FINITE ELEMENT AND RELIABILITY STUDY OF IN-PLANE BEHAVIOUR OF CONCRETE MASONRY INFILLED RC FRAMES

by

Reza Rahimi

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
August 2020

I dedicate this dissertation to all the 176 wonderful people we lost in the crash of the flight PS752 in January 2020, especially Dr. Sharieh Faghihi.

Table of Contents

List of Tables

List of Figures

Abstract

This research provides a finite element and reliability study of the in-plane behaviour of masonry infilled reinforced concrete frames. In the first phase of the research, a reliability analysis was conducted to advance the understanding of the effect of the randomness and uncertainty of masonry materials on the lateral resistance of masonry infill walls from a probabilistic approach. A computationally efficient and robust finite element model was developed in OpenSees and validated using the experimental results obtained in the same research group. Coupled with the Monte-Carlo simulation techniques, the model was used to estimate the failure probability of infilled frames with a spatially varying random field of masonry compressive strength. After 1000 simulations, the resulted failure probability distribution of the infilled frames led to a new recommendation on the masonry resistance reduction factor. Despite the efficiency of the OpenSees model, running simulations of the random fields on the infilled frames on an advanced work station, with 40 Central Processing Units (CPUs) and 128 GBs of memory, took three months to complete. It became evident that a better performing computing technology was needed for any meaningful reliability analysis of this magnitude. This motivated the second phase of the research.

In the second phase, an open-source and modular finite element library for estimating the behaviour of infilled frames was developed. The main feature of the library was to be able to significantly accelerate the numerical simulations for structural applications in general and with a focus on masonry infilled RC frames. The specific algorithms, encoded in C++, were developed for the library to be able to run on either central processing units or graphical processing units (GPUs). The library adopted an advanced smeared crack modeling technique, the Distressed Stress Field Method (DSFM), for modelling of the masonry infilled RC frames. A comparison with the experimental results showed that the DSFM was successfully incorporated in the analysis of masonry infilled RC frames. The performance of the developed library in accelerating the model run speed was demonstrated through comparisons of run speed using a CPU and several commercially available GPUs. It was shown that GPU devices with adequate memory space can lead to significant model run speedup compared with a CPU. The degree of speed-up was highly dependent on the number of elements used in the finite element model and the number of processing core available in the parallel architecture. The greater the number of elements used in the model, the greater rate of acceleration will be achieved.

List of Abbreviations and Symbols Used

Abbreviations

API Application Program Interface

ASA American Standards Association

ATC Applied Technology Council

CMU Concrete Masonry Unit

CPU Central Processing Unit

CSA Canadian Standard Association

CSR Compressed Sparse Row

CUDA Compute Unified Device Architecture

DoD Data-oriented Design

DOF Degree of Freedom

DRAM Dynamic Random-Access Memory

DSFM Distributed Stress Field method

ECC Error-Correcting Code

FORM First Order Reliability Method

FOSM First Order Second Moment

GB Gigabyte

Gbps Gigabit per second

GHz Gigahertz

GPU Graphical Processing Unit

HBM2 High Bandwidth Memory 2

LAS Local Average Subdivision

LRFD Load and Resistance Factor Design

LSD Limit States Design

LVDT Linear Variable Displacement Transducer

MSJC Masonry Standards Joint Committee

NBCC National Building Code of Canada

NNZ Number of Non-Zero

NSERC Natural Sciences and Engineering Research Council

OOP Object Oriented Programming

RAM Random Access Memory

RC Reinforced Concrete

RFEM Random Finite Element method

SGRAM Synchronous Graphics Random Access Memory

SIMD Single Instruction Multiple Data

SM Streaming multiprocessor

STL Standard Template Library

TMS The Masonry Society

VPFEM Vectorized and Parallelized Finite Element Method

WSD Working Stress Design

Symbols

α Angle between direction of mortar thickness and global x direction

α_h Contact length between infill wall and frame column

α_l Contact length between infill wall and frame beam

b Strain hardening ratio

$\mathbf{B}$ Strain matrix

β Reliability index

β_{c_d} Factor to account for softening effect on compression capacity of concrete

β_j friction angle in mortar interface

β_m Factor to account for effect of biaxial stress field in concrete masonry units

β_s Reduction factor to account for bond-slip effect

c Number of CUDA cores on GPU device (Chapter 3)

c Concrete material (Chapter 5)

c_j Cohesion constant in mortar interface

c_{t_c} Degree of stiffening due to reinforcement in concrete

χ Factor to account for the effect of direction of the compressive stress in a masonry member relative to the direction used in the masonry prism compression test

D Material stiffness matrix

d_b Rebar diameter

Δ_a Tip deflection (numerically calculated)

$[D_c]$ Concrete material stiffness matrix

δ^s Shear slip

Δ_y Yield deformation of concrete frame

Δ_n Tip deflection (analytically calculated)

$[D_{s_k}]$ Steel material stiffness matrix

E_f Bounding frame stiffness

E_m Modulus of elasticity of masonry material

E_s Elastic modulus of steel

ϵ_{s_y} Yielding strain of steel

ϵ_{c_1} Principal strain in concrete

ϵ_{c_2} Principal strain in concrete

ϵ_{c_0} Corresponding strain to concrete compressive strength

$\epsilon_{c_{ts}}$ Ultimate tensile strain in concrete

ϵ_{j_0} Strain at compressive capacity of mortar interface

ϵ_{j_1} Principal strain in Mortar interface

ϵ_{j_2} Principal strain in Mortar interface

ϵ_{m_1} Principal strain in masonry

ϵ_{m_2} Principal strain in masonry

ϵ_{m_p} Peak Strain in compressive constitutive model for masonry material

ϵ_0 Corresponding strain at compressive strength of mortar interface

$\epsilon_{j_{tu}}$ Ultimate tensile strain in mortar interface

$[\epsilon_c^n]$ Concrete net strain

ϵ_{s_k} Strain in steel rebar

ϵ_{t_u} Ultimate tensile strength

ϵ_x Normal strains in x direction

ϵ_y Normal strains in y direction

η Natural coordinate

γ_{xy} Shear strain

f_c Concrete stress field in global coordinates

f_c' Concrete compressive strength

F_{exp} Ultimate strength of infilled frame (experimental study)

F_{FE} Ultimate strength of infilled frame (numerical study)

f'_m Compressive strength of masonry prisms

f_{s_k} Stress in steel rebar

f_t Cracking strength

F_y Yielding stress of steel

f_j Compressive strength of mortart interface

f_{c_1} Principal stress in concrete

$f_{c_{1_a}}$ Tensile response of concrete determined by tension softening

$f_{c_{1_b}}$ Tensile response of concrete determined by tension stiffening

f_{c_2} Principal stress in concrete

f_{j_1} Principal stress in Mortar interface

f_{j_2} Principal stress in Mortar interface

$f_{j_{2max}}$ Stress at compressive capacity of mortar interface

f_{j_n} Normal stress at mortar interface element

f'_{j_t} Tensile capacity of mortar

f_{m_1} Principal stress in masonry

f_{m_2} Principal stress in masonry

$f_{m_{2_{max}}}$ Compressive capacity of masonry in direction of principal stresses

$f_{m_{bj}}$ Normal stress acting along mortar bed joint

$f_{m_{hj}}$ Normal stress acting along mortar head joint

f_{m_p} Peak stress in compressive constitutive model for masonry material

f_{m_x} Compressive strength of masonry in direction parallel to bed joints

f_{m_y} compressive strength of masonry in direction perpendicular to bed joints

f'_{t_m} Tensile capacity of masonry prisms

G_{f_c} Fracture energy of concrete

G_{f_m} Fracture energy of masonry

$G_{\ln f'_m}$ Lognormally distributed random field

γ^s average slip strain

G_j Mortar shear modulus

h Height of infill wall

I_b Moments of inertia of RC beam

I_c Moments of inertia of RC column

J Jacobian matrix

j Mortar interface (Chapter 5)

K Global stiffness matrix

k Local stiffness matrix

K_{exp} Stiffness of infilled frame (experimental study)

K_{FE} Stiffness of infilled frame (numerical study)

$[K]$ Stiffness matrix

kN Kilonewtons

L Applied load

l Length of infill wall

l_d Length of diagonal strut

L_{r_c} Characteristic length in concrete

L_{r_m} Characteristic length in masonry

M Safety margin

m Total Number of integration points in an element (Chapter 3)

m Masonry material (Chapter 5)

mm Millimeters

$\mu_{f'_m}$ Mean of masonry compressive strength

$\mu_{f'_{m_V}}$ Mean of masonry compressive strength avaraged over volume of diagonal strut

$\mu_{\ln f'_m}$ Mean of masonry compressive strength in a lognormal space

$\mu_{\ln f'_{m_V}}$ Mean masonry compressive strength in a lognormal space avaraged over volume of diagonal strut

μ_R Mean lateral resistance of masonry infilled frame (analytical-based)

$\mu_{R'}$ Mean lateral resistance of masonry infilled frame (simulation-based)

μ_L Load mean

μ_R Resistance mean

N Number of integration points

n Number of elements

$O()$ Algorithmic efficiency

$\vec{p}$ Spatial position

$\vec{p}_1$ Spatial position

$\vec{p}_2$ Spatial position

$\phi()$ Standard normal (Gaussian) cumulative distribution function

ϕ_m Resistance factor for masonry

ϕ_{st} Stiffness reduction factor

R Structural resistance (Chapter 3)

R Runtime ratio (Chapter 4)

r Depth of compressive zone in diagonal sturt

R_s Parameter that effects shape of transition curve in steel stress-strain relationship

ρ Reinforcement ratio

R' Lateral resistance of masonry infilled frame (simulation-based)

r_p Number of rows in CSR matrix

s_j Average crack spacing

$\sigma_{f'_m}$ Standard deviation of masonry compressive strength

$\sigma_{f'_{mV}}$ Averaged standard deviation of masonry compressive strength avaraged over volume of diagonal strut

$\sigma_{\ln f'_m}$ Standard deviation of masonry compressive strength in a lognormal space

$\sigma_{\ln f'_{mV}}$ Standard deviation of masonry compressive strength in a lognormal space avaraged over volume of diagonal strut

σ_R Standard deviation lateral resistance of masonry infilled frame (analytical-based)

$\sigma_{R'}$ Lateral resistance standard deviation of masonry infilled frame (simulation-based)

σ_L Load standard deviation

σ_R Resistance standard deviation

t_c Depth of the compression zone in diagonal strut

t_e Effective thickness of masonry units

t_f Thickness of face-shell

θ_{bj} Angle between direction of principal stresses and bed joint

$\theta_{\ln f'_m}$ Spatial correlation length of masonry compressive strength

θ_s Angle between diagonal strut and horizontal axis

θ_{s_k} Angle between orientation of reinforcements and cracked direction

V Volume of diagonal strut

$v_{f'_m}$ Coefficient of variation of masonry compressive strength

v_j Shear stress in mortar interface

v_s Shear stress at cracked surface

$v_{j_{max}}$ Maximum shear stress in mortar interface

$v_{m_{bj}}$ Shear stress acting along mortar bed joint

$v_{m_{hj}}$ Shear stress acting along mortar head joint

w_{ij} Weight of integration point

w_j Crack width

w_s Width of diagonal strut

w_{eff} Effective width of diagonal strut

w Maximum crack width limit

x Number of CPU cores

ξ Natural coordinate

Acknowledgements

First, I would like to thank my supervisor, Dr. Yi Liu, who provided valuable guidance throughout the course of this research. This desertion would not have been possible without her guidance and kind advice.

I would like to thank my committee member, Dr. Gordon A. Fenton, for his valuable technical comments and assistance with providing computing hardware and software throughout this research.

I wish to extend my appreciation to the remaining of my committee members, Dr. Pedram Sadeghian, and external examiner, Dr. Tony Yang for taking the time to review this dissertation, attending my defense, and providing valuable feedback.

Additionally, I would like to thank the civil engineering department staff, Mrs. June Ferguson and Mrs. Shelley Parker for their help throughout my studies as well as the laboratory technicians Mr. Blair Nickerson, Mr. Brian Kennedy, Mr. Jesse Keane, and Mr. Brian Liekens for their assistance with my project.

I would like to thank my fellow colleagues, Chuanjia Hu, Ryan Steeves and Ehsan Nasiri for their helpful collaboration in completing the experiments.

Finally, I wish to acknowledge the financial support of the Natural Sciences and Engineering Council of Canada (NSERC). Also, I gratefully thank the support of NVIDIA Corporation through their donation of the Titan Xp GPU used in this research.

Chapter 1

Introduction

1.1 General

Masonry infilled frames commonly refer to frame structures, made of either steel or Reinforced Concrete (RC), with masonry walls built inside. Either as partitions to separate spaces or cladding to complete a building envelope, masonry infills are widely used in modern construction including in seismic regions. Previous research has demonstrated that the masonry infills can significantly increase the stiffness and strength of the infilled system. Even after significant cracking, the infills were shown to be beneficial to the ductility and energy dissipation of the infilled system when subjected to dynamic loading (Mehrabi et al., 1996; Mosalam et al., 1997b; Steeves, 2017). The contribution of the infill to the frame system is dependent on the interaction between the infill and its bounding frame. However, to quantify the exact extent of the infill-to-frame interaction for various combinations of infill and frame materials and geometries has remained a challenge. As a result, the research findings obtained thus far have not been effectively translated into industry practice when it comes to design of masonry infills in the North America. Despite a considerable amount of physical evidence of the benefit of masonry infills, the common industry practice is still to treat infills as non-structural elements and design frames for gravity and lateral loading.

This seeming simplification by ignoring the infill presence does not lead to a safe or economical design for the frame. On the contrary, when not designed and detailed properly, the presence of infill has been shown to cause a catastrophic effect on the frame during an earthquake event.

1.2 State of research of masonry infilled frames

There is a considerable amount of research conducted on the in-plane behaviour of concrete masonry infilled steel or RC frames in the past six decades. An in-depth literature review of the in-plane behaviour of masonry infilled frames can be found in studies conducted by Moghaddam and Dowling (1987), Asteris et al. (2013), Chen (2016), and Nasiri (2019). In general, previous studies have shown that the behaviour of masonry infilled frames is complex and influenced by many factors such as geometric and material properties of both the infill and its bounding frame, boundary conditions, and loading situations to just name a few. Much effort has been dedicated to developing analytical models to accurately estimate the infill effect on the stiffness and strength of the infilled frames. One such model is the well-known "diagonal strut" method. The "diagonal strut" method considers the infill effect by replacing the entire infill with an equivalent strut connecting two loaded compressive corners. Once the width of the strut is known, a simple frame analysis can be performed to obtain the system stiffness and the strength of the infill can also be related to the width of the strut.

Since initial inception of the "diagonal strut" concept in 1960s by (Polyakov, 1960) and (Holmes, 1961), much research has focused on the development of strut width formulations

to consider effects of various potential parameters of the system as mentioned above. To that end, several formulations have been proposed and reported in the literature (Stafford-Smith and Carter, 1969; Mainstone, 1971; Paulay and Priestley, 1992; Flanagan and Bennett, 2001; Moretti et al., 2014). The diagonal strut method is also adopted for design of masonry infills in the Canadian masonry standard (CSA S304, 2014) and in the American masonry standard (TMS402, 2016).However, the proposed equations including those in the current design standards were calibrated against test results of one or two studies with specimens of specific material and geometric properties and often the variation in these properties was limited. Hence, the evaluation of these equations against the existing test results has shown disparity, in many cases, significant, between the analytical values and experimental results. None of the proposed equations was found to be universally applicable for masonry infilled frames of varying materials and geometries. Further, the proposed equations for calculating diagonal strut width are only intended for simple infilled frame situations. Geometric irregularities (for example, infill openings), frame-to-infill interfacial gaps, and/or presence of vertical loading are not covered. However, these conditions are not uncommon in practical application of the infills. These reasons are believed to attribute to the disconnect between the research findings and the industry practice when it comes to infill design.

The CSA S304 standard committee recommends that more research is in need to better quantify the interaction as affected by material and geometric properties of both infill and frame and potential failure modes under different loading conditions. While the physical testing of full-scale masonry infills under realistic loading conditions is the most reliable way to gather more data, the feasibility of conducting tests to cover all potential combination

of design parameters is low. With the advancement of computing technology in the past 20 years, using finite element modeling technique encoded in computers (for example, ANSYS, ABAQUS) has become increasingly accepted as an effective tool to gather reliable data. Thus, one common research approach undertaken by many researchers more recently is experimental testing of physical specimens in tandem with a numerical study using finite element modeling techniques. The experimental testing augments the existing data base of infilled frames and more importantly provides test data for validation of the finite element model. Once validated, the model is used in a comprehensive parametric study where multiple combination of parameters can be studied and results are used to further supplement the physical findings. With the computer technology to aid in structural behaviour simulation becoming increasingly popular, some challenges however have been reported in various studies (Lotfi and Shing, 1994; Stavridis and Shing, 2010; Minaie et al., 2010; Chen, 2016; Nasiri, 2019). One is that the accuracy of the model is a tradeoff of computational cost, which indicates as the complexity and sophistication of model increases, the accuracy of the model may decrease in order to achieve computational efficiency. The second is that the source codes of many commercial finite element packages, such as ANSYS and ABAQUS, are not in public domain and algorithms are performed through so-called "black-box" operations, which makes the modification of algorithms to achieve better simulation results difficult.

1.3 Motivation and methodology of the research

The existing research showed that the random and uncertain nature of masonry materials is one of the key factors influencing the frame-to-infill interaction. The randomness and uncertainty of masonry materials are characterized by probability distributions of masonry compressive strength and its modulus of elasticity. This is partly a result of the fact that masonry material is essentially a masonry assemblage consisting of different materials, i.e., masonry units and mortar, and partly a result of the nature of masonry construction which largely relies on workmanship. In previous studies, this randomness and uncertainty has been studied in a mostly deterministic approach where the infill stiffness and strength were determined based on a set of specific material properties of masonry. Even when the masonry material properties were the variables, both numbers and degrees of variations were limited. Few studies have addressed the strength of masonry infills as affected by the randomness and uncertainty of masonry material properties from a probabilistic approach.

A number of studies using a probabilistic approach in the study of masonry infilled frames included fragility analysis of masonry infilled frames due to seismic actions (Hwang and Huo, 1994; Mosalam et al., 1997a; Dymiotis et al., 2001; Celarec et al., 2012; Jeon et al., 2015). These studies were based on a few finite element simulations considering uncertainties in the geometry, the material model, and/or the seismic loads. However, the main limitations of these studies were that first, the number of simulations used was low, which may lead to inaccurate results due to sampling errors. Secondly, the finite element models used in these studies were based on either the equivalent diagonal strut model (Mosalam et al., 1997a; Celarec et al., 2012) or the three-strut model (Jeon et al., 2015),

neither of which can accurately capture infill cracking formation and development, and the actual effect of infill on the frame. The uncertainty study of the variables was then restricted by the simplified model of the infills.

In the first phase of this research, a random field of masonry material property was applied to the entire area of the infill as opposed to applying on a simplified strut. The theoretical backing of this implementation is the Random Finite Element Method (RFEM). The RFEM, originally developed by (Fenton and Griffiths, 1993), employs continuous variable random field models to study the effect of spatially varying properties. Coupling Monte-Carlo simulations with numerical analysis, RFEM has been used to estimate the failure probability of a structural component. The computerized finite element model made the use of RFEM computationally viable. This research adopted the open source software package, OpenSEES, for the development of the finite element model for the masonry infilled RC frames. 27,000 finite element random field realizations were run in the course of three months using 40 processors with an average runtime of 45.43 minutes for each realization. The computational cost associated with any meaningful reliability analysis becomes an important consideration to motivate the second phase of the research.

The second phase of this research was to develop an open-source and modular finite element library with the goal that this finite element method can significantly accelerate the numerical simulations for infilled frames. The main feature of the library was its ability to parallelize the finite element process on parallel architectures such as Graphical Processing Unit (GPU) devices. The modular nature of the library makes it a finite element simulation tool for general structural applications, not limited to infilled frames.

Concurrent with the finite element work, an experimental program was conducted on concrete masonry infilled RC frames subjected to quasi-dynamic cyclic loading. The results of the specimens were used to verify both the OpenSEES model and the model self-developed at the second phase of the research. In this research, quasi-dynamic loading refers to a type of loading where inertial effects are negligible.

1.4 Objectives

As described previously, this research had two main phases. One was the reliability-based analysis pairing the finite element model using OpenSEES with the Random Finite Element Method (RFEM) to study the effect of spatially varying masonry material properties on the strength of masonry infilled RC frames. The second was the development of a nonlinear finite element open source library for the analysis of the infilled RC frames under lateral loading. The detailed objectives of this research are outlined as follows:

- Augment the experimental study of the in-plane behaviour of masonry infilled RC frames under quasi-dynamic cyclic loading to the existing database.

- Provide reliability based design recommendations for the design of infill wall.

- Recommend a new resistance reduction factor for the design of of infilled walls.

- Develop an open-source and modular library for accelerating finite element processes using parallel architectures.

1.5 Document outline

In this document, Chapter 1 provides a brief introduction of the subject and objectives of the research. Chapter 2 presents a detailed review of the literature on the numerical modeling, reliability assessment and finite element package development. Chapter 3 to 5 consists of three research papers. Chapter 3 illustrates the RFEM study of infilled frame using the OpenSEES finite element model. Chapter 4 describes the development of a parallelized finite element library. Chapter 5 presents further development of the parallelized finite element library in estimating the in-plane behaviour of infilled frames. Finally, chapter 6 provides a summary of the research findings, conclusions and recommendations for future work.

Chapter 2

Literature review

2.1 Introduction

This chapter provides background information and an overview of the existing literatures on subjects related to this research. The following sections focus on the specific subjects of reliability analysis and finite element modeling techniques.

2.2 Design approach

The first document dealing with masonry design was introduced in 1954 by the American Standards Association (ASA A41.1, 1954) which formed the basis for conventional empirical design of masonry. With increasing knowledge of masonry behaviour, enabled through more sophisticated testing in the following many decades, masonry evolved from an empirically designed approach which was largely based on "rules of thumb", to "engineered masonry", where forces, moments, and stresses were considered in elementary design rules. During the 1980s and 1990s, the "engineered masonry" approach made use of the Working Stress Design (WSD) approach, also known as Allowable Stress Design (ASD) in both Canada and the US. In the ASD approach, the resistance of masonry structures was calculated based on allowable stress limits. Although simple to use, the drawbacks of ASD

include the fact that all uncertainties were considered using a single factor of safety, and the variabilities in load and resistance had inconsistent effect on the design. In 2004, the CSA S304.1 (2004a) was published with significant technical changes. Limit States Design (LSD) was adopted and the concept of load and resistance factor was introduced. In 2005, the American Masonry Design Standard (MSJC, 2005) also adopted the LSD. In the case of LSD, the load and resistance factors are calibrated for different materials and loading conditions to achieve a specified safety margin.

2.2.1 Reliability assessment

A series of studies (Turkstra and Ojinaga, 1980; Turkstra et al., 1982, 1983; Hart et al., 1983; Englekirk and Hart, 1984; Turkstra, 1989; Hart and Zorapapel, 1991) led to development of the First Order Second Moment (FOSM) method which was widely used as a theoretical basis for shifting from the ASD to the LSD approach. While FOSM analyses were performed on masonry structures under the combination of dead load and live load, the effect of the lateral loads, such as wind and earthquake, have not yet been considered in the reliability assessment of masonry structures in CSA S304-14 (2014).

FOSM is a simple tool for the determination of the reliability index β for a specific safety margin of a structural element. The safety margin (M) in structural applications is defined as follows (assuming normally distributed load and resistance):

$$M = R - L \tag{2.1}$$

where R is the structural resistance and L is the applied load. Failure occurs when $M < 0$,

i.e. the load exceeds the resistance.

The reliability index, β, corresponding to the safety margin given by the Equation 2.1 may be defined, e.g. Cornell (1969), as follows (assuming that the load and resistance are independent):

$$\beta = \frac{\mu_R - \mu_L}{\sqrt{\sigma_R^2 + \sigma_L^2}} \tag{2.2}$$

where μ_R and μ_L are the mean resistance and load, respectively, and σ_R and σ_L are the resistance and the load standard deviations. From Equation 2.2 the failure probability can be directly determined as:

$$P_f = 1 - \Phi(\beta) = \Phi(-\beta) \tag{2.3}$$

assuming that R and L are both normally distributed and Φ is the standard normal (Gaussian) cumulative distribution function.

An alternative safety (Fenton and Griffiths, 2008) margin which is used for lognormally distributed R and L is as follows:

$$M = \ln\left(\frac{R}{L}\right) \tag{2.4}$$

and the corresponding reliability will be expressed as:

$$\beta = \frac{\mu_{\ln R} - \mu_{\ln L}}{\sqrt{\sigma_{\ln R}^2 + \sigma_{\ln L}^2}} \tag{2.5}$$

Hasofer and Lind (1974) developed an improved approach, referred to as the First Order Reliability Method (FORM). The FORM defines the reliability index using the shortest distance from the mean point (μ_m) to the failure surface ($M = 0$) in the direction of the gradient at the mean point. While accurate for linear failure functions, this solution has limitations for non-linear failure functions having a large number of random variables, such as in the case of masonry infilled frames where failure surface is nonlinear with multiple local minimums with respect to the mean point.

The Random Finite Element Method (RFEM), which is not a first order approximation, was selected as an alternative to the FORM in the reliability analysis. RFEM is based on Monte-Carlo simulations of computerized numerical analysis based on one or more input variables with determined distribution of each random field. RFEM then calculates results over and over, each time using a different set of random values to produce distributions of possible outcomes. These probability distributions are used to estimate the uncertainty in variables of a reliability analysis.

2.2.2 Resistance factor

The Canadian masonry design standard S304.1-14 (2014) specifies a resistance factor, ϕ_m, of 0.60, for masonry. This was increased from 0.55 specified in S304.1-94, the 1994 edition of the standard. This increase was based on a study conducted by Drysdale in 1992 which was published by Laird et al. (2005). This change was more or less influenced by the resistance factor increase for cast-in-place concrete from 0.6 to 0.65 in 2004 (CSA A23.3, 2004b). The new resistance factor of 0.65 for the cast-in-place concrete corresponds to a

reliability index in the range of 3.9 to 4.0. This range of target reliability index is suggested for brittle materials with a normal importance factor by the Canadian standard S408 (CSA S408, 2011) and can then be considered as a target reliability for masonry. While concrete resistance factor change was based on considerable research conducted for cast-in-place concrete, the change of resistance factor from 0.55 to 0.6 for masonry lacked sufficient support of technical studies conducted on masonry structures.

2.3 Numerical modelling

As numerical modeling often implemented using commercial software such as ANSYS and ABAQUS are increasingly used in the studies of masonry infilled frames, two main categories of techniques, i.e., macro-modelling and micro-modelling, are summarized in the following sections.

2.3.1 Macro-modelling

In a macro-modelling technique, the infill is considered to be a continuum with a defined stress-strain relationship or in a more simplified version, as a discrete member represented by a line element. The diagonal strut model is one such example of simplified macro modelling technique. In this case, the effect of the entire infill is supposed to be simulated in theory by a truss element. Several analytical equations have been proposed by various researchers for determination of the strut width or strut configuration of the diagonal strut model. The diagonal strut model, albeit with different strut width formulations, has been adopted in the Canadian and American masonry design standards for design of masonry

infills.

The Canadian design standard S304-14 (2014) adopts the diagonal strut model developed by (Stafford-Smith, 1966), who suggests that the contact length between the infill wall and frame column, α_h, and between the infill wall and the frame beam, α_l, can be calculated as follows and as shown in Figure 2.1:

$$\alpha_h = \frac{\pi}{2} \sqrt[4]{\frac{4\, E_f\, I_c\, h}{E_m\, t_e \sin(2\theta_s)}} \tag{2.6}$$

$$\alpha_l = \pi \sqrt[4]{\frac{4\, E_f\, I_b\, l}{E_m\, t_e \sin(2\theta_s)}} \tag{2.7}$$

where:

- E_f is stiffness of the bounding frame.
- E_m is modulus of elasticity of the masonry material.
- h, l are height and length of the infill wall, respectively.
- t_e is the effective thickness of the masonry.
- $I_c,\ I_b$ are the moments of inertia of the column and the beam, respectively.
- θ_s is the angle between the diagonal strut and the horizontal axis.

Then the width of the strut, w_s, is determined assuming a triangular stress distribution along the strut width as follows:

$$w_s = \sqrt{\alpha_h^2 + \alpha_l^2} \tag{2.8}$$

To have a uniform stress distribution, σ_m, the effective width of the diagonal strut, w_{eff}, is considered to be half of the diagonal strut width and it should not exceed one quarter of the diagonal length.

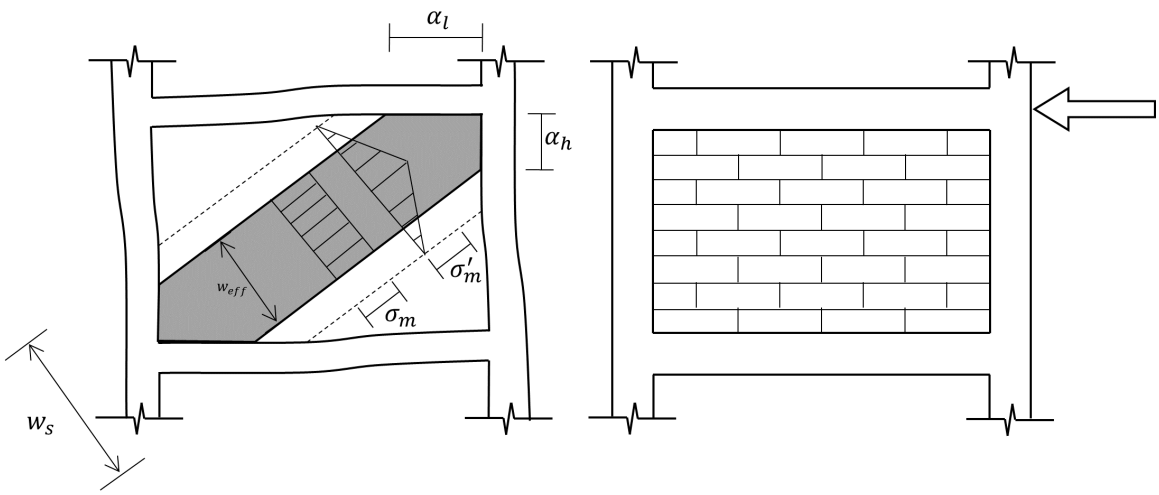


Figure 2.1. Schematic view of the diagonal strut

Stiffness consideration

CSA S304-14 (2014) calculates the initial stiffness of the diagonal strut as:

$$E_{infill} = \frac{\phi_{st}\, w_{eff}\, t_e\, E_m}{l_d} \tag{2.9}$$

where l_d is the length of the strut and ϕ_{st} is a stiffness reduction factor taken as 0.5. This factor was added to the recent edition of the Canadian standard S304-14 based on the research conducted by Chen (2016).

Strength consideration

Corner crushing in the diagonal strut is identified as the most common failure mode for a masonry infilled RC frame of typical material and geometric properties (Hu, 2015; Chen, 2016; Steeves, 2017; Nasiri, 2019). The compressive strength of the diagonal strut is defined using the equivalent masonry stress block calculated as $0.85\ \chi\ f'_m$, uniformly distributed over the compression zone of the cross-section. In this relation f'_m is the compressive strength of the masonry prisms and the χ factor accounts for the effect of direction of the compressive stress in a masonry member relative to the direction used in the masonry prism compression test. The unfactored resistance of the diagonal strut is calculated as:

$$P_r = \left(\frac{l}{\sqrt{h^2 + l^2}}\right)\ \chi\ \left(0.85\ f'_m\right)\ w_{eff}\ t_c \tag{2.10}$$

where t_c is the depth of the compression zone determined as:

$$t_c = \begin{cases} 2\,t_f - r, & r \le t_f \\ t_f, & r > t_f\ \&\ r \le t - t_f \\ t - r, & r > t - t_f \end{cases} \tag{2.11}$$

where r is the distance from the edge of the block within the tension flange to the compressive zone in the cross-section, and t_f is the thickness of the face-shell. Also, the Canadian standard (CSA S304-14, 2014) accounts for the slenderness effect of diagonal strut using moment magnifier method.

2.3.2 Micro-modelling

The micro-modelling technique models the individual blocks and mortar with their respectively defined constitutive relationships. The drawback of this type of modelling is its high computational cost and its accuracy is heavily dependent on the accuracy of material and behaviour model of each component (block, mortar). Thus, a so-called "simplified micro-modelling" technique has been used more commonly (Stavridis and Shing, 2010; Chen and Liu, 2016; Nasiri and Liu, 2017) than the more detailed micro-modelling technique.

In the simplified version, the mortar joints are not physically modelled. Instead, the mortar effect between blocks is represented using contact elements which can include the nonlinear behaviour and failure models of the joints. These models were shown to be able to capture crack initiation and propagation patterns, detailed stress distribution, and failure mechanisms of infilled frames to some degree of success. However, there is commonly a lack of information provided on the input material parameters of these models from various studies. Coupled with "black-box" algorithms commercial software packages, it has been difficult for others to reproduce the model and associated results. In addition, the high computational cost and requirement of comprehensive expertise in finite element modelling make the use of these models in industry design practice a challenge.

2.3.3 Modelling technique adopted for this study

The main problem with micro-modeling technique is the significant computational demand especially in a reliability analysis with potentially thousands of model runs. In this study, a macro-modelling technique, in which the infill is considered as a continuum with a defined

stress-strain relationship was adopted. This type of analysis only requires a 2D model and could be adequate if the interest of the study is to determine the global response of the system. The smeared cracking model (Lotfi and Shing, 1991; Rahimi and Liu, 2017) was used where masonry is modeled with material laws that consider the mechanical properties and behavior of mortar joints and concrete blocks in a "smeared" fashion. The cracking in the infill wall was considered by modifying the masonry material constitutive model progressively to maintain the equilibrium and compatibility at the cracked location. The modelling technique is more computational efficient than the micro-modeling technique while being sufficiently accurate in estimating the lateral load vs. displacement behaviour, cracking patterns and the failure modes of infilled frames.

