ADAPTIVE MULTI-TIME-STEP METHODS FOR DYNAMIC CRACK PROPAGATION

by

Mriganabh Boruah

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science

Lyles School of Civil Engineering

West Lafayette, Indiana

December 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Arun Prakash, Chair

Lyles School of Civil Engineering

Dr. Pablo D. Zavattieri

Lyles School of Civil Engineering

Dr. Mohammad R. Jahanshahi

Lyles School of Civil Engineering

Dr. Akanshu Sharma

Lyles School of Civil Engineering

Approved by:

Dr. Dulcy Abraham

ACKNOWLEDGMENTS

First and foremost, I would like to thank Professor Arun Prakash for his constant support and guidance for the past three and a half years. Coming into masters, I was horrified by the complex math associated with mechanics. I got lost in the formulas and missed the bigger picture. I am forever indebted to him for instilling the ideology that math is just a tool to mimic the real physical world.

I would like to show gratitude to my committee, Professor Pablo D. Zavattieri, Professor Mohammad R. Jahanshahi and Professor Akanshu Sharma for taking the time and effort to go through my work and for their valuable feedback.

I must express my appreciation towards my friends and family for the continuous encouragement when I doubted myself. Their continual reinforcement through thick and thin helped me maintain a level head through the course of my masters degree.

I am grateful to be part of a convivial and competitive lab where I am always learning new things about my research topic and recent developments in the industry.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Problems in structural dynamics that involve rapid evolution of the material at multiple scales of length and time are challenging to solve numerically. One such problem is that of a structure undergoing fracture, where the material in the vicinity of a crack front may experience high stresses and strains while the remainder of the structure may be unaffected by it. Usually, such problems are solved using numerical methods based on a finite element discretization in space and a finite difference time-stepping scheme to capture dynamic response. Regions of interest within the structure, where high transients are expected, are usually modeled with a fine discretization in space and time for better accuracy. In other regions of the model where the response does not change rapidly, a coarser discretization suffices and helps keep the computational cost down. This variation in spatial and temporal discretization is achieved through domain decomposition and multi-time-step coupling methods which allow the use of different levels of mesh discretization and time-steps in different regions of the mesh.

For problems where the region of interest evolves in time, such as those with an advancing crack front, the discretization of the problem domain must evolve as well so that the region of fine spatial and temporal discretization is able to track the region of interest. This need for adaptively refining and coarsening different regions of a numerical model presents several challenges. First, one must establish criteria based on the physical characteristics of the dynamic response of the material to identify the regions of interest that require a fine discretization in space and time. Then, one must devise a strategy to ensure that the region of fine discretization encompasses the region of interest at all times during the simulation without necessarily having to constantly modify the discretization at every time step. As the mesh discretization changes dynamically, an accurate and efficient mapping algorithm must also be implemented to transfer the state variables from the coarse mesh to the fine mesh and vice versa. Lastly, quantities associated with the domain decomposition and multi-time-step coupling method must also be updated every time the mesh discretization changes. These include the subdomain mass, stiffness and damping matrices and the interface coupling matrices needed to ensure spatial continuity of the solution between the subdomains.

In this study, all the challenges associated with dynamically evolving fine and coarse mesh discretizations are addressed by developing an adaptive multi-time-step (AMTS) domain decomposition and coupling method. Some of the challenges associated with adaptivity are also addressed in existing literature on adaptive mesh refinement, for example, but the approach described in this study is distinct from existing methods. First, two different criteria are investigated for identifying the regions of interest in a mesh. One is based on identifying regions of high spatial gradients in the solution, which would be regions of high stress and strain in the model, while the other is based on identifying the location of a dynamically propagating crack front in a brittle material. These regions are encapsulated with a fine discretization, which is periodically updated as the physical characteristics of the solution evolve. A nested mesh refinement and coarsening strategy is developed to ensure accurate mapping of the solution variables and efficient data transfer between the meshes. Finally, a novel method for storing and updating subdomain and interface matrices is described that minimizes the overhead associated with adaptivity.

Two numerical examples, one of a wave propagation in a rectangular plate and another of crack propagation in a notched beam, are used to demonstrate the effectiveness of the AMTS method developed in this study. The solution from this approach is compared to those obtained from three existing approaches. The first is the conventional non-adaptive uniform time-stepping (UTS) approach where all the possible regions of interest at all times during the simulation are identified *a priori*. These regions use a fine discretization in space but the problem is solved with a fine time step over the entire domain. In the second approach, adaptive mesh refinement is used to identify and refine the regions of interest, but the problem is solved with a uniform time step for the entire domain (AUTS). The third is a non-adaptive multi-time-step (MTS) approach, where pre-defined regions of interest are refined in space and different time steps are used in the coarse and fine regions of the mesh for computational efficiency. Reference solutions to these problems are obtained using a fine discretization in space and time over the entire problem domain. Results show that, for comparable solution accuracy, the AMTS method developed in this study is the most computationally efficient way of solving problems with evolving dynamics.

1. INTRODUCTION

Evaluating the response of a structure to external loads is central to the study of structural dynamics. This response is governed by a set of time-dependent partial differential equations which cannot be solved analytically for all but the most simple problems. Instead, approximate numerical solutions are obtained by using a combination of two methods - the finite element method (FEM) and finite difference (FD) method - which are used to discretize the governing partial differential equations in space and time respectively. The finite element method partitions the body into a number of small parts (elements), makes approximations on the kinematic quantities within each element using pre-defined functions, and combines these approximations across all the elements to evaluate the global response of the structure. The finite difference method approximates the time-derivatives in the equation of motion using difference formulas written at discrete instants of time and then advancing the solution by small increment, called a time-step. Chapter-2 provides further details regarding the spatial and temporal discretization used in this study.

When using finite difference methods for numerical time integration, it is common to advance the entire system of equations by a single time-step that must satisfy the accuracy and stability requirements of all elements in the model. The work of Felippa, Park & Underwood [1], [2], [3] allowed the use of larger time-steps in explicit time integration. Researchers, including Bergan & Mollestad [4] and Hulbert & Jang [5] found that updating the time-step size in real-time during a simulation could help reduce the computational run-time. However, for problems involving multiple spatial and temporal scales, usually there are small regions of interest in a large overall model of a structure. Since the regions of interest are usually associated with fine-scale dynamics, the maximum time-step size for the entire model is governed by the stringent accuracy and stability requirements of these regions. This makes the use of a uniform time-stepping (UTS) scheme computationally intensive because it constrains the entire problem domain to use the same time-step.

Related research in the area of multi-scale problems includes the work of Bettinotti, Allix and Malherbe [6], who developed a global/local non-intrusive coupling method to allow an overlapping local analysis to be activated only when necessary along with the global analysis. Ghosh and Cheng [7] proposed a subcycling algorithm that introduces different time steps in each subdomain

which are adaptively partitioned in the evolving computational domain based on the strain-rates. El-Amin et al [8] combined the concepts of automatic selection of time-step size with multi time-stepping and applied them to two-phase flow. Sanchez-Rivadeneira and Duarte [9] presented an approach based on generalized finite element method to reduce the computational time in analysis. This approach involves using enrichment functions within finite elements to capture localized singularities and discontinuities thereby allowing relatively coarser meshes. Soares [10]–[12] introduced explicit time integration schemes where the algorithmic parameters of the scheme are adaptively updated during the simulation to achieve greater accuracy at a reduced computational cost.

Belytschko and co-workers [13]–[15] proposed the method of decomposing the domain into different subdomains and solving them at different time-steps to overcome this problem. Finite element tearing and interconnect (FETI) by Farhat and Roux [16], [17] for static and transient problems is another common method which involves sub-structuring. Gravouil and Combescure (GC-method) [18], [19] extended FETI to enable the use of different time-steps between subdomains by prescribing continuity of velocities at the interface. Prakash and Hjelmstad (PH-method) [20], [21] improved the GC-method by eliminating numerical dissipation and reducing the computational cost of domain decomposition and mult-time-step (MTS) methods for linear and nonlinear problems. However, in all of the studies mentioned above, the subdomains were usually pre-defined and the decomposition was assumed to remain constant for the entire simulation. This strategy works when the region of interest does not evolve as the simulation progresses, but for problems where it does, such as crack propagation, using a static domain decomposition for the entire simulation is not effective.

In this study, a new approach to track the region of interest in a large structural model is developed. Based on the physical characteristics of the solution, the regions of interest are identified and encompassed in subdomains that are integrated with a small time-step while the rest of the model employs a larger time-step for computational efficiency. The decomposition into subdomains is adaptively updated as the region of interest evolves. This adaptive multi-time-step (AMTS) method improves computational performance in the numerical analysis of crack propagation problems.

2. FORMULATION

This chapter briefly discusses the spatial and temporal discretization of the equation of motion along with the primary material model used in this study. A numerical example of crack propagation in a notched beam is presented and used as a benchmark for evaluating the results obtained from methods discussed in the subsequent chapters of this thesis. Detailed descriptions of the formulation summarized in this chapter are available in standard texts (see [22]–[24]).

2.1 SPATIAL DISCRETIZATION

We employ the finite element method (FEM) to discretize the equation of motion in space. This method consists of splitting the body into numerous smaller parts (finite elements) to form a mesh which are combined together to closely resemble the original structure. The vertices of the elements in the mesh are called nodes. Nodes play an integral part in approximating the different kinematic quantities using shape functions. Additionally, each node is assigned a fixed number of degree of freedom depending on the problem.

The semi-discretized governing equation of motion for non-linear dynamic behaviour (assuming no damping) using finite element discretization along with the initial and boundary conditions can be written as:

$$\boldsymbol{M}\ddot{\boldsymbol{u}}(t)+\boldsymbol{p}(t) = \boldsymbol{f}(t) \tag{2.1}$$

$$\boldsymbol{u}(t) = \bar{\boldsymbol{u}}(t) \quad \text{on} \quad \Gamma_D \tag{2.2}$$

$$\boldsymbol{f}(t) = \bar{\boldsymbol{f}}(t) \quad \text{on} \quad \Gamma_N \tag{2.3}$$

$$\boldsymbol{u}(t_0)=\boldsymbol{u}_0 \quad ; \quad \dot{\boldsymbol{u}}(t_0)=\boldsymbol{v}_0 \tag{2.4}$$

Matrices and vectors in the above equations are assembled from the individual element contribution. Matrix $\boldsymbol{M}$ represents the mass matrix, $\ddot{\boldsymbol{u}}$ denotes the acceleration, $\boldsymbol{p}$ is the internal force vector and $\boldsymbol{f}$ is the external force vector. The initial displacement and velocity are specified as $\boldsymbol{u}_0$ and $\boldsymbol{v}_0$ respectively. The Dirichlet boundary with displacements specified is denoted as Γ_D and Γ_N represents the Neumann boundary with force boundary conditions. It is important to note that equations 2.1-2.3 are discrete in space but continuous in time.

2.2 TEMPORAL DISCRETIZATION

The semi-discrete system of ordinary differential equations 2.1 can be converted into a fully discrete system using a finite difference (FD) method for direct numerical time integration. In FD, the equations are solved at discrete instants of time and the solution is advanced by a small increment called a time step. Equation 2.1 is discretized and solved at individual instants of time t_n over the interval of interest [0,T] where $t_n = t_0 + \Sigma_{\mathrm{i}=1}^{n} \Delta t_{\mathrm{i}}$ with $t_0 = 0$, $t_N = T$, and Δt_{i} is called the time-step. Written at t_{n+1}, Equation 2.1 takes the form:

$$\boldsymbol{M}\ddot{\boldsymbol{u}}_{n+1} + \boldsymbol{p}_{n+1} = \boldsymbol{f}_{n+1} \tag{2.5}$$

where $\ddot{\boldsymbol{u}}_{n+1}$, $\boldsymbol{u}_{n+1}$, $\boldsymbol{p}_{n+1}$ and $\boldsymbol{f}_{n+1}$ are the acceleration, displacement, internal force and external force vectors, respectively, at the instant t_{n+1}.

Time integration schemes can be categorized as - implicit and explicit and each has its advantages and disadvantages. Implicit schemes require solving a full system of equations at every time-step but are usually stable for large time steps. Some implicit schemes are unconditionally stable. The Newmark method [25], Bathe method [26] and Wilson method [27]–[29] are examples of implicit schemes. Explicit schemes usually lead to a diagonal mass matrix that helps reduce the computational cost of one time step substantially. However, these schemes are only stable for time steps smaller than the Courant limit [24]:

$$\Delta t \leq \Delta t_{cr} = \frac{L}{c} \tag{2.6}$$

The Courant limit is determined by the highest natural frequency of the model which is influenced by the size of the smallest element in the mesh (L) and the wave speed in the material c. Examples of explicit schemes are the central difference method (which is a special case of the Newmark [25]), Chung and Lee [30], Zhai [31], and Hulbert and Chung [32]. In this study, we use the central difference explicit scheme.

With central difference explicit time integration, the displacement at time t_{n+1} can be evaluated from known quantities at time t_n as:

$$\boldsymbol{u}_{n+1} = \boldsymbol{u}_n + \Delta t\, \dot{\boldsymbol{u}}_n + \frac{\Delta t^2}{2}\ddot{\boldsymbol{u}}_n \tag{2.7}$$

Once $\boldsymbol{u}_{n+1}$ is known, the incremental strain $d\boldsymbol{\varepsilon}$ at instant $t_n + 1$ can be computed using shape functions of individual elements. We use triangular three node (T3) elements for spatial discretization in this study. From incremental strain $d\boldsymbol{\varepsilon}$, incremental stress $d\boldsymbol{\sigma}$ is obtained from a constitutive model of the material, which is then used to compute the local state of stress $\boldsymbol{\sigma}_{n+1}$. In this study, a quasi-brittle material model is used, as described in section 2.3. Internal force vector $\boldsymbol{p}_{n+1}$ is then computed as:

$$\boldsymbol{p}_{n+1} = \underset{\mathrm{e}=1}{\overset{m}{\mathrm{A}}} \blacktriangle^{\mathrm{e}} \boldsymbol{B}^{\mathrm{e}^T} \boldsymbol{\sigma}^{\mathrm{e}}_{n+1} \tag{2.8}$$

where m is the number of elements in the model, $\mathrm{A}_{\mathrm{e}=1}^{m}$ is the assembly operator for combining the internal force vectors of all the elements, $\blacktriangle^{\mathrm{e}}$ is the area-integral of the T3 element, $\boldsymbol{B}^{\mathrm{e}}$ is the matrix of derivatives of the shape functions and $\boldsymbol{\sigma}^{\mathrm{e}}_{n+1}$ is the stress vector in element e.

After computation of the internal force $\boldsymbol{p}_{n+1}$ in equation-(2.8), the acceleration and subsequently velocity at time t_{n+1} can be computed as:

$$\ddot{\boldsymbol{u}}_{n+1} = \boldsymbol{M}^{-1}[\boldsymbol{f}_{n+1} - \boldsymbol{p}_{n+1}] \tag{2.9}$$

and

$$\dot{\boldsymbol{u}}_{n+1} = \dot{\boldsymbol{u}}_n + \frac{\Delta t}{2}\ddot{\boldsymbol{u}}_n + \frac{\Delta t}{2}\ddot{\boldsymbol{u}}_{n+1} \tag{2.10}$$

This advances the solution by one time-step and the entire process is then repeated for the subsequent time steps.

2.3 QUASI-BRITTLE CRACKING

A brittle material behaves like an elastic material up till the ultimate load at which failure occurs, i.e. brittle fracture corresponds to small deformation. Brittle fracture is controlled by flaws or defects present in the material. Glass is a common example of a brittle material. Materials like concrete and polycrystalline ceramics behave like brittle material to an extent. However, they exhibit measurable deformation prior to failure and are classified as quasi-brittle. Once the ultimate stress is reached in these materials, they exhibit negative stiffness until complete failure, i.e. the stress and strain are inversely proportional. However, post failure, they look similar to a classically-brittle failed specimen. In this study, we limit our focus to quasi-brittle materials.

Researchers found that regularization techniques which establish a characteristic length within the discretization and separate crack interface behaviour from the constitutive behaviour of rest of the materials are ideal for representing quasi-brittle behavior. In this study, we focus on a technique called the smeared crack model [33]–[35]. The smeared crack model is based on the principle of spreading the energy release associated with fracture along the width of the localization band which in most cases is within a single element. The width of the band is calibrated to accurately model the dissipated energy.

A version of smeared crack model applied to concrete was developed by Rots and Blaauwendrad [36], [37] which is based on the crack band theory developed by Bazant and Oh [33]. This material model is adopted in this study and described briefly henceforth. In this model, once the maximum principal tensile stress (MPTS) exceeds the tensile strength of the material, a crack initiates in the element. The crack is oriented such that the normal to the crack surfaces aligns with the MPTS direction. The direction of crack in each element is fixed throughout the simulation. Post crack initiation, it is convenient to set up the crack traction-crack strain relations in the rectangular Cartesian system aligned with the crack, see Figure 2.1.

Once the total incremental strain ($d\boldsymbol{\varepsilon}$) is computed as per section 2.2, it is decomposed into strain representing the uncracked continuum between cracks ($d\boldsymbol{\varepsilon}^{el}$), and strain for the cracked part ($d\boldsymbol{\varepsilon}^{ck}$):

$$d\boldsymbol{\varepsilon} = d\boldsymbol{\varepsilon}^{\mathrm{el}} + d\boldsymbol{\varepsilon}^{ck} \tag{2.11}$$

For a 2-D plane stress configuration, the global crack strain vector consists of three components:

$$d\boldsymbol{\varepsilon}^{ck} = [d\varepsilon_{11}^{ck}\ d\varepsilon_{22}^{ck}\ d\gamma_{12}^{ck}]^T \tag{2.12}$$

which are related to the local crack strain components:

$$d\mathbf{e}^{ck} = [d\mathrm{e}_{nn}^{ck}\ dg_{ns}^{ck}]^T \tag{2.13}$$

using the transformation matrix, $\boldsymbol{N}$:

$$d\boldsymbol{\varepsilon}^{ck} = \boldsymbol{N} d\mathbf{e}^{ck} \tag{2.14}$$

The existing mode-I and mode-II crack properties help create a cracking matrix, $\boldsymbol{D}^{ck}$, which provides a relation between local cracking strains, e^{ck} and local cracking stresses, $\boldsymbol{t}$ as:

$$d\boldsymbol{t} = \boldsymbol{D}^{ck} d\mathbf{e}^{ck} \tag{2.15}$$

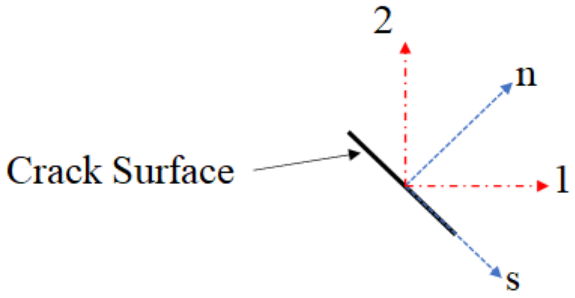


Figure 2.1. Local Coordinate System of crack

Finally, we can write the global stress increment as:

$$d\boldsymbol{\sigma} = \boldsymbol{D}^{\mathrm{el}}(d\boldsymbol{\varepsilon} - \boldsymbol{N} d\mathbf{e}^{ck}) \tag{2.16}$$

Upon arithmetic transformations and using the cracking conditions, we get:

$$d\boldsymbol{\sigma} = [\boldsymbol{D}^{el} - \boldsymbol{D}^{\mathrm{el}}\boldsymbol{N}(\boldsymbol{D}^{ck} + \boldsymbol{N}^T\boldsymbol{D}^{\mathrm{el}}\boldsymbol{N})^{-1}\boldsymbol{N}^T\boldsymbol{D}^{\mathrm{el}}]d\boldsymbol{\varepsilon} = \boldsymbol{D}^{\mathrm{ec}} d\boldsymbol{\varepsilon} \tag{2.17}$$

2.4 NUMERICAL EXAMPLE

In this section, we study crack propagation in a notched beam under three-point bending as shown in Fig. 2.2. The dimensions of the model are chosen to be 10 mm wide and 55 mm long. It contains a 45°notch of depth 2 mm situated halfway along the length. The model is pinned in the y-axis at a distance of 7 mm from each end and a linearly ramped load starting with 0 N/mm at t = 0 sec to 10 N/mm at t = 100 μsec is applied as shown Fig. 2.2.

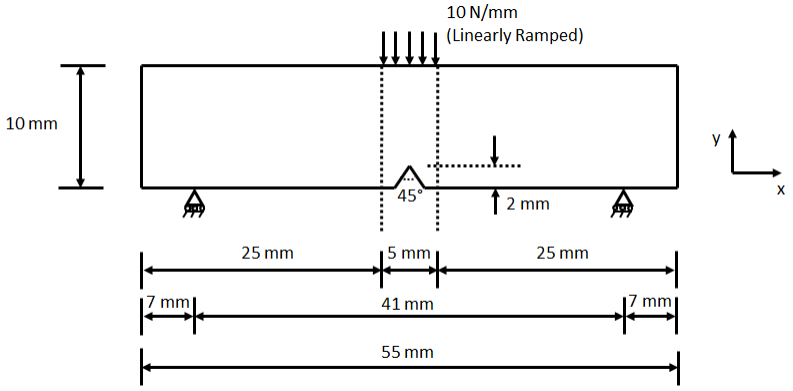


Figure 2.2. Notched three-point bending model

Material properties chosen for this problem are for glass (see [38]). The following material parameters are employed: Young's modulus $E = 70\times10^3$ N/mm^2, Poisson's ratio $\nu = 0.23$, density $\rho = 2.6\times10^{-9}$ tonne/mm^3, tensile strength $\sigma_{tu} = 70$ N/mm^2, critical stress intensity factor $K_c^I =$ 15.8-23.7 N/mm^2 mm$^{1/2}$. Mode-I fracture energy (G_f^I) is computed to be 0.0036-0.008 N/mm using $G_f^I = \frac{K_c^{I2}}{E}$. Glass cracks at a tensile strain of e_u^{cr} = 0.08 - 0.10 % (see [39]).

The quasi-brittle smeared crack material model described in section 2.3 is adopted to model the response of glass in this simulation. Even though glass is more akin to a brittle material, this model is used here with a small strain softening branch to approximate the brittle behavior of glass and to maintain stability of the in-house code. Nevertheless, this smeared crack material model is known to be mesh dependent and one must choose an appropriate element size when using this model (see [36], [37]). The characteristic length l_c for the model is given by:

$$l_c = 2 * \frac{G_f^I}{\sigma_{tu}} * \frac{1}{e_u^{cr}} \tag{2.18}$$

where G_f^I, σ_{tu} and e_u^{cr} are known material properties. From Eqn. 2.18, the characteristic length is computed to be $0.10-0.23$ mm and based on this, one may choose an element size h to adequately represent the material response around the crack-tip. In this study, square root of the area of the element is chosen as a measure of the element size and it is determined that T3 elements with an edge length of $0.125-0.250$ mm are required for this simulation.

