here are considered to be similar to the tensile ones. Therefore, the scalar function of shear damage factor is adopted as tensile concrete-like material here proposed in [31]:

$$d = 1 - \frac{\sigma_s}{\tau} e^{A_s\left(1 - \frac{\tau}{\sigma_s}\right)} \tag{4.5}$$

Where A_s is a parameter present based on the shape of the shear stress-strain curve observed from the experiment, the evaluation of damage coefficient must be monotonic [31], and this irreversible damage process is taken into account by updating the value of d.

(e) Correlation with fracture parameter

The fracture energy in shear (sliding crack model) can be similar to that in tension (first fracture energy). The explicit function of the parameter A_t is proposed in [31], where A_t could be replaced by A_s in shear fracture mechanics [20] in this model. Parameter A_s is then able to be relative to the dissipated energy in shear g^{II} by:

$$g^{II} = \frac{\sigma_s^2}{G}\left(\frac{1}{2} + \frac{1}{A_s}\right) \tag{4.6}$$

Where G is the shear modulus.

As the smeared cracking model is considered in this work, the characteristic length of this model can be:

$$l_s = \frac{G^{II}}{g^{II}}, \qquad l_s = H \tag{4.7}$$

G^{II} is the second fracture energy per unit area (assumed to be a material parameter) and l_s is a characteristic length of finite element, H is finite element size.

Note that A_s can be acquired from equations (4.6) (4.7) as:

$$A_s = \left(\frac{G^{II} G}{l_s \sigma_s^2} - \frac{1}{2}\right)^{-1} \tag{4.8}$$

With equations (4.1) (4.2) (4.3) (4.4) (4.5) (4.8), the damage state variables in shear can be related to the shear strains and stresses of the given cell components. These variables could be relative by only one external shear strain γ_0 by the kinematic relations and the equilibrium equations of the system derived based on the deformed cell assumed in chapter 3 section 3.2.

4.1.1. Kinematic relation

Based on the deformed cell assumed in chapter 3 section 3.2, the relation between the internal shear strain of each component and the external shear strain could be derived according to the displacement equations. Geometrical properties and the deformation of per cell in detail are able to be seen from figure 4-1. Therefore, the equations could be proposed as equations (4.9) to (4.12) show.

$$\Delta u_b = \Delta u_c = \Delta u \tag{4.9}$$

$$\gamma_0 = \frac{2\Delta u}{2(h+t)}, \gamma_b = \gamma_u = \gamma_c = \gamma_h \tag{4.10}$$

$$\gamma_b = \frac{2\Delta u_b}{2t} = \frac{2\Delta u}{2t}, \gamma_c = \frac{2\Delta u_c}{2t} = \frac{2\Delta u}{2t} \tag{4.11}$$

Therefore:

$$\gamma_b = \gamma_u = \gamma_c = \gamma_h = \frac{h+t}{t}\gamma_0 \tag{4.12}$$

Where Δu is the total displacement of the basic cell and Δu_b is the deformations caused by shear stress between the brick unit and the bed joint, Δu_c is the shear deformation between the head and cross joint. The thickness of the mortar is assumed to be a scalar value of $2t$, while the heights of the brick unit and the head joint are assumed to be $2h$.

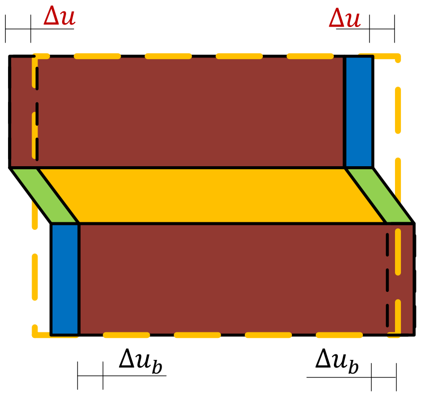


Figure 4-1 Deformed Cell of Model 1

The explicit function proposed in equation (4.9) of the internal shear strain γ_b, γ_u, γ_c and γ_h is related to the half-height of the brick unit (head joint), the thickness of the joints and the external shear strain γ_0. With the defined geometrical properties (value of h and t), γ_b, γ_u, γ_c and γ_h then could only be changed by the value of the variable γ_0.

4.1.2. Equilibrium equations of the system

Considering the deformed cell of model 1 and the stresses selected in chapter 3 section 3.2, the system of the basic cell could be drawn as figure 4-2 shown:

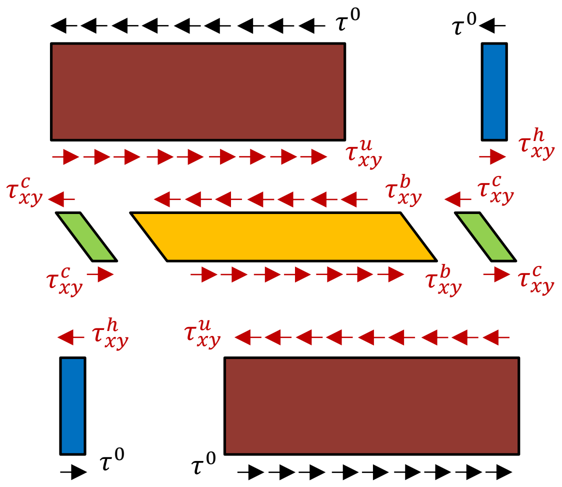


Figure 4-2 Model 1: the internal system of cell

Top boundary condition:

$$\tau^0 \cdot (l + t) = \tau_{xy}^u \cdot l + \tau_{xy}^h \cdot t \tag{4.13}$$

Interface at head & cross joint:

$$\tau_{xy}^h \cdot t = \tau_{xy}^c \cdot t \tag{4.14}$$

Interface at bed joint:

$$\tau_{xy}^u \cdot l = \tau_{xy}^b \cdot (l - t) + \ \tau_{xy}^c \cdot t \tag{4.15}$$

Interface at brick & cross joint:

$$\tau_{xy}^b = \tau_{xy}^c \tag{4.16}$$

Note that the equilibrium equations of the system should be derived at the top boundary and the brick-mortar interfaces. Therefore, equations (4.13) to (4.16) are able to be proposed directly. The function of the internal stress should be satisfied by combining these equations:

$$\tau_{xy}^u = \tau_{xy}^h = \tau_{xy}^c = \tau_{xy}^b = \tau^0 \tag{4.17}$$

Where τ^0 is the external shear stress, τ_{xy}^u and τ_{xy}^b are the shear stresses between the brick and the bed joint, τ_{xy}^c and τ_{xy}^h is the shear stresses between the cross and the head joint.

4.1.3. Constitutive equations

From equation (4.17), the shear stress of each component is equal to each other. Therefore, only one shear stress is needed to be taken into account. τ_{xy}^b, the shear stress between the brick and the bed joint is selected here.

Now, we have 4 unknown strain and stress variables, including the internal shear stress τ_{xy}^b, the external shear stress τ^0, the internal shear strain γ_b and the external shear strain γ_0. That means we need 4 equations in total to solve this system. Now that we have already got equations (4.12) and (4.17), we still need 2 more equations from the constitutive law of the components.

For the shear behaviour at the interface of the brick unit and the bed joint, it is obvious to adopt the relation of the shear strain and stress as:

$$\tau_{xy}^b = G_{xy}^b \gamma_b \tag{4.18}$$

Where G_{xy}^b is the shear modulus of the bed joint.

In this work, displacement control is selected to be the loading approach. Therefore, the value of the horizontal strain ε_0 could be obtained every step. Therefore, the shear strain γ_0 can be computed as:

$$\gamma_0 = 2\varepsilon_0 \tag{4.19}$$

4.1.4. Equilibrium “damage” equations

Considering the damage state evaluated by the shear damage factors calculated following the formulation described in section 4.1, the damaged internal stress should be:

$$\tau_{xy,d}^{b} = (1-d)G_{xy}^{b}\gamma_b \tag{4.20}$$

Where $\tau_{xy,d}^{b}$ is the damaged shear stress between the brick unit and the bed joint.

As the undamaged stress is considered to be the equivalent effective one, equations (4.17) (4.18) should be rederived as:

$$\tau^0 = \tau_{xy,d}^{b} = (1-d)G_{xy}^{b}\gamma_b \tag{4.21}$$

Considering the equation (4.12) (4.18) (4.19) and (4.21), we can propose the “damage” equations for model 1 and find the relations of the unknown variables τ^0, τ_{xy}^{b} and the known ε_0:

$$\tau_{xy}^{b} = 2\frac{h+t}{t}G_{xy}^{b}\varepsilon_0 = \frac{h+t}{t}G_{xy}^{b}\gamma_0 \tag{4.22}$$

$$\tau^0 = 2\frac{h+t}{t}(1-d)G_{xy}^{b}\varepsilon_0 = \frac{h+t}{t}(1-d)G_{xy}^{b}\gamma_0 \tag{4.23}$$

The damage factor d should be calculated according to the damage evaluation law in section 4.1 by τ_{xy}^{b}. Note that the tangent stiffness of model 1 should be:

$$K = \frac{\partial\,\tau^0}{\partial\,\gamma_0} = \frac{h+t}{t}(1-d)G_{xy}^{b} \tag{4.24}$$

4.2. Algorithm

The very simple algorithm that can be used to evaluate the stress of the proposed model has been introduced above as [31] shown. That can be concluded as following steps:

Initial date from time $\boldsymbol{t+1}$: [31]
Material properties: second fracture energy G^{II}, shear modulus G, shear strength σ_s, element size H;
Geometrical properties: half-height of brick unit h, the thickness of mortar t
Current values: internal shear stress τ_t, damage state variable d_t, external shear stress $\varepsilon_{0,\mathrm{t}}$
The boundary condition of the cell here is:

$$\varepsilon_{0,\mathrm{t+1}} = \varepsilon_{0,\mathrm{t}} + \Delta\varepsilon_0 \tag{4.25}$$

Where $\varepsilon_{0,\mathrm{t+1}}$ is the external shear strain at $t = t+1$, $\Delta\varepsilon_0$ is the shear strain increment.

At $t = 0$: initializing the external shear strain and the shear damage factor as $\varepsilon_0 = 0$, $d = 0$.

(1) Determining A_s from eq. (4.8) by known material properties applied in Zucchini’s work [20], see table 4-1,
(2) At $t = t+1$, evaluating the undamaged shear stress $\tau_{xy,t+1}^{b}$ between the brick unit and the bed joint from eq. (4.22) by known $\varepsilon_{0,\mathrm{t+1}}$;
(3) Updating the internal variables τ_{t+1} as eq. (4.4) and d_{t+1} as eq. (4.5) shown;
(4) Updating stresses τ_{t+1}^{0} from eq. (4.23)

5. Model 2: horizontal tension behaviour

In this model, the tension behaviour of the basic cell is considered. Based on the deformed cell assumed in chapter 3 section 3.3, the shear stress between the bed joint and the brick unit and the tension behaviour of each component should be considered. Therefore, both shear and tension damage state variables should be included, which are damage factors using for evaluating damage status caused by internal tension and shear stresses.

5.1. Derive "damage" equations

Formulations for fracture energy in shear has been introduced in chapter 4 section 4.1. The explicit function in tension could be similarly adopted as [20]:

$$\boldsymbol{\sigma}^{\boldsymbol{i}}_{\boldsymbol{xx,d}} = \left(1-d^{i}_{x}\right)\boldsymbol{D}^{\boldsymbol{i}}\boldsymbol{\varepsilon}^{\boldsymbol{i}}_{\boldsymbol{xx}} = \left(1-d^{i}_{t}\right)\boldsymbol{\sigma}^{\boldsymbol{i}}_{\boldsymbol{xx}}, \qquad i = b,u,h,c \tag{5.1}$$

$$\sigma^{i}_{xx,max} = \sigma^{i}_{t} \tag{5.2}$$

$$\sigma^{i} = \max\left\{\sigma^{i}_{xx}, \sigma^{i}_{t}\right\}, \;\; i = b,u,h,c \tag{5.3}$$

$$d^{i}_{t} = 1 - \frac{\sigma^{i}_{t}}{\sigma^{i}}\exp\left[A^{i}_{t}\left(1-\frac{\sigma^{i}}{\sigma^{i}_{t}}\right)\right], \;\; i = b,u,h,c \tag{5.4}$$

Where $\boldsymbol{\sigma}^{\boldsymbol{i}}_{\boldsymbol{xx}}$ is undamaged stress tensor of a given cell component which can be expressed by external strain $\varepsilon_{xx,0}$ with equations listed in section 5.1 below. $\boldsymbol{\sigma}^{\boldsymbol{i}}_{\boldsymbol{xx,d}}$ is damaged stress tensor of each component evaluated by damage factor d^{i}_{t}. The scalar value of maximum tensile stress is the tension strength of each component σ^{i}_{t} while equivalent stress of given component σ^{i} should be a larger value between undamaged stress at the current step and its maximum value.

Parameter A^{i}_{t} could also be related to special fracture energy in uniaxial tension g^{I} (N/mm^{2}) by integration of the deformation energy on full strain path [20]:

$$g^{I} = \frac{\sigma^{2}_{t}}{E}\left(\frac{1}{2}+\frac{1}{A_{t}}\right) \tag{5.5}$$

Therefore, parameter A_{t} of damage model 2 can be generally proposed as:

$$A_{t} = \left(\frac{G^{I}E}{l_{t}\sigma^{2}_{t}}-\frac{1}{2}\right)^{-1}, \qquad l_{t} = H \tag{5.6}$$

Where G^{I} is fracture energy in model I, σ_{t} is tension strength and characteristic length l_{t} of the smeared crack model should be element size H. Formulation of parameter A_{t} from equation (5.6) is able to be applied in components in the basic cell.

5.1.1. Kinematic relation

In this model, the stress-strain curve for tension behaviour of the basic cell is discussed. Therefore, externally horizontal strain and stress are taken into account and strain ε^{0}_{xx} is considered as the known variable with displacement control method for iterative processing. The behaviour of the unit cell under externally vertical displacement will not be considered in this model, which will be discussed in chapter 6.

According to the deformed cell of model 2 assumed in chapter 3 section 3.3, kinematic relation can be derived as following equations based on Zucchini’s work in 2002 [30].

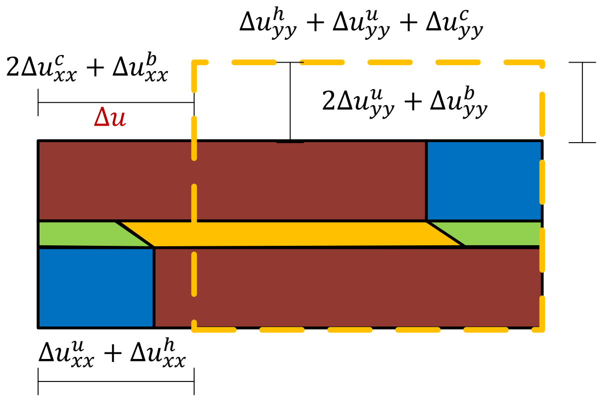

$\Delta u_{yy}^{h}+\Delta u_{yy}^{u}+\Delta u_{yy}^{c}$
$2\Delta u_{xx}^{c}+\Delta u_{xx}^{b}$
Δu
$2\Delta u_{yy}^{u}+\Delta u_{yy}^{b}$
$\Delta u_{xx}^{u}+\Delta u_{xx}^{h}$

Figure 5-1 Deformed cell of model 2

🟥 brick unit 🟨 bed joint 🟩 cross joint 🟦 head joint

Note: “0” means external force, “u” means brick unit (red), “h” means head joint (blue), “c” means cross joint (green), “b” means bed joint (orange)

Where Δu_{xx}^{i} and Δu_{yy}^{i} $(i=u,h,b,c)$ are horizontal and vertical displacement of a given component, Δu is the total displacement of the homogenized cell. Note that the relations between those displacements can be summarized as:

$$\Delta u_{xx}^{u}+\Delta u_{xx}^{h}=2\Delta u_{xx}^{c}+\Delta u_{xx}^{b}=\Delta u \tag{5.7}$$

$$2\Delta u_{yy}^{u}+\Delta u_{yy}^{b}=\Delta u_{yy}^{u}+\Delta u_{yy}^{c}+\Delta u_{yy}^{h} \tag{5.8}$$

The definition of each strain as following:

$$\begin{bmatrix}\Delta u_{xx}^{u}\\ \Delta u_{xx}^{h}\\ \Delta u_{xx}^{c}\\ \Delta u_{xx}^{b}\end{bmatrix}=\begin{bmatrix}l&0&0&0\\0&t&0&0\\0&0&t&0\\0&0&0&l-t\end{bmatrix}\begin{bmatrix}\varepsilon_{xx}^{u}\\ \varepsilon_{xx}^{h}\\ \varepsilon_{xx}^{c}\\ \varepsilon_{xx}^{b}\end{bmatrix},\qquad \begin{bmatrix}\Delta u_{yy}^{u}\\ \Delta u_{yy}^{h}\\ \Delta u_{yy}^{c}\\ \Delta u_{yy}^{b}\end{bmatrix}=\begin{bmatrix}h&0&0&0\\0&h&0&0\\0&0&2t&0\\0&0&0&2t\end{bmatrix}\begin{bmatrix}\varepsilon_{yy}^{u}\\ \varepsilon_{yy}^{h}\\ \varepsilon_{yy}^{c}\\ \varepsilon_{yy}^{b}\end{bmatrix},\qquad \Delta u=\varepsilon_{xx}^{0}\cdot(l+t) \tag{5.9}$$

Where ε_{xx}^{i} and ε_{yy}^{i} $(i=u,h,b,c)$ are the horizontal and vertical strain of each component, ε_{xx}^{0} is homogenized tension strain of the basic cell. The length of the brick unit is $2l$, the thickness of mortar is $2t$, and the heights of the brick unit as well as the head joint are $2h$.

Therefore, eq. (5.7) and (5.8) could be rewritten as:

$$l\varepsilon_{xx}^{u}+t\varepsilon_{xx}^{h}=2t\varepsilon_{xx}^{c}+(l-t)\varepsilon_{xx}^{b} \tag{5.10}$$

$$(l+t)\varepsilon_{xx}^{0}=2t\varepsilon_{xx}^{c}+(l-t)\,\varepsilon_{xx}^{b} \tag{5.11}$$

$$h\varepsilon_{yy}^{u}+2t\varepsilon_{yy}^{b}=2t\varepsilon_{yy}^{c}+h\varepsilon_{yy}^{h} \tag{5.12}$$

The 9 variables shown in kinematic relation include: unknown variables ε_{xx}^{i} and ε_{yy}^{i} $(i = u, h, b, c)$, known variables ε_{xx}^{0}.

5.1.2. Equilibrium equations of the system

Based on assumptions of the deformed cell and selections of stresses in chapter 3 section 3.3, the distribution of stresses of the inside system of model 2 can be drawn as figure 5-2 shown. According to Zucchini's work in 2004 [30], equilibrium equations of the system could be derived at boundary conditions and interfaces of the brick unit and joint mortar.

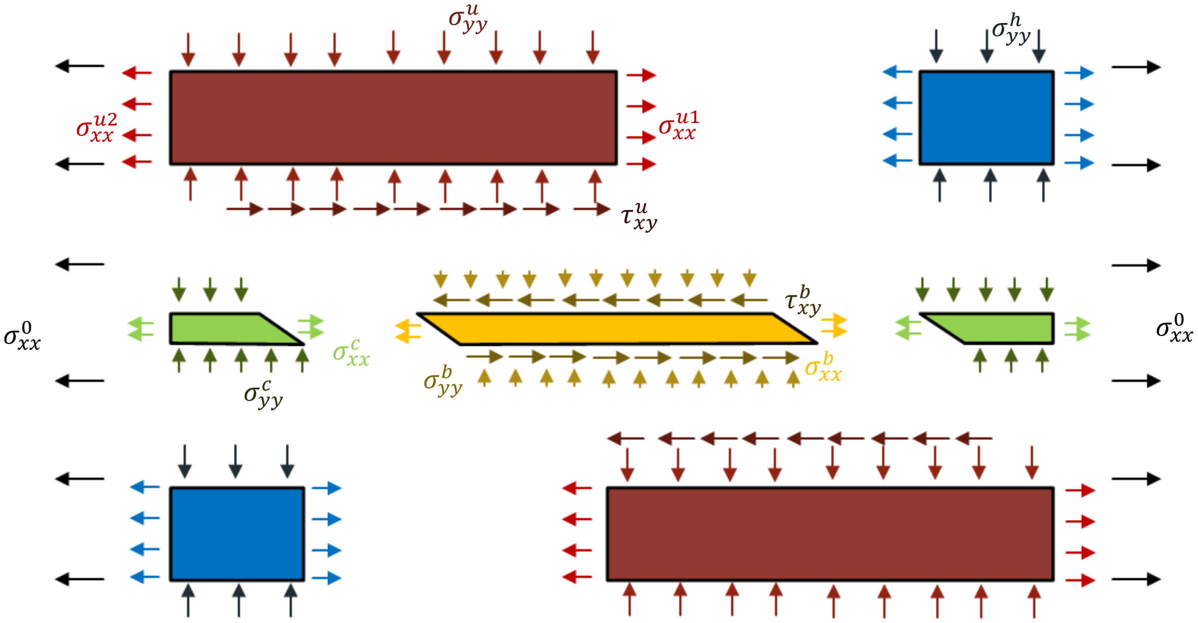


Figure 5-2 Model 2: the internal system of cell

Where σ_{xx}^{i} and σ_{yy}^{i} $(i = u, h, b, c)$ are internally horizontal and vertical stresses of each component, σ_{xx}^{0} is tension stress of the basic cell. Horizontal stresses of the brick unit are different at two sides (σ_{xx}^{u1} at the interface of brick and head joint and σ_{xx}^{u2} at side bounder) as a result of shear behaviour between brick unit and bed joint, see figure 5-3.

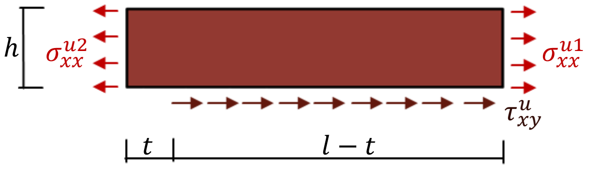


Figure 5-3 Horizontal stresses of the brick unit

Let us introduce the average horizontal stress of the brick unit $\bar{\sigma}_{xx}^{u} = (\sigma_{xx}^{u1} + \sigma_{xx}^{u2})/2$ to simplify the equations.

$$h\sigma_{xx}^{u2} - (l - t)\tau_{xy}^{u} = h\sigma_{xx}^{u1} \tag{5.13}$$

Therefore, the functions of σ_{xx}^{u1} and σ_{xx}^{u2} should be:

$$\sigma_{xx}^{u1} = \bar{\sigma}_{xx}^{u} - \frac{l-t}{h}\tau_{xy}^{b}, \sigma_{xx}^{u2} = \bar{\sigma}_{xx}^{u} + \frac{l-t}{h}\tau_{xy}^{b} \tag{5.14}$$

Shear stress between the brick unit and bed joint $\tau_{xy}^{u} = \tau_{xy}^{b}$ is assumed as linearly increasing along the length direction. Therefore, we can use $\bar{\sigma}_{xx}^{u}$ and τ_{xy}^{b} to derive the function of σ_{xx}^{u1} and σ_{xx}^{u2} above shown.

Note that equilibrium equations of the system can be derived at upper and right boundary conditions as:

1. Upper boundary condition: $l\sigma_{yy}^{u} + t\sigma_{yy}^{h} = 0$ (5.15)
2. Right boundary condition: $h\sigma_{xx}^{h} + 2t\sigma_{xx}^{c} + h\left(\bar{\sigma}_{xx}^{u} + \tau_{xy}^{b}\frac{l-t}{2h}\right) = 2(h+t)\sigma_{xx}^{0}$ (5.16)

At the interface of the brick unit and joint mortar:

3. Interface brick-head joint:$\sigma_{xx}^{h} = \bar{\sigma}_{xx}^{u} - \tau_{xy}^{b}\frac{l-t}{2h}$ (5.17)
4. Interface brick-bed joint: $\sigma_{yy}^{u} = \sigma_{yy}^{b}$ (5.18)

The 9 unknown variables considered in equilibrium equations can be concluded as: σ_{xx}^{i} and σ_{yy}^{i} $(i = u, h, b)$, σ_{xx}^{c}, τ_{xy}^{b} and σ_{xx}^{0}.

5.1.3. Constitutive equations

Bed joint, head joint and brick unit are considered to be elastic materials. Damage variables will evaluate their damage status. Therefore, constitutive equations of those components are:

$$\varepsilon_{xx}^{i} = \frac{1}{E_i}\left[\sigma_{xx}^{i} - \nu_i\sigma_{yy}^{i}\right], \qquad i = b, h, u \tag{5.19}$$

$$\varepsilon_{yy}^{i} = \frac{1}{E_i}\left[\sigma_{yy}^{i} - \nu_i\sigma_{xx}^{i}\right], \qquad i = b, h, u \tag{5.20}$$

Relation of internal shear strain and stress can be derived as:

$$\tau_{xy}^{b} = 2G_b\varepsilon_{xy}^{b} \tag{5.21}$$

Now we have 8 unknown variables in section 5.1.1, 9 unknown variables in section 5.1.2 and 1 unknown variable ε_{xy}^{b} in this section. Therefore, we need 18 equations in total to solve those unknown variables by the function of the known variable ε_{xx}^{0}. We already have eq. (5.10) to (5.12), (5.15) to (5.21), with 14 equations in total. Then, 4 more equations are still needed to be found.

Note that the shear deformation at the interface of the bed joint and the brick unit could be obtained from Zucchini’s work in 2002 [30], see figure 5-4. Then, the shear strain of the bed joint could be derived as:

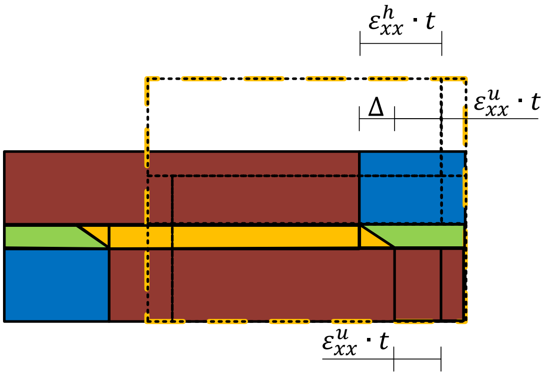

$\varepsilon_{xx}^{h} \cdot t$
Δ
$\varepsilon_{xx}^{u} \cdot t$
$\varepsilon_{xx}^{u} \cdot t$

Figure 5-4 Shear deformation of model 2

$$\Delta \cong t\varepsilon_{xx}^{h} - t\varepsilon_{xx}^{u} \tag{5.22}$$

$$\Delta = 2t\,\gamma_{xy}^{b} = 4t\varepsilon_{xy}^{b} \tag{5.23}$$

$$\varepsilon_{xy}^{b} \cong \frac{\varepsilon_{xx}^{h} - \bar{\varepsilon}_{xx}^{u}}{4} \tag{5.24}$$

According to Zucchini's work in [30], the stress-strain state in the cross joint does not influence the result significantly due to its small volume ratio. Therefore, let us just assume cross and bed joint behaving as horizontal spring to simplify the model when considering strain and stress of cross joint. As a result, eq. (5.25) and (5.26) could be derived at the interface of bed and cross joint as:

$$\sigma_{xx}^{c} = \sigma_{xx}^{b} \tag{5.25}$$

$$\varepsilon_{xx}^{c} = \frac{E_b}{E_c}\varepsilon_{xx}^{b} \tag{5.26}$$

Note that the vertical strain of the cross and bed joint should be the same from figure 5-4.

$$\varepsilon_{yy}^{c} = \varepsilon_{yy}^{b} \tag{5.27}$$

5.1.4. Equilibrium "damage" equations

In this part, the damage status of each component will be evaluated by damage factor d and stresses of each component will be replaced by damaged ones. Therefore, the young's modulus and shear modulus of the given component should be:

$$E_d = (1 - d_t)E_0, \qquad G_d = (1 - d_s)G_0 \tag{5.28}$$

Where E_0 and G_0 are the initial value of elastic and shear modulus, d_t and d_s are tension and shear damage factors. E_d and G_d are damaged elastic and shear modulus.

Damaged stresses of components will be changed as:

$$\sigma_{t,d} = (1 - d_t)\sigma_t, \qquad \tau_d = (1 - d_s)\tau \tag{5.29}$$

Where σ_t and τ are undamaged tension and shear stresses of components at the current time, $\sigma_{t,d}$ and τ_d are damaged ones. Let us assume that the damage status of the bed joint will be evaluated by shear behaviour at the brick-mortar interface, while that of the brick unit, head joint and cross joint will be evaluated by their tension behaviour.

Parameters $r_i = 1 - d_i \ (i = u, h, c, b)$ are introduced here to simplify the equations by introducing them into eq. (5.27) and (5.28) in eq. (5.14) (5.15) (5.16) (5.17) (5.24) (5.25) , those equations are able to be rederived as:

$$\sigma_{xx}^{h} r_h = \bar{\sigma}_{xx}^{u} r_u - \frac{l-t}{2h}\tau_{xy}^{b} r_b \tag{5.30}$$

$$\sigma_{yy}^{u} r_u = \sigma_{yy}^{b} r_b \tag{5.31}$$

$$h\sigma_{xx}^{h} r_h + 2t\sigma_{xx}^{c} r_c + h\left(\bar{\sigma}_{xx}^{u} r_u + \frac{l-t}{2h}\tau_{xy}^{b} r_b\right) = 2(h+t)\sigma_{xx}^{0} \tag{5.32}$$

$$l\sigma_{yy}^{u} r_u + t\sigma_{yy}^{h} r_h = 0 \tag{5.33}$$

$$\varepsilon_{xx}^{c} = \frac{r_b}{r_c}\frac{E_b}{E_c}\varepsilon_{xx}^{b} \tag{5.34}$$

$$\sigma_{xx}^{c} r_c = \sigma_{xx}^{b} r_b \tag{5.35}$$

Once damage factors of components are known, 18 unknown stresses, including 17 internal stresses of components and 1 external stress of the basic cell, could be obtained with those 18 “damage” equations.

5.2. Algorithm

Using the algorithm shown in figure 5-5 given by Zucchini in 2004, the micro-mechanical model of the internal system of damaged masonry cell will be coupled with the isotropic scalar damage model of its components [20].

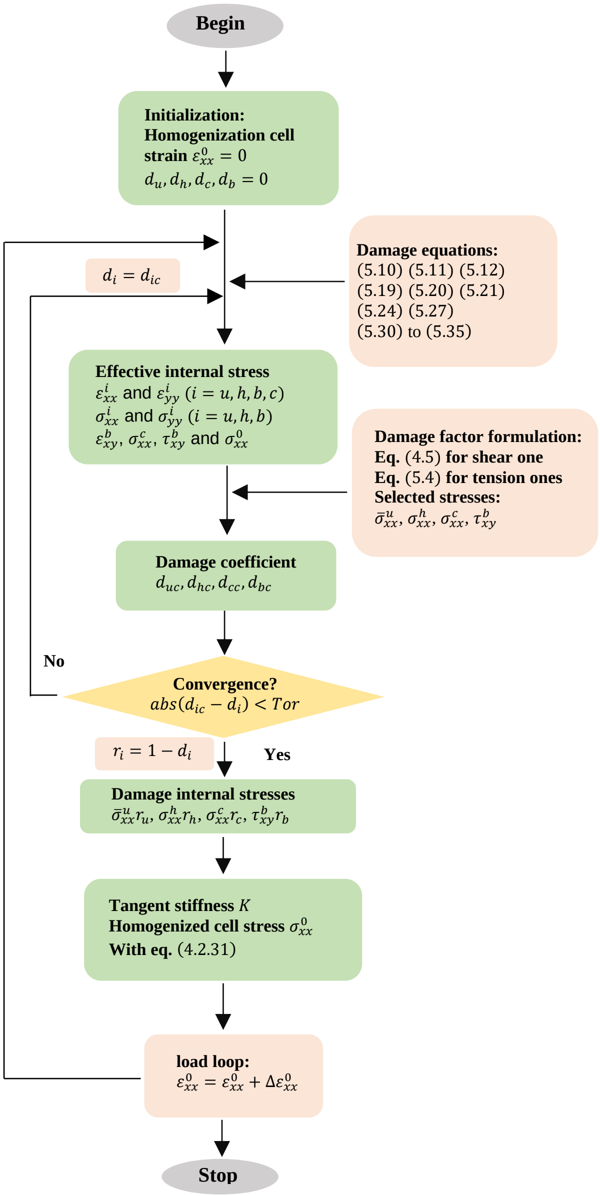

Begin
Initialization:
Homogenization cell
strain $\varepsilon_{xx}^{0} = 0$
$d_u, d_h, d_c, d_b = 0$
Damage equations:
(5.10) (5.11) (5.12)
(5.19) (5.20) (5.21)
(5.24) (5.27)
(5.30) to (5.35)
$d_i = d_{ic}$
Effective internal stress
ε_{xx}^{i} and ε_{yy}^{i} $(i = u, h, b, c)$
σ_{xx}^{i} and σ_{yy}^{i} $(i = u, h, b)$
ε_{xy}^{b}, σ_{xx}^{c}, τ_{xy}^{b} and σ_{xx}^{0}
Damage factor formulation:
Eq. (4.5) for shear one
Eq. (5.4) for tension ones
Selected stresses:
$\bar{\sigma}_{xx}^{u}, \sigma_{xx}^{h}, \sigma_{xx}^{c}, \tau_{xy}^{b}$
Damage coefficient
$d_{uc}, d_{hc}, d_{cc}, d_{bc}$
No
Convergence?
$abs(d_{ic} - d_i) < Tor$
$r_i = 1 - d_i$
Yes
Damage internal stresses
$\bar{\sigma}_{xx}^{u} r_u, \sigma_{xx}^{h} r_h, \sigma_{xx}^{c} r_c, \tau_{xy}^{b} r_b$
Tangent stiffness K
Homogenized cell stress σ_{xx}^{0}
With eq. (4.2.31)
load loop:
$\varepsilon_{xx}^{0} = \varepsilon_{xx}^{0} + \Delta\varepsilon_{xx}^{0}$
Stop

Figure 5-5 Formulation of coupled material model 2 with an iterative algorithm

The outer loop is related to the so-called strain driven problem, known as incremental loading steps. In which, boundary conditions of the basic cell are shown:

$$\varepsilon_{xx,t+1}^{0} = \varepsilon_{xx,t}^{0} + \Delta\varepsilon_{xx}^{0} \tag{5.36}$$

$$\sigma_{yy}^{0} = 0 \tag{5.37}$$

Where $\varepsilon_{xx,t+1}^{0}$ and $\varepsilon_{xx,t}^{0}$ is normal cell strain at time t and $t+1$, $\Delta\varepsilon_{xx}^{0}$ is strain increment. σ_{yy}^{0} is vertical cell stress.

The inner loop is considered an iterative procedure. Values of damage factors are first initialized to be zero at $t = 0$. Then effective internal stresses can be expressed by normal cell strain ε_{xx}^{0} by solving "damage" equilibrium equations of internal structure with the assumed damage factors. To verify values of damage coefficients, damage factors are calculated by components' stresses according to formulation laid out in chapter 4 section 4.1 and chapter 5 section 5.1 again. Damage variables are updated once the difference between calculated and assumed values is larger than the assumed tolerance. This cycle is continually operated every incremental step.

Damaged internal stress σ_{xx}^{0} and tangent stiffness K are finally decided by the value of normal cell strain and damage variables within tolerance, where the function of tangent stiffness of model should be:

$$K = \frac{\partial\, \sigma_{xx}^{0}}{\partial\, \varepsilon_{xx}^{0}} = f(\varepsilon_{xx}^{0}, d_i), i = h, u, c, b \tag{5.38}$$

Note that function of normal stress σ_{xx}^{0} could be proposed with known material and geometrical properties of cell components in the format as:

$$\sigma_{xx}^{0} = f_{sigxx0}(r_u, r_h, r_c, r_b, \varepsilon_{xx}^{0}) \tag{5.39}$$

Here, components are supposed to be damaged by tension behaviour and only one vertical crack in each component is allowed to occur.

6. Model 3: vertical compression behaviour

In this section, the homogenized constitutive relation of vertical compressive behaviour of masonry structures is studied based on Zucchini's research in 2007 [32] and Lourenço's research in 2006 [35]. It is proposed from two main parts: the damage model and elastoplastic theory.

According to the assumptions introduced in chapter 3 section 3.4.1, the damage model is first derived by a similar approach as models 1 and 2 did. And then, plastic deformations of components are proposed according to the theory described in chapter 3 section 3.4.2. The formulation of a combination of the damage model and the plastic deformations is then derived to obtain the final constitutive model.

6.1. Damage model

Formulations of shear and tension damage coefficients are the same as previous formulations stated, see equations (4.8) and (5.6), following the exponential softening processes. In this model, the brick and head joints fail in their equivalent tensile caused by compressive splitting effects. The bed joint is damaged in shear, while the damage status of the cross joint depends on the status of the head joint and the bed joint.

6.1.1. Kinematic relation

The homogenized constitutive law of the basic cell under pure vertical compression loading is studied in this section. Therefore, the vertical strain and the corresponding stress in the vertical compression direction are set in homogeneously distributions. According to the assumptions described in chapter 3 section 3.4.1, the deformed cell of model 3 could be seen from figure 6-1.

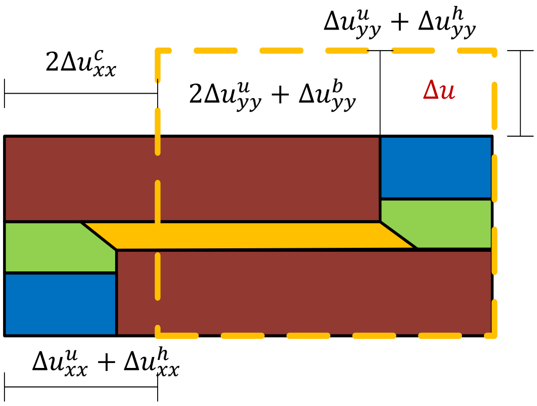

$\Delta u_{yy}^{u} + \Delta u_{yy}^{h}$
$2\Delta u_{xx}^{c}$
$2\Delta u_{yy}^{u} + \Delta u_{yy}^{b}$
Δu
$\Delta u_{xx}^{u} + \Delta u_{xx}^{h}$

Figure 6-1 Deforemed cell of model 3

🟥 brick unit 🟧 bed joint 🟩 cross joint 🟦 head joint

Note: "0" means external force, "u" means brick unit (red), "h" means head joint (blue), "c" means cross joint (green), "b" means bed joint (orange)

It can be noted that only the external loading condition of model 3 differs from that of model 2 from the macro (homogenized) level. Therefore only the equations (5.11) in model 2 proposed in chapter 5 section 5.1.1 is changed, see equation (6.1). Other kinematic relations of model 3 are the same as that of model 2.

$$2(h+t)\varepsilon_{yy}^{0} = 2t\varepsilon_{yy}^{c} + h\varepsilon_{yy}^{h} + h\varepsilon_{yy}^{u} \tag{6.1}$$

6.1.2. Equilibrium equations of the system

Distributions of internal stresses of components could be drawn as figure 6-2 shown based on the assumptions made in chapter 3 section 3.4.1. It is noticed that the horizontal stresses of the bed and cross joint should be compressive considering the uniaxial behaviour of masonry structures introduced in chapter 2 section 2.3 [10]. This simulation differs from Zucchini's research in 2007. [32]

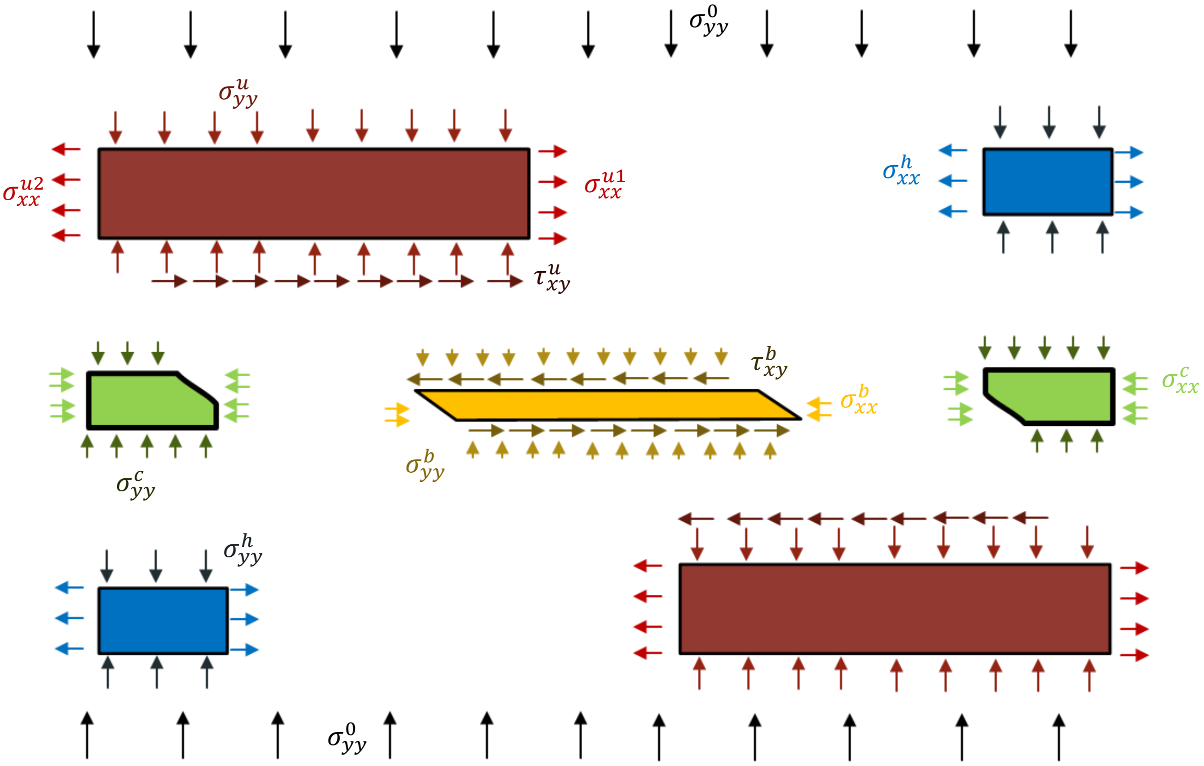


Figure 6-2 Model 3: the internal system of basic cell

As boundary strain is changing from horizontal tensile loading to vertical compressive one, the homogenized stress changes as well and the new definition of it changes from the previous equations (5.32) (5.33) in chapter 5 to (6.2) (6.3)

$$h\sigma_{xx}^{h}r_h - 2t\sigma_{xx}^{c}r_c + h\left(\bar{\sigma}_{xx}^{u}r_u + \frac{l-t}{2h}\tau_{xy}^{b}r_b\right) = 0 \tag{6.2}$$

$$l\sigma_{yy}^{u}r_u + t\sigma_{yy}^{h}r_h = (l+t)\sigma_{yy}^{0} \tag{6.3}$$

6.1.3. Constitutive relation

The brick unit, head and bed joint are still assumed to be elastic materials in the damage model, the constitutive relations of which are the same as the ones proposed in chapter 5, see equations (5.19) to (5.21).

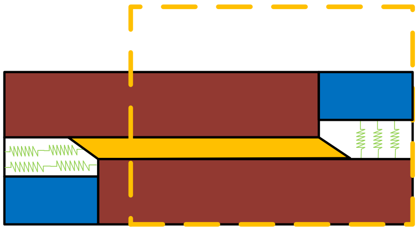


Figure 6-3 simplifications of cross joint

According to Zucchini's research in 2002 [30], the cross joint could be seen as two springs in horizontal and vertical directions rather than only the horizontal one assumed in model 2, see figure 6-3. Therefore, the equation (5.27) could be rewritten as:

$$\varepsilon_{yy}^{c}=\frac{E_h}{E_c}\varepsilon_{yy}^{h} \tag{6.4}$$

6.1.4. Equilibrium "damage" equations

The formulations for the damage variables of the brick unit, head and bed joints are the same as the previous ones, see equation (4.5) in chapter 4 section 4.1 and (5.4) in chapter 5 section 5.1. Based on zucchini's work [20], the value of damage coefficient in the cross joint could be equal to the average value of that in the head d_h and bed joint d_b when the horizontal stress in the cross joint is in compression direction (assumed direction), otherwise this value still is calculated following the formulations proposed in model 2 with the cross joint's tensile stress σ_{xx}^{c} and strength σ_t^c, see equations (6.5) and (6.6).

$$d_c=\begin{cases}\dfrac{d_b+d_h}{2}, & \sigma_{xx}^{c}>0\\ 1-\dfrac{\sigma_t^c}{\sigma^c}\exp\left[A_t^c\left(1-\dfrac{\sigma^c}{\sigma_t^c}\right)\right], & \sigma_{xx}^{c}\leq 0\end{cases} \tag{6.5}$$

$$\sigma^c=\max\{\sigma_{xx}^{c},\sigma_t^c\} \tag{6.6}$$

The internal stresses could be solved by known ε_{yy}^{0} by equations (5.10) $(5.12)(5.19)$ (5.20) (5.21) (5.24) (5.30) (5.31) (5.34) (5.35) from chapter 5 and equations (6.1) to (6.4) from chapter 6. The material and geometrical properties could be obtained from Zucchini's work in 2007 [32], see table 6-1.

Table 6-1 Material and geometrical properties of basic cell [32]

Material properties						
	E (N/mm^2)	ν	σ_t (N/mm^2)	G^I (N/mm^{2}mm)	σ_s (N/mm^2)	G^{II} (N/mm^{2}mm)
Mortar	1178	0.057	0.7	0.35	0.75	0.05
Brick unit	4865	0.094	3.7	1.9	-	-
Geometrical properties						
	h (mm)			l (mm)	t (mm)	
Basic cell	2			12	1	

As a result, the homogenized strain-stress curve can be obtained by implementing the algorithm introduced in chapter 5 section 5.1 figure 5-5 with the changed formulation of the cross-joint's damage variable, see equations (6.5) and (6.6).

This damage model only considers the elastic properties of components, which means all components could only be damaged in shear and tension when shear or tensile cracks occur.

6.2. Elastoplastic phase

Based on the elastoplastic theory introduced in chapter 2 section 2.4, the function of the internal stresses, including the inelastic behaviour, could be calculated by equation (2.35):

$$\boldsymbol{\sigma}_n=\boldsymbol{\sigma}_0+\boldsymbol{D}_e\,\Delta\boldsymbol{\varepsilon}_A+\boldsymbol{D}_e\left(\Delta\boldsymbol{\varepsilon}_B-\frac{\boldsymbol{n}_c^T\boldsymbol{D}_e\Delta\boldsymbol{\varepsilon}_B}{h_c+\boldsymbol{n}_c^T\boldsymbol{D}_e\boldsymbol{m}_c}\boldsymbol{m}_c\right) \tag{2.35}$$

According to figure 2-16 in chapter 2, this equation could be simplified as:

$$\boldsymbol{\sigma}_i = \boldsymbol{D}_{i,e}\left(\boldsymbol{\varepsilon}_i - \sum_{t=m}^{t=t} \frac{\boldsymbol{n}_{i,c}^T \boldsymbol{D}_{i,e} \Delta\boldsymbol{\varepsilon}_i}{h_{i,c} + \boldsymbol{n}_{i,c}^T \boldsymbol{D}_{i,e} \boldsymbol{m}_{i,c}} \boldsymbol{m}_{i,c}\right), \qquad i = u, b, h \tag{6.7}$$

Where subscript "u, b, h" means the brick unit, bed joint and head joint respectively. The inelastic phase begins from load step $t = m$ and $\Delta\boldsymbol{\varepsilon}_i$ is the strain increment of each component at each load step.

$\boldsymbol{\sigma}_i$ is the stress tensor and $\boldsymbol{\varepsilon}_i$ is strain tensor of each component at current step t, see equation (6.8).

$$\boldsymbol{\sigma}_i = \begin{bmatrix} \sigma_{xx}^i \\ \sigma_{yy}^i \\ \tau_{xy}^i \end{bmatrix}, \boldsymbol{\varepsilon}_i = \begin{bmatrix} \varepsilon_{xx}^i \\ \varepsilon_{yy}^i \\ 2\varepsilon_{xy}^i \end{bmatrix} \tag{6.8}$$

$\boldsymbol{D}_{i,e}$ is elastic stiffness matrix and is defined as equation (6.9) shown based on continuum mechanics of plane stress element:

$$\boldsymbol{D}_{i,e} = \frac{E_i}{(1+\nu_i)(1-2\nu_i)} \begin{bmatrix} 1-\nu_i & \nu_i & 0 \\ \nu_i & 1-\nu_i & 0 \\ 0 & 0 & \dfrac{1-2\nu_i}{2} \end{bmatrix} \tag{6.9}$$

Equation (6.7) could also be rewritten as:

$$\boldsymbol{\varepsilon}_i = \boldsymbol{D}_{i,e}{}^{-1}\boldsymbol{\sigma}_i + \boldsymbol{\varepsilon}_{i,p}, \qquad i = u, b, h \tag{6.10}$$

The elastic and elastoplastic properties of components in this model could be coupled by equation (6.10) straight forward. The first part in equation (6.10) is contributed by the damage model proposed in chapter 6 section 6.1. The second part is the plastic strain tensor, which could be obtained by the formulations and algorithm introduced in section 6.2.1.

6.2.1. Flow rule

Plastic strain tensor could be computed by equation (6.11) by:

$$\boldsymbol{\varepsilon}_{i,p} = \sum_{t=m}^{t=t} \Delta\lambda_{i,c}\, \boldsymbol{m}_{i,c} = \sum_{t=m}^{t=t} \frac{\boldsymbol{n}_{i,c}^T \boldsymbol{D}_{i,e} \Delta\boldsymbol{\varepsilon}_i}{h_{i,c} + \boldsymbol{n}_{i,c}^T \boldsymbol{D}_{i,e} \boldsymbol{m}_{i,c}}\, \boldsymbol{m}_{i,c} \tag{6.11}$$

Where $\Delta\lambda_{i,c}$ is the plastic multiplier, $\boldsymbol{n}_{i,c}^T$ and $\boldsymbol{m}_{i,c}$ are the direction of yield function and protentional energy of each component at critical stress point located on the yield surface:

$$\boldsymbol{n}_{i,c} = \begin{bmatrix} \left.\dfrac{\partial f_i}{\partial \sigma_{xx}^i}\right|_c \\ \left.\dfrac{\partial f_i}{\partial \sigma_{yy}^i}\right|_c \\ \left.\dfrac{\partial f_i}{\partial \tau_{xy}^i}\right|_c \end{bmatrix}, \boldsymbol{m}_{i,c} = \begin{bmatrix} \left.\dfrac{\partial g_i}{\partial \sigma_{xx}^i}\right|_c \\ \left.\dfrac{\partial g_i}{\partial \sigma_{yy}^i}\right|_c \\ \left.\dfrac{\partial g_i}{\partial \tau_{xy}^i}\right|_c \end{bmatrix} \tag{6.12}$$

The function of Drucker-Prager yield criteria introduced in chapter 2 section 2.4.2 equation (2.9) and potential

energy in the form of equation (2.22) in chapter 2 section 2.4.2 could be rewritten as equations (6.13) and (6.14) shown when they are applied on the 4-node plane element:

$$f_i = \sqrt{{\sigma_{xx}^i}^2 + {\sigma_{yy}^i}^2 + 3{\tau_{xy}^i}^2 - \sigma_{xx}^i \sigma_{yy}^i} + \frac{2\sin\phi_i}{3-\sin\phi_i}\left(\sigma_{xx}^i + \sigma_{yy}^i\right) - \frac{6\, c_i \sin\phi_i}{3-\sin\phi_i} \tag{6.13}$$

$$g_i = \sqrt{{\sigma_{xx}^i}^2 + {\sigma_{yy}^i}^2 + 3{\tau_{xy}^i}^2 - \sigma_{xx}^i \sigma_{yy}^i} + \frac{3\sin\psi_i}{3-\sin\psi_i}\left(\sigma_{xx}^i + \sigma_{yy}^i\right) - \frac{6\, c_i \sin\psi_i}{3-\sin\psi_i} \tag{6.14}$$

Critical stress point of the given component could be obtained from the cross points of the elastic line and the yield surface f_i in stress space. It can be assumed that we have an elastic stress point $(\sigma_{xx}^{ie}, \sigma_{yy}^{ie}, \tau_{xy}^{ie})$ from the damage model proposed in chapter 6 section 6.1 at the current load step, then the value of critical stress point $(\sigma_{xx}^{ic}, \sigma_{yy}^{ic}, \tau_{xy}^{ic})$ could be found by an equation system:

$$\begin{cases} \sigma_{xx}^{ie} \cdot \sigma_{yy}^{i} = \sigma_{yy}^{ie} \cdot \sigma_{xx}^{i} \\ \tau_{xy}^{ie} \cdot \sigma_{yy}^{i} = \sigma_{yy}^{ie} \cdot \tau_{xy}^{i} \\ f_i = 0 \end{cases} \tag{6.17}$$

Value of hardening modulus $h_{i,c}$, defined in chapter 3 section 3.4 equation (3.1), depends on:

(1) Plastic multiplier $\Delta\lambda_{i,c}$;

(2) The slope of hardening diagram at critical stress point $\left.\frac{\partial f_i}{\partial \kappa_i}\right|_c$, see chapter 3 equations (3.5) (3.6);

(3) Hardening parameter $\Delta\kappa_i$:

$$\Delta\kappa_i = \sqrt{\frac{2}{3}\left[\left(\Delta\varepsilon_{xx}^{ip}\right)^2 + \left(\Delta\varepsilon_{yy}^{ip}\right)^2 + \left(\Delta\varepsilon_{xy}^{ip}\right)^2\right]} \tag{6.18}$$

$$\Delta\boldsymbol{\varepsilon}_{i,p} = \Delta\lambda_{i,c}\, \boldsymbol{m}_{i,c} = \begin{bmatrix} \Delta\varepsilon_{xx}^{ip} \\ \Delta\varepsilon_{yy}^{ip} \\ \Delta\varepsilon_{xy}^{ip} \end{bmatrix} \tag{6.19}$$

6.2.2. Yield surface

The material and geometrical properties could be obtained from Zucchini's work [32], as table 6-2 shown.

Table 6-2 material properties from Zucchini in 2007 [32]

Material properties	Brick unit	Mortar	***Geometry***	Brick unit	Mortar
Young's Modulus E (MPa)	4865	1178	Height h (mm) (Half value only)	2	2
Poisson ratio ν	0.094	0.057			
Tension strength σ_t (MPa)	3.7	0.7			
Compression strength σ_c (MPa)	26.9	3.2	Length l (mm) (Half value only)	12	12
I fracture energy G_I (N/mm)	1.9	0.35			
Compressive fracture energy G_c (N/mm)	29.8	6.43			
Shear strength σ_s (MPa)	-	VAR			
II fracture energy G_{II} (N/mm)	-	VAR	Thickness t (mm) (Half value only)	-	1
Friction angle ϕ (°)	10	10			
Dilatancy angle ψ (°)	5	5			

According to the assumption on the selection of internal stresses of components made in chapter 3 section 3.4, shear stresses of the brick unit τ_{xy}^{u} and the head joint τ_{xy}^{h} should be equal to zero.

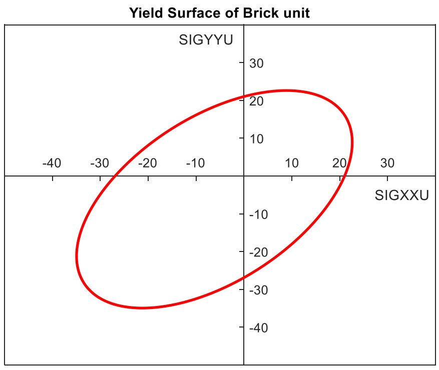


Figure 6-4 yield surface of the brick unit

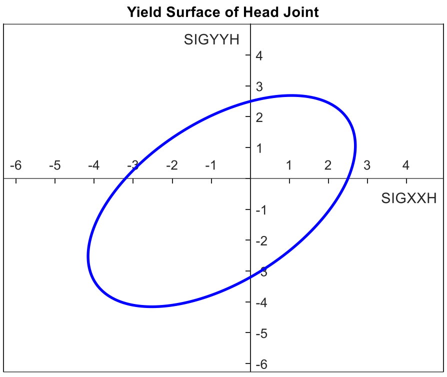


Figure 6-5 yield surface of the head joint

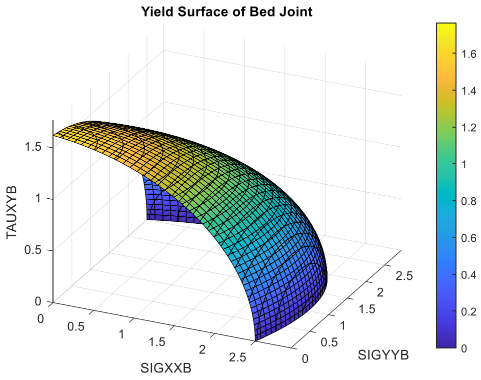


Figure 6-6 representative yield surface of the bed joint

Yield surfaces of the brick unit, head and bed joint could be drawn as figure 6-4 to 6-6 shown.

Notice that all of the yield surfaces are drawn at components' ultimate limit state, and only the bed joint's shear stress is taken into account. The yield surface of the bed joint is a representative surface with only positive values in all directions.

6.3. Algorithm

The formulation of model 3 can be found in figure 6-7. This algorithm could be implemented from a pure brick unit material model to a composited unit cell consisting of the brick units and mortar joints.

Firstly, the homogenized cell can be assumed to consist of only the brick unit, and the elastoplastic behaviour of this cell can be implemented the same as quasi-brittle-like material. In this case, the vertical strain of the brick unit is equal to the homogenously distributed strain ε_{yy}^{0}. The damaged stress of the brick unit σ_{yy}^{u} in vertical direction should be equal to homogenized stress σ_{yy}^{0}, while the horizontal stress of the brick unit σ_{xx}^{u} should be equal the macro stress as $\sigma_{xx}^{u} = \sigma_{xx}^{0} = 0$.

Secondly, the material model considering only under ideal plasticity is implemented by assuming the value of the hardening/softening modulus s_u always to be equal to zero. The cohesion of the brick unit is a constant in this case, and it could be computed by compressive strength f_c and friction angle ϕ, see equation $(2.2b)$ in chapter 2 section 2.4.2. The value of the brick unit's yield function can be computed by the damaged internal stresses (elastic predictors) of the brick unit obtained from the "damage" equilibrium equations once cohesion is obtained. Note that the "damage" equilibrium equations are derived by substituting the damaged internal stresses for the undamaged stresses proposed based on the bricks' elastic properties in the compatibility equations. The damaged stresses are computed according to equation (5.1) shown.

Thirdly, the predicted stress point is indicated by the damaged internal stresses (elastic predictors). It is considered to be located inside or on the yield surface if the yield surface's value is negative or zero, which means the bricks is elastically deforming. Then, the strain increment loop occurs, where the external strain and stress increase linearly.

Fourthly, if the value is positive, the plastic deformation of the basic cell should be considered by introducing a plastic strain tensor of the brick unit $\boldsymbol{\varepsilon}_{i,p}$, with $i = u$ here. To obtain the plastic correctors $\boldsymbol{\varepsilon}_{i,p}$, the critical stress point should be found following equation (6.17) in chapter 6 section 6.2.1 by the stress tensor $\boldsymbol{\sigma}_i$

computed from the damage model, see chapter 6 section 6.1. The directions of the yield surface $\boldsymbol{m}_{i,c}$ together with potential energy $\boldsymbol{n}_{i,c}^{T}$ at this critical stress point should then be calculated following equation (6.12).

Fifthly, the corrected stress $\boldsymbol{\sigma}_{i,p}$ could be calculated by the value of elastic predictors $\boldsymbol{\varepsilon}_{i}$ from the damage model, the plastic correctors $\boldsymbol{\varepsilon}_{i,p}$ and the stiffness matrix $\boldsymbol{D}_{ue}$ of brick unit based on continuum mechanics theory. The damage variable of the brick unit could be defined following the algorithm introduced in chapter 5 section 5.2, see figure 5-5, by the value of $\boldsymbol{\sigma}_{i,p}$. That is the end of the "elastic or elastoplastic phase" loop at the left hand of the algorithm displayed in figure 6-7.

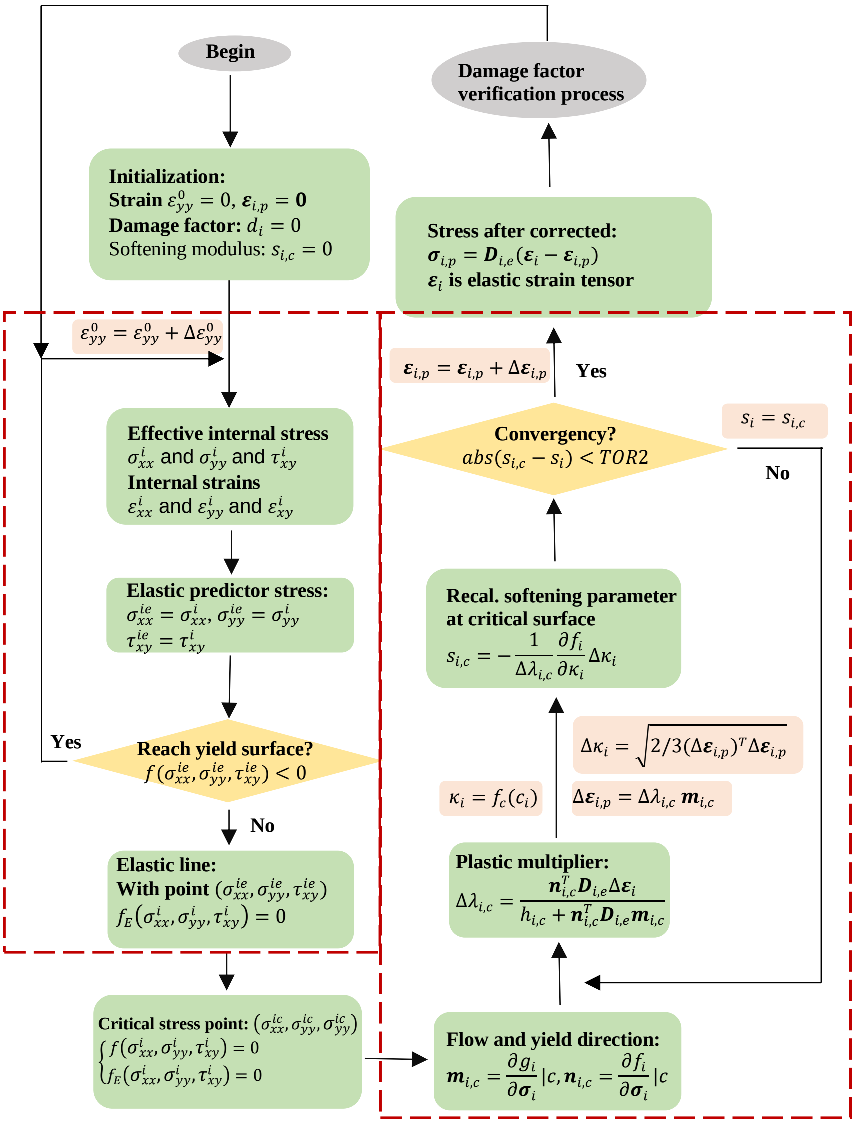

Begin
Damage factor verification process
Initialization:
Strain $\varepsilon_{yy}^{0}=0$, $\boldsymbol{\varepsilon}_{i,p}=\boldsymbol{0}$
Damage factor: $d_i=0$
Softening modulus: $s_{i,c}=0$
Stress after corrected:
$\boldsymbol{\sigma}_{i,p}=\boldsymbol{D}_{i,e}(\boldsymbol{\varepsilon}_i-\boldsymbol{\varepsilon}_{i,p})$
$\boldsymbol{\varepsilon}_i$ is elastic strain tensor
$\varepsilon_{yy}^{0}=\varepsilon_{yy}^{0}+\Delta\varepsilon_{yy}^{0}$
$\boldsymbol{\varepsilon}_{i,p}=\boldsymbol{\varepsilon}_{i,p}+\Delta\boldsymbol{\varepsilon}_{i,p}$
Yes
Convergency?
$abs(s_{i,c}-s_i)<TOR2$
$s_i=s_{i,c}$
No
Effective internal stress
σ_{xx}^{i} and σ_{yy}^{i} and τ_{xy}^{i}
Internal strains
ε_{xx}^{i} and ε_{yy}^{i} and ε_{xy}^{i}
Elastic predictor stress:
$\sigma_{xx}^{ie}=\sigma_{xx}^{i}$, $\sigma_{yy}^{ie}=\sigma_{yy}^{i}$
$\tau_{xy}^{ie}=\tau_{xy}^{i}$
Recal. softening parameter at critical surface
$s_{i,c}=-\frac{1}{\Delta\lambda_{i,c}}\frac{\partial f_i}{\partial\kappa_i}\Delta\kappa_i$
$\Delta\kappa_i=\sqrt{2/3(\Delta\boldsymbol{\varepsilon}_{i,p})^T\Delta\boldsymbol{\varepsilon}_{i,p}}$
Yes
Reach yield surface?
$f(\sigma_{xx}^{ie},\sigma_{yy}^{ie},\tau_{xy}^{ie})<0$
$\kappa_i=f_c(c_i)$
$\Delta\boldsymbol{\varepsilon}_{i,p}=\Delta\lambda_{i,c}\,\boldsymbol{m}_{i,c}$
No
Elastic line:
With point $(\sigma_{xx}^{ie},\sigma_{yy}^{ie},\tau_{xy}^{ie})$
$f_E(\sigma_{xx}^{i},\sigma_{yy}^{i},\tau_{xy}^{i})=0$
Plastic multiplier:
$\Delta\lambda_{i,c}=\frac{\boldsymbol{n}_{i,c}^{T}\boldsymbol{D}_{i,e}\Delta\boldsymbol{\varepsilon}_i}{h_{i,c}+\boldsymbol{n}_{i,c}^{T}\boldsymbol{D}_{i,e}\boldsymbol{m}_{i,c}}$
Critical stress point: $(\sigma_{xx}^{ic},\sigma_{yy}^{ic},\sigma_{yy}^{ic})$
$\begin{cases}f(\sigma_{xx}^{i},\sigma_{yy}^{i},\tau_{xy}^{i})=0\\ f_E(\sigma_{xx}^{i},\sigma_{yy}^{i},\tau_{xy}^{i})=0\end{cases}$
Flow and yield direction:
$\boldsymbol{m}_{i,c}=\frac{\partial g_i}{\partial\boldsymbol{\sigma}_i}|c,\boldsymbol{n}_{i,c}=\frac{\partial f_i}{\partial\boldsymbol{\sigma}_i}|c$

Figure 6-7 Formulation of model 3 with an iterative algorithm

However, the hardening/softening modulus s_u should vary following the hardening or softening processes, and its value could also be relative to the value of cohesion. As s_u could be seen as an input as well as an output in this case. We could use a similar approach as the one introduced in model 2 to find the damage variables, could also be adopted here to find s_u:

(1) Setting the variable as known one with the assumed value to calculate the output coefficient, indicated as the plastic multiplier $\Delta\lambda_{uc}$ here;
(2) Then, re-calculating the variable with the value of output parameter computed in step 1;
(3) If the difference between the assumed and the recalculated values of the variable is smaller than the assumed tolerance, then the assumed value is used in the next step. Otherwise, setting the recalculated one as a new assumed value and repeating steps 1 to 3.

The recalculated value of s_u is defined in equation (3.1) in chapter 3, computed by the plastic multiplier $\Delta\lambda_{uc}$, hardening parameter k (defined in chapter 3) and the slope of the hardening diagram $\alpha f/\alpha k$. According to the theories introduced in chapter 3 and chapter 6, the hardening/softening modulus s_u could be implemented in the algorithm, see the loop located on the right-hand side in figure 6-7. The MATLAB code could be found in Appendix C, leading to results shown in figure 6-8 and 6-9.

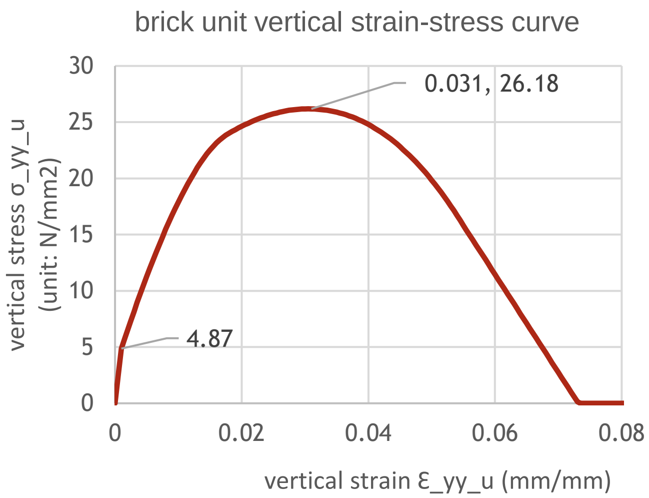


Figure 6-8 strain-stress curve of the brick unit: the vertical stress σ_{yy}^{u} vs, strain ε_{yy}^{u} of the brick unit

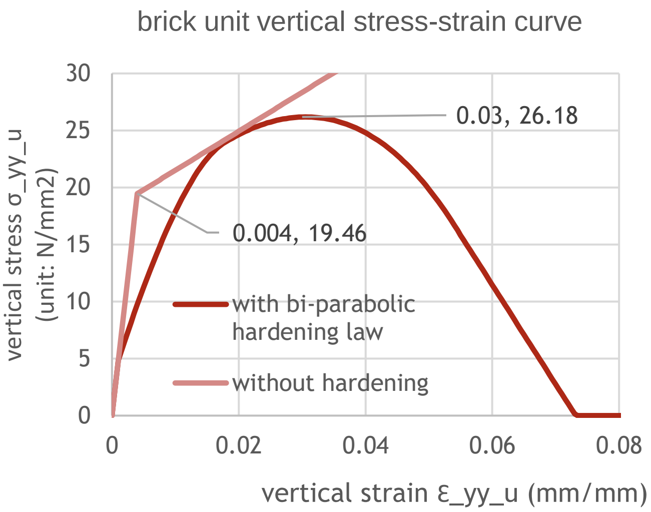


Figure 6-9 differences between ideal and real plasticity: the vertical stress σ_{yy}^{u} vs, strain ε_{yy}^{u} of the brick unit

Finally, applying all of the upper steps in the damage model described in chapter 6 section 6.1. And then, implementing the yield surface of the head and bed joints, see figure 6-5 and 6-6, in each component respectively, with their different compressive strengths, friction and dilatancy angles.

7. Model 4: coupled behaviour

This final model includes all the failure modes of masonry described in section 2.1, namely tension, shear and compression. The mechanical behaviour of masonry is described by means of a newly developed algorithm. Concepts of this chapter are:

(1) Definition of the equations of compatibility of the deformed cell;
(2) Description of the algorithm defined to compute the stresses acting on the cell;
(3) Graphical representation of the stress-strain uniaxial relationships in tension, compression and shear.

7.1. Transverse strains

The transverse strains $\varepsilon_{xx,y}^{0}$ and $\varepsilon_{yy,x}^{0}$ are introduced in chapter 3 section 3.5 in the x- and y-direction, respectively. They are relative to internal strains obtained from models 2 and 3 by equations (7.1) and (7.3):

$$\varepsilon_{xx,2}^{u} \cdot l + \varepsilon_{xx,2}^{h} \cdot t = \varepsilon_{xx,y}^{0} \cdot (l + t) \tag{7.1}$$

$$\varepsilon_{yy,2}^{c} \cdot 2t + \varepsilon_{yy,2}^{h} \cdot h + \varepsilon_{yy,2}^{u} \cdot h = \varepsilon_{yy}^{0} \cdot 2(h + t) \tag{7.2}$$

Where $\varepsilon_{xx,2}^{i}, i = u, h$ are the horizontal strains of the brick unit and the head joint obtained from chapter 5 by adding a new equation (7.2) and $\varepsilon_{yy,2}^{i}, i = u, h, c$ are the vertical strains of the brick unit, the head and the cross joint obtained from chapter 5. The homogenized strain ε_{yy}^{0} is caused by vertical loading.

$$\varepsilon_{yy,3}^{c} \cdot 2t + \varepsilon_{yy,3}^{h} \cdot h + \varepsilon_{yy,3}^{u} \cdot h = \varepsilon_{yy,x}^{0} \cdot 2(h + t) \tag{7.3}$$

$$\varepsilon_{xx,3}^{u} \cdot l + \varepsilon_{xx,3}^{h} \cdot t = \varepsilon_{xx}^{0} \cdot (l + t) \tag{7.4}$$

Similarly, $\varepsilon_{yy,3}^{i}, i = u, h, c$ are the vertical strains of the brick units, the head and the cross joint obtained from chapter 6 and $\varepsilon_{xx,3}^{i}, i = u, h$ are the horizontal strains of the brick unit and the head joint. The macro strain ε_{xx}^{0} is caused by horizontal loading.

7.2. Algorithm

We can distinguish if "diagonal tensile cracking" failure mode, caused by combined shear and vertical compressive loading, occurs or not by determining the magnitude of vertical strain $\varepsilon_{yy,\,total}^{0}$. If the value of $\varepsilon_{yy,\,total}^{0}$ was the positive number or zero, then "shear sliding" failure mode occurs, see figure 2-1 (b) in chapter 2. While "diagonal tensile cracking" failure mode assumed in chapter 3 occurs if the value of $\varepsilon_{yy,\,total}^{0}$ was a negative number, see figure 2-1 (d). Based on these ideas, the formulation of model 4 could be derived as figure 7-1 shown. First, the initial homogenized strains ε_{xx}^{0}, ε_{yy}^{0} and ε_{xy}^{0} together with the damage variables of components for macro constitutive law in shear d_{i}^{xy}, horizontal d_{i}^{x} and vertical direction $d_{i}^{y}(i = u, b, c, h)$ are supposed to be zero. After that, total strains $\varepsilon_{xx,\,total}^{0}$ and $\varepsilon_{yy,\,total}^{0}$ could be computed by equations proposed in chapter 3 and chapter 7. The values of these strains could be positive or negative numbers. Therefore, the strains' magnitudes should be determined by a so-called "in tension?" judgement described in the left loop in figure 7-1. If $\varepsilon_{xx,\,total}^{0} \geq 0$, the horizontal loading is set to be the tensile one. Then, the "horizontal tensile cracking" failure mode proposed in model 2 occurs (see figure 2-1 (c)), and the homogenized constitutive model in horizontal direction could be obtained by implementing the algorithm derived in figure 5-5. Otherwise, the horizontal stress σ_{xx}^{0} is assumed to be zero as the horizontal loading is set to be the compressive one.

If $\varepsilon_{yy,\,total}^{0} \geq 0$, the vertical loading is set to be tensile one. Then, the "joint tensile cracking" failure mode occurs, as figure 2-1 (a) shown in chapter 2. As the vertical tension stresses of the bed joint and other components could be obtained by implementing the algorithm derived in model 3, the damage variables of components could be computed by the exponential relation of damage coefficients and internal stresses (see equation (5.6)). As a result, the constitutive model in the vertical direction could be obtained. Meanwhile, the "shear sliding" failure mode occurs if there is shear loading when $\varepsilon^0_{yy,\,total} \geq 0$, see figure 2-1 (b). In this case, the macro shear stress τ^0_{xy} could be computed by implementing the simple algorithm described in chapter 4. If $\varepsilon^0_{yy,\,total} < 0$, the vertical loading is set to be a compressive one. Then, the "crushing" failure mode introduced in model 3 occurs if there is no shear loading (see figure 2-1 (e)). In this case, the constitutive model in the vertical direction could be built up by implementing the algorithm derived in figure 6-8 chapter 6. Note that the homogenized stress σ^0_{yy} should be negative in this case.

