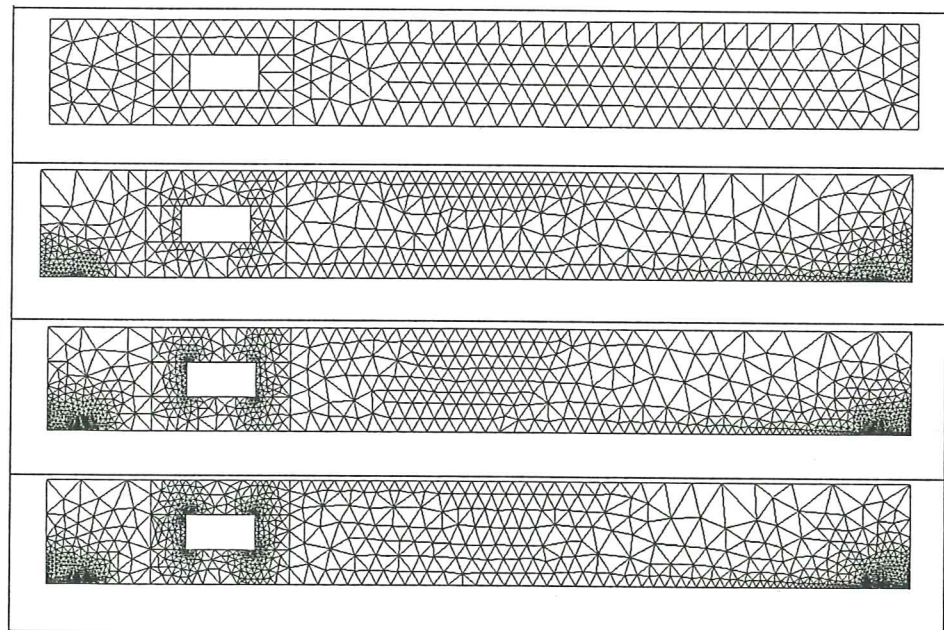


Development of an Adaptive Mesh Generator

J. Olausson
J. Gustafsson
D. Di Capua
E. Oñate



Development of an Adaptive Mesh Generator

**J. Olausson
J. Gustafsson
D. Di Capua
E. Oñate**

Publication CIMNE N°-211, March 2002

Abstract

This study thoroughly explains the process of implementing Adaptive Mesh Refinement into a Finite Element Analysis program. Using the error energy norm, a routine for improving an original mesh for an arbitrary 2D structural problem has been developed. By utilizing a user-defined relative global error, an optimal mesh for a given level of approximation can be obtained in an iterative manner. The routine (called AMG v1.0) runs with the pre-/postprocessor GiD and the calculation module Calsef, but can easily be tailored to work with other FEA programs. AMG operates with two mesh optimality criteria, which have proven to refine the mesh in a non-oscillatory manner. The open structure, along with the Graphical User Interface, makes AMG suitable for educational and experimental purposes. For validation, six examples within the field of structural engineering have been solved, four of which have been compared favorably with analytical solutions.

The mesh optimality routines used are general, why AMG can be extended to handle more types of elements, 3D analysis, and to be applicable to other types of engineering problems. Above all AMG serves as a first step of implementing Adaptive Mesh Refinement procedures into GiD.

Keywords: FEM, mesh, adaptivity, AMR, refinement, discretization error, energy norm, GiD, Calsef

Preface

This master thesis was carried out at the Department of Structural Mechanics at Chalmers, and at CIMNE (Centre Internacional de Mètodes Numèrics en Enginyeria), between July and December 2001. The project was mostly carried out at CIMNE's headquarters in Barcelona, Spain.

The objective of the project was reached after consultation with Professor Eugenio Oñate (director of CIMNE) and Ph.D. Daniel DiCapua. The latter also served as our supervisor during our time in Barcelona.

Examiner for the thesis was Professor Nils-Erik Wiberg, head of the Department of Structural Mechanics.

We would especially like to thank Daniel DiCapua, for his help and endless patience; his good mood inspired us in our work and his guidance has urged us to enjoy Barcelona. We would also like to thank Eugenio Oñate for having confidence in us, and the rest of the staff for always being friendly and helpful. Finally we thank Nils-Erik Wiberg for providing the first contact with Eugenio Oñate, and the staff at the Department of Structural Mechanics for providing the tools to finish the project in Sweden.

Göteborg in December 2001

Joel Gustafsson
Johan Olausson

Notations

Roman letters		Greek letters	
D	Constitutive matrix	β	Ratio between new and old element size
d	Number of dimensions	ε	Strain matrix
E	Young's modulus	Γ	Problem boundary
$\ e\ _E$	Error energy norm	η	Permissible relative global error
$\ e\ _{ri}$	Required error energy norm	ν	Poisson's ratio
f	Nodal force vector	σ	Exact stresses
h	Old element size	$\hat{\sigma}$	FEM stresses
\bar{h}	New element size	$\bar{\sigma}^*$	Smoothed nodal stresses
I	Unit matrix	Ω	Problem domain
K	Stiffness matrix	$\bar{\xi}_i$	Local error parameter
M	Mass matrix	ξ_g	Global error parameter
m	Degree of the shape function	ξ_i	Element refinement parameter
N	Shape function matrix		
n	Number of elements		
$\ u\ $	Strain energy norm		
u	Displacement matrix		
\bar{u}	Nodal displacement matrix		
\hat{u}	Improved displacement		

Acronyms

AMG v1.0	Adaptive Mesh Generator version 1.0 (developed program in thesis)
AMR	Adaptive Mesh Refinement
ASCII	American Standard Code for Information Interchange
C	Programming language
CAD	Computer Aided Design
CALFEM	Computer Aided Learning of the Finite Element Method
Calsef	Calculating module used for structural problems
CIMNE	Centre Internacional de Mètodes Numèrics en Enginyeria, Barcelona Spain
CPU	Central Processing Unit
FDM	Finite Difference Method
FEM	Finite Element Method
FPM	Finite Point Method
FVM	Finite Volume Method
FORTRAN	FORMula TRANslation, programming language
GiD	The personal pre- and post- processor (developed by CIMNE)
GUI	Graphical User Interfaces
MATLAB	Matrix Laboratory
NAFEMS	International Association for the Engineering Analysis Community
PC	Personal Computer
RMS	Root Mean Square
UPC	Universitat Polytècnica de Catalunya

Table of Contents

1	Introduction to the finite element method.....	6
1.1	The generalized process.....	6
1.2	A brief history.....	6
2	Meshing, error estimation and adaptivity	8
2.1	Introduction to mesh generation	8
2.2	Errors in FEM.....	11
2.3	Element discretization	11
2.4	Increasing the accuracy.....	12
2.5	Basic concepts of Adaptive Mesh Refinement	12
3	Theory [7].....	14
3.1	General equations	14
3.2	Error estimator	15
3.3	Definition of acceptable solution.....	16
3.4	Mesh optimality criteria and element sizing	16
4	The tools: GiD and Calsef	19
4.1	GiD	19
4.2	Calsef.....	21
4.3	Validation of Calsef.....	22
5	Development of an adaptive mesh generator.....	26
5.1	Work procedure	26
5.2	AMG v1.0.....	27
5.3	Step-by-step execution of AMG v1.0 on a simple example	29
5.4	Computational speed	30
6	Examples for validation of AMG v1.0	32
6.1	Simply supported beam	32
6.2	Circular plate with normal pressure.....	34
6.3	Thick circular cylinder under internal pressure	36
6.4	Stress concentration in isotropic plates.....	42
6.5	Beam with a hole	45
6.6	Concrete wall on three columns	47
7	Conclusions	50
7.1	General summary.....	50
7.2	Limitations.....	50
7.3	Comparison of optimality strategies	50
7.4	Further studies	50
8	References	51
	Appendix A	52
	Appendix B.....	60

0

Table of figures

Figure 1.1 The general process of FEM	6
Figure 1.2 History of FEM	7
Figure 2.1 Structured and unstructured mesh	8
Figure 2.2 Quadtree decomposition of a simple 2D model	8
Figure 2.3 Example of Delaunay criterion. (a) maintains the criterion, (b) does not.....	9
Figure 2.4 Example of advancing front meshing.....	9
Figure 2.5 Advancing front meshing algorithm.....	9
Figure 2.6 Radii of the circles circumscribed and inscribed in an element	10
Figure 2.7 Example of how the distortion varies with the relative side length.....	10
Figure 2.8 Distortion of meshes in figure 2.1	10
Figure 2.9 h-method refinement	12
Figure 2.10 p-method refinement	12
Figure 2.11 r-method refinement.....	12
Figure 2.12 AMR in the generalized process	13
Figure 4.1 The structure of GiD according to [9]	19
Figure 4.2 Cordal error of elements describing a curved geometry.....	20
Figure 4.3 Example of size assignment using background meshing	21
Figure 4.4 Structure of GiD environment.....	21
Figure 4.5 Elements used in validation.....	22
Figure 4.6 Cube with uniform load and two supports	23
Figure 4.7 Cube with elastic supports.....	24
Figure 5.1 The AMG environment	27
Figure 5.2 The GUI of AMG v1.0.....	28
Figure 5.3 Semi-circular arc	29
Figure 5.4 Computational speed	31
Figure 6.1 Simply supported beam.....	32
Figure 6.2 Structured mesh of simple supported beam.....	33
Figure 6.3 Thick circular plate.....	34
Figure 6.4 Meshes of thick circular plate	35
Figure 6.5 Thick cylinder under internal pressure	36
Figure 6.6 Meshes of thick cylinder	37
Figure 6.7 Strategy B; Ratio between element error and permissible error.....	38
Figure 6.8 Strategy C; Ratio between element error and permissible.....	38
Figure 6.9 Tangential stress	40
Figure 6.10 Radial stress	40
Figure 6.11 Mesh quality, Strategy B, 2207 elements	41
Figure 6.12 Mesh quality, Strategy C, 2544 elements	41
Figure 6.13 Finite plate with semicircular notch	42
Figure 6.14 Meshes for semicircular notch	42
Figure 6.15 Finite plate with central hole	43
Figure 6.16 Meshes of circular hole	43
Figure 6.17 Mesh quality of semicircular notch, using strategy C	44
Figure 6.18 Beam with a hole.....	45
Figure 6.19 Principal stresses in the beam.....	45
Figure 6.20 Concrete wall on three columns	47
Figure 6.21 Principal stresses in wall	48

1 Introduction to the finite element method

All physical problems considered within the area of mechanical engineering are described by differential equations. Usually these problems are too complex to be solved with help of exact mathematical analysis methods. The finite element method (also known as FEM) is a numerical approach by which general differential equations can be solved in an approximate manner.

The differential equations describing a physical problem are assumed to be valid over a certain region. Instead of seeking approximations that are valid over the entire region, the region is divided into smaller parts, so called finite elements, and the approximation is carried out over each element. [1]

1.1 The generalized process

The process of a finite element analysis can be described in a few general steps, shown in figure 1.1 below.

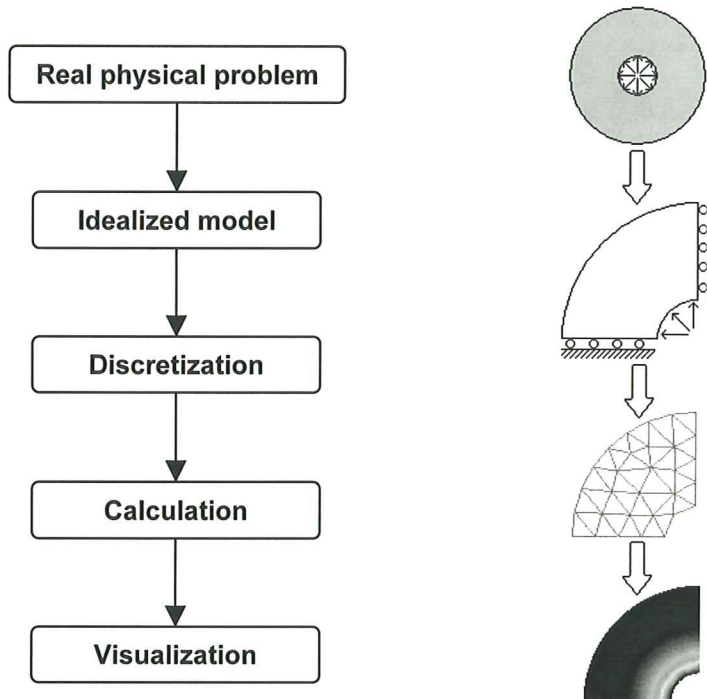


Figure 1.1 The general process of FEM

Idealization means that loads, boundary conditions, material properties etc. are given values or functions interpretable to the FE program. Discretization is the actual division of the problems geometry into a given number of elements.

Here the FE analysis is performed on a thick cylindrical pipe subjected to internal pressure. Only one quarter of the pipe is modeled and the proper boundary conditions are applied. Furthermore the pipe is considered to be infinitely long, thus the strain perpendicular to the plane is supposedly zero. This allows for the pipe to be modeled in only two dimensions.

The idealized model is then divided into triangular elements, allowing for the stresses to be calculated numerically. The stress distribution is then visualized for easy interpretation.

Because of this idealization and discretization, every solution differs from the real physical problem, even though a good model approaches the correct answer with increasing number of elements.

1.2 A brief history

The principles of the finite element were first given by the mathematician Courant in 1943. No development followed until the 1950's, when aeronautical engineers R.W. Clough in USA and J.H. Argyris in England quite independently started applying the theory on structural mechanics for elastic frames.

The first computers that appeared in the 50's could only be programmed in machine code, so the governing matrices had to be established by hand, followed by a computation based on pure matrix

algebra. The computer programs, which were used for the system analysis, were often interpretative matrix programs, written in machine code.

In more modern computer languages, such as FORTRAN, matrix handling could be abandoned for coding directly with scalars.

Although scalar coding is nowadays used for commercial programs, matrix handling is still used in training and for simpler applications, in order to get a better overview of the computational process. A common program for matrix handling is MATLAB[®].

The speed of the computers, the storage capacity and the numerical accuracy have rapidly increased the last decades allowing for the calculation of larger and larger problems.

From being a method used only for problems in structural mechanics, FEM has evolved into being a general computational tool for a wide range of engineering industries. [2]

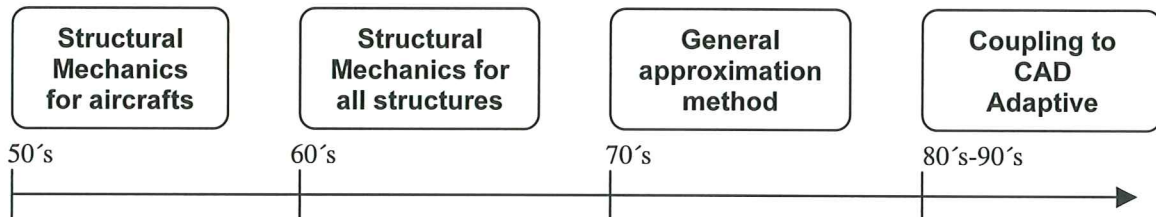


Figure 1.2 History of FEM

The future of FEM seems to involve keywords such as user-friendly and adaptive. This project deals with the latter.

2 Meshing, error estimation and adaptivity

2.1 Introduction to mesh generation

Meshing can be defined as breaking up a continuous physical domain into smaller sub-domains (elements) in order to facilitate a numerical solution to a differential equation. Generally surface domains are subdivided into triangle or quadrilateral shapes, while volumes are subdivided primarily into tetrahedral or hexahedral shapes. Furthermore mesh generation is divided into structured and unstructured meshes. A structured mesh can be recognized by all interior nodes of the mesh having an equal number of adjacent elements, whereas unstructured meshes allow any number of elements meeting at a single node, see figure 2.1.

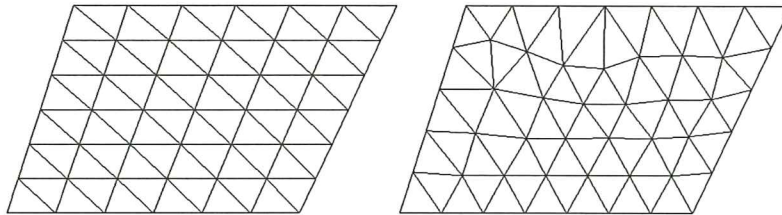


Figure 2.1 Structured and unstructured mesh

Structured meshing is sometimes referred to as 'grid generation', implying that the user has full control over the appearance of the mesh. Unstructured meshing, on the other hand, demands the preprocessor to divide the domain according to the user's requirements. These meshes are usually generated using triangular/tetrahedral elements, although quadrilateral/hexahedral elements also can be used. [3]

2.1.1 Techniques of triangular/tetrahedral meshing

Triangle and tetrahedral meshing are by far the most common forms of unstructured mesh generation. Most techniques currently in use can fit into one of three main categories [4]:

- Octree
- Delaunay
- Advancing Front

Although there is certainly a difference in complexity when moving from 2D to 3D, the algorithms discussed are for the most part applicable for both triangle and tetrahedral mesh generation.

2.1.1.1 Octree

With this method cubes containing the geometric model are subdivided until the desired resolution is reached (see figure 2.2). Irregular cells are then created where cubes intersect the surface, often requiring surface intersection calculations. Tetrahedra are generated from both the irregular cells on the boundary and the internal regular cells. The Octree technique does not match a pre-defined surface mesh, as an advancing front or Delaunay mesh might. Surface facets are instead formed wherever the internal Octree structure intersects the boundary. To ensure element sizes do not change too dramatically, a maximum difference in octree subdivision level between adjacent cubes can be limited to one. Smoothing and cleanup operations can also be employed to improve element shapes.

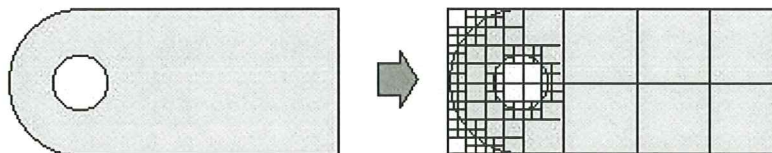


Figure 2.2 Quadtree decomposition of a simple 2D model

Octree meshing techniques are usually not implemented in commercial preprocessors.

2.1.1.2 Delaunay

The Delaunay criterion, sometimes called the "empty sphere", says that any node must not be contained within the circumsphere of any tetrahedra in the mesh. Figure 2.3 is a simple two-dimensional illustration

of the criterion. Since the circumcircles of the triangles in (a) do not contain the other triangle's nodes, the empty circle property is maintained.

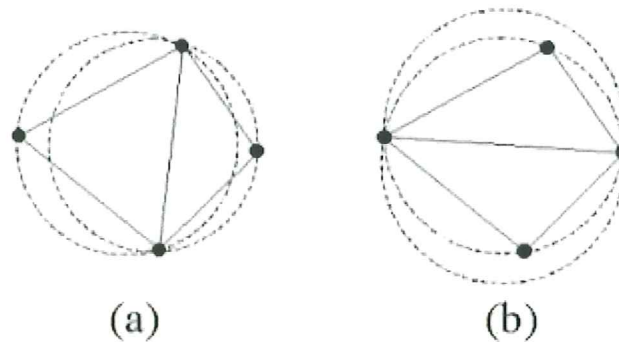


Figure 2.3 Example of Delaunay criterion. (a) maintains the criterion, (b) does not

The Delaunay criterion is not an algorithm for generating a mesh, but it provides a criterion for connection of a set of existing points in space. As such it is necessary to provide a method for generating node locations within the geometry. A typical approach is to first mesh the boundary of the geometry to provide an initial set of nodes. The boundary nodes are then triangulated according to the Delaunay criterion. Nodes are then inserted incrementally into the existing mesh, redefining the triangles or tetrahedra locally as each new node is inserted to maintain the Delaunay criterion.