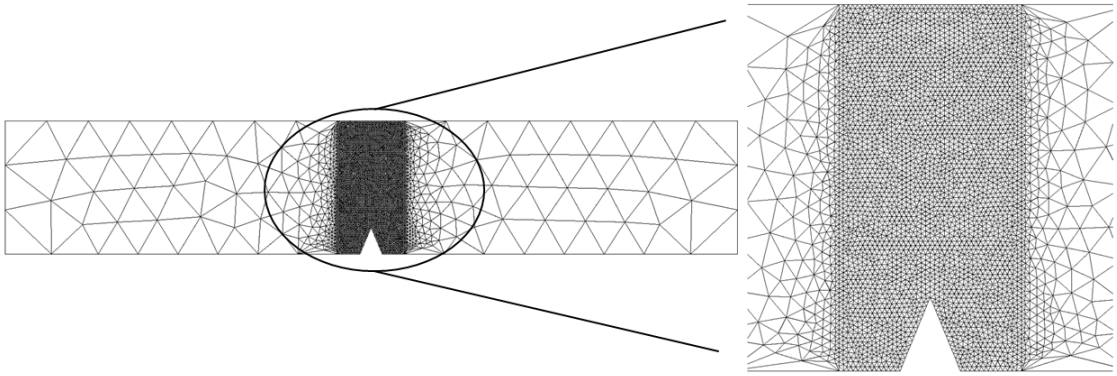


Figure 2.3. 3 point bending test - Reference mesh 1

Figure 2.4. 0.05 mm mesh in ABAQUS at 100 μsec

This problem is first simulated in the commercial FEM software ABAQUS [40] using the explicit version of the software. The material library of ABAQUS also includes a smeared crack model which is primarily intended to model unreinforced concrete but can be useful for other materials like ceramics, glass and brittle rocks. Similar to the material model used in this study, the ABAQUS material model is also based on the research of Rots and Blaauwendraad [36], [37]. The sample is discretized such that finer elements are around the notch tip in a region of 5 mm x 10 mm and rest of the domain is modeled using comparatively larger elements as shown in Figure 2.3. To investigate the mesh-dependence of this model, different discretizations are created where the size of the smallest elements in the cracking region range from 0.05 mm to 0.5 mm. Figure

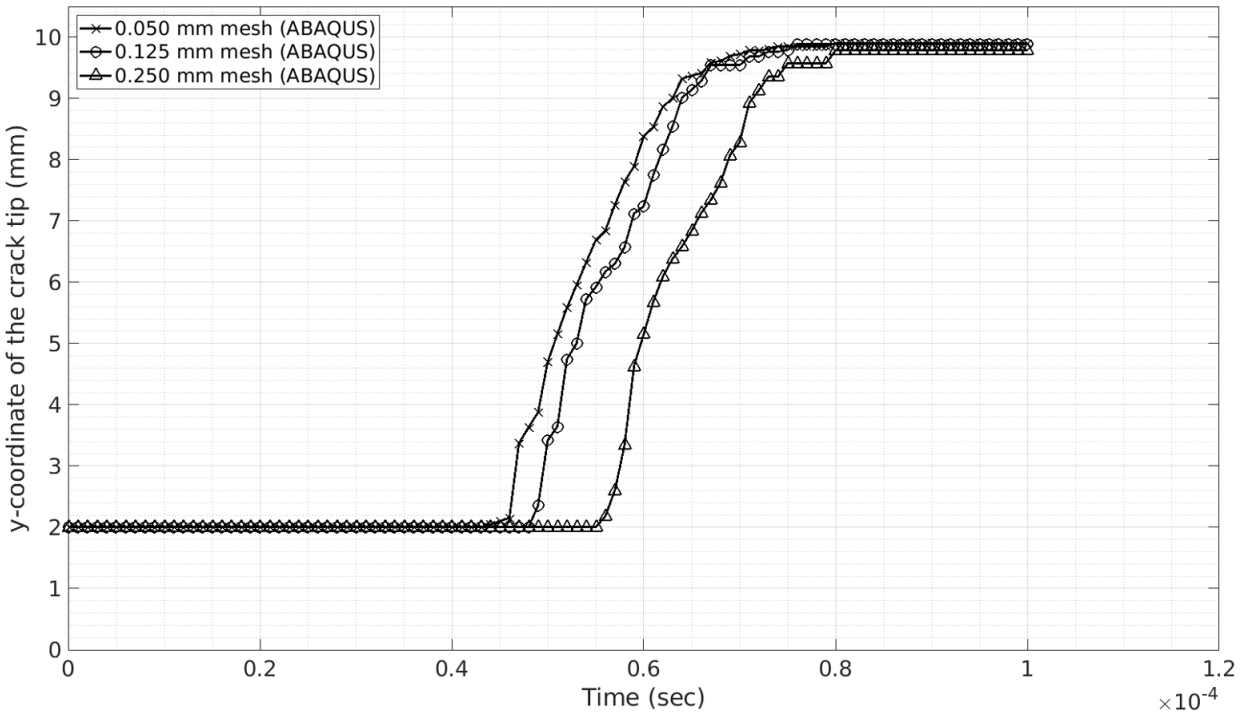


Figure 2.5. Crack-tip time-history for different meshes in ABAQUS

2.4 shows the response of the finest mesh with 0.05 mm elements in the cracking region. Figure 2.5 shows a plot of the evolution of the crack-tip with time. Note that the crack-tip starts at 2 mm due to the presence of 2mm deep notch. It is observed that crack initiation is delayed as the size of the element grows. Since the size of the elements in the coarsest mesh (0.05 mm) are larger than the characteristic length for the model, a crack did not initiate in this mesh during the 100 μsec simulation. Conversely, as the element size decreases, the damage localizes within a thin line of elements along the crack-path and does not exhibit the characteristic length (0.125 mm - 0.25 mm) for the model parameters. This confirms the mesh-dependent behavior of the smeared crack model since the results do not converge as the size of the elements is reduced.

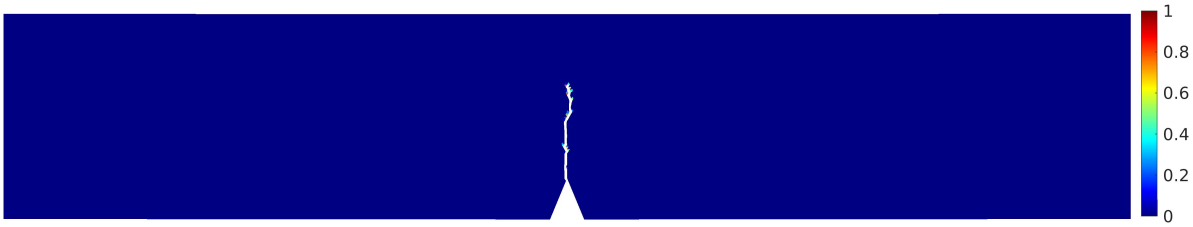
Based on the results from different element sizes, the discretization with 0.125 mm elements in the cracking region is chosen to track the crack propagation and is labelled as 'Reference mesh 1.' In the in-house code, simulations are conducted with a time step of 5×10^{-9} sec using central difference explicit time integration scheme. This is lower than the stable time increment of 2.3×10^{-8} sec for the 0.125 mm present in the mesh. Results from the in-house code developed during this study are compared to results from the commercial FEM software ABAQUS [40] for the same

(a)

(b)

Figure 2.6. Reference mesh 1 in ABAQUS and In-house code at 50 μsec respectively

(a)



(b)

Figure 2.7. Reference mesh 1 in ABAQUS and In-house code at 60 μsec respectively

discretization of Reference mesh 1 in Figures 2.6, 2.7 and 2.8. In ABAQUS the crack initiates at the notch around 49 μsec whereas it initiates at 49.22 μsec in the in-house code, see figure 2.9.

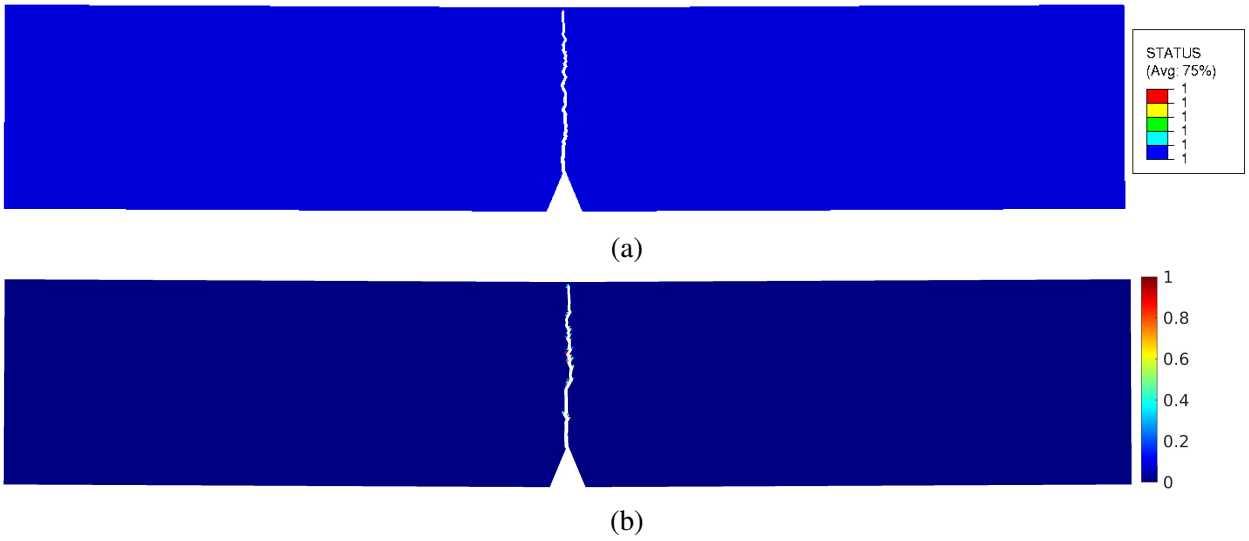


Figure 2.8. Reference mesh 1 in ABAQUS and In-house code at 100 μsec respectively

As discussed in Section 2.3, the maximum principal tensile stress in the element exceeding the tensile strength of the material leads to creation of an initial crack in the element. A damage measure (D_{mg}) was developed to represent the extent of failure in the element beyond this point:

$$D_{mg} = \frac{\mathrm{e}_{cur}^{cr}}{\mathrm{e}_{u}^{cr}} \tag{2.19}$$

where e_{cur}^{cr} is the current crack normal strain in the element and e_{u}^{cr} is the tensile cracking strain which is a material property. Damage D_{mg} can take values from 0 to 1, where 0 signifies absence of initial crack and 1 signifies complete failure of element. To verify the strain-softening behavior of the quasi-brittle model, a plot of crack normal stress and crack normal strain is shown for element-1496 in Figure 2.10 as it undergoes cracking. We observe a linear decrease in traction across the crack surface which is consistent with the quasi-brittle material model used.

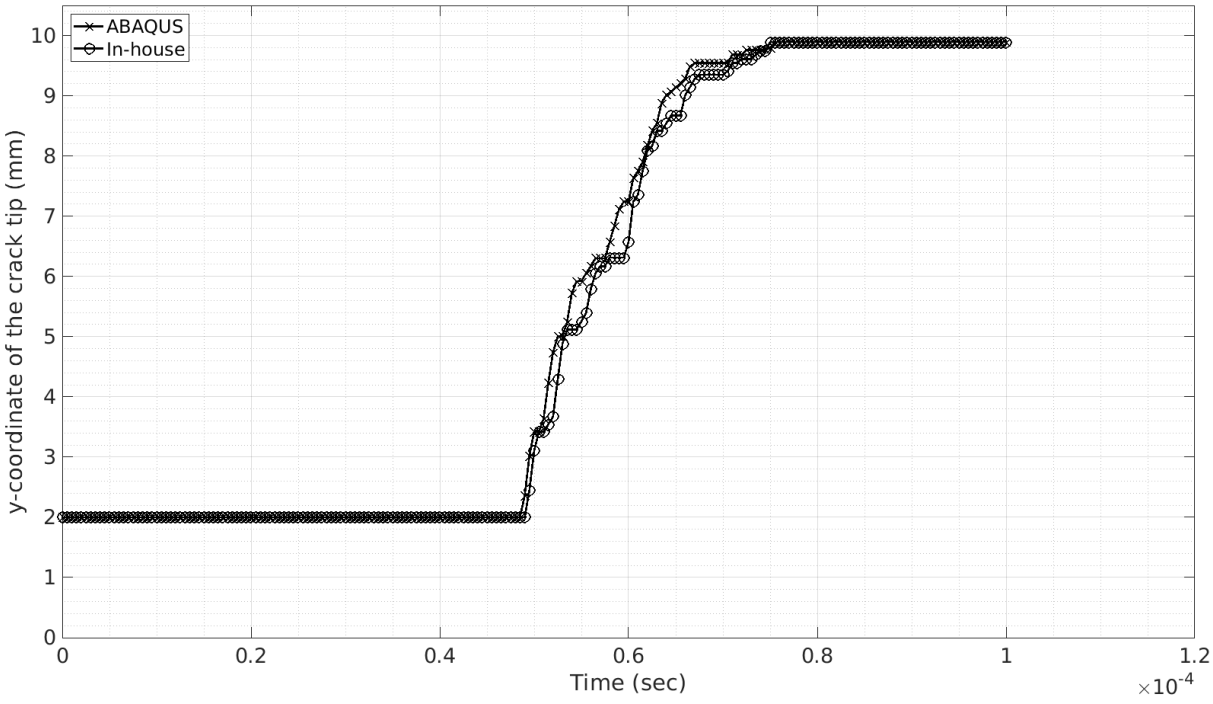


Figure 2.9. Crack-tip Time History in Ref Mesh-1

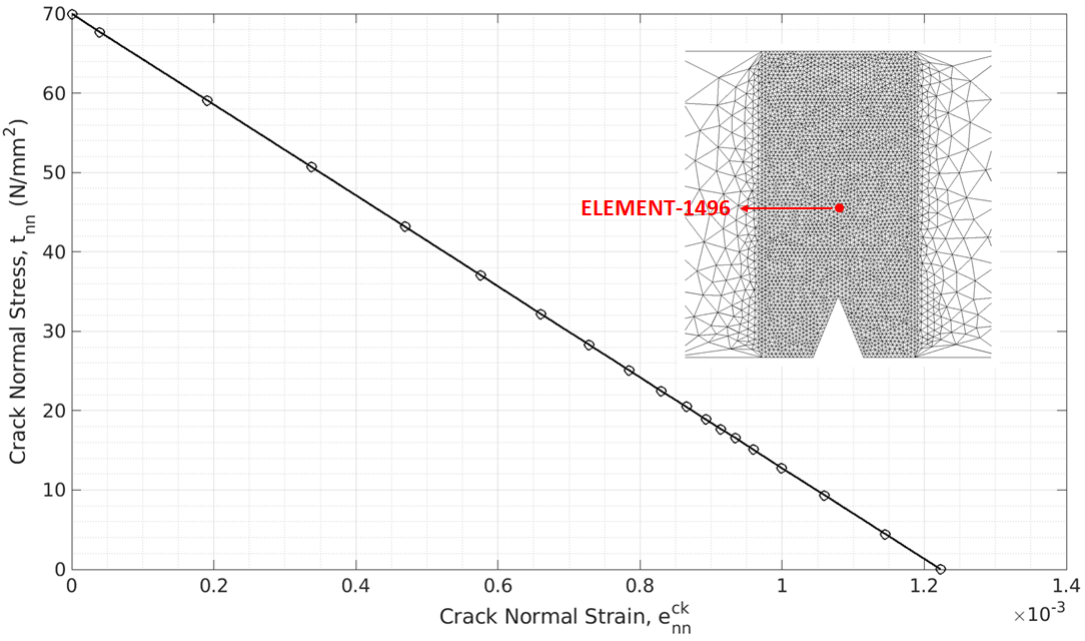


Figure 2.10. Crack Normal stress vs crack Normal strain for element-1496

2.4.1 Measures of Error

To quantify the differences between the ABAQUS vs in-house code results, we have developed four measures of errors -

1. Local Instantaneous Relative Error (LIRE) -

$$\varepsilon_n^m(x) = \frac{|x_n^m - \bar{x}_n^m|}{max(|\bar{x}|)} \tag{2.20}$$

where n refers to the time-step number, m refers to the DOF, x is the quantity under consideration (for example-displacement, velocity, acceleration, nodal stresses, etc.), x_n refers is the subject solution, $\bar{x}_n$ denotes the datum solution, $max(|\bar{x}|)$ denotes the absolute maximum datum value throughout the simulation. LIRE is a local measure of error which is evaluated at every time-step at each DOF or element of the reference solution.

2. Local Cumulative Relative Error (LCRE) -

$$\vartheta_m(x) = \frac{1}{N}\sum_{n=1}^{N} \frac{|x_n^m - \bar{x}_n^m|}{max(|\bar{x}|)} \tag{2.21}$$

where N refers to total number of timesteps. This is also a local measure of error which is the ratio of area enclosed between datum and subject solutions to the total area of the plot under $max(|\bar{x}|)$.

3. Global Instantaneous Relative Error (GIRE) -

$$\varepsilon_n(x) = \frac{1}{M}\sum_{m=1}^{M} \frac{|x_n^m - \bar{x}_n^m|}{max(|\bar{x}|)} \tag{2.22}$$

where M refers to total number of DOFs in the datum solution. GIRE is a global measure of error which is the average LIRE over all nodes or elements, i.e. GIRE is evaluated at every time-step.

4. Global Cumulative Relative Error (GCRE) -

$$\theta(x) = \frac{1}{M}\sum_{m=1}^{M}\frac{1}{N}\sum_{n=1}^{N}\frac{|x_n^m - \bar{x}_n^m|}{max(|\bar{x}|)} \tag{2.23}$$

This is a global measure of error which is the average LCRE over all DOFs. GCRE is a unique value.

While quantifying the accuracy of the in-house code, the datum and subject solutions are the solutions from ABAQUS and in-house code respectively. In Fig-2.11, the kinematic value under consideration is displacement. The maximum LIRE in Fig-2.11 is the maximum value of LIRE of all the nodes at a particular instant of time and GIRE is as per the above definition. The LIRE attains a maximum value of 5.57% whereas the GIRE attains a maximum value of 0.18%. The maximum LCRE is 1.2 % and GCRE is 0.04 %. The small value of local and global cumulative errors suggest that the results from the in-house code and ABAQUS compare well. It should be noted that the there is no difference/error until 49 μs, the time at which the crack starts initiating from the notch (see Figure 2.9).

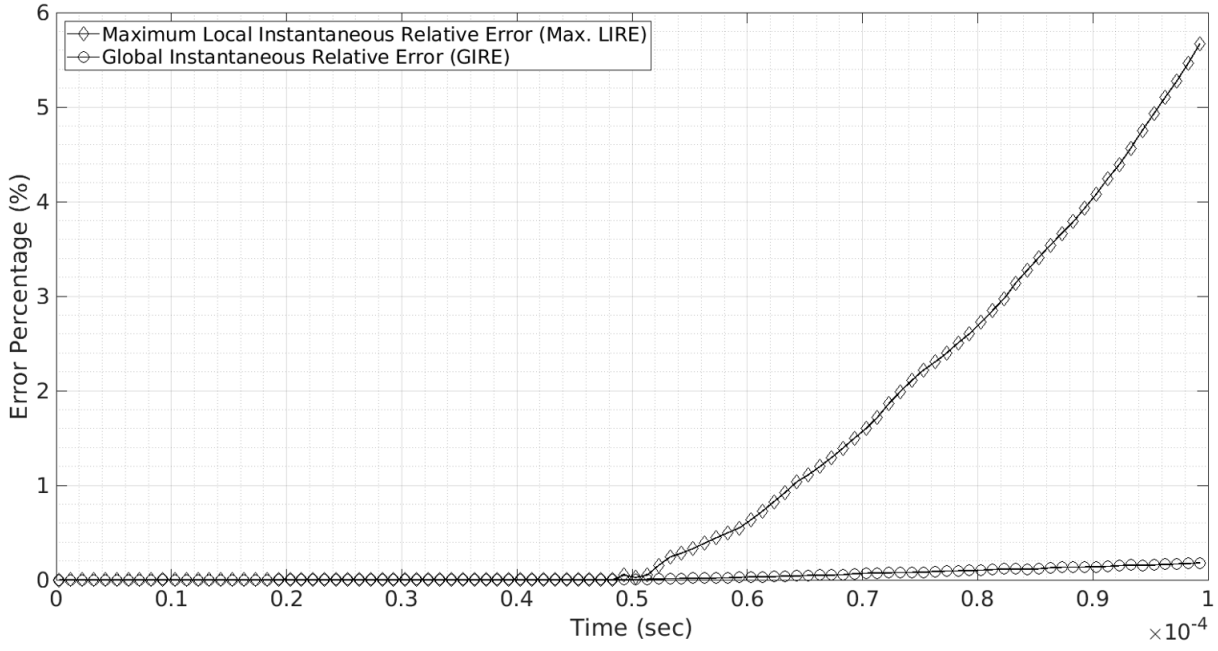


Figure 2.11. Displacement Error - ABAQUS vs In-house

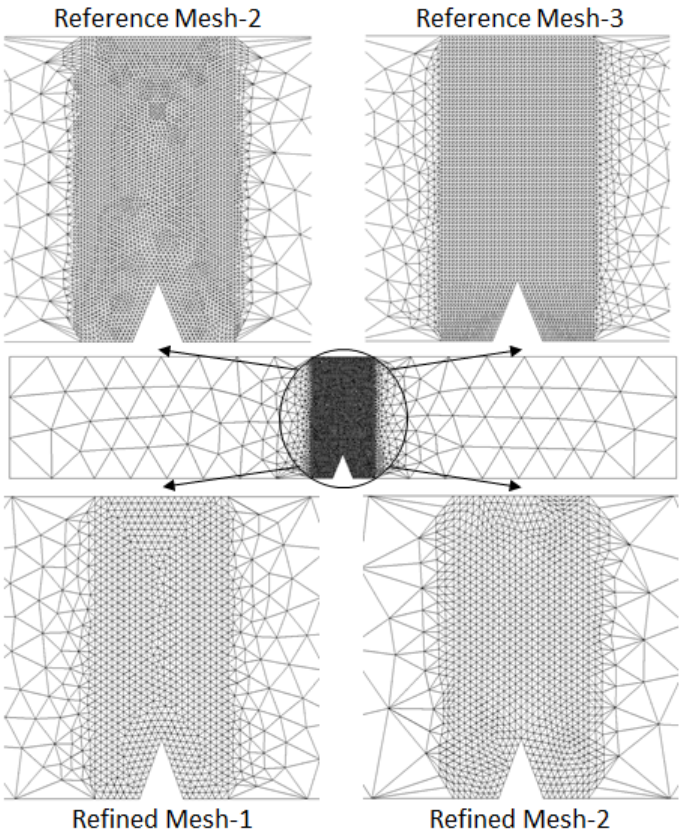


Figure 2.12. 3 point bending test - Different Discretization

2.4.2 Pixelation for Computing Error

Smeared crack models in finite element analysis have been known to exhibit mesh orientation bias or mesh sensitivity, refer Section 2.3. This means the orientation of the smeared crack depends on the orientation of the discretization. To account for this mesh sensitivity, the domain in Fig. 2.2 is discretized into four additional meshes, Reference mesh - 2, Reference mesh - 3, Refined mesh - 1 and Refined mesh - 2 as shown in Figure 2.12. Similar to Reference mesh - 1 (Figure 2.3), Reference meshes 2 and 3 are discretized using 0.125 mm T3 elements in the 5 mm x 10 mm region around the notch tip but with different mesh orientations in that region while Refined meshes 1 and 2 are discretized using 0.250 mm T3 elements.

Crack tip time history for all the Reference and Refined meshes is also shown in Figure 2.13. Note that while the time taken for the crack to propagate through the width of the plate is similar among all meshes, crack initiation is slightly delayed for the refined meshes. Damage patterns at

the end of the simulation (at 100 μs) in Reference meshes 2 and 3 are shown in Figures 2.14a and 2.14b respectively. Comparing these patterns to that of Reference mesh 1 in Figure 2.8b, it is evident that the damage pattern is different for all three cases and this leads to large errors in local quantities, such as displacement, in the cracked region. Global measures of error may still be compared, but they are not a good indicator of the quality of the solution in the region of interest - which is of primary concern in multi-scale problems and is the main objective of multi-time-step methods.