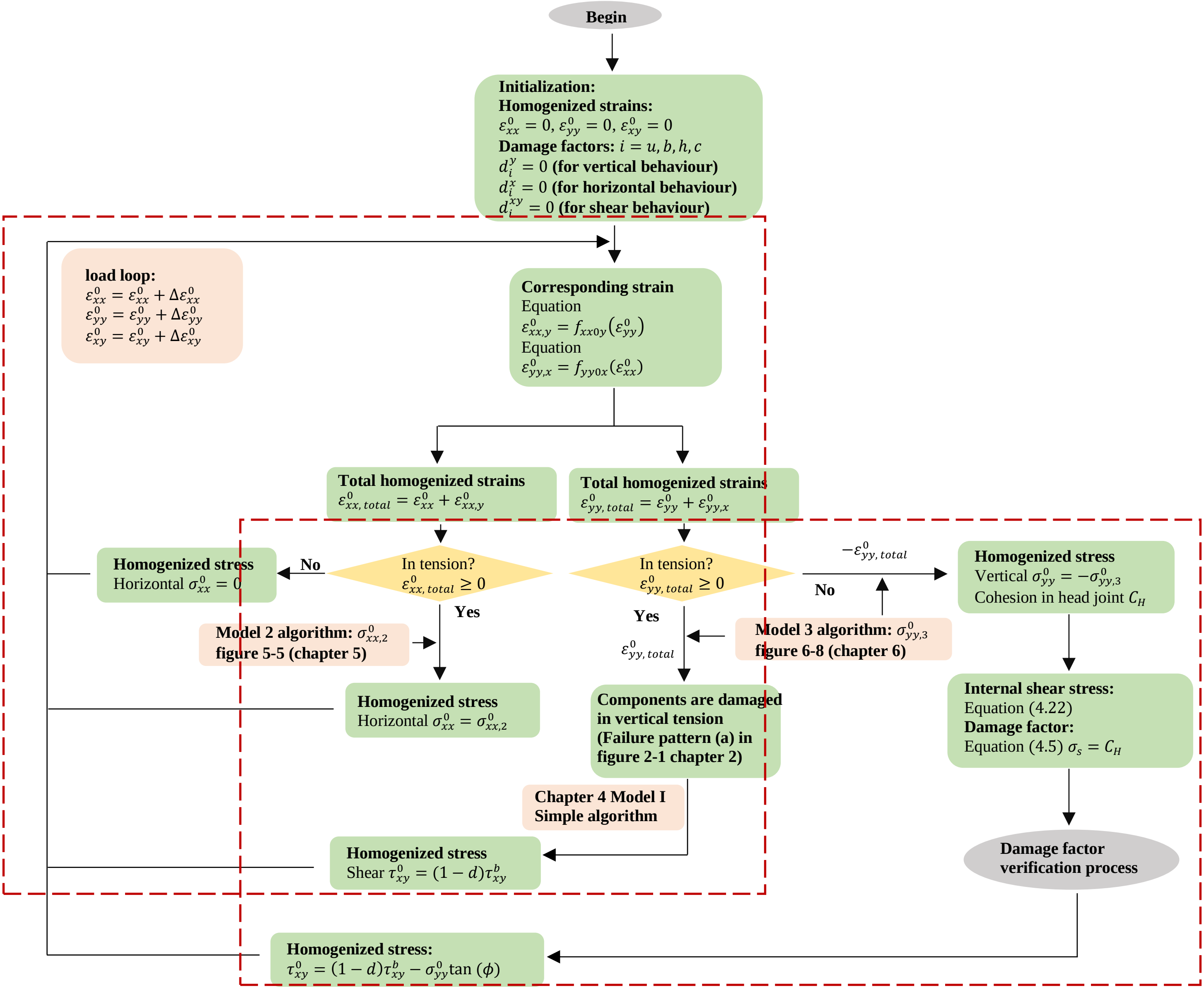

Begin
Initialization:
Homogenized strains:
$\varepsilon^0_{xx} = 0, \varepsilon^0_{yy} = 0, \varepsilon^0_{xy} = 0$
Damage factors: $i = u, b, h, c$
$d^y_i = 0$ (for vertical behaviour)
$d^x_i = 0$ (for horizontal behaviour)
$d^{xy}_i = 0$ (for shear behaviour)
load loop:
$\varepsilon^0_{xx} = \varepsilon^0_{xx} + \Delta\varepsilon^0_{xx}$
$\varepsilon^0_{yy} = \varepsilon^0_{yy} + \Delta\varepsilon^0_{yy}$
$\varepsilon^0_{xy} = \varepsilon^0_{xy} + \Delta\varepsilon^0_{xy}$
Corresponding strain
Equation
$\varepsilon^0_{xx,y} = f_{xx0y}(\varepsilon^0_{yy})$
Equation
$\varepsilon^0_{yy,x} = f_{yy0x}(\varepsilon^0_{xx})$
Total homogenized strains
$\varepsilon^0_{xx,\,total} = \varepsilon^0_{xx} + \varepsilon^0_{xx,y}$
Total homogenized strains
$\varepsilon^0_{yy,\,total} = \varepsilon^0_{yy} + \varepsilon^0_{yy,x}$
Homogenized stress
Horizontal $\sigma^0_{xx} = 0$
No
In tension?
$\varepsilon^0_{xx,\,total} \geq 0$
Yes
In tension?
$\varepsilon^0_{yy,\,total} \geq 0$
Yes
$-\varepsilon^0_{yy,\,total}$
No
Homogenized stress
Vertical $\sigma^0_{yy} = -\sigma^0_{yy,3}$
Cohesion in head joint C_H
Model 2 algorithm: $\sigma^0_{xx,2}$
figure 5-5 (chapter 5)
$\varepsilon^0_{yy,\,total}$
Model 3 algorithm: $\sigma^0_{yy,3}$
figure 6-8 (chapter 6)
Homogenized stress
Horizontal $\sigma^0_{xx} = \sigma^0_{xx,2}$
Components are damaged
in vertical tension
(Failure pattern (a) in
figure 2-1 chapter 2)
Internal shear stress:
Equation (4.22)
Damage factor:
Equation (4.5) $\sigma_s = C_H$
Chapter 4 Model I
Simple algorithm
Homogenized stress
Shear $\tau^0_{xy} = (1-d)\tau^b_{xy}$
Damage factor
verification process
Homogenized stress:
$\tau_{xy} = (1-d)\tau^b_{xy} - \sigma^0_{yy}\tan(\phi)$

Figure 7-1 formulation of model 4 with an iterative algorithm

If there is shear loading, a "diagonal tensile cracking" failure pattern, see figure 2-1 (d), introduced in chapter 3 would occur. In this case, the internal shear stress caused by the external shear strain ε^0_{xy} could be obtained by equation (4.22) in chapter 4. The damage variable relative to this internal stress could be computed by setting the shear strength of the bed joint to be the cohesion of the head joint C_H, see equation (4.5) in chapter 4. The homogenized stress in shear direction τ^0_{xy} could be computed once the damage variable, as well as the internal stress τ^b_{xy}, is obtained by:

$$\tau^0_{xy} = (1-d)\tau^b_{xy} - \sigma^0_{yy}\tan(\phi) \qquad (7.5)$$

The cohesion of the head joint C_H and macro stress σ^0_{yy} could be computed from the previous step.

8. Implementations and comparisons

In this chapter, the material models proposed in chapters 4 to 7 are implemented in MATLAB to find the analytical solutions of the macro constitutive laws. Furthermore, results computed by MATLAB could be used to discover if the algorithms introduced in chapters 4 to 7 and assumptions made in chapter 3 are implemented successfully in the material models.

The validities of the material models stated in chapter 4 for shear behaviour and 5 for horizontal tensile behaviour are then assessed by modelling the single element models in DIANA FEA using user-supplied subroutines. Furthermore, the sensitivity of the material model described in chapter 4 is assessed by changing its material properties.

Finally, the homogenized material model for coupled behaviour introduced in chapter 7 is assessed by comparing the experimental results and the analytical results. The analytical results are obtained by applying material and geometrical properties and the loading condition from the experimental data in model 4.

8.1. Validations of the implementations

In this section, models 1 to 4 are implemented in MATLAB. Furthermore, the validations of the algorithms and the analytical solutions of the macro constitutive laws are discussed.

Furthermore, the single element model is firstly built up in DIANA FEA 10.3. The 4-noded plane stress element with one integration point is selected, and the size is set to be $50 \times 50 - \mathrm{mm}$. Additionally, the homogenized stress-strain curve in shear or horizontal tension is obtained by setting different boundary conditions in this per element model with dcf. File (analysis file, see Appendix F) and dll. File showing the constitutive laws of the material models.

8.1.1. Model 1: shear behaviour

Analytical solution

The parameter A_s introduced in eq. (4.8) in chapter 4 and described in [31] should always be positive. Therefore, the maximum size of the element used in finite element mesh could be obtained by making the function of A_s being equal to zero. The basic unit cell's assumed material and geometrical properties are introduced, as table 8-1 shows.

Table 8-1 Material and geometrical properties of basic cell

	Material properties					
	E (N/mm^2)	ν	σ_t (N/mm^2)	G^I (N/mm^{2}mm)	σ_s (N/mm^2)	G^{II} (N/mm^{2}mm)
Mortar	1000	0.2	0.5	0.01	0.75	0.05
Brick unit	5000	0.2	1.3	0.01	-	-
	Geometrical properties					
	h (mm)			l (mm)	t (mm)	
Basic cell	2			12	1	

Shear modulus G of the bed joint can be calculated as:

$$G = \frac{E}{2(1+\nu)} = 416.67\ \mathrm{N/mm^2}$$

Parameter A_s should always be positive with $l_s = H$:

$$\left(\frac{G^{II}G}{H\sigma_s^2}-\frac{1}{2}\right)>0$$

Therefore, H should be:

$$H<\frac{2G^{II}G}{\sigma_s^2}=74\ \mathrm{mm}$$

The maximum element size should be $74\ \mathrm{mm}$.

The analytical solution could be found from the codes implied in MATLAB and the major codes are:

```
 % outer loop: strain increment
 for i = 1:300
    gama = gama + 0.00001;
   % inner loop: verification of damage factor
   while d < 1
   % shear stress in bed joint:
    tau_b = G*gama*3;
    tau = max(tau_b,sig_s);
   % smeared cracking model
   % characteristic length of element: element size
    l_s = H;
   % A_s must be positive, check maximum mesh size: H < 74
    A_s = (((G_II*G)/(l_s*sig_s^2))-(1/2))^(-1);
   % calculate damage factor from stress
    d_b = 1-sig_s*exp(A_s*(1-(tau/sig_s)))/tau;
    if d_b <= 0
       break;
    end
    if abs(d_b-d) < T
        break
    end
    d = d_b;
   end
   % total damaged stress of cell
    tau_b_d = (1-d)*gama*G*3;
```

Where “$gama$” represents γ_0 in eq. (4.22) and “d_b” is the damage factor from eq. (4.5). while “tau_b_d” is the damaged shear stress, being representative as τ^0 in eq. (4.23), see chapter 4.

The value of tolerance T between d_t and d_{t+1} is introduced here. Let’s assume that if the absolute value of the difference between d_t and d_{t+1} is smaller than the scalar value of tolerance T, the value of damage factor would not change from time t to $t+1$.

The effects on the stress-strain curve caused by the values of element size H and tolerance T could be easily seen from figures 8-1 and 8-2.

Figure 8-1 indicates that the element size of the numerical model has effects on the softening process of the material model.

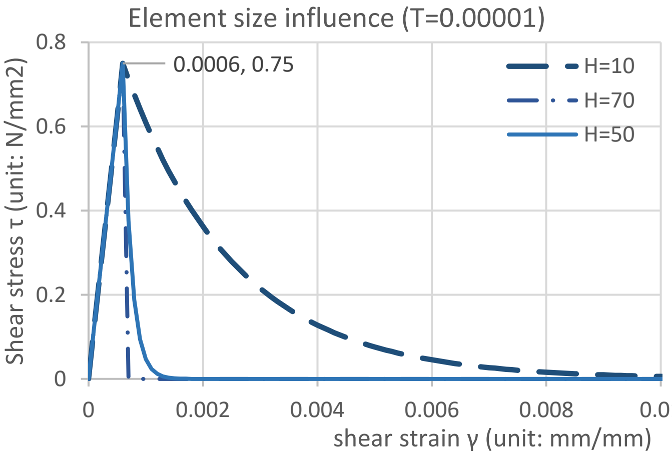


Figure 8-1 Influences of element size: homogenized shear stress τ_{xy}^0 versus macro strain γ_{xy}^0

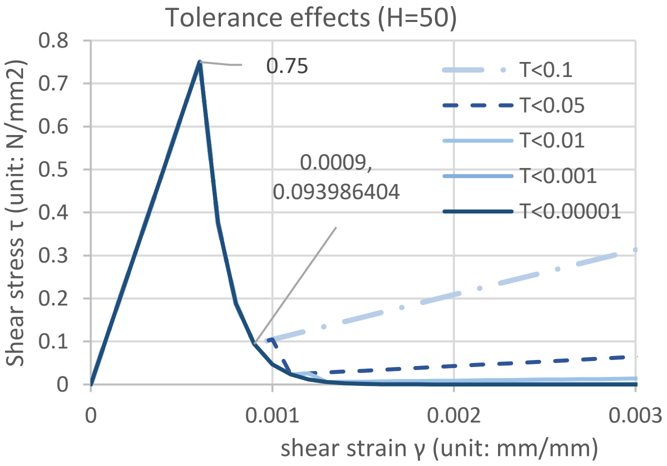


Figure 8-2 Influences of tolerance: homogenized shear stress τ_{xy}^0 versus macro strain γ_{xy}^0

The area of the strain-stress curve should be the correlated special II fracture energy $g_{II} = G_{II}/l_s$ in this model. The l_s is the characteristic length, equal to the element size h as the smeared crack model is applied.

As shown in figure 8-2, the damaged shear stress of the homogenized cell grows up until it reaches the material strength. However, the value of the stress does not drop down to zero if the tolerance is large, which leads to errors occurring. For instance, if the tolerance is assumed to be 0.1 and the value damage state variable d_b tend to be greater than 0.9 at the current load step, the Interpolation of the value at the current step and the next step would never be greater than 0.1 as the damage factor has a range value of $0 \leq d_b \leq 1$. As a result, the damage factor remains the number around 0.9, which leads to residual stiffness.

Therefore, the suitable values of the tolerance and the element size become essential to reduce the errors. Based on the results shown in figure 8-1 and 8-2, the element size $H = 50\ \mathrm{mm}$ and the tolerance $T = 0.00001$ are adopted in this work. With the suitable strain increment of $\Delta\varepsilon_0 = 0.0001$, the analytical result of model 1 can be got as figure 8-3 shown.

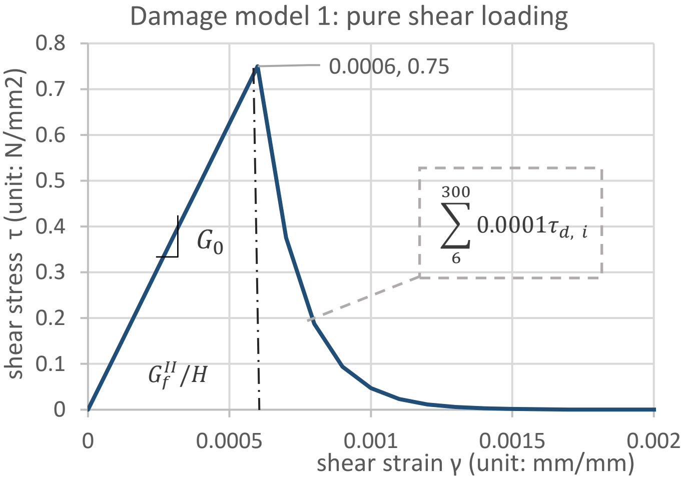


Figure 8-3 Analytical result from MATLAB: homogenized shear stress τ_{xy}^0 versus macro strain γ_{xy}^0

Verification of shear modulus:

(1) Shear modulus of basic cell in figure 8-3:

$$G_{0,model1} = \frac{0.75}{0.0006} = 1250\ \mathrm{N/mm^2}$$

(2) Initial value of shear modulus of basic cell:

$$G_0 = \frac{\partial\,\tau^0}{\partial\,\gamma_0} = \frac{h+t}{t}(1-d)G_{xy}^b$$

At elastic state, d should always be equal to zero.

$$G_0 = \frac{h+t}{t}G_{xy}^b = 3G_{xy}^b = 1250\ N/mm^2$$

Then we can quickly get: $G_{0,model1} = G_0$

Verification of fracture energy:

(1) Second fracture energy of basic cell in figure 8-3:

$$G^{II}_{0,\,model1} = \left(\frac{1}{2}\times 0.75\times 0.0006 + \sum_{6}^{300} 0.0001\tau_{d,\ i}\right)\times 50 = 0.015\ \mathrm{N/mm^2mm}$$

(2) Initial value of second fracture energy of the basic cell:

$$G^{II} = \frac{1}{2}\sigma_s\gamma_b H = \frac{1}{2}\frac{h+t}{t}\sigma_s\gamma_0 H = \frac{3}{2}\sigma_s\gamma_0 H$$

$$G^{II}_0 = \frac{1}{2}\sigma_s\gamma_0 H = \frac{1}{3}G^{II} = 0.017\ \mathrm{N/mm^2mm}$$

Then we can get the ratio of $G^{II}_{0,\,model1}$ and G^{II}_0 and the error of the analytical result as:

$$Ratio\ = \frac{G^{II}_{0,\,model1}}{G^{II}_0} = 0.88, Error\ =\ \frac{G^{II}_{0,model1} - G^{II}_0}{G^{II}_0} = 11.8\%$$

Therefore, this analytical solution of model 1 can be acceptable.

Implement in Fortran code

The algorithm introduced in section 4.2 is implemented by Fortran 77 in this part as User-supplied material (USRMAT) subroutine in Diana 10.3 FEA and then would be modelled by 2D plane stress element as a "single element model". The stresses of the components could be calculated at every step by this file. Therefore, this procedure would be essential to determine the reliability of the material model applied in finite element analysis introduced in model 1.

```
! CALCULATE OF DAMAGED PARAMETERS
      H     = RATIO*T
      LS    = HH
      AS    = (((GII*G)/(LS*SIGS**2.0D0))-(1.0D0/2.0D0))**(-1.0D0)
C
! DEFINATION OF SHEAR STRAIN
      GAMA  = EPS0(3)+DEPS(3)
C
! INNER LOOP: CHECK DAMAGE FACTOR
      DO 30, WHILE(D .LT. 1.0D0)
         TAUBC = TAUB(GAMA,G,H,T,GAMAB)
         TAU   = MAX(TAUBC,SIGS)
         DBC   = DB(TAU,SIGS,AS)
C
! DAMAGE FACTOR FROM STRESS CAL.
         IF (DBC .LE. 0.0D0) THEN
            EXIT
         END IF
         TC    = ABS(DBC-D)
         IF (TC .LT. TOR) THEN
            EXIT
         END IF
C
! FINAL DAMAGE FACTOR
         D     = DBC
30    CONTINUE
C
! DAMAGED STRESS AND DAMGED STIFFNESS
      TAUBD      = (1.0D0-D)*TAUBC
      TAUD       = TAUBD
      SIG(3)     = TAUD
      STIFF(3,3) = (1.0D0-D)*G
C
! STORE OUTPU BY USRSTA MATRIX
      USRSTA(1)  = D
      END SUBROUTINE USRMAT
C
C SUBPROGRAM: EQUILIBRIUM EQUATIONS OF SYSTEM
      REAL FUNCTION TAUB(GAMA,G,H,T,GAMAB)
      IMPLICIT NONE
      REAL, INTENT(IN)::GAMA
      REAL            ::H, T, GAMAB, G
! H IS HALF OF UNIT HEIGHT
! T IS HALF HEIGHT OF JOINT THICKNESS
      GAMAB = (H+T)*GAMA/T
      TAUB  = GAMAB*G
      RETURN
      END FUNCTION TAUB
C
C SUBPROGRAM: DAMAGE FACTOR CAL.
      REAL FUNCTION DB(TAU,SIGS,AS)
      IMPLICIT NONE
      REAL, INTENT(IN)::TAU
      REAL            ::SIGS,AS
      DB    = 1.0D0-SIGS*EXP(AS*(1.0D0-(TAU/SIGS)))/TAU
      RETURN
      END FUNCTION DB
```

The parameters used to describe the material and geometrical properties of the material model should be categorised as "USRSTA" or "USRVAL". The damage factor d is set to be the user state variable "USRSTA", the value of which could be stored every step and could be seen as an input as well as an output coefficient. "USRVAL" is used to set up user parameters, which acts as the unchangeable input parameter. The user parameters could be concluded as:

(1) Shear fracture energy – GII
(2) The ratio of brick height and joint thickness – RATIO
(3) Tolerance of damage factor – TOR
(4) Element size – HH
(5) Shear strength – SIGS
(6) Shear stiffness – G

The principal Fortran codes can be found above.

According to the “2D plane stress element” described in Diana FEA documentation [40], the external shear stress should be SIG(3), and the strain should be EPS(3), and DEPS(3) should be strain increment.

Using the main program to calculate the external stress of basic cell τ^0, the subprogram to find the stress of the proposed component and the damage factors computed by eq. (4.5), see chapter 4.

The shear behaviour of the homogenized material model could be obtained by adding uniformly distributed stress in the shear direction in the single element model, see figure 8-1. The horizontal displacement contour could be viewed as figure 8-2 shown.

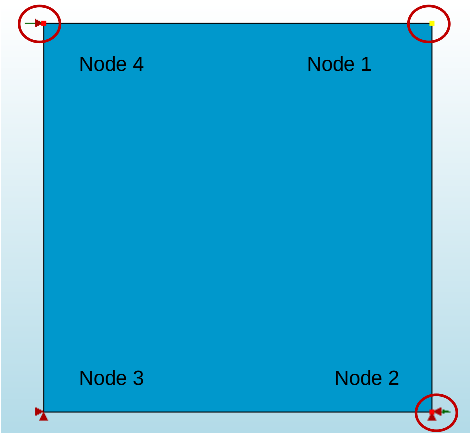


Figure 8-4 Per element model: shear behaviour

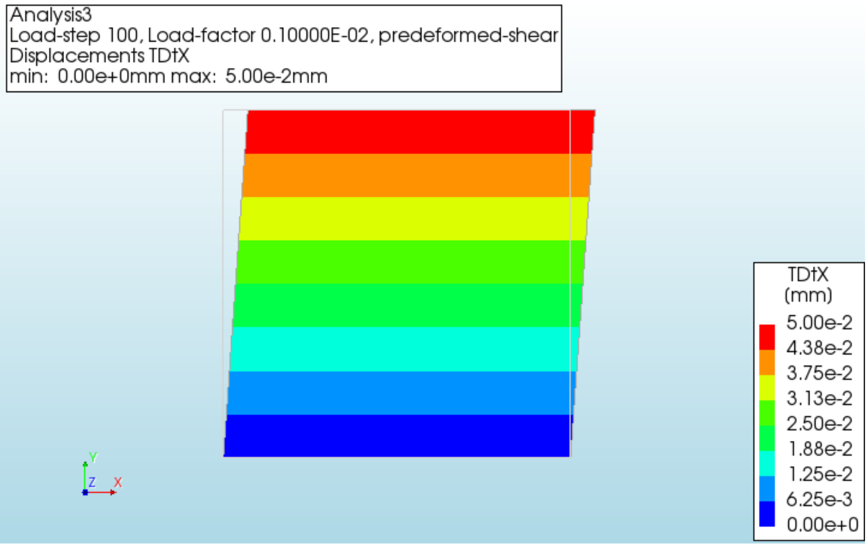


Figure 8-5 Displacement contour of the model 1

The material and geometrical properties described in table 8-1 are used. The comparison of the analytical results computed by MATLAB and the numerical results computed in DIANA FEA is shown in figure 8-6.

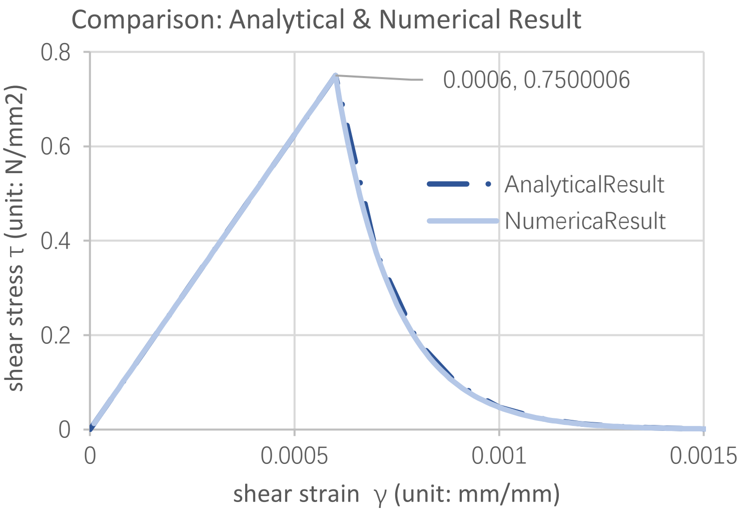


Figure 8-6 Comparison: shear stress τ_{xy}^{0} versus strain γ_{xy}^{0} from the analytical results and the homogenized material model 1

As shown in figure 8-3, the numerical results are the same as the analytical ones, indicating the validity of material model 1 in the single element model.

, the material parameters are set to be the new values to assess the sensitivity of the material model 1:

(1) The shear stiffness of the bed joint changes from $G = 416.67\ \mathrm{N/mm}$ to $G = 1300\ \mathrm{N/mm}$, see figure 8-7.
(2) The shear strength of the bed joint changes from $\sigma_s^b = 0.75\ \mathrm{N/mm^2}$ to $\sigma_s^b = 0.15\ \mathrm{N/mm^2}$, see figure 8-8.

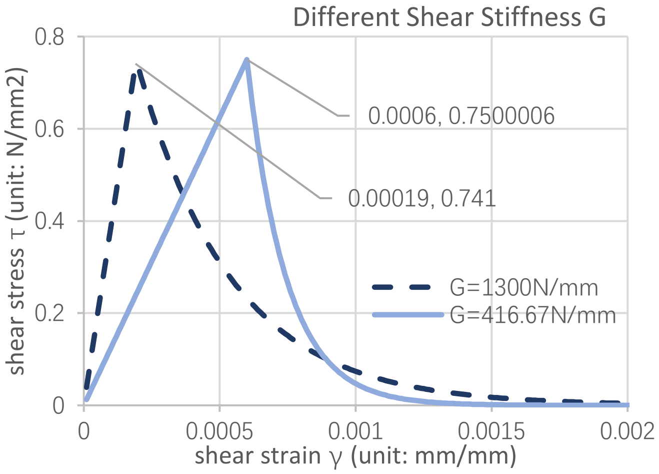


Figure 8-7 Different shear stiffness of the bed joint: shear stress τ_{xy}^0 vs shear strain γ_{xy}^0

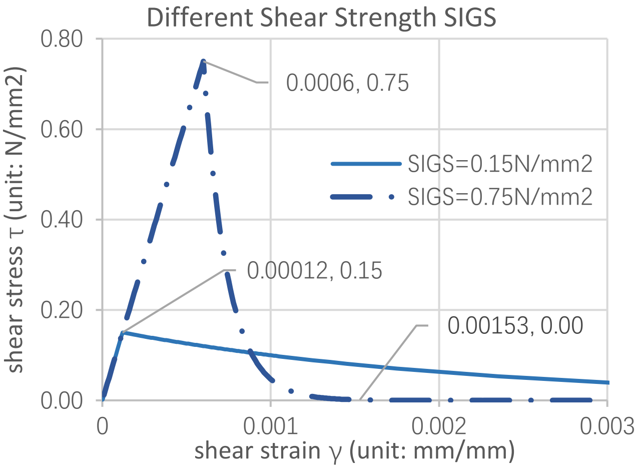


Figure 8-8 Different shear strength of the bed joint: shear stress τ_{xy}^0 vs shear strain γ_{xy}^0

The ultimate limit state strain should become larger if only shear stiffness is reduced, and the area of the strain-stress curve should not be changed. As figure 8-7 shown, model 1 implemented in Diana FEA 10.3 could still be used if the shear stiffness of the bed joint is changed.

Similarly, the ultimate limit state strain should be smaller if only the material's shear strength is reduced, and the area of the strain-stress curve should not be changed. As figure 8-8 shown, model 1 implemented in Diana FEA 10.3 could still be used if the shear strength of the bed joint is changed.

8.1.2. Model 2: horizontal tensile behaviour

Analytical solution

According to the algorithm introduced in figure 5-5, damage variables r_i $(i = h, u, c, b)$ are related to internal stresses σ_{xx}^h, $\bar{\sigma}_{xx}^u$, σ_{xx}^c and τ_{xy}^b solved by "damage" equilibrium equations of system in the format as:

$$\sigma_{xx}^i = f_{sigxxi}(r_u, r_h, r_c, r_b, \varepsilon_{xx}^0), i = h, u, c \tag{8.1}$$

$$\tau_{xy}^b = f_{tauxyb}(r_u, r_h, r_c, r_b, \varepsilon_{xx}^0) \tag{8.2}$$

Expressions of internal and external stress can be first be solved by Maple as a result in the format as eq. (8.1) and (8.2) shown, and we can derive them into MATLAB code to determine the analytical solution of damage model 2.

To simplify the expressions, coefficients C_1, C_2 and C_3 are introduced here:

$$E_h = E_b = E_c = E, E_u = C1 \cdot E \tag{8.3}$$

$$l = C2 \cdot t,\ h = C3 \cdot t \tag{8.4}$$

Where E_h, E_b and E_c are young's modulus of the mortar joint, E_u is young's modulus of the brick unit.

Geometrical and material properties of components can be found in chapter 5, table 2.1-1. Parameter A_s, A_t introduced in eq. (4.5) and (5.4) should always be positive. Therefore, the maximum size of the element used in finite element mesh should be satisfied as:

(1) Calculated from A_{tm} (Parameter of the mortar joint in tension): $H < 80 \text{ mm}$
(2) Calculated from A_{tu} (Parameter of the brick unit in tension): $H < 59 \text{ mm}$
(3) Calculated from A_{sm} (Parameter of the bed joint in shear): $H < 74 \text{ mm}$

Therefore, the maximum element size of model 2 should be $59\ mm$. Let's set $H = 50 \text{ mm}$ here. The MATLAB code in model 2 can be found in Appendix A. Results could be found in figure 8-9.

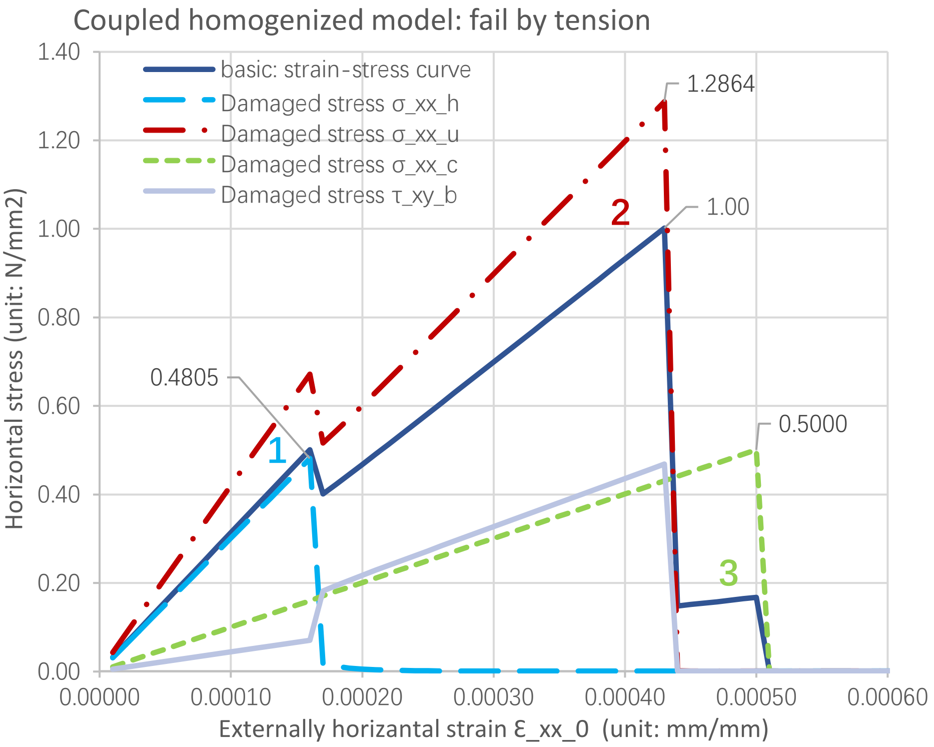


Figure 8-9 Model 2: internal damaged and homogenized stresses -external strain ε_{yy}^{0} curve

🟥 brick unit 🟨 bed joint 🟩 cross joint 🟦 head joint

Note: "0" means external force, "u" means brick unit (red), "h" means head joint (blue), "c" means cross joint (green), "b" means bed joint (orange)

It can be seen that the head joint (shown in blue dashed line) will fail in tension at point 1 as its damaged tension stress "$sigxxh$" reach its strength of $0.5\ N/\text{mm}^2$. However, this localized damage does not have many effects on the macro stiffness of the homogenized cell. The brick unit and cross joint can still expand horizontally, while the bed joint is still moving in shear and horizontal tension direction.

The macro-stress has been increasing until reaching point 2. Vertical crack caused by tension behaviour in brick unit occurs with the value of the brick unit's tension strength being equal to $1.3\ \text{N}/\text{mm}^2$, see the red long-dash-dotted line.

At this stage, shear stress in the bed joint drops down to zero suddenly as the brick unit is damaged as the grey line shown, while the cross joint maintains moving and expanding laterally until point 3, see the green dotted line. As a result, the homogenized cell is damaged after point 3.

Implement in Fortran code

The Fortran codes could be found in Appendix A.

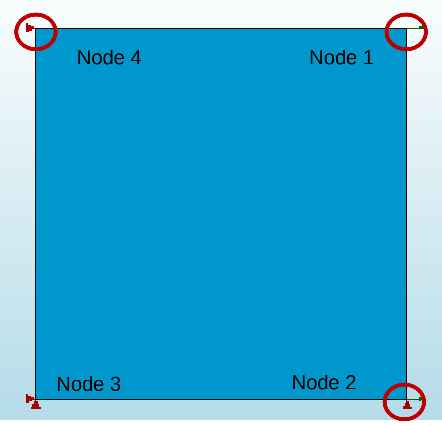


Figure 8-10 Per element model: horizontal tensile behaviour

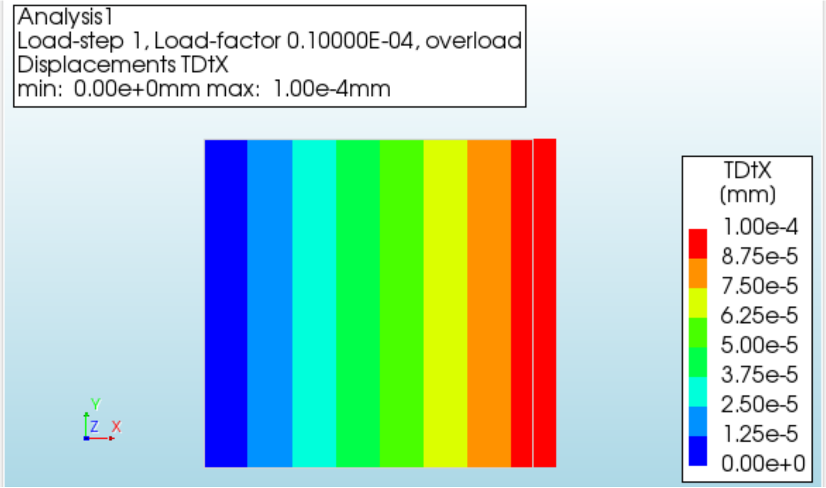

Analysis1
Load-step 1, Load-factor 0.10000E-04, overload
Displacements TDtX
min: 0.00e+0mm max: 1.00e-4mm

Figure 8-11 Displacement contour of the model 2

The horizontal tensile behaviour of the homogenized model introduced in chapter 5 can be obtained once the uniformly distributed stress in horizontal tension is loaded at the side boundaries of the single element model, see figure 8-6. Meanwhile, the horizontal displacement contour could be seen from figure 8-12.

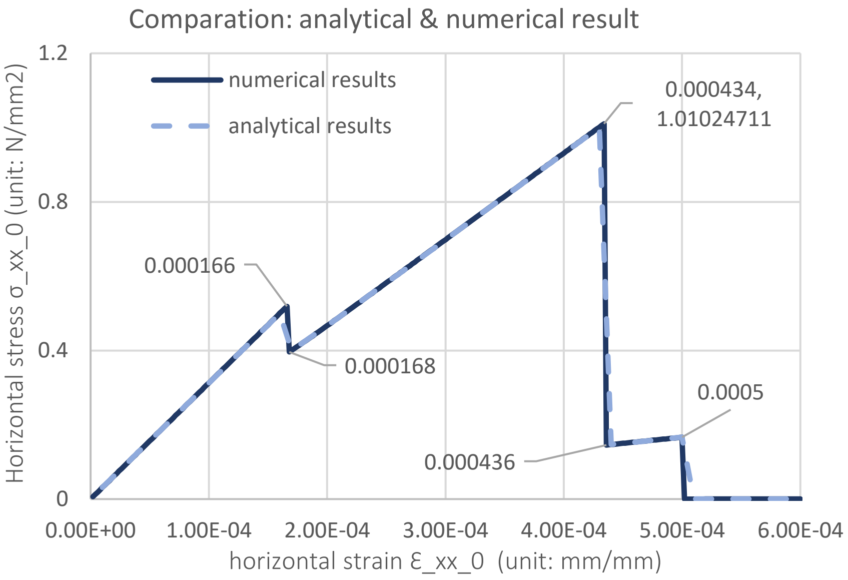


Figure 8-12 Comparison: horizontal stress σ_{yy}^{0} vs strain ε_{yy}^{0} from the analytical results and the homogenized material model 2

The material and geometrical properties of model 2 are set to be the same as that of model 1, see table 8-1. The comparison of the analytical results computed by MATLAB and the numerical results given by the analysis solution from DIANA FEA could be seen in figure 8-13.

As shown in figure 8-13, the numerical results are similar to the analytical ones, indicating the validity of material model 2 in the single element model. The slight differences between the analytical results and numerical results may be caused by different strain increment set in MATLAB and DIANA FEA.

8.1.3. Model 3: vertical compression behaviour

Analytical results

MATLAB Code could be found in Appendix D.

The material and geometrical properties of the unit cell are set as table 6-2 shown, see chapter 6 section 6.2. The length of the element is set to be $10\ \mathrm{mm}$. The tolerances assumed as the convergency conditions for finding the relevant damage variables and the hardening/softening parameters are set to be 0.00001.

The shear stiffness of the bed joint is assumed to be a variable:

(1) Case 1: a large value to limit shear damage occurring in the basic cell, see figure 8-13 to 8-15;
(2) Case 2: a small one to make shear damage in the bed joint occurring before compressive crushing in other components, see figure 8-18 to 8-20.

Case 1: a large shear stiffness of the bed joint

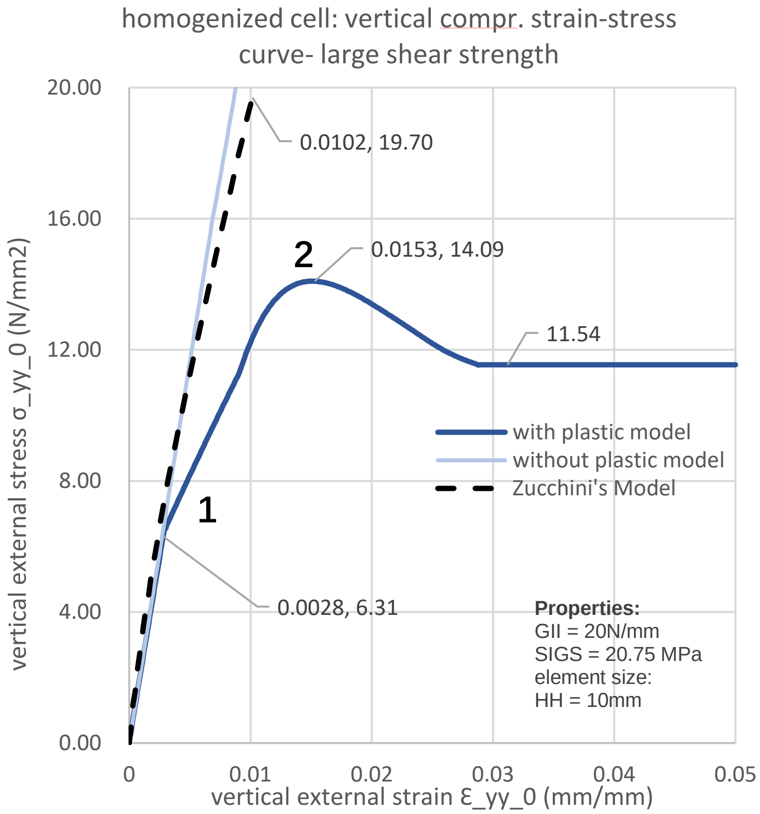


Figure 8-13 vertical compression behaviour of homogenized cell without considering shear damage at interface of brick and bed joint: σ_{yy}^{0} vs the macro strain ε_{yy}^{0}

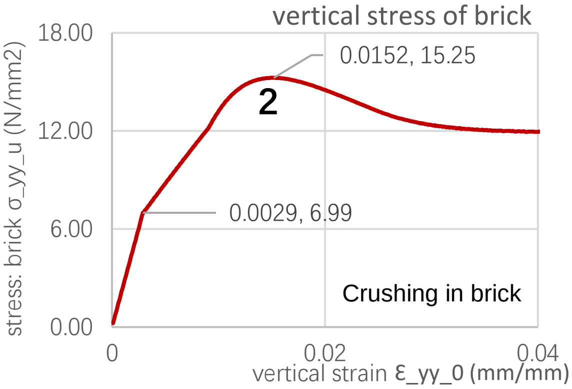


Figure 8-14 the internal stress of brick σ_{yy}^{u} vs. the macro strain ε_{yy}^{0}

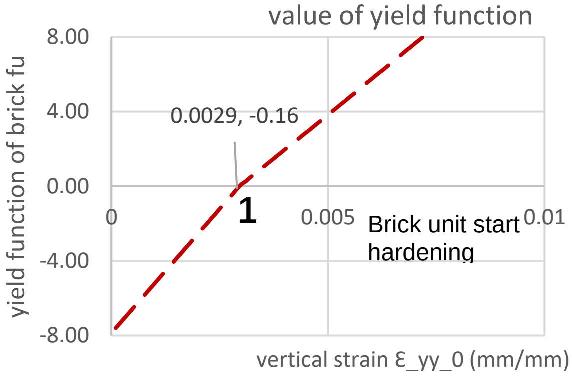


Figure 8-15 The value of yield function f_u with elastic predictors in the brick unit vs. ε_{yy}^{0}

As shown in Figures 8-13 and 8-15, the value of the brick unit's yield function, assessed by the elastic predictors of the brick unit from the damage model, changes from a negative to a positive number at point 1.

That means plastic deformations occur in the brick unit from step 29.

After that, the brick unit deforms plastically until the vertical compression stress of the brick unit reaches its maximum value at point 2, see figure 8-14. At the same time, vertical stress of the homogenized cell is achieving its ultimate limit state, with the value of $14.09\ \mathrm{N/mm^2}$ and then drop to $11.54\ \mathrm{N/mm^2}$ as the softening occurs in the basic unit cell. The brick unit is crushing during the post-peak phase, leading to fewer contact areas between the brick unit and the bed joint. Therefore, the external strain should be transformed to the bed joint directly as the brick unit cannot support load anymore. However, the bed joint is hard to be damaged in shear at that time since its shear deformation depends on the brick unit's and the head joint's horizontal deformations. As a result, the homogenized vertical stress maintains the value of $11.54\ \mathrm{N/mm^2}$.

The internal stress of the brick unit in the x-direction σ_{xx}^{u} is recorded at every load step to investigate that if the split tensile stress was introduced successfully. As shown in figure 8-16, the horizontal stress of the brick unit consists of two parts: the elastic part (long dash-dot line) and the inelastic part, generated from step 30. The elastic part is produced by the elastic deformations of the brick unit in the x and y directions based on the deformed mechanisms of the basic unit cell. After load step 30, the brick unit's vertical compression stress produced plastic deformations in the x and y directions, which is associated with the generation of the micro-cracks. These plastic deformations lead to the inelastic stress of the brick unit in the horizontal direction.

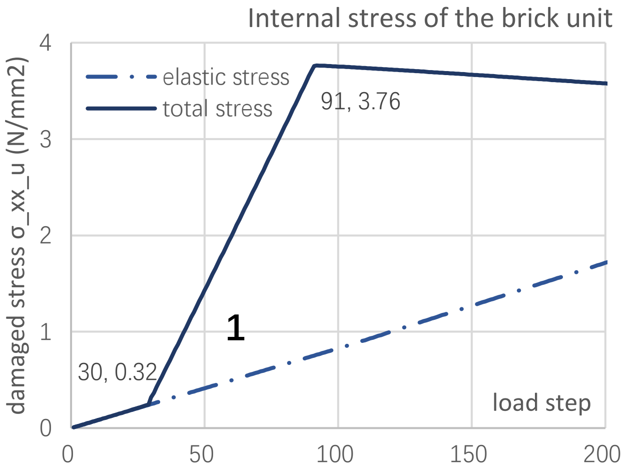


Figure 8-16 horizontal tension stress of the brick unit σ_{xx}^{u} versus load step

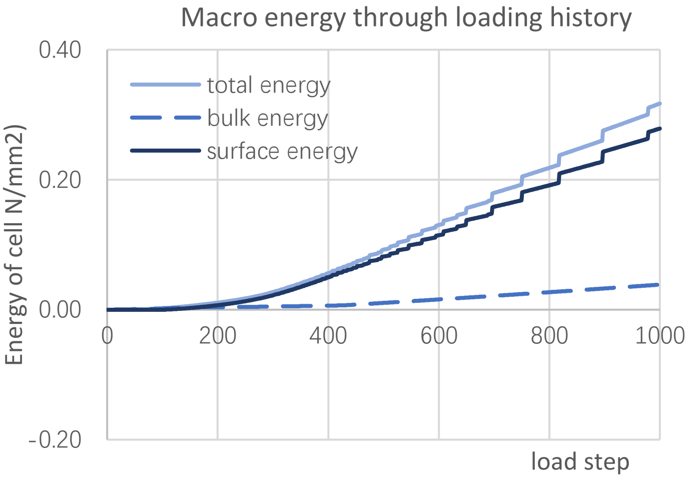


Figure 8-17 energy of the basic cell versus load step

This inelastic tension stress could also be called split tensile stress, which is generated due to the compressive splitting effects.

Furthermore, macro-energy of the basic cell per area could be computed to investigate if the splitting effects are introduced successfully in the whole homogenized cell. Note that the energy of macro horizontal strain ε_{xxy}^{0} and vertical compressive stress σ_{yy}^{0} should be recorded since the splitting effects on the lateral plastic deformation is focused on in this model.

The total energy of the basic unit cell should be the triangle area of the strain-stress curve, which can be computed by equation (8.5).

$$E_{total} = \frac{1}{2} \cdot \varepsilon_{xxy}^{0} \cdot \sigma_{yy}^{0} \tag{8.5}$$

$$\varepsilon_{xxy}^{0} = \frac{l \cdot \varepsilon_{xx}^{u} + t \cdot \varepsilon_{xx}^{h}}{l + t} \tag{8.6}$$

ε_{xxy}^{0} is the homogenized strain in the horizontal direction of the basic unit cell under vertical compressive loading. It could be computed by the horizontal strains of the brick unit ε_{xx}^{u} and head joint ε_{xx}^{h}. The horizontal strains consist of the elastic ones ε_{xx}^{ie} and the plastic ones ε_{xx}^{ip}, see equation (8.7).

$$\varepsilon_{xx}^{i} = \varepsilon_{xx}^{ie} + \varepsilon_{xx}^{ip}, i = u, h \tag{8.7}$$

Let's combine equations (8.7) and (8.6) as:

$$\varepsilon_{xxy}^{0} = \varepsilon_{xxy}^{0e} + \varepsilon_{xxy}^{0p} \tag{8.8}$$

$$E_{elastic} = \frac{1}{2} \cdot \varepsilon_{xxy}^{0e} \cdot \sigma_{yy}^{0}, \varepsilon_{xxy}^{0e} = \frac{l \cdot \varepsilon_{xx}^{ue} + t \cdot \varepsilon_{xx}^{he}}{l + t} \tag{8.9}$$

$$E_{surface} = E_{total} - E_{elastic} \tag{8.10}$$

Equation (8.8) to (8.10) indicate that the total energy should be composed of two components:

(1) The elastic potential energy (bulk energy) stored in the materials has elastic deformations under external forces. This energy would disappear as the deformations tend to recover when the external forces are removed. Therefore, we could use elastic strain ε_{xxy}^{0e} to compute the bulk energy of the homogenized cell, see equation (8.9);
(2) Surface energy is defined as the excess energy at the surface of the materials. The excess energy is the residual part of the total and bulk energy, see equation (8.10). Furthermore, the surface energy should be associated with the formation of micro-cracks by introducing elastoplastic behaviours in the materials.

As a result, figure 8-17 could be obtained. The surface energy of the homogenized cell has value, which indicates the compressive splitting effects are introduced successfully in the whole cell.

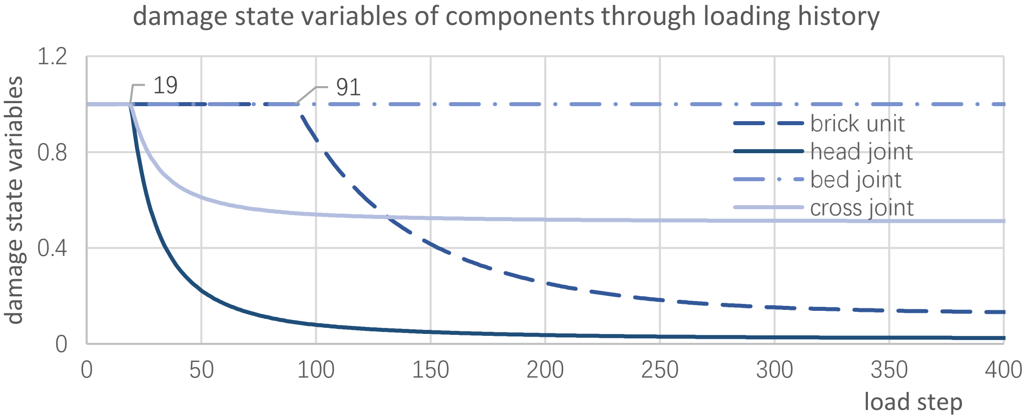


Figure 8-18 Damage state variables of components $r_i = 1 - d_i$ $(i = u, h, b, c)$ versus load step: case 1

Damage state variables $r_i = 1 - d_i$ $(i = u, h, b, c)$ of the component could be recorded at each load step, see figure 8-18. The components are undamaged when $r_i = 1$, while they are partially damaged when $0 < r_i < 1$ and damaged when $r_i = 0$. Therefore, we can investigate each component's damage status with the value of damages state variables at every load step.

As shown in figure 8-18, the head joint and the cross joint start being damaged from step 19 and then the brick unit is damaged from step 91 due to the equivalent tension behaviour. However, the bed joint is not damaged in shear through the whole load path as the shear stiffness of the bed joint is set to be extremely large in this case. Note that the equivalent tension behaviour should include the one caused by deformed mechanisms of the basic cell and the compressive splitting effects.

Case 2: a small shear stiffness of the bed joint

If the shear stiffness of the bed joint is small, which is closer to the reality of physical properties, the basic unit cell would be damaged once the bed joint is damaged in shear at point 2, see figures 8-19 to 8-21. Before point 2, the brick unit and other components still tend to deform elastically before point 1 and plastically after point 1. Note that the deformation of the brick unit changes from plastic one to elastic one again at point 3 due to strain softening in the damaged cell, see figure 8-20.

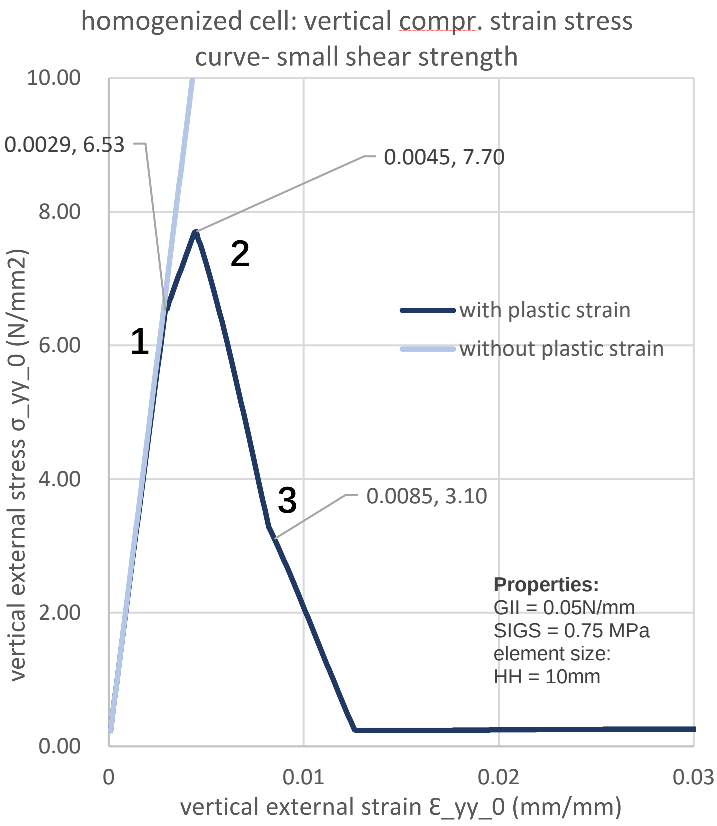


Figure 8-19 vertical compression behaviour of homogenized cell with considering shear damage at interface of brick and bed joint: σ_{yy}^0 vs. the macro strain ε_{yy}^0

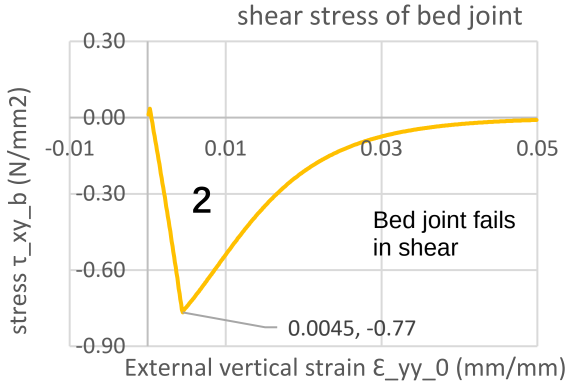


Figure 8-20 the shear stress of bed joint τ_{xy}^b vs. the macro strain ε_{yy}^0

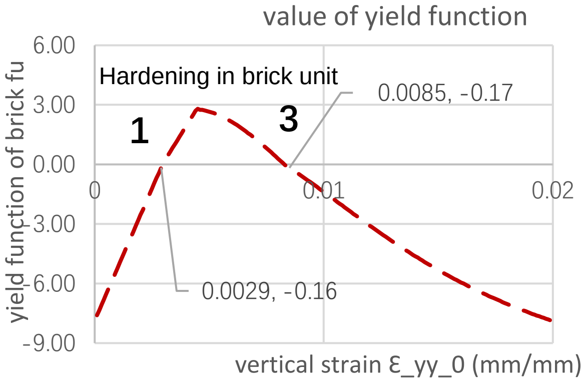


Figure 8-21 The value of yield function f_u with elastic predictors in the brick unit vs. ε_{yy}^0

The energy of the basic cell could be quite complicated as the shear failure occurring in the bed joint is taken into account. Let's record the total energy and bulk energy of the basic cell by using the equations (8.5) and (8.9) again, see figure 8-22.

Figure 8-22 indicates that the bulk energy of the homogenized material model is not consumed as the value of total energy is always equal to or smaller than that of the bulk energy through the whole loading path. At load step 29 (point 1 in figure 8-19), micro-cracks start generating in the brick unit and then the stiffness of the homogenized cell is reduced due to this formation of micro-cracks, see figure 8-19. As a result, the total energy of the basic cell becomes smaller than the bulk energy, see figure 8-22.

The vertical compression stress of the homogenized cell reaches its ultimate limit state strength once the bed joint is damaged in shear, which is provided by figure 8-20. That means the plastic shear deformation at the brick unit and bed joint interface is the dominating one compared with other plastic deformations. Therefore, the plastic shear behaviour of the bed joint could be investigated to figure out how the plastic deformations of components make effects on ultimate limit state strain of the homogenized cell.

The damaged shear stress of the bed joint with and without considering the plastic shear deformation through the loading history could be recorded as figure 8-23 shown. The bed joint would be damaged in shear at step 45 (point 2 in figure 8-19) when the plastic shear deformation is taken into account, while it would be damaged at step 65 if the plastic shear deformation was not included. Note that the value of the bed joint's shear stress changes from positive to negative as the plastic deformations develop in the components. From the physical aspect, this phenomenon appears as the developed brick unit's horizontal plastic strain is larger than the head joint's. According to the equations derived in chapters 5 and 6, the sign of in-plane shear stress should depend on the brick unit's and head joint's horizontal strains. The shear strain is assumed to be positive of the head joint's horizontal strain is larger than the brick unit's.

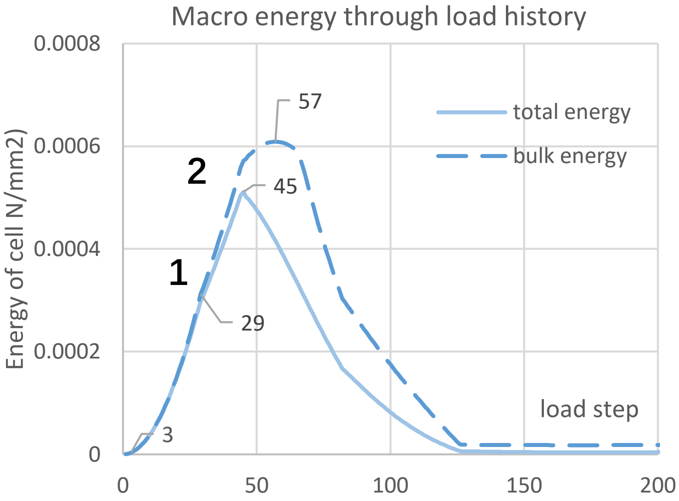


Figure 8-22 energy of the basic cell versus load step

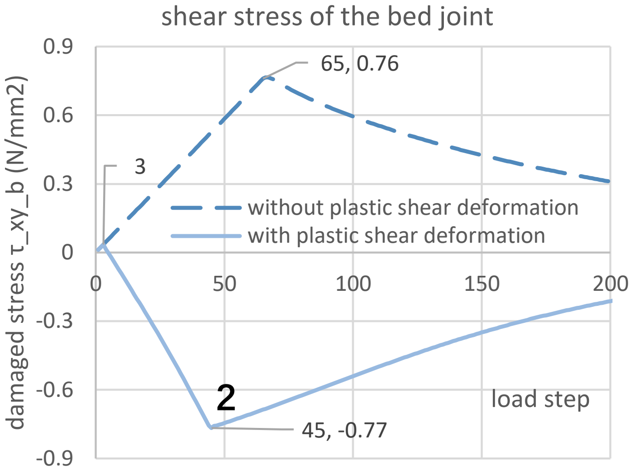


Figure 8-23 Damaged shear stress of bed joint τ_{xy}^{b} versus load step: stress computed without considering plastic shear deformation (long dash line); stress computed with considering plastic shear deformation

Therefore, it can be said that the plastic shear strain of the bed joint causes larger shear stress compared to the one computed based on the deformed mechanisms of the basic cell at every load step. This phenomenon leads to the vertical stress of the basic unit cell achieves its ULS before the elastic potential energy of the material is consumed.

Note that the bulk energy discussed here is indicated as one of the macro material properties of the masonry structure rather than the material properties of the components, such as the bricks or mortars. As shown in figure 8-21, the brick's elastic potential energy should be partially consumed from load step 29 as the value of yield surface, computed by elastic predictors of the brick unit, is changed to a positive number. That indicates that the elastic stress points are located outside the yield surface, and then the elastoplastic phase should start.

The damage state variables of components through load history could be obtained as figure 8-24 shown.

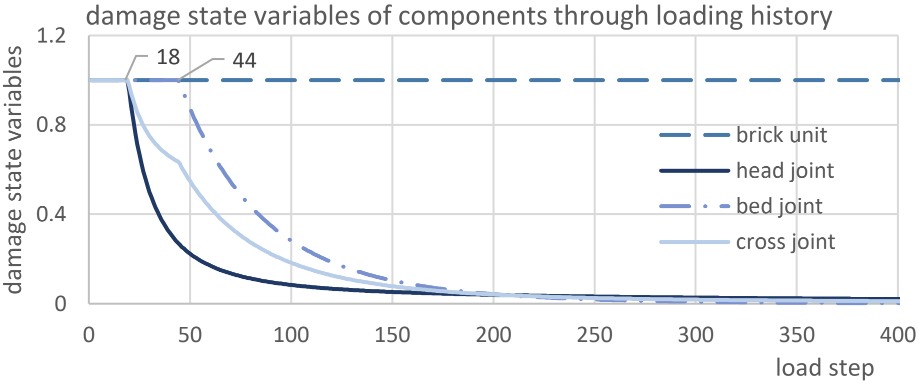


Figure 8-24 Damage state variables of components $r_i = 1 - d_i \ (i = u, h, b, c)$ versus load step: case 2

The head and cross joints are damaged firstly from step 18, and the bed joint is damaged from step 44, leading to the homogenized cell being damaged.

Furthermore, the dilatant effects could be studied by changing the value of the dilatancy angle ψ, see figure 8-25. Let's set up the same values of the dilatancy angle of the bricks and mortars here. As shown in figure 8-25, the ultimate limit state strength and strain of the homogenized cell become larger with a larger dilatancy angle. The dilatancy angle is defined as equation (8.11) shown.

$$\tan(\psi) = \frac{\delta_n}{\delta_s} \tag{8.11}$$

Where δ_n is the vertical deformation and δ_s is the shear deformation of the basic unit cell. That means the shear deformation of the homogenized cell would be smaller with the larger dilatancy angle as the vertical deformation is not changed at every load step. Based on the assumptions made in chapter 3, only shear stress exists at the interface of the brick unit and the bed joint. Therefore, this shear deformation could and only be produced by the shear behaviour of the bed joint.

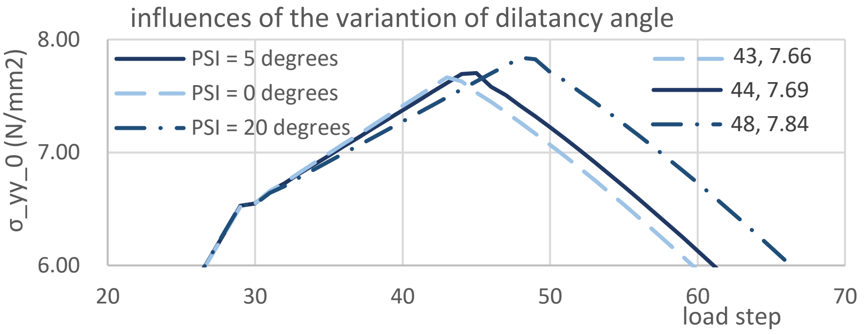


Figure 8-25[4] macro stress in the y-direction σ_{yy}^{0} through load path with different dilatancy angle ψ

If a larger dilatancy angle is adopted, the increment of shear deformation would be smaller if the increment of normal deformation is not changed, see equation (8.11). Meanwhile, the bed joint's shear stiffness is not changed following the variation of the dilatancy. Therefore, the increment of shear stress of the bed joint would be smaller if the increment of the bed joint's shear strain is smaller at each load step. As a result, the bed joint would be damaged in shear for a longer duration. The shear damage in the bed joint is assumed to occur before all other failures. As the bed joint's shear fracture energy is consumed more slowly, more external loading would be absorbed by the brick unit's elastic potential energy if a larger dilatancy angle is applied. Therefore, the ultimate limit state strength and strain of the homogenized material model would be larger if the dilatancy angle is large.

Now, let us compare the results of cases 1 and 2. It can be said that the compressive splitting effects would not make many significant effects on the vertical compressive strength of the basic unit cell in case 2. The bed joint shear failure typically appears before the brick unit crushing failure if the shear stiffness of the bed joint is quite small compared with the stiffness of the brick unit. As a result, the homogenized material model tends to be damaged once the bed joint fails in shear. At the same time, the brick unit still has stiffness in the horizontal direction as its equivalent tensile stress, which consists of the elastic and split tensile stress, does not reach its maximum value.

[4] The bed joint's shear stiffness: shear facture energy 0.05 N/mm, shear strength 0.75 MPa; the element size is set to be 10 mm

8.1.4. Model 4: coupled behaviour

Analytical results

Let us recall back to chapter 3 section 3.5, the homogenized stress tensor $\boldsymbol{\sigma}_0$ could be relative to the macro strain tensor $\boldsymbol{\varepsilon}_0$ in 2D plane stress element by stiffness matrix $\boldsymbol{K}$:

$$\boldsymbol{\sigma}_0 = \boldsymbol{K}\boldsymbol{\varepsilon}_0 \tag{8.12}$$

$$\boldsymbol{\sigma}_0 = \begin{bmatrix} \sigma_{xx}^0 \\ \sigma_{yy}^0 \\ \tau_{xy}^0 \end{bmatrix}, \boldsymbol{\varepsilon}_0 = \begin{bmatrix} \varepsilon_{xx}^0 \\ \varepsilon_{yy}^0 \\ \varepsilon_{xy}^0 \end{bmatrix}, \boldsymbol{K} = \begin{bmatrix} K_{11} & K_{12} & 0 \\ K_{21} & K_{22} & 0 \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \tag{8.13}$$

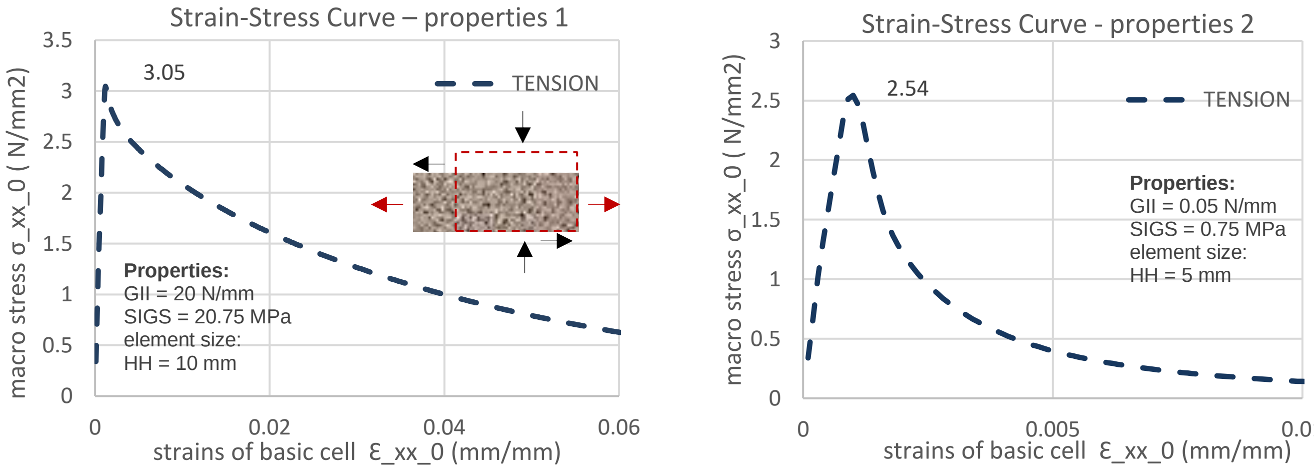
$$K_{11} = \frac{\partial \sigma_{xx}^0}{\partial \varepsilon_{xx}^0}, K_{22} = \frac{\partial \sigma_{yy}^0}{\partial \varepsilon_{yy}^0}, K_{33} = \frac{\partial \tau_{xy}^0}{\partial \varepsilon_{xy}^0} \tag{8.14}$$

$$K_{12} = \frac{\partial \sigma_{xx}^0}{\partial \varepsilon_{xx,y}^0} \cdot \frac{\partial \varepsilon_{xx,y}^0}{\partial \varepsilon_{yy}^0}, K_{21} = K_{31} = \frac{\partial \sigma_{yy}^0}{\partial \varepsilon_{yy,x}^0} \cdot \frac{\partial \varepsilon_{yy,x}^0}{\partial \varepsilon_{xx}^0}, K_{32} = \frac{\partial \sigma_{yy}^0}{\partial \varepsilon_{yy}^0} \tag{8.15}$$

Equation (8.12) could be computed by implementing the expressions of the corresponding strains $\varepsilon_{xx,y}^0$ and $\varepsilon_{yy,x}^0$ defined in section 7.1 and the algorithm introduced in section 7.2.

The strain increments in the x, y and xy direction are set to be the same, with the value of 0.0001 at each load step. In other words, the values of the horizontal, vertical and shear strains of the basic cell are assumed to be increased 0.0001 at every load step.

As a result, the homogenized strain-stress curve in horizontal direction could be obtained as figure 8-26 shown by applying the material and geometrical properties described in table 6-2 in chapter 6 section 6.2.

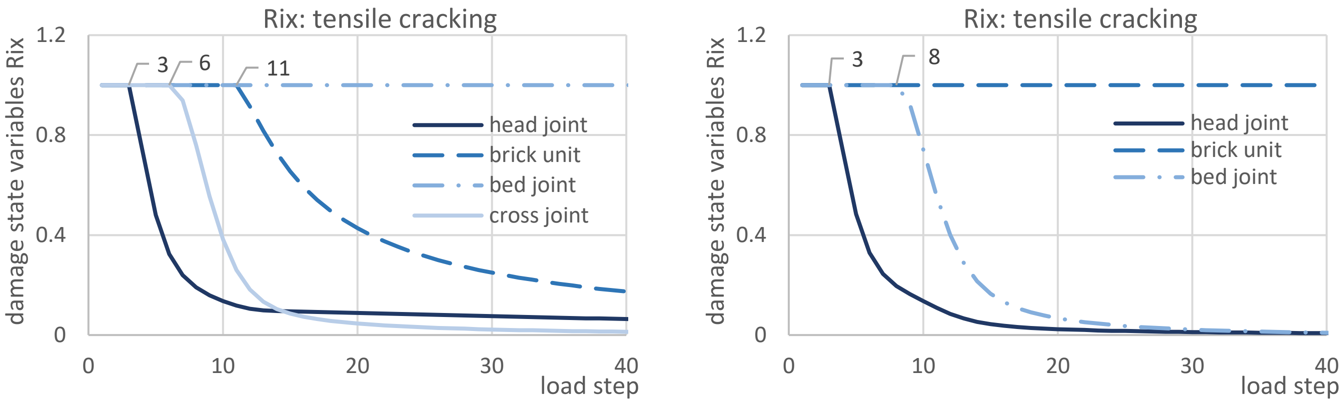


(a) large shear stiffness of the bed joint

(b) small shear stiffness of the bed joint

Figure 8-26 homogenized stress in the horizontal direction σ_{xx}^0 vs. ε_{xx}^0

The damage state variables of components $r_i^x = 1 - d_i^x$ $(i = u, h, b, c)$ could be recorded through the whole load path, as figure 8-27 shown. These variables are only used to compute the macro stress in the horizontal direction, in which case only tensile cracks are assumed to be generated in the components.



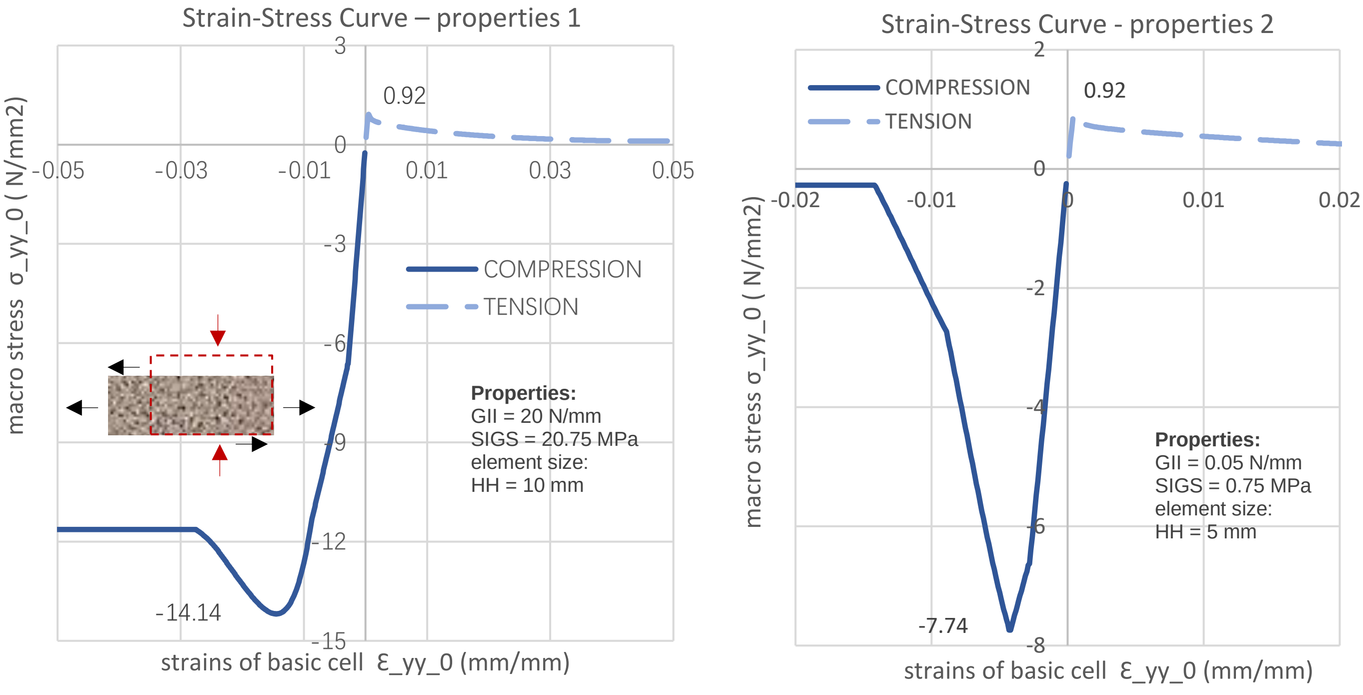
(a) large shear stiffness of the bed joint

(b) small shear stiffness of the bed joint

Figure 8-27 Damage state variables of components $r_i^x = 1 - d_i^x$ $(i = u, h, b, c)$ versus load step: only tensile cracks generated in the components

As figure 8-27 shown, the tensile cracks are firstly generated in the head joint at load step 3 and then in the brick unit at load step 11 if the shear stiffness of the bed joint is considerable. On the contrary, the shear cracks occur in the bed joint at step 8 after the head joint fails in tension if the shear stiffness is small.

The macro stress-strain curve in the vertical direction could be obtained, as figure 8-28 shown.



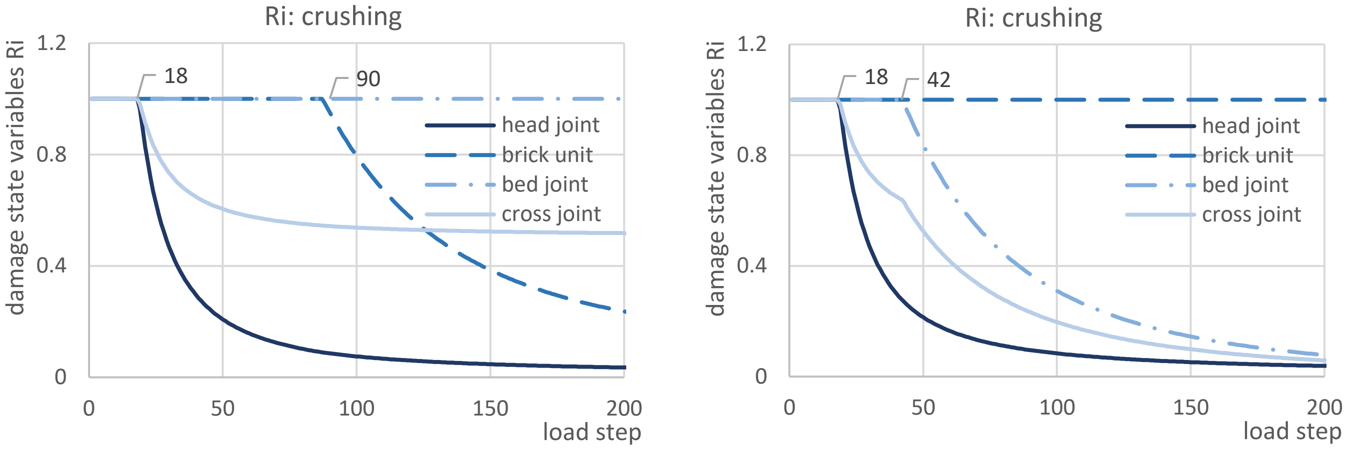
(a) large shear stiffness of the bed joint

(b) small shear stiffness of the bed joint

Figure 8-28 Homogenized stress in the vertical direction σ_{yy}^0 vs. ε_{yy}^0

Note that the results of the macro stress in the vertical direction σ_{yy}^0 are correlated to the results shown in figures 8-13 and 8-19, which indicates that the brick crushing failure mode is introduced successfully.