2.4 Finite element library

One objective of this study was to develop a finite element method into an open-source modular finite element library for simulation of masonry infilled RC frames in particular but also capable of structural applications in general. Its main feature is to accelerate the finite element simulations with the use of Graphical Processing Unit (GPU) devices. Traditionally, the finite element models developed in the existing studies were usually encoded on commercial workstations running on Central Processing Unit (CPU) device. Most personally computers are examples of CPU devices. While the CPU architecture is composed of a number of cores with large cache memory and high clock speed which can handle a few software threads at a time, the advantage of GPU architecture is that there are thousands of cores that can handle thousands of threads simultaneously. This

feature of GPUs is explored to accelerate the finite element model run speed. At the moment, there are more than a thousand open-source finite element repositories available on GitHub (Rahimi et al., 2019). However, only a few of these repositories are actually libraries that provide a clear instruction documentation. Examples of libraries which include documentations are DEAL.ii (Alzetta et al., 2018), MFEM (Kolev and Dobrev, 2010), and FEEL++ (Prud'Homme et al., 2012). However, to the author's knowledge there are no libraries that optimize the runtime of advanced finite element analyses on GPU devices. The library developed in this study is available on GitHub (Wanstrath et al., 2008), a public domain repository, and the library's modular features allow for easy adoption and implementation by others for their specific modeling needs.

2.4.1 Finite element process

One feature of this library was to parallelize the computationally expensive tasks in a finite element process, such as stiffness matrix calculation, assembly and solving the system of equations, to achieve acceleration. In a conventional serial approach to assembling the stiffness matrix, a memory address is allocated to each non-zero entry in the global stiffness matrix and values of entries sharing the same degree of freedom are accumulated at the same memory address. However, this approach, if attempted to be parallelized, will cause a so-called "race condition", where two parallel processes attempt to write data to a specific memory address at the same time. This is an undesirable situation as both processes are "racing" to access/change the data. A solution, proposed by Smith et al. (2013) was to stack entries in a shared memory space between processors. However, this solution cannot be

realized using GPUs because the shared memory is not accessible by all the processing cores available on a GPU device. Some other approaches were suggested for assembling stiffness matrices using GPUs, such as graph colouring (Cecka et al., 2011), graph partitioning (Klöckner et al., 2009) and reduction list (Komatitsch et al., 2009). The main disadvantage of these approaches is that each requires some pre-calculations to determine a list of overlapping elements (sharing the same nodes at the boundary) in the mesh diagram, which is time-consuming and difficult to parallelize. To minimize the preprocessing required for assembly, further studies suggested an approach using "atomic" operations (Fu et al., 2014; Cui et al., 2018). In this approach, a preprocessing is first performed to identify the degrees of freedom required in the global stiffness matrix. Secondly, a sparse matrix, with a compressed sparse row format, is created. Then each entry of the stiffness matrix is calculated and placed in the appropriate position in the global stiffness matrix. To avoid the race condition,"atomic" operations are used to calculate the matrix entries followed by a search procedure to find the relevant memory addresses. Atomic operations typically put a lock on the shared data to ensure only one process can access/change the data at the same time. While this approach minimizes the preprocessing, the serial nature of the "atomic" operation and the following search procedure still increases the overall computational time of the assembly process. Martínez-Frutos and Herrero-Pérez (2015) proposed the use of matrix-free or assembly-free methods which attempt to solve the linear system of equations without forming the global stiffness matrix. This approach maintains the memory at the expense of increasing the load on the processor. While more suited for modeling the overall behaviour of structures, it provides little computing advantage in simulating detailed,

complex, and localized behaviour.

In this study, parallel algorithms were developed to simultaneously run a single instruction on all available processing cores. This type of parallelization accelerates the stiffness matrix calculation process due to the fact that the instructions for calculating the local stiffness matrix remains the same for each integration point. The stiffness matrices then would be stacked on top of each other to avoid race condition. For the assembly step, acceleration is achieved by vectorization and parallelization of the process through the sparse-matrix conversion from a coordinate format to a compressed format.

Solving the global stiffness matrix can be obtained by either the direct method, involving inverting the stiffness matrix and multiplying it by the force vector, or the iterative method, involving iteratively converging on the solution to the linear system of equations. Both solution methods can be implemented on GPU devices using the sparse matrix format and these solver techniques have been widely studied in the literature (NVIDIA Corporation, 2018a; Helfenstein and Koko, 2012; Sharma et al., 2013) and thus were adopted in this study. One additional feature of the algorithm developed here for this process is that the algorithm has the ability of either choosing between the available direct and iterative solvers or looping through possible solvers to reach a desirable level of convergence.

2.4.2 General-purpose graphical processing units

Several recent studies have aimed to employ superior hardware devices and/or high-level software development tools (Martínez-Frutos and Herrero-Pérez, 2015; Wong et al., 2015)

to accelerate finite element analyses. One example (see, e.g., Martínez-Frutos and Herrero-Pérez (2015)) was the development of multi-core technologies on Graphical Processing Unit (GPU) architectures, which accelerates the analyses by employing a large number of processing units for the simultaneous execution of parallelized instructions. A general framework called General-Purpose computing on Graphical Processing Units (Harris, 2005; Owens et al., 2007), for multiple processing cores, has gained considerable interest in the implementation of parallelized numerical methods. However, all numerical simulations on GPU devices require copying data back and forth between the Central Processing Unit (CPU) and the GPU. The overhead of doing so diminishes the computing efficiency gained using parallelization on the GPU and thus poses an obstacle to developing efficient finite element algorithms on GPU devices (Smith et al., 2013). Modern GPU architectures, such as Pascal (NVIDIA Corporation, 2016b) and Volta (NVIDIA Corporation, 2017), provide a solution to this problem. They offer a new hardware capability for memory transfer between the CPU and GPU which allows data synchronization between the GPU and CPU memory without explicit copying between memory locations. The development of Compute Unified Device Architecture (CUDA) by NVIDIA Corporation (2018b), along with a detailed Application Program Interface (API) (NVIDIA Corporation, 2018a), made GPU programming much easier and thus increased its popularity. CUDA is a parallel computing platform that enables software engineers to use GPUs for general purpose computing. However, one drawback of using CUDA is that it requires comprehensive and specific programming skills to be able to efficiently utilize the GPU hardware (Brodtkorb et al., 2013).

2.4.3 GPU architecture

In the past couple of decades, performance of GPU devices advanced rapidly due to the market demand for 3D rendering and game development (Cai et al., 2013). The resulting GPU devices have massive parallel architectures which can be employed in high-performance computing. The concepts and terminologies pertaining to GPU architecture are defined as follows:

- Kernel: an object composed of a scheduler and instructions to be scheduled. The scheduler assigns the instructions to various threads.
- Thread: the sequence of program instructions that can be managed independently and run simultaneously by the scheduler (Butenhof, 1997).
- Compute Unified Device Architecture (CUDA): A parallel programming environment developed by NVIDIA for general computing on GPU devices.
- Streaming Multiprocessor (SM): a part of the GPU device that executes CUDA instructions. Each SM contains execution units capable of performing math operations on both integers and floating-point numbers, as well as a scheduler to organize and assign the instructions to different threads (Wilt, 2013).
- Single Instruction Multiple Data (SIMD): a particular kernel instruction that runs on all processing units in a streaming multiprocessor which can operate on different data types, such as integer and floating-point values (Bolz et al., 2003).
- Block: a group of threads with shared memory space which are executed by a single

multiprocessor.

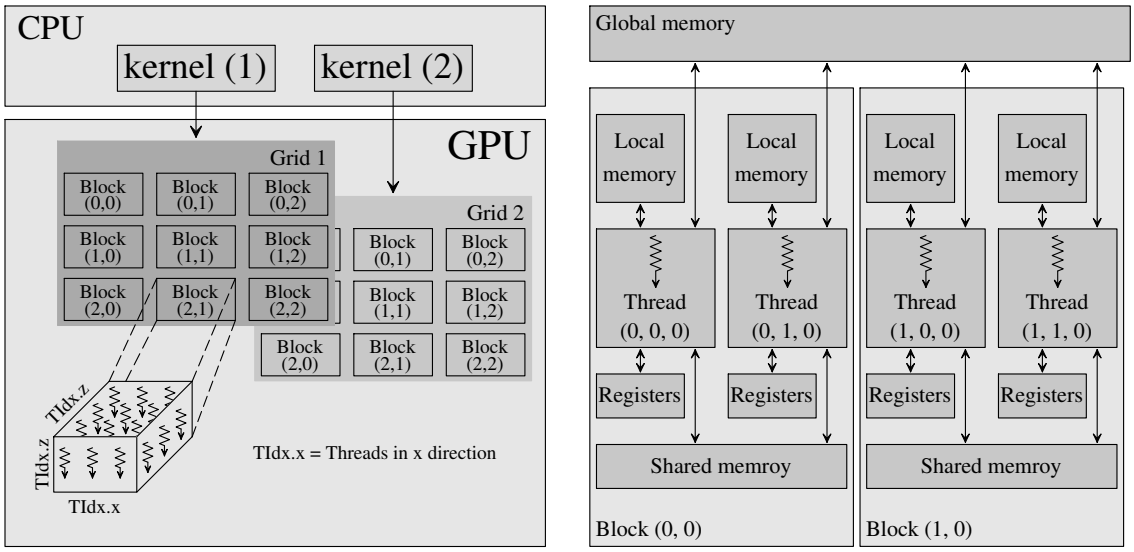
- Grid: a collection of blocks executing a single instruction.

The GPU architecture utilizes a series of streaming multiprocessors (SMs) to enhance the data-level parallelism using single instruction multiple data (SIMD) class of parallel computing. Although SIMD was originally designed for intense graphical processing, e.g. 3D rendering, it has since been implemented in general purpose computing (Harris, 2005; Owens et al., 2007).

Nvidia devices, combined with the CUDA programming platform, provide a comprehensive API for developing GPU accelerated applications using high-level programming languages such as C, C++, Python and Fortran. The saving in computing time is achieved as the result of running a single instruction through many CUDA threads simultaneously. The kernel is responsible for both organizing the CUDA threads and commanding each thread to run the kernel instruction. As shown in Figure 2.2a, the kernel arranges the available CUDA threads into blocks and grids. Optimizing the number of blocks and grids is crucial to the performance gain. If the number of running threads in each block exceeds the number of multiprocessors multiplied by the number of threads available in each SM, a queue of thread blocks would be left waiting to be executed on the GPU, which would result in a relatively poor performance.

The CUDA threads benefit from a type of Dynamic Random-Access Memory (DRAM), with a high bandwidth interface called Synchronous Graphics Random Access Memory (SGRAM), as well as from on-chip memory devices. For the purposes of this research, the memory types of a CUDA enabled device can be categorized into the following groups:

global, local, shared and atomic. All CUDA threads have read/write access to the global memory, which is typically the largest portion of memory. Local memory is on-chip memory which holds the data for each thread. Although local memory is slightly faster than global memory, its smaller size often limits its functionality. Shared memory is memory which is accessible to a restricted number of threads, called a block. The size of the shared memory can be a bottleneck when the kernel includes arrays too large to be stored in the shared memory. The atomic memory holds the information queued to be used by operations which must be sequential rather than parallel. Figure 2.2b presents the memory hierarchy of kernel executions.



(a) (b)

Figure 2.2. GPU architectures' (a) thread batching, and (b) memory hierarchy

The architecture named Pascal, developed by Nvidia and also called GP100, is one of the latest GPU architectures for general purpose applications (NVIDIA Corporation, 2016a). Pascal includes High Bandwidth Memory 2 (HBM2) for its DRAM, which allows a much wider interface (typically means faster access) than the traditional GDDR5 memory

available in Maxwell and Kepler architectures. In their Pascal architecture, Nvidia added a feature to their global memory, referred to as unified memory, allowing applications to have access to the same managed memory allocations by both the CPU and GPU in a workstation. The unified memory is an important asset in optimizing applications that require frequent data transfers between CPU and GPU.

The use of SIMD instructions in parallel computing is best suited for GPU accelerated applications. The main steps in developing a library of optimized and accelerated functions using SIMD within the CUDA framework consist of (i) memory allocation on the GPU, (ii) data transfer from the CPU to the GPU, (iii) running the kernel on the GPU, and (iv) copying the results back to the CPU. Pascal chipsets have been optimized to synchronize data transfer between the CPU and GPU using the unified memory capability. However, the kernel execution still needs to be optimized to improve the GPU performance. The relationship between the number of threads, blocks, grids, and the number of available multiprocessors must be carefully chosen. To accomplish this, a sensitivity analysis was performed in this study to determine the optimal combination of these parameters for best performance of the library. Also, to avoid race conditions, an undesirable situation where multiple processes are "racing" to access/change one data point, memory access was checked using tools such as the *racecheck* function in CUDA (NVIDIA Corporation, 2018a) during each kernel execution.

Chapter 3

Estimating the lateral resistance reduction factor for concrete masonry infills using random field simulations

3.1 Abstract

This paper presents a reliability analysis of concrete masonry infilled frames using the Random Finite Element Method (RFEM). The main objectives of this study were to develop a fast yet reliable finite element model which can be adopted by the RFEM to estimate the resistance factor required to target the design of masonry infill walls at acceptable reliability levels. In this research, the compressive strength of masonry f'_m was selected as the input random field. Based on f'_m, other mechanical properties of masonry, such as the elastic modulus as well as the ultimate strain and the corresponding stress, were determined using established relationships. The developed finite element model was used to analyze the random field simulations and estimate the lateral resistance of masonry infill walls. The lateral resistance distribution was then estimated using the Monte-Carlo simulation method through a large number of realizations of lateral resistance of infill wall, over a range of statistics (mean, standard deviation, and correlation length) of f'_m. To the authors' best knowledge, there are no existing studies which employ continuous spatially variable random field models of masonry compressive strength and study their effect on the lateral

resistance of masonry infill walls. The simulation-based method was then used to calibrate the subsequently developed analytical-based methods for predicting the lateral resistance of infill walls. The analytical methods developed were based on the diagonal strut concept. The predicted analytical-based resistance distribution was then used to estimate the resistance factor for a wide range of target reliability indices.

keywords: reliability-based design, masonry infilled frames, resistance distribution, resistance reduction factor

3.2 Introduction

Masonry walls built inside steel or RC frames are commonly referred to as masonry infills. When built tight against the frame, they have been shown to significantly affect the stiffness, strength, and dynamic characteristics of the frame system under in-plane lateral loading (Moghaddam and Dowling, 1987; Asteris et al., 2013; Hu, 2015; Chen, 2016; Suzuki et al., 2017; Nasiri and Liu, 2017; Rahimi and Liu, 2018). The magnitude of this effect is believed to be dependent on the extent of interaction between the infill and the frame. However, the frame, commonly made of steel or reinforced concrete materials, deforms in a ductile and flexural mode while the masonry infill, made of brittle materials, tends to deform in a non-ductile shear mode. This difference in behaviour, coupled with various combinations of the geometric and material properties of both the frame and the infill, makes it difficult to quantify the exact extent of the infill-frame interaction throughout the entire loading history. Considerable research has been conducted in the past six decades (e.g. Stafford-Smith, 1966, 1967; Mainstone, 1971; Liauw and Kwan, 1985; Flanagan and

Bennett, 1999, 2001; Chen, 2016) in an effort to provide rational methods for considering the infill contribution to the system stiffness and strength. The general observation of these studies underscored the complexity of the infill-frame interaction and identified the masonry compressive strength, frame-to-infill stiffness ratio and infill aspect ratio as being among the most influential factors influencing the combined system behaviour.

For design, both the Canadian and American masonry design standards (CSA S304-14, 2014; TMS 402/602, 2016) specify a "diagonal strut" approach for incorporating infill contribution in the design of infilled frame systems. Conceptually, the approach replaces the entire infill with a compressive strut in the diagonal direction connecting loaded corners. Once the strut width and masonry strength are known, the frame stiffness can then be determined using a frame analysis and the infill strength can be simply related to the strut strength. While being simple to use, the replacement of a two-dimensional infill wall by a one-dimensional strut may lead to loss of accuracy in capturing the effect of many variables in an actual frame configuration. One such variable is the infill compressive strength f'_m, which is identified as one of the most influential variables on the infill stiffness and strength. While its effect on the infill lateral strength is self-evident, it also relates to the masonry's Young's modulus and thus the masonry stiffness. In a realistic infill construction, masonry strength will be different at different locations throughout the wall, due to differences in workmanship, curing conditions, and inherent spatial variability in masonry block and mortar properties. In the diagonal strut model, however, only one masonry strength is used, which does not capture the randomness of masonry properties throughout the infill.

This study was then motivated to investigate the effect of spatially varying masonry

properties on the lateral resistance of masonry infilled frames. The spatially varying masonry properties were modeled by a lognormally distributed random field simulated using the Local Average Subdivision method (LAS, Fenton and Vanmarcke, 1990). The resistance of the resulting frame-infill system was then predicted using the Random Finite Element Method (RFEM, see, e.g., Fenton and Griffiths, 2008) employing a finite element model developed by the author. Although the research on the general topic of masonry infilled frames is large in volume, to the authors' best knowledge, there are no existing studies which employ continuously varying random field models to study the effect of spatially varying masonry properties on the lateral resistance of masonry infilled frames. The methodology used in this study consisted of two components. First, a numerical model to predict the in-plane behaviour of masonry infilled RC frames of varying material and geometric properties subjected to lateral loading was developed and validated against experimental results; secondly, the model was used in a parametric study over various masonry compressive strength statistics using the RFEM to estimate the resistance factors required to safely design the infilled frame system.

The finite element model used in this study implemented the smeared cracking model. The smeared cracking model is commonly chosen to model the cracking behaviour of brittle materials such as masonry infill walls and concrete shear walls (Lotfi and Shing, 1991; Lu et al., 2013). In the smeared crack modeling technique, the effect of cracking is modeled by progressively modifying the constitutive relationships of the material. The finite element package OpenSees, available in the public domain, was used to develop the finite element model. The finite element model was then validated against the test results, achieved in

the experimental study conducted by the same research group (Hu, 2015; Rahimi and Liu, 2018), of masonry infilled RC frames tested under both static and quasi-dynamic cyclic lateral loading.

The RFEM was originally developed by Fenton and Griffiths (Fenton and Griffiths, 1993; Griffiths and Fenton, 1993). In this study, the masonry prism compressive strength, f'_m, was assumed to be a random field having a lognormal distribution with mean values, standard deviations and spatial correlation lengths varied in a parametric study. As mentioned above, the Local Average Subdivision (LAS) method was employed to produce random field realizations of masonry compressive strength over the infill walls. A total of 1,000 simulations were performed for each set of parameters in this study. The lateral resistance of the masonry infilled frame for each realization was then estimated using the finite element analysis developed herein. Subsequently, the analytical-based lateral resistance distributions were estimated using the diagonal strut formulations and were calibrated using the simulation-based distributions. The resulting suite of distributions were used to estimate the probability of failure of infill wall systems designed using a range of resistance factors, resulting in plots of failure probability vs resistance factor. These plots can then be used to determine the resistance factor required to achieve a desired reliability.

3.3 Numerical modelling of the RC frame

A fibre element, available in the OpenSees element library, was employed to model the reinforced concrete frame members. The fibre element is essentially a two-node, beam-column element with 6 degrees of freedom at each node (three translational and three

rotational degrees of freedom). The cross-section of each reinforced concrete frame member was divided into three different regions including the concrete cover, concrete core and steel reinforcement, as shown on Figure 3.1. Figure 3.1 also shows both the fibre discretization and the number of fibres used in each region. The fibres are oriented in the longitudinal direction of the frame member and were extended over the member's full length. The fibers were spaced 9 mm centre to centre. A sensitivity analysis was conducted on the number of fibres required in each region and the number selected (Figure 3.1) was found to provide sufficiently accurate results within a reasonable computational time. The rebars for the steel reinforcement were also modeled using fibers with the "layered straight" command available in OpenSees.

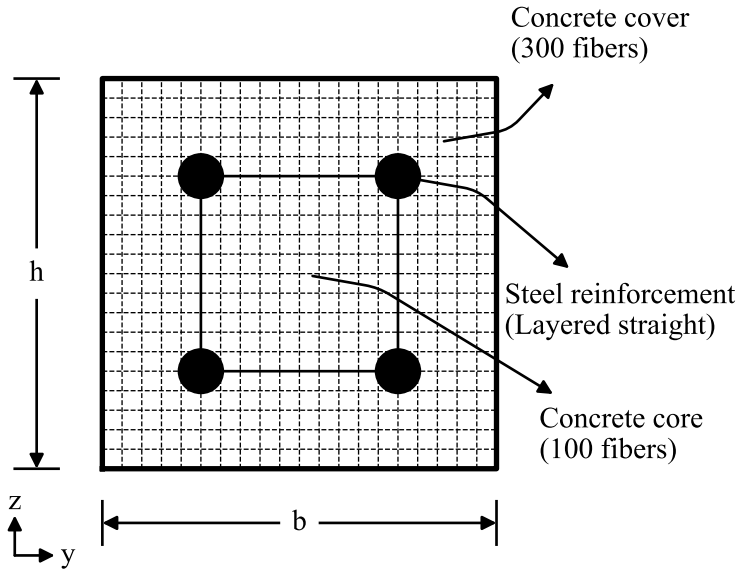


Figure 3.1. Fibre discretization of reinforced concrete section

The static stress-strain model developed by Menegotto and Pinto (1973) and modified by Filippou et al. (1983) was used to model the steel rebar behaviour, including strain hardening. Pinching and softening, observed in the hysteretic load vs. displacement curves

of reinforced concrete frames are believed to be associated with concrete cracking and the bond-slip effect of steel rebar (Filippou et al., 1983). The bond-slip effect refers to a phenomenon where a steel bar, when embedded in concrete, does not show a pronounced yield plateau and the "apparent yield stress" is lower than the yield stress of a bare steel bar. To simulate the bond-slip effect in the proposed model, the loading, unloading and reloading paths in the steel rebar stress-strain relationship were adopted from work by Monti and Spacone (2000). Figure 3.2a shows the material constitutive model used for the steel rebars.

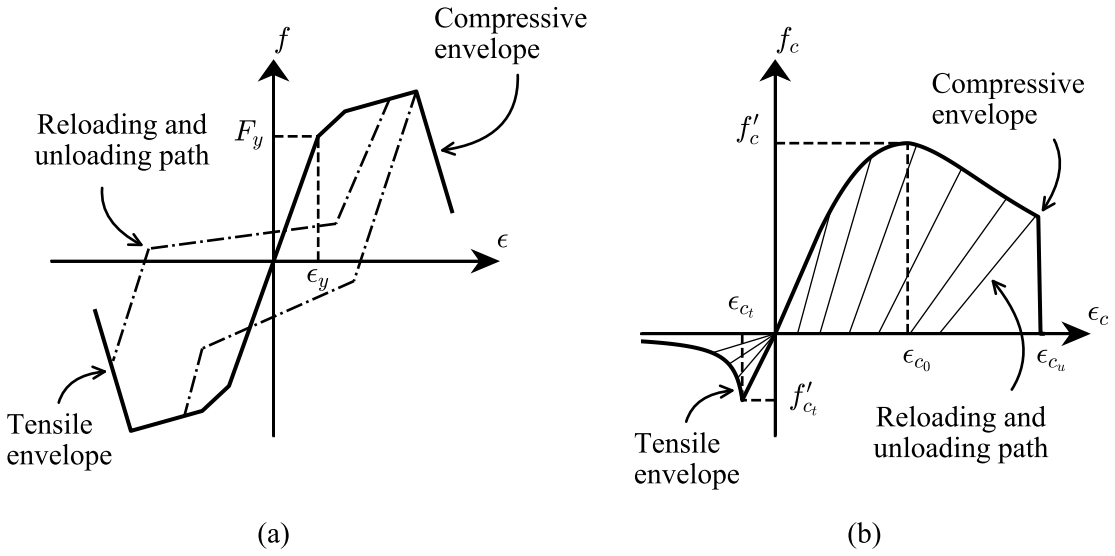


Figure 3.2. Stress-strain relationships; a) steel b) concrete

The compressive stress-strain envelope for concrete was based on the model proposed by Mander et al. (1988) for confined concrete (Figure 3.2b). In the case of cyclic loading, the unloading and reloading responses defined by Karsan and Jirsa (1969) were implemented. The concrete in tension was assumed to be linearly elastic prior to cracking. The falling branch of the tensile response was assumed to follow an exponentially decaying curve, as

depicted in Figure 3.2b.

3.3.1 Modelling the infill wall

The shell element, ShellMITC4 (Dvorkin et al., 1995), was used to model the masonry infill wall. To simulate the stress distribution across the thickness of the masonry infill wall, a multi-layer section, developed by Lu et al. (2013), was employed. By discretizing each face-shell of the masonry block into multiple fully-bounded layers in the thickness direction, the multi-layered approach was used to capture the three-dimensional stress distribution of the masonry infill wall. A sensitivity analysis was conducted to determine the mesh size and number of layers required to achieve reasonable accuracy with tolerable computational effort. Figure 3.3 shows the number of layers used in a single block. The elements used in this study had the width of 23.12 mm and the height of 22.5 mm.

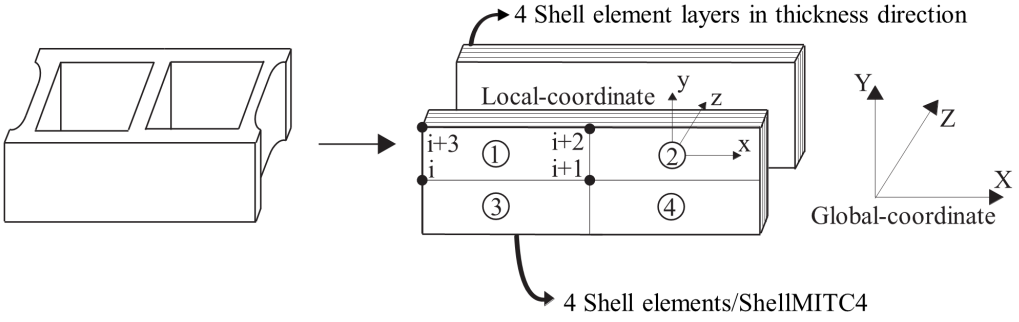


Figure 3.3. Mesh size and number of layers used to model each masonry block

The smeared cracking concept was implemented in the masonry elements by modifying the material constitutive relationship after cracking considering both compression and tension softening effects. The compression softening is the mechanism that results in decreasing the peak masonry compressive capacity due to cracking in the principal tensile direction. The falling branch of the stress-strain curve would follow a linear descending path

until the limit of 20% the peak stress, f'_m, is reached. In tension, the softening mechanism, in a similar manner, occurs after cracking and decays the tensile capacity of the masonry to zero. The ultimate strain ϵ_{t_u} in the tension branch was calculated based on the Mode-I fracture energy (Nasiri and Liu, 2017). Figures 3.4a and 3.4b show the constitutive models used for masonry in tension and compression, respectively.

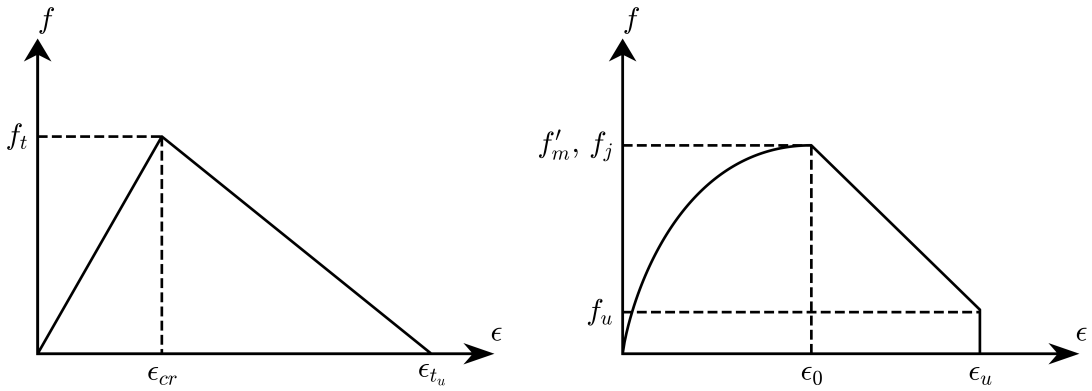


Figure 3.4. Stress-strain relationship for masonry and mortar materials; (a) tensile behaviour; and (b) compressive behaviour

The mortar interface between the concrete frame and masonry infill wall was modelled using the zero-length element available in OpenSees. The zero-length element was placed at each point of contact between the masonry infill wall and the bounding frame, connecting the fibre element and the shell element. The zero-length element is basically a spring with a corresponding uniaxial behaviour defined for a degree of freedom. In this research, a behaviour similar to the masonry material, as shown in Figure 3.4b was considered for the mortar interface, where f_j and ϵ_0 are the stress and the corresponding strain at compressive capacity of the mortar interface, respectively. The failure in compression was assumed to occur when the mortar capacity reached the limit of 20% the peak stress f_j. In tension the

cracking stress f_t is the mortar cracking strength and the ultimate strain ϵ_{t_u} was determined using the Mode-I fracture energy similar to the masonry material.

3.4 Calibration and validation of the numerical model

The model was validated using results of two experimental studies on laterally loaded concrete masonry infilled RC frames. One study was conducted by Hu (2015) with monotonic loading and the other was by Rahimi and Liu (2018) with cyclic loading.

3.4.1 Experimental study

Test set-up

Both studies used a similar test set-up. Figure 3.5 shows a schematic view of the test set-up. A hydraulic actuator with a capacity of 150 kN was used to apply the quasi-dynamic cyclic or the monotonic lateral load. The term quasi-dynamic loading in this paper is referring to the type of loading where the effect of inertial forces are negligible. The actuator was housed in an independent frame which was then attached to the column of a reaction frame. To realize the pulling and pushing action on the specimens subjected to cyclic load, two threaded rods running the full length of the frame beam were used. The actuator head was connected to the specimen through a steel plate and threaded rod assembly. At the loading point between the steel plate and the RC frame, a rubber pad was used to ensure a uniform stress distribution and prevent localized crushing of the concrete. The base beam of the frame was clamped to the strong floor and braced using hydraulic jacks to prevent potential in-plane movements. Linear Variable Displacement Transducers (LVDTs) were

used to monitor the specimen displacement. As shown in Figure 1, LVDT 1 and LVDT 2 were used to measure the lateral displacement of the top beam and bottom beam of the frame specimen, respectively. Another LVDT was used to measure the potential transverse movement at the frame beam center (not shown).

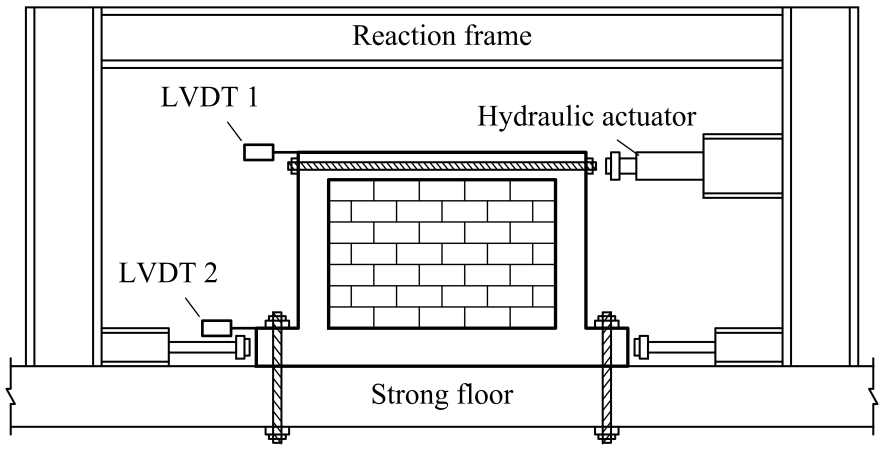


Figure 3.5. Schematic view of the test set-up

Test specimens

Table 3.1 summarizes the specimens used in these two studies. Specimen IF-NG was tested as a control specimen. The remaining infilled specimens incorporated different interfacial gap scenarios, including: 1) at top beam-infill interface (Top Gap), 2) at two column-infill interfaces (Side Gap), 3) at both beam and column-infill interfaces (Full Separation Gap). In addition to the predefined gap, the specimens had a window opening accounting for 20% of the infill area and with an opening aspect ratio of 1:1.5 (IF-W-SG12 and IF-W-TG12).

Figure 3.6 shows the dimensions and reinforcement details used for all specimens. The infill had a height-to-length aspect ratio of 0.73. The half-scale standard 200 mm Concrete Masonry Units (CMUs) were used in a running bond pattern to construct the masonry infill

Table 3.1. Summary of the test specimens

Monatonic Loading Hu (2015)			Cyclic Loading Rahimi and Liu (2018)		
Specimen ID	Gap	Opening/Infill area ratio	Specimen ID	Gap	Opening/Infill area ratio
BF	N.A.	N.A.	BF	N.A.	N.A.
IF-NG	N.A.	-	IF-FG12	12 mm Top Gap & 12 mm Side Gap (6 mm each side)	-
IF-TG7	7 mm Top Gap	-	IF-W-TG12	12 mm Top Gap	20%
IF-TG12	12 mm Top Gap	-	IF-TG25	25 mm Top Gap	
IF-SG7	7 mm Side Gap (3.5 mm each side)	-	IF-W-SG12	12 mm Side Gap (6 mm each side)	20%
IFSG12	12 mm Side Gap (6 mm each side)	-	-	-	-

wall. All pre-defined gap magnitudes were achieved by adjusting mortar thickness with the exception of specimen IF-TG25 in which case the height of the top layer of blocks was also trimmed to achieve the desired gap size. The infills of all specimens were ungrouted while for specimens IF-W-TG12 and IF-W-SG12 with openings, the block cells in the course above the opening were grouted as per industry practice. The top beam and columns of the RC frame were 180 mm by 180 mm square sections reinforced with four 10M deformed steel rebars and 10M stirrups spacing at 100 mm center-to-center. The base beam was a 250 mm square section reinforced with four 15M rebars and 10M stirrups spacing at 100 mm center-to-center. In addition, four 300 mm by 300 mm L-shaped rebars made from 10M rebars were used to further reinforce the top beam-column corners.

Material properties

The mechanical properties of Concrete Masonry Units (CMUs), mortar, and masonry prisms for the infill and those of concrete and reinforcement of the frame were obtained experimentally in accordance with ASTM specifications. Table 3.2 presents the average

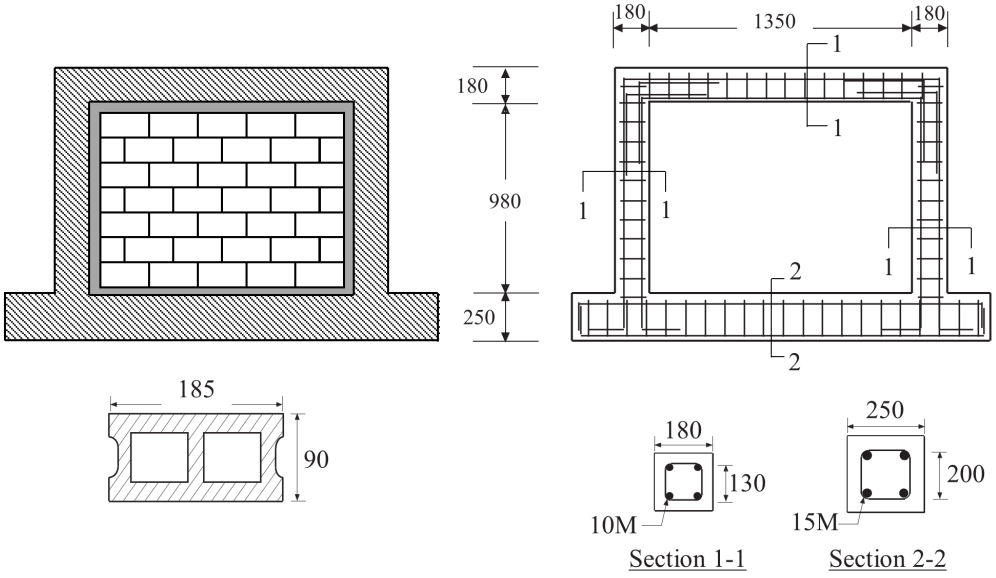


Figure 3.6. Details of test specimens (unit: mm)

strength and stiffness observed during the material tests. The coefficient of variation of the average for each material was below 15% for masonry components, indicating a relatively consistent level of properties.

