AUTOMATED MODELING AND REMESHING

IN METAL FORMING SIMULATION

By

Nitin Vasant Hattangady

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

In Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: Mechanical Engineering

Approved by the
Examining Committee:

_____
Mark S. Shephard, Thesis Advisor

_____
Antoinette M. Maniatty, Member

_____
Robert L. Spilker, Member

_____
Joseph E. Flaherty, Member

Rensselaer Polytechnic Institute
Troy, New York

August 2003
(For Graduation August 2003)

TABLE OF CONTENTS

LIST OF FIGURES

Dedicated to my parents
Vasant & Mira Hattangady

and to my family
Anita, Rohan & Anjali

ACKNOWLEDGEMENTS

There are many people who have helped me along the road that has led to the completion of this dissertation. I thank my parents for the constant encouragement to pursue my dreams. Without the support of my wife, Anita, I would not have successfully reached this goal. I am very grateful for her enduring love throughout the whole process and for her patience and encouragement. To our new family additions, Rohan and Anjali, thanks for providing a big incentive to finish.

I thank my advisor, Dr. Mark Shephard whose dedication and passion for research has been and will always be an inspiration for me. His guidance throughout my research work has been very valuable and is greatly appreciated. My sincere appreciation to the members of my committee, Prof. Maniatty, Prof. Spilker and Prof. Flaherty for their assistance.

I thank my ex-colleagues from UES Software Services, especially Dr. Anil Chaudhary (currently with Applied Optimization, Inc.) for providing guidance and technical information on the Antares™ system. Always enthusiastic, optimistic and supportive, late Dr. Sokka Doraivelu of UES Software Services brought cheer to everyone around and was instrumental in encouraging me to pursue my Ph.D. at RPI. Thanks also to Mr. Larry Clay of UES Software Services for providing me with a copy of the Antares™ system for thesis related use.

My sincere appreciation for the support received from my current employer, Alcoa Technical Center. ATC has been very supportive of my work and thanks to my colleagues Rao Vemuri, Joe Fridy, John Watton, Bill Herbein and Dave Selfridge (Alcoa, Cleveland) for their technical help/assistance.

ABSTRACT

Process modeling has become an effective tool in reducing the lead-time and the cost for designing forming processes. Process simulation can give comprehensive details including material flow, strain/stress/temperature distributions that assist the designer in honing the process design to obtain optimal results. Commercial finite element solvers to model forming processes such as forging, extrusion, etc. are available. However, a major deterrent in their effective use has been the need to provide a new mesh to represent the deformed workpiece. Until recently, this process, known as *remeshing*, was performed manually by process designers. Each workpiece remesh can potentially take several days for a complex 3-D component. Hence, an automated modeling environment that would automatically remesh the deformed workpiece when required and continue the analysis can dramatically reduce the overall modeling time and result in this technology being used in the design of industrial forming processes. This has provided the motivation for the thesis research that has led to the development of an automated modeling system for forming simulation. The research effort has focused on the issues that must be addressed in developing an automated modeling system that is robust, has automated procedures to update the geometry and remeshes the deformed workpiece model with concern for controlling the simulation errors.

Major contributions arising out of this work include procedures for (i) automatic remeshing of the deformed workpiece, (ii) use of CAD model of the die geometry during analysis and remeshing, and, (iii) curvature compensation of nodes on workpiece free

surface. Curvature compensation of nodes allows us to mimic the smooth geometry on the workpiece free surface. The CAD models enable use of the true geometric representation of the die rather than its abstraction – the mesh. Hence, we get better control over the distribution/loss of workpiece volume. These procedures have been integrated with the commercial analysis and CAD tools to provide an automated modeling environment. The system has been very successful in modeling forming processes involving complex 3-D geometries, eliminated the need for manual remeshing and reduced the overall simulation time by more than 90%.

Chapter 1

INTRODUCTION

Process modeling has become an effective tool in reducing the lead-time and the cost for designing forming processes for manufacturing automotive and aerospace components. Several research programs aimed towards the development of the experimental and mathematical analysis of the mechanics of forming operations, computer software for process modeling, and transfer of this technology to the industry have contributed towards the overall objectives of superior technology for process design in net shape and near net shape forming.

Process designers can use these analysis tools and modern visualization techniques for virtual prototyping of the forming process. Simulation of the process on the computer can give comprehensive details of the process including material flow, strain/stress/temperature distributions, etc. These details assist the designer in honing the process design to obtain optimal results. Some of the parameters that the designer can alter include local die geometry, process parameters, etc. Effective application of simulation technologies reduces the number of expensive die trials required, thus leading to a significant reduction in lead time and cost for process design.

The finite element method has been successful in the computer modeling of metal forming operations like forging, extrusion, rolling, etc. In metal forming, plastic strains usually outweigh elastic strains and the idealization of rigid-plastic or rigid-viscoplastic

material behavior is typically acceptable. The resulting analysis based on this assumption is known as the *flow formulation* [1] and the solution to the boundary value problem is obtained when the first order variation ($\delta\pi$) of the functional, π, vanishes where

$$p = \int_V \bar{s}\,\dot{\bar{e}}\,dV - \int_{S_F} F_i u_i\,dS \qquad \textbf{(1)}$$

In the above equation, $\bar{\sigma}$ is the effective stress, $\dot{\bar{\varepsilon}}$ is the effective strain rate, F_i are the surface tractions and u_i represents the kinematically admissible velocity field. Using the penalty method to account for the incompressibility constraint and taking the first order variation results in:

$$dp = \int_V \bar{s}\,d\,\dot{\bar{e}}\,dV + K\int_V \dot{e}_V d\,\dot{e}_V\,dV - \int_{S_F} F_i u_i\,dS = 0 \qquad \textbf{(2)}$$

where K is a penalty constant and $\dot{e}_V = \dot{e}_{ii}$ is the volumetric strain rate [1].

When the velocity solution is obtained, the geometry of the deformed workpiece is obtained by updating the coordinates of the nodes (Lagrangian mesh system). This procedure, know as the updated Lagrangian method, is typically employed to model the deformation mechanics [1-7]. This implies that the workpiece mesh model is allowed to evolve to represent the material flow. However, due to the imposition of large plastic strains, typical to bulk forming processes, the workpiece mesh undergoes severe distortion resulting in a situation that does not allow the analysis to be continued due to zero or a negative jacobian in one or more elements. To continue the analysis, a new, valid mesh representing the deformed workpiece must be generated using a “remeshing

process". A mesh model is of acceptable quality if the elements are not severely distorted and all element jacobians are positive. Note that other "remeshing triggers" or criteria can also be used to indicate the need for remeshing and these are discussed in Chapter 2. The remeshing criteria must be implemented within the solver to avoid unnecessary computations.

The advent of fast computers over the last few years has reduced the solution time once a mesh with an acceptable quality is provided as input. Hence, the bottleneck to obtaining a cost and time effective solution to the forming problem is remeshing of the workpiece. Until recently, the remeshing process was performed manually and potentially took several days for each remeshing (several are typically needed to model the entire process) of a 3-D component. Also, manual remeshing can potentially smooth the geometry thus preventing boundary defects (such as a lap) from being detected, or, introduce constraints that result in false prediction of surface defects from process modeling. Hence, a 3-D modeling system, that would automatically generate a new mesh on the deformed workpiece and continue the analysis, can dramatically reduce the overall modeling time and result in this technology being widely used in the design of industrial forming processes. This has provided the motivation for the thesis research that has led to the development of an automated modeling system for forming simulation. To put the discussion in proper context, a brief description of the typical steps, followed by a designer in designing a forming process, is presented. This will help set the stage for clear identification of technological gaps in the current state-of-the-art and hence, the goals for this thesis research.

1.1 Process Modeling in Metal Forming

The flowchart in Figure 1 lists the steps typical to process modeling of forming operations. The modeling procedure can be broken down into four main steps:

1. Pre-processing, where the CAD, mesh models for the workpiece and the dies are generated and prepared for analysis with the definition of initial and boundary conditions, material and interface properties, etc.,

2. Process analysis, where the finite element solver will analyze the problem defined in step 1 till a remeshing criterion triggers a requirement for a new mesh,

3. Geometry update, where a new geometric representation is constructed based on the deformed workpiece mesh model (available from the solver),

4. Mesh generation, where the newly constructed geometric representation of the deformed workpiece is discretized into finite elements based on the requirements of the solver.

A typical process design phase begins with the building of solid models representing the die and the initial workpiece geometries. The CAD models for the dies are also used in the generation of machining data once the process design is complete. A discretization for the die and workpiece geometries is generated using a mesh generator linked to the CAD system or by transferring the geometry information to a mesh generation environment via a direct translator, an IGES file or a STEP interface. The data transfer

option could potentially result in a situation where the engineer spends additional time in model cleanup due to errors in data translation. In addition to loss of accuracy, data translation also results in loss of feature data. This implies that if the die geometry is altered for design changes, the entire process of data translation, model cleanup and mesh generation may need to be repeated.

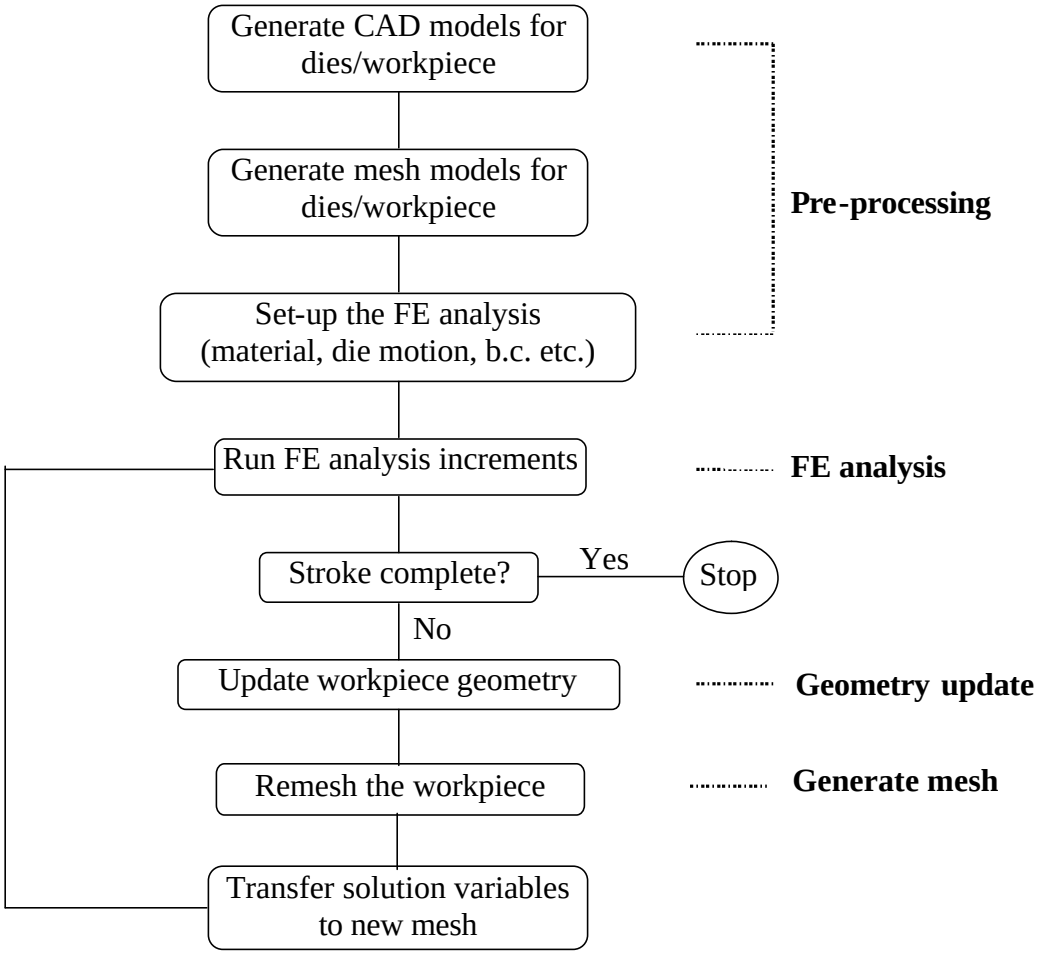


Figure 1: Process modeling in metal forming – typical steps

The finite element analysis system will impose certain restrictions on the type of mesh used to represent the die/workpiece models. Typically 4-node tetrahedral elements are used to represent the volume and the surface is represented by 3-node triangles. The

solver also provides the functionality to transfer the solution information onto a new mesh model. Once the mesh models are generated, the problem is set-up in a pre-processing environment that is typically provided with the finite element solver. This step involves defining the initial and boundary conditions, material and die-workpiece interface properties, and the die motion characteristics. The solver is then invoked to analyze the deformation mechanics of the process.

To continue the analysis after the solver has stopped, we need to build a geometric representation for the deformed workpiece, which can then be discretized to obtain a new mesh containing valid elements. The solver provides the node coordinates and the element connectivity information for the deformed workpiece. This is typically transferred to a finite element pre-processing system or a CAD system where the new geometric representation is carefully constructed. Designers are careful not to introduce surface discontinuities during this stage since this may introduce artificial constraints in the ensuing stage of analysis. This is typically a very time intensive process even for fairly simple problems. Since several of these geometry construction steps are typically needed during the modeling of the entire process this step becomes the biggest bottleneck in the forming simulation process. Once the geometry is constructed, its discretization into tetrahedral elements is quite easy and fast due to the availability of automatic mesh generators.

1.2 Thesis Organization

The goal of this thesis research is to develop a fully automatic procedure to update the geometry of the deformed workpiece and generate its discretization in an effort to develop an automated modeling system for forming analyses. There are several key technical issues that affect the overall accuracy of the simulation that must be considered in the automation of the process.

The use of finite element methods for problems where the domain evolves during the simulation presents a number of challenges that are not present in solutions for a fixed domain. Two high level issues that must be considered when automating process simulations are:

(i) *when* to remesh, and,

(ii) *how* to remesh.

The criteria used to trigger a remesh are collectively called the *remeshing criteria*. Four sources of errors that influence the decision to remesh are: (i) geometric approximation errors, (ii) element distortion errors, (iii) mesh discretization errors, and, (iv) mesh rezoning errors (see chapter 2). The impact of the different types of errors encountered based on metrics to measure them will key a remeshing step. The process of remeshing focuses on controlling these errors so that the simulation can continue.

The objective of the geometry update procedure is to take a finite element mesh representing the current state of the workpiece, and evolve the geometric representation of the workpiece to represent its current deformed shape. This geometric representation is then discretized to generate a mesh consisting of valid, high quality elements focused on controlling the mesh discretization and distortion errors. Chapter 3 discusses the important considerations and requirements for the geometry update procedure, which include the approximation to the geometry of the deformed workpiece, consistent transfer of analysis attributes and consistent geometric interpretation of contact. Domain specification and possible procedures for updating the geometric representation are key issues important to track the deformed workpiece. Maintaining total workpiece volume and its distribution are key to overall accuracy of the simulation.

Procedures for consistent handling of die geometry and use of the CAD model of the dies during analysis are presented in chapter 4. The selected tools are discussed and procedures for determining geometric contact/containment and the die-overlap monitoring algorithm to control geometric approximation errors are discussed. Details of the geometry update procedure used in the automated system developed in this research effort are discussed in chapter 5. This includes the data structures defined to rebuild model topology, the geometry update procedure and procedures for free surface curvature compensation and adaptive refinement of workpiece mesh. Limitations to the geometry update algorithm are also identified. Validation of the implemented procedure is presented in chapter 6 where results from two forging process simulations are presented.

A mesh-coarsening tool has also been developed to eliminate elements not needed to represent the die geometries when the die objects are treated as rigid during forming simulations. Since contact computations have a significant impact on overall compute time, reducing the number of elements representing the die boundary reduces the overall compute time for the process simulation. It is shown that this reduction is obtained without any loss of accuracy.

Limitations to the overall approach and possible solutions to alleviate/eliminate the problems encountered are discussed in chapter 8. Conclusions and major contributions from this thesis research are also identified.

Chapter 2

AUTOMATING FORMING PROCESS MODELING – REMESHING ISSUES AND CRITERIA

Issues relating to the task of deciding *when* to remesh the deformed workpiece geometry will be the focus of the discussion in this section. Due to the finite dimensional nature of finite element analysis and the approximation of the true domain used to represent the die and workpiece geometries, there will always be approximation errors in the solution. To improve overall solution accuracy and make the simulations realistic, these errors need to be detected and a new mesh must be generated when these errors become large. The key sources of errors associated with the finite space approximation in the modeling of forming problems [51] are:

(i) geometric approximation errors,

(ii) element distortion errors,

(iii) mesh discretization errors, and,

(iv) mesh rezoning errors.

In the following sections, we will examine each of these types of errors and metrics to indicate when the error is becoming large and techniques to improve the accuracy and make simulations more realistic are given. From the description presented in this chapter, it will be clear that the finite element solver can monitor the error indicators due

to geometric approximation, element distortion and mesh discretization as the simulation progresses. Hence, they can be used as the basis to decide if the workpiece needs to be remeshed. The decision to remesh will depend on the criteria used to detect the errors and trigger a remesh. The occurrence of mesh rezoning errors is not considered in this thesis. However, others have considered this area and tools developed in [8, 9] have been relied upon for interpolation of solution variables onto the remeshed workpiece.

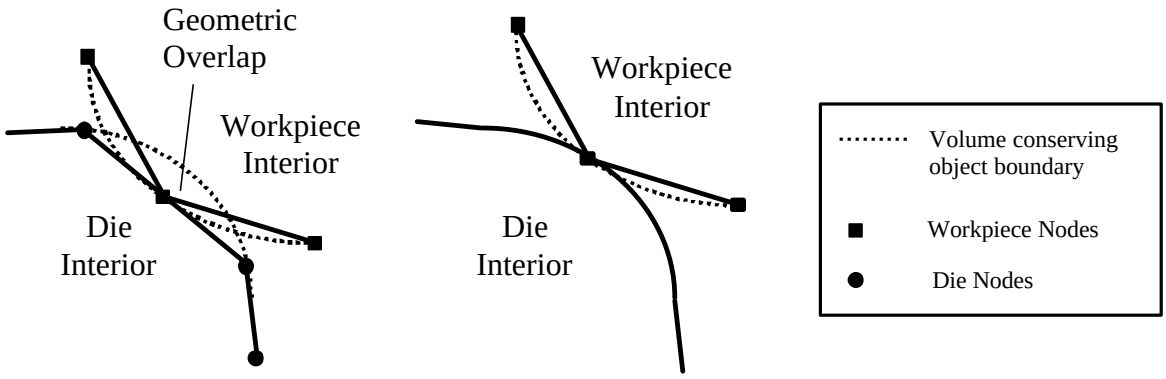
2.1 Geometric Approximation Errors

Geometric approximation errors exist because a finite element mesh approximates the workpiece and the dies during analysis. Use of such an approximation leads to an inaccurate representation of the die-workpiece interaction and the workpiece free surface during the simulation. This could potentially have a serious impact on the material flow computations resulting in an inaccurate prediction of the workpiece deformation pattern and die fill. Small forming defects such as laps (where the material folds onto itself) could go undetected, and workpiece remeshes based on the approximation could lead to volume loss or gain. Each of these issues is explained below.

The impact of geometric approximation on die-workpiece interaction is shown in Figure 2. Typically, the contact algorithms in finite element solvers [8-12] ensure that workpiece nodes do not penetrate the mesh of the die boundary. Hence, initiation of contact with the true curved geometry could occur much before (or after) the analysis code is able to detect it. *Geometric overlap* of the die and workpiece geometries is illustrated in Figure 2a. Since plastic deformation is a path (strain history) dependent

phenomenon, use of an approximation could potentially modify the computed deformation behavior for the workpiece. The reason for this is that, depending on the local die curvature and the discretization used to represent that region, portions of the workpiece (near existing contact regions) are likely to be subjected to more (or less) deformation before the solver determines new contact regions. This is illustrated in Figure 3, which shows the additional die-stroke for the top die before the free surface of the workpiece contacts the bottom die (exaggerated to illustrate the point). This implies that some regions in the workpiece would have a different strain history when an approximation of the die is used in the analysis as opposed to case when the true curved geometry is used during computations. A different strain history would result in a different future path of deformation (loads, material flow). As the simulation progresses and the workpiece remeshed several times, these errors could potentially result in a re-distribution of workpiece volume causing die underfill or overfills and in a different load-stroke curve for the process (as compared to the case when the true curved geometry of the die is used). Note that the issue of volume re-distribution refers to the variation in the workpiece shape for the two (die mesh versus die geometry) scenarios. Obviously, the discretization of the die affects this variation with a finer discretization reducing it. This effect is illustrated for the 3-D analysis of a disk-forging problem in Figure 4, which shows the variation in the load-stroke curves (in the final stages of the analysis) when the mesh and the geometry are used to represent the die during the analysis. The data has been culled from simulation results performed with the Antares^{TM} [8, 9] system and discussed later in this document. The load-stroke curves differ more during the final stages when the material is beginning to flash and all of the corners (curved regions of the

die) begin to fill up. Note that the load-stroke curves do vary throughout the process simulation though only the variation for the final 15% of the stroke is plotted in Figure 4.



(a) contact based on die mesh **(b) contact based on die geometry**

Figure 2: Geometric approximation errors in the contact region

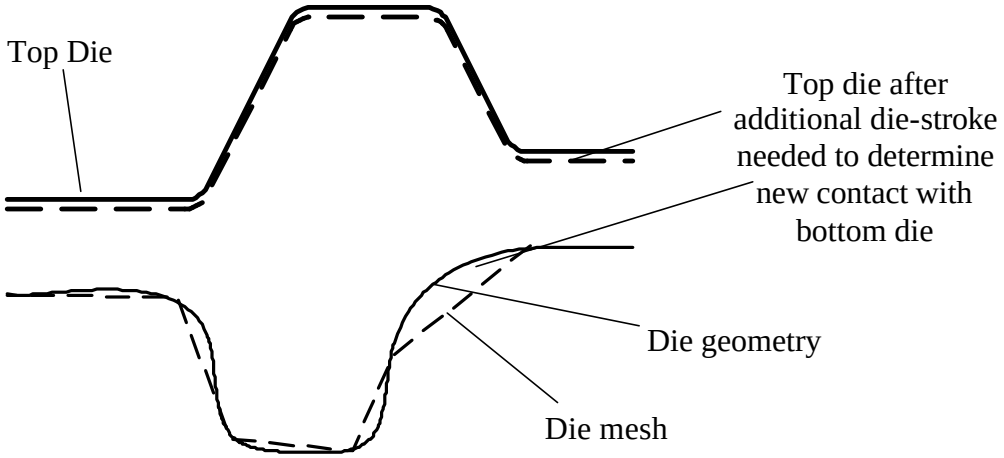


Figure 3: Impact of die geometry approximation on process simulation

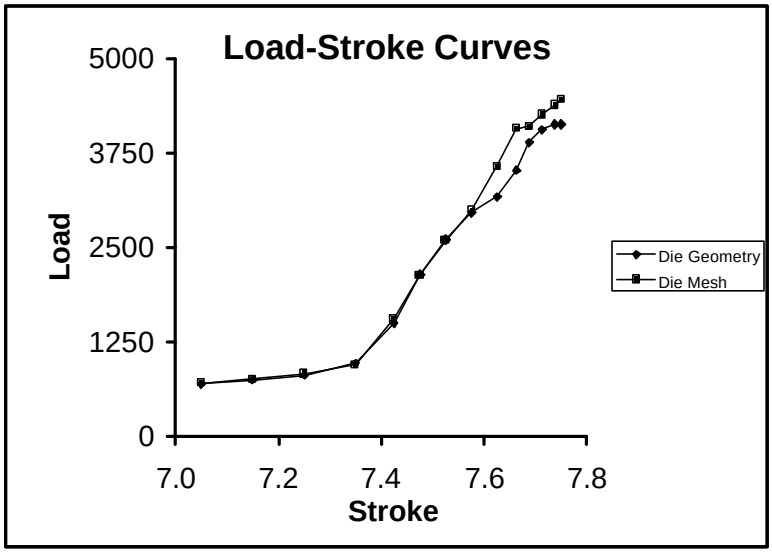


Figure 4: Load-stroke curves (final stages) for disk forging problem

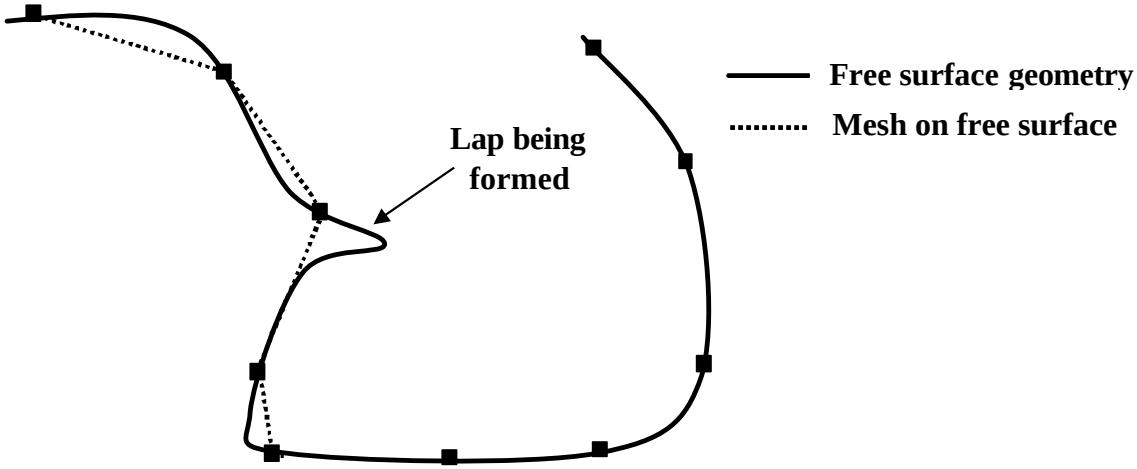


Figure 5: Workpiece free surface approximation in vicinity of a lap

An example of a potential situation where an inaccurate representation of the workpiece free surface during the simulation causes a forging lap to be missed is illustrated in Figure 5. The geometry of the free surface of the workpiece shows the formation of the lap. However, the discretization generated does not capture this feature on the workpiece

boundary. Further simulation with this discretization will result in an erroneous prediction of a process that does not produce any defects.

Use of the finite element mesh of the workpiece as the basis for generating the new workpiece mesh (during a remesh) can potentially result in volume loss/gain. An extreme scenario of volume loss is illustrated in Figure 6 below. It is clear from Figure 6 that each remesh can potentially change the workpiece volume. Obviously, this can have a very serious impact on the accuracy of all predicted simulation parameters. The issue of volume loss during remeshing is discussed further in the next chapter dealing with *how* to remesh the deformed workpiece.

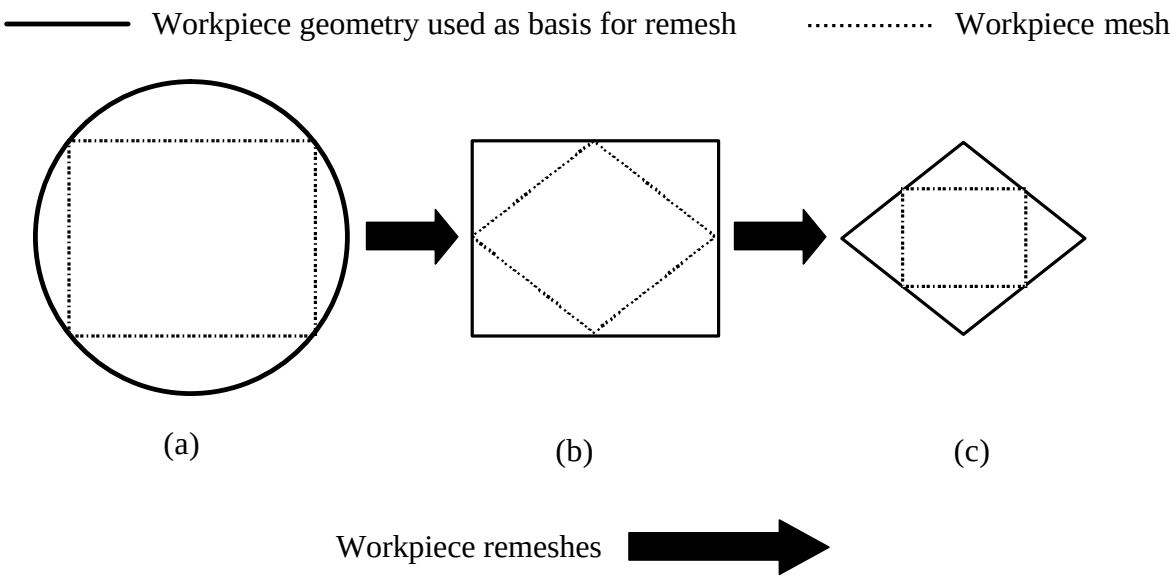


Figure 6: Workpiece volume change due to remeshing

In an application where one of the primary goals is the prediction of workpiece material flow during deformation, geometric approximation errors have the potential of yielding inaccurate results. In a typical 3-D modeling environment where the workpiece and dies

are represented as mesh models with straight edged elements, the severity of the geometric approximation errors strongly depends on how the discretizations of the workpiece and the dies are controlled. Having identified the situations that can be attributed to geometric approximation errors, the rest of this section focuses on actions that will reduce these errors thus making the simulations more realistic.

Geometric approximation errors can be reduced, for example, by using the true geometry of the die as its representation during the analysis. This implies that the die-workpiece interaction illustrated in Figure 2(a) and resulting in geometric overlap would be eliminated and represented by the model shown in Figure 2(b). This can be accomplished by using the geometric definition (lines, arcs, splines in 2-D and planes, complex surfaces in 3-D) of the boundary of the dies (instead of the mesh boundary) in the geometric checks performed during contact computations in the solvers. These geometric checks determine the proximity of a node to the boundary of a die. Use of true geometry of the die is typically implemented in most commercial 2-D solvers [10-12] and in the 2-D automated systems discussed in published literature [15-18]. However, this is not common in 3-D solvers due to the complexity of the algorithms to perform the geometric checks with complex surfaces. None of the published articles describe implementations of 3-D automated systems [38-50] with the capability of dealing with the geometry of the die. All of these systems assume that the geometry of the die is defined by its finite element mesh. However, with the open environments provided by commercial CAD systems, procedures to query the geometric model and perform geometric computations are available and hence, do not have to be duplicated. For this strategy to work, the contact algorithm in the solver must be modified to perform contact

checks using the die geometry from the CAD system. Hence, the contact algorithm should now check for contact between workpiece nodes and the geometric model of the die (instead of the mesh model). This approach has been implemented within the automated system developed in this thesis research and the details of this implementation are provided in Chapter 4.

Geometric approximation errors can also be reduced by building a smooth geometric representation of the workpiece (from the deformed finite element mesh data) and using it as a basis for generating the new mesh. With reference to Figure 6, this would imply that before the mesh in Figure 6(b) is generated, we would build a smooth geometric representation based on the deformed mesh of Figure 6(a), thus recreating the circular boundary in this case. Thus, when the smooth representation is discretized, there would be no loss of volume. This issue of building a smooth representation for the workpiece is discussed further in the next chapter, which deals with how to remesh the workpiece.

Monitoring the "die overlap" can further reduce geometric approximation errors. The die overlap measures the overlap between the workpiece (usually represented by its finite element mesh) and the die (mesh or geometry). A typical situation with die geometry and straight-edged workpiece mesh combination is illustrated in Figure 7. After the workpiece is remeshed, some or all of the volume represented by the shaded region in Figure 7 may be lost (or gained depending on the local die curvature) since nodes on the mesh representing this contact portion of the workpiece are forced to be in contact with the die. Hence, to avoid drastic changes to workpiece volume, such situations must be detected.

A simple procedure to monitor die overlap in 2-D simulations is explained with reference to Figure 7. The procedure would check the distance between the mid-point of the finite element edge e_1, whose end-points (nodes *1* and *2*) are in contact with a die, and the die geometry. If this overlap, δ, is greater than a threshold, a remesh is triggered. It is important to note that this procedure only attempts to keep volume loss (due to remeshing) under control by careful monitoring and does not ensure volume conservation. Though it is not illustrated in Figure 7, monitoring die overlap will also reduce geometric approximation errors when a finite element mesh represents the dies. The procedure described above is a simple procedure to estimate the overlap and would clearly be unsuitable in situations such as the one illustrated in Figure 8 where the die overlap would be computed as zero. However, (it is assumed that) these types of geometric situations occur infrequently in the simulation of an industrial forming process since the workpiece mesh is likely to be finer than the features on the die. Also, the dies typically do not have small features since they can present die fill problems and result in an increase in forming loads. This algorithm can easily be extended to and implemented for complex 3-D geometries due to the availability of the modeling functionality via the open environments provided by the commercial CAD systems. The *die overlap-monitoring* algorithm described above has been implemented within the automated system developed in this thesis research and the implementation details are discussed in Chapter 4.

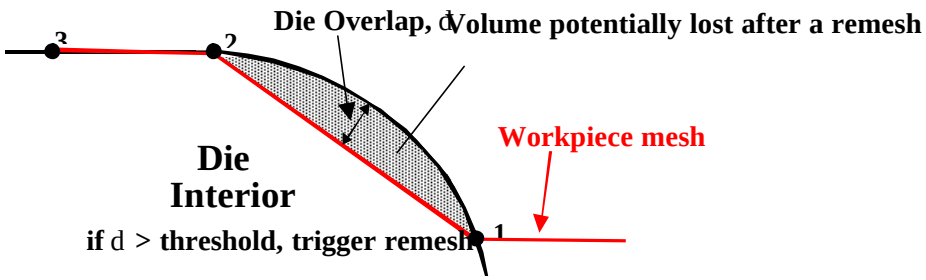


Figure 7: Monitoring die overlap

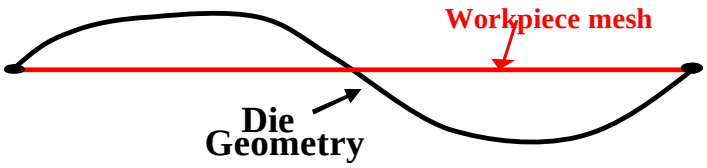


Figure 8: Problem geometry for the overlap-monitoring algorithm

Another technique that helps to reduce the geometric approximation errors is mesh refinement around separation points of the contact region. This results in a finer mesh in the regions that are most likely to come in contact with a die as the simulation (forming operation) progresses and in a better representation of the workpiece as the material flows around the curved surfaces on the die. Baehmann et. al. [16] have implemented a 2-D remeshing scheme that implements this approach. However, this is an *a posteriori* fix to minimize the geometric approximation errors for the next phase of the simulation. A better scheme for minimizing approximation errors would be to develop a procedure that "looks ahead" and refines the mesh in regions that are likely to come in contact. For example, Figure 9 shows the possible introduction of a refinement region where a finer

discretization is generated using the center of the region as a refinement point. Such a procedure would "scan" the boundary of the workpiece (only the free surface) to check for proximity to die surface. Portions of the workpiece free surface within a reasonable distance from the die can be refined to capture the die curvature as the workpiece flows around the curved die surface. The algorithm can also be made smarter by tracking nodal velocity directions to infer regions likely to come in contact with a die.

One such scheme to check proximity with a die boundary would be to define a region, of pre-defined size and centered at the given node location and check if a portion of this region lies inside a die. The region could be a square/circle in 2-D and a cube/sphere for 3-D applications. Figure 9 shows a schematic illustration of this scheme using a square for the region. If a corner of this box lies inside a die (as shown in Figure 9 for nodes 2 and 3), the location of the node is treated as a refinement point when the new mesh is generated. Hence, in Figure 9, the locations of nodes 2 and 3 on the workpiece boundary become points in whose vicinity the mesh is refined. This offers a good solution considering the fact material flow cannot be predicted with certainty by evaluation of the current die-workpiece configuration. A similar "look ahead" algorithm, that examines the proximity of midpoints of edges on the workpiece free surface, has been implemented within the automated system developed in this thesis research and the implementation details are discussed in section 4.6 of Chapter 4.

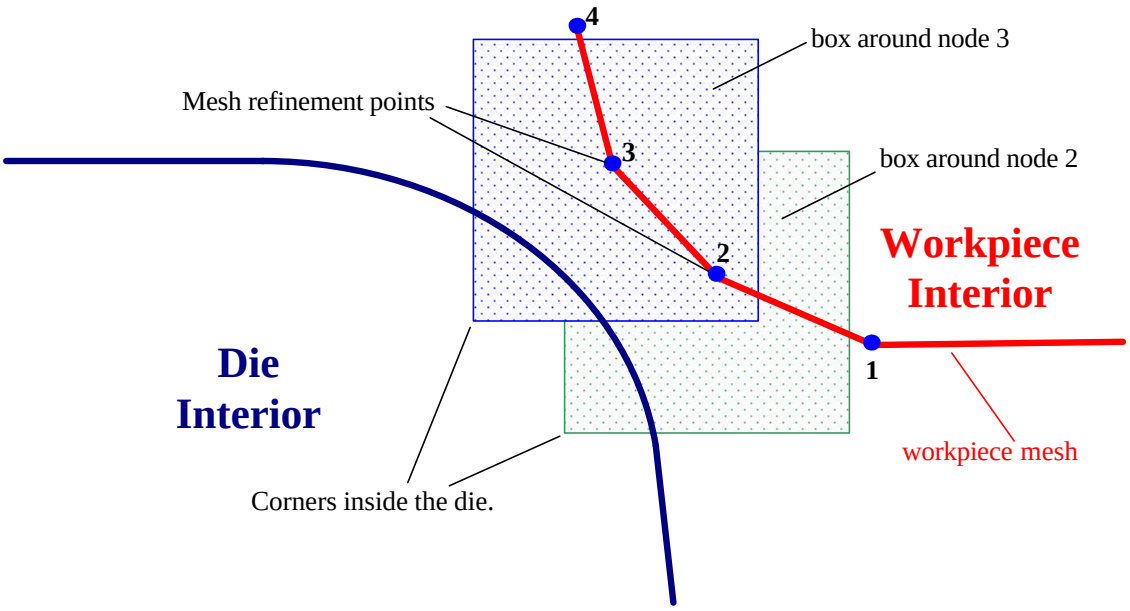


Figure 9: Schematic illustration of the "look-ahead" algorithm

2.2 Element Distortion Errors

The element shape has a bearing on the accuracy of the results of finite element analysis when the elements deform toward the point of being invalid. Meshes with distorted elements make solutions both more difficult to compute and less accurate [52]. Several authors [53-55] have discussed the stiffening of isoparametric elements when they are distorted. Strang and Fix [56] state that "poorly shaped" elements have an effect on the condition number of the linear algebra problem being solved. For triangular elements, Babuska and Aziz [57] have developed the mathematical analysis leading to the requirement that the largest angle should be bounded away from 180^{o}. Krizek [58] has extended the maximum angle criterion to tetrahedral elements and proved that the maximum angle of all triangular faces of the tetrahedron and the maximum angle

between faces of the tetrahedron should be bounded away from 180^{o}. The peak displacement characteristics of a cantilever beam modeled with various distorted isoparametric elements have been presented to show the deleterious effect of element distortion on the accuracy of the results [59-61]. The relationship between element distortion and solution accuracy in finite element analysis are complex and not fully qualified functions of the geometry of the domain, the boundary conditions, the governing differential equations and the geometry of the finite element mesh used for analyzing the problem [62]. Therefore, the most common approach to consider the influence of element shape is to employ a reasonable distortion metric to decide if elements are distorted enough to necessitate remeshing. To be valid, a distortion metric must (i) measure the deviation of the element in the physical (x, y, z) domain from its assumed shape in the computational (ξ, η, ς) domain, (ii) be sensitive to combined stretching and shearing of the element, (iii) not be affected by rigid body displacements, (iv) be independent of element size, and (v) enable a boolean rejection of acceptance decision to be computed conveniently [62].

One measure of element distortion is the element jacobian that relates the computational domain to the physical domain. Since the computational (ξ, η, ς) domain is different from the physical (x, y, z) domain, to transform the variables and the region with respect to which the integration is made, a standard process is used which involves the determinant of the jacobian matrix, J [75, 76]. Thus, for instance a volume element becomes:

$$dx\ dy\ dz = |J|\ d\xi\ d\eta\ d\zeta \qquad \textbf{(3)}$$

When the determinant of the jacobian matrix becomes zero or negative, the transformation from the physical to the computational domain is not valid due to the loss of the uniqueness of the mapping. Under these circumstances, a local remesh to change the shape of the distorted element or, a complete remesh of the deformed workpiece geometry is necessary. A positive jacobian measure will only ensure basic element validity and will not indicate potential of numerical stiffening of the elements, which is likely to reduce accuracy. Another drawback with this method is that the element jacobians are typically monitored only at the integration points. Hence, in some cases, the solution may continue to be accepted as being valid even in the presence of elements that have become invalid (element may have folded onto itself and the largest angle within the element become greater than 180^{o}). Other metrics to measure element distortion have been proposed and some of them are discussed below.

Yang, et. al. [30] monitor the convergence of the solution of the non-linear inverse mapping of the element shape functions. In this method, the local coordinates at node points are found by solving an inverse mapping problem, i.e., mapping from the global coordinate system (x, y, z) to the local coordinate system (ξ, η, ζ). The mapping is defined by the geometric functions for the element:

$$x = \sum_{i=1}^{n} N_i(\xi,\eta,\zeta)\ x_i, \quad y = \sum_{i=1}^{n} N_i(\xi,\eta,\zeta)\ y_i, \quad z = \sum_{i=1}^{n} N_i(\xi,\eta,\zeta)\ z_i \qquad \textbf{(4)}$$

where x_i, y_i and z_i are the coordinates of the node points of an element with n nodes and N are the basis functions. Rearranging the terms and solving for ξ, η, ζ for an 8-node brick element (paper gives the formula for a 4-node quadrilateral element) gives:

$$\begin{Bmatrix} \xi \\ \eta \\ \zeta \end{Bmatrix} = \begin{bmatrix} \sum x_i \xi_i & \sum x_i \eta_i & \sum x_i \zeta_i \\ \sum y_i \xi_i & \sum y_i \eta_i & \sum y_i \zeta_i \\ \sum z_i \xi_i & \sum z_i \eta_i & \sum z_i \zeta_i \end{bmatrix}^{-1} \begin{Bmatrix} 8x - \sum_i x_i f(\xi,\eta,\zeta) \\ 8y - \sum_i x_i f(\xi,\eta,\zeta) \\ 8z - \sum_i x_i f(\xi,\eta,\zeta) \end{Bmatrix} \quad \textbf{(5)}$$

where $f(\xi,\eta,\zeta) = 1 + \xi_i \eta_i \xi\eta + \xi_i \zeta_i \xi\zeta + \eta_i \xi_i \eta\xi + \xi_i \eta_i \zeta_i \xi\eta\zeta$ **(6)**

The direct iteration method is used to solve the non-linear system of equations (5) and it is shown that error in the computed values of ξ, η, ζ is a function of the number of iterations and the element angles. The rate of convergence is influenced by the magnitude of the largest element angle and for angles greater than 180^o, the solution does not converge to the correct values of ξ, η, ζ. In their implementation of the 2-D scheme, Yang, et. al. [30] calculate the values of ξ, η at every corner of the quadrilateral element to determine if there is significant distortion. If the computed values are not within 1% after 20 iterations, a remesh is triggered.

Oddy, et. al. [62] have proposed a distortion metric based on an invariant of the isoparametric transformation. In this method, the jacobian matrix is normalized to eliminate the effect of the element size. Hence,

$$J'_{ij} = \frac{J_{ij}}{\left|J\right|^{1/n}} \tag{7}$$

where J_{ij} are the elements of the jacobian matrix, J, and n is the number of dimensions. An analogy is drawn between element distortion and strain to compute a distortion metric. If $C_{ij} = J'_{ki} J'_{kj}$, then using the analogy to compute the deviatoric strains,

$$e_{ij} = \frac{1}{2}\left(C_{ij} - \frac{1}{n} C_{kk} \delta_{ij} \right) \tag{8}$$

and the second invariant of this tensor is given by:

$$J_2 = \frac{1}{2} e_{ij} e_{ij}$$

$$J_2 = \frac{1}{8}\left(C_{ij} C_{ij} - \frac{1}{n} \left(C_{kk}\right)^2 \right) \tag{9}$$

The factor 8 in the denominator is dropped since it only serves to scale the values obtained. The distortion metric then becomes:

$$D = C_{ij} C_{ij} - \frac{1}{n} \left(C_{kk}\right)^2 \tag{10}$$

This distortion metric is a function of the terms of the jacobian to the fourth power. Hence, the distortion metric is a point-wise measure of the element distortion that grows

rapidly as distortion increases. The user could set a limit for allowable element distortion, which could then be used to trigger a remesh.

Tsukerman [77, 95] derives a maximal eigen-value condition, which shows that it is the maximum eigen-value of the element stiffness matrix that characterizes the impact of the element shape on the energy norm of the error of the finite element approximation. It is shown that the maximum eigen-value of the element stiffness matrix (scaled by the volume of the element) characterizes the geometric "flatness" of the tetrahedral element and tends to increase exponentially as the element becomes flat (zero volume). Hence, it can be used as an *a priori* measure of element distortion.

Empirical rules governing acceptable levels of element distortion are also common. Baehmann, et. al. [15, 16], in their implementation of a 2-D automated modeling system, use 4-node quadrilateral elements to represent the workpiece during finite element computations. The element angles are monitored and a remesh is triggered when an interior angle falls outside the 20^{o}-160^{o} range. This range is chosen through the evaluation of the ability of the mesh to accurately predict solutions in test cases for which an analytical solution exists. This also avoids the case of a physically unrealistic mesh but places fairly strong limit on the amount of deformation the mesh can undergo.

Habraken, et. al. [25, 26] use four different criteria to measure element distortion measures for quadrilateral elements. They are: (i) the criterion of triangles, (ii) the corner angle criterion, (iii) the mid-side node criterion and (iv) the slenderness criterion. In the criterion of triangles, each quadrilateral element is divided into four triangles. The

percent change in the area of each of the triangles (before and after deformation) is monitored. In the corner angle criterion, the percent change in the corner angles (before and after deformation) is monitored. Habraken et. al. [25] recommend a value of 66% as the limiting value which requires the interior angles for a square/rectangle (90^{o}) to be in the 30^{o}-150^{o} range. Though the corner angle criterion is similar to criterion proposed by Baehmann, et. al. [15, 16], the use of a percentage value for allowable change can present problems. It is likely that elements in the starting mesh (especially after a remesh) will consist of elements with interior angles larger than 90^{o}. For example, if the largest interior angle for an element is 120^{o}, the acceptable range changes to 40^{o}-180^{o}. The mid-side node criterion and the slenderness criteria are defined only to be applicable to higher order isoparametric elements. The mid-side node criterion checks to ensure that the mid-side node is within the middle third of the element side. In the slenderness criterion, element slenderness (ratio of length of the longest median to shortest distance between a mid-side node and that median before and after deformation) is monitored.