The damage state variables of components $r_i = 1 - d_i$ $(i = u, h, b, c)$ could be recorded as figure 8-29 shown. In this case, the micro cracks are taken into account.



(a) large shear stiffness of the bed joint

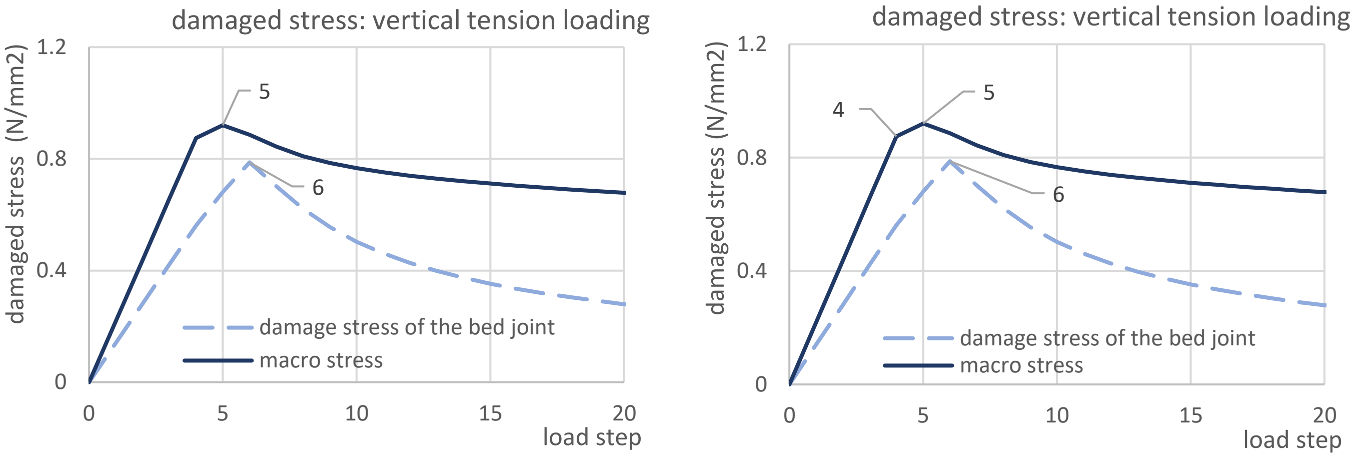
(b) small shear stiffness of the bed joint

Figure 8-29 Damage state variables of components $r_i = 1 - d_i$ $(i = u, h, b, c)$ versus load step: only tensile cracks generated in the components

Again, the micro cracks are firstly generated in the head joint at load step 18 and then in the brick unit at load step 90 if the shear stiffness of the bed joint is substantial. On the contrary, if the shear stiffness is small, the

shear cracks occur in the bed joint at step 42 after the head joint fails due to the compressive splitting effects.

If the vertical load is assumed to be tension, the homogenized cell should be damaged in joint tension failure mode. Therefore, the damaged stress of the bed joint in the y-direction could be recorded as figure 8-30 shown to judge if the joint tension failure mode is introduced successfully.



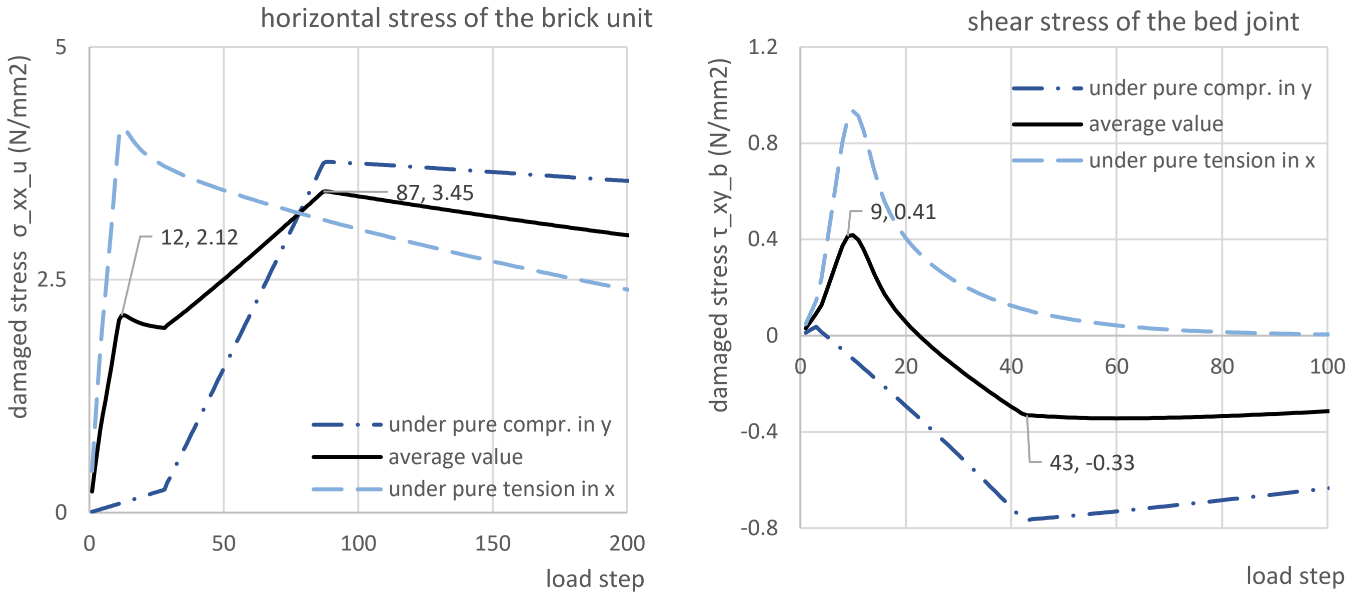
(a) large shear stiffness of the bed joint

(b) small shear stiffness of the bed joint

Figure 8-30 Damaged stress of the bed joint in the y-direction σ_{yy}^{b} and macro stress σ_{yy}^{0} through load path

Figure 8-30 indicates that the homogenized cell is damaged once the bed joint fails in vertical tension. However, horizontal tension loading is set as one of the external forces, and this loading would cause an additional compressive loading in the vertical direction. As a result, the homogenized cell is loaded by vertical tension strain rather than the compressive one from step 4 when the imposed strain is greater than the additional one caused by horizontal loading condition in the y-direction.

The damaged tension stresses of the brick unit σ_{xx}^{u} (large shear stiffness case) and damage shear stresses of bed joint τ_{xy}^{b} (small shear stiffness case) could be obtained as figure 8-31 shown.



(a) large shear stiffness of the bed joint (b) small shear stiffness of the bed joint

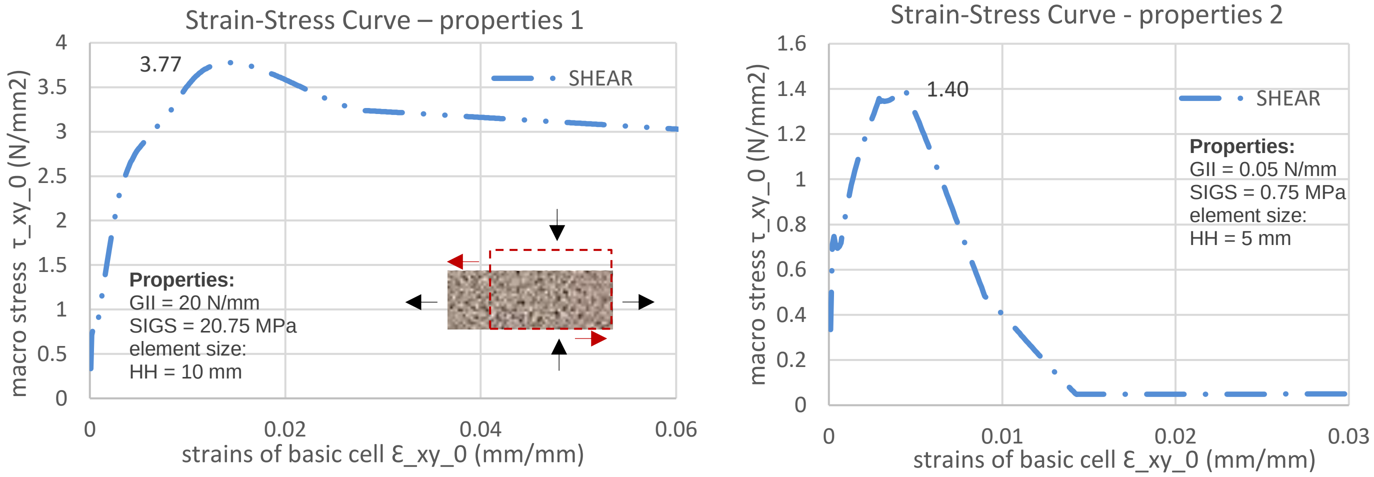
Figure 8-31 Damage stresses of the brick unit σ_{xx}^{u} and the bed joint τ_{xy}^{b} through load path: the value obtained by considering only tensile cracks (dash line); the value obtained by considering micro-cracks (long dash-dot line).

Figures 8-27, 8-29 and 8-31 indicate that the tensile cracks and micro-cracks could be generated in the brick unit (or bed joint), which means the coupled behaviour of the homogenized material model is introduced successfully under the assumed loading conditions.

The homogenized shear strain-stress curves could be obtained when the vertical load is compressive, as shown in figure 8-32.

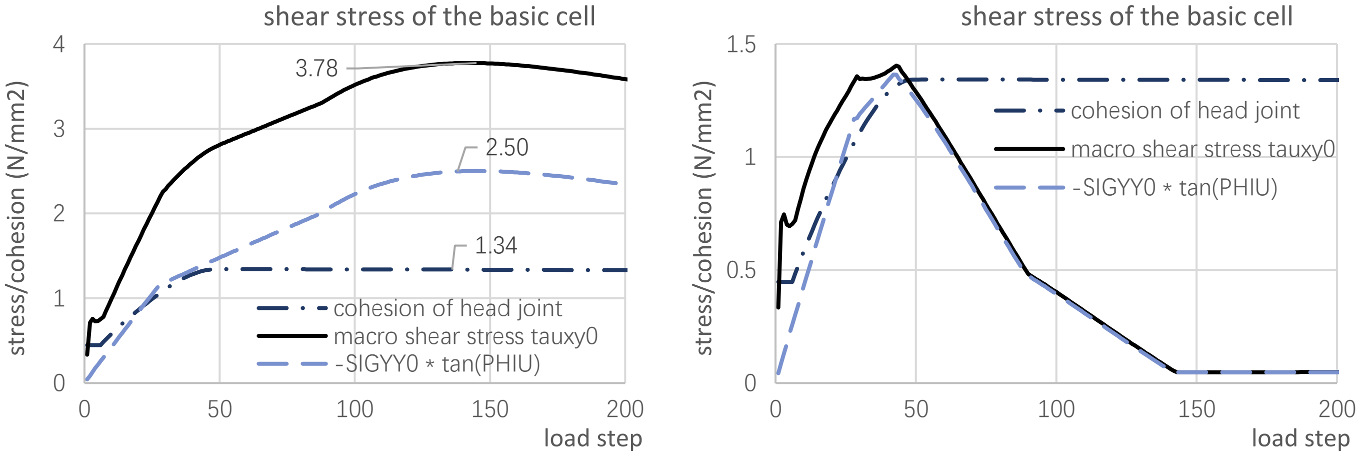
The cohesion of the head joint c_h, the friction stress at the interface of the brick unit and bed joint $\sigma_{yy}^{0} \cdot \tan{(\phi_u)}$ and the macro shear stress τ_{xy}^{0} could be investigated through load history, as figure 8-33 shows.

As shown in figure 8-33, the shear stiffness does not have many effects on the cohesion of the head joint. However, the cohesion of the head joint would be consumed if the shear stiffness of the bed joint is set to be small. When the special II fracture energy $g_{II} = G_{II}/h$ (h is the element size) of the bed joint is set to be smaller. The cohesion of the head joint would be consumed faster in this case.



(a) small shear stiffness of the bed joint (b) large shear stiffness of the bed joint

Figure 8-32 Homogenized stress in shear direction τ_{xy}^{0} vs. shear strain ε_{xy}^{0}

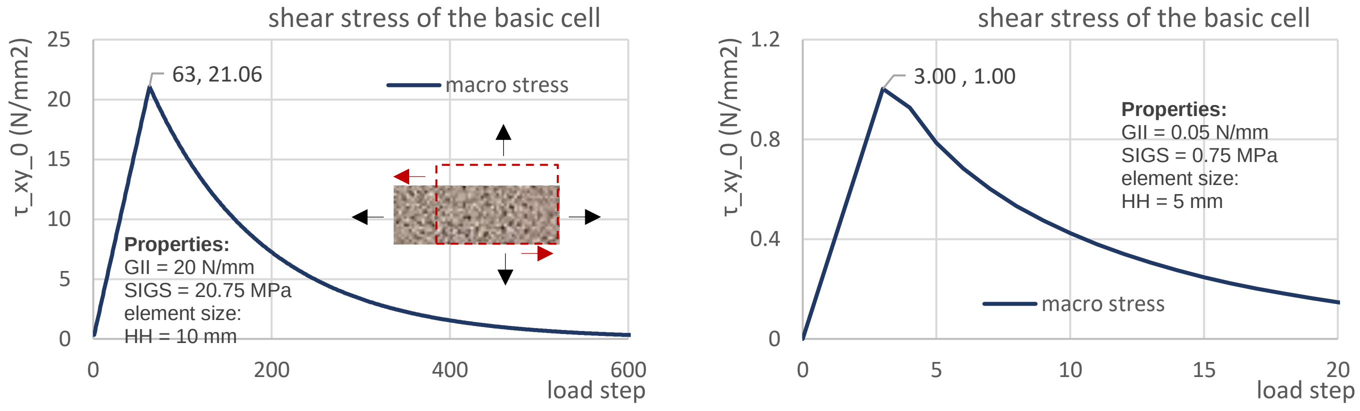


(a) small shear stiffness of the bed joint

(b) large shear stiffness of the bed joint

Figure 8-33 damaged stress of the homogenized cell τ_{xy}^0, cohesion c_h and friction stress $\sigma_{yy}^0 \cdot \tan{(\phi_u)}$ through load step

If the vertical load is set to be tension one, the shear stress of the homogenized cell τ_{xy}^0 could be obtained through the load step, as figure 8-34 shown.



(a) small shear stiffness of the bed joint

(b) large shear stiffness of the bed joint

Figure 8-34 damaged stress of the homogenized cell τ_{xy}^0 through load step when the vertical strain is the compression one

Figure 8-34 indicates that the shear sliding failure mode is introduced successfully as the homogenized cell is damaged once the shear stress of the bed joint reaches its strength.

8.2. Comparison against the experimental result

The experimental data of a set calcium silicate masonry specimens (vertical configuration) cast during the first period (September 2015) under vertical monotonic loading TUD_MAT-11b to TUD_MAT-11g, tested by R. Esposito, F. Messali and J. Rots in [41], are selected here. Note that the experimental results indicated only the vertical compression behaviour of the masonry specimens.

Model 4 constructed in MATLAB is used here to assess the accuracy of the algorithms introduced in chapters 6 and 7 and the analytical solutions of the material model. However, only the macro constitutive law in the vertical direction is assessed here as the experimental data is investigated when the masonry structure is loaded by pure vertical compression loading.

After that, the experimental tests are simulated in this project by implementing model 3 proposed in chapter 6 into the Diana FEA user-supplied subroutine. The comparison of the numerical and analytical results can judge if the model is introduced successfully in the finite element program. The comparison of the numerical and experimental data can assess the accuracy of material model 3 and judge if material model 3 can correctly simulate the specimens' compressive capacity, cracks pattern, failure mechanisms.

Test procedure

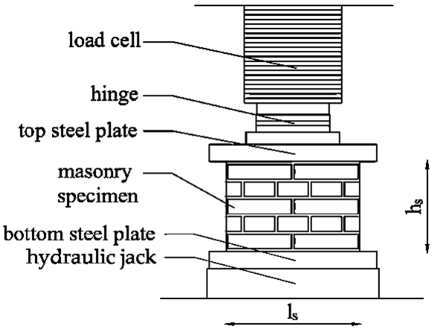


Figure 8-35 Compression test on masonry [41]

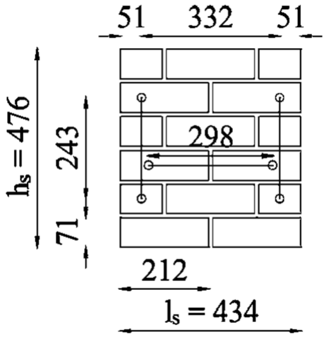


Figure 8-36 Position of LVDTs during the test [41]

Dimensions of this set calcium silicate masonry specimens selected from one of the wallets tested in the experiment are $434 \times 476 \times 102 - \mathrm{mm}$ ($2 \times 6 \times 1$-brick). There is a $10mm$-thickness layer of gypsum applied to the face of the loading plates, which ensures that the loaded faces of this specimen are levelled and parallel to one another. [41] The configuration of the testing apparatus could be seen in figure 8-35. A $3500kN$ hydraulic jack is put on the position at the bottom, which is provided by the testing apparatus and operated in displacement control. There is a steel plate lifted above this hydraulic jack, which is the active side. Meanwhile, there is a passive load plate lifted at the top side. A load cell using to measure the applied force is attached to the top steel plate. The possible eccentricities could be reduced during the loading by putting a hinge between the load cell and the top steel plate.

Vertical relative displacement over the height of the specimen is registered by four LVDTs[5] (linear variable differential transformers) attached to the specimen, see figure 8-36. The measuring range of these LVDTs is $2mm$ with an accuracy of 0.5%. The specimen was tested under monotonic loading, applied with a rate of $0.002mm/s$ to reach the peak stress in $15 - 30\ min$.

Material properties

The strengths of the calcium silicate blocks could be obtained from the experimental data, and the dimension of the bricks and the thickness of mortar joints could be obtained from the configuration of the specimen in [41] straight forward (see table 8-2).

The elastic modulus of the brick unit E_u is equal to $8990\ \mathrm{MPa}$, while the young's modulus of mortars is set to be the secant elastic modulus $E_2 = 5091\mathrm{MPa}$ of masonry subject to a compressive loading perpendicular to the bed joints, evaluated at 1/10 of the maximum stress. The Poisson ratio of the bricks ν_u is set to be the same as the masonry's, while the sum value of the mortars' Poisson ratio $\nu_{m,s}$ is assumed to be the difference value between the masonry's Poisson ratio during the first and second periods. The cross joint's Poisson ratio is zero as we assumed the cross joint deforms the springs in the vertical and horizontal directions. Therefore, the sum value of the mortars' Poisson ratio $\nu_{m,s} = 0.4$ incorporates the head and bed joint's ν_m. As a result, the mortar's Poisson ratio ν_m should be 0.2 as we assumed the head and bed joints have the same mechanical properties.

[5] LVDT: Linear variable differential transformer

Table 8-2 Material and geometrical properties from experimental data [41]

Material properties	Brick unit	Mortar	***Geometry***	Brick unit	Mortar
Young's Modulus E (MPa)	8990	5091	Height h (mm) (Half value only)	35.5	35.5
Poisson ratio ν $(-)$	0.14	0.02			
Tension strength σ_t (MPa)	2.74	2.79			
Compression strength σ_c (MPa)	16	6.59	Length l (mm) (Half value only)	106	106
I fracture energy G_I $(\mathrm{N/mm})$	0.081	0.082			
Compressive fracture energy G_c $(\mathrm{N/mm})$	20.96	17.68			
Shear strength σ_s (MPa)	-	0.14			
II fracture energy G_{II} $(\mathrm{N/mm})$	-	0.012	Thickness t (mm) (Half value only)	-	5
Friction angle ϕ (°)	23.37	23.37			
Dilatancy angle ψ (°)	10	10			

According to the formulations introduced by Maurizio Angelillo et al. in [39], the tensile (first), shear (second) and compressive fracture energy G_I, G_{II} and G_c of the bricks and mortar joints could be computed by equations (8.16) to (8.18).

$$G_I = 0.04 \cdot {\sigma_t}^{0.7} \tag{8.16}$$

$$G_{II} = 0.025 \cdot \left(\frac{\sigma_c}{10}\right)^{0.7} \tag{8.17}$$

$$G_c = 15 + 0.43 \cdot \sigma_c \cdot 0.0036 \cdot {\sigma_c}^2 \tag{8.18}$$

Where σ_t and σ_c are the tension and compression strength of the given component, respectively.

As the results discussed in section 8.1.3, the values of the components' dilatancy angles affect the compressive capacity and softening process. Therefore, the dilatancy angles should be suitably selected. The sensitivity of the dilatancy angle is studied to avoid overestimating the shear deformation with a large dilatancy angle. The macro strain-stress curves of the material model in the vertical direction with different material dilatancy angles could be investigated, as figure 8-37 shown.

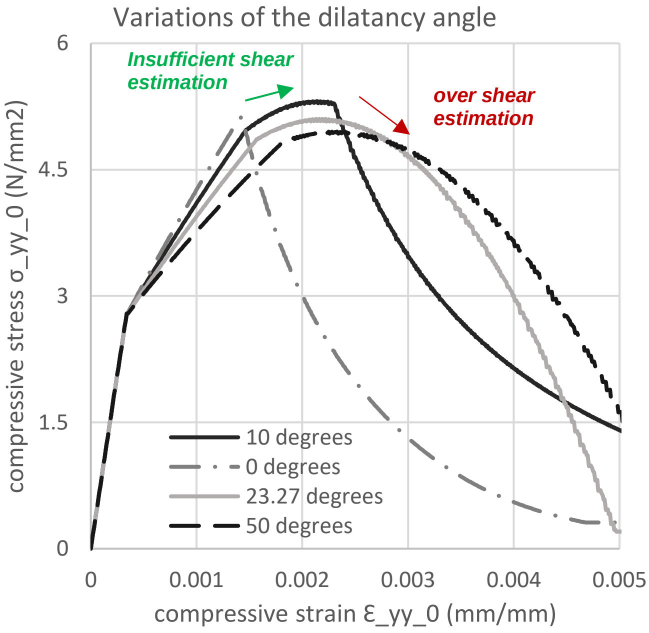


Figure 8-37 the vertical compressive stress σ_{yy}^0 versus compressive strain of the single element $\varepsilon_{yy,se}^0$ with different dilatancy angles of components

As shown in figure 8-37, the capacity of the material is underestimated if the dilatancy angle is very large or small. In this case, the dilatancy angles ψ of the bricks and mortars are assumed to be 10° as the most suitable values.

Many researchers suggest a dilatancy angle of 0 degrees to avoid over shear estimation as they did not consider the shear deformation and shear damage at the brick-mortar interface. In this model, the shear deformation and damage are considered, so components' dilatancy angles have to be suitably chosen.

Comparison: analytical solution versus experimental data

Note that the results computed in MATLAB are the analytical solutions for a single element, while the experimental data is obtained from a specimen modelled by a set of elements.

The vertical strain of the single element $\varepsilon^0_{yy,se}$ should be:

$$\varepsilon^0_{yy,se} = \frac{\Delta u}{H} \tag{8.19}$$

Where Δu is the loaded displacement in the vertical direction at each load step and H is element size, with a value of $50\ \mathrm{mm}$.

The vertical strain of the specimen $\varepsilon^0_{yy,LVDTs}$ evaluated by the LVDTs should be:

$$\varepsilon^0_{yy,LVDTs} = \frac{\Delta u}{L_{LVDTs}} \tag{8.20}$$

Where L_{LVDTs} is the vertical distance between the two LVDTs in the same row. As can be seen from figure 8-36, we can get $L_{LVDTs} = 243\ \mathrm{mm}$.

We can then get the relationship of $\varepsilon^0_{yy,se}$ and $\varepsilon^0_{yy,LVDTs}$ by combining equation (8.19) and (8.20):

$$\varepsilon^0_{yy,se} = \frac{\varepsilon^0_{yy,LVDTs} \cdot L_{LVDTs}}{H} = 4.86\ \varepsilon^0_{yy,LVDTs} \tag{8.21}$$

With this transformation function, we could compare the experimental result and the analytical solution proposed in section 8.1.4, as figure 8-37 shows, by applying geometrical and material properties shown in table 8-2. Note that there is only vertically compressive loading.

As shown in figure 8-38, the analytical results can predict the compressive capacity compared to the experimental results with a small error. The compressive capacity obtained from the test on average is $5.93\ \mathrm{MPa}$ [41], while the one obtained from the analytical solution is $5.23\ \mathrm{MPa}$. Therefore, the error between these two values should be:

$$error = \frac{5.93 - 5.23}{5.93} = 0.12$$

The error is acceptable and may be caused by the variability of mortars as the mortars' mechanical properties are dependent on manual actions.

To investigate if the material model could correctly predict the crack patterns, we should first focus on the cracking process of the specimen and then make the relative comparisons. The cracking process of specimen "TUD_MAT_11b" was recorded and stated in [41], see figure 8-39.

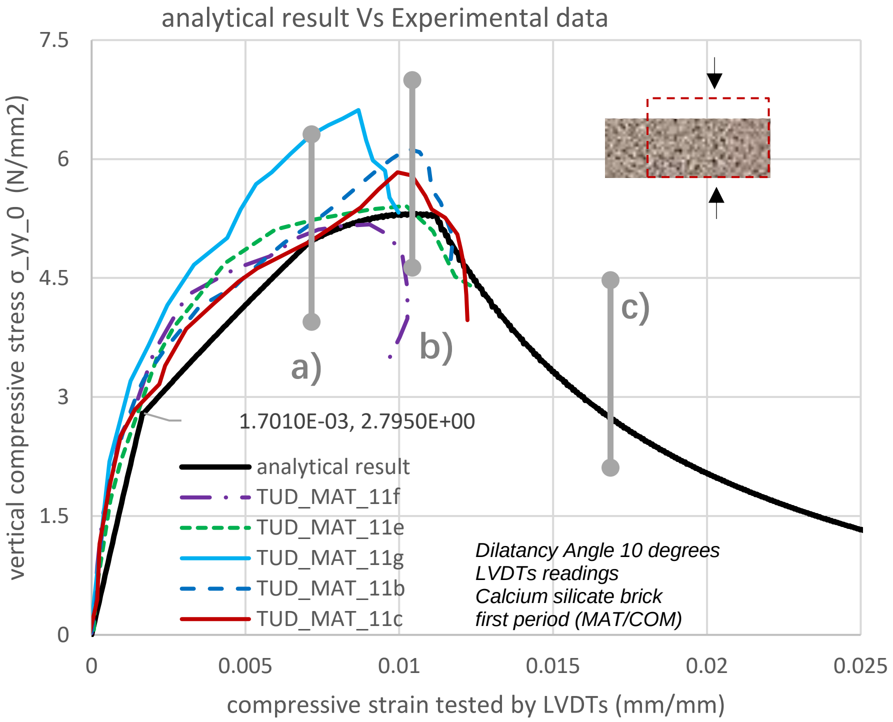
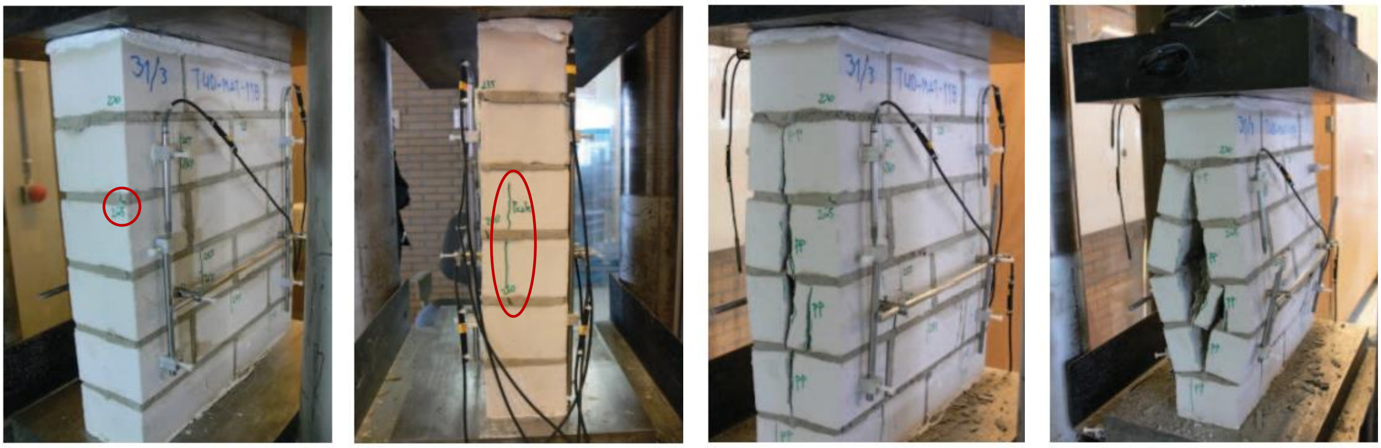


Figure 8-38 Comparison: vertical compressive stress σ_{yy}^{0} versus compressive strain tested by LVDTs $\varepsilon_{yy,LVDTs}^{0}$ from experimental results and the homogenized material model

According to the descriptions of the cracking process of the specimen “TUD_MAT_11b” stated in [41], the shear cracks should first develop at the interface of the brick unit and the bed joint, as figure 8-39 (a) shows. After that, the vertical cracks were generated at the central part of the specimen when the maximum stress was reached, see figure 8-39 (b). In the post-peak phase shown in figures 8-39 (c) and (d), the vertical cracks mainly occurred in the bricks and developed uniformly through the length of the specimen by splitting it into two parts.



(a) $\sigma_n = 5.3\ \mathrm{MPa}$ $\varepsilon_n = 7.4\ ‰$

(b) $\sigma_n = 6.2\ \mathrm{MPa}$ $\varepsilon_n = 11\ ‰$

(c) $\sigma_n = 2.0\ \mathrm{MPa}$

(d) $\sigma_n = 0.7\ \mathrm{MPa}$

Figure 8-39 Crack pattern of specimen TUD_MAT-11b tested under vertical compression test: (a) first crack; (b) maximum stress; (c)-(d) post-peak phase [41].

Note that the cracking phases shown in figures 8-39 (c) and (d) were not recorded by LVDTs reading as the LVDTs can not be attached to the specimen’s surface when the splitting developed throughout the thickness. However, phases (c) and (d) were tested by Jack’s reading. The vertical compressive strain-stress curves of the specimen “TUD_MAT_11d” tested by LVDTs and Jack’s reading were investigated as figure 8-40 shown [41].

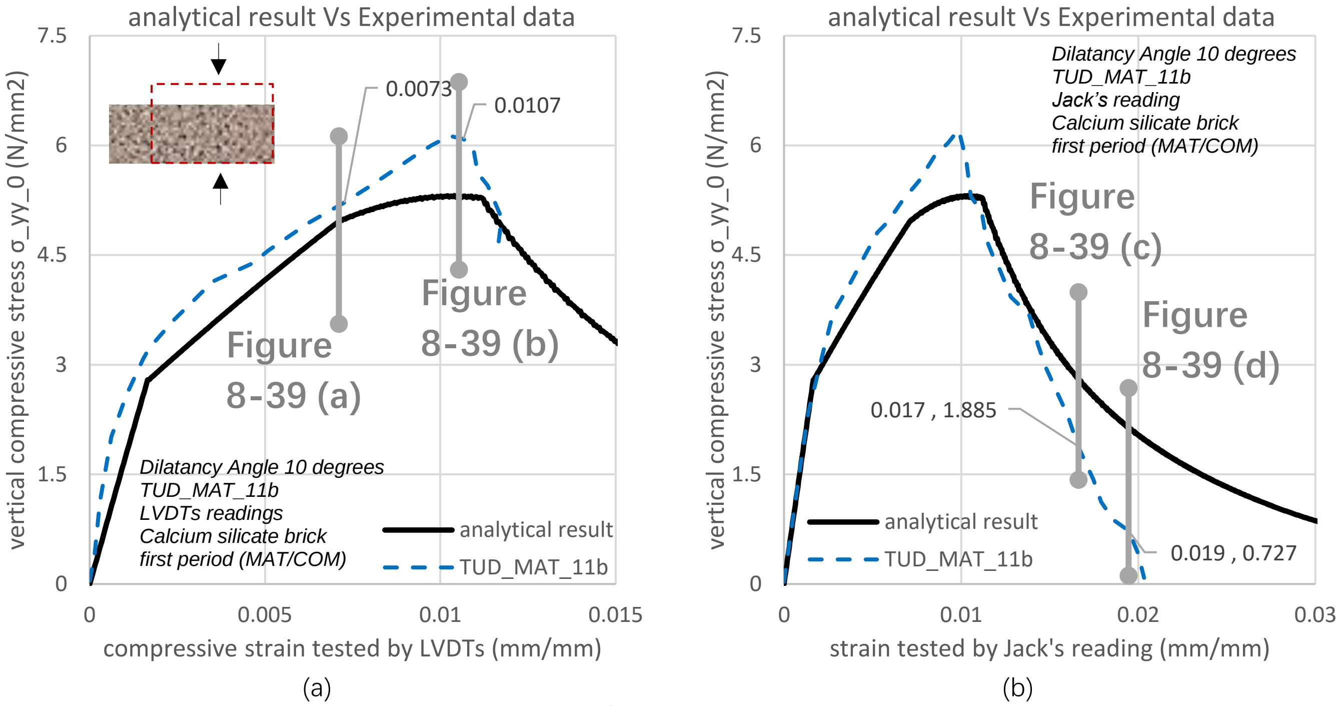


Figure 8-40 Comparisons: vertical compressive stress σ_{yy}^0 versus compressive strain from experimental result of specimen TUD_MAT_11b and the homogenized material model: (a) tested by LVDTs $\varepsilon_{yy,LVDTs}^0$; (b) tested by Jack’s reading $\varepsilon_{yy,Jack}^0$.

The cracking phases described in figures 8-39 (a) to (d) could be specified in figure 8-40 to discover the cracks predicted by the material model at which load step can reflect these phases. The load step n can be obtained as equation (8.22) shown (the load step size for the analytical solution is 0.00001).

$$n = \frac{\varepsilon_{yy,LVDTs}^0 \cdot L_{LVDTs}}{\varepsilon_{yy,se}^0 \cdot H} = 2.06 \times 10^4 \ \varepsilon_{yy,LVDTs}^0, \ \varepsilon_{yy,Jack}^0 \approx \varepsilon_{yy,LVDTs}^0 \tag{8.22}$$

As a result, the relative load steps in numerical calculations of the cracking phases described in figures 8-40 (a) to (c) can be obtained as table 8-3 shown.

Table 8-3 The relative load steps in numerical calculations of the cracking phases

Cracking phase	Vertical strain tested by $LVDTs$	Load step	Cracks
(a)	0.0074	153	Shear cracks at the brick-mortar interface
(b)	0.011	227	Vertical cracks in bricks
(c)	0.017	351	Splitting cracks in bricks
(d)	0.019	392	collapse

Components’ damage state variables $r_i \ (i = u, h, b, c)$ can be investigated through the loading history to study if the cracks generate in each component, see figure 8-41. The cracks are developed when the r_i is reduced from 1 to 0.

As figure 8-41 shows, the shear crack develops in the bed joint at step 139, corresponding to the experimental result. After that, the head joint is cracked at step 203 as its equivalent splitting tensile stress reaches the material tensile strength. At load step 228, the brick unit is damaged in splitting tension, producing a vertical tensile crack, which could also be investigated from the tests. Therefore, it can be said that the material model could simulate all potential cracks of the unit cell in this typical case.

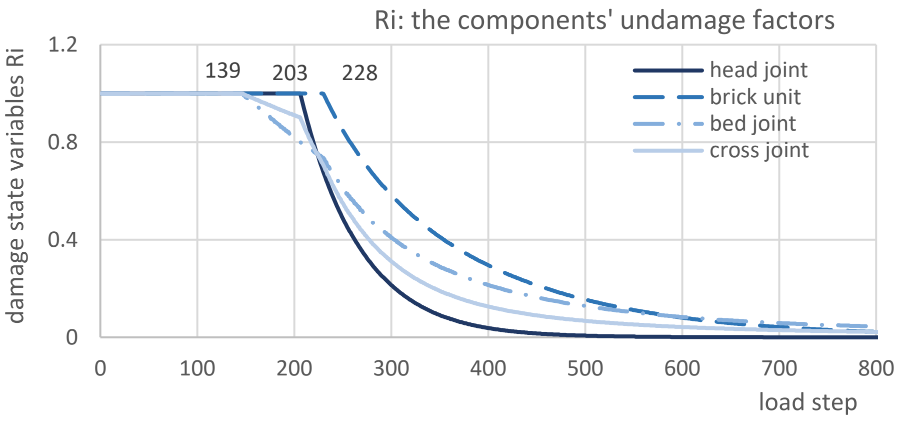
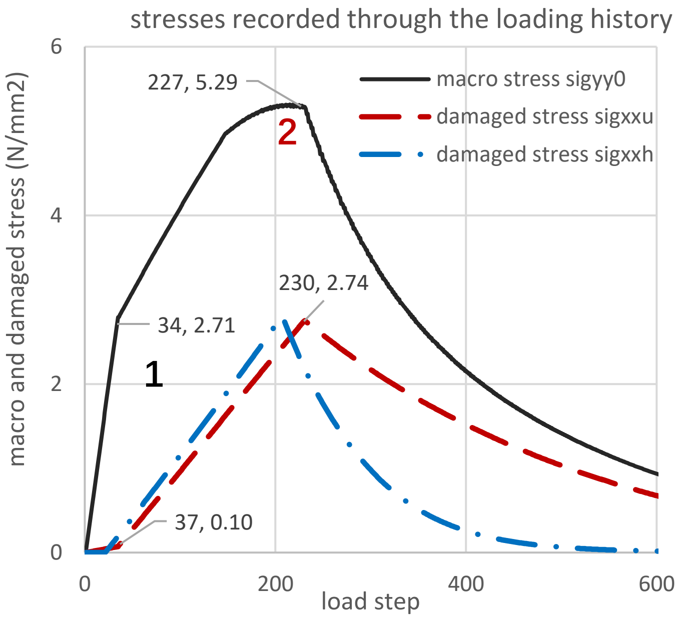


Figure 8-41 Damage state variables of components $r_i = 1 - d_i$ $(i = u, h, b, c)$ versus load step

The damaged internal stresses of the brick unit and head joint could be recorded through the load path, as figure 8-42 shown. These curves can be used to study if the failure mechanisms simulated by the material model is reasonable.

The results in figure 8-42 indicate the failure mechanisms simulated by the material model. The hardening phase starts from step 37 (point 1 in figure 8-42) as the homogenized stiffness of the material model becomes smaller. Description for this phase in the report is: “the nonlinearity occurred at a stress level approximatively of 1/10 of the maximum stress”.

However, the stress at point 1, where the nonlinearity occurred, is $2.71\ \mathrm{MPa}$, about half of the maximum stress rather than 1/10. Note that the analytical solution solved by MATLAB is only a mathematic result, where the numerical integrations are not considered in the integration points. In comparison, the recorded stress from the tests is the average value of the specimen. As a result, differences in the simulations of the hardening process appear.



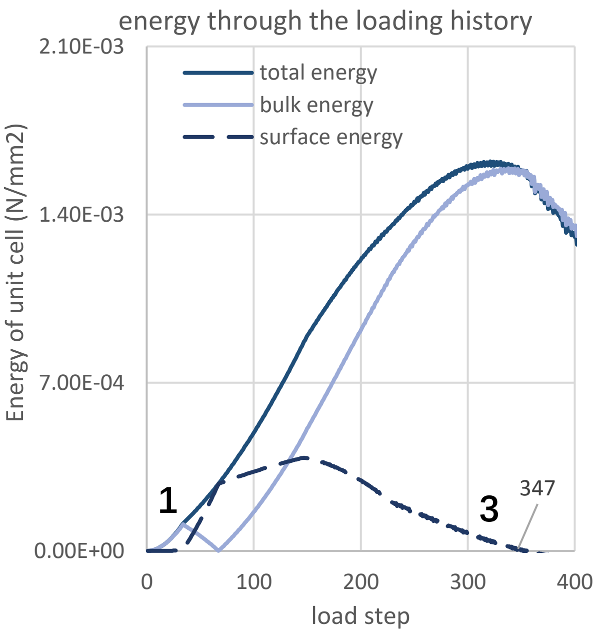


Figure 8-42 the macro stress σ_{yy}^{0}, damaged horizontal stresses of the brick unit σ_{yy}^{u} and head joint σ_{yy}^{h} versus load step

Figure 8-43 the total energy, bulk energy and surface energy of the unit cell through the load path

At point 2, the maximum stress is reached once the brick unit is damaged in splitting tension, which is correlated with the simulations of the crack pattern. After that, an exponential softening branch is investigated from the analytical calculations rather than the linear one reported in [41]. The main reason is that we assume an exponential softening process for the components rather than the linear one in this work.

The total energy and bulk energy can be computed by the same method introduced in section 8.1.3. As a result, the energy status of the unit cell through the loading history could be recorded as figure 8-43 shown. The surface energy has a value once the nonlinearity occurs, indicated in figure 8-42, as the surface energy has a value at point 1. At step 347 (point 3 in figure 8-43), the surface energy drops to zero, indicating that the compressive splitting effects disappear once the vertical splitting cracks split the specimens.

In conclusion, the comparisons between the analytical and experimental results are stated in Table 8-4.

Table 8-4 The comparisons between the analytical results and experimental results

	The experimental results	The analytical results	Agreement
Compressive capacity	5.93 MPa	5.23 MPa	Acceptable
Crack pattern	Cracks started at the mortar-brick interface for the joints orthogonal to the loading direction [41]. The vertical cracks mainly occurred in the central part of the specimens when the maximum stress was reached. In the post-peak phase, the vertical cracks mainly occurred in the bricks and developed uniformly through the length of the specimen by splitting it into two parts.	The micro-cracks were vertically generated in the bricks and mortars after load step 37 due to the splitting effects. Shear cracks appeared in the horizontal joints orthogonal to the loading direction at load step 139. The vertical tensile crack occurred in the head joint at step 203. When the maximum stress was reached at step 227, the vertical tensile crack occurred in the bricks.	Acceptable
Failure mechanisms	The pre-peak stage was characterized by linear-elastic followed by a hardening behaviour until the peak. In this stage, the nonlinearity occurred at a stress level approximatively of 1/10 of the maximum stress [41]. After the maximum stress was reached, a softening behaviour was observed. The softening branch was approximatively linear.	The hardening phase starts from step 37, where the vertical stress is 2.71 MPa. This hardening phase ends at step 227 as the equivalent splitting failure occurs in the brick unit. After that, an exponential softening process is investigated.	Partially acceptable
Energy dissipation	-	Surface energy starts have value at load step 37. Surface energy drops to zero at step 347.	Acceptable

9. Conclusions and recommendations

Although the macro models used to simulate masonry represent a good compromise between accuracy and efficiency, most cannot precisely identify the localized damages. Therefore, an alternative homogenized model developed based on Zucchini and Lourenço’s micro-mechanical models is proposed in this thesis. Zucchini and Lourenco’s work is characterized by an acceptable trade-off between accuracy and computational efforts. However, their material model is based on a complex algorithm that requires a significant computational time to run. Moreover, their work did not consider the following cases: shear damage in the bed joint occurring before other damages, the components’ compressive splitting effects and the hardening phase.

Therefore, a research question can be stated as:

Is that possible to define a homogenized constitutive model for masonry structures under in-plane loading that will consider shearing, tensile cracking, crushing and splitting failures based on a micro-mechanical approach with as few as possible computational efforts?

To be more specific, this research question can be described by three sub-questions:

(1) How to simplify the deformed mechanisms of Zucchini and Lourenço’s model when coupled behaviours in all directions are considered.
(2) How to consider the phenomena that the horizontal shear crack potentially generated in bed joint may appear before the dissipation of compressive energy.
(3) How to implement the new plastic model, in which the hardening phase and compressive splitting effects are considered with less computational costs.

To reply to the research question, the basic unit cell is first defined. Each cell consists of two-quarters of bricks connected through a bed joint, and each one is connected to a head joint on one side. Cracking, crushing and shearing failure are included through four material models. Model 1 (presented in chapter 4) describes shear sliding, and model 2 (presented in chapter 5) describes cracking under horizontal tension. Model 3 (presented in chapter 6) describes compressive crushing and splitting, whereas model 4 (presented in chapter 7) couples the aforementioned failure mechanisms in one final model. The models’ analytical and numerical results shown in chapter 8 for models 1 and 2 indicate that the material model proposed in this work can analytically and numerically simulate the behaviours of bricks and mortars. The comparison of analytical solution and experimental results for model 3 is investigated, indicating that the Drucker-Prager yield criteria and the bi-parabolic hardening law can still be accurately implemented by applying the explicit method instead of the implicit one the load step is small enough. The results in chapter 8 for model 4 indicate that the material model proposed can still analytically predict as many localized damages as Zucchini and Lourenço’s model could if the head joint’s shear behaviour is neglected.

Several new assumptions on deformed mechanisms, components’ failure modes, and elastoplastic behaviours are made in this thesis. An alternative constitutive model for masonry structures under in-plane loading is successfully proposed based on Zucchini and Lourenço’s research [20, 21, 30, 32], where the shearing, tensile cracking, crushing and splitting effects are coupling. This alternative constitutive model maintains the accuracy of the model suggested by Zucchini and Lourenço but reduce the computational efforts. Therefore, the answer to the main research question should indeed be “yes, it is possible”.

In conclusion, the works presented in this thesis can be summarized by the answer to the sub-questions:

(1) The material model for masonry’s coupled behaviour in normal directions is simulated by combining the quarter unit cell’s horizontal tensile and vertical compressive behaviours in a simple manner. This simulation is achieved by neglecting the head joint’s shear deformation so that the number of equations needed to be derived at brick-mortar interfaces could be reduced. As a result, also the numerical efforts are reduced.

(2) In the coupled material model, the bed joint is always damaged before any other damage occurs. When the bed joint is damaged in shear, it moves with the brick unit as a whole. In other words, the shear stress at the horizontal brick-mortar interface drops to zero once the horizontal shear crack is generated at the middle of the bed joint's thickness. As a result, the constitutive model's shear strength is dependent on the values of the head joint's cohesion, vertical compressive loading and the friction angle of the mortars.

(3) The compressive splitting effects are implemented into the final model by introducing the Drucker-Prager yield criterion in a 2D plane. This implementation requires more numerical efforts if we use the implicit method, as several quadratic algebraic equations are needed to be solved to compute the plastic multipliers for every component. Therefore, an explicit algorithm with a bi-parabolic hardening diagram is introduced to incorporate the splitting effects with less computational cost.

9.1. Differences compared to previous works

The work presented in this thesis is very close to the research carried out by Zucchini and Lourenço. Both aims are to derive a final homogenized material model for masonry with a completed stiffness matrix based on the compatibility and equilibrium equations of a representative unit cell. Therefore, the four differences between the final material models proposed in this thesis and Zucchini and Lourenço's works [21] are highlighted in the following.

(i) Two damage parameters define the damage status of bricks and head joints.

The brick unit's and head joint's damage status is controlled by two damage state variables in this final material model. One is obtained based on the deformed mechanisms of the unit cell under pure horizontal tension behaviour, and the other is obtained according to the deformed mechanisms of the unit cell under pure vertical compressive behaviour. These two damage variables are coupled together by incorporating the transverse strain produced by Poisson's effects into the axial one.

From the physical aspect, the brick unit (or the head joint) may first have a vertical tensile crack at the middle of its length, and then the micro-cracks can still be generated in these half bodies if the brick's (or the head joint's) splitting tensile stress does not reach the material tensile strength of its compressive energy is not dissipated.

Vice versa, the brick unit's and head joint's damage status is dependent on only one damage state variable in Zucchini and Lourenço's work [21].

(ii) Shear damage of the bed joints occurs before any other failure mechanisms.

As can be seen from the compatibility and equilibrium equations of the representative plane, the head joint's horizontal stress is influenced by the bed joint's shear stress. Meanwhile, the horizontal behaviour of the bed joint should be correlated with the cross joint as the bed joint is connected with the cross joint.

In Zucchini and Lourenço's works in [21], the bed joint's compressive energy is assumed to be consumed before the horizontal shear crack can occur. Therefore, the shear stress at the horizontal brick-mortar interface kept increasing the head joint's horizontal stress. This shear stress dropped to zero until the bed joint's cohesion and compressive energy were consumed.

In this thesis, a different failure mechanism of the bed joint is assumed. The shear crack is assumed to appear in the bed joint before all other cracks if the masonry is not loaded by vertical tension. This assumption is made based on two facts: first, the bed joint's shear strength is generally smaller than the material compressive strength. Second, the bed joint's tension behaviour is not such essential compared to its shear behaviour. As mentioned above, the bed joint's horizontal behaviour is influenced by the cross joint's. The cross joint's horizontal behaviour is not such important as the volume of the cross joint is very small compared to other components. As a result, the head joint's horizontal stress can be assumed to be no longer influenced by shear behaviour at the brick-mortar interface once the bed joint is fractured in shear.

(iii) The compressive splitting effects are incorporated into the final material model.

Masonry is a composite quasi-brittle material as its components are brittle. Therefore, the tensile crack should develop rapidly in the bricks and mortars when imposed on axial tension forces. However, the micro-cracks are gradually generated when the masonry is loaded by axial compression. These micro-cracks change the components' capacity, making the material yield surface expanding or shrinking. As a result, the plastic deformations in the lateral directions develop once the yield surface expands due to the axial compressive loading, leading to splitting effects. The Drucker-Prager yield surface can be used to find plastic strains in all directions. Therefore, the Drucker-Prager yield criterion is adopted in the components to describe the splitting effects. Such yield criterion is smooth at every corner, making it possible to describe the direction of potential energy or yield surface by a single direction vector for the whole yield surface.

If the implicit return-mapping back algorithm is still used to derive the plastic model with the Drucker-Prager yield criterion, it will take a long time to compute each component's plastic strain tensor as three quadratic algebraic equations need to be solved to compute the three plastic multipliers for the brick unit, head joint and bed joint. In Zucchini and Lourenço's extended unit cell, more quadratic algebraic equations are needed to be solved as more components are incorporated into the extended unit cell. Therefore, Zucchini and Lourenço neglected the splitting effects and assumed that the shear flow only depends on axial or lateral plastic behaviour to save computational time by implementing the Mohr-coulomb yield criterion in the σ-τ plane instead of the Drucker-Prager yield criterion [21].

On the opposite, the splitting effects are considered in this work, and two things were done to save computational time. The first thing is that the explicit Euler-forward method, where the plastic correctors for each component can be computed from an explicit equation based on Prager's consistency equation, is applied. The second one is that the assumptions on deformed mechanisms make it is possible to derive the final material model based on the quarter unit cell.

(iv) The plastic model is introduced by applying an explicit Euler-forward algorithm.

The implicit Euler-backward method guarantees a return to the yield surface every step. Therefore, it has higher robustness and accuracy, especially when the load step size is large. However, it takes time to run if the implicit algorithm is applied to interpret the plastic model for such a complex model. Therefore, an explicit method is adopted in this work. The accuracy of the plastic model is guaranteed by applying a small loading step. However, this simplification leads to the limitation when the model is implemented in nonlinear finite element programs, which is discussed in sections 9.2 and 9.3.

9.2. Limitations

The material model has some limitations, which are described below.

First of all, the micro-mechanical material can only be used to model the masonry structures with staggered alignment bricks. The compatibility and equilibrium equations of the model have to be rederived if it is used to model the masonry structures with other types of brick arrangements, such as the stack bond pattern, as all of the equations are derived based on the deformed mechanisms of the representative unit cell.

Furthermore, the use of the explicit return-mapping algorithm requires a check of the accuracy of the results. Although the forward Euler method provides possibilities for the plastic strain tensor being computed with fewer computation efforts, the algorithm's error may be significant if the relatively large load step is applied. Apparently, the Euler-forward algorithm does not guarantee a rigorous return to the yield surface, which causes an error with a magnitude depending on the curvature of the yield surface. As a strongly curved corner is met, the directions of the yield surface and potential energy will change rapidly from the previous load step, leading to a large error as the Euler forward method solves the quadratic problem by a first-order method. If the relative load step is set to be large, the error accumulations may become significant, leading to numerical instability of the algorithm or scarce accuracy of the predictions. As a result, the material model may become unstable and collapse if the step size is not suitably chosen. That introduces some limitations regarding the step size of the numerical analyses.

Additionally, the coupled material model did not incorporate the horizontal compressive behaviour. The homogenized stress in the horizontal direction is assumed to be zero if the external strain in the horizontal direction is set to be negative. However, the micro-cracks should be generated in the components if the masonry is loaded by horizontal compression. The components may fail once the horizontal splitting cracks occur. Therefore, there should be compressive capacity in the horizontal direction as well.

9.3. Future works

As the instability of the explicit algorithm is mentioned above, it is hard to say that a stable material model is obtained. Although the analytical results of this material model look acceptable from its physical meaning and are correlated with some experimental results in the material scale, the model has not been validated against experimental results on the structural level. Therefore, future studies could focus on the validations of the material model's robustness and accuracy. For instance, the final model could be implemented in a finite element program to assess the model's robustness and accuracy, especially at the structural level.

Furthermore, the sub-stepping techniques could be adopted and connected with the Euler-forward method to improve the stability of the analyses. In the sub-stepping technique, the strain increment tensor $\Delta\boldsymbol{\varepsilon}$ is divided into numbers of strain sub-increment tensor $d\boldsymbol{\varepsilon}$ at every load step. As the explicit algorithm is applied to interpret the model's plastic behaviours in this material model, it will be convenient to introduce this strain sub-increment tensor $d\boldsymbol{\varepsilon}$ into the explicit functions of the plastic multipliers. As a result, the error could be reduced, especially when the strongly curved yield surface is met.

As the sub-stepping techniques could reduce the error produced due to the strong curvature of the yield surface, the extended Drucker-Prager yield surface with the sub-stepping techniques can be introduced to solve the apex problem of the Drucker-Prager yield criterion.

Additionally, some other points could be studied in the future as follows:

- The variations of the friction and dilatancy angles could be studied and implemented into the model. In this material model, only the cohesion of the components could vary during the hardening or softening process. However, other parameters could also vary following the hardening or softening process.

- The horizontal compressive behaviour of the masonry was not taken into account in the material model. It could be studied in future research and incorporated into the material model.

- As shown in [41], the masonry structures with stiff brick units are cracking throughout the thickness during the post-peak under compressive loading. Therefore, the material model could be derived in a 3-dimension space to introduce splitting effects in the thickness direction.

Bibliography

[1] D'Altri A M, Sarhosis V, Milani G, et al., "Modeling strategies for the computational analysis of unreinforced masonry structures: review and classification," *Archives of computational methods in engineering,* pp. 1-33, 2019.

[2] A.T. Vermeltffort, D.R.W.Martens, and G.P.A.G.van Zijl, "Brick-mortar interface effects on masonry under compression," *Canadian Journal of Civil Engineering,* pp. 1475-1485, 2007.

[3] P. A. W, "The biaxial compressive strength of brick masonry," *Proceedings of the Institution of Civil Engineers,* vol. 71, no. 3, pp. 893-906, 1981.

[4] Leuchars J M, Scrivener J C, "Masonry infill panels subjected to cyclic in-plane loading," *Bulletin of the New Zealand Society for Earthquake Engineering,* vol. 9, no. 2, pp. 122-131, 1976.

[5] H. H. K, "Investigation into the failure mechanisms of brick masonry loaded in axial compression," *FH Johnson (ed): Designing, engineering and constructing with masonry products,* pp. 34-41, 1969.

[6] A. W. Page, "Finite element model for masonry," *Journal of the Structural Division,* vol. 104, no. 8, pp. 1267-1285, 1978.

[7] Lourenço P B, Rots J, "Analysis of masonry structures with interface elements," Rep. No. 03-21-22-0, 1994.

[8] P. B. Lourenço, "Computational strategies for masonry structures," Delft University Press, Delft, 1996.

[9] R. J. G, "Numerical simulation of cracking in structural masonry," *Heron,* vol. 36, no. 2, pp. 49-63, 1991.

[10] Paulo B. Lourenço, Jan G. Rots, "Multisurface Interface Model for analysis of Masonry Structure," *J. Eng. Mech.,* pp. 660-668, 1997.

[11] Oliveira D V, Lourenço P B, "Implementation and validation of a constitutive model for the cyclic behaviour of interface elements," *Computers & structures,* vol. 82, no. 17-19, pp. 1451-1461, 2004.

[12] Malomo D, Pinho R, Penna A, "Using the applied element method for modelling calcium silicate brick masonry subjected to in-plane cyclic loading," *Earthquake Engineering & Structural Dynamics,* vol. 47, no. 7, pp. 1610-1630, 2018.

[13] Kuang J S, Yuen Y P, "Simulations of masonry-infilled reinforced concrete frame failure," *Proceedings of the Institution of Civil Engineers-Engineering and Computational Mechanics,* vol. 166, no. 4, pp. 179-193, 2013.

[14] Miglietta P C, Bentz E C, Grasselli G, " Finite/discrete element modelling of reversed cyclic tests on unreinforced masonry structures," *Engineering Structures,* vol. 138, pp. 159-169, 2017.

[15] Ali S S, Page A W, "Finite element model for masonry subjected to concentrated loads," *Journal of structural engineering,* vol. 114, no. 8, pp. 1761-1784, 1988.

[16] Petracca M, Pelà L, Rossi R, et al., "Micro-scale continuous and discrete numerical models for nonlinear analysis of masonry shear walls," *Construction and Building Materials,* vol. 149, pp. 296-314, 2017.

[17] D. P. G, "Constitutive equation and compatibility of the external loads for linear elastic masonry-like materials," *Meccanica,* vol. 24, no. 3, pp. 150-162, 1989.

[18] Maier G, Nappi A, " A theory of no-tension discretized structural systems," *Engineering structures,* vol. 12, no. 4, pp. 227-234, 1990.

[19] A. M, "A finite element approach to the study of no-tension structures," *Finite elements in analysis and design,* vol. 17, no. 1, pp. 57-73, 1994.

[20] A. Zucchini, P.B. Lourenço, "A coupled homogenisation–damage model," *Computers and Structures,* p. 917–929, 2004.

[21] A. Zucchini, P.B. Lourenço, "A micro-mechanical homogenisation model for masonry: Application to shear walls," *International Journal of Solids and Structures,* vol. 46, no. 3-4, pp. 871-886, 2009.

[22] Leonetti L, Greco F, Trovalusci P, et al., "A multiscale damage analysis of periodic composites using a couple-stress/Cauchy multidomain model: Application to masonry structures," *Composites Part B: Engineering,* vol. 141, pp. 50-59, 2018.

[23] Luís C. Silva, Paulo B. Lourenço, Gabriele Milani, "Numerical homogenization-based seismic assessment of an English-bond masonry prototype: Structural level application," *Earthquake Engineering & Structural Dynamics,* vol. 49, no. 9, pp. 841-862, 2020.

[24] Quagliarini E, Maracchini G, Clementi F, “Uses and limits of the Equivalent Frame Model on existing unreinforced masonry buildings for assessing their seismic risk: A review,” *Journal of Building Engineering,* vol. 10, pp. 166-182, 2017.

[25] D. M, “Schematizzazione e modellazione degli edifici in muratura soggetti ad azioni sismiche,” *Industria delle costruzioni,* vol. 25, no. 242, pp. 44-57, 1991.

[26] Block P, Lachauer L, “Three-dimensional (3D) equilibrium analysis of gothic masonry vaults,” *International Journal of Architectural Heritage,* vol. 8, no. 3, pp. 312-335, 2014.

[27] Chiozzi A, Milani G, Grillanda N, et al., “ A fast and general upper-bound limit analysis approach for out-of-plane loaded masonry walls,” *Meccanica,* vol. 53, no. 7, pp. 1875-1898, 2018.

[28] Lopez J, Oller S, Oñate E, et al., “A homogeneous constitutive model for masonry,” *International journal for numerical methods in engineering,* vol. 46, no. 10, pp. 1651-1671, 1999.

[29] A. A, “Derivation of the in-plane elastic characteristics of masonry through homogenization theory,” *International journal of solids and structures,* vol. 32, no. 2, pp. 137-163, 1995.

[30] A. Zucchini, P.B. Lourenço, “A micro-mechanical model for the homogenisation of masonry,” *International Journal of Solids and Structures 39,* p. 3233–3255, 2002.

[31] J. Oliver, M. Cervera, S. Oller and J. Lubliner, “Isotropic Damage Models And Smeared Crack Analysis of Concrete,” *Proc 2nd ICCAADS,* vol. 2, pp. 945-958, 1990.

[32] A. Zucchini, P.B. Lourenço, “Mechanics of masonry in compression: results from a homogenisation approach,” *Computers and Structures,* vol. 85, pp. 193-204, 2007.

[33] Dimitris Theodossopoulos, Braj Sinha, “A review of analytical methods in the current design processes and assessment,” *Construction and Building Materials,* pp. 990-1001, 2013.

[34] Prof. dr. ir. R.de Borst and Prof. dr. ir. L. J. Sluys, “Basic Notions in Elasto-Plasticity,” in *Computational Methods in Non-linear Solid Mechanics,* Delft, Delft University of Technology, 2015, pp. 77-96.

[35] Paulo B. Lourenc¸o, J. Pina-Henriques, “Validation of analytical and continuum numerical methods,” *Computers and Structures 84,* p. 1977–1989, 2006.

[36] M. Salamon, “Elastic moduli of a stratified rock mass,” *International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts,* vol. 5, no. 6, pp. 519-527, 1968.

[37] Prof. dr. ir. R.de Borst and Prof. dr. ir. L. J. Sluys, “Computational Elasto-plasticity,” in *Computational Methods in Non-linear Solid Mechanics,* Delft, Delft University of Technology, 2015, pp. 103-110.

[38] D. F. bv, “DIANA Documentation release 10.3,” 1 March 2019. [Online]. Available: https://dianafea.com/manuals/d93/Analys/node375.html.

[39] Maurizio Angelillo, Paulo B. Lourenço and Gabriele Milani, “Masonry Behaviour and Modelling,” *Mechanics of masonry structures,* pp. 1-26, 2014.

[40] Jonna Manie and Wijtze Pieter Kikstra, “Element Library,” in *Diana Finite Element Analysis User's Manual,* TNO DIANA BV, Release 9.5.

[41] Rita Esposito, Francesco Messali, Jan Rots, “TESTS FOR THE CHARACTERIZATION OF REPLICATED MASONRY AND WALL TIES: Physical Testing and Modelling – Masonry Structures,” Delft University of Technology, Delft, 2016.

Appendix A: MATLAB Code (Model 2)

```
clear all;
% properties setting:
% internal stress selection: SIGXXU(DU),SIGXXH(DH),TAUXYB(DB),SIGXXC(DC)
E = 1000; C1 = 5;
EU = C1*E; EB = E; EH = E; EC = E;
V = 0.2;
% shear modulus of mortar
GB = E/(2*(1+V));
% I and II fracture energy
GI = 0.01; GII = 0.05;
% properties of masonry: L=C2*T, H=C3*T
C2 = 12; C3 = 2;
% initial value of external strain and damage factor:
EPSXX0 = 0;
DH = 0; DU = 0; DB = 0; DC = 0;
% Shear and tension strength:
% Tension strength of brick unit
SIGTU = 1.3;
% Tension strength of mortar
SIGTM = 0.5;
% Shear strength of mortar
SIGS = 0.75;
% element size
% ATM must be positive, check maximum mesh size: HH < 80
% ATU must be positive, check maximum mesh size: HH < 59
% AS must be positive, check maximum mesh size: HH <74
HH = 50;
a = [];
b = [];
c = [];
d = [];
e = [];
% Tolerance of calculted and assumed damge factor
TOR = 0.00001;
% outer loop: strain interation
for i = 1:500
    EPSXX0 = EPSXX0 + 0.00001;
    % inner loop: verification of damage factor
    while DH <= 1 & DU <= 1 & DC <= 1 & DB <= 1
        % undamage factor of each component
        RH = 1-DH;
        RB = 1-DB;
        RC = 1-DC;
        RU = 1-DU;
        % tension stress in head joint
        SIGXXH = -0.4e1 * (C2 + 1) * E * EPSXX0 * (-(GB * RB * (V - 1) * (V + 1) * C2) / 0.4e1 + GB * RB * ((V
^ 2) / 0.4e1 - 0.1e1 / 0.4e1) + C1 * C3 * E * RU) * (C1 * C2 * RU + RH) / (C1 * GB * RB * RU * (V - 1) * (V +
1) * (C2 ^ 3) - 0.4e1 * (GB * RB * (-(V ^ 2) / 0.4e1 + 0.1e1 / 0.4e1) + C1 * C3 * E * RU) * RH * (C2 ^ 2) + (-C1
* GB * RB * RU * (V - 1) * (V + 1) + 0.4e1 * E * C3 * (RU ^ 2 * (V ^ 2 - 1) * C1 ^ 2 - 0.2e1 * C1 * RH * RU * (V
^ 2) + RH ^ 2 * (V ^ 2 - 1))) * C2 - 0.4e1 * (GB * RB * ((V ^ 2) / 0.4e1 - 0.1e1 / 0.4e1) + C1 * C3 * E * RU) *
RH);
        % tension stress in brick unit
        SIGXXU = -0.4e1 * (C2 + 1) * C1 * E * (-(GB * RB * (V - 1) * (V + 1) * C2) / 0.4e1 + GB * RB * ((V ^ 2) /
0.4e1 - 0.1e1 / 0.4e1) + C3 * E * RH) * EPSXX0 * ((C1 * C2 * RU) + RH) / ((C1 * GB * RB * RU * (V - 1) * (V
+ 1) * C2 ^ 3) - 0.4e1 * (GB * RB * (-(V ^ 2) / 0.4e1 + 0.1e1 / 0.4e1) + C1 * C3 * E * RU) * RH * (C2 ^ 2) + (-
(C1 * GB * RB * RU * (V - 1) * (V + 1)) + 0.4e1 * E * C3 * ((RU ^ 2 * (V ^ 2 - 1) * C1 ^ 2) - 0.2e1 * C1 * RH *
RU * (V ^ 2) + RH ^ 2 * (V ^ 2 - 1))) * C2 - 0.4e1 * (GB * RB * ((V ^ 2) / 0.4e1 - 0.1e1 / 0.4e1) + C1 * C3 * E *
RU) * RH);
    % tension stress in cross joint
    SIGXXC = E * EPSXX0 * ((RU * C1 * GB * RB ^ 2 * RC * (V - 1) * (V + 1) * C2 ^ 3) / 0.4e1 - RH * RB *
((-(V ^ 2 * GB) / 0.4e1 + GB / 0.4e1) * RB + (C1 * C3 * E * RU)) * RC * (C2 ^ 2) + (-(RU * C1 * GB * (V - 1) *
(V + 1) * RB ^ 2) / 0.4e1 + E * C3 * ((RU ^ 2 * (V ^ 2 - 1) * C1 ^ 2) - 0.2e1 * C1 * RH * RU * (V ^ 2) + RH ^ 2 *
(V ^ 2 - 1)) * RB + E * RU * (V ^ 2) * C1 * C3 * RH * ((RU * C1) - RH)) * RC * C2 + 0.2e1 * (-(GB * RC * (V -
1) * (V + 1) * RB ^ 2) / 0.8e1 + E * RU * C1 * C3 * (-RC / 0.2e1 + (C1 * RU * V ^ 2) - RH * (V ^ 2)) * RB - E *
RU * (V ^ 2) * C1 * C3 * RC * ((RU * C1) - RH) / 0.2e1) * RH) * (C2 + 1) / RC / ((C1 * GB * RB * RU * (V - 1)
* (V + 1) * C2 ^ 3) / 0.4e1 - RH * ((-(V ^ 2 * GB) / 0.4e1 + GB / 0.4e1) * RB + (C1 * C3 * E * RU)) * (C2 ^ 2) +
(-(C1 * GB * RB * RU * (V - 1) * (V + 1)) / 0.4e1 + E * C3 * ((RU ^ 2 * (V ^ 2 - 1) * C1 ^ 2) - 0.2e1 * C1 * RH *
RU * (V ^ 2) + RH ^ 2 * (V ^ 2 - 1))) * C2 - RH * (((V ^ 2 * GB) / 0.4e1 - GB / 0.4e1) * RB + (C1 * C3 * E * RU)))
/ (RC * C2 + 2 * RB - RC);
    % shear stress between bed joint and brick unit
    TAUXYB = 0.2e1 * (V - 1) * (V + 1) * ((C1 * C2 * RU) + RH) * C3 * E * EPSXX0 * (C2 + 1) * ((RU * C1) -
RH) * GB / ((C1 * GB * RB * RU * (V - 1) * (V + 1) * C2 ^ 3) - 0.4e1 * ((C1 * C3 * E * RU) - (GB * RB * (V - 1)
* (V + 1)) / 0.4e1) * RH * (C2 ^ 2) + ((4 * E * C3 * RU ^ 2 * (V - 1) * (V + 1) * C1 ^ 2) - 0.8e1 * RU * ((V ^ 2) *
C3 * E * RH + (GB * RB * (V - 1) * (V + 1)) / 0.8e1) * C1 + 0.4e1 * E * C3 * RH ^ 2 * (V - 1) * (V + 1)) * C2 -
0.4e1 * ((C1 * C3 * E * RU) + (GB * RB * (V - 1) * (V + 1)) / 0.4e1) * RH);
    % effective stresses of each component based on damage factor
    SXH = max(abs(SIGXXH),SIGTM);
    SXU = max(abs(SIGXXU),SIGTU);
    TXYB = max(abs(TAUXYB),SIGS);
    SXC = max(abs(SIGXXC),SIGTM);
    % with smeared crack model
    % characteristic length of element is element size
    LT = HH;
    LS = HH;
    ATM = (((GI*EH)/(LT*SIGTM^2))-(1/2))^(-1);
    ATU = (((GI*EU)/(LT*SIGTU^2))-(1/2))^(-1);
    ATC = (((GI*EC)/(LT*SIGTM^2))-(1/2))^(-1);
    ASB = (((GII*GB)/(LS*SIGS^2))-(1/2))^(-1);
    % Calculate damage factor from internal stresses
    if RH < TOR
      DHC = 1;
    else
      DHC = 1-(SIGTM*exp(ATM*(1-(SXH/SIGTM)))/SXH);
    end
    DUC = 1-(SIGTU*exp(ATU*(1-(SXU/SIGTU)))/SXU);
    DCC = 1-(SIGTM*exp(ATC*(1-(SXC/SIGTM)))/SXC);
    DBC = 1-(SIGS*exp(ASB*(1-(TXYB/SIGS)))/TXYB);
    % Verification of damage factor
    % Since damage factor will influence stress itself
    % damage factor should be verificated together
    if DHC >= 0 & DUC >=0 & DCC >=0 & DBC >= 0
      if abs(DHC-DH) < TOR
        if abs(DUC-DU) < TOR
          if abs(DCC-DC) < TOR
            if abs(DBC-DB) < TOR
              break;
            else
              DB = DBC;
            end
          else
```

```
            DC = DCC;
          end
        else
          DU = DUC;
        end
      else
        DH = DHC;
      end
    else
      break;
    end
  end
 % total undamaged stress of cell
  sigxxh = RH*SIGXXH;
  sigxxu = RU*SIGXXU;
  sigxxc = RC*SIGXXC;
  tauxyb = RB*TAUXYB;
  SIGXX0 = RH*SIGXXH*C3 + 2*RC*SIGXXC + C3*(RU*SIGXXU + RB*TAUXYB*(C2 - 1)/(2*C3))/2*(C3 +
1);
  STIFF = (C2 + 1) * (((GB * RB * RC * (V - 1) * (V + 1) * C2 ^ 3) / 0.4e1 + (GB * ((V ^ 2) / 0.2e1 - 0.1e1 /
0.2e1) * (RB ^ 2) - (GB * RB * RC * (V - 1) * (V + 1)) / 0.2e1 - (E * C3 * RC * RH)) * (C2 ^ 2) + (GB * (-(V ^ 2)
/ 0.2e1 + 0.1e1 / 0.2e1) * (RB ^ 2) + (-(2 * C3 * E * RH) + (E + GB / 0.4e1) * (V + 1) * (V - 1) * RC) * RB + (E
* RC * RH * (V ^ 2 + C3))) * C2 + 0.2e1 * RH * E * (RB - RC / 0.2e1) * (V ^ 2)) * C3 * RU ^ 2 * C1 ^ 2 - 0.2e1
* (-(GB * RC * (V - 1) * (V + 1) * RB ^ 2 * C2 ^ 3) / 0.8e1 + RH * RB * C3 * (-(V ^ 2 * GB) / 0.4e1 + E + GB /
0.4e1) * RC * (C2 ^ 2) / 0.2e1 + (-GB * (V + 1) * (V - 1) * ((C3 * RH) - RC / 0.2e1) * (RB ^ 2) / 0.4e1 + C3 *
RC * RH * ((E * V ^ 2) + (V ^ 2 * GB) / 0.4e1 - GB / 0.4e1) * RB + (E * C3 * RC * RH ^ 2 * (V ^ 2 + C3)) / 0.2e1)
* C2 + (((V ^ 2) / 0.4e1 - 0.1e1 / 0.4e1) * GB * (RB ^ 2) + ((C3 * E * RH) + (E * V ^ 2 * RH) + (-(V ^ 2 * GB) /
0.4e1 + E + GB / 0.4e1) * RC / 0.2e1) * RB - (E * RC * RH * (V ^ 2 + C3)) / 0.2e1) * RH * C3) * RU * C1 + (V
+ 1) * RH * RB * ((E * C2 * C3 * RH) + (C2 ^ 2 * GB * RB) / 0.4e1 - (GB * RB) / 0.4e1) * (V - 1) * RC) * E / (E
* RU ^ 2 * C2 * C3 * (V - 1) * (V + 1) * C1 ^ 2 - 0.2e1 * (-(GB * RB * (V - 1) * (V + 1) * C2 ^ 3) / 0.8e1 + (E *
C2 ^ 2 * C3 * RH) / 0.2e1 + (GB * RB * ((V ^ 2) / 0.8e1 - 0.1e1 / 0.8e1) + (V ^ 2 * C3 * E * RH)) * C2 + (C3 *
E * RH) / 0.2e1) * RU * C1 + (V + 1) * RH * ((E * C2 * C3 * RH) + (C2 ^ 2 * GB * RB) / 0.4e1 - (GB * RB) /
0.4e1) * (V - 1)) / (C3 + 1) / (C2 * RC + 2 * RB - RC);
  a = [a,SIGXX0];
  b = [b,sigxxh];
  c = [c,sigxxu];
  d = [d,sigxxc];
  e = [e,tauxyb];
end
```