Table 3.2. Material properties of the test specimens

	Monatonic Loading Hu (2015)				Cyclic Loading Rahimi and Liu (2018)			
	Elastic modulus (MPa)	Compressive strength (MPa)	Tensile strength (MPa)	Yield strength (MPa)	Elastic modulus (MPa)	Compressive strength (MPa)	Tensile strength (MPa)	Yield strength (MPa)
Concrete	29295	43.8	3.5	-	12710	29.2	1.7	-
CMUs	-	25.0	1.6	-	-	25.0	1.6	-
Mortar	-	21.3	1.7	-	-	22.1	1.3	-
Masonry prisms	14535	17.1	-	-	2980	10.5	1.04	-
Reinforcement	247357	-	665	446	247357	-	665	446

Loading protocol

During the static test, the lateral load was applied gradually at a rate of 6 kN per minute to the frame top beam until failure of the specimen. In the cyclic loading protocol, a sequential phased displacement technique was used to apply the displacement to the infilled

frame based on the procedure specified by the Applied Technology Council (ATC-24 1994) for cyclic load test. Figure 3.7 shows the lateral quasi-dynamic loading protocol where the peak amplitude for each set of cycles is defined based on the yield deformation, Δ_y. Testing begins with six elastic cycles with at least three of which are performed at 0.75 Δ_y. Following the elastic cycles, three cycles at Δ_y, 2 Δ_y, and 3 Δ_y. respectively are performed. If the specimen has not failed by 3 Δ_y cycles, the loading would continue with sets of two cycles starting at 4 Δ_y increasing by increments of Δ_y until failure. The calculated Δ_y for the RC frame in this study is 7 mm, which occurs at approximately 34 kN of the in-plane lateral force. The displacement amplitudes were applied at a rate of 10 mm per minute to reduce the effect of inertial forces.

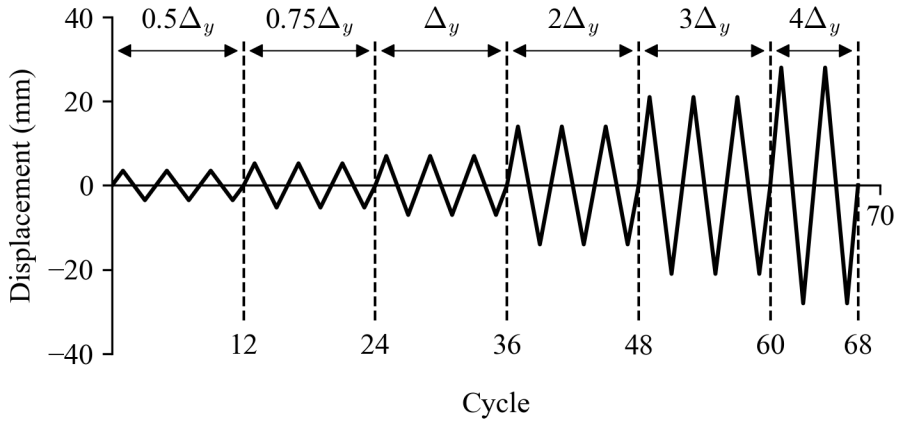


Figure 3.7. Loading protocol for quasi-dynamic loading

3.4.2 Numerical results

Table 3.3 summarizes experimental and numerically predicted results on the strength and stiffness of each specimen in the two studies. The experimental strength was determined to be the maximum load obtained from either the static or hysteretic response curves, and

Table 3.3. Summary of the numerical results vs. experimental data

Monotonic test Hu (2015)							Cyclic test Rahimi and Liu (2018)						
ID	Strength (kN)			Stiffness (kN/mm)			ID	Strength (kN)			Stiffness (kN/mm)		
	F_{exp}	F_{FE}	F_{exp}/F_{FE}	K_{exp}	K_{FE}	K_{exp}/K_{FE}		F_{exp}	F_{FE}	F_{exp}/F_{FE}	K_{exp}	K_{FE}	K_{exp}/K_{FE}
BF	58.5	59.1	0.99	1.7	1.7	0.99	BF	60.5	59.8	1.04	1.19	1.16	1.02
IF-NG	133.6	134.9	0.99	12.2	10.2	1.20	IF-FG12	79.5	72.1	1.10	2.52	2.65	0.95
IF-TG7	129.0	133.3	0.97	8.4	8.3	1.02	IF-W-TG12	71.8	64.2	1.12	2.68	2.66	1.04
IF-TG12	102.0	109.1	0.93	3.6	3.5	1.04	IF-TG25	74.5	74.3	1.00	5.54	6.80	0.81
IF-SG7	134.0	129.5	1.04	7.9	6.6	1.19	IF-W-SG12	66.9	70.0	0.96	2.13	1.92	1.11
IF-SG12	114.0	119.6	0.95	2.5	3.1	0.85	-	-	-	-	-	-	-
AVG			0.98			1.04	AVG			1.04			0.98
C.O.V(%)			3.6			12.7	C.O.V(%)			6.7			11.1

the stiffness is determined as the secant stiffness connecting the maximum load point and the origin. In both loading cases, the experiment-to-numerically-predicted ratios for both strength and stiffness values were close to unity, indicating good agreement. The COV for the stiffness ratio is higher than the COV for the strength ratio, but both were less than 15%, which can be considered relatively low from a practical standpoint for masonry. This suggests that the numerical finite element model is capable of providing accurate values for the important response indicators of masonry infill wall systems, including those with interfacial gaps and infill openings.

The finite element model was further validated by comparing load vs. displace response curves. Figure 3.8 compares the load vs. displacement response of the monotonically loaded infilled frame specimens using IF-NG as an example. It shows that the finite element model accurately estimated the rising branch of the static response of the masonry infilled frame. If the initial stiffness is compared, the experimental-to-numerically-predicted ratio would be even closer to 1.0 than seen for the secant stiffness ratio presented in Table 3.3. The difference observed in the post-ultimate portion of the curves is believed to be associated with the inaccuracies in the modeling of residual strength of masonry beyond its ultimate strain. In the model, this residual strength was considered to be zero while in

the experiment, the masonry material retained residual strength after the ultimate load was reached.

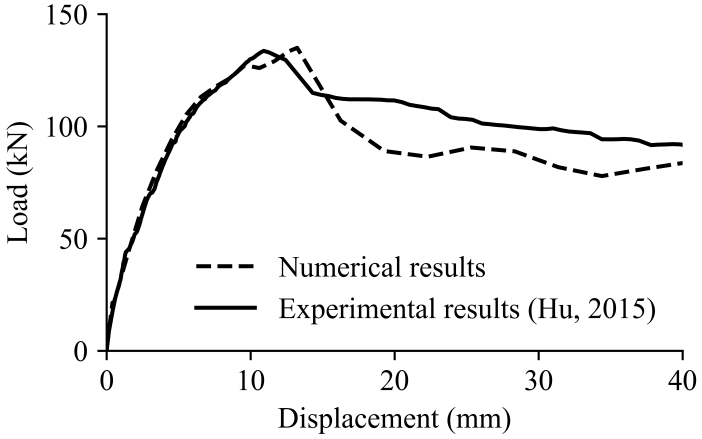


Figure 3.8. Load vs. displacement response of masonry infilled frame under monotonic loading

Figure 3.9 compares the numerical and experimental hysteretic as well as the backbone curves for the cyclically loaded specimens using IF-TG25 as an example. The cyclic loading was applied as quasi-dynamic and thus the effect of time and inertial forces were irrelevant. Again, the numerically predicted hysteresis and backbone curves compare reasonably well with the experimental results.

One important feature of hysteresis response is the degradation of stiffness as loading and unloading progress. Figures 3.10a and 3.10b compare the loading and unloading stiffness of specimen IF-TG25 in each cycle. The loading and reloading stiffness is defined as the secant stiffness from the origin to the peak load at each cycle. The unloading stiffness is defined as the secant stiffness from the peak load to zero load within the half cycle. The degradation of stiffness as loading progressed is seen to be accurately captured by the numerical model.

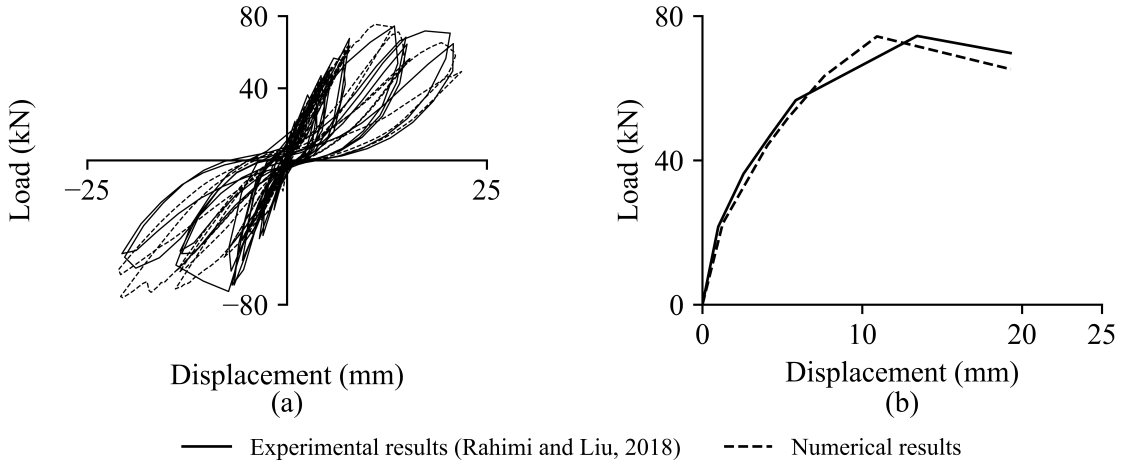


Figure 3.9. Load vs. displacement response of masonry infilled frame under quasi-dynamic cyclic loading; (a) hysteretic response, (b) backbone curve

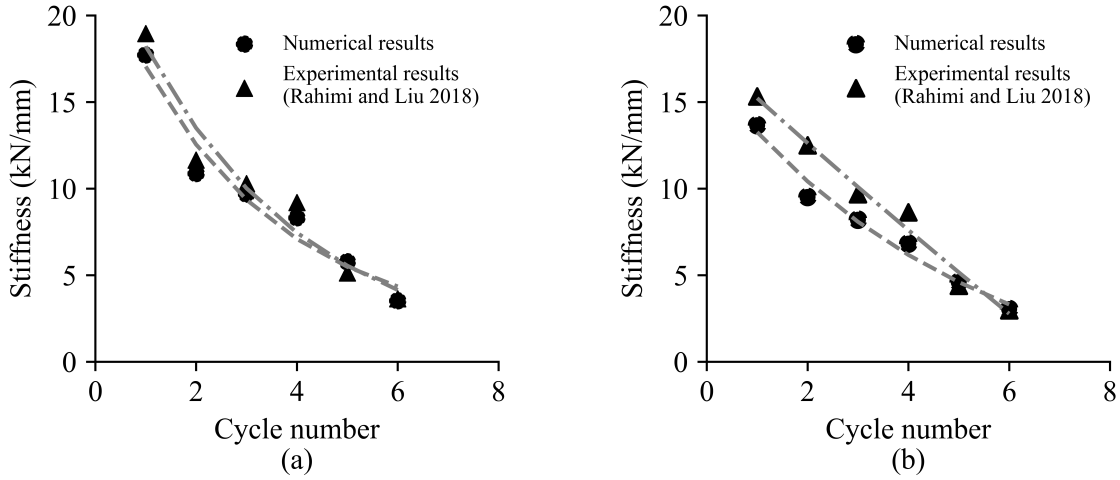


Figure 3.10. Stiffness vs. cycle numbers; (a) loading stiffness, (b) unloading stiffness

The failure mode of all specimens, as predicted by the finite element model, was corner crushing, which was in agreement with the experimental observations. Figure 3.11 shows the comparison of failure modes for specimen IF-TG25, noting that failure in the finite element model (Figure 3.11b) is indicated through the normal stress contours at the ultimate load. In Figure 3.11b the maximum normal stress is shown in dark blue, which coincides with the corner crushing locations at the top left and bottom right corners of the infill. The high normal stresses forming a strip from the top left corner to the bottom right

corner of the infill shown in Figure 3.11b, agreed well with the diagonal cracking pattern observed during the experimental test. It is noted that however, the presented finite element model is not capable of simulating the shear sliding failure as in this smeared model where mechanical behaviour of blocks and mortar joints are "smeared" over the entire infill, the shear behaviour is not explicitly modeled.

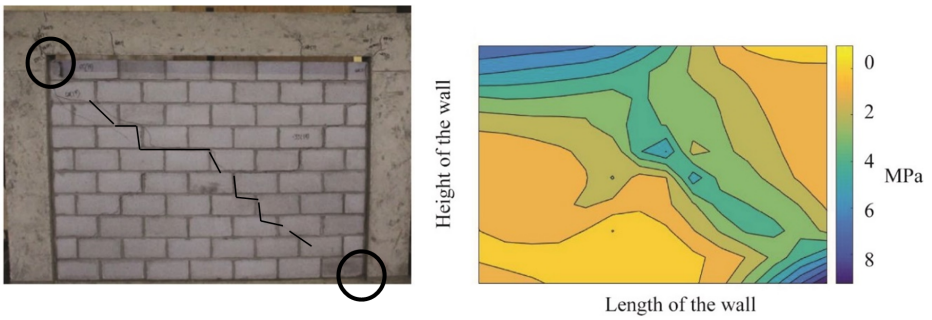


Figure 3.11. Failure mode of specimen IFTG25; (a) experimental observation (b) stress distribution contour of simulated model

3.5 The random finite element method

In this study, the lateral resistance distribution of masonry infills is estimated by modeling the masonry compressive strength, f'_m, as a spatially variable random field. The compressive strength, f'_m, of the masonry prisms was assumed to be lognormally distributed with mean $\mu_{f'_m}$, standard deviation $\sigma_{f'_m}$, and spatial correlation length $\theta_{\ln f'_m}$. The spatial correlation length is a parameter for describing the spatial variability of the masonry compressive strength over the infill, f'_m. A "low" spatial correlation length means that f'_m, varies significantly from point to point within the infill whereas a "high" spatial correlation length indicates a more gradual variation of f'_m from point to point within the infill. The lognormal distribution was assumed because the compressive strength values are non-negative and

so positively skewed, not symmetric like a normal distribution (i.e. as shown in the next section, the lognormal distribution fits well to the histograms of the resistance of infilled frame). Most engineering problems are low-strength dominated (for instance, compressive strength is dominated by the weak zones in the material) and a low-strength dominated average is a geometric average which tends to follow a lognormal distribution by the central limit theorem (Fenton and Griffiths, 2008). Lognormal random fields are fully specified by their mean and covariance structure. A lognormally distributed random field is obtained from a normally (Gaussian) distributed random field, $G_{\ln f'_m}$, having zero mean, unit variance and spatial correlation length, $\theta_{\ln f'_m}$, according to:

$$f'_m(\vec{p}) = \exp\left\{\mu_{\ln f'_m} + \sigma_{\ln f'_m} G_{\ln f'_m}(\vec{p})\right\} \tag{3.1}$$

where $\vec{p}$ is the spatial position. The mean and standard deviation of the lognormal distribution are obtained through the transformations:

$$\sigma^2_{\ln f'_m} = \ln\left(1 + v^2_{f'_m}\right) \tag{3.2}$$

$$\mu_{\ln f'_m} = \ln \mu_{f'_m} - \frac{\sigma^2_{\ln f'_m}}{2} \tag{3.3}$$

where $v_{f'_m}$ is the coefficient of variation of masonry compressive strength. An exponentially decaying (Markovian) correlation function was employed to specify the correlation coefficient between the $\ln f'_m$ at a point ($\vec{p}_1$) and any other point ($\vec{p}_2$) as expressed in the following

(Fenton and Griffiths, 2008):

$$\rho_{\ln f'_m}(\tau) = \exp\left(-\frac{2|\tau|}{\theta_{\ln f'_m}}\right) \tag{3.4}$$

where τ is the distance between the spatial positions, $\vec{p}_1$ and $\vec{p}_2$. The spatial correlation length, $\theta_{\ln f'_m}$, is loosely defined as the distance within which two values of $\ln f'_m$ are significantly correlated. In this random field model, the correlation structure was assumed to be isotropic, so that the correlation length was assumed to have the same value in any direction. The Monte-Carlo simulation was then conducted to produce realizations of the random field of masonry compressive strength. Figure 3.12 shows examples of different random field realizations of the f'_m field, over the masonry infill, where dark colours imply higher masonry compressive strength. As shown in the Figure 3.12 every random field is different in term of the variation of the strength over the wall as expected on a real wall.

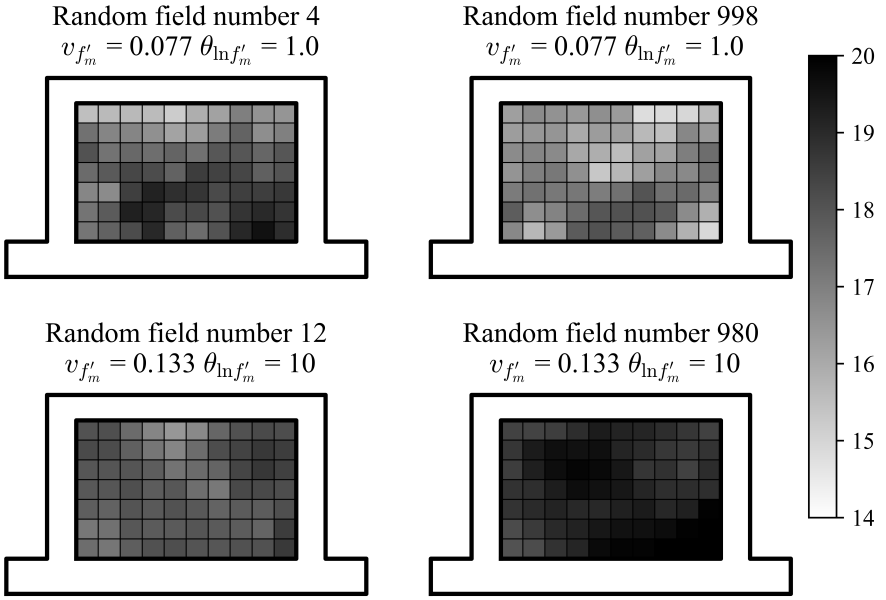


Figure 3.12. Random field simulation of the masonry prism compression strength

3.6 Lateral resistance distribution

3.6.1 Simulation-based

In this study the lateral resistance distribution of masonry infilled RC frames was determined using Monte-Carlo simulation. The lateral resistance of the masonry infilled frame, denoted as R', is defined as the ultimate in-plane strength of the infilled frame under monotonic loading. The primed notations here refers to the simulation-based results. In this study, the f'_m field was assumed to have a mean of 17 MPa, representing an average compressive strength of a typical hollow masonry wall construction with Type S mortar. A total of 27 random field simulation runs, comprising of 1000 realizations each, were performed for each of a variety of spatial correlation lengths $\theta_{\ln f'_m}$, and coefficients of variation $v_{f'_m}$ of the f'_m field. Table 3.4 summarizes the parameters varied. For each run, one thousand random fields of f'_m over the infill (one example is shown in Figure 3.12) were generated. The lateral resistance of the infilled frame for each realization of f'_m was then calculated using the numerical model described in the previous section. The mean of f'_m ($\mu_{f'_m}$ = 17 MPa) was held fixed for all random field simulations. The coefficients of variation of f'_m considered were 7.6%, 9.4%, 10.8%, 13.2%, and 17.1% and so both above and below the value (15%) suggested by the CSA S304-14 (2014). For each coefficient of variation, the correlation length was also varied from 0.1 to 10 m to study the effect of correlation length on the distribution of the lateral resistance.

In each realization the lateral resistance of the infill wall was determined by deducting the RC frame resistance from the infilled frame resistance at the failure point. After 1000 realizations for each simulation set, or random field ID 1 through 27, a resistance histogram

Table 3.4. Summary of the random fields

Random field ID	$\theta_{\ln f'_m}$ (m)	$v_{f'_m}$	random field ID	$\theta_{\ln f'_m}$ (m)	$v_{f'_m}$	random field ID	$\theta_{\ln f'_m}$ (m)	$v_{f'_m}$
1	0.20	0.076	10	10.0	0.094	19	1.00	0.132
2	1.00	0.076	11	0.10	0.108	20	2.00	0.132
3	2.00	0.076	12	0.20	0.108	21	5.00	0.132
4	5.00	0.076	13	1.00	0.108	22	10.0	0.132
5	10.0	0.076	14	2.00	0.108	23	0.20	0.171
6	0.20	0.094	15	5.00	0.108	24	1.00	0.171
7	1.00	0.094	16	10.0	0.108	25	2.00	0.171
8	2.00	0.094	17	0.20	0.132	26	5.00	0.171
9	5.00	0.094	18	0.50	0.132	27	10.0	0.171

Table 3.5. Summary of the random fields results

Random field ID	$\mu_{R'}$ (kN)	$\sigma_{R'}$ (kN)	random field ID	$\mu_{R'}$ (kN)	$\sigma_{R'}$ (kN)	random field ID	$\mu_{R'}$ (kN)	$\sigma_{R'}$ (kN)
1	91.77	3.31	10	90.32	6.95	19	90.42	6.88
2	91.74	4.40	11	90.75	3.32	20	90.13	8.01
3	91.35	4.98	12	90.63	4.10	21	90.33	9.59
4	90.85	5.81	13	90.09	5.66	22	90.65	10.34
5	92.10	6.23	14	89.79	6.60	23	90.79	6.60
6	90.51	3.78	15	90.22	8.07	24	89.86	8.96
7	90.23	4.88	16	90.39	8.11	25	89.26	10.64
8	90.43	5.68	17	90.00	4.82	26	89.23	12.42
9	90.75	6.71	18	90.97	6.12	27	89.61	14.24

was constructed to which a lognormal distribution was fit. Examples of fitted lognormal distributions to the histograms of some of the simulation sets are depicted in Figure 3.13. The figure shows the effect of both the correlation length and the coefficient of variation of the masonry compressive strength on the distribution of the lateral load resistance of infill walls.

For each simulation run, the resulting estimated mean, $\mu_{R'}$, and standard deviation, $\sigma_{R'}$, of the lateral resistance, R, are shown in Table 3.5. The results show how the mean and standard deviation of the resistance changes with different choices of correlation length and coefficient of variation.

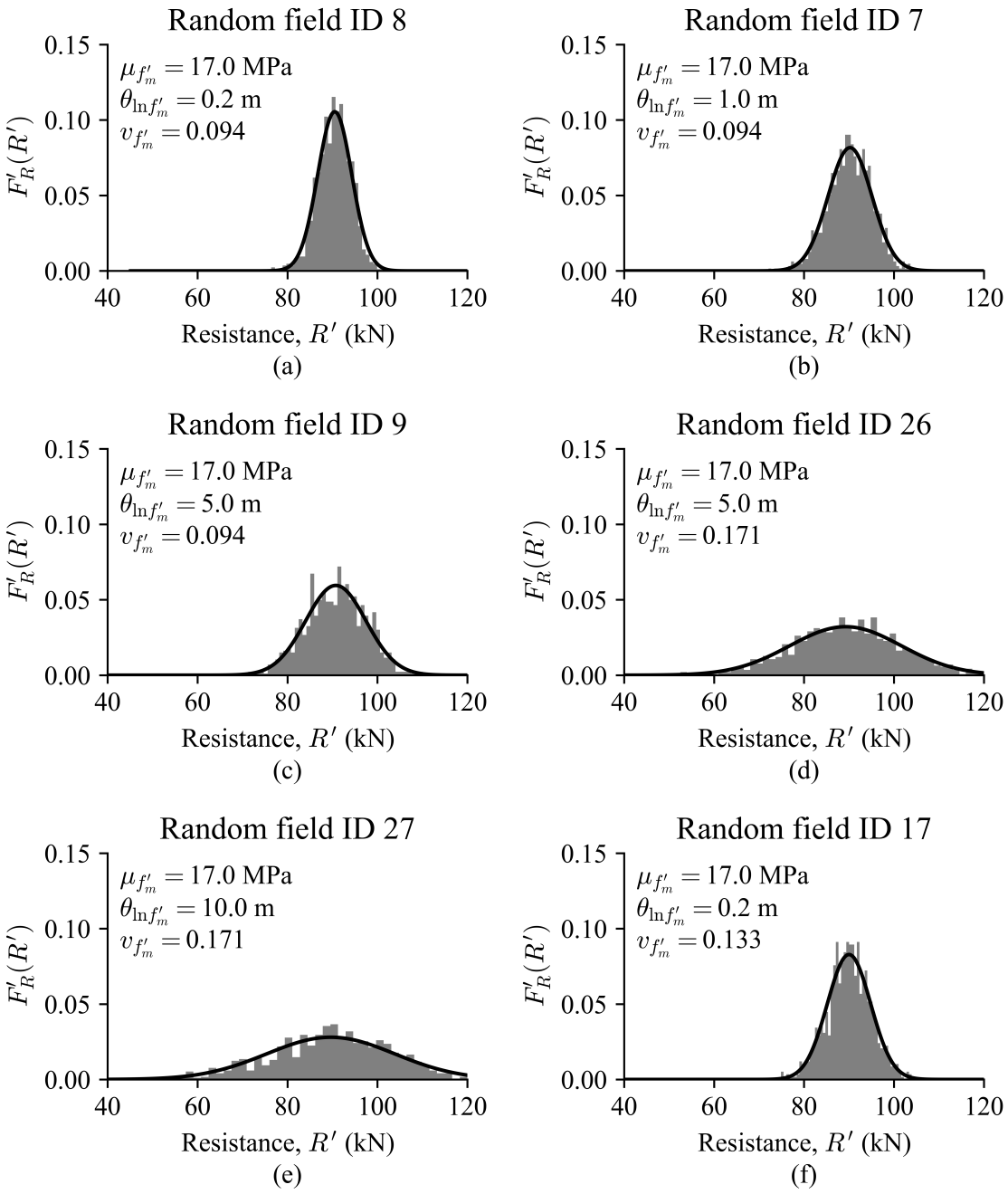


Figure 3.13. Histogram vs. the fitted lognormal distribution for lateral load resistance; (a) random field ID 8, (b) random field ID 7, (c) random field ID 9, (d) random field ID 26, (e) random field ID 27, (f) random field ID 17.

Figures 3.14a and 3.14b show the mean values and the standard deviations of the lateral resistance achieved in each simulation set versus the spatial correlation length for different sets of coefficient of variation, respectively. Figure 3.14a indicates that for a given $\mu_{f'_m}$ (in

this case 17 MPa), changes in the coefficient of variation $v_{f'_m}$ and spatial correlation length $\theta_{\ln f'_m}$, did not significantly alter the mean of the lateral resistance distribution but did change the standard deviation of the lateral load resistance as shown in Figure 3.14b. The latter figure demonstrates that an increase in the coefficient of variation and spatial correlation length of f'_m results in an increase in the standard deviation $\sigma_{R'}$ of the lateral resistance of the infill wall. The higher the coefficient of variation of f'_m, the greater standard deviation of $\sigma_{R'}$ as well as greater effect of spatial correlation of f'_m on $\sigma_{R'}$. Assuming. then, that the lateral load resistance is normally distributed, the probability of failure for the masonry infill wall can then be calculated once the mean, $\mu_{R'}$, and standard deviation, $\sigma_{R'}$, are known.

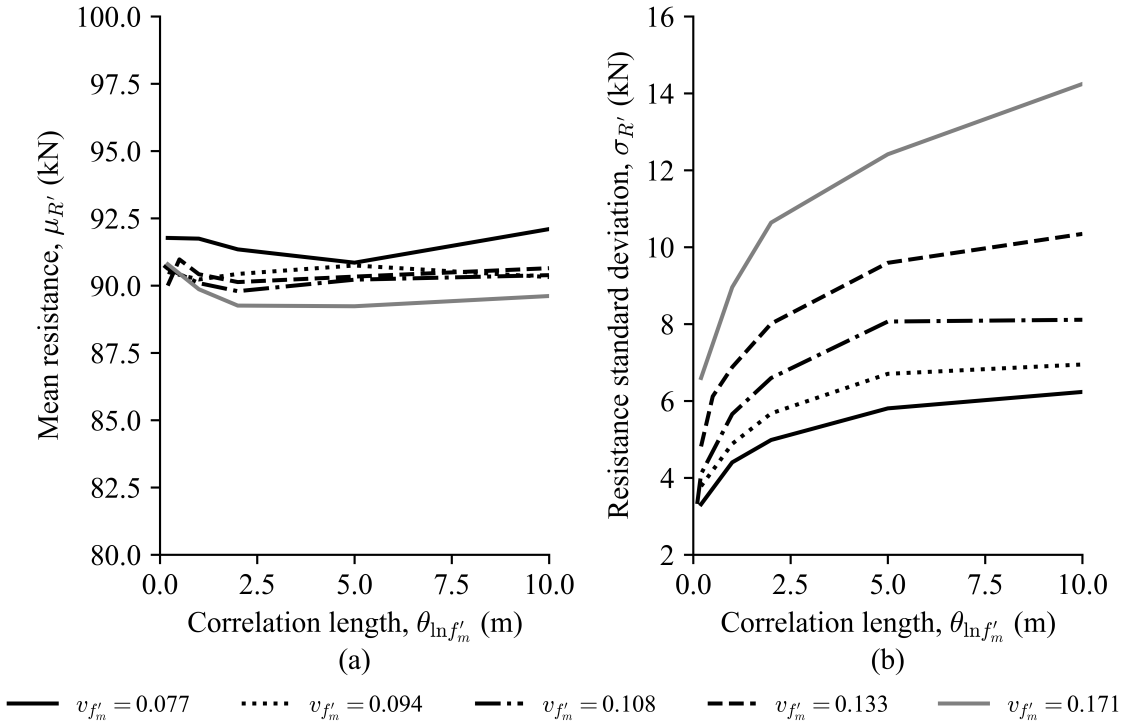


Figure 3.14. Effects of the random filed simulation of f'_m on the lateral resistance of the infill wall; (a) mean value of the lateral resistance $\mu_{R'}$ and (b) standard deviation of the lateral resistance $\sigma_{R'}$ vs. the spatial correlation length of f'_m

3.6.2 Analytical-based formulation

To compare with the finite element simulation-based results, an analytical approach was used to estimate the distribution of the lateral resistance of infill walls without the need for Monte-Carlo simulations. For evaluating the masonry infill strength, the diagonal strut model as specified in the Canadian masonry design standard (CSA S304-14, 2014) was used. According to CSA S304-14, for masonry infills of typical geometric and material properties, corner crushing is the predominant failure mode of masonry infills. The S304-14 also provides an equation for calculating the corner crushing strength as follows:

$$P_r = \chi \ \left(0.85 \ f'_m\right) \ w_{eff} \ t_c \tag{3.5}$$

where P_r is the capacity of the diagonal strut in the direction of the strut. w_{eff} is the effective width which is half of width of the diagonal strut w_s. The factor χ is to account for the direction of the compressive stress in a masonry member relative to the direction used during the material test, in this case, the masonry prism compression test. The factor 0.85 is to account for the equivalent stress block over the cress-section of the diagonal strut (CSA S304-14, 2014). In Equation 3.5 t_c is the depth of compression zone determined as:

$$t_c = \begin{cases} 2\,t_f - r, & r \leq t_f \\ t_f, & r > t_f \ \& \ r \leq t - t_f \\ t - r, & r > t - t_f \end{cases} \tag{3.6}$$

where r is the distance from the edge of the block within the tension flange to the compressive zone in the cross-section, and t_f is the thickness of the face-shell.