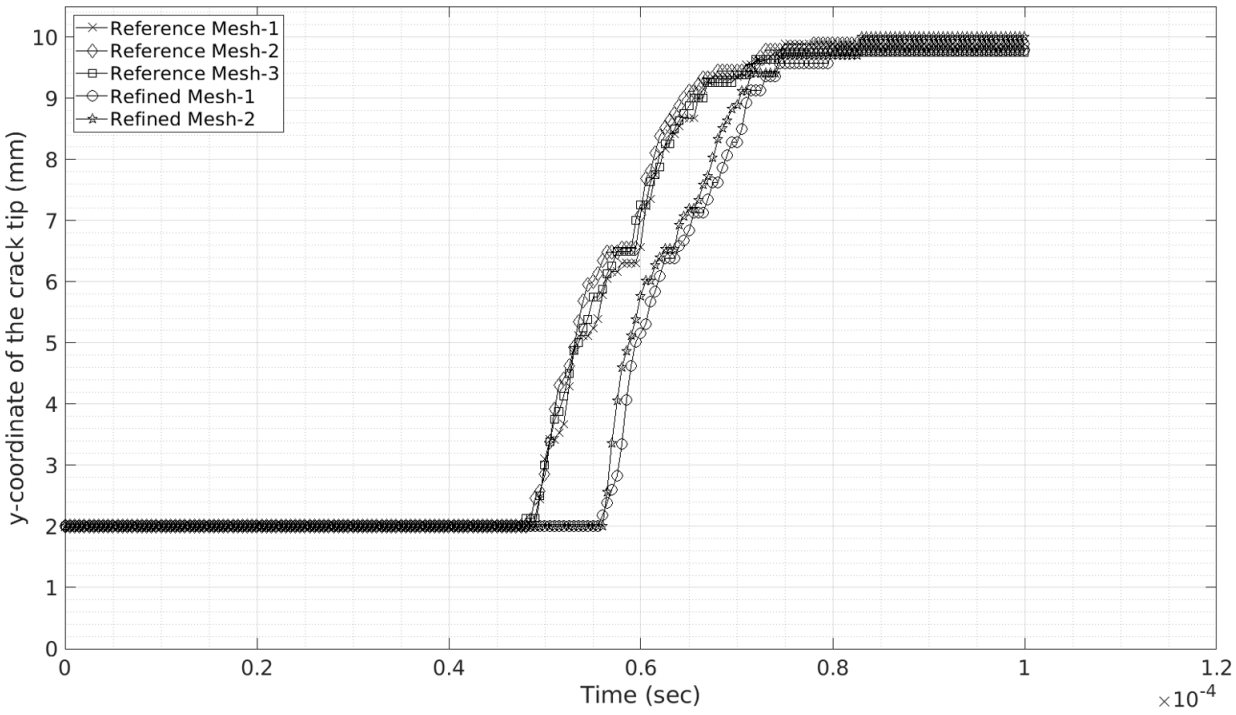


Figure 2.13. Crack-tip time-history for different Reference and Refined Meshes

The measures of error, as defined in section 2.4.1, work well for comparing different solutions obtained from the same spatial discretization. However, when solutions from different discretization have to be compared, they must first be mapped onto a common mesh to allow a direct one-to-one comparison. To facilitate this mapping, we introduce a pixelation of the problem domain, shown in Figure 2.15 onto which solutions from different meshes are mapped. Node-related solution quantities, such as displacements, are mapped to the nodes of the pixel mesh by simply projecting the finite element solution from the Reference and Refined meshes onto the pixel nodes. Element-related quantities, such as damage, are mapped by overlaying the pixel mesh on the finite

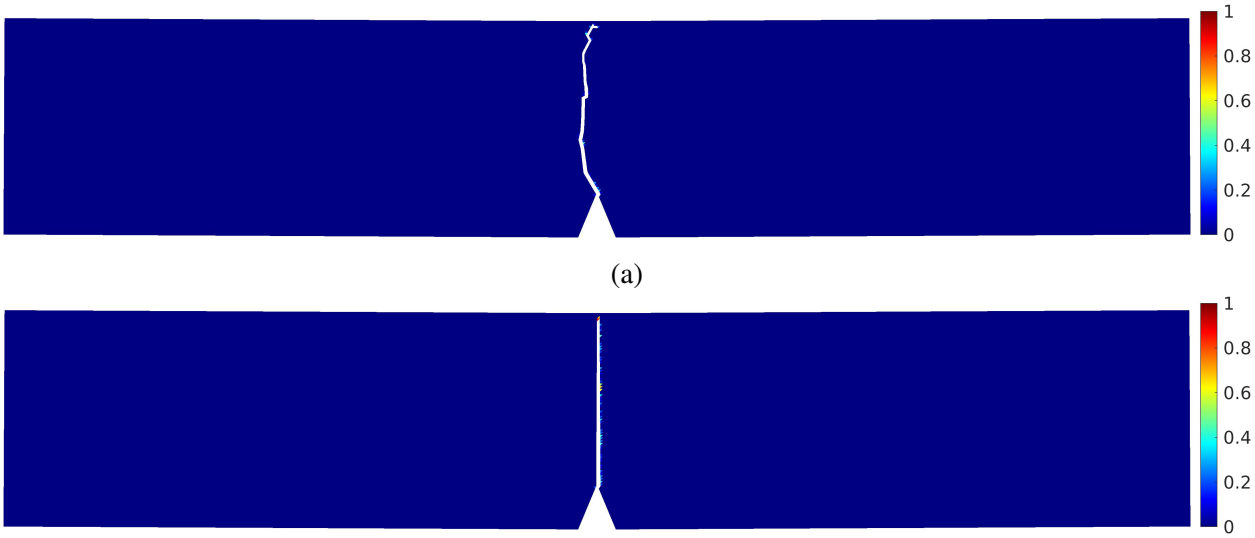


Figure 2.14. Reference mesh 2 and 3 at 100 μsec respectively

element mesh and computing the fraction of area of each pixel that overlaps with a particular element. The value to be assigned to a pixel, of damage for example, is then computed as a weighted average of the damage of its overlapping elements, weighted by the proportion of the overlapping area. The size of the pixel (3 mm) is chosen in such a way that it is insensitive to the variability in crack patterns and yet sensitive to the intensity of damage and its time-history of evolution as the crack propagates.

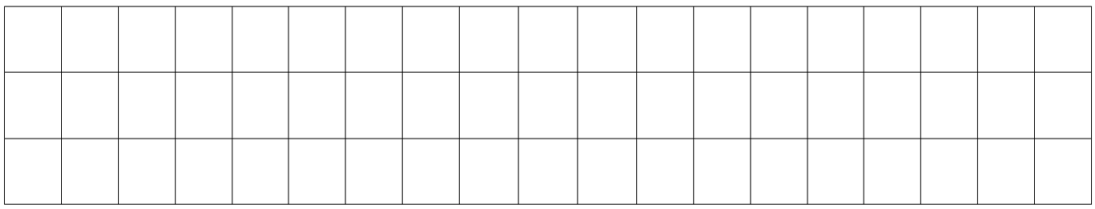


Figure 2.15. 3 mm rectangle pixel mesh

To compare the evolution of damage among all the Reference and Refined meshes, the four measures of error defined in section 2.4.1 are computed on the pixel mesh. Damage variables from Reference meshes 1, 2 and 3 are mapped to the pixel mesh and their average is used as the datum for computing damage error. Time history of evolution of maximum LIRE in damage from different

discretizations is shown in Figure 2.16. The maximum damage error over the entire simulation for the reference and refined meshes are in the range of 0.3-0.5% and 3.5-4% respectively.

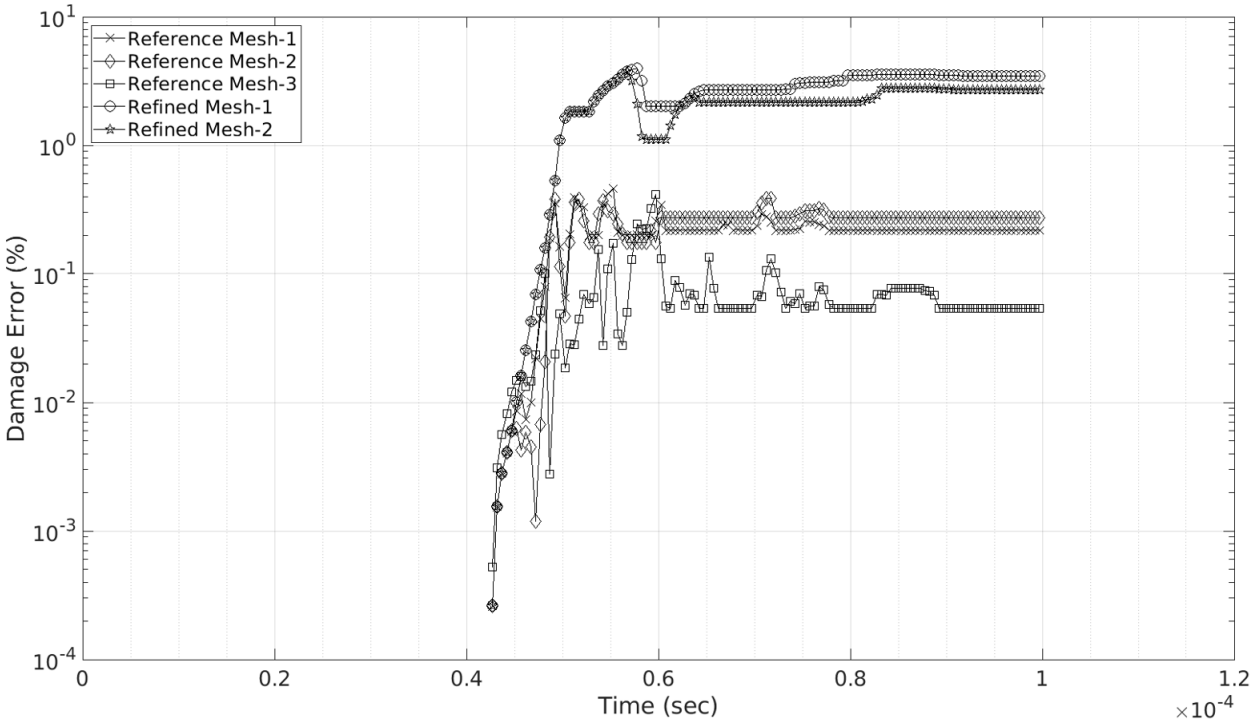


Figure 2.16. Maximum LIRE in Damage translated to 3mm rectangle pixel

2.4.3 Computational Runtime Cost Model

In this section, we develop a computational cost model to estimate the runtimes for different simulations solved using central difference explicit time integration scheme. When a uniform time-step (UTS) is used throughout the domain, then runtime is function of the number of Degrees of Freedom (DOFs) and the material model e.g. Linear Elastic, HyperElastic, Smeared Crack Model etc. To keep the cost model general and to decouple it from the cost associated with different material models, we will not consider the runtime spent in the computation of the internal force which depends on the material model being used. Hence, for UTS simulations, the runtime is just a function of DOFs.

The runtime can be split into two parts - initialization runtime and timeloop runtime. The initialization runtime consists of constructing the matrices required for time-stepping and initial acceleration calculation. The timeloop runtime is the runtime per time-step multiplied by the number of time-steps. For UTS, the number of time-steps are fixed and hence the timeloop runtime is a function of runtime per time-step. Table 2.1 lists the initialization and timeloop runtimes for the in-house code running on a standard desktop computer for meshes with different number of degrees of freedom. These data are plot in Figures 2.18 and 2.17 respectively and the resulting trendline serves as a computational cost model for UTS, allowing us to estimate the runtimes for different meshes used in the 3-point bending simulation of a notched beam. The total runtime can be estimated using the equation

$$(3.761 \times 10^{-8}\mathrm{x}^2 + 2.599 \times 10^{-4}\mathrm{x}) + \mathrm{N} \times (3.423 \times 10^{-7}\mathrm{x}) \tag{2.24}$$

where 'x' is the Dofs in the system and 'N' is the total number of timesteps.

Table 2.1. Uniform Time Step Cost Model Information

DOFs (#)	Initialization Runtime (sec)	Runtime per timestep w/o internal force computation (sec)
1,122	0.46	4.37×10^{-4}
1,904	0.76	6.56×10^{-4}
4,242	1.71	1.48×10^{-3}
8,120	4.56	2.74×10^{-3}
16,482	14.5	5.65×10^{-3}

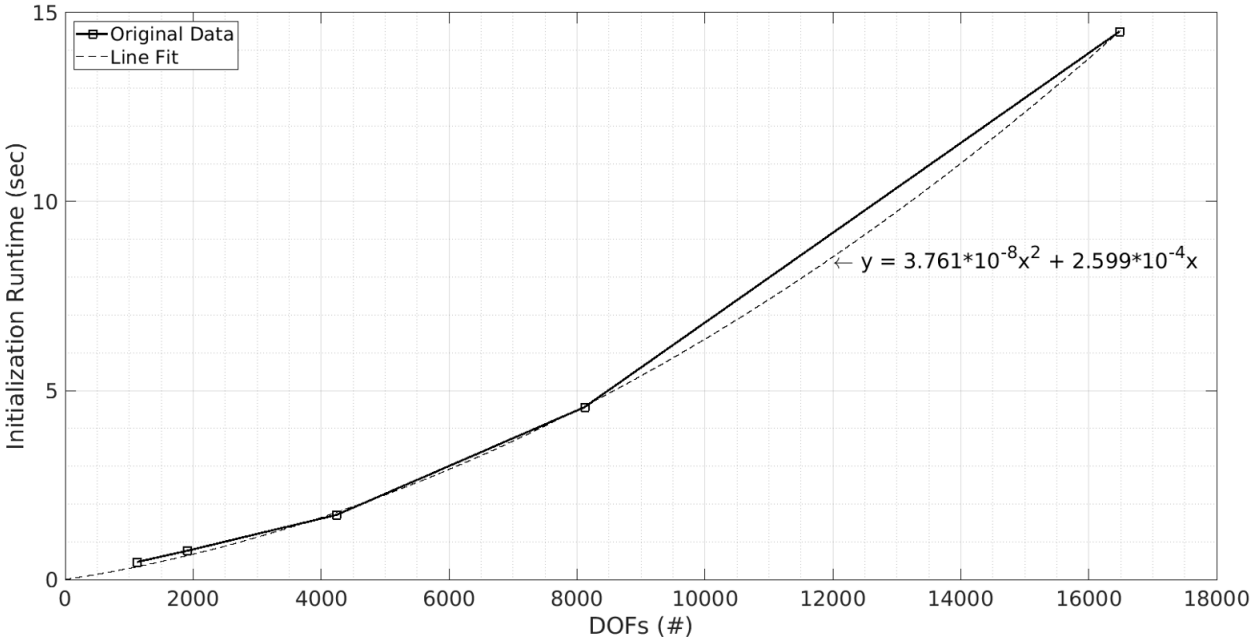


Figure 2.17. UTS Cost Model - Initialization Runtime vs DOFs

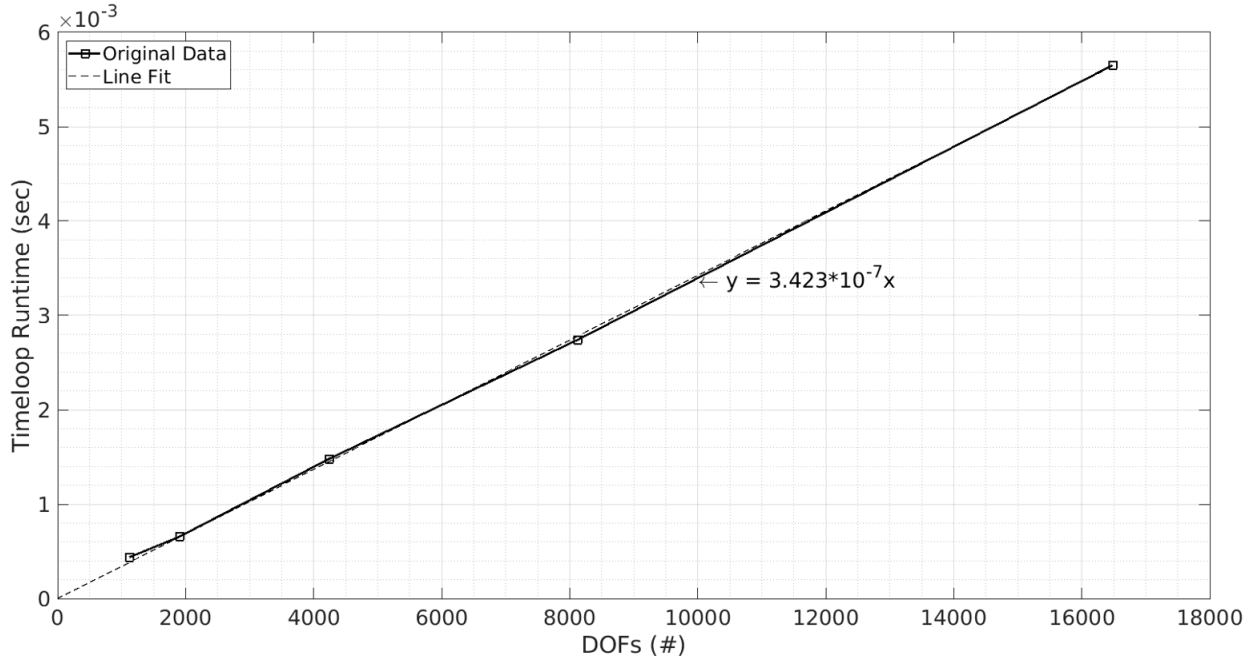


Figure 2.18. UTS Cost Model - Timeloop Runtime vs DOFs

2.4.4 Computational Runtime

Runtimes associated with the different Reference and Refined meshes used in this study are listed in Table 2.2. Employing a time-step of 5×10^{-9} sec for all the discretizations, the system was advanced through 20,000 timesteps during the simulation, (0 sec to 100 μsec).

Table 2.2. UTS Results and Predictions from UTS Cost Model

Mesh	Dofs (#)	Estimated Initialization runtime (sec)	Actual Initialization runtime (sec)	Relative Difference in estimation (%)
Reference mesh-1	7,660	4.20	4.60	9.59
Reference mesh-2	7,792	4.31	4.93	14.4
Reference mesh-3	7,766	4.29	4.44	3.58
Refined mesh-1	2,422	0.85	0.97	14.10
Refined mesh-2	2,430	0.85	0.96	12.46

Estimated Runtime per timestep w/o internal force comp (sec)	Actual Runtime per timestep w/o internal force comp (sec)	Relative Difference in estimation (%)	Total Computational Runtime (sec)	Max. Damage Error (%)
2.62×10^{-3}	2.90×10^{-3}	10.60	12,930	0.47
2.67×10^{-3}	2.81×10^{-3}	5.35	13,216	0.41
2.66×10^{-3}	2.70×10^{-3}	1.57	13,030	0.41
8.29×10^{-4}	9.48×10^{-4}	14.35	4,256	3.98
8.31×10^{-4}	9.60×10^{-4}	15.41	4,073	3.73

Table 2.2 shows that the initialization and runtime per timestep (excluding the internal force computation) predicted by the cost model, as discussed in Section 2.4.3, are 3.5-14.5% and 1.6-15.5% lower respectively than the actual runtimes. The total computational time reported in Table 2.2 is the entire time spent in the simulation which includes initialization runtime, timeloop runtime and internal force computation runtime.

In most dynamic simulations, the timeloop runtime including the internal force computation is the biggest component of the total runtime, much more than the initialization runtime. The timeloop runtime depends on the size of timestep, number of timesteps and the material model. Therefore, if one is either able to reduce the number of time-steps (by using larger time-steps) or reduce the average runtime per timestep, that can lead to a significant reduction in the total

runtime of the simulation. With UTS methods, however, the largest timestep that can be used in a simulation is governed by the fine-scale dynamics and that restricts the timestep for the entire domain. This in-turn places a restriction on the minimum number of timesteps in a simulation.

An alternative way to reduce computational cost is to split the domain into multiple subdomains and to solve subdomains of interest, which have fine-scale dynamics, with a small timestep and solve other subdomains with a large time step. Such methods, called multi-time-step (MTS) methods, are able to capture the fine-scale dynamics in the regions of interest accurately while allowing other regions to be solved at a much larger timestep without being restricted by a global maximum timestep. Chapter-3 discusses MTS methods in detail.

3. MULTI-TIME-STEP METHODS

Using a conventional time-integration method with an uniform time-step for the entire domain in large-scale problems containing elements of varying sizes can be computationally inefficient. One solution to this problem is to split the domain into multiple smaller subdomains and to solve each subdomain with a different time-step in accordance with its stability and accuracy requirements. In this study, we will employ the MTS-PH method which is a FETI-based multi-time-step coupling method [20], [21], which is summarized in this chapter.

For simplicity, we will decompose the domain two subdomains A and B. The semi-discrete governing equations of motion (assuming no damping) for the domain split into two subdomains Ω^A and Ω^B with an interface Γ^l are given by:

$$\boldsymbol{M}^A\ddot{\boldsymbol{u}}^A + \boldsymbol{p}^A + \boldsymbol{C}^{A^T}\boldsymbol{\lambda} = \boldsymbol{f}^A \tag{3.1}$$

$$\boldsymbol{M}^B\ddot{\boldsymbol{u}}^B + \boldsymbol{p}^B + \boldsymbol{C}^{B^T}\boldsymbol{\lambda} = \boldsymbol{f}^B \tag{3.2}$$

$$\boldsymbol{C}^A\dot{\boldsymbol{u}}^A + \boldsymbol{C}^B\dot{\boldsymbol{u}}^B = 0 \tag{3.3}$$

where $\boldsymbol{\lambda}$ denotes the vector of Lagrange multipliers used to enforce the continuity of velocities across the interface Γ^l and the matrices $\boldsymbol{C}^A$ and $\boldsymbol{C}^B$ are connectivity matrices that relate degrees of freedom from subdomains Ω^A and Ω^B respectively to degrees of freedom on the interface Γ^l. Note that Equation (3.3) enforces the constraint of continuity of velocities across Γ^l.

3.1 SINGLE-TIME-STEP FORMULATION

In this section, we first present the single-time-step (STS) method where both subdomains are solved with the same time-step, Δt. This is a special case of MTS and will help towards the formulation of MTS-PH method.