In addition to the above element distortion measures, Habraken, et. al. [25] use two global indicators to monitor overall levels of mesh distortion. The idea here is to trigger a remesh only after detecting a sufficient number of distorted elements as opposed to other techniques [15, 16, 30] where a remesh may be triggered when one of the elements fails the distortion criterion. These global indicators compute (i) the percentage of distorted elements and (ii) the percentage area of the distorted elements, where a distorted element is one for which any of the distortion measures explained above is outside its threshold range. In the implementation described in reference [25], the user has to define

the acceptable limits for η_1 and η_2. However, the advantage of using these global indicators to trigger a remesh is not clear. If an element is considered to be distorted when it fails any one of the above criteria, then not triggering a remesh when the first element fails one of the remeshing criteria can only result in increased solution error. There is no correlation presented between these indicators and solution results and Habraken, et. al. [25, 26] also do not provide any guidance for typical range of values to be used for these indicators.

In this thesis research, we use the determinant of the jacobian matrix as the distortion metric for three reasons. Firstly, this metric is built into every finite element solver and ensures basic element validity [75, 76]. The solver used in this thesis research, Antares™ [8, 9] requires the workpiece to be defined by straight-edged tetrahedral elements. Since only one integration point is needed (for straight-edged tetrahedrons) to integrate the approximating polynomial, the solver would exit immediately for a remesh when the element becomes invalid. The above discussion focuses on distortion metrics in isolation. However, in the forming simulation context, one issue that must be considered when developing a distortion metric is the effect of triggering a remesh on simulation accuracy due to the accumulation of rezoning errors. This issue is discussed further in section 2.4. Finally, on a more practical note, implementation of a distortion metric within typical commercial finite element solvers would be difficult considering the limited access offered for adding such capability.

2.3 Mesh Discretization Errors

Mesh discretization errors are defined as errors in the solution due to use of specific basis functions on individual finite elements. There are two procedures typically employed to identify regions where these errors exceed a desired threshold. The first technique is to use rigorous mathematical constructs to compute *a posteriori* estimates of the errors for a given discretization. The second approach is to use *error indicators* to determine regions where mesh refinement would be most useful. The use of standard C^0 elements results in a discontinuous solution field for the secondary variables. These discontinuities are likely to be reasonable indicators of solution error for linear elements [103, 105]. Hence, regions that show high inter-element jumps in these variables become candidate regions for mesh refinement. Parameters such as gradients in secondary field variables like strain rates, effective strain, etc. are also used as indicators [8-9, 16, 51].

A number of approaches to *a posteriori* error estimation are discussed in the literature and the reader is referred to [65-67] for an overview. These include interpolation methods, projection methods, element residual methods, etc. [67]. These estimates are typically computed for each element and used as a basis for mesh refinement before continuation of analysis. Properly applied, any of these techniques can be used to identify regions for mesh refinement so as to control the discretization errors.

Several implementations of error indication techniques to metal forming analysis have been documented in [25-29, 38, 68, 69]. Almost all of them use some form of the projection type error estimator and follow the basic approach proposed by Zienkiewicz

and Zhu [68, 69] where an error norm is computed based on the difference between the finite element field and an improved solution field computed using specifically constructed projections of the finite element solution. The key to the success of this approach is to carefully define the improved solution field such that it is always more accurate in the norm of interest than the direct finite element solution. The development of these procedures focuses on the secondary solution variables, for which the finite element solution is discontinuous between elements. The finite element values from super-convergent or typically more accurate points are used to define a field with higher order continuity, typically C^0, over a patch of elements. The original procedure [68] employed a global least squares fit to define the improved field, while the more recent procedure [69] performs a least square fit over patches of elements to define values at individual nodes.

Zienkiewicz, et. al. [28, 29, 38] have used variations of this approach in their implementation of adaptive remeshing schemes for modeling forming problems. In reference [29], Zienkiewicz, et. al. use a projection type error estimator to control remeshing in forming problems that are modeled with a flow formulation for viscoplastic materials. In their work, the error in the energy norm is defined using differences between the improved and finite element values of the deviatoric stress. The error in the energy norm is expressed as:

$$\left\|\underline{e}\right\|^2 = \int_{\Omega} \left(\underline{\tilde{s}} - \underline{\hat{s}}\right)^T \left(\mu \underline{D}\right)^{-1} \left(\underline{\tilde{s}} - \underline{\hat{s}}\right) d\Omega \qquad \textbf{(11)}$$

where $\underline{\tilde{s}}$ are the deviatoric stresses computed via the finite element analysis, $\underline{\hat{s}}$ are the improved values of the deviatoric stress, μ is the viscosity and $\underline{D}^{-1}$ is a diagonal matrix with values (2 2 2 1 1 1) obtained from the definition of the constitutive relationship [29]. The finite element values of deviatoric stress are discontinuous, so the improved values of the deviatoric stress are obtained through projection onto a continuous base via the smoothing process of reference [68]. The model is remeshed when the estimated percent error,

$$\eta = \frac{\|\underline{e}\|}{\|\underline{u}\|} \times 100\% \qquad \textbf{(12)}$$

exceeds a user specified maximum.

Habraken and Cescotto [25, 26] and Dyduch, et. al. [27] have also used similar approaches in their implementations of the 2-D remeshing schemes. The difference in their implementations is that for their error norm, they consider L_2 norms of errors in Cauchy stress, $\underline{\sigma}$, and rate of deformation, $\underline{D}$, which are expressed as:

$$\|\underline{e}_{\sigma}\|_{L2}^2 = \int_{\Omega} \left(\underline{\tilde{\sigma}} - \underline{\hat{\sigma}}\right)^T \left(\underline{\tilde{\sigma}} - \underline{\hat{\sigma}}\right) d\Omega \qquad \textbf{(13)}$$

$$\|\underline{e}_{D}\|_{L2}^2 = \int_{\Omega} \left(\underline{\tilde{D}} - \underline{\hat{D}}\right)^T \left(\underline{\tilde{D}} - \underline{\hat{D}}\right) d\Omega \qquad \textbf{(14)}$$

where $\tilde{\underline{\sigma}}$ and $\tilde{\underline{D}}$ are the stress and strain rate values computed via the finite element analysis, and $\hat{\underline{\sigma}}$ and $\hat{\underline{D}}$ are the improved values of stress and strain rate which are computed using the method described in [68].

Other techniques have also been proposed for use in adaptive analysis of metal forming problems. Kikuchi and Cheng present an error measure for 4-node quadrilateral elements based on interpolation error. Bass and Oden [70] examine several estimates of local error for triangular and quadrilateral elements used in modeling viscoplastic problems.

Error indication procedures are typically fast to evaluate and provide a useful technique to quickly identify regions for mesh refinement. Selection of a proper indicator (depending on the application) is key to its successful use. They present a qualitative measure of the solution error at different locations. Since errors are related to dissatisfaction of the governing equations and boundary conditions [104], the inter-element jump is a dissatisfaction of the governing (equilibrium, in our case) equation and contributes to the error. In the case of linear finite elements (used in this thesis), the second derivative of the functions used in the approximation is identically zero. Hence, the inter-element jumps become the error contributors when there are no body forces [103, 105].

Baehmann, et. al. [16] have used error indicators in their implementation of the 2-D automated modeling system. Inter-element jumps for total strain are used as the indicator to guide remeshing. The solution error for element i, E_i, is measured in the L_2 norm as:

$$e_i = \max\left|\varepsilon_i^p - \varepsilon_j^p\right| \quad j = 1,....N$$
$$\left\|E\right\|_i^2 = \int_{\Omega_i} e_i e_i d\Omega_i \qquad \textbf{(15)}$$

where ε_i^p is the effective plastic strain and N is the number of adjacent elements. The areas with the largest change (top 10%) in effective plastic strain are refined.

In this thesis research, we have also used an error indicator to indicate regions for mesh refinement. The error indicator is based on inter-element jumps in effective strain rate. This is based on the fact that the strain rate provides a snapshot of the workpiece deformation at the time of the remesh and hence, is a useful indicator of the ensuing deformation (after the remesh). This is in contrast to the use of effective strain as an indicator [16], which addresses the path dependent nature of the deformation process and may indicate a need for refinement in regions that have undergone substantial deformation and may not necessarily be subjected to more deformation as the simulation progresses. This is illustrated further in Figure 10, which shows the strain and strain rate contours for the deformed part in the late stages of the analysis of a valve forging process. The strain based criterion of Baehmann, et. al. [16] resulted in the identification of 3 points in the regions shown in Figure 10. The proposed strain-rate based criterion resulted in the identification of 137 points in the regions shown in Figure 10. The regions that need refinement are the regions where most of the deformation is taking place (in this case in the flash gutter). The inter-element strain rate based criterion seems to be a better indicator of these regions, especially in the latter stages of the forming process. It is to be noted that both identify the same regions in the initial stages of the forming

process simulation. Hence, the error indicator used in the thesis research (for linear displacement tetrahedral elements) is defined as:

$$E_i = \max\left|\dot{\bar{e}}_i^p - \dot{\bar{e}}_j^p\right| \quad j = 1,....N, \tag{16}$$

where $\dot{\bar{e}}_i^p$ is the effective plastic strain rate for element *i* and N is the number of adjacent tetrahedral elements (4 for an interior element).

Another variation with respect to the approach of Baehmann, et. al. [15, 16] is in the procedure to determine the candidate regions for refinement once the inter-element strain rate gradients are computed. It is common to expect a fairly large range for the strain rates (and for the inter-element gradients) as the workpiece undergoes deformation. The reason for this is that in a typical forming process involving complex shapes, a very small region of the workpiece is undergoing deformation at any given time. This is illustrated in Figure 11 showing the strain rate distribution in the workpiece (values from 0 to 72), for a hammer forging operation, when it needs to be remeshed. The criterion to select elements with gradients greater than (0.9 * Maximum gradient) [16] will not refine the mesh in all of the regions undergoing severe deformation. Hence, to indicate the regions for mesh refinement, we first compute the median value of the error indicator as:

$$E_{med} = \left(E_{i,\max} + E_{i,\min}\right) \times 0.5 \tag{17}$$

where $E_{i,\max}$ is the maximum value and $E_{i,\min}$ is the minimum value of the error indicator across all elements in the model. Elements with error greater than the median $\left(E_i \geq E_{med}\right)$ are flagged for refinement. Results using this approach to be presented in

chapter 6 demonstrate that this provides useful guidance to identify regions for mesh refinement. The refined mesh for the deformed workpiece in Figure 11 is shown in Figure 12. Note that the mesh on the symmetry surfaces reflects the desired mesh size specified by the user. Specifics on the implementation of the procedure are discussed in chapter 5.

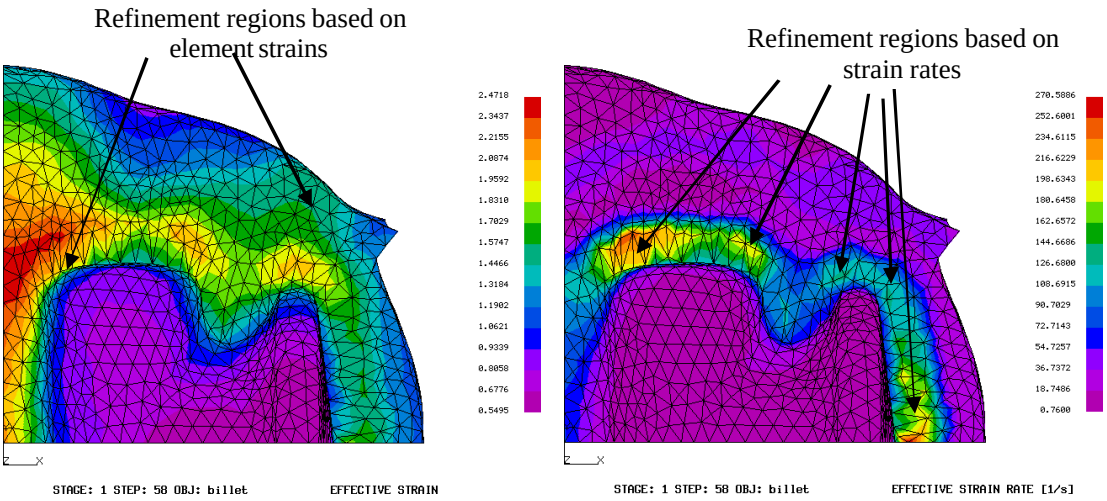


Figure 10: Strain and strain rate based mesh refinement criteria

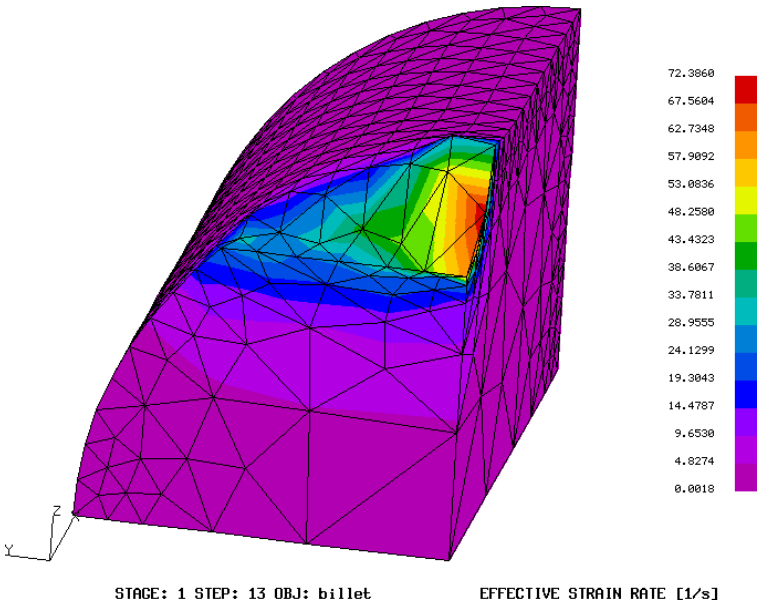


Figure 11: Strain rate distribution in the workpiece

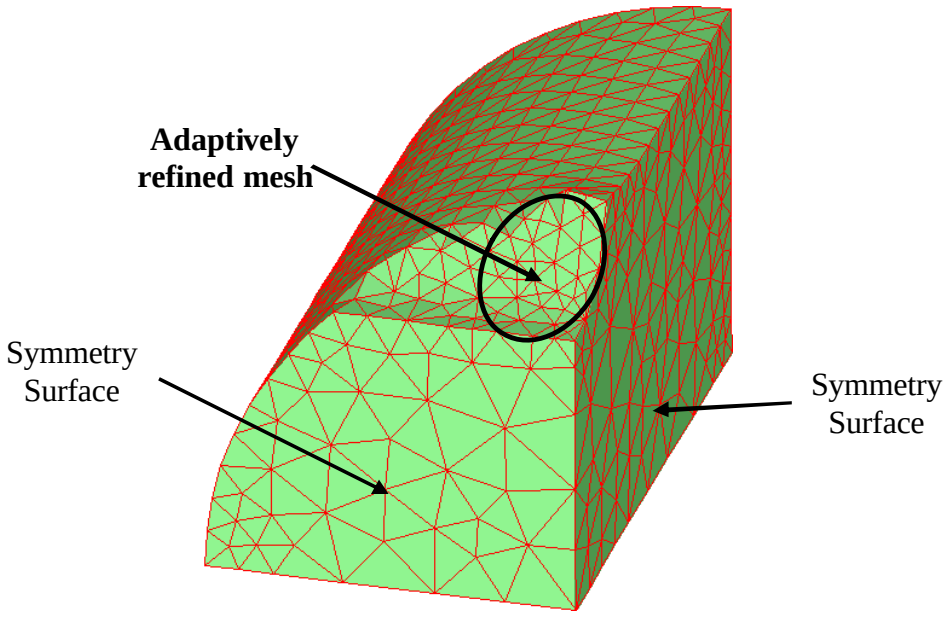


Figure 12: Mesh refinement in (red) regions of high strain rate

2.4 Mesh Rezoning Errors

An issue critical to the successful use of remeshing in history dependent solution procedures is the transfer of the appropriate solution variables between meshes. The specific requirements placed on the mapping procedures depend on the type of analysis performed [51]. For example, in an elastic-plastic analysis, the stress-state of all points within the plastic region must stay on the yield surface after mapping. In the modeling of bulk forming problems, element strains are to be mapped from the deformed (sending) onto the new (receiving) mesh. Errors in this mapping will alter the deformation path since plastic deformation is a history dependent process. These errors, termed mesh-rezoning errors, occur due to the fact that (i) the solution variables need to be interpolated and, (ii) the sending and receiving meshes may represent approximations of slightly

different geometric domains. The interpolation errors are present in any approach used [88-94] for transferring the solution variables. A careful analysis of the numerical method yields an expression in which the error is a function of a number of parameters including typically the element size to a power. This exponent defines the rate of convergence of the method. If the rate of convergence for the finite element method is of a specific order, we want to be sure to use a rezoning method that is of that order or higher. If the rezoning method has a lower order of convergence, it becomes the dominant error, thus dictating the accuracy of the results obtained. The problem of sending and receiving meshes representing slightly different domains is observed during remeshing when a geometric representation is developed for the deformed workpiece based on the analysis results and then discretized. A typical case illustrating mesh-rezoning errors is shown below. Strain distribution in the workpiece when the solver exits for a remesh are shown in Figure 13 and the results from the Antares™ solver with the new mesh (after the solution variables are mapped) after a small time-step of 1.0e-06 seconds are shown in Figure 14. Though the strain patterns appear alike, the maximum strain value in the two cases is different. In Figure 13, the maximum strain is 1.4244 where as it is 1.0701 in Figure 14.

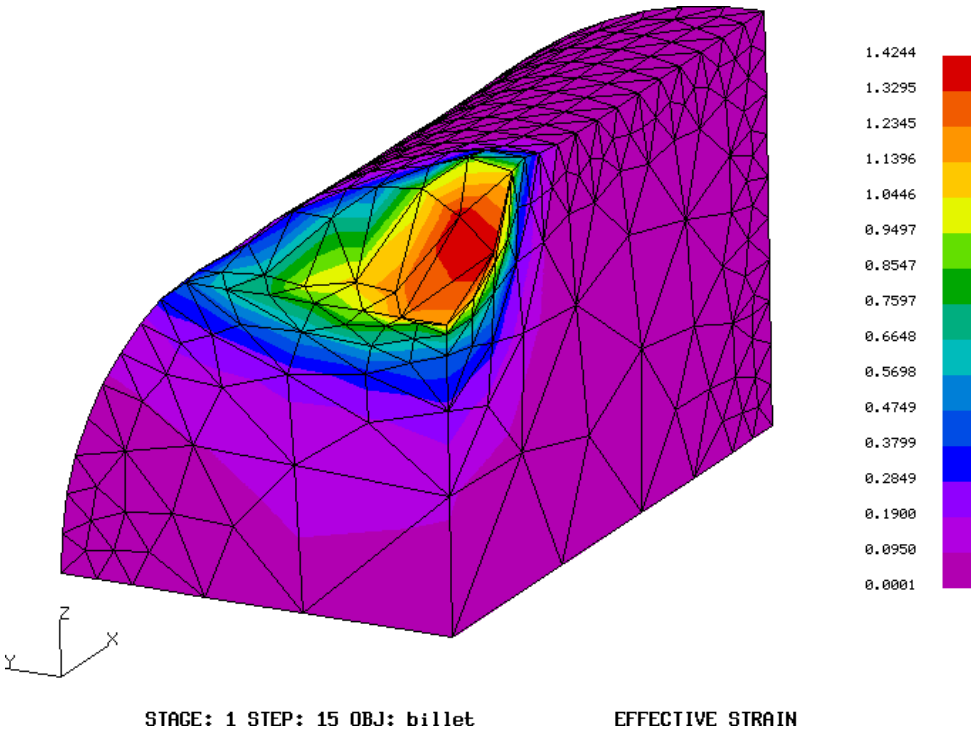


Figure 13: Strains in workpiece when it needs a remesh (in AntaresÔ)

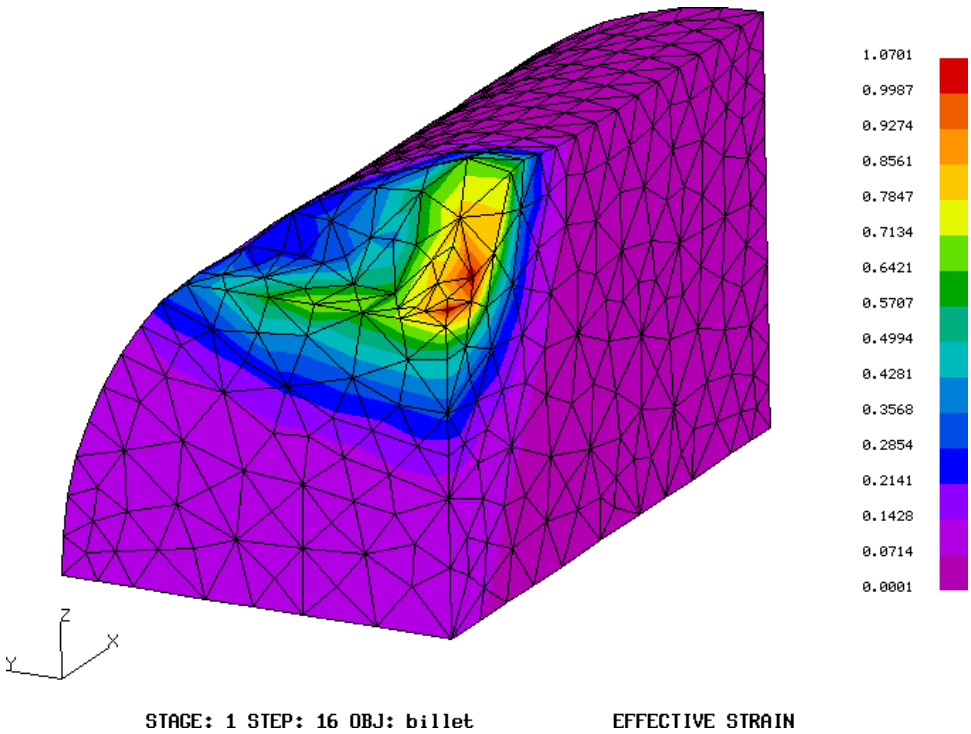


Figure 14: Strains in workpiece upon remesh (in AntaresÔ)

The variation in element strains in the workpiece before and after a remesh can potentially affect further material flow calculations. Consider, for example, the constitutive response of a strain-hardening material in such a scenario. The flow stress would be different when the strain is 1.4244 as opposed to the case when the strain is 1.0701. This difference in the constitutive response of the material (lower stress with the lower strain value) can affect the localized material flow computations. So, each remesh is likely to introduce such mesh rezoning errors. So, the number of remeshes necessary to simulate the entire process may influence simulation accuracy. Since the criteria to detect/control geometric approximation and element distortion errors trigger a remesh when an element exceeds the threshold, further research is needed to study the impact of remeshes on overall simulation accuracy. In this thesis research, since we use the sign of the determinant of the jacobian to monitor element distortion errors, only the die-overlap-monitoring criterion may result in more remeshes. Other than raising this issue as a valid concern, the thesis research does not address this issue further, leaving it for future work.

Chapter 3

AUTOMATING FORMING PROCESS MODELING – GEOMETRY UPDATE ISSUES

The application of finite element methods to problems where the domain evolves during the simulation introduces a number of complexities in the solution process that are not present in the solution for fixed domain problems. This chapter considers the geometry-based aspects of evolving geometry simulations, which include specification of the domain geometry and evolving it to reflect the simulation status. The automatic generation and control of finite element meshes is also an important issue in evolving geometry simulations since the mesh and the mesh control information associated with the mesh must evolve as the geometry evolves. This thesis research does not focus on mesh generation related issues since this aspect of the system is reasonably well developed [72, 79, 80, 106, 107]. For the purposes of this thesis research, it is assumed that an automatic mesh generator is available to discretize a given domain.

The perspective taken in this thesis research is that the finite element meshes used in a simulation are simply an idealized representation of the domain of the evolving geometry. It is this geometric representation that must play a central role in the simulation process. The framework implied by this perspective allows for the convenient discussion of the geometry-based issues associated with the simulation of evolving geometry problems. In addition, and more importantly, it forces a careful consideration of the influence of the finite element discretizations on the modeling process.

The next three sections discuss domain specification, evolution and discretization. This represents the "geometry update" and the "generate mesh" phase in the flow chart of Figure 1. A discussion on the requirements of the geometry update algorithm is presented followed by a discussion on possible approaches for model update. The implications of using one of the possible update schemes on workpiece volume is discussed with the intent of clearly identifying the volume change attributable to the update scheme. A classification scheme for the geometry update procedures is also presented. This leads to the identification of the selected approach implemented within our 3-D automated system. Specifics on the actual implementation scheme and the algorithms developed as part of this system are the focus of our discussion in chapter 4.

3.1 Domain Specification in Evolving Geometry Problems

In evolving geometry problems such as forging, extrusion, etc., the geometry of the model changes continuously during the simulation. The automation of evolving geometry problems must explicitly consider the definition and evolution of the domain and its numerical analysis discretizations during the simulation. The interactions between the evolving geometric domain description and its numerical discretization is complicated by the fact that the numerical discretizations are derived from the geometric description of the domain, while the analysis results of the discretized model dictate the evolution of the domain geometry [51].

The automatic generation of a numerical discretization, such as a finite element mesh, requires the existence of a complete and unique representation of that domain [79, 80].

Initial research efforts focused on two-manifold representations referred to as solid models [82, 83]. More recently, the domain coverage of complete and unique representations has been extended to cover more general combination of solids, surfaces and curves referred to as non-manifold representations [84, 85].

It is typically convenient in the description of a geometric domain to consider the hierarchy of entities that define and bound the domain of interest. The abstraction of topology and topological adjacencies provide an effective means to identify the 0- through 3-dimensional entities that constitute a geometric domain and allows separate consideration of shape description and entity adjacencies [86, 101, 102]. The use of topological entities of vertices, edges, faces and regions also provides a convenient means to uniquely specify the attribute information of loads, material properties and boundary conditions needed to complete the description of the problem to be analyzed [86].

Starting with a complete description of the original domain, it is possible to automatically generate the mesh required to begin an evolving geometry simulation. In those cases where multiple finite element discretizations are needed, it is necessary to use the results of the simulation to generate an updated geometric representation. Before investigating the possible approaches to update the geometric representation, we need to discuss the key issues that the geometry update procedure must address. These are presented in the next section.

3.2 Geometry Update – Key Issues

The objective of the geometry update procedure is to take a finite element mesh representing the current state of the workpiece, and evolve the geometric representation of the workpiece to represent its current deformed shape. This geometric representation is then discretized to generate a mesh consisting of valid, high quality mesh (in our case, consisting of 4-node tetrahedral elements). The geometry update procedure has to ensure the following, each of which is addressed in greater detail below:

1. a good approximation to the geometry of the deformed workpiece that maintains workpiece volume within acceptable limits,

2. consistent definition of the boundary condition surfaces to allow continuation of analysis, and,

3. consistent geometric interpretation of contact between objects with respect to the solver and the geometry update procedure.

A poor approximation to the deformed workpiece geometry and inability to maintain workpiece volume can lead to gross errors in the simulation. Maintaining volume refers to both the enclosed volume of the workpiece as well as its distribution. For an acceptable updated geometric representation, it is imperative that the contact portions of the updated workpiece geometry inherit appropriate portions of the die geometry. This will ensure that the discretization of the evolved geometric representation provided to continue the analysis will result in the imposition of the same loading conditions as in the

deformed workpiece. The approaches used to represent the contact portion of the workpiece boundary by various researchers and in this thesis research are discussed later in this chapter.

To consider how the free surface should be represented, we need to understand the continuity requirements for the free surface. For a single material with a continuous forcing function and boundary conditions, the free surface will be smooth [81]. The methodology used to represent the free surface in the updated workpiece representation plays an important role in its ability to maintain workpiece volume. The approaches used to represent the free portion of the workpiece boundary by various researchers and in this thesis research are discussed later in this chapter.

In order to continue the analysis, the new workpiece mesh model must inherit the appropriate boundary conditions. In finite element analysis solvers, these are typically imposed on finite element faces. Boundary condition information that needs to be *consistently* transferred across remeshes includes symmetry, contact, pressure, and velocity. Improper transfer of boundary conditions onto the new workpiece mesh will alter the deformation behavior in the ensuing computations.

Consistent interpretation of geometric contact between the geometry update procedure and the finite element analysis solver is maintained by making the same decision on the contact status of a node. In order to have consistent interpretation/definition of contact and free regions of the workpiece, we need to enforce consistent logic in the solver and the geometry update module to perform the geometric computations for contact. This

also implies that the geometry update procedure must use the same representation of the dies (as the solver). A typical problem, encountered when different representations are used, is illustrated in Figure 15. Figure 15(a) shows the output from the solver with workpiece nodes in contact with the die mesh (represented by straight-edged elements). Geometry update (using the true die geometry) and the resulting discretization of the contact region of Figure 15(a) would result in nodes that are in contact with the die geometry, as shown in Figure 15(b). However, upon continuation of analysis, these nodes are treated as free surface nodes, as shown in Figure 15(b). This different interpretation of the contact region could potentially affect the simulation results (loads, stresses, etc.) and the workpiece volume.

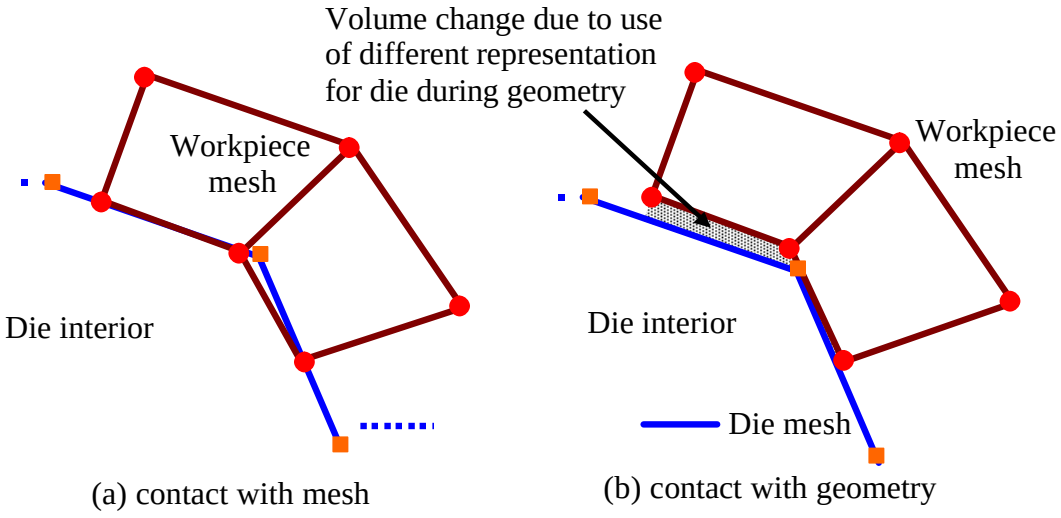


Figure 15: Contact with die mesh in solver and die geometry in update

3.3 Geometry Update – Possible Approaches

The model update procedure has to handle the interactions between the representations of the dies and the workpiece in a *consistent* manner, i.e., all procedures must make the same geometric decisions. In this section, we will define conditions for consistent geometry update, i.e., the conditions that must be ensured by the geometry update procedure given the requirements of the analysis procedure used. The ideal conditions are first defined followed by a description of the possible approaches (and their consistency conditions) for evolving the geometric representation of the workpiece.

The ideal scenario would be to evolve the geometric representations of the die-workpiece model ensuring no interference in the geometric representations of the dies and the workpiece (since they are two distinct objects). Using the notations W and D to represent the workpiece and die representations respectively, the *consistency condition* for the geometry update procedure can therefore be stated as

$$W_R \cap D_R = 0 \qquad \textbf{(18)}$$

where R is the representation of choice (for the die and the workpiece) and can take the value of G when the geometric representation is being used or M when the mesh representation is used. Since the mesh model is an abstraction of the true geometry, the *consistency condition* for geometry update can be stated as:

$$W_G \cap D_G = 0 \qquad \textbf{(19)}$$

This condition implies that the workpiece geometry must be evolved ensuring that there is no interference between the geometric representations for the die and workpiece objects. In the context of an implementation scheme, the consistency condition must reflect the level of the geometric approximation used. Since we have two types of objects, viz., the dies and the workpiece and two possible representations for each of them (geometry or mesh), there are four different methods of model update. In each of these cases, to avoid situations illustrated in Figure 15, it is assumed that the same representation (mesh or geometry) is used for the die during the analysis and geometry update. Even though the discussion below also applies to cases where higher order elements are used to represent an object, we will assume (for illustration purposes only) that the mesh model consists of straight-edged elements. Each of the methods and its governing consistency condition are discussed below.

3.3.1 Geometry Update - Die mesh and workpiece mesh

In this scheme, the mesh models for the dies and the workpiece would be used to represent the geometry of the objects during the analysis and geometry update. A schematic illustration of this scheme for 2-D applications is shown in Figure 16. The consistency condition when die and workpiece meshes are used for model update would be:

$$W_M \cap D_M = 0 \qquad \textbf{(20)}$$

However, due to geometric approximation, as shown in Figure 16, it is not possible to maintain a fully consistent approximation unless the individual mesh entities on the

boundary of the die and workpiece are exactly aligned. For practical use of this method, the consistency condition has to be relaxed to ensure that workpiece nodes do not penetrate the die boundary. This condition can be stated as:

$$W_M^0 \cap D_M = 0 \qquad \textbf{(21)}$$

where W_M^0 indicate the workpiece nodes. The use of this combination is common in most commercial 3-D finite element codes [8-12] and in most automated 3-D systems described in the literature [43-50]. During model update, the contact portion of the workpiece inherits the appropriate portion of the die mesh and the appropriate portion of the faceted model of the workpiece (or a slight variation) is used to define the free surface. This is then re-discretized while the die mesh model is used again to continue the simulation.

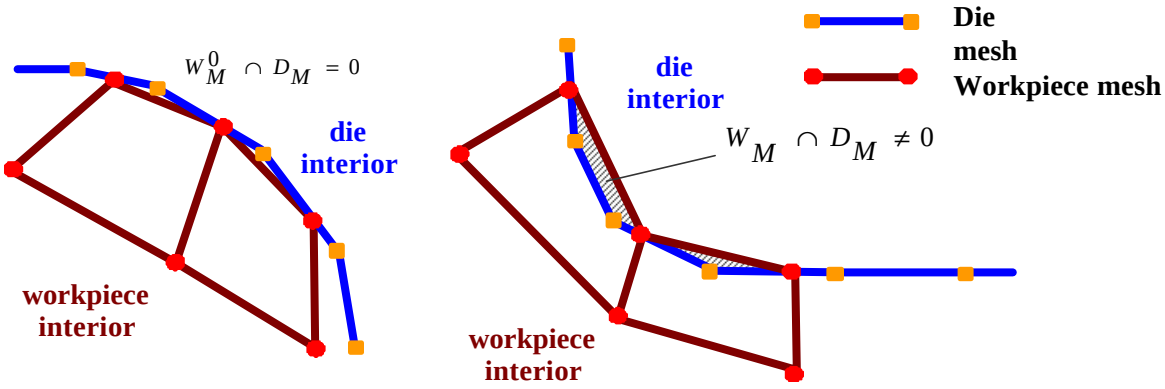


Figure 16: Consistent model update – die and workpiece mesh

This scheme is also used in several 2-D automated forming systems described in the literature. Schneiders, et. al. [22] are of the opinion that when the boundary of the mesh is defined as the geometry to be meshed, the number of elements in the new mesh grows.

This follows from the fact that the boundary of the new mesh will contain nodes representing the boundary of the old mesh and several inserted nodes. This could lead to problems after several meshes and hence, they use a simplified boundary for generating a new mesh and then project the new mesh onto the old boundary. This approach has been demonstrated in 2-D modeling of forming problems. Lee, et. al. [23] use the boundary of the workpiece as the basis to generate the new discretization and have developed a technique termed as "automatic renoding". In this approach, the workpiece elements likely to undergo mesh degeneracy and geometric interference with the die (elements with at least one edge on the boundary) are monitored. When these elements violate the interference or the degeneracy criterion, the element configuration is changed. Hence, a four node quadrilateral element violating the interference criterion may be changed to a five-node element to account for the proper shape of the die. This is defined as "renoding" and the shape functions are changed to reflect the new element configuration. The deformed workpiece mesh with these "renoded" elements forms the updated workpiece representation. Lee, et. al. [23] have used this technique to demonstrate its application to 2-D and 3-D forming operations. However, application of this technique to model forming processes involving complex workpiece geometries that undergo severe deformation would be very difficult since element distortion would eventually necessitate a remesh of the workpiece. Successful procedures for automated 2-D modeling of forming problems have been developed by Yukawa, et. al. [19], Tezuka [20], Park and Hwang [21] using the boundary of the deformed workpiece as the definition of the geometry to be meshed. Some more relevant references that use similar procedures for

remeshing of the workpiece in 2-D modeling of forming problems have also been listed [28-38].

Coupez, et. al. [39-42] describe an automated procedure for 3-D analyses. The boundary of the workpiece is defined as a fine mesh of linear triangular elements. This is used to generate a coarse, quadratic surface mesh that is used as the boundary of the workpiece mesh for analysis. From the analysis results, the geometric definition of the workpiece (fine mesh of linear triangles) is deduced with the goal of controlling the variation in the volume of the workpiece to less than a user-specified value. The technique has been applied to model the forging of a tripod. Tack, et. al. [43] and Tekkaya et. al [44-47] also use the boundary of the deformed workpiece mesh as the basis for generating the new discretization. In this approach, termed as the grid based approach, the characteristic points on the boundary are first determined and then used to define the updated (simplified) geometry for the workpiece.

Commercial special-purpose modeling systems such as Deform™ [10] and Antares™ [8, 9] require the dies to be represented by their mesh models and not their true geometric (CAD) models. Both of these codes use the deformed workpiece mesh as the basis to generate the new mesh. General-purpose commercial FEA codes such as Dyna™ [11], Abaqus™ [12] provide the capability for large deformation plasticity analysis but do not provide any automatic remeshing capability.

3.3.2 Geometry Update - Die mesh and workpiece geometry

In this method, the dies would be represented by their mesh models and the workpiece would be represented by a geometric representation generated based on the mesh data from the solver. The consistency condition when die mesh and workpiece geometry are used for model update would be:

$$W_G \cap D_M = 0 \qquad \textbf{(22)}$$

During model update, the contact portion of the workpiece would inherit the appropriate portion of the die mesh. The free portion of the workpiece boundary would be represented by a smooth surface. The interactions between the workpiece and die representations in the contact region would be the same as in the previous case shown in Figure 16. Also, as shown in Figure 17, overlap between the die mesh and the workpiece geometry must be avoided in convex regions of the workpiece (in close proximity of die boundary). For practical use of this method, the consistency condition has to be relaxed to ensure that die nodes do not penetrate the workpiece mesh or the geometry. These conditions can be stated as:

$$W_G \cap D_M^0 = 0, \qquad W_M \cap D_M^0 = 0 \qquad \textbf{(23)}$$

Zhu, et. al. [24], Habraken, et. al. [25] and Dyduch, et. al. [26, 27] have implemented automated 2-D geometry update schemes where the free surface is represented with a cubic spline and the portion of the workpiece that is in contact inherits the appropriate portion of the die mesh.

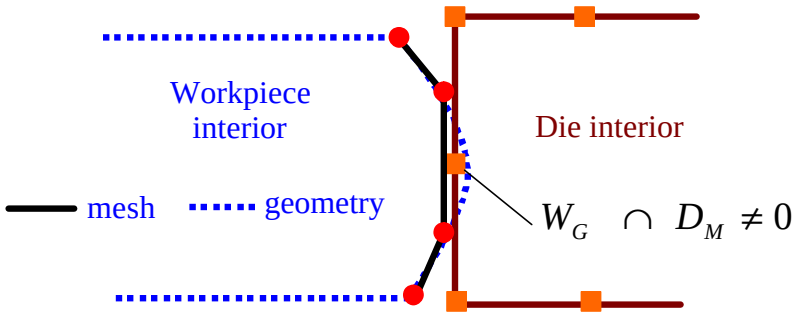


Figure 17: Consistent model update – die mesh and workpiece geometry

3.3.3 Geometry Update - Die geometry and workpiece mesh

In this method, the dies would be represented by their geometric representation and the mesh model would represent the workpiece. A schematic illustration of this scheme for 2-D applications is shown in Figure 18. The consistency condition when die geometry and workpiece mesh are used for model update is:

$$W_M \cap D_G = 0 \qquad \textbf{(24)}$$

However, as shown in Figure 18, it is not possible to avoid overlap between the workpiece mesh and the die geometry and the consistency condition is violated in the convex regions of the die. For practical use of this method, the consistency condition has to be relaxed to ensure that workpiece nodes do not penetrate the die boundary. This condition can be stated as:

$$W_M^0 \cap D_G = 0 \qquad \textbf{(25)}$$

During model update, the contact portion of the workpiece inherits the appropriate portion of the die geometry. A faceted model represents the free surface of the updated workpiece geometry.

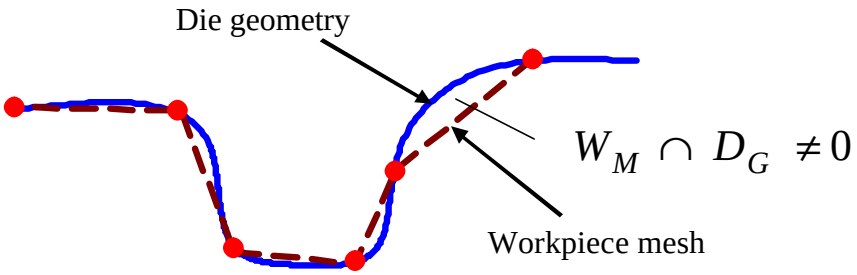


Figure 18: Consistent model update – die geometry and workpiece mesh

3.3.4 Geometry Update - Die and workpiece geometry

In this method, geometric models represent dies and the workpiece. The consistency condition when die and workpiece geometries are used for model update is:

$$W_G \cap D_G = 0 \qquad \textbf{(26)}$$

During model update, the contact portion of the workpiece would inherit the appropriate portion of the die geometry and a smooth surface represents the workpiece free surface. Baehmann, et. al. [15, 16] have developed an automated system for 2-D modeling of forming operations based on this concept. The system provides the capability for geometry based problem specification within the framework of a 2-D geometric modeling system. The updated model is represented with a cubic spline through the nodes on the free surface of the workpiece and the appropriate portion of the die

geometry (facilitated via modification of the topology of the die model in a non-manifold representational scheme) for the contact regions. Specific techniques have been incorporated to ensure that there is no overlap between the billet and die geometries. Cheng [17, 18] has also developed techniques for 2-D modeling that use cubic splines to represent the free surface and the die geometry to represent the contact regions.

3.4 Implications on Workpiece Volume

The intent of the discussion in this section is to qualify the impact of the model update procedure on total workpiece volume. Note that we assume the workpiece volume is defined as the volume enclosed by the mesh model representing the workpiece during the analysis (since available solvers only see the mesh model). We will first discuss the factors affecting the workpiece volume during the simulation and then identify ones that are attributable to the geometry update procedure used in the automated system. The volume change during the simulation, V_s, can be attributed to the following components:

1. Volume change during the analysis, V_a,
2. Volume change due to die curvature, V_c, and,
3. Volume change due to the approximation of the free surface, V_f,

The change in the workpiece volume during the entire simulation, V_s, is expressed as:

$$V_s = V_a + V_c + V_f \qquad \textbf{(27)}$$

The geometry update procedure used within the (updated Lagrangian) finite element procedure can result in change of workpiece volume with the change typically being a loss of volume. This represents workpiece *volume change during analysis* (V_a) and Kobayashi, et. al. [1] discuss this in further detail. Other than identifying volume loss during analysis as a potential source of workpiece volume change, the thesis research does not address this issue further.

The bulk of the change in workpiece volume during geometry update is likely to result from enforcing the *relaxed* consistency constraints (as the one in equation 25) in the contact regions. This represents the *volume change due to die curvature*, V_c, in equation 27. To illustrate this further, consider the situation (die geometry and workpiece mesh) illustrated in Figure 19, when a contact edge (both end points in contact with the same die) is to be split in the process of generating the new discretization. The location of the new node is adjusted from the mid-point of the original finite element edge to the die surface (if the mid-point lies inside the die) to enforce the relaxed consistency condition (equation 25). This impacts the local mesh volume and as shown in Figure 19, will result in a lower local volume in convex regions of the die. Note that the mid-point will lie outside the die volume in concave regions of the die and hence will be treated as a node on the free surface of the workpiece. It is to be noted that geometries with sharp transitions typically present significant difficulty for remeshing and can cause significant local volume changes and hence, contribute to a higher V_c. Volume change due to die curvature can be reduced by:

(a) using a smaller value of average mesh size in high curvature areas (for the workpiece and if necessary, the dies) and near contact regions during remeshing, and,

(b) implementing geometric checks within the finite element solver that would trigger a remesh (for example, when the mid-point of a contact edge is more than some distance tolerance within the die).

These options increase the computational effort but will result in improved accuracy. The geometric checks referred to in option (b) are implemented via the die-overlap monitoring algorithm referred to in chapter 2.

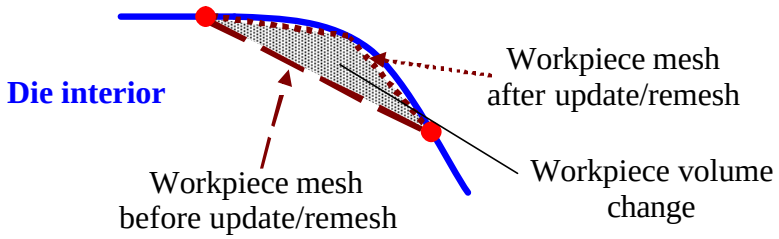


Figure 19: Impact on workpiece volume due to relaxation of consistency constraints

The true volume change that can be attributed to the model update procedure is the change in volume due to approximation of the workpiece free surface. This represents *volume change due to free surface curvature* (V_f) and depends on how the model update procedure handles the workpiece free surface. The magnitude of V_f is lower when the discretization on the free portion of the workpiece boundary reflects a smooth boundary as illustrated in Figure 6. This concept, in fact, has been used in the development of several 2-D modeling systems [15-17] where the free surface is represented by cubic

splines. Unfortunately, in the 3-D modeling world, representing a model with higher order surfaces generated automatically from an arbitrarily distributed set of connected points (nodes/elements on the boundary of the workpiece) is an extremely difficult task even in the most sophisticated commercial software systems. In addition, even when it is possible, it is manually done. Hence, in this thesis research, the workpiece free surface is represented by sets of piecewise smooth surfaces defined over a localized domains. Details of the implementation are provided in chapter 5.

3.5 Classification of Model Update Approaches

Algorithms used to update the geometry of the deformed workpiece can be classified into the following three categories:

1. Evolving geometry method.
2. Mesh based method.
3. Hybrid or the combination method.

Details and characteristics on each of these methods are given below.

3.5.1 Evolving Geometry Method

In this method, the geometry for the dies and the workpiece is stored as a single geometric model that is *evolved* based on results from the solver. The procedure ensures that portions of the workpiece boundary that are in contact with a die inherit the appropriate portions of the die geometry. This is accomplished by tracking the contact

boundaries as they evolve during the simulation, which in turn dictates updates to the topology of the die. The boundary conditions for the problem are defined as attributes on the geometric model and hence, automatically inherited by the updated representation of the workpiece. The free surface of the workpiece is typically represented with higher order (at least C^1 continuous) geometric entities in an attempt to mimic the smooth surface of the workpiece boundary.