The capacity of the diagonal strut is dependent on the masonry compressive strength f'_m. The masonry compressive strength is, in reality, an averaged strength. In this research, geometric averaging was selected as opposed to arithmetic averaging due to the fact that geometric averaging more strongly weights the lower values of f'_m which are crucial in determining the resistance of the diagonal strut (both arithmetic and geometric averaging are discussed by Fenton and Griffiths (2008) in detail). Therefore, the geometric averaging of the masonry compressive strength over the volume of the diagonal strut was determined as follows:

$$\ln f'_m(v) = \frac{1}{V} \int_V \ln f'_m(\vec{p})\, d\vec{p} \tag{3.7}$$

where $\ln f'_m(v)$ is the local average of $\ln f'_m(x)$ over the volume of the diagonal strut V. $\ln f'_m(x)$ is the natural logarithm of the masonry compressive strength at any point, $\vec{p}$, in the continuum of the diagonal strut. The following relation shows that the local averaging does not change the mean $\mu_{\ln f'_m}$ as the expected value of $\mu_{\ln f'_m}$ does not change by averaging over the volume of the diagonal strut.

$$\begin{aligned} E\left[\ln f'_m(v)\right] &= E\left[\frac{1}{V} \int_V \ln f'_m(\vec{p})\ d\vec{p}\right] \\ &= \frac{1}{V} \int_V E\left[\ln f'_m(\vec{p})\right]\ d\vec{p} = E\left[\ln f'_m(\vec{p})\right] \end{aligned} \tag{3.8}$$

However, averaging would affect and reduce the variance of the resistance as follows:

$$
\begin{aligned}
Var\left[\ln f'_m(v)\right] &= E\left[\left(\frac{1}{V}\int_V \ln f'_m(\vec{p}_1)\,d\vec{p}_1\right)\left(\frac{1}{V}\int_V \ln f'_m(\vec{p}_2)\,d\vec{p}_2\right)\right] \\
&= \frac{1}{V^2}\left(\int_V\int_V E\left[\left(\ln f'_m(\vec{p}_1)-\mu_{\ln f'_m}\right)\left(\ln f'_m(\vec{p}_2)-\mu_{\ln f'_m}\right)\right]d\vec{p}_1\,d\vec{p}_2\right) \\
&= Var\left[\ln f'_m(v)\right]\,\gamma(V)
\end{aligned}
\tag{3.9}
$$

where $\gamma(V)$ is the variance reduction function. Vectors $\vec{p}_1$ and $\vec{p}_2$ are two independent points in the volume of the diagonal strut. The variance reduction function is the integral of the correlation function (Equation 3.4) over the volume of the diagonal strut. In this study, the compressive resistance of the masonry was considered to be random in the plane of the wall. Therefore, the resistance reduction function can be calculated using the following quadruple integral:

$$
\begin{aligned}
\gamma(V) = \frac{1}{(w\,l_d)^2}\int_0^{l_d}\int_0^{w}\int_0^{l_d}\int_0^{w} \\
\exp\left(\frac{-2\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}}{\theta_{\ln f'_m}}\right)dx_1\,dy_1\,dx_2\,dy_2
\end{aligned}
\tag{3.10}
$$

where x_1, y_1, x_2 and y_2 are the coordinates of the physical points $\vec{p}_1$ and $\vec{p}_2$ in the in-plane continuum of the diagonal strut. Also, l_d is the length of the diagonal strut.As the result of averaging, the averaged mean, $\mu_{\ln f'_{m_V}}$, and standard deviation $\sigma_{\ln f'_{m_V}}$ of the masonry

compressive strength in logarithmic space is calculated as:

$$\begin{aligned} \mu_{\ln f'_{m_V}} &= \mu_{\ln f'_m} \\ \sigma_{\ln f'_{m_V}} &= \sqrt{\gamma(V)}\,\sigma_{\ln f'_m} \end{aligned} \tag{3.11}$$

The averaged mean and standard deviation of f'_m was then calculated using Equations 3.2 and 3.3. To find the mean and standard deviation of the infill wall strength, the strength equation presented in Equation 3.5 was reformulated in the following to relate the lateral resistance distribution of the infill wall in terms of its mean μ_R and standard deviation σ_R with the averaged mean $\mu_{f'_{m_v}}$ and standard deviation $\sigma_{f'_{m_v}}$ of the masonry compressive strength f'_m.

$$\mu_R = \left(\frac{l}{\sqrt{h^2 + l^2}}\right)\ \chi\ \left(0.85\ \mu_{f'_{m_v}}\right)\ w_{eff}\ t_c \tag{3.12}$$

$$\sigma_R = \left(\frac{l}{\sqrt{h^2 + l^2}}\right)\ \chi\ \left(0.85\ \sigma_{f'_{m_v}}\right)\ w_{eff}\ t_c \tag{3.13}$$

where h and l are the height and width of the infill wall, respectively.

3.6.3 Calibration of the analytical prediction using the simulated-based estimate

The analytical resistance P_r of the masonry infill in the horizontal direction was calculated to be 95.8 kN according to the Equation 3.5, which is in a good agreement with the experimental result. The lateral resistance of the masonry infill in the experimental study

for the control specimen (IF-NG) was determined to be 94.3 kN (the system strength minus the bare frame load at failure). The χ factor was found to be 0.54 in this calibration which is in agreement with the S304-14 suggestion. Assuming $\chi = 0.54$, the mean resistance μ_R and resistance standard deviation σ_R are calculated using Equations 3.12 and 3.13, respectively.

Figure 3.15 compares the analytical- and simulation-based, mean and standard deviation of the infill wall for different coefficient of variations of masonry compressive strength, f'_m. In Figures 3.15a and 3.15b, the mean and standard deviation of the infill wall resistance are plotted against the correlation length of f'_m. Figure 3.15 shows that the analytical-based results agree with the simulation-based results and can be adopted to estimate the failure probability of the infill wall.

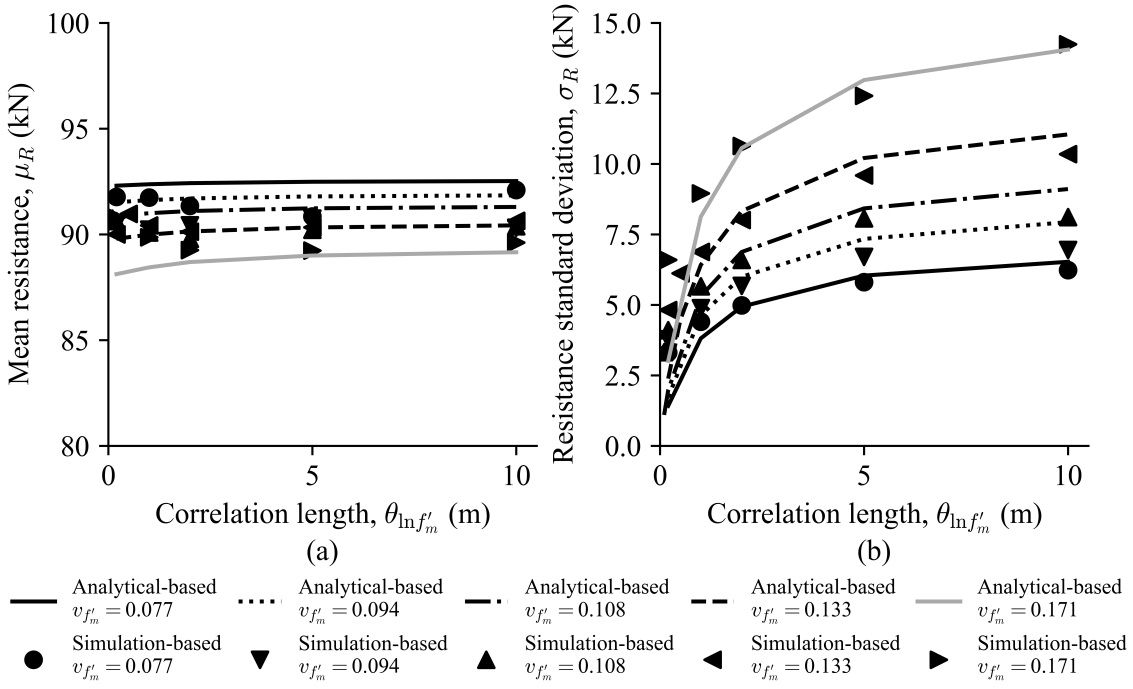


Figure 3.15. The analytical- and simulation-based standard deviation of the infill wall resistance vs. the correlation length

3.7 Estimating the design resistance factor

The resistance factor is a reduction factor which applies a safety margin to the design. The resistance factor is defined by the Load and Resistance Factor Design (LRFD) methodology (Fenton and Griffiths, 2008). Based on LRFD, a successful design of an infill wall should satisfy the following relation:

$$\phi_{mi}\,\hat{R} \geq \alpha \hat{L} \tag{3.14}$$

where $\hat{L_i}$ and $\hat{R}$ are the characteristic values for the load and the resistance, respectively. α is the load factor and ϕ_{mi} is the infill wall's resistance factor. In this research, as the masonry infill walls are mainly lateral load resisting systems, the seismic load was considered as the loading to be designed for. It was assumed that the design load is deterministic and resistance has a lognormal distribution. The design load was based on a deterministic earthquake load with a known return period (ie. see Naghibi and Fenton, 2019). The resistance factor required to achieve a target failure probability for infill walls was estimated to be dependent on the return period of the earthquake. The mean value of the lateral load μ_L is assumed to be defined in terms of the characteristic load, $\hat{L}$, as follows:

$$\mu_L = k_L\,\hat{L} \tag{3.15}$$

where k_L is the bias factor. The bias factor is used to provide margin of safety by decreasing the probability of load being higher than the characteristic load. The mean value

of resistance is also defined using the characteristic resistance as follows:

$$\mu_R = k_R \, \hat{R} \tag{3.16}$$

where k_R is the bias factor defined as the ratio of the mean resistance to the characteristic resistance. Normally k_R is less than or equal to 1.0. In this research, as the mean resistance μ_R is known, through both simulation and analytical prediction, the equivalent design load can be calculated using Equations 3.14, 3.15, and 3.16 as follows:

$$\mu_L = \left(\frac{k_L}{k_R}\right)\left(\frac{\phi_{mi}}{\alpha}\right) \mu_R \tag{3.17}$$

In Equation 3.17, both k_L and k_R are assumed to be 1.0. The load factor, α for seismic loading was considered 1.0 according to the National Building Code of Canada (NBCC, 2015). The next step is to determine the resistance factor, ϕ_{mi}, for a target reliability index β. The reliability index is determined from the maximum allowable probability of failure, p_m, as follows:

$$p_m = p\left[\frac{R}{L} < 1\right] = \Phi\left(\frac{\ln \mu_L - \mu_{\ln R}}{\sigma_{\ln R}}\right) = \Phi\left(-\beta\right) \tag{3.18}$$

from which the target reliability can be expressed as follows:

$$\beta = \frac{\mu_{\ln R} - \ln \mu_L}{\sigma_{\ln R}} \tag{3.19}$$

The target reliability index β for each limit state is available in CSA S408 (2011).

Figure 3.16 plots the target reliability index, β, vs. the resistance reduction factor, ϕ_{mi}, for infill walls. For instance, considering the coefficient of variation of 15% for the masonry compressive strength, as suggested by CSA S304-14 (2014), results in $\phi_{mi} = 0.80$ for the target reliability of $\beta = 3$. The resistance factor currently specified by CSA S304-14 for masonry is 0.60. This study indicates that the current resistance factor may be conservative.

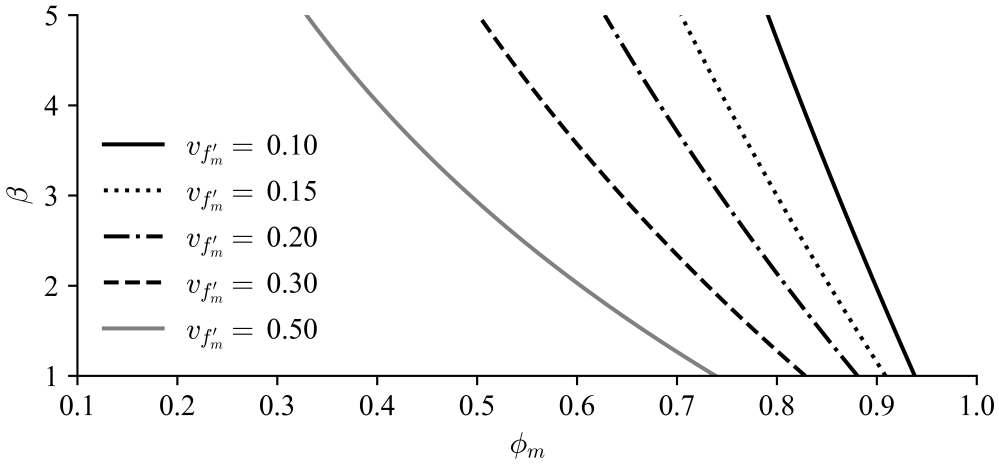


Figure 3.16. The target reliability index vs. the resistance reduction factor

It should be noted that the recommended ϕ_{mi} was determined based on random field simulation of only one parameter, the masonry compressive strength f'_m, and the seismic load was considered to be deterministic. While studies on other parameters are needed for a comprehensive recommendation, the study provides a framework for using the RFEM for reliability assessment of masonry infilled frames.

3.8 Conclusions

This paper presents the implementation of the Random Finite Element Method (RFEM) in estimating the resistance factor for designing the masonry infill walls. The masonry compressive strength was treated as a spatially variable random field to generate different

realization sets of masonry infills with varying standard deviations and correlation lengths. Each realization was numerically analyzed using the developed finite element model. The good agreement of the ultimate load and stiffness as well as the load vs. displacement response between the numerical and experimental results suggested that the model can successfully be incorporated in the RFEM simulations. The lognormal distributions fitted to the histogram of lateral load resistances of masonry infills revealed that the mean resistance remains the same while the resistance standard deviation depends on the correlation length and the coefficient of variation of masonry compressive strength. The successful calibration of the proposed analytical formulations using the simulation-based distributions resulted in proposing a resistance factor of 0.80 for reducing the lateral resistance of infill walls when designing for the target reliability index of 3.

3.9 Acknowledgement

The authors wish to recognize the contribution of financial assistance by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

Chapter 4

Parallelized finite element library for modeling applications in structural engineering

Reza Rahimi, Yi Liu, Gordon A. Fenton

Submitted to the Canadian Journal of Civil Engineering.

4.1 Abstract

This paper presents the development of a finite element algorithm using a vectorized and parallelized finite element method as a modular and open-source framework for finite element applications in structural engineering. Encoded in C++, the unique feature of the algorithm is its use of the Data-oriented Design (DoD) paradigm with the single instruction multiple data (SIMD) class of parallel computing to accelerate the stiffness matrix calculation and assembly of the global stiffness matrix. The algorithm is also designed to utilize the compressed sparse matrix format for data storage to achieve memory efficiency for finite element simulations. This parallel computing algorithm can run finite element models on either central processing units (CPUs) or graphical processing units (GPUs). Numerical examples of both linear and non-linear engineering problems are provided to show that the algorithm is capable of providing accurate results. The comparison of finite element

runtimes on CPU's vs GPU's demonstrates that current GPU's can easily outperform even the most advanced current CPU's.

Keywords: Finite element methods, GPU accelerated applications, Stiffness matrix calculations, Global stiffness matrix assembly, Data-oriented design.

4.2 Introduction

The Finite Element modeling technique has been widely used as an effective analysis tool in the prediction of structural behaviour for both research and design practice. In recent years, advancements in computing technology and hardware have enabled the use of finite element techniques in increasingly sophisticated and complex predictions of structural behaviour incorporating material and geometric non-linearities and large numbers of elements. However, the accuracy of these sophisticated structural models are often at the expense of increased computing time and cost due to the simulation complexity. For example, while the Modified Compression Field Theory (MCFT, Vecchio and Collins, 1986; Sadeghian and Vecchio, 2018) is an efficient non-linear finite element technique used to model the detailed cracking behaviour of brittle materials such as masonry and concrete, the implementation of such a method is computationally expensive due to the large number of iterations needed to achieve a specified level of precision in the non-linear solution process. Another example of computationally expensive problems is reliability analysis using stochastic methods, such as the Random Finite Element Method (RFEM) developed by Fenton and Griffiths (2008), to evaluate the failure probability of structural systems. RFEM employs the Monte Carlo simulation technique to determine the reliability index of a structural system as influenced

by a range of material, geometric, and loading characteristics of the system. This type of analysis typically involves thousands of random field realizations, each of which involves a finite element analysis (see, e.g., Rahimi et al., 2017), which has been taking a considerable computing time on conventional computers.

In an effort to develop more computationally efficient finite element analysis packages, the Object Oriented Programming (OOP) paradigm has been carefully studied (McKenna et al., 2010; Prud'Homme et al., 2012; Alnæs et al., 2015; Abhyankar et al., 2018; Alzetta et al., 2018) where the key features in the OOP design, such as abstraction, encapsulation, modularity and code reuse have been shown to increase computational efficiency (Mckenna, 1999). The OOP programming is based on grouping data and their functionalities in an class. Objects are instances of classes and can interact with each other. The key advantage of OOP is polymorphism which allow grouping objects with similar functionalities. For instance in a finite element code two different types of elements could be two different objects while still having similar functionalities inherited from the element class. The drawbacks of the OOP paradigm are that first, the functions are restricted to pieces of data and second, accessing an object, such as an element object in a finite element code, results in "cache pollution", a situation in which unnecessary data is loaded into the CPU cache while the useful data is moved to a lower level in the memory hierarchy (Fabian, 2013). However, the main limitation of OOP design is in parallel processing due to the risk of multiple threads of the processor attempting to concurrently access the same data. One solution to this problem is to arrange the data in contiguous blocks which allows the data to be sequentially processed. The resulting programming method is referred to as the data-oriented design

(DoD) paradigm, where the focus is on the arrangement of the data in memory to increase performance. In DoD the data could be structured close to the desired output so a minimum amount of operations is needed to be preformed through the functions. DoD has gained a considerable amount of interest in the game design industry (Fabian, 2013). Another key feature of the DoD algorithms the development of parallel algorithms using SIMD class of parallel computing as the functions and data are septate, and the functions can be applied on any piece of data. Such features enable the DoD algorithms to be implemented on the massively parallel structures available on modern graphical processing units (GPUs) to increase the computational efficiency of finite element analyses.

Recent developments in parallel processing algorithms on GPU devices have been focused on either optimizing the linear algebraic operations, such as sparse matrix-vector multiplications (Wong et al., 2015), or on porting the source code developed for the CPU to the GPU device (e.g., Komatitsch et al., 2009). Several recent studies have aimed to employ such algorithms to accelerate finite element analyses. One example is the work conducted by Bui et al. (see, e.g. Bui et al., 2017, 2019) for real-time surgical simulations where finite element analyses were greatly accelerated by employing a large number of processing units for the simultaneous execution of parallelized instructions. Many of the available finite element packages which run on GPU devices focus on solving the linear system of equations in parallel (Georgescu et al., 2013; Tian et al., 2015) while leaving the rest of the finite element process (stiffness matrix calculation and assembly) on the CPU device. The main limitations of such finite element applications are the serial assembly step and the high overhead associated with copying data back and forth from CPU memory to GPU memory.

The main objective of this study was then to design data-oriented algorithms to take advantage of modern GPU parallel architectures for the entire finite element analysis process, specifically utilizing parallelization capabilities available on CPU and GPU devices to accelerate finite element simulations. The DoD algorithms were developed to allow simultaneous execution of a single instruction on all available processing cores to achieve acceleration. For the stiffness matrix calculation step, the instructions for calculating the local stiffness matrix remains the same for each integration point. For the matrix assembly step, acceleration was achieved by vectorization and parallelization of the process through the sparse-matrix conversion from a coordinate format to a compressed format. For solving the linear system of equations, both direct and iterative methods were implemented in the code. Both solution methods have been widely studied in the literature and their implementation using DoD programming can be found in CUDA documentations (NVIDIA Corporation, 2018a), Helfenstein and Koko (2012) and Sharma et al. Sharma et al. (2013).

Another goal of this study was to develop the above-mentioned algorithms of an entire finite element process into an open-source modular finite element library available in the public domain. At the moment, there are more than a thousand open-source finite element repositories available on GitHub (Wanstrath et al., 2008). However, only a few of these repositories are libraries that actually provide clear instruction documentation. Examples of libraries which include documentation are MFEM (Kolev and Dobrev, 2010), FEEL++ (Prud'Homme et al., 2012), DEAL.ii (Alzetta et al., 2018), and FEniCS (Alnæs et al., 2015). A plate and shell finite element model simulated using FEniCS (Alnæs et al., 2015) indicates that such libraries are capable of solving complex finite element problems. Both DEAL.ii

and FEniCS libraries are built on top of the PETSc library (Abhyankar et al., 2018) to parallelize the finite element process. However, PETSc lacks the capability of assembling matrices on GPU devices. The main application of the above-mentioned libraries is to solve general differential equation problems such as Laplace, Poisson, etc., using the finite element method. However, structural analysis finite element models involve a variety of specific solutions to differential equation models of structural elements in one-, two-, and three-dimensions which are not provided by the MFEM, FEEL++, DEAL.ii and FEniCS libraries.

The few structural analysis finite element packages which employ parallel architectures are summarized by Georgescu et al. (2013). These packages are all proprietary, including names such as ANSYS, ABAQUES, etc., and are all expensive with high maintenance costs. Despite these parallelized proprietary packages, to the authors' knowledge, there are no finite element packages that actually optimize the runtime of advanced finite element analyses, in particular with respect to the stiffness matrix calculation and assembly steps, on multiple CPU and GPU devices for structural analysis purposes.

The algorithms developed in this paper were encoded using the C++ programming language and they can be run on a single CPU core, multiple CPU cores or GPU threads. The primary application for the algorithms is to accelerate structural finite element simulations, especially important for cases where running the finite element models numerous times is required. This is often encountered in parametric studies or random field simulations where a finite element analysis needs to be performed multiple times with different geometric or material parameters.

4.3 GPU architecture

The detailed explanation of GPU architecture is available in several recent studies (Kirk et al., 2007; Lindholm et al., 2008; NVIDIA Corporation, 2016a). For easy reference, the key terminologies and their definitions pertaining to this paper are summarized as follows:

- Kernel: an object composed of a scheduler and instructions to be scheduled. The scheduler assigns the instructions to various threads.
- Thread: the sequence of program instructions that can be managed independently and run simultaneously by the scheduler (Butenhof, 1997).
- Compute Unified Device Architecture (CUDA): A parallel programming environment developed by NVIDIA for general computing on GPU devices.
- Streaming multiprocessor (SM): a part of the GPU device that executes CUDA instructions. Each SM contains execution units capable of performing math operations on both integers and floating-point numbers, as well as a scheduler to organize and assign the instructions to different threads (Wilt, 2013).
- Single instruction multiple data (SIMD): a particular kernel instruction that runs on all processing units in a streaming multiprocessor which can operate on different data types, such as integer and floating-point values (Bolz et al., 2003).
- Block: a group of threads with shared memory space which are executed by a single multiprocessor.
- Grid: a collection of blocks executing a single instruction.

4.4 Data-oriented design of the finite element method

The algorithm developed in this study is referred to as the Vectorized and Parallelized Finite Element Method (VPFEM) (Rahimi et al., 2019), https://github.com/vpfem. The method vectorizes and parallelizes finite element processes by transforming matrices into column vectors and optimizing the algorithm for running on multiple processing cores. This data-oriented algorithm shifts the focus in finite element programming from the elements to the data. In particular, the global stiffness matrix is now viewed in terms of its data, rather than as a series of elements. Figure 4.1 presents the difference in data management by DoD and OOP techniques in finite element modeling of a structural shear wall as an example. Figure 4.1a, and Figure 4.1b show the example shear wall under lateral loading and its two-dimensional finite element model, respectively. Figure 4.1c represents the data arrangement used in DoD algorithms which groups the data based on their usage while Figure 4.1d depicts the data arrangement into different objects (elements) to be used by OOP designed algorithms. The algorithms developed herein utilized the SIMD techniques which can be implemented on GPU devices through their massively parallel architecture to achieve acceleration. In this study, since the DoD algorithms are less modular than OOP designed algorithm, only the most computationally expensive steps, such as the stiffness matrix calculation and assembly, and solving the system of equations, were designed using the DoD technique while the rest of the finite element code, such as input geometry and material behaviour, was designed with OOP to utilize its modularity.

A class hierarchy of the system, presented in Figure 4.2, was designed to allow the use of different modules (such as higher-order elements or advanced constitutive material

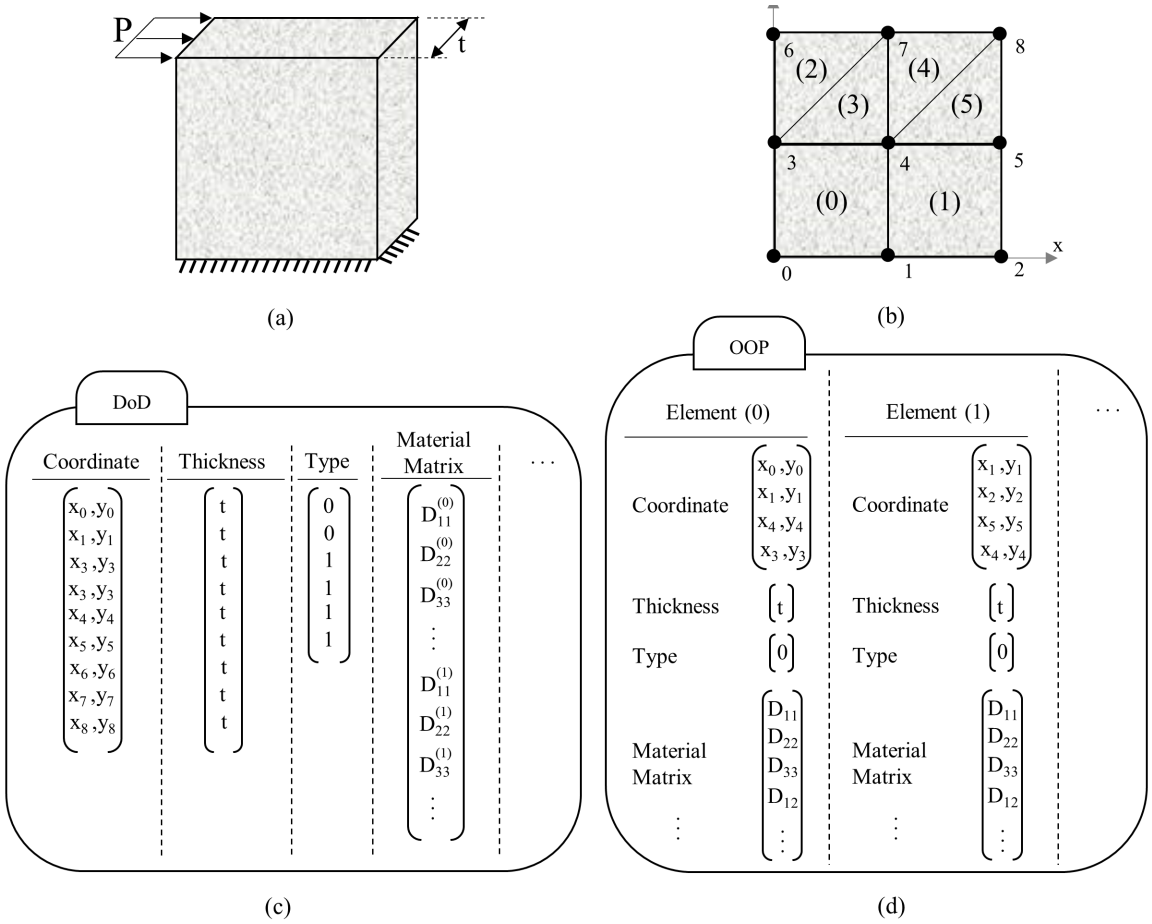


Figure 4.1. Data representation in DoD vs OOP programing styles a) an example shear wall subjected to lateral loading b) a 2-dimensional finite element presentation of the shear wall c) data arrangement in DoD algorithms d) data arrangement in OOP designed algorithms

models) and the choice of processor type (e.g., single CPU, multiple parallelized CPUs, or GPUs). These choices can be made by choosing objects from the relevant classes rather than by writing specific instructions for each processor. The entire source code of the VPFEM algorithm is available in the GitHub repository (Rahimi et al., 2019) along with multiple applications, which are test/examples of a finite element analysis using the VPFEM algorithm. The following subsections describe the details of the class hierarchy system used and the algorithms developed in the VPFEM to build a parallelized finite element framework.

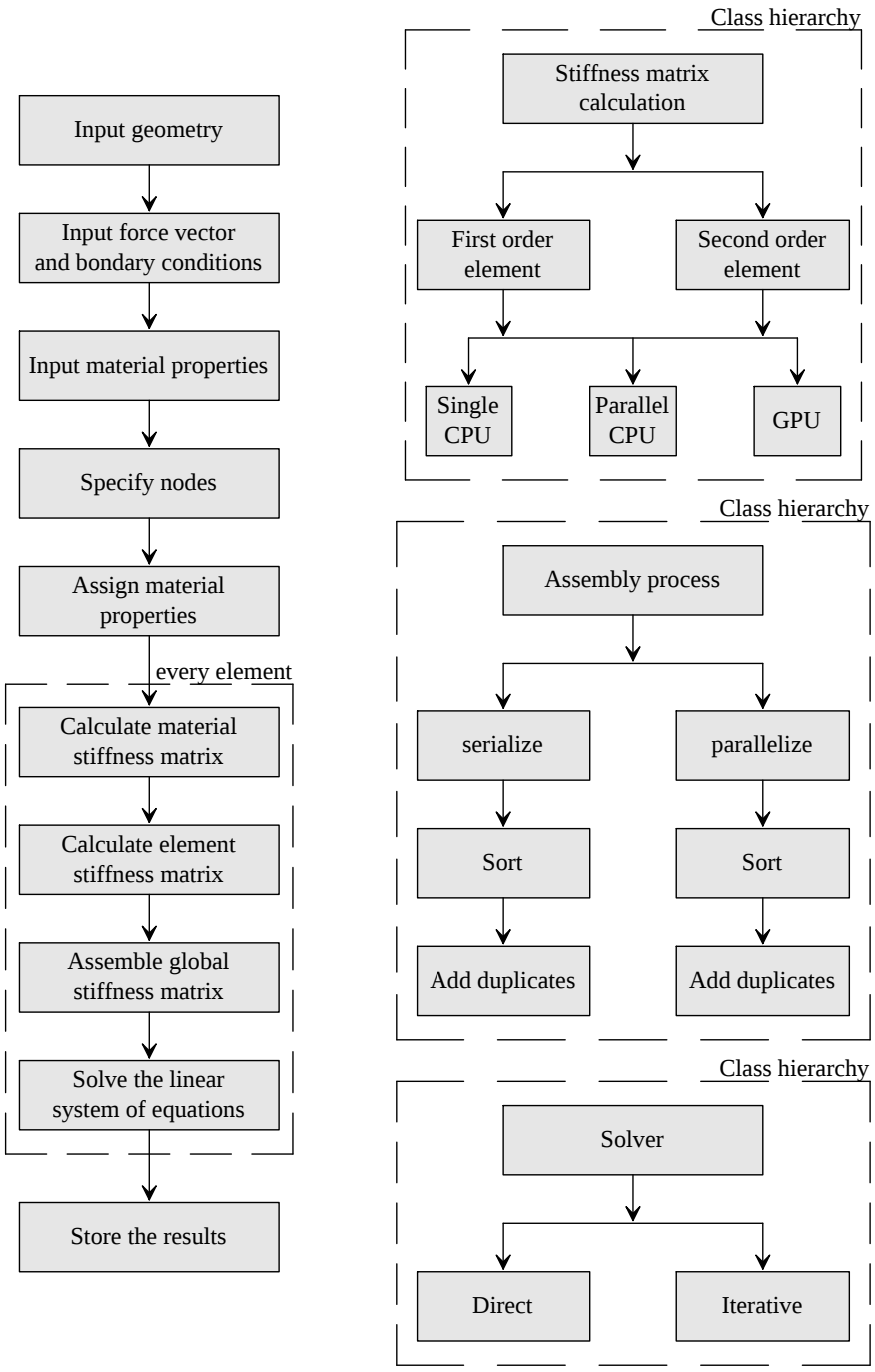


Figure 4.2. VPFEM flowchart for running nonlinear finite element analysis

4.4.1 Geometry and material classes

The input variables relating to the geometry, loads, and boundary conditions of a structure for finite element analyses are categorized under the Geometry Class. In particular, these

variables are the coordinates, forces, and boundary conditions at each node, as well as the list of nodes composing each element. These variables are managed using the functions *Node*, *Fix*, *Load* and *Mesh*, which use dynamic arrays to store the data. The main role of the Geometry Class is to organize the data so that they can be efficiently loaded into the processor's cache memory for the stiffness matrix calculation step. The data are constructed of low-level arrays containing structures consisting of coordinates, free Degrees of Freedom (DOFs), mesh, and material stiffness matrices as seen in Figure 4.1c.

The Material Class is responsible for filling in the material stiffness matrices. The Material Class assigns material behaviour through child classes, such as elasticMaterial, elasticPerfectlyPlasticMaterial and parabolicMaterial. The modular feature of the library allows the user to add as many material model child classes as required. This Class calculates the material stiffness matrix using the material constitutive model and updates the material stiffness matrix if the analysis requires an iterative process, as in some non-linear finite element methods (see, e.g., Vecchio, 2000). The Material Class also sorts the material stiffness matrices according to the element numbers and stacks them into an array to be used by the DoD algorithms in the Stiffness Matrix Class (Figure 4.1c), as discussed next.

4.4.2 Stiffness matrix class

The Stiffness Matrix Class is responsible for both creating a sparse matrix that stores all of the local stiffness matrices (Figure 4.3a) and assembles the local stiffness matrices into the global stiffness matrix (Figure 4.3b). The formulation used to calculate a fully populated stiffness matrix is described in the literature (Vecchio, 2000; Bathe, 2006). In this study, the

equations used to calculate the stiffness matrix were simplified and encoded to minimize the number of math operations required at each integration point. The stiffness matrix of each integration point can be calculated by any available thread. Therefore, the total number of instructions which can be run in parallel is equal to the total number of integration points in the finite element model. For example, a finite element model having n quadrilateral elements, each having $m = N \times N$ integration points, can run $n \times m$ instructions in parallel using the algorithm developed herein.

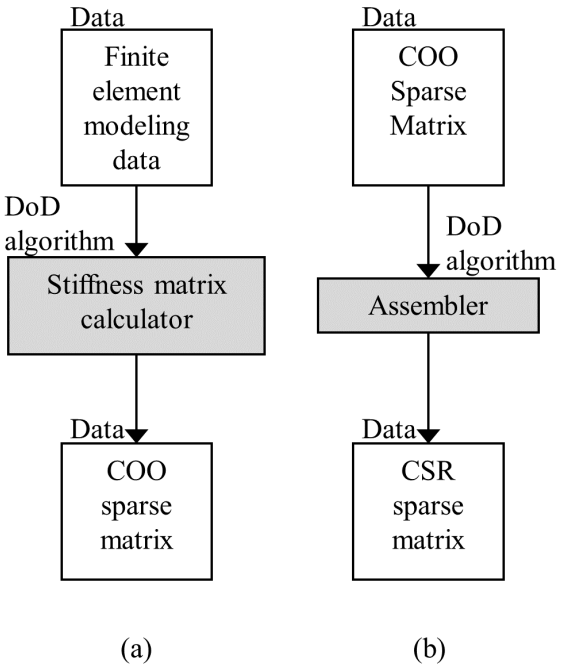


Figure 4.3. DoD algorithms used in a) calculating b) assembly of the global stiffness matrix

For the assembly step, the conventional serial approach consists of allocating a memory address to each non-zero entry in the global stiffness matrix and accumulating values of entries sharing the same degree of freedom at that memory address. However, this approach, if attempted to be performed in parallel, will result in a so-called “race condition”, where two parallel processes attempt to write data to a specific memory address at the same time.

This is an undesirable situation as both processes are "racing" to access/change the data. Different solutions have been proposed by several researchers, such as graph colouring (Cecka et al., 2011), graph partitioning (Klöckner et al., 2009), reduction list (Komatitsch et al., 2009), atomic operation (Fu et al., 2014; Talebi et al., 2014; Cui et al., 2018), and matrix-free or assembly-free methods (Margetts, 2002; Smith et al., 2013; Martínez-Frutos and Herrero-Pérez, 2015; Shterenlikht et al., 2018). The main drawback of these suggested methods is the overhead cost by preprocessing to identify the DOFs required in the global stiffness matrix.

This study addressed the overhead problem by converting the global stiffness matrix from a coordinate format (COO) to a Compressed Sparse Row format (CSR), as illustrated in Figure 4.3b. The CSR format focuses on the data which allows the algorithm to be parallelized. The theory and algorithms required to convert a COO format matrix into a CSR matrix are discussed in detail in the next several subsections.