The most common meshing techniques utilize the Delaunay criterion.

2.1.1.3 Advancing front

Another very popular family of triangle and tetrahedral mesh generation algorithms is the advancing front, also called moving front method. In this method, the tetrahedra are built progressively inward from the triangulated surface. An active front is maintained where new tetrahedra are formed. Figure 2.4 is a simple two-dimensional example of the advancing front, where triangles have been formed at the boundary. As the algorithm progresses, the front will advance to fill the remainder of the area with triangles.

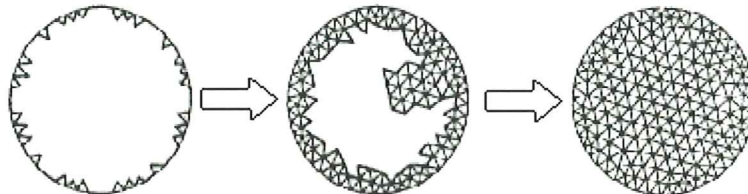


Figure 2.4 Example of advancing front meshing

In three-dimensions, for each triangular facet on the front, an ideal location for a new fourth node is computed. Also determined are any existing nodes on the front that may form a well-shaped tetrahedron with the facet. The algorithm selects either the new fourth node or an existing node to form the new tetrahedron based on which will form the best tetrahedron. Also required are intersection checks to ensure that tetrahedron do not overlap as opposing fronts advance towards each other.

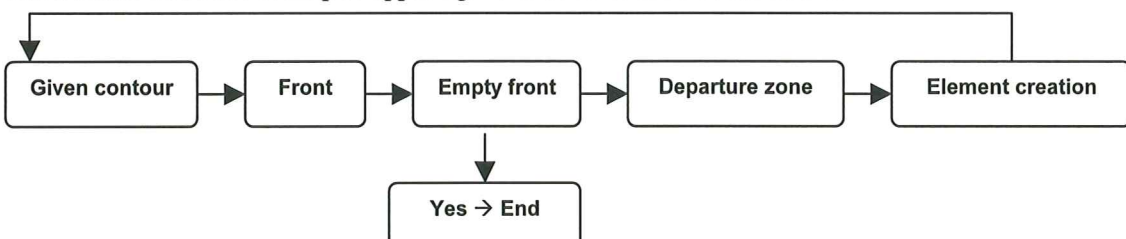


Figure 2.5 Advancing front meshing algorithm

2.1.2 Mesh quality

A good mathematical model and a sufficiently dense mesh are both vital for achieving a good approximation using FE analysis. Apart from this, one needs a criterion for measuring the quality of each element in the mesh. The most essential condition is the element distortion, as described by [5]. This is obtained, as shown in figure 2.6, by circumscribing and inscribing a circle in the 2D element.

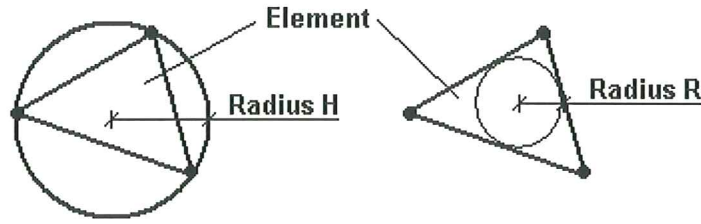


Figure 2.6 Radii of the circles circumscribed and inscribed in an element

The element distortion factor D is defined as the ratio between the two radii, $D=H/R$, where:

$$H = \frac{a \cdot b \cdot c}{4A} \quad \text{and} \quad R = \frac{2A}{a + b + c}$$

Here a, b and c denote the element side length and A denotes the area.

2.1.2.1 The acceptable geometry

The optimal shape of a three-node element is of course the equilateral triangle. With $a=b=c$, the distortion factor becomes:

$$D = \frac{H}{R} = \frac{3a^4}{8A^2} = \frac{3a^4}{8 \cdot \left(\frac{\sqrt{3}}{4} a^2\right)^2} = 2$$

Creating a mesh consisting of only optimal elements is of course only possible for certain geometries. The distortion should usually be below 5, but values in the order of 10 are acceptable. D-values above 15 usually imply a mesh not suitable for FE-calculations [5].

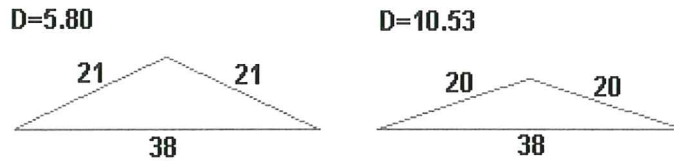


Figure 2.7 Example of how the distortion varies with the relative side length

Plotting the distortion over the domain can reveal areas with poor element quality, thus allowing for the user to manually edit the mesh. An example of the quality concerning distortion for the meshes shown in figure 2.1 is given below.

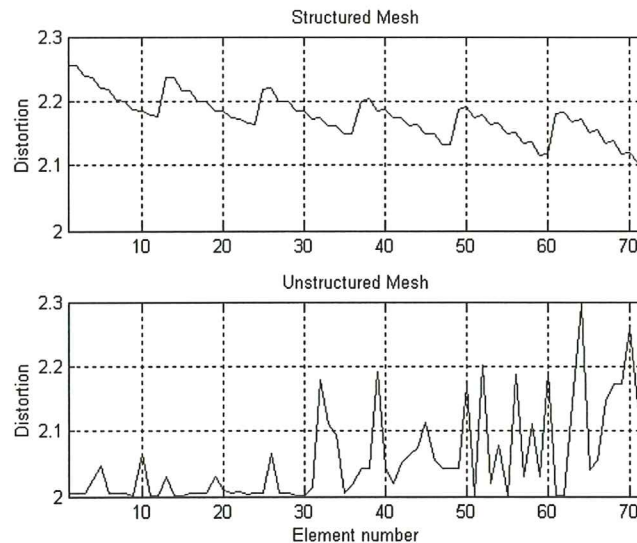


Figure 2.8 Distortion of meshes in figure 2.1

Both meshes prove to be of high quality, concerning distortion

2.2 Errors in FEM

Since FE analysis is based on different types of approximations, every solution has errors. These errors can be grouped as follows [2]:

Modeling errors (physical modeling)

1. Choice of continuum for the structure (e.g. approximation of a 3D-model to simplified models of 2D or 1D)
2. Uncertainties in input data (e.g. geometry, load, constitutive parameters)

Discretization errors (numerical modeling)

3. Choice of mathematical model (differential equation + variational formulation)
4. Choice of approximation polynomial within element and choice of coupling between elements.
5. Element discretization
6. Cordal error (i.e. curved boundaries are described by polygons)

Manipulation errors (numerical calculations)

7. Errors in input data
8. Truncation errors (e.g. numerical integration)
9. Rounding-off errors during calculations due to the computer handling a limited amount of digits

The errors 4-6 above will decrease with a finer mesh, whilst error number 9 will increase, although this error is becoming less significant with better computers.

2.3 Element discretization

This project deals with errors associated with discretization. For structural analysis these errors are found in displacement u , stresses σ and strains ε , and can be written as [2]:

$$e(x) = u(x) - u^h(x)$$

$$e_\sigma(x) = \sigma(x) - \sigma^h(x)$$

$$e_\varepsilon(x) = \varepsilon(x) - \varepsilon^h(x)$$

where $v(x)$ is the exact solution and $v^h(x)$ the approximate one obtained from the FE-calculation, $v = u, \sigma, \varepsilon$ and $x = x, y, z$. Clearly the real error cannot be calculated if the exact solution is not known, which is most commonly the case. Even so, it is possible to estimate the error and estimate how this decreases with increasing number of elements.

The specification of local error in the manner given in equations above is a simplified approach to adaptivity, and can be misleading. For instance, under a point load both errors in displacements and stresses will be locally infinite but the overall solution may well be acceptable. Similar situations exist near re-entrant where stress singularities exist in elastic analysis. For this reason various norms representing some integral scalar quantity are often introduced to measure the error [6]. This is discussed in Chapter 3.

2.4 Increasing the accuracy

The accuracy in a FE-calculation can be increased in mainly three ways [2]:

- The mesh density is increased by using more elements of the same type (h-method)

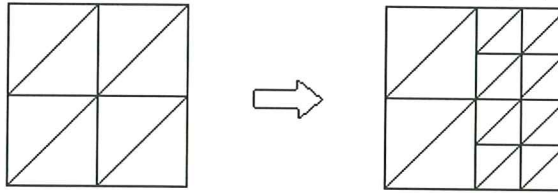


Figure 2.9 h-method refinement

- The polynomial degree for the element approximation functions is increased, while the mesh is kept the same (p-method)

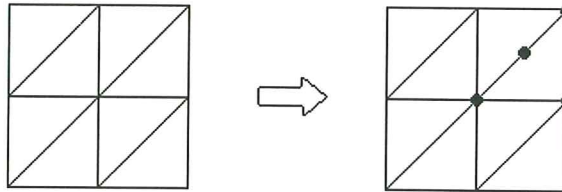


Figure 2.10 p-method refinement

- Change of the element geometry with the same element mesh and element approximation (r-method)

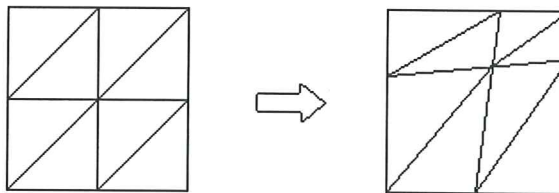


Figure 2.11 r-method refinement

The most common methods are the h- and p-methods. A combination of these (known as the hp-method) has shown to be very effective.

Furthermore the h-method can be divided into two subclasses [6]:

1. The first h-refinement is element subdivision (as seen in figure 2.9). Here refinement can be conveniently implemented and existing elements are simply divided into smaller elements, keeping the original element boundaries intact. The problem with this method is the *hanging points*, which are created where an element with mid-side nodes is joined to an element with no such nodes. On such occasions it is necessary to provide local constraints at the hanging points and the calculations become more involved. In addition de-refinement requires rather complex data management, which may reduce the efficiency of the method. Nevertheless, the method of element subdivision is quite widely used.
2. The second method is that of complete mesh regeneration. Here, on the basis of a given solution, a new element size is predicted in the entire domain and a totally new mesh is generated. This allows for a simultaneous refinement and de-refinement. This of course can be expensive, especially in three dimensions where mesh generation is difficult for certain types of elements. However, the results are generally superior, why this method has been utilized in this project.

2.5 Basic concepts of Adaptive Mesh Refinement

Increasingly larger and more complex designs are being simulated using the finite element method. With its increasing popularity comes the incentive to improve automatic meshing algorithms.

At the inception of the finite element method, most users were satisfied to simulate vastly simplified forms of their final design utilizing only tens or hundreds of elements. Thorough preprocessing was required to subdivide domains into usable elements. Market forces have now pushed meshing technology to a point where users expect to mesh complex domains with thousands or millions of elements with no

more interaction than the push of a button. Clearly a very dense mesh will produce a fairly accurate solution, given that the modeling/manipulation errors are small. However, a large number of elements result in a large number of equations and a long calculation time. Although the computers of today have a much greater capacity than say ten years ago, this is still a matter of great concern for computational engineers. The solution is obviously to create mesh with a non-uniform distribution of element sizes, i.e. smaller elements in some parts of the geometry and larger in others [3].

Adaptive Mesh Refinement (AMR) is a technique of refining (and derefining) an initial mesh with the aim of creating a mesh that is optimal for the user's needs. This refining is based on the approximated error described in Chapter 3.

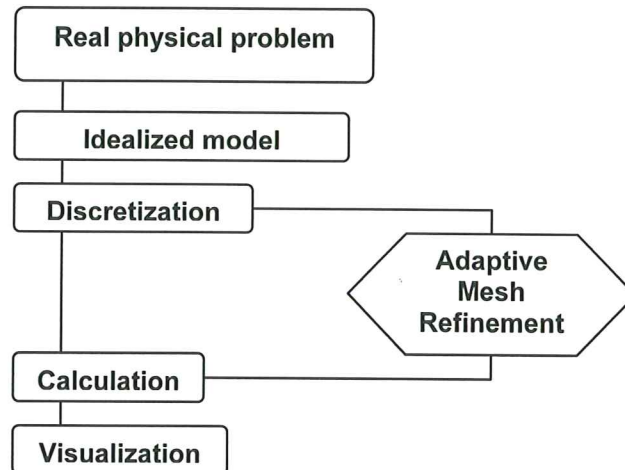


Figure 2.12 AMR in the generalized process

The AMR process above is run as long as the mesh optimality criterion is unfulfilled.

Estimation of the discretization error is a crucial matter of Adaptive Mesh Refinement. Achieving a mesh that is optimal for the user's needs is a matter of efficiency as well as economy. A mesh is considered as optimal when the error is equally distributed over all the elements, which in practice means that the element size is unequally distributed over the model.

The AMR procedures largely rely on the decoupling of the concepts of error measure and mesh optimality criteria. This leads to the definitions of the global and local error parameters, which are the foundation of the element refinement procedure. Since AMR is an iterative procedure the rate of convergence is of great importance, as well as the speed of each iteration in the process.

Basically two mesh optimality criteria have been used [7]:

1. Equal distribution of the global error over all elements.
2. Equal distribution of the specific error (i.e. the error per unit area for the 2D case)

The latter allows the concentration of more and smaller elements in areas with higher gradients of stress.

3 Theory [7]

3.1 General equations

In 2D elasticity two cases are considered, namely plane stress and plane strain.

Consider the solution of a problem governed by a system of differential equations written in the general matrix form:

$$\mathbf{A}(\mathbf{u}) = 0 \text{ in } \Omega \quad (1)$$

with boundary conditions

$$\mathbf{B}(\mathbf{u}) = 0 \text{ in } \Gamma \quad (2)$$

where Ω is the area of the analysis domain and Γ its boundary. In (1) and (2) \mathbf{u} is the vector containing the displacements, which relates to the strain stress vectors as:

$$\boldsymbol{\varepsilon} = \nabla \mathbf{u} \quad (3)$$

$$\boldsymbol{\sigma} = \mathbf{D}\boldsymbol{\varepsilon} \quad (4)$$

where

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{bmatrix}, \quad \nabla = \begin{bmatrix} \partial/\partial x & 0 \\ 0 & \partial/\partial y \\ \partial/\partial y & \partial/\partial x \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix}, \quad \boldsymbol{\sigma} = \begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix}$$

and

$$\mathbf{D} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1}{2}(1-\nu) \end{bmatrix}, \text{ for plane stress}$$

$$\mathbf{D} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & \frac{1}{2}(1-2\nu) \end{bmatrix}, \text{ for plane strain}$$

\mathbf{D} is the constitutive matrix, containing the material properties, namely E (Young's modulus of elasticity) and ν (Poisson's ration).

In the finite element solution the displacements \mathbf{u} are interpolated as:

$$\mathbf{u} \approx \hat{\mathbf{u}} = \sum_{i=1}^n \mathbf{N}_i \bar{\mathbf{u}} = \mathbf{N} \bar{\mathbf{u}} \quad (5a)$$

where

$$\mathbf{N}_i = \mathbf{N}_i \mathbf{I} \quad (5b)$$

N_i are the elements' shape functions, $\bar{\mathbf{u}}$ are the nodal displacements and n is the total number of nodes in the mesh.

Substituting (5) in (1) and (2) and using the weighted residual method leads to an equation system of the form:

$$\mathbf{K} \bar{\mathbf{u}} = \mathbf{f} \quad (6)$$

\mathbf{K} is the stiffness matrix and \mathbf{f} is the equivalent nodal force vector, obtained by assembly of the elements' contributions.

When these nodal variables have been calculated, the strains and stresses are approximated as follows:

$$\boldsymbol{\varepsilon} \approx \hat{\boldsymbol{\varepsilon}} = \sum_{i=1}^n \mathbf{B}_i \bar{\mathbf{u}}_i \quad (7a)$$

$$\boldsymbol{\sigma} \approx \hat{\boldsymbol{\sigma}} = \sum_{i=1}^n \mathbf{D} \mathbf{B}_i \bar{\mathbf{u}}_i \quad (7b)$$

where

$$\mathbf{B}_i = \nabla \mathbf{N}_i$$

3.2 Error estimator

As a measure of the error function e a non-negative number, called a norm $\|e\|$, is used. Basically there are three different norms used in connection with finite element analysis, i.e. the **L₂-norm** $\|e\|_0$, the **max-norm** $\|e\|_\infty$ and the **energy norm** $\|e\|_E$. We will now take a closer look at the latter, which is often used for elliptic formulations (e.g. structural problems):

$$\|e\|_E = \left[\int_{\Omega} \left[\boldsymbol{\sigma} - \hat{\boldsymbol{\sigma}} \right]^T \mathbf{D}^{-1} \left[\boldsymbol{\sigma} - \hat{\boldsymbol{\sigma}} \right] d\Omega \right]^{1/2} \quad (8)$$

Here $\boldsymbol{\sigma}$ are the exact stresses and $\hat{\boldsymbol{\sigma}}$ are the stresses obtained from the finite element solution (7b). In practical use the exact stresses are not known, why they are approximated as:

$$\boldsymbol{\sigma} \approx \boldsymbol{\sigma}^* = \mathbf{N}_\sigma \bar{\boldsymbol{\sigma}}^* \quad (9)$$

\mathbf{N}_σ are stress interpolating functions (based on the element shape functions) and $\bar{\boldsymbol{\sigma}}^*$ are nodal stress values derived from e.g. local or global least square smoothing. We will hereafter consider global smoothing, which yields the nodal smoothed values as:

$$\boldsymbol{\sigma}^* = \mathbf{M}^{-1} \int_{\Omega} \mathbf{N}_\sigma \hat{\boldsymbol{\sigma}} d\Omega \quad (10)$$

where

$$\mathbf{M} = \begin{bmatrix} \mathbf{N}_1 \mathbf{N}_1 & \mathbf{N}_1 \mathbf{N}_2 & \dots & \mathbf{N}_1 \mathbf{N}_N \\ \mathbf{N}_2 \mathbf{N}_1 & \mathbf{N}_2 \mathbf{N}_2 & \dots & \mathbf{N}_2 \mathbf{N}_N \\ \dots & \dots & \dots & \dots \\ \mathbf{N}_N \mathbf{N}_1 & \dots & \dots & \mathbf{N}_N \mathbf{N}_N \end{bmatrix}, \quad \text{with } \mathbf{N} \text{ as (5b)}$$

Equation (10) will provide an accurate smoothed stress field for linear elements. The strain energy U of the exact solution is estimated as:

$$U = \|\mathbf{u}\|^2 \approx \int_{\Omega} \boldsymbol{\sigma}^{*T} \mathbf{D}^{-1} \boldsymbol{\sigma}^* d\Omega + \|\mathbf{e}\|^2 \quad (11)$$

$\|\mathbf{e}\|$ and $\|\mathbf{u}\|$ can be evaluated as the sum of their element contributions as:

$$\|\mathbf{e}\|^2 = \sum_{i=1}^n \|\mathbf{e}\|_i^2 \quad \|\mathbf{u}\|^2 = \sum_{i=1}^n \|\mathbf{u}\|_i^2 \quad (12)$$

3.3 Definition of acceptable solution

An acceptable solution satisfies the following two conditions:

1. The global error in the energy norm is smaller than a specified value of the total strain energy:

$$\|\mathbf{e}\| \leq \eta \cdot \|\mathbf{u}\| \quad (13)$$

where η is the user-specified value of the permissible relative global error. Equation (6) allows to define the global error parameter ξ_g as:

$$\xi_g = \frac{\|\mathbf{e}\|}{\eta \cdot \|\mathbf{u}\|} \quad (14)$$

A value $\xi_g \leq 1$ implies satisfaction of the global error condition, whilst $\xi_g > 1$ indicates that further mesh refinement is necessary. An AMR strategy exclusively based on the global error condition will result in a uniform mesh refinement/derefinement until $\xi_g = 1$. A local error indicator is therefore needed.