Among the automated systems described in published literature and discussed earlier, Baehmann, et. al. [15, 16] and Cheng, et. al. [17, 18] have implemented evolving geometry schemes to automate 2-D forming analysis. Complexities of the geometric interactions and computations make it extremely difficult to implement this approach for 3-D modeling and none of the published articles describing 3-D modeling systems use this approach.

3.5.2 Mesh Based Method

In this approach, mesh models represent die and workpiece geometries. The boundary of the deformed mesh is assumed to represent the geometry of the workpiece. Since the new mesh is generated on a faceted representation, proper care has to be taken to:

(i) preserve workpiece features,

(ii) minimize volume loss, and,

(iii) ensure proper transfer/inheritance of analysis attribute information.

The use of discretized models as a basis for generation of the new workpiece mesh results in the introduction of an additional level of approximation. Hence, workpiece nodes that are identified as being in contact with a die are forced (geometrically) to lie on the boundary of the die mesh (rather than the die geometry). Since the modeling of a typical forming process involves a number of remeshing steps, the error (in the representation of the workpiece volume) can accumulate and introduce inaccuracies in the simulation. Using a fine mesh to represent the die boundary can result in a better approximation of the contact surface. However, the change in workpiece volume, due to the fact that the free surface discretization does not reflect a smooth boundary, cannot be eliminated from the simulation.

There have been several implementations of the mesh-based method described in the literature and summarized earlier. To recap, Schneiders, et. al. [22], Lee, et. al. [23], Habraken, et. al. [25] , Barata Marques, et. al. [35], and Dyduch, et. al. [26, 27] have used mesh based schemes in their automation of 2-D forming analysis. Coupez, et. al. [39-42], Tack, et. al. [43], Tekkaya, et. al. [44-47], Lee and Yang [48], Yoon and Yang [49-50] have used mesh based schemes in their automation of 3-D forming analysis.

3.5.3 Hybrid or the Combination Method

In this method, a combination of the above two approaches is used to update the workpiece geometry. A typical scenario would be to use the faceted boundary of the deformed workpiece as its updated geometry and discretize it. This mesh is appropriately modified so that nodes in the contact regions lie on the true die geometry (rather than the

die mesh in the mesh based method). Hence, the scheme would provide us with the same accuracy (in the representation of the workpiece mesh model) as the evolving geometry method in the contact regions. In this method, the discretization on the free surface of the workpiece is generated in the same manner as in the mesh-based method. This approach, especially for 3-D applications, appears to be a pragmatic approach considering the fact that available commercial geometric modeling software does not support the automated generation of a solid model (with smooth surfaces) from a discrete representation. The implementation constraints are essentially the same as the ones mentioned for the two previous methods. The procedure can be improved if the discretization on the free surface can be compensated so that it reflects a discretization on a smooth higher order surface defining the workpiece boundary. This would involve relocating new nodes (not on the free surface of the original workpiece mesh) so that the resulting discretization reflects the curvature of the workpiece free surface. Curvature compensation of the free surface nodes can provide a similar level of accuracy in terms of volume control as the evolving geometry method. This hybrid method has been implemented in this thesis research and details of the implementation are provided in chapters 4 and 5.

Chapter 4

THE HYBRID METHOD – CONSISTENT HANDLING OF DIE GEOMETRY

The overall approach to automate the generation of new meshes to represent the deformed workpiece is referred to as a hybrid method [87] due to its use of both model and mesh information. A brief overview of the hybrid model update process is first presented. In the hybrid method, representation and consistent handling of the geometry of the dies is a key requirement. Published literature and commercial software codes dealing with 3-D forming analysis [8-10, 39-50] do not address the issue of using the geometric representation of the die during the analysis. In this chapter, algorithms to integrate the finite element solver and the geometry update procedures with a geometric modeler that houses the die-workpiece geometry information are given. Specifics of the procedures used to generate the new mesh representing the boundary of the deformed workpiece are discussed in the next chapter.

4.1 The Hybrid Method – An Overview

From the discussion on the possible approaches for geometry update and their classification presented in the previous chapter, it is clear that the evolving geometry method is the ideal method for implementation. The major hurdle in extending the evolving geometry method to 3-D is in dealing with complex geometric interactions (such as the ones in determining die contact) and in the construction of a 3-D geometric

representation for the deformed workpiece using its mesh model. It is to be noted that computations performed on geometric models can result in a significant increase in compute time (compared to the use of mesh models to represent die geometry). The central theme of the hybrid method is to select specific features of the evolving geometry and the mesh-based methods and combine and extend them to get results comparable to the evolving geometry method.

In the hybrid method, geometric models are used to represent dies during analysis and remeshing for better capture and representation of the evolution of die-workpiece contact. Representation of the die geometry as a geometric model can be accomplished by using a geometric modeler such as a commercial CAD system. The commercially available modelers allow representation of complex geometries, can perform geometric computations and answer geometric interrogations on any geometry they control. The choice of the geometric modeler and reasoning for its selection are given in section 4.3.

In order to use the die geometry in the finite element computations, the contact algorithm in the solver is modified to determine contact with the true geometry of the die rather than its discretization. Though finite element solvers use different techniques to determine contact, the need for geometric checks is common to all methods. This typically involves providing (for nodes inside/on the die boundary) the location of the closest point and the normal at that point. The proximity/containment check implemented in this work is discussed in section 4.4.1. A die overlap monitoring procedure has been implemented to control/monitor geometric approximation errors (section 4.4.2) and triggers a remesh, if necessary. Results from the solver are written out

to a text file to enable workpiece geometry update and remeshing. Consistent interpretation of geometric contact during analysis and remeshing is ensured by use of identical procedures to perform the geometric checks in both instances.

The workpiece is not defined as a standard geometric model since the geometric representations used by the geometric modeling systems are not capable of effectively tracking the type of shape changes involved with forming problems. Coupez, et. al. [39-42], Tack, et. al. [43], Tekkaya, et. al. [44-47], Lee and Yang [48], Yoon and Yang [49-50] and commercial codes [8-10] have used mesh-based schemes in their automation of 3-D forming analysis. In this thesis research, a new geometry update and remeshing procedure has been developed [71, 73, 87] specifically to address the limitations of the typical mesh-based procedure with respect to the representation of the deformed workpiece. A specific step forward is to introduce a curvature capturing method (section 5.8) that will place new nodes on a curved local representation thus introducing improved local geometry control. Comparisons of this procedure with published literature [39-50] are provided in Chapter 5. The procedure, briefly explained, uses mesh manipulation and procedures to update the geometry of the deformed workpiece. Element faces on the boundary are used in the representation of the geometry of the workpiece. Boundary nodes representing the contact portion of the workpiece are constrained to be in contact with appropriate portions of the die geometry. Appropriate mesh refinement procedures are used to control the geometric approximation and discretization errors. The central idea of the free surface curvature compensation algorithm [87] is to create local C^1 surfaces based on the workpiece mesh model to capture the local curvature of the deformed workpiece geometry. New nodes representing the workpiece boundary are

then positioned appropriately on these local C^1 surfaces. The combination of the mesh-based method and free surface curvature compensation represents an effective method to track the evolving workpiece geometry.

In the hybrid method, use of the true die geometry, die-overlap monitoring, adaptive and look-ahead mesh refinement and free surface curvature compensation procedures help to preserve workpiece features and minimize the impact on total workpiece volume and its distribution. Analysis attribute information is properly transferred to the new mesh by using special data structures (explained in Chapter 5) that track and update the specific information during workpiece geometry update.

4.2 Implementation of the Hybrid Method

The procedure described here is applicable immaterial of the choice of the geometric modeler, finite element solver and the geometry update/remeshing procedure. The problem definition and set-up supports use of geometric models to represent the dies. The geometric model to represent the dies is created only once, before the simulation is begun. In the current work, the dies are considered to be rigid and hence updating the die models involves applying rigid body transformations to reflect the die motion. Elastic deformations in the die are considered small enough to justify not updating the geometric representation of the dies. Obviously, this process would not be applicable in cases where the die deflections are important. However, such cases are not common in the design of industrial forming processes.

Problem set-up for analysis includes defining the boundary conditions, material specifications, etc. as per the requirements of the finite element solver. Once the problem is setup, the finite element solver can simulate the deformation of the workpiece till one of the following situations is encountered: (a) the elements are too distorted to continue, (b) a distortion metric or a geometric approximation monitoring criterion triggers a remesh, (c) the iterative procedure in the finite element solver fails to converge, or, (d) the required die stroke has been reached and the simulation is complete.

For circumstances other than when the required die stroke has been reached, it is necessary to update the workpiece geometry and mesh it in order to continue the analysis. Upon successful completion of a time-step during finite element analysis, the solver writes out specific information used to drive workpiece geometry update and remeshing. Information written out includes the node coordinates and element connectivities, node contact data (if in contact then with which die), element results (strains, stresses, temperatures), boundary condition information (groups of elements with associated boundary condition information). The simulation is continued after the geometry update and remeshing procedure uses this information to build a new mesh for the deformed workpiece.

4.3 Selected Software Tools

The generic outline of the hybrid method described above is suitable for use with a number of software tools for analysis and model representation. However, the implementation details for the algorithms developed are affected by the choice of the

software tools used. In this research effort, commercial software has been used to perform the finite element analysis, generate meshes to fill the workpiece volume and store the 3-D geometric model of the dies. The commercial software tools used are:

1. The finite element solver: The Antares™ [8, 9] solver from UES Software Services, Dayton, Ohio has been used to analyze the deformation mechanics. It requires that the workpiece be represented by 4-node tetrahedral elements. The primary reason for its selection was that Antares™ is a commercial tool, available with an automatic mesh generator. The version provided for this thesis research also included access to the appropriate portions of the code to allow modification of the geometric checks within the contact algorithm, add checks to control geometric approximation errors and retrieve relevant results data.

2. The automatic mesh generator: The MeshCast™ mesh generator from UES Software Services, Annapolis, Maryland has been used to discretize the workpiece. This is included as a standard component of the Antares™ system [8, 9]. Given a surface mesh (of triangles) representing the boundary of the workpiece, this software uses the advancing front method [72, 79, 80, 106, 107] to fill the volume with straight-edged tetrahedrons. The geometry update and remeshing procedure developed in this thesis research has focused on the generating a surface that properly represents the boundary of the deformed workpiece. Hence, any of the volume meshing techniques that operate from a surface mesh are well suited to use with these procedures.

3. The CAD system: The Unigraphics™ CAD system [13, 14] has been used to house the geometric models for the dies. It was selected primarily because of the extensive

functionality provided to create, manipulate and query the geometric model. It is a widely used commercial tool and is very robust and extensive in its geometry creation functionality. The UG/Open™ API [13, 14] is used to extract the geometry and topology related information from the CAD models and to perform desired geometric computations needed during model update and analysis.

4.4 Using CAD Geometry in Analysis and Remeshing

The containment/proximity check, to determine if a point on the workpiece boundary lies inside, outside or on the die boundary, forms the basis of geometric checks required in contact computations, die-overlap monitoring and the look-ahead mesh refinement procedures performed during analysis and remeshing. The containment/proximity check implemented in this thesis research is described in section 4.4.1. The die-overlap monitoring algorithm, which uses this geometric check, is explained in section 4.4.2. Details of the CAD interactions needed for the look-ahead mesh refinement and the free surface compensation procedures are provided in chapter 5 as part of the discussion on geometry update and remeshing.

To facilitate integration of the solver and the CAD system, a data structure has been defined to track die geometry related information during the simulation. The data structure, shown in Figure 20, is updated as needed to ensure that it reflects the current state in the process simulation. For each die, its name, the Unigraphics™ object identifier (called the *tag)*, the bounding box information for the die and for each of its bounding faces, translation applied to get it to its initial position (at time, t = 0 for the

process simulation), die stroke and direction of translation are stored. Information on the faces and their bounding boxes is used in the containment/proximity procedure described below. Since the finite element solver and the geometry update procedure are two separate modules, handshaking between these modules is accomplished by writing the information in this data structure, at the end of execution of each of these modules, to a text file, *dietrans.dat*. The geometry update procedure reads this file written out by the solver and first synchronizes the die locations to reflect the die locations for the last converged solution. If there is any difference in the die stroke data for the converged solution (obtained from solver results) and the applied die stroke (defined in the file *dietrans.dat*), the die models need to be appropriately translated. These situations occur when the solution fails to converge within the user-defined number of iterations (but the dies have already been translated in order to perform the computations).

```
typedef struct GeoDieInfo
        {
           tag_t objTag;            /*  object identifier in Unigraphics  */
           int numFaces;            /*  number of faces bounding the object  */
           tag_t *faceTags;         /*  array of tags (object ids) of the faces  */
           int dieNum;              /*  die number in the process model  */
          double bBox[6];           /*  bounding box for the object  */
          double (*faceBox)[6];     /*  bounding box for each face */
          char (*faceName)[24];     /*  name for each face */
          char dieName[24];         /*  name for the object  */
          double posTrans[3];       /*  translation applied to position die before
                                        commencing the process simulation  */
          double simTrans;          /*  translation applied as die stroke  */
          double transDir[3];       /*  direction of translation (for stroke)  */
        } GeoDies;
```

Figure 20: Data structure to store solver and CAD model integration information

4.4.1 Geometric Proximity/Containment Check

There are a variety of methods that can be applied to determine if a node point is inside a die region. Since the amount of motion is easily bounded by a conservative multiple of the die stroke in that step, the points are first checked against an expanded bounding box of the region's faces instead of the standard, but somewhat expensive for curved domains, ray firing algorithm. If the given node point location lies inside or on the boundary of the expanded bounding box for a face, this face becomes a candidate face on which the closest point could potentially lie. The closest point on the candidate face and its distance from the given node point are computed using an iterative procedure such as the one described below. Repeating this procedure for each bounding face of the die, storing the minimum distance and the associated point on the die allows us to compute the closest point on the die and hence, the die penetration or proximity (if the node remains outside the die).

In modeling systems, the underlying geometry defining the face (topological entity) is typically a patch or a curve-bounded collection of points whose coordinates are given by a continuous, two parameter, single-valued mathematical functions [126, 127]. The parametric variables (u, w) are constrained to the interval $u, w \in [0, 1]$. In Figure 21, $\mathbf{q}$ is the location vector of the node point and $\mathbf{p}$ the location vector of the closest point on the surface. Hence, the minimum distance between $\mathbf{p}$ and the surface is $|\mathbf{p} - \mathbf{q}|$ along the surface normal direction ($\mathbf{p} - \mathbf{q}$). The normal vector at a point, whose location is known in parametric space, is also obtained by taking a cross product of the two tangent vectors ($\mathbf{p}^{\mathbf{u}}$ and $\mathbf{p}^{\mathbf{w}}$ with respect to the u and w parameters) at that point [126, 127]. Hence, we

must have the cross product $(\mathbf{p} - \mathbf{q}) \times (\mathbf{p}^u \times \mathbf{p}^w) \sim 0$ (within a small tolerance). The tolerance value used is typically the model tolerance or the minimum distance between points for the modeler to consider them as non-coincident (0.001 inches in Unigraphics™). An iterative procedure such as the Newton-Raphson method is used to determine **p**. The specific geometric computations to determine the closest point on a face are performed using the UG/Open™ API.

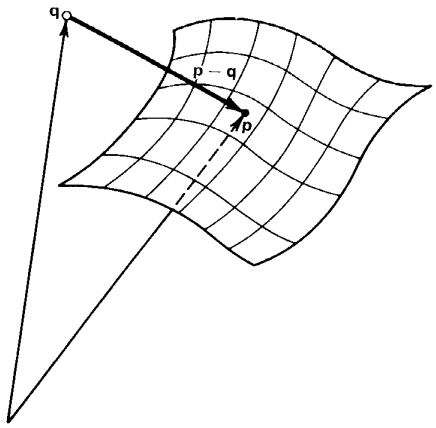


Figure 21: Determining closest point to a face [126]

The bounding boxes for the faces are padded by an amount equal to four times the die stroke per time step to ensure that all of the nodes penetrating the die are taken into account. Padding the bounding boxes and using simple bounding box containment checks to reduce the number of proximity checks performed, allows us to reduce the CPU time needed per check as compared to the case when all of the faces are subjected to complete closest point checks. These checks are essential since, as the simulation progresses, an increasing percentage of the nodes on the workpiece boundary are subjected to these proximity/containment checks during the analysis. The impact on the simulation time is significant since these checks are performed once per iteration of a

time step (several iterations may be needed to obtain a converged solution). Padding the bounding boxes by four times the die stroke per time step provides a large enough safety net to catch all of the nodes that have penetrated the die. Appropriate UG/Open™ API calls are then used to obtain the normal to the die surface ($\mathbf{p}^u \times \mathbf{p}^w$) at the closest point $\mathbf{p}$ when the solver performs this check as part of the contact computations.

To further speed up the process of determining contact between the workpiece and the die during analysis, the inherently compute intensive geometric proximity/containment checks are applied only to a subset of the nodes on the workpiece boundary. The geometric checks in the mesh based contact algorithm are first used to obtain the classification information for the workpiece boundary nodes. This identifies the set of nodes ($\left\{ {}_W M_j^0 \mid {}_W M_j^0 \subset \partial\left({}_D M\right)\right\}$) to be subjected to the geometric proximity/containment checks. The geometric checks in the mesh based contact algorithm of Antares™ [8, 9] are explained with respect to Figure 22. A prism is built around each of the triangular faces (like face *abc* in Figure 22a) on the die boundary and nodes that lie inside the prism are classified as contact nodes. The height of the prism is set based on a user specified parameter termed as the *overlap band*, typically set to the die stroke per time step. The height of the prism extends to four times the overlap band into the die volume (prism *abcdef*) and one times the overlap band outside the die region (prism *abcghi*) as shown in Figure 22a. Note that the prism is extended outside the die region specifically to account for the approximation of the geometry by the finite element mesh representing the die. This ensures that nodes in close proximity of the die mesh will also be picked up for the geometric checks. Since the prism check will miss nodes in wedges (node *q* in Figure

22b), a distance/dot product check is performed on nodes not classified as contact nodes after the prism check. In Figure 22b, if (a) the dot product between vector (**q** – **b)** and face normal $\tilde{n}$ is positive and the distance **qb** is less than 4d, or, (b) the dot product between vector (**q** – **b)** and face normal $\tilde{n}$ is negative and the distance **qb** is less than d, the node is classified as a contact node. Selective application of the geometric proximity/containment checks has resulted in a considerable reduction in the CPU time required (as compared to the case when all the nodes are directly subjected to the checks). The reduction in CPU time can be especially large in the early stages of the forming simulation when only a small portion of the workpiece boundary nodes are in contact with a die.

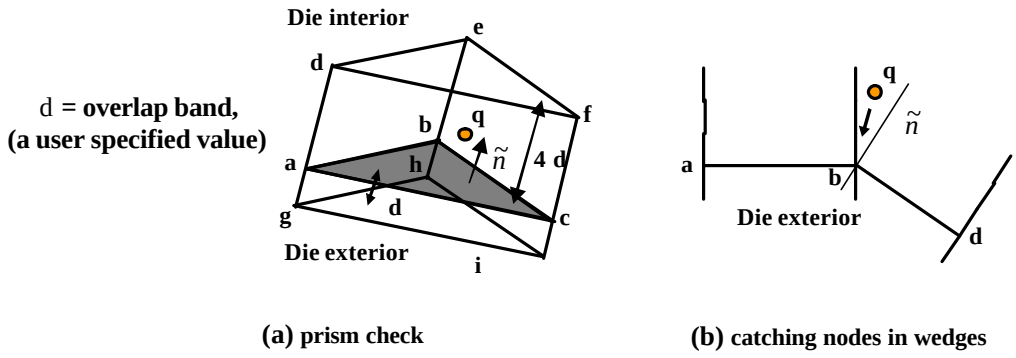


Figure 22: Geometric checks for contact in Antares™

4.4.2 Die-Overlap Monitoring Algorithm

As described in section 2.1, monitoring the "die-overlap" and triggering a remesh when die-overlap is detected, can reduce geometric approximation errors. In the hybrid method, die-overlap is the overlap between the workpiece (represented by its finite element mesh) and the die geometry. A typical situation with die geometry and straight-

edged workpiece mesh combination is illustrated in Figure 23. If the workpiece is remeshed based on the workpiece mesh geometry, the volume represented by the shaded region in Figure 23 is lost (or gained depending on the local die curvature).

The central idea of the die-overlap monitoring scheme is to use sampling point(s) on the edges of the contact portion of the workpiece boundary and determine die penetration, if any. The proximity/containment check forms the basis of this check. A single sample point at the mid-point of the contact edges is used to approximately check for die-overlap. Since a complete proximity/containment check is quite compute-intensive, a single sample point limits the increase in CPU time while providing an acceptable measure of the geometry approximation error. In the procedure implemented (Figure 23), the proximity/containment check described above is used to compute the distance between the mid-point of the finite element edge e_1, whose end-points (nodes *1* and *2*) are in contact with a die, and the die geometry.

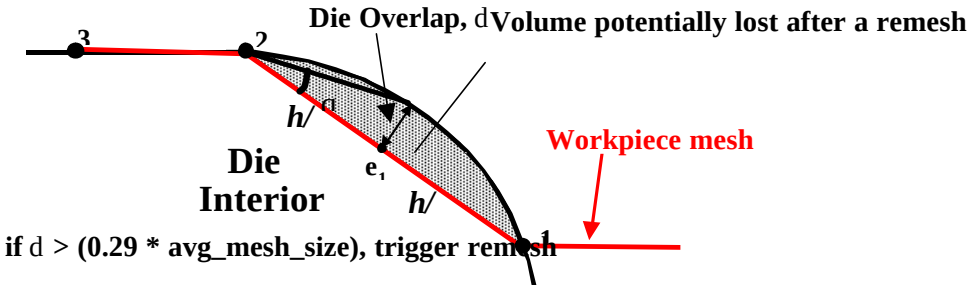


Figure 23: Die-overlap monitoring algorithm

The threshold value for triggering a remesh has been based on a limiting value for the angle θ in Figure 23 for the case where the length of the contact edge is equal to the

average mesh size specified by the user as input for the problem. The *average mesh size* represents the desired local mesh size for the workpiece. A small limiting value for θ is likely to result in frequent remeshes. As explained in section 2.4, frequent remeshes of the workpiece can cause mesh-rezoning errors to increase. On the contrary, a large value for θ implies a looser control on the geometric approximation error. A limiting value of 30^{o} has been selected in this implementation to limit the possible impact of mesh rezoning errors. This threshold may need to be altered when further studies on the mesh rezoning errors are completed. For a finite element edge of *average mesh* size length (h) in Figure 23, the overlap, $\delta = (h/2) * \tan 30^{o}$, i.e., $\delta = 0.29h$. Hence, this limiting value (30^{o}) for θ yields a threshold die penetration distance of 0.29 times the *average mesh size* specified by the user for the problem. A single threshold value based on a user specified parameter gives a single criterion for triggering a remesh as opposed to case where a threshold value is computed as a fraction of the contact edge length.

The impact of the die-overlap monitoring scheme on the workpiece volume is evident in the simulation of the valve body (hammer) forging process, for which full simulation results are presented in chapter 6. The simulation was performed using identical process parameters with and without die-overlap monitoring. With the original workpiece volume at 9.116 cu. in., the first remesh was triggered at time t = 0.0225 seconds when die-overlap was monitored and at time t = 0.03 seconds when die overlap was not monitored. The resulting workpiece volumes after geometry update and remeshing were 9.108 cu. in. (volume loss of 0.09%) and 9.098 cu. in. (volume loss of 0.19%) respectively in the two cases, showing a 52% improvement in volume preservation. The

results are similar when simulation results from the two cases were compared at time t = 0.1 seconds (70% of total die stroke). The workpiece volumes in the two cases were 9.063 cu. in. (volume loss of 0.58%) and 9.025 cu. in. (volume loss of 1.0 %) respectively in the two cases, showing a 42% improvement in volume preservation.

Chapter 5

THE HYBRID METHOD – GEOMETRY UPDATE AND REMESHING

This chapter discusses the procedures and algorithms developed, as part of the hybrid method, to update the geometry of the deformed workpiece and generate a new mesh to represent it for further analysis [71, 73, 87]. A brief overview of the overall geometry update process is first presented. The geometry update and remeshing procedures developed in this thesis research focus only on the generation of a new mesh representing the boundary of the deformed workpiece. An automatic mesh generator uses this updated workpiece boundary mesh to discretize the volume. This chapter first presents the data structures used and specifics of the model update algorithm. Procedures that enrich the surface (boundary) mesh are then described. These include algorithms for free surface curvature compensation, adaptive mesh and look-ahead mesh refinement.

5.1 Geometry Update Procedure – An Overview

The model update algorithm works in tandem with the solver. The solver simulates the process until the workpiece mesh is too distorted to continue at which point the model update and remeshing algorithm takes over. Information needed to update the workpiece geometry includes:

(i) the current geometry of the deformed workpiece as understood by the analysis code (only node coordinates and element connectivities – other classification and adjacency information is regenerated),

(ii) boundary conditions, if any, applied on the elements (contact status, symmetry, pressure, velocity),

(iii) the distribution of the field variable (strain rate, in our case) that is to be used to guide adaptive mesh refinement,

(iv) incremental die-strokes for each die, and,

(v) a mesh size field specified by the user during problem setup. This information is assigned to the workpiece like a boundary condition in the Antares™ [8,9] pre-processor and is provided to the geometry update procedure. Ideally, an error estimation process should define this mesh size field. In the present case, an *average mesh size* is used to guide the remeshing process in the portions of the domain where no other mesh size information is available.

The contact status information indicates if a workpiece boundary node lies on the free surface or is in contact with a die (geometry). For contact nodes, identification of the die with which it is in contact is also needed.

The objective of the geometry update procedure is to take the mesh model for the deformed workpiece from the solver and generate a surface mesh consisting of valid, 3-node triangular elements of acceptable shape to represent the workpiece boundary. Previous implementations of mesh-based schemes for 3-D geometry update and

remeshing [8-12, 38-50] are described in the next section along with their limitations so as to set the stage for describing the method developed in this thesis.

During geometry update, new nodes may be generated on the workpiece boundary. In order to enforce consistent interpretation of geometric contact, it is imperative that identical geometric containment/proximity procedures be used in the solver and the update procedure. The geometric containment/proximity procedure described in section 4.4.1 is also used within the update procedure. Geometric computations of interest are the containment/proximity checks to determine if new nodes created lie inside, outside or on the boundary of any of the dies.

Procedures have been incorporated within the geometry update scheme to address geometric approximation and aspects of the mesh discretization errors (discussed in chapter 2). An adaptive mesh refinement algorithm identifies regions where the solution error has exceeded a threshold. However, only the discretization on the workpiece boundary is refined because interior refine points cannot be specified as input to the version of the MeshCAST™ [8, 9] mesh generator that comes with the Antares™ system. Using a mesh generator that takes a general mesh size field as input is recommended though not implemented here.

A look-ahead mesh refinement procedure has also been implemented to refine the mesh on the free portion of the workpiece boundary that is in the vicinity of the die. This ensures better capture of die-workpiece contact as the simulation progresses. The look-ahead mesh refinement procedure and the mesh-based update procedure are likely to

create new nodes on the free portion of the workpiece boundary. To ensure that the free surface of the updated workpiece reflects the discretization of a smooth boundary, the locations of new nodes on the free surface are modified based on the local curvature.

5.2 Previous Work

In this section, 3-D geometry update procedures described in published literature are compared to help set the stage for describing the procedure developed here. The relevant technical issues of interest are the procedures used to create the updated geometry, treatment of contact and free surface areas, use of special procedures to control geometric approximation errors, and application constraints (if any).

Most 3-D applications [8-11, 43-50] use the boundary of the deformed workpiece (mesh) without any modifications as the updated workpiece geometry. Although the simplest approach, it has the potential of making simulation results unrealistic due to volume gain/loss when the discretization based on this faceted boundary is used to represent the workpiece in the ensuing analysis. A typical situation encountered in 2-D cases for a convex region of the workpiece is illustrated in Figure 24. The new mesh (node sequence 1, 5, 6, 7, 4) is generated using the boundary of the deformed mesh (node sequence 1, 2, 3, 4) as the workpiece geometry. This results in a localized volume loss and further computations for local material flow based on this discretization will be inaccurate. Simulations performed by researchers [139] with early versions of some commercial codes using this approach [10, 11] have shown volume losses of up to 20%.

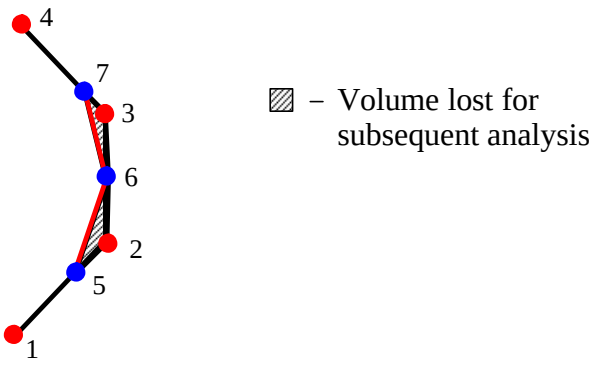


Figure 24: Volume change due to meshing of faceted boundary

Lee, et. al. [23] have developed a technique termed “automatic renoding” where the boundary of the workpiece is locally modified by adding new nodes (along boundary edges) and modifying the element shape functions in subsequent analyses. The decision on where to make the local change is based on mesh degeneracy and geometric interference criteria. In 3-D simulations, the boundary is initially represented by 4-node quadrilaterals (bricks representing the volume). Different types of renoded elements used in 3-D simulations where the top element face of the brick element is on the workpiece boundary are shown in Figure 25. Lee, et. al. [23] have used this technique to model a simple 3-D forging case. Though the local changes imply limited impact on workpiece volume, it is difficult to envisage application of this technique to model forming processes involving complex workpiece geometries since element distortion would eventually necessitate a remesh of the workpiece.

In the procedure developed by Coupez, et. al. [39-42], the workpiece is represented by 10-node tetrahedral elements during analysis (6-node triangles to define the boundary). The updated workpiece geometry is represented by fine mesh of linear triangles deduced from the boundary of the deformed workpiece. Each quadratic triangle defining the

workpiece boundary is divided into a several small linear triangles with the "goal of controlling the variation in the volume of the workpiece to less than a user-specified value" [39-42]. New nodes, created by this process, that lie inside a die (mesh) are pulled to the die boundary. It is not clear if any special procedures are in place to handle cases where the surface mesh contains degenerate elements. A coarse mesh of linear triangles is then generated (based on the fine mesh of linear triangles) while "maintaining a good precision on the workpiece geometry" [39]. This mesh of linear triangles is curved to match "as precisely as possible" [39], the boundary represented by the fine mesh of linear triangles and represents the boundary of the workpiece for the ensuing simulation. The procedure has been applied to forging of a tripod [39] but workpiece volume variation as the simulation progresses has not been shown.

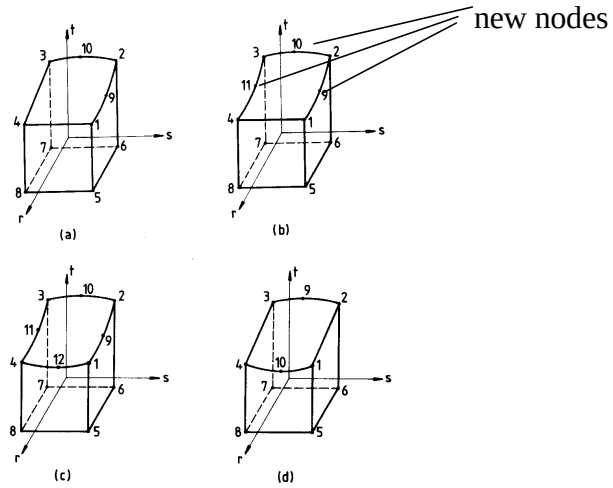


Figure 25: Renoded element configurations in 3-D simulations [23]

All of the published 3-D implementations [23, 39-50] and the commercial codes [8-12] use the die mesh during analysis and geometry update. Though some of the 2-D

procedures have implemented adaptive remeshing, the 3-D implementations do not address the refinement of meshes for discretization or geometric approximation errors.

The geometry update procedure developed in this thesis research [71, 73, 87] uses *mesh manipulation* operations to generate the updated representation of the boundary of the deformed workpiece. The basic methodology can be applied to a variety of evolving geometry problems such as bulk forming (forging, extrusion, etc.) and sheet forming. Results from the procedure implemented for bulk forming simulations are presented in the next chapter. Though the proposed methodology overcomes key technical limitations of the approaches described above, there are issues requiring further consideration. These issues relate to mesh gradation/transitioning, application of the procedure to higher order elements, and full versus local remeshing of workpiece. These topics are discussed further in section 5.10.

Before presenting details of the model update procedure, some relevant preliminaries are discussed. First, we examine the data structures used to store the needed mesh model adjacency information. Next, the *local geometry variation check* (LGVC) is discussed. This check forms the cornerstone of the model update procedure and serves to limit the change in the local workpiece geometry. Next, some of the basic mesh manipulation operations are reviewed and the sequence of operations to commit a topological change (with respect to the data structures) is given.

5.3 Data Structures to Store Mesh Adjacencies

In order to implement efficient procedures to update the workpiece geometry, data structures must be developed to store the mesh adjacency (topology) information. Mesh adjacencies are important because query/interpretation of the local geometry/topology is key to the implementation of the update procedure developed. Beall and Shephard [78] address the issue of topology based mesh data structures and the various options available. We could either store all the necessary adjacency information in the data structures or store only the primary adjacency information and derive the rest as needed. A structure, that stores all the information required by the model update procedure, has been implemented here and is given in Figure 26. Hence, there is no need to search for topological adjacency information. The decision to pursue this form of the implementation was based on the typical sizes of the mesh models for workpiece (up to 15000 nodes) and the die objects and also the RAM available (at least 512 Mb) on typical workstations used to perform the simulations. It is to be noted that the amount of memory needed to store all the adjacency information is far less than the memory typically used by the solver during the analysis. For example, in the disk-forging example, full simulation results for which are presented in the next chapter, the workpiece near the end of simulation had 3298 nodes (1860 on the boundary), 13393 tetrahedral elements, 3715 boundary faces and 5574 boundary edges. The memory requirement reported by the Antares solver (for the stiffness matrix) in this case is 77 Mb where as the memory requirement to store the required adjacency information into the data structures of Figure 26 is 1.295 Mb. Note that the FE_NODE structure needs (64 + (4 * no. of edges using node) + (4 * no. of faces adjacent to node)) bytes, the FE_EDGE

structure needs 56 bytes and the FE_FACE structure needs (112 + (32 * no. of labels on face)) bytes (a maximum of 32 characters allowed for each label).

```
typedef struct fe_node_info
{
   double nodeCoords[3];     /*  coordinates of the node  */
   int bcNode;               /*  indicates a boundary or corner node  */
   int contactStatus         /*  die with which node is in contact  */
   int edgesUsingNode;       /*  number of edges adjacent to this node  */
   FE_EDGE **edgeList;       /*  pointers to edges adjacent to node  */
   int facesUsingNode;       /*  number of faces adjacent to this node  */
   FE_FACE **faceList;       /*  pointers to faces adjacent to this node  */
} FE_NODE;

typedef struct fe_edge_info
{
   int edgeNodes[2];         /*  nodes of the edge  */
   int edgeFaces[2];         /*  faces adjacent to an edge  */
   int biEdge;               /*  indicates boundary or incident edge  */
   int forceSplit;           /*  indicates if length checks
                                 are to be ignored for splitting  */
   double meshSize;          /*  mesh size for the edge  */
   double edgeLen;           /*  edge  length  */
} FE_EDGE;

typedef struct fe_face_info
{
   int faceConn[3];          /*  face connectivity  */
   int adjElem;              /*  adjacent tetrahedron  */
   FE_EDGE **faceEdges;      /*  edges of the face  */
   double faceNormal[3];     /*  components of the normal to the face  */
   double faceFactor;        /*  shape factor for the face  */
   double LocalMeshSize;     /*  user specified local mesh size  */
   int numLabels;            /*  number of labels for the face  */
   char **labels;            /*  array of labels on the face  */
 } FE_FACE;
```

Figure 26: Data structures to store mesh model adjacencies

To ensure proper transfer and inheritance of analysis attribute information, we build a model topology for the workpiece represented by “model faces” bounding it. The model topology provides a convenient abstraction for defining the domain and allows the

convenient specification of analysis attributes such as boundary conditions [78-80]. For example, in Antares, boundary conditions are assigned to a named group of finite element faces (termed *mesh surfaces*), which, in our case, are the topological model faces of the workpiece geometric model. The geometric representation of these model faces is defined as the union of mesh entities and the data structure developed to facilitate this representation is illustrated in Figure 27. The notation proposed by Beall and Shephard [78] and described in Figure 28 is used to define the classification and adjacency information. Note that the procedure only deals with the workpiece mesh model and hence, there is no need for a separate identifier as used to distinguish between the mesh models for the die and the workpiece. Under this notation, mesh faces are termed as being classified on the model faces, i.e., ($\{M_i^2 | M_i^2 ?\ ^W G_i^2\}$). The *label* element of the FE_FACE data structure of Figure 27 identifies the boundary condition associated with the model face and is used to sort the element faces into groups representing model faces. Figure 29 shows the constructed model topology for the workpiece with model faces defined for symmetry, contact and free portions of the workpiece boundary. Restrictions are placed on the mesh manipulation operations when dealing with nodes/edges bounding the model faces during geometry update (explained in section 5.6).

```
typedef struct model_face_info
{
   int numFaces;              /*  no.  of FE faces in the "model face"  */
   FE_FACE **faceList;        /*  FE faces defining the "model face"  */
   char *label;               /*  label for the "model face"  */
 } MFACE;
```

Figure 27: Data structures to store model topology

${}^{W}G_i^d$	the i[th] entity of the workpiece geometric model of dimension d, $0 = d = 3$
${}^{D}_{n}G_i^d$	the i[th] entity of the geometric model of the n[th] die of dimension d, $0 = d = 3$
M_i^d	the i[th] entity of the mesh model of dimension d, $0 = d = 3$
$\partial(G_i^d)$	the entities on the boundary of G^d (${}^{W}G^d$ or ${}^{D}G^d$)
?	classification symbol used to associate mesh entities with model entities
$M_i^{d_i}\{M^{d_j}\}$	the unordered group of topological entities of dimension d_j that are adjacent to the entity $M_i^{d_i}$ of the mesh model of dimension d_i

Figure 28: Adjacency definition nomenclature from Beall and Shephard [78]

Mesh adjacency information necessary to perform the mesh manipulation operations is illustrated in Figure 30. For a node, *i*, its coordinates, adjacent edges ($M_i^0\{M^1\}$) and adjacent faces ($M_i^0\{M^2\}$) are stored. Adjacent edges are needed during smoothing and the adjacent faces are needed to build the local surface during free surface curvature compensation. In Figure 30, node 1 has 6 adjacent edges and 6 adjacent faces. The list of adjacent edges has $\{ M_i^0\{M^1\} = M_j^1, j = 1,..6 \}$ and the list of adjacent faces has $\{ M_i^0\{M^2\} = M_j^2, j = A,..F \}$. The *contactStatus* flag indicates the node's contact status and its classification with respect to the dies. Hence, the classification for a node, *i*, in contact with die *n*, can be represented as M_i^0 ? $\partial({}^{D}_{n}G)$. The *contactStatus* flag is set to 0 for nodes on the free surface and *n* for nodes in contact with die *n*. The *bcNode* element of the FE_NODE data structure is used to identify the classification of the node with respect to the model topology. Nodes in the interior of a model face are classified to be

on the model face ($\{M_i^0 | M_i^0 ?\ {}^W G_i^2\}$) and the *bcNode* element of the FE_NODE data structure for these nodes is set to 0. Nodes classified on model edges, defined as the set $\{M_i^0 | M_i^0 ?\ {}^W G_i^1\}$, are the nodes with adjacent faces ($M_i^0 \{M^2\}$) bounding two different model faces. For these nodes, the *bcNode* element of their FE_NODE data structure is set to 1. Nodes classified on model vertices, defined as the set $\{M_i^0 | M_i^0 ?\ {}^W G_i^0\}$, are the nodes with adjacent faces ($M_i^0 \{M^2\}$) bounding more than two different model faces. For these nodes, the *bcNode* element of their FE_NODE data structure is set to 2. These parameters are used in the mesh modification procedures discussed later. The geometry update procedure maintains the location of the mesh surface corner nodes.

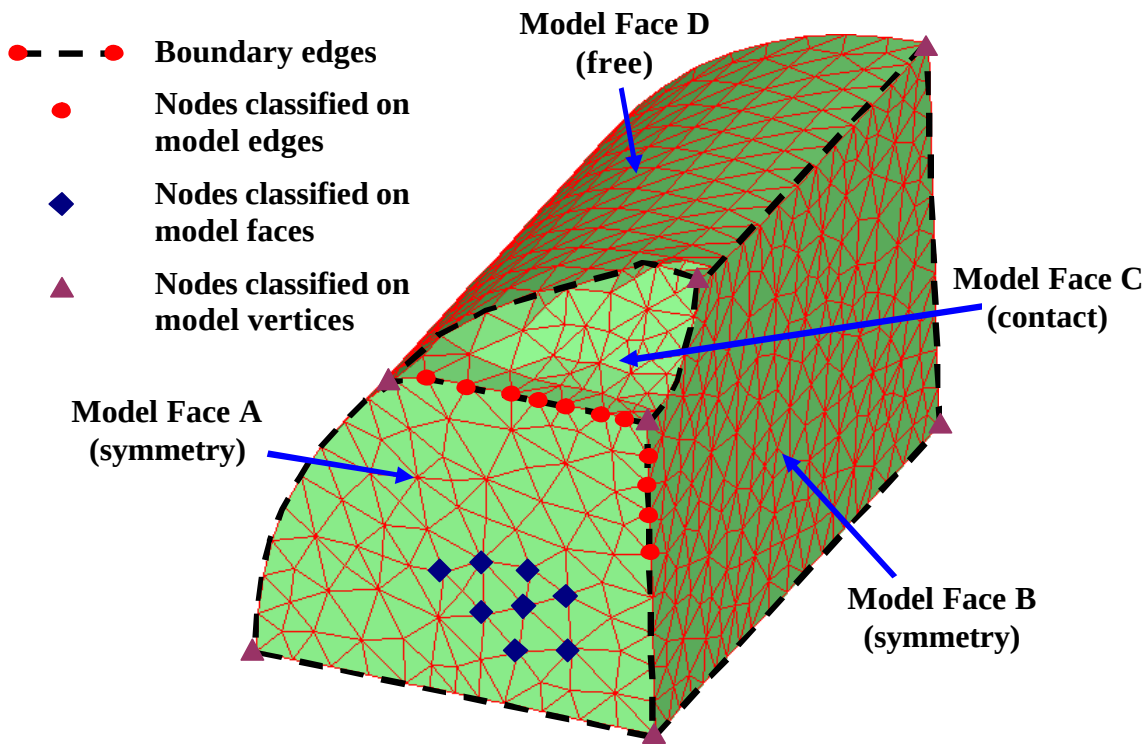


Figure 29: Classification for consistent transfer of analysis attributes

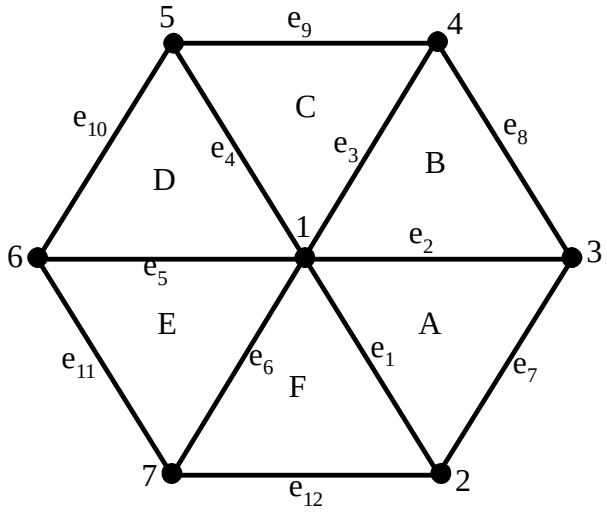


Figure 30: Mesh adjacencies required for mesh manipulation

For an edge, *i*, its bounding nodes ($\{M_i^1\{M^0\}\}$), adjacent faces ($\{M_i^1\{M^2\}\}$) and the mesh size to be used for discretization are stored. Information on the two adjacent faces is needed during edge swapping. The square of the length of each finite element edge is computed and stored in the *edgeLen* element of the FE_EDGE data structure to avoid repeated computations during geometry update. Squares of the lengths are used to avoid square root computations. The mesh manipulation operators that create or delete mesh entities (splitting and collapsing operators) make their decisions based on the *edgeLen* values. Since the user can specify a varying mesh size requirement in different areas of the workpiece boundary, a desired mesh size value is associated with each edge in the *meshSize* element of the FE_EDGE data structure. The initial value for desired mesh size is set as the square of the user specified *average mesh size* for the analysis. The *meshSize* value is then modified for edges in regions to be refined and where the user has defined a local mesh size requirement. The goal of the mesh manipulation operations is to bring the *edgeLen* value close to the *meshSize* value for an edge. To control geometric

approximation errors, it may be necessary to split the edge even if it is not required to satisfy the length-based criteria. Hence, an additional flag, *forceSplit*, is associated with every edge. When set to 0 (default) the length-based checks take precedence in deciding if the edge is to be split. A setting of 1 implies that the edge is to be split regardless of the outcome of the length-based checks. The *biEdge* element of the FE_EDGE data structure is used to indicate its classification with respect to the model topology and is initialized to 0 to indicate its location in the interior of the model face ($\{M_i^1 | M_i^1 ?\ ^W G_i^2\}$).

For mesh edges classified on model edges, defined as the set $\{M_i^1 | M_i^1 ?\ ^W G_i^1\}$, the *biEdge* element of the FE_EDGE data structure is set to 1. These are edges whose adjacent faces ($\{M_i^1 \{M^2\}\}$) bound two different model faces. Note that the *bcNode* element of the FE_NODE data structure of the two bounding nodes ($\{M_i^1 \{M^0\}\}$) has a non-zero value. Edges bounded by one node classified on a model edge (*bcNode* element of its FE_NODE data structure is non-zero) and the other classified on a model face (*bcNode* element of its FE_NODE data structure is zero) are also identified and the *biEdge* for these edges is set to 2. It is to be noted that Antares recomputes contact information when a new mesh is given. Consistency is maintained via use of identical algorithms are used to determine contact. Hence, there is no need to identify and store edges and/or faces bounding contact and free surfaces. However, this information may be needed if a different analysis code is used.