Parallelization of the local stiffness matrix on GPU devices

On GPU devices, a kernel function is used to schedule tasks amongst the GPU threads and separate them into blocks and grids. The number of threads that form a block and the number of blocks that form a grid are dependent on the number of streaming multi-processors available in the GPU architecture. The GPU scheduler used for this study is the CUDA scheduler (NVIDIA Corporation, 2018b). The local element stiffness matrices were calculated at all the integration points of an element and were locally added in shared memory to form the element stiffness matrix. Algorithm 4.1 shows the instruction used to

calculate the local stiffness matrices.

Algorithm 4.1. Instructions for parallelizing the stiffness matrix calculation

Data: x,y, D and mesh (vectors)

Result: k (COO sparse matrix)

Operations ← number of elements × number of integration points

Kernel *parallelStiffnessMatrixCalculation(Operations)*

```
/* divide the Operations between available threads        */
stiffnessMatrixCalculation(x^(e), y^(e), D^(e), mesh^(e))    // SIMD instruction
```

Function *stiffnessMatrixCalculation*($x^{(e)}$, $y^{(e)}$, $D^{(e)}$, $mesh^{(e)}$)

```
/* Initialize the local matrix to zero                     */
K ← 0
/* Find the natural coordinate values and the integration
   weights integral at each integration point              */
set ξ_i, η_j, w_ij
/* calculate the Jacobian and the strain matrix            */
calculate J and B
/* calculate the entries of the local stiffness matrix     */
k_ij ← Σ w_ij |J| (B^T D B)_ij
```

For CPU devices, a task manager was developed in the VPFEM algorithm to schedule instructions as a series of tasks to be executed on the available threads in a CPU device. Each task is basically a group of instructions required to calculate the stiffness matrix of an element, and each task stacks the instructions required to be run for all the integration points in each element on a single thread. Each thread sums the local stiffness matrices of all element integration points in the local memory and stores the resultant local stiffness matrix into the global memory. Because of the high speed of the local memory, adding the stiffness matrix of different integration points in the local memory results in an accelerated process and improved efficiency in memory use. To accomplish this, the task manager first

creates a pool of parallel threads based on either the user defined number of threads or the maximum number of threads available in the system. Then the task manager creates a queued system of jobs, which holds a certain number of tasks depending on the size of the finite element model, waiting to enter the thread pool. Also, the task manager constantly monitors the threads in the parallel pool to ensure that all the threads are running at full potential.

Storing local stiffness matrices

The next step for both the CPU and GPU algorithms is to stack the resultant local stiffness matrices of all elements into global memory using a sparse matrix format COO (Algorithm 4.1). The key advantages of the algorithm developed herein, compared to OOP designed algorithms, are first that the stiffness matrices are not parts of different objects, such as element objects. Therefore, loading the data into the processor memory would not pollute the cache. Secondly, the process can be performed in parallel as instructions and the data are independent, allowing the use of SIMD class of parallel computing.

The resulting COO matrix consists of three vectors of equal length containing the DOFs and values of each non-zero entry of the global stiffness matrix. The length of the vectors is the number of non-zero entries of the global stiffness matrix, which corresponds to the number of DOFs. The first two vectors, called DOF_i and DOF_j, store the DOFs associated with each entry of the global stiffness matrix. The third vector, called the $data$ vector, holds the stiffness value of each entry.

Parallelization of the global stiffness matrix assembly

The instructions for assembling the global stiffness matrix using a parallel algorithm are depicted in (Algorithm 4.2). As can be seen, the parallel algorithm first requires the entries in the data vector to be sorted numerically according to both the DOF_i and DOF_j vectors. The next step is to add duplicate entries in the sorted vectors, setting the extra entries to zero and separating non-zero and zero entries. Each thread in the parallel algorithm is responsible for adding the duplicates in each row of the global stiffness matrix simultaneously. The algorithm finally updates the Number of Non-Zero (NNZ) entries upon completion. The non-zero entries are then used to form the row pointer and column indices vectors in a stiffness matrix having a CSR format. The steps in the algorithm are discussed in more details as follows.

Algorithm 4.2. Assembly step

Data: $k.DOF_i, k.DOF_j, k.data, NNZ$

Result: K `// global stiffness matrix in CSR format`

Function *Assemble(matrix)*

```
/* sort local stiffness matrix                                  */
K = sort(k, compare(m, n))
/* Add duplicates in the local stiffness matrices               */
K = addDuplicates(K)
/* separate the local stiffness matrices into non-zero and
   zero entries                                                 */
K = partition (K)
/* convert the COO matrix to CSR format                         */
K = csrRow(K)
```

Sorting algorithm The output of the stiffness matrix calculation step is a COO matrix which is the bundle of local stiffness matrices of all elements. The matrix needs to be sorted row-by-row to be converted to a CSR format. The sorting methodology adopted herein uses a comparison operation that determines which of the two entries of the stiffness matrix should occur first in the global stiffness matrix. The comparison function sorts both the rows and entries in each row of the sparse coordinate format matrix by reordering the DOF_i and DOF_j vectors in an ascending order (Algorithm 4.3). Depending on either single or multiple processors, different sorting algorithms were used in this study to convert the stack of local stiffness matrices to a global stiffness matrix.

Algorithm 4.3. Comparison sort algorithm

Function *Compare(m, n)*
  if $k.DOF_{i_m} == k.DOF_{i_n}$ **then**
    boolean $\leftarrow k.DOF_{j_m} < k.DOF_{j_n}$
  else
    boolean $\leftarrow k.DOF_{i_m} < k.DOF_{i_n}$
  end
return boolean

For a single CPU core, the serial comparison sorting algorithm, IntroSort (Musser, 1997), was adopted to sort the stiffness matrix. IntroSort is a hybrid sorting algorithm which combines Quicksort, Heapsort and insertion sort to minimize the runtime. The time complexity of the IntroSort algorithm in worst case is $O(N \log N)$ when sorting N values. For multi CPU or GPU cores, a parallel sorting algorithm based on the MergeSort (invented by Neumann in 1945, i.e. see Knuth, 1998) algorithm, was developed in this study. To parallelize the MergeSort algorithm, the stack of local stiffness matrices was first divided

into sub-matrices, as shown in Figure 4.4. A vector of unsigned integer numbers was created to hold the address of the first and last entry of each sub-matrix. The task manager and scheduler discussed in the stiffness matrix calculation step was used to manage the parallel processes on CPU and GPU devices, respectively. The goal was to create a thread pool and send each of the sub-matrices to a separate thread for sorting.

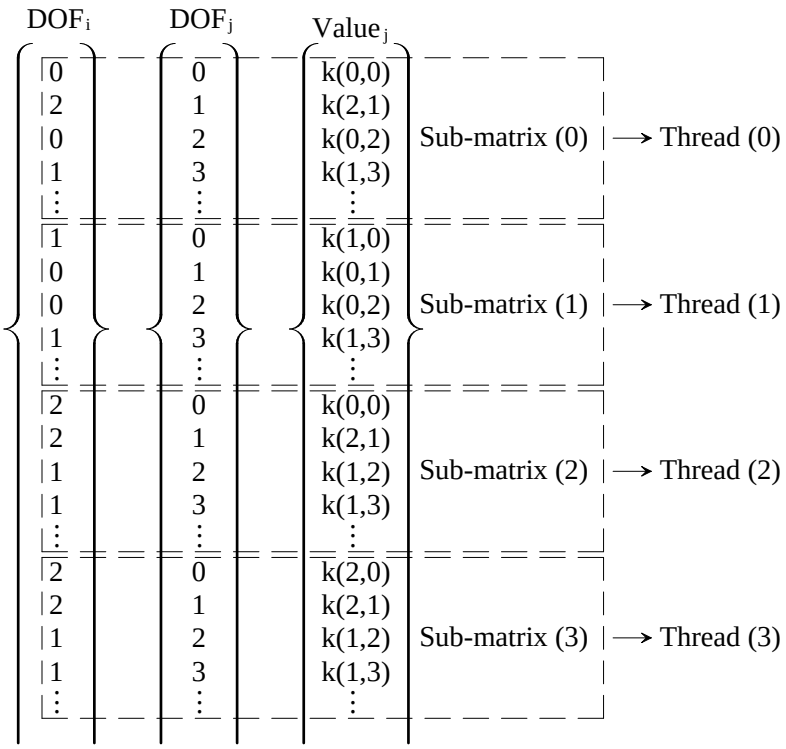


Figure 4.4. Dividing the stiffness matrix into sub-matrices for parallel sorting

In this study, the sorting algorithm used in each thread was IntroSort (Algorithm 4.4).

After sorting sub-matrices, the next task was to merge the sorted sub-matrices into a sorted global stiffness matrix. The merge function inputs two vectors of the same size, then picks a pivot and uses a rotate function available in Standard Template Library (STL, 2017) to rearrange the data into an ordered set of elements. The merge instruction can then be performed recursively on subranges of the input vectors to achieve a sorted vector with

Algorithm 4.4. Parallel sort algorithm

```
Function ParallelSort()
    for all sub-matrices do
        EachSort(sub-matrix[i])
    end
Kernel EachSort(k, sub-matrix[i], Compare(m,n))
    IntroSort(k.begin() + sub-matrix[i][0], k.begin() + sub-matrix[i][1],
     compare(m,n))
```

twice the size of the input vectors. The merge is usually a serial operation and difficult to parallelize. The VPFEM algorithm parallelized the merge process by merging adjacent pairs of sub-matrices using separate threads. The merged sub-matrices are twice the size of the original sub-matrices and are referred to as updated sub-matrices. Then, the task manager recursively repeats the process until the global stiffness matrix is fully merged and sorted.

Figure 4.5 illustrates the successive merging steps required to achieve the final sorted global stiffness matrix. For example, sub-matrices 1 and 2 in the left hand column of the figure are merged into the larger submatrix in the central column of the figure. The merge process involves secondary sorting process so that the larger sub-matrix is also properly sorted (see STL , 2017). The maximum number of merges required is $M - 1$ where M is the total number of sub-matrices in the left hand column of Figure 4.5.

The VPFEM algorithm sorts the matrices in-place which only involves moving indices so that no data copying is required. Using moves considerably decreases the computational expense of the sorting process. As mentioned above, the time complexity of the MergeSort algorithm is $O(N \log N)$ when sorting N values on a single serial processor. If both the

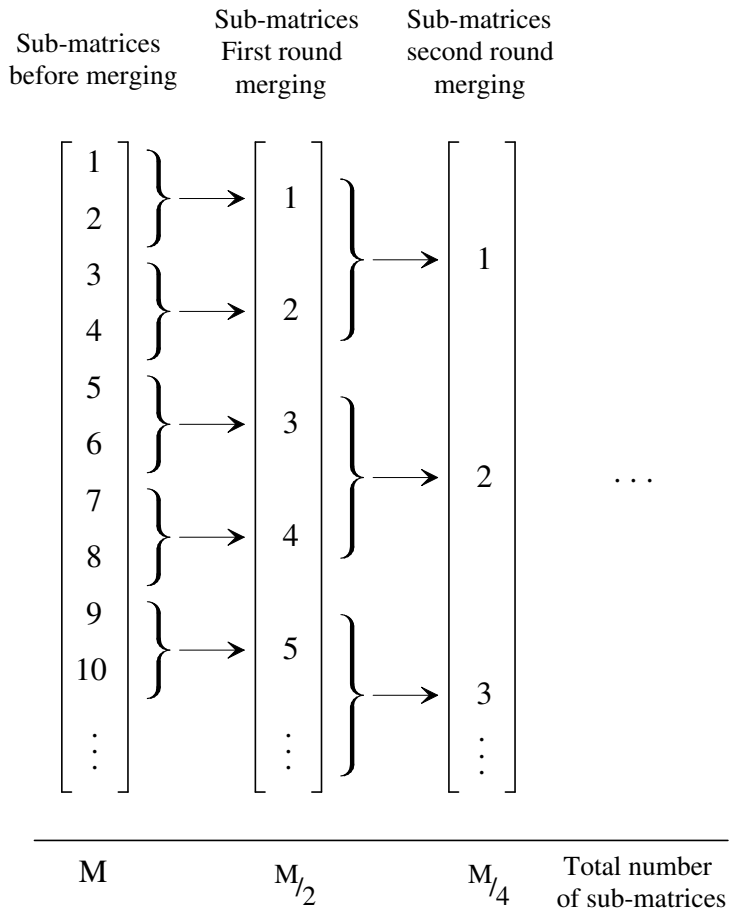


Figure 4.5. Different steps in merging the sorted sub-matrices to a sorted matrix

sort and merge algorithms are done in parallel using many processors the time complexity changes to $O((\log N)^3)$ (Cormen et al., 2009). In other words, parallelizing the sorting algorithm on GPU devices with many available processors can result in a substantial reduction in runtime. For instance, the time complexity of a case where $N = 1000$ in the serial approach is $O(3000)$ while it could be reduced to $O(27)$ if parallelized using many core architectures.

Adding duplicate entries After the stiffness matrix has been sorted, the next task is to remove the duplicate entries. Duplicate entries in a sparse COO format of a stiffness

matrix are the entries with identical DOF_i and DOF_j values. Duplicate entries occur in the stack of local stiffness matrices whenever there is at least one node shared amongst two or more elements in the finite element model. The stiffness values of these shared entries must be added together, then the duplicate entries can be removed. This process, if done using a serial algorithm, is generally computationally expensive. The algorithm developed herein parallelizes the process of removing duplicate entries using the instructions shown in Algorithm 4.5 using the SIMD class of parallel computing. After adding the duplicate entries, the duplicate value is set to zero to be removed from the stiffness matrix later. The main reason for not removing the duplicate values now is it might cause race conditions if attempted in parallel. In Algorithm 4.5, each thread is to go through a row of the global stiffness matrix and find the duplicates, add the value to the original entry, and set the duplicate value to zero.

Upon completion of Algorithm 4.5 for the entire stiffness matrix, the zeroes are then removed from the matrix to optimize for the solution step. Erasing an entry from a dynamic array in C++ is a computationally expensive task, since the remainder of the array must be shifted by one index which involves many copy operations. The time complexity of the erase operation is the linear distance between the first and last iterator in a vector. The solution, developed for the VPFEM algorithm, was to partition the stiffness matrix so that all the zeros were placed at the end of the array rather than deleting them one by one. A stable partitioning algorithm, called *Stable Partition Position*, was developed in this study to reorder the global stiffness matrix and put all the zeros at the end of the array. This algorithm travels through the array to find the zeros and reorders the array using the *rotate*

Algorithm 4.5. Instructions for removing the duplicate entries

```
Function addDuplicate(rowNumber, rowPtr, Dof_j, value)
    while counter < rowPtr [rowNumber +1] do
        increment ← 1
        while col[counter] == col[counter + increment] do
            value[counter] += value[counter + increment]
            value[counter + increment] = 0
            increment++
        end
        counter += increment
    end
Kernel addDuplicateKernel(TotalRow, rowPtr, Dof_j, value)
    int i = blockDim.x * blockIdx.x + threadIdx.x
    if i < TotalRow then
        addDuplicate(i, row, col, value)
    end
```

function. The rotate function, available in STL, (2017), is an instruction that rotates the order of entries in a range using a circular shift operation. The circular shift operation avoids having to re-order the entire vector on each zero removal. While the complexity of the rotate function is also the linear distance between the first and the last iterator, the implementation is more efficient as it performs a move rather than a copy assignment, where in C++, move is more efficient than copy because the move operation casts the reference of an object instead of cloning the object.

To convert the sorted global stiffness matrix from COO to CSR, the DOF_i and DOF_j vectors were simply changed to row pointer and column index vectors. The row pointer vector is of size $r_p + 1$ where r_p is the number of rows in the matrix. The last entry of the row pointer vector contains the NNZ entries in the matrix. The column index vector is of

size NNZ and stores the column index of each non-zero entry in the global stiffness matrix.

4.4.3 Solving the system of equations

Once the global stiffness matrix is formed, the next step is to solve the linear system of equations. The global stiffness matrix is usually a Hermitian, positive-definite matrix. Direct and iterative solvers were adopted in the VPFEM library to solve the linear system of equations (e.g. see Martínez-Frutos and Herrero-Pérez, 2015; Vetterling et al., 1999; Chandrupatla et al., 2002). Direct methods are more useful for small sized stiffness matrices but would likely fail for larger sized stiffness matrices. It is difficult to define an exact upper bound for the size of the stiffness matrix for direct solvers as they are dependent on the shape and condition number of the stiffness matrix, which in turn depend on the specific finite element problem. As the main goal of the algorithm developed herein was to accelerate the finite element analysis for parametric and random field studies where finite element simulation needs to be repeated numerous times, the first run of the finite element model is used to investigate the efficiency of the various solvers. Then the algorithm can choose the most efficient solver for the subsequent simulations.

Figure 4.6 shows the efficiency of different solvers versus the number of elements in a nonlinear finite element problem. It can be seen that the direct solver (Cholesky decomposition) was the fastest up to about 2500 elements. Above 2500 elements iterative conjugate gradient methods became more efficient. These approximate “rules” were built into the VPFEM library to choose a solver in the event that the solver type was not specified. If the selected solver is unable to reach convergence, the next fastest solver would be tried.

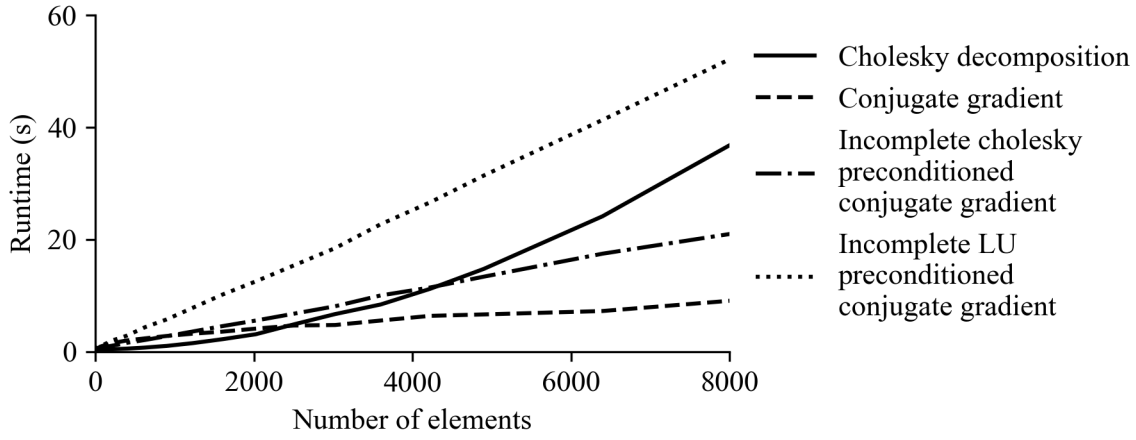


Figure 4.6. Performance of different solvers versus number of elements used in a nonlinear finite element example

It should be noted that the results in Figure 4.6 are highly dependent on the type and size of the finite element problem as well as on the computing system configuration. Iterative methods may require less memory but at the cost of increased computations, depending on the convergence behaviour. For example, if the convergence is only achieved in a large number of iterations then the iterative method may require more computations than the direct method.

4.5 Implementation example

The VPFEM algorithms discussed in this paper are encoded in C++ as a static library called the VPFEM library. The library, including example applications, is available in the public domain (Rahimi et al., 2019) at https://github.com/vpfem. Two examples of structural problems, one linear and one nonlinear, are presented as follows.

4.5.1 Linear elastic modeling of a cantilever beam

To demonstrate the performance of the algorithms developed in the VPFEM library, an example application was constructed consisting of a finite element model of a cantilever beam as shown in Figure 4.7.

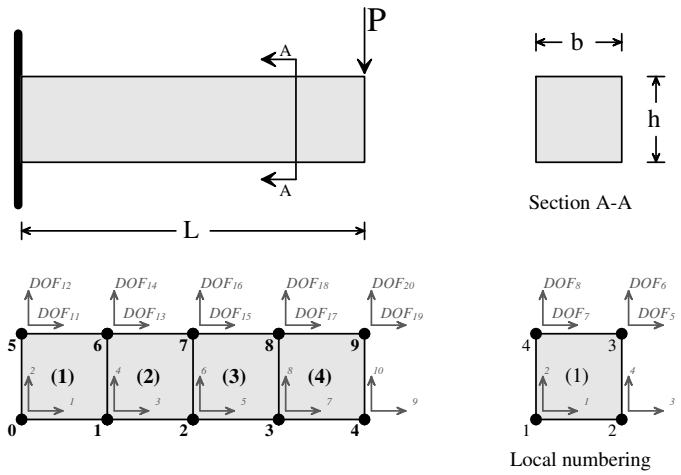


Figure 4.7. Geometry and mesh discretization of the example cantilever beam

First, the accuracy of the numerically estimated tip deflection, Δ_n, provided by the VPFEM library was compared to the analytical solution of the tip deflection, Δ_a. Figure 4.8 presents the results of Δ_n/Δ_a as a function of the number of elements used in the model. In this example four node isoparametric quadrilateral elements, each having four integration points, were used to model the cantilever beam. Also, in the figure, the results obtained using OpenSees (McKenna, 2011), are shown for comparison. OpenSees is a modular, open-source framework for developing finite element modeling of structural and geotechnical systems. Note that "Ex" and "Ey" in the figure represent the number of elements used in the beam in the x and y directions, respectively. It can be seen that the VPFEM library provides more accurate displacement estimate with fewer elements than

OpenSees.

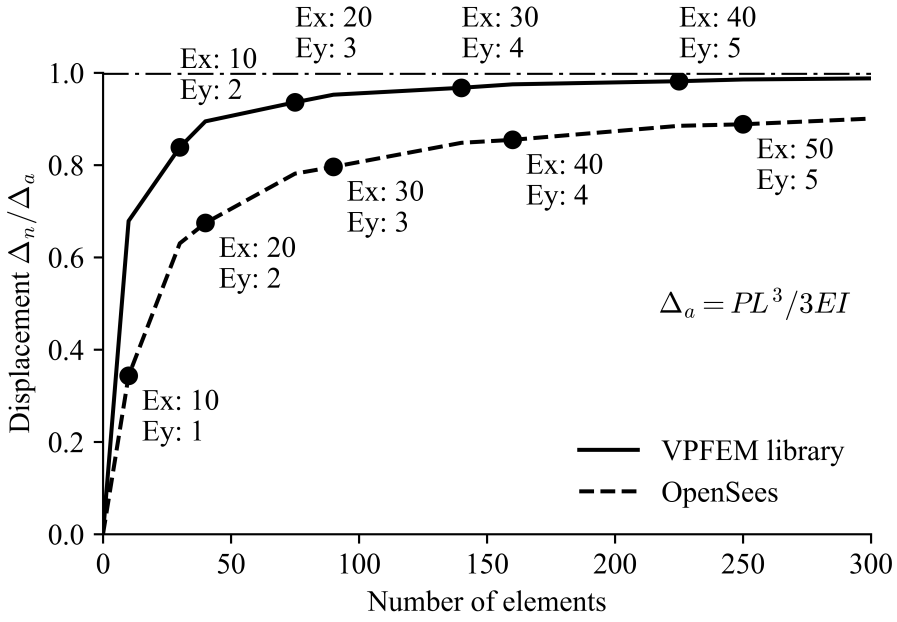


Figure 4.8. Numerical displacement vs. number of elements obtained in the VPFEM library and OpenSees

Next, the example cantilever was analyzed using the VPFEM library on different processor configurations including multiple CPUs, GPU devices, and a single CPU. In this example, 10 million quadrilateral elements having 16 integration points each were used to model the cantilever beam. The resulting finite element model has 20,022,002 degrees of freedom and 160 million local stiffness matrices. At the assembly stage the input COO matrix, which is the stack of calculated local stiffness matrices, has approximately 5.76 billion entries. It is recognized that this level of complexity is not necessary to calculate the behaviour of a cantilever beam. However, a large finite element analysis such as this allows the speed of the developed parallel algorithms to be properly tested against serial

Table 4.1. Details of the processing devices used to test the VPFEM library

Name	Device	Cores in use	Clock speed (GHz)	Memory (GB)	Memory type	Memory speed	Price ratio*
CPU(1)	Intel Xeon CPU E5-2630 v4	1	2.20	128	DDR4	2400 (MHz)	1
CPU(4)	Intel Xeon CPU E5-2630 v4	4	2.20	128	DDR4	2400 (MHz)	1
CPU(8)	Intel Xeon CPU E5-2630 v4	8	2.20	128	DDR4	2400 (MHz)	1
CPU(10)	Intel Xeon CPU E5-2630 v4	10	2.20	128	DDR4	2400 (MHz)	1
CPU(20)	Intel Xeon CPU E5-2630 v4	20	2.20	128	DDR4	2400 (MHz)	2
GT 1030	Nvidia GPU	384	1.47	2	DDR5	3 (Gbps)	0.06
GTX 1050	Nvidia GPU	640	1.46	2	DDR5	7 (Gbps)	0.09
GTX 1060	Nvidia GPU	1152	1.71	6	DDR5	8 (Gbps)	0.16
GTX 1070	Nvidia GPU	1920	1.68	8	DDR5	8 (Gbps)	0.18
Titan Xp	Nvidia GPU	3840	1.58	12	DDR5	11.4 (Gbps)	0.8

* The price ratios (GPU cost divided by cost of a 10 core Xeon CPU) are calculated based on the prices of the CPU and GPU devices in Canada at the time of writing the paper, not including the price of other components of the host computer.

algorithms. The hardware specifications of the CPUs and GPUs used in this comparison are presented in Table 4.1.

The computational runtime ratio of the VPFEM library using different processor configurations is compared in Figure 5.15. The runtime ratio, R, is defined as

$$R = \frac{\text{single CPU runtime}}{\text{multiple CPU or GPU runtime}} \tag{4.1}$$

Figures 5.15a and 5.15b plot the runtime ratio, R, of different hardware configurations in calculating and assembling the stiffness matrix, respectively. The runtime ratios calculated here are the result of running the example application multiple times and averaging the various runtimes to estimate the true execution time. This averaging removes variability due to operating system scheduling. Figure 5.15 shows that the parallel instructions and task managers implemented in the stiffness and assembly classes have the potential to improve the performance of the library significantly, depending on the hardware used. In the stiffness matrix calculation, the parallel instructions resulted in up to 189 times acceleration comparing to the serial instruction on a single CPU whereas the acceleration was up to 122

times for the assembly process. The better improvement achieved for the stiffness matrix calculation was due to the fact that the stiffness matrix algorithm was fully parallelized while the assembly algorithm was only partially parallelized due to the limitations caused by race conditions.

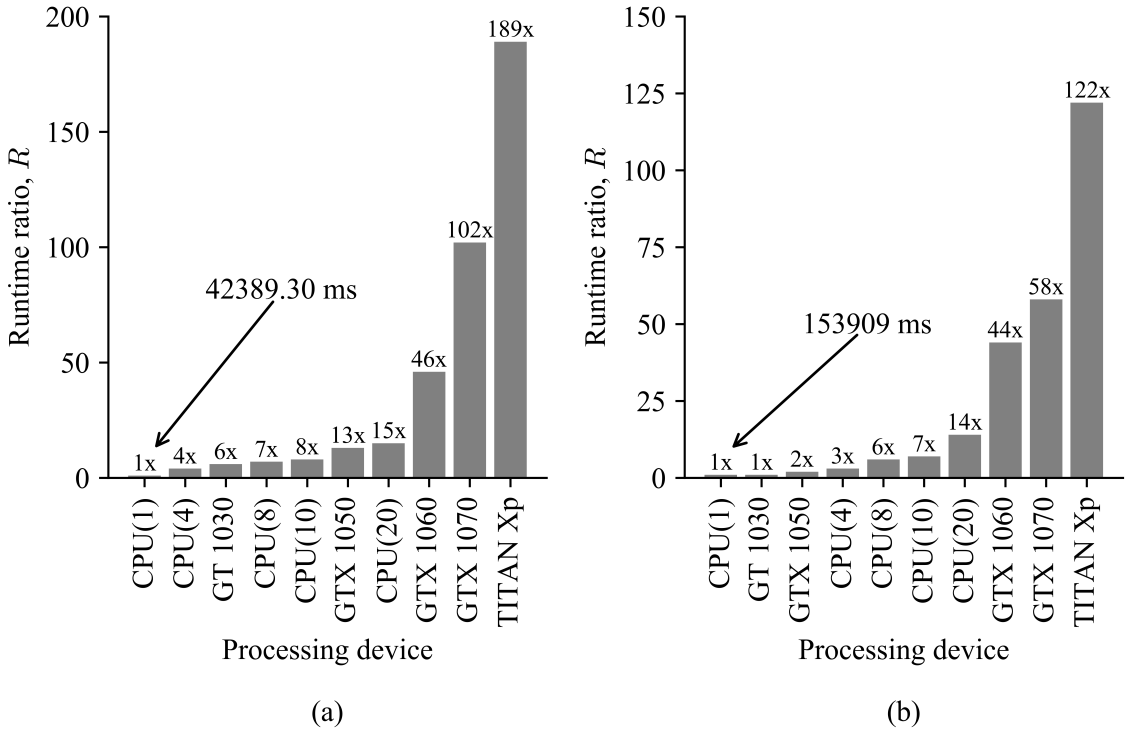


Figure 4.9. Runtime ratio of different hardware configuration in a) calculating local stiffness matrices b) assembling the global stiffness matrix

Figure 4.10 compares the total combined runtime of stiffness and assembly steps vs. the number of cores in a CPU device (Figure 4.10a) and a GPU device (Figure 4.10b). Both Figure 4.10a and Figure 4.10b show an approximately linear relationship between the runtime ratio and the number of CPU or GPU CUDA cores used. On average, the runtime

ratio using x number of CPU cores (threads) is estimated to be:

$$R = 0.74\,x + 0.2 \tag{4.2}$$

while the runtime ratio using c CUDA cores on a GPU device is approximately

$$R = \frac{1}{30}c + 3.38 \tag{4.3}$$

Comparing Equations 4.2 and 4.3 reveals that for a specific number of cores a CPU could out perform a GPU device. However, at time of writing the paper the most advanced CPU devices have up to 256 processing cores while GPU devices can have thousands of processing cores. Also, it should be noted the price per core for GPU devices is much less than the price per core ratio for CPU devices 4.1.

Note that the run time ratios of the GPU devices GTX1030 and 1050 (384 and 640 CUDA cores, corresponding to first and second dots on the horizontal axis of Figure 4.10b) do not well fit the linear trend of the more advanced GPU devices because of their limited memory capacities, which becomes the bottleneck in their total execution time.

It can be seen in Figure 5.15 that the finite element process experiences considerably more speed-up using the GPU device GTX 1060 (1152 CUDA cores, $R = 44$) than the CPU device with 20 processing cores (two Intel Xeon E5-2630 v4 processors, 20 cores, 40 threads, $R = 15$). In addition, the price of a CPU device is much higher than that of a GPU device. Table 4.1 shows the price ratios of GPU devices divided by the price of the 10 core Xeon CPU device used in this research. It is clear that utilizing a GPU device not only

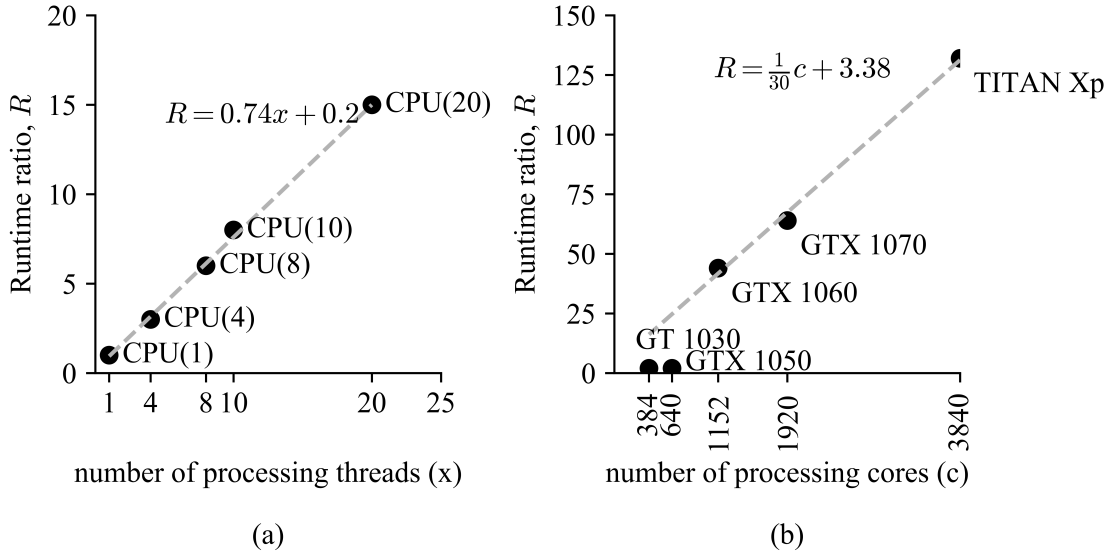


Figure 4.10. Runtime ratio vs. the number of cores in a) CPU devices and b) GPU devices

results in much faster runtimes, but also considerably decreases the computational cost of finite element simulations.

Next the cantilever example was used to test the performance of the VPFEM library against OpenSees and the results are shown in Figure 4.11. In this test, OpenSees 3.0.3 was used to run the model on a single CPU core. The VPFEM library was used to run the model on both a single CPU core (serial algorithm) and a GPU device (parallel algorithm). The CPU had the base clock speed of 2.2 GHz, 128 GBs of Error-Correcting Code (ECC) Random Access Memory (RAM) with a clock speed of 2400 MHz whereas the GPU device used houses 3840 CUDA cores with 12 GB RAM (Table 4.1). Figure 4.11 compares the runtime vs. number of elements obtained by the VPFEM library and OpenSees in simulating the cantilever beam example. It shows that the overall computational time of the VPFEM library is increasing much more slowly, if at all, than OpenSees, as the number of elements increases. At 841,000 elements, the VPFEM library parallel algorithm was able to run the

simulation of the cantilever beam 117 times faster than OpenSees. Even in the case of the VPFEM library run on a single CPU (serially), the 841,000 elements simulation achieved by the VPFEM library was still approximately 4 times faster than OpenSees.