2. The element distribution satisfies a ‘mesh optimality criterion’, which can be described as:

$$\|\mathbf{e}\|_i = \|\mathbf{e}\|_{ri} \quad (15)$$

where $\|\mathbf{e}\|_i$ is the actual error norm in each element and $\|\mathbf{e}\|_{ri}$ is the required error norm in the element. The local error parameter is defined as:

$$\bar{\xi}_i = \frac{\|\mathbf{e}\|_i}{\|\mathbf{e}\|_{ri}} \quad (16)$$

A value of $\bar{\xi}_i = 1$ indicates an optimal element size, while $\bar{\xi}_i > 1$ and $\bar{\xi}_i < 1$ indicate that the element needs further refinement or derefinement respectively. The definition of the required error norm $\|\mathbf{e}\|_{ri}$ is of great importance, since it strongly affects the distribution of element sizes in the mesh. This definition can be based on different mesh optimality criteria, of which this project considers mainly two: *equal distribution of the global* and *specific error*, respectively.

3.4 Mesh optimality criteria and element sizing

Generally one aims to satisfy both local and global error conditions when using AMR. This calls for the definition of an element refinement parameter using (14) and (16) as:

$$\xi_i = \bar{\xi}_i \cdot \xi_g = \frac{\|\mathbf{e}\| \cdot \|\mathbf{e}\|_i}{\eta \cdot \|\mathbf{u}\| \cdot \|\mathbf{e}\|_{ri}} \quad (17)$$

The element refinement parameter was first introduced by Zienkiewicz and Zhu [8] for defining the new element size in a general AMR strategy. The expression in (17) can be interpreted as the result of trying to satisfy the global and local error conditions in a successive manner. Individually the terms in (17) can play a very different role, as explained in sections 3.4.1-3.4.2.

3.4.1 Strategy A and B: Equal distribution of the global error

This criterion is based on the definition of a mesh being optimal when the energy norm is equally distributed between all elements. On the basis of this, the required error for each element can be defined as the ratio between the global error and the total number of elements in the mesh. Bearing in mind that only the square of the error norm is additive yields:

$$\|e\|_{ri} = \frac{\|e\|_i}{\sqrt{n}} \quad (18)$$

Using (18) in combination with (16) and (17) yields the local error parameter and element refinement parameter as:

$$\bar{\xi}_i = \frac{\|e\|_i}{\|e\|_n^{-1/2}} \quad (19)$$

$$\xi_i = \bar{\xi}_i \cdot \xi_g = \frac{\|e\|_i}{\eta \cdot \|u\|_n^{-1/2}} \quad (20)$$

The element refinement parameter ξ_i can now be interpreted as the ratio between the element error and the distributed value of the permissible error over the mesh. Again $\xi_i > 1$ means that the mesh needs further refinement, whereas $\xi_i \leq 1$ implies that both local and global error conditions are satisfied.

Next, let us consider the convergence rates of the element and global error. Assuming that m is the degree of the shape function polynomials in (5) ($m=1$ for linear elements, $m=2$ for quadratic elements etc.) and that only first derivatives of \mathbf{u} are involved in the strain operator \mathbf{L} of (3), it is easy to find the global error norm (8):

$$\|e\|_E = \left[\int_{\Omega} \mathbf{O}(h^m) \mathbf{D}^{-1} \mathbf{O}(h^m) d\Omega \right]^{1/2} \approx \mathbf{O}(h^m) \quad (21)$$

where h is the average element size of all elements in the mesh. The element error norm is obtained as:

$$\|e\|_{Ei} = \left[\int_{\Omega} \mathbf{O}(h_i^m) \mathbf{D}^{-1} \mathbf{O}(h_i^m) d\Omega \right]^{1/2} \approx \mathbf{O}(h_i^m) \Omega^{1/2} \approx \mathbf{O}(h_i^{m+(d/2)}) \quad (22)$$

where h_i and d are the element size and the number of dimensions of the problem, respectively.

Using (21) and (22) allows for the definition of the new element size in terms of the existing element size as:

$$\bar{h}_i = \frac{h_i}{\beta_i} \quad (23)$$

with β_i as:

$$\beta_i = (C \xi_i)^{1/m} \quad (\text{Strategy A}) \quad (24a)$$

$$\beta_i = \xi_g^{1/m} \bar{\xi}_i^{2/(2m+d)} \quad (\text{Strategy B}) \quad (24b)$$

where m is the degree of the shape functions ($m=1$ for linear elements, $m=2$ for quadratic elements and so on), and d is the number of dimensions in the FE analysis. C is the relaxation factor (generally $C=1$ is taken) and $m'=m$ except for elements adjacent to singularities, where $m=\lambda$ is used.

Strategy A is not considered from now on, since it has proven to adapt the mesh in an oscillatory manner, see [7].

3.4.2 Strategy C: Equal distribution of the specific error

An alternative definition of mesh optimality is the equal distribution of the square error norm per unit area or volume, i.e. for the optimal mesh the following conditions is fulfilled:

$$\frac{\|e\|_i^2}{\Omega_i} = \frac{\|e\|^2}{\Omega} \quad (25)$$

where Ω_i and Ω denote the area or volume of the element and total mesh, respectively. Comparing (25) with (15) yields the expression for the required error norm for each element as:

$$\|e\|_{ri} = \|e\| \left(\frac{\Omega}{\Omega_i} \right)^{1/2} \quad (26)$$

Using (17) and (26) the local error parameter is obtained as:

$$\bar{\xi}_i = \frac{\|e\|_i}{\|e\|} \left(\frac{\Omega}{\Omega_i} \right)^{1/2} \quad (27)$$

The element refinement parameter is obtained as:

$$\xi_i = \bar{\xi}_i \xi_g = \frac{\|e\|_i}{\eta \|u\|} \left(\frac{\Omega}{\Omega_i} \right)^{1/2} \quad (28)$$

The new element size can now be obtained by using (21), with β_i now given by:

$$\beta_i = (\xi_i \xi_g)^{1/m} = (\xi_i)^{1/m} \quad (29)$$

4 The tools: GiD and Calsef

4.1 GiD

GiD is a pre- and post-processing tool developed by CIMNE (International Center for Numerical Methods in Engineering) in Barcelona, Spain. The aim has been to create an adaptive and user-friendly environment for universities and companies, who want the ability to customize their FE modeling. The graphical user interface allows for the user to define all data related to numerical simulation, including definition of geometry, materials, conditions, solution information etc. The environment of GiD is somewhat different compared with traditional pre- and postprocessors, see figure 4.1.

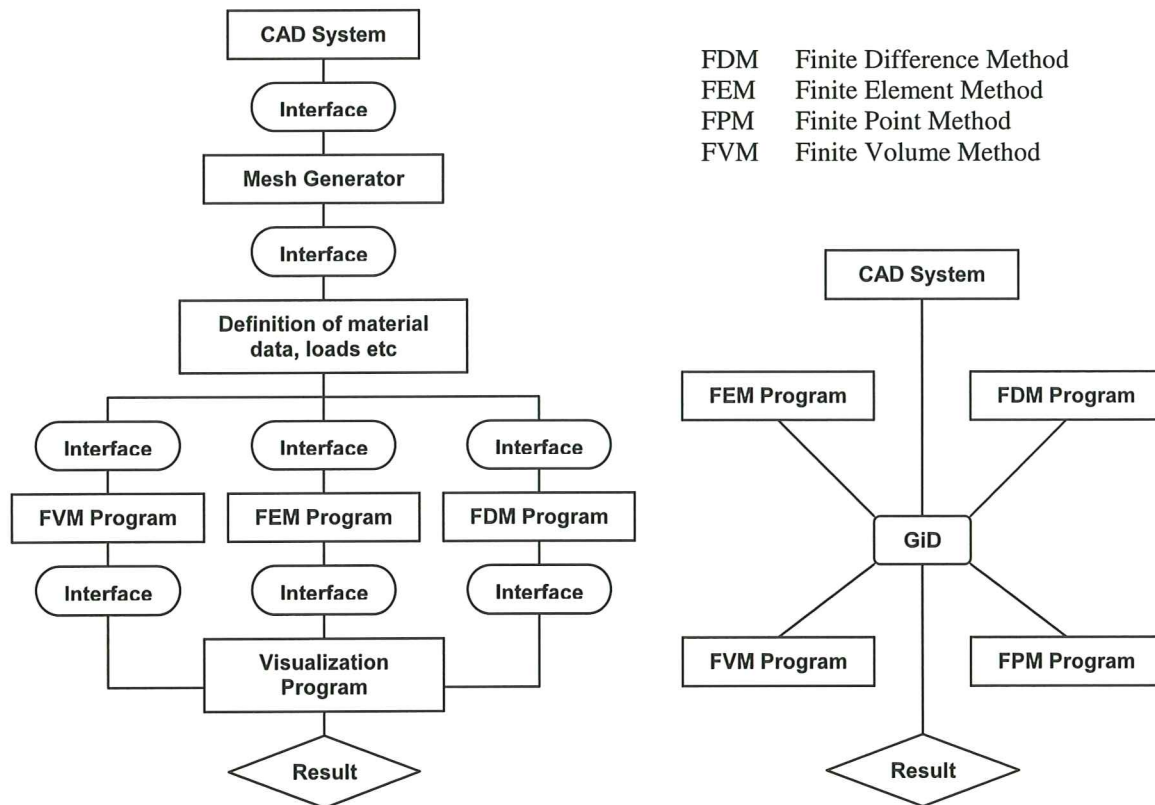


Figure 4.1 The structure of GiD according to [9]

GiD can be customized so that the data relevant to the user is generated from the calculation module. This is done using a number of external files written in a local high-level language, for easy editing.

GiD also possesses the ability to run the numerical simulation from within the program. Since the program is highly customizable, any simulation code can be used, although CIMNE provides a range of calculation modules:

- Emant – 2D magnetostatic problems
- Caltep – Heat transfer and electromagnetic analysis
- Shyne – Fluid dynamics
- Calsef – Structural problems

4.1.1 Mesh generation in GiD [10]

GiD supports both structured and unstructured meshing, of which the latter naturally is of most interest. In order to ensure speed and portability GiD utilizes the advancing front technique for unstructured mesh generation (see section 2.1.1.3). Unstructured sizes can be assigned by a number of criteria:

4.1.1.1 Geometric entities

Sizes can be assigned to geometric entities such as points, lines, surfaces and volumes. This results in the element sizes in the vicinity of the chosen entity being approximately that. The transition between different sizes can also be controlled in order to allow for the generation of a 'smooth' mesh.

4.1.1.2 Cordal error

Cordal error is defined as the maximum distance between an element defining a boundary and the actual geometry.

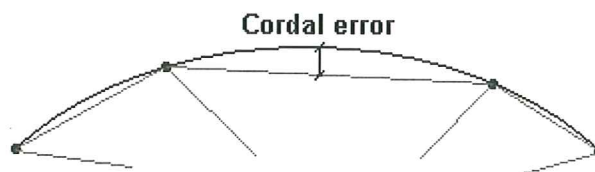


Figure 4.2 Cordal error of elements describing a curved geometry

GiD assigns the corresponding sizes to all the entities to accomplish this condition. It will only change the current sizes if the new one is smaller than the previously defined.

4.1.1.3 Correct sizes

With this option one can prescribe the minimum and maximum element sizes in a mesh. It is also possible to assign sizes according to the shape of the geometry, which implies that smaller elements will be assigned to smaller surfaces/volumes.

4.1.1.4 Background mesh

A recent additional feature of GiD is the possibility of 'background meshing', i.e. mesh refining by reading an ASCII file containing the desired size of elements. The implementation so far is done for triangular elements as shown in figure 4.3. Apart from this the file must include information about the original geometry, such as nodal coordinates and connectivity, see example below.

```
BackgroundMesh V 1.0
MESH dimension 2 ElemType Triangle Nnode 3
Coordinates
1 5.61705 4.81504 0.00000
...
51 -5.64191 -1.53335 0.00000
end coordinates
Elements
1 24 16 26
...
76 34 31 28
end elements
DesiredSize Elements
1 0.20000
...
76 1.50000
End DesiredSize
```

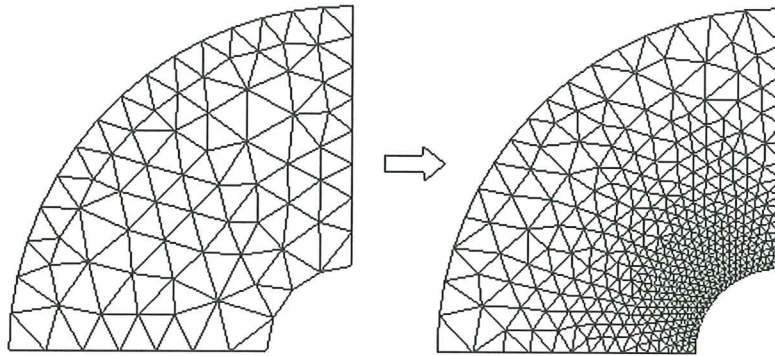



Figure 4.3 Example of size assignment using background meshing

Unfortunately the feature is not fully implemented yet, e.g. quadrilateral/hexahedra elements are not yet supported. Nevertheless background meshing is a first step towards adaptive mesh generation in GiD. In its current state, the feature allows for an external program to automatically create a file with the appropriate information and read it into GiD. The development of such a program is described in Chapter 5.

4.1.1.5 Note on element sizes

Although the size of elements is often mentioned in literature concerning meshing, a definition of it is harder to find. Typically element size is referred to as the area (for 2D elements), the average side length or the average height. In GiD the two latter are used; side length for normal meshing and height for background meshing. In this report element size is always referred to as the triangular elements mean side length.

4.2 Calsef

Calsef is the in-house numerical simulation code for structural analysis. The program is divided into several separated calculation modules dealing with 2D and 3D solids, plates, 3D shells and axisymmetric solids and shells [11].

When executed along with GiD, a number of output files are produced. These files are entirely in ASCII format, and can therefore be read by either GiD's postprocessor or by an external program (e.g. an AMR module), see figure 4.4.

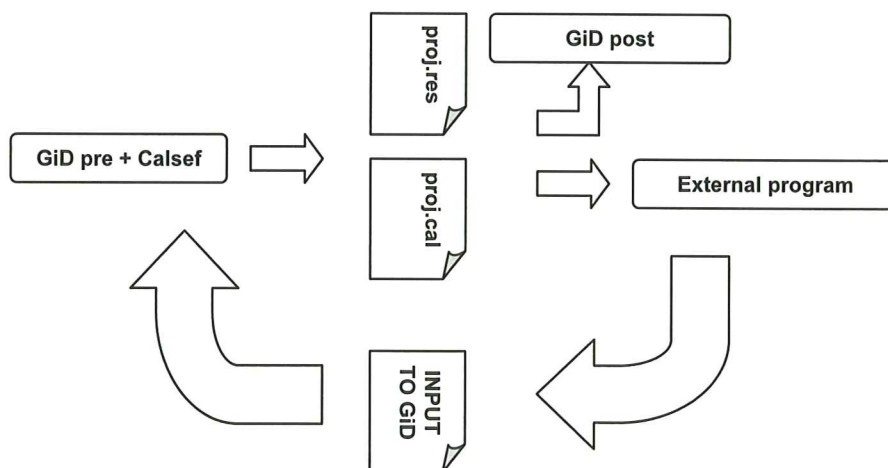


Figure 4.4 Structure of GiD environment

GiD can be customized so that the first output file (proj.res) contains all relevant data about the mesh (number of elements and nodes, coordinates, connectivity and element type), material properties and the type of analysis. The second file (proj.cal) will contain information about nodal displacements and the smoothed nodal stress values.

The Adaptive Mesh Generator described in Chapter 5 is based entirely on the output files created by the 2D plane state module, why a closer look on this is necessary. The fact that a new beta-version of Calsef (Calsef 2001) recently has been released calls for a validation before this can be safely relied on.

4.3 Validation of Calsef

In all scientific work it is important to verify all methods and tools that are used in the project. It is therefore important to this project that a validation of the program Calsef is made, since the calculation of the error norm is based on the results from this program. A beta version of the program, named Calsef 2001 has been released and needs to be tested. The beta version has included more options, e.g. elastic constraints and linear loads. For validation some examples are calculated and compared with analytical solutions. Problems related with the boundary conditions have been noticed and these three simple tests are therefore required. Each example is modeled four times using different types of elements; triangles (3 and 6 nodes) and quadrilateral (4 and 8 nodes). When the triangles are used, 2 elements describe the geometry, and for the quadrilaterals only 1 element is used (see figure 4.5).

4.3.1 Meshes used in validation

The different elements, which are used in the examples below, are triangular 3 and 6 nodes and rectangular 4 and 8 nodes. In the examples the same type of element structure is used, see figure 4.5.

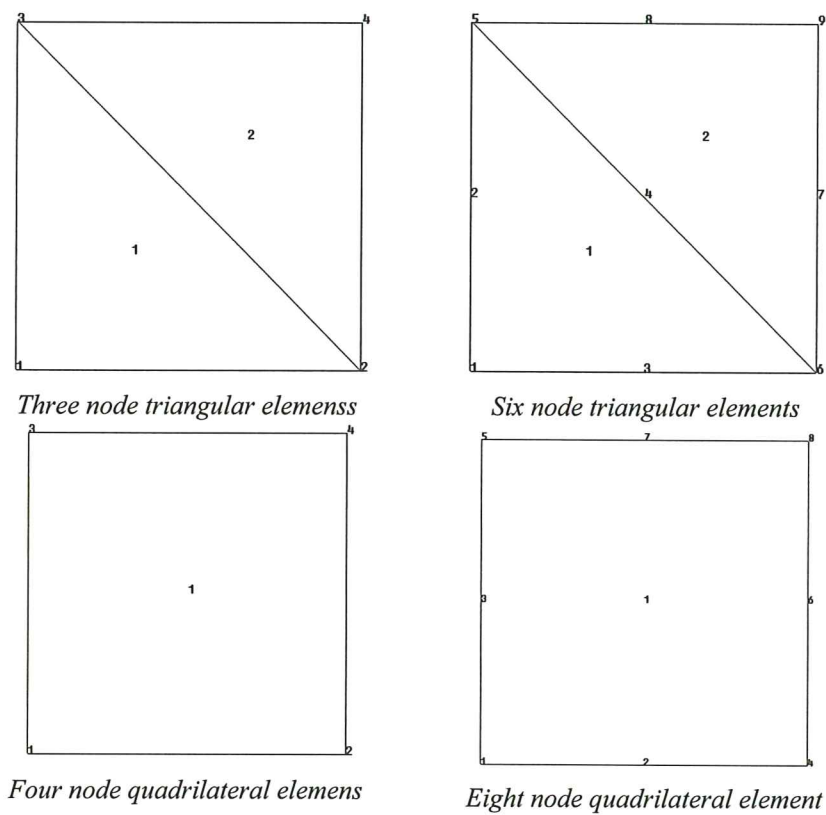


Figure 4.5 Elements used in validation

4.3.2 Example 1 - Cube with a uniform load and 2 supports

This example is chosen because it is easy to model and the analytical solution is well known. No self-weight is considered.