Appendix B: Fortran Code (Model 2)

```
!DEC$ ATTRIBUTES DLLEXPORT::USRMAT
      SUBROUTINE USRMAT( EPS0, DEPS, NS, AGE0, DTIME, TEMP0,
     $                   DTEMP, ELEMEN, INTPT, COORD, SE, ITER,
     $                   USRMOD, USRVAL, NUV, USRSTA, NUS,
     $                   USRIND, NUI, SIG, STIFF )
               IMPLICIT NONE
C
! IN  DBL EPS0(NS)     STRAIN VECTOR AT START OF STEP
! IN  DBL DEPS(NS)     TOTAL STRAIN INCREMENT
! IN  INT NS           NUMBER OF STRESS COMPONENTS
! IN  DBL AGE0         AGE OF ELEMENT
! IN  DBL DTIME        TOTAL TIME INCREMENT
! IN  DBL TEMP0        TEMPERATURE
! IN  DBL DTEMP        TOTAL TEMPERATURE INCREMENT
! IN  INT ELEMEN       CURRENT ELEMENT NUMBER
! IN  INT INTPT        CURRENT INTEGRATION POINT NUMBER
! IN  DBL COORD(3)     COORDINATES OF INTERGRATION POINT
! IN  DBL SE(NS,NS)    ELASTICITY MATRIX
! IN  INT ITER         CURRENT ITERATION NUMBER
! IN  CHA USRMOD*6     USER MODEL NAME
! IN  DBL USRVAL(NUV)  USER PARAMETER
! IN  INT NUV          NUMBER OF USER PARAMETERS
! IN  DBL USRSTA(NUS)  USER STATE VARIABLES AT START OF STEP
! OUT DBL USRSTA(NUS)  UPDATED USER STATE VARIABLES
! IN  INT NUS          NUMBER OF USER STATE VARIABLES
! IN  INT USRIND(NUI)  USER INDICATORS AT START OF STEP
! OUT INT USRIND(NUI)  UPDATED USER INDICATORS
! IN  INT NUI          NUMBER OF USER INDICATORS
! IN  DBL SIG(NS)      TOTAL STRESS AT START OF STEP
! OUT DBL SIG(NS)      CURRENT TOTAL STRESS
! IN  DBL STIFF(NS,NS) PREVIOUS TANGENT STIFFNESS
! OUT DBL STIFF(NS,NS) CURRENT TANGENT STIFFNESS
C
! MATHMATIC IDENTIFY OF VARIABLES
      CHARACTER*6      USRMOD
C
! VARIABLES DEFINED BY USRMAT
      INTEGER          NS, NUV, NUS, NUI, ELEMEN, INTPT, ITER
      DOUBLE PRECISION EPS0(NS), DEPS(NS), AGE0, DTIME, TEMP0,
     $                 DTEMP, COORD(3), SE(NS,NS), USRVAL(NUV),
     $                 USRSTA(NUS), SIG(NS), STIFF(NS,NS)
      INTEGER          USRIND(NUI)
C
! VAIABLES DEFINED BY SELF
      DOUBLE PRECISION DH, DU, DC, DB, GI, GII, SIGTM, SIGTU,
     $                 SIGS, E, V, TOR, HH, GB, C1, C2, C3
      DOUBLE PRECISION EPSXX0, SIGXX0
      DOUBLE PRECISION EU, EH, EC, EB, LT, LS, ATM, ATU, ASM,
     $                 RH, RU, RC, RB, SIGXXHC, SIGXXUC,
     $                 SIGXXCC, TAUXYBC, SXH, SXU, SXC, TXYB,
     $                 DHC, DUC, DCC, DBC
      DOUBLE PRECISION SIGXXH, SIGXXU, SIGXXC, TAUXYB, DMT,
     $                 DUT, DMS, THC, TUC, TCC, TBC, SIGXX0C,
     $                 STIFFNESS
```

```
C
C MAIN PROGRAM: MODELII
! MODEL II: COUPLED SHEAR AND TENSION BEHAVIOURS
! MATERIAL PROPERTIES
      DH    = USRSTA(1)
      DU    = USRSTA(2)
      DC    = USRSTA(3)
      DB    = USRSTA(4)
      GI    = USRVAL(1)
      GII   = USRVAL(2)
      SIGTM = USRVAL(3)
      SIGTU = USRVAL(4)
      SIGS  = USRVAL(5)
      E     = USRVAL(6)
      V     = USRVAL(7)
      TOR   = USRVAL(8)
      HH    = USRVAL(9)
      GB    = USRVAL(10)
! INDENTIFY RELATIVE PARAMETERS
      C1    = USRVAL(11)
! GEOMETRICAL PROPERTIES: L=C2*T, H=C3*T
! L IS HALF LENGTH OF BRICK UNIT
! T IS HALF THICKNESS OF JOINT
! H IS HALF HEIGHT OF BRICK UNIT AND HEAD JOINT
      C2    = USRVAL(12)
      C3    = USRVAL(13)
C
! MATERIAL PROPERTIES OF EACH COMPONENT
      EU    = C1*E
      EH    = E
      EC    = E
      EB    = E
      LT    = HH
      LS    = HH
! CALCULATE OF DAMAGED PARAMETERS
      ATM   = (((GI*EH)/(LT*SIGTM**2.0D0))-(1.0D0/2.0D0))**(-1.0D0)
      ATU   = (((GI*EU)/(LT*SIGTU**2.0D0))-(1.0D0/2.0D0))**(-1.0D0)
      ASM   = (((GII*GB)/(LS*SIGS**2.0D0))-(1.0D0/2.0D0))**(-1.0D0)
! DEFINATION OF EXTERNALLY HORIZONTAL STRAIN
      EPSXX0  = EPS0(1)+DEPS(1)
C
! INNER LOOP: CHECK DAMAGE FACTOR
      DO 30, WHILE(DH .LE. 1.0D0) .AND. (DU .LE. 1.0D0) .AND.
     $           (DC .LE. 1.0D0) .AND. (DB .LE. 1.0D0)
! UNDAMAGE FACTOR OF EACH COMPONENT
        RH    = 1-DH
        RU    = 1-DU
        RC    = 1-DC
        RB    = 1-DB
! TENSION STRESS IN HEAD JOINT
        SIGXXHC = SIGXXH(EPSXX0, RH, RU, RC, RB, E, GB, V,
       $                C1, C2, C3)
! TENSION STRESS IN BRICK UNIT
        SIGXXUC = SIGXXU(EPSXX0, RH, RU, RC, RB, E, GB, V,
       $                C1, C2, C3)
! TENSION STRESS IN CROSS JOINT
```

```
        SIGXXCC = SIGXXC(EPSXX0, RH, RU, RC, RB, E, GB, V,
       $                C1, C2, C3)
! SHEAR STRESS BETWEEN BED JOINT AND BRICK UNIT
        TAUXYBC = TAUXYB(EPSXX0, RH, RU, RC, RB, E, GB, V,
       $                C1, C2, C3)
C
! EFFECTIVE STRESSES OF EACH COMPONENT
        SXH     = MAX(ABS(SIGXXHC),SIGTM)
        SXU     = MAX(ABS(SIGXXUC),SIGTU)
        SXC     = MAX(ABS(SIGXXCC),SIGTM)
        TXYB    = MAX(ABS(TAUXYBC),SIGS)
! CALCULATE DAMAGE FACTOR FROM INTERNAL STRESSES
        DHC    = DMT(SIGTM, ATM, SXH)
        DUC    = DUT(SIGTU, ATU, SXU)
        DCC    = DMT(SIGTM, ATM, SXC)
        DBC    = DMS(SIGS, ASM, TXYB)
C
! DAMAGE FACTOR FROM STRESS CAL.
        IF (DHC .LT. 0.0D0) THEN
           EXIT
        END IF
        IF (DUC .LT. 0.0D0) THEN
           EXIT
        END IF
        IF (DCC .LT. 0.0D0) THEN
           EXIT
        END IF
        IF (DBC .LT. 0.0D0) THEN
           EXIT
        END IF
! FINAL DAMAGE FACTOR
        THC   = ABS(DHC - DH)
        TUC   = ABS(DUC - DU)
        TCC   = ABS(DCC - DC)
        TBC   = ABS(DBC - DB)
        IF (THC .LT. TOR) THEN
           IF (TUC .LT. TOR) THEN
              IF (TCC .LT. TOR) THEN
                 IF (TBC .LT. TOR) THEN
                    EXIT
                 ELSE
                    DB = DBC
                 END IF
              ELSE
                 DC = DCC
              END IF
           ELSE
              DU = DUC
           END IF
        ELSE
           DH = DHC
        END IF
30    CONTINUE
C
! DAMAGED STRESS AND DAMGED STIFFNESS
      SIGXX0C    = SIGXX0(EPSXX0, RH, RU, RC, RB, E, GB, V,
     $                C1, C2, C3)
```

```
      SIG(1)     = SIGXX0C
      STIFFC     = STIFFNESS(RH, RU, RC, RB, E, GB, V, C1,
     $                       C2, C3)
      STIFF(1,1) = STIFFC
C
! STORE OUTPU BY USRSTA MATRIX
      USRSTA(1)  = DH
      USRSTA(2)  = DU
      USRSTA(3)  = DC
      USRSTA(4)  = DB
      END SUBROUTINE USRMAT
C
C SUBPROGRAM: INTERNAL STRESS TENSION STRESS IN HEAD JOINT
      REAL FUNCTION SIGXXH(EPSXX0, RH, RU, RC, RB, E, GB, V,
     $                     C1, C2, C3)
      IMPLICIT NONE
      REAL, INTENT(IN)::EPSXX0, RH, RU, RC, RB
      REAL             ::E, GB, V, C1, C2, C3
      SIGXXH = -0.4D1 * DBLE(C2 + 1) * E * EPSXX0 * (-DBLE(GB
     $ * RB * (V - 1) * (V + 1) * C2) / 0.4D1 + DBLE(GB) *
     $ DBLE(RB) * (DBLE(V ** 2) / 0.4D1 - 0.1D1 / 0.4D1) +
     $ C1 * C3 * E * RU) * (C1 * DBLE(C2) * RU + RH) /
     $ (C1 * DBLE(GB) * DBLE(RB) * RU * DBLE(V - 1) *
     $ DBLE(V + 1) * DBLE(C2 ** 3) - 0.4D1 * (DBLE(GB) *
     $ DBLE(RB) * (-DBLE(V ** 2) / 0.4D1 + 0.1D1 / 0.4D1) +
     $ C1 * C3 * E * RU) * RH * DBLE(C2 ** 2) + (-C1 *
     $ DBLE(GB) * DBLE(RB) * RU * DBLE(V - 1) * DBLE(V + 1)
     $ + 0.4D1 * E * C3 * (RU ** 2 * DBLE(V ** 2 - 1) * C1
     $ ** 2 - 0.2D1 * C1 * RH * RU * DBLE(V ** 2) + RH ** 2
     $ * DBLE(V ** 2 - 1))) * DBLE(C2) - 0.4D1 * (DBLE(GB)
     $ * DBLE(RB) * (DBLE(V ** 2) / 0.4D1 - 0.1D1 / 0.4D1)
     $ + C1 * C3 * E * RU) * RH)
      RETURN
      END FUNCTION SIGXXH
C
C SUBPROGRAM: INTERNAL STRESS TENSION STRESS IN HEAD JOINT
      REAL FUNCTION SIGXXU(EPSXX0, RH, RU, RC, RB, E, GB, V,
     $                     C1, C2, C3)
      IMPLICIT NONE
      REAL, INTENT(IN)::EPSXX0, RH, RU, RC, RB
      REAL             ::E, GB, V, C1, C2, C3
      SIGXXU = -0.4D1 * DBLE(C2 + 1) * DBLE(C1) * E *
     $ (-DBLE(GB * RB * (V - 1) * (V + 1) * C2) / 0.4D1 +
     $ DBLE(GB) * DBLE(RB) * (DBLE(V ** 2) / 0.4D1 - 0.1D1 /
     $ 0.4D1) + C3 * E * RH) * EPSXX0 * (DBLE(C1 * C2 * RU)
     $ + RH) / (DBLE(C1 * GB * RB * RU * (V - 1) * (V + 1) *
     $ C2 ** 3) - 0.4D1 * (DBLE(GB) * DBLE(RB) * (-DBLE(V **
     $ 2) / 0.4D1 + 0.1D1 / 0.4D1) + DBLE(C1) * C3 * E *
     $ DBLE(RU)) * RH * DBLE(C2 ** 2) + (-DBLE(C1 * GB * RB *
     $ RU * (V - 1) * (V + 1)) + 0.4D1 * E * C3 * (DBLE(RU **
     $ 2 * (V ** 2 - 1) * C1 ** 2) - 0.2D1 * DBLE(C1) * RH *
     $ DBLE(RU) * DBLE(V ** 2) + RH ** 2 * DBLE(V ** 2 - 1)))
     $ * DBLE(C2) - 0.4D1 * (DBLE(GB) * DBLE(RB) * (DBLE(V **
     $ 2) / 0.4D1 - 0.1D1 / 0.4D1) + DBLE(C1) * C3 * E *
     $ DBLE(RU)) * RH)
      RETURN
      END FUNCTION SIGXXU
C
C SUBPROGRAM: INTERNAL STRESS TENSION STRESS IN HEAD JOINT
      REAL FUNCTION SIGXXC(EPSXX0, RH, RU, RC, RB, E, GB, V,
     $                    C1, C2, C3)
      IMPLICIT NONE
      REAL, INTENT(IN)::EPSXX0, RH, RU, RC, RB
      REAL           ::E, GB, V, C1, C2, C3
      SIGXXC = DBLE(C2 + 1) * DBLE(E) * EPSXX0 * (DBLE(RU *
     $ C1 * GB * RB ** 2 * RC * (V - 1) * (V + 1) * C2 ** 3)
     $ / 0.4D1 - RH * DBLE(RB) * ((-DBLE(V ** 2 * GB) / 0.4D1
     $ + DBLE(GB) / 0.4D1) * DBLE(RB) + DBLE(C1 * C3 * E *
     $ RU)) * DBLE(RC) * DBLE(C2 ** 2) + (-DBLE(RU * C1 * GB
     $ * (V - 1) * (V + 1) * RB ** 2) / 0.4D1 + DBLE(E) *
     $ DBLE(C3) * (DBLE(RU ** 2 * (V ** 2 - 1) * C1 ** 2) -
     $ 0.2D1 * DBLE(C1) * RH * DBLE(RU) * DBLE(V ** 2) + RH
     $ ** 2 * DBLE(V ** 2 - 1)) * DBLE(RB) + DBLE(E) *
     $ DBLE(RU) * DBLE(V ** 2) * DBLE(C1) * DBLE(C3) * RH
     $ * (DBLE(RU * C1) - RH)) * DBLE(RC) * DBLE(C2) + 0.2D1
     $ * (-DBLE(GB * RC * (V - 1) * (V + 1) * RB ** 2) /
     $ 0.8D1 + DBLE(E) * DBLE(RU) * DBLE(C1) * DBLE(C3) *
     $ (-DBLE(RC) / 0.2D1 + DBLE(C1 * RU * V ** 2) - RH *
     $ DBLE(V ** 2)) * DBLE(RB) - DBLE(E) * DBLE(RU) *
     $ DBLE(V ** 2) * DBLE(C1) * DBLE(C3) * DBLE(RC) *
     $ (DBLE(RU * C1) - RH) / 0.2D1) * RH) / (DBLE(C1 *
     $ GB * RB * RU * (V - 1) * (V + 1) * C2 ** 3) / 0.4D1
     $ - RH * ((-DBLE(V ** 2 * GB) / 0.4D1 + DBLE(GB) / 0.4D1)
     $ * DBLE(RB) + DBLE(C1 * C3 * E * RU)) * DBLE(C2 ** 2) +
     $ (-DBLE(C1 * GB * RB * RU * (V - 1) * (V + 1)) / 0.4D1 +
     $ DBLE(E) * DBLE(C3) * (DBLE(RU ** 2 * (V ** 2 - 1) * C1
     $ ** 2) - 0.2D1 * DBLE(C1) * RH * DBLE(RU) * DBLE(V ** 2)
     $ + RH ** 2 * DBLE(V ** 2 - 1))) * DBLE(C2) - RH *
     $ ((DBLE(V ** 2 * GB) / 0.4D1 - DBLE(GB) / 0.4D1) *
     $ DBLE(RB) + DBLE(C1 * C3 * E * RU))) / DBLE(RC) /
     $ DBLE(RC * C2 + 2 * RB - RC)
      RETURN
      END FUNCTION SIGXXC
C
C SUBPROGRAM: INTERNAL STRESS SHEAR STRESS IN BED JOINT
      REAL FUNCTION TAUXYB(EPSXX0, RH, RU, RC, RB, E, GB, V,
     $                    C1, C2, C3)
      IMPLICIT NONE
      REAL, INTENT(IN)::EPSXX0, RH, RU, RC, RB
      REAL           ::E, GB, V, C1, C2, C3
      TAUXYB = 0.2D1 * DBLE(V - 1) * DBLE(V + 1) *
     $ (DBLE(C1 * C2 * RU) + RH) * DBLE(C3) * DBLE(E) *
     $ EPSXX0 * DBLE(C2 + 1) * (DBLE(RU * C1) - RH) *
     $ DBLE(GB) / (DBLE(C1 * GB * RB * RU * (V - 1) *
     $ (V + 1) * C2 ** 3) - 0.4D1 * (DBLE(C1 * C3 * E
     $ * RU) - DBLE(GB * RB * (V - 1) * (V + 1)) / 0.4D1)
     $ * RH * DBLE(C2 ** 2) + (DBLE(4 * E * C3 * RU ** 2
     $ * (V - 1) * (V + 1) * C1 ** 2) - 0.8D1 * DBLE(RU)
     $ * (DBLE(V ** 2) * DBLE(C3) * DBLE(E) * RH +
     $ DBLE(GB * RB * (V - 1) * (V + 1)) / 0.8D1) *
     $ DBLE(C1) + 0.4D1 * DBLE(E) * DBLE(C3) * RH ** 2
     $ * DBLE(V - 1) * DBLE(V + 1)) * DBLE(C2) - 0.4D1
     $ * (DBLE(C1 * C3 * E * RU) + DBLE(GB * RB * (V - 1)
     $ * (V + 1)) / 0.4D1) * RH)
      RETURN
      END FUNCTION TAUXYB
C
C SUBPROGRAM: INTERNAL STRESS SHEAR STRESS IN BED JOINT
      REAL FUNCTION SIGXX0(EPSXX0, RH, RU, RC, RB, E, GB, V,
     $                    C1, C2, C3)
      IMPLICIT NONE
      REAL, INTENT(IN)::EPSXX0, RH, RU, RC, RB
      REAL            ::E, GB, V, C1, C2, C3
      SIGXX0 = DBLE(C2 + 1) * ((DBLE(GB * RB * RC * (V - 1)
     $ * (V + 1) * C2 ** 3) / 0.4D1 + (DBLE(GB) * (DBLE(V ** 2)
     $ / 0.2D1 - 0.1D1 / 0.2D1) * DBLE(RB ** 2) - DBLE(GB *
     $ RB * RC * (V - 1) * (V + 1)) / 0.2D1 - DBLE(E * C3 *
     $ RC * RH)) * DBLE(C2 ** 2) + (DBLE(GB) * (-DBLE(V **
     $ 2) / 0.2D1 + 0.1D1 / 0.2D1) * DBLE(RB ** 2) +
     $ (-DBLE(2 * C3 * E * RH) + (DBLE(E) + DBLE(GB) / 0.4D1)
     $ * DBLE(V + 1) * DBLE(V - 1) * DBLE(RC)) * DBLE(RB) +
     $ DBLE(E * RC * RH * (V ** 2 + C3))) * DBLE(C2) + 0.2D1
     $ * DBLE(RH) * DBLE(E) * (DBLE(RB) - DBLE(RC) / 0.2D1)
     $ * DBLE(V ** 2)) * DBLE(C3) * RU ** 2 * C1 ** 2 - 0.2D1
     $ * (-DBLE(GB * RC * (V - 1) * (V + 1) * RB ** 2 * C2
     $ ** 3) / 0.8D1 + DBLE(RH) * DBLE(RB) * DBLE(C3) *
     $ (-DBLE(V ** 2 * GB) / 0.4D1 + DBLE(E) + DBLE(GB) /
     $ 0.4D1) * DBLE(RC) * DBLE(C2 ** 2) / 0.2D1 + (-DBLE(GB)
     $ * DBLE(V + 1) * DBLE(V - 1) * (DBLE(C3 * RH) - DBLE(RC)
     $ / 0.2D1) * DBLE(RB ** 2) / 0.4D1 + DBLE(C3) * DBLE(RC)
     $ * DBLE(RH) * (DBLE(E * V ** 2) + DBLE(V ** 2 * GB) /
     $ 0.4D1 - DBLE(GB) / 0.4D1) * DBLE(RB) + DBLE(E * C3 *
     $ RC * RH ** 2 * (V ** 2 + C3)) / 0.2D1) * DBLE(C2) +
     $ ((DBLE(V ** 2) / 0.4D1 - 0.1D1 / 0.4D1) * DBLE(GB) *
     $ DBLE(RB ** 2) + (DBLE(C3 * E * RH) + DBLE(E * V ** 2
     $ * RH) + (-DBLE(V ** 2 * GB) / 0.4D1 + DBLE(E) + DBLE(GB)
     $ / 0.4D1) * DBLE(RC) / 0.2D1) * DBLE(RB) - DBLE(E * RC
     $ * RH * (V ** 2 + C3)) / 0.2D1) * DBLE(RH) * DBLE(C3))
     $ * RU * C1 + DBLE(V + 1) * DBLE(RH) * DBLE(RB) * (DBLE(E
     $ * C2 * C3 * RH) + DBLE(C2 ** 2 * GB * RB) / 0.4D1 -
     $ DBLE(GB * RB) / 0.4D1) * DBLE(V - 1) * DBLE(RC)) *
     $ DBLE(E) * EPSXX0 / (DBLE(E) * RU ** 2 * DBLE(C2) *
     $ DBLE(C3) * DBLE(V - 1) * DBLE(V + 1) * C1 ** 2 -
     $ 0.2D1 * (-DBLE(GB * RB * (V - 1) * (V + 1) * C2 ** 3)
     $ / 0.8D1 + DBLE(E * C2 ** 2 * C3 * RH) / 0.2D1 +
     $ (DBLE(GB) * DBLE(RB) * (DBLE(V ** 2) / 0.8D1 - 0.1D1
     $ / 0.8D1) + DBLE(V ** 2 * C3 * E * RH)) * DBLE(C2) +
     $ DBLE(C3 * E * RH) / 0.2D1) * RU * C1 + DBLE(V + 1) *
     $ DBLE(RH) * (DBLE(E * C2 * C3 * RH) + DBLE(C2 ** 2 *
     $ GB * RB) / 0.4D1 - DBLE(GB * RB) / 0.4D1) * DBLE(V
     $ - 1)) / DBLE(C3 + 1) / DBLE(RC * C2 + 2 * RB - RC)
      RETURN
      END FUNCTION SIGXX0
C
C SUBPROGRAM: STIFFNESS OF HOMOGENIZED CELL
      REAL FUNCTION STIFFNESS( RH, RU, RC, RB, E, GB, V,
     $                       C1, C2, C3)
      IMPLICIT NONE
      REAL, INTENT(IN)::RH, RU, RC, RB
      REAL          ::E, GB, V, C1, C2, C3
      STIFF = DBLE(C2 + 1) * ((DBLE(GB * RB * RC * (V - 1) *
     $ (V + 1) * C2 ** 3) / 0.4D1 + (DBLE(GB) * (DBLE(V **
     $ 2) / 0.2D1 - 0.1D1 / 0.2D1) * DBLE(RB ** 2) -
     $ DBLE(GB * RB * RC * (V - 1) * (V + 1)) / 0.2D1 -
     $ DBLE(E * C3 * RC * RH)) * DBLE(C2 ** 2) + (DBLE(GB)
     $ * (-DBLE(V ** 2) / 0.2D1 + 0.1D1 / 0.2D1) *
     $ DBLE(RB ** 2) + (-DBLE(2 * C3 * E * RH) + (DBLE(E)
     $ + DBLE(GB) / 0.4D1) * DBLE(V + 1) * DBLE(V - 1) *
     $ DBLE(RC)) * DBLE(RB) + DBLE(E * RC * RH * (V ** 2
     $ + C3))) * DBLE(C2) + 0.2D1 * DBLE(RH) * DBLE(E) *
     $ (DBLE(RB) - DBLE(RC) / 0.2D1) * DBLE(V ** 2)) *
     $ DBLE(C3) * RU ** 2 * C1 ** 2 - 0.2D1 * (-DBLE(GB
     $ * RC * (V - 1) * (V + 1) * RB ** 2 * C2 ** 3) /
     $ 0.8D1 + DBLE(RH) * DBLE(RB) * DBLE(C3) * (-DBLE(V
     $ ** 2 * GB) / 0.4D1 + DBLE(E) + DBLE(GB) / 0.4D1) *
     $ DBLE(RC) * DBLE(C2 ** 2) / 0.2D1 + (-DBLE(GB) *
     $ DBLE(V + 1) * DBLE(V - 1) * (DBLE(C3 * RH) -
     $ DBLE(RC) / 0.2D1) * DBLE(RB ** 2) / 0.4D1 + DBLE(C3)
     $ * DBLE(RC) * DBLE(RH) * (DBLE(E * V ** 2) + DBLE(V
     $ ** 2 * GB) / 0.4D1 - DBLE(GB) / 0.4D1) * DBLE(RB) +
     $ DBLE(E * C3 * RC * RH ** 2 * (V ** 2 + C3)) / 0.2D1)
     $ * DBLE(C2) + ((DBLE(V ** 2) / 0.4D1 - 0.1D1 / 0.4D1)
     $ * DBLE(GB) * DBLE(RB ** 2) + (DBLE(C3 * E * RH) +
     $ DBLE(E * V ** 2 * RH) + (-DBLE(V ** 2 * GB) / 0.4D1
     $ + DBLE(E) + DBLE(GB) / 0.4D1) * DBLE(RC) / 0.2D1) *
     $ DBLE(RB) - DBLE(E * RC * RH * (V ** 2 + C3)) / 0.2D1)
     $ * DBLE(RH) * DBLE(C3)) * RU * C1 + DBLE(V + 1) *
     $ DBLE(RH) * DBLE(RB) * (DBLE(E * C2 * C3 * RH) +
     $ DBLE(C2 ** 2 * GB * RB) / 0.4D1 - DBLE(GB * RB) /
     $ 0.4D1) * DBLE(V - 1) * DBLE(RC)) * DBLE(E) / (DBLE(E)
     $ * RU ** 2 * DBLE(C2) * DBLE(C3) * DBLE(V - 1) *
     $ DBLE(V + 1) * C1 ** 2 - 0.2D1 * (-DBLE(GB * RB *
     $ (V - 1) * (V + 1) * C2 ** 3) / 0.8D1 + DBLE(E * C2
     $ ** 2 * C3 * RH) / 0.2D1 + (DBLE(GB) * DBLE(RB) *
     $ (DBLE(V ** 2) / 0.8D1 - 0.1D1 / 0.8D1) + DBLE(V **
     $ 2 * C3 * E * RH)) * DBLE(C2) + DBLE(C3 * E * RH) /
     $ 0.2D1) * RU * C1 + DBLE(V + 1) * DBLE(RH) * (DBLE(E
     $ * C2 * C3 * RH) + DBLE(C2 ** 2 * GB * RB) / 0.4D1
     $ - DBLE(GB * RB) / 0.4D1) * DBLE(V - 1)) /
     $ DBLE(C3 + 1) / DBLE(C2 * RC + 2 * RB - RC)
      RETURN
      END FUNCTION STIFFNESS
C
C SUBPROGRAM: DAMAGE FACTOR CAL. FOR MORTAR IN TENSION
      REAL FUNCTION DMT(SIGTM, ATM, SXM)
      IMPLICIT NONE
      REAL, INTENT(IN)::SXM
      REAL          ::SIGTM, ATM
      DMT   = 1.0D0-SIGTM*EXP(ATM*(1.0D0-(SXM/SIGTM)))/SXM
      RETURN
      END FUNCTION DMT
C
C SUBPROGRAM: DAMAGE FACTOR CAL. FOR MORTAR IN SHEAR
      REAL FUNCTION DMS(SIGS, ASM, TXY)
      IMPLICIT NONE
      REAL, INTENT(IN)::TXY
      REAL           ::SIGS, ASM
      DMS    = 1.0D0-SIGS*EXP(ASM*(1.0D0-(TXY/SIGS)))/TXY
      RETURN
      END FUNCTION DMS
C
C SUBPROGRAM: DAMAGE FACTOR CAL. FOR BRICK IN TENSION
      REAL FUNCTION DUT(SIGTU, ATU, SXU)
      IMPLICIT NONE
      REAL, INTENT(IN)::SXU
      REAL           ::SIGTU, ATU
      DUT    = 1.0D0-SIGTU*EXP(ATU*(1.0D0-(SXU/SIGTU)))/SXU
      RETURN
      END FUNCTION DUT
```

Appendix C: MATLAB Code (Model 3 Brick Unit)

```
clear all;
% properties setting:
% internal stress selection: SIGXXU(DU),SIGXXH(DH),TAUXYB(DB),SIGXXC(DC)
E = 1178; C1 = 4.13;
EU = C1*E;
V = 0.094;
VU = V;
phi = (10*pi)/180;
psi = (5*pi)/180;
GC = 29.8;
% I fracture energy
GI = 1.9;
% initial value of external strain and damage factor:
EPSYY0 = 0;
DEPSYY0 = 0.0005;
EPSPXXU = 0;
EPSPYYU = 0;
K = 0;
% initialize value of variables:
DU = 0; SU = 0;
DEPSPXXU = 0;
DEPSPYYU = 0;
% Tension and compression strength of brick unit
SIGTU = 3.7; fc = 26.9;
eps0 = 2*fc/EU; % ultimate strain
% element size
% ATU must be positive, check maximum mesh size: HH < 59
HH = 50;
LC = HH;
a = [];
b = [];
d = [];
% Tolerance of calculted and assumed damge factor
TOR = 0.00001;
TOR2 = 0.00001;
% outer loop: strain interation
for i = 1:300
   EPSYY0 = EPSYY0 + DEPSYY0;
   DK = sqrt((6 * DEPSPXXU ^ 2 + 6 * DEPSPYYU ^ 2)) / 0.3e1;
   K  = K + DK;
   % find compressive strength by hardening parameter K:
   KMAX = ((2 * LC * eps0 * fc + 3 * GC) / LC / fc) / 0.2e1;
   if K <= eps0
      SIGC = (fc * (-2 * K ^ 2 / eps0 ^ 2 + 4 * K / eps0 + 1)) / 0.3e1;
      KC = (fc * (-4 * K / eps0 ^ 2 + 4 / eps0)) / 0.3e1;
   else
      if K < KMAX
         SIGC = fc * (0.1e1 - 0.4e1 / 0.9e1 * fc ^ 2 * LC ^ 2 / GC ^ 2 * (K - eps0) ^ 2);
         KC = -0.8e1 / 0.9e1 * fc ^ 3 * LC ^ 2 / GC ^ 2 * (K - eps0);
      else
         SIGC = 0;
         KC = -0.8e1 / 0.9e1 * fc ^ 3 * LC ^ 2 / GC ^ 2 * (KMAX - eps0);
      end
   end
```

```
    % find critical stress:
    c = (0.1e1 - sin(phi)) / cos(phi) * SIGC / 0.2e1;
    SIGYYUC = (0.6e1 * (phi)) * SIGC / (sin(phi) + 0.3e1);
    SIGXXUC = 0;
    % judge if yielding:
    FU = sqrt(EPSYY0 ^ 2 * EU ^ 2) + 0.2e1 * sin(phi) / (0.3e1 - sin(phi)) * EPSYY0 * EU - 0.6e1 * c * cos(phi)
/ (0.3e1 - sin(phi));
    if FU <= 0
        DEPSPXXU = 0;
        DEPSPYYU = 0;
    else
        while c > 0
            % calculate plastic strain increment:
            DEPSPXXU = 0.6e1 * DEPSYY0 * (((((VU ^ 2 - 0.7e1 / 0.3e1 * VU + 0.7e1 / 0.3e1) * SIGXXUC ^ 2 -
0.3e1 / 0.2e1 * (VU ^ 2 - 0.11e2 / 0.9e1 * VU + 0.11e2 / 0.9e1) * SIGYYUC * SIGXXUC + (VU ^ 2 - 0.14e2 /
0.3e1 * VU + 0.14e2 / 0.3e1) * SIGYYUC ^ 2 / 0.2e1) * sin(phi) - 0.3e1 * ((VU ^ 2 + VU / 0.3e1 - 0.1e1 / 0.3e1)
* SIGXXUC - (VU ^ 2 + 0.2e1 / 0.3e1 * VU - 0.2e1 / 0.3e1) * SIGYYUC) * (-SIGYYUC / 0.2e1 + SIGXXUC)) *
sin(psi) - 0.3e1 * (-0.3e1 + sin(phi)) * ((VU ^ 2 + VU / 0.3e1 - 0.1e1 / 0.3e1) * SIGXXUC - (VU ^ 2 + 0.2e1 /
0.3e1 * VU - 0.2e1 / 0.3e1) * SIGYYUC) * (-SIGYYUC / 0.2e1 + SIGXXUC)) * sqrt(SIGXXUC ^ 2 - SIGXXUC
* SIGYYUC + SIGYYUC ^ 2) - 0.2e1 * ((((VU ^ 2 - VU / 0.3e1 + 0.1e1 / 0.3e1) * SIGXXUC - (VU ^ 2 + VU /
0.3e1 - 0.1e1 / 0.3e1) * SIGYYUC) * sin(phi) + (-0.3e1 * VU ^ 2 - VU + 0.1e1) * SIGXXUC + SIGYYUC *
(0.3e1 * VU ^ 2 + 0.2e1 * VU - 0.2e1)) * sin(psi) + 0.2e1 * (VU - 0.1e1) * (-SIGYYUC / 0.2e1 + SIGXXUC) *
sin(phi)) * (SIGXXUC ^ 2 - SIGXXUC * SIGYYUC + SIGYYUC ^ 2)) * EU / (((((0.8e1 * SU * VU ^ 2 + ((9 * EU)
+ 0.4e1 * SU) * VU - (37 * EU) - 0.4e1 * SU) * SIGXXUC ^ 2 - 0.18e2 * SIGYYUC * (0.4e1 / 0.9e1 * SU * VU
^ 2 + (EU + 0.2e1 / 0.9e1 * SU) * VU - 0.20e2 / 0.9e1 * EU - 0.2e1 / 0.9e1 * SU) * SIGXXUC + 0.9e1 *
SIGYYUC ^ 2 * (0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1 / 0.9e1 * SU) * VU - 0.37e2 / 0.9e1 * EU - 0.4e1 /
0.9e1 * SU)) * sin(phi) + (-0.24e2 * SU * VU ^ 2 + (-(27 * EU) - 0.12e2 * SU) * VU + (15 * EU) + 0.12e2 * SU)
* SIGXXUC ^ 2 + 0.54e2 * SIGYYUC * (0.4e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.2e1 / 0.9e1 * SU) * VU - 0.4e1
/ 0.9e1 * EU - 0.2e1 / 0.9e1 * SU) * SIGXXUC - 0.27e2 * SIGYYUC ^ 2 * (0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU
+ 0.4e1 / 0.9e1 * SU) * VU - 0.5e1 / 0.9e1 * EU - 0.4e1 / 0.9e1 * SU)) * sin(psi) - 0.27e2 * (-0.3e1 + sin(phi))
* ((0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1 / 0.9e1 * SU) * VU - 0.5e1 / 0.9e1 * EU - 0.4e1 / 0.9e1 * SU) *
SIGXXUC ^ 2 - 0.2e1 * SIGYYUC * (0.4e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.2e1 / 0.9e1 * SU) * VU - 0.4e1 /
0.9e1 * EU - 0.2e1 / 0.9e1 * SU) * SIGXXUC + SIGYYUC ^ 2 * (0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1 /
0.9e1 * SU) * VU - 0.5e1 / 0.9e1 * EU - 0.4e1 / 0.9e1 * SU))) * sqrt(SIGXXUC ^ 2 - SIGXXUC * SIGYYUC +
SIGYYUC ^ 2) + 0.8e1 * ((sin(phi) - 0.3e1 / 0.2e1) * sin(psi) - 0.3e1 / 0.2e1 * sin(phi)) * (SIGXXUC + SIGYYUC)
* (SIGXXUC ^ 2 - SIGXXUC * SIGYYUC + SIGYYUC ^ 2) * EU);

            DEPSPYYU = 0.3e1 * (((((VU ^ 2 + 0.17e2 / 0.3e1 * VU - 0.17e2 / 0.3e1) * SIGXXUC ^ 2 - 0.3e1 *
(VU ^ 2 + 0.20e2 / 0.9e1 * VU - 0.20e2 / 0.9e1) * SIGYYUC * SIGXXUC + 0.2e1 * (VU ^ 2 + 0.10e2 / 0.3e1 *
VU - 0.10e2 / 0.3e1) * SIGYYUC ^ 2) * sin(phi) - 0.3e1 * ((VU ^ 2 + VU / 0.3e1 - 0.1e1 / 0.3e1) * SIGXXUC -
(VU ^ 2 + 0.2e1 / 0.3e1 * VU - 0.2e1 / 0.3e1) * SIGYYUC) * (SIGXXUC - 0.2e1 * SIGYYUC)) * sin(psi) - 0.3e1
* (-0.3e1 + sin(phi)) * ((VU ^ 2 + VU / 0.3e1 - 0.1e1 / 0.3e1) * SIGXXUC - (VU ^ 2 + 0.2e1 / 0.3e1 * VU - 0.2e1
/ 0.3e1) * SIGYYUC) * (SIGXXUC - 0.2e1 * SIGYYUC)) * sqrt(SIGXXUC ^ 2 - SIGXXUC * SIGYYUC +
SIGYYUC ^ 2) + 0.4e1 * ((((VU ^ 2 + 0.2e1 / 0.3e1 * VU - 0.2e1 / 0.3e1) * SIGXXUC - (VU + 0.2e1) * (VU -
0.2e1 / 0.3e1) * SIGYYUC) * sin(phi) + (-0.3e1 * VU ^ 2 - VU + 0.1e1) * SIGXXUC + SIGYYUC * (0.3e1 * VU
^ 2 + 0.2e1 * VU - 0.2e1)) * sin(psi) - sin(phi) * (SIGXXUC - 0.2e1 * SIGYYUC) * (VU - 0.1e1)) * (SIGXXUC ^
2 - SIGXXUC * SIGYYUC + SIGYYUC ^ 2)) * DEPSYY0 * EU / (((((0.8e1 * SU * VU ^ 2 + ((9 * EU) + 0.4e1 *
SU) * VU - (37 * EU) - 0.4e1 * SU) * SIGXXUC ^ 2 - 0.18e2 * SIGYYUC * (0.4e1 / 0.9e1 * SU * VU ^ 2 + (EU
+ 0.2e1 / 0.9e1 * SU) * VU - 0.20e2 / 0.9e1 * EU - 0.2e1 / 0.9e1 * SU) * SIGXXUC + 0.9e1 * SIGYYUC ^ 2 *
(0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1 / 0.9e1 * SU) * VU - 0.37e2 / 0.9e1 * EU - 0.4e1 / 0.9e1 * SU)) *
sin(phi) + (-0.24e2 * SU * VU ^ 2 + (-(27 * EU) - 0.12e2 * SU) * VU + (15 * EU) + 0.12e2 * SU) * SIGXXUC ^
2 + 0.54e2 * SIGYYUC * (0.4e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.2e1 / 0.9e1 * SU) * VU - 0.4e1 / 0.9e1 * EU
- 0.2e1 / 0.9e1 * SU) * SIGXXUC - 0.27e2 * SIGYYUC ^ 2 * (0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1 /
0.9e1 * SU) * VU - 0.5e1 / 0.9e1 * EU - 0.4e1 / 0.9e1 * SU)) * sin(psi) - 0.27e2 * (-0.3e1 + sin(phi)) * ((0.8e1
/ 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1 / 0.9e1 * SU) * VU - 0.5e1 / 0.9e1 * EU - 0.4e1 / 0.9e1 * SU) * SIGXXUC
^ 2 - 0.2e1 * SIGYYUC * (0.4e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.2e1 / 0.9e1 * SU) * VU - 0.4e1 / 0.9e1 * EU
- 0.2e1 / 0.9e1 * SU) * SIGXXUC + SIGYYUC ^ 2 * (0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1 / 0.9e1 * SU)
* VU - 0.5e1 / 0.9e1 * EU - 0.4e1 / 0.9e1 * SU))) * sqrt(SIGXXUC ^ 2 - SIGXXUC * SIGYYUC + SIGYYUC ^
2) + 0.8e1 * ((sin(phi) - 0.3e1 / 0.2e1) * sin(psi) - 0.3e1 / 0.2e1 * sin(phi)) * (SIGXXUC + SIGYYUC) *
(SIGXXUC ^ 2 - SIGXXUC * SIGYYUC + SIGYYUC ^ 2) * EU);

         % recalculate softening modulus SUC:
         DL = 0.6e1 * ((-0.3e1 + sin(phi)) * (((-DEPSYY0 * VU - DEPSYY0) * VU + 0.2e1 / 0.3e1 * DEPSYY0
* VU + DEPSYY0 / 0.3e1) * SIGXXUC - SIGYYUC * ((-DEPSYY0 * VU - DEPSYY0) * VU + DEPSYY0 * VU
/ 0.3e1 + 0.2e1 / 0.3e1 * DEPSYY0)) * sqrt(SIGXXUC ^ 2 - SIGXXUC * SIGYYUC + SIGYYUC ^ 2) + 0.4e1 /
0.3e1 * sin(phi) * (SIGXXUC ^ 2 - SIGXXUC * SIGYYUC + SIGYYUC ^ 2) * (-DEPSYY0 * VU + DEPSYY0))
* EU * (-0.3e1 + sin(psi)) / (0.8e1 * (SIGXXUC + SIGYYUC) * ((sin(psi) - 0.3e1 / 0.2e1) * sin(phi) - 0.3e1 /
0.2e1 * sin(psi)) * EU * sqrt(SIGXXUC ^ 2 - SIGXXUC * SIGYYUC + SIGYYUC ^ 2) + (((0.8e1 * SU * VU ^ 2
+ (0.9e1 * EU + 0.4e1 * SU) * VU - 0.37e2 * EU - 0.4e1 * SU) * SIGXXUC ^ 2 - 0.18e2 * SIGYYUC * (0.4e1 /
0.9e1 * SU * VU ^ 2 + (EU + 0.2e1 / 0.9e1 * SU) * VU - 0.20e2 / 0.9e1 * EU - 0.2e1 / 0.9e1 * SU) * SIGXXUC
+ 0.9e1 * (0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1 / 0.9e1 * SU) * VU - 0.37e2 / 0.9e1 * EU - 0.4e1 / 0.9e1
* SU) * SIGYYUC ^ 2) * sin(psi) + (-0.24e2 * SU * VU ^ 2 + (-0.27e2 * EU - 0.12e2 * SU) * VU + 0.15e2 * EU
+ 0.12e2 * SU) * SIGXXUC ^ 2 + 0.54e2 * SIGYYUC * (0.4e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.2e1 / 0.9e1 *
SU) * VU - 0.4e1 / 0.9e1 * EU - 0.2e1 / 0.9e1 * SU) * SIGXXUC - 0.27e2 * SIGYYUC ^ 2 * (0.8e1 / 0.9e1 *
SU * VU ^ 2 + (EU + 0.4e1 / 0.9e1 * SU) * VU - 0.5e1 / 0.9e1 * EU - 0.4e1 / 0.9e1 * SU)) * sin(phi) - 0.27e2 *
((0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1 / 0.9e1 * SU) * VU - 0.5e1 / 0.9e1 * EU - 0.4e1 / 0.9e1 * SU) *
SIGXXUC ^ 2 - 0.2e1 * SIGYYUC * (0.4e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.2e1 / 0.9e1 * SU) * VU - 0.4e1 /
0.9e1 * EU - 0.2e1 / 0.9e1 * SU) * SIGXXUC + SIGYYUC ^ 2 * (0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1 /
0.9e1 * SU) * VU - 0.5e1 / 0.9e1 * EU - 0.4e1 / 0.9e1 * SU)) * (-0.3e1 + sin(psi)));

         fk = 0.3e1 / (0.3e1 - sin(phi)) * (0.1e1 - sin(phi)) * KC;

         SUC = 0.1e1 / (0.3e1 - sin(phi)) * (0.1e1 - sin(phi)) * KC * sqrt((6 * DEPSPXXU ^ 2 + 6 * DEPSPYYU
^ 2)) / DL;

         if abs(SUC-SU) < TOR2
            break;
         else
            SU = SUC;
         end
         d = [d,DL];
      end
   end
   EPSPXXU = EPSPXXU + DEPSPXXU;
   EPSPYYU = EPSPYYU + DEPSPYYU;
   if EPSYY0-EPSPYYU < 0
      EPSPYYU = EPSYY0;
   else
   end
   % inner loop: verification of damage factor
   % damage factor:
   while DU < 1
      % undamage factor of each component
      RU = 1-DU;
      SIGXXUP = 0;
      % stress in brick unit
      SXU = max(SIGXXUP,SIGTU);
      % characteristic length of element is element size
      LT = HH;
      ATU = (((GI*C1*E)/(LT*SIGTU^2))-(1/2))^(-1);
      % Calculate damage factor from internal stresses
      DUC = 1-(SIGTU*exp(ATU*(1-(SXU/SIGTU)))/SXU);
      % Verification of damage factor
```

```
        % Since damage factor and aplhai will influence stress itself
        % damage factor and coefficent alpha should be verificated
        if DUC >= 0
            if abs(DUC-DU) < TOR
               break;
            else
                DUC = DU;
            end
        else
            break;
        end
    end
% total undamaged stress of cell
  SIGYY0 = RU * EU * (-EPSPYYU + EPSYY0);
  a = [a,SIGYY0];
  b = [b,SIGYYUC];
end
```