Recall that the central difference explicit time integration scheme allows the state of subdomain A to be advanced through one time-step (from t_n to t_{n+1}) using:

$$\begin{aligned} \boldsymbol{M}^A \ddot{\boldsymbol{u}}^A_{n+1} + \boldsymbol{p}^A_{n+1} + \boldsymbol{C}^{A^T} \boldsymbol{\lambda}_{\boldsymbol{n+1}} &= \boldsymbol{f}^A_{n+1} && (3.4) \\ \dot{\boldsymbol{u}}^A_{n+1} &= \dot{\boldsymbol{u}}^A_n + \frac{\Delta t}{2}\ddot{\boldsymbol{u}}^A_n + \frac{\Delta t}{2}\ddot{\boldsymbol{u}}^A_{n+1} && (3.5) \\ \boldsymbol{u}^A_{n+1} &= \boldsymbol{u}^A_n + \Delta t\, \dot{\boldsymbol{u}}^A_n + \frac{\Delta t^2}{2}\ddot{\boldsymbol{u}}^A_n && (3.6) \end{aligned}$$

Equations (3.5), (3.6) and (3.6) can be written as:

$$\mathbb{M}^A \mathbb{U}^A_{n+1} + \mathbb{P}^A_{n+1} + \mathbb{C}^A \boldsymbol{\lambda}_{n+1} = \mathbb{F}^A_{n+1} - \mathbb{N}^A \mathbb{U}^A_n \tag{3.7}$$

where

$$\begin{aligned} \mathbb{M}^A &= \begin{bmatrix} \boldsymbol{M}^A & 0 & 0 \\ -\frac{1}{2}\Delta t \boldsymbol{I}^A & \boldsymbol{I}^A & 0 \\ 0 & 0 & \boldsymbol{I}^A \end{bmatrix}, \mathbb{N}^A = \begin{bmatrix} 0 & 0 & 0 \\ -\frac{1}{2}\Delta t \boldsymbol{I}^A & -\boldsymbol{I}^A & 0 \\ -\frac{1}{2}\Delta t^2 \boldsymbol{I}^A & -\Delta t \boldsymbol{I}^A & \boldsymbol{I}^A \end{bmatrix} \\ \mathbb{P}^A_{n+1} &= \begin{bmatrix} \boldsymbol{p}^A_{n+1} \\ 0 \\ 0 \end{bmatrix}, \mathbb{U}^A = \begin{bmatrix} \ddot{\boldsymbol{u}}^A_{n+1} \\ \dot{\boldsymbol{u}}^A_{n+1} \\ \boldsymbol{u}^A_{n+1} \end{bmatrix}, \mathbb{F}^A = \begin{bmatrix} \boldsymbol{f}^A_{n+1} \\ 0 \\ 0 \end{bmatrix}, \mathbb{C}^A = \begin{bmatrix} \boldsymbol{C}^{A^T} \\ 0 \\ 0 \end{bmatrix} \end{aligned} \tag{3.8}$$

Since subdomain A and B have the same timesteps, the system of equations for subdomain B can be written as:

$$\mathbb{M}^B \mathbb{U}^B_{n+1} + \mathbb{P}^B_{n+1} + \mathbb{C}^B \boldsymbol{\lambda}_{n+1} = \mathbb{F}^B_{n+1} - \mathbb{N}^B \mathbb{U}^B_n \tag{3.9}$$

The continuity of velocities across the interface is ensured at the end of each time-step in this section:

$$\boldsymbol{C}^A \dot{\boldsymbol{u}}^A_{n+1} + \boldsymbol{C}^B \dot{\boldsymbol{u}}^B_{n+1} = 0 \tag{3.10}$$

Equations (3.7), (3.9) and (3.10) can be combined together and written in the form:

$$\left[\begin{array}{cc|c} \mathbb{M}^A & 0 & \mathbb{C}^A \\ 0 & \mathbb{M}^B & \mathbb{C}^B \\ \hline \mathbb{B}^A & \mathbb{B}^B & \mathbb{C}^A \end{array}\right] \left[\begin{array}{c} \mathbb{U}^A_{n+1} \\ \mathbb{U}^B_{n+1} \\ \hline \boldsymbol{\lambda}_{n+1} \end{array}\right] = \left[\begin{array}{c} \mathbb{F}^A_{n+1} - \mathbb{N}^A \mathbb{U}^A_n - \text{I}^A_{n+1} \\ \mathbb{F}^B_{n+1} - \mathbb{N}^A \mathbb{U}^B_n - \text{I}^B_{n+1} \\ \hline 0 \end{array}\right] \tag{3.11}$$

where $\mathbb{B}^k = [0 \;\; C^k \;\; 0]$. Note that the first row represents the system of equations for subdomain A, the second row for subdomain B and the last row imposes the continuity of velocity constraint at the final time step.

Equation (3.11) as can be solved using a bordered system approach. The subdomain blocks can be collected together and the system expressed as:

$$\begin{bmatrix} \mathsf{M} & \mathsf{C} \\ \mathsf{B} & 0 \end{bmatrix} \begin{bmatrix} \mathsf{u} \\ \boldsymbol{\lambda}_{n+1} \end{bmatrix} = \begin{bmatrix} \mathsf{f} \\ 0 \end{bmatrix} \tag{3.12}$$

Let

$$\mathsf{u} = \mathsf{v} + \mathsf{w} \;\text{ where } \mathsf{v} = \mathsf{M}^{-1}\mathsf{f},\ \mathsf{w} = -\mathsf{Y}\boldsymbol{\lambda}_{n+1} \text{ and } \mathsf{Y} = \mathsf{M}^{-1}\mathsf{C} \tag{3.13}$$

The assumption from Equation (3.13) ensures that the first row in (3.12) is satisfied as:

$$\begin{aligned} \mathsf{Mu} + \mathsf{C}\boldsymbol{\lambda}_{n+1} = \mathsf{f} &\Longrightarrow \mathsf{Mv} + \mathsf{Mw} + \mathsf{C}\boldsymbol{\lambda}_{n+1} = \mathsf{f} \\ &\Longrightarrow (\mathsf{Mv} - \mathsf{f}) = (\mathsf{MY} - \mathsf{C})\,\boldsymbol{\lambda}_{n+1} \end{aligned} \tag{3.14}$$

Using the second row of Equation (3.12), the interface forces can be computed as

$$\mathsf{Bu} = 0 \Longrightarrow \mathsf{Bv} + \mathsf{Bw} = 0 \Longrightarrow [\mathsf{BY}]\boldsymbol{\lambda}_{n+1} = \mathsf{Bv} \tag{3.15}$$

3.1.1 Computing v

Use $\mathsf{Mv} = \mathsf{f}$ to compute v.

$$\begin{bmatrix} \mathbb{M}^A & 0 \\ 0 & \mathbb{M}^B \end{bmatrix} \begin{bmatrix} \mathbb{V}^A_{n+1} \\ \mathbb{V}^B_{n+1} \end{bmatrix} = \begin{bmatrix} \mathbb{F}^A_{n+1} - \mathbb{N}^A\mathbb{U}^A_n - \mathbb{P}^A_{n+1} \\ \mathbb{F}^B_{n+1} - \mathbb{N}^A\mathbb{U}^B_n - \mathbb{P}^B_{n+1} \end{bmatrix} \tag{3.16}$$

The first row of Equation (3.16) can be expanded to:

$$\begin{aligned} \boldsymbol{M}^A\ddot{\boldsymbol{v}}^A_{n+1} &= \boldsymbol{f}^A_{n+1} - \boldsymbol{p}^A_{n+1} \\ \dot{\boldsymbol{v}}^A_{n+1} &= \dot{\boldsymbol{u}}^A_n + \frac{\Delta t}{2}\ddot{\boldsymbol{u}}^A_n + \frac{\Delta t}{2}\ddot{\boldsymbol{v}}^A_{n+1} \\ \boldsymbol{v}^A_{n+1} &= \boldsymbol{u}^A_n + \Delta t\,\dot{\boldsymbol{u}}^A_n + \frac{\Delta t^2}{2}\ddot{\boldsymbol{u}}^A_n \end{aligned} \tag{3.17}$$

3.1.2 Computing Y

Use $\mathsf{MY} = \mathsf{C}$ to compute Y.

$$\begin{bmatrix} \mathbb{M}^A & 0 \\ 0 & \mathbb{M}^B \end{bmatrix} \begin{bmatrix} \mathbb{Y}^A_{n+1} \\ \mathbb{Y}^B_{n+1} \end{bmatrix} = \begin{bmatrix} \mathbb{C}^A_{n+1} \\ \mathbb{C}^B_{n+1} \end{bmatrix} \tag{3.18}$$

The first row of Equation (3.18) can be expanded to:

$$\begin{aligned} \boldsymbol{M}^A \ddot{\boldsymbol{Y}}^A_{n+1} &= \boldsymbol{C}^{A^T} \\ \dot{\boldsymbol{Y}}^A_{n+1} &= \dot{\boldsymbol{Y}}^A_n + \frac{\Delta t}{2}\ddot{\boldsymbol{Y}}^A_n + \frac{\Delta t}{2}\ddot{\boldsymbol{Y}}^A_{n+1} \\ \boldsymbol{Y}^A_{n+1} &= \boldsymbol{Y}^A_n + \Delta t\, \dot{\boldsymbol{Y}}^A_n + \frac{\Delta t^2}{2}\ddot{\boldsymbol{Y}}^A_n \end{aligned} \tag{3.19}$$

3.1.3 Computing $\boldsymbol{\lambda}_{n+1}$

Use $(\mathsf{BY})\boldsymbol{\lambda}_{n+1} = \mathsf{BV}$ to compute $\boldsymbol{\lambda}_{n+1}$.

$$\begin{bmatrix} \mathbb{B}^A & \mathbb{B}^B \end{bmatrix} \begin{bmatrix} \mathbb{Y}^A_{n+1} \\ \mathbb{Y}^B_{n+1} \end{bmatrix} \boldsymbol{\lambda}_{n+1} = \begin{bmatrix} \mathbb{B}^A & \mathbb{B}^B \end{bmatrix} \begin{bmatrix} \mathbb{V}^A_{n+1} \\ \mathbb{V}^B_{n+1} \end{bmatrix} \tag{3.20}$$

The Equation (3.20) can be expanded to:

$$(\boldsymbol{C}^A \dot{\boldsymbol{Y}}^A_{n+1} + \boldsymbol{C}^B \dot{\boldsymbol{Y}}^B_{n+1})\boldsymbol{\lambda}_{n+1} = \boldsymbol{C}^A \dot{\boldsymbol{V}}^A_{n+1} + \boldsymbol{C}^B \dot{\boldsymbol{V}}^B_{n+1} \tag{3.21}$$

3.1.4 Computing w

Use $\mathsf{w} = \text{-}\mathsf{Y}\boldsymbol{\lambda}_{n+1}$ to compute w.

$$\begin{bmatrix} \mathbb{W}^A_{n+1} \\ \mathbb{W}^B_{n+1} \end{bmatrix} = - \begin{bmatrix} \mathbb{Y}^A_{n+1} \\ \mathbb{Y}^B_{n+1} \end{bmatrix} \boldsymbol{\lambda}_{n+1} \tag{3.22}$$

The first row of Equation (3.22) can be expanded to:

$$\begin{aligned} \boldsymbol{w}^A_{n+1} &= -\boldsymbol{Y}^A_{n+1}\boldsymbol{\lambda}_{n+1} \\ \dot{\boldsymbol{w}}^A_{n+1} &= -\dot{\boldsymbol{Y}}^A_{n+1}\boldsymbol{\lambda}_{n+1} \\ \ddot{\boldsymbol{w}}^A_{n+1} &= -\ddot{\boldsymbol{Y}}^A_{n+1}\boldsymbol{\lambda}_{n+1} \end{aligned} \tag{3.23}$$

3.1.5 Computing u

From sections - 3.1.1 and 3.1.4, $\mathsf{u} = \mathsf{v} + \mathsf{w}$ is computed. When using smeared crack model, damage in the domain is updated after the computation of u.

3.2 MULTI-TIME-STEP FORMULATION

This section has the same semi-discretized equations of motion as Section 3.1 which are Equations (3.2), (3.3 and (3.3). However, the subdomain A and B are solved at different time-step, ΔT and Δt respectively where $\Delta T = m\Delta t$. MTS-PH method allows both the subdomains to be solved at their respective time-steps and require the continuity Equation (3.3) to be solved only at the bigger time-step i.e. ΔT. For convenience, the coupling method is shown for proceeding the solution by ΔT from t_0 to $t_m = t_0 + \Delta T$. This can be easily extended for proceeding the solution from t_n to t_{n+m}. The fully discretized equations of motion for subdomain A and B and continuity equation can be written as:

$$\mathbb{M}^A\mathbb{U}^A_m + \mathbb{P}^A_m + \mathbb{C}^A\boldsymbol{\lambda}_m = \mathbb{F}^A_m - \mathbb{N}^A\mathbb{U}^A_0 \tag{3.24}$$

$$\mathbb{M}^B\mathbb{U}^B_{\mathrm{j}} + \mathbb{P}^B_{\mathrm{j}} + \mathbb{C}^B\boldsymbol{\lambda}_{\mathrm{j}} = \mathbb{F}^B_{\mathrm{j}} - \mathbb{N}^B\mathbb{U}^B_{\mathrm{j}-1} \;\; \forall\, \mathrm{j}\; \varepsilon\; [1,2,...,m] \tag{3.25}$$

$$\boldsymbol{C}^A\dot{\boldsymbol{u}}^A_m + \boldsymbol{C}^B\dot{\boldsymbol{u}}^B_m = 0 \tag{3.26}$$

Upon arithmetic transformations (see MTS-PH method for details), Equations (3.25), (3.26) and (3.26) can be combined together and written in the form:

$$\left[\begin{array}{cccc:c:c} \mathbb{M}^B & & & & & \frac{1}{m}\mathbb{C}^B \\ \mathbb{N}^B & \mathbb{M}^B & & & & \frac{2}{m}\mathbb{C}^B \\ & \ddots & \ddots & & & \vdots \\ & & \mathbb{N}^B & \mathbb{M}^B & & \mathbb{C}^B \\ \hdashline & & & & \mathbb{M}^A & \mathbb{C}^A \\ \hline & & & \mathbb{B}^B & \mathbb{B}^A & 0 \end{array}\right] \left[\begin{array}{c} \mathbb{U}_1^B \\ \mathbb{U}_2^B \\ \vdots \\ \mathbb{U}_m^B \\ \hdashline \mathbb{U}_m^A \\ \hline \boldsymbol{\lambda}_{n+1} \end{array}\right] = \left[\begin{array}{c} \mathbb{F}_1^B - \mathbb{N}^B\mathbb{U}_0^B - \mathbb{P}_1^B - \mathbb{C}^B\mathbf{S}_1 \\ \mathbb{F}_2^B - \mathbb{P}_2^B - \mathbb{C}^B\mathbf{S}_2 \\ \vdots \\ \mathbb{F}_m^B - \mathbb{P}_m^B - \mathbb{C}^B\mathbf{S}_{\mathrm{m}} \\ \hdashline \mathbb{F}_m^A - \mathbb{P}_m^A - \mathbb{N}^A\mathbb{U}_0^A \\ \hline 0 \end{array}\right] \tag{3.27}$$

where $\mathbf{S}_{\mathrm{j}} = (1-\frac{\mathrm{j}}{m})\boldsymbol{\lambda}_0 + \boldsymbol{C}^A[\boldsymbol{f}_{\mathrm{j}}^A - (1-\frac{\mathrm{j}}{m})\boldsymbol{f}_0^A - \frac{\mathrm{j}}{m}\boldsymbol{f}_m^A]$. Using the bordered procedure again, Equation (3.27) is solved in the following sections.

3.2.1 Computing v

Use $\mathsf{M}\mathsf{v} = \mathsf{f}$ to compute v:

$$\left[\begin{array}{cccc:c} \mathbb{M}^B & & & & \\ \mathbb{N}^B & \mathbb{M}^B & & & \\ & \ddots & \ddots & & \\ & & \mathbb{N}^B & \mathbb{M}^B & \\ \hdashline & & & & \mathbb{M}^A \end{array}\right] \left[\begin{array}{c} \mathbb{V}_1^B \\ \mathbb{V}_2^B \\ \vdots \\ \mathbb{V}_m^B \\ \hdashline \mathbb{V}_m^A \end{array}\right] = \left[\begin{array}{c} \mathbb{F}_1^B - \mathbb{N}^B\mathbb{U}_0^B - \mathbb{P}_1^B - \mathbb{C}^B\mathbf{S}_1 \\ \mathbb{F}_2^B - \mathbb{P}_2^B - \mathbb{C}^B\mathbf{S}_2 \\ \vdots \\ \mathbb{F}_m^B - \mathbb{P}_m^B - \mathbb{C}^B\mathbf{S}_{\mathrm{m}} \\ \hdashline \mathbb{F}_m^A - \mathbb{P}_m^A - \mathbb{N}^A\mathbb{U}_0^A \end{array}\right] \tag{3.28}$$

3.2.2 Computing Y

Use $\mathsf{M}\mathsf{Y} = \mathsf{C}$ to compute Y.

$$\left[\begin{array}{cccc:c} \mathbb{M}^B & & & & \\ \mathbb{N}^B & \mathbb{M}^B & & & \\ & \ddots & \ddots & & \\ & & \mathbb{N}^B & \mathbb{M}^B & \\ \hdashline & & & & \mathbb{M}^A \end{array}\right] \left[\begin{array}{c} \mathbb{Y}_1^B \\ \mathbb{Y}_2^B \\ \vdots \\ \mathbb{Y}_m^B \\ \hdashline \mathbb{Y}_m^A \end{array}\right] = \left[\begin{array}{c} \frac{1}{m}\mathbb{C}^B \\ \frac{2}{m}\mathbb{C}^B \\ \vdots \\ \mathbb{C}^B \\ \hdashline \mathbb{C}^A \end{array}\right] \tag{3.29}$$

3.2.3 Computing $\boldsymbol{\lambda}_{n+1}$

Use $(\mathsf{BY})\boldsymbol{\lambda}_{n+1} = \mathsf{BV}$ to compute $\boldsymbol{\lambda}_{n+1}$.

$$(\boldsymbol{C}^A\dot{\boldsymbol{Y}}^A_m + \boldsymbol{C}^B\dot{\boldsymbol{Y}}^B_m)\boldsymbol{\lambda}_m = \boldsymbol{C}^A\dot{\boldsymbol{V}}^A_m + \boldsymbol{C}^B\dot{\boldsymbol{V}}^B_m \tag{3.30}$$

3.2.4 Computing w

Use $\mathsf{w} = \text{-}\mathsf{Y}\boldsymbol{\lambda}_{n+1}$ to compute w.

$$\begin{bmatrix} \mathbb{W}^B_1 \\ \mathbb{W}^B_2 \\ \vdots \\ \mathbb{W}^B_m \\ \hdashline \mathbb{W}^A_m \end{bmatrix} = - \begin{bmatrix} \mathbb{Y}^A_1 \\ \mathbb{Y}^A_2 \\ \vdots \\ \mathbb{Y}^A_m \\ \hdashline \mathbb{Y}^B_m \end{bmatrix} \boldsymbol{\lambda}_m \tag{3.31}$$

3.2.5 Computing u

From sections - 3.2.1 and 3.2.4, $\mathsf{u} = \mathsf{v} + \mathsf{w}$ is computed. When using smeared crack model, damage in the domain is updated after the computation of u.

3.3 NUMERICAL EXAMPLE

In this section, we solve the same problem discussed in Section 2.4 using MTS-PH method and evaluate the computational gain achieved over the uniform time-step (UTS) method. We choose "Refined mesh-2" (see Figure 2.12) as the discretization for the sample. The 5 mm x 10 mm region around the notch tip is specified to be subdomain B and is always solved using a timestep of 5×10^{-9}, i.e. $\Delta t_B = 5 \times 10^{-9}$. The rest of the domain, i.e. subdomain A is either solved at 5×10^{-9}, 1×10^{-8}, 2.5×10^{-8} or 5×10^{-8} with timestep ratios (mAB) 1, 2, 5 and 10, respectively. The final state at 100 μsec of the beam for these four MTS simulations is shown in Figure 3.1. The corresponding crack-tip time-histories are plot in Figure 3.2 and Figure 3.3 shows the maximum LIRE in Damage.

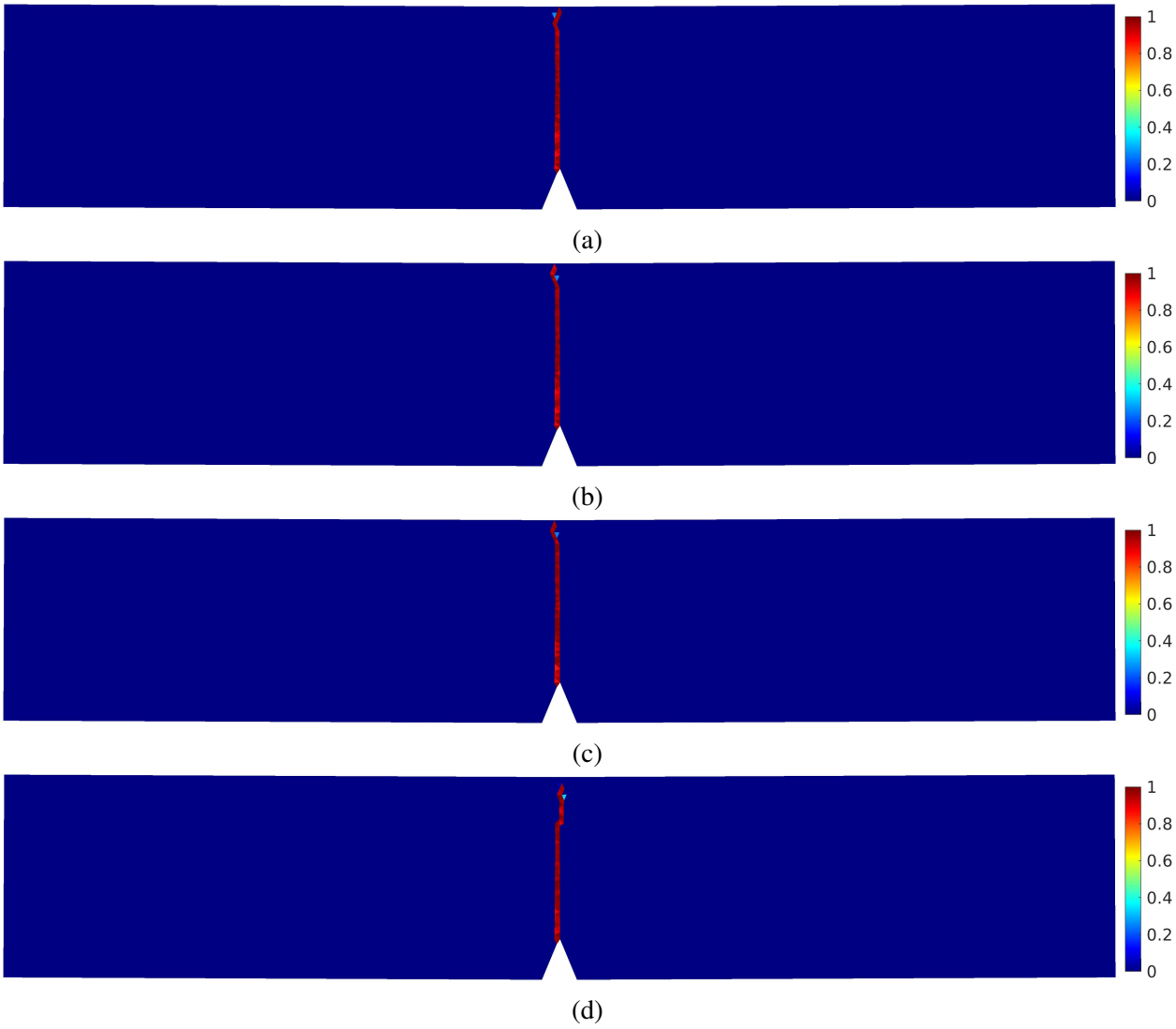


Figure 3.1. Refined mesh 2 at 100 μsec for mAB = 1, 2, 5 & 10 respectively

3.3.1 MTS Computational Runtime Cost Model

Similar to Section 2.4.3, we develop a runtime cost model for MTS as well. The initialization runtime cost for MTS depends only the total number of degrees of freedom and is similar to UTS cost model (Table 2.1 and Figure 2.17). Hence, we focus on the timeloop runtime in MTS which depends on a number of factors listed below:

1. Degrees of Freedom in subdomain A (DOFs-A)

2. Degrees of Freedom in subdomain B (DOFs-B)

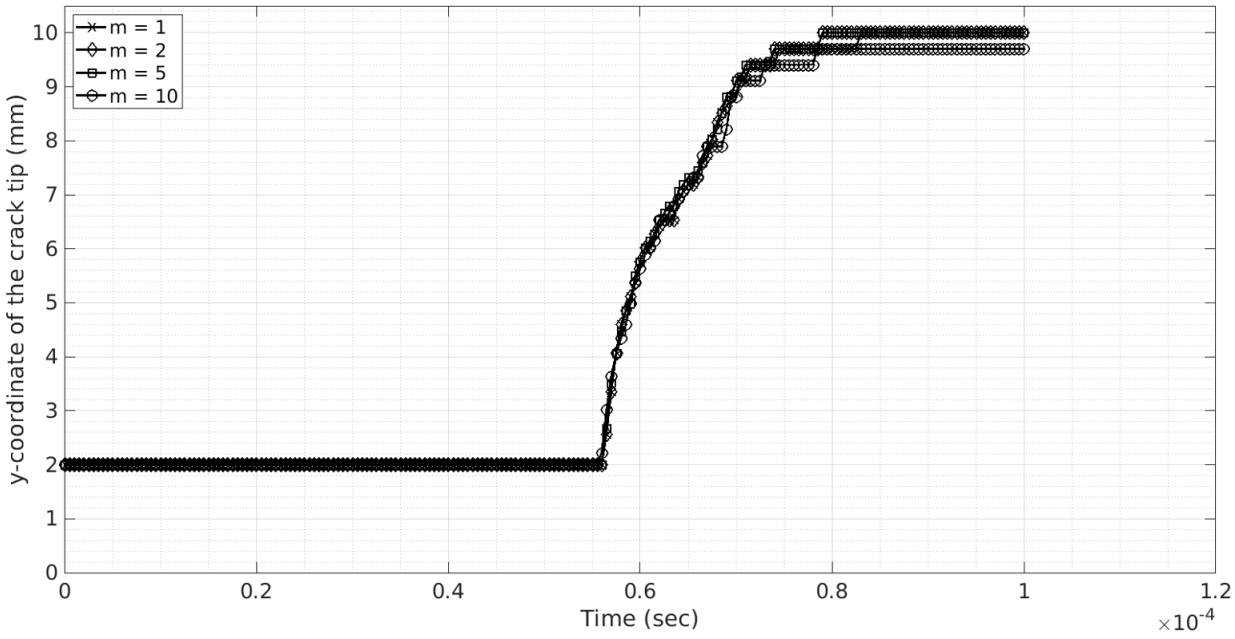


Figure 3.2. Crack-tip time-history in refined mesh-2 for mAB = 1, 2, 5 & 10

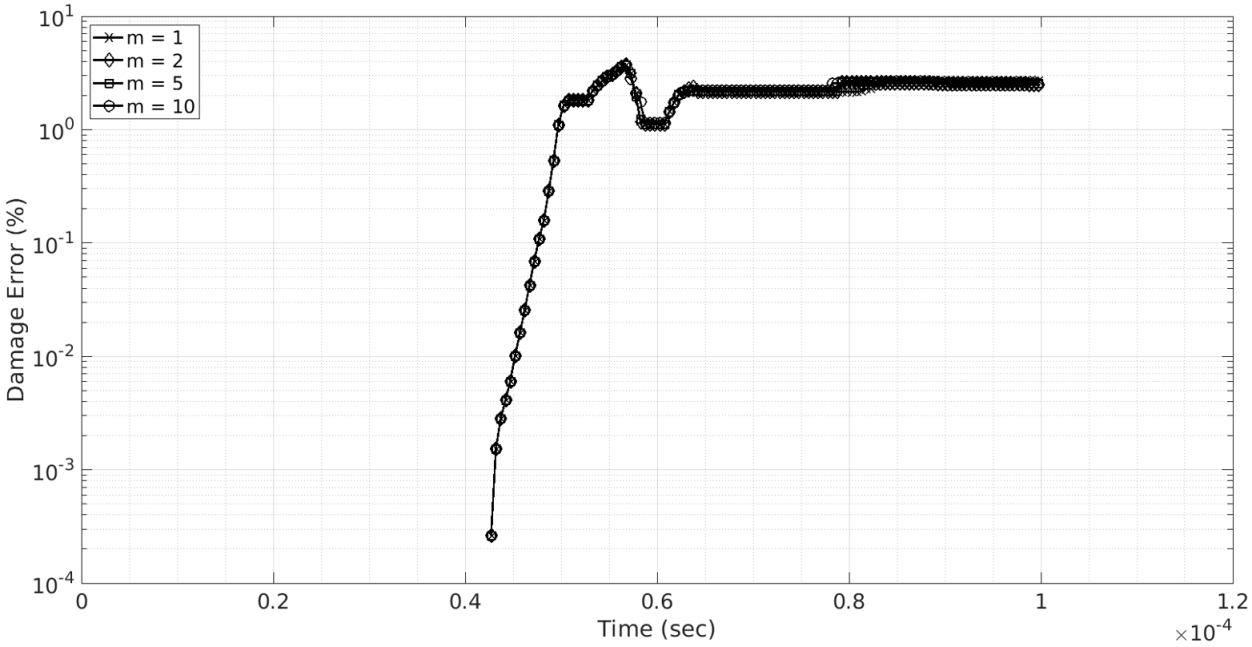


Figure 3.3. Maximum Damage LIRE in refined mesh-2 for mAB = 1, 2, 5 & 10

3. Degrees of Freedom along the interface (iDOFs)

4. Time-step ratio (*mAB*)

Note that the total degrees of freedom is the sum of DOFs-A and DOFs-B minus the iDOFs. The timeloop runtime can be further split into five parts - Predictor-A, Predictor-B, Interface, Corrector-A and Corrector-B. This segregation allows us to determine the effects of the individual factors on the timeloop runtime, see Figures 3.4, 3.5, 3.6, 3.7 and 3.8. It is evident from Figures 3.5 and 3.8 that the timestep ratio (*mAB*) has no effect on the runtimes for subdomain-B. The computational gain when using MTS is due to the use of coarser time-steps in subdomain-A. One of the primary reasons to develop the MTS cost model is to determine the conditions when it is beneficial to use MTS over UTS. The total timeloop runtime per Δt_B can be estimated using the equation

$$\begin{aligned} &\text{mAB} \times (3.316 \times 10^{-8} \times \text{Dofs-A}) + (3.534 \times 10^{-7} \times \text{Dofs-B}) \\ &+\text{mAB} \times (4.628 \times 10^{-7} \times \text{idofs}) + \text{mAB} \times (1.206 \times 10^{-8} \times \text{Dofs-A}) \\ &+ (9.749 \times 10^{-7} \times \text{Dofs-B}) \end{aligned} \tag{3.32}$$

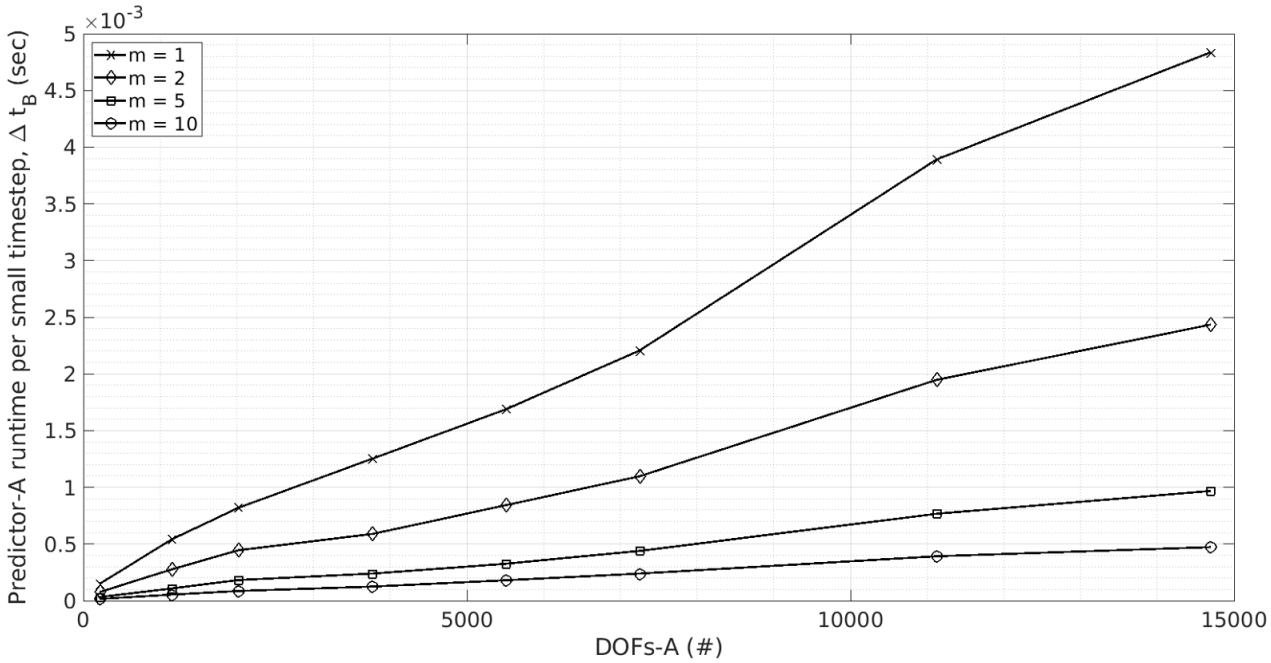


Figure 3.4. MTS Cost Model: Predictor-A Runtime per small timestep, Δt_B vs DOFs

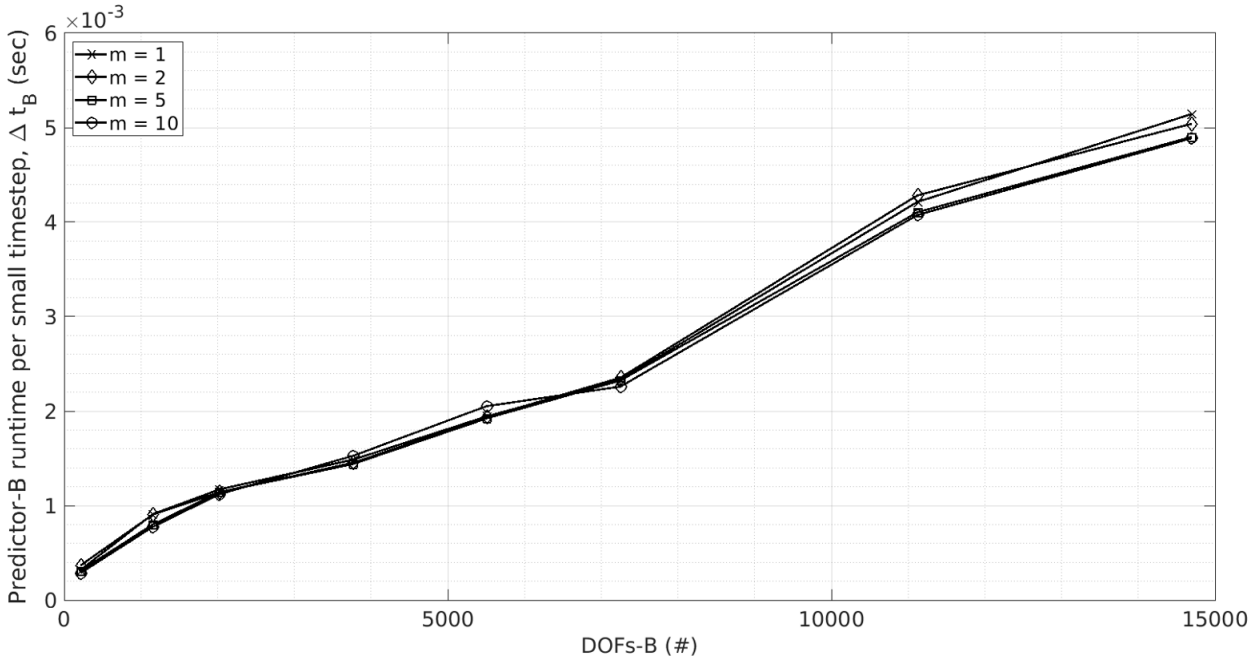


Figure 3.5. MTS Cost Model: Predictor-B Runtime per small timestep, Δt_B vs DOFs

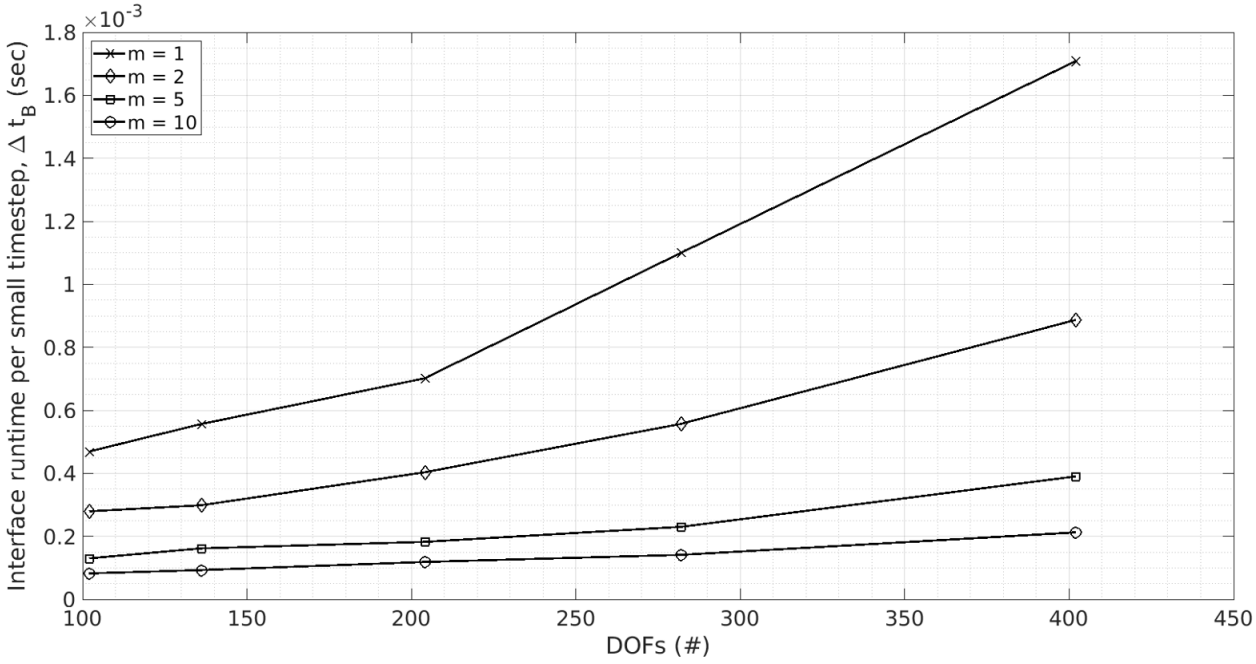


Figure 3.6. MTS Cost Model: Interface Runtime per small timestep, Δt_B vs DOFs

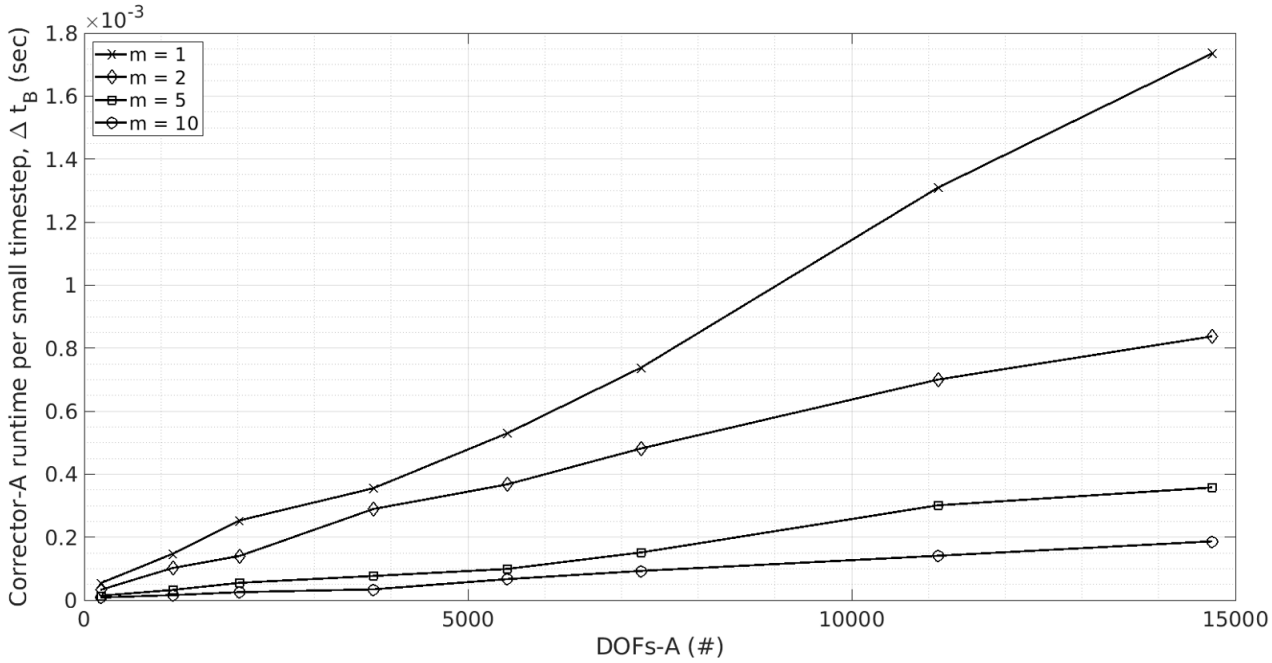


Figure 3.7. MTS Cost Model: Corrector-A Runtime per small timestep, Δt_B vs DOFs

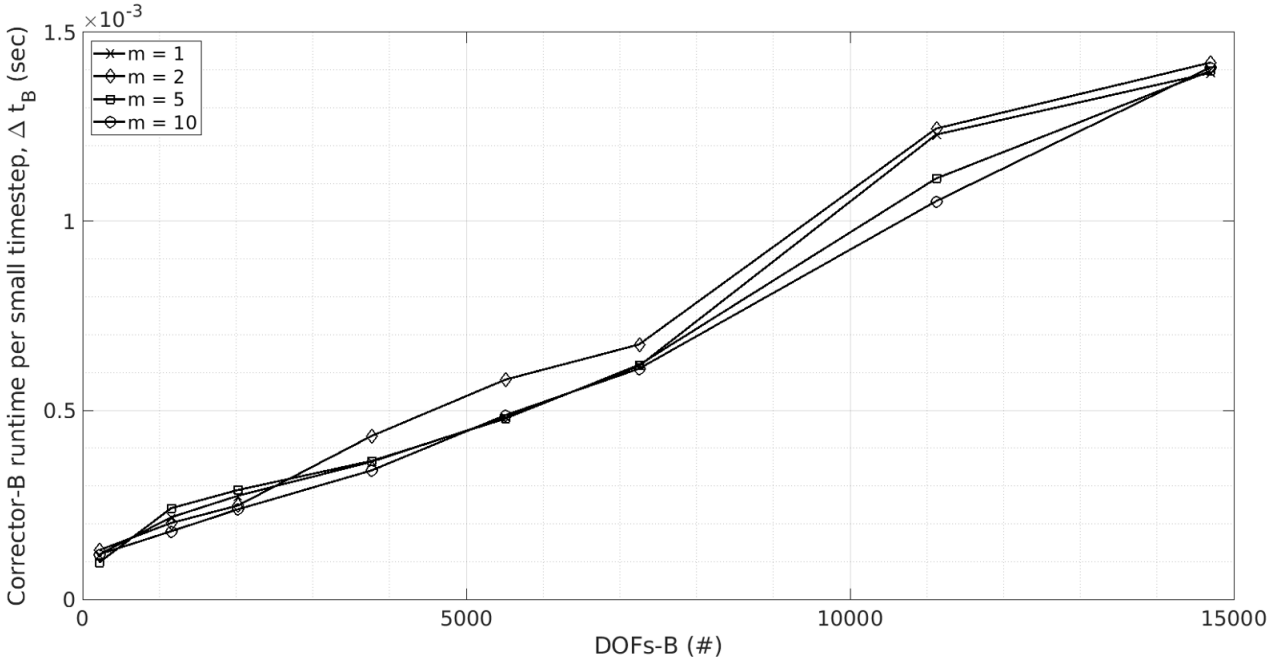


Figure 3.8. MTS Cost Model: Corrector-B Runtime per small timestep, Δt_B vs DOFs

3.3.2 Computational Runtime

This section outlines the runtime details associated with using the Refined mesh-2 discretization for different timestep ratios (*mAB*) in the notched three-point bending problem. The sub-

domains decomposition is kept constant throughout the simulation, i.e. the fine discretization around the notch tip is specified as the subdomain-B while the rest of the domain is specified as subdomain-A. The values of Dofs-A, Dofs-B and idofs for this discretization are 266, 2272 and 108 respectively. Employing a small time-step (Δt_B) of 5×10^{-9} sec, the simulation is conducted for 100×10^{-6} sec. In Table 3.1, we compare the actual runtimes against the runtimes predicted by the MTS cost model. We take the UTS runtime for Refined Mesh-2 (4,073 sec from Table 2.2) as the baseline.

Table 3.1. MTS Results and Predictions from MTS Cost Model

mAB (#)	Estimated Predictor-A Runtime (sec)	Estimated Predictor-B Runtime (sec)	Estimated Interface Runtime (sec)	Estimated Corrector-A Runtime (sec)	Estimated Corrector-B Runtime (sec)	Estimated Runtime per Timestep (sec)
UTS	N/A	N/A	N/A	N/A	N/A	8.31×10^{-4}
1	9.10×10^{-5}	8.17×10^{-4}	5.02×10^{-4}	3.57×10^{-5}	2.23×10^{-4}	1.67×10^{-3}
2	4.41×10^{-5}	8.13×10^{-4}	2.30×10^{-4}	1.63×10^{-5}	2.32×10^{-4}	1.34×10^{-3}
5	1.75×10^{-5}	7.90×10^{-4}	1.02×10^{-4}	6.42×10^{-6}	2.17×10^{-4}	1.13×10^{-3}
10	8.86×10^{-6}	7.90×10^{-4}	5.91×10^{-5}	3.37×10^{-6}	2.14×10^{-4}	1.08×10^{-3}

Actual Predictor-A Runtime (sec)	Actual Predictor-B Runtime (sec)	Actual Interface Runtime (sec)	Actual Corrector-A Runtime (sec)	Actual Corrector-B Runtime (sec)	Actual Runtime per Timestep (sec)
N/A	N/A	N/A	N/A	N/A	9.60×10^{-4}
3.13×10^{-4}	1.03×10^{-3}	5.37×10^{-4}	7.15×10^{-5}	2.23×10^{-4}	2.17×10^{-3}
1.53×10^{-4}	9.72×10^{-4}	2.82×10^{-4}	3.63×10^{-5}	2.19×10^{-4}	1.66×10^{-3}
6.00×10^{-5}	9.31×10^{-4}	1.15×10^{-4}	1.58×10^{-5}	2.22×10^{-4}	1.34×10^{-3}
3.64×10^{-5}	1.04×10^{-3}	6.47×10^{-5}	9.18×10^{-6}	2.24×10^{-4}	1.37×10^{-3}

In Table 3.1, “+/-" runtime difference denote increase and decrease respectively, with respect to UTS. It is observed that $mAB = 1$ (STS) has worse performance than UTS due to the additional computational cost of coupling required at every time-step, while MTS with $mAB = 10$ has the best performance. We observe a 12.4% reduction in runtime from using MTS $mAB = 10$ versus UTS while maintaining similar level of maximum damage error.