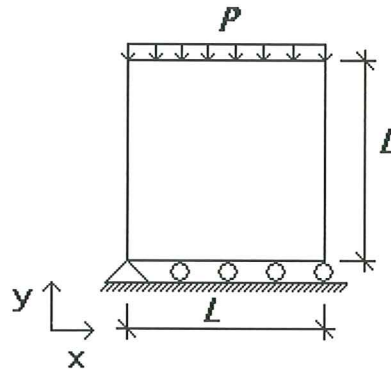


Figure 4.6 Cube with uniform load and two supports

The example is studied using different types of element and number of nodes and is performed in plane stress. The material and geometry parameters are:

$$P = 10 \text{ kN/m}$$

$$L = 1 \text{ m}$$

$$\text{thickness} = 1 \text{ m}$$

$$\nu = 0$$

$$E = 1 \text{ MPa}$$

The analytical solution of this example is [12]:

$$\delta = \frac{P \cdot L}{E \cdot A} = \frac{1 \cdot 10^4 \cdot 1 \cdot 1}{1 \cdot 10^6 \cdot 1 \cdot 1} = 1 \cdot 10^{-2} \text{ m}$$

$$\sigma_x = 0$$

$$\sigma_y = P \cdot A = 10 \cdot 10^3 \cdot 1 \cdot 1 = 1 \cdot 10^4 \text{ Pa}$$

$$\tau_{xy} = 0$$

4.3.2.1 The Calsef 2001 solutions for Example 1

The results from Calsef 2001 (See Appendix B) for each element type all correspond with the analytical solution.

$$\delta = 1 \cdot 10^{-2} \text{ m}$$

$$\sigma_x \approx 0$$

$$\sigma_y = P = 1 \cdot 10^4 \text{ Pa}$$

$$\tau_{xy} \approx 0$$

All nodes with the same y-coordinates have the same y-displacement, as they are supposed to have because of the symmetry in the model.

4.3.3 Example 2 – 45° rotation of example 1

Example 1 is now rotated 45° and calculated in the same manner as example 1. In previous versions of Calsef this has produced strange results. All properties (material, geometry and elements) are the same as in Example 1. Naturally the analytical result is the same as before, which ought to be the case for the FEM solution as well.

4.3.3.1 The Calsef 2001 solutions for Example 2

The results from Calsef 2001 (See Appendix B) for each type of element correspond to the analytical solution. Since the result is displayed in the global coordinate system a transformation is needed. Pythagora's thesis and the circle of Mohr [12] give the result below. As the results are quite accurate, independently of what kind of element is used, only the 3 node triangular element is calculated. The 3 other element types results in the same conclusion.

$$\delta_4 = \delta_3 = \sqrt{\delta_{3x}^2 + \delta_{3y}^2} = \sqrt{(-0.7071450 \cdot 10^{-2})^2 + (-0.7070850 \cdot 10^{-2})^2} = 1.00001162 \cdot 10^{-2} \\ \approx 1.00 \cdot 10^{-2} m$$

$$\delta_2 = \sqrt{\delta_{2x}^2 + \delta_{2y}^2} = \sqrt{(-0.5000035 \cdot 10^{-7})^2 + (-0.5000035 \cdot 10^{-7})^2} = 7.071117 \cdot 10^{-8} \approx 0m$$

$$\sigma_{principalstress1,2} = \frac{1}{2} \left((\sigma_y + \sigma_x) \pm \sqrt{(\sigma_y - \sigma_x)^2 + 4 \cdot \tau_{xy}^2} \right)$$

$$\sigma_x = -0.5000298 \cdot 10^4 Pa$$

$$\sigma_y = -0.4999823 \cdot 10^4 Pa$$

$$\tau_{xy} = -0.5000011 \cdot 10^4 Pa$$

$$\Rightarrow \sigma_{principalstress1} = -1.000007150564061 \cdot 10^4 \approx -1 \cdot 10^4 Pa$$

Although the result is satisfying, one can notice the approximate manner of FEM. The error is small enough to be considered as neglectable in this example. When more complex structures are modeled the error can increase and the error might not be ignorable.

The Calsef result files only include 7 decimals, which provide an additional round-off error to the model. As GiD's postprocessor is based on these result files, GiD will show the same error.

4.3.4 Example 3 - Cube with elastic supports and point load

Cube with elastic supports and a point load at the upper right corner, according to the left part of figure 4.7. To the right the analytical resultants are shown. Elastic constraints is a new feature of Calsef, why this simple test is needed. The only things that have been changed from the first example are the supports and the load. The figure to the right shows how the displacements are, according to the analytical solution. No self-weight is considered. The cube is considered as stiff, only allowing rotation without deformation.

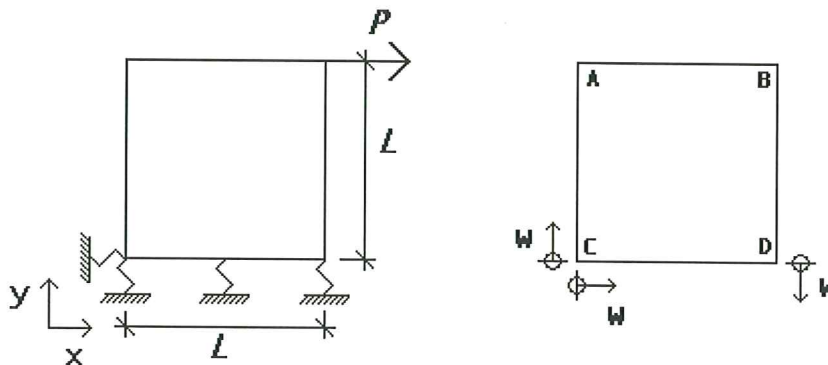


Figure 4.7 Cube with elastic supports

$$P = 10N$$

$$K = 1000 \frac{N}{m}$$

$$L = 1m$$

$$thickness = 1m$$

The analytical solution of this example is:

$$w = \frac{P}{K} = \frac{10}{1000} = 0.01m$$

The results coincide with the analytical solution for all types of elements, see appendix B.

4.3.5 Conclusions from validation

The results are satisfying for these examples and the beta version seems to be correct. More examples need to be studied to get a better picture of the accuracy of the program Calsef 2001, but as validation of the calculator used for the program AMG v1.0 these examples are enough. The beta testing might seem rather basic. Nevertheless some really severe problems have been detected and corrected. These have concerned the use of elastic constraints and uniform load on more complex problems. The calculation module Calsef is now according to the validation all right and the development of the AMG v1.0 can begin.

5 Development of an adaptive mesh generator

5.1 Work procedure

In order to calculate the error energy norm from the Calsef solution several approaches were possible. As Calsef is written in Fortran the first idea was to use Fortran to create an algorithm that could be incorporated in Calsef. Another approach was to create an independent program, working outside the GiD environment. To do this the calculated values from the solver had to be read and implemented in the program. Familiarity of the programming language MATLAB[®] and the open structure of that code were the main reasons why the program was composed in MATLAB[®]. Since the code does not require to be compiled before being executed it is easy to modify the program for different needs. As Calsef writes the files that are required for GiD in ASCII format, these files could be read by MATLAB[®] as well. The program is based on these values and is called AMG v1.0.

Since Calsef does not provide all necessary information in the results files (e.g. shape functions, the Jacobian matrix and the values in the Gauss points) this had to be calculated again. This wouldn't have been necessary if the program would have been connected with Calsef. This caused more programming and the program also became a little more time consuming. On the other hand a modification of the Calsef program was not necessary and the work could be concentrated on AMG v1.0.

A few modifications were done in the input files (*.bas) in order to read the output files (*.cal) and calculate the error. The program is designed for problems in plane state (plane stress or plane strain), using three-node triangular linear elastic elements. The basic structure can be studied on the next page.

Some functions from the program Calfem [13] created at Lund Institute of Technology have been used to create AMG v1.0.

5.2 AMG v1.0

The structure of the program can be described by the 12 steps below (for the complete code with user comments see Appendix A).

1. The GUI is invoked and the file path, relative global error and type of strategy are read
2. Geometry, number of elements and nodes, analysis type, constitutive matrix, nodal displacements and smoothed stress values are read from the Calsef files described in section 4.2
3. The Jacobian is calculated for each element in order to determine the error energy norm.
4. The error energy and strain energy norms are calculated for each element
5. The sum of the square error and strain energy norms are calculated
6. Global error ξ_g and local error $\bar{\xi}_i$ are calculated
7. The element refinement parameter ξ_i is calculated for each element. If ξ_i is too big point 8-11 is done, otherwise point 12
8. New element sizes are calculated
9. A Background file readable by GiD is created (see section 4.1.1.4)
10. GiD is started and a batch file is started within the program. The batch file is written so that GiD reads the background file and remeshes the structure. After that Calsef starts to calculate a new solution
11. Back to point 2
12. GiD is started and the final mesh is read and shown

Below a schematic figure shows the process of the different programs used in the calculations. To be able to run the MATLAB[®] program the problem must have been calculated once, in order for the output files to be created. The process after that is automatic and continues until the criteria is fulfilled and a satisfying mesh is established.

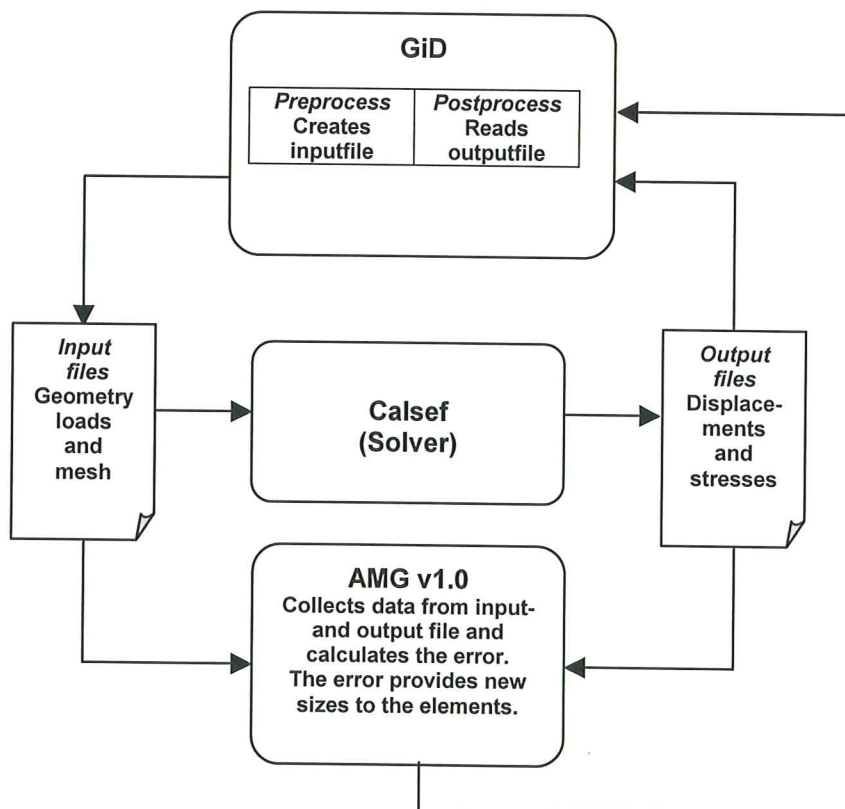


Figure 5.1 The AMG environment

5.2.1 The program files of AMG v1.0

AMG v1.0 consist of the following functions and programs:

- *AMG* (Main program)
- *Usermenu* (GUI, see figure 5.2)
- *Usermenu* (GUI-controller)
- *Connect_menu* (Reads input values)
- *Writebat* (Writes two batch files)
- *Read_gid* (Reads GiD-files)
- *Coordxtr* (Collects global coordinates into element coordinates)
- *Jacobtriangle* (Calculates the B-matrix and the Jacobian determinant)
- *Backgroundsize* (Calculates new sizes for the elements)
- *Write_mesh* (Writes the background file with the new sizes)

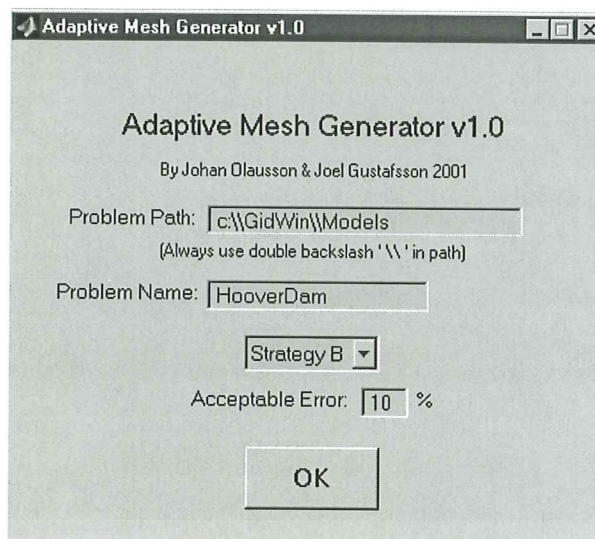


Figure 5.2 The GUI of AMG v1.0

These files make up the complete program, with a total size of about 44kB.

5.2.2 Modifications to be made in order to run AMG with GiD

Some modifications must be made in a Calsef input file to be able to run AMG together with GiD. This must be done in order to make GiD provide the parameters needed to calculate the error. The file modified is under the directory:

Gidwin\problemtypes\calsef_PlaneState_V1.0.gid

, which is created when GiD is installed. The following additions must be made under this directory in the following file:

calsef_PlaneState_V1.0.bas (This is a text based *bas file)

At the end at this file the following text must be added:

```

Number of Elements & Nodes:
Identifier
*nelem *npoin
*loop materials
*MatProp(1) *MatProp(2) *Matprop(3) *Matprop(4)
*end materials
*if( (strcmp(GenData(Problem_Type), "Plane-Stress")==0) )
    1
*else
    2
*endif
*if(nelem(Triangle)>0)
    0
*else

```



```

1
*endif
*if(IsQuadratic)
2
*else
1
*endif

```

After this modification is made it is possible to use AMG along with GiD.

5.3 Step-by-step execution of AMG v1.0 on a simple example

To run the program some things need to be considered, a simple example is therefore shown. The following example is modeled in GiD with a global element size of 0.1, which gives 153 elements. Plane stress is assumed. The limit of the element refinement parameter is in the program chosen to $\xi_i=2.3$.

Boundary condition: CD and EF is locked in x- and y-direction

Applied Load: A uniformed pressure load of 1000kN/m is applied on GH

Material: Isotropic and linear elastic material, Concrete, $E=30$ GPa, $\nu=0.2$, thickness=0.2m

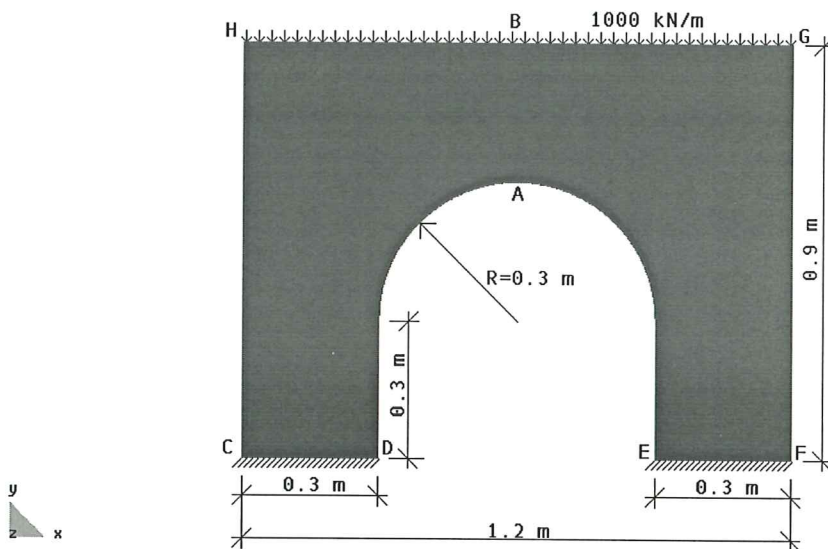
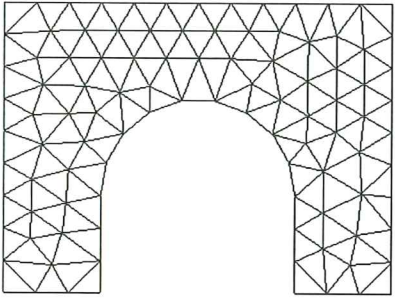
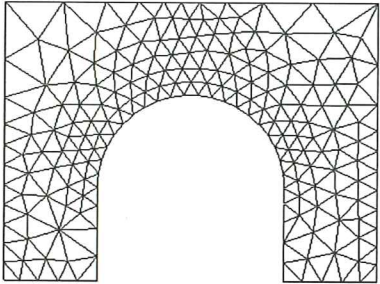
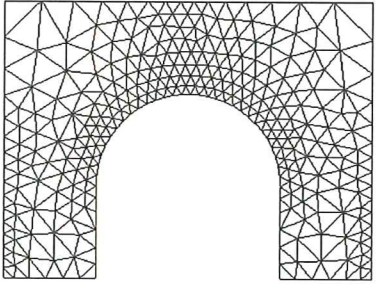


Figure 5.3 Semi-circular arc

5.3.1 Step-by-step explanation

- Model and calculate the example in GiD
- Close GiD in order to free more CPU and to minimize the amount of windows.
- Start AMG v.1.0 in MATLAB[®] by writing “AMG” on the MATLAB[®] prompt.
- A window opens where the Path, Problem name, Strategy and error must be chosen. In this case: Strategy B and 10% error
- The calculation starts and afterwards a plot with the refinement parameter are shown and GiD is executed.
- GiD remeshes the problem to 338 elements and starts calculating.
- When the calculation is finished save the project and close GiD.
- Press any button in the MATLAB[®] prompt and the calculation starts again.
- The plot with the new refinement parameter is shown and GiD is executed.
- GiD remeshes the problem to 535 elements and starts calculating.
- When the calculation is finished save the project and close GiD.
- Press any button in the MATLAB[®] prompt and the calculation starts again.
- The element refinement parameter $\xi_i < 2.3$ and therefore only this plot is shown.
- Press any button in the MATLAB[®] prompt GiD is executed.
- GiD now shows the final mesh with 535 elements and the program AMG v1.0 is finished.
- Study the example in GiD as normal but with a more accurate mesh.