Appendix D: MATLAB Code (Model 3)

```
clear all;
% properties setting:
E = 1178; C1 = 4.13;
EU = C1*E; EH = E; EB = E; EC = E;
V = 0.057; % Poisson's ratio of mortar
VU = 0.094; % Poisson's ratio of brick
GB = E/(2*(1+V)); % shear modulus of mortar
% friction and dilatancy angel:
PHIU = (10*pi)/180;
PSIU = (5*pi)/180;
PHIM = (10*pi)/180;
PSIM = (5*pi)/180;
% I and II fracture energy
GIU = 1.9; GIM = 0.35;
GII = 0.05;
% compressive fracture energy
GCU = 29.8; GCM = 6.43;
% Shear, tension and compressive strength:
SIGTU = 3.7; % Tension strength of brick unit
SIGTM = 0.7; % Tension strength of mortar
FCU   = 26.9; %compressive strength of brick
FCM   = 3.2; % compressive strength of mortar
% Shear strength of mortar:
SIGS  = 0.75; % should always be smaller than "2c*cos(phi)^2/(1-sin(phi))" with cmax = fc
% maximum strain of strain-stress curve under compression
EPS0U = 2*FCU/EU;
EPS0M = 2*FCM/E;

% geometrical properties:
C2 = 12; C3 = 2; % properties of masonry: L=C2*T, H=C3*T

% initial value of external strain and damage factor:
EPSYY0 = 0; % external strain
DEPSYY0 = 0.0001; % external strain increment
EPSPXXU = 0;EPSPYYU = 0; % initial value of plastic strain
EPSPXXH = 0;EPSPYYH = 0;
EPSPXXB = 0;EPSPYYB = 0;EPSPXYB = 0;
KU = 0; KH = 0; KB = 0; % initial value of hardening(softening) parameter
% initialize value of variables:
DH = 0; DU = 0; DB = 0; DC = 0; % damage varaiables
SU = 0; SH = 0; SB = 0; %softening modulus
DEPSPXXU = 0;DEPSPYYU = 0; % initialized value of plastic strain increment
DEPSPXXH = 0;DEPSPYYH = 0;
DEPSPXXB = 0;DEPSPYYB = 0;DEPSPXYB = 0;

% ATM must be positive, check maximum mesh size: HH < 80
% ATU must be positive, check maximum mesh size: HH < 59
% AS must be positive, check maximum mesh size: HH <74
HH = 10; % element size
LT = HH; LS = HH; LC = HH; % characteristic length of element is element size

% Tolerance of calculted and assumed damge factor
TOR = 0.00001; % verify damage factor
TOR2 = 0.00001; % verify hardening modulus
```

```
SWC = 0;

a = []; b = []; d = []; e = []; f = []; g = []; h = [];

% outer loop: strain integration
for i = 1:500
   EPSYY0 = EPSYY0 + DEPSYY0;

   % inner loop: verification of damage factor
   % undamage factor of each component
   RH = 1-DH;
   RB = 1-DB;
   RC = 1-DC;
   RU = 1-DU;

   % drucker pragar plasiticy model of brick unit
   % elastic predictor of brick unit
      SIGXXUE = -0.4e1 * (C3 + 1) * (-GB * (V - 1) * (V + 1) * ((-C2 / 0.2e1 - 0.1e1 / 0.2e1) * RC + RU * C1) *
(RC * C3 + (2 * RH)) * (C2 - 0.1e1) * RB ^ 2 / 0.2e1 + (0.2e1 * RC * E * RH * ((-C2 / 0.2e1 - 0.1e1 / 0.2e1) *
RC + RU * C1) * (C3 ^ 2) + (-RU * C1 * GB * (C2 - 0.1e1) ^ 2 * (V - 1) * (V + 1) * RC ^ 2 / 0.4e1 - 0.2e1 * RC
* (RH ^ 2) * E * C2 + 0.4e1 * RU * C1 * E * (RH ^ 2)) * C3 - RU * C1 * GB * RC * RH * (C2 - 0.1e1) ^ 2 * (V -
1) * (V + 1) / 0.2e1) * RB + RC * E * RU * C1 * C3 * (RC * (C2 - 0.1e1) * C3 - 0.2e1 * RC + (0.2e1 * C2 - 0.2e1)
* RH) * RH) * E * C1 * RB * V * EPSYY0 / (-0.2e1 * GB * (V - 1) * (V + 1) * C3 * (-RU * C1 * C3 - (V - 1) * (V
+ 1) * (C2 + 0.1e1) * RC / 0.2e1 + RU * C1 * (V ^ 2)) * (RC * C3 + (2 * RH)) * (C2 - 0.1e1) * RB ^ 3 - 0.4e1 *
(0.2e1 * RU * C1 * E * (C3 ^ 3) * RH + ((V - 1) * (RU * (-(C2 - 0.1e1) ^ 2 * GB / 0.4e1 + E) * C1 + E * C2 * RH)
* (V + 1) * RC - 0.2e1 * RU * C1 * E * RH * (V ^ 2)) * (C3 ^ 2) + GB * (V - 1) * RU * (V + 1) * C1 * (-(V ^ 2) *
RC + RU * (V - 1) * (V + 1) * C1) * (C2 - 0.1e1) * C3 / 0.2e1 + RU * C1 * GB * RC * (C2 - 0.1e1) * (C2 + 0.1e1)
* (V - 1) * (V + 1) / 0.4e1) * (RC * C3 + (2 * RH)) * RB ^ 2 + 0.8e1 * RU * C1 * C3 * (-E * RC ^ 2 * RH * (C2 -
0.1e1) * (C3 ^ 3) / 0.2e1 + RC * E * RH * (RU * (V ^ 2 - 1) * C1 - RH * (C2 - 0.1e1)) * (C3 ^ 2) + ((-RU * (V -
1) * (V + 1) * ((C2 - 0.1e1) ^ 2 * (V - 1) * (V + 1) * GB / 0.4e1 + E) * C1 / 0.2e1 - E * RH * ((V ^ 2) - C2) / 0.2e1)
* RC ^ 2 - E * (V ^ 2) * RC * (RH ^ 2) + 0.2e1 * RU * C1 * E * (RH ^ 2) * (V - 1) * (V + 1)) * C3 - RC * RH * (RU
* (V - 1) * (V + 1) * ((C2 - 0.1e1) ^ 2 * (V - 1) * (V + 1) * GB / 0.4e1 + E) * C1 - E * C2 * RH)) * RB + 0.4e1 * E
* RU ^ 2 * C1 ^ 2 * (C3 ^ 2) * RC * RH * (V - 1) * (V + 1) * (RC * C3 + (2 * RH)) * (C2 - 0.1e1));

      SIGYYUE = -0.4e1 * (C3 + 1) * E * ((RC * E * RU * C1 * RH * (2 * RB + RC * (C2 - 1)) * C3 ^ 3) + (-(RU
* C1 * GB * RC * (V - 1) * (V + 1) * (C2 - 1) * RB ^ 2) / 0.2e1 + (((V - 1) * RU * (V + 1) * (-((C2 - 1) ^ 2 * GB) /
0.4e1 + E) * C1 - (E * RH * (V ^ 2 + C2))) * (RC ^ 2) + (4 * RU * C1 * E * RH ^ 2)) * RB - (2 * RC * E * RU * (V
^ 2 * RC - RH * (C2 - 1)) * C1 * RH)) * (C3 ^ 2) + 0.2e1 * (-GB * (V - 1) * ((-C2 / 0.4e1 - 0.1e1 / 0.4e1) * (RC ^
2) + (RU * C1 * RH)) * (V + 1) * (C2 - 1) * RB / 0.2e1 + RC * ((V - 1) * RU * (V + 1) * (-((C2 - 1) ^ 2 * GB) /
0.4e1 + E) * C1 - (E * C2 * RH)) * RH) * RB * C3 + (GB * RB ^ 2 * RC * RH * (C2 - 1) * (C2 + 1) * (V - 1) * (V
+ 1)) / 0.2e1) * C1 * RB * EPSYY0 / (-(4 * RC * E * RU * C1 * RB * RH * (2 * RB + RC * (C2 - 1)) * C3 ^ 4) +
((2 * RU * C1 * GB * RC * (V - 1) * (V + 1) * (C2 - 1) * RB ^ 3) + (-0.4e1 * (V - 1) * (RU * (-((C2 - 1) ^ 2 * GB) /
0.4e1 + E) * C1 + (E * C2 * RH)) * (V + 1) * (RC ^ 2) + (8 * E * RU * V ^ 2 * C1 * RC * RH) - (16 * RU * C1 * E
* RH ^ 2)) * (RB ^ 2) + (8 * RC * E * RU * C1 * RH * (RU * (V ^ 2 - 1) * C1 - RH * (C2 - 1)) * RB) + (4 * E * RU
^ 2 * C1 ^ 2 * RC ^ 2 * RH * (V - 1) * (V + 1) * (C2 - 1))) * (C3 ^ 3) + (-0.2e1 * GB * (V - 1) * (V + 1) * (-((V - 1)
* (V + 1) * (C2 + 1) * RC ^ 2) / 0.2e1 + (C1 * RC * RU * V ^ 2) - (2 * RU * C1 * RH)) * (C2 - 1) * (RB ^ 3) + ((2
* RU * C1 * V ^ 2 * GB * (V - 1) * (V + 1) * (C2 - 1) * RC ^ 2) - 0.8e1 * (V - 1) * (V + 1) * ((RU ^ 2 * GB * (V - 1)
* (V + 1) * (C2 - 1) * C1 ^ 2) / 0.4e1 + RU * (-((C2 - 1) ^ 2 * GB) / 0.4e1 + E) * RH * C1 + (E * C2 * RH ^ 2)) *
RC + (16 * E * RU * V ^ 2 * C1 * RH ^ 2)) * (RB ^ 2) - 0.4e1 * ((RU * (V - 1) * (V + 1) * (((C2 - 1) ^ 2 * (V - 1) *
(V + 1) * GB) / 0.4e1 + E) * C1 + (E * RH * (V ^ 2 - C2))) * (RC ^ 2) + (2 * E * V ^ 2 * RC * RH ^ 2) - (4 * RU *
C1 * E * RH ^ 2 * (V - 1) * (V + 1))) * RU * C1 * RB + (8 * E * RU ^ 2 * C1 ^ 2 * RC * RH ^ 2 * (V - 1) * (V + 1)
* (C2 - 1))) * (C3 ^ 2) - 0.8e1 * (GB * (V - 1) * (V + 1) * RH * (C2 - 1) * (-((V - 1) * (V + 1) * (C2 + 1) * RC) /
0.2e1 + (RU * C1 * V ^ 2)) * (RB ^ 2) / 0.2e1 + GB * (V - 1) * RU * (V + 1) * C1 * ((C2 / 0.4e1 + 0.1e1 / 0.4e1)
* (RC ^ 2) - (V ^ 2 * RC * RH) + (RU * C1 * RH * (V - 1) * (V + 1))) * (C2 - 1) * RB / 0.2e1 + RC * RU * C1 *
RH * (RU * (V - 1) * (V + 1) * (((C2 - 1) ^ 2 * (V - 1) * (V + 1) * GB) / 0.4e1 + E) * C1 - (E * C2 * RH))) * RB *
C3 - (2 * RU * C1 * GB * RB ^ 2 * RC * RH * (C2 - 1) * (C2 + 1) * (V - 1) * (V + 1)));

    % increment of elastic strain exceed yiled surface
        DEPSXXU0 = 0.4e1 * DEPSYY0 * ((VU * RH * (2 * RB + RC * (C2 - 1)) * RU * C1 * RC * E * C3 ^ 3) +
(-(RU * VU * C1 * GB * RC * (V - 1) * (V + 1) * (C2 - 1) * RB ^ 2) / 0.2e1 + (RU * ((V + 1) * VU * (V - 1) * (-((C2
- 1) ^ 2 * GB) / 0.4e1 + E) * (RC ^ 2) - (2 * V * RH * RC * E) + (4 * E * VU * RH ^ 2)) * C1 - (E * V * RC ^ 2 *
RH * (VU - 1) * (VU + 1))) * RB - 0.2e1 * (V * ((V * VU) + C2 / 0.2e1 - 0.1e1 / 0.2e1) * RC - (VU * RH * (C2 -
1))) * RC * RU * E * C1 * RH) * (C3 ^ 2) + 0.2e1 * (-GB * (V + 1) * (V - 1) * (C2 - 1) * ((VU * RH) - (V * RC) /
0.2e1) * (RB ^ 2) / 0.2e1 + ((V * GB * (C2 - 1) ^ 2 * (V - 1) * (V + 1) * RC ^ 2) / 0.8e1 + (V + 1) * VU * (V - 1) *
RH * (-((C2 - 1) ^ 2 * GB) / 0.4e1 + E) * RC - (2 * E * V * RH ^ 2)) * RB - (V * RH * (-RC + RH * (C2 - 1)) * RC
* E)) * RU * C1 * C3 + (V * GB * (V + 1) * (V - 1) * RH * RB * (C2 - 1) * (2 * RB + RC * (C2 - 1)) * RU * C1) /
0.2e1) * RB * (C3 + 1) / (-(4 * RH * RB * (2 * RB + RC * (C2 - 1)) * RU * C1 * RC * E * C3 ^ 4) + ((2 * RU * C1
* GB * RC * (V - 1) * (V + 1) * (C2 - 1) * RB ^ 3) + (0.8e1 * RU * (-(V + 1) * (V - 1) * (-((C2 - 1) ^ 2 * GB) / 0.4e1
+ E) * (RC ^ 2) / 0.2e1 + (E * VU * V * RC * RH) - (2 * E * RH ^ 2)) * C1 - (4 * E * C2 * RC ^ 2 * RH * (VU - 1)
* (VU + 1))) * (RB ^ 2) + 0.8e1 * RC * RU * E * ((RU * (V ^ 2 - 1) * C1) - (V * (VU - V) * RC) / 0.2e1 - (RH * (C2
- 1))) * C1 * RH * RB + (4 * E * RU ^ 2 * C1 ^ 2 * RC ^ 2 * RH * (V - 1) * (V + 1) * (C2 - 1))) * (C3 ^ 3) + (-0.2e1
* GB * (V - 1) * (C2 - 1) * (V + 1) * ((RU * (V * RC * VU - 2 * RH) * C1) - (RC ^ 2 * (VU - 1) * (VU + 1) * (C2 +
1)) / 0.2e1) * (RB ^ 3) + (-(2 * RU ^ 2 * GB * RC * (V - 1) ^ 2 * (V + 1) ^ 2 * (C2 - 1) * C1 ^ 2) + 0.16e2 * RU *
((V * VU * GB * (V - 1) * (V + 1) * (C2 - 1) * RC ^ 2) / 0.8e1 - (V + 1) * (V - 1) * RH * (-((C2 - 1) ^ 2 * GB) /
0.4e1 + E) * RC / 0.2e1 + (E * VU * V * RH ^ 2)) * C1 - (8 * E * C2 * RC * RH ^ 2 * (VU - 1) * (VU + 1))) * (RB
^ 2) - 0.4e1 * RU * C1 * ((V - 1) * RU * ((((C2 - 1) ^ 2 * (V - 1) * (V + 1) * GB) / 0.4e1 + E) * (RC ^ 2) - (4 * E *
RH ^ 2)) * (V + 1) * C1 + (((V * VU - C2) * RC + 2 * V * VU * RH) * RH * RC * E)) * RB + (8 * E * RU ^ 2 * C1
^ 2 * RC * RH ^ 2 * (V - 1) * (V + 1) * (C2 - 1))) * (C3 ^ 2) - 0.8e1 * (GB * ((RU * VU * V * C1) - (RC * (VU - 1)
* (VU + 1) * (C2 + 1)) / 0.2e1) * (V - 1) * (C2 - 1) * (V + 1) * RH * (RB ^ 2) / 0.2e1 + GB * (V - 1) * RU * C1 *
(C2 - 1) * ((RU * RH * (V - 1) * (V + 1) * C1) - RC * ((-C2 / 0.4e1 - 0.1e1 / 0.4e1) * RC + (V * VU * RH))) * (V
+ 1) * RB / 0.2e1 + RC * RU * C1 * (RU * (V - 1) * (V + 1) * (((C2 - 1) ^ 2 * (V - 1) * (V + 1) * GB) / 0.4e1 + E)
* C1 - (E * C2 * RH)) * RH) * RB * C3 - (2 * RU * C1 * GB * RB ^ 2 * RC * RH * (C2 - 1) * (C2 + 1) * (V - 1) *
(V + 1)));

    DEPSYYU0 = 0.4e1 * RB * DEPSYY0 * (-GB * (V + 1) * (V - 1) * ((RU * VU * V * C1) - (RU * C1 * C3) - (RC
* (VU - 1) * (VU + 1) * (C2 + 1)) / 0.2e1) * (C3 * RC + 2 * RH) * (C2 - 1) * RB ^ 2 / 0.2e1 + 0.2e1 * (C3 * RC +
2 * RH) * (-(RU * C1 * E * C3 ^ 2 * RH) + ((-(V + 1) * (V - 1) * (-((C2 - 1) ^ 2) * GB / 0.4e1 + E) * RU * C1 /
0.2e1 - (E * C2 * RH * (VU - 1) * (VU + 1)) / 0.2e1) * RC + (RU * C1 * E * VU * V * RH)) * C3 - RU * VU * V *
C1 * GB * RC * ((C2 - 1) ^ 2) * (V - 1) * (V + 1) / 0.8e1) * RB + (RH * (-RC * (C2 - 1) * C3 ^ 2 + (V * (2 * V +
VU * (C2 - 1)) * RC - 2 * RH * (C2 - 1)) * C3 + 2 * V * VU * (-RC + RH * (C2 - 1))) * C3 * RU * C1 * RC * E)) *
(C3 + 1) / (-0.2e1 * GB * (V + 1) * (V - 1) * ((RU * VU * V * C1) - (RU * C1 * C3) - (RC * (VU - 1) * (VU + 1) *
(C2 + 1)) / 0.2e1) * (C3 * RC + 2 * RH) * (C2 - 1) * C3 * RB ^ 3 + 0.8e1 * (C3 * RC + 2 * RH) * (-(RU * C1 * E
* C3 ^ 3 * RH) + ((-(V + 1) * (V - 1) * (-((C2 - 1) ^ 2) * GB / 0.4e1 + E) * RU * C1 / 0.2e1 - (E * C2 * RH * (VU
- 1) * (VU + 1)) / 0.2e1) * RC + (RU * C1 * E * VU * V * RH)) * (C3 ^ 2) - GB * (V + 1) * (V - 1) * (C2 - 1) * (-V
* RC * VU + RU * (V - 1) * (V + 1) * C1) * RU * C1 * C3 / 0.4e1 - RU * C1 * RC * GB * (C2 - 1) * (C2 + 1) * (V
- 1) * (V + 1) / 0.8e1) * RB ^ 2 + 0.8e1 * C3 * RU * (-(E * RC ^ 2 * RH * (C2 - 1) * C3 ^ 3) / 0.2e1 + RH * ((RU
* (V ^ 2 - 1) * C1) - (V * (VU - V) * RC) / 0.2e1 - (RH * (C2 - 1))) * RC * E * (C3 ^ 2) + ((-RU * (V - 1) * (V + 1)
* (((C2 - 1) ^ 2) * (V - 1) * (V + 1) * GB / 0.4e1 + E) * C1 / 0.2e1 - (E * RH * (V * VU - C2)) / 0.2e1) * (RC ^ 2)
- (E * RC * RH ^ 2 * V * VU) + (2 * RU * C1 * E * RH ^ 2 * (V - 1) * (V + 1))) * C3 - RH * (RU * (V - 1) * (V + 1)
* (((C2 - 1) ^ 2) * (V - 1) * (V + 1) * GB / 0.4e1 + E) * C1 - (E * C2 * RH)) * RC) * C1 * RB + (4 * E * RU ^ 2 *
C1 ^ 2 * C3 ^ 2 * RC * RH * (V - 1) * (V + 1) * (C3 * RC + 2 * RH) * (C2 - 1)));

    % caculated cohesion of brick unit by value of hardening(softening) parameter K
    DKU = sqrt((6 * DEPSPXXU ^ 2 + 6 * DEPSPYYU ^ 2)) / 0.3e1;
    KU  = KU + DKU;
    KUMAX = ((2 * LC * EPS0U * FCU + 3 * GCU) / LC / FCU) / 0.2e1;
    % find compressive strength by hardening parameter K:
    if KU <= EPS0U
        SIGCU = (FCU * (-2 * KU ^ 2 / EPS0U ^ 2 + 4 * KU / EPS0U + 1)) / 0.3e1;
        KCU = (FCU * (-4 * KU / EPS0U ^ 2 + 4 / EPS0U)) / 0.3e1;
    else
        if KU < KUMAX
```

```
         SIGCU = FCU * (0.1e1 - 0.4e1 / 0.9e1 * FCU ^ 2 * LC ^ 2 / GCU ^ 2 * (KU - EPS0U) ^ 2);
         KCU = -0.8e1 / 0.9e1 * FCU ^ 3 * LC ^ 2 / GCU ^ 2 * (KU - EPS0U);
      else
         SIGCU = 0;
         KCU   = -0.8e1 / 0.9e1 * FCU ^ 3 * LC ^ 2 / GCU ^ 2 * (KUMAX - EPS0U);
      end
   end
   % find critical stress:
      CU = (0.1e1 - sin(PHIU)) / cos(PHIU) * SIGCU / 0.2e1;

      SIGXXUC1 = -0.18e2 * (-0.6e1 * SIGXXUE * sin(PHIU) / (0.3e1 - sin(PHIU)) - 0.6e1 * SIGYYUE *
sin(PHIU) / (0.3e1 - sin(PHIU)) + 0.3e1 * sqrt(SIGXXUE ^ 2 - SIGXXUE * SIGYYUE + SIGYYUE ^ 2)) *
SIGXXUE * CU * cos(PHIU) / (0.3e1 - sin(PHIU)) / (0.36e2 * SIGXXUE ^ 2 * sin(PHIU) ^ 2 / (0.3e1 - sin(PHIU))
^ 2 + 0.72e2 * SIGXXUE * SIGYYUE * sin(PHIU) ^ 2 / (0.3e1 - sin(PHIU)) ^ 2 + 0.36e2 * SIGYYUE ^ 2 *
sin(PHIU) ^ 2 / (0.3e1 - sin(PHIU)) ^ 2 - 0.9e1 * SIGXXUE ^ 2 + 0.9e1 * SIGXXUE * SIGYYUE - 0.9e1 *
SIGYYUE ^ 2);

      SIGYYUC1 = 0.18e2 * (0.6e1 * SIGXXUE * sin(PHIU) / (0.3e1 - sin(PHIU)) + 0.6e1 * SIGYYUE *
sin(PHIU) / (0.3e1 - sin(PHIU)) + 0.3e1 * sqrt(SIGXXUE ^ 2 - SIGXXUE * SIGYYUE + SIGYYUE ^ 2)) *
SIGYYUE * CU * cos(PHIU) / (0.3e1 - sin(PHIU)) / (0.36e2 * SIGXXUE ^ 2 * sin(PHIU) ^ 2 / (0.3e1 - sin(PHIU))
^ 2 + 0.72e2 * SIGXXUE * SIGYYUE * sin(PHIU) ^ 2 / (0.3e1 - sin(PHIU)) ^ 2 + 0.36e2 * SIGYYUE ^ 2 *
sin(PHIU) ^ 2 / (0.3e1 - sin(PHIU)) ^ 2 - 0.9e1 * SIGXXUE ^ 2 + 0.9e1 * SIGXXUE * SIGYYUE - 0.9e1 *
SIGYYUE ^ 2);

      SIGXXUC2 = 0.18e2 * (0.6e1 * SIGXXUE * sin(PHIU) / (0.3e1 - sin(PHIU)) + 0.6e1 * SIGYYUE *
sin(PHIU) / (0.3e1 - sin(PHIU)) + 0.3e1 * sqrt(SIGXXUE ^ 2 - SIGXXUE * SIGYYUE + SIGYYUE ^ 2)) *
SIGXXUE * CU * cos(PHIU) / (0.3e1 - sin(PHIU)) / (0.36e2 * SIGXXUE ^ 2 * sin(PHIU) ^ 2 / (0.3e1 - sin(PHIU))
^ 2 + 0.72e2 * SIGXXUE * SIGYYUE * sin(PHIU) ^ 2 / (0.3e1 - sin(PHIU)) ^ 2 + 0.36e2 * SIGYYUE ^ 2 *
sin(PHIU) ^ 2 / (0.3e1 - sin(PHIU)) ^ 2 - 0.9e1 * SIGXXUE ^ 2 + 0.9e1 * SIGXXUE * SIGYYUE - 0.9e1 *
SIGYYUE ^ 2);

      SIGYYUC2 = -0.18e2 * (-0.6e1 * SIGXXUE * sin(PHIU) / (0.3e1 - sin(PHIU)) - 0.6e1 * SIGYYUE *
sin(PHIU) / (0.3e1 - sin(PHIU)) + 0.3e1 * sqrt(SIGXXUE ^ 2 - SIGXXUE * SIGYYUE + SIGYYUE ^ 2)) *
SIGYYUE * CU * cos(PHIU) / (0.3e1 - sin(PHIU)) / (0.36e2 * SIGXXUE ^ 2 * sin(PHIU) ^ 2 / (0.3e1 - sin(PHIU))
^ 2 + 0.72e2 * SIGXXUE * SIGYYUE * sin(PHIU) ^ 2 / (0.3e1 - sin(PHIU)) ^ 2 + 0.36e2 * SIGYYUE ^ 2 *
sin(PHIU) ^ 2 / (0.3e1 - sin(PHIU)) ^ 2 - 0.9e1 * SIGXXUE ^ 2 + 0.9e1 * SIGXXUE * SIGYYUE - 0.9e1 *
SIGYYUE ^ 2);

   if SIGXXUE/SIGXXUC1 > 0
      SIGXXUC = SIGXXUC1;
   else
      SIGXXUC = SIGXXUC2;
   end
   if SIGYYUE/SIGYYUC1 > 0
      SIGYYUC = SIGYYUC1;
   else
      SIGYYUC = SIGYYUC2;
   end

   % drucker pragar plasiticy model of head joint
   % elastic predictor of head joint:
      SIGXXHE = -0.4e1 * (C3 + 1) * E * RB * ((((2 * RU * C1 - RC * (C2 + 1)) * RB + RU * C1 * RC * (C2 - 1))
* RC * E * RU * C1 * C3 ^ 2) + (-GB * ((RU * (V ^ 2 * RC - RC + 2 * RH) * C1) - ((V - 1) * (V + 1) * (C2 + 1) *
RC ^ 2) / 0.2e1) * (C2 - 1) * (RB ^ 2) / 0.2e1 + 0.4e1 * RU * C1 * (RU * (-GB * (V - 1) * (V + 1) * (C2 - 1) * RC
/ 0.4e1 + (E * RH)) * C1 - (-GB * ((C2 + 3) * V ^ 2 + C2 - 1) * (C2 - 1) * RC / 0.8e1 + (((C2 - 1) ^ 2) * GB /
0.4e1 + (E * C2)) * RH) * RC / 0.2e1) * RB + 0.2e1 * RC * (RU ^ 2) * (C1 ^ 2) * ((-((C2 - 1) ^ 2) * (V - 1) * (V +
1) * GB / 0.4e1 - E) * RC + (E * RH * (C2 - 1)))) * C3 - GB * RU * C1 * RB * (-2 * RB * RH + RC * (RC * (C2 +
1) - RH * (C2 - 1))) * (C2 - 1) / 0.2e1) * V * EPSYY0 / (-(4 * RC * E * RU * C1 * RB * RH * (2 * RB + RC * (C2
- 1)) * C3 ^ 4) + (0.2e1 * RU * C1 * GB * RC * (V - 1) * (V + 1) * (C2 - 1) * (RB ^ 3) + (-0.4e1 * RU * ((V - 1) *
(V + 1) * (-((C2 - 1) ^ 2) * GB / 0.4e1 + E) * (RC ^ 2) - (2 * E * V ^ 2 * RC * RH) + (4 * E * RH ^ 2)) * C1 - (4 *
E * C2 * RC ^ 2 * RH * (V - 1) * (V + 1))) * (RB ^ 2) + (8 * RC * E * RU * C1 * RH * (RU * (V ^ 2 - 1) * C1 - RH
* (C2 - 1)) * RB) + (4 * E * RU ^ 2 * C1 ^ 2 * RC ^ 2 * RH * (V - 1) * (V + 1) * (C2 - 1))) * (C3 ^ 3) + (-0.2e1 *
GB * (V - 1) * (V + 1) * ((RU * (V ^ 2 * RC - 2 * RH) * C1) - ((V - 1) * (V + 1) * (C2 + 1) * RC ^ 2) / 0.2e1) * (C2
- 1) * (RB ^ 3) + (-0.2e1 * (RU ^ 2) * GB * RC * ((V - 1) ^ 2) * ((V + 1) ^ 2) * (C2 - 1) * (C1 ^ 2) - 0.8e1 * (-(V ^
2) * GB * (V - 1) * (V + 1) * (C2 - 1) * (RC ^ 2) / 0.4e1 + (V - 1) * (V + 1) * (-((C2 - 1) ^ 2) * GB / 0.4e1 + E) *
RH * RC - (2 * E * V ^ 2 * RH ^ 2)) * RU * C1 - (8 * RC * RH ^ 2 * E * C2 * (V - 1) * (V + 1))) * (RB ^ 2) - 0.4e1
* ((V - 1) * RU * (V + 1) * ((((C2 - 1) ^ 2) * (V - 1) * (V + 1) * GB / 0.4e1 + E) * (RC ^ 2) - (4 * E * RH ^ 2)) * C1
+ (RC * E * ((V ^ 2 - C2) * RC + 2 * V ^ 2 * RH) * RH)) * RU * C1 * RB + (8 * E * RU ^ 2 * C1 ^ 2 * RC * RH ^
2 * (V - 1) * (V + 1) * (C2 - 1))) * (C3 ^ 2) - 0.8e1 * (GB * (V - 1) * (V + 1) * RH * (C2 - 1) * (-((V - 1) * (V + 1) *
(C2 + 1) * RC) / 0.2e1 + (RU * C1 * V ^ 2)) * (RB ^ 2) / 0.2e1 + GB * (V - 1) * RU * (V + 1) * C1 * ((C2 / 0.4e1
+ 0.1e1 / 0.4e1) * (RC ^ 2) - (V ^ 2 * RC * RH) + (RU * C1 * RH * (V - 1) * (V + 1))) * (C2 - 1) * RB / 0.2e1 +
RC * RU * C1 * RH * (RU * (V - 1) * (V + 1) * (((C2 - 1) ^ 2) * (V - 1) * (V + 1) * GB / 0.4e1 + E) * C1 - (E * C2
* RH))) * RB * C3 - 0.2e1 * RU * C1 * GB * (RB ^ 2) * RC * RH * (C2 - 1) * (C2 + 1) * (V - 1) * (V + 1));

      SIGYYHE = -0.4e1 * (C3 + 1) * E * ((RC * E * RU * C1 * RB * RH * (2 * RB + RC * (C2 - 1)) * C3 ^ 3) +
RC * (-(RU * C1 * GB * (V - 1) * (V + 1) * (C2 - 1) * RB ^ 3) / 0.2e1 + ((2 * C1 ^ 2 * E * RU ^ 2 * V ^ 2) - RU *
((((C2 - 1) ^ 2 * (V - 1) * (V + 1) * GB) / 0.4e1 + (E * (C2 * V ^ 2 + 1))) * RC + (2 * E * RH * V ^ 2)) * C1 + (E *
C2 * RC * RH * (V - 1) * (V + 1))) * (RB ^ 2) + (E * RU * C1 * ((V ^ 2 * (C2 - 1) * RC - 4 * V ^ 2 * RH + 4 * RH)
* RU * C1 + V ^ 2 * RC * RH * (C2 + 1)) * RB) - (2 * E * RU ^ 2 * C1 ^ 2 * RC * RH * (V - 1) * (V + 1) * (C2 -
1))) * (C3 ^ 2) + 0.2e1 * (-GB * (-((V - 1) * (V + 1) * (C2 + 1) * RC ^ 2) / 0.4e1 + (C1 * RH * RU * V ^ 2)) * (C2
- 1) * (RB ^ 2) / 0.2e1 + 0.2e1 * (RU * (-(GB * (V - 1) * (V + 1) * (C2 - 1) * RC) / 0.4e1 + (E * RH * V ^ 2)) * C1
- RC * (GB * (0.1e1 / 0.4e1 - (C2 ^ 2) / 0.4e1) * RC + (((C2 - 1) ^ 2 * GB) / 0.4e1 + (E * C2)) * RH) * (V ^ 2) /
0.2e1) * RU * C1 * RB + RC * RU * C1 * (((-((C2 - 1) ^ 2 * (V - 1) * (V + 1) * GB) / 0.4e1 - E) * RC + (E * V ^ 2
* RH * (C2 - 1))) * RU * C1 - (E * C2 * RC * RH))) * RB * C3 + (GB * RU * C1 * RB ^ 2 * (2 * V ^ 2 * RB * RH
+ RC * ((-C2 - 1) * RC + V ^ 2 * RH * (C2 - 1))) * (C2 - 1)) / 0.2e1) * EPSYY0 / (-(4 * RC * E * RU * C1 * RB *
RH * (2 * RB + RC * (C2 - 1)) * C3 ^ 4) + ((2 * RU * C1 * GB * RC * (V - 1) * (V + 1) * (C2 - 1) * RB ^ 3) + (-
0.4e1 * RU * ((V - 1) * (V + 1) * (-((C2 - 1) ^ 2 * GB) / 0.4e1 + E) * (RC ^ 2) - (2 * E * V ^ 2 * RC * RH) + (4 * E
* RH ^ 2)) * C1 - (4 * E * C2 * RC ^ 2 * RH * (V - 1) * (V + 1))) * (RB ^ 2) + (8 * RC * E * RU * C1 * RH * (RU *
(V ^ 2 - 1) * C1 - RH * (C2 - 1)) * RB) + (4 * E * RU ^ 2 * C1 ^ 2 * RC ^ 2 * RH * (V - 1) * (V + 1) * (C2 - 1))) *
(C3 ^ 3) + (-0.2e1 * GB * (V - 1) * (V + 1) * ((RU * (V ^ 2 * RC - 2 * RH) * C1) - ((V - 1) * (V + 1) * (C2 + 1) *
RC ^ 2) / 0.2e1) * (C2 - 1) * (RB ^ 3) + (-(2 * RU ^ 2 * GB * RC * (V - 1) ^ 2 * (V + 1) ^ 2 * (C2 - 1) * C1 ^ 2) -
0.8e1 * (-(V ^ 2 * GB * (V - 1) * (V + 1) * (C2 - 1) * RC ^ 2) / 0.4e1 + (V - 1) * (V + 1) * (-((C2 - 1) ^ 2 * GB) /
0.4e1 + E) * RH * RC - (2 * E * V ^ 2 * RH ^ 2)) * RU * C1 - (8 * RC * RH ^ 2 * E * C2 * (V - 1) * (V + 1))) * (RB
^ 2) - 0.4e1 * ((V - 1) * RU * (V + 1) * ((((C2 - 1) ^ 2 * (V - 1) * (V + 1) * GB) / 0.4e1 + E) * (RC ^ 2) - (4 * E *
RH ^ 2)) * C1 + (RC * E * ((V ^ 2 - C2) * RC + 2 * V ^ 2 * RH) * RH)) * RU * C1 * RB + (8 * E * RU ^ 2 * C1 ^
2 * RC * RH ^ 2 * (V - 1) * (V + 1) * (C2 - 1))) * (C3 ^ 2) - 0.8e1 * (GB * (V - 1) * (V + 1) * RH * (C2 - 1) * (-((V
- 1) * (V + 1) * (C2 + 1) * RC) / 0.2e1 + (RU * C1 * V ^ 2)) * (RB ^ 2) / 0.2e1 + GB * (V - 1) * RU * (V + 1) * C1
* ((C2 / 0.4e1 + 0.1e1 / 0.4e1) * (RC ^ 2) - (V ^ 2 * RC * RH) + (RU * C1 * RH * (V - 1) * (V + 1))) * (C2 - 1) *
RB / 0.2e1 + RC * RU * C1 * RH * (RU * (V - 1) * (V + 1) * (((C2 - 1) ^ 2 * (V - 1) * (V + 1) * GB) / 0.4e1 + E) *
C1 - (E * C2 * RH))) * RB * C3 - (2 * RU * C1 * GB * RB ^ 2 * RC * RH * (C2 - 1) * (C2 + 1) * (V - 1) * (V + 1)));

   % increment of elastic strain exceed yiled surface
      DEPSXXH0 = 0.4e1 * (-(RU * C1 * GB * (V - 1) * (V + 1) * (C3 * RC + 2 * RH) * (C2 - 1) * (C3 * VU - V)
* RB ^ 3) / 0.2e1 + ((2 * C1 * C3 ^ 3 * E * RC * RH * RU * V) + 0.2e1 * ((E * RU ^ 2 * V * (V - 1) * (V + 1) * C1
^ 2) - ((((C2 - 1) ^ 2 * GB) / 0.4e1 + (E * C2)) * (V + 1) * (V - 1) * RC + (2 * E * V ^ 2 * RH)) * VU * RU * C1 /
0.2e1 + (E * V * C2 * RC * RH * (VU - 1) * (VU + 1)) / 0.2e1) * RC * (C3 ^ 2) + 0.4e1 * (V + 1) * (V - 1) * ((C1
* E * RH * RU * V) - (-(V * GB * (C2 - 1) ^ 2 * RC) / 0.8e1 + (((C2 - 1) ^ 2 * GB) / 0.4e1 + (E * C2)) * VU * RH)
* RC / 0.2e1) * RU * C1 * C3 + (RU * V * C1 * GB * RC * RH * (C2 - 1) ^ 2 * (V - 1) * (V + 1)) / 0.2e1) * (RB ^
2) + ((RH * RC * (C2 - 1) * C3 ^ 2 + ((V + 1) * (V - 1) * (RC * (C2 - 1) - 4 * RH) * RU * C1 + VU * V * RC * RH
* (C2 + 1)) * C3 + 2 * RH * (RU * (V - 1) * (V + 1) * (C2 - 1) * C1 - RC * C2)) * V * C3 * RU * C1 * RC * E * RB)
- (2 * E * RU ^ 2 * V * C1 ^ 2 * C3 ^ 2 * RC ^ 2 * RH * (V - 1) * (V + 1) * (C2 - 1))) * DEPSYY0 * (C3 + 1) / (-
0.2e1 * GB * (V + 1) * (V - 1) * ((RU * VU * V * C1) - (RU * C1 * C3) - (RC * (VU - 1) * (VU + 1) * (C2 + 1)) /
0.2e1) * (C3 * RC + 2 * RH) * (C2 - 1) * C3 * (RB ^ 3) + 0.8e1 * (C3 * RC + 2 * RH) * (-(E * RU * C1 * C3 ^ 3
* RH) + ((-(V + 1) * (V - 1) * (-((C2 - 1) ^ 2 * GB) / 0.4e1 + E) * RC / 0.2e1 + (E * VU * V * RH)) * RU * C1 - (E
* C2 * RC * RH * (VU - 1) * (VU + 1)) / 0.2e1) * (C3 ^ 2) - (GB * (V + 1) * (V - 1) * (C2 - 1) * (RU * (V ^ 2 - 1) *
C1 - VU * V * RC) * RU * C1 * C3) / 0.4e1 - (RU * C1 * RC * GB * (C2 - 1) * (C2 + 1) * (V - 1) * (V + 1)) / 0.8e1)
* (RB ^ 2) + 0.8e1 * C3 * RU * (-(E * RC ^ 2 * RH * (C2 - 1) * C3 ^ 3) / 0.2e1 + RH * ((RU * (V ^ 2 - 1) * C1) -
(V * (VU - V) * RC) / 0.2e1 - (RH * (C2 - 1))) * RC * E * (C3 ^ 2) + (-(V + 1) * (V - 1) * ((((C2 - 1) ^ 2 * GB * V ^
2) / 0.4e1 - ((C2 - 1) ^ 2 * GB) / 0.4e1 + E) * (RC ^ 2) - (4 * E * RH ^ 2)) * RU * C1 / 0.2e1 - (((V * VU - C2) *
RC + 2 * VU * V * RH) * RH * RC * E) / 0.2e1) * C3 - RH * (RU * (V - 1) * (V + 1) * (((C2 - 1) ^ 2 * GB * V ^ 2)
/ 0.4e1 - ((C2 - 1) ^ 2 * GB) / 0.4e1 + E) * C1 - (E * C2 * RH)) * RC) * C1 * RB + (4 * E * RU ^ 2 * C1 ^ 2 * C3
^ 2 * RC * RH * (V - 1) * (V + 1) * (C3 * RC + 2 * RH) * (C2 - 1)));

    DEPSYYH0 = 0.8e1 * (-(RH * RB * (2 * RB + RC * (C2 - 1)) * RU * C1 * E * C3 ^ 3) / 0.2e1 + ((RU * C1 *
GB * (V - 1) * (V + 1) * (C2 - 1) * RB ^ 3) / 0.4e1 + ((-(V + 1) * (V - 1) * (-((C2 - 1) ^ 2 * GB) / 0.4e1 + E) * RC
/ 0.2e1 + (E * VU * V * RH)) * RU * C1 - (E * C2 * RC * RH * (VU - 1) * (VU + 1)) / 0.2e1) * (RB ^ 2) + 0.2e1 *
RH * ((RU * (V ^ 2 - 1) * C1) - (VU * V * (C2 + 1) * RC) / 0.4e1) * RU * C1 * E * RB + (RH * RU ^ 2 * C1 ^ 2 *
RC * E * (V - 1) * (V + 1) * (C2 - 1))) * (C3 ^ 2) - RB * (GB * (V + 1) * (V - 1) * ((RU * VU * V * C1) - (RC * (VU
- 1) * (VU + 1) * (C2 + 1)) / 0.2e1) * (C2 - 1) * (RB ^ 2) / 0.4e1 + GB * ((RU * (V ^ 2 - 1) * C1) - (VU * V * RC *
(C2 + 3)) / 0.4e1) * (V + 1) * (V - 1) * (C2 - 1) * RU * C1 * RB / 0.2e1 + ((V + 1) * (V - 1) * RU * (((C2 - 1) ^ 2 *
(V - 1) * (V + 1) * GB) / 0.4e1 + E) * C1 - (E * C2 * RH)) * RU * C1 * RC) * C3 - (RU * C1 * GB * RB ^ 2 * RC
* (C2 - 1) * (C2 + 1) * (V - 1) * (V + 1)) / 0.4e1) * DEPSYY0 * RC * (C3 + 1) / (-(4 * RH * RB * (2 * RB + RC *
(C2 - 1)) * RU * C1 * RC * E * C3 ^ 4) + ((2 * RU * C1 * GB * RC * (V - 1) * (V + 1) * (C2 - 1) * RB ^ 3) + (0.8e1
* RU * (-(V + 1) * (V - 1) * (-((C2 - 1) ^ 2 * GB) / 0.4e1 + E) * (RC ^ 2) / 0.2e1 + (E * VU * V * RC * RH) - (2 *
E * RH ^ 2)) * C1 - (4 * E * C2 * RC ^ 2 * RH * (VU - 1) * (VU + 1))) * (RB ^ 2) + 0.8e1 * RH * ((RU * (V ^ 2 -
1) * C1) - (V * (VU - V) * RC) / 0.2e1 - (RH * (C2 - 1))) * RU * C1 * RC * E * RB + (4 * E * RU ^ 2 * C1 ^ 2 *
RC ^ 2 * RH * (V - 1) * (V + 1) * (C2 - 1))) * (C3 ^ 3) + (-0.2e1 * GB * (V + 1) * (V - 1) * (C2 - 1) * ((RU * (VU *
V * RC - 2 * RH) * C1) - (RC ^ 2 * (VU - 1) * (VU + 1) * (C2 + 1)) / 0.2e1) * (RB ^ 3) + (-(2 * RU ^ 2 * GB * RC
* (V - 1) ^ 2 * (V + 1) ^ 2 * (C2 - 1) * C1 ^ 2) + 0.16e2 * ((VU * V * GB * (V - 1) * (V + 1) * (C2 - 1) * RC ^ 2) /
0.8e1 - (V + 1) * (V - 1) * RH * (-((C2 - 1) ^ 2 * GB) / 0.4e1 + E) * RC / 0.2e1 + (E * VU * V * RH ^ 2)) * RU *
C1 - (8 * E * C2 * RC * RH ^ 2 * (VU - 1) * (VU + 1))) * (RB ^ 2) - 0.4e1 * ((V + 1) * (V - 1) * ((((C2 - 1) ^ 2 * (V
- 1) * (V + 1) * GB) / 0.4e1 + E) * (RC ^ 2) - (4 * E * RH ^ 2)) * RU * C1 + (((V * VU - C2) * RC + 2 * VU * V *
RH) * RH * RC * E)) * RU * C1 * RB + (8 * E * RU ^ 2 * C1 ^ 2 * RC * RH ^ 2 * (V - 1) * (V + 1) * (C2 - 1))) *
(C3 ^ 2) - 0.8e1 * RB * (GB * (V + 1) * (V - 1) * ((RU * VU * V * C1) - (RC * (VU - 1) * (VU + 1) * (C2 + 1)) /
0.2e1) * RH * (C2 - 1) * (RB ^ 2) / 0.2e1 + RU * C1 * GB * (V - 1) * (V + 1) * (C2 - 1) * ((RU * RH * (V - 1) * (V
+ 1) * C1) - ((-C2 / 0.4e1 - 0.1e1 / 0.4e1) * RC + (VU * V * RH)) * RC) * RB / 0.2e1 + ((V + 1) * (V - 1) * RU *
(((C2 - 1) ^ 2 * (V - 1) * (V + 1) * GB) / 0.4e1 + E) * C1 - (E * C2 * RH)) * RH * RU * C1 * RC) * C3 - (2 * RU *
C1 * GB * RB ^ 2 * RC * RH * (C2 - 1) * (C2 + 1) * (V - 1) * (V + 1)));

    % caculated cohesion of brick unit by value of hardening(softening) parameter K
    DKH = sqrt((6 * DEPSPXXH ^ 2 + 6 * DEPSPYYH ^ 2)) / 0.3e1;
    KH  = KH + DKH;
    KHMAX = ((2 * LC * EPS0M * FCM + 3 * GCM) / LC / FCM) / 0.2e1;
    % find compressive strength by hardening parameter K:
    if KH <= EPS0M
       SIGCH = (FCM * (-2 * KH ^ 2 / EPS0M ^ 2 + 4 * KH / EPS0M + 1)) / 0.3e1;
       KCH = (FCM * (-4 * KH / EPS0M ^ 2 + 4 / EPS0M)) / 0.3e1;
    else
       if KH < KHMAX
          SIGCH = FCM * (0.1e1 - 0.4e1 / 0.9e1 * FCM ^ 2 * LC ^ 2 / GCM ^ 2 * (KH - EPS0M) ^ 2);
          KCH = -0.8e1 / 0.9e1 * FCM ^ 3 * LC ^ 2 / GCM ^ 2 * (KH - EPS0M);
       else
          SIGCH = 0;
          KCH = -0.8e1 / 0.9e1 * FCM ^ 3 * LC ^ 2 / GCM ^ 2 * (KHMAX - EPS0M);
       end
    end
    % find critical stress:
       CH = (0.1e1 - sin(PHIM)) / cos(PHIM) * SIGCH / 0.2e1;
```

```
        SIGXXHC1 = -0.18e2 * (-0.6e1 * SIGXXHE * sin(PHIM) / (0.3e1 - sin(PHIM)) - 0.6e1 * SIGYYHE *
sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.3e1 * sqrt(SIGXXHE ^ 2 - SIGXXHE * SIGYYHE + SIGYYHE ^ 2)) *
SIGXXHE * CH * cos(PHIM) / (0.3e1 - sin(PHIM)) / (0.36e2 * SIGXXHE ^ 2 * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM))
^ 2 + 0.72e2 * SIGXXHE * SIGYYHE * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 + 0.36e2 * SIGYYHE ^ 2 *
sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 - 0.9e1 * SIGXXHE ^ 2 + 0.9e1 * SIGXXHE * SIGYYHE - 0.9e1 *
SIGYYHE ^ 2);

        SIGYYHC1 = 0.18e2 * (0.6e1 * SIGXXHE * sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.6e1 * SIGYYHE *
sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.3e1 * sqrt(SIGXXHE ^ 2 - SIGXXHE * SIGYYHE + SIGYYHE ^ 2)) *
SIGYYHE * CH * cos(PHIM) / (0.3e1 - sin(PHIM)) / (0.36e2 * SIGXXHE ^ 2 * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM))
^ 2 + 0.72e2 * SIGXXHE * SIGYYHE * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 + 0.36e2 * SIGYYHE ^ 2 *
sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 - 0.9e1 * SIGXXHE ^ 2 + 0.9e1 * SIGXXHE * SIGYYHE - 0.9e1 *
SIGYYHE ^ 2);

        SIGXXHC2 = 0.18e2 * (0.6e1 * SIGXXHE * sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.6e1 * SIGYYHE *
sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.3e1 * sqrt(SIGXXHE ^ 2 - SIGXXHE * SIGYYHE + SIGYYHE ^ 2)) *
SIGXXHE * CH * cos(PHIM) / (0.3e1 - sin(PHIM)) / (0.36e2 * SIGXXHE ^ 2 * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM))
^ 2 + 0.72e2 * SIGXXHE * SIGYYHE * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 + 0.36e2 * SIGYYHE ^ 2 *
sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 - 0.9e1 * SIGXXHE ^ 2 + 0.9e1 * SIGXXHE * SIGYYHE - 0.9e1 *
SIGYYHE ^ 2);

        SIGYYHC2 = -0.18e2 * (-0.6e1 * SIGXXHE * sin(PHIM) / (0.3e1 - sin(PHIM)) - 0.6e1 * SIGYYHE *
sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.3e1 * sqrt(SIGXXHE ^ 2 - SIGXXHE * SIGYYHE + SIGYYHE ^ 2)) *
SIGYYHE * CH * cos(PHIM) / (0.3e1 - sin(PHIM)) / (0.36e2 * SIGXXHE ^ 2 * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM))
^ 2 + 0.72e2 * SIGXXHE * SIGYYHE * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 + 0.36e2 * SIGYYHE ^ 2 *
sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 - 0.9e1 * SIGXXHE ^ 2 + 0.9e1 * SIGXXHE * SIGYYHE - 0.9e1 *
SIGYYHE ^ 2);

    if SIGXXHE/SIGXXHC1 > 0
        SIGXXHC = SIGXXHC1;
    else
        SIGXXHC = SIGXXHC2;
    end
    if SIGYYHE/SIGYYHC1 > 0
        SIGYYHC = SIGYYHC1;
    else
        SIGYYHC = SIGYYHC2;
    end

    % drucker pragar plasiticy model of bed joint
    % elastic predictor of bed joint:
        SIGXXBE = -0.4e1 * (C3 + 1) * (-GB * (V - 1) * (V + 1) * ((-C2 / 0.2e1 - 0.1e1 / 0.2e1) * RC + RU * C1) *
(C3 * RC + (2 * RH)) * (C2 - 0.1e1) * RB ^ 2 / 0.2e1 + (0.2e1 * RC * E * RH * ((-C2 / 0.2e1 - 0.1e1 / 0.2e1) *
RC + RU * C1) * (C3 ^ 2) + (-RU * C1 * GB * (C2 - 0.1e1) ^ 2 * (V - 1) * (V + 1) * RC ^ 2 / 0.4e1 - 0.2e1 * RC
* (RH ^ 2) * E * C2 + 0.4e1 * RU * C1 * E * (RH ^ 2)) * C3 - RU * C1 * GB * RC * RH * (C2 - 0.1e1) ^ 2 * (V -
1) * (V + 1) / 0.2e1) * RB + RC * E * RU * C1 * C3 * (RC * (C2 - 0.1e1) * C3 - 0.2e1 * RC + (0.2e1 * C2 - 0.2e1)
* RH) * RH) * E * RU * C1 * C3 * V * EPSYY0 / (-0.2e1 * GB * (V - 1) * (V + 1) * C3 * (-RU * C1 * C3 - (V - 1)
* (V + 1) * (C2 + 0.1e1) * RC / 0.2e1 + RU * C1 * (V ^ 2)) * (C3 * RC + (2 * RH)) * (C2 - 0.1e1) * RB ^ 3 - 0.4e1
* (0.2e1 * RU * C1 * E * (C3 ^ 3) * RH + ((V - 1) * (RU * (-(C2 - 0.1e1) ^ 2 * GB / 0.4e1 + E) * C1 + E * C2 *
RH) * (V + 1) * RC - 0.2e1 * RU * C1 * E * RH * (V ^ 2)) * (C3 ^ 2) + GB * (V - 1) * RU * (V + 1) * C1 * (-(V ^
2) * RC + RU * (V - 1) * (V + 1) * C1) * (C2 - 0.1e1) * C3 / 0.2e1 + RU * C1 * GB * RC * (C2 - 0.1e1) * (C2 +
0.1e1) * (V - 1) * (V + 1) / 0.4e1) * (C3 * RC + (2 * RH)) * RB ^ 2 + 0.8e1 * RU * C1 * C3 * (-E * RC ^ 2 * RH *
(C2 - 0.1e1) * (C3 ^ 3) / 0.2e1 + RC * E * RH * (RU * (V ^ 2 - 1) * C1 - RH * (C2 - 0.1e1)) * (C3 ^ 2) + ((-RU *
(V - 1) * (V + 1) * ((C2 - 0.1e1) ^ 2 * (V - 1) * (V + 1) * GB / 0.4e1 + E) * C1 / 0.2e1 - E * RH * ((V ^ 2) - C2) /
0.2e1) * RC ^ 2 - E * (V ^ 2) * RC * (RH ^ 2) + 0.2e1 * RU * C1 * E * (RH ^ 2) * (V - 1) * (V + 1)) * C3 - RC *
RH * (RU * (V - 1) * (V + 1) * ((C2 - 0.1e1) ^ 2 * (V - 1) * (V + 1) * GB / 0.4e1 + E) * C1 - E * C2 * RH)) * RB +
0.4e1 * E * RU ^ 2 * C1 ^ 2 * (C3 ^ 2) * RC * RH * (V - 1) * (V + 1) * (C3 * RC + (2 * RH)) * (C2 - 0.1e1));
```

```
        SIGYYBE = -0.4e1 * (C3 + 1) * E * RU * ((RC * E * RU * C1 * RH * (2 * RB + RC * (C2 - 1)) * C3 ^ 3) +
(-(RU * C1 * GB * RC * (V - 1) * (V + 1) * (C2 - 1) * RB ^ 2) / 0.2e1 + (((V - 1) * RU * (V + 1) * (-((C2 - 1) ^ 2 *
GB) / 0.4e1 + E) * C1 - (E * RH * (V ^ 2 + C2))) * (RC ^ 2) + (4 * RU * C1 * E * RH ^ 2)) * RB - (2 * RC * E *
RU * (V ^ 2 * RC - RH * (C2 - 1)) * C1 * RH)) * (C3 ^ 2) + 0.2e1 * (-GB * (V - 1) * ((-C2 / 0.4e1 - 0.1e1 / 0.4e1)
* (RC ^ 2) + (RU * C1 * RH)) * (V + 1) * (C2 - 1) * RB / 0.2e1 + RC * ((V - 1) * RU * (V + 1) * (-((C2 - 1) ^ 2 *
GB) / 0.4e1 + E) * C1 - (E * C2 * RH)) * RH) * RB * C3 + (GB * RB ^ 2 * RC * RH * (C2 - 1) * (C2 + 1) * (V -
1) * (V + 1)) / 0.2e1) * C1 * EPSYY0 / (-(4 * RC * E * RU * C1 * RB * RH * (2 * RB + RC * (C2 - 1)) * C3 ^ 4)
+ ((2 * RU * C1 * GB * RC * (V - 1) * (V + 1) * (C2 - 1) * RB ^ 3) + (-0.4e1 * (V - 1) * (RU * (-((C2 - 1) ^ 2 * GB)
/ 0.4e1 + E) * C1 + (E * C2 * RH)) * (V + 1) * (RC ^ 2) + (8 * E * RU * V ^ 2 * C1 * RC * RH) - (16 * RU * C1 *
E * RH ^ 2)) * (RB ^ 2) + (8 * RC * E * RU * C1 * RH * (RU * (V ^ 2 - 1) * C1 - RH * (C2 - 1)) * RB) + (4 * E *
RU ^ 2 * C1 ^ 2 * RC ^ 2 * RH * (V - 1) * (V + 1) * (C2 - 1))) * (C3 ^ 3) + (-0.2e1 * GB * (V - 1) * (V + 1) * (-((V
- 1) * (V + 1) * (C2 + 1) * RC ^ 2) / 0.2e1 + (C1 * RC * RU * V ^ 2) - (2 * RU * C1 * RH)) * (C2 - 1) * (RB ^ 3) +
((2 * RU * C1 * V ^ 2 * GB * (V - 1) * (V + 1) * (C2 - 1) * RC ^ 2) - 0.8e1 * (V - 1) * (V + 1) * ((RU ^ 2 * GB * (V
- 1) * (V + 1) * (C2 - 1) * C1 ^ 2) / 0.4e1 + RU * (-((C2 - 1) ^ 2 * GB) / 0.4e1 + E) * RH * C1 + (E * C2 * RH ^
2)) * RC + (16 * E * RU * V ^ 2 * C1 * RH ^ 2)) * (RB ^ 2) - 0.4e1 * ((RU * (V - 1) * (V + 1) * (((C2 - 1) ^ 2 * (V
- 1) * (V + 1) * GB) / 0.4e1 + E) * C1 + (E * RH * (V ^ 2 - C2))) * (RC ^ 2) + (2 * E * V ^ 2 * RC * RH ^ 2) - (4 *
RU * C1 * E * RH ^ 2 * (V - 1) * (V + 1))) * RU * C1 * RB + (8 * E * RU ^ 2 * C1 ^ 2 * RC * RH ^ 2 * (V - 1) * (V
+ 1) * (C2 - 1))) * (C3 ^ 2) - 0.8e1 * (GB * (V - 1) * (V + 1) * RH * (C2 - 1) * (-((V - 1) * (V + 1) * (C2 + 1) * RC)
/ 0.2e1 + (RU * C1 * V ^ 2)) * (RB ^ 2) / 0.2e1 + GB * (V - 1) * RU * (V + 1) * C1 * ((C2 / 0.4e1 + 0.1e1 / 0.4e1)
* (RC ^ 2) - (V ^ 2 * RC * RH) + (RU * C1 * RH * (V - 1) * (V + 1))) * (C2 - 1) * RB / 0.2e1 + RC * RU * C1 *
RH * (RU * (V - 1) * (V + 1) * (((C2 - 1) ^ 2 * (V - 1) * (V + 1) * GB) / 0.4e1 + E) * C1 - (E * C2 * RH))) * RB *
C3 - (2 * RU * C1 * GB * RB ^ 2 * RC * RH * (C2 - 1) * (C2 + 1) * (V - 1) * (V + 1)));

        TAUXYBE = -0.4e1 * (C3 + 1) * (-GB * (V - 1) * (V + 1) * ((-C2 / 0.2e1 - 0.1e1 / 0.2e1) * RC + RU * C1)
* (C3 * RC + (2 * RH)) * (C2 - 0.1e1) * RB ^ 2 / 0.2e1 + (0.2e1 * RC * E * RH * ((-C2 / 0.2e1 - 0.1e1 / 0.2e1) *
RC + RU * C1) * (C3 ^ 2) + (-RU * C1 * GB * (C2 - 0.1e1) ^ 2 * (V - 1) * (V + 1) * RC ^ 2 / 0.4e1 - 0.2e1 * RC
* (RH ^ 2) * E * C2 + 0.4e1 * RU * C1 * E * (RH ^ 2)) * C3 - RU * C1 * GB * RC * RH * (C2 - 0.1e1) ^ 2 * (V -
1) * (V + 1) / 0.2e1) * RB + RC * E * RU * C1 * C3 * (RC * (C2 - 0.1e1) * C3 - 0.2e1 * RC + (0.2e1 * C2 - 0.2e1)
* RH) * RH) * E * RU * C1 * C3 * V * EPSYY0 / (-0.2e1 * GB * (V - 1) * (V + 1) * C3 * (-RU * C1 * C3 - (V - 1)
* (V + 1) * (C2 + 0.1e1) * RC / 0.2e1 + RU * C1 * (V ^ 2)) * (C3 * RC + (2 * RH)) * (C2 - 0.1e1) * RB ^ 3 - 0.4e1
* (0.2e1 * RU * C1 * E * (C3 ^ 3) * RH + ((V - 1) * (RU * (-(C2 - 0.1e1) ^ 2 * GB / 0.4e1 + E) * C1 + E * C2 *
RH) * (V + 1) * RC - 0.2e1 * RU * C1 * E * RH * (V ^ 2)) * (C3 ^ 2) + GB * (V - 1) * RU * (V + 1) * C1 * (-(V ^
2) * RC + RU * (V - 1) * (V + 1) * C1) * (C2 - 0.1e1) * C3 / 0.2e1 + RU * C1 * GB * RC * (C2 - 0.1e1) * (C2 +
0.1e1) * (V - 1) * (V + 1) / 0.4e1) * (C3 * RC + (2 * RH)) * RB ^ 2 + 0.8e1 * RU * C1 * C3 * (-E * RC ^ 2 * RH *
(C2 - 0.1e1) * (C3 ^ 3) / 0.2e1 + RC * E * RH * (RU * (V ^ 2 - 1) * C1 - RH * (C2 - 0.1e1)) * (C3 ^ 2) + ((-RU *
(V - 1) * (V + 1) * ((C2 - 0.1e1) ^ 2 * (V - 1) * (V + 1) * GB / 0.4e1 + E) * C1 / 0.2e1 - E * RH * ((V ^ 2) - C2) /
0.2e1) * RC ^ 2 - E * (V ^ 2) * RC * (RH ^ 2) + 0.2e1 * RU * C1 * E * (RH ^ 2) * (V - 1) * (V + 1)) * C3 - RC *
RH * (RU * (V - 1) * (V + 1) * ((C2 - 0.1e1) ^ 2 * (V - 1) * (V + 1) * GB / 0.4e1 + E) * C1 - E * C2 * RH)) * RB +
0.4e1 * E * RU ^ 2 * C1 ^ 2 * (C3 ^ 2) * RC * RH * (V - 1) * (V + 1) * (C3 * RC + (2 * RH)) * (C2 - 0.1e1));

    % increment of elastic strain exceed yiled surface
        DEPSXXB0 = 0.4e1 * (E * RB * RC * RH * (C2 * VU + V) * C3 ^ 3 + (-(VU * GB * RC * (C2 - 1) * (C2 +
1) * (V - 1) * (V + 1) * RB ^ 2) / 0.4e1 + (((RU * (V ^ 2 - 1) * C1 - RH * (V * VU + C2)) * V * RC + 2 * VU * C2 *
RH ^ 2) * E * RB) - (2 * E * RU * V * C1 * RC * RH * (V - 1) * (V + 1))) * C3 ^ 2 + 0.2e1 * (-GB * (V + 1) * (V -
1) * (C2 - 1) * (C2 + 1) * ((VU * RH) - (V * RC) / 0.2e1) * RB / 0.4e1 + (V * (RU * (V ^ 2 - 1) * C1 - RH * C2) *
RH * E)) * RB * C3 + (V * GB * RB ^ 2 * RH * (C2 - 1) * (C2 + 1) * (V - 1) * (V + 1)) / 0.2e1) * DEPSYY0 * RU
* C1 * RC * (C3 + 0.1e1) / (-0.4e1 * RH * RB * (2 * RB + RC * (C2 - 1)) * RU * C1 * RC * E * C3 ^ 4 + ((2 * RU
* C1 * GB * RC * (V - 1) * (V + 1) * (C2 - 1) * RB ^ 3) + ((-0.4e1 * (V + 1) * (V - 1) * (-((C2 - 1) ^ 2 * GB) / 0.4e1
+ E) * RU * C1 - (4 * E * C2 * RH * (VU - 1) * (VU + 1))) * (RC ^ 2) + (8 * RU * C1 * E * VU * V * RC * RH) -
(16 * RU * C1 * E * RH ^ 2)) * (RB ^ 2) + 0.8e1 * RH * ((RU * (V ^ 2 - 1) * C1) - (V * (VU - V) * RC) / 0.2e1 -
(RH * (C2 - 1))) * RU * C1 * RC * E * RB + (4 * E * RU ^ 2 * C1 ^ 2 * RC ^ 2 * RH * (V - 1) * (V + 1) * (C2 - 1)))
* C3 ^ 3 + (-0.2e1 * GB * (V + 1) * (V - 1) * (C2 - 1) * (-(RC ^ 2 * (VU - 1) * (VU + 1) * (C2 + 1)) / 0.2e1 + (C1
* RC * RU * V * VU) - (2 * C1 * RH * RU)) * (RB ^ 3) + ((2 * RU * VU * V * C1 * GB * (V - 1) * (V + 1) * (C2 - 1)
* RC ^ 2) + (-(2 * RU ^ 2 * GB * (V - 1) ^ 2 * (V + 1) ^ 2 * (C2 - 1) * C1 ^ 2) - 0.8e1 * (V + 1) * (V - 1) * RH * (-
((C2 - 1) ^ 2 * GB) / 0.4e1 + E) * RU * C1 - (8 * E * C2 * RH ^ 2 * (VU - 1) * (VU + 1))) * RC + (16 * RU * C1 *
E * VU * V * RH ^ 2)) * (RB ^ 2) - 0.4e1 * (((V + 1) * (V - 1) * RU * (((C2 - 1) ^ 2 * (V - 1) * (V + 1) * GB) / 0.4e1
+ E) * C1 + (E * RH * (V * VU - C2))) * (RC ^ 2) + (2 * E * RC * RH ^ 2 * V * VU) - (4 * E * RU * C1 * RH ^ 2 *
(V - 1) * (V + 1))) * RU * C1 * RB + (8 * E * RU ^ 2 * C1 ^ 2 * RC * RH ^ 2 * (V - 1) * (V + 1) * (C2 - 1))) * C3 ^
2 - 0.8e1 * (GB * ((RU * VU * V * C1) - (RC * (VU - 1) * (VU + 1) * (C2 + 1)) / 0.2e1) * (V + 1) * (V - 1) * RH *
(C2 - 1) * (RB ^ 2) / 0.2e1 + GB * (V + 1) * (V - 1) * (C2 - 1) * ((C2 / 0.4e1 + 0.1e1 / 0.4e1) * (RC ^ 2) - (VU *
V * RC * RH) + (RU * RH * (V - 1) * (V + 1) * C1)) * RU * C1 * RB / 0.2e1 + ((V + 1) * (V - 1) * RU * (((C2 - 1)
^ 2 * (V - 1) * (V + 1) * GB) / 0.4e1 + E) * C1 - (E * C2 * RH)) * RH * RU * C1 * RC) * RB * C3 - (2 * RU * C1 *
GB * RB ^ 2 * RC * RH * (C2 - 1) * (C2 + 1) * (V - 1) * (V + 1)));

        DEPSYYB0 = 0.4e1 * DEPSYY0 * ((RH * (((-C2 * V * VU - V ^ 2) * RC + (2 * V ^ 2 - 2) * RU * C1) * RB
+ RU * C1 * RC * (V - 1) * (V + 1) * (C2 - 1)) * RC * E * C3 ^ 3) + (-GB * (V + 1) * (V - 1) * (-(V * VU * (C2 + 1)
* RC) / 0.2e1 + ((V + 1) * (V - 1) * RU * C1)) * (C2 - 1) * RC * (RB ^ 2) / 0.2e1 + ((-(V + 1) * (V - 1) * RU * (((C2
- 1) ^ 2) * (V - 1) * (V + 1) * GB / 0.4e1 + E) * C1 + (E * RH * (V * VU + C2))) * (RC ^ 2) - (2 * E * VU * V * C2
* RC * RH ^ 2) + (4 * E * RU * C1 * RH ^ 2 * (V - 1) * (V + 1))) * RB + (2 * E * RU * C1 * RC * RH ^ 2 * (V - 1)
* (V + 1) * (C2 - 1))) * (C3 ^ 2) - 0.2e1 * (GB * (V + 1) * (V - 1) * (C2 - 1) * ((C2 / 0.4e1 + 0.1e1 / 0.4e1) * (RC
^ 2) - (VU * V * RC * RH * (C2 + 1)) / 0.2e1 + (RU * RH * (V - 1) * (V + 1) * C1)) * RB / 0.2e1 + RH * ((V + 1)
* (V - 1) * RU * (((C2 - 1) ^ 2) * (V - 1) * (V + 1) * GB / 0.4e1 + E) * C1 - (E * C2 * RH)) * RC) * RB * C3 - GB
* RH * (RB ^ 2) * RC * (C2 - 1) * (C2 + 1) * (V - 1) * (V + 1) / 0.2e1) * RU * C1 * (C3 + 1) / (-(4 * RH * RB * (2
* RB + RC * (C2 - 1)) * RU * C1 * RC * E * C3 ^ 4) + (0.2e1 * RU * C1 * GB * RC * (V - 1) * (V + 1) * (C2 - 1)
* (RB ^ 3) + ((-0.4e1 * (V + 1) * (V - 1) * (-((C2 - 1) ^ 2) * GB / 0.4e1 + E) * RU * C1 - (4 * E * C2 * RH * (VU -
1) * (VU + 1))) * (RC ^ 2) + (8 * RU * C1 * E * VU * V * RC * RH) - (16 * RU * C1 * E * RH ^ 2)) * (RB ^ 2) +
0.8e1 * RH * ((RU * (V ^ 2 - 1) * C1) - (V * (VU - V) * RC) / 0.2e1 - (RH * (C2 - 1))) * RU * C1 * RC * E * RB +
(4 * E * RU ^ 2 * C1 ^ 2 * RC ^ 2 * RH * (V - 1) * (V + 1) * (C2 - 1))) * (C3 ^ 3) + (-0.2e1 * GB * (V + 1) * (V -
1) * (C2 - 1) * (-(RC ^ 2 * (VU - 1) * (VU + 1) * (C2 + 1)) / 0.2e1 + (C1 * RC * RU * V * VU) - (2 * C1 * RH *
RU)) * (RB ^ 3) + (0.2e1 * RU * VU * V * C1 * GB * (V - 1) * (V + 1) * (C2 - 1) * (RC ^ 2) + (-0.2e1 * (RU ^ 2) *
GB * ((V - 1) ^ 2) * ((V + 1) ^ 2) * (C2 - 1) * (C1 ^ 2) - 0.8e1 * (V + 1) * (V - 1) * RH * (-((C2 - 1) ^ 2) * GB /
0.4e1 + E) * RU * C1 - (8 * E * C2 * RH ^ 2 * (VU - 1) * (VU + 1))) * RC + (16 * RU * C1 * E * VU * V * RH ^
2)) * (RB ^ 2) - 0.4e1 * (((V + 1) * (V - 1) * RU * (((C2 - 1) ^ 2) * (V - 1) * (V + 1) * GB / 0.4e1 + E) * C1 + (E *
RH * (V * VU - C2))) * (RC ^ 2) + (2 * E * RC * RH ^ 2 * V * VU) - (4 * E * RU * C1 * RH ^ 2 * (V - 1) * (V + 1)))
* RU * C1 * RB + (8 * E * RU ^ 2 * C1 ^ 2 * RC * RH ^ 2 * (V - 1) * (V + 1) * (C2 - 1))) * (C3 ^ 2) - 0.8e1 * (GB
* ((RU * VU * V * C1) - (RC * (VU - 1) * (VU + 1) * (C2 + 1)) / 0.2e1) * (V + 1) * (V - 1) * RH * (C2 - 1) * (RB ^
2) / 0.2e1 + GB * (V + 1) * (V - 1) * (C2 - 1) * ((C2 / 0.4e1 + 0.1e1 / 0.4e1) * (RC ^ 2) - (VU * V * RC * RH) +
(RU * RH * (V - 1) * (V + 1) * C1)) * RU * C1 * RB / 0.2e1 + ((V + 1) * (V - 1) * RU * (((C2 - 1) ^ 2) * (V - 1) *
(V + 1) * GB / 0.4e1 + E) * C1 - (E * C2 * RH)) * RH * RU * C1 * RC) * RB * C3 - 0.2e1 * RU * C1 * GB * (RB
^ 2) * RC * RH * (C2 - 1) * (C2 + 1) * (V - 1) * (V + 1));

    DEPSXYB0 = DEPSYY0 * C3 * ((-(2 * RH * RU * C1 * RC * (VU - V) * C3 ^ 2) + ((-(RU * VU * (V - 1) * (V
+ 1) * C1 + (-VU ^ 2 + 1) * V * RH) * (C2 + 1) * RC ^ 2 + 2 * V * RU * C1 * (RU * (V ^ 2 - 1) * C1 + (-V * VU +
1) * RH) * RC - 4 * RU * VU * C1 * RH ^ 2) * C3) + 0.4e1 * (-(VU * (V - 1) * (V + 1) * (C2 + 1) * RC) / 0.2e1 +
(V * (RU * (V ^ 2 - 1) * C1 + RH))) * RH * RU * C1) * RB ^ 2 + (-(RH * RC * (C2 - 1) * (VU - V) * C3 ^ 2) + ((V
* (RU * (V - 1) * (V + 1) * (C2 - 1) * C1 + (VU * (C2 + 3) * V + C2 - 1) * RH) * RC) - 0.4e1 * (((V ^ 3 - V) * RU *
C1) + (VU * RH * (C2 - 1)) / 0.2e1) * RH) * C3 + (2 * V * RH * ((-C2 - 1) * RC + (C2 - 1) * (RU * (V ^ 2 - 1) *
C1 + RH)))) * RU * C1 * RC * RB - (2 * RU ^ 2 * V * C1 ^ 2 * C3 * RC ^ 2 * RH * (V - 1) * (V + 1) * (C2 - 1))) *
(C3 + 1) * E / (-0.2e1 * GB * (V + 1) * (V - 1) * ((RU * VU * V * C1) - (RU * C1 * C3) - (RC * (VU - 1) * (VU +
1) * (C2 + 1)) / 0.2e1) * (C3 * RC + 2 * RH) * (C2 - 1) * C3 * RB ^ 3 + 0.8e1 * (C3 * RC + 2 * RH) * (-(E * RU
* C1 * C3 ^ 3 * RH) + ((-(V + 1) * (V - 1) * (-((C2 - 1) ^ 2) * GB / 0.4e1 + E) * RU * C1 / 0.2e1 - (E * C2 * RH *
(VU - 1) * (VU + 1)) / 0.2e1) * RC + (E * RU * C1 * VU * V * RH)) * (C3 ^ 2) - GB * (V + 1) * (V - 1) * (C2 - 1) *
(-V * RC * VU + (V + 1) * (V - 1) * RU * C1) * RU * C1 * C3 / 0.4e1 - RU * C1 * RC * GB * (C2 - 1) * (C2 + 1)
* (V - 1) * (V + 1) / 0.8e1) * RB ^ 2 + 0.8e1 * C3 * RU * (-(E * RC ^ 2 * RH * (C2 - 1) * C3 ^ 3) / 0.2e1 + RH *
((RU * (V ^ 2 - 1) * C1) - (V * (VU - V) * RC) / 0.2e1 - (RH * (C2 - 1))) * RC * E * (C3 ^ 2) + ((-(V + 1) * (V - 1)
* RU * (((C2 - 1) ^ 2) * GB * (V ^ 2) / 0.4e1 - ((C2 - 1) ^ 2) * GB / 0.4e1 + E) * C1 / 0.2e1 - (E * RH * (V * VU -
C2)) / 0.2e1) * (RC ^ 2) - (E * RC * RH ^ 2 * V * VU) + (2 * E * RU * C1 * RH ^ 2 * (V - 1) * (V + 1))) * C3 - RH
* ((V + 1) * (V - 1) * RU * (((C2 - 1) ^ 2) * GB * (V ^ 2) / 0.4e1 - ((C2 - 1) ^ 2) * GB / 0.4e1 + E) * C1 - (E * C2
* RH)) * RC) * C1 * RB + (4 * E * RU ^ 2 * C1 ^ 2 * C3 ^ 2 * RC * RH * (V - 1) * (V + 1) * (C3 * RC + 2 * RH) *