Relative Difference in estimation (%)	Total computational runtime (sec)	Runtime Difference from UTS (%)	Max. Damage Error (%)
15.4	4,073	NA	3.73
29.9	4,235	+4.0	3.73
23.9	3,860	-5.2	3.77
18.6	3,578	-12.2	3.82
26.9	3,568	-12.4	3.88

The computational gain through MTS in this crack propagation problem using Refined mesh-2 as the discretization is limited because of the relatively large size of subdomain B in comparison to subdomain A. If we can limit the fine discretization region to a small region around the crack tip instead of having fine discretization throughout the width of the specimen, we can significantly reduce the number of Dofs in subdomain B, which can in-turn reduce the total computational cost. This would require adaptation of the region around the crack tip as the crack propagates. An adaptive multi-time-step (AMTS) method that enables the evolution of the region of interest during the simulation is discussed in Chapter-4.

4. ADAPTIVE MULTI-TIME-STEP METHOD

Fine-scale dynamics in problems involving fracture are usually are focussed around the crack tip. The rest of the domain away from the crack does not typically require a fine spatial discretization allowing one to reduce the total number of Dofs present in the system. However, this reduction in Dofs requires the focused fine-mesh region to be updated dynamically as the crack propagates through the material. In this chapter, we develop a method which updates the fine-mesh region of interest around the crack tip adaptively.

4.1 ADAPTIVITY SCHEME

Three key steps are needed to adaptively track the region of interest in a dynamically evolving problem:

- Criterion to define the region of interest as it evolves during runtime
- Mesh refinement and coarsening strategy
- Mapping of data from one mesh discretization to another

Since undertaking any of these steps during a simulation can add a significant computational overhead, their implementation in a computationally efficient manner is key to realizing any gains from adaptivity.

4.1.1 Criteria to define regions of interest

In this study, two different criteria are used for two different types of problems. One criterion is based on identifying high spatial gradients in stress in the problem domain and can be used for tracking the wave-front in wave-propagation problems. Usually, one needs a fine discretization at the wave-front to resolve the wave form while a coarse discretization suffices elsewhere. Another criterion is based on locating the crack-tip in problems involving fracture and is used to adpatively refine the region around the crack tip as it moves through the problem domain.

For the stress-gradient criterion, one can compute approximate spatial gradients by taking the difference of a stress-measure between neighboring elements and dividing it by the distance be-

tween their centroids. We adopt a simple measure of stress in an element as the trace of the stress tensor i.e. volumetric stress and we define neighboring elements as those which share at least one node. Once stress gradients are computed for all neighboring element-pairs in the mesh, we create a distribution of this data and use a threshold value to determine which element-pairs have high gradients compared to the rest of the mesh. Elements associated with a high stress gradients are assumed to be experiencing fine-scale dynamics and these elements, along with their first and second neighbors, are identified as being in the region of interest.

For crack-propagation problems, we use damage, as defined in Equation 2.19, and its rate of increase with time as an indicator of the location of the crack-tip. Elements that are associated with a high rate of increase of damage are assumed to be at or near the crack-tip and are therefore included in the region of interest, along with their first and second neighbors.

4.1.2 Mesh refinement and coarsening

Once regions of interest are identified, the mesh is decomposed into subdomains A and B accordingly, which is updated subsequently as the regions of interest evolve. The number of times we need to update the subdomains depends on the rate of deformation or the rate of crack propagation. Each update creates a new discretization - a new mesh state - where some elements may be refined while others are merged together into coarser elements. Updating the mesh can be done using techniques similar to adaptive mesh refinement.

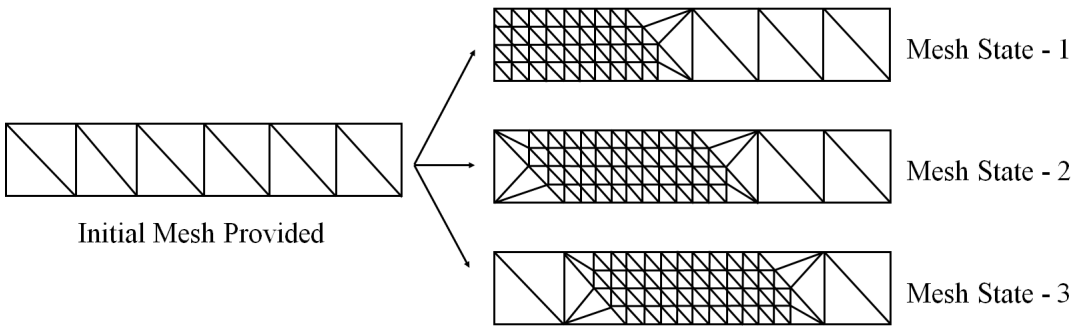


Figure 4.1. Example of evolving mesh states generated from an initial mesh

The in-house code developed in this study allows the user to provide an initial mesh which is adaptively refined by creating finer elements in the regions of interest in a nested manner as shown

in Figure 4.1. To create the different mesh states shown in Figure 4.1, elements in the initial mesh are divided into smaller elements. Elements in the initial mesh are referred to as 'Parent (Level-0)' elements and subsequent smaller elements created from them in different mesh states are referred to as 'Children (Level-1)' and 'Grandchildren (Level-2)' elements. In the mesh states shown in Figure 4.1, parent elements undergo two levels of refinement, first into children elements and then into grandchildren elements. A sample two-level refinement of a parent element is shown in Figure 4.2.

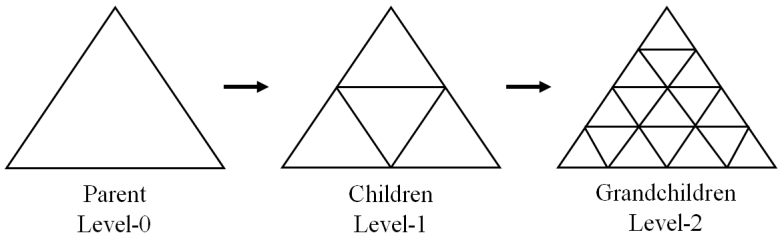


Figure 4.2. Sample Level-0, Level-1, Level-2 refinements

We adopt the following rules for refinement of parent elements to ensure that element quality is maintained upon refinement. Elements are first classified into three types based on their shape:

Type 1: Elements for which all three edges satisfy the condition: $\left(\frac{l_{\mathrm{i}}}{l_{max}}\right) > 0.6$

Type 2: Elements for which two edges satisfy the condition: $\left(\frac{l_{\mathrm{i}}}{l_{max}}\right) > 0.6$

Type 3: Elements for which only one edge satisfies the condition: $\left(\frac{l_{\mathrm{i}}}{l_{max}}\right) > 0.6$

where l_{i} $(\mathrm{i} = 1,2,3)$ is length of edge i and l_{max} is the length of the longest edge. Depending upon its type, a parent element may lead to different number of children elements after one level of refinement. As shown Figures 4.3, 4.4 and 4.5, a Type-1 parent element has 10 possible children elements, a Type-2 has 8 possible children elements and Type-3 has 6 possible children elements . Once all the possible children elements are generated, subsequent grandchildren elements are created from each of the children elements using the same procedure. Thus, a Type-1 element can have a maximum of 100 possible grandchildren elements after two levels of refinement, if each of its 10 children elements are also of Type-1.

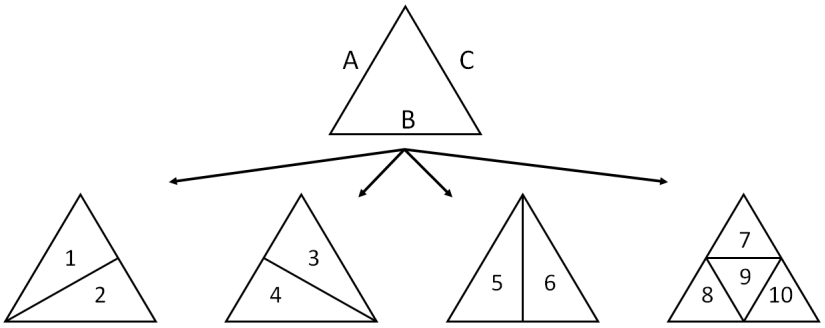


Figure 4.3. Possible children elements of Type-1 parent element

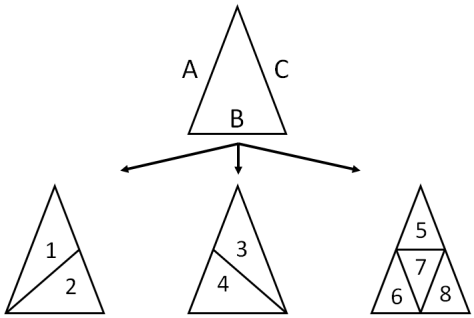


Figure 4.4. Possible children elements of Type-2 parent element

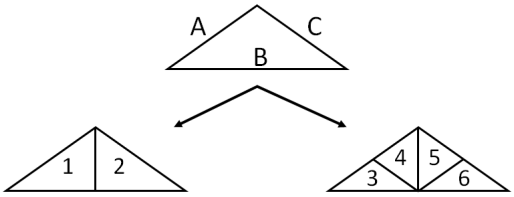


Figure 4.5. Possible children elements of Type-3 parent element

Using the rules defined above, the mesh is refined in the regions of interest (ROI) as determined by the criteria from section 4.1.1. Elements in the level-0 mesh with 2 or more nodes within the ROI are divided into their respective level-1 elements. In Figure 4.6 , the first 5 elements on the left are in the ROI. However, this leads to the creation of hanging nodes as shown in Figure 4.7, where element 'X' contains the hanging node. Depending upon the type of element 'X' is and the edge containing the hanging node, element 'X' is divided into either 2 or 4 level-1 elements.

Table 4.1. Hanging node element refinement information

Element Type	Hanging Node Edge	Level-1 Elements
1	A	3,4
1	B	5,6
1	C	1,2
2	A	3,4
2	B	5,6,7,8
2	C	1,2
3	A	3,4,5,6
3	B	1,2
3	C	3,4,5,6

Table-4.1 lists the possible level-1 elements obtained from a level-0 element shown in Figures 4.3, 4.4 and 4.5. In Figure 4.7, element 'X' is a type-1 element. Hence, it is divided into two level-1 elements and the final level-1 mesh is shown in Figure 4.8. The process is repeated to refine level-1 mesh into level-2 mesh (that leads to Mesh State-1) as shown in Figure 4.9. Once Level-2 mesh is created, smaller elements are categorized as subdomain-B while the rest of the system is classified as subdomain-A for the MTS method. As the ROI evolves, the process to create mesh states 2 and 3 is identical to that of mesh state-1, as discussed above.

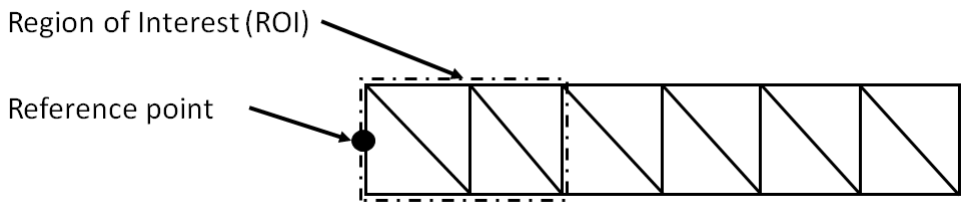


Figure 4.6. Reference point and ROI in Level-0 mesh for Mesh State-1

Note on Implementation: Conducting mesh refinement to compute mesh states during the simulation can be very time consuming and has the potential to negate any computational gains that one may hope to achieve with adaptivity. To facilitate fast transitions from one mesh state to another when the ROI evolves, we pre-compute the possible mesh states before commencing time-stepping. This is done by creating all possible level-1 and level-2 children of parent elements in the initial mesh that are likely to be refined during the simulation. Quantities associated with all such children

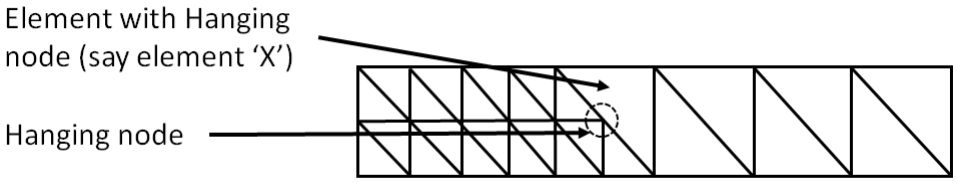


Figure 4.7. Level-1 mesh with Hanging nodes for Mesh State-1

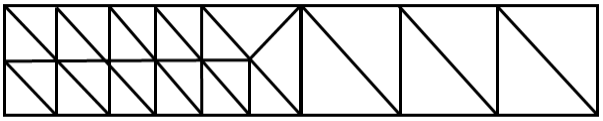


Figure 4.8. Level-1 mesh for Mesh State-1

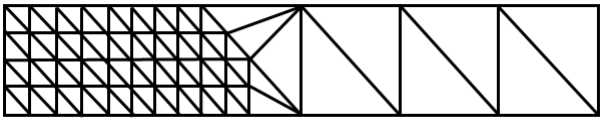


Figure 4.9. Level-2 mesh or Mesh State-1

and grandchildren elements - such as derivatives of shape functions, mass and stiffness matrices - are stored in a hierarchical manner linked to their parent element. During runtime, elements are simply switched on and off as refinement and coarsening occurs. Subdomain matrices are also easily modified using this strategy because when an element is turned on or off, its contribution to the subdomain mass and stiffness matrix is added to or subtracted from the previous subdomain matrix. Even though saving all this information requires a lot more storage than a conventional mesh, it helps reduce the overhead associated with implementation of adaptivity and allows rapid transitions between mesh states. For moderate-size problems, consisting of 10,000-20,000 Dofs, this implementation can drastically reduce the runtime at the expense of some additional storage. However, for large problems involving millions of Dofs, this implementation would lead to non-trivial storage issues.

4.1.3 Data transfer between two successive mesh states

The final piece of the puzzle to implementing adaptivity is transferring the state of the problem from one mesh-state to another. Data that need to be transferred between successive mesh states can be classified into elemental and nodal quantities. Elemental quantities include the element stress, strain and crack properties and nodal quantities include displacement, velocity and acceleration.

Element data to be transferred from the present mesh-state to the next mesh-state may be classified into one of the following:

1. No change: For unchanged elements in between the two mesh states, the values are directly copied from the present to future mesh state.

2. Refinement: When the elements undergo refinement, the newly created elements in future mesh state are assigned the same values as that of the original element in the present mesh state.

3. Coarsening: For fine elements coarsened from present to future mesh state, area weights of the present elements combining to form the future element are used to compute the values.

For example, Figure 4.10 shows the elements undergoing refinement and coarsening in mesh state-1 for data transfer between mesh state-1 and mesh state-2 in Figure 4.1. All other elements are unchanged between the two mesh states.

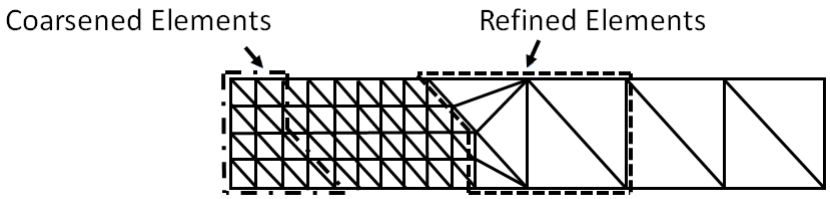


Figure 4.10. Regions for Elemental Data Transfer in mesh state-1

Nodes in the present mesh-state and the next mesh-state can be classified into one of the following and the values of the nodal variables determined as noted:

1. Nodes in both mesh states and not a part of newly refined or coarsened elements: Values are directly copied from the present to next mesh-state.

2. Nodes created as part of newly formed refined elements in the next mesh state: Values are interpolated using the shape functions of elements in the present mesh-state.

3. Nodes in both mesh states and part of newly formed coarse elements in the next mesh-state: Values are computed using a minimization of the error between the present and the next mesh-state as described next.

Figure 4.11 shows the nodes which are a part of newly formed coarse elements in mesh state-2 for data transfer between mesh state-1 and mesh state-2 in Figure 4.1.

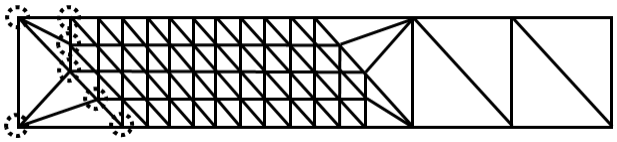


Figure 4.11. Nodes part of newly formed coarse elements in mesh state-2

The following data transfer strategy is employed to compute the future mesh state values. We define a measure of error, ε, between the fine vs coarse element values as:

$$\varepsilon = \iint_B \left(f^F - f^C\right)^2 dB \tag{4.1}$$

$$= \sum_{\mathrm{j}=1}^{n} \iint_{B^{\mathrm{j}}} \left(f^F - f^C\right)^2 dB^{\mathrm{j}} \tag{4.2}$$

$$= \sum_{\mathrm{j}=1}^{n} \sum_{k=1}^{m} \iint_{\Omega^k} \left(\sum_{\alpha=1}^{3} N_\alpha^k f_\alpha^{kF} - \sum_{\beta=1}^{3} S_\beta^{\mathrm{j}} f_\beta^{\mathrm{j}C} \right)^2 d\Omega^k \tag{4.3}$$

where B is the combined domain of all the coarse elements under consideration, B^{j} is the area under 'j' coarse element, n is the total number of coarse elements in the domain, Ω^k is the area under 'k' fine element, m is the total number of fine elements in the 'j' coarse element, f^F and f^{kF} are the global and elemental values in the fine elements, f^C and $f^{\mathrm{j}C}$ are the global and elemental values in the coarse elements, N_α^k are the shape functions in the 'k' fine element and S_β^{j} are the shape functions in the 'j' coarse element.

Minimizing the error ε with respect to the coarse element values f_γ^C, we get:

$$\frac{d\varepsilon}{df_\gamma^C} = 0 \tag{4.4}$$

$$\Rightarrow \sum_{\mathrm{j}=1}^{n}\sum_{k=1}^{m}\iint_{\Omega^k} 2\left(\sum_{\alpha=1}^{3} N_\alpha^k f_\alpha^{kF} - \sum_{\beta=1}^{3} S_\beta^{\mathrm{j}} f_\beta^{\mathrm{j}C}\right) S_\gamma^{\mathrm{j}} d\Omega^k = 0 \tag{4.5}$$

Using 3-point Gauss quadrature to compute the above integral, we get:

$$\sum_{\mathrm{j}=1}^{n}\sum_{k=1}^{m}\sum_{gp=1}^{3}\frac{1}{3}\left(\sum_{\alpha=1}^{3} \bar{N}_\alpha^k f_\alpha^{kF} - \sum_{\beta=1}^{3} \bar{S}_\beta^{\mathrm{j}^k} f_\beta^{\mathrm{j}C}\right) \bar{S}_\gamma^{\mathrm{j}^k}\,\Omega^k = 0 \tag{4.6}$$

where $\bar{N}_\alpha^k$ and $\bar{S}_\beta^{\mathrm{j}^k}$ denote the fine and coarse shape function values at the gauss points of the fine element 'k' respectively.

$$\sum_{\mathrm{j}=1}^{n}\sum_{k=1}^{m}\sum_{gp=1}^{3}\frac{1}{3}\left(\bar{S}_\gamma^{\mathrm{j}^k}\Omega^k\sum_{\alpha=1}^{3}\bar{N}_\alpha^k f_\alpha^{kF}\right) - \sum_{\mathrm{j}=1}^{n}\sum_{k=1}^{m}\sum_{gp=1}^{3}\frac{1}{3}\left(\bar{S}_\gamma^{\mathrm{j}^k}\Omega^k\sum_{\beta=1}^{3}\bar{S}_\beta^{\mathrm{j}^k} f_\beta^{\mathrm{j}C}\right) = 0 \tag{4.7}$$

Upon simplification, the above equation can be written in matrix form as:

$$\sum_{\mathrm{j}=1}^{n}\sum_{k=1}^{m}\sum_{gp=1}^{3}\frac{1}{3}\Omega^k \bar{\boldsymbol{S}}^{\mathrm{j}^k}\bar{\boldsymbol{S}}^{\mathrm{j}^{kT}}\boldsymbol{f}^{\mathrm{j}C} = \sum_{\mathrm{j}=1}^{n}\boldsymbol{b}^{\mathrm{j}} \tag{4.8}$$

where

$$\bar{\boldsymbol{S}}^{\mathrm{j}^k} = \begin{bmatrix} \bar{S}_1^{\mathrm{j}^k} \\ \bar{S}_2^{\mathrm{j}^k} \\ \bar{S}_3^{\mathrm{j}^k} \end{bmatrix} ; \boldsymbol{f}^{\mathrm{j}^c} = \begin{bmatrix} f_1^{\mathrm{j}C} \\ f_2^{\mathrm{j}C} \\ f_3^{\mathrm{j}C} \end{bmatrix} ; \bar{\boldsymbol{N}}^k = \begin{bmatrix} \bar{N}_1^k \\ \bar{N}_2^k \\ \bar{N}_3^k \end{bmatrix} ; \boldsymbol{f}^{kF} = \begin{bmatrix} f_1^{kF} \\ f_2^{kF} \\ f_3^{kF} \end{bmatrix} ;$$

$$\boldsymbol{b}^{\mathrm{j}} = \boldsymbol{P}\boldsymbol{Q}^{\mathrm{j}F} \tag{4.9}$$

$$\boldsymbol{P} = \begin{bmatrix} \frac{\Omega^1}{3}\sum_{gp=1}^{3} \bar{\boldsymbol{S}}^{\mathrm{j}^1}\bar{\boldsymbol{N}}^{1^T} & .... & \frac{\Omega^m}{3}\sum_{gp=1}^{3} \bar{\boldsymbol{S}}^{\mathrm{j}^m}\bar{\boldsymbol{N}}^{m^T} \end{bmatrix} ; \boldsymbol{Q}^{\mathrm{j}F} = \begin{bmatrix} \boldsymbol{f}^{1F} \\ . \\ . \\ \boldsymbol{f}^{mF} \end{bmatrix}$$

Equation-(4.8) is in the elemental form. The vectors and matrices can be assembled as:

$$\boldsymbol{L}\boldsymbol{f}^C = \boldsymbol{b}^G \tag{4.10}$$

where

$$\boldsymbol{L} = \mathop{\mathrm{A}}_{\mathrm{j}=1}^{n}\left(\sum_{k=1}^{m}\sum_{gp=1}^{3}\frac{1}{3}\Omega^k\bar{\boldsymbol{S}}^{\mathrm{j}^k}\bar{\boldsymbol{S}}^{\mathrm{j}^{kT}}\right), \boldsymbol{f}^C = \mathop{\mathrm{A}}_{\mathrm{j}=1}^{n}\boldsymbol{f}^{\mathrm{j}C}, \boldsymbol{b}^G = \mathop{\mathrm{A}}_{\mathrm{j}=1}^{n}\boldsymbol{b}^{\mathrm{j}} \tag{4.11}$$

Equation 4.10 can be solved for $\boldsymbol{f}^C$ to obtain the coarse mesh values from fine scale variables.

4.2 DATA TRANSFER VERIFICATION

To evaluate the accuracy of the data transfer strategy described in section 4.1.3, we consider a wave propagation problem in a rectangular plate that is 1 m wide and 8 m long, see Figure 4.12. One edge of the model is fixed and a uniform step-load of 1 N/m is applied throughout the simulation (0-8 sec). The plate is composed of a linear-elastic material (Young's Modulus E = 2 Pa, Poisson's ratio ν = 0.25, Density ρ = 2 kg/m^3) with 10% mass proportional Rayleigh damping.