The three iterations give the following result:

Mesh	NE	ξ_{R}	$\sigma_{x,A}$ [MPa]	$\sigma_{x,B}$ [MPa]
	153	1.6705	1.74	-6.4
	338	1.2012	5.92	-8.02
	535	0.9939	7.17	-8.54

In each iteration it is possible to use GiD as normal to study results as long as the problem is saved before any button is pressed in the MATLAB[®] prompt. It is not necessary to close GiD every time but if there are many iterations a loss in computer power caused by all the open windows will slow down the calculations.

5.4 Computational speed

AMG has been developed in MATLAB[®], which is a high-level matrix handling computer language. Because of this, the program is slower than similar applications. A simple test of the speed of AMG v1.0 has been made, using meshes containing between 8 and 7066 elements. For all meshes the CPU has been clocked with the internal MATLAB[®]-command *cputime*, using an ordinary PC equipped with a 350 MHz Pentium II[™] processor.

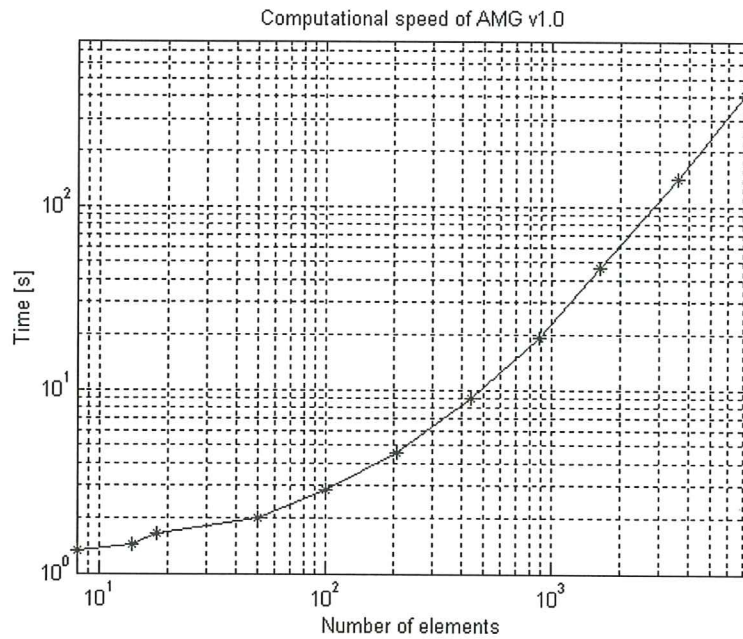


Figure 5.4 Computational speed

The time for treating 7066 elements is about 7.5 minutes, which can be considered as quite much. In order to increase the time efficiency one would need to either optimize the code, or rewrite the program in a low-level computer language such as Fortran or C.

6 Examples for validation of AMG v1.0

In order to validate the program AMG v1.0 and the two strategies B and C a few examples are studied. Some of the examples have analytical solutions, which allows for an easy comparison with the results obtained. The number and structure of the elements in the examples compares the effectiveness of the strategies B and C. Some of the examples have been found in books, others are familiar from practical structural engineering. After these examples a good validation can be made of AMG v1.0.

6.1 Simply supported beam

The example is an analysis of a simply supported beam as shown in figure 6.1. Plane stress is assumed. The problem is very common in engineering and therefore useful to study.

Geometry: The beam has a height of 0.5 m, a width of 0.3 m and the span is 10 m

Boundary condition: The left support is fixed in x- and y-direction and the right support is fixed in y-direction

Applied Load: A uniform line load of 1 kN/m along the beam

Material: Isotropic and linear elastic material, $E=30$ GPa, $\nu=0.2$

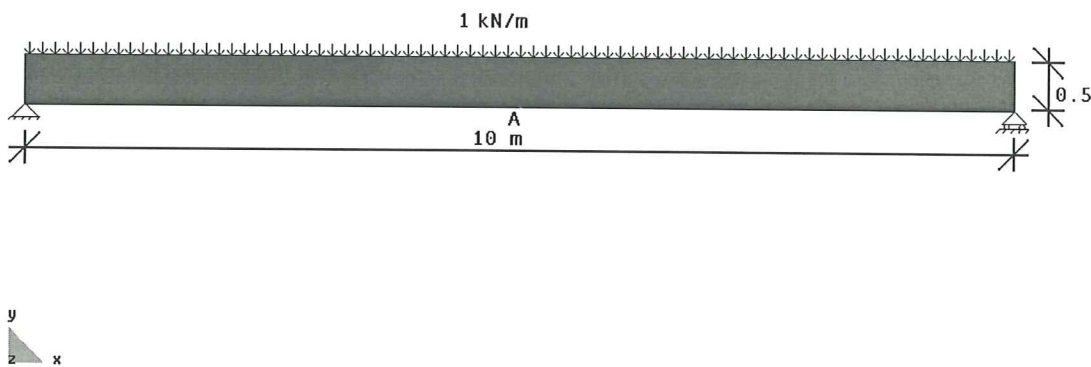


Figure 6.1 Simply supported beam

The beam can be calculated according to the beam theory [12]. The stresses and the displacement in the middle of beam at point A will be:

$$\sigma_{x,\text{bottompart}} = \frac{M_x}{W_x} = \frac{\frac{q \cdot L^2}{8}}{\frac{w \cdot h^2}{6}} = \frac{1 \cdot 10^3 \cdot 10^2}{\frac{0.3 \cdot 0.5^2}{6}} = 1 \text{ MPa} = |\sigma_{x,\text{upperpart}}|$$

$$\delta_{\text{middle}} = \frac{5 \cdot q \cdot L^4}{384 \cdot E \cdot I} = \frac{5 \cdot 1 \cdot 10^3 \cdot 10^4}{384 \cdot 30 \cdot 10^9 \cdot \frac{w \cdot h^3}{12}} = 0.0013899 \text{ m} \approx 1.390 \text{ mm}$$

6.1.1 Results







The stresses will have same value on the upper part of the beam, with opposite signs. AMG v1.0 is set to the permissible error $\eta_g=10\%$, and after only 3 iteration with the 2 methods a good result is achieved.

Strategy B (AMG v1.0)					Strategy C (AMG v1.0)				
NE	ξ_g	$\sigma_{x,u}$ (MPa)	$\sigma_{x,b}$ (MPa)	δ (mm)	NE	ξ_g	$\sigma_{x,u}$ (MPa)	$\sigma_{x,b}$ (MPa)	δ (mm)
50	6.95	-0.2900	0.2936	0.421	50	6.95	-0.2900	0.2936	0.421
1792	1.983	-0.9174	0.9211	1.350	1950	1.94	-0.9772	0.9506	1.353
6290	1.059	-0.9623	0.9561	1.385	6923	1.092	-0.9720	0.9676	1.384

This gives the following error in each iteration.

Iteration	Strategy B		Strategy C	
	Stress Error (%)	Disp. Error (%)	Stress Error (%)	Disp. Error (%)
0	71	67	71	67
1	8.3	2.8	4.9	2.6
2	4.4	0.29	3.2	0.36

The result is satisfying after one iteration. The errors in the two methods are almost identical, so in this example a comparison between the errors for the strategies can be ignored. Strategy B has fewer elements so this strategy can be considered as slightly more effective for this example. Also, strategy B does not concentrate the elements as much as strategy C does. This can avoid convergence problems, which occurs in problems with large stress concentrations.

Strategy B	Strategy C
 50 elements	 50 elements
 1792 elements	 1950 elements
 6290 elements	 6923 elements

The meshes in the different methods are as expected and further comments on them are not necessary. For comparison, this example is also calculated without AMG v1.0, using 4 different uniform meshes produced by GiD.

NE	$\sigma_{x,u}$ (MPa)	$\sigma_{x,b}$ (MPa)	δ (mm)
2278	-0.8933	0.8904	1.3544
7002	-0.9439	0.9494	1.3841
12452	-0.9974	0.9938	1.3919
28846	-0.9920	0.9747	1.3963

As an example the mesh with 2278 elements can be seen in fig 6.2.



Figure 6.2 Structured mesh of simple supported beam

A comparison with the meshes made with help of AMG v1.0 and the one without shows that the stresses are more accurate in the first case, whereas the displacement are almost the same. In this example the elements are evenly distributed except on the ends of the beam. This is due to that the error is quite uniformly spread throughout the beam. The example still verifies that the mesh provided by AMG v1.0 is more accurate than the uniform mesh provided by GiD.

6.2 Circular plate with normal pressure

Comparison with an analytical solution from NAFEMS [14] is made in order to test AMG v1.0. The example consists of a 1m thick plate with normal pressure from the outside. Only one fourth of the pipe needs to be modeled, due to symmetry. The properties of the structure is as follows:

Geometry: Inner radius: 10m, outer radius: 11m

Boundary condition: AB is fixed in x-direction and CD is fixed in y-direction

Applied Load: A uniform load of 100MN/m is applied on BC

Material: Isotropic and linear elastic material, $E=210.103\text{MPa}$, $\nu=0.3$, thickness=1m

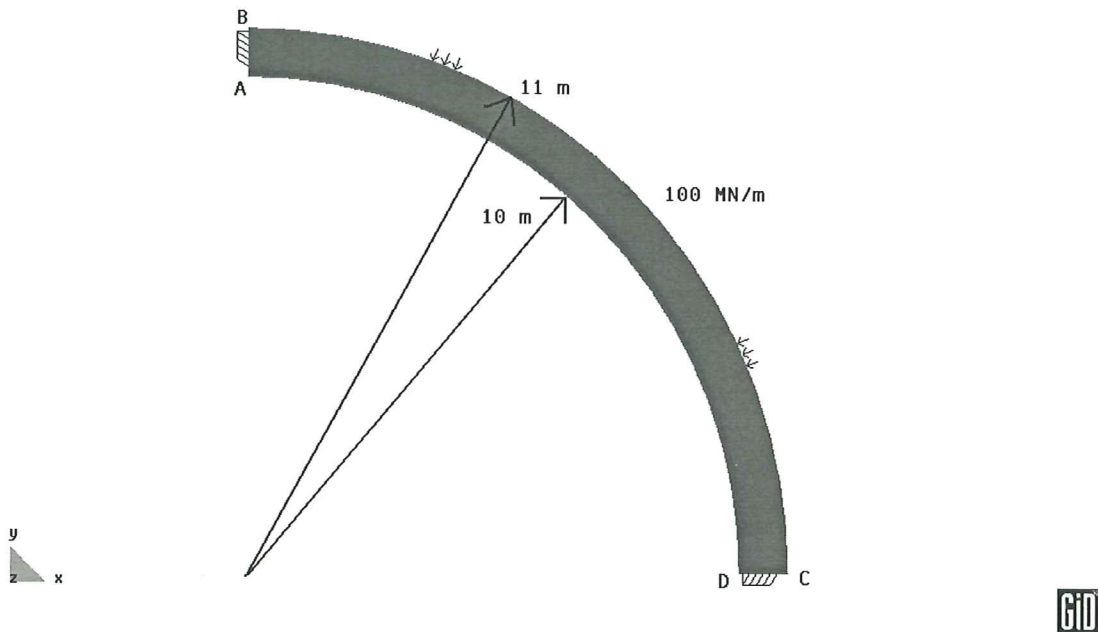


Figure 6.3 Thick circular plate

According to NAFEMS this problem has the analytical result:

The stresses in y-direction at point D is $\sigma_y = -1150\text{MPa}$

AMG v1.0 is set to run until the element refinement parameter $\xi_i < 1.50$. This is to avoid oscillation and to decrease the time of the calculations. The error limit is 2% in this example.

6.2.1 Results

The result of the iterations:

Strategy B			Strategy C		
NE	ξ_g	σ_y (point D)	NE	ξ_g	σ_y (point D)
34	3.2926	-1095.89	34	3.2926	-1095.89
324	1.2213	-1156.77	311	1.2249	-1156.90
508	0.9781	-1154.37	473	0.9938	-1152.03
Total error at point D (%) = 0.38%			Total error at point D (%) = 0.18%		

As a comparison, an unstructured mesh with 518 elements give the result $\sigma_y = -1156.7\text{MPa}$ at point D. This gives an error of 0.58%. The result is very close to the analytical solution, which is very satisfying and also shows that the mesh created by the AMG v1.0 is better than a normal unstructured mesh created by GiD. The error of 2% was in this example very easy to fulfill. The load is uniformly distributed in this example, which decreases the chance of stress concentrations, which contributes to a very large amount of small elements. On the next page the element distribution can be studied for each iteration and strategy.

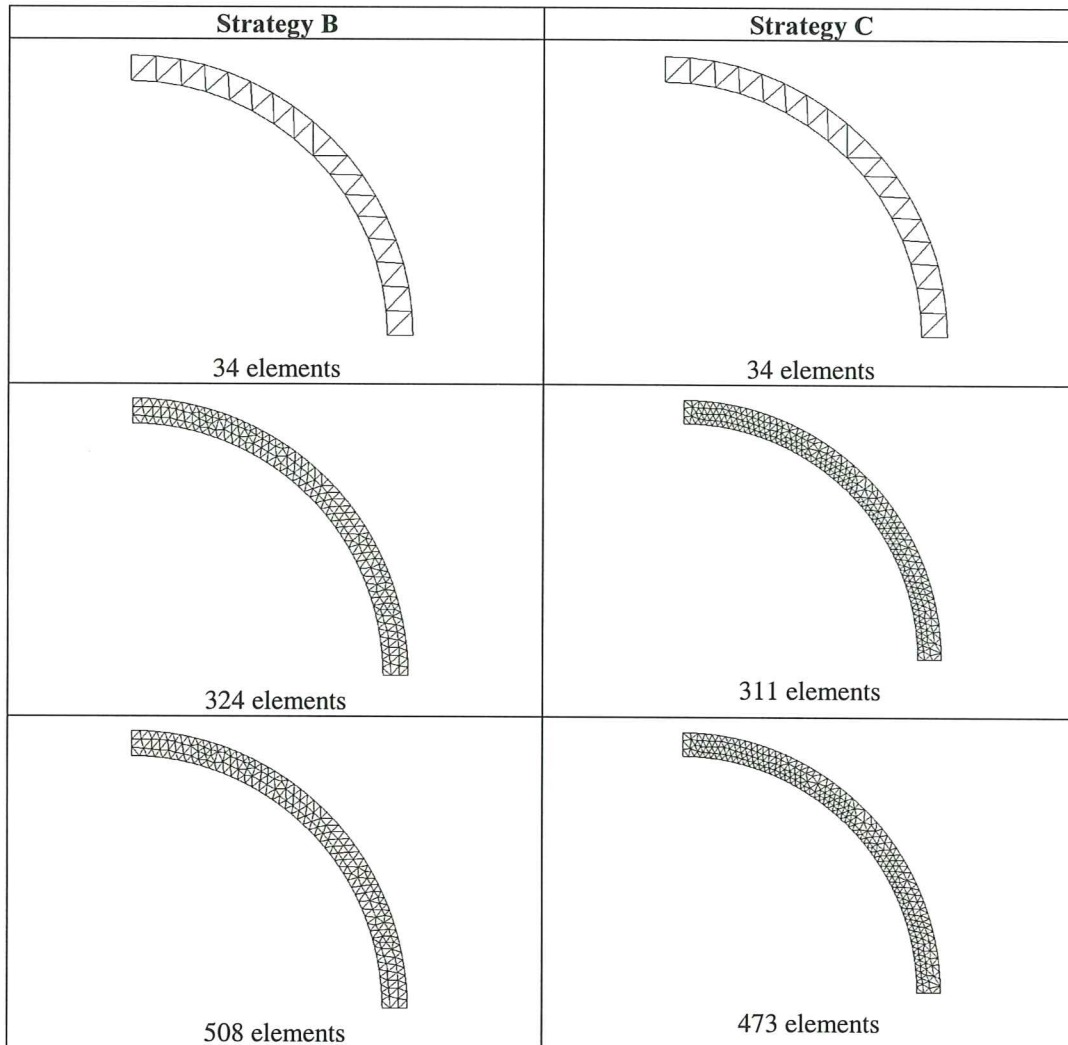


Figure 6.4 Meshes of thick circular plate

As the element distribution is very similar in both strategies not many comments to the figures is necessary. One thing to point out is that the elements in strategy C are more concentrated at the inner line (AD) and that some larger elements exist in the middle. Strategy C creates fewer elements than B, which is unusual.

The distortion is low for both strategies and is therefore not plotted.

6.3 Thick circular cylinder under internal pressure

The example is an analysis of a thick circular cylinder under internal pressure as shown in figure 6.5. Due to symmetry only one fourth of the cylinder is studied. Plane strain is assumed. The permissible error is set to $\eta=5\%$.

Geometry: Inner radius: 5m, outer radius: 20m

Boundary condition: AB is fixed in x-direction and CD is fixed in y-direction

Applied Load: A uniform pressure load of 1N/m is applied on AD

Material: Isotropic and linear elastic material, $E=1$ MPa, $\nu=0.3$, thickness=1m

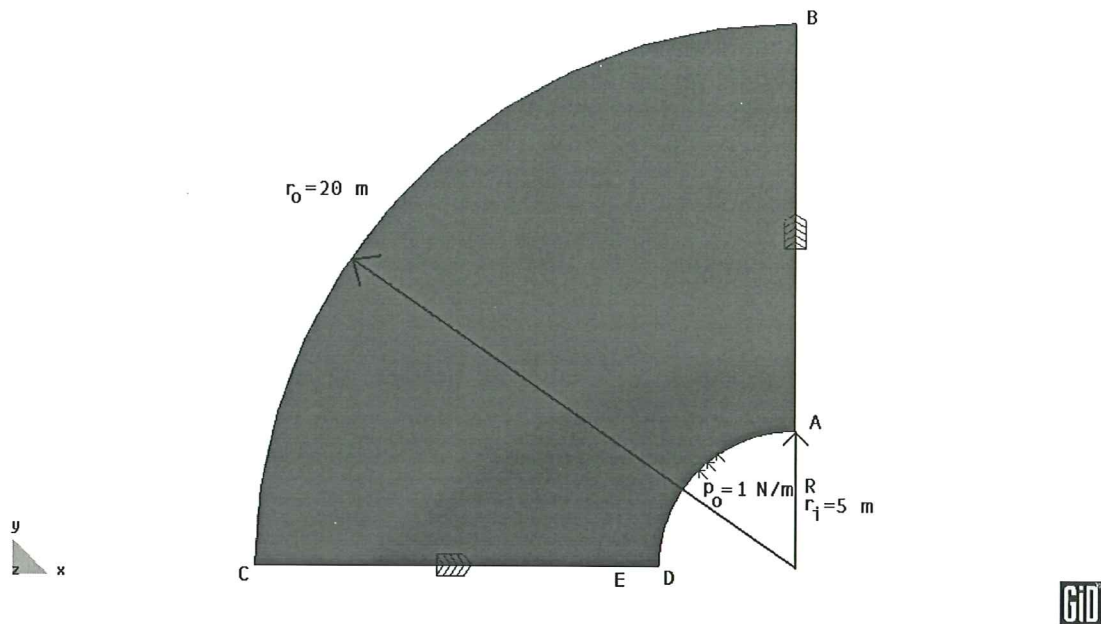


Figure 6.5 Thick cylinder under internal pressure

6.3.1 Study of the mesh

After the iteration the following result can be studied. The different meshes can be seen on the below.