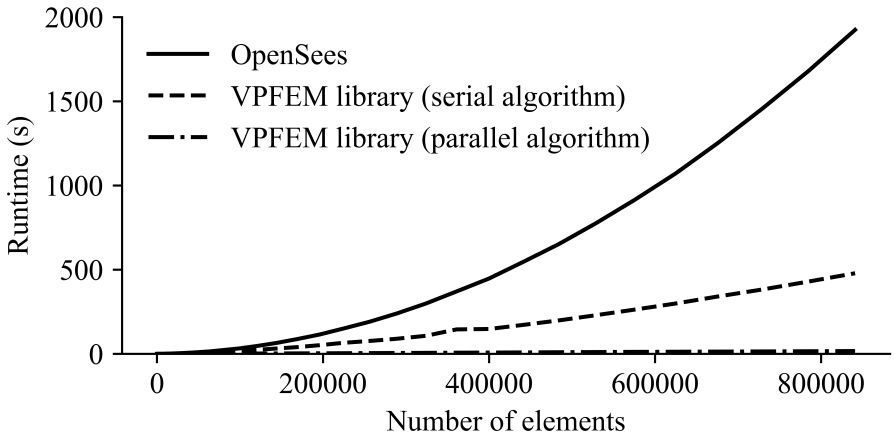


Figure 4.11. Performance comparison of the VPFEM library and OpenSees

4.5.2 Nonlinear modeling of a reinforced concrete shear wall

A reinforced concrete panel shear wall (specimen PB21 in Bhide, 1987) was used to test the performance of the parallelized algorithms in a nonlinear finite element problem. The material properties of the panel are presented in Table 4.2.

Table 4.2. Material properties of the reinforced concrete panel (Vecchio, 1990)

Material property	Value
Modulus of elasticity of steel	200,000 MPa
Yield stress of steel	402 MPa
Modulus of elasticity of concrete	24200 MPa
Compressive strength of concrete cylinder	21.8 MPa
Concrete cracking stress	1.54 MPa
Poisson's ratio	0.3
Strain in concrete cylinder at peak stress	0.0018

The geometry, loading and boundary conditions of the model is depicted in Figures

Figure 4.12a and 4.12b, respectively. The nonlinear behaviour of the reinforced concrete was modeled using the MCFT technique (Vecchio and Collins, 1986) which is based on an iterative secant stiffness formulation. The shear wall was also numerically modeled by Vecchio (1990).

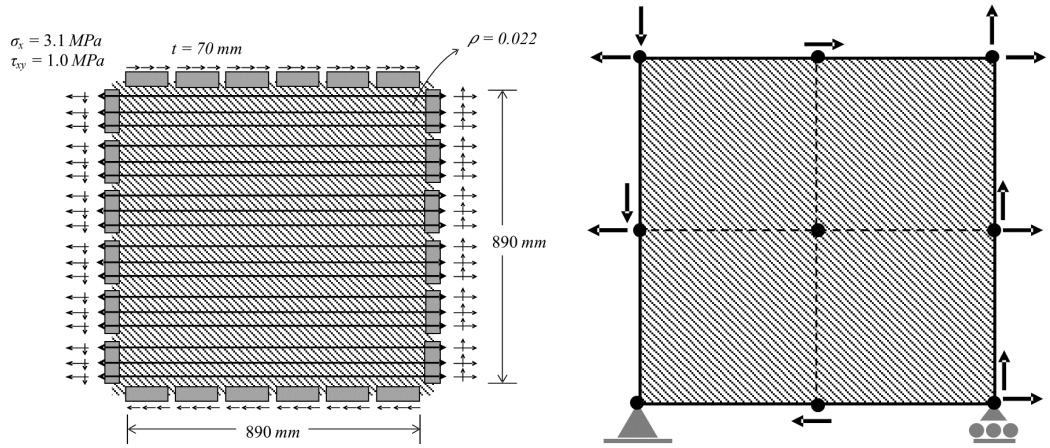


Figure 4.12. Details of the reinforced concrete panel PB21 a) specimen properties (Bhide, 1987), b) finite element model (Vecchio, 1990) (simplest four element case only shown)

Table 4.3 compares strain values computed using the VPFEM library and by Vecchio (1990) for each iteration of the nonlinear simulation where ϵ_x, ϵ_y and γ_{xy} are the normal strains in x and y directions and shear strain, respectively. The strains were calculated at each integration point then averaged to find the element strain. The max error is defined as the maximum percentage difference among the three strain values obtained by the VPFEM library and Vecchio (1990). The table shows that for all iterations, the VPFEM simulation results compare well with the results obtained by Vecchio (1990) with a maximum error of less than 1 percent.

The nonlinear example was then used to test the performance of the VPFEM library implemented on GPU devices. In this comparison, a single CPU (serial algorithm) with the

Table 4.3. Comparison of strain values obtained by VPFEM library and Vecchio (1990)

Iteration number	(Vecchio, 1990)			VPFEM library			Max error (%)
	ϵ_x 10^{-3}	ϵ_y 10^{-3}	γ_{xy} 10^{-3}	ϵ_x 10^{-3}	ϵ_y 10^{-3}	γ_{xy} 10^{-3}	
1	0.128	-0.038	0.107	0.128	-0.038	0.107	0.000
2	0.234	0.018	0.189	0.234	0.018	0.189	0.000
3	0.355	0.057	0.345	0.355	0.057	0.345	0.000
4	0.458	0.122	0.540	0.459	0.123	0.540	0.000
5	0.530	0.207	0.736	0.530	0.209	0.736	0.000
6	0.579	0.299	0.908	0.579	0.300	0.908	0.000
7	0.613	0.383	1.048	0.614	0.385	1.045	0.286
8	0.638	0.455	1.159	0.638	0.457	1.155	0.345
9	0.656	0.514	1.245	0.656	0.515	1.240	0.402
10	0.669	0.560	1.310	0.669	0.561	1.300	0.763
15	0.698	0.672	1.462	0.698	0.672	1.457	0.342
20	0.704	0.669	1.497	0.705	0.701	1.494	0.200
25	0.706	0.705	1.505	0.706	0.704	1.500	0.332
30	0.706	0.706	1.507	0.706	0.706	1.501	0.398
35	0.706	0.707	1.507	0.706	0.706	1.502	0.332

base clock speed of 2.2 GHz (Table 4.1) and the TITAN Xp GPU (parallel algorithm) (Table 4.1) were considered. The results are shown in Figure 4.13. As the number of elements used to model the wall increases, the acceleration of the simulation using a GPU device becomes increasingly pronounced. For example, at 235,225 elements, the respective model runtime was 6901.3 s on the single CPU device and 280.1s on the GPU device resulting in 24 times acceleration by the latter. The improvement in computational time shown by both examples suggests that the VPFEM library is capable of producing accurate simulation results as well as accelerating a simulation process significantly when implemented on GPU devices. The low cost of GPU devices and demonstrated computational savings make VPFEM a viable solution for engineering finite element problems, especially where numerous finite element model runs for multiple parameters are required.

It should be noted that the VPFEM library is open-source and an evolving project. At

the time of writing the paper the library can only perform finite element analysis under monotonic loading.

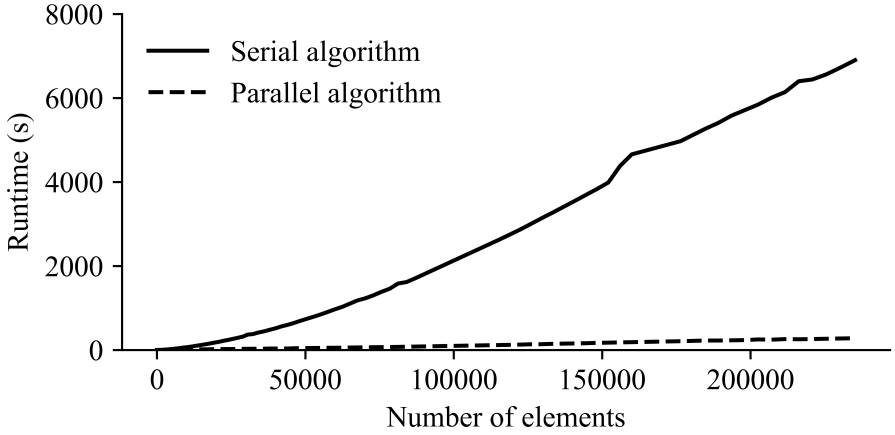


Figure 4.13. Performance of the VPFEM library in modeling the nonlinear behaviour of the reinforced concrete panel

4.6 Conclusions

This paper presents the development of parallelized algorithms for a modular, open source finite element library, VPFEM, to accelerate structural finite element simulations using parallelization capability on CPU and GPU devices. Encoded in the C++ programming language, the unique feature of the algorithm is its use of the Data-oriented Design (DoD) paradigm with the single instruction multiple data (SIMD) class of parallel computing to accelerate the stiffness matrix calculation and assembly of the global stiffness matrix. The algorithm also provides the flexibility of running finite element simulations on a single CPU core, multiple CPU cores, or multiple GPU threads. The performance of the developed algorithm was assessed through two numerical examples of structural engineering problems, one linear, the other non-linear. The results of both examples suggest that the VPFEM library

is capable of producing accurate simulation results and, when implemented on GPU devices, results in a significant acceleration of the finite element analysis. While this paper focuses on the development of a finite element library which can be run on multiple processor configurations, especially GPU devices, the benefits of clustering GPU devices, also in combination with CPUs to form a network for large finite element simulations, are being considered for future research.

4.7 Acknowledgement

The authors wish to recognize the contribution of financial assistance by the Natural Sciences and Engineering Research Council (NSERC) of Canada. Also, the authors gratefully acknowledge the support of NVIDIA Corporation through their donation of the Titan Xp GPU used in this research.

Chapter 5

Study of the in-plane behaviour of concrete masonry infilled RC frames using a parallelized finite element modelling technique with the distressed stress field method

Reza Rahimi, Yi Liu, Gordon A. Fenton

Submitted to the Canadian Journal of Civil Engineering.

5.1 Abstract

This paper presents further development of the Vectorized and Parallelized Finite Element Method (VPFEM) for modelling the in-plane behaviour of masonry infilled RC frames using the Distributed Stress Field Method (DSFM). The main objectives of the study were to implement the DSFM using the VPFEM library for masonry infilled frames, and to further develop algorithms specifically for this implementation to accelerate the finite element model run speed. In this study, the DSFM is used to model the three components of masonry infilled RC frames, i.e. the bounding frame, infill wall and mortar interface. The equilibrium and compatibility conditions, as well as material constitutive models, for each component, are discussed in the paper. The iterative analysis process of the DSFM in the VPFEM library was carried out using the Single Instruction Multiple Data (SIMD)

class of parallel computing techniques at each element level to achieve acceleration. A comparison with the experimental results showed that the DSFM is able to accurately model the behaviour of masonry infilled RC frames. The predicted ultimate load, load vs. displacement response and the cracking pattern were in a reasonably good agreement with the experimental results. The performance of the VPFEM library in accelerating the model run speed was demonstrated through comparisons of run speeds obtained using a single CPU and several commercially available GPUs. It was shown that GPU devices with adequate memory space can accelerate up to six times the model run speed when compared with a CPU and the degree of speed-up was highly dependent on the number of elements used in the finite element model.

Keywords: Finite element method, masonry infilled frame, VPFEM, GPU, Distributed stress field method, Secant stiffness approach, SIMD.

5.2 Introduction

The previous paper by Rahimi et al. (2020) described the development of a finite element library aiming to accelerate finite element simulations in structural engineering applications. Referred to as the Vectorized and Parallelized Finite Element Method (VPFEM), the algorithms of the library were developed to utilize the massively parallel architectures available in Graphical Processing Units (GPUs) to accelerate the key steps of a finite element simulation. The paper showed the library's capability of accurately predicting structural behaviour as well as its performance in accelerating the model run speed when implemented on GPUs. Building upon the previous paper, this paper employs the VPFEM library to be

used in a numerical study of in-plane behaviour of masonry infilled Reinforced Concrete (RC) frames.

Finite element modelling has been increasingly used as an effective tool in studies of in-plane behaviour of masonry infilled frames (Madan et al., 1997; Shing and Mehrabi, 2002; El-Dakhakhni et al., 2003; Hashemi, 2007; Mohebkhah et al., 2008; Stavridis and Shing, 2010; Nasiri and Liu, 2017; Rahimi and Liu, 2017). Due to the numerous possible combinations of various geometric, material, as well as loading parameters for the infilled frame system, the finite element modelling technique is a viable alternative for analyses to supplement physical data. Two main categories of modelling techniques, i.e., micro-modelling and macro-modelling, are commonly used to model masonry infilled frames. The micro-modelling technique models the individual blocks and mortar with their respective constitutive relationships. While capable of capturing the localized stress and failure patterns, the accuracy of this type of modelling is often at the expense of high computational cost (Mohebkhah et al., 2008; Sarhosis et al., 2014). On the other hand, in a macro-modelling technique, the infill is considered to be a continuum with a defined stress-strain relationship. The smeared cracking model is an example of a macro-model (Lotfi and Shing, 1991; Rahimi and Liu, 2017) where masonry is modeled with material laws that consider the mechanical properties and behavior of mortar joints and concrete blocks in a "smeared" fashion. This modelling technique is computational efficient and is reasonably accurate in modelling structures when the effect of local shear stress is negligible (Lu et al., 2013). However, for masonry infilled frames where shear behaviour is critical, this technique may be inadequate in capturing cracking patterns and thus potential failure modes. To overcome

these limitations, this paper implemented the Distributed Stress Field method (DSFM), originally developed by Vecchio (2000), to model the in-plane behaviour of infilled frames.

The DSFM is an advanced smeared cracking model based on the rotating crack concept. It has been used to model the in-plane behaviour of concrete and masonry shear walls (Sadeghian and Vecchio, 2018) and was shown to result in better simulation of cracking patterns and post-crack behaviour than conventional smeared cracking models (Vecchio, 2000; Facconi et al., 2014). The DSFM applies the rotating crack concept to the average macroscopic representation of the material behaviour with local shear stress. The method is also capable of incorporating the local rigid body movements, such as shear slip and mortar joint slip, and thus is believed to be a suitable modelling technique for simulation of masonry infilled frames. To the best knowledge of the authors, this study represents the first effort in implementing the DSFM in the numerical study of in-plane behaviour of masonry infilled frames.

This study has two main objectives. One is to implement the DSFM to model the in-plane behaviour of concrete masonry infilled RC frames. The second is to further develop the VPFEM library (Rahimi et al., 2020) to allow modelling masonry infilled RC frames using the DSFM. One key characteristic of the DSFM is that the material stiffness matrix of each element is modified progressively, as a result of the change in crack orientation, to achieve convergence in the joint displacement field. The process of updating stiffness matrix needs to be repeated at each iteration and as the number of iterations increases, the DSFM becomes computationally expensive. The parallelized algorithms of the VPFEM library can provide a significant acceleration of the DSFM in a finite element simulation

and achieve computational efficiency. Hence, the ultimate goal of this study is to develop a computationally efficient and technically reliable finite element modelling package for predicting the in-plane behaviour of masonry infill RC frames.

5.3 Finite element formulation of DSFM

In this study, the three components of an infilled frame, i.e. the bounding frame, infill wall, and infill-frame mortar interface were all modeled using the DSFM technique. Figure 5.1 illustrates the general state of stress (Figure 5.1a) as well as the specific local state of stress in a typical element (Figures 5.1b, 5.1c and 5.1d) for the aforementioned three components. The subscript c, m, and j, denote the stresses in the concrete bounding frame, the masonry infill, and the infill-frame mortar interface, respectively. As shown in Figure 5.1a, an element subjected to tensile and compressive stresses may experience cracking in the principal stress direction. In the DSFM method, the net strain field of the element is converted to the principal stresses which are used to estimate the direction of the cracked surface and determine the new secant stiffnesses for each load increment. Then, the secant stiffnesses are used to update the net strain field. The process continues until a desired level of convergence in either the displacements or the secant stiffnesses is achieved. Also, in each component region, appropriate constitutive laws need to be defined and equilibrium and compatibility conditions need to be satisfied over the element as well as at the cracked surface. The equilibrium and compatibility conditions, and constitutive laws for each component are discussed in the following section.

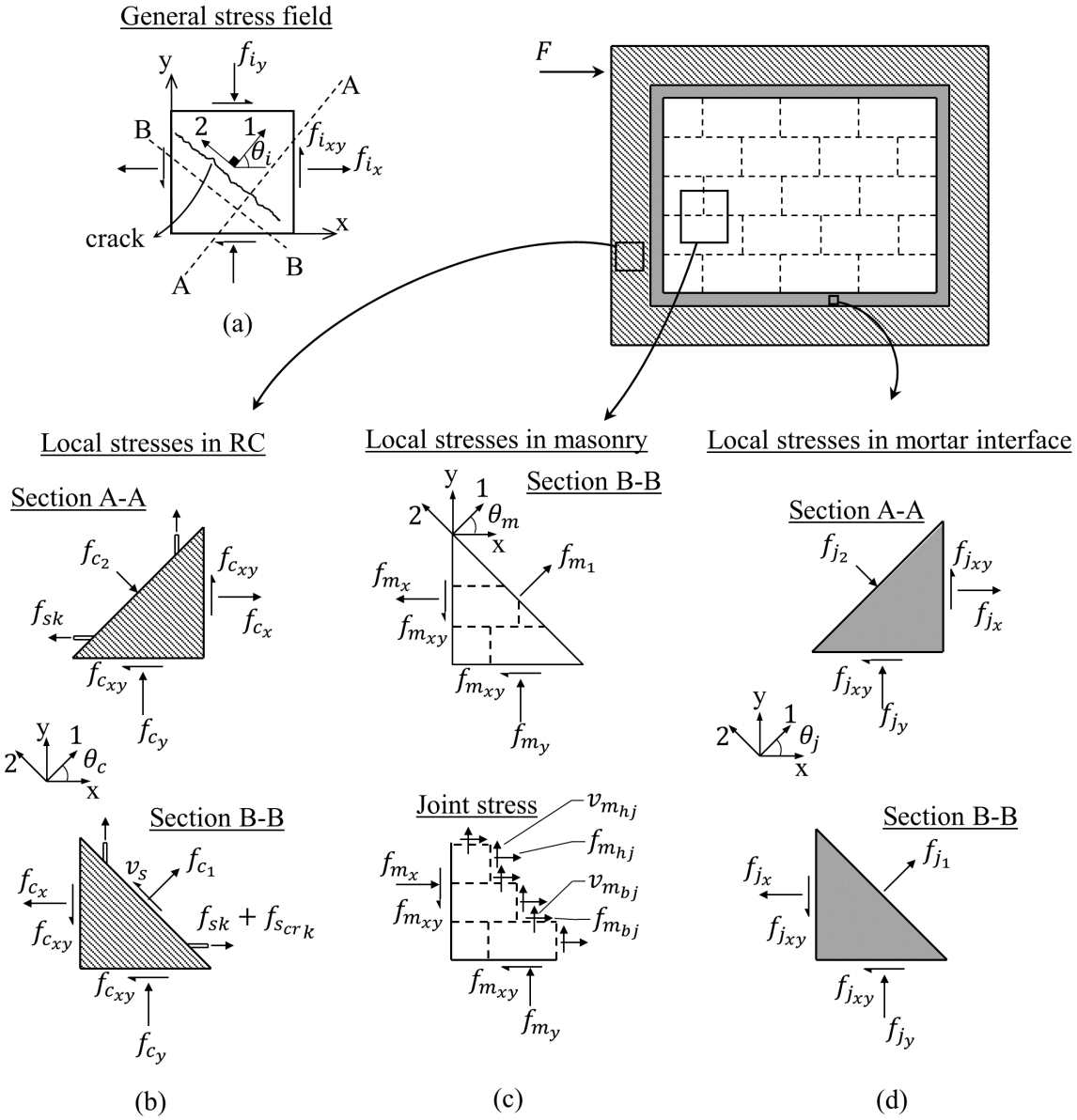


Figure 5.1. State of stress representation in an infilled frame: a) general stress field in a typical element; local stress field in b) concrete, c) masonry, and d) mortar interface elements

5.3.1 Equilibrium and compatibility conditions

In the RC frame (Figure 5.1b), the external force is assumed to be resisted by internal stress fields in the concrete and the strut and tie forces in the rebars. In the smeared crack model, the equilibrium within an element in the RC frame is achieved in an averaged smeared

condition as follows:

$$[f_c] = \begin{Bmatrix} f_{c_x} \\ f_{c_y} \\ f_{c_{xy}} \end{Bmatrix} = [D_c]\left[\epsilon_c^n\right] + \sum_{k=1}^{N}\left[D_{s_k}\right]\left[\epsilon_{s_k}\right] \tag{5.1}$$

where $[f_c]$ is the stress vector in the global coordinates, $[D_c]$ is the material stiffness matrix, and $[\epsilon_c^n]$ is the net strain vector. The term $[D_{s_k}][\epsilon_{s_k}]$ is the average stress field acting on the k^{th} reinforcement bar and N is the total number of bars. It should be noted that the average stress in the rebars is considered to be smeared throughout the element.

When cracking occurs, the cracked surface is estimated to be aligned perpendicular to the direction of the principal tensile stresses f_{c_1} (Figure 5.1b). After cracking, the loss in concrete tensile capacity causes the average stress in the rebars f_{s_k} to increase by $f_{s_{cr_k}}$ at the crack surface to ensure that the equilibrium condition is satisfied using the following relation:

$$f_{c_1} = \sum_{k=1}^{N} \rho_k \left(f_{s_{cr_k}} - f_{s_k}\right) \cos^2 \theta_{s_k} \tag{5.2}$$

where ρ_k is the reinforcement ratio of the k^{th} rebar and θ_{s_k} is the angle between the orientation of the reinforcement and the crack. The increase in the local stress of the steel rebars results in a shear stress v_s at the crack surface, as shown in Figure 5.1b. The shear stress v_s can then be used to estimate the amount of crack slippage in the crack direction (Vecchio, 2000).

The compatibility condition also needs to be satisfied for all the elements in the RC frame. The deformation of each element in the RC frame is a combination of the continuum

straining and the discrete slip along the cracked surface. While the continuum strain is calculated using the constitutive laws of the material, the slip strains are calculated using shear stress v_s along the cracked surface within each cracked element. While the detailed calculation is described in Vecchio (2000), a simple summarization is that the average slip strain can be calculated as the ratio of the slip along the cracked surface to the average crack spacing.

In the masonry infill wall (Figure 5.1c) the external force is resisted by the masonry assemblage consisting of Concrete Masonry Units (CMUs) and mortar joints. Similar to concrete, each masonry element subjected to a principal stress field cracks when the tensile stress reaches its limit. At the cracked surface, the local shear ($v_{m_{bj}}, v_{m_{hj}}$) and normal stresses ($f_{m_{bj}}, f_{m_{hj}}$) acting along the mortar head and bed joints, as shown in Figure 5.1c, can be derived based on the equilibrium of internal and external forces. The local shear stresses may cause a rigid body slip in the direction of the mortar head and bed joints. In this study, only the shear slip along mortar bed joints was considered. This assumption was supported by the experimental observation on masonry infilled RC frames under lateral loading (Rahimi and Liu, 2018) where shear sliding was observed to be only pronounced in mortar bed joints.

In the frame-infill mortar interface region (Figure 5.1d), the interface element, acting as the contact between the frame and infill, transfers forces applied from the frame to the infill wall. Such a contact element has not been considered in DSFM shear wall applications in previous studies and its modelling using the DSFM technique needs special consideration. This study proposes the following method for modelling contact elements using the DSFM

technique. As shown in Figure 5.2, the contact elements crack due to the tensile stresses and the crack propagates under both tensile and shear stresses with an average crack spacing s_j (Figure 5.2b) and crack width w_j (Figure 5.2c). The shear stress in the interface elements causes a discrete slip δ_j^s. The average slip strain γ_j^s in the interface element is then calculated by the following equation:

$$\gamma_j^s = \frac{\delta_j^s}{s_j} \tag{5.3}$$

As a result, the orthogonal components of the slip strain (Figure 5.2d), with reference to global coordinates, are calculated using Mohr's circle (Figure 5.2e) as follows:

$$\left[\epsilon_j^s\right] = \begin{Bmatrix} -\frac{\gamma_j^s}{2}\sin 2\alpha \\ \frac{\gamma_j^s}{2}\sin 2\alpha \\ \gamma_j^s \cos 2\alpha \end{Bmatrix} \tag{5.4}$$

where α is the angle between the direction of the thickness and the global x direction. For the modelling purposes of this paper the angle α is either 0 or 90 degrees for either the infill-column or infill-beam interactions, respectively.

One additional point to note is that the presence interfacial gap between the masonry infill and the bounding frame can be a common occurrence in infilled frame construction. The effect of the presence of such gap can be considered by including an elastic strain offset in combination with net and shear slip strains in the compatibility equations.

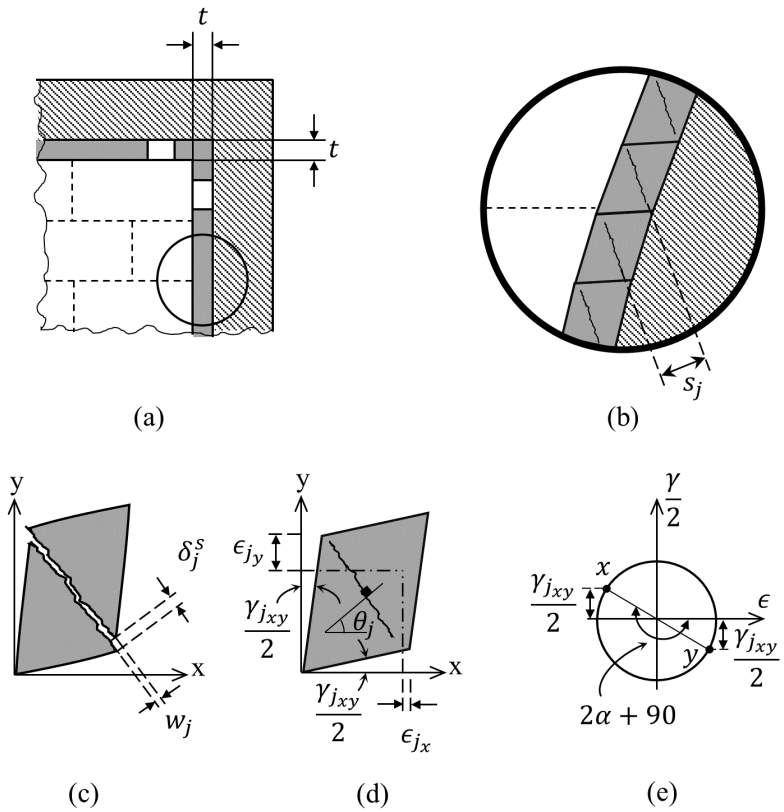


Figure 5.2. Rigid body movement in the mortar interface contact element

5.3.2 Constitutive models

Concrete

The compressive behaviour of the concrete used in this study was based on the model suggested by Vecchio and Collins (1986) and further refined by Vecchio and Collins (1993). The latter study showed that transverse cracking causes a softening effect on the compression capacity of the concrete which can be captured using a factor, β_{c_d}, expressed as a function of principal strains (ϵ_{c_1}, ϵ_{c_2}) in a crack as follows:

$$\beta_{c_d} = \frac{1}{1 + 0.193\left(-\frac{\epsilon_{c1}}{\epsilon_{c_2}} - 0.28\right)^{0.8}} \leq 1.0 \tag{5.5}$$

The β_{c_d} factor is used to modify both the peak stress f_c' and the corresponding strain ϵ_{c_0} in the compressive behaviour obtained from a standard concrete cylinder test. Adopting β_{c_d}, the compressive behaviour of the cracked concrete in terms of principal stress-strain relationship is shown in Figure 5.3a as a solid line, which can be expressed as follows:

$$f_{c_2} = f_{c_p} \frac{n \frac{\epsilon_{c_2}}{\epsilon_{c_p}}}{n - 1 + \left(\frac{\epsilon_{c_2}}{\epsilon_{c_p}}\right)^{nk}} \tag{5.6}$$

where

$$n = 0.80 - \frac{f_{c_p}}{17} \tag{5.7}$$

and

$$k = \begin{cases} 1.0, & \epsilon_{c_p} < \epsilon_{c_2} < 0 \\ 0.67 - f_{c_p}/62, & \epsilon_{c_2} < \epsilon_{c_p} \end{cases} \tag{5.8}$$

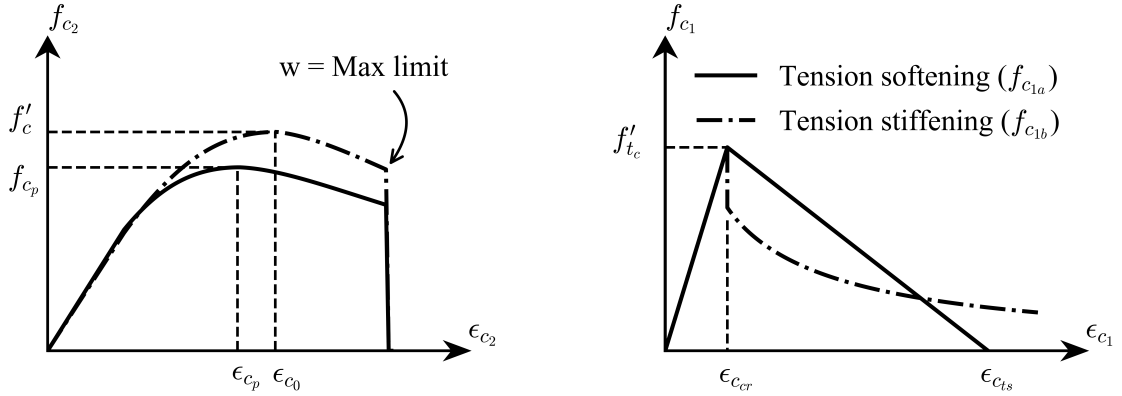


Figure 5.3. Stress-strain relationship of a) concrete in compression and b) concrete in tension

In Figure 5.3a, *w* corresponds to the point where the crack width exceeds a limit beyond which the compressive capacity of the concrete drops to zero. In this study the maximum

crack width was assumed to be 5 mm, as suggested by Vecchio (2000).

The response of the concrete in tension is depicted in Figure 5.3b. In tension, the concrete behaves linearly elastically up to the cracking stress f'_{t_c}, which is the tensile strength of the concrete. The tensile strength of the concrete can be estimated to be a function of its compressive strength using an empirical relation suggested by Vecchio (2000).

$$f'_{t_c} = 0.65 \left(f'_c\right)^{0.33} \tag{5.9}$$

After cracking, the concrete can follow two independent paths, i.e., tension softening or tension stiffening. While tension softening refers to the decrease in concrete tensile stress with increase in crack width, tension stiffening refers to the contribution of the bond between the reinforcement and the concrete in transferring the tensile stress at the crack to uncracked segments of the concrete. The tension softening was assumed to be linear in this study as several finite element models (Vecchio, 2000; Facconi et al., 2014) revealed that linear formulation is sufficiently accurate. Tension softening can occur in elements with or without reinforcements. However, tension stiffening is due to interactions between the reinforcement bars and the concrete and thus can only occur in elements containing reinforcement. The tensile response of the concrete after cracking is governed by the greater of the stress determined by either the tension softening f_{c1_a} or tension stiffening f_{c1_b}.

The path for tensioning softening is dependent on the ultimate strain $\epsilon_{c_{ts}}$. In this study, the ultimate strain was calculated as follows based on the fracture energy concept.

$$\epsilon_{c_{ts}} = 2.0 \frac{G_{f_c}}{f'_{t_c} L_{r_c}} \tag{5.10}$$

where G_{f_c} is the fracture energy parameter and L_{r_c} is the characteristic length (Darwin and Pecknold, 1977). The fracture energy was considered to be 75 N/m for concrete in this study.

The path for tension stiffening was determined using the following:

$$f_{c1b} = \frac{f'_{t_c}}{1 + \sqrt{c_{t_c}\,\epsilon_{c_1}}} \tag{5.11}$$

where c_{t_c} is the degree of stiffening which is dependent on the reinforcement ratio ρ, and rebar diameter d_b (Bentz, 1999).

Steel

The stress-strain model developed by Menegotto and Pinto (1973) and modified by Filippou et al. (1983) was used to model the behaviour of steel reinforcement bars. Both strain hardening and bond-slip effects were considered. The bond-slip effect refers to a phenomenon where a steel bar, when embedded in concrete, does not show a pronounced yield plateau and the "apparent yield stress" is lower than the yield stress of the bare steel bar. This phenomenon was mimicked by reducing both the elastic modulus and yield stress of the steel bars. Figure 5.4 shows the stress-strain constitutive model used to predict the behaviour of steel rebars, where the dotted line incorporates both bond-slip and strain hardening effects, which can be expressed as follows:

$$f_s = b\,\epsilon_s\,\beta_s\,E_s + \frac{(1-b)\,\epsilon_s\,\beta_s\,E_s}{\left[1 + \left(\frac{\epsilon_s}{\epsilon_{s_y}}\right)^{R_s}\right]^{\frac{1}{R_s}}} \tag{5.12}$$

where E_s is the elastic modulus, F_y and ϵ_{s_y} in Equation 5.12 and Figure 5.4 denote the yielding stress and strain of the steel, respectively. The reduction factor β_s, used to account for the bond-slip effect, was assumed to have a value of 0.6 based on a study conducted by Dehestani and Mousavi (2015). The strain hardening ratio b was considered to be 0.01 based on the experimental coupon test. R_s is a parameter that effects the shape of the transition curve and a value of 18 was used, as suggested by Filippou et al. (1983).

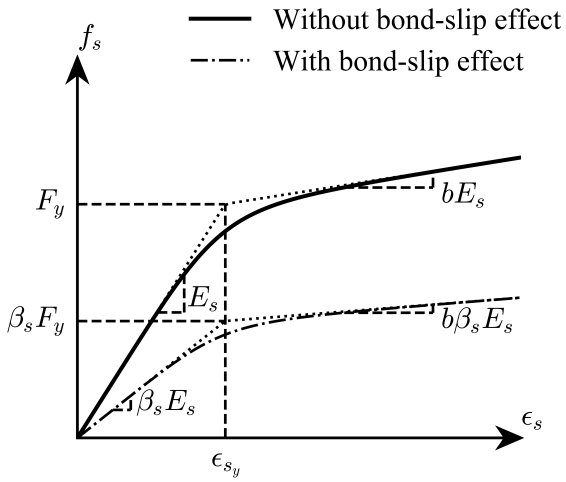


Figure 5.4. Stress-strain relationship of the steel

Masonry

Figure 5.5 shows the constitutive models used for the masonry in both compression and tension, expressed as principal stress-strain relationships. For masonry in compression, similar to the concrete model, the principal stress in compression f_{m_2} is calculated based on the principal compressive strain ϵ_{m_2} using the following relation (Facconi et al., 2014):

$$f_{m_2} = \epsilon_{m_2} E_m \left| 1 - \frac{1}{n} \left(\frac{\epsilon_{m_2}}{\epsilon_{m_p}} \right)^{n-1} \right| \tag{5.13}$$

where n is

$$n = \frac{E_m}{E_m - \frac{f_{mp}}{|\epsilon_{mp}|}} \tag{5.14}$$

and f_{m_p} and ϵ_{m_p} are the peak stress and strain in the compressive constitutive model for the masonry, respectively. They can be determined through the compressive strength f'_m, and the corresponding strain from a uniaxial compression test on sample masonry prisms considering compression softening effect and the orthotropic nature of masonry.