The model is discretized using nine different meshes - six regular (as shown in Figure 4.13) and three adaptive meshes with predefined mesh-states (as shown in Figure 4.14, 4.15 and 4.16

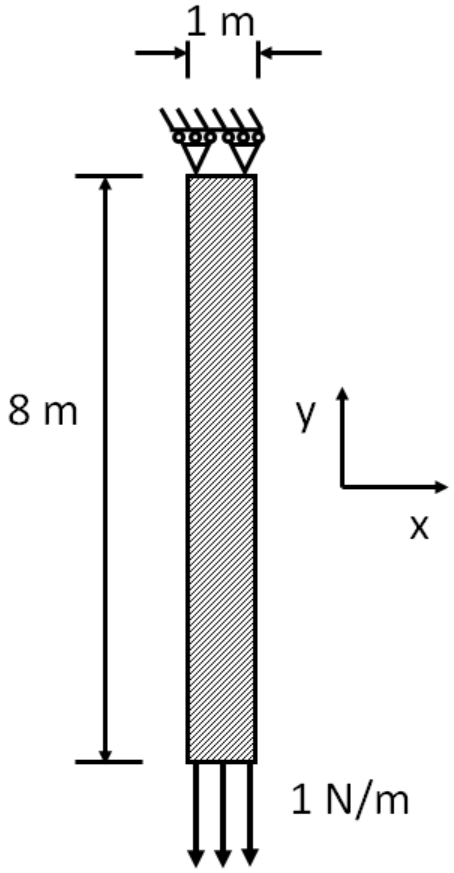


Figure 4.12. Wave propagation

respectively). For the adaptive meshes, the input mesh is provided by the user and is used only to create the multiple mesh-states - it is not used during the simulation. The simulation starts with the model in an initial discretization of mesh state-1. Adaptive meshes 1, 2 and 3 undergo 3, 4 and 5 mesh transitions respectively.

All nine discretizations of the model are run using central-difference explicit time integration scheme. The stable time-increment for the most refined mesh (16 elements along width) is 6.25×10^{-2} sec. Hence, all meshes are solved with a time-step of 4×10^{-2} sec and measures of error are computed. Accuracy of the data transfer is evaluated using the local instantaneous relative error (LIRE) in displacement (see section-2.4), which requires mapping of results of all the discretizations to a reference or pixel mesh (see section sec:pixelation). The reference or pixel mesh adopted for error computations is the 16 elem along width mesh in Figure 4.13. The datum and

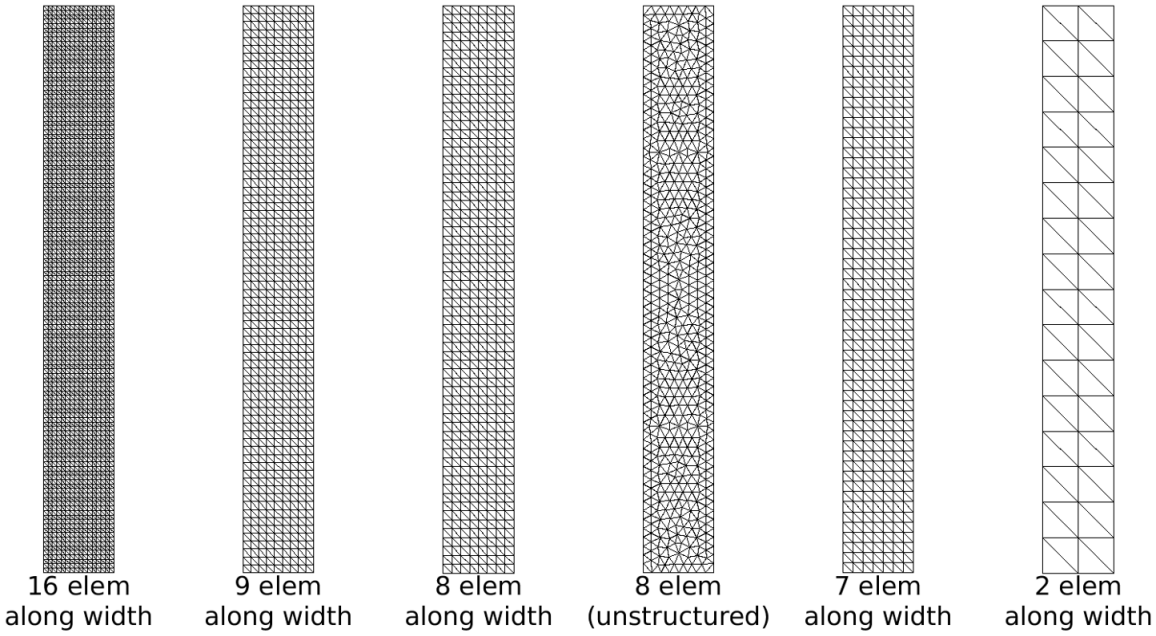


Figure 4.13. Wave propagation : Six regular meshes

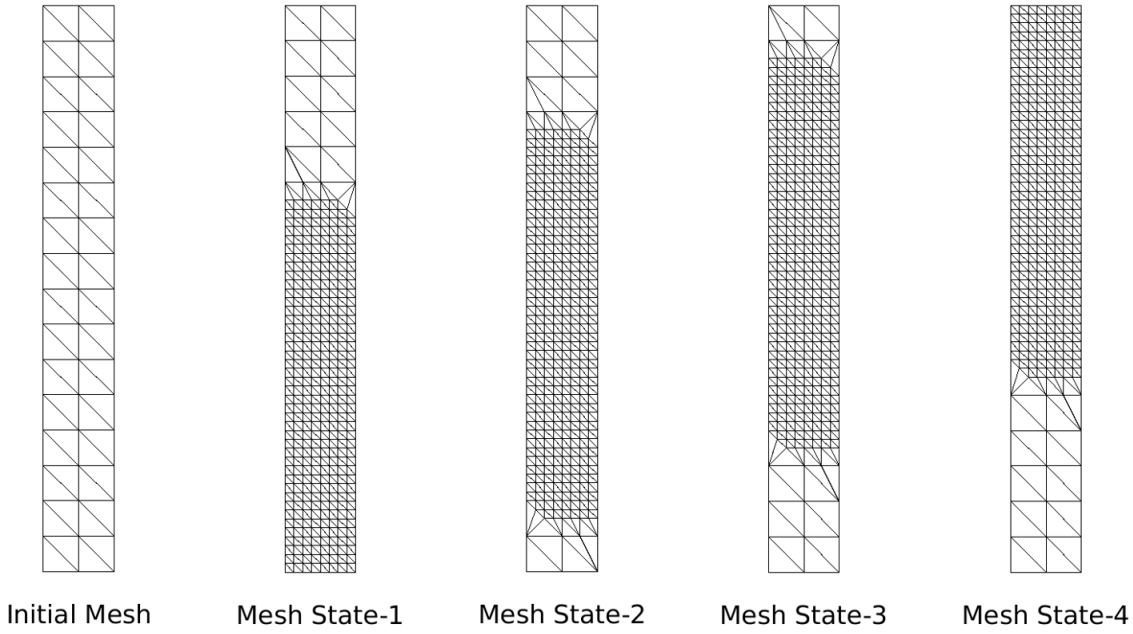


Figure 4.14. Wave propagation : Adaptive Mesh-1 (3 transitions)

subject solutions for the LIRE are the original reference mesh values and the mapped reference mesh values for different discretizations.

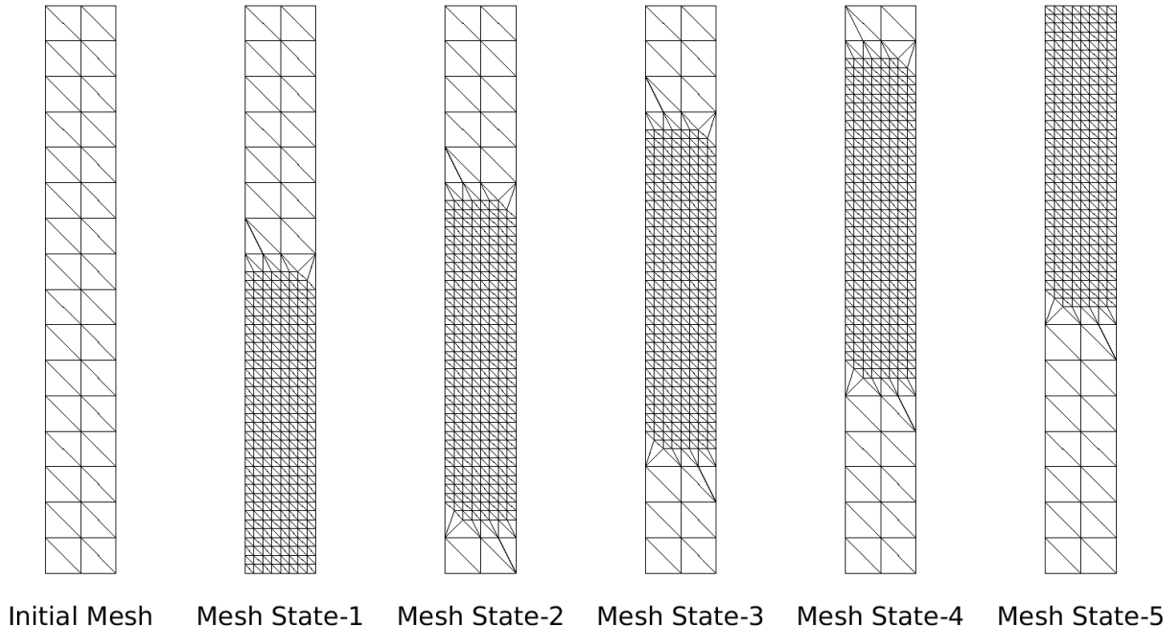


Figure 4.15. Wave propagation : Adaptive Mesh-2 (4 transitions)

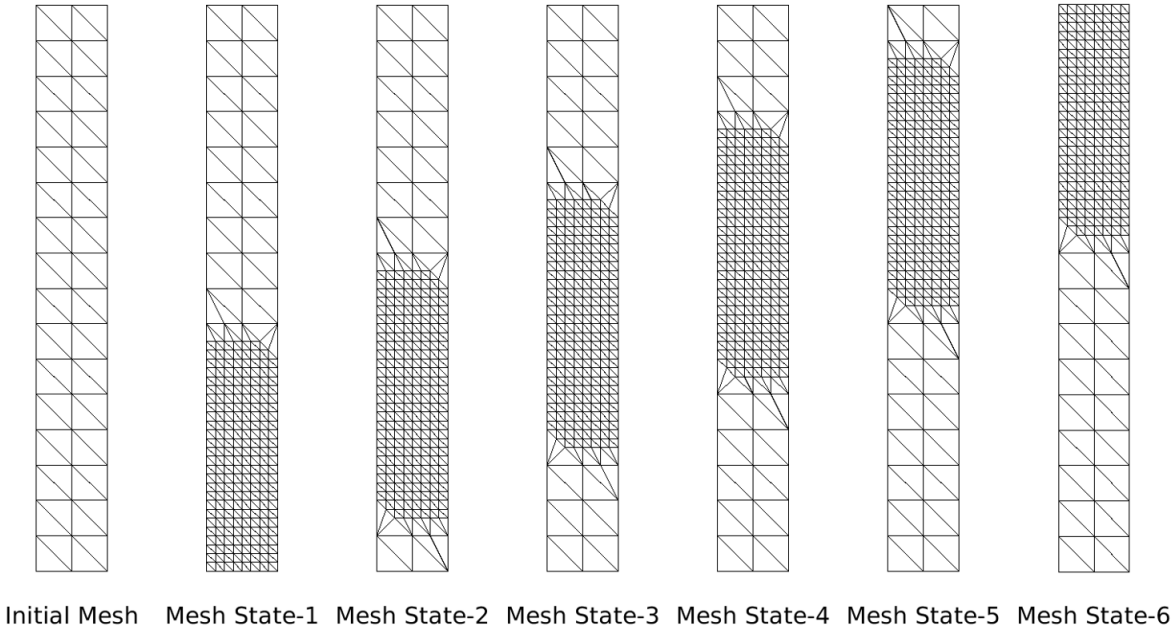


Figure 4.16. Wave propagation : Adaptive Mesh-3 (5 transitions)

LIRE in displacement for different subject meshes with respect to the datum mesh are shown in Figure 4.17. It is apparent from the plots that as the refinement of the meshes increases, the errors tend to decrease. An important thing to note is that the first transition from mesh state-1

to mesh state-2 occurs at 2 sec in adaptive mesh-3 occurs and the errors are comparable to the refined regular meshes up until about 2.5-2.6 sec. The errors increase after this point in time due to dispersion of the wave and presence of non-causal waves that travel in advance of the wave-front and end up outside the refinement region. It can be argued that since errors do not jump up immediately after the mesh transition and stay within the range of the refined and regular meshes, therefore the data transfer strategy is accurate. A similar trend is observed in Adaptive meshes 1 and 2 where the first transition occurs at 4 sec and 3 sec respectively.

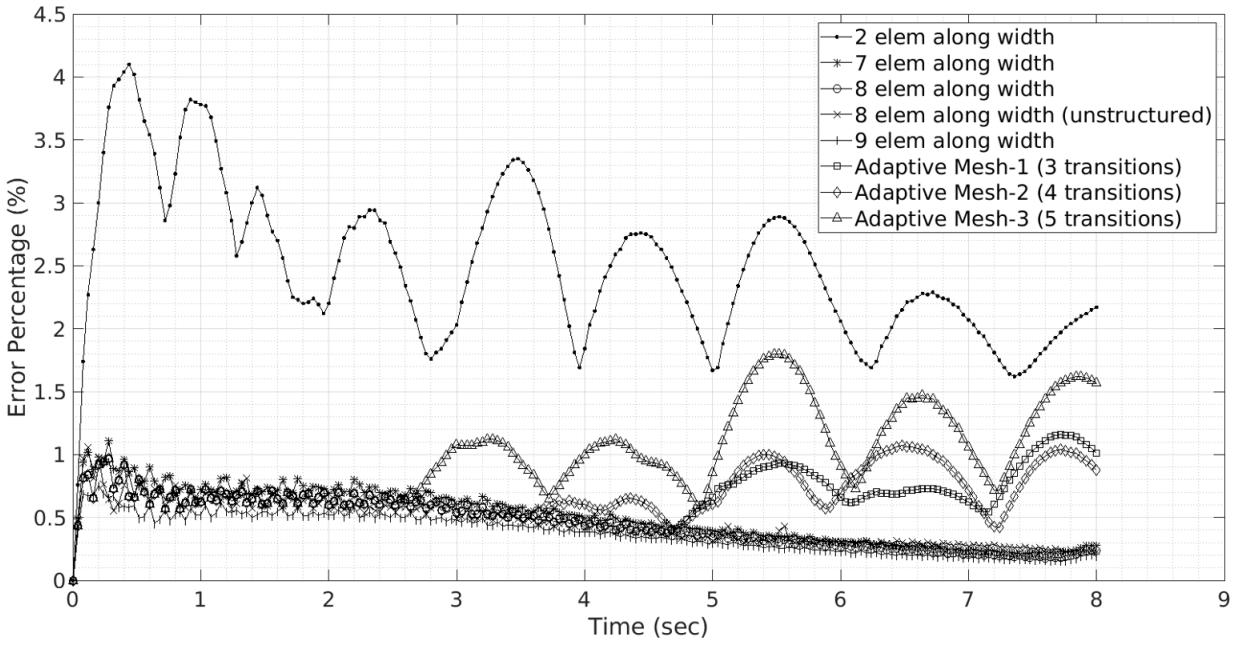


Figure 4.17. Displacement LIRE for different discretizations

4.3 NUMERICAL EXAMPLE

We consider the notched 3-point bending problem discussed previously in Section-2.4 and 3.3. The initial input mesh for this problem, shown in Figure 4.18, is only used to create the different adaptive mesh states and does not serve any other purpose during the simulation. Three different adaptive meshes created from this initial mesh are used to compare the performance of the adaptive uniform time-step method (AUTS) method and the adaptive multi-time-step (AMTS) method to

their original (non-adaptive) UTS and MTS counterparts. Key characteristics of these meshes are listed as follows:

1. Adaptive mesh-1 consists of a fine mesh region of 5 mm x 7 mm and undergoes 2 transitions to track the crack propagation - see Figure 4.19

2. Adaptive mesh-2 consists of a fine mesh region of 5 mm x 8 mm and undergoes 2 transitions to track the crack propagation - see Figure 4.20

3. Adaptive mesh-3 consists of a fine mesh region of 5 mm x 9 mm and undergoes 1 transitions to track the crack propagation - see Figure 4.21

All three adaptive meshes are made up of 0.250 mm elements in the fine mesh region.

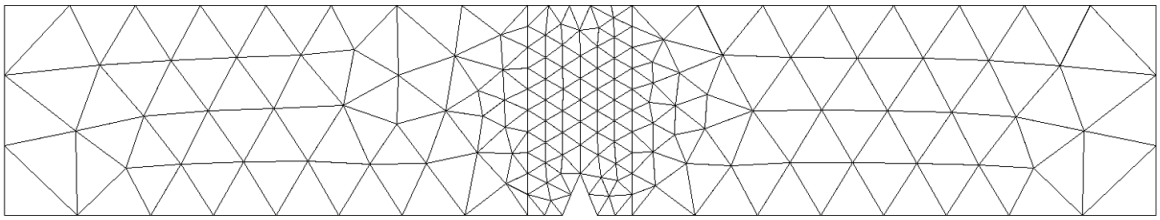


Figure 4.18. Input Mesh for Adaptive meshes

First, the three adaptive meshes are used with the AUTS method and maximum LIRE in damage is compared to that obtained from Refined meshes 1 and 2 run with the UTS method in Figure 4.22. The maximum damage error over the entire simulation for the refined meshes is in the range of 3.82-3.98% respectively, whereas maximum damage error in the adaptive mesh-1, 2 and 3 is around 3.88%, 3.87% and 4.47%, respectively. Since errors from the adaptive meshes are similar to those from the refined meshes, this validates the results from the adaptive meshes.

4.3.1 Reduction in runtimes due to adaptivity

Table-4.2 compares the runtimes for different adaptive meshes against refined mesh-2 when an uniform time-step (UTS) is used throughout the domain. The total DOFs for the adaptive meshes are shown as a range since each adaptive mesh is made up of various mesh-states. The mesh-states runtime is the time spent in pre-computing the various mesh states prior to the time-loop.

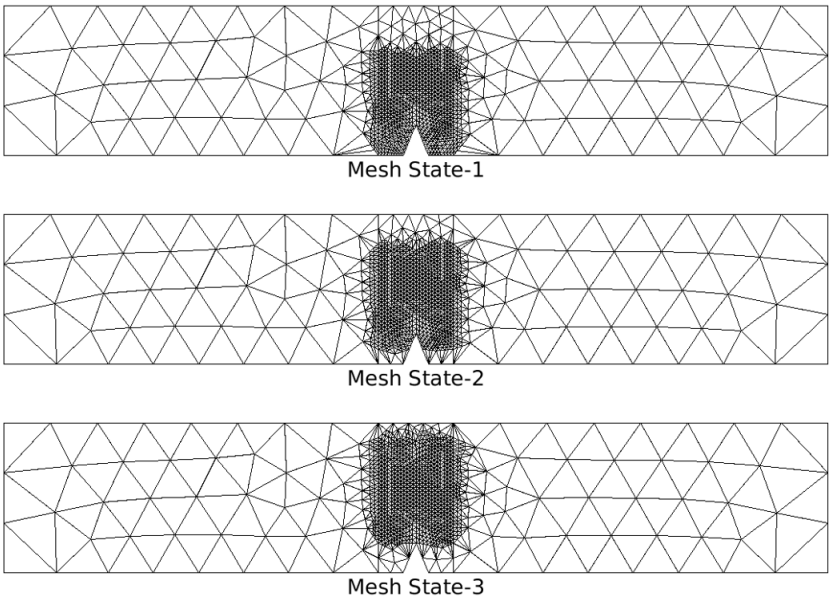


Figure 4.19. Mesh States of Adaptive mesh 1 (5 mm x 7 mm with 2 transitions)

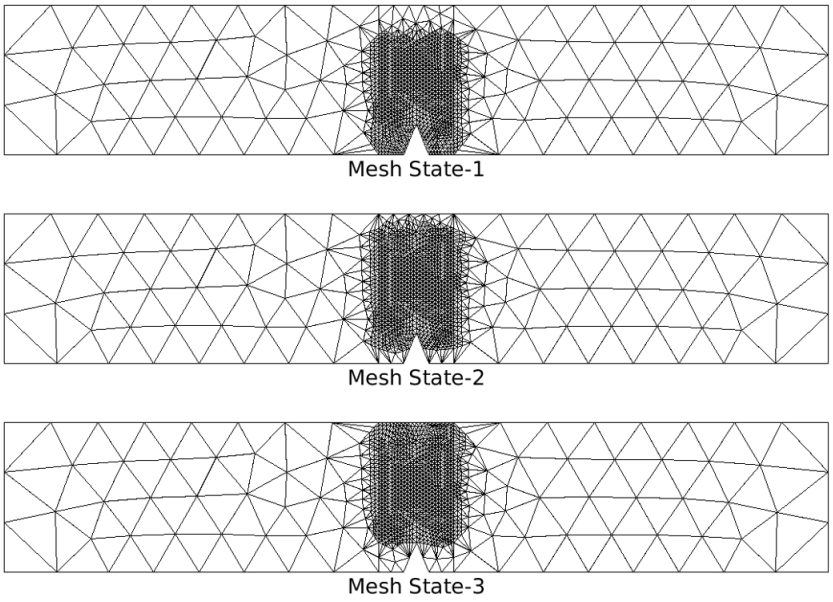


Figure 4.20. Mesh States of Adaptive mesh 2 (5 mm x 8 mm with 2 transitions)

We note that the computational overhead associated with adaptivity is relatively small: 2.7 sec, 5.18 sec and 5.02 sec for adaptive mesh-3, adaptive mesh-2 and adaptive mesh-1 respectively. However, adaptive mesh-1 and 2 undergo two transitions each and hence, the runtime overhead associated with adaptivity is around 2.5-2.7 sec per transition which is much smaller compared to

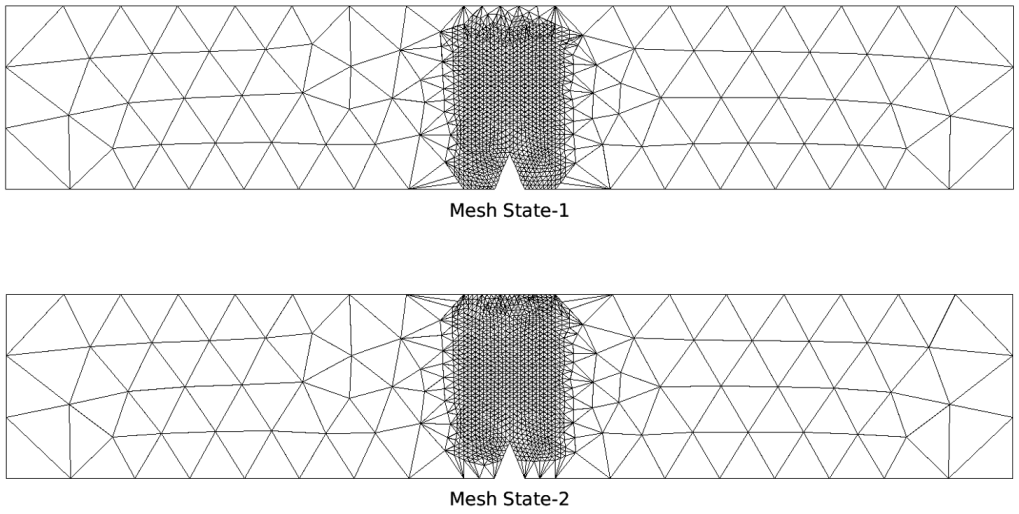


Figure 4.21. Mesh States of Adaptive mesh 3 (5 mm x 9 mm with 1 transition)