Strategy B (AMG v1.0)		Strategy C (AMG v1.0)	
NE	ξ_g	NE	ξ_g
199	4.994	199	4.994
2207	0.882	2544	1.0377
1755	0.977	4975	0.9254
1748	0.976	5719	0.9265

It can be clearly seen that the number of elements in strategy C is larger than in strategy B. In this case the additional elements do not provide a significantly better solution. After one iteration Strategy B fulfils the global error criterion. Both methods correctly increase the number of elements against the internal boundary, since the error is largest there. This is due to higher stress gradients in that area. Strategy B is in this example a better method because of less number of elements produced, and the fact that the criterion is still fulfilled. The smaller number of elements saves a lot of computer time and could in more complex models be very important.

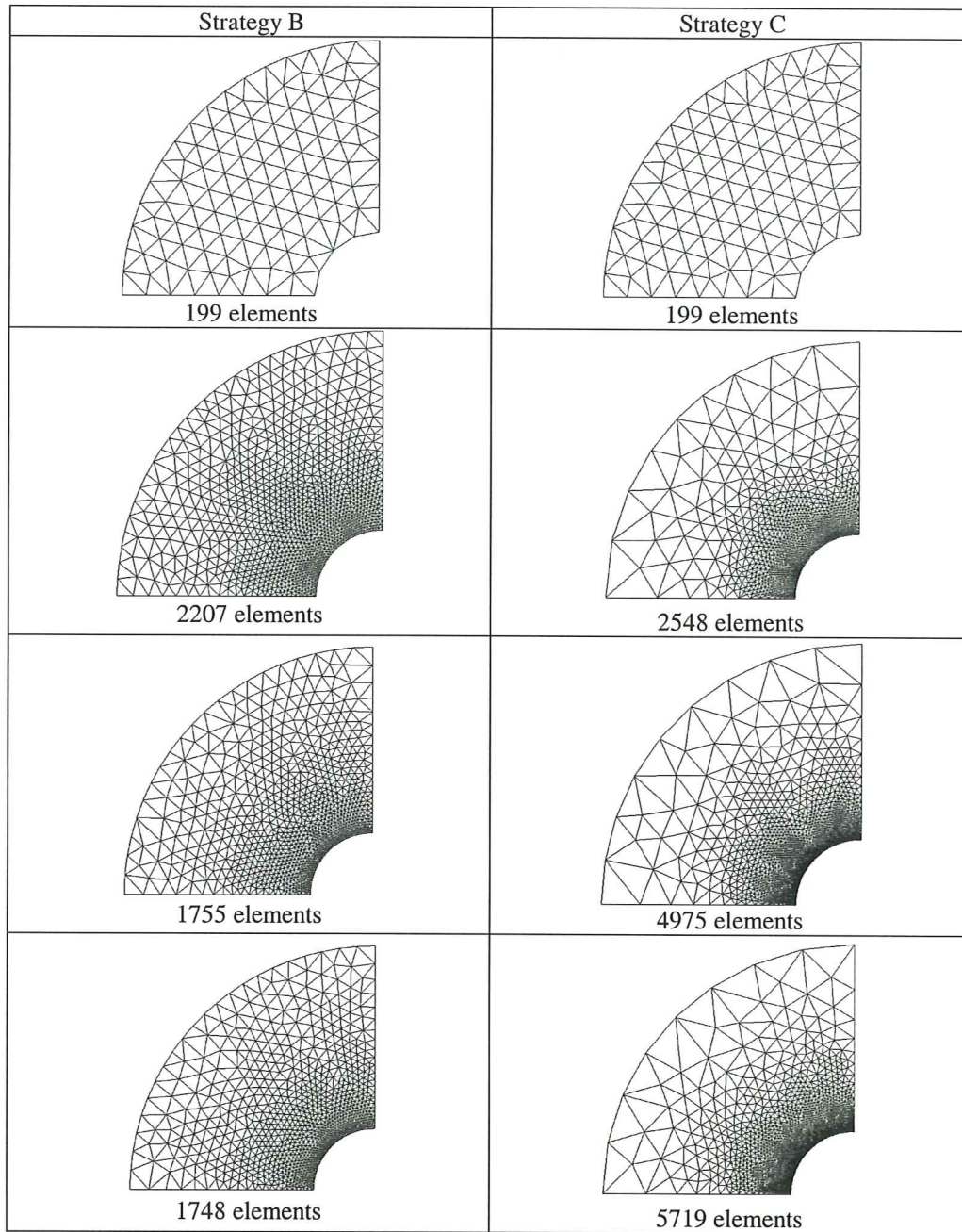


Figure 6.6 Meshes of thick cylinder

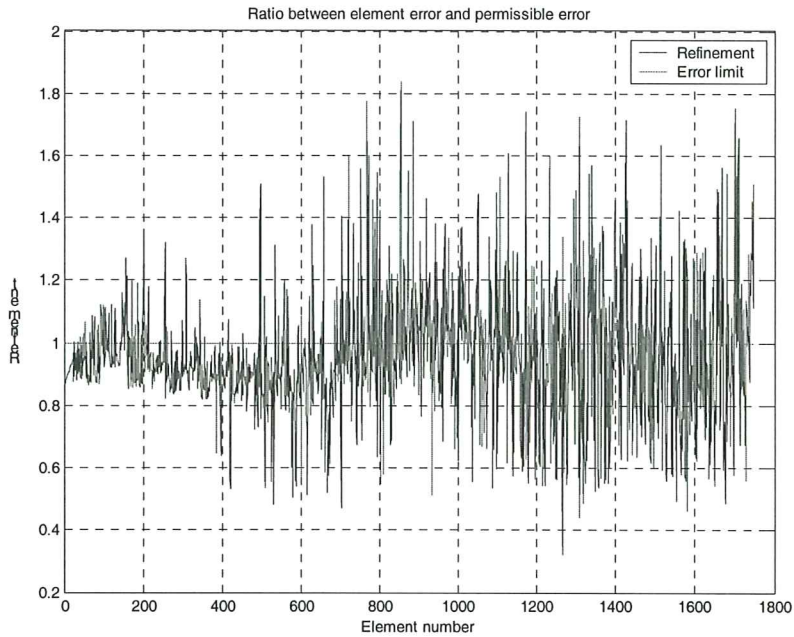


Figure 6.7 Strategy B; Ratio between element error and permissible error

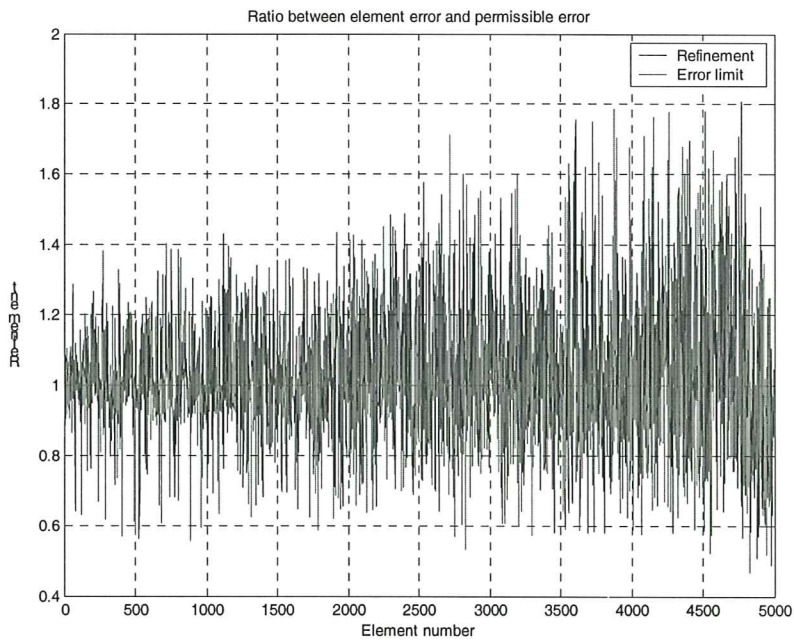


Figure 6.8 Strategy C; Ratio between element error and permissible

The figures above show the element refinement perimeter, which for an optimal mesh should be 1.0 in each element. In strategy B the curve is not as spread as in strategy C. This is probably because strategy B converges against 1750 elements while strategy C needs more iteration to converge and therefore the graph is more spread. This also shows that strategy B is a better method for this example.

6.3.2 Results

Comparison with the analytical solution of a cylinder under internal pressure. The analytical solution is according to [15]:

$$\sigma_{\tan} = \frac{r_i^2 \cdot p_i}{r_o^2 - r_i^2} \left(1 + \frac{r_o^2}{r^2} \right)$$

$$\sigma_{\text{radial}} = \frac{r_i^2 \cdot p_i}{r_o^2 - r_i^2} \left(1 - \frac{r_o^2}{r^2} \right)$$

r_o = outer radius

r_i = inner radius

p_i = inner pressure

r = point of calculation

Two points are chosen to compare with the analytical solution. Point D is the origo in figure 6.5. These points are:

- $(x,y) = (20,5)$ Point A in figure 6.5
- $(x,y) = (14,0)$ Point E in figure 6.5

The analytical values in these points are with the above formulas:

$$\sigma_{\tan A} = 1.133\text{MPa}$$

$$\sigma_{\text{radial}A} = -1.000\text{MPa}$$

$$\sigma_{\tan E} = 0.8074074074\text{MPa}$$

$$\sigma_{\text{radial}E} = -0.6740740741\text{MPa}$$

The different meshes used gives in these points the following stresses.

Strategy B (AMG v1.0)				
NE	$\sigma_{\tan A}$	$\sigma_{\text{radial}A}$	$\sigma_{\tan E}$	$\sigma_{\text{radial}E}$
199	0.835	-0.531	0.692	-0.540
2207	0.1154	-0.9520	0.8271	-0.6840
1755	0.1151	-0.9555	0.8187	-0.6789
1748	0.1146	-0.9509	0.7953	-0.6573
Strategy C (AMG v1.0)				
NE	$\sigma_{\tan A}$	$\sigma_{\text{radial}A}$	$\sigma_{\tan E}$	$\sigma_{\text{radial}E}$
199	0.835	-0.531	0.692	-0.540
2544	0.1146	-0.9662	0.7994	0.6680
4975	0.1137	-0.9854	0.7980	-0.6669
5719	0.1136	-0.9842	0.8167	-0.6837

After one iteration the stress values are quite close to the analytical values. $\sigma_{\text{radial}A}$ carries the largest error, which is probably due to the stresses concentrations in this direction. The B strategy gives with approximately 1800 elements an error of 5%, which is good. The C strategy produces a smaller error but a larger number of elements.

To get a better look on the results the tangential- and the radial- stresses are calculated both for the FEM- and the analytical solution. This is made in MATLAB[®] and the x- and y- components from the FEM solution is translated into tangential and radial stress components. This is made in every node point then collected in increasing radius order and then plotted and compared in the graphs below. The graphs are plotted for the C strategy after one iteration, which means 2544 elements. Some additional error occur when the transformation between coordinate system is made and that must be taken into account when studying the graphs.

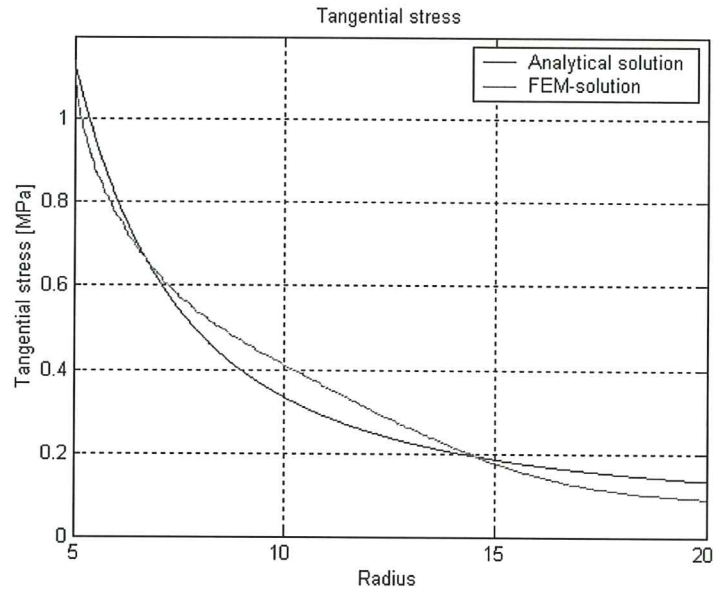


Figure 6.9 Tangential stress

For the tangential stresses the two curves are quite similar and the result is satisfying. One strange thing is that the error is not largest when the radius = 5m, which would have been logical because that is where the number of elements are increasing most during the second iteration.

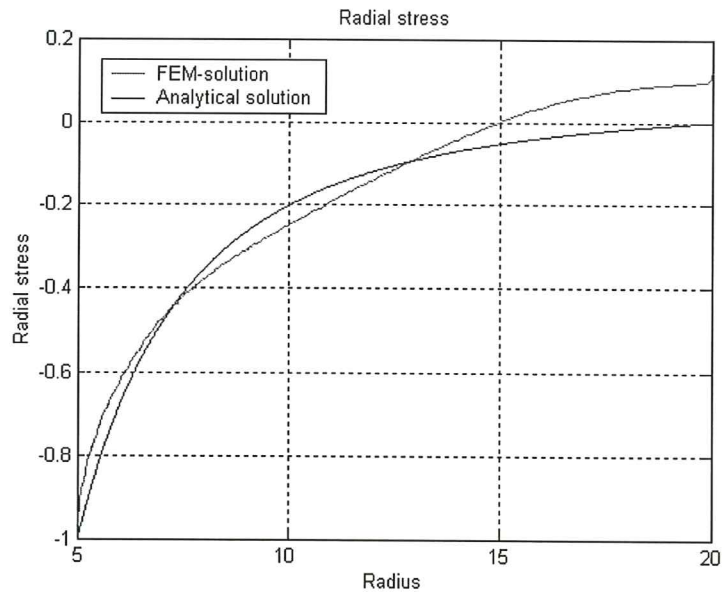


Figure 6.10 Radial stress

The radial stresses have rather large error when the radius is between 15 – 20 meters otherwise the curves are similar.

6.3.3 Mesh quality

The mesh quality is measured according to section 2.4.2. The graphs are based on the mesh achieved after one iteration of AMG. Although the mesh quality of both strategies are very good, strategy C produces some high values of distortion. This is probably due to the larger difference in element size that strategy C creates. In total the quality is very good and gets even better after even more iterations.

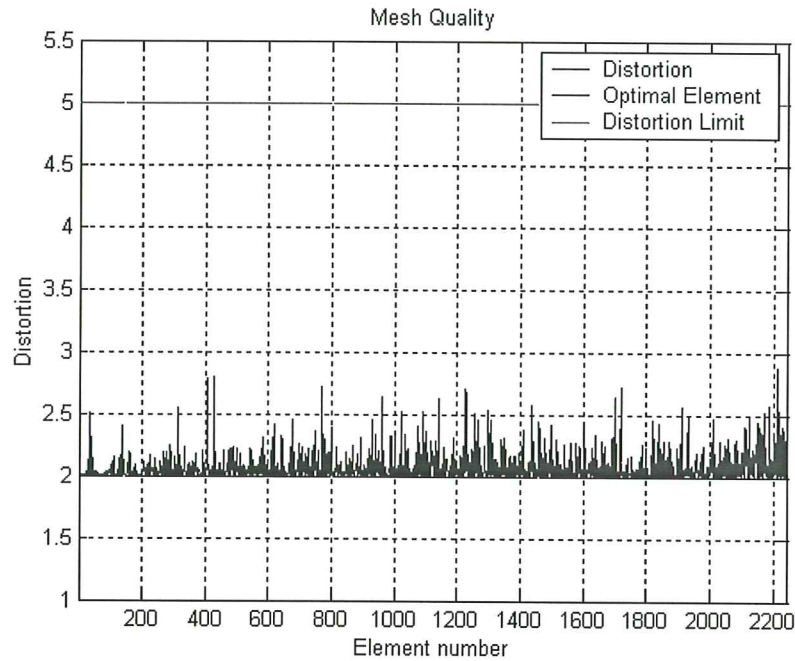


Figure 6.11 Mesh quality, Strategy B, 2207 elements

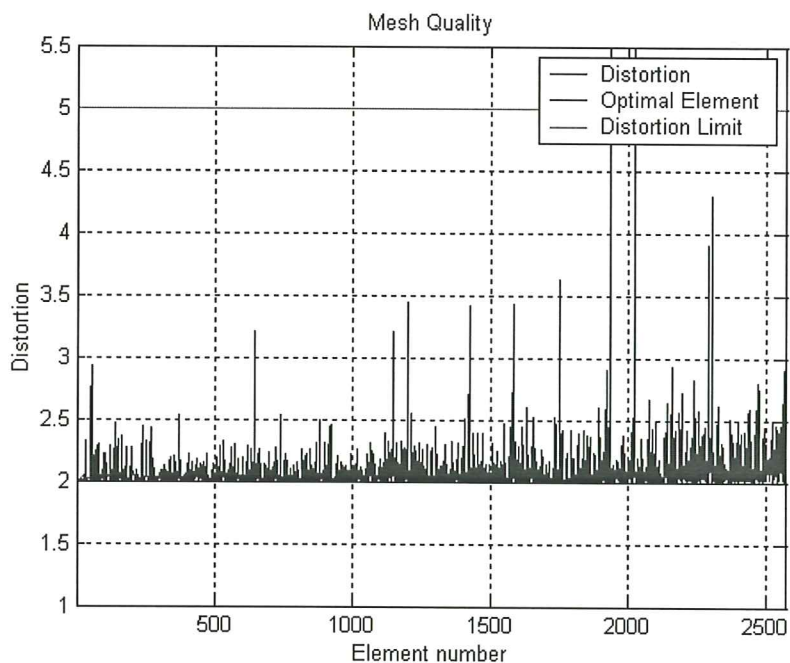


Figure 6.12 Mesh quality, Strategy C, 2544 elements

6.4 Stress concentration in isotropic plates

Stress concentration factors in finite plates have been compared with theoretical elasticity solutions. In both cases only one quarter of the plate is modeled and boundary conditions are applied as shown in figure 6.13 and 6.15. Plane stress and ordinary triangular elements are used for the simulations.

The initial mesh is created in GiD, using an element size of 0.2 over the entire area. Nevertheless the size varies quite substantially over the geometry, which can be seen in the tables below. AMG v1.0 is then executed using a relative global error of $\eta=3\%$. In order to avoid getting an extremely coarse mesh the maximum element size is set to 0.2, i.e. the initial mesh will only be subject to refining.

6.4.1 Semicircular notch

The first example is a rectangular plate with a semicircular notch (with radius r) in the middle of each of the longer sides (a). The shorter sides (b) carry a distributed load. For the aspect ratio $a/b=4$ and $b/r=10$, the theoretical elasticity solutions for the stress concentration around the notch is 3.018 [15].