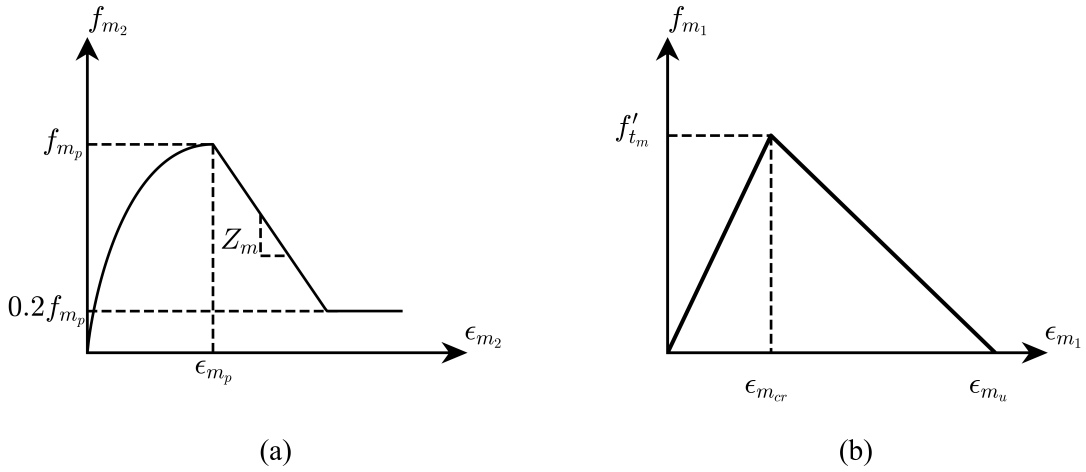


Figure 5.5. Stress-strain relationship of masonry a) in compression b) in tension

The compression softening effect due to tensile strains in the transverse direction (Lotfi and Shing, 1991) can be considered in a similar manner as in concrete and Equation 5.5 still applies. The effect of biaxial stress field in concrete masonry units of orthotropic nature on the calculation of the principal stresses was considered using a factor β_m. The concept of β_m was proposed by Facconi et al. (2014), and the formulation was developed in this study based on the modified Mohr Coulomb's yield criteria. As suggested by the Canadian masonry design standard S304-14 (2014), the compressive strength of masonry in the direction parallel to the bed joints, f_{m_x}, is approximately 50% of the strength in

the direction perpendicular to the bed joints, f_{m_y}. Using this assumption, the maximum compressive strength in the cracking direction was then derived as follows and the associated failure surface is plotted in Figure 5.6.

$$f_{m_{2max}} = \frac{1}{2}\left(f_{m_y} + f_{m_x}\right) + \frac{1}{2}\left(f_{m_y} - f_{m_x}\right)\cos 2\theta_{bj} \tag{5.15}$$

where θ_{bj} is the angle between the direction of the principal stresses and the bed joint, and $f_{m_{2max}}$ is the compressive capacity of the masonry in the direction of the principal stresses.

The β_m factor is then calculated to be:

$$\beta_m = \frac{f_{m_{2max}}}{f_{m_y}} \tag{5.16}$$

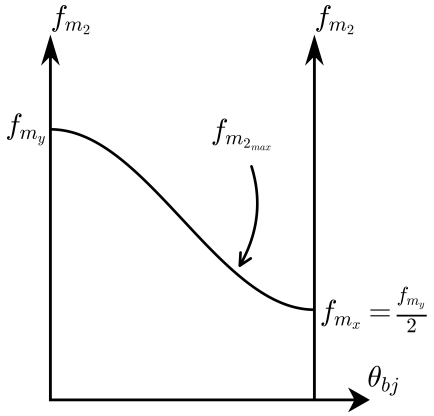


Figure 5.6. Modified Mohr-Coulomb yield criteria

The falling branch of the stress-strain curve would follow a linear descending path until 20% of the peak stress, f_{m_p}, is reached. The slope of the descending branch is calculated as

follows (Priestley and Elder, 1983):

$$Z_m = \frac{1}{\left(\frac{3 + 0.29\, f_j}{145\, f_j - 1000} - 0.0002\right)} \tag{5.17}$$

where f_j is the mortar compressive strength.

For masonry in tension (Figure 5.5b), a linearly elastic behaviour was considered before cracking, where f_{m_1} and ϵ_{m_1} are the masonry principal tensile stress and strain, respectively. Similar to concrete, the effect of the compressive principal stress f_{m_2} on tensile strength can be considered negligible, according to Facconi et al. (2014). The tension softening portion of the masonry was also considered to be linear, following a path based on the fracture energy G_{f_m}. The ultimate strain is calculated as follows:

$$\epsilon_{m_u} = 5.136 \frac{G_{f_m}}{f'_{t_m} L_{r_m}} \tag{5.18}$$

where L_{r_m} is the characteristic length and f'_{t_m} is the tensile capacity of the masonry prisms. The value of the fracture energy, G_{f_m}, was considered to be 40 N/m for the masonry infilled frames, as suggested by Nasiri and Liu (2017).

Mortar interface

Figure 5.7 plots the constitutive models used for the mortar interface between the frame and the infill for both compression and tension. The mortar interface in compression follows a

parabolic path until failure as follows (Vecchio and Collins, 1986):

$$f_{j_2} = f_{j_2 max}\left[2\left(\frac{\epsilon_{j_2}}{\epsilon_{j_0}}\right) - \left(\frac{\epsilon_{j_2}}{\epsilon_{j_0}}\right)^2\right] \tag{5.19}$$

where $f_{j_2 max}$ and ϵ_{j_0} are the stress and strain at the compressive capacity of the mortar interface, which can be estimated using mortar cube samples tested under uniaxial compression with the compression softening effect included. The compression softening due to cracking in the principal tensile direction was again considered using the same reduction factor as defined in Equation 5.5. The ultimate failure was assumed to occur when the crack width, w, reached 4 mm.

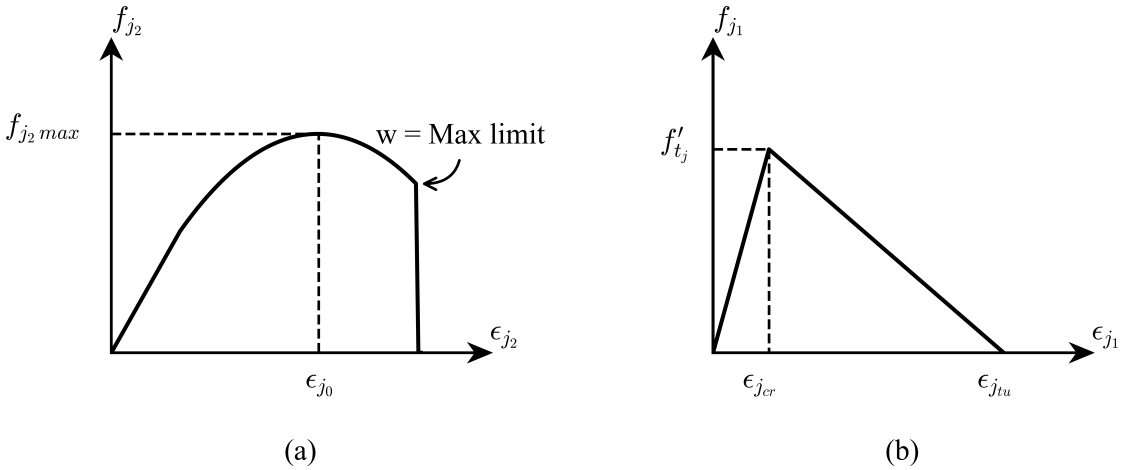


Figure 5.7. Stress-strain relationship for the mortar interface in a) compression and b) tension

In tension, the stress-strain relationship was considered to be linearly elastic up to the cracking stress (Figure 5.7b). After cracking the falling branch of the response was assumed to be linear until the ultimate strain $\epsilon_{j_{tu}}$. Equation 5.18 was also adopted to calculate the ultimate strain.

A bi-linear shear stress-strain relationship of the mortar interface was assumed as shown

in Figure 5.8. To determine the maximum shear stress $v_{j_{max}}$ of the mortar interface element, the Drucker-Prager yield criteria was employed as depicted in Figure 5.8b and also expressed as follows.

$$v_{j_{max}} = \sqrt{\left(f_{j_n}\mu + c_j\right)^2 - \left(f'_{j_t}\mu - c_j\right)^2} \tag{5.20}$$

where $\mu = \tan\beta_j$ is the slope of the hyperbola (Figure 5.8b), β_j is the friction angle, c_j is the cohesion, f_{j_n} is the normal stress at the surface of the mortar interface element, and f'_{j_t} is the tensile capacity of the mortar. Once v_j exceeds $v_{j_{max}}$, the Mohr-Coulomb friction rule governs the shear behaviour and the shear slip over the element width t was calculated based on the following:

$$\delta_j^s = \frac{\mu f_{j_n}}{G_j} t \tag{5.21}$$

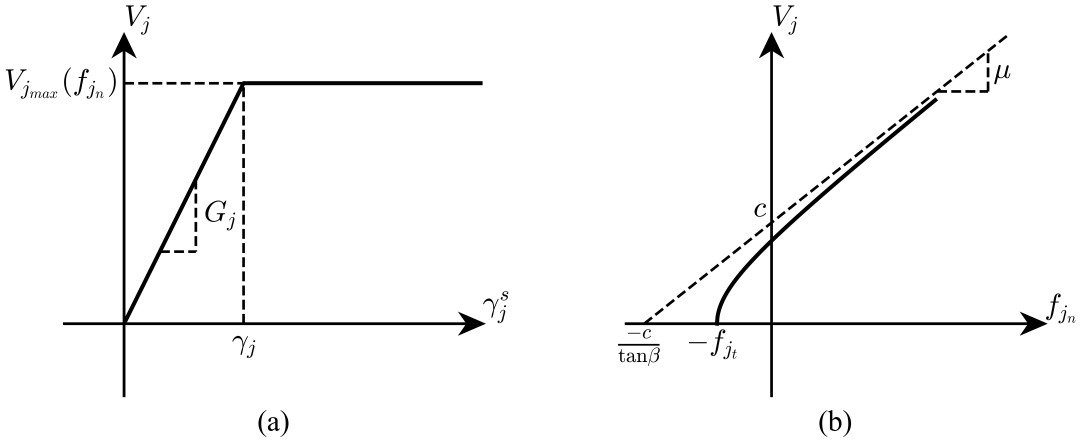


Figure 5.8. Drucker-Prager yield criterion

5.4 Finite element implementation

In a previous paper (Rahimi et al., 2020), the algorithms of the VPFEM library (`github.com/vpfem`) were developed to focus on key steps of the finite element analysis, i.e. stiffness matrix calculation, assembly, and solving the linear system of equations that are common to all structural problems. This study further develops the VPFEM library algorithms to incorporate the DSFM technique for modelling the in-plane behaviour of masonry infilled frames. Figure 5.9 shows the flow chart which the library follows in the nonlinear analysis of masonry infilled frame systems. As mentioned earlier, the uniqueness of the DSFM is that, at each load increment, the material stiffness matrix of each element is checked, based on principal stresses and strains, and modified progressively to reflect the change in crack orientation. The process needs to be carried out for each element at each iteration of each load increment, which leads to very high computational costs. To achieve computational efficiency of the DSFM, the algorithms the steps 2 through 8 (Figure 5.9) are the focus of this study, where parallelization is achieved using the framework developed in the VPFEM library. This is, to the authors' best knowledge, a first effort to parallelize the DSFM technique.

Figure 5.10 shows the discretization of a representative portion of the infilled frame. In this study, bilinear four-node quadrilateral elements were used to mesh all three components, i.e. the RC frame, the masonry infill wall, and the mortar interface between the two. All the elements were considered to be elastic, isotropic, and in a plane stress state. The VPFEM library mesh tool enables the discretization so that the elements of each component at the boundary are aligned as shown in Figure 5.10b. A sensitivity analysis was conducted at the

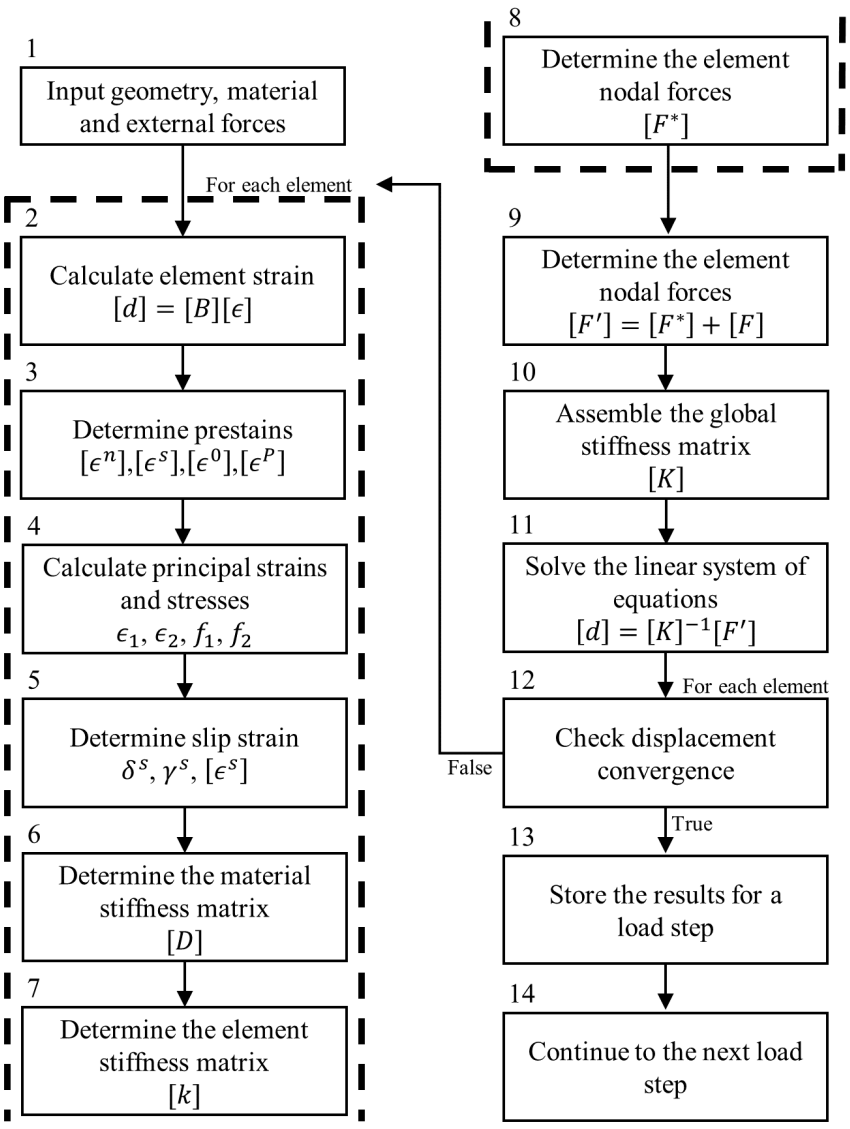


Figure 5.9. Flowchart for nonlinear analysis of masonry infilled frames

outset of modeling and it was found that on average 11 x 11 mm for the RC frame, 10 x 10 mm for the infill wall and 3 x 3 mm for the mortar interface were the optimal mesh sizes in order to converge to an acceptable level of accuracy.

This research employed a load controlled iterative secant-stiffness approach. At each load increment the material stiffness matrix $[D]$, and consequently the element stiffness

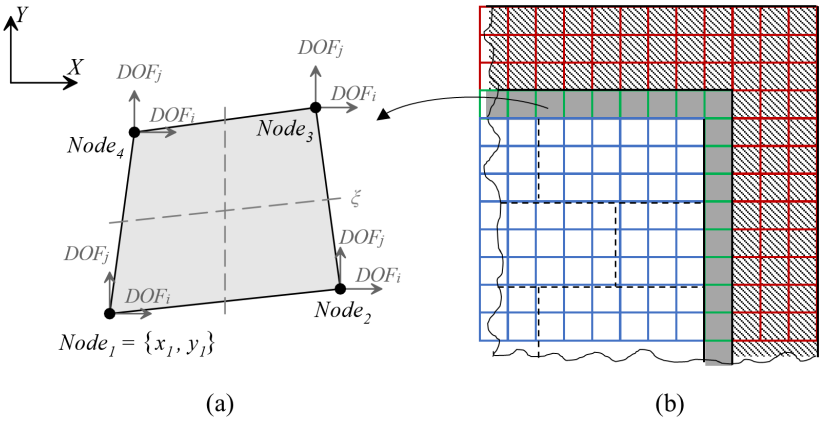


Figure 5.10. Mesh discretization of the infilled frame

matrix $[K]$, were continuously modified until an acceptable level of convergence in displacement was achieved (see step 12 in Figure 5.9). Convergence was considered to be achieved when the relative difference in displacements was less than 10^{-5}. The iterative process involved steps 2 to 8 in Figure 5.9. As the operations in steps 2 to 8 is been performed for each element with specific input data stored for that element and thus independent from other elements, this process can be parallelized using GPU architectures. A kernel function was designed to schedule the task between the available threads (Algorithm 5.1). Since the data is separate for each element while the instructions for updating the element stiffness matrix is the same, the process was parallelized through the Single Instruction Multiple Data (SMID) class of parallel computing.

The detailed step algorithms for steps 2 to 8 can be found in Appendix A. To summarize, the kernel inputs the structured data including the element information, the global displacement and element nodal forces and outputs a coordinate format (COO) sparse matrix of stacked stiffness matrices and a vector of updated nodal forces. The VPFEM library takes the output of the kernel function and assembles the stack of local stiffness matrices into the

Algorithm 5.1. Parallel algorithm for updating the element stiffness matrix and nodal forces

Data: d_{global}: global displacement, $ElementDetails$: element information
Result: k: stack of stiffness matrices, F^*: updated nodal forces
Kernel *UpdateStiffnessMatrix(d_{global}, $ElementDetails$, ϵ)*

```
/* Divide the operations between available threads                    */
ElementNumber ← ThreadId.x
/* SIMD instructions                                                  */
```

$\epsilon \leftarrow$ ElementDisplacementToStrain($ElementNumber$,d_{global}) // step 2

$\epsilon^n \leftarrow$ ElementStrainToNetStrain(ϵ) // step 3

$f \leftarrow$ ElementStrainToStress(ϵ^n) // step 4

$\epsilon^s \leftarrow$ ElementShearStrain(γ^s) // step 5

$D \leftarrow$ ElementMaterialMatrix(f) // step 6

$K \leftarrow$ stiffnessMatrixCalculation($ElementDetails$) // step 7 (Rahimi et al., 2020)

$F^* \leftarrow$ ElementNodalForces(ϵ^s) // step 8

return k F^*

global stiffness matrix in a Compressed Sparse Row (CSR) format, and solves the system of equations using the updated nodal force vector (steps 10 and 11 in Figure 5.9). After the system of equations is solved, the convergence of the displacement is checked by comparing relatively to the displacement achieved in the previous iteration. If the desired level of convergence was achieved, the current state is recorded, and the application moves to the next load increment.

5.5 Performance of the VPFEM library

The efficacy of the VPFEM library in achieving the objectives of this study is demonstrated through 1) its performance in predicting the behaviour of masonry infilled RC frames in comparison with test results; and 2) its performance in accelerating the model run speed

when implemented on GPUs.

First, the finite element model was validated against experimental results of an infilled frame specimen tested by Rahimi and Liu (2018). The geometry and reinforcement details of the infilled frame specimen are presented in Figure 5.11. The masonry infill was constructed using custom-made, half-scale 200 mm standard concrete masonry units laid in a running bond. The RC frame was designed according to CSA A23.3 (2004b) and reinforcement detailing, including size, spacing, arrangement of longitudinal bars and stirrups, complied with code requirements to provide ductility and avoid brittle shear failure.

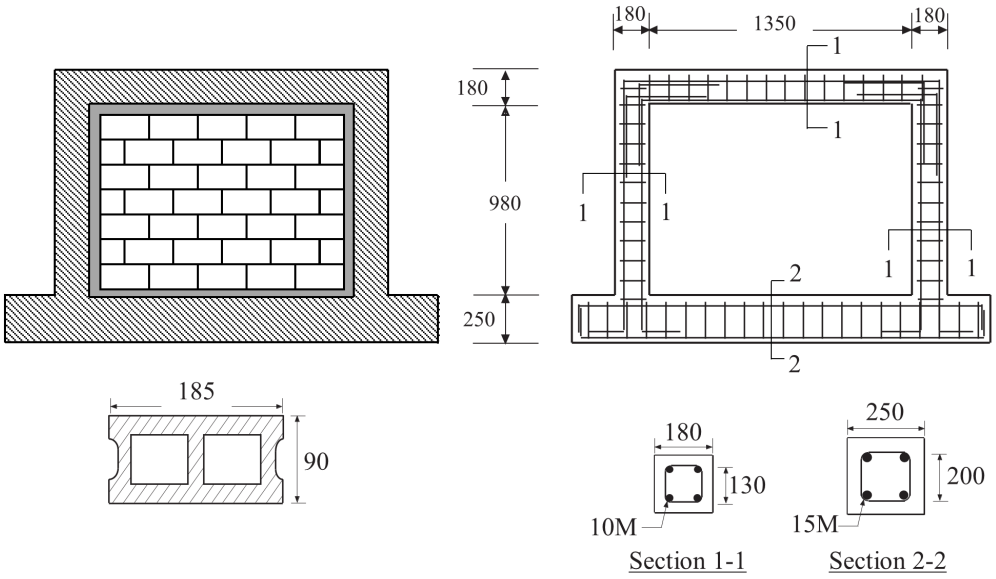


Figure 5.11. Geometry and reinforcement details of the specimen

The experimental set-up is illustrated in Figure 5.12. The specimen was connected to the strong floor through high strength bolts and the lateral load was applied at the top beam level using a hydraulic actuator with a capacity of 250 kN. Two Linear Variable Displacement Transducers, LVDTs, (LVDT 1 and 2) were mounted at the centerline of the top and bottom beam respectively to measure the in-plane lateral displacements. Another two LVDTs (not

shown) were positioned at the mid-height of the masonry infill wall and at the central point of the top beam respectively, both on the back side, to monitor any possible out-of-plane movements of the infill wall and the frame, respectively.

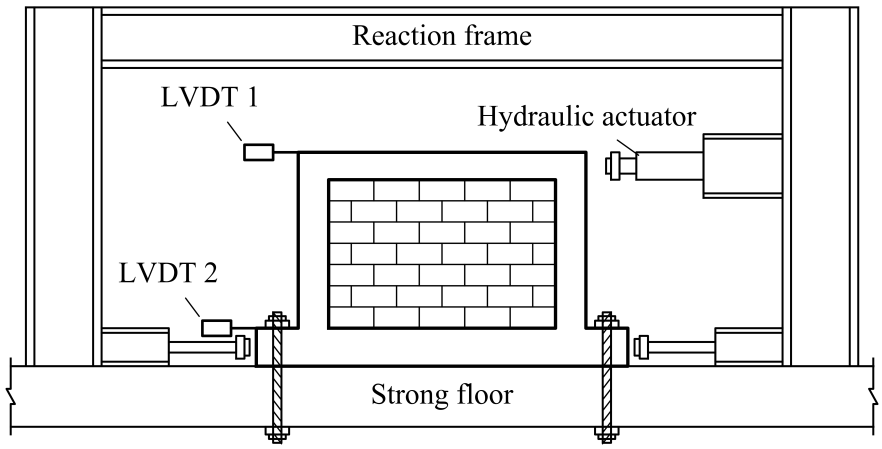


Figure 5.12. Schematic view of test set-up

The mechanical properties of Concrete Masonry Units (CMUs), mortar, and masonry prisms for the infill and those of the concrete and reinforcement for the frame were obtained experimentally in accordance with ASTM specifications. A summary of the material properties is presented in Table 5.1.

Table 5.1. Material properties of the specimens

	Elastic modulus E (MPa)	Compressive strength (MPa)	Tensile strength (MPa)	Yield strength (MPa)
Concrete	12710	29.2	1.7	-
CMUs	3500	25.0	1.6	-
Mortar	2600	22.1	1.3	-
Masonry prisms	2980	10.5	1.04	-
Reinforcement	247357	-	665	446

Figure 5.13 compares the load vs. lateral displacement response obtained using the VPFEM application with the experimental results. It can be seen that the VPFEM predicted

an ultimate load of 98.9 kN, which is within 2% of the experimental value of 100.7 kN. Overall, the VPFEM curve compares very well with the experimental response. The slightly stiffer behaviour in the rising branch of the finite element curve is believed to be attributed to the use of first order elements. It is also noted that the VPFEM model showed marked accuracy in predicting the post-ultimate behaviour.

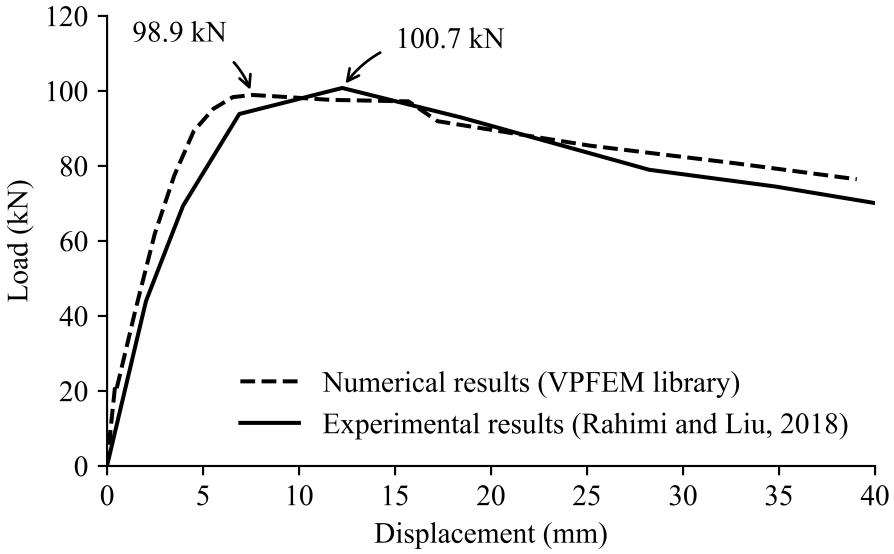


Figure 5.13. Comparison of load vs. displacement behaviour of the masonry infilled frame specimen

Figure 5.14 shows the cracking pattern observed at the ultimate load in the experimental program and the finite element simulations, respectively. The experimental cracking was predominately in the diagonal direction and corner crushing was the mode of failure. In Figure 5.14b the shaded elements are those with yielded reinforcements. The lines in the elements indicate cracking and the direction of the line is the crack direction. The elements with darker solid lines are the ones that have reached the maximum crack width limit (failed). The cracking pattern resulted from the finite element simulations shows pronounced diagonal cracking and compression failure at the corners as well as frame

cracking, all of which are in agreement with the test observations.

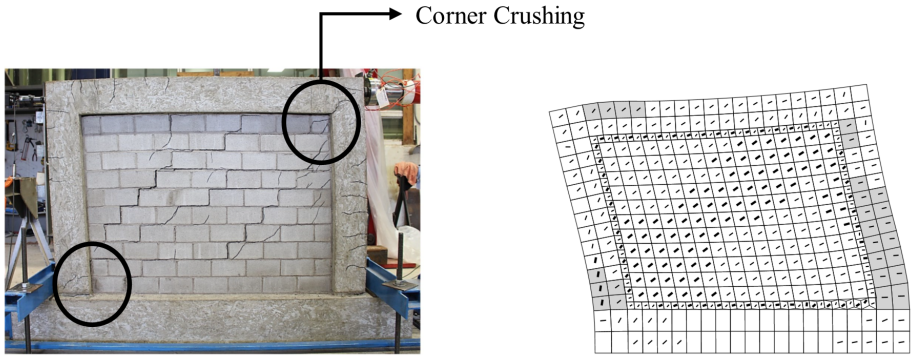


Figure 5.14. Comparison of cracking pattern a) experimental observations b) numerical simulations

The runtime of the VPFEM analysis using different GPU devices was compared to a single CPU device for modeling the infilled frame. One comparison is presented in Figure 5.15 where the runtimes associated with the analysis of one load increment in the DSFM technique are compared. The CPU device in Figure 5.15 refers to a single core in the Intel Xeon CPU E5-2630 v4. The GPU devices are consumer grade GPU devices with Compute Unified Device Architecture (CUDA) capabilities, ranging from low-end (GT 1050) to high-end (GTX TITAN Xp). These GPU devices are proved to be cost efficient for finite element simulations (Rahimi et al., 2020). All the GPU devices were hosted in the same workstation. The details of the workstation are provided in Rahimi et al. (2020).

The runtimes presented in Figure 5.15 were the finite element results based on 5144 elements for the infilled specimen. It shows that the GPU device (TITAN Xp) could run up to 6 times faster than a CPU. However, it is noted that the GPU devices, GT 1030 and GTX 1050, were not able to speed-up the process due to their limited memory capacity and speed. The low memory capacity in a GPU device results in the scheduler to cut the global

data into smaller pieces and copy the data when seemed necessary. The computational costs associated with transferring data results in a runtime penalty for the mentioned devices. The model discussed in this paper required twice the memory capacity available in the GPU devices GT 1030 and GTX 1050. The GPU devices GTX 1060, GTX 1070 and TITAN xp, on the other hand, were able to accelerate the process due to the higher capacity and speed in their memory modules. Also, to note that the comparison was conducted on the model with 5144 elements. If the number of elements were to increase significantly for modeling a more sophisticated or complex structure, the acceleration as a result of using parallel algorithm could be more pronounced. For example, for a two-bay, two-storey infilled frame system with 20160 elements, the run speed in GPU device TITAN Xp would be more than 18 times faster than the CPU device.

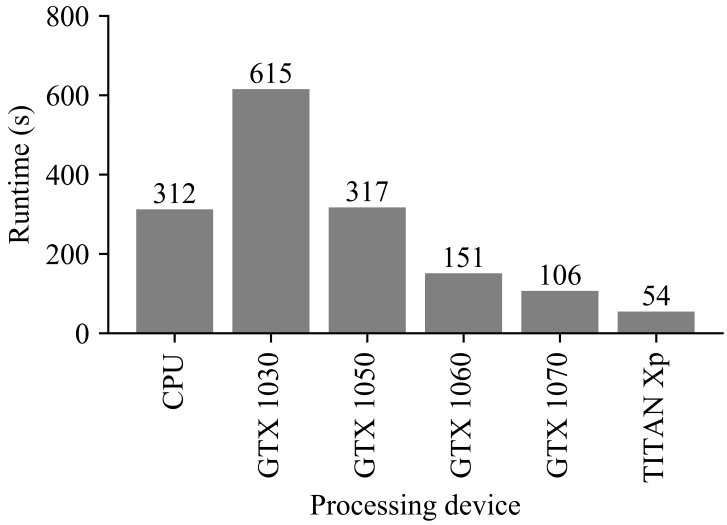


Figure 5.15. Runtime vs. the processing device employed in the VPFEM library for simulating the in-plane behaviour of the masonry infilled frame

5.6 Conclusions

This paper presents the implementation of the Distributed Stress Field Method (DSFM) in a self-developed finite element library VPFEM for modelling the in-plane behaviour of masonry infilled RC frames. Extending application of the DSFM to the analysis of masonry infilled frames, the paper further developed the mortar interface contact element between the infill wall and the bounding frame with a shear slip model capable of capturing the rigid body movement at the cracked surface. When implementing the DSFM in the VPFEM library, the Single Instruction Multiple Data (SMID) class of parallel computing technique was used to parallelize the iterative analysis process at each element level to achieve acceleration. The good agreement of the ultimate load as well as the load vs. displacement response between the finite element and experimental results suggested that the DSFM was successfully incorporated in the analysis of masonry infilled RC frames. Further, the DSFM was also shown to be capable of predicting with a reasonable accuracy the cracking pattern of the infill. The performance of the VPFEM library in accelerating the model run speed was demonstrated through comparisons of using a CPU and several configurations of GPUs. It was shown that GPU devices can accelerate up to six times the model run speed when compared with a CPU for simulating an infill wall. The degree of speed-up is highly dependent on the number of elements. The greater number of elements, the more significant speed-up can be expected.

5.7 Acknowledgement

The authors wish to recognize the contribution of financial assistance by the Natural Sciences and Engineering Research Council (NSERC) of Canada. Also, the authors gratefully acknowledge the support of NVIDIA Corporation through their donation of the Titan Xp GPU used in this research.

Chapter 6

Summary and conclusions

This research consisted of two distinctive objectives. In phase one, a reliability analysis was conducted which implemented the Random Finite Element Method (RFEM) in estimating the resistance factor for designing the masonry infill walls. The randomness of masonry materials was considered by random field simulation of masonry compressive strength over masonry infills. A total of 1000 random field realizations for each set of statistical parameters of masonry compressive strength was generated. A numerical model developed in OpenSees software was used to produce a histogram of lateral load resistance of masonry infills by numerically analyzing each random field realization. A lognormal distribution was fitted to the histogram of each parameter set which results in a total of 27 simulation-based distributions of lateral load resistance of masonry infills. The simulation-based distributions were used to calibrate the analytically-based formulation for design strength of masonry infills. The calibrated analytical simulations were then used to derive the lateral resistance factor of masonry infills. In phase two, parallelized algorithms for a modular, open source finite element library were developed to accelerate structural finite element simulations, using parallelization capability on CPU and GPU devices. The library was encoded in the C++ programming language, the unique feature of the algorithm is its use of the data-oriented design (DoD) paradigm with the single instruction multiple data (SIMD) class of

parallel computing to accelerate the finite element process i.e. stiffness matrix calculation, assembly of the global stiffness matrix and solving the system of equations. The algorithm also provides the flexibility of running finite element simulations on a single CPU core, multiple CPU cores, or multiple GPU threads. The finite element library also implemented the Distressed Stress Field Method (DSFM) for modelling the in-plane behaviour of masonry infilled RC frames. Extending application of the DSFM to the analysis of masonry infilled frames, the study further developed the mortar interface contact element between the infill wall and the bounding frame with a shear slip model capable of capturing the rigid body movement at the cracked surface. When implementing the DSFM in the library the SIMD class of parallel computing technique was again used to parallelize the iterative analysis process at each element level to achieve acceleration. The ultimate load as well as the load vs. displacement response between the finite element and experimental results were compared and discussed. The performance of the library in accelerating numerical simulations of masonry infilled RC frames was examined. The conclusions drawn from this study are listed in the following.