For a face, *i*, on the boundary of the workpiece, its bounding nodes ($\{M_i^2 \{M^0\}\}$), and edges ($\{M_i^2 \{M^1\}\}$) are stored. The face normal is stored to compute angles between

faces as needed during edge swapping and free surface curvature computation (explained in sections 5.6 and 5.8). The *LocalMeshSize* element of the FE_FACE data structure represents the user specified local mesh size requirement. The *meshSize* element of the FE_EDGE data structure for the bounding edges inherits this value. To facilitate proper transfer of the boundary conditions, the number of attribute labels and the attribute labels assigned to the face (each label representing a boundary condition) are stored. Note that multiple boundary conditions (thermal/mechanical) may be assigned to faces. The tetrahedral element adjacent to this face ($\{M_i^2\{M^3\}\}$), is stored in the *adjElem* element of the FE_FACE data structure. This information is used to identify refinement regions on the workpiece boundary to control mesh discretization errors.

Element quality plays an important role in the geometry update procedure and the metric for element quality is used as a basis for deciding the edges to collapse. The quality of an element with straight edges is typically measured in terms of one or more of the following or similar shape metrics [128-131]:

1. Aspect ratio measure:

    a. Normalized ratio of the radius of the inscribed circle, r, to the radius of the circumscribed circle, R,

$$Q_1 = \frac{2r}{R} \qquad \textbf{(28)}$$

    b. Normalized ratio of the minimum height, d_{min}, to the largest base length of the triangular face, l_{max},

$$Q_2 = \frac{2}{\sqrt{3}} \frac{d_{\min}}{l_{\max}} \quad \textbf{(29)}$$

2. Angle measure: Smallest and the largest angle for the element f_{min}, $f_{max} \in [0, 180]$ degrees which can be measured as the following normalized metrics:

   a. Small angle metric, $Q_3 = 2\left(1 - \cos\left(f_{\min}\right)\right)$ **(30)**

   b. Large angle metric, $Q_4 = \frac{2}{3}\left(1 + \cos\left(f_{\max}\right)\right)$ **(31)**

For these quality measures, $0 \le Q_i \le 1$ for $1 \le i \le 4$. In the thesis implementation, aspect ratio measure Q_2 has been used due to its lower computational cost compared to explicit computation of angles. The element quality as measured by Q_2 is computed for each element face and stored in the *faceFactor* element of the FE_FACE data structure.

5.4 Local Geometry Variation Check

The purpose of the *local geometry variation check* (LGVC) is to measure the potential impact of a mesh manipulation operation on the volume between the two affected portions (before and after configurations) of the surface mesh. If the change in local workpiece volume is within acceptable limits (defined below), the mesh manipulation operation is performed.

For the case of swapping, the affected region is a tetrahedron defined by the two triangles adjacent to the edge being considered for swapping ($\{M_i^1 \{M^2\}\}$). In Figure 31, the nodes *1234* define the tetrahedron formed by the triangles *A* and *B* before swapping and the

nodes *1243* define the tetrahedron after swapping. To determine if swapping would result in an acceptable geometry, we compute and compare the volumes of these tetrahedrons. The volume, V_i, of a tetrahedral element is computed as the determinant,

$$V_i = \frac{1}{6}\begin{vmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{vmatrix}, \text{ or, } V_i = \frac{1}{6}\begin{vmatrix} (x_2 - x_1) & (y_2 - y_1) & (z_2 - z_1) \\ (x_3 - x_1) & (y_3 - y_1) & (z_3 - z_1) \\ (x_4 - x_1) & (y_4 - y_1) & (z_4 - z_1) \end{vmatrix} \quad \textbf{(32)}$$

where $\{x_i, y_i, z_i, i = 1,..4\}$ represent the coordinates of the nodes of the tetrahedron [141]. The criterion to decide if the change in local volume is acceptable is discussed below.

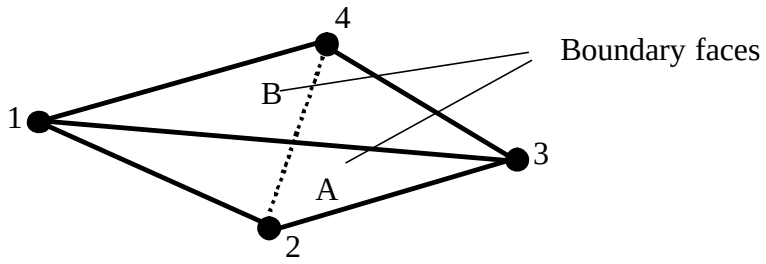
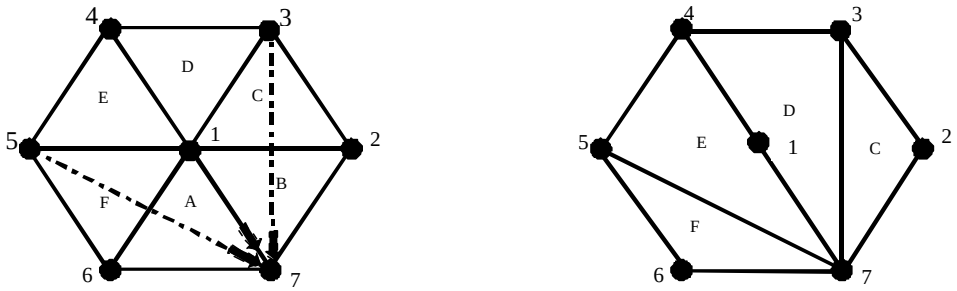


Figure 31: LGVC for edge swapping

A similar procedure is used for the collapsing operation and the volume of the affected region is computed using the faces adjacent to the node being deleted ($\{M_i^0 \{M^2\}\}$) and the location of the node being deleted as the apex of the tetrahedrons. This selection for the apex of the tetrahedrons ensures that we do not need to compute the volume of the affected region before collapsing (since it will always be zero). Figure 32 shows the local mesh configuration before and after collapsing of edge *17* with node *1* being deleted. The volume, V^{aft}, of the affected region after collapsing would be:

$$V^{aft} = \sum_{i=1}^{n} V_i \tag{33}$$

where i =1…n, the number of faces adjacent to the node being deleted ($\{M_i^0 \{M^2\}\}$), and V_i is the volume of prism computed as described above.



(a) before collapsing **(a) after collapsing**

Figure 32: LGVC for edge collapsing

To decide if the change in local geometry is acceptable, we compute the percentage change in local workpiece volume, V_c^l, defined as

$$V_c^l = \left| \left(\frac{V_b^{aft} - V_b^{bef}}{V^w} \right) \times 100 \right| \tag{34}$$

where V_b^{bef} is the volume of the affected portion of the workpiece boundary before mesh manipulation, V_b^{aft} is the volume of the same portion after mesh manipulation and V^w is the volume of the workpiece before any mesh manipulation. The volume of the

workpiece is the sum of the volumes of each of the tetrahedrons defining the workpiece mesh computed as described above.

For a non-coplanar set of bounding faces, V_c^l is greater than zero. The ideal scenario obviously is to have no volume change but this is not practical since we have to deal with complex forming geometries defined by curved boundaries. Hence, the threshold value for proceeding with the mesh manipulation operation is based on empirical guidelines with the explicit objective of limiting total workpiece volume change. Typically, the preform is designed with about 4% more volume than the desired forging shape to allow for flash [139]. For simulation purposes, this gives a narrow range for allowable total workpiece volume change. With this in mind, a threshold value of 0.1% per operation (swapping/collapsing) has been selected and as shown below, found to yield satisfactory results. The LGVC is applied only when considering a mesh manipulation on the workpiece free surface in order to control the volume change due to the approximation of the free surface (V_f in section 3.4). The nodes in the contact region are always forced to remain in contact with the die geometry and the local volume changes there are due to the changes in local die curvature and unavoidable in order to enforce the relaxed consistency requirements. The efficacy of LGVC as a basis for applying the mesh manipulation is confirmed by small volume change (Figure 33) in the early stages of the disk-forging example (results presented in the next chapter) when only a small portion of the workpiece boundary is in contact with the die.

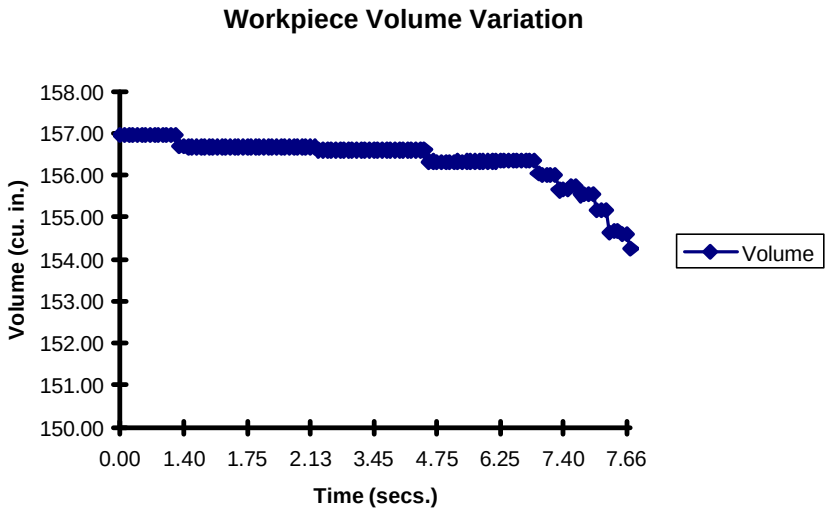


Figure 33: Volume change for disk-forging process

5.5 Mesh Topology Manipulation Operators

Four different mesh manipulation operations are used to update the workpiece mesh model. These are standard mesh manipulation operations encountered in the mesh generation literature [72]. The mesh manipulation operations used are:

(a) edge collapsing,

(b) edge swapping,

(c) edge splitting, and,

(d) smoothing.

Changes to the workpiece mesh model entail continuous updates to the mesh data structures that store the mesh adjacency information. This is required so that the mesh

data structures reflect the current state of the mesh model. The updates necessary to keep the adjacency information current, when a topological operation is performed, are described below. Details of the criteria for performing each of the mesh manipulation operations are explained in section 5.6. Note that though these operations are standard, updates to the adjacency information depend on the data structures used. In our case, the updates are described with reference to the data structures described in Figure 26.

5.5.1 Edge Collapsing

The edge collapsing operation is performed to remove small edges and results in the deletion of one node, three edges and two faces from the mesh model. Consider the mesh topology set-up as shown in Figure 34 where the dotted lines indicate the local mesh topology after collapsing. Assuming that edge 1 is to be collapsed (with node *a* to be deleted), the topological modifications needed, explained with reference to Figure 34 and data structures detailed in Figure 26, are as follows:

1. Node *a* is deleted.

2. Edges *1, 11* and *3* are deleted and removed from the node->edge adjacency list (*edgeList* array of FE_NODE) of their bounding nodes.

3. Faces *A* and *B*, adjacent to edge *1*, are deleted and removed from the node-face adjacency list (*faceList* array of FE_NODE) of their bounding nodes.

4. The face->node adjacency information (*faceConn* array of FE_FACE) for all faces using node *a*, excluding faces *A* and *B*, i.e., faces *C, D, E, F* is updated. Node *a* in the face-

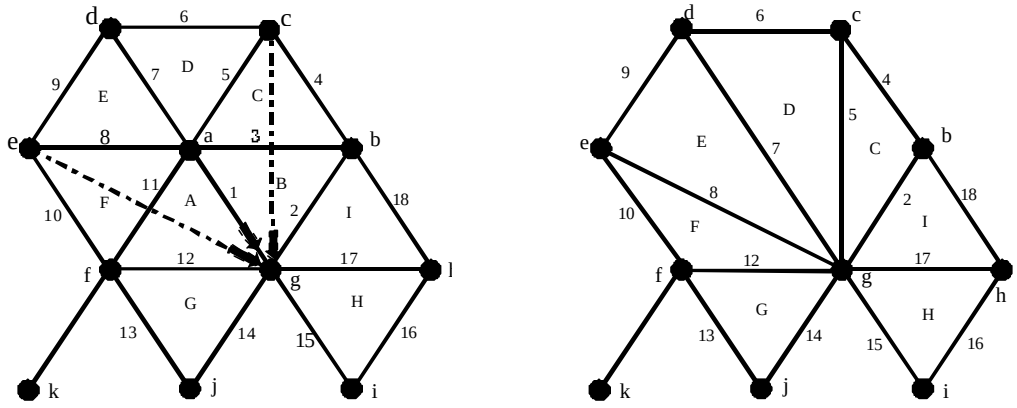
>node adjacency list of these faces is replaced by node *g*. These faces are also added to the node->face adjacency list (*faceList* array of FE_NODE) for node *g*.

5. The edge->node adjacency information (*edgeNodes* array of FE_EDGE) for the remaining edges using node *a*, i.e., edges *5, 7, 8* is updated. Node *a* in the edge->node adjacency list is replaced by node *g*. These edges are also added to the node->edge adjacency list (*edgeList* array of FE_NODE) for node *g*.

6. The face->edge adjacency information (*faceEdges* array of FE_FACE) for faces *C* and *F* is updated. Deleted edges *3* and *11* are replaced by edges *2* and *12* respectively.

7. The edge->face adjacency information (*edgeFaces* array of FE_EDGE) for edges *2* and *12* is updated. Faces *C* and *F* replace the deleted faces *B* and *A* respectively.

8. The deleted faces *A* and *B* are removed from the definition of the appropriate model faces (*faceList* array of the MFACE data structure).



(a) before collapsing **(b) after collapsing**

Figure 34: Mesh topology set-up for edge collapsing

5.5.2 Edge Swapping

In edge swapping, the connectivity of the face is rearranged with the intention of improving element quality. Assuming that edge *c* in Figure 35 is to be swapped, the connectivity for face *A* would then be nodes *1, 2, 4* instead of nodes *1, 2, 3*. The connectivity for face *B* would be *1, 4, 3* instead of nodes *2, 4, 3*. The topological modifications needed, explained with reference to Figure 35 and the data structures detailed in Figure 26, area as follows:

1. The edge->node adjacency information (*edgeNodes* array of FE_EDGE) is modified. Identify the two nodes, one from each of the faces *A* and *B*, which do not use edge *c*. In our case these would be nodes *1* and *4*. These two nodes would form the new edge->node adjacency for edge *c*. Replace node *2* with node *1* and node *3* with node *4* in the edge->node adjacency list for edge *c*.

2. The face->node adjacency information (*faceConn* array of FE_FACE) for faces *A* and *B* is updated to reflect the new connectivities.

3. The node->edge adjacency information (*edgeList* array of FE_NODE) is updated. Edge *c* is removed from the lists of node *2* and node *3* and added to lists of node *1* and node *4*.

4. The node->face adjacency information (*faceList* array of FE_FACE) is updated. Face *B* is deleted from the list of node *2* and added to list of node *1*. Face *A* is deleted from the list of node *3* and added to the list of node *4*.

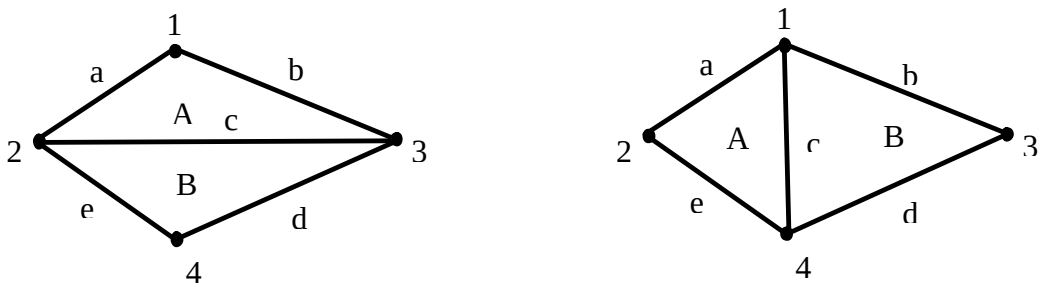


Figure 35: Mesh topology before and after swapping

5.5.3 Edge Splitting

The edge splitting operation is performed to refine the mesh in the vicinity of long edges and results in the creation of one new node, three new edges and two new elements. The initial location split point (where the edge is split) is the mid-point of the edge. Assuming edge *b* in Figure 36 is to be split, the topological modifications needed, explained with reference to Figure 36 and the data structures detailed in Figure 26, area as follows:

1. A new node (FE_NODE data structure) is created to save information for node *5*. The coordinates (*nodeCoords* array of FE_NODE) are set to the midpoint of edge *b*.

2. The face->node adjacency information (*faceConn* array of FE_FACE) for faces *A* and *B* is updated with node *5* replacing node *4*.

3. Two new faces (FE_FACE data structure) are created to save information for faces *C* and *D*. The face->node adjacency information (*faceConn* array of FE_FACE) is set. These faces are also added to the node->face (*faceList* array of FE_NODE) information for node *5*.

4. The edge->node adjacency (*edgeNodes* array of FE_EDGE) information for edge *b* is updated. Node *4* is replaced by node *5*. Edge *b* is added to the node->edge adjacency list (*edgeList* array of FE_NODE) of node *5* and deleted from the list for node *4*.

5. Three new edges (FE_EDGE data structure) are created to store information on the new edges adjacent to node *5*, i.e., *f, g, h.* The edge->node (*edgeNodes* array of FE_EDGE) and the edge->face (*edgeFaces* array of FE_EDGE) adjacencies for these new edges are set. The node->edge (*edgeList* array of FE_NODE) information for nodes *5, 1, 3* and *4* is updated to reflect the new edges.

6. The face->edge adjacency information (*faceEdges* array of FE_FACE) for faces *A, B, C,* and *D* is updated.

7. The new faces *C* and *D* are added to the definition of the appropriate model faces (*faceList* array of the MFACE data structure).

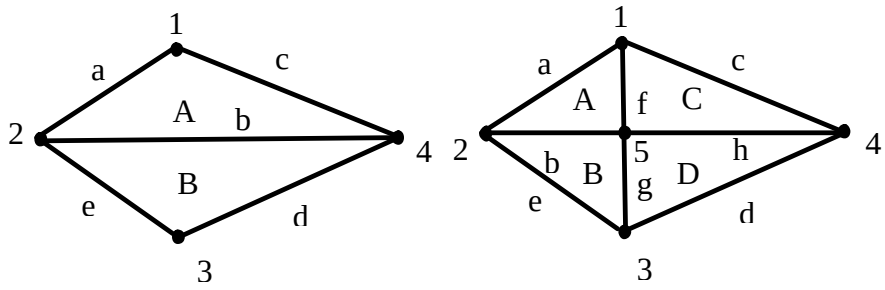


Figure 36: Mesh topology before and after splitting

5.6 Updating the Mesh Model

In this section, we explain how the mesh manipulation operators described in the previous section are used in conjunction with the local geometry variation check to

update the geometry of the deformed workpiece. Since handshaking between the solver and the update procedure is accomplished via a text file containing the results data, the first step in the update procedure is to read the results data from this file. The mesh data for the dies and the workpiece along with information on the boundary conditions (contact, symmetry, pressure, velocity) are read in and the adjacency information generated and stored in data structures described in Figure 26. The mesh information is continuously updated as the mesh manipulation operations are performed.

Node data including the node number, coordinates and contact status are available directly from the results file and stored in the *nodeNumber*, *nodeCoords* and the *contactStatus* elements of the FE_NODE data structure of Figure 26. Similarly, data for each triangular face on the boundary of the workpiece including the element number, its connectivity and information on the boundary conditions applied to the face are available directly from the results file. These are stored in the *faceNumber*, *faceConn*, *numLabels* and *labels* fields respectively of the FE_FACE data structure of Figure 26. Since Antares is not able to directly provide the tetrahedral element using a boundary face, information on all tetrahedral elements is read in. Comparing the connectivities of the tetrahedral elements with the connectivity of the face allows us to identify the tetrahedral element adjacent to a given face. This is then stored in the *adjElem* element of the FE_FACE data structure. Note that tetrahedral element adjacencies are not maintained since they are not needed during workpiece geometry update. The normal to each element face and the aspect ratio measure, Q_2, are computed using the node coordinates and stored in the *faceNormal* and *faceFactor* elements of the FE_FACE data structure of Figure 26.

As the element connectivity information is read in, the node->face adjacency is built by adding the face to the *faceList* element of the FE_NODE data structure of Figure 26. To generate the edge related information, we loop over the 3 edges of each of the element faces. The node->edge adjacency list (*edgeList* array) of one of the bounding nodes is evaluated to determine if an edge with the given bounding nodes has already been created. A new edge is created if one with the given edge->node adjacency does not exist. The bounding nodes of the edge under consideration define the elements of the *edgeNodes* array of the FE_EDGE data structure of Figure 26. The new edge is given an *edgeNumber*, which is the current value of the running count of the edges created. The node->edge adjacency for both of the bounding nodes is updated by adding the new edge to the *edgeList* array of their FE_NODE data structure. The bounding face for this edge is added to the first element of the *edgeFaces* array of the FE_EDGE data structure. If an edge with the two bounding nodes exists, the bounding face is added to the second element of the *edgeFaces* array.

Before updating the workpiece mesh model, the dies (stored in the Unigraphics CAD model) need to be translated to reflect the current simulation state. This is accomplished by applying a rigid-body transformation to the die object using the UG/Open API provided by Unigraphics™. The interaction with CAD models for the dies has been discussed in section 4.4.

The workpiece surface mesh is updated by performing collapsing, splitting and swapping operations in a sequence. Collapsing is selected as the first operation to eliminate distorted (long, sliver-like) triangular elements (identified by the aspect ratio measure)

that may be present in the deformed workpiece mesh model. Sliver elements are poor quality elements that have a deleterious effect on the solutions [142]. Also, using another operation to begin the update procedure could potentially result in more poor quality elements on the workpiece boundary. Examples of potential situations are shown with respect to the sliver elements in Figure 37(a). If we were to use edge splitting as the first operation, the resulting local mesh would be as shown in Figure 37(c). If edge swapping is to be used as the first operation, the resulting mesh would first be as shown in Figure 37(b) and then after a subsequent splitting operation be as shown in Figure 37(d). In both cases (Figure 37(c) and Figure 37(d)), there are more sliver elements created. Hence, collapsing is used as the first operation.

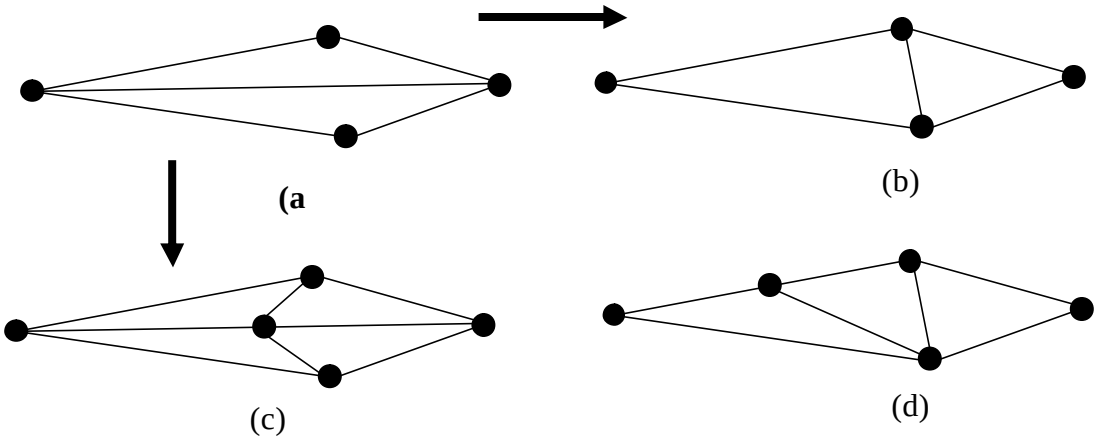


Figure 37: Dealing with sliver elements

Due to the sequential procedure of the collapsing and splitting operations, we need to be careful in setting the criteria that trigger these operations. Improper selection of the criteria could potentially result in redundant operations such as the one shown in Figure 38. In the initial configuration shown in Figure 38 (a), let us assume that edges *a* and *b* are smaller than their desired mesh size. If the desired mesh size is the trigger, collapsing

edge *a* results in the configuration of Figure 38 (b). The length of edge *b* will now trigger a splitting operation and result in the configuration of Figure 38(c), which in turn could result in a swapping operation for edges *c* and *d.* The net result is no change in the workpiece geometry but three redundant operations. The approach used here is to set criteria that would result in a mesh where the edge lengths are within a range around the desired mesh size. The lower bound, termed the *minimum edge length* defines the minimum threshold for edge length, i.e., edges with length less than *minimum edge length* are deleted. The upper bound, termed the *maximum edge length* defines the maximum threshold for edge length, i.e., edges with length more than *maximum edge length* are split. Both are set as scaled values of the *desired mesh size* for the edge (the *meshSize* element of the FE_EDGE data structure). Scaling the desired mesh size allows us to set the values for these bounds based on the local mesh size requirement. The *minimum edge length* and the *maximum edge length* must be related in that (2.0 * *minimum edge length*) must be greater than the *maximum edge length*. The values for the scale factors are set to 0.65 and 1.35 for the lower and upper bounds respectively. This is the only set of numbers that gives an equal range (0.35) on either side of the desired mesh size and also satisfies the aforementioned constraint. For example, a value of 0.7 for the scale factor for *minimum edge length* requires that the scale factor for the *minimum edge length* is greater than 1.4 resulting in an unequal range around the desired mesh size.

For the element quality, a minimum threshold of 0.1 has been chosen for Q_2 in our implementation. Note that the element quality factor will be 1 for an equilateral triangle. The value of 0.1 as the limit was selected by computing the factor (and rounding) for a hypothetical flattened triangle whose largest angle is 160^{o} and the other two angles are

10^{o} each. The limiting angle of 160^{o} has been selected since Babuska and Aziz [57] have developed mathematical analysis leading to the requirement that the largest angle for triangles should be bounded away from 180^{o} in order to maintain theoretical rates of solution convergence.

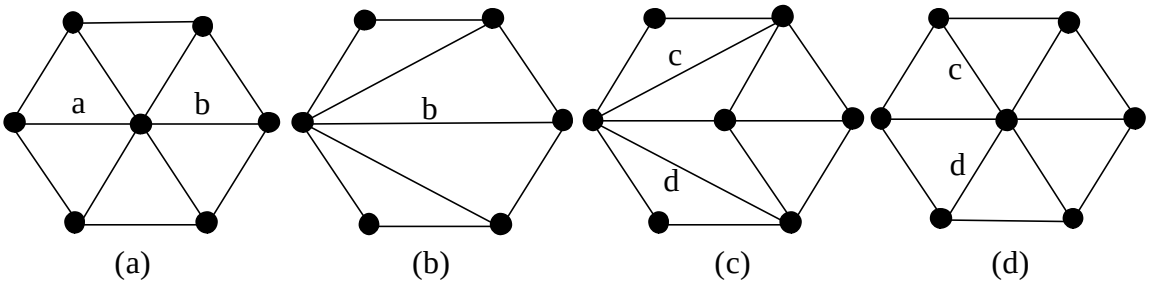


Figure 38: Redundant mesh manipulation operations

The procedure for collapsing involves identification of the edge to collapse followed by the local geometry variation check to determine if the local geometry after collapsing will be within acceptable limits of variation. For collapsing and swapping operations, changes to the mesh model are made only if the variation in local geometry is within acceptable limits as defined by the LGVC. The decision to collapse an edge is based on evaluation of the length of the edge and the quality of the elements adjacent to the element. If the edge length or the element quality factor (*faceFactor* element of the FE_FACE data structure) for either of its adjacent faces is less than desired, the edge is to be collapsed.

The procedure loops through each edge on the workpiece boundary to determine if the length of any of the edges is less than the *minimum edge length* for that edge. Note that

the actual length checks (for collapsing and splitting) compare the *edgeLen* variable of the FE_EDGE data structure for the edge and the square of *minimum/maximum edge length*. The decision on which of the two nodes, of a candidate edge (*node1* and *node2* for our discussion) being evaluated for deletion, is to be deleted is made after checking the node flag (*bcNode* element of the FE_NODE data structure) that indicates if it lies on a mesh surface boundary and the effect of deleting the node on the local geometry. The following scenarios exist:

1. If a node is a *corner node* (*bcNode* is 2), it cannot be deleted.
2. If the nodes are in contact with different dies, the edge cannot be deleted. Comparing the *contactStatus* flag of each of the nodes allows us to check for this condition.
3. For *boundary incident edges* (*biEdge* is 2), if *node1* lies on the boundary of a mesh surface (*bcNode* is 1) on which a boundary condition has been defined and *node2* does not (*bcNode* is 0), then only *node2* can be deleted and vice versa.
4. If an edge does not bound any mesh surface (*biEdge* is 0), the node to be deleted is selected based on the results of the LGVC check applied to both the nodes. The node that produces the smaller local change in mesh geometry is deleted.

The procedure for splitting involves identification of edges that are longer than the *maximum edge length* value for that edge or ones that are forced to split (element *forceSplit* of the FE_EDGE data structure is set to 1) as part of mesh refinement requirements (discussed in section 5.7). The procedure loops through each edge on the workpiece boundary to identify candidate edges for splitting. The initial location of the point where an edge is to be split is set as the mid-point of the edge. The mid-point is

used purely to keep the procedure simple and another location (for example, a computed optimal location) could be used, if desired. However, immaterial of the location of the point along the edge to be split, we need to determine its contact status. The split point location may fall in one of the two situations shown in Figure 39:

1. In the scenario illustrated in Figure 39(a), the bounding nodes of the edge being split are in contact with the die.
2. In the scenario illustrated in Figure 39(b), the bounding nodes of the edge being split are not in contact with the die but the location of the split point is inside the die.

We use the classification information for the bounding nodes of the edge being split and the proximity/containment check to deduce the classification information and the final location for the split point. For edges with both nodes ($\{M_i^1 \{M^0\}\}$) in contact with the same die ($M_i^0 \; ? \; \partial\left({}_n^D G\right)$) as shown in Figure 39(a), the split point is also classified as being in contact with the same die ($M_i^0 \; ? \; \partial\left({}_n^D G\right)$). For edges with one or no contact nodes, the geometric proximity/containment check described in section 4.4.1 is used to determine the proper classification for the split point - if it is inside, outside or on the die. When the split point is inside or on a die, it is classified as being in contact with that die ($M_i^0 \; ? \; \partial\left({}_n^D G\right)$).

The locations of all new nodes classified on a die are altered so as to lie on the die geometry and the *biNode* element of their FE_NODE data structure is set to n. When the split point is outside the die, the *biNode* element of the FE_NODE data structure is set to 0 and the location of the new node is altered to reflect the local curvature of the workpiece. This curvature compensation procedure is explained in section 5.8.

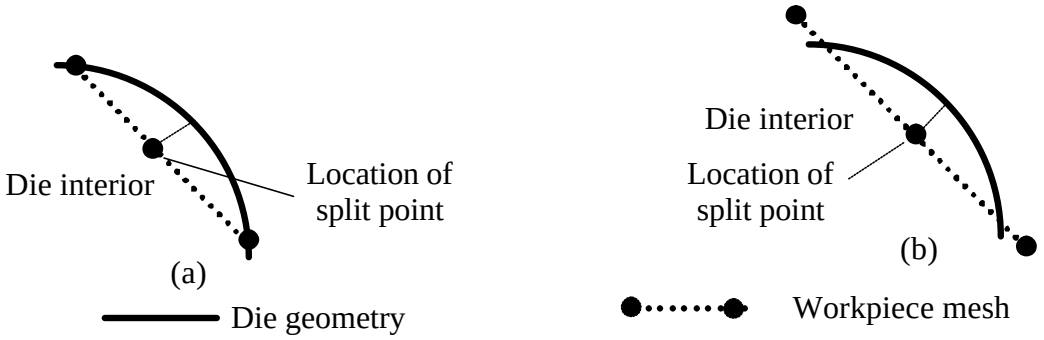


Figure 39: Proper location of split-points

In our implementation, the edge swapping procedure loops through each edge on the workpiece boundary to determine if it is swappable. An edge is swappable if:

1. The two faces adjacent to the edge being swapped (for example, faces A and B in Figure 35 when edge *c* is to be swapped) belong to the same model face. This ensures that mesh surface boundary edges (with element *biEdge* of the FE_EDGE data structure set to 1) are not swapped in order to preserve the topological definition of the surface on which a boundary condition has been defined. Note that boundary incident edges (with element *biEdge* of the FE_EDGE data structure set to 2) can be swapped because its adjacent faces belong to the same mesh surface.

2. The local geometry variation check applied to the current and the *as swapped* configuration shows local volume changes within the acceptable limits defined in section 5.4. This ensures that swapping does not introduce large changes to the local geometry.

3. Swapping does not introduce artificial flow constraints such as kinks on the workpiece boundary. This is monitored by measuring the dihedral angle between the faces in the

before and after configurations. If this change in dihedral angle (computed using the face normal information) is less than a threshold value, the edge can be swapped. This threshold is set to 5^{o} in our case to avoid formation of kinks that could result (more likely with a higher threshold) in artificial defects such as folds/laps.

4. The quality of the affected faces (for example, faces A and B in Figure 35 when edge *c* is to be swapped), as measured by the element quality factor Q_2, improves after the swapping operation. This constraint is mainly to quantify and use a metric to ensure that the swapping operation improves local mesh quality. Hence, the minimum shape (quality) factor (for the two affected elements) must be higher after swapping.

5.7 Adaptive Mesh Refinement

In chapter 2, the different sources of errors in the simulation were identified and methods for addressing them were discussed. Procedures implemented in this thesis to adaptively refine the discretization on the workpiece boundary and control mesh discretization and geometric approximation errors are discussed in this section. Since geometry update is facilitated by the mesh manipulation operations described in the previous section, procedures to identify regions for mesh refinement are invoked right after the mesh model for the deformed workpiece is loaded into the data structures. Mesh refinement is achieved by forcing the appropriate finite element edges to split. This is accomplished by setting the *forceSplit* flag of the FE_EDGE data structure for the edge to 1. The length-based checks are ignored for this edge, thus resulting in a finer discretization in the vicinity of this edge. Procedures used to identify regions for mesh refinement (edges to be flagged) are described below.

5.7.1 Controlling Mesh Discretization Errors

An error indicator, based on inter-element jumps in effective strain rate, is used to identify regions for mesh refinement. Since errors are related to dissatisfaction of the governing equations and boundary conditions [104], the inter-element jump is a dissatisfaction of the governing equation (equilibrium, in our case) and contributes to the error. In the case of linear tetrahedral elements, the second derivative of the functions used in the approximation is identically zero. Hence, the inter-element jumps become the error contributors when there are no body forces [103, 105]. As illustrated in chapter 2, the strain rate provides a snapshot of the workpiece deformation at the time of the remesh and hence, is a useful indicator of the regions likely to undergo further deformation. The solution error indicator for a linear displacement tetrahedral element i, E_i, is measured as the inter-element jump:

$$E_i = \max\left|\dot{\bar{e}}_i^{\,p} - \dot{\bar{e}}_j^{\,p}\right| \quad j = 1,....N, \qquad \textbf{(35)}$$

where $\dot{\bar{e}}_i^{\,p}$ is the effective plastic strain rate for element i and N is the number of adjacent tetrahedral elements (4 for an interior element).

It is common to see a fairly large range for the strain rates and for the inter-element jumps as the workpiece undergoes deformation. The reason for this is that in a typical forming process involving complex shapes, a very small region of the workpiece is undergoing deformation at any given time. A typical situation is illustrated in Figure 40 showing the strain rate distribution (for a hammer forging operation) in the workpiece that needs remeshing (values from 0 to 72). To accommodate the large range of strain

rate jumps typically seen in forming analysis, the median value of the jumps is used as a threshold to indicate the regions for mesh refinement. The median value of the jumps is computed as:

$$E_{med} = (E_{\max} + E_{\min}) \times 0.5 \quad \textbf{(36)}$$

where E_{max} is the maximum value and E_{min} is the minimum value of the strain rate jumps across all the tetrahedral elements in the model. Elements, where the jump is greater than the median $(E_i \geq E_{med})$, are flagged for refinement.

To identify regions for refinement on the workpiece boundary, we first construct the list of adjacent elements (to each tetrahedral element) from element connectivity information. Using the element adjacency information, the maximum jump in strain rate across each element, E_i, is computed as suggested in equation (35). For the set of inter-element jumps, the median value, E_{med}, is computed as suggested in equation (36). Elements where the jump is greater than the median $(E_i \geq E_{med})$ are candidates for refinement. Let this set of elements be denoted *R* (for refinement set). As noted earlier, only the discretization on the workpiece boundary is refined because interior refine points cannot be specified as input to the version of the MeshCAST™ [8, 9] mesh generator that comes with the Antares™ system. This is a limitation that should be removed for construction of complete adaptive control. Hence, we loop over the faces on the workpiece boundary to determine if any of the elements in *R* are adjacent to any of the boundary faces. Checking if the *adjElem* element of the FE_FACE data structure for the face is in *R* allows us to confirm this. If the *adjElem* element is in *R*, the *forceSplit* flag for the edges

bounding this face (*faceEdges* elements of the FE_FACE data structure) is set to 1, thus forcing mesh refinement.

In the current implementation, we have refrained from coarsening the mesh mainly because we have not implemented safeguards to prevent coarsening of the region that is in close proximity of a die (and is likely to come in contact). Coarsening in these regions could result in a less accurate capture of material flow around the die boundary once contact is developed. Contact regions are not coarsened since a poor approximation of the contact region can result in an inaccurate computation of forming loads. Also, the evolving nature of the forming problems could potentially result in situations where the contact surface (at the time of remeshing) will not remain a contact surface as the process progresses, resulting, potentially, in a less accurate capture of material flow as the simulation progresses. Figure 40 shows the strain rate distribution in the workpiece at an intermediate stage of a valve forging process and Figure 41 shows the new mesh with adaptively refined regions. Full simulation results using this approach to be presented in chapter 6 demonstrate that this provides useful guidance to identify regions for mesh refinement.

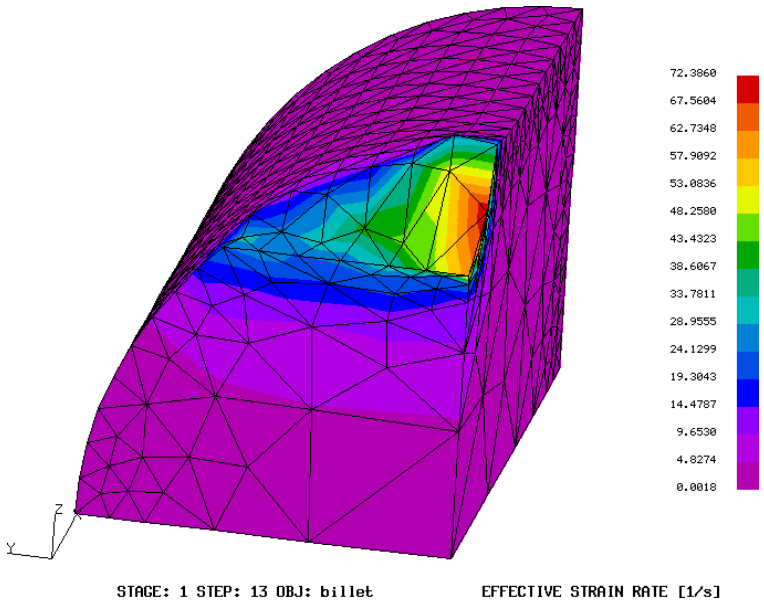


Figure 40: Strain rate distribution in the workpiece

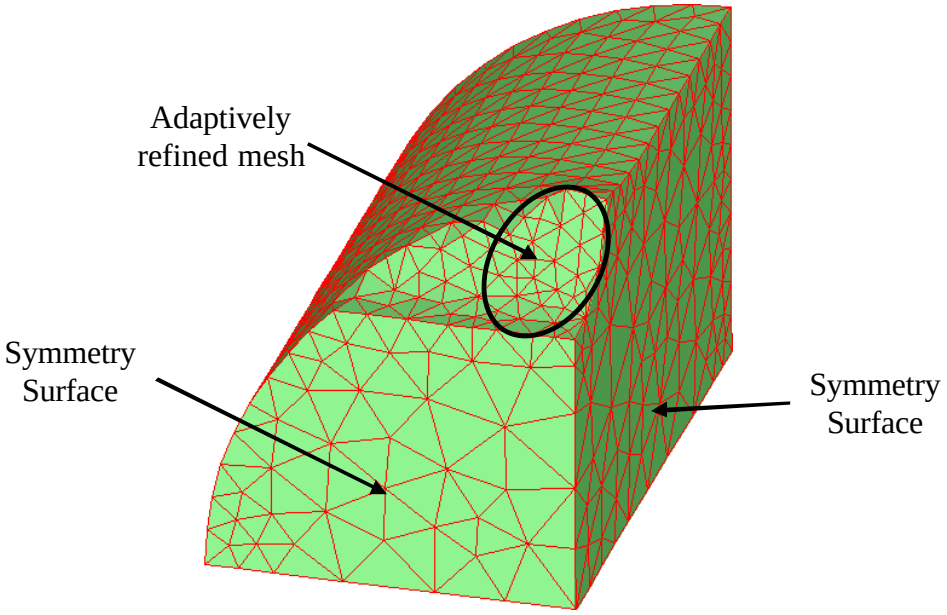


Figure 41: Mesh refinement in (red) regions of high strain rate

5.7.2 Controlling Geometric Approximation Errors

Different approaches to control geometric approximation errors were discussed in chapter 2. The previous chapter discussed the details of using the true geometry of the die and the die overlap monitoring procedure to control these errors during the analysis. Additional procedures have been implemented to control these errors during workpiece geometry update. These include (i) a procedure to refine the mesh due to die overlap, (ii) a look-ahead mesh refinement procedure to refine the mesh on the free portion of the workpiece boundary that is in close proximity to a die, and, (iii) a free-surface compensation algorithm to reposition new nodes generated on the workpiece free surface based on local C^1 surfaces created to capture the local curvature of the deformed workpiece geometry.

To control geometric approximation errors due to die overlap, the procedure described in section 4.4.2 is used again during geometry update. The procedure loops over all edges on the workpiece boundary to identify ones with bounding nodes in contact with the same die (element *contactStatus* of the FE_NODE data structure). The same threshold value is also used here to determine if any of these edges is to be split. To recap, we check if the mid-point of a contact edge is inside the die by a distance greater than 0.29 times the value for the *average mesh size* specified by the user (Figure 23 in section 4.4.2). If the mid-point of the contact edge is inside the die by more than this threshold distance, the finite element edge is forced to split. This is accomplished by setting the *forceSplit* flag of the FE_EDGE data structure to 1.

A schematic illustration of the "look-ahead" mesh refinement procedure is shown in Figure 42. The central theme is to refine the mesh in regions that are likely to come in contact with the die. The first set of candidate edges (that will be forced to split), R_c, consists of the ones in the immediate vicinity of the current contact regions. These are edges that have one of its nodes in contact with a die (*contactStatus* flag of one of the nodes is non-zero). We then build a second list of edges, R_v, that are in close vicinity of the die, with "close vicinity" being defined as a scale factor, f_v, times the *average mesh size* specified for the problem. Obviously, the value of f_v used will control the refinement area. A value of 3 has been selected for f_v and found to give satisfactory results as measured by the volume change due to the approximation of the free surface (V_f in section 3.4). Figure 33 shows the small volume change in the early stages of the disk-forging example when only a small portion of the workpiece boundary is in contact with the die. The mid-point of each edge on the free surface (both bounding nodes have *contactStatus* of zero), is subjected to the geometric proximity check defined in section 4.4.1. If the minimum distance to a die, d, is less than the threshold, the edge is added to R_v. The *forceSplit* flag for the edges in R_c and R_v are set to 1. Refinement due to the look-ahead algorithm is explained for a valve-forging problem with reference to Figure 43 and Figure 44. In order to illustrate the impact of the "look-ahead" algorithm, the simulation is started and the workpiece is forced to remesh after one small time step. Figure 43 shows the initial setup for the problem and identifies the regions where the die is in contact with the workpiece when the simulation begins. Figure 44 shows the workpiece mesh after remeshing and identifies the regions where the free surface has been refined because of proximity to the die.

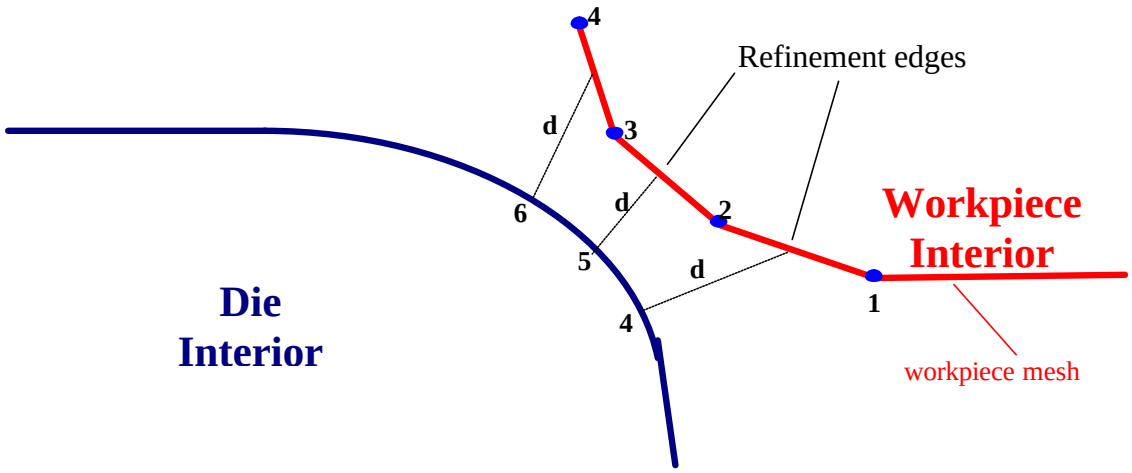


Figure 42: Schematic illustration of the "look-ahead" algorithm

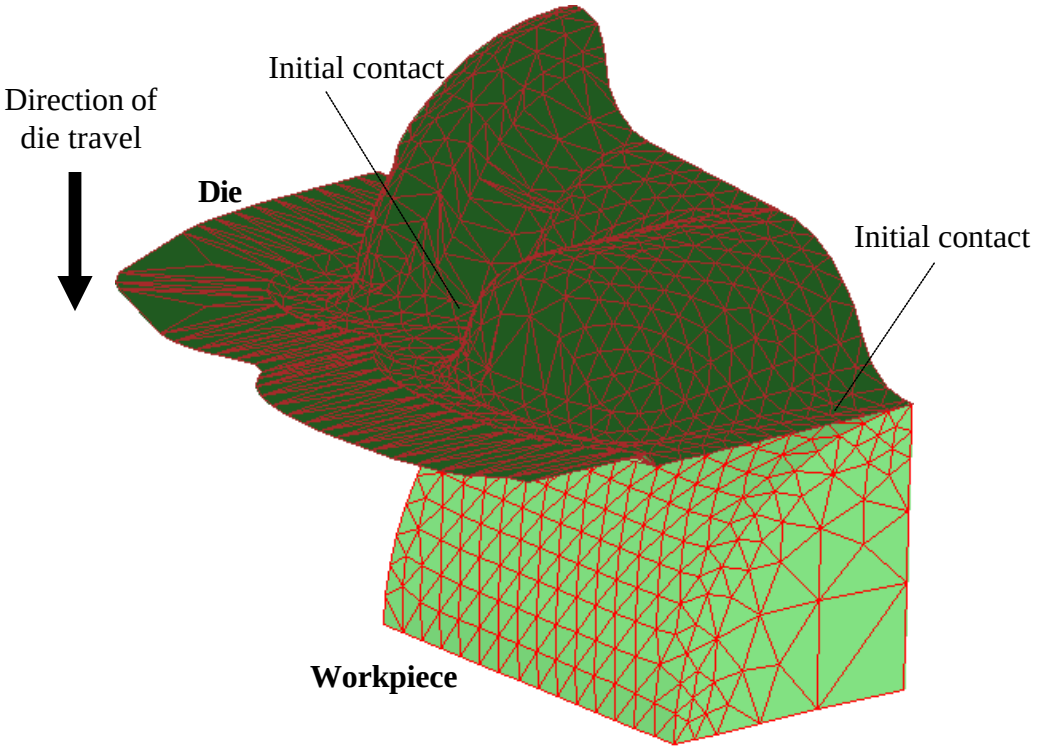


Figure 43: Die-workpiece setup at start of the valve forging simulation

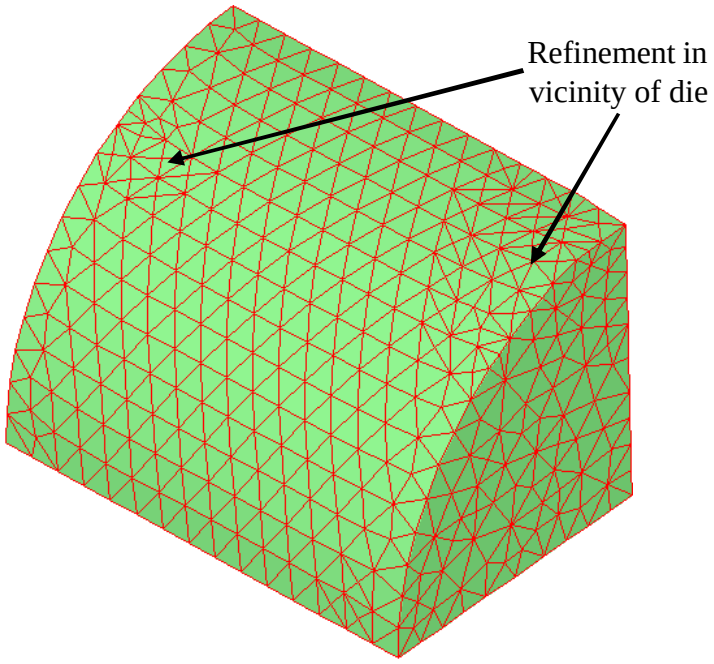


Figure 44: Remeshed workpiece with refinement from the "look-ahead" algorithm

5.8 Free Surface Curvature Compensation

Published literature [38-50] as well as commercial codes [8-12] for 3-D modeling of forming problems do not address the issue of relocating new nodes, that may be generated on the workpiece boundary during remeshing, to reflect the discretization of a smooth boundary. This issue is addressed to some extent in published literature on subdivision surfaces [143, 144] developed specifically with computer graphics applications in mind [145]. Subdivision defines a smooth curve or surface as the limit of a sequence of successive refinements [143, 144]. The refinement (for surfaces) is obtained by splitting each triangle into four according to a particular subdivision rule. In the present work, an alternative approach, termed as *free surface curvature compensation*, has been developed and implemented.