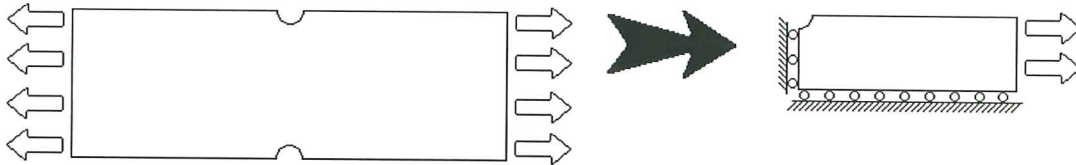


Figure 6.13 Finite plate with semicircular notch

Strategy B:

Iteration	Stress concentration factor	Error	Elements	Max size	Min size
0	1.5246	49.5	70	0.2709	0.0921
1	2.7694	8.24	143	0.2539	0.0264
2	2.8886	4.29	197	0.2618	0.0157

Strategy C:

Iteration	Stress concentration factor	Error	Elements	Max size	Min size
0	1.5246	49.5	70	0.2709	0.0921
1	2.8675	4.99	341	0.2344	0.0079
2	2.9989	0.63	2339	0.2334	0.0013

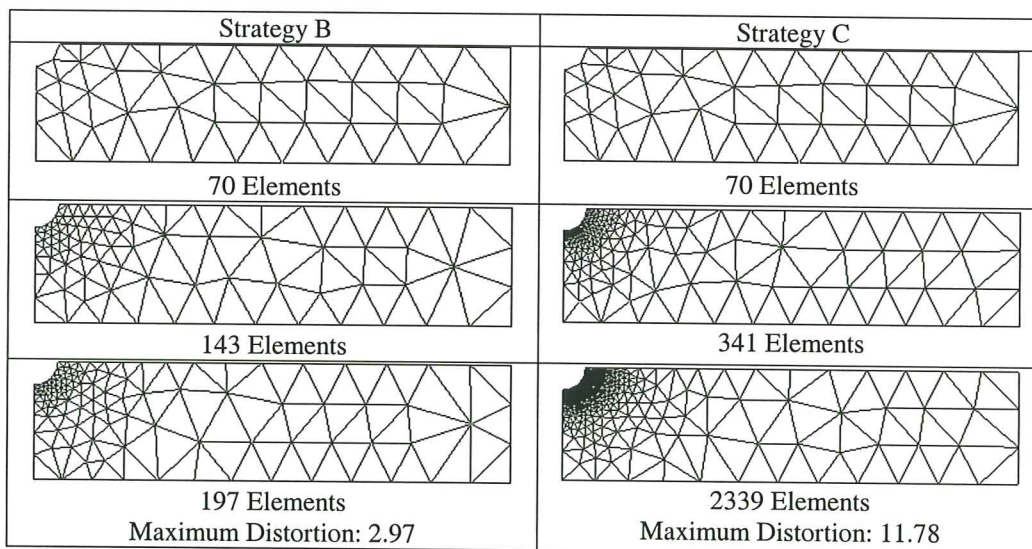


Figure 6.14 Meshes for semicircular notch

6.3.2 Central hole

Here the notch is replaced by a single, central hole. The same aspect ratios yield a stress concentration factor of 3.133 [15].

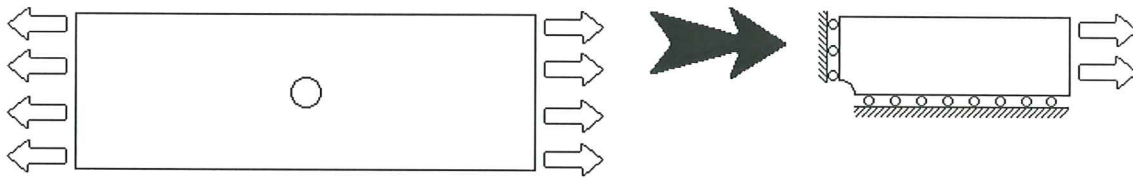


Figure 6.15 Finite plate with central hole

Strategy B:

Iteration	Stress concentration factor	Error	Elements	Max size	Min size
0	1.572	49.8	71	0.2637	0.0854
1	2.7953	10.8	136	0.2687	0.0267
2	2.954	5.71	212	0.2538	0.0142

Strategy C:

Iteration	Stress concentration factor	Error	Elements	Max size	Min size
0	1.572	49.8	71	0.2637	0.0854
1	2.9778	4.95	398	0.2517	0.0069
2	3.1072	0.82	3226	0.2611	0.0011

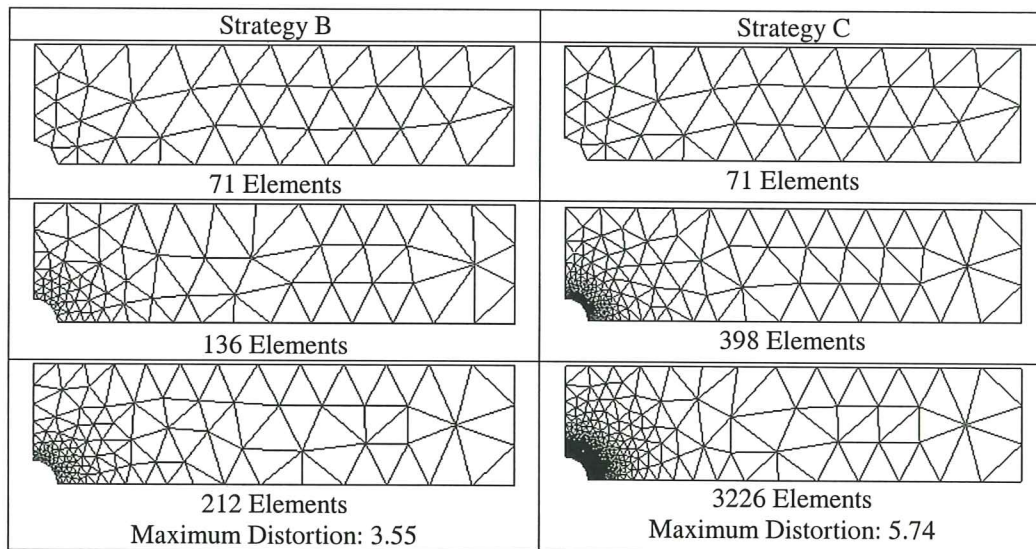


Figure 6.16 Meshes of circular hole

6.4.3 Efficiency of the adaptive mesh refinement

For these examples reasonable solutions are achieved, for both strategies, after only two iterations. Strategy C yields a significantly higher convergence rate, at the expense of a larger amount of elements. In order to avoid these kinds of large meshes, one could introduce a minimum element size limit along with the maximum one, as proposed by [7]

6.4.4 Quality of the final meshes

The maximum D-values (see section 2.4.2) for the elements are low in three of the four final meshes, but for the mesh produced using the C-method on example 1, $D_{\max}=11.78$. Though the mean value is still well below 5, the distortion peaks in some areas, see figure 6.17.

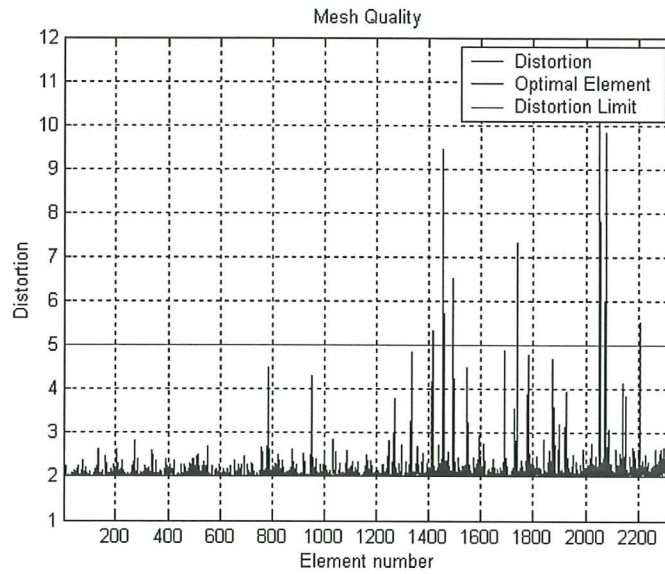


Figure 6.17 Mesh quality of semicircular notch, using strategy C

Note that GiD is the actual mesh generator, whereas AMG only calculates the element sizes (i.e. mean side length). For this reason, the possible implications of the locally poor mesh quality have not been investigated.

6.5 Beam with a hole

A 5 m long concrete beam with a hole for ventilation is studied. Stress concentrations around the hole and under the supports are expected. In order to decrease the amounts of elements due to the concentrations a minimum size must be chosen.

Geometry: 5m long with a height of 0.6m. The hole is 0.2x0.4m placed 0.6 meters from the left support.

Boundary condition: Both supports are fixed in both x- and y-direction.

Applied Load: A uniform load of 20kN/m is applied on the beam

Material: Isotropic and linear elastic material, $E=30$ GPa, $\nu=0.2$, thickness=0.2m

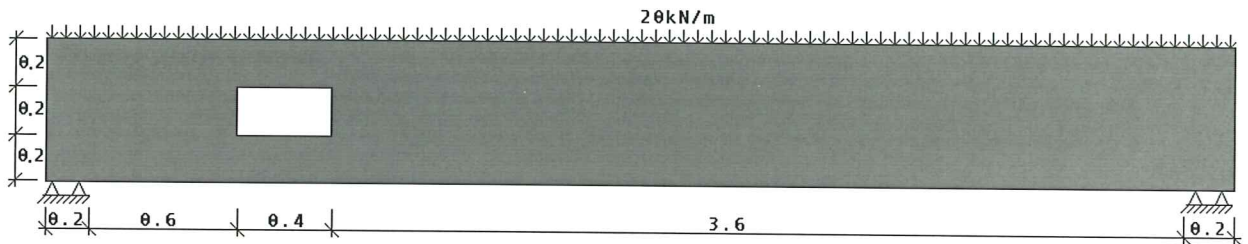


Figure 6.18 Beam with a hole

To make the mesh efficient a minimum size of 0.01 is chosen. Otherwise convergence problems will occur. A permissible error $\eta=20\%$ is selected. Five iterations give the following result.

Strategy B (AMG v1.0)		Strategy C (AMG v1.0)	
NE	ξ_g	NE	ξ_g
398	1.9383	398	1.9383
1130	1.1797	1125	1.1721
1434	1.0087	1922	1.0224
1453	0.9990	2183	1.0120
1479	0.9983	2293	0.9995

Strategy B is more efficient and produces a mesh with ca 35% less amount of elements. This example requires a few more iterations to have the global error $\xi_g < 1$ than the earlier examples. This due to that more places on the structure has stress concentrations than on example 1-4. This example is on the other hand more useful and the number of iterations is still reasonably low.

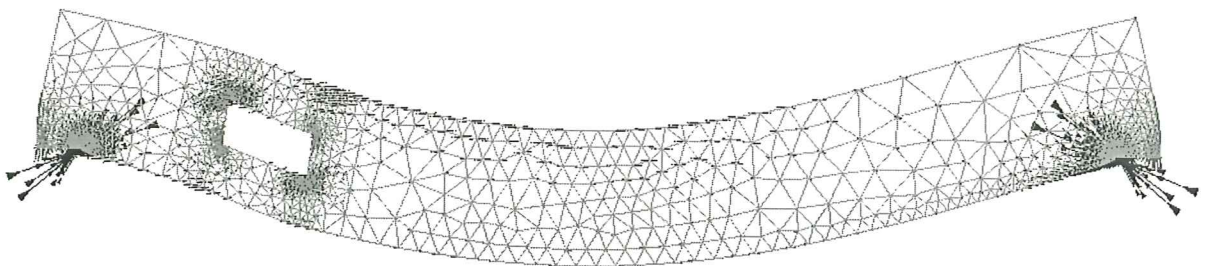
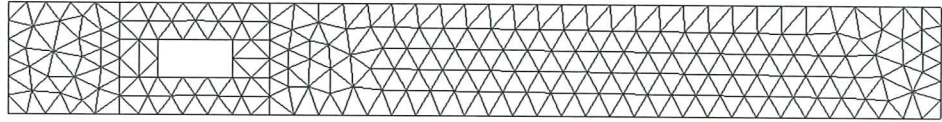
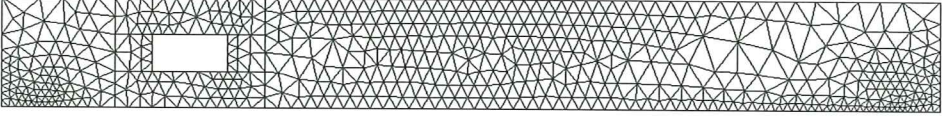
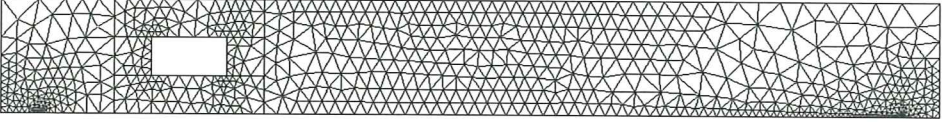
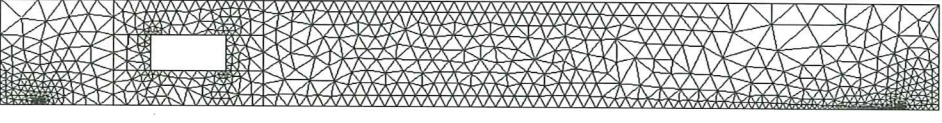
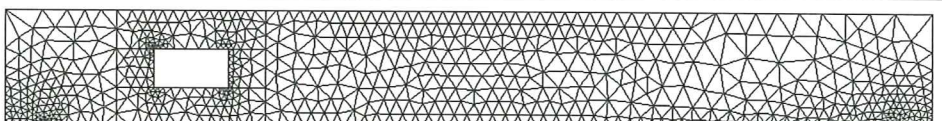
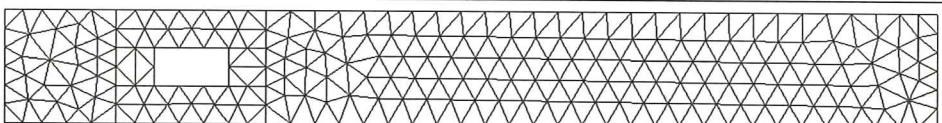
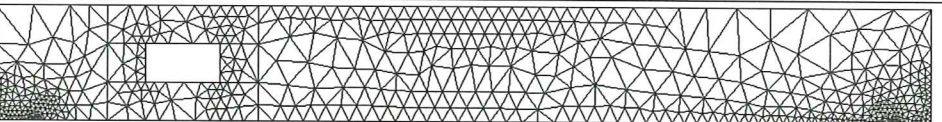
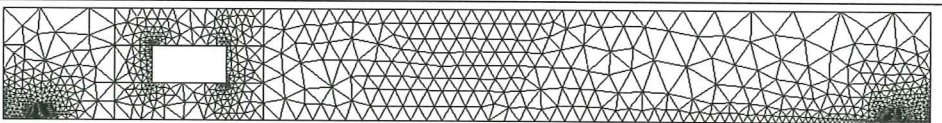
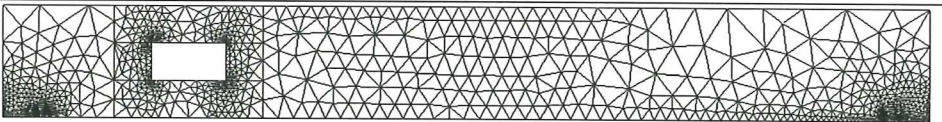


Figure 6.19 Principal stresses in the beam

The figure illustrates all the principal stresses in the structure with mesh from strategy C (2293 elements). It can clearly be seen that element sizes are small where the stresses are large and that the program is acting as wanted. The minimum size given to some of the elements are at the inner end of the support where the stresses are greatest. The deformation is approximately enlarged 810 times to see the effect of the ventilation hole. The different meshes can be seen on the next page.

	Strategy B 398 elements
	Strategy B 1130 elements
	Strategy B 1434 elements
	Strategy B 1453 elements
	Strategy B 1479 elements
	Strategy C 398 elements
	Strategy C 1125 elements
	Strategy C 1922 elements
	Strategy C 2183 elements

After each iteration, the elements are concentrated around the corners of the ventilation hole and in the vicinity of the supports, which is as expected. After only three iterations a good result is achieved for strategy B and the model starts to converge against around 1400- 1500 elements. The C strategy does not really converge as the strategy B does and requires 4-5 iterations to stabilize around 2200-2300 elements.

6.6 Concrete wall on three columns

An 8m long and 4m high concrete wall is fixed to three columns as illustrated in figure 6.20. The columns are fixed in both x- and y- direction and uniform load is applied on the wall. Two ventilation holes are cut out in the wall with a diameter of 1m each.

Geometry: 4m high columns (0.3x0.2) are supporting a wall (4x8m). The wall has two holes (d=1m).

Boundary condition: All three columns are clamped in the bottom.

Applied Load: A uniformed pressure load of 30kN/m is applied on the beam

Material: Isotropic and linear elastic material, $E=30$ GPa, $\nu=0.2$, thickness=0.2m

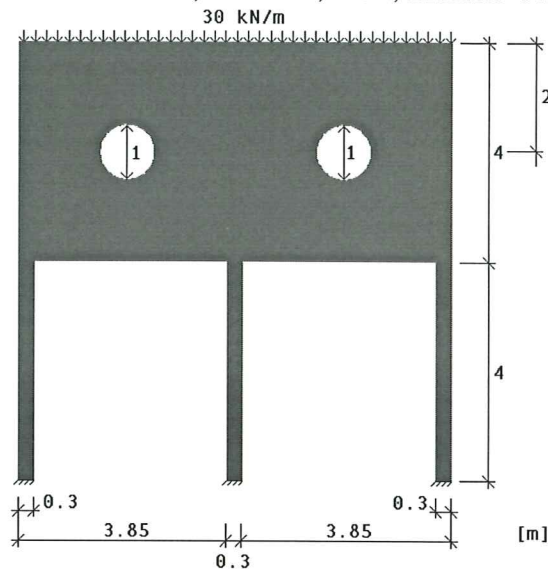


Figure 6.20 Concrete wall on three columns

Four iterations of each strategy give the result below. A minimum size of the elements is put to 0.01, due to stress singularities. Strategy C shows a strange behavior between the third and the fourth iteration; the global error increases and the elements in the mid column become distorted. The quality of the mesh must therefore be studied.

Strategy B (AMG v1.0)		Strategy C (AMG v1.0)	
NE	ξ_g	NE	ξ_g
147	3.2851	147	3.2851
1175	1.6460	2084	1.4448
2596	1.1203	8124	1.1415
3099	1.0134	9124	2.0546

Strategy B is much better in this example because C does not converge at all.