(C2 - 1)));

    % caculated cohesion of brick unit by value of hardening(softening) parameter K
```

```
  DKB = sqrt((6 * DEPSPXXB ^ 2 + 6 * DEPSPXYB ^ 2 + 6 * DEPSPYYB ^ 2)) / 0.3e1;
  KB  = KB + DKB;
  KBMAX = ((2 * LC * EPS0M * FCM + 3 * GCM) / LC / FCM) / 0.2e1;
  % find compressive strength by hardening parameter K:
  if KB <= EPS0M
     SIGCB = (FCM * (-2 * KB ^ 2 / EPS0M ^ 2 + 4 * KB / EPS0M + 1)) / 0.3e1;
     KCB = (FCM * (-4 * KB / EPS0M ^ 2 + 4 / EPS0M)) / 0.3e1;
  else
     if KB < KBMAX
        SIGCB = FCM * (0.1e1 - 0.4e1 / 0.9e1 * FCM ^ 2 * LC ^ 2 / GCM ^ 2 * (KB - EPS0M) ^ 2);
        KCB = -0.8e1 / 0.9e1 * FCM ^ 3 * LC ^ 2 / GCM ^ 2 * (KB - EPS0M);
     else
        SIGCB = 0;
        KCB = -0.8e1 / 0.9e1 * FCM ^ 3 * LC ^ 2 / GCM ^ 2 * (KBMAX - EPS0M);
     end
  end
  % find critical stress:
     CB = (0.1e1 - sin(PHIM)) / cos(PHIM) * SIGCB / 0.2e1;

     SIGXXBC1 = 0.18e2 * (0.6e1 * SIGXXBE * sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.6e1 * SIGYYBE *
sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.3e1 * sqrt(SIGXXBE ^ 2 - SIGXXBE * SIGYYBE + SIGYYBE ^ 2 + (3 *
TAUXYBE ^ 2))) * SIGXXBE * CB * cos(PHIM) / (0.3e1 - sin(PHIM)) / (0.36e2 * SIGXXBE ^ 2 * sin(PHIM) ^
2 / (0.3e1 - sin(PHIM)) ^ 2 + 0.72e2 * SIGXXBE * SIGYYBE * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 + 0.36e2
* SIGYYBE ^ 2 * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 - 0.9e1 * SIGXXBE ^ 2 + 0.9e1 * SIGXXBE * SIGYYBE
- 0.9e1 * SIGYYBE ^ 2 - (27 * TAUXYBE ^ 2));

     SIGYYBC1 = 0.18e2 * (0.6e1 * SIGXXBE * sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.6e1 * SIGYYBE *
sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.3e1 * sqrt(SIGXXBE ^ 2 - SIGXXBE * SIGYYBE + SIGYYBE ^ 2 + (3 *
TAUXYBE ^ 2))) * SIGYYBE * CB * cos(PHIM) / (0.3e1 - sin(PHIM)) / (0.36e2 * SIGXXBE ^ 2 * sin(PHIM) ^
2 / (0.3e1 - sin(PHIM)) ^ 2 + 0.72e2 * SIGXXBE * SIGYYBE * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 + 0.36e2
* SIGYYBE ^ 2 * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 - 0.9e1 * SIGXXBE ^ 2 + 0.9e1 * SIGXXBE * SIGYYBE
- 0.9e1 * SIGYYBE ^ 2 - (27 * TAUXYBE ^ 2));

     TAUXYBC1 = 0.18e2 * (0.6e1 * SIGXXBE * sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.6e1 * SIGYYBE *
sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.3e1 * sqrt(SIGXXBE ^ 2 - SIGXXBE * SIGYYBE + SIGYYBE ^ 2 + (3 *
TAUXYBE ^ 2))) * CB * cos(PHIM) / (0.3e1 - sin(PHIM)) * TAUXYBE / (0.36e2 * SIGXXBE ^ 2 * sin(PHIM) ^
2 / (0.3e1 - sin(PHIM)) ^ 2 + 0.72e2 * SIGXXBE * SIGYYBE * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 + 0.36e2
* SIGYYBE ^ 2 * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 - 0.9e1 * SIGXXBE ^ 2 + 0.9e1 * SIGXXBE * SIGYYBE
- 0.9e1 * SIGYYBE ^ 2 - (27 * TAUXYBE ^ 2));

     SIGXXBC2 = -0.18e2 * (-0.6e1 * SIGXXBE * sin(PHIM) / (0.3e1 - sin(PHIM)) - 0.6e1 * SIGYYBE *
sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.3e1 * sqrt(SIGXXBE ^ 2 - SIGXXBE * SIGYYBE + SIGYYBE ^ 2 + (3 *
TAUXYBE ^ 2))) * SIGXXBE * CB * cos(PHIM) / (0.3e1 - sin(PHIM)) / (0.36e2 * SIGXXBE ^ 2 * sin(PHIM) ^
2 / (0.3e1 - sin(PHIM)) ^ 2 + 0.72e2 * SIGXXBE * SIGYYBE * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 + 0.36e2
* SIGYYBE ^ 2 * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 - 0.9e1 * SIGXXBE ^ 2 + 0.9e1 * SIGXXBE * SIGYYBE
- 0.9e1 * SIGYYBE ^ 2 - (27 * TAUXYBE ^ 2));

     SIGYYBC2 = -0.18e2 * (-0.6e1 * SIGXXBE * sin(PHIM) / (0.3e1 - sin(PHIM)) - 0.6e1 * SIGYYBE *
sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.3e1 * sqrt(SIGXXBE ^ 2 - SIGXXBE * SIGYYBE + SIGYYBE ^ 2 + (3 *
TAUXYBE ^ 2))) * SIGYYBE * CB * cos(PHIM) / (0.3e1 - sin(PHIM)) / (0.36e2 * SIGXXBE ^ 2 * sin(PHIM) ^
2 / (0.3e1 - sin(PHIM)) ^ 2 + 0.72e2 * SIGXXBE * SIGYYBE * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 + 0.36e2
* SIGYYBE ^ 2 * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 - 0.9e1 * SIGXXBE ^ 2 + 0.9e1 * SIGXXBE * SIGYYBE
- 0.9e1 * SIGYYBE ^ 2 - (27 * TAUXYBE ^ 2));

     TAUXYBC2 = -0.18e2 * (-0.6e1 * SIGXXBE * sin(PHIM) / (0.3e1 - sin(PHIM)) - 0.6e1 * SIGYYBE *
sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.3e1 * sqrt(SIGXXBE ^ 2 - SIGXXBE * SIGYYBE + SIGYYBE ^ 2 + (3 *
TAUXYBE ^ 2))) * CB * cos(PHIM) / (0.3e1 - sin(PHIM)) * TAUXYBE / (0.36e2 * SIGXXBE ^ 2 * sin(PHIM) ^
2 / (0.3e1 - sin(PHIM)) ^ 2 + 0.72e2 * SIGXXBE * SIGYYBE * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 + 0.36e2
* SIGYYBE ^ 2 * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 - 0.9e1 * SIGXXBE ^ 2 + 0.9e1 * SIGXXBE * SIGYYBE
- 0.9e1 * SIGYYBE ^ 2 - (27 * TAUXYBE ^ 2));

  if SIGXXBE/SIGXXBC1 > 0
    SIGXXBC = SIGXXBC1;
  else
    SIGXXBC = SIGXXBC2;
  end
  if SIGYYBE/SIGYYBC1 > 0
    SIGYYBC = SIGYYBC1;
  else
    SIGYYBC = SIGYYBC2;
  end
  if TAUXYBE/TAUXYBC1 > 0
    TAUXYBC = TAUXYBC1;
  else
    TAUXYBC = TAUXYBC2;
  end

  while CU >= 0
    % yield function of brick unit:
    FU = (sqrt(SIGXXUE ^ 2 - SIGXXUE * SIGYYUE + SIGYYUE ^ 2) * (-0.3e1 + sin(PHIU)) + (-0.2e1 *
SIGXXUE - 0.2e1 * SIGYYUE) * sin(PHIU) + 0.6e1 * CU * cos(PHIU)) / (-0.3e1 + sin(PHIU));
    if FU <= 0 % before yielding, plastic strain = 0
      DEPSPXXU = 0;
      DEPSPYYU = 0;
      break;
    else
      % calculate plastic strain increment:
      DEPSPXXU = 0.6e1 * ((((((DEPSXXU0 - DEPSYYU0) * VU - 0.10e2 / 0.3e1 * DEPSXXU0 - 0.7e1 /
0.3e1 * DEPSYYU0) * SIGXXUC ^ 2 - 0.3e1 / 0.2e1 * ((DEPSXXU0 - DEPSYYU0) * VU - 0.20e2 / 0.9e1 *
DEPSXXU0 - 0.11e2 / 0.9e1 * DEPSYYU0) * SIGYYUC * SIGXXUC + ((DEPSXXU0 - DEPSYYU0) * VU -
0.17e2 / 0.3e1 * DEPSXXU0 - 0.14e2 / 0.3e1 * DEPSYYU0) * SIGYYUC ^ 2 / 0.2e1) * sin(PHIU) - 0.3e1 *
(((DEPSXXU0 - DEPSYYU0) * VU - 0.2e1 / 0.3e1 * DEPSXXU0 + DEPSYYU0 / 0.3e1) * SIGXXUC -
((DEPSXXU0 - DEPSYYU0) * VU - DEPSXXU0 / 0.3e1 + 0.2e1 / 0.3e1 * DEPSYYU0) * SIGYYUC) * (-
SIGYYUC / 0.2e1 + SIGXXUC)) * sin(PSIU) - 0.3e1 * (((DEPSXXU0 - DEPSYYU0) * VU - 0.2e1 / 0.3e1 *
DEPSXXU0 + DEPSYYU0 / 0.3e1) * SIGXXUC - ((DEPSXXU0 - DEPSYYU0) * VU - DEPSXXU0 / 0.3e1 +
0.2e1 / 0.3e1 * DEPSYYU0) * SIGYYUC) * (-SIGYYUC / 0.2e1 + SIGXXUC) * (-0.3e1 + sin(PHIU))) *
sqrt(SIGXXUC ^ 2 - SIGXXUC * SIGYYUC + SIGYYUC ^ 2) - 0.2e1 * (((((DEPSXXU0 - DEPSYYU0) * VU -
0.4e1 / 0.3e1 * DEPSXXU0 - DEPSYYU0 / 0.3e1) * SIGXXUC - ((DEPSXXU0 - DEPSYYU0) * VU - 0.2e1 /
0.3e1 * DEPSXXU0 + DEPSYYU0 / 0.3e1) * SIGYYUC) * sin(PHIU) + ((-0.3e1 * DEPSXXU0 + 0.3e1 *
DEPSYYU0) * VU + 0.2e1 * DEPSXXU0 - DEPSYYU0) * SIGXXUC + 0.3e1 * ((DEPSXXU0 - DEPSYYU0) *
VU - DEPSXXU0 / 0.3e1 + 0.2e1 / 0.3e1 * DEPSYYU0) * SIGYYUC) * sin(PSIU) + 0.2e1 * sin(PHIU) *
(DEPSXXU0 + DEPSYYU0) * (-SIGYYUC / 0.2e1 + SIGXXUC)) * (SIGXXUC ^ 2 - SIGXXUC * SIGYYUC +
SIGYYUC ^ 2)) * EU / (((((0.8e1 * SU * VU ^ 2 + ((9 * EU) + 0.4e1 * SU) * VU - (37 * EU) - 0.4e1 * SU) *
SIGXXUC ^ 2 - 0.18e2 * (0.4e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.2e1 / 0.9e1 * SU) * VU - 0.20e2 / 0.9e1 * EU
- 0.2e1 / 0.9e1 * SU) * SIGYYUC * SIGXXUC + 0.9e1 * (0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1 / 0.9e1 *
SU) * VU - 0.37e2 / 0.9e1 * EU - 0.4e1 / 0.9e1 * SU) * SIGYYUC ^ 2) * sin(PHIU) + (-0.24e2 * SU * VU ^ 2 +
(-(27 * EU) - 0.12e2 * SU) * VU + (15 * EU) + 0.12e2 * SU) * SIGXXUC ^ 2 + 0.54e2 * (0.4e1 / 0.9e1 * SU *
VU ^ 2 + (EU + 0.2e1 / 0.9e1 * SU) * VU - 0.4e1 / 0.9e1 * EU - 0.2e1 / 0.9e1 * SU) * SIGYYUC * SIGXXUC -
0.27e2 * (0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1 / 0.9e1 * SU) * VU - 0.5e1 / 0.9e1 * EU - 0.4e1 / 0.9e1 *
SU) * SIGYYUC ^ 2) * sin(PSIU) - 0.27e2 * ((0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1 / 0.9e1 * SU) * VU -
0.5e1 / 0.9e1 * EU - 0.4e1 / 0.9e1 * SU) * SIGXXUC ^ 2 - 0.2e1 * (0.4e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.2e1
/ 0.9e1 * SU) * VU - 0.4e1 / 0.9e1 * EU - 0.2e1 / 0.9e1 * SU) * SIGYYUC * SIGXXUC + (0.8e1 / 0.9e1 * SU *
VU ^ 2 + (EU + 0.4e1 / 0.9e1 * SU) * VU - 0.5e1 / 0.9e1 * EU - 0.4e1 / 0.9e1 * SU) * SIGYYUC ^ 2) * (-0.3e1
```

```
+ sin(PHIU))) * sqrt(SIGXXUC ^ 2 - SIGXXUC * SIGYYUC + SIGYYUC ^ 2) + 0.8e1 * EU * (SIGXXUC ^ 2 -
SIGXXUC * SIGYYUC + SIGYYUC ^ 2) * (SIGXXUC + SIGYYUC) * ((sin(PHIU) - 0.3e1 / 0.2e1) * sin(PSIU)
- 0.3e1 / 0.2e1 * sin(PHIU)));

        DEPSPYYU = -0.3e1 * ((((((DEPSXXU0 - DEPSYYU0) * VU + 0.14e2 / 0.3e1 * DEPSXXU0 + 0.17e2
/ 0.3e1 * DEPSYYU0) * SIGXXUC ^ 2 - 0.3e1 * ((DEPSXXU0 - DEPSYYU0) * VU + 0.11e2 / 0.9e1 *
DEPSXXU0 + 0.20e2 / 0.9e1 * DEPSYYU0) * SIGYYUC * SIGXXUC + 0.2e1 * SIGYYUC ^ 2 * ((DEPSXXU0
- DEPSYYU0) * VU + 0.7e1 / 0.3e1 * DEPSXXU0 + 0.10e2 / 0.3e1 * DEPSYYU0)) * sin(PHIU) - 0.3e1 *
(((DEPSXXU0 - DEPSYYU0) * VU - 0.2e1 / 0.3e1 * DEPSXXU0 + DEPSYYU0 / 0.3e1) * SIGXXUC -
((DEPSXXU0 - DEPSYYU0) * VU - DEPSXXU0 / 0.3e1 + 0.2e1 / 0.3e1 * DEPSYYU0) * SIGYYUC) *
(SIGXXUC - 0.2e1 * SIGYYUC)) * sin(PSIU) - 0.3e1 * (((DEPSXXU0 - DEPSYYU0) * VU - 0.2e1 / 0.3e1 *
DEPSXXU0 + DEPSYYU0 / 0.3e1) * SIGXXUC - ((DEPSXXU0 - DEPSYYU0) * VU - DEPSXXU0 / 0.3e1 +
0.2e1 / 0.3e1 * DEPSYYU0) * SIGYYUC) * (-0.3e1 + sin(PHIU)) * (SIGXXUC - 0.2e1 * SIGYYUC)) *
sqrt(SIGXXUC ^ 2 - SIGXXUC * SIGYYUC + SIGYYUC ^ 2) + 0.4e1 * (SIGXXUC ^ 2 - SIGXXUC * SIGYYUC
+ SIGYYUC ^ 2) * (((((DEPSXXU0 - DEPSYYU0) * VU - DEPSXXU0 / 0.3e1 + 0.2e1 / 0.3e1 * DEPSYYU0) *
SIGXXUC - ((DEPSXXU0 - DEPSYYU0) * VU + DEPSXXU0 / 0.3e1 + 0.4e1 / 0.3e1 * DEPSYYU0) *
SIGYYUC) * sin(PHIU) + ((-0.3e1 * DEPSXXU0 + 0.3e1 * DEPSYYU0) * VU + 0.2e1 * DEPSXXU0 -
DEPSYYU0) * SIGXXUC + 0.3e1 * ((DEPSXXU0 - DEPSYYU0) * VU - DEPSXXU0 / 0.3e1 + 0.2e1 / 0.3e1 *
DEPSYYU0) * SIGYYUC) * sin(PSIU) - sin(PHIU) * (DEPSXXU0 + DEPSYYU0) * (SIGXXUC - 0.2e1 *
SIGYYUC))) * EU / (((((0.8e1 * SU * VU ^ 2 + ((9 * EU) + 0.4e1 * SU) * VU - (37 * EU) - 0.4e1 * SU) * SIGXXUC
^ 2 - 0.18e2 * (0.4e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.2e1 / 0.9e1 * SU) * VU - 0.20e2 / 0.9e1 * EU - 0.2e1 /
0.9e1 * SU) * SIGYYUC * SIGXXUC + 0.9e1 * (0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1 / 0.9e1 * SU) * VU
- 0.37e2 / 0.9e1 * EU - 0.4e1 / 0.9e1 * SU) * SIGYYUC ^ 2) * sin(PHIU) + (-0.24e2 * SU * VU ^ 2 + (-(27 * EU)
- 0.12e2 * SU) * VU + (15 * EU) + 0.12e2 * SU) * SIGXXUC ^ 2 + 0.54e2 * (0.4e1 / 0.9e1 * SU * VU ^ 2 + (EU
+ 0.2e1 / 0.9e1 * SU) * VU - 0.4e1 / 0.9e1 * EU - 0.2e1 / 0.9e1 * SU) * SIGYYUC * SIGXXUC - 0.27e2 * (0.8e1
/ 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1 / 0.9e1 * SU) * VU - 0.5e1 / 0.9e1 * EU - 0.4e1 / 0.9e1 * SU) * SIGYYUC
^ 2) * sin(PSIU) - 0.27e2 * ((0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1 / 0.9e1 * SU) * VU - 0.5e1 / 0.9e1 *
EU - 0.4e1 / 0.9e1 * SU) * SIGXXUC ^ 2 - 0.2e1 * (0.4e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.2e1 / 0.9e1 * SU) *
VU - 0.4e1 / 0.9e1 * EU - 0.2e1 / 0.9e1 * SU) * SIGYYUC * SIGXXUC + (0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU
+ 0.4e1 / 0.9e1 * SU) * VU - 0.5e1 / 0.9e1 * EU - 0.4e1 / 0.9e1 * SU) * SIGYYUC ^ 2) * (-0.3e1 + sin(PHIU)))
* sqrt(SIGXXUC ^ 2 - SIGXXUC * SIGYYUC + SIGYYUC ^ 2) + 0.8e1 * EU * (SIGXXUC ^ 2 - SIGXXUC *
SIGYYUC + SIGYYUC ^ 2) * (SIGXXUC + SIGYYUC) * ((sin(PHIU) - 0.3e1 / 0.2e1) * sin(PSIU) - 0.3e1 /
0.2e1 * sin(PHIU)));

        % recalculate softening modulus SUC:
        DLU = 0.6e1 * EU * ((-0.3e1 + sin(PHIU)) * (((DEPSXXU0 - DEPSYYU0) * VU - 0.2e1 / 0.3e1 *
DEPSXXU0 + DEPSYYU0 / 0.3e1) * SIGXXUC - ((DEPSXXU0 - DEPSYYU0) * VU - DEPSXXU0 / 0.3e1 +
0.2e1 / 0.3e1 * DEPSYYU0) * SIGYYUC) * sqrt(SIGXXUC ^ 2 - SIGXXUC * SIGYYUC + SIGYYUC ^ 2) +
0.4e1 / 0.3e1 * sin(PHIU) * (SIGXXUC ^ 2 - SIGXXUC * SIGYYUC + SIGYYUC ^ 2) * (DEPSXXU0 +
DEPSYYU0)) * (-0.3e1 + sin(PSIU)) / (0.8e1 * EU * (SIGXXUC + SIGYYUC) * ((sin(PSIU) - 0.3e1 / 0.2e1) *
sin(PHIU) - 0.3e1 / 0.2e1 * sin(PSIU)) * sqrt(SIGXXUC ^ 2 - SIGXXUC * SIGYYUC + SIGYYUC ^ 2) + (((0.8e1
* SU * VU ^ 2 + (0.9e1 * EU + 0.4e1 * SU) * VU - 0.37e2 * EU - 0.4e1 * SU) * SIGXXUC ^ 2 - 0.18e2 * (0.4e1
/ 0.9e1 * SU * VU ^ 2 + (EU + 0.2e1 / 0.9e1 * SU) * VU - 0.20e2 / 0.9e1 * EU - 0.2e1 / 0.9e1 * SU) * SIGYYUC
* SIGXXUC + 0.9e1 * (0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1 / 0.9e1 * SU) * VU - 0.37e2 / 0.9e1 * EU -
0.4e1 / 0.9e1 * SU) * SIGYYUC ^ 2) * sin(PSIU) + (-0.24e2 * SU * VU ^ 2 + (-0.27e2 * EU - 0.12e2 * SU) *
VU + 0.15e2 * EU + 0.12e2 * SU) * SIGXXUC ^ 2 + 0.54e2 * (0.4e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.2e1 /
0.9e1 * SU) * VU - 0.4e1 / 0.9e1 * EU - 0.2e1 / 0.9e1 * SU) * SIGYYUC * SIGXXUC - 0.27e2 * (0.8e1 / 0.9e1
* SU * VU ^ 2 + (EU + 0.4e1 / 0.9e1 * SU) * VU - 0.5e1 / 0.9e1 * EU - 0.4e1 / 0.9e1 * SU) * SIGYYUC ^ 2) *
sin(PHIU) - 0.27e2 * ((0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1 / 0.9e1 * SU) * VU - 0.5e1 / 0.9e1 * EU -
0.4e1 / 0.9e1 * SU) * SIGXXUC ^ 2 - 0.2e1 * (0.4e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.2e1 / 0.9e1 * SU) * VU -
0.4e1 / 0.9e1 * EU - 0.2e1 / 0.9e1 * SU) * SIGYYUC * SIGXXUC + (0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1
/ 0.9e1 * SU) * VU - 0.5e1 / 0.9e1 * EU - 0.4e1 / 0.9e1 * SU) * SIGYYUC ^ 2) * (-0.3e1 + sin(PSIU)));

        FKU = -0.3e1 / (0.3e1 - sin(PHIU)) * (0.1e1 - sin(PHIU)) * KCU;
        SUC = -FKU*DKU/DLU;
        if abs(SUC-SU) < TOR2
```

```
                break;
            else
                SU = SUC;
            end
        end
    end

    while CH >= 0
        % yield function of brick unit:
        FH = (sqrt(SIGXXHE ^ 2 - SIGXXHE * SIGYYHE + SIGYYHE ^ 2) * (-0.3e1 + sin(PHIM)) + (-0.2e1 *
SIGXXHE - 0.2e1 * SIGYYHE) * sin(PHIM) + 0.6e1 * CH * cos(PHIM)) / (-0.3e1 + sin(PHIM));
        if FH <= 0 % before yielding, plastic strain = 0
            DEPSPXXH = 0;
            DEPSPYYH = 0;
            break;
        else
            % calculate plastic strain increment:
            DEPSPXXH = ((((((((20 * DEPSYYH0 - 6 * DEPSXXH0) * E - 14 * EU * DEPSYYH0) * V + 20 * E *
DEPSXXH0 + 14 * EU * DEPSYYH0) * SIGXXHC ^ 2) - 0.20e2 * (((DEPSYYH0 - 0.9e1 / 0.20e2 * DEPSXXH0)
* E - 0.11e2 / 0.20e2 * EU * DEPSYYH0) * V + (E * DEPSXXH0) + 0.11e2 / 0.20e2 * EU * DEPSYYH0) *
SIGYYHC * SIGXXHC + 0.17e2 * (((DEPSYYH0 - 0.3e1 / 0.17e2 * DEPSXXH0) * E - 0.14e2 / 0.17e2 * EU *
DEPSYYH0) * V + (E * DEPSXXH0) + 0.14e2 / 0.17e2 * EU * DEPSYYH0) * SIGYYHC ^ 2) * sin(PHIM) -
0.12e2 * (SIGXXHC - SIGYYHC / 0.2e1) * ((((DEPSYYH0 - 0.3e1 / 0.2e1 * DEPSXXH0) * E + (EU *
DEPSYYH0) / 0.2e1) * V + (E * DEPSXXH0) - (EU * DEPSYYH0) / 0.2e1) * SIGXXHC - SIGYYHC *
(((DEPSYYH0 - 3 * DEPSXXH0) * E + 2 * EU * DEPSYYH0) * V + E * DEPSXXH0 - 2 * EU * DEPSYYH0) /
0.2e1)) * sin(PSIM) - 0.12e2 * (SIGXXHC - SIGYYHC / 0.2e1) * (sin(PHIM) - 0.3e1) * ((((DEPSYYH0 - 0.3e1
/ 0.2e1 * DEPSXXH0) * E + (EU * DEPSYYH0) / 0.2e1) * V + (E * DEPSXXH0) - (EU * DEPSYYH0) / 0.2e1)
* SIGXXHC - SIGYYHC * (((DEPSYYH0 - 3 * DEPSXXH0) * E + 2 * EU * DEPSYYH0) * V + E * DEPSXXH0
- 2 * EU * DEPSYYH0) / 0.2e1)) * sqrt((SIGXXHC ^ 2) - SIGXXHC * SIGYYHC + SIGYYHC ^ 2) - 0.16e2 *
((SIGXXHC ^ 2) - SIGXXHC * SIGYYHC + SIGYYHC ^ 2) * ((((((DEPSYYH0 - 0.3e1 / 0.4e1 * DEPSXXH0) *
E - (EU * DEPSYYH0) / 0.4e1) * V + (E * DEPSXXH0) + (EU * DEPSYYH0) / 0.4e1) * SIGXXHC -
(((DEPSYYH0 - 0.3e1 / 0.2e1 * DEPSXXH0) * E + (EU * DEPSYYH0) / 0.2e1) * V + (E * DEPSXXH0) - (EU
* DEPSYYH0) / 0.2e1) * SIGYYHC / 0.2e1) * sin(PHIM) + (((-0.3e1 / 0.2e1 * DEPSYYH0 + 0.9e1 / 0.4e1 *
DEPSXXH0) * E - 0.3e1 / 0.4e1 * EU * DEPSYYH0) * V - 0.3e1 / 0.2e1 * E * DEPSXXH0 + 0.3e1 / 0.4e1 *
EU * DEPSYYH0) * SIGXXHC + 0.3e1 / 0.4e1 * SIGYYHC * (((DEPSYYH0 - 3 * DEPSXXH0) * E + 2 * EU *
DEPSYYH0) * V + E * DEPSXXH0 - 2 * EU * DEPSYYH0)) * sin(PSIM) - 0.3e1 / 0.2e1 * (DEPSYYH0 * (E -
EU) * V + E * DEPSXXH0 + EU * DEPSYYH0) * (SIGXXHC - SIGYYHC / 0.2e1) * sin(PHIM))) / ((((((-8 * V ^
2 * SH + (8 * E - 17 * EU - 4 * SH) * V + 20 * E + 17 * EU + 4 * SH) * SIGXXHC ^ 2) - 0.2e1 * SIGYYHC * (-4
* V ^ 2 * SH + (E - 10 * EU - 2 * SH) * V + 10 * E + 10 * EU + 2 * SH) * SIGXXHC + 0.11e2 * SIGYYHC ^ 2 *
(-0.8e1 / 0.11e2 * (V ^ 2) * SH + (E - 0.20e2 / 0.11e2 * EU - 0.4e1 / 0.11e2 * SH) * V + 0.17e2 / 0.11e2 * E +
0.20e2 / 0.11e2 * EU + 0.4e1 / 0.11e2 * SH)) * sin(PHIM) + ((24 * V ^ 2 * SH + (24 * E + 3 * EU + 12 * SH) *
V - 12 * E - 3 * EU - 12 * SH) * SIGXXHC ^ 2) - 0.42e2 * (0.4e1 / 0.7e1 * (V ^ 2) * SH + (E + 0.2e1 / 0.7e1 *
EU + 0.2e1 / 0.7e1 * SH) * V - 0.2e1 / 0.7e1 * E - 0.2e1 / 0.7e1 * EU - 0.2e1 / 0.7e1 * SH) * SIGYYHC *
SIGXXHC + 0.15e2 * (0.8e1 / 0.5e1 * (V ^ 2) * SH + (E + 0.4e1 / 0.5e1 * EU + 0.4e1 / 0.5e1 * SH) * V - E /
0.5e1 - 0.4e1 / 0.5e1 * EU - 0.4e1 / 0.5e1 * SH) * SIGYYHC ^ 2) * sin(PSIM) + 0.24e2 * (sin(PHIM) - 0.3e1)
* (((V ^ 2 * SH) + (E + EU / 0.8e1 + SH / 0.2e1) * V - E / 0.2e1 - EU / 0.8e1 - SH / 0.2e1) * (SIGXXHC ^ 2) -
0.7e1 / 0.4e1 * (0.4e1 / 0.7e1 * (V ^ 2) * SH + (E + 0.2e1 / 0.7e1 * EU + 0.2e1 / 0.7e1 * SH) * V - 0.2e1 / 0.7e1
* E - 0.2e1 / 0.7e1 * EU - 0.2e1 / 0.7e1 * SH) * SIGYYHC * SIGXXHC + 0.5e1 / 0.8e1 * (0.8e1 / 0.5e1 * (V ^
2) * SH + (E + 0.4e1 / 0.5e1 * EU + 0.4e1 / 0.5e1 * SH) * V - E / 0.5e1 - 0.4e1 / 0.5e1 * EU - 0.4e1 / 0.5e1 *
SH) * SIGYYHC ^ 2)) * sqrt((SIGXXHC ^ 2) - SIGXXHC * SIGYYHC + SIGYYHC ^ 2) + 0.8e1 * ((SIGXXHC
^ 2) - SIGXXHC * SIGYYHC + SIGYYHC ^ 2) * ((((E - EU) * V - 2 * E + EU) * SIGXXHC) - 0.2e1 * (((E - EU)
* V) - E / 0.2e1 + EU) * SIGYYHC) * ((sin(PHIM) - 0.3e1 / 0.2e1) * sin(PSIM) - 0.3e1 / 0.2e1 * sin(PHIM)));

            DEPSPYYH = ((((((((14 * DEPSYYH0 + 3 * DEPSXXH0) * E - 17 * EU * DEPSYYH0) * V + 14 * E *
DEPSXXH0 + 17 * EU * DEPSYYH0) * SIGXXHC ^ 2) - 0.11e2 * (((DEPSYYH0 + 0.9e1 / 0.11e2 * DEPSXXH0)
* E - 0.20e2 / 0.11e2 * EU * DEPSYYH0) * V + (E * DEPSXXH0) + 0.20e2 / 0.11e2 * EU * DEPSYYH0) *
SIGYYHC * SIGXXHC + 0.14e2 * (((DEPSYYH0 + 0.3e1 / 0.7e1 * DEPSXXH0) * E - 0.10e2 / 0.7e1 * EU *
DEPSYYH0) * V + (E * DEPSXXH0) + 0.10e2 / 0.7e1 * EU * DEPSYYH0) * SIGYYHC ^ 2) * sin(PHIM) +
0.6e1 * ((((DEPSYYH0 - 0.3e1 / 0.2e1 * DEPSXXH0) * E + (EU * DEPSYYH0) / 0.2e1) * V + (E * DEPSXXH0)
- (EU * DEPSYYH0) / 0.2e1) * SIGXXHC - SIGYYHC * (((DEPSYYH0 - 3 * DEPSXXH0) * E + 2 * EU *
DEPSYYH0) * V + E * DEPSXXH0 - 2 * EU * DEPSYYH0) / 0.2e1) * (SIGXXHC - 0.2e1 * SIGYYHC)) *
sin(PSIM) + 0.6e1 * (sin(PHIM) - 0.3e1) * ((((DEPSYYH0 - 0.3e1 / 0.2e1 * DEPSXXH0) * E + (EU *
DEPSYYH0) / 0.2e1) * V + (E * DEPSXXH0) - (EU * DEPSYYH0) / 0.2e1) * SIGXXHC - SIGYYHC *
(((DEPSYYH0 - 3 * DEPSXXH0) * E + 2 * EU * DEPSYYH0) * V + E * DEPSXXH0 - 2 * EU * DEPSYYH0) /
0.2e1) * (SIGXXHC - 0.2e1 * SIGYYHC)) * sqrt((SIGXXHC ^ 2) - SIGXXHC * SIGYYHC + SIGYYHC ^ 2) -
0.4e1 * ((SIGXXHC ^ 2) - SIGXXHC * SIGYYHC + SIGYYHC ^ 2) * (((((((DEPSYYH0 - 3 * DEPSXXH0) * E
+ 2 * EU * DEPSYYH0) * V + E * DEPSXXH0 - 2 * EU * DEPSYYH0) * SIGXXHC) + (((DEPSYYH0 + 3 *
DEPSXXH0) * E - 4 * EU * DEPSYYH0) * V + E * DEPSXXH0 + 4 * EU * DEPSYYH0) * SIGYYHC) * sin(PHIM)
+ ((((-6 * DEPSYYH0 + 9 * DEPSXXH0) * E - 3 * EU * DEPSYYH0) * V - 6 * E * DEPSXXH0 + 3 * EU *
DEPSYYH0) * SIGXXHC) + 0.3e1 * SIGYYHC * (((DEPSYYH0 - 3 * DEPSXXH0) * E + 2 * EU * DEPSYYH0)
* V + E * DEPSXXH0 - 2 * EU * DEPSYYH0)) * sin(PSIM) + 0.3e1 * (DEPSYYH0 * (E - EU) * V + E *
DEPSXXH0 + EU * DEPSYYH0) * (SIGXXHC - 0.2e1 * SIGYYHC) * sin(PHIM))) / ((((((-8 * V ^ 2 * SH + (8 *
E - 17 * EU - 4 * SH) * V + 20 * E + 17 * EU + 4 * SH) * SIGXXHC ^ 2) - 0.2e1 * SIGYYHC * (-4 * V ^ 2 * SH
+ (E - 10 * EU - 2 * SH) * V + 10 * E + 10 * EU + 2 * SH) * SIGXXHC + 0.11e2 * SIGYYHC ^ 2 * (-0.8e1 /
0.11e2 * (V ^ 2) * SH + (E - 0.20e2 / 0.11e2 * EU - 0.4e1 / 0.11e2 * SH) * V + 0.17e2 / 0.11e2 * E + 0.20e2 /
0.11e2 * EU + 0.4e1 / 0.11e2 * SH)) * sin(PHIM) + ((24 * V ^ 2 * SH + (24 * E + 3 * EU + 12 * SH) * V - 12 *
E - 3 * EU - 12 * SH) * SIGXXHC ^ 2) - 0.42e2 * (0.4e1 / 0.7e1 * (V ^ 2) * SH + (E + 0.2e1 / 0.7e1 * EU +
0.2e1 / 0.7e1 * SH) * V - 0.2e1 / 0.7e1 * E - 0.2e1 / 0.7e1 * EU - 0.2e1 / 0.7e1 * SH) * SIGYYHC * SIGXXHC
+ 0.15e2 * (0.8e1 / 0.5e1 * (V ^ 2) * SH + (E + 0.4e1 / 0.5e1 * EU + 0.4e1 / 0.5e1 * SH) * V - E / 0.5e1 - 0.4e1
/ 0.5e1 * EU - 0.4e1 / 0.5e1 * SH) * SIGYYHC ^ 2) * sin(PSIM) + 0.24e2 * (sin(PHIM) - 0.3e1) * (((V ^ 2 * SH)
+ (E + EU / 0.8e1 + SH / 0.2e1) * V - E / 0.2e1 - EU / 0.8e1 - SH / 0.2e1) * (SIGXXHC ^ 2) - 0.7e1 / 0.4e1 *
(0.4e1 / 0.7e1 * (V ^ 2) * SH + (E + 0.2e1 / 0.7e1 * EU + 0.2e1 / 0.7e1 * SH) * V - 0.2e1 / 0.7e1 * E - 0.2e1 /
0.7e1 * EU - 0.2e1 / 0.7e1 * SH) * SIGYYHC * SIGXXHC + 0.5e1 / 0.8e1 * (0.8e1 / 0.5e1 * (V ^ 2) * SH + (E
+ 0.4e1 / 0.5e1 * EU + 0.4e1 / 0.5e1 * SH) * V - E / 0.5e1 - 0.4e1 / 0.5e1 * EU - 0.4e1 / 0.5e1 * SH) * SIGYYHC
^ 2)) * sqrt((SIGXXHC ^ 2) - SIGXXHC * SIGYYHC + SIGYYHC ^ 2) + 0.8e1 * ((SIGXXHC ^ 2) - SIGXXHC *
SIGYYHC + SIGYYHC ^ 2) * ((((E - EU) * V - 2 * E + EU) * SIGXXHC) - 0.2e1 * (((E - EU) * V) - E / 0.2e1 +
EU) * SIGYYHC) * ((sin(PHIM) - 0.3e1 / 0.2e1) * sin(PSIM) - 0.3e1 / 0.2e1 * sin(PHIM)));

        % recalculate softening modulus SUC:
        DLH = 0.4e1 * (((((DEPSYYH0 - 0.3e1 / 0.2e1 * DEPSXXH0) * E + EU * DEPSYYH0 / 0.2e1) * V + E
* DEPSXXH0 - EU * DEPSYYH0 / 0.2e1) * SIGXXHC - SIGYYHC * (((DEPSYYH0 - 0.3e1 * DEPSXXH0) * E
+ 0.2e1 * EU * DEPSYYH0) * V + E * DEPSXXH0 - 0.2e1 * EU * DEPSYYH0) / 0.2e1) * (sin(PHIM) - 0.3e1)
* sqrt(SIGXXHC ^ 2 - SIGXXHC * SIGYYHC + SIGYYHC ^ 2) - 0.2e1 * sin(PHIM) * (DEPSYYH0 * (E - EU) *
V + E * DEPSXXH0 + EU * DEPSYYH0) * (SIGXXHC ^ 2 - SIGXXHC * SIGYYHC + SIGYYHC ^ 2)) * (-0.3e1
+ sin(PSIM)) / (0.8e1 * ((sin(PSIM) - 0.3e1 / 0.2e1) * sin(PHIM) - 0.3e1 / 0.2e1 * sin(PSIM)) * (((E - EU) * V -
0.2e1 * E + EU) * SIGXXHC - 0.2e1 * ((E - EU) * V - E / 0.2e1 + EU) * SIGYYHC) * sqrt(SIGXXHC ^ 2 -
SIGXXHC * SIGYYHC + SIGYYHC ^ 2) + (((-0.8e1 * V ^ 2 * SH + (0.8e1 * E - 0.17e2 * EU - 0.4e1 * SH) * V
+ 0.20e2 * E + 0.17e2 * EU + 0.4e1 * SH) * SIGXXHC ^ 2 - 0.2e1 * SIGYYHC * (-0.4e1 * V ^ 2 * SH + (E -
0.10e2 * EU - 0.2e1 * SH) * V + 0.10e2 * E + 0.10e2 * EU + 0.2e1 * SH) * SIGXXHC + 0.11e2 * SIGYYHC ^
2 * (-0.8e1 / 0.11e2 * V ^ 2 * SH + (E - 0.20e2 / 0.11e2 * EU - 0.4e1 / 0.11e2 * SH) * V + 0.17e2 / 0.11e2 * E
+ 0.20e2 / 0.11e2 * EU + 0.4e1 / 0.11e2 * SH)) * sin(PSIM) + (0.24e2 * V ^ 2 * SH + (0.24e2 * E + 0.3e1 * EU
+ 0.12e2 * SH) * V - 0.12e2 * E - 0.3e1 * EU - 0.12e2 * SH) * SIGXXHC ^ 2 - 0.42e2 * (0.4e1 / 0.7e1 * V ^ 2
* SH + (E + 0.2e1 / 0.7e1 * EU + 0.2e1 / 0.7e1 * SH) * V - 0.2e1 / 0.7e1 * E - 0.2e1 / 0.7e1 * EU - 0.2e1 /
0.7e1 * SH) * SIGYYHC * SIGXXHC + 0.15e2 * (0.8e1 / 0.5e1 * V ^ 2 * SH + (E + 0.4e1 / 0.5e1 * EU + 0.4e1
/ 0.5e1 * SH) * V - E / 0.5e1 - 0.4e1 / 0.5e1 * EU - 0.4e1 / 0.5e1 * SH) * SIGYYHC ^ 2) * sin(PHIM) + 0.24e2
* (-0.3e1 + sin(PSIM)) * ((V ^ 2 * SH + (E + EU / 0.8e1 + SH / 0.2e1) * V - E / 0.2e1 - EU / 0.8e1 - SH / 0.2e1)
* SIGXXHC ^ 2 - 0.7e1 / 0.4e1 * (0.4e1 / 0.7e1 * V ^ 2 * SH + (E + 0.2e1 / 0.7e1 * EU + 0.2e1 / 0.7e1 * SH) *
V - 0.2e1 / 0.7e1 * E - 0.2e1 / 0.7e1 * EU - 0.2e1 / 0.7e1 * SH) * SIGYYHC * SIGXXHC + 0.5e1 / 0.8e1 *
(0.8e1 / 0.5e1 * V ^ 2 * SH + (E + 0.4e1 / 0.5e1 * EU + 0.4e1 / 0.5e1 * SH) * V - E / 0.5e1 - 0.4e1 / 0.5e1 *
EU - 0.4e1 / 0.5e1 * SH) * SIGYYHC ^ 2));

        FKH = -0.3e1 / (0.3e1 - sin(PHIM)) * (0.1e1 - sin(PHIM)) * KCH;
        SHC = -FKH* DKH / DLH ;
```

```
            if abs(SHC-SH) < TOR2
                break;
            else
                SH = SHC;
            end
        end
    end

    while CB >= 0
        % yield function of brick unit:
        FB = (sqrt((SIGXXBE ^ 2 - SIGXXBE * SIGYYBE + SIGYYBE ^ 2 + 3 * TAUXYBE^2)) * (-0.3e1 +
sin(PHIM)) + (-2 * SIGXXBE - 2 * SIGYYBE) * sin(PHIM) + 0.6e1 * CB * cos(PHIM)) / (-0.3e1 + sin(PHIM));
        if FB <= 0
            DEPSPXXB = 0;
            DEPSPYYB = 0;
            DEPSPXYB = 0;
            break;
        else
            % calculate plastic strain increment:
            DEPSPXXB = ((((((((18 * DEPSXXB0 - 28 * DEPSYYB0) * E + 10 * EU * DEPSYYB0) * V - 28 * E *
DEPSXXB0 - 10 * EU * DEPSYYB0) * SIGXXBC ^ 2) + (((((-27 * DEPSXXB0 + 28 * DEPSYYB0) * E - EU *
DEPSYYB0) * V + 28 * E * DEPSXXB0 + EU * DEPSYYB0) * SIGYYBC) + 0.72e2 * DEPSXYB0 * TAUXYBC
* (V - 0.1e1 / 0.2e1) * E) * SIGXXBC + ((((9 * DEPSXXB0 - 19 * DEPSYYB0) * E + 10 * EU * DEPSYYB0) *
V - 19 * E * DEPSXXB0 - 10 * EU * DEPSYYB0) * SIGYYBC ^ 2) - 0.36e2 * DEPSXYB0 * TAUXYBC * (V -
0.1e1 / 0.2e1) * E * SIGYYBC - 0.48e2 * TAUXYBC ^ 2 * (DEPSYYB0 * (E - EU) * V + E * DEPSXXB0 + EU
* DEPSYYB0)) * sin(PHIM) - 0.54e2 * (SIGXXBC - SIGYYBC / 0.2e1) * ((((DEPSXXB0 - 0.2e1 / 0.3e1 *
DEPSYYB0) * E - (EU * DEPSYYB0) / 0.3e1) * V - 0.2e1 / 0.3e1 * E * DEPSXXB0 + (EU * DEPSYYB0) /
0.3e1) * SIGXXBC + (((-DEPSXXB0 + DEPSYYB0 / 0.3e1) * E + 0.2e1 / 0.3e1 * EU * DEPSYYB0) * V + (E
* DEPSXXB0) / 0.3e1 - 0.2e1 / 0.3e1 * EU * DEPSYYB0) * SIGYYBC + 0.4e1 * DEPSXYB0 * TAUXYBC * (V
- 0.1e1 / 0.2e1) * E)) * sin(PSIM) - 0.54e2 * (SIGXXBC - SIGYYBC / 0.2e1) * (sin(PHIM) - 0.3e1) *
((((DEPSXXB0 - 0.2e1 / 0.3e1 * DEPSYYB0) * E - (EU * DEPSYYB0) / 0.3e1) * V - 0.2e1 / 0.3e1 * E *
DEPSXXB0 + (EU * DEPSYYB0) / 0.3e1) * SIGXXBC + (((-DEPSXXB0 + DEPSYYB0 / 0.3e1) * E + 0.2e1 /
0.3e1 * EU * DEPSYYB0) * V + (E * DEPSXXB0) / 0.3e1 - 0.2e1 / 0.3e1 * EU * DEPSYYB0) * SIGYYBC +
0.4e1 * DEPSXYB0 * TAUXYBC * (V - 0.1e1 / 0.2e1) * E)) * sqrt((SIGXXBC ^ 2) - (SIGXXBC * SIGYYBC) +
(SIGYYBC ^ 2) + 0.3e1 * TAUXYBC ^ 2) - 0.12e2 * ((((((DEPSXXB0 - 0.8e1 / 0.3e1 * DEPSYYB0) * E + 0.5e1
/ 0.3e1 * EU * DEPSYYB0) * V - 0.5e1 / 0.3e1 * EU * DEPSYYB0 - 0.8e1 / 0.3e1 * E * DEPSXXB0) * SIGXXBC
+ (((-DEPSXXB0 + 0.4e1 / 0.3e1 * DEPSYYB0) * E - (EU * DEPSYYB0) / 0.3e1) * V + 0.4e1 / 0.3e1 * E *
DEPSXXB0 + (EU * DEPSYYB0) / 0.3e1) * SIGYYBC + 0.4e1 * DEPSXYB0 * TAUXYBC * (V - 0.1e1 / 0.2e1)
* E) * sin(PHIM) + ((((-3 * DEPSXXB0 + 2 * DEPSYYB0) * E + EU * DEPSYYB0) * V + 2 * E * DEPSXXB0 -
EU * DEPSYYB0) * SIGXXBC) + ((((3 * DEPSXXB0 - DEPSYYB0) * E - 2 * EU * DEPSYYB0) * V - E *
DEPSXXB0 + 2 * EU * DEPSYYB0) * SIGYYBC) - 0.12e2 * DEPSXYB0 * TAUXYBC * (V - 0.1e1 / 0.2e1) *
E) * sin(PSIM) + 0.6e1 * (SIGXXBC - SIGYYBC / 0.2e1) * (DEPSYYB0 * (E - EU) * V + E * DEPSXXB0 + EU
* DEPSYYB0) * sin(PHIM)) * ((SIGXXBC ^ 2) - (SIGXXBC * SIGYYBC) + (SIGYYBC ^ 2) + 0.3e1 * TAUXYBC
^ 2)) / ((((((24 * V ^ 2 * SB + (8 * E + 19 * EU + 12 * SB) * V - 28 * E - 19 * EU - 12 * SB) * SIGXXBC ^ 2) -
0.26e2 * (0.12e2 / 0.13e2 * (V ^ 2) * SB + (E + 0.14e2 / 0.13e2 * EU + 0.6e1 / 0.13e2 * SB) * V - 0.14e2 /
0.13e2 * E - 0.14e2 / 0.13e2 * EU - 0.6e1 / 0.13e2 * SB) * SIGYYBC * SIGXXBC + ((24 * V ^ 2 * SB + (-E +
28 * EU + 12 * SB) * V - 19 * E - 28 * EU - 12 * SB) * SIGYYBC ^ 2) + 0.168e3 * (0.3e1 / 0.7e1 * (V ^ 2) * SB
+ (E + 0.2e1 / 0.7e1 * EU + 0.3e1 / 0.14e2 * SB) * V - 0.13e2 / 0.14e2 * E - 0.2e1 / 0.7e1 * EU - 0.3e1 / 0.14e2
* SB) * TAUXYBC ^ 2) * sin(PHIM) + ((-72 * V ^ 2 * SB + (-72 * E - 9 * EU - 36 * SB) * V + 36 * E + 9 * EU +
36 * SB) * SIGXXBC ^ 2) + 0.126e3 * (0.4e1 / 0.7e1 * (V ^ 2) * SB + (E + 0.2e1 / 0.7e1 * EU + 0.2e1 / 0.7e1
* SB) * V - 0.2e1 / 0.7e1 * E - 0.2e1 / 0.7e1 * EU - 0.2e1 / 0.7e1 * SB) * SIGYYBC * SIGXXBC + ((-72 * V ^ 2
* SB + (-45 * E - 36 * EU - 36 * SB) * V + 9 * E + 36 * EU + 36 * SB) * SIGYYBC ^ 2) - 0.648e3 * TAUXYBC
^ 2 * (V - 0.1e1 / 0.2e1) * ((V * SB) / 0.3e1 + E + SB / 0.3e1)) * sin(PSIM) - 0.72e2 * (((V ^ 2 * SB) + (E + EU
/ 0.8e1 + SB / 0.2e1) * V - E / 0.2e1 - EU / 0.8e1 - SB / 0.2e1) * (SIGXXBC ^ 2) - 0.7e1 / 0.4e1 * (0.4e1 /
0.7e1 * (V ^ 2) * SB + (E + 0.2e1 / 0.7e1 * EU + 0.2e1 / 0.7e1 * SB) * V - 0.2e1 / 0.7e1 * E - 0.2e1 / 0.7e1 *
EU - 0.2e1 / 0.7e1 * SB) * SIGYYBC * SIGXXBC + ((V ^ 2 * SB) + (0.5e1 / 0.8e1 * E + EU / 0.2e1 + SB /
0.2e1) * V - E / 0.8e1 - EU / 0.2e1 - SB / 0.2e1) * (SIGYYBC ^ 2) + 0.9e1 * TAUXYBC ^ 2 * (V - 0.1e1 / 0.2e1)
* ((V * SB) / 0.3e1 + E + SB / 0.3e1)) * (sin(PHIM) - 0.3e1)) * sqrt((SIGXXBC ^ 2) - (SIGXXBC * SIGYYBC) +
(SIGYYBC ^ 2) + 0.3e1 * TAUXYBC ^ 2) - 0.16e2 * ((SIGXXBC ^ 2) - (SIGXXBC * SIGYYBC) + (SIGYYBC ^
2) + 0.3e1 * TAUXYBC ^ 2) * ((sin(PHIM) - 0.3e1 / 0.4e1) * sin(PSIM) - 0.9e1 / 0.4e1 * sin(PHIM)) * ((((E -
EU) * V - 2 * E + EU) * SIGXXBC) - 0.2e1 * (((E - EU) * V) - E / 0.2e1 + EU) * SIGYYBC));

        DEPSPYYB = ((((((((-9 * DEPSXXB0 - 10 * DEPSYYB0) * E + 19 * EU * DEPSYYB0) * V - 10 * E *
DEPSXXB0 - 19 * EU * DEPSYYB0) * SIGXXBC ^ 2) + (((((27 * DEPSXXB0 + DEPSYYB0) * E - 28 * EU *
DEPSYYB0) * V + E * DEPSXXB0 + 28 * EU * DEPSYYB0) * SIGYYBC) - 0.36e2 * DEPSXYB0 * TAUXYBC
* (V - 0.1e1 / 0.2e1) * E) * SIGXXBC + ((((-18 * DEPSXXB0 - 10 * DEPSYYB0) * E + 28 * EU * DEPSYYB0)
* V - 10 * E * DEPSXXB0 - 28 * EU * DEPSYYB0) * SIGYYBC ^ 2) + 0.72e2 * DEPSXYB0 * TAUXYBC * (V
- 0.1e1 / 0.2e1) * E * SIGYYBC - 0.48e2 * TAUXYBC ^ 2 * (DEPSYYB0 * (E - EU) * V + E * DEPSXXB0 + EU
* DEPSYYB0)) * sin(PHIM) + 0.27e2 * (SIGXXBC - 2 * SIGYYBC) * ((((DEPSXXB0 - 0.2e1 / 0.3e1 *
DEPSYYB0) * E - (EU * DEPSYYB0) / 0.3e1) * V - 0.2e1 / 0.3e1 * E * DEPSXXB0 + (EU * DEPSYYB0) /
0.3e1) * SIGXXBC + (((-DEPSXXB0 + DEPSYYB0 / 0.3e1) * E + 0.2e1 / 0.3e1 * EU * DEPSYYB0) * V + (E
* DEPSXXB0) / 0.3e1 - 0.2e1 / 0.3e1 * EU * DEPSYYB0) * SIGYYBC + 0.4e1 * DEPSXYB0 * TAUXYBC * (V
- 0.1e1 / 0.2e1) * E)) * sin(PSIM) + 0.27e2 * (SIGXXBC - 2 * SIGYYBC) * (sin(PHIM) - 0.3e1) * ((((DEPSXXB0
- 0.2e1 / 0.3e1 * DEPSYYB0) * E - (EU * DEPSYYB0) / 0.3e1) * V - 0.2e1 / 0.3e1 * E * DEPSXXB0 + (EU *
DEPSYYB0) / 0.3e1) * SIGXXBC + (((-DEPSXXB0 + DEPSYYB0 / 0.3e1) * E + 0.2e1 / 0.3e1 * EU *
DEPSYYB0) * V + (E * DEPSXXB0) / 0.3e1 - 0.2e1 / 0.3e1 * EU * DEPSYYB0) * SIGYYBC + 0.4e1 *
DEPSXYB0 * TAUXYBC * (V - 0.1e1 / 0.2e1) * E)) * sqrt((SIGXXBC ^ 2) - (SIGXXBC * SIGYYBC) +
(SIGYYBC ^ 2) + 0.3e1 * TAUXYBC ^ 2) - 0.12e2 * ((SIGXXBC ^ 2) - (SIGXXBC * SIGYYBC) + (SIGYYBC ^
2) + 0.3e1 * TAUXYBC ^ 2) * ((((((DEPSXXB0 + DEPSYYB0 / 0.3e1) * E - 0.4e1 / 0.3e1 * EU * DEPSYYB0)
* V + (E * DEPSXXB0) / 0.3e1 + 0.4e1 / 0.3e1 * EU * DEPSYYB0) * SIGXXBC + (((-DEPSXXB0 - 0.5e1 /
0.3e1 * DEPSYYB0) * E + 0.8e1 / 0.3e1 * EU * DEPSYYB0) * V - 0.5e1 / 0.3e1 * E * DEPSXXB0 - 0.8e1 /
0.3e1 * EU * DEPSYYB0) * SIGYYBC + 0.4e1 * DEPSXYB0 * TAUXYBC * (V - 0.1e1 / 0.2e1) * E) * sin(PHIM)
+ ((((-3 * DEPSXXB0 + 2 * DEPSYYB0) * E + EU * DEPSYYB0) * V + 2 * E * DEPSXXB0 - EU * DEPSYYB0)
* SIGXXBC) + ((((3 * DEPSXXB0 - DEPSYYB0) * E - 2 * EU * DEPSYYB0) * V - E * DEPSXXB0 + 2 * EU *
DEPSYYB0) * SIGYYBC) - 0.12e2 * DEPSXYB0 * TAUXYBC * (V - 0.1e1 / 0.2e1) * E) * sin(PSIM) - 0.3e1 *
(SIGXXBC - 2 * SIGYYBC) * (DEPSYYB0 * (E - EU) * V + E * DEPSXXB0 + EU * DEPSYYB0) * sin(PHIM)))
/ ((((((24 * V ^ 2 * SB + (8 * E + 19 * EU + 12 * SB) * V - 28 * E - 19 * EU - 12 * SB) * SIGXXBC ^ 2) - 0.26e2
* (0.12e2 / 0.13e2 * (V ^ 2) * SB + (E + 0.14e2 / 0.13e2 * EU + 0.6e1 / 0.13e2 * SB) * V - 0.14e2 / 0.13e2 * E
- 0.14e2 / 0.13e2 * EU - 0.6e1 / 0.13e2 * SB) * SIGYYBC * SIGXXBC + ((24 * V ^ 2 * SB + (-E + 28 * EU +
12 * SB) * V - 19 * E - 28 * EU - 12 * SB) * SIGYYBC ^ 2) + 0.168e3 * (0.3e1 / 0.7e1 * (V ^ 2) * SB + (E +
0.2e1 / 0.7e1 * EU + 0.3e1 / 0.14e2 * SB) * V - 0.13e2 / 0.14e2 * E - 0.2e1 / 0.7e1 * EU - 0.3e1 / 0.14e2 * SB)
* TAUXYBC ^ 2) * sin(PHIM) + ((-72 * V ^ 2 * SB + (-72 * E - 9 * EU - 36 * SB) * V + 36 * E + 9 * EU + 36 *
SB) * SIGXXBC ^ 2) + 0.126e3 * (0.4e1 / 0.7e1 * (V ^ 2) * SB + (E + 0.2e1 / 0.7e1 * EU + 0.2e1 / 0.7e1 * SB)
* V - 0.2e1 / 0.7e1 * E - 0.2e1 / 0.7e1 * EU - 0.2e1 / 0.7e1 * SB) * SIGYYBC * SIGXXBC + ((-72 * V ^ 2 * SB
+ (-45 * E - 36 * EU - 36 * SB) * V + 9 * E + 36 * EU + 36 * SB) * SIGYYBC ^ 2) - 0.648e3 * TAUXYBC ^ 2 *
(V - 0.1e1 / 0.2e1) * ((V * SB) / 0.3e1 + E + SB / 0.3e1)) * sin(PSIM) - 0.72e2 * (((V ^ 2 * SB) + (E + EU / 0.8e1
+ SB / 0.2e1) * V - E / 0.2e1 - EU / 0.8e1 - SB / 0.2e1) * (SIGXXBC ^ 2) - 0.7e1 / 0.4e1 * (0.4e1 / 0.7e1 * (V
^ 2) * SB + (E + 0.2e1 / 0.7e1 * EU + 0.2e1 / 0.7e1 * SB) * V - 0.2e1 / 0.7e1 * E - 0.2e1 / 0.7e1 * EU - 0.2e1
/ 0.7e1 * SB) * SIGYYBC * SIGXXBC + ((V ^ 2 * SB) + (0.5e1 / 0.8e1 * E + EU / 0.2e1 + SB / 0.2e1) * V - E /
0.8e1 - EU / 0.2e1 - SB / 0.2e1) * (SIGYYBC ^ 2) + 0.9e1 * TAUXYBC ^ 2 * (V - 0.1e1 / 0.2e1) * ((V * SB) /
0.3e1 + E + SB / 0.3e1)) * (sin(PHIM) - 0.3e1)) * sqrt((SIGXXBC ^ 2) - (SIGXXBC * SIGYYBC) + (SIGYYBC
^ 2) + 0.3e1 * TAUXYBC ^ 2) - 0.16e2 * ((SIGXXBC ^ 2) - (SIGXXBC * SIGYYBC) + (SIGYYBC ^ 2) + 0.3e1
* TAUXYBC ^ 2) * ((sin(PHIM) - 0.3e1 / 0.4e1) * sin(PSIM) - 0.9e1 / 0.4e1 * sin(PHIM)) * ((((E - EU) * V - 2 *
E + EU) * SIGXXBC) - 0.2e1 * (((E - EU) * V) - E / 0.2e1 + EU) * SIGYYBC));

        DEPSPXYB = -0.54e2 * (((((DEPSXXB0 - 0.2e1 / 0.3e1 * DEPSYYB0) * E - EU * DEPSYYB0 / 0.3e1)
* SIGXXBC + ((-DEPSXXB0 + DEPSYYB0 / 0.3e1) * E + 0.2e1 / 0.3e1 * EU * DEPSYYB0) * SIGYYBC +
0.4e1 * TAUXYBC * E * DEPSXYB0) * V + (-0.2e1 / 0.3e1 * E * DEPSXXB0 + EU * DEPSYYB0 / 0.3e1) *
SIGXXBC + (E * DEPSXXB0 / 0.3e1 - 0.2e1 / 0.3e1 * EU * DEPSYYB0) * SIGYYBC - 0.2e1 * TAUXYBC * E
* DEPSXYB0) * (sin(PHIM) - 0.3e1) * sqrt(SIGXXBC ^ 2 - SIGXXBC * SIGYYBC + SIGYYBC ^ 2 + 0.3e1 *
TAUXYBC ^ 2) + 0.4e1 / 0.3e1 * sin(PHIM) * (SIGXXBC ^ 2 - SIGXXBC * SIGYYBC + SIGYYBC ^ 2 + 0.3e1
* TAUXYBC ^ 2) * (DEPSYYB0 * (E - EU) * V + E * DEPSXXB0 + EU * DEPSYYB0)) * (-0.3e1 + sin(PSIM))
* TAUXYBC / (0.16e2 * ((SIGXXBC - 0.2e1 * SIGYYBC) * (E - EU) * V + (-0.2e1 * E + EU) * SIGXXBC +
SIGYYBC * (E - 0.2e1 * EU)) * ((sin(PSIM) - 0.9e1 / 0.4e1) * sin(PHIM) - 0.3e1 / 0.4e1 * sin(PSIM)) *
sqrt(SIGXXBC ^ 2 - SIGXXBC * SIGYYBC + SIGYYBC ^ 2 + 0.3e1 * TAUXYBC ^ 2) + ((-0.24e2 * SB *
(SIGXXBC ^ 2 - SIGXXBC * SIGYYBC + SIGYYBC ^ 2 + 0.3e1 * TAUXYBC ^ 2) * V ^ 2 + ((-0.8e1 * E -
0.19e2 * EU - 0.12e2 * SB) * SIGXXBC ^ 2 + 0.26e2 * SIGYYBC * (E + 0.14e2 / 0.13e2 * EU + 0.6e1 / 0.13e2
* SB) * SIGXXBC + (E - 0.28e2 * EU - 0.12e2 * SB) * SIGYYBC ^ 2 - 0.168e3 * TAUXYBC ^ 2 * (E + 0.2e1 /
0.7e1 * EU + 0.3e1 / 0.14e2 * SB)) * V + (0.28e2 * E + 0.19e2 * EU + 0.12e2 * SB) * SIGXXBC ^ 2 - 0.28e2
* (E + EU + 0.3e1 / 0.7e1 * SB) * SIGYYBC * SIGXXBC + (0.19e2 * E + 0.28e2 * EU + 0.12e2 * SB) *
SIGYYBC ^ 2 + 0.156e3 * (E + 0.4e1 / 0.13e2 * EU + 0.3e1 / 0.13e2 * SB) * TAUXYBC ^ 2) * sin(PSIM) +
0.72e2 * SB * (SIGXXBC ^ 2 - SIGXXBC * SIGYYBC + SIGYYBC ^ 2 + 0.3e1 * TAUXYBC ^ 2) * V ^ 2 +
((0.72e2 * E + 0.9e1 * EU + 0.36e2 * SB) * SIGXXBC ^ 2 - 0.126e3 * SIGYYBC * (E + 0.2e1 / 0.7e1 * EU +
0.2e1 / 0.7e1 * SB) * SIGXXBC + (0.45e2 * E + 0.36e2 * EU + 0.36e2 * SB) * SIGYYBC ^ 2 + 0.648e3 *
TAUXYBC ^ 2 * (E + SB / 0.6e1)) * V + (-0.36e2 * E - 0.9e1 * EU - 0.36e2 * SB) * SIGXXBC ^ 2 + 0.36e2 *
SIGYYBC * (E + EU + SB) * SIGXXBC + (-0.9e1 * E - 0.36e2 * EU - 0.36e2 * SB) * SIGYYBC ^ 2 - 0.324e3
* TAUXYBC ^ 2 * (E + SB / 0.3e1)) * sin(PHIM) + 0.72e2 * (-0.3e1 + sin(PSIM)) * (SB * (SIGXXBC ^ 2 -
SIGXXBC * SIGYYBC + SIGYYBC ^ 2 + 0.3e1 * TAUXYBC ^ 2) * V ^ 2 + ((E + EU / 0.8e1 + SB / 0.2e1) *
SIGXXBC ^ 2 - 0.7e1 / 0.4e1 * SIGYYBC * (E + 0.2e1 / 0.7e1 * EU + 0.2e1 / 0.7e1 * SB) * SIGXXBC + (0.5e1
/ 0.8e1 * E + EU / 0.2e1 + SB / 0.2e1) * SIGYYBC ^ 2 + 0.9e1 * TAUXYBC ^ 2 * (E + SB / 0.6e1)) * V + (-E /
0.2e1 - EU / 0.8e1 - SB / 0.2e1) * SIGXXBC ^ 2 + SIGYYBC * (E + EU + SB) * SIGXXBC / 0.2e1 + (-E / 0.8e1
- EU / 0.2e1 - SB / 0.2e1) * SIGYYBC ^ 2 - 0.9e1 / 0.2e1 * TAUXYBC ^ 2 * (E + SB / 0.3e1))) * (SIGXXBC ^
2 - SIGXXBC * SIGYYBC + SIGYYBC ^ 2 + 0.3e1 * TAUXYBC ^ 2) ^ (-0.1e1 / 0.2e1);

        % recalculate softening modulus SUC:
        DLB = -0.18e2 * (((((DEPSXXB0 - 0.2e1 / 0.3e1 * DEPSYYB0) * E - EU * DEPSYYB0 / 0.3e1) *
SIGXXBC + ((-DEPSXXB0 + DEPSYYB0 / 0.3e1) * E + 0.2e1 / 0.3e1 * EU * DEPSYYB0) * SIGYYBC + 0.4e1
* TAUXYBC * E * DEPSXYB0) * V + (-0.2e1 / 0.3e1 * E * DEPSXXB0 + EU * DEPSYYB0 / 0.3e1) * SIGXXBC
+ (E * DEPSXXB0 / 0.3e1 - 0.2e1 / 0.3e1 * EU * DEPSYYB0) * SIGYYBC - 0.2e1 * TAUXYBC * E *
DEPSXYB0) * (sin(PHIM) - 0.3e1) * sqrt(SIGXXBC ^ 2 - SIGXXBC * SIGYYBC + SIGYYBC ^ 2 + 0.3e1 *
TAUXYBC ^ 2) + 0.4e1 / 0.3e1 * sin(PHIM) * (SIGXXBC ^ 2 - SIGXXBC * SIGYYBC + SIGYYBC ^ 2 + 0.3e1
* TAUXYBC ^ 2) * (DEPSYYB0 * (E - EU) * V + E * DEPSXXB0 + EU * DEPSYYB0)) * (-0.3e1 + sin(PSIM))
/ (0.16e2 * ((SIGXXBC - 0.2e1 * SIGYYBC) * (E - EU) * V + (-0.2e1 * E + EU) * SIGXXBC + SIGYYBC * (E -
0.2e1 * EU)) * ((sin(PSIM) - 0.9e1 / 0.4e1) * sin(PHIM) - 0.3e1 / 0.4e1 * sin(PSIM)) * sqrt(SIGXXBC ^ 2 -
SIGXXBC * SIGYYBC + SIGYYBC ^ 2 + 0.3e1 * TAUXYBC ^ 2) + ((-0.24e2 * SB * (SIGXXBC ^ 2 - SIGXXBC
* SIGYYBC + SIGYYBC ^ 2 + 0.3e1 * TAUXYBC ^ 2) * V ^ 2 + ((-0.8e1 * E - 0.19e2 * EU - 0.12e2 * SB) *
SIGXXBC ^ 2 + 0.26e2 * SIGYYBC * (E + 0.14e2 / 0.13e2 * EU + 0.6e1 / 0.13e2 * SB) * SIGXXBC + (E -
0.28e2 * EU - 0.12e2 * SB) * SIGYYBC ^ 2 - 0.168e3 * TAUXYBC ^ 2 * (E + 0.2e1 / 0.7e1 * EU + 0.3e1 /
0.14e2 * SB)) * V + (0.28e2 * E + 0.19e2 * EU + 0.12e2 * SB) * SIGXXBC ^ 2 - 0.28e2 * (E + EU + 0.3e1 /
0.7e1 * SB) * SIGYYBC * SIGXXBC + (0.19e2 * E + 0.28e2 * EU + 0.12e2 * SB) * SIGYYBC ^ 2 + 0.156e3 *
(E + 0.4e1 / 0.13e2 * EU + 0.3e1 / 0.13e2 * SB) * TAUXYBC ^ 2) * sin(PSIM) + 0.72e2 * SB * (SIGXXBC ^ 2
- SIGXXBC * SIGYYBC + SIGYYBC ^ 2 + 0.3e1 * TAUXYBC ^ 2) * V ^ 2 + ((0.72e2 * E + 0.9e1 * EU + 0.36e2
* SB) * SIGXXBC ^ 2 - 0.126e3 * SIGYYBC * (E + 0.2e1 / 0.7e1 * EU + 0.2e1 / 0.7e1 * SB) * SIGXXBC +
(0.45e2 * E + 0.36e2 * EU + 0.36e2 * SB) * SIGYYBC ^ 2 + 0.648e3 * TAUXYBC ^ 2 * (E + SB / 0.6e1)) * V
+ (-0.36e2 * E - 0.9e1 * EU - 0.36e2 * SB) * SIGXXBC ^ 2 + 0.36e2 * SIGYYBC * (E + EU + SB) * SIGXXBC
+ (-0.9e1 * E - 0.36e2 * EU - 0.36e2 * SB) * SIGYYBC ^ 2 - 0.324e3 * TAUXYBC ^ 2 * (E + SB / 0.3e1)) *
sin(PHIM) + 0.72e2 * (-0.3e1 + sin(PSIM)) * (SB * (SIGXXBC ^ 2 - SIGXXBC * SIGYYBC + SIGYYBC ^ 2 +
0.3e1 * TAUXYBC ^ 2) * V ^ 2 + ((E + EU / 0.8e1 + SB / 0.2e1) * SIGXXBC ^ 2 - 0.7e1 / 0.4e1 * SIGYYBC *
(E + 0.2e1 / 0.7e1 * EU + 0.2e1 / 0.7e1 * SB) * SIGXXBC + (0.5e1 / 0.8e1 * E + EU / 0.2e1 + SB / 0.2e1) *
SIGYYBC ^ 2 + 0.9e1 * TAUXYBC ^ 2 * (E + SB / 0.6e1)) * V + (-E / 0.2e1 - EU / 0.8e1 - SB / 0.2e1) *
SIGXXBC ^ 2 + SIGYYBC * (E + EU + SB) * SIGXXBC / 0.2e1 + (-E / 0.8e1 - EU / 0.2e1 - SB / 0.2e1) *
SIGYYBC ^ 2 - 0.9e1 / 0.2e1 * TAUXYBC ^ 2 * (E + SB / 0.3e1)));

        FKB = -0.3e1 / (0.3e1 - sin(PHIM)) * (0.1e1 - sin(PHIM)) * KCB;
        SBC = -FKB* DKB / DLB ;
        if abs(SBC-SB) < TOR2
           break;
        else
           SB = SBC;
        end
```

```
      end
   end

   EPSPXXU = EPSPXXU + DEPSPXXU;
   EPSPYYU = EPSPYYU + DEPSPYYU;
   % vertical elastic stress should always be positive:
   EPSXXUE = (SIGXXUE-VU*SIGYYUE)/EU;
   EPSYYUE = (SIGYYUE-VU*SIGXXUE)/EU;
   if EPSPYYU > EPSYYUE
      if EPSPXXU > EPSXXUE
         EPSPXXU = EPSXXUE;
      else
      end
      EPSPYYU = EPSYYUE;
   else
   end

   EPSPXXH = EPSPXXH + DEPSPXXH;
   EPSPYYH = EPSPYYH + DEPSPYYH;
   % elastic stress should always be positive:
   EPSXXHE = (SIGXXHE-V*SIGYYHE)/E;
   EPSYYHE = (SIGYYHE-V*SIGXXHE)/E;
   if EPSPYYH > EPSYYHE
      if EPSPXXH > EPSXXHE
         EPSPXXH = EPSXXHE;
      else
      end
      EPSPYYH = EPSYYHE;
   else
   end

   EPSPXXB = EPSPXXB + DEPSPXXB;
   EPSPYYB = EPSPYYB + DEPSPYYB;
   EPSPXYB = EPSPXYB + DEPSPXYB;
   % vertical elastic predicted stress should always be positive:
   EPSXXBE = (SIGXXBE-V*SIGYYBE)/E;
   EPSYYBE = (SIGYYBE-V*SIGXXBE)/E;
   EPSXYBE = TAUXYBE/(2*GB);
   if EPSPYYB > EPSYYBE
      if EPSPXXB > EPSXXBE
         if EPSPXYB > EPSXYBE
            EPSPXYB = EPSXYBE;
         else
         end
         EPSPXXB = EPSXXBE;
      else
      end
      EPSPYYB = EPSYYBE;
   else
   end

%     EPSPXXU = 0;
%     EPSPYYU = 0;
%     EPSPXXH = 0;
%     EPSPYYH = 0;
%     EPSPXXB = 0;
%     EPSPYYB = 0;
%     EPSPXYB = 0;

   % damage factor:
   while DH < 1 & DU < 1 & DC < 1 & DB < 1
      % damage model
      % Stresses with plastic corrector of each component in x direction:
      SIGXXUP = (EU*(1-VU)*(EPSXXUE-EPSPXXU))/((1+VU)*(1-2*VU))+(EU*(VU)*(EPSYYUE-EPSPYYU))/((1+VU)*(1-2*VU));

      SIGXXHP = (EH*(1-V)*(EPSXXHE-EPSPXXH))/((1+V)*(1-2*V))+(EH*(V)*(EPSYYHE-EPSPYYH))/((1+V)*(1-2*V));

      SIGXXCP = RB*(EB*(1-V)*(EPSXXBE-EPSPXXB))/((1+V)*(1-2*V))+(EB*(V)*(EPSYYBE-EPSPYYB))/((1+V)*(1-2*V))/RC;
      % shear stresses with plastic corrector of each component in shear direction:
      TAUXYBP = 2*GB*(EPSXYBE-EPSPXYB);

      %find maximum stress between stress at n step and intial maximum value
      SXH = max(SIGXXHP,SIGTM); % head joint
      SXU = max(SIGXXUP,SIGTU); % brick unit
      SXC = max(SIGXXCP,SIGTM); % cross joint
      TXYB = max(abs(TAUXYBP),SIGS); % bed joint
      % with smeared crack model
      ATM = (((GIM*EH)/(LT*SIGTM^2))-(1/2))^(-1); % paramter AT/AS of each component
      ATU = (((GIU*EU)/(LT*SIGTU^2))-(1/2))^(-1);
      ASB = (((GII*GB)/(LS*SIGS^2))-(1/2))^(-1);

      % Calculate damage factor from internal stresses
      DHC = 1-(SIGTM*exp(ATM*(1-(SXH/SIGTM)))/SXH); % DH should not increasing
      DUC = 1-(SIGTU*exp(ATU*(1-(SXU/SIGTU)))/SXU); % once DH tend to increased, brick is damaged
      DBC = 1-(SIGS*exp(ASB*(1-(TXYB/SIGS)))/TXYB);
      if abs(DHC/DH) < 1
         if abs(DUC/DU) < 1
            if abs(DBC/DB) < 1
              DBC = DB;
            else
            end
            DUC = DU;
         else
         end
         DHC = DH;
         SWC = 1;
      else
      end
      if SIGXXCP < 0 % cross joint failure in tension only happened once sigxxc < 0 (tension stress)
         DCC = 1-(SIGTM*exp(ATM*(1-(SXC/SIGTM)))/SXC);
      else
         DCC = (DB+DH)/2;
      end

      % Verification of damage factor
      % Since damage factor will influence stress itself
      % damage factor should be verificated together
      if DHC >= 0 & DUC >=0 & DCC >=0 & DBC >= 0
         if abs(DHC-DH) < TOR
            if abs(DUC-DU) < TOR
```

```
                if abs(DCC-DC) < TOR
                    if abs(DBC-DB) < TOR
                        break;
                    else
                        DB = DBC;
                    end
                else
                    DC = DCC;
                end
            else
                DU = DUC;
            end
        else
            DH = DHC;
        end
    else
        break;
    end
end

EPSXX0YT        =        - ((EPSXXUE)* C2 + (EPSXXHE))/(C2+1);
EPSXX0YP        =        - ((EPSPXXU)* C2 + (EPSPXXH))/(C2+1);

SIGYYUP         =         (EU*(1-VU)*(EPSYYUE-EPSPYYU))/((1+VU)*(1-2*VU))+(EU*(VU)*(EPSXXUE-
EPSPXXU))/((1+VU)*(1-2*VU));

SIGYYHP              =              (EH*(1-V)*(EPSYYHE-EPSPYYH))/((1+V)*(1-2*V))+(EH*(V)*(EPSXXHE-
EPSPXXH))/((1+V)*(1-2*V));

% total undamaged stress of cell
SIGYY0C = (RH*SIGYYHP + C2*RU*SIGYYUP)/(C2+1);
if SWC == 1
    SIGYY0 = SIGYY0;
else
    SIGYY0 = SIGYY0C;
end

% record value:
a = [a,SIGYY0];
b = [b,RH*SIGYYHP];
d = [d,FU];
e = [e,RB*TAUXYBP];
f =  [f,RU*SIGYYUP];
g = [g,EPSXX0YT];
h = [h,EPSXX0YP];
end
```