The new nodes generated during the splitting operation are initially located on the mid-point of the edge being split. New nodes classified on the die geometry are relocated on the die boundary as described earlier. New nodes created on the free portion of the workpiece present a different challenge since we do not have a proper curved geometric representation of the workpiece. Leaving the new node at the mid-point does not reflect any surface curvature. This is illustrated for a 2-D case in Figure 45 where node *f* is created at the mid-point of edge *bc*. The free surface compensation procedure adjusts the location of the nodes that lie on the free surface of the workpiece to reflect the discretization of a smooth surface by building a local geometric representation, G_L, of the workpiece boundary using the mesh geometry/topology adjacency data for the edge being split. The local geometric representation, G_L, consists of curves/surfaces constructed using the locations of nodes adjacent to the bounding nodes of the edge being split. Building a local geometric representation at both the end points of the edge being split allows us to approximate the variation in local curvature of the workpiece boundary. The new location of the node being created can then be obtained by projecting the original location of the split point (mid-point of the edge) on to G_L.

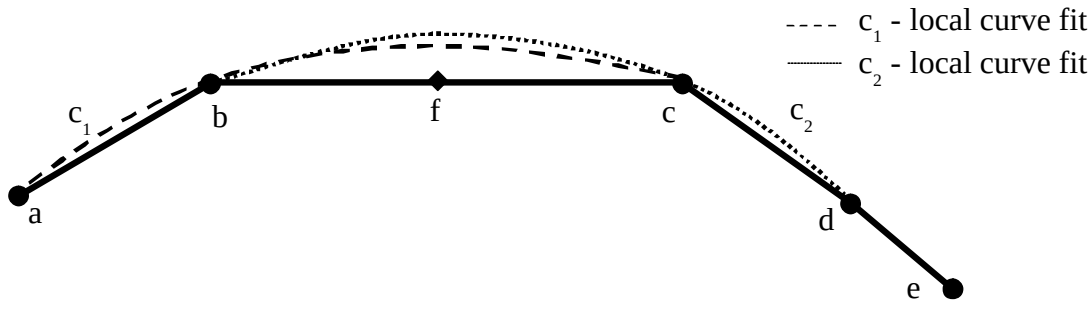


Figure 45: A piecewise local C^1 fit for free surface compensation in 2-D

The geometric representation G_L^i, created at each of the bounding nodes of the edge being split, is a *local piecewise C^1 curve/surface* fit. The actual curves and surfaces created are *standard* parametric curves and surfaces of degree 2. The algebraic form of a parametric quadratic curve segment is given by the polynomials [126, 127]:

$$\begin{aligned} x(u) &= a_{2x}u^2 + a_{1x}u + a_{0x} \\ y(u) &= a_{2y}u^2 + a_{1y}u + a_{0y} \\ z(u) &= a_{2z}u^2 + a_{1z}u + a_{0z} \end{aligned} \qquad \textbf{(35)}$$

where the parameter u is constrained to the interval $\in$ [0, 1]. A unique set of 9 constant coefficients $(a_{ix}, a_{iy}, a_{iz},\ i = 1,2,3)$ called algebraic coefficients determines a unique quadratic curve. Parametric bi-quadratic surfaces are a generalization of the parametric quadratic curves. The C^1 surface is a patch or a curve-bounded collection of points whose coordinates are given by continuous, two parameter, single-valued mathematical functions [126, 127]. The algebraic form of the bi-quadratic surface is given by [126]:

$$p(u,w) = \sum_{i=0}^{2}\sum_{j=0}^{2} \mathbf{a}_{ij} u^i w^j \qquad \textbf{(36)}$$

The parametric variables (u, w) are constrained to the interval u, $w \in$ [0, 1]. The $\mathbf{a}_{ij}$ vectors are the algebraic coefficients of the surface. The reason for the term bi-quadratic is that both parametric variables can be quadratic terms. The UG/Open API [13, 14] provides the necessary functionality to create/query these C^1 curves and surface.

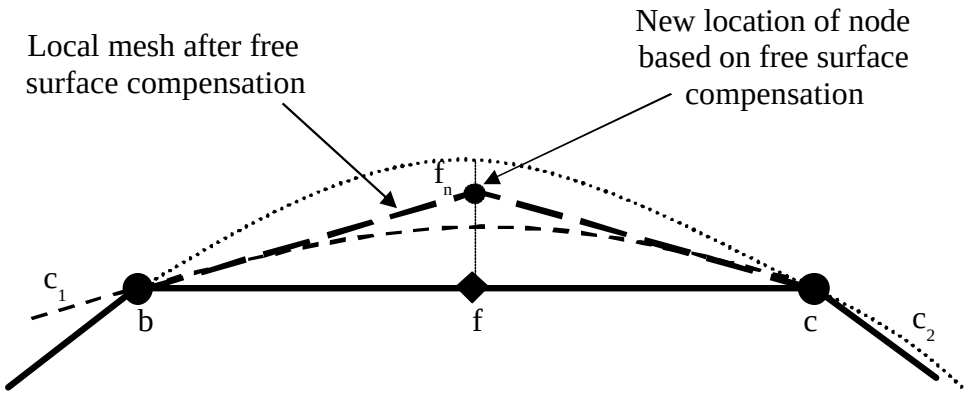


Figure 46: New location of node based on free surface compensation

The procedure to create a local C^1 curve is now described with reference to Figure 45. Let us assume that edge *bc* is to be split. The initial location of the new node, *f*, would be the mid-point of edge *bc*. Before creating the curves, we identify nodes adjacent to the bounding nodes of the edge being split, i.e., nodes adjacent to *b* and *c* but not belonging to edge *bc*. In this case, node *a* is adjacent to node *b* and node *d* is adjacent to node *c*. The first quadratic curve, c_1, is fit through points *a*, *b* and *c* while the second quadratic curve, c_2, is fit through points *b*, *c* and *d*. The curvature compensated location of the new node, f_n (in Figure 46), can then be obtained by projecting the mid-point of edge *bc* onto G_i^i, i.e., the two curves c_1 and c_2. One possibility, for the sake of simplicity, could be to use an average of the projections as the curvature compensated location, i.e.,

$$\tilde{X} = \frac{\sum_{i=1}^{2} \tilde{X}_i}{2} \qquad \textbf{(34)}$$

where $\tilde{X}_i$ is the projected location of the split point on G_L^i and $\tilde{X}$ is the final curvature compensated location for the split point. The new local mesh configuration (in dashed lines) is illustrated in Figure 46.

The Unigraphics [13, 14] option to create a bi-quadratic surface through three curves is used in this implementation and the procedure is described with reference to Figure 47. In order to capture the local curvature, one of the curves has to pass through the node at which the surface is being built and is termed as the *spine curve*. The other two curves are called *boundary curves.* In essence, the three curves represent the surface cross-sections. Procedural details for creating a surface using this method are given below.

Consider the case of the coarse mesh on the boundary of the sphere in Figure 47 where edge *ab* is to be split at point *p*. We now identify the two sets of finite element faces adjacent to nodes *a* and *b*, $\{M_i^0\{M^2\}\}$. These are obtained from the node->face adjacency information stored in the *faceList* array of the FE_NODE data structures for nodes *a* and *b*. In this example, the two sets represented by face connectivity are {*acd, ade, aef, afb, abc*} and {*baf, bfg, bgh, bhi, bic, bca*} respectively. The procedure for creating a C^1 surface, S_1, at node *a* is as follows:

1. Identify the set of nodes that bound the region defined by the above set of faces. The sequence is built using the node->face adjacency information $\{M_i^0\{M^2\}\}$ in the *faceList* array of the FE_NODE data structure for node *a*. For each face in this list, the nodes, other than node *a*, adjacent to the face are identified. This information is obtained from the face->node adjacencies $\{M_i^2\{M^0\}\}$ contained in the *faceConn*

elements of the FE_FACE data structure and only the first occurrence of a node is added to the set. The ordering of the element connectivities allows us to properly sequence the nodes. So, we first add nodes *c* and *d,* from the first face in the face adjacency list, i.e. face *acd,* to the set. Then we identify the face adjacent to edge *ad* that is not face *acd*. In this example, it is face *ade*. From the face->node adjacency information for this face, the node *e* is added to the node set. The procedure is necessary since the faces in the *faceList* array are not ordered in any specific manner procedure and is repeated until all the nodes that bound the region have been identified. In Figure 47, the nodes that bound the region to be defined by S_1 are {*c, d, e, f, b*}.

2. Next, we identify the nodes to be used to define the spine curve, c_s. From the bounding node set {*c, d, e, f, b*}, the best set of points for construction of the spine curve of the surface is determined. Since the spine curve must pass through node *a*, selecting the other two points that are closest to being diametrically opposite to each other (from node *a*) will provide the best local representation. Since this is a 3-D representation, we cannot compute angles between the different edges directly. Instead, we calculate and compare angles between successive edges adjacent to node *a* to find the two edges that subtend an angle closest to 180^{o}. So, in this case, we compute angles between edges *ac* and *ad*, edges *ad* and *ae*, edges *ae* and *af*, edges *af* and *ab* and finally edges *ab* and *ac*. The angle between two edges is obtained by traversing in sequence, i.e., the angle between edge *ac* and edge *ae* is the sum of the angle between edges *ac* and *ad* and edges *ad* and *ae*. In Figure 47, edges *ab* and *ad* represent the set that subtends an angle closest to 180^{o}. So, the spine curve would be

defined through nodes {*d, a, b*}, with node *d* as the first since it occurs first in the bounding node set.

3. Next, we identify the sequence of nodes to define the boundary curves. These are obtained from the bounding node set {*c, d, e, f, b*}. Since the spine curve is defined by nodes {*d, a, b*}, the nodes defining the first boundary curve, c_1, are obtained by traversing forward in the bounding node set from node *d* to node *b*. The nodes for the second boundary curve, c_2, are obtained by traversing backward in the bounding node set from node *d* to node *b*. This procedure ensures that the curves are oriented identically (parametric directions). In our example, the boundary curves are to be generated from nodes {*d, e, f, b*} and {*b, c, d*}.
4. Generate the three C^1 curves passing through {*d, e, f, b*}, {*b, a, d*} and {*b, c, d*}.
5. Generate a C^1 surface through these 3 curves. Providing the proper order of the curves is important in that it defines the parameterization of the surface generated. The order used is $\{c_1, c_s, c_2\}$.

The procedure described above is repeated for node *b* resulting in the creation of surface S_2 as shown in Figure 47. The final location of the split point, p, is the average of the projection of its initial location (mid-point of the edge *ab*) onto surfaces S_1 and S_2. The geometric proximity check defined in the previous chapter is used to compute the projections.

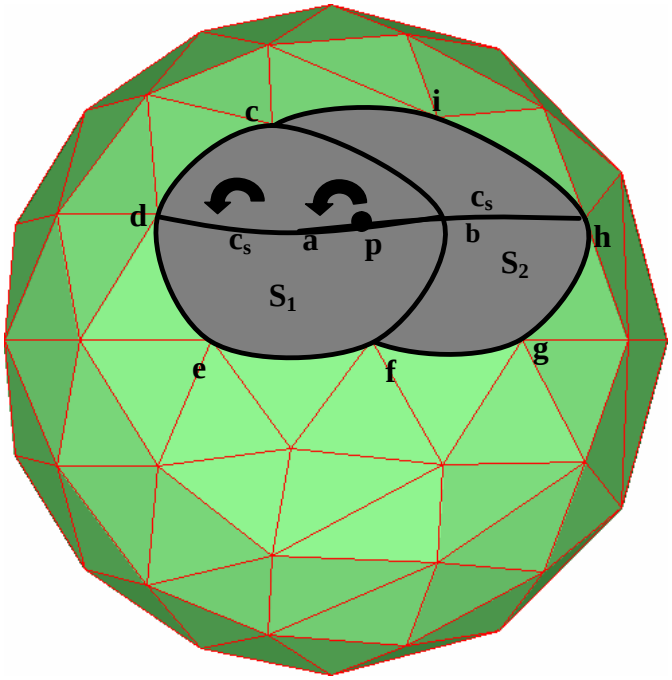


Figure 47: Coarse mesh on the boundary of a sphere

Depending on the local geometry, it may be necessary to use only a local curve fit (instead of a surface fit) as the local geometric representation. The angle between faces adjacent to the edge being split, e_s, is used to decide the type of local geometric representation to be constructed. The central idea here is to identify edges that can be considered *feature edges*, i.e., edges that bound geometric features as shown in Figure 51. A surface fit near feature edges may not capture the local curvature well. The option of using a curve fit for the edges classified on the boundary of model faces ($\{M_i^1 | M_i^1 ?\ \partial\left({}^{W}G_i^2\right)\}$) is not used since the general case of feature edges occurring inside a model face is not handled with this procedure. Two such situations are frequently encountered and need to be anticipated by the geometry update and remeshing procedure in the modeling of metal forming problems. Firstly, surface features such as folds/laps

may form on the workpiece free surface during the forming process. Secondly, the definition of the free surface itself may contain feature edges. For example, in Figure 29, the definition of model face *D* representing the free surface also includes the set of element faces on the plane directly opposite (but hidden in the figure) model face *A* representing a symmetry surface. If the faces adjacent to e_s (elements of the *edgeFaces* array for the FE_EDGE data structure for e_s) are coplanar, there is no need for applying curvature compensation. The angle between faces is computed using the face normal information stored in the *faceNormal* array for the FE_FACE data structure for the adjacent faces. To account for rounding errors, a tolerance of 0.001 degrees is used, i.e., if the angle between adjacent faces is less than 0.001, no compensation is applied. If the angle is greater than the tolerance, we need to decide if this edge can be considered a *feature edge*. An edge is considered to be a feature edge if the angle between its adjacent faces is more than 45^o. This feature edge angle tolerance of 45^o has been selected based on our experience with typical 3-D modeling scenarios where quarter or one-eighth symmetry models may be analyzed which would result in angles between symmetry surfaces of 90^o and 45^o respectively. The following two scenarios exist for e_s:

1. If e_s is not a feature edge, a C^1 surface is constructed at both its end points as described above. This represents a situation with no shape/feature related constraints and hence, the surface fit is used to approximate the local variation in curvature. The procedure described above is used to generate this representation.

2. If e_s is a feature edge, a C^1 curve is constructed at both its end points as described above. This represents a situation with shape/feature related constraints and hence,

the curve fit is used to approximate the local variation in curvature. In order to determine the adjacent points to be used for curve fitting, the procedure similar to the one used to identify points for the spine curve is used. For example, if edge *ab* in Figure 47 was a feature edge, steps 1 and 2 of the surface generation procedure would be applied to the two sets of finite element faces adjacent to nodes *a* and *b*. These adjacent faces are obtained from the node->face adjacency information stored in the *faceList* array of the FE_NODE data structures for nodes *a* and *b*. In this example, the two sets represented by face connectivity are {*acd, ade, aef, afb, abc*} and {*baf, bfg, bgh, bhi, bic, bca*} respectively. The procedure would result in the identification of nodes *d* and *h* and thus sets {*d, a, b*} and {*h, b, a*} to generate the two quadratic curves. Figure 51 illustrates the presence of feature edges and curvature compensation applied to new nodes generated on curved portion of the object.

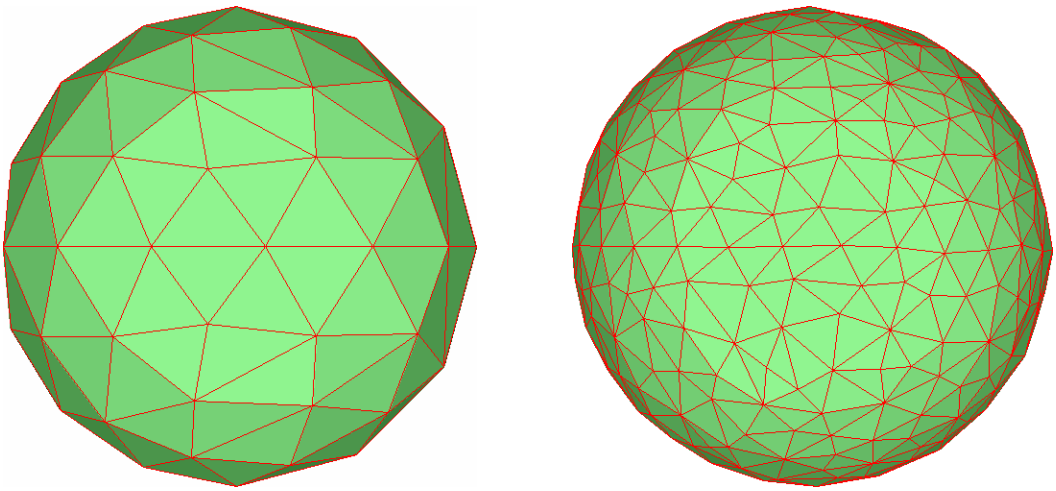
The effect of applying this procedure is demonstrated for the coarse initial discretization of the sphere (of diameter 20 in.) in Figure 48. In this example, the entire boundary is treated as a free surface and hence, free surface compensation is applied to all nodes on the boundary. This has been done to evaluate the effectiveness of the procedure under ideal conditions. The volume enclosed by the coarse mesh is 3816.38 cu. in. as compared to the theoretical volume of 4188.79 cu. in for the sphere. The coarse discretization on the sphere was generated using a commercial mesh generator, MSC-Patran™ version 2000r2 [146] with a mesh size of 5.0 in. The new mesh, for a mesh size of 2.5 in., after application of mesh manipulation operations and curvature compensation is shown in Figure 48. The volume enclosed by the new mesh is 4107.27 cu. in. with an average compensation of 0.19 in. for the 313 new nodes generated on the workpiece boundary.

To compare the effectiveness of this procedure, the sphere was also meshed in MSC-Patran™ where for a mesh size of 2.5 in., the volume enclosed by the mesh was 4098.93 cu. in. The convergence of the sphere volumes, from the curvature compensation procedure described here and from MSC-Patran™, towards the theoretical volume is demonstrated in Figure 49. Note that the big jump in the volume (from 3816.39 to 3995.03 cu. in.) in the first remeshing (for a mesh size of 4.5 in.) with the curvature compensation procedure is to be expected since all edges more than 5 in. in length are split. One of the limitations of the update procedure is its ability to generate good quality elements as we transition from a very coarse mesh (initial mesh size of 5.0 in.) to a very fine mesh (final mesh size of 0.5 in.). This issue is described further in section 5.10 and is the main reason for the difference in volume of about 0.5% for a mesh size of 0.5 in. where the mesh from this procedure results in a volume of 4161.01 cu. in. and the mesh from MSC-Patran™ results in a volume of 4185.00 cu. in. A much better correlation of the sphere volumes from the update procedure and from MSC-Patran™ is observed when the coarse mesh on the sphere is generated using a mesh size of 2.5 in. instead of 5.0 in. The convergence of the sphere volume in this case is illustrated in Figure 50. In this case, the sphere volume using the curvature compensation procedure for a mesh size of 0.5 in. is 4184.06 cu. in. as compared to a volume of 4185.00 cu. in. from MSC-Patran™ and true volume of the sphere of 4188.79 cu. in.

Note that the volume of the mesh from a mesh generator such as MSC-PATRAN™ will converge to the exact value in the limit because the procedure has explicit knowledge of the underlying geometry of the object. On the other hand, the curvature compensation procedure described here has no prior knowledge of the object geometry and hence,

cannot be expected to provide a mesh whose volume converges to the exact value in the limit. This procedure has been developed specifically for use in the analysis of evolving geometry problems where geometry information is not available during remeshing but the new discretization must reflect the physical reality of a smooth boundary.

Results for another test case to illustrate refinement along curved portions of the object boundary are shown in Figure 51. The theoretical volume is 2888.83 cu. in and the volume computed from the initial mesh of 4.0 in. is 2845.77 cu. in. The volume, with a mesh size of 2.0 in., of the remeshed object in Figure 51 is 2870.35 cu. in. The efficacy of free surface compensation procedure for arbitrary geometries is confirmed by small volume change (Figure 33) in the early stages of the disk-forging example when only a small portion of the workpiece boundary is in contact with the die. Simulation results applying this procedure to modeling of forming problems are given in the next chapter.



Coarse mesh **Curvature compensated mesh**

Figure 48: Curvature compensation on a sphere

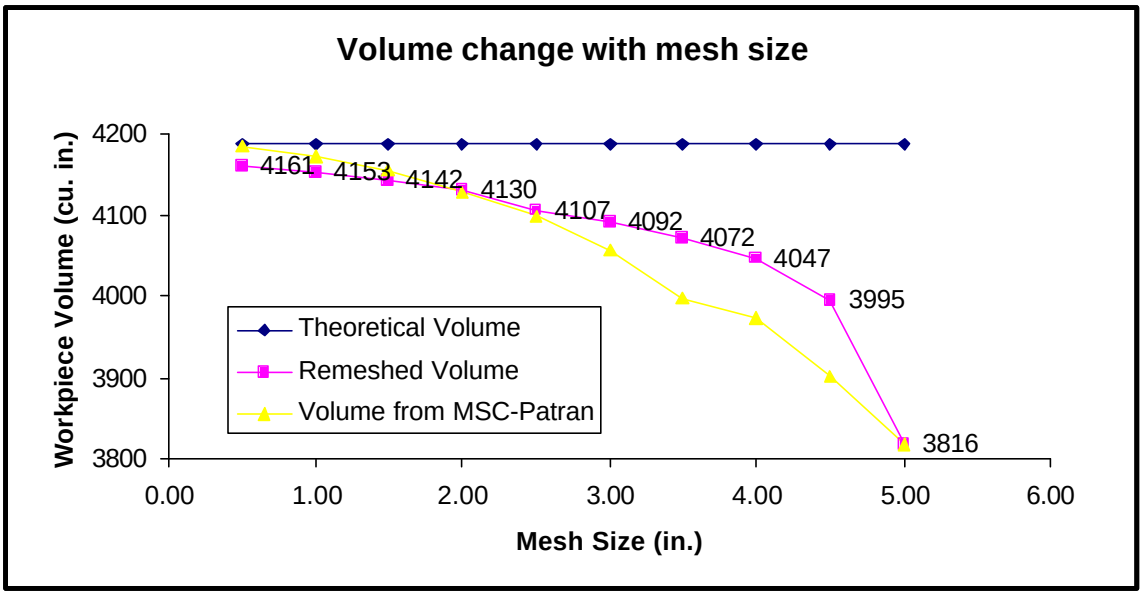


Figure 49: Sphere volume change with initial mesh size of 5.0 inches

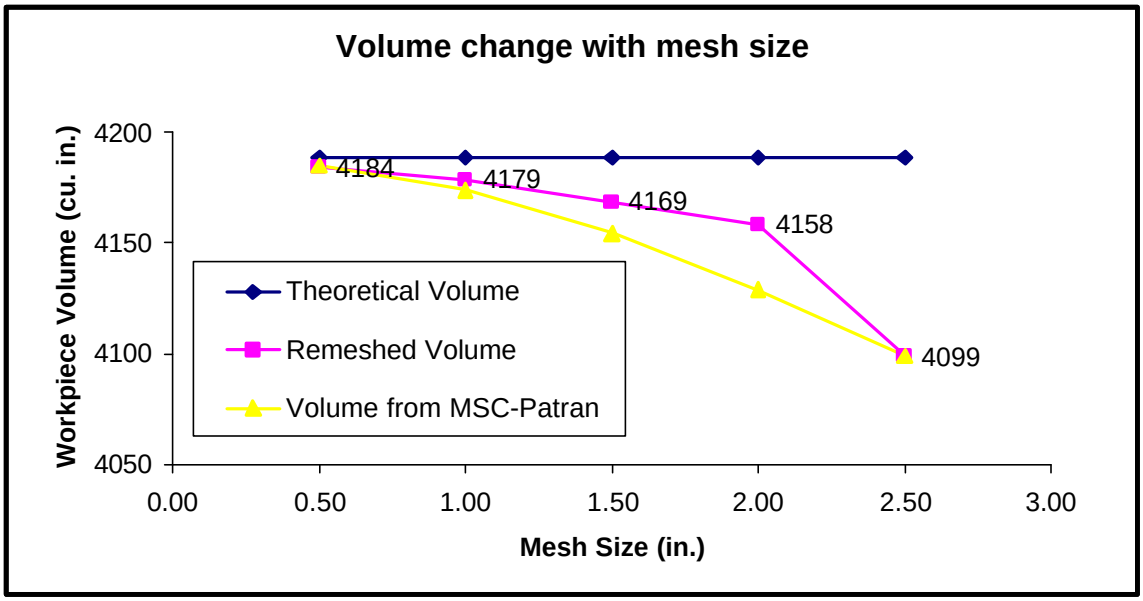


Figure 50: Sphere volume change with initial mesh size of 2.5 inches

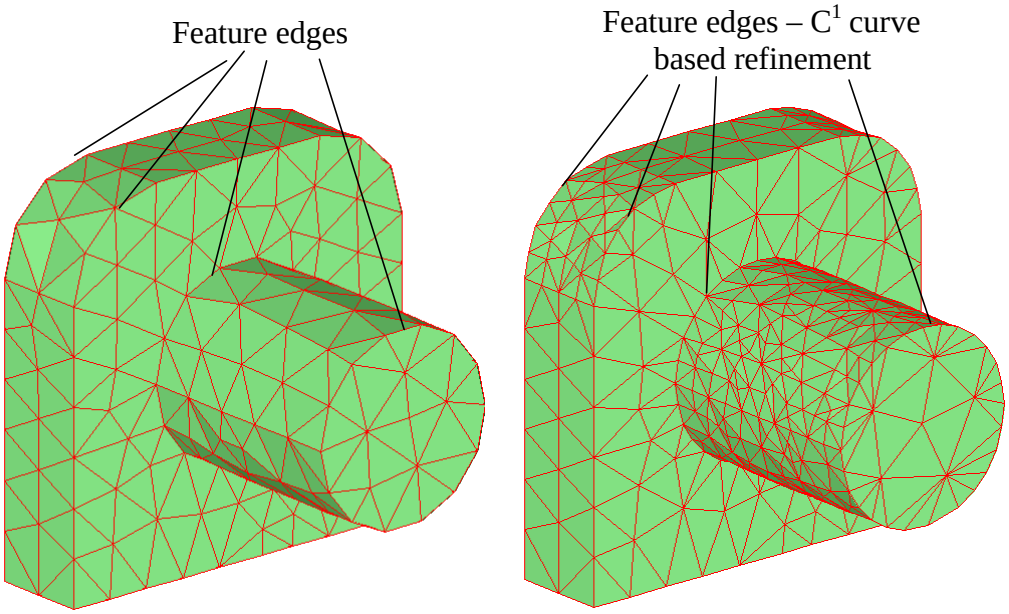


Figure 51: Curvature compensation on test geometry

5.9 Discretization of the Updated Workpiece

Using the procedure described in this chapter so far, the updated representation of the boundary of the workpiece is obtained. However, for the continuation of the analysis, we need to fill the workpiece volume with 4-node tetrahedrons and prepare the required input information. In our case, the Antares™ solver is to be provided with the following information:

1. Mesh model information: This includes the coordinates of all the nodes and connectivity information for all of the tetrahedral elements representing the new mesh.

2. Boundary condition information: This includes providing the *mesh surface* information (list of elements on which a boundary condition is defined) identified by names and contact status information.

Note that the history dependent solution parameters are not provided since the Antares [8, 9] solver interpolates the parameters from the deformed to the remeshed workpiece geometries before proceeding with further analysis.

Once the workpiece boundary is defined by valid 3-node triangles after geometry update, the volume of the workpiece can be filled with tetrahedral elements by an automatic mesh generator. The MeshCAST™ [8, 9] automatic mesh generator is used to fill the workpiece volume with tetrahedral elements. It is to be noted that another automatic mesh generator could easily be used provided it is able to generate a tetrahedral mesh for the deformed workpiece using boundary representation provided in the form of 3-node triangles.

The data structures of Figure 26 and Figure 27 are key to consistent transfer of boundary conditions. These data structures are constantly updated during the geometry update process. The model faces identify the *mesh surfaces* on which boundary conditions are defined. The contact status of a node is available from the *contactStatus* element of the FE_NODE data structure. The volume mesh and the boundary condition information maintained through the geometry update procedure are used to construct the input files for continuation of the analysis.

5.10 Limitations of the Geometry Update Procedure

The geometry update procedure discussed in this chapter presents a method to update the workpiece geometry and generate a new mesh representation to continue the analysis. Limitations to this implementation include:

1. **Mesh adaptation:** Gradation or transition of the workpiece meshes from a coarse mesh to a fine mesh and vice versa, especially on the workpiece free surface, could be a potential problem encountered with the use of this approach. Figure 52(b) shows the degradation in element quality when the element configuration of Figure 52(a) is remeshed with two edge splits. The fundamental reason for this is that the updated boundary (mesh) for the workpiece is obtained by applying a set of mesh manipulation operations on the mesh representation of the deformed workpiece and not by building a geometric representation and then discretizing that representation. Hence, the transitioning cannot be enforced over a wider region which would likely help in improving element shapes. For example, in this procedure, the nodes on the free surface of the deformed workpiece are fixed since these are locations obtained from analysis results. When there are sudden, large changes in mesh size requirements on the free portion of the workpiece boundary (either user specified or based on the indications of the adaptive mesh refinement algorithm), these known locations can act as constraints and affect element quality (if the procedure is to maintain mesh size requirements). The impact of this limitation is seen in the volume convergence pattern of Figure 49 where the volume from the update procedure is closer to the theoretical volume than the volume of the mesh generated in MSC-

Patran™ till a mesh size of 2.0 in. However, when the required mesh size is reduced further, the volume from the update procedure lags the volume of the mesh generated in MSC-Patran™. To overcome this limitation, improved mesh control procedures are needed to handle regions with sudden transitions in mesh size. The current approach is to limit the variation in mesh sizes and require that the user provide an initial mesh for the workpiece that does not contain a large variation in mesh sizes in different regions of the workpiece. This requirement may necessitate putting more elements to define the workpiece, but any affect on CPU time is more than offset by the savings achieved by elimination of manual remeshing.

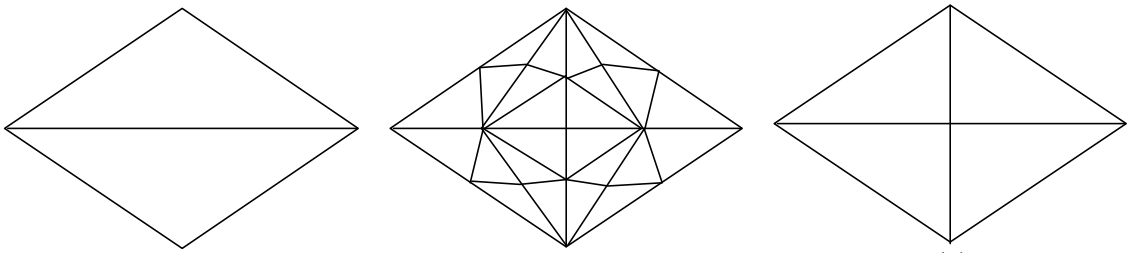


Figure 52: Effect of large changes in mesh size on element quality

2. **Definition of the workpiece boundary:** The procedure used is applicable only when the boundary of the workpiece is defined by straight-edged triangles. For example, if the workpiece were to be defined by hexahedral elements, its boundary would be represented by quadrilateral elements. If the workpiece were to be defined by 10 node tetrahedral elements, its boundary would be represented by 6 node triangles. In both of these cases, the procedure cannot be used as described and the geometry update procedure may need significant changes.

3. **Full remeshing of workpiece:** Even though the deformed workpiece model may contain only a few distorted elements most likely in the vicinity of the boundary, the entire workpiece boundary is updated and the volume re-discretized. This imposes a need for transferring solution data from the deformed mesh onto the new mesh representing the workpiece, an interpolation process that is a potential source of errors as explained in section 2.4 of chapter 2. To avoid accumulation of rezoning errors, it may be beneficial to develop algorithms that focus on a local remesh in regions of element distortion. This may result in some improvement to the solution accuracy as well as a potential reduction in CPU time.

4. **Adaptive remeshing:** The adaptive remeshing procedure implemented only refines the mesh on the workpiece boundary and ignores the fact that there may be regions in the interior of the workpiece that may need refinement. Though the error indicator is already capable of identifying interior regions for refinement, a different volume mesh generator, capable of using this information to refine the interior mesh, is needed.

Chapter 6

RESULTS FROM THE AUTOMATED MODELING SYSTEM

There are several issues that determine the accuracy of the overall finite element solution as compared to the actual forming process. Theoretical considerations related to the finite element formulation, modeling of material constitutive behavior, friction, accuracy of data transfer across remeshes, and volume preservation during analysis, etc. will impact simulation accuracy. In addition, the ability of the geometry update procedure to provide a new mesh for the deformed workpiece while (i) maintaining total workpiece volume and its distribution, and, (ii) effectively transferring analysis attributes necessary to continue the analysis plays a significant role in the overall accuracy of the simulation.

This chapter discusses simulation results from the system, specifically concentrating on central research focus of this thesis - the geometry update and remeshing procedure. We will first demonstrate the impact of the special procedures developed for (i) controlling geometric approximation and mesh discretization errors, and (ii) curvature compensation on the workpiece free surface. Later, full simulation results, performed using all error control procedures and curvature compensation, are presented for two forming processes.

6.1 Impact of Procedures to Control Simulation Errors

The error control procedures used to improve simulation accuracy include procedures for adaptive mesh refinement, die-overlap monitoring, look-ahead mesh refinement, and

curvature compensation of new nodes on the workpiece free surface. In order to facilitate evaluation of the impact of these error control procedures, process simulations have been run for a valve body forging with and without each of the error control procedures. The first simulation was run with no controls, i.e., no die-overlap monitoring during analysis and refinement based on this procedure during geometry update, no look-ahead and adaptive mesh refinement and curvature compensation during geometry update. This procedure would give us the set of baseline results with which the results from the second simulation, run with the specific error control measure being monitored, can be compared. The starting configuration for the valve body forging process is shown in Figure 53.

6.1.1 Adaptive Mesh Refinement

The procedure to control discretization errors refines the mesh based on jumps in element strain rates. Refining in regions based on jumps in strain rates allows us to hone in on regions of action (at the time of remeshing), thus enabling the capture of surface defects such as laps/folds. To evaluate the effectiveness of the strain rate based error indicator, jumps in element strain rates are plotted before and after a remesh. The expectation would be that the jumps in strain rate would be smaller in the regions that were refined using the error indicator to guide mesh refinement. Since there is no way in Antares™ to get interpolated strain rate values for the remeshed workpiece without continuing the analysis, the simulation is continued with the new mesh for a small time step. For the present example, the first remesh was needed at a die stroke of 24%. The deformed workpiece geometry at 24% die stroke is shown in Figure 55 along with its strain rate

distribution. The corresponding jumps in element strain rates for this configuration are shown in Figure 56. The maximum strain rate jump is 18.082 and the minimum jump is 0.0 giving a median jump of 9.041. The error indicator identifies regions with jumps more than this median value for refinement. The remeshed workpiece is shown in Figure 56. The distribution of jumps in element strain rates on the new mesh, after a small time step, is shown in Figure 57. Note that the region of high strain rate jumps has shifted (since the results are not for the same time value) and the jumps in strain rates, in regions that were refined, have gone from a high of 18 before remeshing to between 1.0 and 5.0.

Since this procedure typically results in more elements in regions of deformation, it also has an impact on controlling workpiece volume during the simulation. The variation in workpiece volume for the two cases is shown in Figure 58. In the simulation where no controls are used, the workpiece lost 3.29% at the end of die stroke compared to 2.29% for the case when all controls were used, a difference of 30.39%. More importantly, the difference is pronounced till around 75% of stroke when material is just beginning to flash out. At a die stroke of 75%, the volume loss is 0.84% compared to 0.21%, a reduction of 75% due to use of the adaptive mesh refinement procedures. The spike in volume loss is due to the sharp transitions in die geometry in the flash region. In this case, the use of monitoring die overlap would trigger a remesh much each earlier resulting in better control over volume loss.

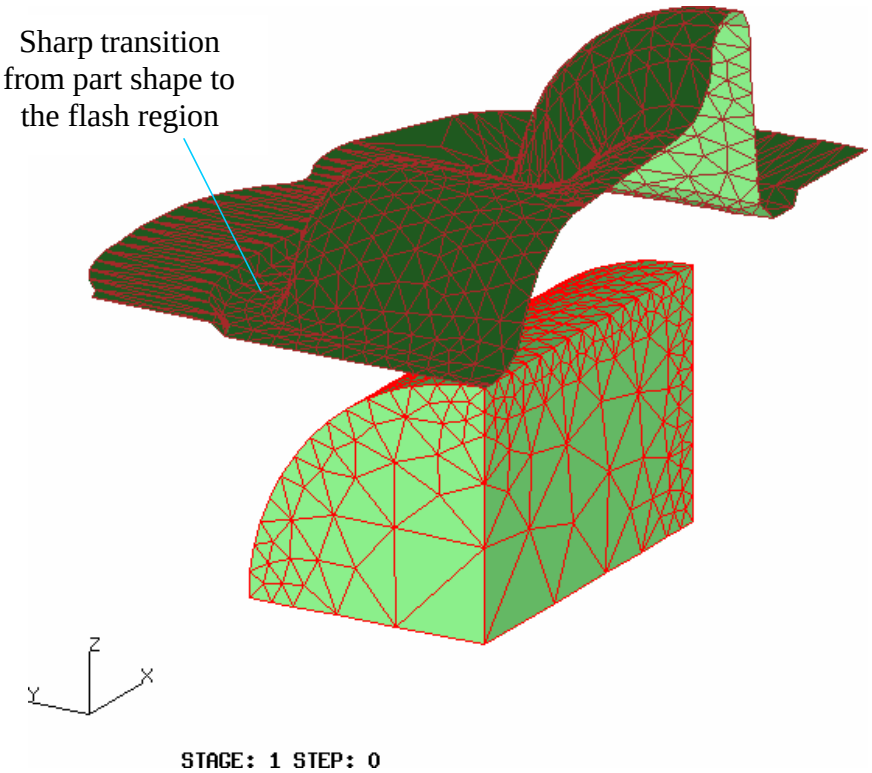


Figure 53: Initial configuration for the valve-forging problem

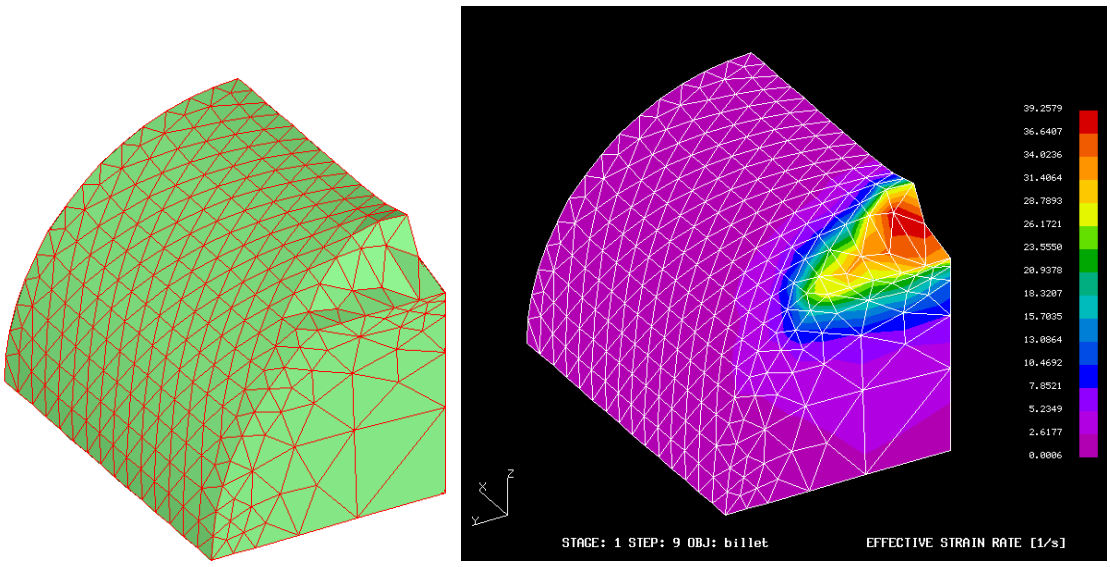


Figure 54: Deformed workpiece at 24% die-stroke (1st remesh)

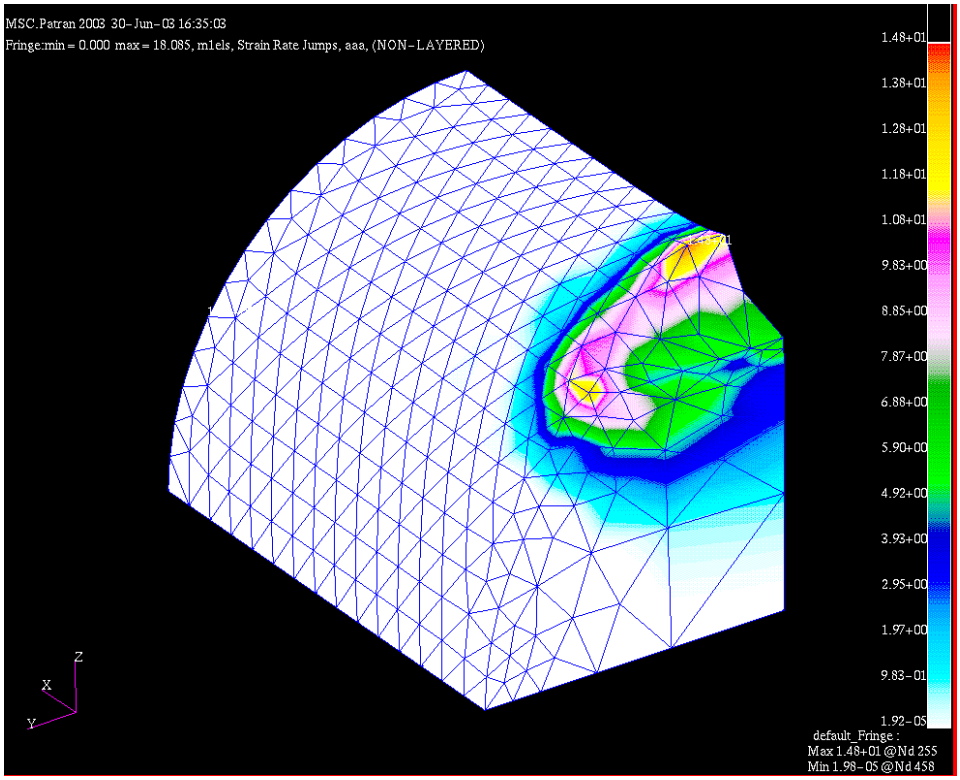


Figure 55: Jumps in strain rates at 24% die stroke

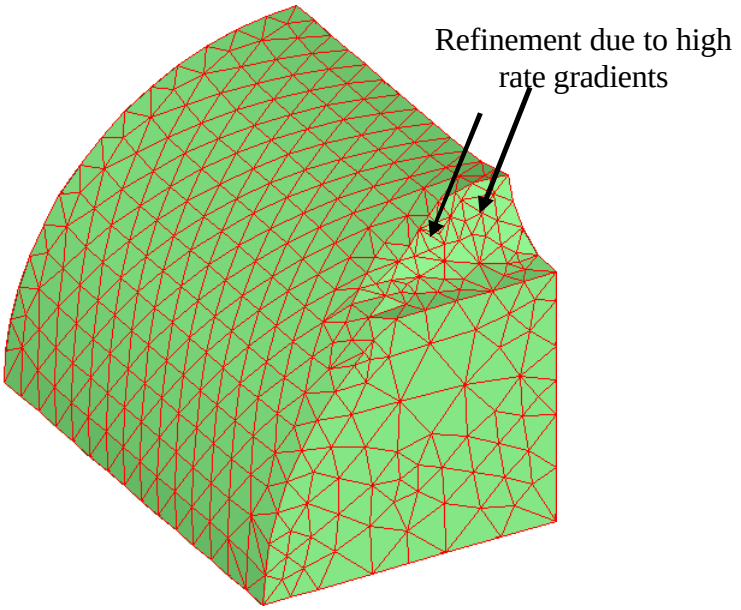


Figure 56: Strain rate based adaptive refinement of workpiece mesh

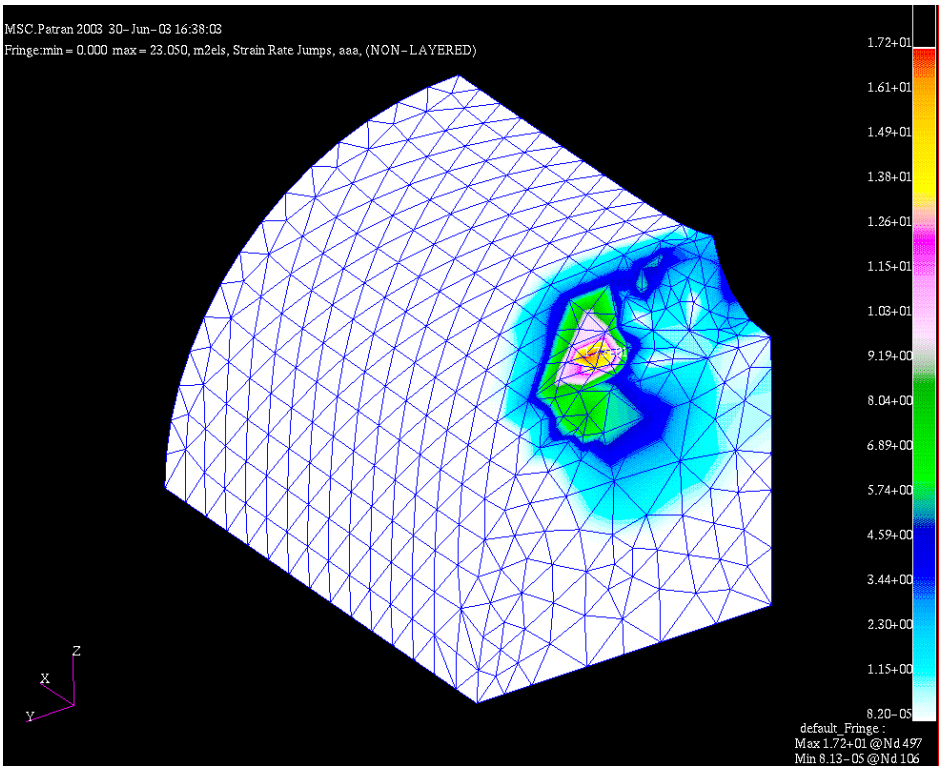


Figure 57: Strain rate jumps after remeshing (at 24% stroke)

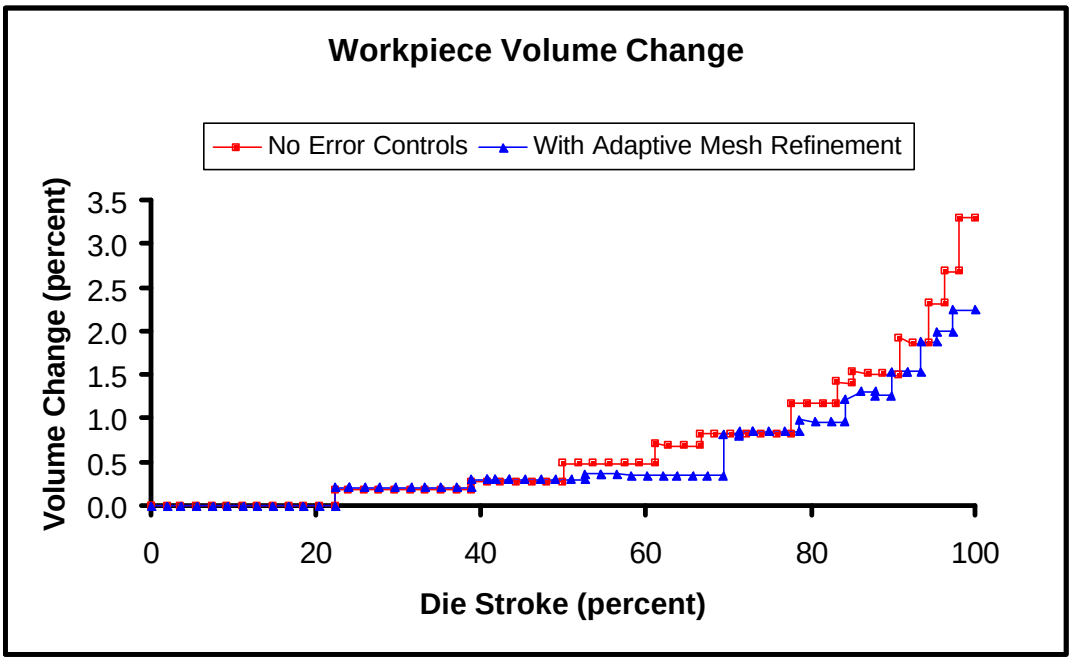


Figure 58: Impact of adaptive mesh refinement on workpiece volume

Use of the adaptive mesh refinement procedure also resulted in better prediction of die loads. Results (load-stroke curve) from simulations for an average mesh size of 0.40 inches for the cases with no error controls and only adaptive mesh refinement are shown in Figure 59 along with the load-stroke curve from a simulation run with no error controls and a mesh size requirement of 0.25 inches. At the end of die-stroke, the simulation predicted a die load of 282.15 Kips for the case with a mesh size of 0.40 inches and no controls, as compared to a load prediction of 314.68 Kips for the case with adaptive mesh refinement (a difference of 11.53%). The load at the end of die stroke was 386.25 Kips for the case with a mesh size of 0.25 inches and no error controls. The results for the final 5% of the stroke when the loads typically spike up as the material flashes out and the die cavities fill, are shown in Figure 59.