6.1 Reliability analysis

- The two-dimensional finite element model developed in OpenSees was shown to produce a good agreement of the ultimate load and stiffness as well as the load vs. displacement response against the experimental results under both static and cyclic loading.

- The macro-modeling technique was selected to be a more suitable tool when compared

with the micro-modeling technique in reliability analyses of infilled frames with potentially thousands of numerical-model runs.

- Random field simulation of masonry compressive strength using lognormal distributions was shown to be a good choice when fitting distributions to the histograms of the lateral load resistance of masonry infills.

- Geometric averaging of the masonry compressive strength over the areal of the diagonal strut was essential in accurately calibrating the analytical-based distributions of the lateral load resistance.

- The analytical formulations for estimating the lateral resistance distribution of masonry infills were successfully calibrated using the simulation-based distributions.

- The resistance factor of 0.8 for the target reliability index of 3 was achieved for strength design of masonry infills.

6.2 Finite element library

- The data-oriented design (DoD) paradigm along with the single instruction multiple data (SIMD) class of parallel computing allowed running finite element analysis on devices with massive parallel architecture.

- The finite element library developed in this study, VPFEM, can accelerate the finite element process up to 122 times using a consumer level GPU device, GTX TITAN Xp.

- The VPFEM library is available in public domain.
- The VPFEM library is encoded using C++ programming language.
- The Distressed Stress Field Method (DSFM) used to model the in-plane behaviour of infilled frames for the first time in the developed library showed good agreement between the numerical and experimental results.
- The developed finite element for modeling the in-plane behaviour of infilled frames using the VPFEM library was shown to accelerate the finite element runtime significantly.
- The degree of acceleration in the VPFEM library is highly dependent to the number of elements in the finite element model and the number of cores in the parallel device. The greater number of elements and the number of cores, the more significant speed-up can be expected.

6.3 Recommendations for future work

The following is recommended for future research:

- All the experimental studies used to calibrate the finite element models were on specimens made with half-scaled CMUs and one set of material properties. Experimental investigations on full-scaled specimens as well as varying material properties for both masonry and concrete would provide valuable information on the behaviour masonry infilled RC frames.

- Reliability analysis used in this study was based on a random field simulation of the masonry compressive strength and the focus was on the masonry infills. Parameters such as such as masonry unit geometry might be of interest. Further, studying the reliability of infilled frames by introducing the randomness of the frame member properties, such as concrete compressive strength, would afford a reliability analysis from a system point of view.

- In this study the lateral load was considered to be deterministic seismic load. Considering both the randomness and type of the lateral loads would result in a more reliable resistance factor.

- This research focused on developing a finite element library that can be run on multiple processor configurations, especially GPU devices. A study on the benefits of clustering GPU devices, also in combination with CPUs to form a network for large finite element simulations can be a topic of future work.

Bibliography

Abhyankar, S., Brown, J., Constantinescu, E. M., Ghosh, D., Smith, B. F., and Zhang, H. 2018. Petsc/ts: A modern scalable ode/dae solver library. *arXiv preprint arXiv:1806.01437*.

Alnæs, M., Blechta, J., Hake, J., Johansson, A., Kehlet, B., Logg, A., Richardson, C., Ring, J., Rognes, M. E., and Wells, G. N. 2015. The fenics project version 1.5. *Archive Numerical Software*, **3**(100).

Alzetta, G., Arndt, D., Bangerth, W., Boddu, V., Brands, B., Davydov, D., Gassmoeller, R., Heister, T., Heltai, L., Kormann, K., Kronbichler, M., Maier, M., Pelteret, J.-P., Turcksin, B., and Wells, D. 2018. The `deal.II` library, version 9.0. *Journal of Numerical Mathematics*. doi: 10.1515/jnma-2018-0054.

American Standards Association. Sectional Committee on Building Code Requirements and Good Practice Recommendations for Masonry, A41, 1954. American standard building code requirements for masonry. National Bureau of Standards. National Bureau of Standards Miscellaneous publication and United States. URL `https://www.govinfo.gov`.

Asteris, P., Cotsovos, D., Chrysostomou, C., Mohebkhah, A., and Al-Chaar, G. 2013. Mathematical micromodeling of infilled frames: State of the art. *Journal of Engineering Structures*, 56:1905 – 1921. ISSN 0141-0296. doi: 10.1016/j.engstruct.2013.08.010.

Bathe, K.-J. 2006. *Finite element procedures*. Klaus-Jurgen Bathe. ISBN 097900490X.

Bentz, E., 1999. *Sectional analysis of reinforced concrete structures*. PhD thesis, University of Toronto, Toronto, Canada.

Bhide, S. B. 1987, Reinforced concrete elements in shear and tension. Technical Report Publication 87-02, University of Toronto, Toronto, Canada.

Bolz, J., Farmer, I., Grinspun, E., and Schröoder, P. 2003. Sparse matrix solvers on the GPU: Conjugate gradients and multigrid. *Journal of ACM Trans. Graph.*, **22**(3):917–924. ISSN 0730-0301. doi: 10.1145/882262.882364.

Brodtkorb, A. R., Hagen, T. R., and Sætra, M. L. 2013. Graphics processing unit (GPU) programming strategies and trends in GPU computing. *Journal of Parallel and Distributed Computing*, **73**(1):4 – 13. ISSN 0743-7315. doi: 10.1016/j.jpdc.2012.04.003. Metaheuristics on GPUs.

Bui, H. P., Tomar, S., Courtecuisse, H., Cotin, S., and Bordas, S. P. 2017. Real-time error control for surgical simulation. *Journal of IEEE Transactions on Biomedical Engineering*, **65**(3):596–607.

Bui, H. P., Tomar, S., and Bordas, S. P. 2019. Corotational cut finite element method for real-time surgical simulation: application to needle insertion simulation. *Journal of Computer Methods in Applied Mechanics and Engineering*, 345:183–211.

Butenhof, D. R. 1997. *Programming with POSIX threads*. Addison-Wesley Professional. ISBN 0201633922.

C++ Standards Committee and others. 2017, ISO international standard ISO/IEC 14882: 2017, Programming Language C++. Technical report, C++ Standards Committee and others, Geneva, Switzerland. URL `open-std.org/jtc1/sc22/wg21`.

Cai, Y., Li, G., and Wang, H. 2013. A parallel node-based solution scheme for implicit finite element method using GPU. *Journal of Procedia Engineering*, 61:318 – 324. ISSN 1877-7058. doi: 10.1016/j.proeng.2013.08.022. 25th International Conference on Parallel Computational Fluid Dynamics.

Canadian Standards Association, 1994. CSA S304-94 masonry design for buildings (limit states design). Canadian Standards Association, Mississauga, Ontario, Canada.

Canadian Standards Association, 2004a. CSA S304-04: Design of masonry structures. Canadian Standards Association, Mississauga, Ontario, Canada. URL `http://www.nrc-cnrc.gc.ca`.

Canadian Standards Association, 2004b. CSA A23.3: Design of concrete structures. Canadian Standards Association, Mississauga, Ontario, Canada. URL `http://www.nrc-cnrc.gc.ca`.

Canadian Standards Association, 2011. CSA S408: Guidelines for the development of limit states design. Canadian Standards Association, Mississauga, Ontario, Canada. URL `http://www.nrc-cnrc.gc.ca`.

Canadian Standards Association, 2014. CSA S304-14: Design of masonry structures. Canadian Standards Association, Mississauga, Ontario, Canada. URL `http://www.nrc-cnrc.gc.ca`.

Cecka, C., Lew, A. J., and Darve, E. 2011. Assembly of finite element methods on graphics processors. *International Journal for Numerical Methods in Engineering*, **85** (5):640–669. doi: 10.1002/nme.2989.

Celarec, D., Ricci, P., and DolÅąek, M. 2012. The sensitivity of seismic response parameters to the uncertain modelling variables of masonry-infilled reinforced concrete frames. *Journal of Engineering Structures*, 35:165 – 177. ISSN 0141-0296. doi: https://doi.org/10.1016/j.engstruct.2011.11.007.

Chandrupatla, T. R., Belegundu, A. D., Ramesh, T., and Ray, C. 2002. *Introduction to finite elements in engineering*, volume 10. Prentice Hall Upper Saddle River, NJ.

Chen, X., 2016. *Numerical study of the in-plane behaviour of concrete masonry infills bounded by steel frames*. PhD thesis, Dalhosuie University, Halifax, Canada.

Chen, X. and Liu, Y. 2016. A finite element study of the effect of vertical loading on the in-plane behavior of concrete masonry infills bounded by steel frames. *Journal of Engineering Structures*, 117:118–129.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. 2009. *Introduction to algorithms*. MIT press.

Cornell, C. A. 1969, Structual safety specification based on second-moment reliability analysis. In *Proceedings of Symposium on Concepts of Safety of Structures and Methods of Design*, Zurich, IABSE, London, 1969.

Cui, T., Guo, X., and Liu, H. 2018. GPU accelerated finite element assembly with runtime compilation. *arXiv preprint arXiv:1802.03433*.

Darwin, D. and Pecknold, D. 1977. Nonlinear biaxial stress-strain law for concrete. *Journal of Engineering Mechanics*, 103:229.

Dehestani, M. and Mousavi, S. 2015. Modified steel bar model incorporating bond-slip effects for embedded element method. *Journal of Construction and Building Materials*, 81:284 – 290. ISSN 0950-0618. doi: 0.1016/j.conbuildmat.2015.02.027.

Dvorkin, E. N., Pantuso, D., and Repetto, E. A. 1995. A formulation of the MITC4 shell element for finite strain elasto-plastic analysis. *Journal of Computer methods in applied mechanics and engineering*, **125**(1-4):17–40.

Dymiotis, C., Kappos, A. J., and Chryssanthopoulos, M. K. 2001. Seismic reliability of masonry-infilled RC frames. *Journal of Structural Engineering*, **127**(3):296–305.

El-Dakhakhni, W. W., Elgaaly, M., and Hamid, A. A. 2003. Three-strut model for concrete masonry-infilled steel frames. *Journal of Structural Engineering*, **129**(2):177–185.

Englekirk, R. E. and Hart, G. C. 1984. Earthquake design of concrete masonry buildings, vol. 2: Strength design of one-to-four-story buildings.

Fabian, R., 2013. Data-oriented design. URL `dataorienteddesign.com`.

Facconi, L., Plizzari, G., and Vecchio, F. 2014. Disturbed stress field model for unreinforced masonry. *Journal of Structural Engineering*, **140**(4):04013085–1–11.

Fenton, G. and Griffiths, D. 2008. Risk assessment in geotechnical engineering. *John wiley&Sons, Inc*, pages 381–400.

Fenton, G. A. and Griffiths, D. V. 1993. Statistics of block conductivity through a simple bounded stochastic medium. *Journal of Water Resources Research*, **29**(6):1825–1830.

Fenton, G. A. and Vanmarcke, E. H. 1990. Simulation of random fields via local average subdivision. *Journal of Engineering Mechanics*, **116**(8):1733–1749.

Filippou, F. C., Bertero, V. V., and Popov, E. P. 1983, Effects of bond deterioration on hysteretic behavior of reinforced concrete joints. Technical Report UCB/EERC-83/19, Earthquake Engineering Research Center, Berkeley, California.

Flanagan, R. D. and Bennett, R. M. 1999. In-plane behavior of structural clay tile infilled frames. *Journal of structural engineering*, **125**(6):590–599.

Flanagan, R. D. and Bennett, R. M. 2001. In-plane analysis of masonry infill materials. *Journal of Practice Periodical on Structural Design and Construction*, **6**(4):176–182.

Fu, Z., Lewis, T. J., Kirby, R. M., and Whitaker, R. T. 2014. Architecting the finite element method pipeline for the GPU. *Journal of Computational and Applied Mathematics*, 257: 195 – 211. ISSN 0377-0427. doi: 10.1016/j.cam.2013.09.001.

Georgescu, S., Chow, P., and Okuda, H. 2013. GPU acceleration for fem-based structural analysis. *Archives of Computational Methods in Engineering*, **20**(2):111–121.

Griffiths, D. and Fenton, G. A. 1993. Seepage beneath water retaining structures founded on spatially random soil. *Journal of Géotechnique*, **43**(4):577–587.

Harris, M. 2005, GPGPU: General-purpose computation on GPUs. In *proceedings of SGGRAPH 2005 GPGPU COURSE*, pages 1–51.

Hart, G. C. and Zorapapel, G. T. 1991, Reliability of concrete masonry wall structures. Technical Report 8.2-1, Department of Civil Engineering, University of California, Los Angeles, Los Angeles, California.

Hart, G. C., Huang, S. C., and Sabol, T. A. 1983. Calibration of strength reduction factors for concrete masonry. *Journal of Civil Engineering Systems*, **1**(1):33–36.

Hashemi, S. A., 2007. *Seismic evaluation of reinforced concrete buildings including effects of masonry infill walls*. PhD thesis, University of California Berkeley, Berkeley, California.

Hasofer, A. M. and Lind, N. C. 1974. Exact and invariant second-moment code format. *Journal of the Engineering Mechanics division*, **100**(1):111–121.

Helfenstein, R. and Koko, J. 2012. Parallel preconditioned conjugate gradient algorithm on GPU. *Journal of Computational and Applied Mathematics*, **236**(15):3584 – 3590. ISSN 0377-0427. doi: 10.1016/j.cam.2011.04.025.

Holmes, M. 1961, Steel frames with brickwork and concrete infilling. In *proceedings of the Institution of civil Engineers*, volume 19, pages 473–478. Thomas Telford-ICE Virtual Library.

Hu, C., 2015. Experimental study of the effect of interfacial gaps on the in-plane behaviour of masonry infilled RC frames. Master's thesis, Dalhosuie University, Halifax, Canada.

Hwang, H. H. and Huo, J.-R. 1994. Generation of hazard-consistent fragility curves. *Journal of Soil Dynamics and Earthquake Engineering*, **13**(5):345–354.

Jeon, J.-S., Park, J.-H., and DesRoches, R. 2015. Seismic fragility of lightly reinforced concrete frames with masonry infills. *Journal of Earthquake Engineering & Structural Dynamics*, **44**(11):1783–1803. doi: 10.1002/eqe.2555.

Karsan, I. D. and Jirsa, J. O. 1969. Behavior of concrete under compressive loadings. *Journal of the Structural Division*, **95**(12):2543–2563.

Kirk, D. et al. 2007, NVIDIA CUDA software and GPU parallel computing architecture. In *Proceeding of the International Symposium on Memory Management*, volume 7, pages 103–104.

Klöckner, A., Warburton, T., Bridge, J., and Hesthaven, J. 2009. Nodal discontinuous galerkin methods on graphics processors. *Journal of Computational Physics*, **228**(21): 7863 – 7882. ISSN 0021-9991. doi: 10.1016/j.jcp.2009.06.041.

Knuth, D. E. 1998. *The art of computer programming: Volume 3: Sorting and Searching*, volume 3. Addison-Wesley Professional, 2 edition. ISBN 0-201-89685-0.

Kolev, T. and Dobrev, V., 2010. MFEM: Modular finite element methods library. `mfem.org`.

Komatitsch, D., Michéa, D., and Erlebacher, G. 2009. Porting a high-order finite-element earthquake modeling application to NVIDIA graphics cards using cuda. *Journal of Parallel and Distributed Computing*, **69**(5):451 – 460. ISSN 0743-7315. doi: 10.1016/j.jpdc.2009.01.006.

Laird, D. A., Drysdale, R. G., Stubbs, D. W., and Sturgeon, G. R. 2005, The new csa s304. 1-04 "design of masonry structures". In *Proceedings of the 10th Canadian Masonry Symposium.*, pages 8–12, Banff, Alberta, 2005.

Liauw, T. and Kwan, K. 1985. Unified plastic analysis for infilled frames. *Journal of Structural Engineering*, **111**(7):1427–1448.

Lindholm, E., Nickolls, J., Oberman, S., and Montrym, J. 2008. NVIDIA Tesla: A unified graphics and computing architecture. *IEEE micro*, **28**(2):39–55.

Lotfi, H. R. and Shing, P. B. 1991. An appraisal of smeared crack models for masonry shear wall analysis. *Journal of Computers & structures*, **41**(3):413–425.

Lotfi, H. and Shing, P. 1994. Interface model applied to fracture of masonry structures. *Journal of structural engineering*, **120**(1):63–80.

Lu, X., Lu, X., Guan, H., and Ye, L. 2013. Collapse simulation of reinforced concrete high-rise building induced by extreme earthquakes. *Journal of Earthquake Engineering & Structural Dynamics*, **42**(5):705–723.

Madan, A., Reinhorn, A., Mander, J., and Valles, R. 1997. Modeling of masonry infill panels for structural analysis. *Journal of structural engineering*, **123**(10):1295–1302.

Mainstone, R. 1971, On the stiffness and strengths of infilled frame. In *Proceedings Institution of Civil Engineers, Supplement IV*, pages 57–90.

Mander, J., Priestley, M., and Park, R. 1988. Observed stress-strain behavior of confined concrete. *Journal of structural engineering*, **114**(8):1827–1849.

Margetts, L., 2002. *Parallel finite element analysis*. PhD thesis, University of Manchester, Manchester, UK.

Martínez-Frutos, J. and Herrero-Pérez, D. 2015. Efficient matrix-free GPU implementation of fixed grid finite element analysis. *Journal of Finite Elements in Analysis and Design*, 104:61–71.

Masonry Standards Joint Committee, 2005. Building code requirements and specification for masonry structures (ACI 530/ASCE 5/TMS 402 and ACI 530.1/ASCE 6/TMS 602).

Mckenna, F. T., 1999. *Object-oriented finite element programming: Frameworks for analysis, algorithms and parallel computing*. PhD thesis, University of California, Berkeley, Berkeley, California.

McKenna, F. 2011. OpenSees: a framework for earthquake engineering simulation. *Journal of Computing in Science & Engineering*, **13**(4):58–66.

McKenna, F., Scott, M. H., and Fenves, G. L. 2010. Nonlinear finite-element analysis software architecture using object composition. *Journal of Computing in Civil Engineering*, **24**(1):95–107.

Mehrabi, A. B., Benson Shing, P., Schuller, M. P., and Noland, J. L. 1996. Experimental evaluation of masonry-infilled RC frames. *Journal of Structural engineering*, **122**(3): 228–237.

Menegotto, M. and Pinto, P. 1973. Method of analysis for cyclically loaded reinforced concrete plane frames including changes in geometry and nonelastic behavior of elements under combined normal force and bending. *IABSE. Symposium on resistance and ultimate deformability of structures acted on by well-defined repeated loads. Lisbon*, 15:22.

Minaie, E., Mota, M., Moon, F., and Hamid, A. 2010. In-plane behavior of partially grouted reinforced concrete masonry shear walls. *Journal of Structural Engineering*, **136** (9):1089–1097.

Moghaddam, H. and Dowling, P. 1987, The state of the art in infilled frames. Technical Report ESEE Research Report No. 87-2, Imperial College of Science and Technology, London, UK.

Mohebkhah, A., Tasnimi, A., and Moghadam, H. 2008. Nonlinear analysis of masonry-infilled steel frames with openings using discrete element method. *Journal of constructional steel research*, **64**(12):1463–1472.

Monti, G. and Spacone, E. 2000. Reinforced concrete fiber beam element with bond-slip. *Journal of Structural Engineering*, **126**(6):654–661.

Moretti, M. L., Papatheocharis, T., and Perdikaris, P. C. 2014. Design of reinforced concrete infilled frames. *Journal of Structural Engineering*, **140**(9):04014062.

Mosalam, K. M., Ayala, G., White, R. N., and Roth, C. 1997a. Seismic fragility of LRC frames with and without masonry infill walls. *Journal of Earthquake Engineering*, **1**(04): 693–720.

Mosalam, K. M., White, R. N., and Gergely, P. 1997b. Static response of infilled frames using quasi-static experimentation. *Journal of Structural Engineering*, **123**(11):1462–4169.

Musser, D. R. 1997. Introspective sorting and selection algorithms. *Software: Practice and Experience*, **27**(8):983–993.

Naghibi, F. and Fenton, G. A. 2019. Calibration of resistance factors for geotechnical seismic design. *Canadian Geotechnical Journal*, **56**(8):1134–1141.

Nasiri, E. and Liu, Y. 2017. Development of a detailed 3d FE model for analysis of the in-plane behaviour of masonry infilled concrete frames. *Journal of Engineering Structures*, 143:603 – 616. ISSN 0141-0296. doi: 10.1016/j.engstruct.2017.04.049.

Nasiri, E., 2019. *Experimental and numerical study of the out-of-plane behaviour of unreinforced masonry infills bounded by RC frames*. PhD thesis, Dalhosuie University, Halifax, Canada.

National Research Council Canada, 2015. National building code of canada. URL `http://www.nrc-cnrc.gc.ca`.

NVIDIA Corporation, 2016a. NVIDIA tesla P100: The most advanced datacenter accelerator ever built. URL `http://images.nvidia.com/`. v01.1.

NVIDIA Corporation. 2016b, P100 the most advanced datacenter accelerator ever built featuring pascal gp100, the world's fastest. Technical report. URL `http://images.nvidia.com/`.

NVIDIA Corporation. 2017, Nvidia tesla v100 gpu architecture. Technical report. URL `http://images.nvidia.com/`.

NVIDIA Corporation, 2018a. CUDA toolkit documentation. URL `https://docs.nvidia.com/cuda/`. v10.0.130.

NVIDIA Corporation, 2018b. NVIDIA CUDA C programming guide. URL https://docs.nvidia.com/cuda/. v10.0.

Owens, J. D., Luebke, D., Govindaraju, N., Harris, M., Krüger, J., Lefohn, A. E., and Purcell, T. J. 2007, A survey of general-purpose computation on graphics hardware. In *Proceedings of Computer graphics forum*, volume 26, pages 80–113. Wiley Online Library.

Paulay, T. and Priestley, M. N. 1992. *Seismic design of reinforced concrete and masonry buildings*. Wiley New York.

Polyakov, S. 1960. On the interaction between masonry filler walls and enclosing frame when loaded in the plane of the wall. *Journal of Translations in earthquake engineering*, **2**(3):36–42.

Priestley, M. J. N. and Elder, D. M. 1983. Stress-strain curves for unconfined and confined concrete masonry. *ACI Journal Proceedings*, **80**(3). doi: 10.14359/10834.

Prud'Homme, C., Chabannes, V., Doyeux, V., Ismail, M., Samake, A., and Pena, G. 2012, Feel++: A computational framework for galerkin methods and advanced numerical methods. In *ESAIM: Proceedings*, volume 38, pages 429–455. EDP Sciences. doi: 10.1051/proc/201238024.

Rahimi, R. and Liu, Y. 2017, Numerical investigation on the in-plane behaviour of concrete masonry infilled RC frames under quasi-static cyclic loading. In *Proceedings of the 13th Canadian Masonry Symposium (13CMS)*, Halifax, NS, Canada, June 2017.

Rahimi, R., Fenton, G. A., and Liu, Y. 2017, Reliability of unreinforced concrete masonry walls bounded by RC frames. In *Proceedings of the 13th Canadian Masonry Symposium (13CMS)*, Halifax, NS, Canada, June 2017.

Rahimi, R. and Liu, Y. 2018, In-plane behaviour of masonry infilled RC frames subjected to quasi-static cyclic loading. In *Proceedings of the 2018 Fredericton Annual Conference*, Canadian Society of Civil Engineers, Fredericton, NB, Canada, June 2018.

Rahimi, R., Liu, Y., and Fenton, G., 2019. VPFEM: Vectorized and parallelized finite element method. https://zenodo.org/badge/latestdoi/144008731.

Rahimi, R., Liu, Y., and Fenton, G. 2020. Parallelized finite element library for modeling applications in structural engineering. *Submitted to the Canadian Journal of Civil Engineering*.

Sadeghian, V. and Vecchio, F. 2018. The modified compression field theory: then and now. *ACI Symposium Publication*, 328.

Sarhosis, V., Tsavdaridis, K., and Giannopoulos, I. 2014. Discrete element modelling (DEM) for masonry infilled steel frames with multiple window openings subjected to lateral load variations. *Open Construction and Building Technology Journal*, 8:93–103.

Sharma, G., Agarwala, A., and Bhattacharya, B. 2013. A fast parallel gauss jordan algorithm for matrix inversion using cuda. *Computers & Structures*, 128:31 – 37. ISSN 0045-7949. doi: 10.1016/j.compstruc.2013.06.015.

Shing, P. B. and Mehrabi, A. B. 2002. Behaviour and analysis of masonry-infilled frames. *Journal of Progress in Structural Engineering and Materials*, **4**(3):320–331.

Shterenlikht, A., Margetts, L., and Cebamanos, L. 2018. Modelling fracture in heterogeneous materials on hpc systems using a hybrid mpi/fortran coarray multi-scale cafe framework. *Journal of Advances in Engineering Software*, 125:155–166.

Smith, I. M., Griffiths, D. V., and Margetts, L. 2013. *Programming the finite element method*. John Wiley & Sons. ISBN 1119973341.

Stafford-Smith, B. 1966. Behavior of square infilled frames. *Journal of the Structural Division*, **92**(1):381–404.

Stafford-Smith, B. 1967. Methods for predicting the lateral stiffness and strength of multi-storey infilled frames. *Building Science*, **2**(3):247–257.

Stafford-Smith, B. and Carter, C. 1969, A method of analysis for infilled frames. In *proceedings of the institution of civil engineers*, volume 44, pages 31–48. Thomas Telford-ICE Virtual Library.

Stavridis, A. and Shing, P. B. 2010. Finite-element modeling of nonlinear behavior of masonry-infilled RC frames. *Journal of structural engineering*, **136**(3):285–296.

Steeves, R., 2017. In-plane behaviour of masonry infilled RC frames with interfacial gaps subjected to quasi-static loading. Master's thesis, Dalhosuie University, Halifax, Canada.

Suzuki, T., Choi, H., Sanada, Y., Nakano, Y., Matsukawa, K., Paul, D., Gülkan, P., and Binici, B. 2017. Experimental evaluation of the in-plane behaviour of masonry wall infilled RC frames. *Bulletin of Earthquake Engineering*, **15**(10):4245–4267.

Talebi, H., Silani, M., Bordas, S. P., Kerfriden, P., and Rabczuk, T. 2014. A computational library for multiscale modeling of material failure. *Journal of Computational Mechanics*, **53**(5):1047–1071.

The Masonry Society, 2016. Building code requirements for masonry structures and specification for masonry structures, TMS 402/602. The Masonry Society, Longmont, CO.

Tian, Y., Xie, L., Xu, Z., and Lu, X., 2015. *GPU-powered high-performance computing for the analysis of large-scale structures based on OpenSEES*, pages 411–418. doi: 10.1061/9780784479247.051.

Turkstra, C. and Ojinaga, J. 1980, Towards a canadian limit states masonry design code. In *Proceedings of the 2nd Canadian Masonry Symposium*, pages 133–141, Ottawa, Ontario, Canada, June 1980.

Turkstra, C., Ojinaga, J., and Shyu, C. 1982, Safety index analysis for a limit states code. In *Proceedings of the 2nd North American Masonry Conference*, pages 22.1–22.14, College Park, Maryland, August 1982.

Turkstra, C., Ojinaga, J., and Shyu, C. 1983, Development of a limit states masonry code. In *Proceedings of the 3rd Canadian Masonry Symposium*, pages 2.1–2.13, Edmonton, Alberta, Canada, June 1983.

Turkstra, C. J. 1989, Limit states design in masonry. In *Proceedings of Structural Safety and Reliability*, pages 2043–2050. ASCE.

Vecchio, F. 1992. Finite element modeling of concrete expansion and confinement. *Journal of Structural Engineering*, **118**(9):2390–2406.

Vecchio, F. 2000. Disturbed stress field model for reinforced concrete: formulation. *Journal of structural engineering*, **126**(9):1070–1077.

Vecchio, F. J. 1990. Reinforced concrete membrane element formulations. *Journal of Structural Engineering*, **116**(3):730–750.

Vecchio, F. J. and Collins, M. P. 1986. The modified compression-field theory for reinforced concrete elements subjected to shear. *ACI Journal*, **83**(2):219–231.

Vecchio, F. J. and Collins, M. P. 1993. Compression response of cracked reinforced concrete. *Journal of structural engineering*, **119**(12):3590–3610.

Vetterling, W. T., Teukolsky, S. A., Press, W. H., and Flannery, B. P. 1999. *Numerical recipes in C: the art of scientific computing*. Cambridge university press.

Wanstrath, C., Hyett, P. J., Preston-Werner, T., and Chacon, S., 2008. Github, inc. URL `https://github.com`.

Wilt, N. 2013. *The cuda handbook: A comprehensive guide to GPU programming*. Pearson Education. ISBN 0321809467.

Wong, J., Kuhl, E., and Darve, E. 2015. A new sparse matrix vector multiplication graphics processing unit algorithm designed for finite element problems. *International Journal for Numerical Methods in Engineering*, **102**(12):1784–1814. doi: 10.1002/nme.4865.

Appendices

Appendix A

Detailed overview of the kernel function

This section describes some of the data oriented algorithms used in the kernel function (Algorithm 5.1). For each step in the flowchart depicted in Figure 5.9 there is a function in the kernel function. In step 2, the total strain in an element is determined using the deformation vector $[d]$ and the element strain matrix $[B]$. As the same element type was used for all three continuum spaces, the process is the same for all elements. Algorithm A.1 shows the procedure for calculating the strain at each element. The input data of the procedure are the element number, the global displacement vector, and the element information such as the coordinates of the nodes and integration points. First, the displacement at each node of the element is extracted from the displacement vector. Then, the strain matrix $[B]$ of the element is calculated using the element information (Bathe, 2006). The strain matrix is then multiplied by the element displacement vector to calculate the element strain $[\epsilon]$.

Algorithm A.1. Casting the global displacement to element strain, step 2.

Data: d_{global}: global displacement ,$ElementDetails$: information on Integration points and coordinates
Result: ϵ: strain field at each element
Function *ElementDisplacementToStrain(*$ElementNumber, d_{global}$*)*
    `/* Find element displacement using global degrees of freedoms */`
    $d \leftarrow$ ElementDisplacement($ElementNumber$,d_{global})
    `/* Find strain matrix using the element shape function */`
    $B \leftarrow$ StrainMatrix($ElementNumber$, $ElementDetails$)
    `/* Calculate the element strain using the following relation */`
    $\epsilon \leftarrow Bd$
return ϵ

In step 3, the element net strain is determined, while preserving the compatibility conditions (Algorithm A.2):

Algorithm A.2. Calculating the net strain field, step 3.

Data: ϵ: element total strain, ϵ^s: element slip strain, ϵ^0: element elastic strain offset, ϵ^p: element plastic strain offset
Result: ϵ^n: element net strain
Function *ElementStrainToNetStrain(ϵ, ϵ^s, ϵ^0, ϵ^p)*
  $[\epsilon^n] \leftarrow [\epsilon] - \left([\epsilon^s] + [\epsilon^p] + \left[\epsilon^0\right]\right)$
return ϵ^n

In step 4, the principal strains are calculated from the net strain field $[\epsilon_n^i]$. Then, the principal stresses are evaluated and the cracked surface is determined (Algorithm A.3).

Algorithm A.3. Calculating the principal stresses, step 4.

Data: ϵ^n: element net strain, constitutive relations
Result: f: element principal stresses
Function *ElementStrianToStress(ϵ^n)*
  `/* Calculate the principal strains using Mohr circular formulation */`
  `/* Translate strains to stress using the constitutive relations */`
return f

In Step 5, the shear slip strain is recalculated, as the crack reorients, based on the instruction discussed for each continuum space. In step 6, the material stiffness matrix is constructed based on the secant stiffness method. As the element stiffness matrix of each element is defined in the global coordinates directions the material stiffness matrix is transformed by the inclination angle of the principal stresses (Algorithm A.4).

In step 7, as discuss by Rahimi et al. (2020), the element stiffness matrix $[k]$ is constructed through numerical integrations. To consider the strain offset into account the pre-strain nodal forces need to be evaluated (Vecchio, 1992, 1990). In step 8, slip strain offset is calculated. The nodal forces caused by shear slip is refined at each iteration. The element nodal forces caused by shear slip is determined by multiplying element stiffness matrix by the free nodal

Algorithm A.4. Calculating the material stiffness matrix, step 6.

Data: f: element principal stresses
Result: D: element Material matrix
Function *ElementMaterialMatrix(f)*

```
/* Calculate the secant moduli at the principal stress direction      */
```

$$[D_p] \leftarrow \begin{bmatrix} E_1^i = \frac{f_1^i}{\epsilon_1^i} & 0 & 0 \\ 0 & E_2^i = \frac{f_2^i}{\epsilon_2^i} & 0 \\ 0 & 0 & G^i = \frac{E_1^i E_2^i}{E_1^i + E_2^i} \end{bmatrix}$$

```
/* Transformation matrix                                              */
```

$$[T] \leftarrow \begin{bmatrix} \cos^2\theta_\sigma & \sin^2\theta_\sigma & \cos\theta_\sigma \sin\theta_\sigma \\ \sin^2\theta_\sigma & \cos^2\theta_\sigma & -\cos\theta_\sigma \sin\theta_\sigma \\ -2\cos\theta_\sigma \sin\theta_\sigma & 2\cos\theta_\sigma \sin\theta_\sigma & \cos^2\theta_\sigma - \sin^2\theta_\sigma \end{bmatrix}$$

```
                                   // θ_σ is inclination of principal stresses
/* Calculate the material matrix                                      */
```

$[D] \leftarrow [T]^T [D_p][T]$

return D

displacement. The free nodal displacement was determined by integrating the element slip strain over the area of the element (Algorithm A.5).

Algorithm A.5. Calculating the pre-strain nodal forces, step 8.

Data: ϵ^s: element slip strain
Result: F^s: pre-strain nodal forces
Function *ElementNodalForces(ϵ^s)*

```
/* Calculate the free nodal displacement                              */
```

$$[r^s] \leftarrow \int [\epsilon^s] dA$$

```
/* Calculate the pre-strain nodal forces                              */
```

$[F^s] \leftarrow [k][r^s]$

return F^s

In step 9, the pre-strain forces are assembled then added to the nodal force vector from the last loading step (Algorithm A.6). The pre-strain nodal forces caused by elastic and plastic strain offsets are calculated at onset of each load increment using the similar procedure described in Algorithm A.5.

Algorithm A.6. Updating the nodal force vector

Data: F: Nodal forces from the current loading step, F^{*s}, F^{*0}, F^{*p}: Nodal forces due to strain offsets

Result: F': updated nodal force vector

Kernel *NodalForcesAssembly*(F, F^{*s}, F^{*0}, F^{*p})

```
/* Assemble pre-strain nodal forces                    */
```

$F^* \leftarrow F^{*s} + F^{*0} + F^{*p}$

```
/* update the force vector                             */
```

$F' \leftarrow F^* + F$

return F'