Appendix E: MATLAB Code (Model 4)

```
clear all;
% properties setting:
E = 5091; C1 = 1.77;
EU = C1*E; EH = E; EB = E; EC = E;
V = 0.02; % Poisson's ratio of mortar
VU = 0.14; % Poisson's ratio of brick
GB = E/(2*(1+V)); % shear modulus of mortar

% friction and dilatancy angle:
PHIU = (23.27*pi)/180;
PSIU = (10*pi)/180;
PHIM = (23.27*pi)/180;
PSIM = (10*pi)/180;

% I and II fracture energy
GIU = 0.081; GIM = 0.082;
GII = 0.012;
% compressive fracture energy
GCU = 20.96; GCM = 17.68;

% Shear, tension and compressive strength:
SIGTU = 2.74; % Tension strength of brick unit
SIGTM = 2.79; % Tension strength of mortar
FCU   = 16; %compressive strength of brick
FCM   = 6.59; % compressive strength of mortar
% Shear strength of mortar:
SIGS  = 0.14; % should always be smaller than "2c*cos(phi)^2/(1-sin(phi))" with cmax = fc

% maximum strain of strain-stress curve under compression
EPS0U = 2*FCU/EU;
EPS0M = 2*FCM/E;

% geometrical properties:
C2 = 21.2; C3 = 7.1; % properties of masonry: L=C2*T, H=C3*T

% initial value of external strain and damage factor:
EPSYY0 = 0; EPSXX0 = 0; EPSXY0 = 0; % external strain
DEPSYY0 = - 0.00001; % external vertical strain increment
DEPSXX0 = 0.00001; % external horizantal strain increment
DEPSXY0 = 0.00001; % external shear strain increment

% initial value of plastic strain of each component:
EPSPXXU = 0;EPSPYYU = 0; % initial value of plastic strain
EPSPXXH = 0;EPSPYYH = 0;
EPSPXXB = 0;EPSPYYB = 0;EPSPXYB = 0;
KU = 0; KH = 0; KB = 0; % initial value of hardening(softening) parameter

% initialize value of variables:
DH = 0; DU = 0; DB = 0; DC = 0; % damage varaiables of compression splitting
SU = 0; SH = 0; SB = 0; %softening modulus
DEPSPXXU = 0;DEPSPYYU = 0; % initialized value of plastic strain increment
DEPSPXXH = 0;DEPSPYYH = 0;
DEPSPXXB = 0;DEPSPYYB = 0;DEPSPXYB = 0;
% damage variables of tension behaviour
```

```
DHX = 0; DUX = 0; DBX = 0; DCX = 0;
% damage variable of shear behaviour
DBXY = 0;

% ATM must be positive, check maximum mesh size: HH < 80
% ATU must be positive, check maximum mesh size: HH < 59
% AS must be positive, check maximum mesh size: HH <74
HH = 50; % element size
LT = HH; LS = HH; LC = HH; % characteristic length of element is element size

% Tolerance of calculted and assumed damge factor
TOR = 0.00001; % verify damage factor
TOR2 = 0.00001; % verify hardening modulus

% switch code: SWC
SWC = 0; % when damage factor decresed, error happened then SWC = 1
SIGYY0 = 0;
a = []; b = []; d = []; e = []; f = [];g = []; h = [];

% outer loop: strain integration
for i = 1:1000
   EPSYY0 = EPSYY0 + DEPSYY0;
   EPSXX0 = EPSXX0 + DEPSXX0;
   EPSXY0 = EPSXY0 + DEPSXY0;

   % inner loop: verification of damage factor
   % undamage factor of each component for compression splitting
   RH = 1-DH;
   RB = 1-DB;
   RC = 1-DC;
   RU = 1-DU;

   % undamage factor of each component for horizontal tension behaviour
   RHX = 1-DHX;
   RBX = 1-DBX;
   RCX = 1-DCX;
   RUX = 1-DUX;

   EPSXX0Y = -(-(2 * RBX + (C2 - 1) * RCX) * C1 * RBX * ((E * RHX * (C2 * VU + V) * C3 ^ 2) + (-(VU * GB
* (C2 - 1) * (C2 + 1) * (V - 1) * (V + 1) * RBX) / 0.4e1 + (((-V ^ 2 + 1) * C1 * RUX + RHX * (V * VU + C2)) * E
* V)) * C3 - (V * GB * RBX * (C2 - 1) * (C2 + 1) * (V - 1) * (V + 1)) / 0.4e1) * (C3 + 1) * EPSYY0 * RUX / (-
(RBX * RHX * C1 * (2 * RBX + (C2 - 1) * RCX) * RUX * E * C3 ^ 3) + ((RUX * C1 * GB * (V - 1) * (V + 1) *
(C2 - 1) * RBX ^ 3) / 0.2e1 + (-0.2e1 * RUX * (-(RCX * (C2 - 1) ^ 2 * (V - 1) * (V + 1) * GB) / 0.8e1 + E * ((-(V
^ 2) / 0.2e1 + 0.1e1 / 0.2e1) * RCX + (RHX * V * VU))) * C1 + (E * C2 * RCX * RHX * (VU - 1) * (VU + 1))) *
(RBX ^ 2) + 0.2e1 * C1 * ((RUX * (V ^ 2 - 1) * C1) - (V * RCX * (2 * C2 * VU + V - VU)) / 0.2e1) * RHX * E *
RUX * RBX + (E * RUX ^ 2 * C1 ^ 2 * RCX * RHX * (V - 1) * (V + 1) * (C2 - 1))) * (C3 ^ 2) - (-(C2 - 1) * (V +
1) * ((V * VU * RUX * C1) - (RCX * (VU - 1) * (VU + 1) * (C2 + 1)) / 0.2e1) * GB * (V - 1) * (RBX ^ 2) / 0.2e1
+ ((C2 - 1) * (V + 1) * C1 * RUX * GB * (RUX * (V ^ 2 - 1) * C1 - C2 * RCX * V * VU) * (V - 1) * RBX) / 0.2e1
+ C1 * (-(V + 1) * RUX * (-((C2 - 1) ^ 2 * (V - 1) * (V + 1) * GB) / 0.4e1 + E) * (V - 1) * C1 + (E * RHX * (V *
VU + C2))) * RUX * RCX) * RBX * C3 + (RUX * C1 * GB * RBX ^ 2 * RCX * (C2 - 1) * (C2 + 1) * (V - 1) * (V
+ 1)) / 0.4e1) / (C2 + 1));

   EPSYY0X = -(EPSXX0 * ((C3 * GB * (V - 1) * (V + 1) * (C3 * RC + 2 * RH) * (C2 - 1) * (C1 * C2 * RU +
RH) * RB ^ 3) / 0.2e1 + (-(2 * E * RU * C1 * RC * RH * (C2 + 1) * C3 ^ 3) + ((GB * (C2 - 1) ^ 2 * (V - 1) * (V +
1) * (C1 * C2 * RU + RH) * RC ^ 2) / 0.4e1 - (4 * E * RU * C1 * RH ^ 2 * (C2 + 1))) * (C3 ^ 2) + (GB * (V - 1) *
(C2 - 1) * RC * (V + 1) * ((C2 + 1) * (C1 * C2 * RU - RH) * RC + 2 * RH * ((C2 - 1) * RH + C1 * RU * (C2 ^ 2 -
C2 + 2))) * C3) / 0.4e1 + (RH * C1 * GB * (V - 1) * (C2 - 1) * RU * ((C2 ^ 2 + C2) * RC + 2 * RH) * (V + 1)) /
0.2e1) * (RB ^ 2) + (-(E * RU * C1 * RC ^ 2 * RH * (C2 - 1) * (C2 + 1) * C3 ^ 3) + ((((-V ^ 2 + 1) * C2 * RH ^ 2
+ (-C2 ^ 2 + 1) * RU * C1 * RH + RU ^ 2 * C1 ^ 2 * C2 * (V - 1) * (V + 1)) * RC + 2 * RH * C1 * ((-C2 ^ 2 + V ^
2) * RH + RU * (V - 1) * (V + 1) * C1) * RU) * RC * E * C3 ^ 2) + 0.2e1 * RH * C1 * ((GB * (C2 - 1) ^ 2 * (V -
1) * (V + 1) * RC ^ 2) / 0.4e1 + (((-V ^ 2 - C2) * RH + RU * (V - 1) * (V + 1) * C1) * E * C2 * RC) + (2 * E * RU
* C1 * RH * (V - 1) * (V + 1))) * RU * C3 + (C1 * RU * GB * RC * RH ^ 2 * (C2 - 1) ^ 2 * (V - 1) * (V + 1)) /
0.2e1) * RB + (RH * C1 * (C3 * (C1 * RU + RH) * RC + 2 * RH * C1 * RU) * (V - 1) * (C2 - 1) * RC * E * RU *
C3 * (V + 1))) * (C2 + 1) * V / ((C2 - 1) * RC + 2 * RB) / (C3 + 1) / ((GB * (V - 1) * (C2 - 1) * (C2 + 1) * (V + 1)
* (C3 * (C1 * C2 * RU + RH) * RC + 2 * C1 * C2 * RU * RH) * RB ^ 2) / 0.4e1 + (RC * E * ((C2 * (V ^ 2 - 1) *
RH ^ 2) - 0.2e1 * C1 * RU * ((C2 * V ^ 2) + (C2 ^ 2) / 0.2e1 + 0.1e1 / 0.2e1) * RH + (RU ^ 2 * C1 ^ 2 * C2 *
(V - 1) * (V + 1))) * (C3 ^ 2) + (2 * RH * C1 * ((-V ^ 2 - C2) * RH + RU * (V - 1) * (V + 1) * C1) * E * RU * C3 *
C2) + (C1 * RU * GB * RC * RH * (C2 - 1) * (C2 + 1) * (V - 1) * (V + 1)) / 0.2e1) * RB + (2 * E * RU * C1 * C3
* RC * RH * (V - 1) * (V + 1) * (C1 * RU + C2 * RH))));

    DEPSXX0Y = -(-RUX * (E * RHX * (C2 * VU + V) * C3 ^ 2 + (((-0.25e0 * C2 ^ 2 * VU + 0.25e0 * VU) * V ^
2 + 0.25e0 * C2 ^ 2 * VU - 0.25e0 * VU) * GB * RBX + (-0.1e1 * E * V ^ 3 + E * V) * C1 * RUX + E * RHX * V
* (V * VU + C2)) * C3 + (-0.25e0 * C2 ^ 2 * V ^ 3 + 0.25e0 * C2 ^ 2 * V + 0.25e0 * V ^ 3 - 0.25e0 * V) * GB *
RBX) * (0.2e1 * RBX + (C2 - 0.1e1) * RCX) * DEPSYY0 * C1 * RBX * (C3 + 0.1000000000e1) / (C2 +
0.1000000000e1) / ((-0.2e1 * C1 * E * RBX ^ 2 * RHX * RUX + (E * RHX - 0.1e1 * C2 * E * RHX) * RCX *
C1 * RUX * RBX) * C3 ^ 3 + (((0.5e0 * C2 - 0.5e0) * V ^ 2 + 0.5e0 - 0.5e0 * C2) * GB * C1 * RUX * RBX ^ 3
+ (((0.25e0 * (C2 - 0.1e1) ^ 2 * V ^ 2 - 0.25e0 * (C2 - 0.1e1) ^ 2) * RCX * GB + (E * V ^ 2 - 0.1e1 * E) * RCX
- 0.2e1 * E * RHX * V * VU) * C1 * RUX + (RHX * VU ^ 2 - 0.1e1 * RHX) * E * C2 * RCX) * RBX ^ 2 +
((0.2e1 * RHX * E * V ^ 2 - 0.2e1 * E * RHX) * C1 ^ 2 * RUX ^ 2 + (-0.1e1 * RHX * E * V ^ 2 + (E * RHX * VU
- 0.2e1 * C2 * E * RHX * VU) * V) * RCX * C1 * RUX) * RBX + ((-0.1e1 * E * RHX + C2 * E * RHX) * V ^ 2 +
E * RHX - 0.1e1 * C2 * E * RHX) * RCX * C1 ^ 2 * RUX ^ 2) * C3 ^ 2 + ((((0.5e0 * C2 * VU - 0.5e0 * VU) * V
^ 3 + (-0.5e0 * C2 * VU + 0.5e0 * VU) * V) * GB * C1 * RUX + (((0.25e0 - 0.25e0 * VU ^ 2) * C2 ^ 2 + 0.25e0
* VU ^ 2 - 0.25e0) * V ^ 2 + (0.25e0 * VU ^ 2 - 0.25e0) * C2 ^ 2 + 0.25e0 - 0.25e0 * VU ^ 2) * RCX * GB) *
RBX ^ 3 + (((0.5e0 - 0.5e0 * C2) * V ^ 4 + (C2 - 0.1e1) * V ^ 2 + 0.5e0 - 0.5e0 * C2) * GB * C1 ^ 2 * RUX ^ 2
+ ((0.5e0 * C2 ^ 2 * VU - 0.5e0 * C2 * VU) * V ^ 3 + (-0.5e0 * C2 ^ 2 * VU + 0.5e0 * C2 * VU) * V) * RCX *
GB * C1 * RUX) * RBX ^ 2 + (((-0.25e0 * (C2 - 0.1e1) ^ 2 * V ^ 4 + 0.5e0 * (C2 - 0.1e1) ^ 2 * V ^ 2 - 0.25e0 *
(C2 - 0.1e1) ^ 2) * RCX * GB + (E * V ^ 2 - 0.1e1 * E) * RCX) * C1 ^ 2 * RUX ^ 2 + (-0.1e1 * E * RHX * V *
VU - 0.1e1 * C2 * E * RHX) * RCX * C1 * RUX) * RBX) * C3 + ((0.25e0 * C2 ^ 2 - 0.25e0) * V ^ 2 - 0.25e0 *
C2 ^ 2 + 0.25e0) * RCX * GB * C1 * RUX * RBX ^ 2));

    DEPSYY0X = -(DEPSXX0 * ((C3 * GB * (V - 1) * (V + 1) * (C3 * RC + 2 * RH) * (C2 - 1) * (C1 * C2 * RU +
RH) * RB ^ 3) / 0.2e1 + (-(2 * E * RU * C1 * RC * RH * (C2 + 1) * C3 ^ 3) + (((C2 - 1) ^ 2 * (V - 1) * (V + 1) *
GB * (C1 * C2 * RU + RH) * RC ^ 2) / 0.4e1 - (4 * E * RU * C1 * RH ^ 2 * (C2 + 1))) * (C3 ^ 2) + (GB * (V - 1)
* (C2 - 1) * RC * (V + 1) * ((C2 + 1) * (C1 * C2 * RU - RH) * RC + 2 * RH * ((C2 - 1) * RH + RU * C1 * (C2 ^ 2
- C2 + 2))) * C3) / 0.4e1 + (RH * C1 * GB * (V - 1) * (C2 - 1) * RU * ((C2 ^ 2 + C2) * RC + 2 * RH) * (V + 1)) /
0.2e1) * (RB ^ 2) + (-(E * RU * C1 * RC ^ 2 * RH * (C2 - 1) * (C2 + 1) * C3 ^ 3) + ((((-V ^ 2 + 1) * C2 * RH ^ 2
+ (-C2 ^ 2 + 1) * RU * C1 * RH + RU ^ 2 * C1 ^ 2 * C2 * (V - 1) * (V + 1)) * RC + 2 * RH * C1 * ((-C2 ^ 2 + V ^
2) * RH + RU * (V - 1) * (V + 1) * C1) * RU) * RC * E * C3 ^ 2) + 0.2e1 * RH * C1 * (((C2 - 1) ^ 2 * (V - 1) * (V
+ 1) * GB * RC ^ 2) / 0.4e1 + (((-V ^ 2 - C2) * RH + RU * (V - 1) * (V + 1) * C1) * E * C2 * RC) + (2 * E * RU *
C1 * RH * (V - 1) * (V + 1))) * RU * C3 + (RU * C1 * GB * RC * RH ^ 2 * (C2 - 1) ^ 2 * (V - 1) * (V + 1)) /
0.2e1) * RB + (RH * C1 * (C3 * (RU * C1 + RH) * RC + 2 * RH * C1 * RU) * (V - 1) * (C2 - 1) * RC * E * RU *
C3 * (V + 1))) * (C2 + 1) * V / ((C2 - 1) * RC + 2 * RB) / (C3 + 1) / ((GB * (V - 1) * (C2 - 1) * (C2 + 1) * (V + 1)
* (C3 * (C1 * C2 * RU + RH) * RC + 2 * C1 * C2 * RU * RH) * RB ^ 2) / 0.4e1 + (RC * E * ((C2 * (V ^ 2 - 1) *
RH ^ 2) - 0.2e1 * RU * C1 * ((C2 * V ^ 2) + (C2 ^ 2) / 0.2e1 + 0.1e1 / 0.2e1) * RH + (RU ^ 2 * C1 ^ 2 * C2 *
(V - 1) * (V + 1))) * (C3 ^ 2) + (2 * RH * C1 * ((-V ^ 2 - C2) * RH + RU * (V - 1) * (V + 1) * C1) * E * RU * C3 *
C2) + (RU * C1 * GB * RC * RH * (C2 - 1) * (C2 + 1) * (V - 1) * (V + 1)) / 0.2e1) * RB + (2 * E * RU * C1 * C3
* RC * RH * (V - 1) * (V + 1) * (RU * C1 + RH * C2))));

    % vertical stress:
    EPSYY0TC = EPSYY0 + EPSYY0X;
    DEPSYY0TC = DEPSYY0 + DEPSYY0X;

    % horizantal stress:
```

```
    EPSXX0TC = EPSXX0 + EPSXX0Y;
    DEPSXX0TC = DEPSXX0 + DEPSXX0Y;

    % material properties: with smeared crack model
    ATM = (((GIM*EH)/(LT*SIGTM^2))-(1/2))^(-1); % paramter AT/AS of each component
    ATU = (((GIU*EU)/(LT*SIGTU^2))-(1/2))^(-1);
    ASB = (((GII*GB)/(LS*SIGS^2))-(1/2))^(-1);

    % compression behaviour (without shear loading): brick crushing
    EPSYY0T = abs(EPSYY0TC);
    DEPSYY0T = abs(DEPSYY0TC);

    % drucker pragar plasiticy model of brick unit
    % elastic predictor of brick unit
    SIGXXUE = -0.4e1 * (C3 + 1) * (-GB * (V - 1) * (V + 1) * ((-C2 / 0.2e1 - 0.1e1 / 0.2e1) * RC + RU * C1) *
(RC * C3 + (2 * RH)) * (C2 - 0.1e1) * RB ^ 2 / 0.2e1 + (0.2e1 * RC * E * RH * ((-C2 / 0.2e1 - 0.1e1 / 0.2e1) *
RC + RU * C1) * (C3 ^ 2) + (-RU * C1 * GB * (C2 - 0.1e1) ^ 2 * (V - 1) * (V + 1) * RC ^ 2 / 0.4e1 - 0.2e1 * RC
* (RH ^ 2) * E * C2 + 0.4e1 * RU * C1 * E * (RH ^ 2)) * C3 - RU * C1 * GB * RC * RH * (C2 - 0.1e1) ^ 2 * (V -
1) * (V + 1) / 0.2e1) * RB + RC * E * RU * C1 * C3 * (RC * (C2 - 0.1e1) * C3 - 0.2e1 * RC + (0.2e1 * C2 -
0.2e1) * RH) * RH) * E * C1 * RB * V * EPSYY0T / (-0.2e1 * GB * (V - 1) * (V + 1) * C3 * (-RU * C1 * C3 - (V
- 1) * (V + 1) * (C2 + 0.1e1) * RC / 0.2e1 + RU * C1 * (V ^ 2)) * (RC * C3 + (2 * RH)) * (C2 - 0.1e1) * RB ^ 3 -
0.4e1 * (0.2e1 * RU * C1 * E * (C3 ^ 3) * RH + ((V - 1) * (RU * (-(C2 - 0.1e1) ^ 2 * GB / 0.4e1 + E) * C1 + E *
C2 * RH) * (V + 1) * RC - 0.2e1 * RU * C1 * E * RH * (V ^ 2)) * (C3 ^ 2) + GB * (V - 1) * RU * (V + 1) * C1 * (-
(V ^ 2) * RC + RU * (V - 1) * (V + 1) * C1) * (C2 - 0.1e1) * C3 / 0.2e1 + RU * C1 * GB * RC * (C2 - 0.1e1) *
(C2 + 0.1e1) * (V - 1) * (V + 1) / 0.4e1) * (RC * C3 + (2 * RH)) * RB ^ 2 + 0.8e1 * RU * C1 * C3 * (-E * RC ^ 2
* RH * (C2 - 0.1e1) * (C3 ^ 3) / 0.2e1 + RC * E * RH * (RU * (V ^ 2 - 1) * C1 - RH * (C2 - 0.1e1)) * (C3 ^ 2) +
((-RU * (V - 1) * (V + 1) * ((C2 - 0.1e1) ^ 2 * (V - 1) * (V + 1) * GB / 0.4e1 + E) * C1 / 0.2e1 - E * RH * ((V ^ 2)
- C2) / 0.2e1) * RC ^ 2 - E * (V ^ 2) * RC * (RH ^ 2) + 0.2e1 * RU * C1 * E * (RH ^ 2) * (V - 1) * (V + 1)) * C3 -
RC * RH * (RU * (V - 1) * (V + 1) * ((C2 - 0.1e1) ^ 2 * (V - 1) * (V + 1) * GB / 0.4e1 + E) * C1 - E * C2 * RH))
* RB + 0.4e1 * E * RU ^ 2 * C1 ^ 2 * (C3 ^ 2) * RC * RH * (V - 1) * (V + 1) * (RC * C3 + (2 * RH)) * (C2 -
0.1e1));

    SIGYYUE = -0.4e1 * (C3 + 1) * E * ((RC * E * RU * C1 * RH * (2 * RB + RC * (C2 - 1)) * C3 ^ 3) + (-(RU *
C1 * GB * RC * (V - 1) * (V + 1) * (C2 - 1) * RB ^ 2) / 0.2e1 + (((V - 1) * RU * (V + 1) * (-((C2 - 1) ^ 2 * GB) /
0.4e1 + E) * C1 - (E * RH * (V ^ 2 + C2))) * (RC ^ 2) + (4 * RU * C1 * E * RH ^ 2)) * RB - (2 * RC * E * RU *
(V ^ 2 * RC - RH * (C2 - 1)) * C1 * RH)) * (C3 ^ 2) + 0.2e1 * (-GB * (V - 1) * ((-C2 / 0.4e1 - 0.1e1 / 0.4e1) *
(RC ^ 2) + (RU * C1 * RH)) * (V + 1) * (C2 - 1) * RB / 0.2e1 + RC * ((V - 1) * RU * (V + 1) * (-((C2 - 1) ^ 2 *
GB) / 0.4e1 + E) * C1 - (E * C2 * RH)) * RH) * RB * C3 + (GB * RB ^ 2 * RC * RH * (C2 - 1) * (C2 + 1) * (V -
1) * (V + 1)) / 0.2e1) * C1 * RB * EPSYY0T / (-(4 * RC * E * RU * C1 * RB * RH * (2 * RB + RC * (C2 - 1)) *
C3 ^ 4) + ((2 * RU * C1 * GB * RC * (V - 1) * (V + 1) * (C2 - 1) * RB ^ 3) + (-0.4e1 * (V - 1) * (RU * (-((C2 - 1)
^ 2 * GB) / 0.4e1 + E) * C1 + (E * C2 * RH)) * (V + 1) * (RC ^ 2) + (8 * E * RU * V ^ 2 * C1 * RC * RH) - (16 *
RU * C1 * E * RH ^ 2)) * (RB ^ 2) + (8 * RC * E * RU * C1 * RH * (RU * (V ^ 2 - 1) * C1 - RH * (C2 - 1)) * RB)
+ (4 * E * RU ^ 2 * C1 ^ 2 * RC ^ 2 * RH * (V - 1) * (V + 1) * (C2 - 1))) * (C3 ^ 3) + (-0.2e1 * GB * (V - 1) * (V
+ 1) * (-((V - 1) * (V + 1) * (C2 + 1) * RC ^ 2) / 0.2e1 + (C1 * RC * RU * V ^ 2) - (2 * RU * C1 * RH)) * (C2 - 1)
* (RB ^ 3) + ((2 * RU * C1 * V ^ 2 * GB * (V - 1) * (V + 1) * (C2 - 1) * RC ^ 2) - 0.8e1 * (V - 1) * (V + 1) * ((RU
^ 2 * GB * (V - 1) * (V + 1) * (C2 - 1) * C1 ^ 2) / 0.4e1 + RU * (-((C2 - 1) ^ 2 * GB) / 0.4e1 + E) * RH * C1 + (E
* C2 * RH ^ 2)) * RC + (16 * E * RU * V ^ 2 * C1 * RH ^ 2)) * (RB ^ 2) - 0.4e1 * ((RU * (V - 1) * (V + 1) * (((C2
- 1) ^ 2 * (V - 1) * (V + 1) * GB) / 0.4e1 + E) * C1 + (E * RH * (V ^ 2 - C2))) * (RC ^ 2) + (2 * E * V ^ 2 * RC *
RH ^ 2) - (4 * RU * C1 * E * RH ^ 2 * (V - 1) * (V + 1))) * RU * C1 * RB + (8 * E * RU ^ 2 * C1 ^ 2 * RC * RH ^
2 * (V - 1) * (V + 1) * (C2 - 1))) * (C3 ^ 2) - 0.8e1 * (GB * (V - 1) * (V + 1) * RH * (C2 - 1) * (-((V - 1) * (V + 1) *
(C2 + 1) * RC) / 0.2e1 + (RU * C1 * V ^ 2)) * (RB ^ 2) / 0.2e1 + GB * (V - 1) * RU * (V + 1) * C1 * ((C2 /
0.4e1 + 0.1e1 / 0.4e1) * (RC ^ 2) - (V ^ 2 * RC * RH) + (RU * C1 * RH * (V - 1) * (V + 1))) * (C2 - 1) * RB /
0.2e1 + RC * RU * C1 * RH * (RU * (V - 1) * (V + 1) * (((C2 - 1) ^ 2 * (V - 1) * (V + 1) * GB) / 0.4e1 + E) * C1
- (E * C2 * RH))) * RB * C3 - (2 * RU * C1 * GB * RB ^ 2 * RC * RH * (C2 - 1) * (C2 + 1) * (V - 1) * (V + 1)));

    % increment of elastic strain exceed yiled surface
    DEPSXXU0 = 0.4e1 * RB * (C3 + 1) * DEPSYY0T * ((VU * RH * (2 * RB + RC * (C2 - 1)) * RU * C1 * RC *
E * C3 ^ 3) + (-(RU * VU * C1 * GB * RC * (V - 1) * (V + 1) * (C2 - 1) * RB ^ 2) / 0.2e1 + (((V + 1) * VU * (V -
1) * (-((C2 - 1) ^ 2 * GB) / 0.4e1 + E) * (RC ^ 2) - (2 * V * RH * RC * E) + (4 * E * VU * RH ^ 2)) * RU * C1 -
(E * V * RC ^ 2 * RH * (VU - 1) * (VU + 1))) * RB - 0.2e1 * (V * ((V * VU) + C2 / 0.2e1 - 0.1e1 / 0.2e1) * RC -
(VU * RH * (C2 - 1))) * RH * C1 * RC * RU * E) * (C3 ^ 2) + 0.2e1 * (-GB * (V + 1) * (V - 1) * (C2 - 1) * ((VU *
RH) - (V * RC) / 0.2e1) * (RB ^ 2) / 0.2e1 + ((V * GB * (C2 - 1) ^ 2 * (V - 1) * (V + 1) * RC ^ 2) / 0.8e1 + (V +
1) * VU * (V - 1) * RH * (-((C2 - 1) ^ 2 * GB) / 0.4e1 + E) * RC - (2 * E * V * RH ^ 2)) * RB - (V * RH * (-RC +
RH * (C2 - 1)) * RC * E)) * RU * C1 * C3 + (V * GB * (V + 1) * (V - 1) * RH * RB * (C2 - 1) * (2 * RB + RC *
(C2 - 1)) * RU * C1) / 0.2e1) / (-(4 * RH * RB * (2 * RB + RC * (C2 - 1)) * RU * C1 * RC * E * C3 ^ 4) + ((2 *
RU * C1 * GB * RC * (V - 1) * (V + 1) * (C2 - 1) * RB ^ 3) + (0.8e1 * (-(V + 1) * (V - 1) * (-((C2 - 1) ^ 2 * GB) /
0.4e1 + E) * (RC ^ 2) / 0.2e1 + (E * VU * V * RC * RH) - (2 * E * RH ^ 2)) * RU * C1 - (4 * E * C2 * RC ^ 2 *
RH * (VU - 1) * (VU + 1))) * (RB ^ 2) + 0.8e1 * ((RU * (V ^ 2 - 1) * C1) - (V * (VU - V) * RC) / 0.2e1 - (RH *
(C2 - 1))) * RH * C1 * RC * RU * E * RB + (4 * E * RU ^ 2 * C1 ^ 2 * RC ^ 2 * RH * (V - 1) * (V + 1) * (C2 -
1))) * (C3 ^ 3) + (-0.2e1 * (V - 1) * GB * (C2 - 1) * (V + 1) * ((RU * (V * RC * VU - 2 * RH) * C1) - (RC ^ 2 *
(VU - 1) * (VU + 1) * (C2 + 1)) / 0.2e1) * (RB ^ 3) + (-(2 * RU ^ 2 * GB * RC * (V - 1) ^ 2 * (V + 1) ^ 2 * (C2 -
1) * C1 ^ 2) + 0.16e2 * ((V * VU * GB * (V - 1) * (V + 1) * (C2 - 1) * RC ^ 2) / 0.8e1 - (V + 1) * (V - 1) * RH * (-
((C2 - 1) ^ 2 * GB) / 0.4e1 + E) * RC / 0.2e1 + (E * VU * V * RH ^ 2)) * RU * C1 - (8 * E * C2 * RC * RH ^ 2 *
(VU - 1) * (VU + 1))) * (RB ^ 2) - 0.4e1 * ((V - 1) * (V + 1) * ((((C2 - 1) ^ 2 * (V - 1) * (V + 1) * GB) / 0.4e1 + E)
* (RC ^ 2) - (4 * E * RH ^ 2)) * RU * C1 + (((V * VU - C2) * RC + 2 * V * VU * RH) * RH * RC * E)) * C1 * RU
* RB + (8 * E * RU ^ 2 * C1 ^ 2 * RC * RH ^ 2 * (V - 1) * (V + 1) * (C2 - 1))) * (C3 ^ 2) - 0.8e1 * ((V - 1) * GB *
RH * ((RU * VU * V * C1) - (RC * (VU - 1) * (VU + 1) * (C2 + 1)) / 0.2e1) * (C2 - 1) * (V + 1) * (RB ^ 2) / 0.2e1
+ (V - 1) * GB * ((RU * RH * (V - 1) * (V + 1) * C1) - RC * ((-C2 / 0.4e1 - 0.1e1 / 0.4e1) * RC + (V * VU *
RH))) * (C2 - 1) * C1 * (V + 1) * RU * RB / 0.2e1 + (RU * (V - 1) * (V + 1) * (((C2 - 1) ^ 2 * (V - 1) * (V + 1) *
GB) / 0.4e1 + E) * C1 - (E * C2 * RH)) * RH * C1 * RC * RU) * RB * C3 - (2 * RU * C1 * GB * RB ^ 2 * RC *
RH * (C2 - 1) * (C2 + 1) * (V - 1) * (V + 1)));
    DEPSYYU0 = 0.4e1 * RB * DEPSYY0T * (-GB * (V + 1) * (V - 1) * ((RU * VU * V * C1) - (RU * C1 * C3) -
(RC * (VU - 1) * (VU + 1) * (C2 + 1)) / 0.2e1) * (C3 * RC + 2 * RH) * (C2 - 1) * RB ^ 2 / 0.2e1 + 0.2e1 * (C3 *
RC + 2 * RH) * (-(RU * C1 * E * C3 ^ 2 * RH) + ((-(V + 1) * (V - 1) * (-((C2 - 1) ^ 2) * GB / 0.4e1 + E) * RU *
C1 / 0.2e1 - (E * C2 * RH * (VU - 1) * (VU + 1)) / 0.2e1) * RC + (RU * C1 * E * VU * V * RH)) * C3 - RU * VU
* V * C1 * GB * RC * ((C2 - 1) ^ 2) * (V - 1) * (V + 1) / 0.8e1) * RB + (RH * (-RC * (C2 - 1) * C3 ^ 2 + (V * (2 *
V + VU * (C2 - 1)) * RC - 2 * RH * (C2 - 1)) * C3 + 2 * V * VU * (-RC + RH * (C2 - 1))) * C3 * RU * C1 * RC *
E)) * (C3 + 1) / (-0.2e1 * GB * (V + 1) * (V - 1) * ((RU * VU * V * C1) - (RU * C1 * C3) - (RC * (VU - 1) * (VU +
1) * (C2 + 1)) / 0.2e1) * (C3 * RC + 2 * RH) * (C2 - 1) * C3 * RB ^ 3 + 0.8e1 * (C3 * RC + 2 * RH) * (-(RU *
C1 * E * C3 ^ 3 * RH) + ((-(V + 1) * (V - 1) * (-((C2 - 1) ^ 2) * GB / 0.4e1 + E) * RU * C1 / 0.2e1 - (E * C2 *
RH * (VU - 1) * (VU + 1)) / 0.2e1) * RC + (RU * C1 * E * VU * V * RH)) * (C3 ^ 2) - GB * (V + 1) * (V - 1) *
(C2 - 1) * (-V * RC * VU + RU * (V - 1) * (V + 1) * C1) * RU * C1 * C3 / 0.4e1 - RU * C1 * RC * GB * (C2 - 1)
* (C2 + 1) * (V - 1) * (V + 1) / 0.8e1) * RB ^ 2 + 0.8e1 * C3 * RU * (-(E * RC ^ 2 * RH * (C2 - 1) * C3 ^ 3) /
0.2e1 + RH * ((RU * (V ^ 2 - 1) * C1) - (V * (VU - V) * RC) / 0.2e1 - (RH * (C2 - 1))) * RC * E * (C3 ^ 2) + ((-
RU * (V - 1) * (V + 1) * (((C2 - 1) ^ 2) * (V - 1) * (V + 1) * GB / 0.4e1 + E) * C1 / 0.2e1 - (E * RH * (V * VU -
C2)) / 0.2e1) * (RC ^ 2) - (E * RC * RH ^ 2 * V * VU) + (2 * RU * C1 * E * RH ^ 2 * (V - 1) * (V + 1))) * C3 -
RH * (RU * (V - 1) * (V + 1) * (((C2 - 1) ^ 2) * (V - 1) * (V + 1) * GB / 0.4e1 + E) * C1 - (E * C2 * RH)) * RC) *
C1 * RB + (4 * E * RU ^ 2 * C1 ^ 2 * C3 ^ 2 * RC * RH * (V - 1) * (V + 1) * (C3 * RC + 2 * RH) * (C2 - 1)));

    % caculated cohesion of brick unit by value of hardening(softening) parameter K
    DKU = sqrt((6 * DEPSPXXU ^ 2 + 6 * DEPSPYYU ^ 2)) / 0.3e1;
    KU  = KU + DKU;
    KUMAX = ((2 * LC * EPS0U * FCU + 3 * GCU) / LC / FCU) / 0.2e1;
    % find compressive strength by hardening parameter K:
    if KU <= EPS0U
        SIGCU = (FCU * (-2 * KU ^ 2 / EPS0U ^ 2 + 4 * KU / EPS0U + 1)) / 0.3e1;
        KCU = (FCU * (-4 * KU / EPS0U ^ 2 + 4 / EPS0U)) / 0.3e1;
    else
        if KU < KUMAX
            SIGCU = FCU * (0.1e1 - 0.4e1 / 0.9e1 * FCU ^ 2 * LC ^ 2 / GCU ^ 2 * (KU - EPS0U) ^ 2);
            KCU = -0.8e1 / 0.9e1 * FCU ^ 3 * LC ^ 2 / GCU ^ 2 * (KU - EPS0U);
        else
            SIGCU = 0;
```

```
         KCU   = -0.8e1 / 0.9e1 * FCU ^ 3 * LC ^ 2 / GCU ^ 2 * (KUMAX - EPS0U);
      end
   end
   % find critical stress:
   CU = (0.1e1 - sin(PHIU)) / cos(PHIU) * SIGCU / 0.2e1;

   SIGXXUC1 = -0.18e2 * (-0.6e1 * SIGXXUE * sin(PHIU) / (0.3e1 - sin(PHIU)) - 0.6e1 * SIGYYUE *
sin(PHIU) / (0.3e1 - sin(PHIU)) + 0.3e1 * sqrt(SIGXXUE ^ 2 - SIGXXUE * SIGYYUE + SIGYYUE ^ 2)) *
SIGXXUE * CU * cos(PHIU) / (0.3e1 - sin(PHIU)) / (0.36e2 * SIGXXUE ^ 2 * sin(PHIU) ^ 2 / (0.3e1 -
sin(PHIU)) ^ 2 + 0.72e2 * SIGXXUE * SIGYYUE * sin(PHIU) ^ 2 / (0.3e1 - sin(PHIU)) ^ 2 + 0.36e2 *
SIGYYUE ^ 2 * sin(PHIU) ^ 2 / (0.3e1 - sin(PHIU)) ^ 2 - 0.9e1 * SIGXXUE ^ 2 + 0.9e1 * SIGXXUE *
SIGYYUE - 0.9e1 * SIGYYUE ^ 2);

   SIGYYUC1 = 0.18e2 * (0.6e1 * SIGXXUE * sin(PHIU) / (0.3e1 - sin(PHIU)) + 0.6e1 * SIGYYUE *
sin(PHIU) / (0.3e1 - sin(PHIU)) + 0.3e1 * sqrt(SIGXXUE ^ 2 - SIGXXUE * SIGYYUE + SIGYYUE ^ 2)) *
SIGYYUE * CU * cos(PHIU) / (0.3e1 - sin(PHIU)) / (0.36e2 * SIGXXUE ^ 2 * sin(PHIU) ^ 2 / (0.3e1 -
sin(PHIU)) ^ 2 + 0.72e2 * SIGXXUE * SIGYYUE * sin(PHIU) ^ 2 / (0.3e1 - sin(PHIU)) ^ 2 + 0.36e2 *
SIGYYUE ^ 2 * sin(PHIU) ^ 2 / (0.3e1 - sin(PHIU)) ^ 2 - 0.9e1 * SIGXXUE ^ 2 + 0.9e1 * SIGXXUE *
SIGYYUE - 0.9e1 * SIGYYUE ^ 2);

   SIGXXUC2 = 0.18e2 * (0.6e1 * SIGXXUE * sin(PHIU) / (0.3e1 - sin(PHIU)) + 0.6e1 * SIGYYUE *
sin(PHIU) / (0.3e1 - sin(PHIU)) + 0.3e1 * sqrt(SIGXXUE ^ 2 - SIGXXUE * SIGYYUE + SIGYYUE ^ 2)) *
SIGXXUE * CU * cos(PHIU) / (0.3e1 - sin(PHIU)) / (0.36e2 * SIGXXUE ^ 2 * sin(PHIU) ^ 2 / (0.3e1 -
sin(PHIU)) ^ 2 + 0.72e2 * SIGXXUE * SIGYYUE * sin(PHIU) ^ 2 / (0.3e1 - sin(PHIU)) ^ 2 + 0.36e2 *
SIGYYUE ^ 2 * sin(PHIU) ^ 2 / (0.3e1 - sin(PHIU)) ^ 2 - 0.9e1 * SIGXXUE ^ 2 + 0.9e1 * SIGXXUE *
SIGYYUE - 0.9e1 * SIGYYUE ^ 2);

   SIGYYUC2 = -0.18e2 * (-0.6e1 * SIGXXUE * sin(PHIU) / (0.3e1 - sin(PHIU)) - 0.6e1 * SIGYYUE *
sin(PHIU) / (0.3e1 - sin(PHIU)) + 0.3e1 * sqrt(SIGXXUE ^ 2 - SIGXXUE * SIGYYUE + SIGYYUE ^ 2)) *
SIGYYUE * CU * cos(PHIU) / (0.3e1 - sin(PHIU)) / (0.36e2 * SIGXXUE ^ 2 * sin(PHIU) ^ 2 / (0.3e1 -
sin(PHIU)) ^ 2 + 0.72e2 * SIGXXUE * SIGYYUE * sin(PHIU) ^ 2 / (0.3e1 - sin(PHIU)) ^ 2 + 0.36e2 *
SIGYYUE ^ 2 * sin(PHIU) ^ 2 / (0.3e1 - sin(PHIU)) ^ 2 - 0.9e1 * SIGXXUE ^ 2 + 0.9e1 * SIGXXUE *
SIGYYUE - 0.9e1 * SIGYYUE ^ 2);

   if SIGXXUE/SIGXXUC1 > 0
      SIGXXUC = SIGXXUC1;
   else
      SIGXXUC = SIGXXUC2;
   end
   if SIGYYUE/SIGYYUC1 > 0
      SIGYYUC = SIGYYUC1;
   else
      SIGYYUC = SIGYYUC2;
   end

   % drucker pragar plasiticy model of head joint
   % elastic predictor of head joint:
   SIGXXHE = -0.4e1 * (C3 + 1) * E * RB * ((((2 * RU * C1 - RC * (C2 + 1)) * RB + RU * C1 * RC * (C2 - 1)) *
RC * E * RU * C1 * C3 ^ 2) + (-GB * ((RU * (V ^ 2 * RC - RC + 2 * RH) * C1) - ((V - 1) * (V + 1) * (C2 + 1) *
RC ^ 2) / 0.2e1) * (C2 - 1) * (RB ^ 2) / 0.2e1 + 0.4e1 * RU * C1 * (RU * (-GB * (V - 1) * (V + 1) * (C2 - 1) *
RC / 0.4e1 + (E * RH)) * C1 - (-GB * ((C2 + 3) * V ^ 2 + C2 - 1) * (C2 - 1) * RC / 0.8e1 + (((C2 - 1) ^ 2) * GB /
0.4e1 + (E * C2)) * RH) * RC / 0.2e1) * RB + 0.2e1 * RC * (RU ^ 2) * (C1 ^ 2) * ((-((C2 - 1) ^ 2) * (V - 1) * (V
+ 1) * GB / 0.4e1 - E) * RC + (E * RH * (C2 - 1)))) * C3 - GB * RU * C1 * RB * (-2 * RB * RH + RC * (RC *
(C2 + 1) - RH * (C2 - 1))) * (C2 - 1) / 0.2e1) * V * EPSYY0T / (-(4 * RC * E * RU * C1 * RB * RH * (2 * RB +
RC * (C2 - 1)) * C3 ^ 4) + (0.2e1 * RU * C1 * GB * RC * (V - 1) * (V + 1) * (C2 - 1) * (RB ^ 3) + (-0.4e1 * RU *
((V - 1) * (V + 1) * (-((C2 - 1) ^ 2) * GB / 0.4e1 + E) * (RC ^ 2) - (2 * E * V ^ 2 * RC * RH) + (4 * E * RH ^ 2)) *
```

```
C1 - (4 * E * C2 * RC ^ 2 * RH * (V - 1) * (V + 1))) * (RB ^ 2) + (8 * RC * E * RU * C1 * RH * (RU * (V ^ 2 - 1)
* C1 - RH * (C2 - 1)) * RB) + (4 * E * RU ^ 2 * C1 ^ 2 * RC ^ 2 * RH * (V - 1) * (V + 1) * (C2 - 1))) * (C3 ^ 3) +
(-0.2e1 * GB * (V - 1) * (V + 1) * ((RU * (V ^ 2 * RC - 2 * RH) * C1) - ((V - 1) * (V + 1) * (C2 + 1) * RC ^ 2) /
0.2e1) * (C2 - 1) * (RB ^ 3) + (-0.2e1 * (RU ^ 2) * GB * RC * ((V - 1) ^ 2) * ((V + 1) ^ 2) * (C2 - 1) * (C1 ^ 2) -
0.8e1 * (-(V ^ 2) * GB * (V - 1) * (V + 1) * (C2 - 1) * (RC ^ 2) / 0.4e1 + (V - 1) * (V + 1) * (-((C2 - 1) ^ 2) * GB /
0.4e1 + E) * RH * RC - (2 * E * V ^ 2 * RH ^ 2)) * RU * C1 - (8 * RC * RH ^ 2 * E * C2 * (V - 1) * (V + 1))) *
(RB ^ 2) - 0.4e1 * ((V - 1) * RU * (V + 1) * ((((C2 - 1) ^ 2) * (V - 1) * (V + 1) * GB / 0.4e1 + E) * (RC ^ 2) - (4 *
E * RH ^ 2)) * C1 + (RC * E * ((V ^ 2 - C2) * RC + 2 * V ^ 2 * RH) * RH)) * RU * C1 * RB + (8 * E * RU ^ 2 *
C1 ^ 2 * RC * RH ^ 2 * (V - 1) * (V + 1) * (C2 - 1))) * (C3 ^ 2) - 0.8e1 * (GB * (V - 1) * (V + 1) * RH * (C2 - 1) *
(-((V - 1) * (V + 1) * (C2 + 1) * RC) / 0.2e1 + (RU * C1 * V ^ 2)) * (RB ^ 2) / 0.2e1 + GB * (V - 1) * RU * (V +
1) * C1 * ((C2 / 0.4e1 + 0.1e1 / 0.4e1) * (RC ^ 2) - (V ^ 2 * RC * RH) + (RU * C1 * RH * (V - 1) * (V + 1))) *
(C2 - 1) * RB / 0.2e1 + RC * RU * C1 * RH * (RU * (V - 1) * (V + 1) * (((C2 - 1) ^ 2) * (V - 1) * (V + 1) * GB /
0.4e1 + E) * C1 - (E * C2 * RH))) * RB * C3 - 0.2e1 * RU * C1 * GB * (RB ^ 2) * RC * RH * (C2 - 1) * (C2 + 1)
* (V - 1) * (V + 1));

    SIGYYHE = -0.4e1 * (C3 + 1) * E * ((RC * E * RU * C1 * RB * RH * (2 * RB + RC * (C2 - 1)) * C3 ^ 3) +
RC * (-(RU * C1 * GB * (V - 1) * (V + 1) * (C2 - 1) * RB ^ 3) / 0.2e1 + ((2 * C1 ^ 2 * E * RU ^ 2 * V ^ 2) - RU *
((((C2 - 1) ^ 2 * (V - 1) * (V + 1) * GB) / 0.4e1 + (E * (C2 * V ^ 2 + 1))) * RC + (2 * E * RH * V ^ 2)) * C1 + (E *
C2 * RC * RH * (V - 1) * (V + 1))) * (RB ^ 2) + (E * RU * C1 * ((V ^ 2 * (C2 - 1) * RC - 4 * V ^ 2 * RH + 4 *
RH) * RU * C1 + V ^ 2 * RC * RH * (C2 + 1)) * RB) - (2 * E * RU ^ 2 * C1 ^ 2 * RC * RH * (V - 1) * (V + 1) *
(C2 - 1))) * (C3 ^ 2) + 0.2e1 * (-GB * (-((V - 1) * (V + 1) * (C2 + 1) * RC ^ 2) / 0.4e1 + (C1 * RH * RU * V ^ 2))
* (C2 - 1) * (RB ^ 2) / 0.2e1 + 0.2e1 * (RU * (-(GB * (V - 1) * (V + 1) * (C2 - 1) * RC) / 0.4e1 + (E * RH * V ^
2)) * C1 - RC * (GB * (0.1e1 / 0.4e1 - (C2 ^ 2) / 0.4e1) * RC + (((C2 - 1) ^ 2 * GB) / 0.4e1 + (E * C2)) * RH) *
(V ^ 2) / 0.2e1) * RU * C1 * RB + RC * RU * C1 * (((-((C2 - 1) ^ 2 * (V - 1) * (V + 1) * GB) / 0.4e1 - E) * RC +
(E * V ^ 2 * RH * (C2 - 1))) * RU * C1 - (E * C2 * RC * RH))) * RB * C3 + (GB * RU * C1 * RB ^ 2 * (2 * V ^ 2
* RB * RH + RC * ((-C2 - 1) * RC + V ^ 2 * RH * (C2 - 1))) * (C2 - 1)) / 0.2e1) * EPSYY0T / (-(4 * RC * E *
RU * C1 * RB * RH * (2 * RB + RC * (C2 - 1)) * C3 ^ 4) + ((2 * RU * C1 * GB * RC * (V - 1) * (V + 1) * (C2 -
1) * RB ^ 3) + (-0.4e1 * RU * ((V - 1) * (V + 1) * (-((C2 - 1) ^ 2 * GB) / 0.4e1 + E) * (RC ^ 2) - (2 * E * V ^ 2 *
RC * RH) + (4 * E * RH ^ 2)) * C1 - (4 * E * C2 * RC ^ 2 * RH * (V - 1) * (V + 1))) * (RB ^ 2) + (8 * RC * E *
RU * C1 * RH * (RU * (V ^ 2 - 1) * C1 - RH * (C2 - 1)) * RB) + (4 * E * RU ^ 2 * C1 ^ 2 * RC ^ 2 * RH * (V - 1)
* (V + 1) * (C2 - 1))) * (C3 ^ 3) + (-0.2e1 * GB * (V - 1) * (V + 1) * ((RU * (V ^ 2 * RC - 2 * RH) * C1) - ((V - 1)
* (V + 1) * (C2 + 1) * RC ^ 2) / 0.2e1) * (C2 - 1) * (RB ^ 3) + (-(2 * RU ^ 2 * GB * RC * (V - 1) ^ 2 * (V + 1) ^ 2
* (C2 - 1) * C1 ^ 2) - 0.8e1 * (-(V ^ 2 * GB * (V - 1) * (V + 1) * (C2 - 1) * RC ^ 2) / 0.4e1 + (V - 1) * (V + 1) * (-
((C2 - 1) ^ 2 * GB) / 0.4e1 + E) * RH * RC - (2 * E * V ^ 2 * RH ^ 2)) * RU * C1 - (8 * RC * RH ^ 2 * E * C2 *
(V - 1) * (V + 1))) * (RB ^ 2) - 0.4e1 * ((V - 1) * RU * (V + 1) * ((((C2 - 1) ^ 2 * (V - 1) * (V + 1) * GB) / 0.4e1 +
E) * (RC ^ 2) - (4 * E * RH ^ 2)) * C1 + (RC * E * ((V ^ 2 - C2) * RC + 2 * V ^ 2 * RH) * RH)) * RU * C1 * RB +
(8 * E * RU ^ 2 * C1 ^ 2 * RC * RH ^ 2 * (V - 1) * (V + 1) * (C2 - 1))) * (C3 ^ 2) - 0.8e1 * (GB * (V - 1) * (V + 1)
* RH * (C2 - 1) * (-((V - 1) * (V + 1) * (C2 + 1) * RC) / 0.2e1 + (RU * C1 * V ^ 2)) * (RB ^ 2) / 0.2e1 + GB * (V
- 1) * RU * (V + 1) * C1 * ((C2 / 0.4e1 + 0.1e1 / 0.4e1) * (RC ^ 2) - (V ^ 2 * RC * RH) + (RU * C1 * RH * (V -
1) * (V + 1))) * (C2 - 1) * RB / 0.2e1 + RC * RU * C1 * RH * (RU * (V - 1) * (V + 1) * (((C2 - 1) ^ 2 * (V - 1) *
(V + 1) * GB) / 0.4e1 + E) * C1 - (E * C2 * RH))) * RB * C3 - (2 * RU * C1 * GB * RB ^ 2 * RC * RH * (C2 - 1)
* (C2 + 1) * (V - 1) * (V + 1)));

    % increment of elastic strain exceed yiled surface
    DEPSXXH0 = 0.4e1 * (-(RU * C1 * GB * (V - 1) * (V + 1) * (C3 * RC + 2 * RH) * (C2 - 1) * (C3 * VU - V) *
RB ^ 3) / 0.2e1 + ((2 * C1 * C3 ^ 3 * E * RC * RH * RU * V) + 0.2e1 * ((E * RU ^ 2 * V * (V - 1) * (V + 1) * C1
^ 2) - ((((C2 - 1) ^ 2 * GB) / 0.4e1 + (E * C2)) * (V + 1) * (V - 1) * RC + (2 * E * V ^ 2 * RH)) * VU * RU * C1 /
0.2e1 + (E * V * C2 * RC * RH * (VU - 1) * (VU + 1)) / 0.2e1) * RC * (C3 ^ 2) + 0.4e1 * (V + 1) * (V - 1) * ((C1
* E * RH * RU * V) - (-(V * GB * (C2 - 1) ^ 2 * RC) / 0.8e1 + (((C2 - 1) ^ 2 * GB) / 0.4e1 + (E * C2)) * VU *
RH) * RC / 0.2e1) * RU * C1 * C3 + (RU * V * C1 * GB * RC * RH * (C2 - 1) ^ 2 * (V - 1) * (V + 1)) / 0.2e1) *
(RB ^ 2) + ((RH * RC * (C2 - 1) * C3 ^ 2 + ((V + 1) * (V - 1) * (RC * (C2 - 1) - 4 * RH) * RU * C1 + VU * V *
RC * RH * (C2 + 1)) * C3 + 2 * RH * (RU * (V - 1) * (V + 1) * (C2 - 1) * C1 - RC * C2)) * V * C3 * RU * C1 *
RC * E * RB) - (2 * E * RU ^ 2 * V * C1 ^ 2 * C3 ^ 2 * RC ^ 2 * RH * (V - 1) * (V + 1) * (C2 - 1))) * DEPSYY0T
* (C3 + 1) / (-0.2e1 * GB * (V + 1) * (V - 1) * ((RU * VU * V * C1) - (RU * C1 * C3) - (RC * (VU - 1) * (VU + 1)
* (C2 + 1)) / 0.2e1) * (C3 * RC + 2 * RH) * (C2 - 1) * C3 * (RB ^ 3) + 0.8e1 * (C3 * RC + 2 * RH) * (-(E * RU *
C1 * C3 ^ 3 * RH) + ((-(V + 1) * (V - 1) * (-((C2 - 1) ^ 2 * GB) / 0.4e1 + E) * RC / 0.2e1 + (E * VU * V * RH)) *
RU * C1 - (E * C2 * RC * RH * (VU - 1) * (VU + 1)) / 0.2e1) * (C3 ^ 2) - (GB * (V + 1) * (V - 1) * (C2 - 1) * (RU
* (V ^ 2 - 1) * C1 - VU * V * RC) * RU * C1 * C3) / 0.4e1 - (RU * C1 * RC * GB * (C2 - 1) * (C2 + 1) * (V - 1) *
(V + 1)) / 0.8e1) * (RB ^ 2) + 0.8e1 * C3 * RU * (-(E * RC ^ 2 * RH * (C2 - 1) * C3 ^ 3) / 0.2e1 + RH * ((RU *
(V ^ 2 - 1) * C1) - (V * (VU - V) * RC) / 0.2e1 - (RH * (C2 - 1))) * RC * E * (C3 ^ 2) + (-(V + 1) * (V - 1) * ((((C2
- 1) ^ 2 * GB * V ^ 2) / 0.4e1 - ((C2 - 1) ^ 2 * GB) / 0.4e1 + E) * (RC ^ 2) - (4 * E * RH ^ 2)) * RU * C1 / 0.2e1
- (((V * VU - C2) * RC + 2 * VU * V * RH) * RH * RC * E) / 0.2e1) * C3 - RH * (RU * (V - 1) * (V + 1) * (((C2 -
1) ^ 2 * GB * V ^ 2) / 0.4e1 - ((C2 - 1) ^ 2 * GB) / 0.4e1 + E) * C1 - (E * C2 * RH)) * RC) * C1 * RB + (4 * E *
RU ^ 2 * C1 ^ 2 * C3 ^ 2 * RC * RH * (V - 1) * (V + 1) * (C3 * RC + 2 * RH) * (C2 - 1)));

    DEPSYYH0 = 0.8e1 * (-(RH * RB * (2 * RB + RC * (C2 - 1)) * RU * C1 * E * C3 ^ 3) / 0.2e1 + ((RU * C1 *
GB * (V - 1) * (V + 1) * (C2 - 1) * RB ^ 3) / 0.4e1 + ((-(V + 1) * (V - 1) * (-((C2 - 1) ^ 2 * GB) / 0.4e1 + E) * RC
/ 0.2e1 + (E * VU * V * RH)) * RU * C1 - (E * C2 * RC * RH * (VU - 1) * (VU + 1)) / 0.2e1) * (RB ^ 2) + 0.2e1
* RH * ((RU * (V ^ 2 - 1) * C1) - (VU * V * (C2 + 1) * RC) / 0.4e1) * RU * C1 * E * RB + (RH * RU ^ 2 * C1 ^ 2
* RC * E * (V - 1) * (V + 1) * (C2 - 1))) * (C3 ^ 2) - RB * (GB * (V + 1) * (V - 1) * ((RU * VU * V * C1) - (RC *
(VU - 1) * (VU + 1) * (C2 + 1)) / 0.2e1) * (C2 - 1) * (RB ^ 2) / 0.4e1 + GB * ((RU * (V ^ 2 - 1) * C1) - (VU * V *
RC * (C2 + 3)) / 0.4e1) * (V + 1) * (V - 1) * (C2 - 1) * RU * C1 * RB / 0.2e1 + ((V + 1) * (V - 1) * RU * (((C2 -
1) ^ 2 * (V - 1) * (V + 1) * GB) / 0.4e1 + E) * C1 - (E * C2 * RH)) * RU * C1 * RC) * C3 - (RU * C1 * GB * RB ^
2 * RC * (C2 - 1) * (C2 + 1) * (V - 1) * (V + 1)) / 0.4e1) * DEPSYY0T * RC * (C3 + 1) / (-(4 * RH * RB * (2 *
RB + RC * (C2 - 1)) * RU * C1 * RC * E * C3 ^ 4) + ((2 * RU * C1 * GB * RC * (V - 1) * (V + 1) * (C2 - 1) * RB
^ 3) + (0.8e1 * RU * (-(V + 1) * (V - 1) * (-((C2 - 1) ^ 2 * GB) / 0.4e1 + E) * (RC ^ 2) / 0.2e1 + (E * VU * V *
RC * RH) - (2 * E * RH ^ 2)) * C1 - (4 * E * C2 * RC ^ 2 * RH * (VU - 1) * (VU + 1))) * (RB ^ 2) + 0.8e1 * RH *
((RU * (V ^ 2 - 1) * C1) - (V * (VU - V) * RC) / 0.2e1 - (RH * (C2 - 1))) * RU * C1 * RC * E * RB + (4 * E * RU
^ 2 * C1 ^ 2 * RC ^ 2 * RH * (V - 1) * (V + 1) * (C2 - 1))) * (C3 ^ 3) + (-0.2e1 * GB * (V + 1) * (V - 1) * (C2 - 1)
* ((RU * (VU * V * RC - 2 * RH) * C1) - (RC ^ 2 * (VU - 1) * (VU + 1) * (C2 + 1)) / 0.2e1) * (RB ^ 3) + (-(2 *
RU ^ 2 * GB * RC * (V - 1) ^ 2 * (V + 1) ^ 2 * (C2 - 1) * C1 ^ 2) + 0.16e2 * ((VU * V * GB * (V - 1) * (V + 1) *
(C2 - 1) * RC ^ 2) / 0.8e1 - (V + 1) * (V - 1) * RH * (-((C2 - 1) ^ 2 * GB) / 0.4e1 + E) * RC / 0.2e1 + (E * VU *
V * RH ^ 2)) * RU * C1 - (8 * E * C2 * RC * RH ^ 2 * (VU - 1) * (VU + 1))) * (RB ^ 2) - 0.4e1 * ((V + 1) * (V - 1)
* ((((C2 - 1) ^ 2 * (V - 1) * (V + 1) * GB) / 0.4e1 + E) * (RC ^ 2) - (4 * E * RH ^ 2)) * RU * C1 + (((V * VU - C2)
* RC + 2 * VU * V * RH) * RH * RC * E)) * RU * C1 * RB + (8 * E * RU ^ 2 * C1 ^ 2 * RC * RH ^ 2 * (V - 1) *
(V + 1) * (C2 - 1))) * (C3 ^ 2) - 0.8e1 * RB * (GB * (V + 1) * (V - 1) * ((RU * VU * V * C1) - (RC * (VU - 1) *
(VU + 1) * (C2 + 1)) / 0.2e1) * RH * (C2 - 1) * (RB ^ 2) / 0.2e1 + RU * C1 * GB * (V - 1) * (V + 1) * (C2 - 1) *
((RU * RH * (V - 1) * (V + 1) * C1) - ((-C2 / 0.4e1 - 0.1e1 / 0.4e1) * RC + (VU * V * RH)) * RC) * RB / 0.2e1 +
((V + 1) * (V - 1) * RU * (((C2 - 1) ^ 2 * (V - 1) * (V + 1) * GB) / 0.4e1 + E) * C1 - (E * C2 * RH)) * RH * RU *
C1 * RC) * C3 - (2 * RU * C1 * GB * RB ^ 2 * RC * RH * (C2 - 1) * (C2 + 1) * (V - 1) * (V + 1)));

    % caculated cohesion of brick unit by value of hardening(softening) parameter K
    DKH = sqrt((6 * DEPSPXXH ^ 2 + 6 * DEPSPYYH ^ 2)) / 0.3e1;
    KH  = KH + DKH;
    KHMAX = ((2 * LC * EPS0M * FCM + 3 * GCM) / LC / FCM) / 0.2e1;
    % find compressive strength by hardening parameter K:
    if KH <= EPS0M
        SIGCH = (FCM * (-2 * KH ^ 2 / EPS0M ^ 2 + 4 * KH / EPS0M + 1)) / 0.3e1;
        KCH = (FCM * (-4 * KH / EPS0M ^ 2 + 4 / EPS0M)) / 0.3e1;
    else
        if KH < KHMAX
            SIGCH = FCM * (0.1e1 - 0.4e1 / 0.9e1 * FCM ^ 2 * LC ^ 2 / GCM ^ 2 * (KH - EPS0M) ^ 2);
            KCH = -0.8e1 / 0.9e1 * FCM ^ 3 * LC ^ 2 / GCM ^ 2 * (KH - EPS0M);
        else
            SIGCH = 0;
            KCH = -0.8e1 / 0.9e1 * FCM ^ 3 * LC ^ 2 / GCM ^ 2 * (KHMAX - EPS0M);
        end
    end
    % find critical stress:
    CH = (0.1e1 - sin(PHIM)) / cos(PHIM) * SIGCH / 0.2e1;
    SIGXXHC1 = -0.18e2 * (-0.6e1 * SIGXXHE * sin(PHIM) / (0.3e1 - sin(PHIM)) - 0.6e1 * SIGYYHE *
sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.3e1 * sqrt(SIGXXHE ^ 2 - SIGXXHE * SIGYYHE + SIGYYHE ^ 2)) *
SIGXXHE * CH * cos(PHIM) / (0.3e1 - sin(PHIM)) / (0.36e2 * SIGXXHE ^ 2 * sin(PHIM) ^ 2 / (0.3e1 -
sin(PHIM)) ^ 2 + 0.72e2 * SIGXXHE * SIGYYHE * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 + 0.36e2 *
SIGYYHE ^ 2 * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 - 0.9e1 * SIGXXHE ^ 2 + 0.9e1 * SIGXXHE *
SIGYYHE - 0.9e1 * SIGYYHE ^ 2);

   SIGYYHC1 = 0.18e2 * (0.6e1 * SIGXXHE * sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.6e1 * SIGYYHE *
sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.3e1 * sqrt(SIGXXHE ^ 2 - SIGXXHE * SIGYYHE + SIGYYHE ^ 2)) *
SIGYYHE * CH * cos(PHIM) / (0.3e1 - sin(PHIM)) / (0.36e2 * SIGXXHE ^ 2 * sin(PHIM) ^ 2 / (0.3e1 -
sin(PHIM)) ^ 2 + 0.72e2 * SIGXXHE * SIGYYHE * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 + 0.36e2 *
SIGYYHE ^ 2 * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 - 0.9e1 * SIGXXHE ^ 2 + 0.9e1 * SIGXXHE *
SIGYYHE - 0.9e1 * SIGYYHE ^ 2);

   SIGXXHC2 = 0.18e2 * (0.6e1 * SIGXXHE * sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.6e1 * SIGYYHE *
sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.3e1 * sqrt(SIGXXHE ^ 2 - SIGXXHE * SIGYYHE + SIGYYHE ^ 2)) *
SIGXXHE * CH * cos(PHIM) / (0.3e1 - sin(PHIM)) / (0.36e2 * SIGXXHE ^ 2 * sin(PHIM) ^ 2 / (0.3e1 -
sin(PHIM)) ^ 2 + 0.72e2 * SIGXXHE * SIGYYHE * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 + 0.36e2 *
SIGYYHE ^ 2 * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 - 0.9e1 * SIGXXHE ^ 2 + 0.9e1 * SIGXXHE *
SIGYYHE - 0.9e1 * SIGYYHE ^ 2);

   SIGYYHC2 = -0.18e2 * (-0.6e1 * SIGXXHE * sin(PHIM) / (0.3e1 - sin(PHIM)) - 0.6e1 * SIGYYHE *
sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.3e1 * sqrt(SIGXXHE ^ 2 - SIGXXHE * SIGYYHE + SIGYYHE ^ 2)) *
SIGYYHE * CH * cos(PHIM) / (0.3e1 - sin(PHIM)) / (0.36e2 * SIGXXHE ^ 2 * sin(PHIM) ^ 2 / (0.3e1 -
sin(PHIM)) ^ 2 + 0.72e2 * SIGXXHE * SIGYYHE * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 + 0.36e2 *
SIGYYHE ^ 2 * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 - 0.9e1 * SIGXXHE ^ 2 + 0.9e1 * SIGXXHE *
SIGYYHE - 0.9e1 * SIGYYHE ^ 2);

   if SIGXXHE/SIGXXHC1 > 0
      SIGXXHC = SIGXXHC1;
   else
      SIGXXHC = SIGXXHC2;
   end
   if SIGYYHE/SIGYYHC1 > 0
      SIGYYHC = SIGYYHC1;
   else
      SIGYYHC = SIGYYHC2;
   end

   % drucker pragar plasiticy model of bed joint
   % elastic predictor of bed joint:
   SIGXXBE = -0.4e1 * (C3 + 1) * (-GB * (V - 1) * (V + 1) * ((-C2 / 0.2e1 - 0.1e1 / 0.2e1) * RC + RU * C1) *
(C3 * RC + (2 * RH)) * (C2 - 0.1e1) * RB ^ 2 / 0.2e1 + (0.2e1 * RC * E * RH * ((-C2 / 0.2e1 - 0.1e1 / 0.2e1) *
RC + RU * C1) * (C3 ^ 2) + (-RU * C1 * GB * (C2 - 0.1e1) ^ 2 * (V - 1) * (V + 1) * RC ^ 2 / 0.4e1 - 0.2e1 * RC
* (RH ^ 2) * E * C2 + 0.4e1 * RU * C1 * E * (RH ^ 2)) * C3 - RU * C1 * GB * RC * RH * (C2 - 0.1e1) ^ 2 * (V -
1) * (V + 1) / 0.2e1) * RB + RC * E * RU * C1 * C3 * (RC * (C2 - 0.1e1) * C3 - 0.2e1 * RC + (0.2e1 * C2 -
0.2e1) * RH) * RH) * E * RU * C1 * C3 * V * EPSYY0T / (-0.2e1 * GB * (V - 1) * (V + 1) * C3 * (-RU * C1 * C3
- (V - 1) * (V + 1) * (C2 + 0.1e1) * RC / 0.2e1 + RU * C1 * (V ^ 2)) * (C3 * RC + (2 * RH)) * (C2 - 0.1e1) * RB
^ 3 - 0.4e1 * (0.2e1 * RU * C1 * E * (C3 ^ 3) * RH + ((V - 1) * (RU * (-(C2 - 0.1e1) ^ 2 * GB / 0.4e1 + E) * C1
+ E * C2 * RH) * (V + 1) * RC - 0.2e1 * RU * C1 * E * RH * (V ^ 2)) * (C3 ^ 2) + GB * (V - 1) * RU * (V + 1) *
C1 * (-(V ^ 2) * RC + RU * (V - 1) * (V + 1) * C1) * (C2 - 0.1e1) * C3 / 0.2e1 + RU * C1 * GB * RC * (C2 -
0.1e1) * (C2 + 0.1e1) * (V - 1) * (V + 1) / 0.4e1) * (C3 * RC + (2 * RH)) * RB ^ 2 + 0.8e1 * RU * C1 * C3 * (-E
* RC ^ 2 * RH * (C2 - 0.1e1) * (C3 ^ 3) / 0.2e1 + RC * E * RH * (RU * (V ^ 2 - 1) * C1 - RH * (C2 - 0.1e1)) *
(C3 ^ 2) + ((-RU * (V - 1) * (V + 1) * ((C2 - 0.1e1) ^ 2 * (V - 1) * (V + 1) * GB / 0.4e1 + E) * C1 / 0.2e1 - E *
RH * ((V ^ 2) - C2) / 0.2e1) * RC ^ 2 - E * (V ^ 2) * RC * (RH ^ 2) + 0.2e1 * RU * C1 * E * (RH ^ 2) * (V - 1) *
(V + 1)) * C3 - RC * RH * (RU * (V - 1) * (V + 1) * ((C2 - 0.1e1) ^ 2 * (V - 1) * (V + 1) * GB / 0.4e1 + E) * C1 -
E * C2 * RH)) * RB + 0.4e1 * E * RU ^ 2 * C1 ^ 2 * (C3 ^ 2) * RC * RH * (V - 1) * (V + 1) * (C3 * RC + (2 *
RH)) * (C2 - 0.1e1));

   SIGYYBE = -0.4e1 * (C3 + 1) * E * RU * ((RC * E * RU * C1 * RH * (2 * RB + RC * (C2 - 1)) * C3 ^ 3) + (-
(RU * C1 * GB * RC * (V - 1) * (V + 1) * (C2 - 1) * RB ^ 2) / 0.2e1 + (((V - 1) * RU * (V + 1) * (-((C2 - 1) ^ 2 *
GB) / 0.4e1 + E) * C1 - (E * RH * (V ^ 2 + C2))) * (RC ^ 2) + (4 * RU * C1 * E * RH ^ 2)) * RB - (2 * RC * E *
RU * (V ^ 2 * RC - RH * (C2 - 1)) * C1 * RH)) * (C3 ^ 2) + 0.2e1 * (-GB * (V - 1) * ((-C2 / 0.4e1 - 0.1e1 /
0.4e1) * (RC ^ 2) + (RU * C1 * RH)) * (V + 1) * (C2 - 1) * RB / 0.2e1 + RC * ((V - 1) * RU * (V + 1) * (-((C2 -
1) ^ 2 * GB) / 0.4e1 + E) * C1 - (E * C2 * RH)) * RH) * RB * C3 + (GB * RB ^ 2 * RC * RH * (C2 - 1) * (C2 +
1) * (V - 1) * (V + 1)) / 0.2e1) * C1 * EPSYY0T / (-(4 * RC * E * RU * C1 * RB * RH * (2 * RB + RC * (C2 - 1))
* C3 ^ 4) + ((2 * RU * C1 * GB * RC * (V - 1) * (V + 1) * (C2 - 1) * RB ^ 3) + (-0.4e1 * (V - 1) * (RU * (-((C2 -
1) ^ 2 * GB) / 0.4e1 + E) * C1 + (E * C2 * RH)) * (V + 1) * (RC ^ 2) + (8 * E * RU * V ^ 2 * C1 * RC * RH) -
(16 * RU * C1 * E * RH ^ 2)) * (RB ^ 2) + (8 * RC * E * RU * C1 * RH * (RU * (V ^ 2 - 1) * C1 - RH * (C2 - 1))
* RB) + (4 * E * RU ^ 2 * C1 ^ 2 * RC ^ 2 * RH * (V - 1) * (V + 1) * (C2 - 1))) * (C3 ^ 3) + (-0.2e1 * GB * (V -
1) * (V + 1) * (-((V - 1) * (V + 1) * (C2 + 1) * RC ^ 2) / 0.2e1 + (C1 * RC * RU * V ^ 2) - (2 * RU * C1 * RH)) *
(C2 - 1) * (RB ^ 3) + ((2 * RU * C1 * V ^ 2 * GB * (V - 1) * (V + 1) * (C2 - 1) * RC ^ 2) - 0.8e1 * (V - 1) * (V +
1) * ((RU ^ 2 * GB * (V - 1) * (V + 1) * (C2 - 1) * C1 ^ 2) / 0.4e1 + RU * (-((C2 - 1) ^ 2 * GB) / 0.4e1 + E) * RH
* C1 + (E * C2 * RH ^ 2)) * RC + (16 * E * RU * V ^ 2 * C1 * RH ^ 2)) * (RB ^ 2) - 0.4e1 * ((RU * (V - 1) * (V +
1) * (((C2 - 1) ^ 2 * (V - 1) * (V + 1) * GB) / 0.4e1 + E) * C1 + (E * RH * (V ^ 2 - C2))) * (RC ^ 2) + (2 * E * V ^
2 * RC * RH ^ 2) - (4 * RU * C1 * E * RH ^ 2 * (V - 1) * (V + 1))) * RU * C1 * RB + (8 * E * RU ^ 2 * C1 ^ 2 *
RC * RH ^ 2 * (V - 1) * (V + 1) * (C2 - 1))) * (C3 ^ 2) - 0.8e1 * (GB * (V - 1) * (V + 1) * RH * (C2 - 1) * (-((V -
1) * (V + 1) * (C2 + 1) * RC) / 0.2e1 + (RU * C1 * V ^ 2)) * (RB ^ 2) / 0.2e1 + GB * (V - 1) * RU * (V + 1) * C1
* ((C2 / 0.4e1 + 0.1e1 / 0.4e1) * (RC ^ 2) - (V ^ 2 * RC * RH) + (RU * C1 * RH * (V - 1) * (V + 1))) * (C2 - 1) *
RB / 0.2e1 + RC * RU * C1 * RH * (RU * (V - 1) * (V + 1) * (((C2 - 1) ^ 2 * (V - 1) * (V + 1) * GB) / 0.4e1 + E)
* C1 - (E * C2 * RH))) * RB * C3 - (2 * RU * C1 * GB * RB ^ 2 * RC * RH * (C2 - 1) * (C2 + 1) * (V - 1) * (V +
1)));

    TAUXYBE = -0.4e1 * (C3 + 1) * (-GB * (V - 1) * (V + 1) * ((-C2 / 0.2e1 - 0.1e1 / 0.2e1) * RC + RU * C1) *
(C3 * RC + (2 * RH)) * (C2 - 0.1e1) * RB ^ 2 / 0.2e1 + (0.2e1 * RC * E * RH * ((-C2 / 0.2e1 - 0.1e1 / 0.2e1) *
RC + RU * C1) * (C3 ^ 2) + (-RU * C1 * GB * (C2 - 0.1e1) ^ 2 * (V - 1) * (V + 1) * RC ^ 2 / 0.4e1 - 0.2e1 * RC
* (RH ^ 2) * E * C2 + 0.4e1 * RU * C1 * E * (RH ^ 2)) * C3 - RU * C1 * GB * RC * RH * (C2 - 0.1e1) ^ 2 * (V -
1) * (V + 1) / 0.2e1) * RB + RC * E * RU * C1 * C3 * (RC * (C2 - 0.1e1) * C3 - 0.2e1 * RC + (0.2e1 * C2 -
0.2e1) * RH) * RH) * E * RU * C1 * C3 * V * EPSYY0T / (-0.2e1 * GB * (V - 1) * (V + 1) * C3 * (-RU * C1 * C3
- (V - 1) * (V + 1) * (C2 + 0.1e1) * RC / 0.2e1 + RU * C1 * (V ^ 2)) * (C3 * RC + (2 * RH)) * (C2 - 0.1e1) * RB
^ 3 - 0.4e1 * (0.2e1 * RU * C1 * E * (C3 ^ 3) * RH + ((V - 1) * (RU * (-(C2 - 0.1e1) ^ 2 * GB / 0.4e1 + E) * C1
+ E * C2 * RH) * (V + 1) * RC - 0.2e1 * RU * C1 * E * RH * (V ^ 2)) * (C3 ^ 2) + GB * (V - 1) * RU * (V + 1) *
C1 * (-(V ^ 2) * RC + RU * (V - 1) * (V + 1) * C1) * (C2 - 0.1e1) * C3 / 0.2e1 + RU * C1 * GB * RC * (C2 -
0.1e1) * (C2 + 0.1e1) * (V - 1) * (V + 1) / 0.4e1) * (C3 * RC + (2 * RH)) * RB ^ 2 + 0.8e1 * RU * C1 * C3 * (-E
* RC ^ 2 * RH * (C2 - 0.1e1) * (C3 ^ 3) / 0.2e1 + RC * E * RH * (RU * (V ^ 2 - 1) * C1 - RH * (C2 - 0.1e1)) *
(C3 ^ 2) + ((-RU * (V - 1) * (V + 1) * ((C2 - 0.1e1) ^ 2 * (V - 1) * (V + 1) * GB / 0.4e1 + E) * C1 / 0.2e1 - E *
RH * ((V ^ 2) - C2) / 0.2e1) * RC ^ 2 - E * (V ^ 2) * RC * (RH ^ 2) + 0.2e1 * RU * C1 * E * (RH ^ 2) * (V - 1) *
(V + 1)) * C3 - RC * RH * (RU * (V - 1) * (V + 1) * ((C2 - 0.1e1) ^ 2 * (V - 1) * (V + 1) * GB / 0.4e1 + E) * C1 -
E * C2 * RH)) * RB + 0.4e1 * E * RU ^ 2 * C1 ^ 2 * (C3 ^ 2) * RC * RH * (V - 1) * (V + 1) * (C3 * RC + (2 *
RH)) * (C2 - 0.1e1));

    % increment of elastic strain exceeds yieLd surface
    DEPSXXB0 = 0.4e1 * (E * RB * RC * RH * (C2 * VU + V) * C3 ^ 3 + (-(VU * GB * RC * (C2 - 1) * (C2 + 1)
* (V - 1) * (V + 1) * RB ^ 2) / 0.4e1 + (((RU * (V ^ 2 - 1) * C1 - RH * (V * VU + C2)) * V * RC + 2 * VU * C2 *
RH ^ 2) * E * RB) - (2 * E * RU * V * C1 * RC * RH * (V - 1) * (V + 1))) * C3 ^ 2 + 0.2e1 * (-GB * (V + 1) * (V -
1) * (C2 - 1) * (C2 + 1) * ((VU * RH) - (V * RC) / 0.2e1) * RB / 0.4e1 + (V * (RU * (V ^ 2 - 1) * C1 - RH * C2) *
RH * E)) * RB * C3 + (V * GB * RB ^ 2 * RH * (C2 - 1) * (C2 + 1) * (V - 1) * (V + 1)) / 0.2e1) * DEPSYY0T *
RU * C1 * RC * (C3 + 0.1e1) / (-0.4e1 * RH * RB * (2 * RB + RC * (C2 - 1)) * RU * C1 * RC * E * C3 ^ 4 + ((2
* RU * C1 * GB * RC * (V - 1) * (V + 1) * (C2 - 1) * RB ^ 3) + ((-0.4e1 * (V + 1) * (V - 1) * (-((C2 - 1) ^ 2 * GB)
/ 0.4e1 + E) * RU * C1 - (4 * E * C2 * RH * (VU - 1) * (VU + 1))) * (RC ^ 2) + (8 * RU * C1 * E * VU * V * RC *
RH) - (16 * RU * C1 * E * RH ^ 2)) * (RB ^ 2) + 0.8e1 * RH * ((RU * (V ^ 2 - 1) * C1) - (V * (VU - V) * RC) /
0.2e1 - (RH * (C2 - 1))) * RU * C1 * RC * E * RB + (4 * E * RU ^ 2 * C1 ^ 2 * RC ^ 2 * RH * (V - 1) * (V + 1) *
(C2 - 1))) * C3 ^ 3 + (-0.2e1 * GB * (V + 1) * (V - 1) * (C2 - 1) * (-(RC ^ 2 * (VU - 1) * (VU + 1) * (C2 + 1)) /
0.2e1 + (C1 * RC * RU * V * VU) - (2 * C1 * RH * RU)) * (RB ^ 3) + ((2 * RU * VU * V * C1 * GB * (V - 1) * (V
+ 1) * (C2 - 1) * RC ^ 2) + (-(2 * RU ^ 2 * GB * (V - 1) ^ 2 * (V + 1) ^ 2 * (C2 - 1) * C1 ^ 2) - 0.8e1 * (V + 1) *
(V - 1) * RH * (-((C2 - 1) ^ 2 * GB) / 0.4e1 + E) * RU * C1 - (8 * E * C2 * RH ^ 2 * (VU - 1) * (VU + 1))) * RC +
(16 * RU * C1 * E * VU * V * RH ^ 2)) * (RB ^ 2) - 0.4e1 * (((V + 1) * (V - 1) * RU * (((C2 - 1) ^ 2 * (V - 1) * (V
+ 1) * GB) / 0.4e1 + E) * C1 + (E * RH * (V * VU - C2))) * (RC ^ 2) + (2 * E * RC * RH ^ 2 * V * VU) - (4 * E *
RU * C1 * RH ^ 2 * (V - 1) * (V + 1))) * RU * C1 * RB + (8 * E * RU ^ 2 * C1 ^ 2 * RC * RH ^ 2 * (V - 1) * (V +
1) * (C2 - 1))) * C3 ^ 2 - 0.8e1 * (GB * ((RU * VU * V * C1) - (RC * (VU - 1) * (VU + 1) * (C2 + 1)) / 0.2e1) * (V
+ 1) * (V - 1) * RH * (C2 - 1) * (RB ^ 2) / 0.2e1 + GB * (V + 1) * (V - 1) * (C2 - 1) * ((C2 / 0.4e1 + 0.1e1 /
0.4e1) * (RC ^ 2) - (VU * V * RC * RH) + (RU * RH * (V - 1) * (V + 1) * C1)) * RU * C1 * RB / 0.2e1 + ((V + 1)
* (V - 1) * RU * (((C2 - 1) ^ 2 * (V - 1) * (V + 1) * GB) / 0.4e1 + E) * C1 - (E * C2 * RH)) * RH * RU * C1 * RC)
* RB * C3 - (2 * RU * C1 * GB * RB ^ 2 * RC * RH * (C2 - 1) * (C2 + 1) * (V - 1) * (V + 1)));

    DEPSYYB0 = 0.4e1 * DEPSYY0T * ((RH * (((-C2 * V * VU - V ^ 2) * RC + (2 * V ^ 2 - 2) * RU * C1) * RB
+ RU * C1 * RC * (V - 1) * (V + 1) * (C2 - 1)) * RC * E * C3 ^ 3) + (-GB * (V + 1) * (V - 1) * (-(VU * V * (C2 +
1) * RC) / 0.2e1 + ((V + 1) * (V - 1) * RU * C1)) * (C2 - 1) * RC * (RB ^ 2) / 0.2e1 + ((-(V + 1) * (V - 1) * RU *
(((C2 - 1) ^ 2) * (V - 1) * (V + 1) * GB / 0.4e1 + E) * C1 + (E * RH * (V * VU + C2))) * (RC ^ 2) - (2 * E * VU *
V * C2 * RC * RH ^ 2) + (4 * E * RU * C1 * RH ^ 2 * (V - 1) * (V + 1))) * RB + (2 * E * RU * C1 * RC * RH ^ 2
* (V - 1) * (V + 1) * (C2 - 1))) * (C3 ^ 2) - 0.2e1 * (GB * (V + 1) * (V - 1) * (C2 - 1) * ((C2 / 0.4e1 + 0.1e1 /
0.4e1) * (RC ^ 2) - (VU * V * RC * RH * (C2 + 1)) / 0.2e1 + (RU * RH * (V - 1) * (V + 1) * C1)) * RB / 0.2e1 +
RH * ((V + 1) * (V - 1) * RU * (((C2 - 1) ^ 2) * (V - 1) * (V + 1) * GB / 0.4e1 + E) * C1 - (E * C2 * RH)) * RC) *
RB * C3 - GB * RH * (RB ^ 2) * RC * (C2 - 1) * (C2 + 1) * (V - 1) * (V + 1) / 0.2e1) * RU * C1 * (C3 + 1) / (-(4
* RH * RB * (2 * RB + RC * (C2 - 1)) * RU * C1 * RC * E * C3 ^ 4) + (0.2e1 * RU * C1 * GB * RC * (V - 1) * (V
+ 1) * (C2 - 1) * (RB ^ 3) + ((-0.4e1 * (V + 1) * (V - 1) * (-((C2 - 1) ^ 2) * GB / 0.4e1 + E) * RU * C1 - (4 * E *
C2 * RH * (VU - 1) * (VU + 1))) * (RC ^ 2) + (8 * RU * C1 * E * VU * V * RC * RH) - (16 * RU * C1 * E * RH ^
2)) * (RB ^ 2) + 0.8e1 * RH * ((RU * (V ^ 2 - 1) * C1) - (V * (VU - V) * RC) / 0.2e1 - (RH * (C2 - 1))) * RU * C1
* RC * E * RB + (4 * E * RU ^ 2 * C1 ^ 2 * RC ^ 2 * RH * (V - 1) * (V + 1) * (C2 - 1))) * (C3 ^ 3) + (-0.2e1 * GB
* (V + 1) * (V - 1) * (C2 - 1) * (-(RC ^ 2 * (VU - 1) * (VU + 1) * (C2 + 1)) / 0.2e1 + (C1 * RC * RU * V * VU) - (2
* C1 * RH * RU)) * (RB ^ 3) + (0.2e1 * RU * VU * V * C1 * GB * (V - 1) * (V + 1) * (C2 - 1) * (RC ^ 2) + (-
0.2e1 * (RU ^ 2) * GB * ((V - 1) ^ 2) * ((V + 1) ^ 2) * (C2 - 1) * (C1 ^ 2) - 0.8e1 * (V + 1) * (V - 1) * RH * (-((C2
- 1) ^ 2) * GB / 0.4e1 + E) * RU * C1 - (8 * E * C2 * RH ^ 2 * (VU - 1) * (VU + 1))) * RC + (16 * RU * C1 * E *
VU * V * RH ^ 2)) * (RB ^ 2) - 0.4e1 * (((V + 1) * (V - 1) * RU * (((C2 - 1) ^ 2) * (V - 1) * (V + 1) * GB / 0.4e1 +
E) * C1 + (E * RH * (V * VU - C2))) * (RC ^ 2) + (2 * E * RC * RH ^ 2 * V * VU) - (4 * E * RU * C1 * RH ^ 2 *
(V - 1) * (V + 1))) * RU * C1 * RB + (8 * E * RU ^ 2 * C1 ^ 2 * RC * RH ^ 2 * (V - 1) * (V + 1) * (C2 - 1))) * (C3
^ 2) - 0.8e1 * (GB * ((RU * VU * V * C1) - (RC * (VU - 1) * (VU + 1) * (C2 + 1)) / 0.2e1) * (V + 1) * (V - 1) *
RH * (C2 - 1) * (RB ^ 2) / 0.2e1 + GB * (V + 1) * (V - 1) * (C2 - 1) * ((C2 / 0.4e1 + 0.1e1 / 0.4e1) * (RC ^ 2) -
(VU * V * RC * RH) + (RU * RH * (V - 1) * (V + 1) * C1)) * RU * C1 * RB / 0.2e1 + ((V + 1) * (V - 1) * RU *
(((C2 - 1) ^ 2) * (V - 1) * (V + 1) * GB / 0.4e1 + E) * C1 - (E * C2 * RH)) * RH * RU * C1 * RC) * RB * C3 -
0.2e1 * RU * C1 * GB * (RB ^ 2) * RC * RH * (C2 - 1) * (C2 + 1) * (V - 1) * (V + 1));

    DEPSXYB0 = DEPSYY0T * C3 * ((-(2 * RH * RU * C1 * RC * (VU - V) * C3 ^ 2) + ((-(RU * VU * (V - 1) *
(V + 1) * C1 + (-VU ^ 2 + 1) * V * RH) * (C2 + 1) * RC ^ 2 + 2 * V * RU * C1 * (RU * (V ^ 2 - 1) * C1 + (-V *
VU + 1) * RH) * RC - 4 * RU * VU * C1 * RH ^ 2) * C3) + 0.4e1 * (-(VU * (V - 1) * (V + 1) * (C2 + 1) * RC) /
0.2e1 + (V * (RU * (V ^ 2 - 1) * C1 + RH))) * RH * RU * C1) * RB ^ 2 + (-(RH * RC * (C2 - 1) * (VU - V) * C3 ^
2) + ((V * (RU * (V - 1) * (V + 1) * (C2 - 1) * C1 + (VU * (C2 + 3) * V + C2 - 1) * RH) * RC) - 0.4e1 * (((V ^ 3 -
V) * RU * C1) + (VU * RH * (C2 - 1)) / 0.2e1) * RH) * C3 + (2 * V * RH * ((-C2 - 1) * RC + (C2 - 1) * (RU * (V
^ 2 - 1) * C1 + RH)))) * RU * C1 * RC * RB - (2 * RU ^ 2 * V * C1 ^ 2 * C3 * RC ^ 2 * RH * (V - 1) * (V + 1) *
(C2 - 1))) * (C3 + 1) * E / (-0.2e1 * GB * (V + 1) * (V - 1) * ((RU * VU * V * C1) - (RU * C1 * C3) - (RC * (VU -
1) * (VU + 1) * (C2 + 1)) / 0.2e1) * (C3 * RC + 2 * RH) * (C2 - 1) * C3 * RB ^ 3 + 0.8e1 * (C3 * RC + 2 * RH) *
(-(E * RU * C1 * C3 ^ 3 * RH) + ((-(V + 1) * (V - 1) * (-((C2 - 1) ^ 2) * GB / 0.4e1 + E) * RU * C1 / 0.2e1 - (E *
C2 * RH * (VU - 1) * (VU + 1)) / 0.2e1) * RC + (E * RU * C1 * VU * V * RH)) * (C3 ^ 2) - GB * (V + 1) * (V - 1)
* (C2 - 1) * (-V * RC * VU + (V + 1) * (V - 1) * RU * C1) * RU * C1 * C3 / 0.4e1 - RU * C1 * RC * GB * (C2 -
1) * (C2 + 1) * (V - 1) * (V + 1) / 0.8e1) * RB ^ 2 + 0.8e1 * C3 * RU * (-(E * RC ^ 2 * RH * (C2 - 1) * C3 ^ 3) /
0.2e1 + RH * ((RU * (V ^ 2 - 1) * C1) - (V * (VU - V) * RC) / 0.2e1 - (RH * (C2 - 1))) * RC * E * (C3 ^ 2) + ((-
(V + 1) * (V - 1) * RU * (((C2 - 1) ^ 2) * GB * (V ^ 2) / 0.4e1 - ((C2 - 1) ^ 2) * GB / 0.4e1 + E) * C1 / 0.2e1 - (E
* RH * (V * VU - C2)) / 0.2e1) * (RC ^ 2) - (E * RC * RH ^ 2 * V * VU) + (2 * E * RU * C1 * RH ^ 2 * (V - 1) *
(V + 1))) * C3 - RH * ((V + 1) * (V - 1) * RU * (((C2 - 1) ^ 2) * GB * (V ^ 2) / 0.4e1 - ((C2 - 1) ^ 2) * GB / 0.4e1
+ E) * C1 - (E * C2 * RH)) * RC) * C1 * RB + (4 * E * RU ^ 2 * C1 ^ 2 * C3 ^ 2 * RC * RH * (V - 1) * (V + 1) *
(C3 * RC + 2 * RH) * (C2 - 1)));

    % caculated cohesion of brick unit by value of hardening(softening) parameter K
```

```
   DKB = sqrt((6 * DEPSPXXB ^ 2 + 6 * DEPSPXYB ^ 2 + 6 * DEPSPYYB ^ 2)) / 0.3e1;
   KB  = KB + DKB;
   KBMAX = ((2 * LC * EPS0M * FCM + 3 * GCM) / LC / FCM) / 0.2e1;
   % find compressive strength by hardening parameter K:
   if KB <= EPS0M
      SIGCB = (FCM * (-2 * KB ^ 2 / EPS0M ^ 2 + 4 * KB / EPS0M + 1)) / 0.3e1;
      KCB = (FCM * (-4 * KB / EPS0M ^ 2 + 4 / EPS0M)) / 0.3e1;
   else
      if KB < KBMAX
         SIGCB = FCM * (0.1e1 - 0.4e1 / 0.9e1 * FCM ^ 2 * LC ^ 2 / GCM ^ 2 * (KB - EPS0M) ^ 2);
         KCB = -0.8e1 / 0.9e1 * FCM ^ 3 * LC ^ 2 / GCM ^ 2 * (KB - EPS0M);
      else
         SIGCB = 0;
         KCB = -0.8e1 / 0.9e1 * FCM ^ 3 * LC ^ 2 / GCM ^ 2 * (KBMAX - EPS0M);
      end
   end
   % find critical stress:
   CB = (0.1e1 - sin(PHIM)) / cos(PHIM) * SIGCB / 0.2e1;
   SIGXXBC1 = 0.18e2 * (0.6e1 * SIGXXBE * sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.6e1 * SIGYYBE *
sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.3e1 * sqrt(SIGXXBE ^ 2 - SIGXXBE * SIGYYBE + SIGYYBE ^ 2 + (3 *