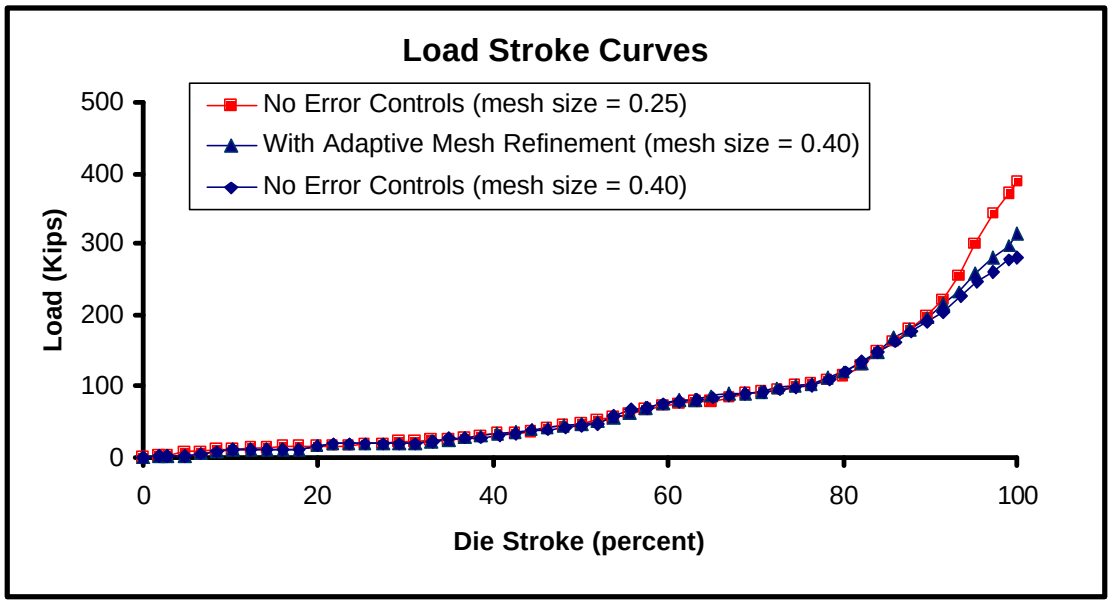


Figure 59: Load-stroke variation with adaptive mesh refinement

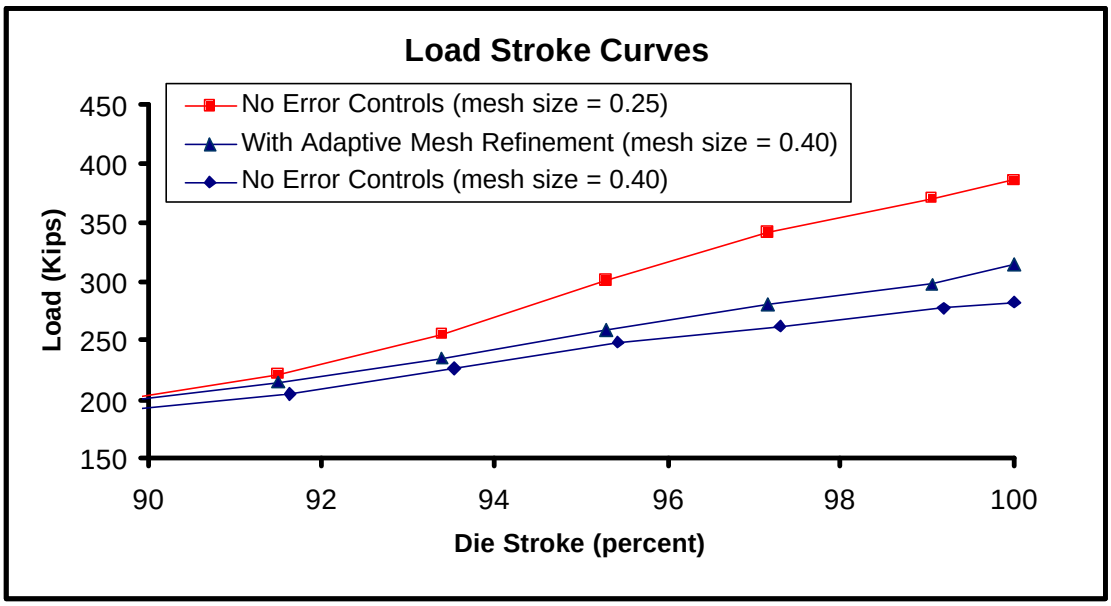


Figure 60: Load-stroke variation with adaptive mesh refinement – final 5%

6.1.2 Die-Overlap Monitoring

The die-overlap monitoring procedure is implemented in the solver and also in the geometry update procedure. In the solver, a remesh is triggered when midpoints of contact edges have penetrated the die by more than a threshold distance of 0.29 times the *average mesh size* specified by the user for the problem. The geometry update procedure then splits these edges. The premise here has been that triggering a remesh would result in reducing workpiece volume loss during the simulation occurring due to the need to enforce consistency.

To evaluate the impact of this procedure, simulations were run with and without this control for two separate mesh sizes. The first case was with an average mesh size of 0.25 inches and the second was with an average mesh size of 0.40 inches. For the simulation

with a mesh size of 0.25 inches, the first remesh is triggered at a die stroke of 17% compared to a die stroke of 22% for the case with no error controls. For the simulation with a mesh size of 0.40 inches, the first remesh is triggered at a die stroke of 13% compared to a die stroke of 35% for the case with no error controls. As can be seen in the two charts for workpiece volume change in Figure 61 and Figure 62, the workpiece loses less volume at the end of the simulation when die-overlap monitoring is used compared to the case when no error controls are used. The workpiece volume at the end of simulation for a mesh size of 0.25 inches was 3.29% with no error controls compared to 2.30% when die-overlap monitoring was used, a difference of 30.09%. For a mesh size of 0.4 inches, the volume loss at the end of the simulation was 7.68% with no error controls and 4.42% when die-overlap monitoring was used, a difference of 42.45%.

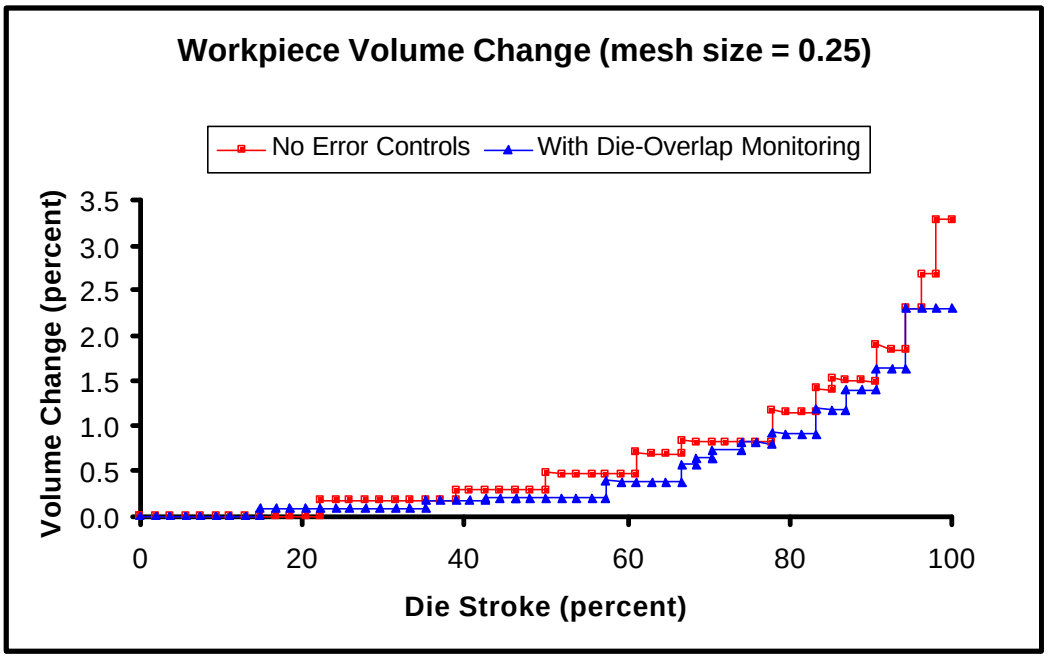


Figure 61: Workpiece volume change (mesh size = 0.25) – overlap monitoring

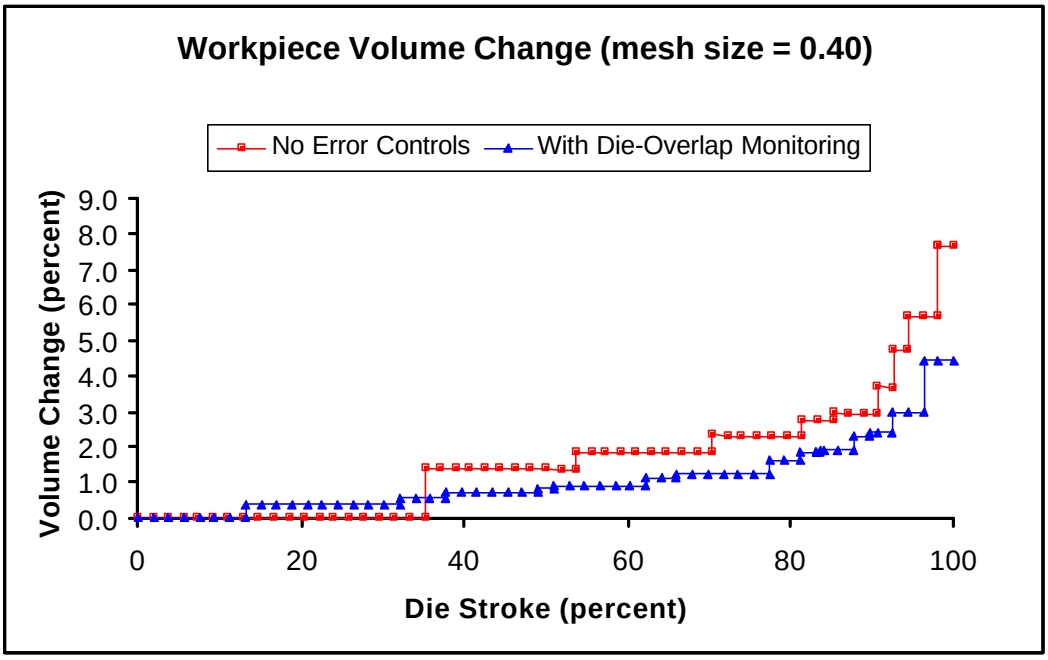


Figure 62: Workpiece volume change (mesh size = 0.40) – overlap monitoring

6.1.3 Look-Ahead Mesh Refinement

The look-ahead mesh refinement procedure is implemented in the geometry update procedure and refines the mesh on the free surface of the workpiece that is in close proximity of the die. The premise here has been that this refinement would result in better representation of die-workpiece contact as the workpiece geometry evolves during the process and reduce workpiece volume loss by anticipating contact and avoiding situations that may potentially result in significant die-overlap.

Once again, simulations of the valve body forging process were run with mesh size requirements of 0.25 inches and 0.4 inches respectively with only look-ahead mesh refinement. The variation in workpiece volume as the simulation progresses for the two

cases is shown in Figure 63 and Figure 64. For the case with a mesh size of 0.25 inches, the initial impact of the look-ahead mesh refinement procedure is not noticeable due to the nature of the contact geometry. The initial contact with the die is in locations that do not have sharp transitions. However, as the simulation progresses and contact ensues with regions of sharp transitions, the effect of the look-ahead refinement is evident in the resulting lower volume loss. In this case, at the end of die stroke, the workpiece lost 2.53% compared to a volume loss of 3.29% for the case without any error control procedures, a difference of 23.10%. For a mesh size of 0.4 inches, the look-ahead mesh refinement results in a better representation of the die-workpiece contact after the first remesh and throughout the simulation. The workpiece lost 5.30% with look-ahead mesh refinement compared to 7.68% with no error-controls, a difference of 30.98%.

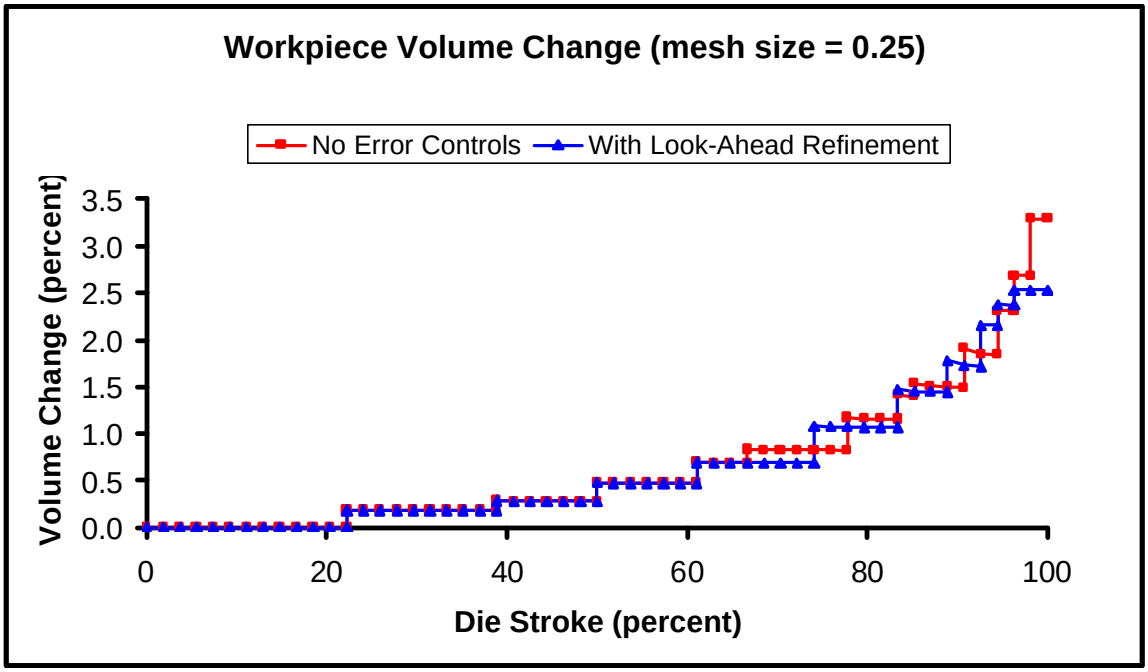


Figure 63: Workpiece volume change (mesh size = 0.25) – look-ahead refinement

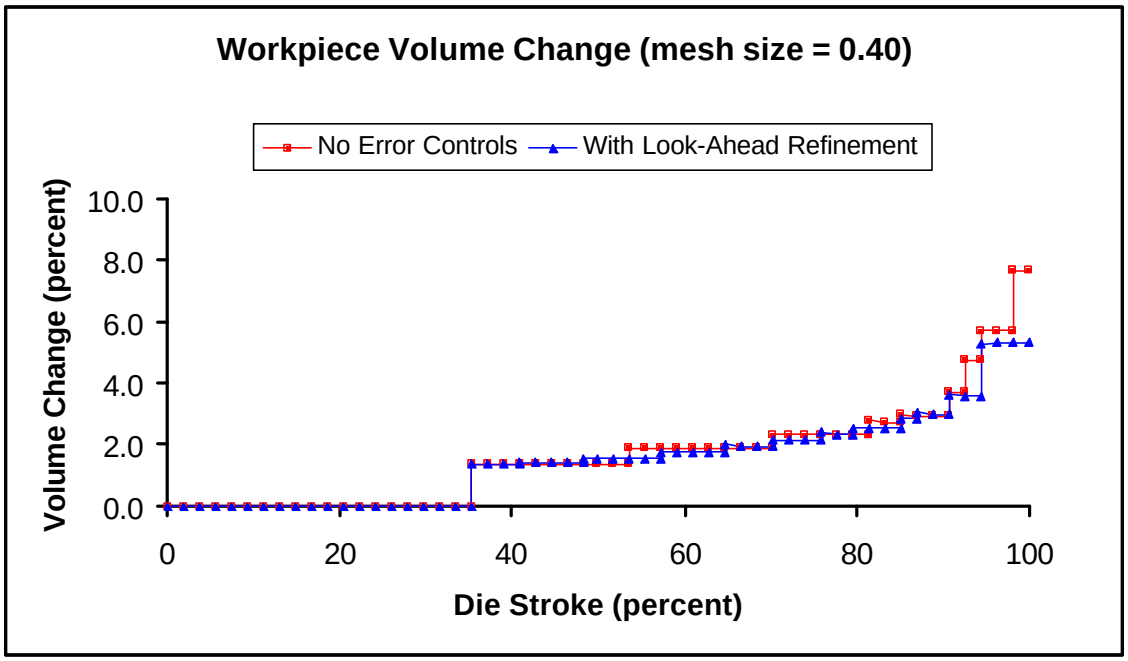


Figure 64: Workpiece volume change (mesh size = 0.40) – look-ahead refinement

6.1.4 Curvature Compensation on Workpiece Free Surface

Curvature compensation is applied only to new nodes generated on the workpiece free surface. The location of these nodes is modified to reflect the discretization of a smooth boundary. Hence, the curvature compensation procedure contributes to volume preservation only when new nodes are generated on the workpiece free surface. To evaluate its effectiveness, the valve body forging analysis was run for the case where the initial mesh size was 0.4 inches. The contribution of this procedure towards the control of overall workpiece volume will diminish as a bigger portion of the workpiece boundary comes in contact with the die. Volume changes during the later stages are due to the need for enforcing consistency in the contact regions. Hence, for this case, changes to workpiece volume are compared only until a die stroke of 70% because a significant

portion of the workpiece is in contact with the die at this point. The impact of the curvature compensation procedure on the workpiece volume is evident in Figure 65. Using curvature compensation resulted in a volume loss of 1.65% at a die stroke of 70% as compared to a loss of 1.87% when no curvature compensation was applied to the new free surface nodes, a reduction of 11.8%.

The curvature compensation procedure was also evaluated by running a simulation with all error control procedures except curvature compensation and another simulation with all error control procedures including curvature compensation. Without curvature compensation, the workpiece lost 0.29% as compared to 0.06% with curvature compensation, a reduction of 79.3%. Note that by 65% die stroke, the workpiece was remeshed 7 times in both cases.

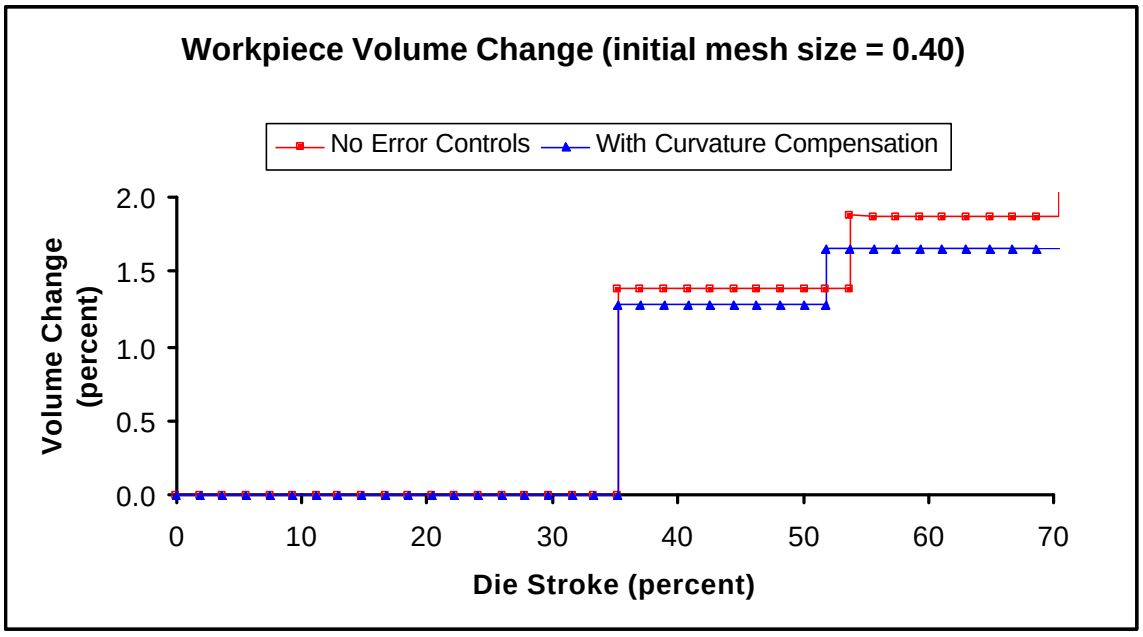


Figure 65: Workpiece volume change (mesh size = 0.40) – curvature compensation

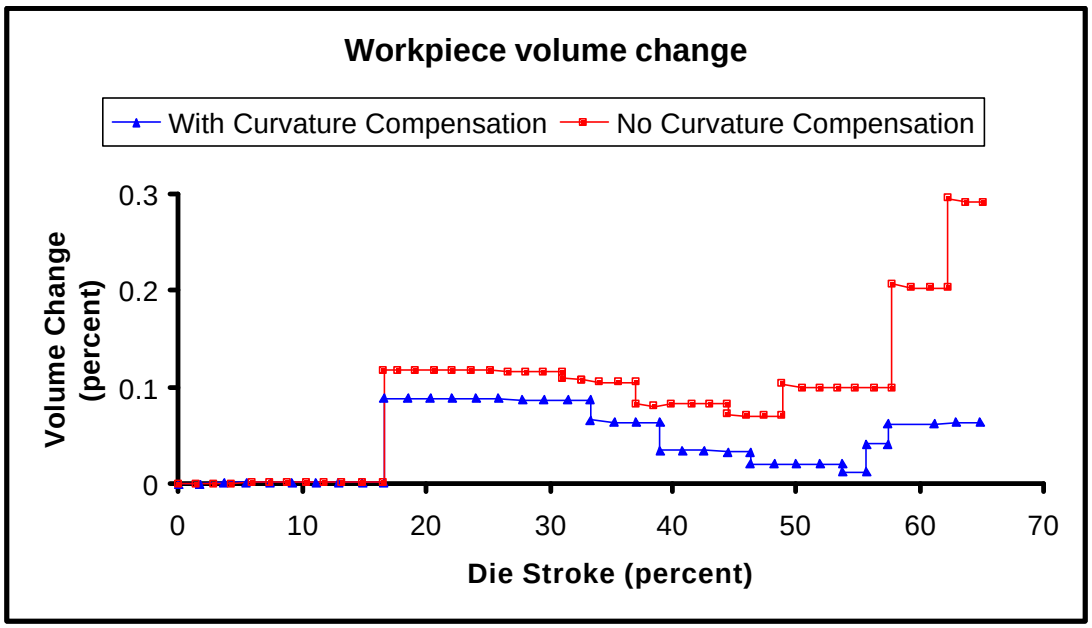


Figure 66: Workpiece volume change (mesh size = 0.25) – curvature compensation

6.1.5 Combined Effect of Error Control Procedures

The combined effect of the error control procedures has been evaluated by running simulations with all error control procedures and comparing the results with those of simulations run with no error control procedures. Once again, simulations were run with a mesh size of 0.25 inches and 0.4 inches. The variation in workpiece volume for the two cases is shown in Figure 67 and Figure 68. The results are consistent with our expectations in that the simulation with no controls shows the maximum volume loss and the simulation with all of the geometric approximation and discretization error controls show the least amount of volume loss.

For the simulation with a mesh size of 0.25 inches, the workpiece lost 3.29% at the end of die stroke when no error controls were used as compared to 1.83% for the case when all

controls were used, a reduction of 44.38%. More importantly, the difference is even more pronounced till around 75% of stroke when material is just beginning to flash out. At a die stroke of 75%, the volume loss is 0.91% compared to 0.21%, a reduction of 76.92% due to use of error control procedures. For the simulation with a mesh size of 0.40 inches, the workpiece lost 7.68% at the end of die stroke when no error controls were used as compared to 3.19% for the case when all controls were used, a reduction of 58.46%. In this case also, the difference is pronounced till around 75% of stroke when material is just beginning to flash out. At a die stroke of 75%, the volume loss is 2.33% compared to 0.86%, a reduction of 63.09% due to use of error control procedures.

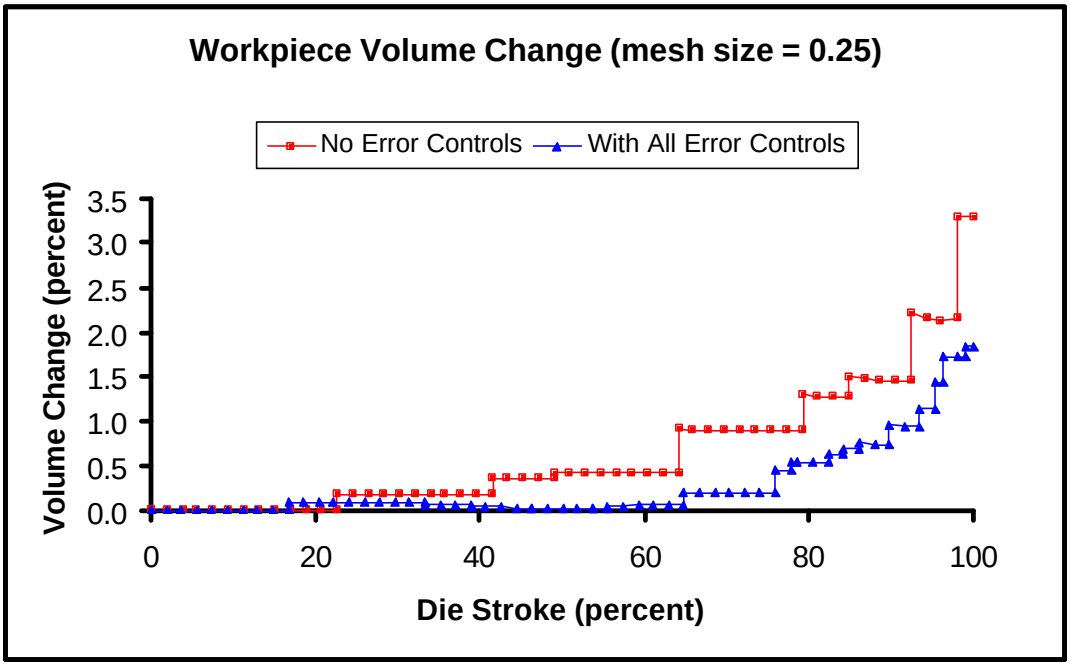


Figure 67: Combined effect of error control procedures – mesh size of 0.25 inches

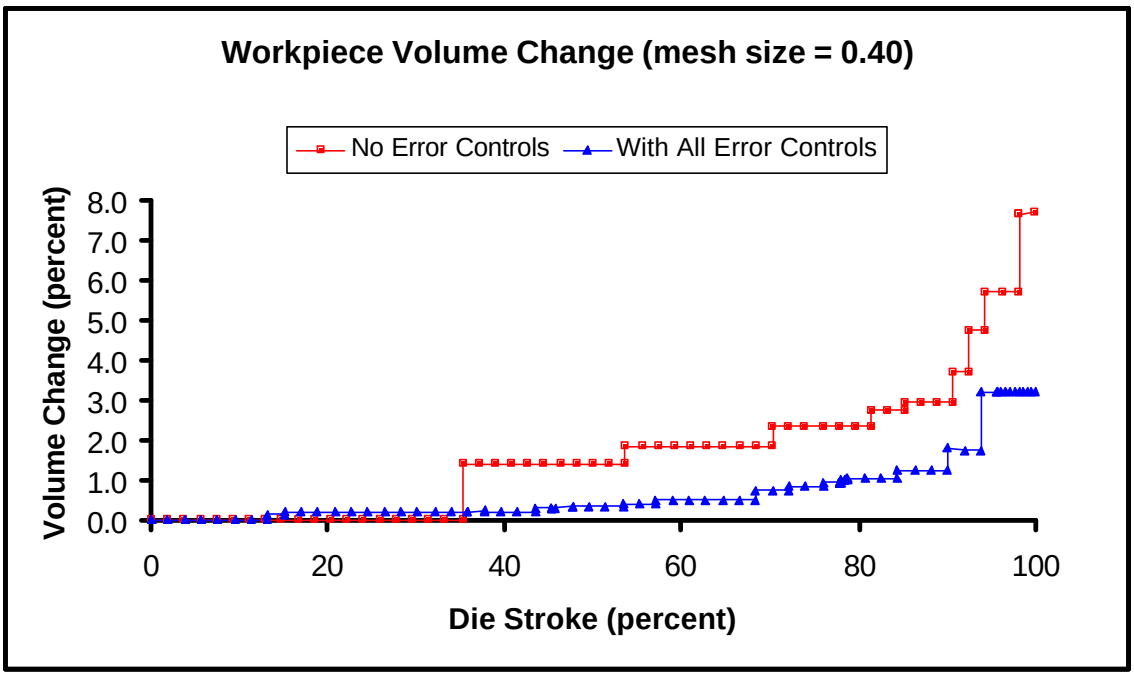


Figure 68: Combined effect of error control procedures – mesh size of 0.40 inches

The procedures to control geometric approximation and mesh discretization errors have a significant impact on the simulation accuracy as the workpiece evolves during the process simulation. Use of the error control procedures also resulted in better prediction of die loads. The full load-stroke curves when a mesh size of 0.40 inches is used are shown in Figure 69. At the end of die-stroke, the simulation predicted a die load of 282.157 Kips when no error controls were used, as compared to a load prediction of 356.02 Kips when all error controls were used (a difference of 26.18%) and 386.25 Kips for the case with a mesh size of 0.25 inches and no error controls. The results for the final 10% of the stroke when the loads typically spike up as the material flashes out and the die cavities fill, are shown in Figure 70.

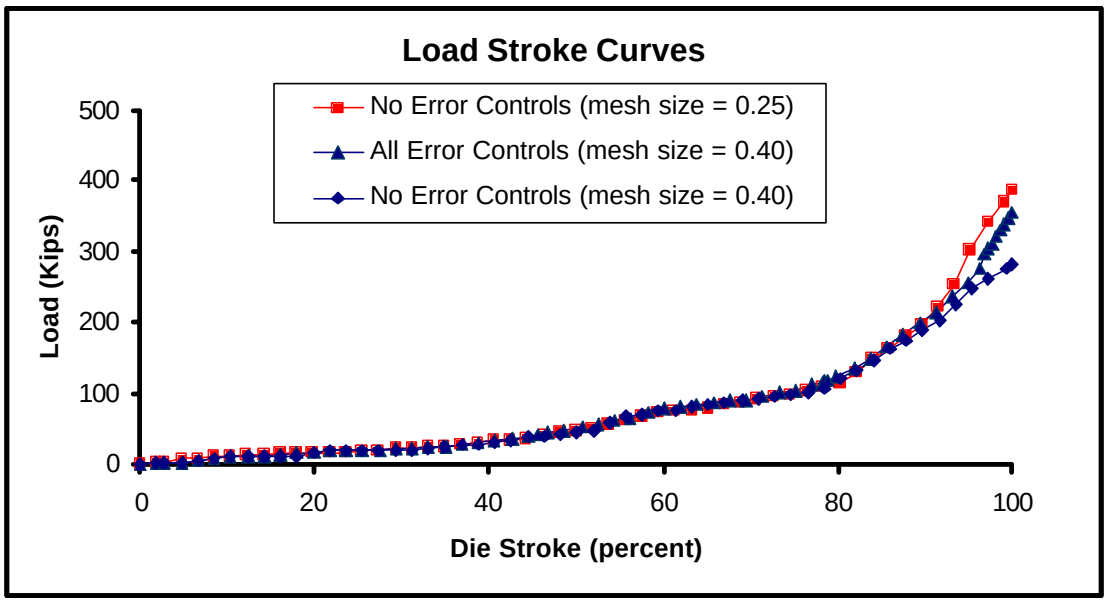


Figure 69: Load-stroke comparison with all error controls

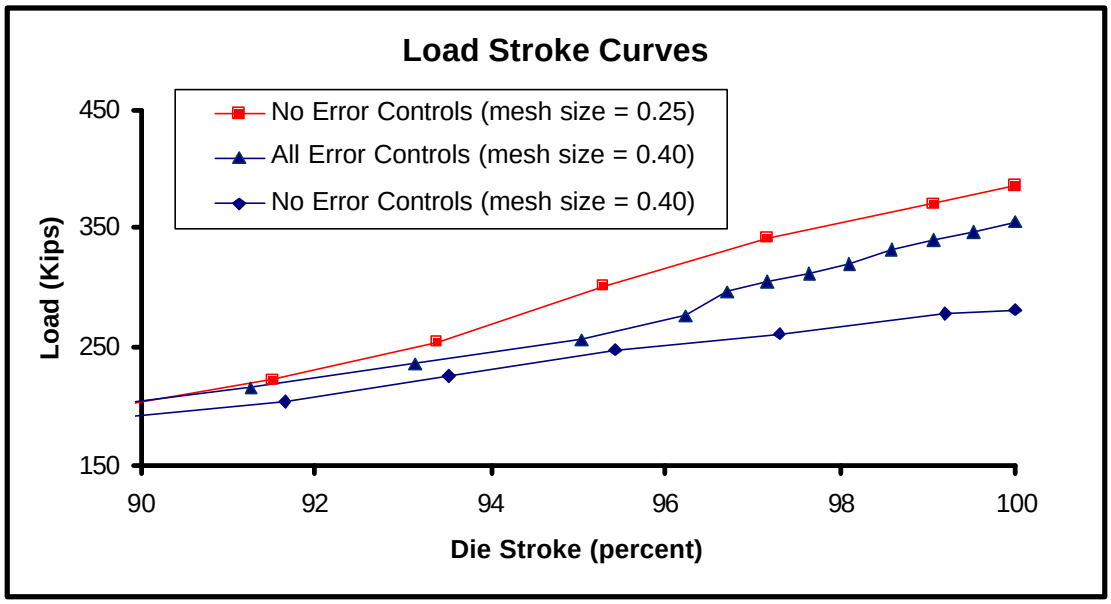


Figure 70: Load-stroke comparison with all error controls – final 10% die stroke

6.1.6 Element Quality

The geometry update procedure uses element quality as one of the criteria on which decision to perform a mesh manipulation is made. The lower bound on the quality of the triangular elements, as measured by the Q2 shape metric, generated on the workpiece boundary is typically more than the threshold value of 0.1. However, as the simulation progresses, this lower bound may slide below this threshold after remeshing. Potential causes include the nature of the die geometry (sharp transitions) and the evolution of die-workpiece contact and the mesh size transitions on the workpiece boundary. An example of the mesh quality before and after remeshing at 17% die stroke is shown in Figure 71 and Figure 72. All error controls were used when generating the new mesh for the deformed workpiece. The element quality measure varies from 0.124 to 0.997 before remeshing and from 0.236 to 0.997 after remeshing. Another example of the mesh quality before and after remeshing for 50% die stroke is shown in Figure 73 and Figure 74. The element quality measure varies from 0.092 to 0.996 before remeshing and from 0.171 to 0.998 after remeshing.

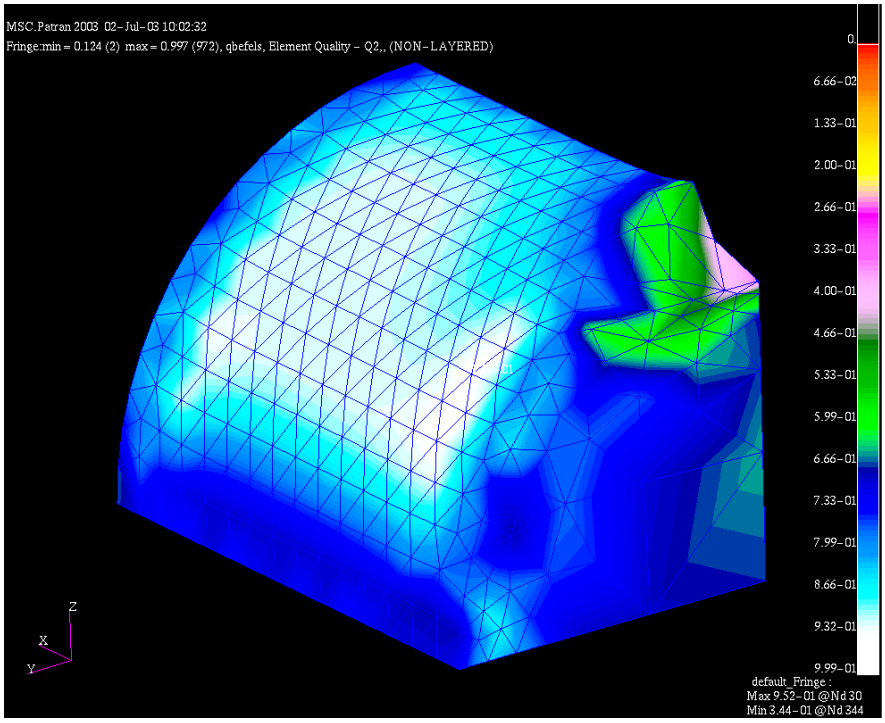


Figure 71: Element quality before remeshing (17% die stroke)

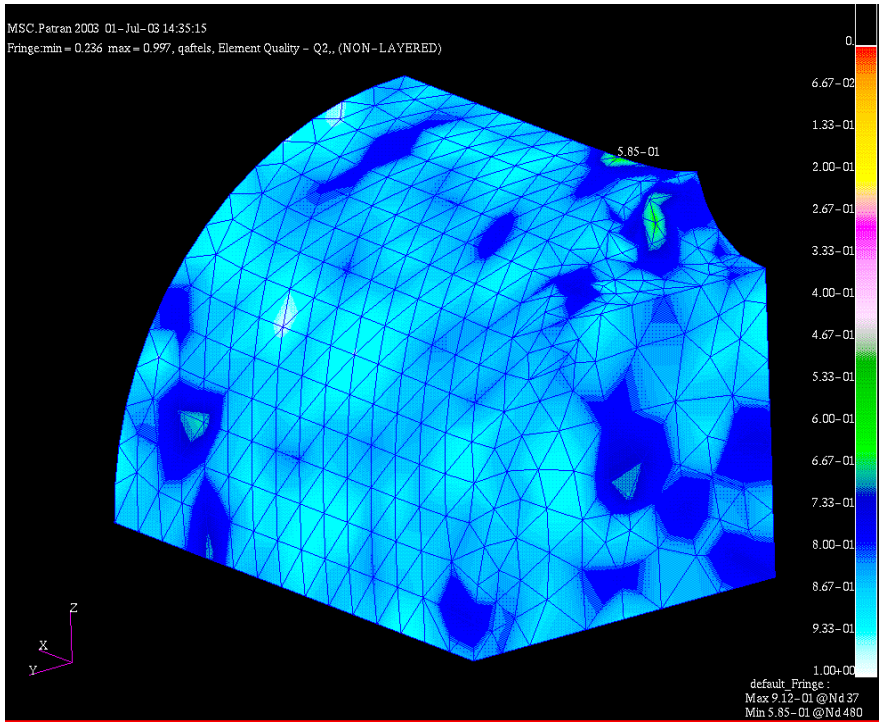


Figure 72: Element quality after remeshing (17% die stroke)

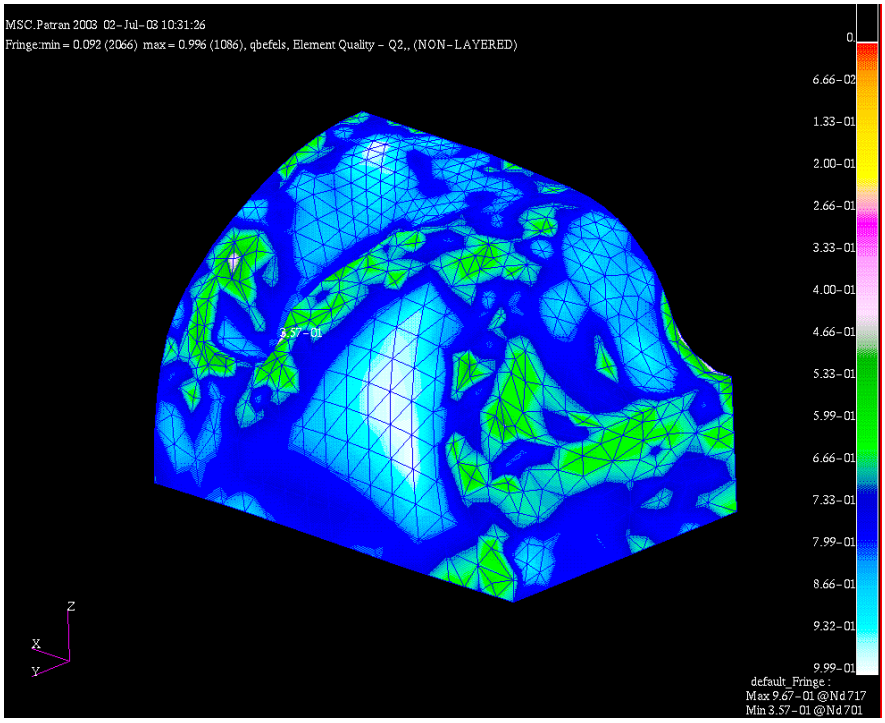


Figure 73: Element quality before remeshing (50% die stroke)

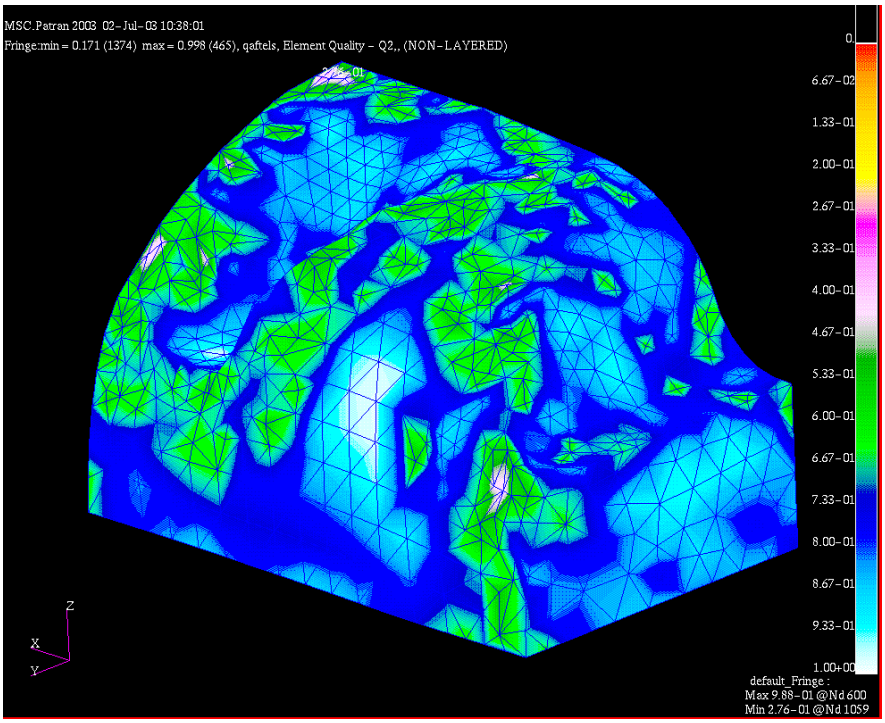


Figure 74: Element quality after remeshing (50% die stroke)

6.2 Simulation Results from the Automated System

Full simulation results from the geometry update system are presented in this section. The geometry update scheme has been validated on the simulation of a forging of a valve body and a disk forging process. The examples have been selected to demonstrate the ability of the automated system to handle large geometry changes typically encountered in industrial forming problems. Each of these examples was run with all error control procedures.