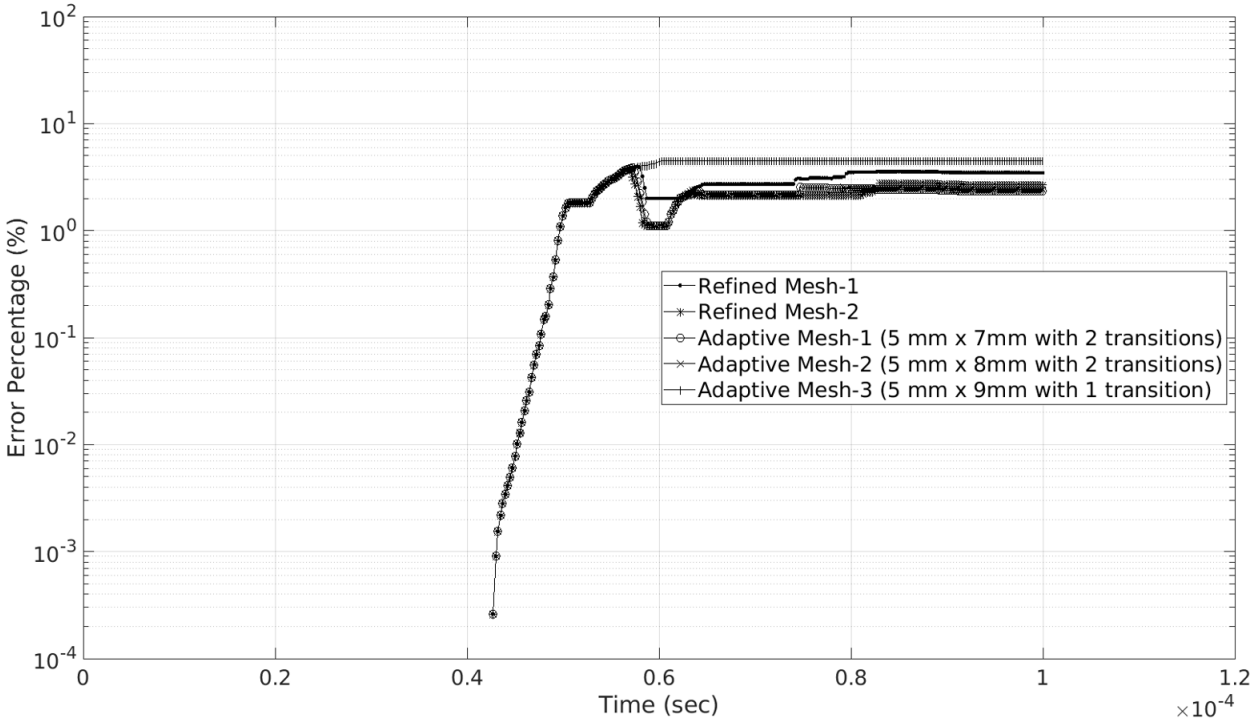


Figure 4.22. Damage Error Translated to 3mm rectangle pixel

the reduction in runtime achieved through adaptivity. Adaptive mesh-1 has the smallest fine mesh region among the adaptive meshes and contains the minimum number of DOFs which results in

Table 4.2. Different Adaptive Meshes vs Refined Mesh-2 with UTS

Mesh	Total DOFs (#)	Number of Transitions (#)	Mesh States Runtime (sec)	Initialization Runtime (sec)	Runtime per Timestep (sec)	Adaptivity Runtime (sec)	Total Runtime (sec)	Runtime Difference from UTS (%)	Maximum Damage Error (%)
Refined Mesh-2	2,430	N/A	N/A	0.96	9.60×10^{-4}	N/A	4,073	N/A	3.82
Adaptive Mesh-3	2,270-2,278	1	137.2	1.15	1.07×10^{-3}	2.79	4,016	-1.4	4.47
Adaptive Mesh-2	2,040-2,120	2	209.5	1.35	8.76×10^{-4}	6.01	3,597	-11.7	3.87
Adaptive Mesh-1	1,828-1,924	2	191.5	1.22	8.81×10^{-4}	5.65	3,528	-13.4	3.84

the maximum reduction in runtime of about 13.4%. This is not surprising as this is the idea behind adaptive mesh refinement where the fine discretization is limited to regions of interest.

Next, we study if similar gains in computational efficiency can be achieved with the AMTS method in comparison to the original (non-adaptive) MTS method. Factors governing the runtime of the AMTS method are the same as the ones discussed in section-3.3 for the MTS method. Table-4.3 presents this comparison for adaptive mesh 1 compared to refined mesh 2. Note that $mAB = 0$ denotes the UTS method which is taken as the baseline case. We note that the runtime reduces by about 32% with AMTS method compared to the UTS method and is also less than the original (non-adaptive) MTS method by about 12%. Additionally, we observe that the maximum damage error with respect to the datum mesh is around 3-4 % for different time step ratios in both refined mesh-2 and adaptive mesh-1. Hence, we conclude that AMTS method can help reduce the runtime while maintaining the accuracy of the solution.

4.3.2 Limitations to adaptivity

Having demonstrated the benefits of the AMTS method, we now define the boundaries to such adaptivity. As discussed in Section-4.1.2, prior knowledge of the body's response under the

Table 4.3. Adaptive Mesh-1 Results Comparison

Mesh	mAB	Adaptivity	Total Runtime (sec)	Runtime Difference from UTS (%)	Max. Damage Error (%)
Refined mesh-2	0	No	4,073	N/A	3.73
Refined mesh-2	1	No	4,245	+4.0	3.73
Refined mesh-2	2	No	3,860	-5.2	3.77
Refined mesh-2	5	No	3,587	-12.2	3.82
Refined mesh-2	10	No	3,568	-12.4	3.88
Adaptive mesh-1	1	Yes	3,633	-10.8	3.84
Adaptive mesh-1	2	Yes	3,191	-21.7	3.86
Adaptive mesh-1	5	Yes	2,820	-30.8	3.93
Adaptive mesh-1	10	Yes	2,784	-31.6	3.97

specified loads is needed to create pre-defined mesh states. In addition, the current data transfer strategy requires the original notch tip to always have fine discretization even when the actual crack tip is away from it. When the fine element at the notch tip, which is fully damaged, undergoes coarsening, the coarsened element might not be fully damaged since not all of its children elements in the previous mesh state were fully damaged. This is an inaccurate representation of the crack which leads to inaccurate results. In the numerical problem that has been discussed in this study has a 2mm deep notch and since the fine mesh travels 1 mm with every transition, we are currently limited to a maximum of two transitions.

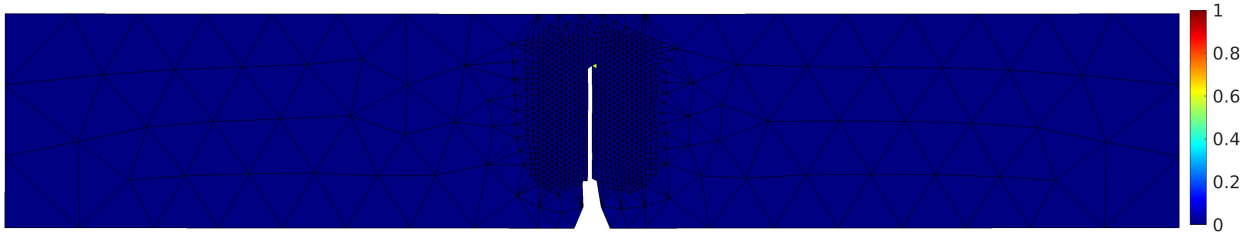


Figure 4.23. Response before the third transition in Adaptive mesh-4

A new mesh, Adaptive mesh-4 with a fine mesh region of 5 mm x 7 mm with 3 transitions is tested as well. The response before and after the third transition is shown in Figures 4.23 and 4.24 respectively. We note that the original notch tip is intact immediately after third transition which is inaccurate. This leads to inaccurate results after the third transition as shown in Figure 4.25. The

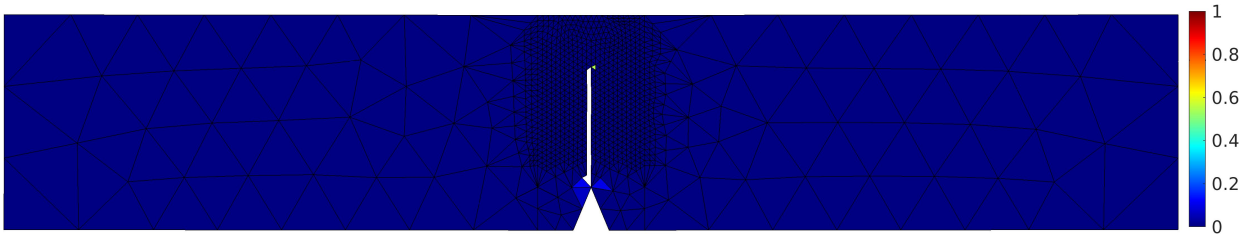


Figure 4.24. Response immediately after the third transition in Adaptive mesh-4

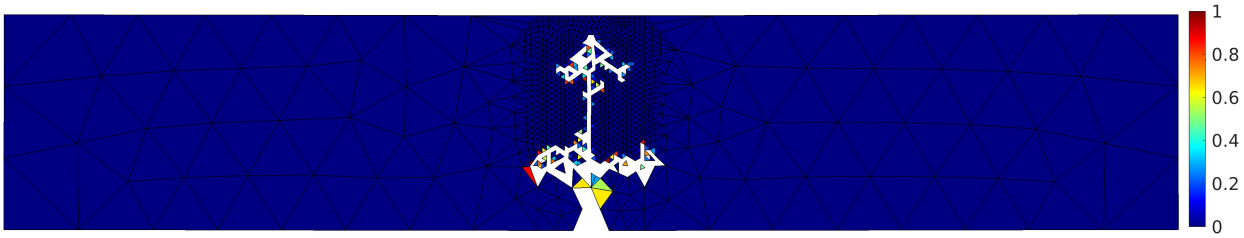


Figure 4.25. Response at 100 μsec in Adaptive mesh-4

inaccuracy in the data transfer when a fully damaged element is coarsened is a limitation of the current study. One possible way to overcome this limitation is to use element weakening method instead of element elimination to represent the crack. The element weakening method will allow assignment of weak properties or reduced stiffness to selected elements once the they undergo complete failure. Consequently, a newly coarsened element after a transition may be assigned a reduced stiffness based on the area weights of all the fine elements which are a part of it.

Despite these limitations and the challenges involved with adaptivity, we conclude that the AMTS method can improve the computational efficiency for simulating dynamically evolving problems in comparison to most existing methods in the literature.

5. SUMMARY AND CONCLUSIONS

The present Adaptive Multi Time-step (AMTS) method provides an efficient way to solve multi-scale problems where the region of interest evolves dynamically during the simulation. This method is an extension of the unconditionally stable, energy preserving multi-time-step method developed by Prakash and co-workers [20], [21] which allows domain decomposition alonog with the use of multiple time steps and different time-stepping schemes.

The current study uses the central difference explicit time integration method to advance the solution in time. We focus on crack propagation in a notched 3-point bending test for quasi-brittle material and use the smeared crack model to simulate cracking. The smeared crack model introduces a parameter called the characteristic length which depends upon the material properties. The characteristic length in turn governs the element size in the mesh. For glass, the element size has to be in the range of 0.125-0.250 mm for accurate representation of cracks. The material model is first validated against a commercial finite element software ABAQUS which also allows simulation of brittle cracking using a smeared crack model. Comparison of results from ABAQUS and the in-house code show that the maximum instantaneous difference in the displacement response is about 5-6%. This difference is due to minor inconsistencies between the respective material models and implementation. Additionally, the damage error is compared for different discretizations against a datum mesh. We find that using the 0.250 mm mesh instead of the 0.125 mm mesh reduces the computational runtime upto 70% when a uniform time-step (UTS) method is employed throughout the domain. Discretizations used for UTS simulations consist of a refined region of small elements throughout the width of the domain along the prospective crack path. The maximum instantaneous damage error in the 0.125 mm mesh is found to be about 0.4% and in the 0.250 mm mesh it is about 4.0%.

A multi time-step (MTS) method where the domain is split into two fixed subdomains and the refined region is solved at the same timestep as the UTS method and the remainder of the mesh is solved at larger time-step with a ratios of 2, 5 and 10, is also used for the 3-point bending simulation. For a a time-step ratio of 10, the MTS method is found to be 12.4% more efficient that the UTS method for similar levels of error.

The adaptive multi time-step method (AMTS) where the subdomain of fine mesh refinement is adaptively updated to track the crack tip is developed. Unlike UTS and MTS, with AMTS, one is able to choose a smaller region of the domain for mesh refinement and therefore reduce the computational cost of the simulation. Adaptivity requires three components: a criterion to determine the region of interest experiencing fine-scale dynamics at runtime, a strategy for adaptive mesh refinement, and a scheme for mapping data from one mesh to another. To save computational time during the simulation, we precompute the possible mesh states of the domain with focused refined region along the prospective crack path. Upon simulating the notched 3-point bending test using AMTS, a 32% runtime reduction is observed over the UTS runtime while maintaining the damage error in the same range.

5.1 Future Directions

Precomputation of various the mesh states of the domain requires prior knowledge or estimation of the body's response to the loading. For the notched 3-point bending test, the crack path is predictable but for general crack propagation problems, this is not the case. Hence, incorporating a method which does not require precomputation of mesh states would be the first step towards improving this method. This would also help overcome the challenge of storage and memory requirements for larger problems.

The central difference explicit time integration restricts the maximum time-step which can be used due to stability reasons. Exploring damage models which are not limited to explicit time integration is another direction for improvement.

Lastly, the current method uses element elimination to delete fully damaged elements. However, using an element weakening method which allows assignment of weakened properties or reduced stiffness to selected elements once the they undergo complete failure may be helpful during data transfer from fine to coarse elements.

REFERENCES

[1] C. Felippa and K. Park, "Direct time integration methods in nonlinear structural dynamics," *Computer Methods in Applied Mechanics and Engineering*, vol. 17-18, pp. 277–313, 1979. DOI: https://doi.org/10.1016/0045-7825(79)90023-9.

[2] K. Park and P. Underwood, "A variable-step central difference method for structural dynamics analysis part 1. theoretical aspects," *Computer Methods in Applied Mechanics and Engineering*, vol. 22, pp. 241–258, 1980. DOI: https://doi.org/10.1016/0045-7825(80)90087-0.

[3] P. Underwood and K. Park, "A variable-step central difference method for structural dynamics analysis- part 2. implementation and performance evaluation," *Computer Methods in Applied Mechanics and Engineering*, vol. 23, pp. 259–279, 1980. DOI: https://doi.org/10.1016/0045-7825(80)90009-2.

[4] P. Bergan and E. Mollestad, "An automatic time-stepping algorithm for dynamic problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 49, pp. 299–318, 1985. DOI: https://doi.org/10.1016/0045-7825(85)90127-6.

[5] G. Hulbert and I. Jang, "Automatic time step control algorithms for structural dynamics," *Computer Methods in Applied Mechanics and Engineering*, vol. 126, pp. 155–178, 1995. DOI: https://doi.org/10.1016/0045-7825(95)00791-X.

[6] O. Bettinotti, O. Allix, and B. Malherbe, "A coupling strategy for adaptive local refinement in space and time with a fixed global model in explicit dynamics," *Computational Mechanics*, vol. 53, pp. 561–574, 2013. DOI: https://doi.org/10.1007/s00466-013-0917-9.

[7] S. Ghosh and J. Cheng, "Adaptive multi-time-domain subcycling for crystal plasticity FE modeling of discrete twin evolution," *Computational Mechanics*, vol. 61, pp. 33–54, 2018. DOI: https://doi.org/10.1007/s00466-017-1421-4.

[8] M. El-Amin, J. Kou, and S. Sun, "Adaptive time-splitting scheme for nanoparticles transport with two-phase flow in heterogeneous porous media," *Computational Science ICCS 2018*, pp. 366–378, 2018. DOI: https://doi.org/10.1007/978-3-319-93713-7_30.

[9] C. Sanchez-Rivadeneira A.G. aand Duarte, "A high-order generalized finite element method for multiscale structural dynamics and wave propagation," *Computer Methods in Applied Mechanics and Engineering*, vol. 384, 2021. DOI: https://doi.org/10.1016/j.cma.2021.113934.

[10] D. Soares, "A simple and effective single-step time marching technique based on adaptive time integrators," *International Journal for Numerical Methods in Engineering*, vol. 109, pp. 1344–1368, 2016. DOI: https://doi.org/10.1002/nme.5329.

[11] D. Soares, "Nonlinear dynamic analysis considering explicit and implicit time marching techniques with adaptive time integration parameters," *Acta Mech*, vol. 229, pp. 2097–2116, 2018. DOI: https://doi.org/10.1007/s00707-017-2104-0.

[12] D. Soares, "An enhanced explicit time-marching technique for wave propagation analysis considering adaptive time integrators," *Computer Methods in Applied Mechanics and Engineering*, vol. 363, 2020. DOI: https://doi.org/10.1016/j.cma.2020.112882.

[13] T. Belytschko, H. Yen, and R. Mullen, "Mixed methods for time integration," *Computer Methods in Applied Mechanics and Engineering*, vol. 17-18, pp. 259–275, 2 1979. DOI: https://doi.org/10.1016/0045-7825(79)90022-7.

[14] W. Liu and T. Belytschko, "Mixed-time implicit-explicit finite elements for transient analysis," *Computers & Structures*, vol. 15, pp. 445–450, 4 1982. DOI: https://doi.org/10.1016/0045-7949(82)90079-7.

[15] P. Smolinski, T. Belytschko, and M. Neal, "Multi-time-step integration using nodal partitioning," *International Journal for Numerical Methods in Engineering*, vol. 26, pp. 349–359, 1988.

[16] C. Farhat and F. Roux, "A method for finite element tearing and interconnecting and its parallel solution algorithm," *International Journal for Numerical Methods in Engineering*, vol. 32, pp. 1205–1227, 1991. DOI: https://doi.org/10.1002/nme.1620320604.

[17] C. Farhat, L. Crivelli, and F. Roux, "A transient FETI methodology for large-scale parallel implicit computations in structural mechanics," *International Journal for Numerical Methods in Engineering*, vol. 37, pp. 1945–1975, 1994. DOI: https://doi.org/10.1002/nme.1620371111.

[18] A. Gravouil and A. Combescure, "Multi-time-step explicitimplicit method for non-linear structural dynamics," *International Journal for Numerical Methods in Engineering*, vol. 50, pp. 199–225, 2001. DOI: https://doi.org/10.1002/1097-0207(20010110)50:1<199::AID-NME132>3.0.CO;2-A.

[19] A. Gravouil and A. Combescure, "A numerical scheme to couple subdomains with different time-steps for predominantly linear transient analysis," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, pp. 1129–1157, 11-12 2002. DOI: https://doi.org/10.1016/S0045-7825(01)00190-6.

[20] A. Prakash and K. Hjelmstad, "A FETI-based multi-time-step coupling method for newmark schemes in structural dynamics," *International Journal for Numerical Methods in Engineering*, vol. 61, pp. 2183–2204, 2004. DOI: https://doi.org/10.1002/nme.1136.

[21] A. Prakash, E. Taciroglu, and K. Hjelmstad, "Computationally efficient multi-time-step method for partitioned time integration of highly nonlinear structural dynamics," *Computers and Structures*, vol. 133, pp. 51–63, 2014. DOI: https://doi.org/10.1016/j.compstruc.2013.11.013.

[22] K. D. Hjelmstad, *Fundamentals of Structural Mechanics, Second Edition*. Springer, 2005, ISBN: 978-0-387-23330-7.

[23] W. Lai, D. Rubin, and E. E. Krempl, *Introduction to Continuum Mechanics, Fourth Edition*. Butterworth-Heinemann, 2010, ISBN: 978-0-7506-8560-3.

[24] O. Zienkiewicz, R. Taylor, and J. Zhu, *The Finite Element Method: Its Basis and Fundamentals, Seventh Edition*. Butterworth-Heinemann, 2013, ISBN: 978-1-85617-633-0.

[25] N. Newmark, "A method of computation for structural dynamics," *Journal of Engineering Mechanics*, vol. 85, pp. 67–94, 1959.

[26] K. Bathe and M. Baig, "On a composite implicit time integration procedure for nonlinear dynamics," *Computers & Structures*, vol. 83, pp. 2513–2524, 2007. DOI: https://doi.org/10.1016/j.compstruc.2005.08.001.

[27] E. Wilson, "A computer program for the dynamic stress analysis of underground structures," *Earthquake Engineering & Structural Dynamics*, vol. 1, pp. 241–252, 1968. DOI: https://doi.org/10.1002/eqe.4290010305.

[28] E. Wilson, I. Farhoomand, and K. Bathe, "Nonlinear dynamic analysis of complex structures," *SESM Report No.68-1, Division of Structural Engineering Structural Mechanics, University of California, Berkeley*, 1972.

[29] K. Bathe and E. Wilson, "Stability and accuracy analysis of direct integration methods," *Earthquake Engineering & Structural Dynamics*, vol. 1, pp. 283–291, 1972. DOI: https://doi.org/10.1002/eqe.4290010308.

[30] J. Chung and J. Lee, "A new family of explicit time integration methods for linear and nonlinear structural dynamics," *International Journal for Numerical Methods in Engineering*, vol. 37, pp. 3961–3976, 1994. DOI: https://doi.org/10.1002/nme.1620372303.

[31] W. Zhai, "Two simple fast integration methods for large scale dynamic problems in engineering," *International Journal for Numerical Methods in Engineering*, vol. 39, pp. 4199–4214, 1996. DOI: https://doi.org/10.1002/(SICI)1097-0207(19961230)39:24<4199::AID-NME39>3.0.CO;2-Y.

[32] G. Hulbert and J. Chung, "Explicit time integration algorithms for structural dynamics with optimal numerical dissipation," *Computer Methods in Applied Mechanics and Engineering*, vol. 137, pp. 175–188, 1996. DOI: https://doi.org/10.1016/S0045-7825(96)01036-5.

[33] Z. Bazant and B. Oh, "Crack band theory for fracture of concrete," *Materials and Construction*, vol. 16, pp. 155–177, 1983.

[34] M. Jirasek and T. Zimmermann, "Analysis of rotating crack model," *Journal of Engineering Mechanics*, vol. 124, pp. 842–851, 1998. DOI: https://doi.org/10.1061/(ASCE)0733-9399(1998)124:8(842).

[35] M. Jirasek and T. Zimmermann, "Rotating crack model with transition to scalar damage," *Journal of Engineering Mechanics*, vol. 124, pp. 277–284, 1998. DOI: https://doi.org/10.1061/(ASCE)0733-9399(1998)124:3(277).

[36] J. Rots and J. Blaauwendraad, "Crack models for concrete: Discrete or smeared? fixed, multi-directional or rotating?" *Heron*, vol. 34, pp. 1–59, 1989.

[37] J. Rots, P. Nauta, G. Kusters, and J. Blaauwendraad, "Smeared crack approach and fracture localization in concrete," *Heron*, vol. 30, pp. 1–48, 1989.

[38] A. Varshneya and J. Mauro, *Fundamentals of Inorganic Glasses, Third Edition*. Elsevier, 2019, ISBN: 978-0-12-816225-5.

[39] W. Callister Jr. and D. Rethwisch, *Materials Science and Engineering: An Introduction, 10th Edition*. Wiley, 2018, ISBN: 978-1-119-40549-8.

[40] *Abaqus unified FEA*, URL: https://www.3ds.com/products-services/simulia/products/abaqus/, 2021.