TAUXYBE ^ 2))) * SIGXXBE * CB * cos(PHIM) / (0.3e1 - sin(PHIM)) / (0.36e2 * SIGXXBE ^ 2 * sin(PHIM) ^
2 / (0.3e1 - sin(PHIM)) ^ 2 + 0.72e2 * SIGXXBE * SIGYYBE * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 +
0.36e2 * SIGYYBE ^ 2 * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 - 0.9e1 * SIGXXBE ^ 2 + 0.9e1 * SIGXXBE
* SIGYYBE - 0.9e1 * SIGYYBE ^ 2 - (27 * TAUXYBE ^ 2));

   SIGYYBC1 = 0.18e2 * (0.6e1 * SIGXXBE * sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.6e1 * SIGYYBE *
sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.3e1 * sqrt(SIGXXBE ^ 2 - SIGXXBE * SIGYYBE + SIGYYBE ^ 2 + (3 *
TAUXYBE ^ 2))) * SIGYYBE * CB * cos(PHIM) / (0.3e1 - sin(PHIM)) / (0.36e2 * SIGXXBE ^ 2 * sin(PHIM) ^
2 / (0.3e1 - sin(PHIM)) ^ 2 + 0.72e2 * SIGXXBE * SIGYYBE * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 +
0.36e2 * SIGYYBE ^ 2 * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 - 0.9e1 * SIGXXBE ^ 2 + 0.9e1 * SIGXXBE
* SIGYYBE - 0.9e1 * SIGYYBE ^ 2 - (27 * TAUXYBE ^ 2));

   TAUXYBC1 = 0.18e2 * (0.6e1 * SIGXXBE * sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.6e1 * SIGYYBE *
sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.3e1 * sqrt(SIGXXBE ^ 2 - SIGXXBE * SIGYYBE + SIGYYBE ^ 2 + (3 *
TAUXYBE ^ 2))) * CB * cos(PHIM) / (0.3e1 - sin(PHIM)) * TAUXYBE / (0.36e2 * SIGXXBE ^ 2 * sin(PHIM) ^
2 / (0.3e1 - sin(PHIM)) ^ 2 + 0.72e2 * SIGXXBE * SIGYYBE * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 +
0.36e2 * SIGYYBE ^ 2 * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 - 0.9e1 * SIGXXBE ^ 2 + 0.9e1 * SIGXXBE
* SIGYYBE - 0.9e1 * SIGYYBE ^ 2 - (27 * TAUXYBE ^ 2));

   SIGXXBC2 = -0.18e2 * (-0.6e1 * SIGXXBE * sin(PHIM) / (0.3e1 - sin(PHIM)) - 0.6e1 * SIGYYBE *
sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.3e1 * sqrt(SIGXXBE ^ 2 - SIGXXBE * SIGYYBE + SIGYYBE ^ 2 + (3 *
TAUXYBE ^ 2))) * SIGXXBE * CB * cos(PHIM) / (0.3e1 - sin(PHIM)) / (0.36e2 * SIGXXBE ^ 2 * sin(PHIM) ^
2 / (0.3e1 - sin(PHIM)) ^ 2 + 0.72e2 * SIGXXBE * SIGYYBE * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 +
0.36e2 * SIGYYBE ^ 2 * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 - 0.9e1 * SIGXXBE ^ 2 + 0.9e1 * SIGXXBE
* SIGYYBE - 0.9e1 * SIGYYBE ^ 2 - (27 * TAUXYBE ^ 2));

   SIGYYBC2 = -0.18e2 * (-0.6e1 * SIGXXBE * sin(PHIM) / (0.3e1 - sin(PHIM)) - 0.6e1 * SIGYYBE *
sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.3e1 * sqrt(SIGXXBE ^ 2 - SIGXXBE * SIGYYBE + SIGYYBE ^ 2 + (3 *
TAUXYBE ^ 2))) * SIGYYBE * CB * cos(PHIM) / (0.3e1 - sin(PHIM)) / (0.36e2 * SIGXXBE ^ 2 * sin(PHIM) ^
2 / (0.3e1 - sin(PHIM)) ^ 2 + 0.72e2 * SIGXXBE * SIGYYBE * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 +
0.36e2 * SIGYYBE ^ 2 * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 - 0.9e1 * SIGXXBE ^ 2 + 0.9e1 * SIGXXBE
* SIGYYBE - 0.9e1 * SIGYYBE ^ 2 - (27 * TAUXYBE ^ 2));

   TAUXYBC2 = -0.18e2 * (-0.6e1 * SIGXXBE * sin(PHIM) / (0.3e1 - sin(PHIM)) - 0.6e1 * SIGYYBE *
sin(PHIM) / (0.3e1 - sin(PHIM)) + 0.3e1 * sqrt(SIGXXBE ^ 2 - SIGXXBE * SIGYYBE + SIGYYBE ^ 2 + (3 *
TAUXYBE ^ 2))) * CB * cos(PHIM) / (0.3e1 - sin(PHIM)) * TAUXYBE / (0.36e2 * SIGXXBE ^ 2 * sin(PHIM) ^
```

```
2 / (0.3e1 - sin(PHIM)) ^ 2 + 0.72e2 * SIGXXBE * SIGYYBE * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 +
0.36e2 * SIGYYBE ^ 2 * sin(PHIM) ^ 2 / (0.3e1 - sin(PHIM)) ^ 2 - 0.9e1 * SIGXXBE ^ 2 + 0.9e1 * SIGXXBE
* SIGYYBE - 0.9e1 * SIGYYBE ^ 2 - (27 * TAUXYBE ^ 2));

    if SIGXXBE/SIGXXBC1 > 0
        SIGXXBC = SIGXXBC1;
    else
        SIGXXBC = SIGXXBC2;
    end
    if SIGYYBE/SIGYYBC1 > 0
        SIGYYBC = SIGYYBC1;
    else
        SIGYYBC = SIGYYBC2;
    end
    if TAUXYBE/TAUXYBC1 > 0
        TAUXYBC = TAUXYBC1;
    else
        TAUXYBC = TAUXYBC2;
    end

    while CU >= 0
        % yield function of brick unit:
        FU = (sqrt(SIGXXUE ^ 2 - SIGXXUE * SIGYYUE + SIGYYUE ^ 2) * (-0.3e1 + sin(PHIU)) + (-0.2e1 *
SIGXXUE - 0.2e1 * SIGYYUE) * sin(PHIU) + 0.6e1 * CU * cos(PHIU)) / (-0.3e1 + sin(PHIU));
        if FU <= 0 % before yielding, plastic strain = 0
            DEPSPXXU = 0;
            DEPSPYYU = 0;
            break;
        else

            % calculate plastic strain increment:
            DEPSPXXU = 0.6e1 * ((((((DEPSXXU0 - DEPSYYU0) * VU - 0.10e2 / 0.3e1 * DEPSXXU0 - 0.7e1 /
0.3e1 * DEPSYYU0) * SIGXXUC ^ 2 - 0.3e1 / 0.2e1 * ((DEPSXXU0 - DEPSYYU0) * VU - 0.20e2 / 0.9e1 *
DEPSXXU0 - 0.11e2 / 0.9e1 * DEPSYYU0) * SIGYYUC * SIGXXUC + ((DEPSXXU0 - DEPSYYU0) * VU -
0.17e2 / 0.3e1 * DEPSXXU0 - 0.14e2 / 0.3e1 * DEPSYYU0) * SIGYYUC ^ 2 / 0.2e1) * sin(PHIU) - 0.3e1 *
(((DEPSXXU0 - DEPSYYU0) * VU - 0.2e1 / 0.3e1 * DEPSXXU0 + DEPSYYU0 / 0.3e1) * SIGXXUC -
((DEPSXXU0 - DEPSYYU0) * VU - DEPSXXU0 / 0.3e1 + 0.2e1 / 0.3e1 * DEPSYYU0) * SIGYYUC) * (-
SIGYYUC / 0.2e1 + SIGXXUC)) * sin(PSIU) - 0.3e1 * (((DEPSXXU0 - DEPSYYU0) * VU - 0.2e1 / 0.3e1 *
DEPSXXU0 + DEPSYYU0 / 0.3e1) * SIGXXUC - ((DEPSXXU0 - DEPSYYU0) * VU - DEPSXXU0 / 0.3e1 +
0.2e1 / 0.3e1 * DEPSYYU0) * SIGYYUC) * (-SIGYYUC / 0.2e1 + SIGXXUC) * (-0.3e1 + sin(PHIU))) *
sqrt(SIGXXUC ^ 2 - SIGXXUC * SIGYYUC + SIGYYUC ^ 2) - 0.2e1 * (((((DEPSXXU0 - DEPSYYU0) * VU -
0.4e1 / 0.3e1 * DEPSXXU0 - DEPSYYU0 / 0.3e1) * SIGXXUC - ((DEPSXXU0 - DEPSYYU0) * VU - 0.2e1 /
0.3e1 * DEPSXXU0 + DEPSYYU0 / 0.3e1) * SIGYYUC) * sin(PHIU) + ((-0.3e1 * DEPSXXU0 + 0.3e1 *
DEPSYYU0) * VU + 0.2e1 * DEPSXXU0 - DEPSYYU0) * SIGXXUC + 0.3e1 * ((DEPSXXU0 - DEPSYYU0)
* VU - DEPSXXU0 / 0.3e1 + 0.2e1 / 0.3e1 * DEPSYYU0) * SIGYYUC) * sin(PSIU) + 0.2e1 * sin(PHIU) *
(DEPSXXU0 + DEPSYYU0) * (-SIGYYUC / 0.2e1 + SIGXXUC)) * (SIGXXUC ^ 2 - SIGXXUC * SIGYYUC +
SIGYYUC ^ 2)) * EU / (((((0.8e1 * SU * VU ^ 2 + ((9 * EU) + 0.4e1 * SU) * VU - (37 * EU) - 0.4e1 * SU) *
SIGXXUC ^ 2 - 0.18e2 * (0.4e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.2e1 / 0.9e1 * SU) * VU - 0.20e2 / 0.9e1 *
EU - 0.2e1 / 0.9e1 * SU) * SIGYYUC * SIGXXUC + 0.9e1 * (0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1 /
0.9e1 * SU) * VU - 0.37e2 / 0.9e1 * EU - 0.4e1 / 0.9e1 * SU) * SIGYYUC ^ 2) * sin(PHIU) + (-0.24e2 * SU *
VU ^ 2 + (-(27 * EU) - 0.12e2 * SU) * VU + (15 * EU) + 0.12e2 * SU) * SIGXXUC ^ 2 + 0.54e2 * (0.4e1 /
0.9e1 * SU * VU ^ 2 + (EU + 0.2e1 / 0.9e1 * SU) * VU - 0.4e1 / 0.9e1 * EU - 0.2e1 / 0.9e1 * SU) * SIGYYUC
* SIGXXUC - 0.27e2 * (0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1 / 0.9e1 * SU) * VU - 0.5e1 / 0.9e1 * EU -
0.4e1 / 0.9e1 * SU) * SIGYYUC ^ 2) * sin(PSIU) - 0.27e2 * ((0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1 /
0.9e1 * SU) * VU - 0.5e1 / 0.9e1 * EU - 0.4e1 / 0.9e1 * SU) * SIGXXUC ^ 2 - 0.2e1 * (0.4e1 / 0.9e1 * SU *
VU ^ 2 + (EU + 0.2e1 / 0.9e1 * SU) * VU - 0.4e1 / 0.9e1 * EU - 0.2e1 / 0.9e1 * SU) * SIGYYUC * SIGXXUC
+ (0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1 / 0.9e1 * SU) * VU - 0.5e1 / 0.9e1 * EU - 0.4e1 / 0.9e1 * SU) *
SIGYYUC ^ 2) * (-0.3e1 + sin(PHIU))) * sqrt(SIGXXUC ^ 2 - SIGXXUC * SIGYYUC + SIGYYUC ^ 2) +
0.8e1 * EU * (SIGXXUC ^ 2 - SIGXXUC * SIGYYUC + SIGYYUC ^ 2) * (SIGXXUC + SIGYYUC) *
((sin(PHIU) - 0.3e1 / 0.2e1) * sin(PSIU) - 0.3e1 / 0.2e1 * sin(PHIU)));

        DEPSPYYU = -0.3e1 * ((((((DEPSXXU0 - DEPSYYU0) * VU + 0.14e2 / 0.3e1 * DEPSXXU0 +
0.17e2 / 0.3e1 * DEPSYYU0) * SIGXXUC ^ 2 - 0.3e1 * ((DEPSXXU0 - DEPSYYU0) * VU + 0.11e2 / 0.9e1 *
DEPSXXU0 + 0.20e2 / 0.9e1 * DEPSYYU0) * SIGYYUC * SIGXXUC + 0.2e1 * SIGYYUC ^ 2 *
((DEPSXXU0 - DEPSYYU0) * VU + 0.7e1 / 0.3e1 * DEPSXXU0 + 0.10e2 / 0.3e1 * DEPSYYU0)) *
sin(PHIU) - 0.3e1 * (((DEPSXXU0 - DEPSYYU0) * VU - 0.2e1 / 0.3e1 * DEPSXXU0 + DEPSYYU0 / 0.3e1) *
SIGXXUC - ((DEPSXXU0 - DEPSYYU0) * VU - DEPSXXU0 / 0.3e1 + 0.2e1 / 0.3e1 * DEPSYYU0) *
SIGYYUC) * (SIGXXUC - 0.2e1 * SIGYYUC)) * sin(PSIU) - 0.3e1 * (((DEPSXXU0 - DEPSYYU0) * VU -
0.2e1 / 0.3e1 * DEPSXXU0 + DEPSYYU0 / 0.3e1) * SIGXXUC - ((DEPSXXU0 - DEPSYYU0) * VU -
DEPSXXU0 / 0.3e1 + 0.2e1 / 0.3e1 * DEPSYYU0) * SIGYYUC) * (-0.3e1 + sin(PHIU)) * (SIGXXUC - 0.2e1
* SIGYYUC)) * sqrt(SIGXXUC ^ 2 - SIGXXUC * SIGYYUC + SIGYYUC ^ 2) + 0.4e1 * (SIGXXUC ^ 2 -
SIGXXUC * SIGYYUC + SIGYYUC ^ 2) * (((((DEPSXXU0 - DEPSYYU0) * VU - DEPSXXU0 / 0.3e1 + 0.2e1
/ 0.3e1 * DEPSYYU0) * SIGXXUC - ((DEPSXXU0 - DEPSYYU0) * VU + DEPSXXU0 / 0.3e1 + 0.4e1 / 0.3e1
* DEPSYYU0) * SIGYYUC) * sin(PHIU) + ((-0.3e1 * DEPSXXU0 + 0.3e1 * DEPSYYU0) * VU + 0.2e1 *
DEPSXXU0 - DEPSYYU0) * SIGXXUC + 0.3e1 * ((DEPSXXU0 - DEPSYYU0) * VU - DEPSXXU0 / 0.3e1 +
0.2e1 / 0.3e1 * DEPSYYU0) * SIGYYUC) * sin(PSIU) - sin(PHIU) * (DEPSXXU0 + DEPSYYU0) *
(SIGXXUC - 0.2e1 * SIGYYUC))) * EU / (((((0.8e1 * SU * VU ^ 2 + ((9 * EU) + 0.4e1 * SU) * VU - (37 * EU) -
0.4e1 * SU) * SIGXXUC ^ 2 - 0.18e2 * (0.4e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.2e1 / 0.9e1 * SU) * VU -
0.20e2 / 0.9e1 * EU - 0.2e1 / 0.9e1 * SU) * SIGYYUC * SIGXXUC + 0.9e1 * (0.8e1 / 0.9e1 * SU * VU ^ 2 +
(EU + 0.4e1 / 0.9e1 * SU) * VU - 0.37e2 / 0.9e1 * EU - 0.4e1 / 0.9e1 * SU) * SIGYYUC ^ 2) * sin(PHIU) + (-
0.24e2 * SU * VU ^ 2 + (-(27 * EU) - 0.12e2 * SU) * VU + (15 * EU) + 0.12e2 * SU) * SIGXXUC ^ 2 + 0.54e2
* (0.4e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.2e1 / 0.9e1 * SU) * VU - 0.4e1 / 0.9e1 * EU - 0.2e1 / 0.9e1 * SU) *
SIGYYUC * SIGXXUC - 0.27e2 * (0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1 / 0.9e1 * SU) * VU - 0.5e1 /
0.9e1 * EU - 0.4e1 / 0.9e1 * SU) * SIGYYUC ^ 2) * sin(PSIU) - 0.27e2 * ((0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU
+ 0.4e1 / 0.9e1 * SU) * VU - 0.5e1 / 0.9e1 * EU - 0.4e1 / 0.9e1 * SU) * SIGXXUC ^ 2 - 0.2e1 * (0.4e1 / 0.9e1
* SU * VU ^ 2 + (EU + 0.2e1 / 0.9e1 * SU) * VU - 0.4e1 / 0.9e1 * EU - 0.2e1 / 0.9e1 * SU) * SIGYYUC *
SIGXXUC + (0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1 / 0.9e1 * SU) * VU - 0.5e1 / 0.9e1 * EU - 0.4e1 /
0.9e1 * SU) * SIGYYUC ^ 2) * (-0.3e1 + sin(PHIU))) * sqrt(SIGXXUC ^ 2 - SIGXXUC * SIGYYUC +
SIGYYUC ^ 2) + 0.8e1 * EU * (SIGXXUC ^ 2 - SIGXXUC * SIGYYUC + SIGYYUC ^ 2) * (SIGXXUC +
SIGYYUC) * ((sin(PHIU) - 0.3e1 / 0.2e1) * sin(PSIU) - 0.3e1 / 0.2e1 * sin(PHIU)));

        % recalculate softening modulus SUC:
        DLU = 0.6e1 * EU * ((-0.3e1 + sin(PHIU)) * (((DEPSXXU0 - DEPSYYU0) * VU - 0.2e1 / 0.3e1 *
DEPSXXU0 + DEPSYYU0 / 0.3e1) * SIGXXUC - ((DEPSXXU0 - DEPSYYU0) * VU - DEPSXXU0 / 0.3e1 +
0.2e1 / 0.3e1 * DEPSYYU0) * SIGYYUC) * sqrt(SIGXXUC ^ 2 - SIGXXUC * SIGYYUC + SIGYYUC ^ 2) +
0.4e1 / 0.3e1 * sin(PHIU) * (SIGXXUC ^ 2 - SIGXXUC * SIGYYUC + SIGYYUC ^ 2) * (DEPSXXU0 +
DEPSYYU0)) * (-0.3e1 + sin(PSIU)) / (0.8e1 * EU * (SIGXXUC + SIGYYUC) * ((sin(PSIU) - 0.3e1 / 0.2e1) *
sin(PHIU) - 0.3e1 / 0.2e1 * sin(PSIU)) * sqrt(SIGXXUC ^ 2 - SIGXXUC * SIGYYUC + SIGYYUC ^ 2) +
(((0.8e1 * SU * VU ^ 2 + (0.9e1 * EU + 0.4e1 * SU) * VU - 0.37e2 * EU - 0.4e1 * SU) * SIGXXUC ^ 2 -
0.18e2 * (0.4e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.2e1 / 0.9e1 * SU) * VU - 0.20e2 / 0.9e1 * EU - 0.2e1 / 0.9e1
* SU) * SIGYYUC * SIGXXUC + 0.9e1 * (0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1 / 0.9e1 * SU) * VU -
0.37e2 / 0.9e1 * EU - 0.4e1 / 0.9e1 * SU) * SIGYYUC ^ 2) * sin(PSIU) + (-0.24e2 * SU * VU ^ 2 + (-0.27e2 *
EU - 0.12e2 * SU) * VU + 0.15e2 * EU + 0.12e2 * SU) * SIGXXUC ^ 2 + 0.54e2 * (0.4e1 / 0.9e1 * SU * VU ^
2 + (EU + 0.2e1 / 0.9e1 * SU) * VU - 0.4e1 / 0.9e1 * EU - 0.2e1 / 0.9e1 * SU) * SIGYYUC * SIGXXUC -
0.27e2 * (0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1 / 0.9e1 * SU) * VU - 0.5e1 / 0.9e1 * EU - 0.4e1 / 0.9e1
* SU) * SIGYYUC ^ 2) * sin(PHIU) - 0.27e2 * ((0.8e1 / 0.9e1 * SU * VU ^ 2 + (EU + 0.4e1 / 0.9e1 * SU) * VU
- 0.5e1 / 0.9e1 * EU - 0.4e1 / 0.9e1 * SU) * SIGXXUC ^ 2 - 0.2e1 * (0.4e1 / 0.9e1 * SU * VU ^ 2 + (EU +
0.2e1 / 0.9e1 * SU) * VU - 0.4e1 / 0.9e1 * EU - 0.2e1 / 0.9e1 * SU) * SIGYYUC * SIGXXUC + (0.8e1 / 0.9e1
* SU * VU ^ 2 + (EU + 0.4e1 / 0.9e1 * SU) * VU - 0.5e1 / 0.9e1 * EU - 0.4e1 / 0.9e1 * SU) * SIGYYUC ^ 2) *
(-0.3e1 + sin(PSIU)));

        FKU = -0.3e1 / (0.3e1 - sin(PHIU)) * (0.1e1 - sin(PHIU)) * KCU;
        SUC = -FKU*DKU/DLU;
```

```
            if abs(SUC-SU) < TOR2
                break;
            else
                SU = SUC;
            end
        end
    end

    while CH >= 0
        % yield function of brick unit:
        FH = (sqrt(SIGXXHE ^ 2 - SIGXXHE * SIGYYHE + SIGYYHE ^ 2) * (-0.3e1 + sin(PHIM)) + (-0.2e1 *
SIGXXHE - 0.2e1 * SIGYYHE) * sin(PHIM) + 0.6e1 * CH * cos(PHIM)) / (-0.3e1 + sin(PHIM));
        if FH <= 0 % before yielding, plastic strain = 0
            DEPSPXXH = 0;
            DEPSPYYH = 0;
            break;
        else

            % calculate plastic strain increment:
            DEPSPXXH = ((((((((20 * DEPSYYH0 - 6 * DEPSXXH0) * E - 14 * EU * DEPSYYH0) * V + 20 * E *
DEPSXXH0 + 14 * EU * DEPSYYH0) * SIGXXHC ^ 2) - 0.20e2 * (((DEPSYYH0 - 0.9e1 / 0.20e2 *
DEPSXXH0) * E - 0.11e2 / 0.20e2 * EU * DEPSYYH0) * V + (E * DEPSXXH0) + 0.11e2 / 0.20e2 * EU *
DEPSYYH0) * SIGYYHC * SIGXXHC + 0.17e2 * (((DEPSYYH0 - 0.3e1 / 0.17e2 * DEPSXXH0) * E - 0.14e2
/ 0.17e2 * EU * DEPSYYH0) * V + (E * DEPSXXH0) + 0.14e2 / 0.17e2 * EU * DEPSYYH0) * SIGYYHC ^ 2)
* sin(PHIM) - 0.12e2 * (SIGXXHC - SIGYYHC / 0.2e1) * ((((DEPSYYH0 - 0.3e1 / 0.2e1 * DEPSXXH0) * E +
(EU * DEPSYYH0) / 0.2e1) * V + (E * DEPSXXH0) - (EU * DEPSYYH0) / 0.2e1) * SIGXXHC - SIGYYHC *
(((DEPSYYH0 - 3 * DEPSXXH0) * E + 2 * EU * DEPSYYH0) * V + E * DEPSXXH0 - 2 * EU * DEPSYYH0) /
0.2e1)) * sin(PSIM) - 0.12e2 * (SIGXXHC - SIGYYHC / 0.2e1) * (sin(PHIM) - 0.3e1) * ((((DEPSYYH0 -
0.3e1 / 0.2e1 * DEPSXXH0) * E + (EU * DEPSYYH0) / 0.2e1) * V + (E * DEPSXXH0) - (EU * DEPSYYH0) /
0.2e1) * SIGXXHC - SIGYYHC * (((DEPSYYH0 - 3 * DEPSXXH0) * E + 2 * EU * DEPSYYH0) * V + E *
DEPSXXH0 - 2 * EU * DEPSYYH0) / 0.2e1)) * sqrt((SIGXXHC ^ 2) - SIGXXHC * SIGYYHC + SIGYYHC ^
2) - 0.16e2 * ((SIGXXHC ^ 2) - SIGXXHC * SIGYYHC + SIGYYHC ^ 2) * ((((((DEPSYYH0 - 0.3e1 / 0.4e1 *
DEPSXXH0) * E - (EU * DEPSYYH0) / 0.4e1) * V + (E * DEPSXXH0) + (EU * DEPSYYH0) / 0.4e1) *
SIGXXHC - (((DEPSYYH0 - 0.3e1 / 0.2e1 * DEPSXXH0) * E + (EU * DEPSYYH0) / 0.2e1) * V + (E *
DEPSXXH0) - (EU * DEPSYYH0) / 0.2e1) * SIGYYHC / 0.2e1) * sin(PHIM) + (((-0.3e1 / 0.2e1 * DEPSYYH0
+ 0.9e1 / 0.4e1 * DEPSXXH0) * E - 0.3e1 / 0.4e1 * EU * DEPSYYH0) * V - 0.3e1 / 0.2e1 * E * DEPSXXH0 +
0.3e1 / 0.4e1 * EU * DEPSYYH0) * SIGXXHC + 0.3e1 / 0.4e1 * SIGYYHC * (((DEPSYYH0 - 3 *
DEPSXXH0) * E + 2 * EU * DEPSYYH0) * V + E * DEPSXXH0 - 2 * EU * DEPSYYH0)) * sin(PSIM) - 0.3e1 /
0.2e1 * (DEPSYYH0 * (E - EU) * V + E * DEPSXXH0 + EU * DEPSYYH0) * (SIGXXHC - SIGYYHC / 0.2e1)
* sin(PHIM))) / ((((((-8 * V ^ 2 * SH + (8 * E - 17 * EU - 4 * SH) * V + 20 * E + 17 * EU + 4 * SH) * SIGXXHC
^ 2) - 0.2e1 * SIGYYHC * (-4 * V ^ 2 * SH + (E - 10 * EU - 2 * SH) * V + 10 * E + 10 * EU + 2 * SH) *
SIGXXHC + 0.11e2 * SIGYYHC ^ 2 * (-0.8e1 / 0.11e2 * (V ^ 2) * SH + (E - 0.20e2 / 0.11e2 * EU - 0.4e1 /
0.11e2 * SH) * V + 0.17e2 / 0.11e2 * E + 0.20e2 / 0.11e2 * EU + 0.4e1 / 0.11e2 * SH)) * sin(PHIM) + ((24 *
V ^ 2 * SH + (24 * E + 3 * EU + 12 * SH) * V - 12 * E - 3 * EU - 12 * SH) * SIGXXHC ^ 2) - 0.42e2 * (0.4e1 /
0.7e1 * (V ^ 2) * SH + (E + 0.2e1 / 0.7e1 * EU + 0.2e1 / 0.7e1 * SH) * V - 0.2e1 / 0.7e1 * E - 0.2e1 / 0.7e1 *
EU - 0.2e1 / 0.7e1 * SH) * SIGYYHC * SIGXXHC + 0.15e2 * (0.8e1 / 0.5e1 * (V ^ 2) * SH + (E + 0.4e1 /
0.5e1 * EU + 0.4e1 / 0.5e1 * SH) * V - E / 0.5e1 - 0.4e1 / 0.5e1 * EU - 0.4e1 / 0.5e1 * SH) * SIGYYHC ^ 2) *
sin(PSIM) + 0.24e2 * (sin(PHIM) - 0.3e1) * (((V ^ 2 * SH) + (E + EU / 0.8e1 + SH / 0.2e1) * V - E / 0.2e1 -
EU / 0.8e1 - SH / 0.2e1) * (SIGXXHC ^ 2) - 0.7e1 / 0.4e1 * (0.4e1 / 0.7e1 * (V ^ 2) * SH + (E + 0.2e1 / 0.7e1
* EU + 0.2e1 / 0.7e1 * SH) * V - 0.2e1 / 0.7e1 * E - 0.2e1 / 0.7e1 * EU - 0.2e1 / 0.7e1 * SH) * SIGYYHC *
SIGXXHC + 0.5e1 / 0.8e1 * (0.8e1 / 0.5e1 * (V ^ 2) * SH + (E + 0.4e1 / 0.5e1 * EU + 0.4e1 / 0.5e1 * SH) * V
- E / 0.5e1 - 0.4e1 / 0.5e1 * EU - 0.4e1 / 0.5e1 * SH) * SIGYYHC ^ 2)) * sqrt((SIGXXHC ^ 2) - SIGXXHC *
SIGYYHC + SIGYYHC ^ 2) + 0.8e1 * ((SIGXXHC ^ 2) - SIGXXHC * SIGYYHC + SIGYYHC ^ 2) * ((((E - EU)
* V - 2 * E + EU) * SIGXXHC) - 0.2e1 * (((E - EU) * V) - E / 0.2e1 + EU) * SIGYYHC) * ((sin(PHIM) - 0.3e1 /
0.2e1) * sin(PSIM) - 0.3e1 / 0.2e1 * sin(PHIM)));

            DEPSPYYH = ((((((((14 * DEPSYYH0 + 3 * DEPSXXH0) * E - 17 * EU * DEPSYYH0) * V + 14 * E *
DEPSXXH0 + 17 * EU * DEPSYYH0) * SIGXXHC ^ 2) - 0.11e2 * (((DEPSYYH0 + 0.9e1 / 0.11e2 *
DEPSXXH0) * E - 0.20e2 / 0.11e2 * EU * DEPSYYH0) * V + (E * DEPSXXH0) + 0.20e2 / 0.11e2 * EU *
DEPSYYH0) * SIGYYHC * SIGXXHC + 0.14e2 * (((DEPSYYH0 + 0.3e1 / 0.7e1 * DEPSXXH0) * E - 0.10e2
/ 0.7e1 * EU * DEPSYYH0) * V + (E * DEPSXXH0) + 0.10e2 / 0.7e1 * EU * DEPSYYH0) * SIGYYHC ^ 2) *
sin(PHIM) + 0.6e1 * ((((DEPSYYH0 - 0.3e1 / 0.2e1 * DEPSXXH0) * E + (EU * DEPSYYH0) / 0.2e1) * V + (E
* DEPSXXH0) - (EU * DEPSYYH0) / 0.2e1) * SIGXXHC - SIGYYHC * (((DEPSYYH0 - 3 * DEPSXXH0) * E
+ 2 * EU * DEPSYYH0) * V + E * DEPSXXH0 - 2 * EU * DEPSYYH0) / 0.2e1) * (SIGXXHC - 0.2e1 *
SIGYYHC)) * sin(PSIM) + 0.6e1 * (sin(PHIM) - 0.3e1) * ((((DEPSYYH0 - 0.3e1 / 0.2e1 * DEPSXXH0) * E +
(EU * DEPSYYH0) / 0.2e1) * V + (E * DEPSXXH0) - (EU * DEPSYYH0) / 0.2e1) * SIGXXHC - SIGYYHC *
(((DEPSYYH0 - 3 * DEPSXXH0) * E + 2 * EU * DEPSYYH0) * V + E * DEPSXXH0 - 2 * EU * DEPSYYH0) /
0.2e1) * (SIGXXHC - 0.2e1 * SIGYYHC)) * sqrt((SIGXXHC ^ 2) - SIGXXHC * SIGYYHC + SIGYYHC ^ 2) -
0.4e1 * ((SIGXXHC ^ 2) - SIGXXHC * SIGYYHC + SIGYYHC ^ 2) * (((((((DEPSYYH0 - 3 * DEPSXXH0) * E
+ 2 * EU * DEPSYYH0) * V + E * DEPSXXH0 - 2 * EU * DEPSYYH0) * SIGXXHC) + (((DEPSYYH0 + 3 *
DEPSXXH0) * E - 4 * EU * DEPSYYH0) * V + E * DEPSXXH0 + 4 * EU * DEPSYYH0) * SIGYYHC) *
sin(PHIM) + ((((-6 * DEPSYYH0 + 9 * DEPSXXH0) * E - 3 * EU * DEPSYYH0) * V - 6 * E * DEPSXXH0 + 3 *
EU * DEPSYYH0) * SIGXXHC) + 0.3e1 * SIGYYHC * (((DEPSYYH0 - 3 * DEPSXXH0) * E + 2 * EU *
DEPSYYH0) * V + E * DEPSXXH0 - 2 * EU * DEPSYYH0)) * sin(PSIM) + 0.3e1 * (DEPSYYH0 * (E - EU) *
V + E * DEPSXXH0 + EU * DEPSYYH0) * (SIGXXHC - 0.2e1 * SIGYYHC) * sin(PHIM))) / ((((((-8 * V ^ 2 *
SH + (8 * E - 17 * EU - 4 * SH) * V + 20 * E + 17 * EU + 4 * SH) * SIGXXHC ^ 2) - 0.2e1 * SIGYYHC * (-4 *
V ^ 2 * SH + (E - 10 * EU - 2 * SH) * V + 10 * E + 10 * EU + 2 * SH) * SIGXXHC + 0.11e2 * SIGYYHC ^ 2 *
(-0.8e1 / 0.11e2 * (V ^ 2) * SH + (E - 0.20e2 / 0.11e2 * EU - 0.4e1 / 0.11e2 * SH) * V + 0.17e2 / 0.11e2 * E +
0.20e2 / 0.11e2 * EU + 0.4e1 / 0.11e2 * SH)) * sin(PHIM) + ((24 * V ^ 2 * SH + (24 * E + 3 * EU + 12 * SH) *
V - 12 * E - 3 * EU - 12 * SH) * SIGXXHC ^ 2) - 0.42e2 * (0.4e1 / 0.7e1 * (V ^ 2) * SH + (E + 0.2e1 / 0.7e1 *
EU + 0.2e1 / 0.7e1 * SH) * V - 0.2e1 / 0.7e1 * E - 0.2e1 / 0.7e1 * EU - 0.2e1 / 0.7e1 * SH) * SIGYYHC *
SIGXXHC + 0.15e2 * (0.8e1 / 0.5e1 * (V ^ 2) * SH + (E + 0.4e1 / 0.5e1 * EU + 0.4e1 / 0.5e1 * SH) * V - E /
0.5e1 - 0.4e1 / 0.5e1 * EU - 0.4e1 / 0.5e1 * SH) * SIGYYHC ^ 2) * sin(PSIM) + 0.24e2 * (sin(PHIM) - 0.3e1)
* (((V ^ 2 * SH) + (E + EU / 0.8e1 + SH / 0.2e1) * V - E / 0.2e1 - EU / 0.8e1 - SH / 0.2e1) * (SIGXXHC ^ 2) -
0.7e1 / 0.4e1 * (0.4e1 / 0.7e1 * (V ^ 2) * SH + (E + 0.2e1 / 0.7e1 * EU + 0.2e1 / 0.7e1 * SH) * V - 0.2e1 /
0.7e1 * E - 0.2e1 / 0.7e1 * EU - 0.2e1 / 0.7e1 * SH) * SIGYYHC * SIGXXHC + 0.5e1 / 0.8e1 * (0.8e1 / 0.5e1
* (V ^ 2) * SH + (E + 0.4e1 / 0.5e1 * EU + 0.4e1 / 0.5e1 * SH) * V - E / 0.5e1 - 0.4e1 / 0.5e1 * EU - 0.4e1 /
0.5e1 * SH) * SIGYYHC ^ 2)) * sqrt((SIGXXHC ^ 2) - SIGXXHC * SIGYYHC + SIGYYHC ^ 2) + 0.8e1 *
((SIGXXHC ^ 2) - SIGXXHC * SIGYYHC + SIGYYHC ^ 2) * ((((E - EU) * V - 2 * E + EU) * SIGXXHC) -
0.2e1 * (((E - EU) * V) - E / 0.2e1 + EU) * SIGYYHC) * ((sin(PHIM) - 0.3e1 / 0.2e1) * sin(PSIM) - 0.3e1 /
0.2e1 * sin(PHIM)));

        % recalculate softening modulus SUC:
        DLH = 0.4e1 * (((((DEPSYYH0 - 0.3e1 / 0.2e1 * DEPSXXH0) * E + EU * DEPSYYH0 / 0.2e1) * V +
E * DEPSXXH0 - EU * DEPSYYH0 / 0.2e1) * SIGXXHC - SIGYYHC * (((DEPSYYH0 - 0.3e1 * DEPSXXH0)
* E + 0.2e1 * EU * DEPSYYH0) * V + E * DEPSXXH0 - 0.2e1 * EU * DEPSYYH0) / 0.2e1) * (sin(PHIM) -
0.3e1) * sqrt(SIGXXHC ^ 2 - SIGXXHC * SIGYYHC + SIGYYHC ^ 2) - 0.2e1 * sin(PHIM) * (DEPSYYH0 * (E
- EU) * V + E * DEPSXXH0 + EU * DEPSYYH0) * (SIGXXHC ^ 2 - SIGXXHC * SIGYYHC + SIGYYHC ^ 2))
* (-0.3e1 + sin(PSIM)) / (0.8e1 * ((sin(PSIM) - 0.3e1 / 0.2e1) * sin(PHIM) - 0.3e1 / 0.2e1 * sin(PSIM)) * (((E -
EU) * V - 0.2e1 * E + EU) * SIGXXHC - 0.2e1 * ((E - EU) * V - E / 0.2e1 + EU) * SIGYYHC) * sqrt(SIGXXHC
^ 2 - SIGXXHC * SIGYYHC + SIGYYHC ^ 2) + (((-0.8e1 * V ^ 2 * SH + (0.8e1 * E - 0.17e2 * EU - 0.4e1 *
SH) * V + 0.20e2 * E + 0.17e2 * EU + 0.4e1 * SH) * SIGXXHC ^ 2 - 0.2e1 * SIGYYHC * (-0.4e1 * V ^ 2 * SH
+ (E - 0.10e2 * EU - 0.2e1 * SH) * V + 0.10e2 * E + 0.10e2 * EU + 0.2e1 * SH) * SIGXXHC + 0.11e2 *
SIGYYHC ^ 2 * (-0.8e1 / 0.11e2 * V ^ 2 * SH + (E - 0.20e2 / 0.11e2 * EU - 0.4e1 / 0.11e2 * SH) * V + 0.17e2
/ 0.11e2 * E + 0.20e2 / 0.11e2 * EU + 0.4e1 / 0.11e2 * SH)) * sin(PSIM) + (0.24e2 * V ^ 2 * SH + (0.24e2 *
E + 0.3e1 * EU + 0.12e2 * SH) * V - 0.12e2 * E - 0.3e1 * EU - 0.12e2 * SH) * SIGXXHC ^ 2 - 0.42e2 *
(0.4e1 / 0.7e1 * V ^ 2 * SH + (E + 0.2e1 / 0.7e1 * EU + 0.2e1 / 0.7e1 * SH) * V - 0.2e1 / 0.7e1 * E - 0.2e1 /
0.7e1 * EU - 0.2e1 / 0.7e1 * SH) * SIGYYHC * SIGXXHC + 0.15e2 * (0.8e1 / 0.5e1 * V ^ 2 * SH + (E +
0.4e1 / 0.5e1 * EU + 0.4e1 / 0.5e1 * SH) * V - E / 0.5e1 - 0.4e1 / 0.5e1 * EU - 0.4e1 / 0.5e1 * SH) *
SIGYYHC ^ 2) * sin(PHIM) + 0.24e2 * (-0.3e1 + sin(PSIM)) * ((V ^ 2 * SH + (E + EU / 0.8e1 + SH / 0.2e1) *
V - E / 0.2e1 - EU / 0.8e1 - SH / 0.2e1) * SIGXXHC ^ 2 - 0.7e1 / 0.4e1 * (0.4e1 / 0.7e1 * V ^ 2 * SH + (E +
0.2e1 / 0.7e1 * EU + 0.2e1 / 0.7e1 * SH) * V - 0.2e1 / 0.7e1 * E - 0.2e1 / 0.7e1 * EU - 0.2e1 / 0.7e1 * SH) *
SIGYYHC * SIGXXHC + 0.5e1 / 0.8e1 * (0.8e1 / 0.5e1 * V ^ 2 * SH + (E + 0.4e1 / 0.5e1 * EU + 0.4e1 /
0.5e1 * SH) * V - E / 0.5e1 - 0.4e1 / 0.5e1 * EU - 0.4e1 / 0.5e1 * SH) * SIGYYHC ^ 2));

        FKH = -0.3e1 / (0.3e1 - sin(PHIM)) * (0.1e1 - sin(PHIM)) * KCH;
        SHC = -FKH* DKH / DLH ;
        if abs(SHC-SH) < TOR2
            break;
        else
            SH = SHC;
        end
    end
end

while CB >= 0
    % yield function of brick unit:
    FB = (sqrt((SIGXXBE ^ 2 - SIGXXBE * SIGYYBE + SIGYYBE ^ 2 + 3 * TAUXYBE^2)) * (-0.3e1 +
sin(PHIM)) + (-2 * SIGXXBE - 2 * SIGYYBE) * sin(PHIM) + 0.6e1 * CB * cos(PHIM)) / (-0.3e1 + sin(PHIM));
    if FB <= 0
        DEPSPXXB = 0;
        DEPSPYYB = 0;
        DEPSPXYB = 0;
        break;
    else

        % calculate plastic strain increment:
        DEPSPXXB = ((((((((18 * DEPSXXB0 - 28 * DEPSYYB0) * E + 10 * EU * DEPSYYB0) * V - 28 * E *
DEPSXXB0 - 10 * EU * DEPSYYB0) * SIGXXBC ^ 2) + (((((-27 * DEPSXXB0 + 28 * DEPSYYB0) * E - EU *
DEPSYYB0) * V + 28 * E * DEPSXXB0 + EU * DEPSYYB0) * SIGYYBC) + 0.72e2 * DEPSXYB0 *
TAUXYBC * (V - 0.1e1 / 0.2e1) * E) * SIGXXBC + ((((9 * DEPSXXB0 - 19 * DEPSYYB0) * E + 10 * EU *
DEPSYYB0) * V - 19 * E * DEPSXXB0 - 10 * EU * DEPSYYB0) * SIGYYBC ^ 2) - 0.36e2 * DEPSXYB0 *
TAUXYBC * (V - 0.1e1 / 0.2e1) * E * SIGYYBC - 0.48e2 * TAUXYBC ^ 2 * (DEPSYYB0 * (E - EU) * V + E *
DEPSXXB0 + EU * DEPSYYB0)) * sin(PHIM) - 0.54e2 * (SIGXXBC - SIGYYBC / 0.2e1) * ((((DEPSXXB0 -
0.2e1 / 0.3e1 * DEPSYYB0) * E - (EU * DEPSYYB0) / 0.3e1) * V - 0.2e1 / 0.3e1 * E * DEPSXXB0 + (EU *
DEPSYYB0) / 0.3e1) * SIGXXBC + (((-DEPSXXB0 + DEPSYYB0 / 0.3e1) * E + 0.2e1 / 0.3e1 * EU *
DEPSYYB0) * V + (E * DEPSXXB0) / 0.3e1 - 0.2e1 / 0.3e1 * EU * DEPSYYB0) * SIGYYBC + 0.4e1 *
DEPSXYB0 * TAUXYBC * (V - 0.1e1 / 0.2e1) * E)) * sin(PSIM) - 0.54e2 * (SIGXXBC - SIGYYBC / 0.2e1) *
(sin(PHIM) - 0.3e1) * ((((DEPSXXB0 - 0.2e1 / 0.3e1 * DEPSYYB0) * E - (EU * DEPSYYB0) / 0.3e1) * V -
0.2e1 / 0.3e1 * E * DEPSXXB0 + (EU * DEPSYYB0) / 0.3e1) * SIGXXBC + (((-DEPSXXB0 + DEPSYYB0 /
0.3e1) * E + 0.2e1 / 0.3e1 * EU * DEPSYYB0) * V + (E * DEPSXXB0) / 0.3e1 - 0.2e1 / 0.3e1 * EU *
DEPSYYB0) * SIGYYBC + 0.4e1 * DEPSXYB0 * TAUXYBC * (V - 0.1e1 / 0.2e1) * E)) * sqrt((SIGXXBC ^ 2)
- (SIGXXBC * SIGYYBC) + (SIGYYBC ^ 2) + 0.3e1 * TAUXYBC ^ 2) - 0.12e2 * ((((((DEPSXXB0 - 0.8e1 /
0.3e1 * DEPSYYB0) * E + 0.5e1 / 0.3e1 * EU * DEPSYYB0) * V - 0.5e1 / 0.3e1 * EU * DEPSYYB0 - 0.8e1 /
0.3e1 * E * DEPSXXB0) * SIGXXBC + (((-DEPSXXB0 + 0.4e1 / 0.3e1 * DEPSYYB0) * E - (EU *
DEPSYYB0) / 0.3e1) * V + 0.4e1 / 0.3e1 * E * DEPSXXB0 + (EU * DEPSYYB0) / 0.3e1) * SIGYYBC +
0.4e1 * DEPSXYB0 * TAUXYBC * (V - 0.1e1 / 0.2e1) * E) * sin(PHIM) + ((((-3 * DEPSXXB0 + 2 *
DEPSYYB0) * E + EU * DEPSYYB0) * V + 2 * E * DEPSXXB0 - EU * DEPSYYB0) * SIGXXBC) + ((((3 *
DEPSXXB0 - DEPSYYB0) * E - 2 * EU * DEPSYYB0) * V - E * DEPSXXB0 + 2 * EU * DEPSYYB0) *
SIGYYBC) - 0.12e2 * DEPSXYB0 * TAUXYBC * (V - 0.1e1 / 0.2e1) * E) * sin(PSIM) + 0.6e1 * (SIGXXBC -
SIGYYBC / 0.2e1) * (DEPSYYB0 * (E - EU) * V + E * DEPSXXB0 + EU * DEPSYYB0) * sin(PHIM)) *
((SIGXXBC ^ 2) - (SIGXXBC * SIGYYBC) + (SIGYYBC ^ 2) + 0.3e1 * TAUXYBC ^ 2)) / ((((((24 * V ^ 2 * SB
+ (8 * E + 19 * EU + 12 * SB) * V - 28 * E - 19 * EU - 12 * SB) * SIGXXBC ^ 2) - 0.26e2 * (0.12e2 / 0.13e2 *
(V ^ 2) * SB + (E + 0.14e2 / 0.13e2 * EU + 0.6e1 / 0.13e2 * SB) * V - 0.14e2 / 0.13e2 * E - 0.14e2 / 0.13e2 *
EU - 0.6e1 / 0.13e2 * SB) * SIGYYBC * SIGXXBC + ((24 * V ^ 2 * SB + (-E + 28 * EU + 12 * SB) * V - 19 * E
- 28 * EU - 12 * SB) * SIGYYBC ^ 2) + 0.168e3 * (0.3e1 / 0.7e1 * (V ^ 2) * SB + (E + 0.2e1 / 0.7e1 * EU +
0.3e1 / 0.14e2 * SB) * V - 0.13e2 / 0.14e2 * E - 0.2e1 / 0.7e1 * EU - 0.3e1 / 0.14e2 * SB) * TAUXYBC ^ 2) *
sin(PHIM) + ((-72 * V ^ 2 * SB + (-72 * E - 9 * EU - 36 * SB) * V + 36 * E + 9 * EU + 36 * SB) * SIGXXBC ^
2) + 0.126e3 * (0.4e1 / 0.7e1 * (V ^ 2) * SB + (E + 0.2e1 / 0.7e1 * EU + 0.2e1 / 0.7e1 * SB) * V - 0.2e1 /
0.7e1 * E - 0.2e1 / 0.7e1 * EU - 0.2e1 / 0.7e1 * SB) * SIGYYBC * SIGXXBC + ((-72 * V ^ 2 * SB + (-45 * E -
36 * EU - 36 * SB) * V + 9 * E + 36 * EU + 36 * SB) * SIGYYBC ^ 2) - 0.648e3 * TAUXYBC ^ 2 * (V - 0.1e1 /
0.2e1) * ((V * SB) / 0.3e1 + E + SB / 0.3e1)) * sin(PSIM) - 0.72e2 * (((V ^ 2 * SB) + (E + EU / 0.8e1 + SB /
0.2e1) * V - E / 0.2e1 - EU / 0.8e1 - SB / 0.2e1) * (SIGXXBC ^ 2) - 0.7e1 / 0.4e1 * (0.4e1 / 0.7e1 * (V ^ 2) *
SB + (E + 0.2e1 / 0.7e1 * EU + 0.2e1 / 0.7e1 * SB) * V - 0.2e1 / 0.7e1 * E - 0.2e1 / 0.7e1 * EU - 0.2e1 /
0.7e1 * SB) * SIGYYBC * SIGXXBC + ((V ^ 2 * SB) + (0.5e1 / 0.8e1 * E + EU / 0.2e1 + SB / 0.2e1) * V - E /
0.8e1 - EU / 0.2e1 - SB / 0.2e1) * (SIGYYBC ^ 2) + 0.9e1 * TAUXYBC ^ 2 * (V - 0.1e1 / 0.2e1) * ((V * SB) /
0.3e1 + E + SB / 0.3e1)) * (sin(PHIM) - 0.3e1)) * sqrt((SIGXXBC ^ 2) - (SIGXXBC * SIGYYBC) + (SIGYYBC
^ 2) + 0.3e1 * TAUXYBC ^ 2) - 0.16e2 * ((SIGXXBC ^ 2) - (SIGXXBC * SIGYYBC) + (SIGYYBC ^ 2) + 0.3e1
* TAUXYBC ^ 2) * ((sin(PHIM) - 0.3e1 / 0.4e1) * sin(PSIM) - 0.9e1 / 0.4e1 * sin(PHIM)) * ((((E - EU) * V - 2 *
E + EU) * SIGXXBC) - 0.2e1 * (((E - EU) * V) - E / 0.2e1 + EU) * SIGYYBC));

        DEPSPYYB = ((((((((-9 * DEPSXXB0 - 10 * DEPSYYB0) * E + 19 * EU * DEPSYYB0) * V - 10 * E *
DEPSXXB0 - 19 * EU * DEPSYYB0) * SIGXXBC ^ 2) + (((((27 * DEPSXXB0 + DEPSYYB0) * E - 28 * EU *
DEPSYYB0) * V + E * DEPSXXB0 + 28 * EU * DEPSYYB0) * SIGYYBC) - 0.36e2 * DEPSXYB0 *
TAUXYBC * (V - 0.1e1 / 0.2e1) * E) * SIGXXBC + ((((-18 * DEPSXXB0 - 10 * DEPSYYB0) * E + 28 * EU *
DEPSYYB0) * V - 10 * E * DEPSXXB0 - 28 * EU * DEPSYYB0) * SIGYYBC ^ 2) + 0.72e2 * DEPSXYB0 *
TAUXYBC * (V - 0.1e1 / 0.2e1) * E * SIGYYBC - 0.48e2 * TAUXYBC ^ 2 * (DEPSYYB0 * (E - EU) * V + E *
DEPSXXB0 + EU * DEPSYYB0)) * sin(PHIM) + 0.27e2 * (SIGXXBC - 2 * SIGYYBC) * ((((DEPSXXB0 -
0.2e1 / 0.3e1 * DEPSYYB0) * E - (EU * DEPSYYB0) / 0.3e1) * V - 0.2e1 / 0.3e1 * E * DEPSXXB0 + (EU *
DEPSYYB0) / 0.3e1) * SIGXXBC + (((-DEPSXXB0 + DEPSYYB0 / 0.3e1) * E + 0.2e1 / 0.3e1 * EU *
DEPSYYB0) * V + (E * DEPSXXB0) / 0.3e1 - 0.2e1 / 0.3e1 * EU * DEPSYYB0) * SIGYYBC + 0.4e1 *
DEPSXYB0 * TAUXYBC * (V - 0.1e1 / 0.2e1) * E)) * sin(PSIM) + 0.27e2 * (SIGXXBC - 2 * SIGYYBC) *
(sin(PHIM) - 0.3e1) * ((((DEPSXXB0 - 0.2e1 / 0.3e1 * DEPSYYB0) * E - (EU * DEPSYYB0) / 0.3e1) * V -
0.2e1 / 0.3e1 * E * DEPSXXB0 + (EU * DEPSYYB0) / 0.3e1) * SIGXXBC + (((-DEPSXXB0 + DEPSYYB0 /
0.3e1) * E + 0.2e1 / 0.3e1 * EU * DEPSYYB0) * V + (E * DEPSXXB0) / 0.3e1 - 0.2e1 / 0.3e1 * EU *
DEPSYYB0) * SIGYYBC + 0.4e1 * DEPSXYB0 * TAUXYBC * (V - 0.1e1 / 0.2e1) * E)) * sqrt((SIGXXBC ^ 2)
- (SIGXXBC * SIGYYBC) + (SIGYYBC ^ 2) + 0.3e1 * TAUXYBC ^ 2) - 0.12e2 * ((SIGXXBC ^ 2) -
(SIGXXBC * SIGYYBC) + (SIGYYBC ^ 2) + 0.3e1 * TAUXYBC ^ 2) * ((((((DEPSXXB0 + DEPSYYB0 /
0.3e1) * E - 0.4e1 / 0.3e1 * EU * DEPSYYB0) * V + (E * DEPSXXB0) / 0.3e1 + 0.4e1 / 0.3e1 * EU *
DEPSYYB0) * SIGXXBC + (((-DEPSXXB0 - 0.5e1 / 0.3e1 * DEPSYYB0) * E + 0.8e1 / 0.3e1 * EU *
DEPSYYB0) * V - 0.5e1 / 0.3e1 * E * DEPSXXB0 - 0.8e1 / 0.3e1 * EU * DEPSYYB0) * SIGYYBC + 0.4e1 *
DEPSXYB0 * TAUXYBC * (V - 0.1e1 / 0.2e1) * E) * sin(PHIM) + ((((-3 * DEPSXXB0 + 2 * DEPSYYB0) * E +
EU * DEPSYYB0) * V + 2 * E * DEPSXXB0 - EU * DEPSYYB0) * SIGXXBC) + ((((3 * DEPSXXB0 -
DEPSYYB0) * E - 2 * EU * DEPSYYB0) * V - E * DEPSXXB0 + 2 * EU * DEPSYYB0) * SIGYYBC) - 0.12e2
* DEPSXYB0 * TAUXYBC * (V - 0.1e1 / 0.2e1) * E) * sin(PSIM) - 0.3e1 * (SIGXXBC - 2 * SIGYYBC) *
(DEPSYYB0 * (E - EU) * V + E * DEPSXXB0 + EU * DEPSYYB0) * sin(PHIM))) / ((((((24 * V ^ 2 * SB + (8 *
E + 19 * EU + 12 * SB) * V - 28 * E - 19 * EU - 12 * SB) * SIGXXBC ^ 2) - 0.26e2 * (0.12e2 / 0.13e2 * (V ^ 2)
* SB + (E + 0.14e2 / 0.13e2 * EU + 0.6e1 / 0.13e2 * SB) * V - 0.14e2 / 0.13e2 * E - 0.14e2 / 0.13e2 * EU -
0.6e1 / 0.13e2 * SB) * SIGYYBC * SIGXXBC + ((24 * V ^ 2 * SB + (-E + 28 * EU + 12 * SB) * V - 19 * E - 28
* EU - 12 * SB) * SIGYYBC ^ 2) + 0.168e3 * (0.3e1 / 0.7e1 * (V ^ 2) * SB + (E + 0.2e1 / 0.7e1 * EU + 0.3e1 /
0.14e2 * SB) * V - 0.13e2 / 0.14e2 * E - 0.2e1 / 0.7e1 * EU - 0.3e1 / 0.14e2 * SB) * TAUXYBC ^ 2) *
sin(PHIM) + ((-72 * V ^ 2 * SB + (-72 * E - 9 * EU - 36 * SB) * V + 36 * E + 9 * EU + 36 * SB) * SIGXXBC ^
2) + 0.126e3 * (0.4e1 / 0.7e1 * (V ^ 2) * SB + (E + 0.2e1 / 0.7e1 * EU + 0.2e1 / 0.7e1 * SB) * V - 0.2e1 /
0.7e1 * E - 0.2e1 / 0.7e1 * EU - 0.2e1 / 0.7e1 * SB) * SIGYYBC * SIGXXBC + ((-72 * V ^ 2 * SB + (-45 * E -
36 * EU - 36 * SB) * V + 9 * E + 36 * EU + 36 * SB) * SIGYYBC ^ 2) - 0.648e3 * TAUXYBC ^ 2 * (V - 0.1e1 /
0.2e1) * ((V * SB) / 0.3e1 + E + SB / 0.3e1)) * sin(PSIM) - 0.72e2 * (((V ^ 2 * SB) + (E + EU / 0.8e1 + SB /
0.2e1) * V - E / 0.2e1 - EU / 0.8e1 - SB / 0.2e1) * (SIGXXBC ^ 2) - 0.7e1 / 0.4e1 * (0.4e1 / 0.7e1 * (V ^ 2) *
SB + (E + 0.2e1 / 0.7e1 * EU + 0.2e1 / 0.7e1 * SB) * V - 0.2e1 / 0.7e1 * E - 0.2e1 / 0.7e1 * EU - 0.2e1 /
0.7e1 * SB) * SIGYYBC * SIGXXBC + ((V ^ 2 * SB) + (0.5e1 / 0.8e1 * E + EU / 0.2e1 + SB / 0.2e1) * V - E /
0.8e1 - EU / 0.2e1 - SB / 0.2e1) * (SIGYYBC ^ 2) + 0.9e1 * TAUXYBC ^ 2 * (V - 0.1e1 / 0.2e1) * ((V * SB) /
0.3e1 + E + SB / 0.3e1)) * (sin(PHIM) - 0.3e1)) * sqrt((SIGXXBC ^ 2) - (SIGXXBC * SIGYYBC) + (SIGYYBC
^ 2) + 0.3e1 * TAUXYBC ^ 2) - 0.16e2 * ((SIGXXBC ^ 2) - (SIGXXBC * SIGYYBC) + (SIGYYBC ^ 2) + 0.3e1
* TAUXYBC ^ 2) * ((sin(PHIM) - 0.3e1 / 0.4e1) * sin(PSIM) - 0.9e1 / 0.4e1 * sin(PHIM)) * ((((E - EU) * V - 2 *
E + EU) * SIGXXBC) - 0.2e1 * (((E - EU) * V) - E / 0.2e1 + EU) * SIGYYBC));

        DEPSPXYB = -0.54e2 * (((((DEPSXXB0 - 0.2e1 / 0.3e1 * DEPSYYB0) * E - EU * DEPSYYB0 /
0.3e1) * SIGXXBC + ((-DEPSXXB0 + DEPSYYB0 / 0.3e1) * E + 0.2e1 / 0.3e1 * EU * DEPSYYB0) *
SIGYYBC + 0.4e1 * TAUXYBC * E * DEPSXYB0) * V + (-0.2e1 / 0.3e1 * E * DEPSXXB0 + EU *
DEPSYYB0 / 0.3e1) * SIGXXBC + (E * DEPSXXB0 / 0.3e1 - 0.2e1 / 0.3e1 * EU * DEPSYYB0) * SIGYYBC -
0.2e1 * TAUXYBC * E * DEPSXYB0) * (sin(PHIM) - 0.3e1) * sqrt(SIGXXBC ^ 2 - SIGXXBC * SIGYYBC +
SIGYYBC ^ 2 + 0.3e1 * TAUXYBC ^ 2) + 0.4e1 / 0.3e1 * sin(PHIM) * (SIGXXBC ^ 2 - SIGXXBC * SIGYYBC
+ SIGYYBC ^ 2 + 0.3e1 * TAUXYBC ^ 2) * (DEPSYYB0 * (E - EU) * V + E * DEPSXXB0 + EU *
DEPSYYB0)) * (-0.3e1 + sin(PSIM)) * TAUXYBC / (0.16e2 * ((SIGXXBC - 0.2e1 * SIGYYBC) * (E - EU) * V
+ (-0.2e1 * E + EU) * SIGXXBC + SIGYYBC * (E - 0.2e1 * EU)) * ((sin(PSIM) - 0.9e1 / 0.4e1) * sin(PHIM) -
0.3e1 / 0.4e1 * sin(PSIM)) * sqrt(SIGXXBC ^ 2 - SIGXXBC * SIGYYBC + SIGYYBC ^ 2 + 0.3e1 * TAUXYBC
^ 2) + ((-0.24e2 * SB * (SIGXXBC ^ 2 - SIGXXBC * SIGYYBC + SIGYYBC ^ 2 + 0.3e1 * TAUXYBC ^ 2) * V
^ 2 + ((-0.8e1 * E - 0.19e2 * EU - 0.12e2 * SB) * SIGXXBC ^ 2 + 0.26e2 * SIGYYBC * (E + 0.14e2 / 0.13e2 *
EU + 0.6e1 / 0.13e2 * SB) * SIGXXBC + (E - 0.28e2 * EU - 0.12e2 * SB) * SIGYYBC ^ 2 - 0.168e3 *
TAUXYBC ^ 2 * (E + 0.2e1 / 0.7e1 * EU + 0.3e1 / 0.14e2 * SB)) * V + (0.28e2 * E + 0.19e2 * EU + 0.12e2 *
SB) * SIGXXBC ^ 2 - 0.28e2 * (E + EU + 0.3e1 / 0.7e1 * SB) * SIGYYBC * SIGXXBC + (0.19e2 * E +
0.28e2 * EU + 0.12e2 * SB) * SIGYYBC ^ 2 + 0.156e3 * (E + 0.4e1 / 0.13e2 * EU + 0.3e1 / 0.13e2 * SB) *
TAUXYBC ^ 2) * sin(PSIM) + 0.72e2 * SB * (SIGXXBC ^ 2 - SIGXXBC * SIGYYBC + SIGYYBC ^ 2 + 0.3e1
* TAUXYBC ^ 2) * V ^ 2 + ((0.72e2 * E + 0.9e1 * EU + 0.36e2 * SB) * SIGXXBC ^ 2 - 0.126e3 * SIGYYBC *
(E + 0.2e1 / 0.7e1 * EU + 0.2e1 / 0.7e1 * SB) * SIGXXBC + (0.45e2 * E + 0.36e2 * EU + 0.36e2 * SB) *
SIGYYBC ^ 2 + 0.648e3 * TAUXYBC ^ 2 * (E + SB / 0.6e1)) * V + (-0.36e2 * E - 0.9e1 * EU - 0.36e2 * SB) *
SIGXXBC ^ 2 + 0.36e2 * SIGYYBC * (E + EU + SB) * SIGXXBC + (-0.9e1 * E - 0.36e2 * EU - 0.36e2 * SB) *
SIGYYBC ^ 2 - 0.324e3 * TAUXYBC ^ 2 * (E + SB / 0.3e1)) * sin(PHIM) + 0.72e2 * (-0.3e1 + sin(PSIM)) *
(SB * (SIGXXBC ^ 2 - SIGXXBC * SIGYYBC + SIGYYBC ^ 2 + 0.3e1 * TAUXYBC ^ 2) * V ^ 2 + ((E + EU /
0.8e1 + SB / 0.2e1) * SIGXXBC ^ 2 - 0.7e1 / 0.4e1 * SIGYYBC * (E + 0.2e1 / 0.7e1 * EU + 0.2e1 / 0.7e1 *
SB) * SIGXXBC + (0.5e1 / 0.8e1 * E + EU / 0.2e1 + SB / 0.2e1) * SIGYYBC ^ 2 + 0.9e1 * TAUXYBC ^ 2 *
(E + SB / 0.6e1)) * V + (-E / 0.2e1 - EU / 0.8e1 - SB / 0.2e1) * SIGXXBC ^ 2 + SIGYYBC * (E + EU + SB) *
SIGXXBC / 0.2e1 + (-E / 0.8e1 - EU / 0.2e1 - SB / 0.2e1) * SIGYYBC ^ 2 - 0.9e1 / 0.2e1 * TAUXYBC ^ 2 *
(E + SB / 0.3e1))) * (SIGXXBC ^ 2 - SIGXXBC * SIGYYBC + SIGYYBC ^ 2 + 0.3e1 * TAUXYBC ^ 2) ^ (-
0.1e1 / 0.2e1);

        % recalculate softening modulus SUC:
        DLB = -0.18e2 * (((((DEPSXXB0 - 0.2e1 / 0.3e1 * DEPSYYB0) * E - EU * DEPSYYB0 / 0.3e1) *
SIGXXBC + ((-DEPSXXB0 + DEPSYYB0 / 0.3e1) * E + 0.2e1 / 0.3e1 * EU * DEPSYYB0) * SIGYYBC +
0.4e1 * TAUXYBC * E * DEPSXYB0) * V + (-0.2e1 / 0.3e1 * E * DEPSXXB0 + EU * DEPSYYB0 / 0.3e1) *
SIGXXBC + (E * DEPSXXB0 / 0.3e1 - 0.2e1 / 0.3e1 * EU * DEPSYYB0) * SIGYYBC - 0.2e1 * TAUXYBC *
E * DEPSXYB0) * (sin(PHIM) - 0.3e1) * sqrt(SIGXXBC ^ 2 - SIGXXBC * SIGYYBC + SIGYYBC ^ 2 + 0.3e1
* TAUXYBC ^ 2) + 0.4e1 / 0.3e1 * sin(PHIM) * (SIGXXBC ^ 2 - SIGXXBC * SIGYYBC + SIGYYBC ^ 2 +
0.3e1 * TAUXYBC ^ 2) * (DEPSYYB0 * (E - EU) * V + E * DEPSXXB0 + EU * DEPSYYB0)) * (-0.3e1 +
sin(PSIM)) / (0.16e2 * ((SIGXXBC - 0.2e1 * SIGYYBC) * (E - EU) * V + (-0.2e1 * E + EU) * SIGXXBC +
SIGYYBC * (E - 0.2e1 * EU)) * ((sin(PSIM) - 0.9e1 / 0.4e1) * sin(PHIM) - 0.3e1 / 0.4e1 * sin(PSIM)) *
sqrt(SIGXXBC ^ 2 - SIGXXBC * SIGYYBC + SIGYYBC ^ 2 + 0.3e1 * TAUXYBC ^ 2) + ((-0.24e2 * SB *
(SIGXXBC ^ 2 - SIGXXBC * SIGYYBC + SIGYYBC ^ 2 + 0.3e1 * TAUXYBC ^ 2) * V ^ 2 + ((-0.8e1 * E -
0.19e2 * EU - 0.12e2 * SB) * SIGXXBC ^ 2 + 0.26e2 * SIGYYBC * (E + 0.14e2 / 0.13e2 * EU + 0.6e1 /
0.13e2 * SB) * SIGXXBC + (E - 0.28e2 * EU - 0.12e2 * SB) * SIGYYBC ^ 2 - 0.168e3 * TAUXYBC ^ 2 * (E +
0.2e1 / 0.7e1 * EU + 0.3e1 / 0.14e2 * SB)) * V + (0.28e2 * E + 0.19e2 * EU + 0.12e2 * SB) * SIGXXBC ^ 2 -
0.28e2 * (E + EU + 0.3e1 / 0.7e1 * SB) * SIGYYBC * SIGXXBC + (0.19e2 * E + 0.28e2 * EU + 0.12e2 * SB)
* SIGYYBC ^ 2 + 0.156e3 * (E + 0.4e1 / 0.13e2 * EU + 0.3e1 / 0.13e2 * SB) * TAUXYBC ^ 2) * sin(PSIM) +
0.72e2 * SB * (SIGXXBC ^ 2 - SIGXXBC * SIGYYBC + SIGYYBC ^ 2 + 0.3e1 * TAUXYBC ^ 2) * V ^ 2 +
((0.72e2 * E + 0.9e1 * EU + 0.36e2 * SB) * SIGXXBC ^ 2 - 0.126e3 * SIGYYBC * (E + 0.2e1 / 0.7e1 * EU +
0.2e1 / 0.7e1 * SB) * SIGXXBC + (0.45e2 * E + 0.36e2 * EU + 0.36e2 * SB) * SIGYYBC ^ 2 + 0.648e3 *
TAUXYBC ^ 2 * (E + SB / 0.6e1)) * V + (-0.36e2 * E - 0.9e1 * EU - 0.36e2 * SB) * SIGXXBC ^ 2 + 0.36e2 *
SIGYYBC * (E + EU + SB) * SIGXXBC + (-0.9e1 * E - 0.36e2 * EU - 0.36e2 * SB) * SIGYYBC ^ 2 - 0.324e3
* TAUXYBC ^ 2 * (E + SB / 0.3e1)) * sin(PHIM) + 0.72e2 * (-0.3e1 + sin(PSIM)) * (SB * (SIGXXBC ^ 2 -
SIGXXBC * SIGYYBC + SIGYYBC ^ 2 + 0.3e1 * TAUXYBC ^ 2) * V ^ 2 + ((E + EU / 0.8e1 + SB / 0.2e1) *
SIGXXBC ^ 2 - 0.7e1 / 0.4e1 * SIGYYBC * (E + 0.2e1 / 0.7e1 * EU + 0.2e1 / 0.7e1 * SB) * SIGXXBC +
(0.5e1 / 0.8e1 * E + EU / 0.2e1 + SB / 0.2e1) * SIGYYBC ^ 2 + 0.9e1 * TAUXYBC ^ 2 * (E + SB / 0.6e1)) * V
+ (-E / 0.2e1 - EU / 0.8e1 - SB / 0.2e1) * SIGXXBC ^ 2 + SIGYYBC * (E + EU + SB) * SIGXXBC / 0.2e1 + (-
E / 0.8e1 - EU / 0.2e1 - SB / 0.2e1) * SIGYYBC ^ 2 - 0.9e1 / 0.2e1 * TAUXYBC ^ 2 * (E + SB / 0.3e1)));
```

```
        FKB = -0.3e1 / (0.3e1 - sin(PHIM)) * (0.1e1 - sin(PHIM)) * KCB;
        SBC = -FKB* DKB / DLB ;
        if abs(SBC-SB) < TOR2
            break;
        else
            SB = SBC;
        end
    end
end

EPSPXXU = EPSPXXU + DEPSPXXU;
EPSPYYU = EPSPYYU + DEPSPYYU;
% vertical elastic stress should always be positive:
EPSXXUE = (SIGXXUE-VU*SIGYYUE)/EU;
EPSYYUE = (SIGYYUE-VU*SIGXXUE)/EU;
if EPSPYYU > EPSYYUE
    if EPSPXXU > EPSXXUE
        EPSPXXU = EPSXXUE;
    else
    end
    EPSPYYU = EPSYYUE;
else
end

EPSPXXH = EPSPXXH + DEPSPXXH;
EPSPYYH = EPSPYYH + DEPSPYYH;
% elastic stress should always be positive:
EPSXXHE = (SIGXXHE-V*SIGYYHE)/E;
EPSYYHE = (SIGYYHE-V*SIGXXHE)/E;
if EPSPYYH > EPSYYHE
    if EPSPXXH > EPSXXHE
        EPSPXXH = EPSXXHE;
    else
    end
    EPSPYYH = EPSYYHE;
else
end

EPSPXXB = EPSPXXB + DEPSPXXB;
EPSPYYB = EPSPYYB + DEPSPYYB;
EPSPXYB = EPSPXYB + DEPSPXYB;
% vertical elastic predicted stress should always be positive:
EPSXXBE = (SIGXXBE-V*SIGYYBE)/E;
EPSYYBE = (SIGYYBE-V*SIGXXBE)/E;
EPSXYBE = TAUXYBE/(2*GB);
if EPSPYYB > EPSYYBE
    if EPSPXXB > EPSXXBE
        if EPSPXYB > EPSXYBE
            EPSPXYB = EPSXYBE;
        else
        end
        EPSPXXB = EPSXXBE;
    else
    end
    EPSPYYB = EPSYYBE;
```

```
  else
  end

  % Stresses with plastic corrector of each component in x direction:
  SIGXXUP = (EU*(1-VU)*(EPSXXUE-EPSPXXU))/((1+VU)*(1-2*VU))+(EU*(VU)*(EPSYYUE-
EPSPYYU))/((1+VU)*(1-2*VU));

  SIGXXHP = (EH*(1-V)*(EPSXXHE-EPSPXXH))/((1+V)*(1-2*V))+(EH*(V)*(EPSYYHE-
EPSPYYH))/((1+V)*(1-2*V));

  SIGXXCP = RB*(EC*(1-V))*(EPSXXBE-EPSPXXB)/((1+V)*(1-2*V))+(EH*(V)*(EPSYYBE-
EPSPYYB))/((1+V)*(1-2*V))/RC;

  % Stresses with plastic corrector in y direction:
  SIGYYUP = (EU*(1-VU)*(EPSYYUE-EPSPYYU))/((1+VU)*(1-2*VU))+(EU*(VU)*(EPSXXUE-
EPSPXXU))/((1+VU)*(1-2*VU));

  SIGYYHP = (EH*(1-V)*(EPSYYHE-EPSPYYH))/((1+V)*(1-2*V))+(EH*(V)*(EPSXXHE-
EPSPXXH))/((1+V)*(1-2*V));

  SIGYYCE = EC * (RH * EPSYYHE / RC);

  % shear stresses with plastic corrector of each component in shear direction:
  TAUXYBP = 2*GB*(EPSXYBE-EPSPXYB);

  if EPSYY0TC < 0
    % damage factor:
    while DH < 1 & DU < 1 & DC < 1 & DB < 1
      % damage model
      %find maximum stress between stress at n step and intial maximum value
      SXH = max(SIGXXHP,SIGTM); % head joint
      SXU = max(SIGXXUP,SIGTU); % brick unit
      SXC = max(-SIGXXCP,SIGTM); % cross joint
      TXYB = max(abs(TAUXYBP),SIGS); % bed joint

      % Calculate damage factor from internal stresses
      DHC = 1-(SIGTM*exp(ATM*(1-(SXH/SIGTM)))/SXH); % DH should not increasing
      DUC = 1-(SIGTU*exp(ATU*(1-(SXU/SIGTU)))/SXU); % once DH tend to increased, brick is
damaged
      DBC = 1-(SIGS*exp(ASB*(1-(TXYB/SIGS)))/TXYB);
      if abs(DHC/DH) < 1
        DHC = DH;
        SWC = 1;
      else
      end
      if abs(DUC/DU) < 1
        DUC = DU;
      else
      end
      if abs(DBC/DB) < 1
        DBC = DB;
      else
      end

      if SIGXXCP < 0
        DCC = 1-(SIGTM*exp(ATM*(1-(SXC/SIGTM)))/SXC);
      else
```

```
         DCC = (DBC+DHC)/2;% For compressive deformed cell, sigxxc is in compression direction
      end

      % Verification of damage factor
      % Since damage factor will influence stress itself
      % damage factor should be verificated together
      if DHC >= 0 & DUC >=0 & DCC >=0 & DBC >= 0
         if abs(DHC-DH) < TOR
            if abs(DUC-DU) < TOR
               if abs(DCC-DC) < TOR
                  if abs(DBC-DB) < TOR
                     break;
                  else
                     DB = DBC;
                  end
               else
                  DC = DCC;
               end
            else
               DU = DUC;
            end
         else
            DH = DHC;
         end
      else
         break;
      end
   end
   SIGYY0C = -(RH*SIGYYHP + C2*RU*SIGYYUP)/(C2+1);

   % shear behaviour
   RBXY = 1 - DBXY; % with vertical compression load
   while DBXY < 1
      TAUXYBXY = 2 * GB * EPSXY0 * (C3+1);
      CC = CH;
      TXYBXY = max(abs(TAUXYBXY),CC);
      ASBXY = (((GII*GB)/(LS*CC^2))-(1/2))^(-1);
      DBCXY =  1-(CC*exp(ASBXY*(1-(TXYBXY/CC)))/TXYBXY);
      if DBCXY >= 0
         if abs(DBCXY-DBXY) < TOR
            break;
         else
            DBXY = DBCXY;
         end
      else
         break;
      end
   end
   TAUXY0 = - SIGYY0 * tan(PHIM)+ RBXY * TAUXYBXY;
else
   while DH < 1 & DU < 1 & DC < 1 & DB < 1
      % damage model for vertical tension cracking
      %find maximum stress between stress at n step and intial maximum value
      SYH = max(SIGYYHE,SIGTM); % head joint
      SYU = max(SIGYYUE,SIGTU); % brick unit
      SYC = max(-SIGYYCE,SIGTM); % cross joint
```

```
        SYB = max(abs(SIGYYBE),SIGTM); % bed joint

        % Calculate damage factor from internal stresses
        DHC = 1-(SIGTM*exp(ATM*(1-(SYH/SIGTM)))/SYH); % DH should not increasing
        DUC = 1-(SIGTU*exp(ATU*(1-(SYU/SIGTU)))/SYU); % once DH tend to increased, brick is damaged
        DBC = 1-(SIGTM*exp(ATM*(1-(SYB/SIGTM)))/SYB);
        if abs(DHC/DH) < 1
            DHC = DH;
            SWC = 1;
        else
        end
        if abs(DUC/DU) < 1
            DUC = DU;
        else
        end
        if abs(DBC/DB) < 1
            DBC = DB;
        else
        end

        if SIGYYCE < 0
            DCC = 1-(SIGTM*exp(ATM*(1-(SYC/SIGTM)))/SYC);
        else
            DCC = (DBC+DHC)/2;% For compressive deformed cell, sigxxc is in compression direction
        end

        % Verification of damage factor in y direction
        % Since damage factor will influence stress itself
        % damage factor should be verificated together
        if DHC >= 0 & DUC >=0 & DCC >=0 & DBC >= 0
            if abs(DHC-DH) < TOR
                if abs(DUC-DU) < TOR
                    if abs(DCC-DC) < TOR
                        if abs(DBC-DB) < TOR
                            break;
                        else
                            DB = DBC;
                        end
                    else
                        DC = DCC;
                    end
                else
                    DU = DUC;
                end
            else
                DH = DHC;
            end
        else
            break;
        end
    end
    SIGYY0C = (RH*SIGYYHE + C2*RU*SIGYYUE)/(C2+1);

    % shear behaviour
    RBXY = 1 - DBXY; % shear sliding failure pattern
    while DBXY < 1
```

```
         TAUXYBXY = 2 * GB * EPSXY0 * (C3+1);
         CC = SIGS;
         TXYBXY = max(abs(TAUXYBXY),CC);
         ASBXY = (((GII*GB)/(LS*CC^2))-(1/2))^(-1);
         DBCXY =  1-(CC*exp(ASBXY*(1-(TXYBXY/CC)))/TXYBXY);
         if DBCXY >= 0
            if abs(DBCXY-DBXY) < TOR
               break;
            else
               DBXY = DBCXY;
            end
         else
            break;
         end
      end
      TAUXY0 = RBXY * TAUXYBXY;
   end

   % undamaged stresses of basic cell in y direction:
   if SWC == 1
      SIGYY0 = SIGYY0;
   else
      SIGYY0 = SIGYY0C;
   end

   % tension behaviour (without shear loading): brick tension cracking
   EPSXX0T = abs(EPSXX0TC);
   DEPSXX0T = abs(DEPSXX0TC);

   % horizontal stress in components
   SIGXXHX = -0.4e1 * E * (C2 + 1) * (-(C1 * GB * RBX * RUX * (V * VU - 1) * C2 ^ 2) / 0.4e1 + (((C1 * (V *
VU - 1) * RUX + (-VU ^ 2 + 1) * RHX) * GB * RBX) / 0.4e1 + (C1 ^ 2 * C3 * E * RUX ^ 2)) * C2 + (GB * RBX
* ((VU ^ 2) / 0.4e1 - 0.1e1 / 0.4e1) + (C1 * C3 * E * RUX)) * RHX) * EPSXX0T / ((C1 * GB * RBX * RUX * (V
- 1) * (V + 1) * C2 ^ 3) - 0.4e1 * RHX * (GB * RBX * (-(VU ^ 2) / 0.4e1 + 0.1e1 / 0.4e1) + (C1 * C3 * E *
RUX)) * (C2 ^ 2) + ((-C1 * GB * RBX * RUX * (V - 1) * (V + 1) + 4 * C3 * E * (RUX ^ 2 * (V ^ 2 - 1) * C1 ^ 2 -
2 * C1 * RHX * RUX * V * VU + RHX ^ 2 * (VU ^ 2 - 1))) * C2) - 0.4e1 * (GB * RBX * ((VU ^ 2) / 0.4e1 - 0.1e1
/ 0.4e1) + (C1 * C3 * E * RUX)) * RHX);

   SIGXXUX = -0.4e1 * E * C1 * (C2 + 1) * EPSXX0T * (-(C1 * GB * RBX * RUX * (V - 1) * (V + 1) * C2 ^ 2) /
0.4e1 + ((GB * (C1 * RUX * V ^ 2 - RHX * V * VU - RUX * C1 + RHX) * RBX) / 0.4e1 + (C1 * RHX * C3 * E *
RUX)) * C2 + ((C3 * E * RHX) + (GB * RBX * (V * VU - 1)) / 0.4e1) * RHX) / ((C1 * GB * RBX * RUX * (V - 1)
* (V + 1) * C2 ^ 3) - 0.4e1 * RHX * (GB * RBX * (-(VU ^ 2) / 0.4e1 + 0.1e1 / 0.4e1) + (C1 * C3 * E * RUX)) *
(C2 ^ 2) + ((-C1 * GB * RBX * RUX * (V - 1) * (V + 1) + 4 * E * (RHX ^ 2 * (VU ^ 2 - 1) - 2 * C1 * RHX * RUX
* V * VU + RUX ^ 2 * C1 ^ 2 * (V - 1) * (V + 1)) * C3) * C2) - 0.4e1 * (GB * RBX * ((VU ^ 2) / 0.4e1 - 0.1e1 /
0.4e1) + (C1 * C3 * E * RUX)) * RHX);

   SIGXXCX = -E * (C2 + 1) * EPSXX0T * (-(((V * C1 * (V - VU) * RUX) + (RCX * (VU - 1) * (VU + 1) * (C2 +
1)) / 0.2e1) * RHX + (RUX * C1 * C2 * RCX * (V - 1) * (V + 1) * (C2 + 1)) / 0.2e1) * (C2 - 1) * GB * RBX ^ 2 /
0.2e1 + (0.2e1 * E * ((RUX * V * C1 * VU) - (C2 * RCX * (VU - 1) * (VU + 1)) / 0.2e1) * C3 * RHX ^ 2 - 0.2e1
* (E * RUX * (V ^ 2) * C1 * C3 - ((-V * (V - VU) * GB / 0.8e1 + C3 * E / 0.2e1) * (C2 ^ 2) + V * ((-VU / 0.4e1 +
V / 0.4e1) * GB + E * VU * C3) * C2 - V * (V - VU) * GB / 0.8e1 + C3 * E / 0.2e1) * RCX) * C1 * RUX * RHX -
E * (RUX ^ 2) * (C1 ^ 2) * C2 * C3 * RCX * (V - 1) * (V + 1)) * RBX + E * RHX * RUX * V * C1 * C3 * RCX *
(C2 - 1) * (-(C1 * RUX * V) + RHX * VU)) / (((C2 - 1) * RCX) + 0.2e1 * RBX) / ((C2 + 1) * (C2 - 1) * GB *
((RUX * (V - 1) * (V + 1) * C1 * C2) + RHX * (VU ^ 2 - 1)) * RBX / 0.4e1 + (C2 * (VU ^ 2 - 1) * RHX ^ 2 -
0.2e1 * RHX * ((VU * V * C2) + (C2 ^ 2) / 0.2e1 + 0.1e1 / 0.2e1) * RUX * C1 + (RUX ^ 2 * C2 * (V - 1) * (V +
1) * C1 ^ 2)) * E * C3) / RCX;
```

```
   SIGXXBX = RC * SIGXXCX / RB;

   % vertical stresses in components
   SIGYYUX = 0.4e1 * EPSXX0T * (GB * RBX * (V - VU) * C2 / 0.4e1 - GB * RBX * (V - VU) / 0.4e1 + E * C3 * (C1 * RUX * V - RHX * VU)) * RHX * C1 * E * (C2 + 0.1e1) / (C1 * GB * RBX * RUX * (V - 0.1e1) * (V + 0.1e1) * C2 ^ 3 - 0.4e1 * RHX * (GB * RBX * (-VU ^ 2 / 0.4e1 + 0.1e1 / 0.4e1) + C1 * C3 * E * RUX) * C2 ^ 2 + (-C1 * GB * RBX * RUX * (V - 0.1e1) * (V + 0.1e1) + 0.4e1 * (RHX ^ 2 * (VU ^ 2 - 0.1e1) - 0.2e1 * C1 * RHX * RUX * V * VU + RUX ^ 2 * C1 ^ 2 * (V - 0.1e1) * (V + 0.1e1)) * C3 * E) * C2 - 0.4e1 * (GB * RBX * (VU ^ 2 / 0.4e1 - 0.1e1 / 0.4e1) + C1 * C3 * E * RUX) * RHX);

   SIGYYHX = -0.4e1 * C2 * EPSXX0T * C1 * RUX * E * (GB * RBX * (V - VU) * C2 / 0.4e1 - GB * RBX * (V - VU) / 0.4e1 + E * C3 * (C1 * RUX * V - RHX * VU)) * (C2 + 0.1e1) / (C1 * GB * RBX * RUX * (V - 0.1e1) * (V + 0.1e1) * C2 ^ 3 - 0.4e1 * RHX * (GB * RBX * (-VU ^ 2 / 0.4e1 + 0.1e1 / 0.4e1) + C1 * C3 * E * RUX) * C2 ^ 2 + (-C1 * GB * RBX * RUX * (V - 0.1e1) * (V + 0.1e1) + 0.4e1 * E * C3 * (RUX ^ 2 * (V ^ 2 - 0.1e1) * C1 ^ 2 - 0.2e1 * C1 * RHX * RUX * V * VU + RHX ^ 2 * (VU ^ 2 - 0.1e1))) * C2 - 0.4e1 * (GB * RBX * (VU ^ 2 / 0.4e1 - 0.1e1 / 0.4e1) + C1 * C3 * E * RUX) * RHX);

   SIGYYBX = EPSXX0T * RHX * C1 * RUX * E * (GB * RBX * (V - VU) * C2 / 0.4e1 - GB * RBX * (V - VU) / 0.4e1 + E * C3 * (C1 * RUX * V - RHX * VU)) * (C2 + 0.1e1) / (C1 * GB * RBX * RUX * (V - 0.1e1) * (V + 0.1e1) * C2 ^ 3 / 0.4e1 - RHX * (GB * RBX * (-VU ^ 2 / 0.4e1 + 0.1e1 / 0.4e1) + C1 * C3 * E * RUX) * C2 ^ 2 + (-C1 * GB * RBX * RUX * (V - 0.1e1) * (V + 0.1e1) / 0.4e1 + (RHX ^ 2 * (VU ^ 2 - 0.1e1) - 0.2e1 * C1 * RHX * RUX * V * VU + RUX ^ 2 * C1 ^ 2 * (V - 0.1e1) * (V + 0.1e1)) * C3 * E) * C2 - (GB * RBX * (VU ^ 2 / 0.4e1 - 0.1e1 / 0.4e1) + C1 * C3 * E * RUX) * RHX) / RBX;

   EPSYYBX = (SIGYYBX - V * SIGXXBX) / EB;

   SIGYYCX = EC * (RB * EPSYYBX / RC);

   % shear stress between bed joint and brick unit
   TAUXYBX =  0.2e1 * (RUX * C1 * ((-V * VU + 1) * RHX + RUX * C1 * (V - 1) * (V + 1)) * C2 + RHX * (C1 * (V * VU - 1) * RUX + (-VU ^ 2 + 1) * RHX)) * E * (C2 + 1) * C3 * EPSXX0T * GB / ((C1 * GB * RBX * RUX * (V - 1) * (V + 1) * C2 ^ 3) - 0.4e1 * RHX * ((C1 * C3 * E * RUX) - (GB * RBX * (VU - 1) * (VU + 1)) / 0.4e1) * (C2 ^ 2) + (((4 * VU ^ 2 - 4) * E * C3 * RHX ^ 2) - (8 * E * RUX * V * C1 * C3 * VU * RHX) + 0.4e1 * RUX * C1 * (V - 1) * ((C1 * C3 * E * RUX) - (GB * RBX) / 0.4e1) * (V + 1)) * C2 - 0.4e1 * ((C1 * C3 * E * RUX) + (GB * RBX * (VU - 1) * (VU + 1)) / 0.4e1) * RHX);

   if EPSXX0TC >= 0
      % inner loop: find damage factor
      while DHX < 1 & DUX < 1 & DCX < 1 & DBX < 1
         % effective stresses of each component based on damage factor
         SXHX = max(SIGXXHX,SIGTM);
         SXUX = max(SIGXXUX,SIGTU);
         SXCX = max(SIGXXCX,SIGTM);
         TXYBX = max(abs(TAUXYBX),SIGS);

         % Calculate damage factor from internal stresses
         DUCX = 1-(SIGTU*exp(ATU*(1-(SXUX/SIGTU)))/SXUX);
         DHCX = 1-(SIGTM*exp(ATM*(1-(SXHX/SIGTM)))/SXHX);
         DCCX = 1-(SIGTM*exp(ATM*(1-(SXCX/SIGTM)))/SXCX);
         DBCX = 1-(SIGS*exp(ASB*(1-(TXYBX/SIGS)))/TXYBX);
         if abs(DHCX/DHX) < 1
            if abs(DUCX/DUX) < 1
               if abs(DBCX/DBX) < 1
                  if abs(DCCX/DCX) < 1
                     DCCX = DCX;
                  else
                  end
                  DBCX = DBX;
               else
               end
               DUCX = DUX;
            else
            end
            DHCX = DHX;
         else
         end

         % Verification of damage factor
         % Since damage factor will influence stress itself
         % damage factor should be verificated together
         if DHCX >= 0 & DUCX >=0 & DCCX >=0 & DBCX >= 0
            if abs(DHCX-DHX) < TOR
               if abs(DUCX-DUX) < TOR
                  if abs(DCCX-DCX) < TOR
                     if abs(DBCX-DBX) < TOR
                        break;
                     else
                        DBX = DBCX;
                     end
                  else
                     DCX = DCCX;
                  end
               else
                  DUX = DUCX;
               end
            else
               DHX = DHCX;
            end
         else
            break;
         end
      end
      % total horizontal undamaged stress of cell
      SIGXX0 = (RHX*SIGXXHX*C3 + 2*RCX*SIGXXCX + C3*(RUX*SIGXXUX + RBX*TAUXYBX*(C2 -
1)/(2*C3)))/(2*(C3 + 1));
   else
      SIGXX0 = 0;
   end

   % Stiffness of basic cell:
   % K11, K12, K13=0, K21, K22, K23=0, 0, K32, K33

   % record value:
   a = [a,SIGYY0];
   b = [b,SIGXX0];
   d = [d,TAUXY0];
   e = [e,RH*SIGXXHP+RHX*SIGXXHX];
   f = [f,RU*SIGXXUP+RUX*SIGXXUX];
   g = [g,RB*TAUXYBP+RBX*TAUXYBX+RBXY*TAUXYBXY];
   h = [h,RC*SIGXXCP+RCX*SIGXXCX];
end
```

Appendix F: dcf. File (Single Element Model)

```
*FILOS
INITIA
*INPUT
*FORTRAN
USE "model1.dll"
*NONLIN LABEL="Structural nonlinear"
  TYPE PHYSIC PLASTI MITERA 100
  BEGIN EXECUT
    LOAD STEPS EXPLIC SIZES 1e-5(300)
    BEGIN ITERAT
      MAXITE 500
      METHOD NEWTON
      BEGIN CONVER
        FORCE OFF
        DISPLA OFF
        ENERGY
      END CONVER
    END ITERAT
  END EXECUT
  SOLVE PARDIS
  BEGIN OUTPUT
    TEXT "Output"
    BINARY
    SELECT STEPS ALL /
    DISPLA TOTAL TRANSL GLOBAL
    BEGIN STRAIN
      BEGIN TOTAL
        BEGIN GREEN
          BEGIN GLOBAL
            INTPNT
            ERROR OFF
          END GLOBAL
        END GREEN
      END TOTAL
    END STRAIN
    STRESS TOTAL CAUCHY GLOBAL INTPNT
    FORCE REACTI TRANSL GLOBAL
  END OUTPUT
*END
```