6.2.1 Valve Body Forging

The valve body forging process involves making two components in one hit on a hammer forging press. The presence of symmetry allows us to model one-eighth of the problem. The billet is a 4 inches round cylinder and the process takes it down to a thickness 0.14 inches at the center resulting in a large amount of flash. Figure 53 shows the initial die-workpiece configuration for the analysis. Note that a geometric model (from which the discretization shown in Figure 53 was created) has been used to represent the die during analysis and geometry update.

The problem needed 20 remeshes and took about 6.5 hours of CPU time on a HP C-180 workstation with 10 remeshes in the last 20% of die stroke when material begins to flash out. The remeshes in the later stages are triggered by the geometric overlap monitoring algorithm due to the sharp transition in the die geometry from the valve body to the flash gutter region as shown in Figure 53. The problem ran to completion in about 2.5 hours on the same workstation when the die was represented by its mesh model instead of the

CAD geometry and the workpiece was updated using the mesh-based method instead of the hybrid method. It is to be noted that earlier efforts to model the problem with manual remeshing took about 3-4 weeks [73, 74]. The increased CPU time when using the CAD model can be attributed to geometric checks performed in the contact algorithm via the CAD API which are typically very compute intensive (especially for 3-D operations) and hence, result in more than doubling of the CPU time required to solve this problem. Note that these geometric checks are performed for each node of the contact surface pair during an iteration (of a time step) and once during each of the remeshing steps.

The process was modeled in 60 time steps with an average of 3 iterations needed to obtain a converged solution per step. The workpiece has 757 nodes and 3784 4-node tetrahedral elements when the simulation commences with the number increasing to 3064 nodes and 16052 elements in the final stage when there is a large amount of flash. Results presented in Figure 75 through Figure 81 are for an average mesh size of 0.25 inches. Figure 75 shows the deformed workpiece mesh and the strain rate distribution at 17% die stroke and Figure 76 shows the remeshed workpiece with an accumulated volume loss of 0.09% compared to the original workpiece model. Figure 77 shows the deformed workpiece mesh and the strain rate distribution at 39% die stroke and Figure 78 shows the remeshed workpiece with an accumulated volume loss of 0.06% compared to the original workpiece model. Figure 79 shows the deformed workpiece mesh and the strain rate distribution at 76% die stroke and Figure 80 shows the remeshed workpiece with an accumulated volume loss of 0.46% compared to the original workpiece model. The full mesh model (mirrored) at the end of die stroke to give a realistic image of the parts being forged is shown in Figure 81. In the figures below, as the workpiece deforms,

it may seem as though the mesh gets more refined. However, this is an illusion due to the fact that the image sizes have been kept fixed in order to fit them within page limits even though the workpiece deformation has resulted in a significant increase in its footprint (bounding box size).

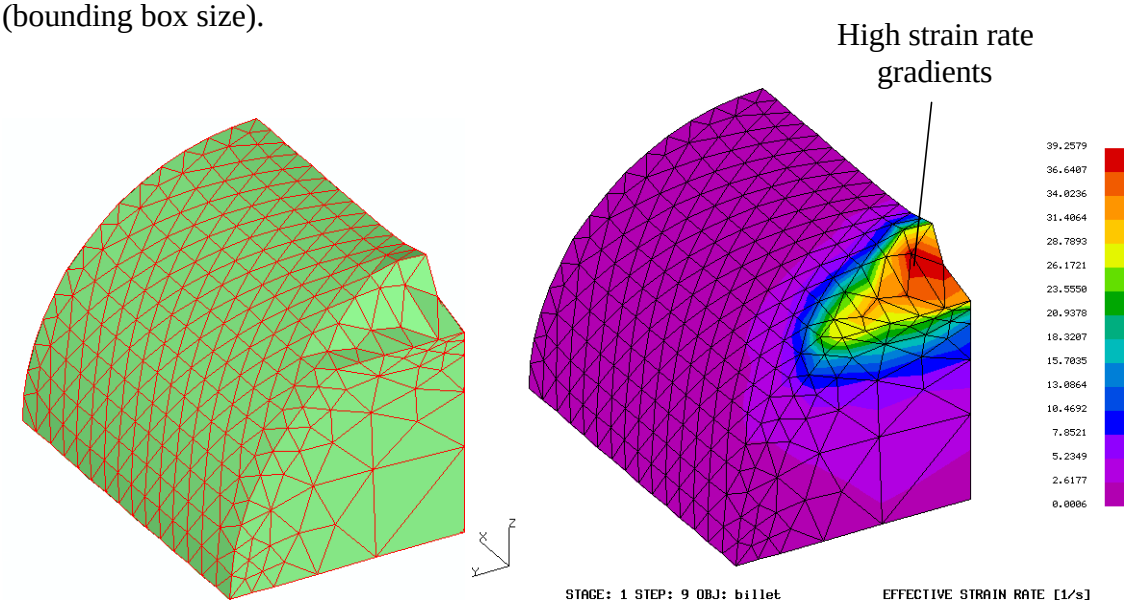


Figure 75: Deformed workpiece at 17% stroke

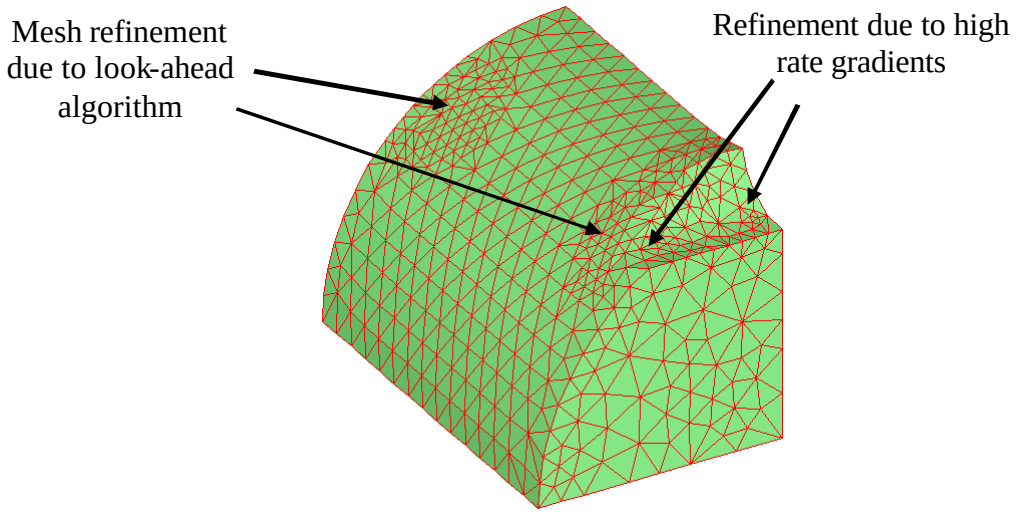


Figure 76: Remeshed workpiece at 17% stroke

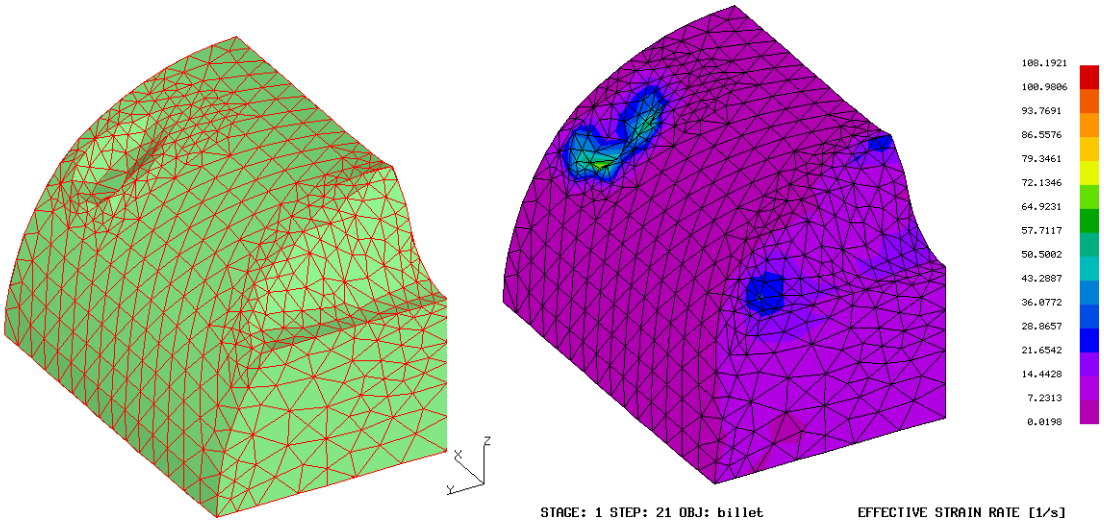


Figure 77: Deformed workpiece at 39% stroke

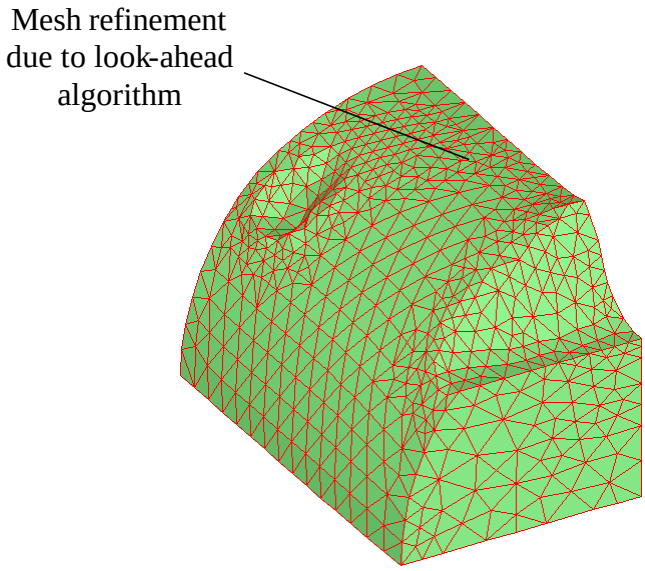


Figure 78: Remeshed workpiece at 39% stroke

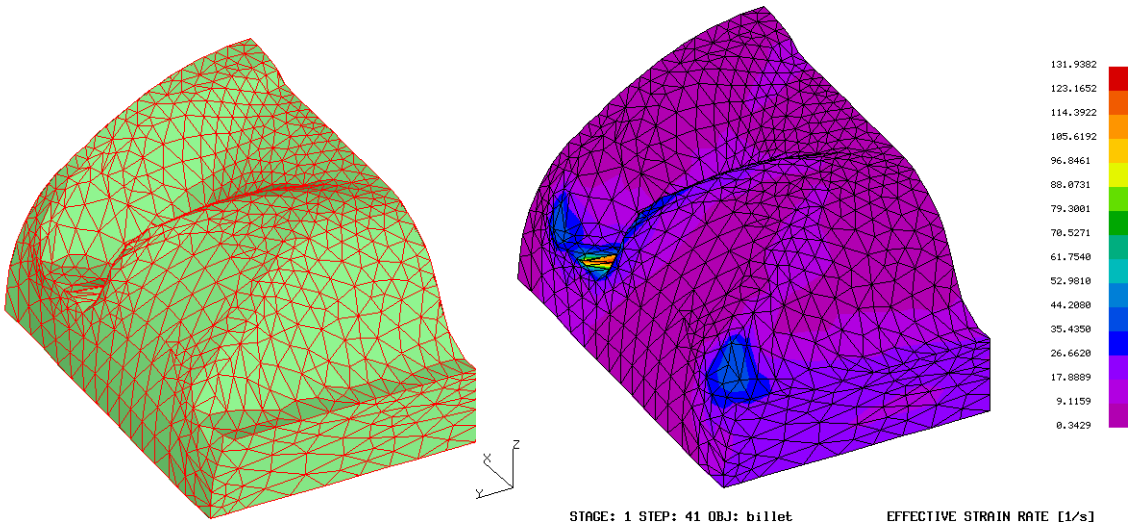


Figure 79: Deformed workpiece at 76% stroke

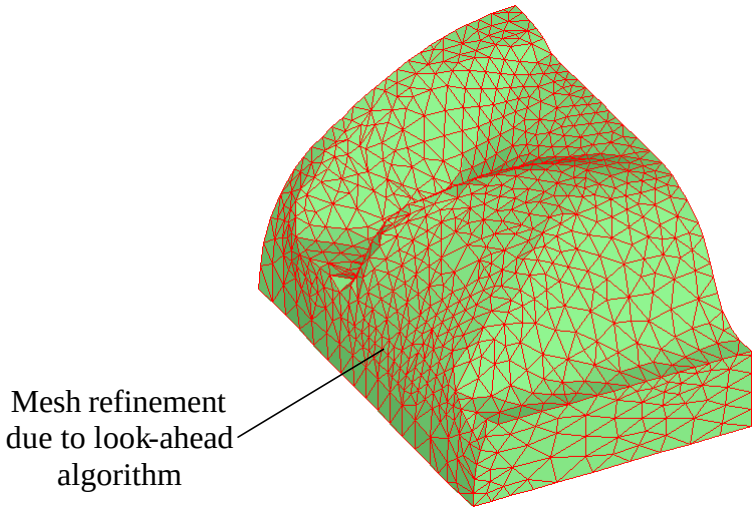


Figure 80: Remeshed workpiece at 76% stroke

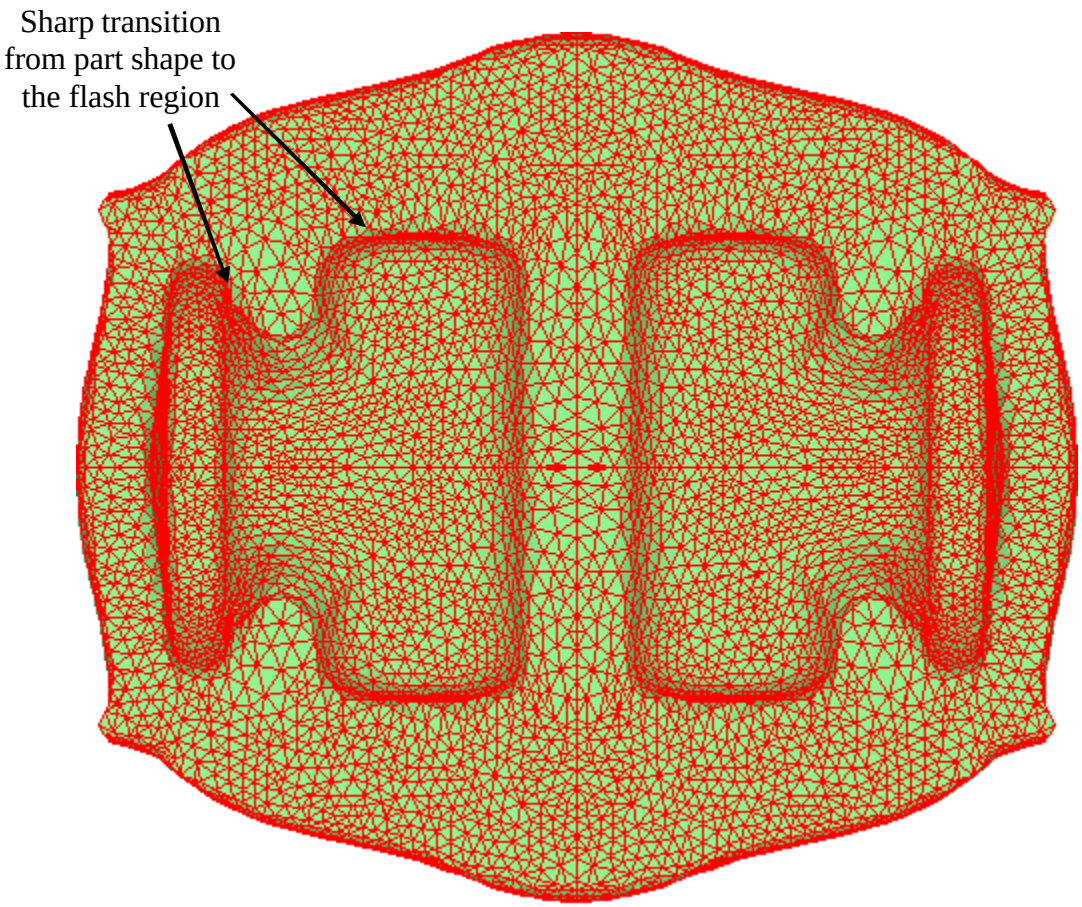


Figure 81: Mirrored workpiece mesh model at end of simulation

The percent change in the volume of the workpiece (enclosed by its mesh model) as the simulation progresses is shown in Figure 82. The volume of the workpiece remains almost constant during the analysis [8, 9, 74] and typically (though not necessarily) changes during geometry update due to enforcement of consistency requirements. As explained in section 3.4, the true volume change that can be attributed to the geometry update procedure is the change in volume due to approximation of the workpiece free surface, defined as V_f for *volume change due to free surface curvature*. As shown in Figure 82, the geometry update and remeshing procedure does well in terms of maintaining the workpiece volume with a total loss of 0.54% after 82% of die stroke.

The sharp transition from the part shape to the flash region has a big impact on the workpiece volume, which starts to change more rapidly once the material begins to flash at around 85% of the die stroke and as most of the workpiece comes in contact with the die. This drop is due to enforcement of relaxed consistency constraints (as the one in equation 25) in the contact regions. This represents *volume change due to die curvature* (V_c) as explained in section 3.4. The sharp transitions in the die geometry near the flash regions result in the die-overlap monitoring algorithm to trigger frequent remeshes as the material flows around these regions into the flash gutter. In this case, the workpiece was remeshed 10 times in the last 20% of the stroke. The ideal solution to avoid such a scenario is to design the die geometry with smooth transitions to the flash region. In fact, geometries with sharp transitions typically present significant difficulty for remeshing and one of the objectives in presenting this example was to demonstrate the ability of this algorithm to successfully model such cases. The total change in workpiece volume was 1.83% at the end of the simulation, well below the typically accepted norm of 4% [139] for forming analyses in industrial designs.

The impact of error control procedures on workpiece volume, for this example, was presented for two different mesh sizes in Figure 67 and Figure 68. As shown in these figures, use of error controls has a significant impact on controlling workpiece volume.

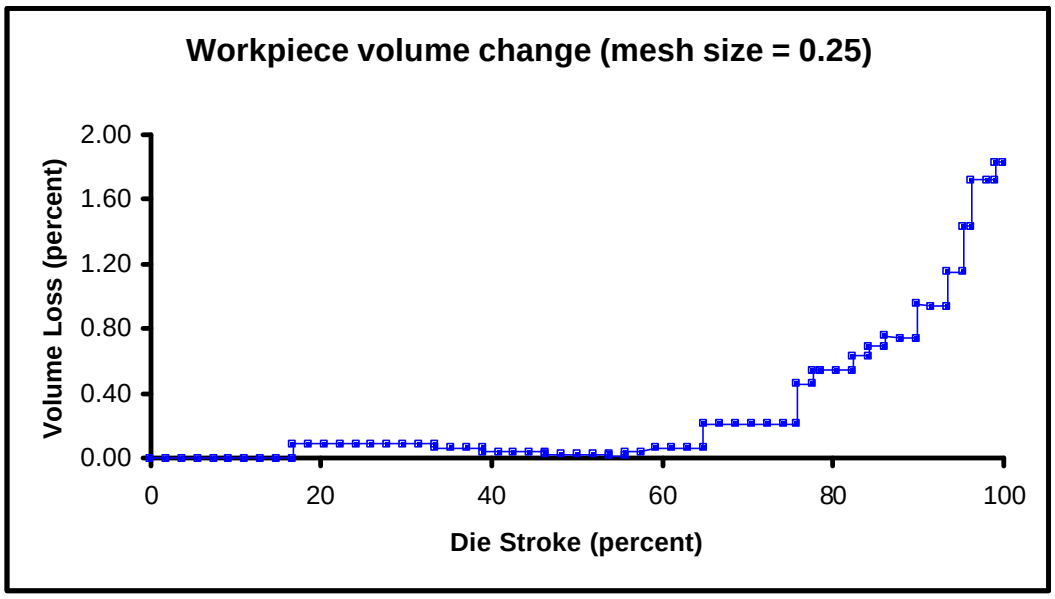


Figure 82: Percent change in workpiece volume in valve body forging

6.2.2 Disk Forging

The automated modeling system has also been used to model a disk-forging problem for which the initial die-workpiece mesh models are shown in Figure 83. This example is an axi-symmetric case that has been modeled in 3-D to demonstrate the ability of the geometry update procedure to handle fairly large changes in the geometry. Once again, one-eighth of the workpiece is modeled to take advantage of workpiece symmetry. The workpiece height at the center is reduced from 8 inches to 0.5 inches. The simulation was run with a mesh size requirement of 0.6 inches and completed in about one hour on the HP C-180 workstation with 8 remeshes for the workpiece. The initial mesh on the workpiece had a graded discretization with a mesh size of 0.4 inches in the regions of initial contact to a mesh size of 0.8 inches at the bottom (in Figure 83). Results presented in Figure 83 through Figure 90 are from the simulation performed using all error control

procedures. Figure 84 shows the deformed workpiece mesh and the strain rate distribution at 10% die stroke (first remesh) and Figure 85 shows the remeshed workpiece with an accumulated volume loss of 0.30% compared to the original workpiece model. Figure 86 shows the deformed workpiece mesh and the strain rate distribution at 36% die stroke and Figure 87 shows the remeshed workpiece with an accumulated volume loss of 0.52% compared to the original workpiece model. Figure 88 shows the deformed workpiece mesh and the strain rate distribution at 80% die stroke and Figure 89 shows the remeshed workpiece with an accumulated volume loss of only 0.61% compared to the original workpiece model. The mesh model, mirrored to give a realistic image of the part being forged, at the end of die stroke is shown in Figure 90.

The variation in workpiece volume during the simulation is shown in Figure 91. The change in workpiece volume at the end of the simulation in this case was only 1.42%. The impact of a die design with gradual transitions in the die geometry (as compared to the previous example) on preserving workpiece volume is evident here with the workpiece losing only 0.84% volume at 98% die stroke.

The impact of error control procedures on volume loss is shown in Figure 92. Using error controls consistently keeps volume loss less than the case with no error controls. The workpiece lost 1.58% at the end of die stroke when no error controls were used as compared to a loss of 1.42% when all error controls were used, a difference of 10.13%. However, note that at 98% die stroke, the volume loss is 0.84% with all error controls and 1.51% when no error controls are used, a difference of 44.37%.

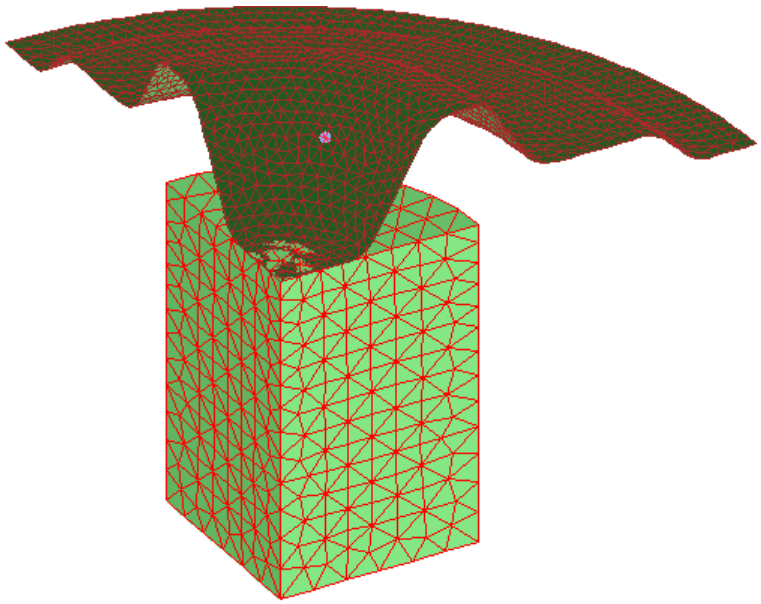


Figure 83: Initial die-workpiece configuration for the disk-forging problem

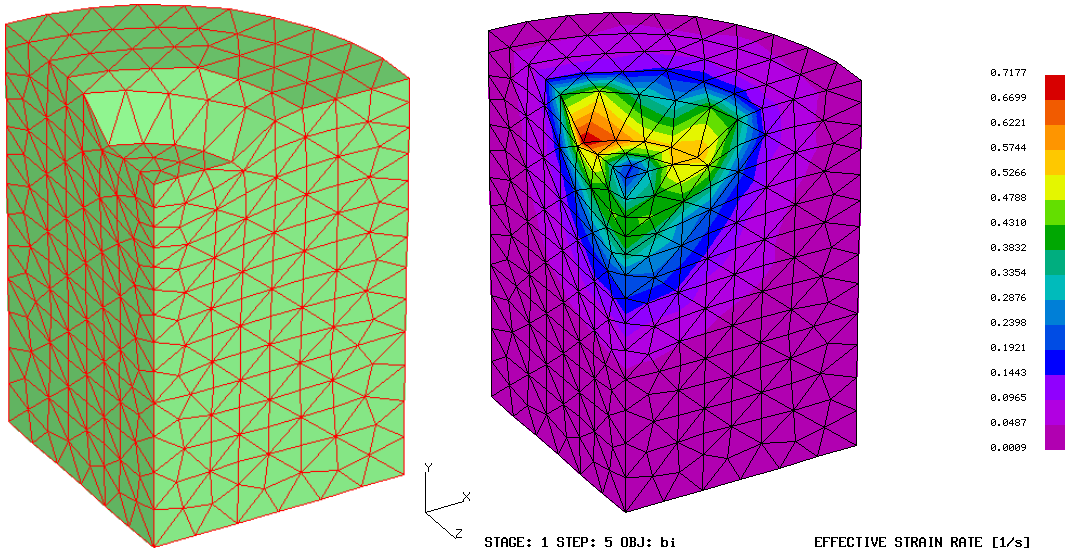


Figure 84: Deformed workpiece at 10% die stroke

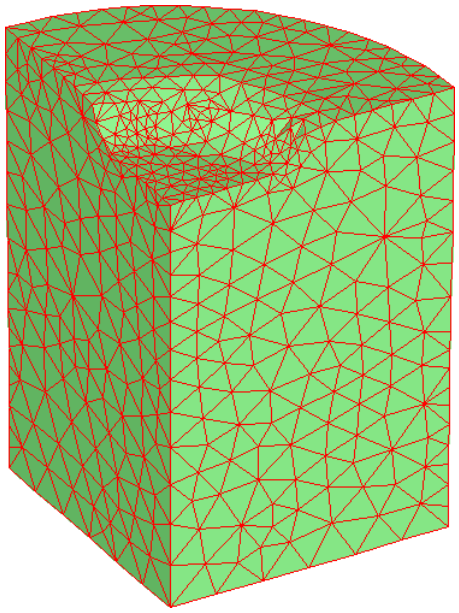


Figure 85: Remeshed workpiece at 10% die stroke

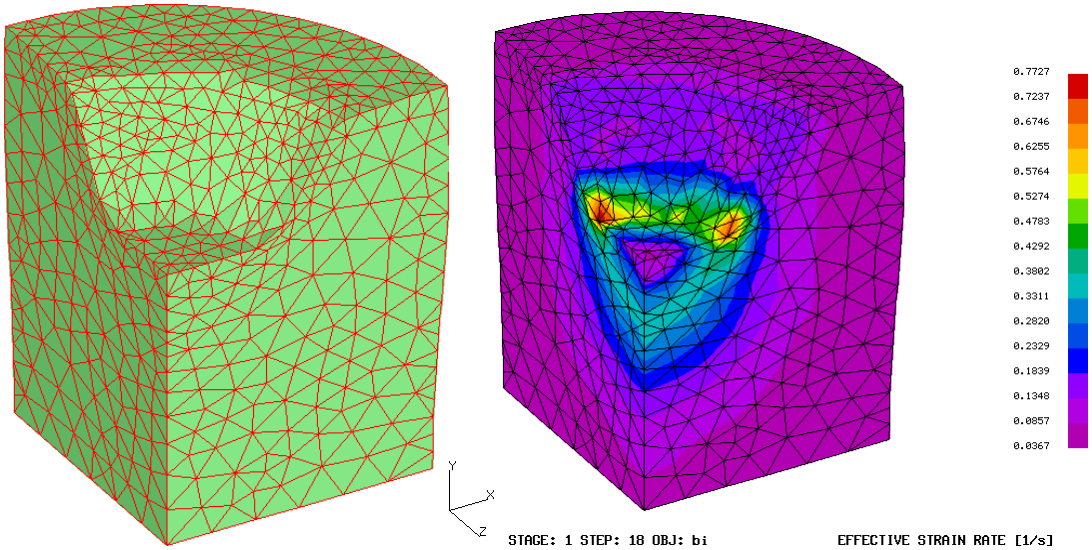


Figure 86: Deformed workpiece at 36% die stroke

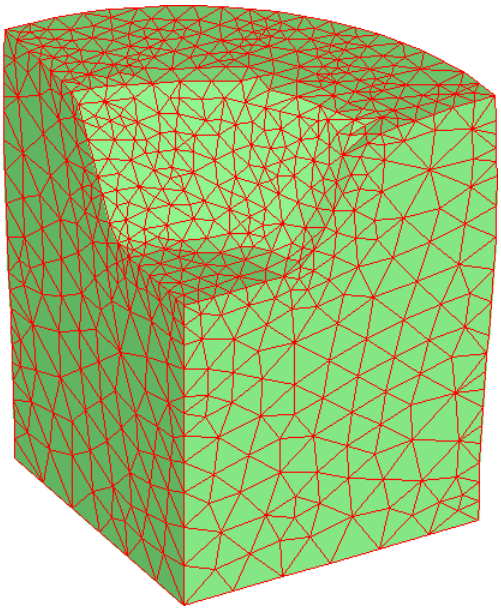


Figure 87: Remeshed workpiece at 36% die stroke

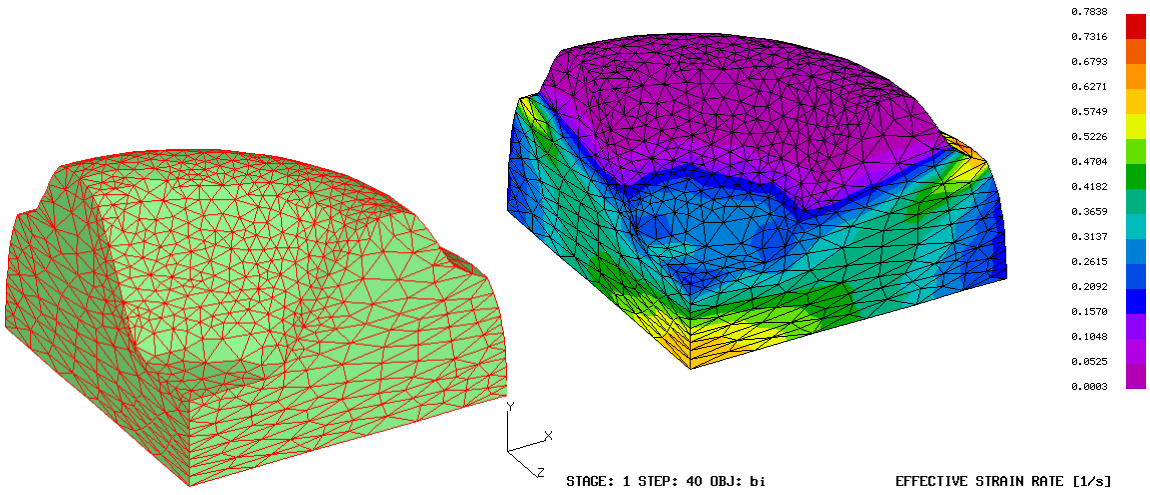


Figure 88: Deformed workpiece at 80% die stroke

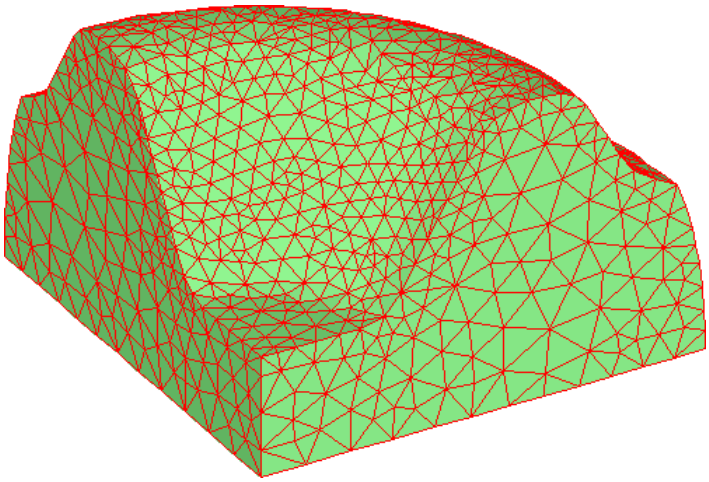


Figure 89: Remeshed workpiece at 80% die stroke

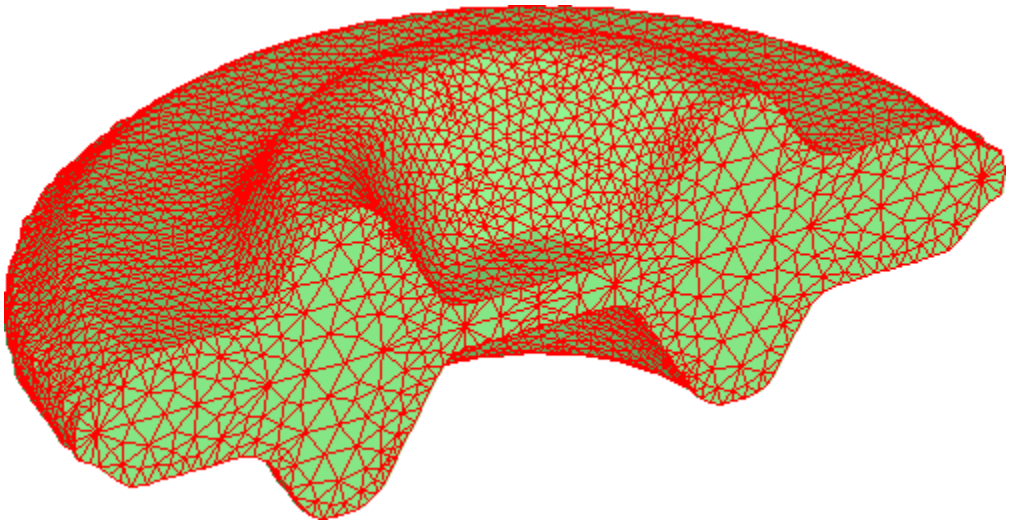


Figure 90: Mirrored workpiece mesh model at the end of simulation

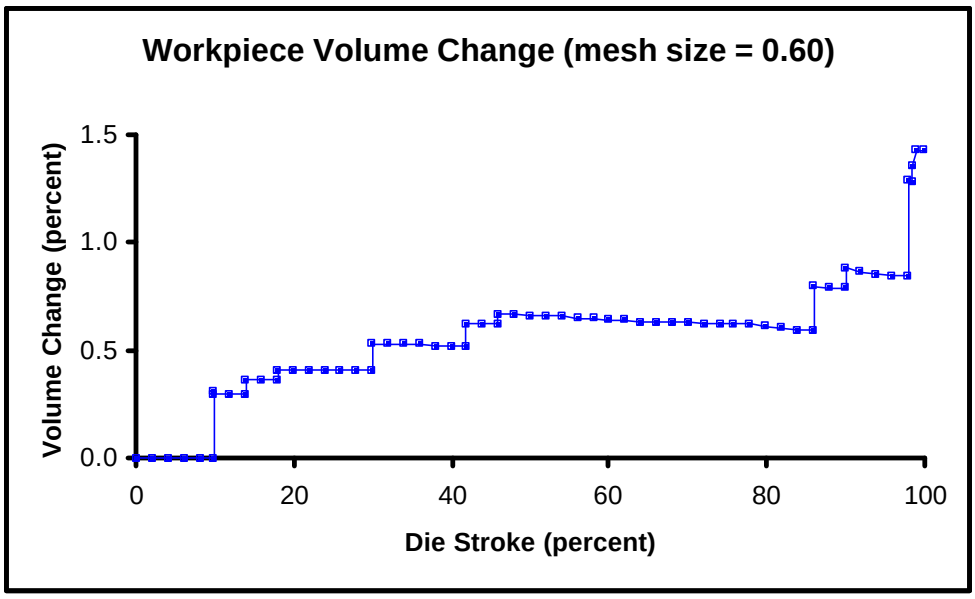


Figure 91: Percent change in workpiece volume in disk forging

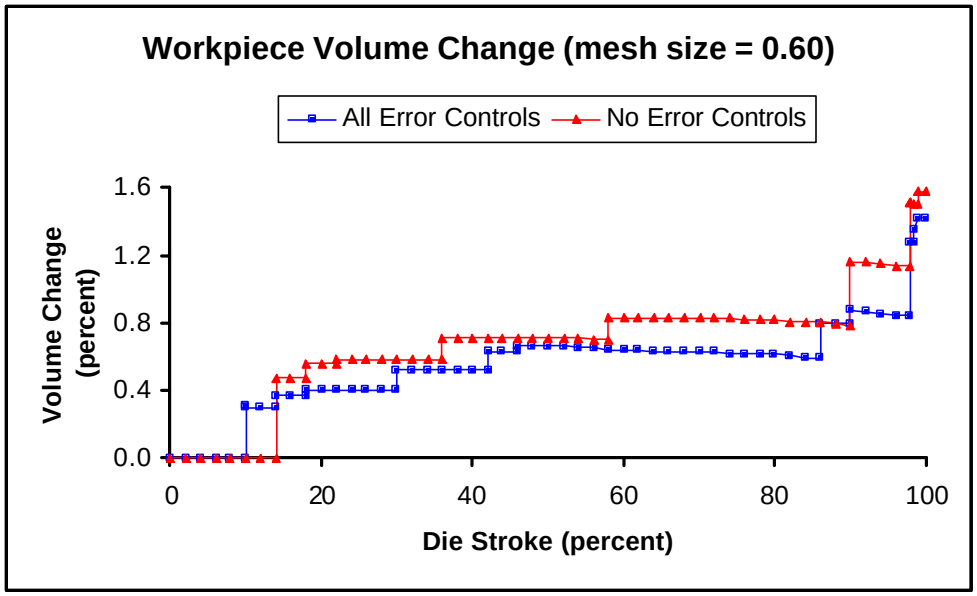


Figure 92: Effect of error controls on volume loss in disk forging

Chapter 7

TOOLS FOR AUTOMATED MODELING – MESH COARSENING

In some forming analysis cases, it may be acceptable not to perform any specific analysis on the dies and treat them as being *rigid*. This is especially true for cases when material flow is of primary concern and die deflections/stresses are assumed to be within limits. In such cases and in automated modeling systems that are based on the mesh-based method discussed in section 3.5.2, coarse mesh models could be used to represent the dies. The key notion here is that the mesh model for the die is only used to represent its geometry and since no analysis is performed on the dies, we can afford to have elements that are of poor quality with respect to being equilateral. The geometric checks that are performed to determine if a node of the workpiece is in contact with an element on the die boundary need to be performed for every iteration of every time step. As discussed in chapter 6, this can add up to a fairly significant amount of CPU time. Hence, the smaller number of triangles representing the die boundary can significantly reduce the time spent by the solver in geometric checks for contact between objects.

Automatic mesh generators are typically used to discretize the geometric model and create finite element meshes for analysis. They typically discretize the geometric models with an explicit objective of creating good quality finite elements. Hence, even if constraints are defined on the geometric model to force the mesh generator to create a

graded mesh, the discretization is going to result in more elements/facets than needed to represent the geometry of the object within acceptable error limits.

In order to obtain results with minimal influence on results accuracy, a mesh-coarsening module has been developed that eliminates unnecessary nodes/elements from the mesh models for the dies. The approach recommended here is that a mesh be generated with curvature control and a mesh size that equals the size of the smallest feature to be maintained. The setup time to construct such meshes by coarsening more standard meshes is small compared to the other components of the process. The coarsened mesh is then used to represent the rigid object during finite element analysis.

7.1 Mesh Coarsening

The fundamental goal of mesh coarsening is to reduce the number of triangles in a mesh while preserving the important geometric features in the model. The new mesh representation must preserve the original topology of the mesh and must form a "good" geometric approximation to the original mesh. These constraints on a good approximation to the original mesh are similar to the ones encountered in the workpiece geometry update procedure and hence the metrics from chapter 5 can also be used here. The process of reducing the elements to represent the die boundary is also called mesh decimation, coarsening, or compaction. Most of the related work [109-124] has dealt with this problem mostly for graphical visualization applications. Heckbert and Garland [109] present an excellent survey of the various techniques used in simplification of polygonal models. The decimation methods are classified into four categories [109]: (a)

vertex decimation methods delete a vertex and re-triangulate its neighborhood, (b) edge decimation methods delete one edge and two triangles and merge two faces, (c) triangle decimation methods delete one triangle and three edges, merge three vertices, and retriangulate the neighborhood, and, (d) patch decimation methods delete several adjacent triangles and retriangulate the neighborhood. Erikson [110] and Krus, et. al. [111] also review some of the techniques [112-124] in their papers.

The mesh coarsening operation used here involves using the edge collapsing procedure described in section 5.5.1. The collapsing operation is performed only after the local geometry variation check suggests changes within user specified limits. To represent the die geometry during analysis, the dies can be defined by surface meshes that define closed volumes or by a surface mesh that represents portion of the die with which the workpiece is likely to come in contact. For meshes that represent closed volumes, the local geometry variation check defined in chapter 5 is used to check if geometry changes are within limits. For surface meshes that do not represent a closed volume, an edge is collapsed if the dihedral angle between its adjacent faces is less than a user specified value for *plane_tolrnc*. A default value of 1^{o} is chosen to keep the geometry shape variation small. However, since the mesh coarsening operation is very quick and performed prior to running the simulation, it could easily be re-run with a different value if results with the default are not satisfactory.

An additional check is introduced to control the minimum angle of a triangle. This check has been included to avoid potential numerical round off problems in visualization and contact routines (that need face normal information for their calculations) when triangles

with very small angles (areas) are present. Before edge collapsing, the angles of a triangle are evaluated to identify its smallest angle in the collapsed local configuration. The user defines the minimum allowable angle for a triangle, *min_tri_angle*. If collapsing results in a situation where the minimum angle is less than the user-specified value for *min_tri_angle*, the element is not collapsed. This constraint is likely to result in a few more elements but will still provide significant time-savings as demonstrated later in this chapter.

Examples of some mesh models before and after coarsening are shown below. The numbers in parentheses for coarsened meshes are the compaction ratios. They indicate the percentage of initial number of nodes/elements that have been eliminated. Each of these examples was run with a *min_tri_angle* value of 2^{o}. Simulation results from using coarse mesh models are presented in the next section.

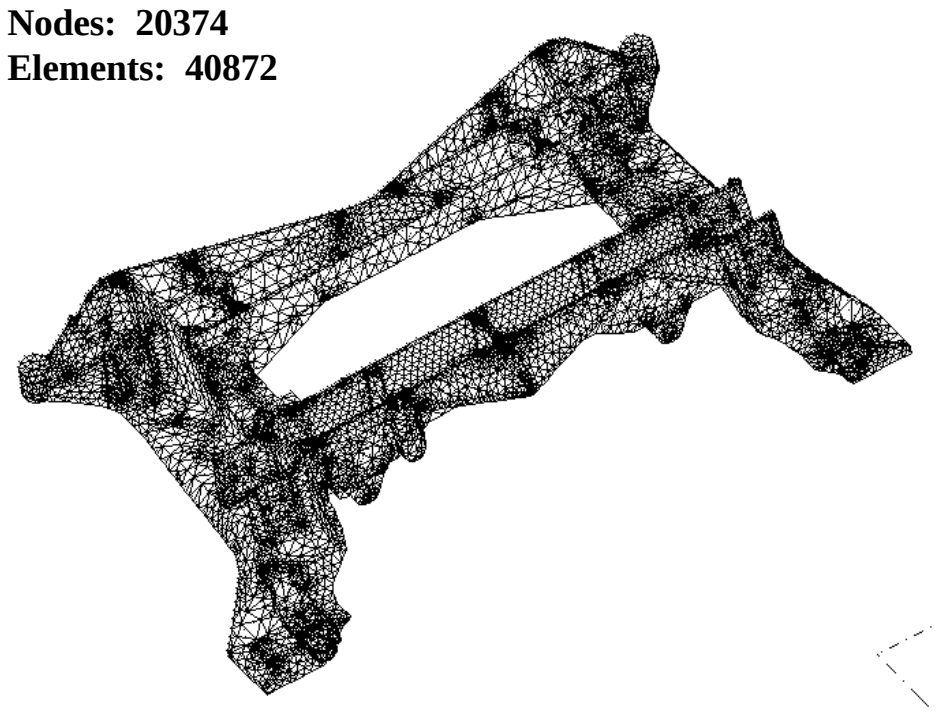


Figure 93: Fine mesh model on a casting geometry

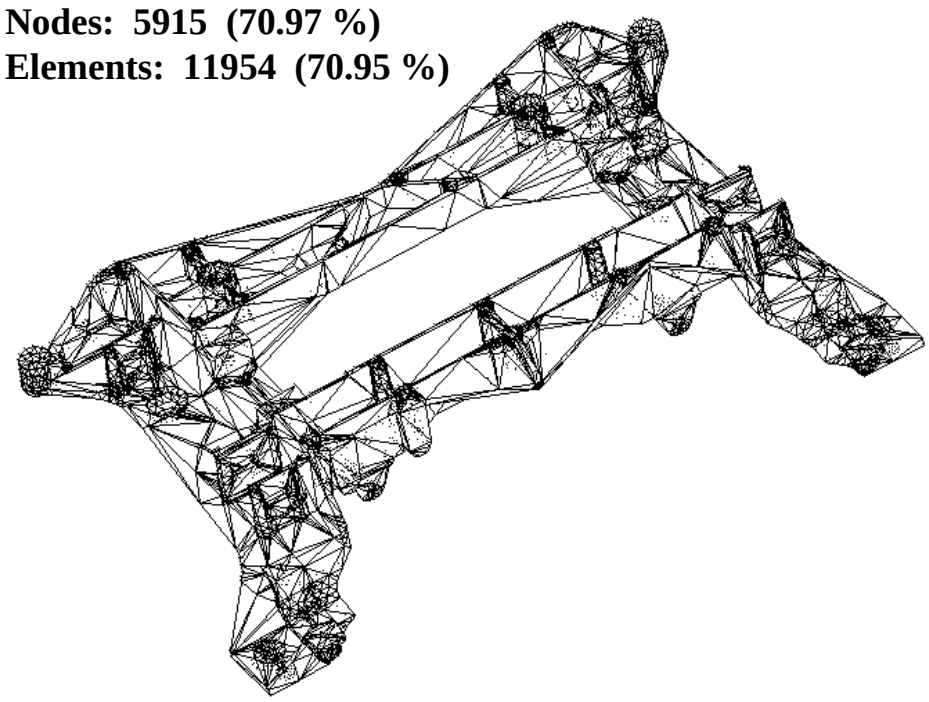


Figure 94: Coarse mesh model on the casting geometry

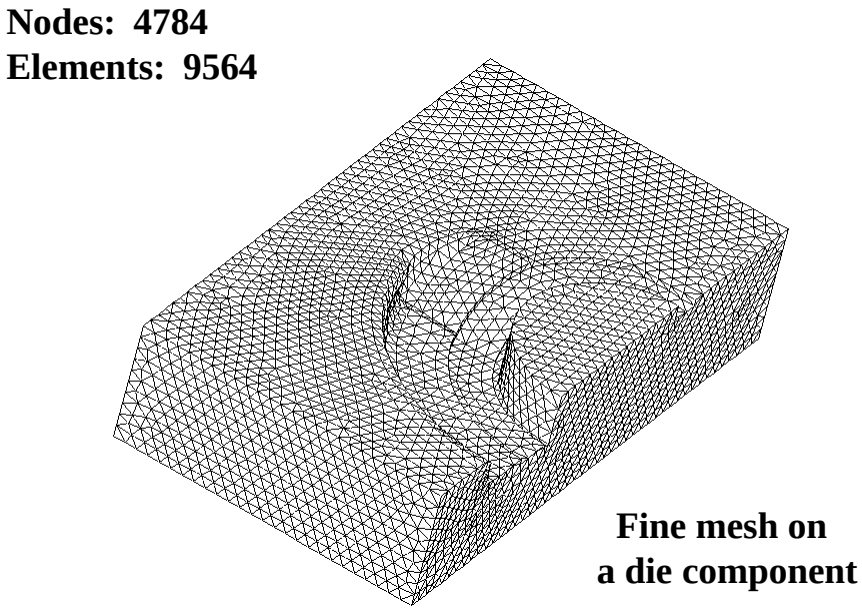


Figure 95: Fine mesh on the bottom die for a die casting application

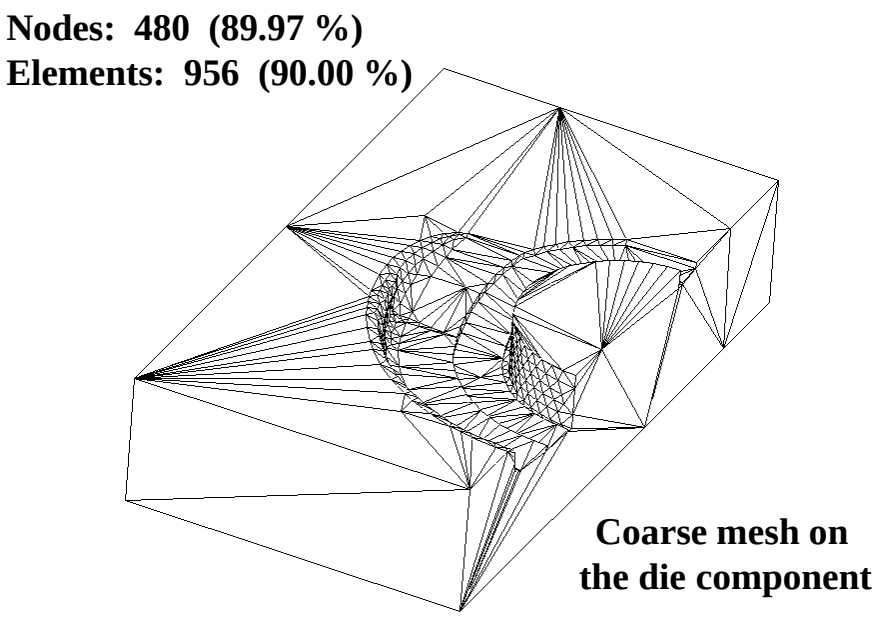


Figure 96: Coarse mesh on the bottom die for the casting application

7.2 Process Simulation Using Coarsened Die Models

The mesh-coarsening module was tested in the finite element simulation of a forging process. The simulations were run with the Antares^{TM} [8, 9] software for 3-D process modeling of forming operations. Two analyses were run to determine the impact of mesh coarsening on analysis time/results. The valve body forging described in chapter 6 was again used for this evaluation. The first simulation used a mesh model without coarsening and the second simulation used a coarsened mesh model for the die. A *min_tri_angle* value of 2^{o} was used to obtain the coarsened models. The initial setups for the two cases are shown in Figure 97 and Figure 98.

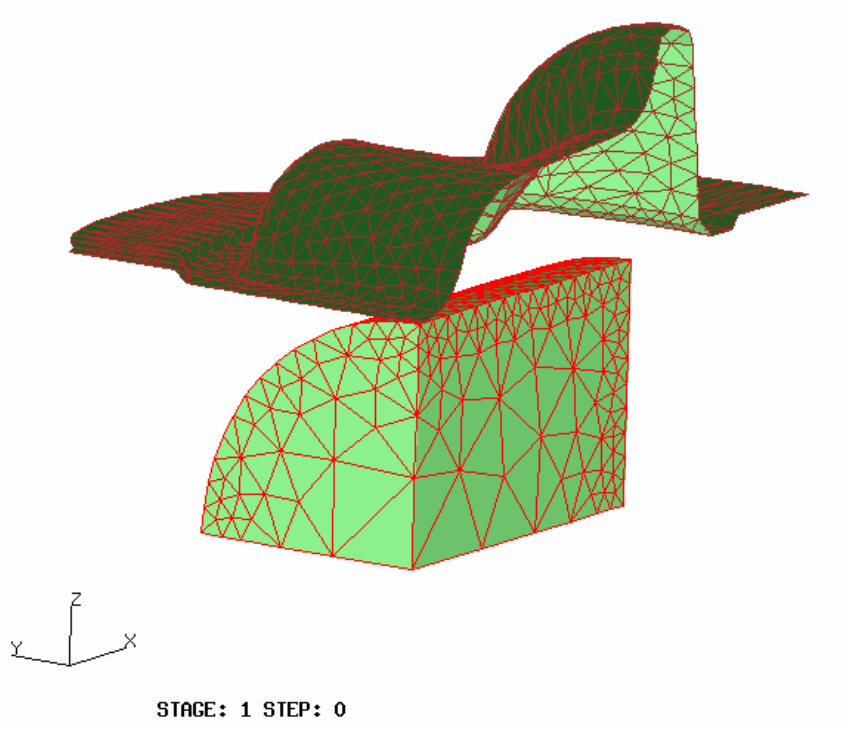


Figure 97: Initial setup for valve body forging with original die mesh

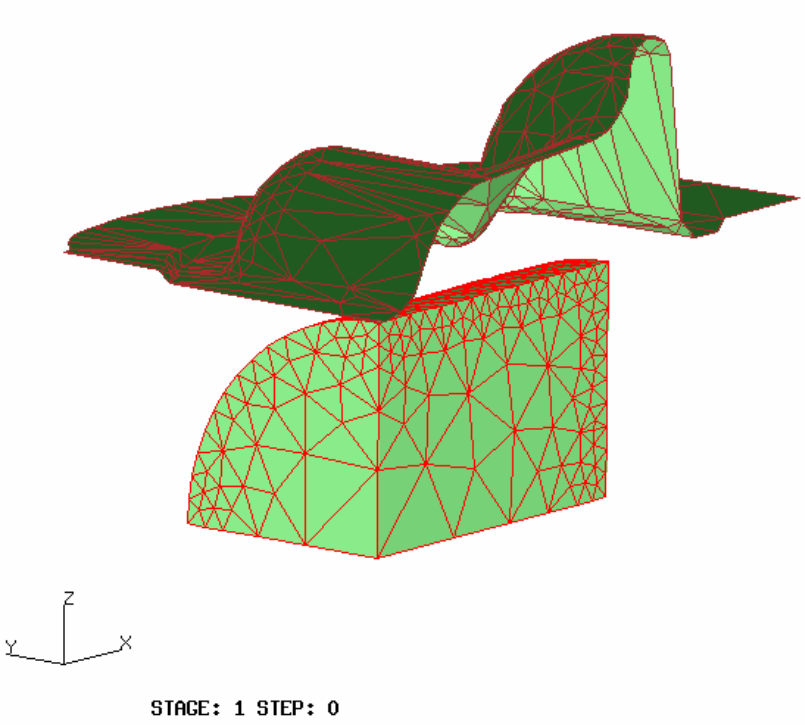


Figure 98: Initial setup for valve body forging with coarsened die mesh

Mesh coarsening reduced total simulation time by 20.55%. The results from mesh coarsening and the process simulation are presented below in Table 1.

	Elements in Die Mesh	**Nodes in Die Mesh**	**CPU Time (seconds)**	**Reduction in CPU Time**
Without Mesh Coarsening	1389	740	8084.3	-
With Mesh Coarsening	615 (**-55.72%**)	337 (**-54.46%**)	6422.6	**-20.55 %**

Table 1: CPU times for forging analysis run with and without mesh coarsening

As can be seen from the load stroke curves plots for the two cases shown in Figure 99, *the results of the simulations performed with and without mesh coarsening are almost identical*. Results of other parameters such as strains in the workpiece, etc. (from the coarse mesh simulation) are also observed to closely match those obtained from the simulation performed with the fine mesh representing the die. In general, there is likely to be a small variation in the results due to the approximation of the die geometry (especially when the die has regions of substantial curvature) with fewer elements. It is expected that cases where the original die model contains a more uniform mesh (as compared to the large triangles representing planar regions in this case), the reduction in CPU time will be even more dramatic. Note that the time required to generate the coarse mesh for this problem was only about half a minute. The more "curved" the die geometry, the more facets we need for an acceptable representation of the die geometry. Also, a majority of the CPU time is spent in solving for the primary field variables and

hence there is a limit to the speed up achieved from using a compacted mesh model for the dies.

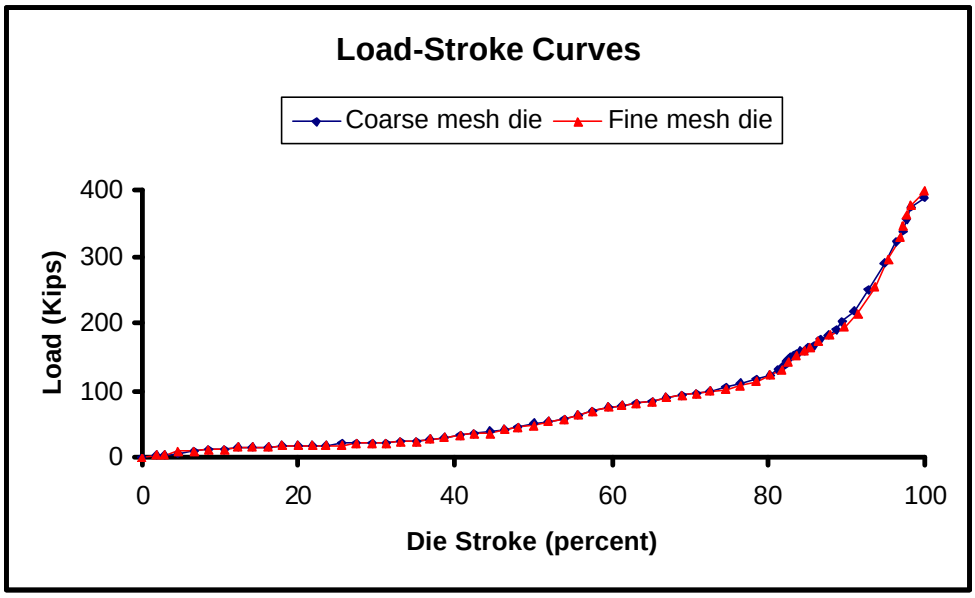


Figure 99: Load-stroke curves with original and coarse mesh die

Chapter 8

CONCLUSIONS AND FUTURE WORK

This dissertation has presented the key issues that must be considered in the development of an automated system for modeling metal forming operations.

8.1 Summary of Contributions

The major contribution from this thesis research is the development of the hybrid method for updating the geometry of the deformed workpiece in 3-D forming simulations. New techniques developed to accomplish this include procedures to:

1. update the workpiece geometry using mesh manipulation operations,
2. control/monitor volume losses using the local geometry variation check,
3. use the true geometric representation of the die during analysis and remeshing,
4. compensate new nodes on the workpiece surface based on local curvature,
5. adaptively refine the mesh based on die-overlap, a strain-rate based error indicator and proximity to the region of deformation (look-ahead algorithm).

A mesh coarsening procedure has also been developed based using the mesh manipulation approach to reduce overall simulation time for cases where no analysis is to be performed on the dies.

8.2 Future Work

This section introduces areas for potential future research efforts in automated modeling of forming processes. First, research areas specific to the hybrid method of this thesis are given. This is followed by a discussion of research topics that would affect overall simulation accuracy/speed.

8.2.1 Topics Related to the Hybrid Method

The hybrid method developed in this thesis is applicable only when the workpiece boundary is defined by straight-edged triangles. New procedures have to be developed to handle cases where other types of elements represent the workpiece boundary, for example 6-node triangles or 4-node quadrilaterals. For 6-node triangles, one possibility is to ignore the mid-side nodes for the triangles for the purpose of geometry update and remeshing. The workpiece boundary would now be made up of straight-edged triangles, and hence, the procedure developed here can be used to remesh the deformed workpiece. A post-processing operation can then generate mid-side nodes and adjust their location based on local curvature using the curvature compensation algorithm presented here.

In this thesis research, a completely new mesh is generated to represent the deformed workpiece. However, only local changes to the workpiece mesh guided by the remeshing criteria may result in overall CPU time-savings as well as potential improvement in accuracy due to reduced mesh rezoning errors. This may also need changes in the mesh-

rezoning algorithm of the solver that specifically identifies only new nodes/elements and interpolates field variables for them.

In the implemented method, the error indicator is used to identify regions for mesh refinement on workpiece boundary. Full consideration should be given to discretization errors throughout the domain since localizations of importance can form in the interior of the workpiece.

Geometric checks to determine contact has resulted in more than doubling of the CPU time required for the simulation. This is despite the fact that the geometric checks are applied only to a subset of the boundary nodes obtained by using the mesh based contact algorithm as a filter (to identify nodes that are likely to be in contact with the dies). Since complex industrial forming processes will have complex die shapes and the workpiece represented by several thousand nodes, a faster algorithm for geometric checks can potentially have a big impact in the total time required to run the simulations. One possible approach that may save compute time is to store the information of the face (geometry) with which a node has come in contact. As the simulation progresses, the contact procedure can first check for contact with this face and then with adjacent faces rather than the current approach of cycling through all faces with a bounding box filter.

In the current implementation, load calculations reported by the solver reflect the contact with die mesh and not the die geometry. It was not possible to implement this change since access to the appropriate portion of the finite element solver's source code.

However, to reflect the updated contact calculations, this change must also be made in the solver's load calculation procedure.

8.2.2 Related Topics of Interest

More general curved representations of the die and workpiece based on a complete model topology could be considered. This is an ideal approach for implementation within an automated environment since:

(i) the die-workpiece geometric model with the attribute data is a complete and unambiguous representation of the problem domain from which a discretization is generated by an automatic mesh generator,

(ii) the evolved geometry inherits the true geometry of the die in contact regions resulting in an accurate geometric description of the contact areas before discretization, and,

(iii) the workpiece free surface in the geometric representation could better represent the smooth free surface of the workpiece.

Since mesh rezoning requires interpolation of field variables from one mesh to another, it is important to carefully control the accuracy of this process. Improved procedures for general mesh-to-mesh rezoning and incremental mesh modification rezoning should be considered.

8.3 Closing Remarks

This research effort has led to the development of an automated modeling system for metal forming. The automated system has been quite successful in modeling industrial forming processes involving complex 3-D geometries, eliminated the need for manual remeshing and reduced the overall simulation time by more than 90%.

Chapter 9

BIBLIOGRAPHY

1. Kobayashi, S., Oh, S. I., Altan, T., "Metal Forming and the Finite-Element Method", Oxford Series on Advanced Manufacturing, Oxford University Press, 1989.

2. Lee, C. H., Kobayashi, S., "New Solutions to Rigid-Plastic Deformation Problems Using a Matrix Method", Trans. ASME, *J. Engng. for Ind.*, Vol. 95, 1973.

3. Oh, S. I., "Finite Element Analysis of Metal Forming Processes with Arbitrarily Shaped Dies", *Int. J. Mech. Sci*., Vol. 24, pp. 479-493, 1982.

4. Park, J. J. and Kobayashi, S., "Three Dimensional Finite Element Analysis of Block Forging", ", *Int. J. Mech*. *Sci*., Vol. 26, pp. 165-176, 1984.

5. Surdon, G. and Chenot, J. L., "Finite Element Calculation of Three Dimensional Hot Forging", *Int. J. Numer. Meth. Engng.*, vol. 24, pp. 2107-2117, 1987.

6. Owen, D. R. J., Peric, DeSouza Neto, E. A., Yu, J., Dutko, M. and Crook, A. J. L., "Advanced Computational Strategies for 3-D Large Scale Metal Forming Simulations", In *Simulation of Materials Processing: Theory, Practice, Methods and Applications*, Shan-Fu Shen and P. R. Dawson (eds.), Balkema, Rotterdam, pp. 7-22, 1995.

7. Ghosh, S., "Arbitrary Lagrangian-Eulerian Finite Element Analysis of Large Deformation in Contacting Bodies", *Int. J. Numer. Meth. Engng.*, vol. 33, pp. 1891-1925, 1992.

8. Antares™ Primer Manual, UES Software Services, Inc., 4401 Dayton-Xenia Road, Dayton, OH 45432.

9. Antares™ Theory and Reference Manual, UES Software Services, Inc., 4401 Dayton-Xenia Road, Dayton, OH 45432.

10. Deform™ Users Manual, Scientific Forming Technologies, Columbus, OH.

11. LS-Dyna™ Users Manual, Livermore Software Technology Corp., Livermore, CA.

12. Abaqus™ Users Manual, Hibbitt, Karlson & Sorensen, Inc., Providence, RI.

13. Unigraphics™ User Manual, Electronics Data Systems Corp., Unigraphics Division, Version 13.0, 1997.

14. UG/Open API Reference, Electronics Data Systems Corp., Unigraphics Division, Version 13.0, Vol. 1-6, 1997.

15. Baehmann, P. L., Shephard, M. S., Ashley, R. A., Jay, A., "Automated Metal Forming Modeling Utilizing Adaptive Remeshing and Evolving Geometry", *Comp. Struct.*, vol. 30 (1/2), pp. 319-325, 1988.

16. Baehmann, P. L., Collar, R. R., Hattangady, N. V., and Shephard, M. S., "Geometry and Mesh Control for Automated Bulk Forming Simulations", *Proc. ASME Winter Annual Meeting*, Anaheim, CA, Nov. 8-13, 1992, pp. 47-57.

17. Cheng, J. H., “Automatic Adaptive Remeshing for Finite Element Simulation of Forming Processes”, *Int. J. Numer. Meth. Engng.*, vol. 26, pp. 1-18, 1989.

18. Cheng, J. H. and Kikuchi, N., “A Mesh Rezoning Technique for Finite Element Simulations of Metal Forming Processes”, *Int. J. Numer. Meth. Engng.*, vol. 23, pp. 219-228, 1986.

19. Yukawa, N., Kikuchi, N., and Tezuka, A., “An Adaptive Remeshing Method for Analysis of Metal Forming Processes”, *Adv. Tech. Plasticity*, 4:1719-1728, 1990.

20. Tezuka, A., “Adaptive Remeshing Process with Quadrangular Finite Elements”, *Adv. Engng. Software*, vol. 15, 185-201, 1992.

21. Park, J. S. and Hwang, S. W., “Automatic Remeshing in Finite Element Simulation of Metal Forming Processes by the Guide Grid Method”, *J. Mater. Proc. Tech.*, Vol. 27, 73-89, 1991.

22. Schneiders, R., Oberschelp, W., Kopp, R, and Becker, M., “New and Effective Remeshing Scheme for the Simulation of Metal Forming Processes”, *Engng. Comp.*, 8:163-176, 1992.

23. Lee, N. K., Yoon, J. H., and Yang, D. Y., “Finite Element Deformation of Large Deformation by Automatic Renoding as a Weak Remeshing Technique”, *Int. J. Mech. Sci.*, 34(4):255-273, 1992.

24. Zhu, Y. Y., Zacharia, T., and Cescotto, S., "Application of Fully Automatic Remeshing to Complex Metal Forming Analyses", *Comp. Struct*., Vol. 62, No. 3, pp. 417-427, 1997.

25. Habraken, A. M. and Cescotto, S., "An Automatic Remeshing Technique for Finite Element Simulation of Forming Processes", *Int. J. Numer. Meth. Engng.*, vol. 30, pp. 1503-1525, 1990.

26. Dyduch, M., Habraken, A. M., and Cescotto, S., "Automatic Adaptive Remeshing for Numerical Simulations of Metal Forming", *Comp. Meth. App. Mech. Engng*., vol. 101, pp. 282-298, 1992.

27. Dyduch, M, Cescotto, S and Habraken, A. M., "Efficient Error Estimates for Adaptive Remeshing in 2-D Metal Forming Modeling", In *Simulation of Materials Processing: Theory, Practice, Methods and Applications*, Shan-Fu Shen and P. R. Dawson (eds.), Balkema, Rotterdam, pp. 419-424, 1995.

28. Zienkiewicz, O. C., Huang, G. C., and Liu, Y. C., "Adaptive FEM Computation of Forming Processes - Application to Porous and Non-porous Materials", *Int. J. Numer. Meth. Engng.*, vol. 30, pp. 1527-1553, 1990.

29. Zienkiewicz, O. C., Liu, Y. C. and Huang, G. C., "Error Estimation and Adaptivity in Flow Formulation for Forming Problems", *Int. J. Numer. Meth. Engng.*, vol. 25, pp. 23, 1990.

30. Yang, H. T. Y., Heinstein, M., and Shih, J. M., “Adaptive 2-D Finite Element Simulation of Metal Forming Processes”, *Int. J. Numer. Meth. Engng.*, Vol. 28, pp. 1409-1428, 1989.

31. Dick, R. E. and Harris, W. E., “Fully Automatic Rezoning of Evolving Problems”, In *Numiform ‘92*, Chenot, Wood, and Zienkiewicz (eds.), Balkema, Rotterdam, 1992.

32. Joun, M. S. and Lee, M. C., “Quadrilateral Finite-Element Generation and Mesh Quality Control for Metal Forming Simulation”, *Int. J. Numer. Meth. Engng.*, Vol. 40, pp. 4059-4075, 1997.

33. Petersen, S. B. and Martins, P. A. F., “Finite Element Remeshing: A Metal Forming Approach for Quadrilateral Mesh Generation and Refinement”, *Int. J. Numer. Meth. Engng.*, Vol. 40, pp. 1449-1464, 1997.

34. Wu, W. T., Oh, S. I., Altan, T and Miller, R. A., “Automated Mesh Generation for Forming Simulation”, Proc. Comput. Engng., ASME, New York, Vol. 1, pp. 506-513, 1990.

35. Barata Marques, M. J. M and Martins, P. A. F., “An Algorithm for Remeshing in Metal Forming”, *J. Mater. Proc. Tech.*, Vol. 24, pp. 157-167, 1990.

36. Lee, N. S., Bathe, K. J., “Error Indicators and Adaptive Remeshing in Large Deformation Finite Element Analysis", *Fin. Elem. Anal. Des*., 16, 99-139, 1994.

37. Habraken, A. M., Radu, J. P., “Simulation of Forging Applications with the Finite Element Method”, *Numiform ‘89*, Thompson, et. al. (eds.), Balkema, Rotterdam, pp. 543-548, 1989.

38. Zienkiewicz, O. C., Huang, G. C., “Adaptive Modeling of Transient Coupled Metal Forming Processes”, *Numiform ‘89*, Thompson, et. al. (eds.), Balkema, Rotterdam, pp. 3-10, 1989.

39. Coupez, T., Soyris, N., and Chenot, J. L., “3-D Finite Element Modeling of the Forging Process with Automatic Remeshing”, *J. Mater. Proc. Tech.*, 27:119-133, 1991.

40. Coupez, T., and Chenot, J. L., “Large Deformations and Automatic Remeshing”, In Computational Plasticity (COMPLAS III), E. Hinton, D. R. J. Owen, E. Onate (eds.), Pineridge Press, pp. 1077-1087, 1992.

41. Coupez, T., and Chenot, J. L., “Mesh Topology for Mesh Generation Problems - Application to Three-Dimensional Remeshing”, In *Numerical Methods in Industrial Forming Processes*, Chenot, Wood, and Zienkiewicz (eds.), Balkema, Rotterdam, pp. 237-242, 1992.

42. Coupez, T., “Automatic Remeshing in Three-Dimensional Moving Mesh Finite Element Analysis of Industrial Forming”, In *Simulation of Materials Processing: Theory, Practice, Methods and Applications*, Shan-Fu Shen and P. R. Dawson (eds.), Balkema, Rotterdam, pp. 407-412, 1995.

43. Tack, L. H., Schneiders, R, Debye, J., Kopp, R., Oberschelp, W., "Two- and Three-dimensional Remeshing, Mesh Refinement and Application to Simulation of Micromechanical Processes", *Computational Materials Science*, Vol. 3, pp. 241-246, 1994.

44. Tekkaya, A. E. and Kavakli, S., "3-D Simulation of Metal Forming Processes with Automatic Mesh Generation", *Steel Research*, Vol. 66, no. 9, pp. 377-383, 1995.

45. Tekkaya, A. E., Kavakli, S., Savkilioglu, M., "Rigid-Plastic Finite Element Modeling of Three-Dimensional Bulk-Metal Forming Processes", Proc. Int. Conf. And Workshop on Metal Forming Process Simulation in Industry, Baden-Baden, Germany, Vol. 1, pp. 60-79, 1994.

46. Kavakli, S. and Tekkaya, A. E., "Automatic Hexahedral Mesh Generation for the Simulation of Metal Forming Processes", Proc. 4th Int. Conf. on Computational Plasticity, Barcelona, pp. 431-442, 1995.

47. Tekkaya, A. E., "Fully Automatic Simulation of Bulk Metal Forming Processes", In *Simulation of Materials Processing: Theory, Methods and Applications*, J. Huetink and F. P. T. Baaijens (eds.), Balkema, Rotterdam, pp. 529-534, 1998.

48. Lee, Y., K. and Yang, D. Y., "Automatic Generation of Meshes with Surface Elements and Core Mesh for Finite Element Simulation of Metal Forming Processes", In *Simulation of Materials Processing: Theory, Methods and Applications*, J. Huetink and F. P. T. Baaijens (eds.), Balkema, Rotterdam, pp. 121-128, 1998.

49. Yoon, J. H. and Yang, D. Y., “A Three-Dimensional Rigid-Plastic Finite Element Analysis of Bevel Gear Forging by Using a Remeshing Technique”, *Int. J. Mech. Sci*., 32:277-291, 1990.

50. Yang, D. Y., Yoon, J. H. and Lee, N. K., “Modular Remeshing: A Practical Method of 3-D Remeshing in Forging of Complicated Parts”, *Adv. Tech. of Plasticity*, Vol. 1, pp. 171-178, 1990.

51. Shephard, M. S., Baehmann, P. L., Collar, R. R., Hattangady, N. V. and Niu, Q. "Automated Remodeling Techniques in Finite Element Analysis", in *Advances in CAD/CAE*, Academic Press, 1993.

52. Freitag, L. A. and Ollivier-Gooch, C., “The Effect of Mesh Quality on Solution Efficiency”, Proc. Of 6th International Meshing Roundtable, 1997, pp. 249.

53. Fried, I., “Accuracy of Complex Finite Elements”, *AIAA Journal*, Vol. 10, pp. 347-349, 1972.

54. Henshell, R. D., Walters, D. and Warburton, G. B., “On Possible Loss of Accuracy in Curved Finite Elements”, *J. Sound Vibration*, Vol. 23, pp. 51-513, 1972.

55. Fried, I., “Possible Loss of Accuracy in Curved (Isoparametric) Finite Elements - Comment on Paper by Henshell, et. al.”, *J. Sound Vibration*, Vol. 23, pp. 507-510, 1972.

56. Strang, G. and Fix, G. J., *An Analysis of the Finite Element Method, Prentice Hall Series in Automatic Computation*, Englewood Cliffs, New Jersey.

57. Babuska, M. and Aziz, K., "On the Angle Condition in the Finite Element Method", *SIAM J. Numer. Anal.*, Vol. 13., no. 2, pp. 214-226, 1976.

58. Krizek, M., "On the Maximum Angle Condition for Linear Tetrahedral Elements", *SIAM J. Numer. Anal.*, Vol. 29, no. 2, pp. 513-520, 1992.

59. Stricklin, J. A., Ho, W. S., Richardson, E. Q. and Haisler, W. E., "On Isoparametric vs. Linear Strain Triangular Elements", *Int. J. Numer. Meth. Engng.*, Vol. 11, pp. 1041-1043, 1977.

60. Backlund, J., "On Isoparametric Elements", *Int. J. Numer. Meth. Engng.*, Vol. 12, pp. 731-732, 1978.

61. Gifford, L. N., "More on Distorted Isoparametric Elements", *Int. J. Numer. Meth. Engng.*, Vol. 14, pp. 290-291, 1979.

62. Oddy, A., Goldak, J., McDill, M. and Bibby, M., "A Distortion Metric for Isoparametric Elements", *Transactions of the CSME*, **12**(4), pp. 213-217, 1988.

63. Berzins, M., "A Solution-Based Triangular and Tetrahedral Mesh Quality Indicator", *SIAM Journal on Scientific Computing*, Vol. 19, pp. 2051-2060, 1998.

64. Bank, R. E. and Smith, R. K., "Mesh Smoothing Using A Posteriori Error Estimates", *SIAM J. Numer. Anal.*, Vol. 34, pp. 979-997, 1997.

65. Babuska, I., Zienkiewicz, O. C., Gago, J. and Olivera D. A., eds., *Accuracy Estimates and Adaptive Refinements in Finite Element Computations*, John Wiley, 1986.

66. Flaherty, J. E., Paslow, P. J., Shephard, M. S. and Vasilakis, J. D., eds., *Adaptive Methods for Partial Differential Equations*, SIAM 1989, Philadelphia, PA.

67. Oden, J. T., Demkowicz, L., Rachowicz, W. and Westermann, T. A., "Towards a Universal h-p Adaptive Finite Element Strategy, Part 2. A posteriori Error Estimation", ", *Comp. Meth. App. Mech. Engng*., vol. 77, pp.113-180, 1989.

68. Zienkiewicz, O. C. and Zhu, J. Z., "A Simple Error Estimator and Adaptive Procedure for Practical Engineering Analysis", *Int. J. Numer. Meth. Engng.*, Vol. 24, pp. 337-357, 1987.

69. Zienkiewicz, O. C. and Zhu, J. Z., "Superconvergent Derivative Recovery Techniques and a posteriori Error Estimation in the Finite Element Method, part 1: A General Superconvergent Recovery Technique", *Int. J. Numer. Meth. Engng.*, Vol. 33, pp. 1331-1364, 1992.

70. Bass, J. M. and Oden, J. T., "Adaptive Finite Element Methods for a Class of Evolution Problems in Viscoplasticity", *Int. J. Mech. Sci.*, Vol. 26, No. 6, pp. 623-653, 1987.

71. Hattangady, N. V., "Automatic Remeshing in 3-D Analysis of Forming Processes", *Int. J. Numer. Meth. Engng*, vol. 45, pp. 553-568, 1999.

72. George, P. L., *Automatic Mesh Generation*, Addison-Wesley, 1991.

73. Hattangady, N. V., Dewasurendra, L., and Chaudhary, A. B., "Fully Automated Analysis of 3-D Metal Forming Processes", In *Simulation of Materials Processing:*

Theory, Practice, Methods and Applications, Shan-Fu Shen and P. R. Dawson (eds.), Balkema, Rotterdam, pp. 441-444, 1995.

74. Private communication, UES Software Services, Inc., 4401 Dayton-Xenia Road, Dayton, OH 45432.

75. Hughes, T. J. R., *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Prentice Hall, Englewood Cliffs, NJ 1987.

76. Zienkiewicz, O. C. and Taylor, R. L., *The Finite Element Method – Volume 1*, McGraw Hill Book Co., New York, 1987.

77. Tsukerman, I., "Comparison of Accuracy Criteria for Approximation of Conservative Fields on Tetrahedra", *IEEE Transactions on Magnetics*, Vol. 34, no. 5, September 1998.

78. Beall, M. W. and Shephard, M. S., "A General Topology-Based Mesh Data Structure", *Int. J. Numer. Meth. Engng.*, Vol. 40, pp. 1573-1596, 1997.

79. Shephard, M. S., "Approaches to the Automatic Generation and Control of Finite Element Meshes", *Appl. Mech. Rev.*, vol. 41, no. 4, 1988.

80. Shephard, M. S., "Update to: Approaches to the Automatic Generation and Control of Finite Element Meshes", *Appl. Mech. Rev.*, vol. 49, no. 10, 1996.

81. Zauderer, E., *Partial Differential Equations of Applied Mathematics*, John Wiley and Sons, New York, 1998.

82. Requicha, A. A. G. and Voelcker, H. B., "Solid Modeling: A Historic Summary and Contemporary Assessment", *IEEE Computer Graphics and Applications*, vol. 2, no. 3, pp. 9-24, 1982.

83. Requicha, A. A. G. and Voelcker, H. B., "Solid Modeling: Current Status and Research Directions", *IEEE Computer Graphics and Applications*, vol. 3, no. 7, pp. 25-37, 1983.

84. Gursoz, E. L., Choi, Y., and Priz, F. B., "Vertex-Based Representation of Non-manifold Boundaries", in *Geometric Modeling Product Engineering* (M. J. Wozny, J. U. Turner, and K. Preiss, eds.), pp. 107-130, North Holland, 1990.

85. Weiler, K. J., "The Radial-Edge Data Structure: A Topological Representation for Non-manifold Geometric Boundary Representation", in *Geometric Modeling for CAD Applications* (M. J. Wozny, H. W. MacLaughlin, and, J. L. Encarnacao, eds.), pp. 3-36, North Holland, 1988.

86. Shephard, M. S. and Finnigan, P. M., "Toward Automatic Model Generation", in *State-of-the-art Surveys in Computational Mechanics* (A. K. Noor and J. T. Oden, eds.), pp. 335-366, ASME, 1989.

87. Hattangady, N. V., Shephard, M. S. and Chaudhary, A. B., "Towards Realistic Automated 3D Modeling of Metal Forming Problems", *Engineering with Computers*, vol. 15, no. 4, pp. 356-374, 1999.

88. Dukowicz, J. K., "Conservative Rezoning (Remapping) for General Quadrilateral Mesh", *J. Comput. Physics*, vol. 54, pp. 411-424, 1984.

89. Crawfort, R. H., Anderson, D. C. and Waggenspack, W. N., “Mesh Rezoning of 2-D Isoparametric Elements by Inversion”, *Int. J. Numer. Meth. Engng.*, vol. 28, pp. 523-531, 1989.

90. Oh, S. I., Tang, S. and Badawy, A., “Finite Element Mesh Rezoning and its Applications to Metal Forming Analysis”, *Adv. Tech. Plasticity*, vol. II, pp. 1051-1058, 1984.

91. Ramshaw, J. D., “Simplified Second-Order Rezoning Algorithm for Generalized Two-Dimensional Meshes”, *J. Comput. Physics*, vol. 67, pp. 214-222, 1986.

92. Cheng, J. H. and Kikuchi, N., “A Mesh Rezoning Technique for Finite Element Simulations of Metal Forming Processes”, *Int. J. Numer. Meth. Engng.*, vol. 23, pp. 219-228, 1986.

93. Murti, V. and Valliappan, S., “Numerical Inverse Isoparametric Mapping in Remeshing and Nodal Quantity Contouring”, *Computers and Structures*, vol. 22, pp. 1011-1021, 1986.

94. Niu, Q. and Shephard, M. S., “Transfer of Solution Variables Between Finite Element Meshes”, SCOREC Tech. Report #4-1990, Scientific Computation Research Center, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, 1990.

95. Tsukerman, I., “Approximation of Conservative Fields and the Element Edge Shape Matrix”, *IEEE Transactions on Magnetics*, vol. 34, no. 5, 1998.

96. Parthasarthy, V. N., Graichen, C. M. and Hathaway, A. F., “A Comparison of Tetrahedron Quality Measures”, *Finite Elements in Analysis and Design*, vol. 15, pp. 255-261, 1993.

97. Robinson, J., “Some New Distortion Measures for Quadrilaterals”, *Finite Elements in Analysis and Design*, vol. 3, pp. 183-197, 1987.

98. MSC/PATRAN™ version 8.5, MSC.Software Corp., Los Angeles, CA.

99. I-DEAS™ version 6.0, Structural Dynamics Research Corp., Cincinnati, OH.

100. Hypermesh™ version 3.0, Altair Computing, Troy, MI.

101. Finnigan, P, Kela, A. and Davis, J., “Geometry as a Basis for Finite Element Automation”, *Engng. With Computers*, vol. 5, pp. 147-160, 1989.

102. Weiler, K. J., “Edge-Based Data Structures for Solid Modeling in Curved-Surface Environments”, *IEEE Computer Graphics and Applications*, vol. 5, no. 2, 1985.

103. Babuska, I. and Yu, D., “Asymptotically Exact A-posteriori Error Estimator for Bi-quadratic Elements”, Technical Note BN-1050, Institute for Physical Science and Technology, Laboratory for Numerical Analysis, University of Maryland, College Park, MD, 1986.

104. Babuska, I., “Adaptive Mathematical Modeling”, In *Adaptive Methods for Partial Differential Equations*, Flaherty, J. E., Paslow, P. J., Shephard, M. S. and Vasilakis, J. D., eds., SIAM 1989, Philadelphia, PA.

105. Baehmann, P. L., Shephard, M. S. and Flaherty, J. E., “A posteriori Error Estimation for Triangular and Tetrahedral Quadratic Elements Using Interior Residuals”, *Int. J. Numer. Meth. Engng.*, vol. 34, pp. 979-996, 1992.

106. Owen, Steven J., "A Survey of Unstructured Mesh Generation Technology", *Proc. 7th International Meshing Roundtable,* Dearborn, MI, October 1998.

107. Ho-Le, K., “Finite Element Mesh Generation Methods: A Review and Classification", *Computer Aided Design*, Vol 1, Num 20, pp.27-38, 1988.

108. Hattangady, N. V., "Coarsening of Mesh Models for Representation of Rigid Objects in Finite Element Analysis", *Int. J. Numer. Meth. Engng*, **44**, pp. 313-326, 1999.

109. Heckbert, P. S. and Garland, M., “Survey of Polygonal Surface Simplification Algorithms”, *Multiresolution Surface Modeling course, Siggraph ’97.*

110. Erickson, C., “Polygonal Simplification : An Overview”, Technical Report TR96-016, Department of Computer Science, UNC-Chapel Hill, January 1996.

111. Krus, M., Bourdot, P., Guisnel, F., amd Thibault, G., “Levels of Detail and Polygonal Simplification”, *Computer Graphics 3.4 Summer ’97*, *Crossroads* - ACM’s electronic publication.

112. Schroeder, W. J., Zarge, J. A., and Lorensen, W. E., “Decimation of Triangular Meshes”, *Computer Graphics* (Proc. Siggraph), Vol. 26, No. 2, July, 1992, pp. 65-70.

113. Turk, G., “Re-tiling of Polygonal Surfaces”, *Computer Graphics* (Proc. Siggraph), Vol. 26, No. 2, July 1992, pp. 55-64.

114. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W., “Mesh Optimization”, *Computer Graphics* (Proc. Siggraph), Vol. 27, No. 3, Aug. 1993, pp. 19-26.

115. Kalvin, A. D. and Taylor, R. H., “Superfaces: Polygonal Mesh Simplification with Bounded Error”, *IEEE Computer Graphics and Applications*, Feature Article, May 1996, pp. 64-77.

116. Gueziec, A., “Surface Simplification with Variable Tolerance”, *Second Annual Intl. Symposium on Medical Robotics and Computer Assisted Surgery* (MRCAS ’95), November 1995, pp. 132-139.

117. Hinker, P. and Hansen, C., “Geometric Optimization”, *Proc. Visualization ’93*, October, 1993, pp. 189-195.

118. Hamann, B., “A Data Reduction Scheme for Triangulated Surfaces”, *Computer Aided Geometric Design*, Vol. 11, No, 2, Apr. 1994, pp. 197-214.

119. DeHaemer, M. J. and Zyda, M. J., “Simplification of Objects Rendered by Polygonal Approximations”, *Computer Graphics*, Vol. 15, no. 2, 1991, pp. 175-184.

120. Varshney, A., “Hierarchic Geometric Approximations”, Ph.D thesis, Dept. of Computer Science, Univ. of North Carolina, Chapel Hill, 1994.

121. Hoppe, H., “Progressive Meshes”, *Proc. Siggraph ’96*, August 1996, pp. 99-108.

122. Klein, R., Leibich, G., and, Strasser, W., “Mesh Reduction with Error Control”, *Proc. Visualization ’96*, October, 1996, pp. 311-318.

123. Reddy, M., “SCROOGE: Perpetually-Driven Polygon Reduction”, *Computer Graphics Forum*, 15(4): 191-203, 1996.

124. Gueziec, A., “Surface Simplification Inside a Tolerance Volume”, IBM Research Report RC20440 (90191), Computer Science/Mathematics, T. J. Watson Research Center, Yorktown Heights, NY.

125. Foley, J. D., van Dam, A., Feiner, S. K. and Hughes, J. F. “Computer Graphics - Principles and Practice”, Addison-Wesley Publishing Company, New York, 1990.

126. Mortensen, M., “Geometric Modeling, John Wiley & Sons, New York, 1985.

127. Farin, G., “Curves and Surfaces for Computer Aided Geometric Design, A Practical Guide”, Second Edition, Academic Press, 1990.

128. Dey, S., Shephard, M. S. and Georges, M. K., “Elimination of the Adverse Effects of Small Model Features by the Local Modification of Automatically Generated Meshes”, *Engineering with Computers*, vol. 13, no. 3, 1997, pp. 134-152.

129. Baker, T. J., “Automatic Mesh Generation for Complex Three-Dimensional Regions Using a Constrained Delaunay Triangulation”, *Engineering with Computers*, vol. 5, 1989, pp. 161-175.

130. Cavendish, J. C., Field, D. A. and Frey, W. H., “An Approach to Automatic Three-Dimensional Mesh Generation”, *IJNME*, vol. 21, 1985, pp. 329-347.

131. de l'Isle, B. E. and George, P. L., "Optimization of Tetrahedral Meshes", Modeling, Mesh Generation and Adaptive Numerical Methods for Partial Differential Equations (Babuska, I., Flaherty, J. E., Hopcroft, J. E., Henshaw, W. D., Oliger, J. E. and Tezduyar, T., editors), Springer-Verlag, New York, 1995, pp. 97-128.

132. Schumaker, L. L., "Computing Optimal Triangulations using Simulated Annealing", *Computer Aided Geometric Design*, vol. 10, 1993, pp. 329-345.

133. Canann, S. A., Tristano, J. R. and Staten M. L., "An Approach to Combined Laplacian and Optimization-Based Smoothing for Triangular, Quadrilateral and Quad-Dominant Meshes", Ansys, Inc., Canonsburg, PA., paper available for download at *http://ansys.net/ansys/papers/meshing/combined_l_o_smoothing.pdf*.

134. Field, D., "Laplacian Smoothing and Delaunay Triangulations", *Comm. App. Num. Meth.*, vol. 4, 1988, pp. 709-712.

135. Lo, S. H., "A New Mesh Generation Scheme for Planar Domains", *IJNME*, vol. 21, 1985, pp. 1403-1426.

136. Freitag, L. A., Jones, M. T. and Plassman, P. E., "An Efficient Parallel Algorithm for Mesh Smoothing", *4th Int. Meshing Roundtable*, Sandia Labs., 1995, pp. 47-58.

137. Amenta, N., Bern, M. and Eppstein, D., "Optimal Point Placement for Mesh Smoothing", *Proc. 8th Symp. Discrete Algorithms*, SIAM, Philadelphia, 1997, pp. 528-537.

138. Canann, S. A., Muthukrishnan, S. N. and Blacker, T. D., "Optismoothing: An Optimization Driven Approach to Mesh Smoothing", *Finite Elements in Analysis and Design*, vol. 13, 1993, pp. 185-190.

139. Selfridge, D. R. (Alcoa Cleveland Works) and Watton, J. D. (Alcoa Technical Center), Private Communication.

140. *ASM Handbook of Engineering Mathematics*, American Society for Metals, Metals Park, Ohio, 1989.

141. Spiegel, Murray, *Mathematical Handbook of Formulas and Tables*, Schaum's Outline Series, McGraw Hill, Inc. 1992.

142. Batdorf, M., Freitag, L. A., and Ollivier-Gooch, C., "Computational Study of the Effect of Unstructured Mesh Quality on Solution Efficiency", *Proc. 13th AIAA Computational Fluid Dynamics Conf.*, 1997.

143. Zorin, D. and Schroder, P., "Subdivision for Modeling and Animation", Siggraph 1999 Course Notes available at http://multires.caltech.edu/pubs/sig99notes.pdf.

144. Zorin, D. and Schroder, P., "Using Subdivision Surfaces", GDC 2001 Course Notes available at http://www.mrl.nyu.edu/publications/gdc-tutorial2001.

145. Schroeder, P., Multi-resolution modeling group, http://www.multires.caltech.edu/ and http://mrsed.caltech.edu/.

146. MSC-Patran™ version 2000r2, MacNeal Schwendler Corporation, Costa Mesa, CA.