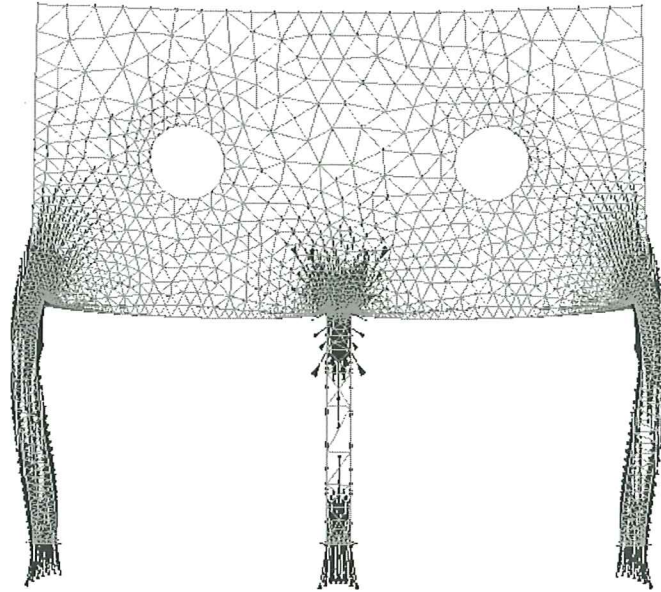
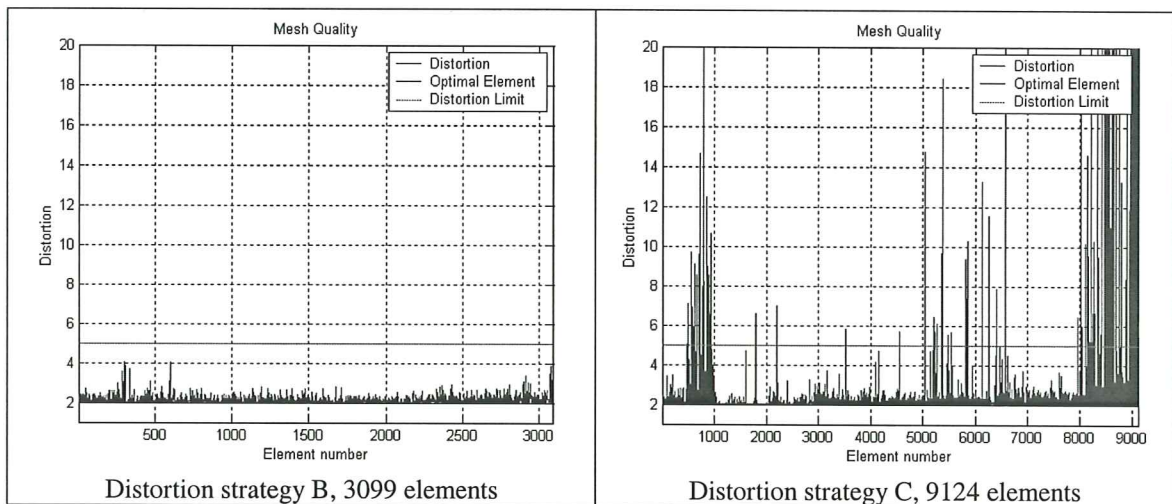


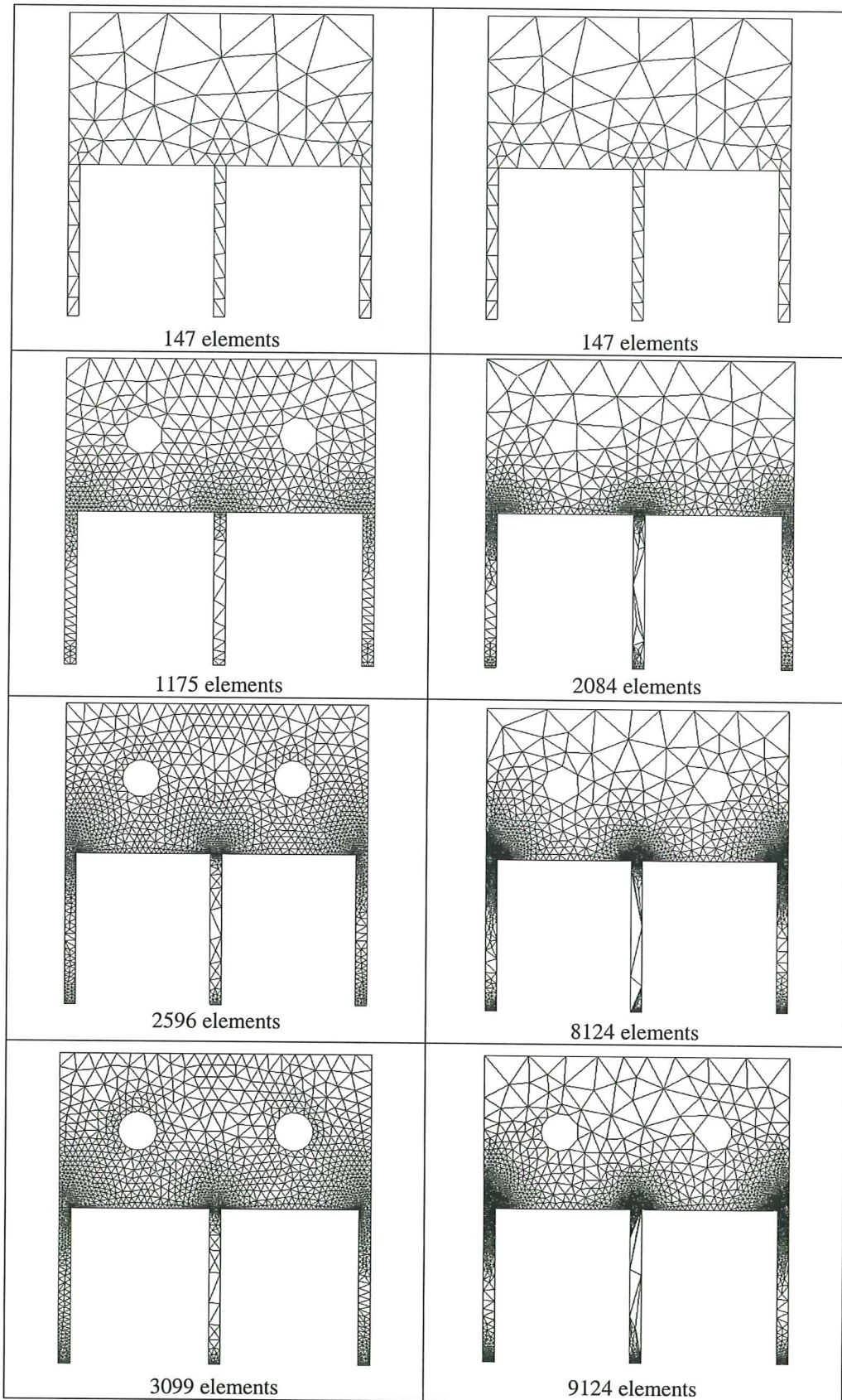
Figure 6.21 Principal stresses in wall

The principal stresses for the mesh with 3099 elements using strategy B can be studied in figure 6.21. It can clearly be seen how the forces go down in the columns. The mid column is only subjected to compression while the other columns are being bent outwards and experience both tension and compression. The elements concentrates where the highest stresses occur, as they are supposed to, and the result are satisfying.



The figures of mesh quality after the fourth iteration clearly shows that the mesh in strategy C contains severely distorted elements. The problem with the distortion is on several elements almost all over the structure and therefore strategy C is inappropriate for this example.

In the figures on the next page the mesh for strategy C shows large and strange elements in the mid column. These elements show very high distortional values. When meshing narrow structures with small errors, strategy C does not provide a satisfying mesh. Strategy B is a much better way for mesh optimization in this example.



7 Conclusions

7.1 General summary

A program for mesh improvement in 2D finite element analysis has been developed. The routine is currently compatible with GiD and Calsef, developed by CIMNE in Barcelona. The open structure of the code allows the user to monitor the process of error estimation and to customize the program for use with other applications. The intention has been to provide an educational tool for adaptive mesh generation and also serves as a first step towards implementing adaptivity in GiD.

7.2 Limitations

When using AMG v1.0, more or less satisfactory results have been obtained for all examples analyzed in this report. However, one should be aware that the program has been slightly modified before running some of the models, in order for it to produce a high quality mesh. Examples of such modifications are:

1. Raising of the limit of the element refinement parameter, which normally should be 1
For some models the parameter peaks in a few elements, even if the global error is well below 1. This action relaxes the mesh optimality criteria, in order to produce a decent mesh in fewer iterations.
2. Prescribing the maximum and/or minimum element size of the mesh produced
When dealing with models with near uniform stress distribution one needs to prescribe the maximum element size. When the optimality algorithm encounters areas with low local error it tries to increase the size, which often results in poor quality elements. In the same manner one needs to prescribe the minimum element size for models with high stress concentrations (e.g. areas near point loads and re-entrant corners).

7.3 Comparison of optimality strategies

The main disparity between strategies B and C is the number of elements produced by the two. The latter (for some cases) proves to create as much as 15 times more elements for the same number of iterations (see section 6.3.2). However, the strategy cannot be discarded since the solutions obtained sometimes are more accurate. As proposed by Oñate and Bugeda, the solution is to use this mesh optimality criterion together with a minimum element size prescription. An alternative would be to introduce a parameter concerning stress singularities, as could be found in strategy A.

Strategy B is generally more stable and can be used with less user interaction. It also proves to be more effective, mostly providing a reasonable mesh in a few iterations.

7.4 Further studies

The current version of AMG can be used as a platform for further development, incorporating quadrilateral and 3D elements. Since the mesh optimality criteria used are general, the program can be tailored to handle other types of problems such as structural dynamics, thermal analysis and fluid dynamics. The program can also be modified to work with other FE applications.

In its current state AMG is semi-automatic, i.e. the program pauses after each iteration and waits for the user to validate the mesh. A step further would be to make the program fully automatic so that only the final mesh, which fulfills the chosen criterion, is visualized. An even larger improvement would be to in some way eliminate the necessary modifications mentioned in 7.2. Whether this means discarding the two strategies completely or merely modifying the mesh optimality criteria, is very difficult to say. However, the nature of real problems in practical structural engineering shows a great diversity, why an easily modified program can be an asset.

8 References

- [1] Ottosen, N. and Petersson, H. (1992) *Introduction to the Finite Element Method*. Wiltshire: Prentice Hall ISBN 0-13-473877-2
- [2] Samuelsson, A. and Wiberg, N-E. (1998) *Finite Element Method, Basics*. Lund: Studentlitteratur ISBN 91-44-00927-5
- [3] Owen, S. *Mesh Generation: A quick introduction*. Carnegie Mellon University Computing Services. URL: <http://www.andrew.cmu.edu/~sowen/mintro.html> (2001-11-26)
- [4] Owen, S. *A Survey of Unstructured Mesh Generation Technology*. Carnegie Mellon University Computing Services. URL: <http://www.andrew.cmu.edu/~sowen/survey/index.html> (2001-11-26)
- [5] Grosz, Lutz. *Mesh Quality*. University of Karlsruhe. URL: <http://www.uni-karlsruhe.de/~vecfem/guide/xvem/help/Tutorial.FEM.html> (2001-10-24)
- [6] Zienkiewicz, O.C. and Taylor, R.L. (2000) *The finite element method Volume 1, The basis, 5th edition*. Oxford: Butterworth-Heinemann ISBN 0-7506-5049-4
- [7] Oñate, E. and Bugeda, G. (1993) *A study of mesh optimality criteria in adaptive finite element analysis*. Swansea: Pineridge Press Limited ISSN 0264-4401
- [8] Zienkiewicz, O.C. and Zhu, J.Z. (1987) *A simple error estimator and adaptive procedure for practical engineering analysis*, pp 337-357
- [9] *GiD introduction*, URL: <http://gid.cimne.upc.es/intro/in02.subst> (2001-10-15)
- [10] Ribó, R., Pasenau, M., Escolano, E. *GiD Reference Manual*. URL: <http://gid.cimne.upc.es/download/do01.subst> (2001-10-16)
- [11] *Official homepage of Calsef*, URL: <http://www.cimne.upc.es/calsef/frame.htm> (2001-10-04)
- [12] Samuelsson, A. and Wiberg, N-E. (1993) *Byggnadsmekanik Hållfasthetslära*. Lund: Studentlitteratur ISBN 91-44-28242-7
- [13] *Official homepage of Calfem*, URL: <http://www.byggmek.lth.se/Calfem/> (2001-10-08)
- [14] NAFEMS, *Linear static's benchmarks vol.1*, October 1987, test IC5
- [15] Petersson, R E, (1974) *Stress concentration factors*, Wiley, New York

Appendix A

AMG.m

The main program executable from MATLAB®, which calls the subroutines.

```
%-----
%-----Adaptive Mesh Generator v1.0-----
%-----Created by-----
%-----Joel Gustafsson and Johan Olausson-----
%-----
%-----Last edited 2001-09-21-----
%-----
%-----
%-----The program calculates the error for a three node triangular element -----
%-----and assigns new sizes to each element in the initial mesh-----
%-----
clear all;
zloop=0;
%-----Invokes GUI for user input data-----
usermenu
waitfor(usermenu);
elem_ref=5;
%-----Collects the input data from GUI-----
[xfilepath,xfilename,xeta,xstrategy]=connectmenu(1);
%-----Starts the program loop using the condition of the element refinement-----
%-----parameter-----
while max(elem_ref) > 1.00 %Change for certain problems
close all;
zloop=zloop+1;
%-----Clear all variables except those associated with input data-----
clear B* D* M* N* a* b* c* d* e* f* g* h* i* j* m* n* s* t* u* w*
%-----Clocks each iteration-----
time=cputime;
%-----Creates the full path for the problem files-----
fullpath=[xfilepath,xfilename,'.gid\'',xfilename];
%-----Calls writebat which writes 2 batch files for GiD-----
writebat(fullpath);

%-----Calls connectold which returns connectivity, coordinates, displacements----
%-----number of elements and nodes, smoothed stress values, the Constitutive ----
%-----matrix, selfweight, thickness, analyze type and element type from the GiD-
%-----files-----
[edof,coord,ex,ey,disp,nelem,nnode,sigma_smooth_gid,D,selfweight,t,an_type,el_type]=read
_gid(fullpath);

if el_type==0
w=1/6;
else
error('Only 3 node triangular elements can be used!!!!')
end

h = waitbar(0,'Assembling B matrix and the Jacobian matrix');
%-----Calls jacobtriangle which calculates the B matrix and the Jacobian -----
%-----determinant-----
%-----for each element. The Jacobian determinant and the B matrix are assembled--
%-----into the matrices detjacobian and B1-----
%-----Calculates the element stresses sigma_fem-----
for i=1:nelem
[B,detjacob]=jacobtriangle(ex(i,:),ey(i,:));
sigma_fem(:,i)=D*B*(disp(i,:))';
detjacobian(i)=detjacob;
B1(3*i-2:3*i,1:6)=B;
waitbar(i/nelem,h);
end
close(h)

sigma_fem=transpose(sigma_fem);
%-----Calculation points in natural coordinates-----
NG=[0.5 0 0.5; 0.5 0.5 0; 0 0.5 0.5];

%-----Mean value nodal stress calculation which is suitable for 3 node triangular-
%-----elements-----
[nie,n]=size(edof);
te=edof(:,2:n);
nneigh=zeros(nnode,1);
sigma_smooth_x=zeros(nnode,1);sigma_smooth_y=zeros(nnode,1);sigma_smooth_xy=zeros(nnode,
1);
%-----Collects all neighbour elements stresses on each node and calculates the----
%-----number of neighbour elements for each node-----
for i=1:nie
for j=1:n-1
nneigh(te(i,j))=nneigh(te(i,j))+1;
sigma_smooth_x(te(i,j))=sigma_smooth_x(te(i,j))+sigma_fem(i,1);
sigma_smooth_y(te(i,j))=sigma_smooth_y(te(i,j))+sigma_fem(i,2);
end
end
end
```



```

        sigma_smooth_xy(te(i,j))=sigma_smooth_xy(te(i,j))+sigma_fem(i,3);
    end
end
%-----Divides the nodal stress values with the number of neighbouring elements-----
for i=1:nnode
    sigma_smooth_x(i)=sigma_smooth_x(i)/nneigh(i);
    sigma_smooth_y(i)=sigma_smooth_y(i)/nneigh(i);
    sigma_smooth_xy(i)=sigma_smooth_xy(i)/nneigh(i);
end
sigma_smooth=[sigma_smooth_x sigma_smooth_y sigma_smooth_xy];

%-----Calculates the error energy norm and the strain energy norm in each element---
%-----Calculates the Global Error-----
unit=diag(ones(1,3));
e2=zeros(1,nelem);u2=zeros(1,nelem);

h = waitbar(0,'Calculating the Global Error');
for i=1:nelem
    clear sigma_correct;

    sigmaspec=[(sigma_smooth(te(i,1),:))';(sigma_smooth(te(i,2),:))';(sigma_smooth(te(i,3),:
    ))'];
        for j=1:3
            sigma_correct=[NG(1,j)*unit,NG(2,j)*unit,NG(3,j)*unit]*sigmaspec;
            e2(i)=e2(i)+(sigma_correct-sigma_fem(i,:))'*inv(D)*(sigma_correct-
            sigma_fem(i,:))*w*detjacobian(i);
            u2(i)=u2(i)+sigma_correct'*inv(D)*sigma_correct*w*detjacobian(i)+e2(i);
        end
        waitbar(i/nelem,h)
    end
    esum=sqrt(sum(e2));
    usum=sqrt(sum(u2));
    close(h)
%-----Calls Backgroundsize which calculates new element sizes according to one-----
%-----of two mesh optimality criteria-----
%-----Also returns the local error and the element refinement parameter-----
[backgroundfile,err_glob,err_loc,elem_ref]=backgroundsize(e2,u2,esum,usum,xeta,nelem,ex,
ey,el_type,detjacobian,xstrategy);
%-----Calls write_mesh if the criteria is not fulfilled-----
%-----Write_mesh writes a backgroundfile called c:\mesh.txt which GiD can read-----
if max(elem_ref) >1.00 %Change for certain problems
    write_mesh;
    err_glob
end
%-----Plots the element refinement parameter for each element-----
err_graph=figure;
    plot([1:nelem],elem_ref);grid on;hold on;
        title('Ratio between element error and permissible error');xlabel('Element
        number');ylabel('Refinement');
    plot([1:nelem],ones(1,nelem),'r');
    legend('Refinement','Error limit')
cpu_tid=cputime-time;
%-----If the criteria is not fulfilled GiD is executed using the batchfile which--
%-----generates the new mesh-----
%-----After the mesh has been created GiD calculates the problem-----
if max(elem_ref) >1.00 %Change for certain problems
    dos_call=['c:\cimne\gid.exe -p calsef_PlaneState_V1.0 -b c:\bat.bat
    ',xfilepath,xfilename,'.gid &'];
    gid_call=dos(dos_call);
end
    pause
%-----When the criteria is fulfilled GiD is executed using the batchfile which----
%-----only shows the final mesh-----
end
    dos_call=['c:\cimne\gid.exe -p calsef_PlaneState_V1.0 -b c:\bat2.bat
    ',xfilepath,xfilename,'.gid &'];
    gid_call=dos(dos_call);

```

Usermenu.m

```

function varargout = usermenu(varargin)
%-----
%-----The Function associated with each button-----
%-----in the GUI-----
%-----
%-----
if nargin == 0 % LAUNCH GUI

    fig = openfig(mfilename,'reuse');

    % Use system color scheme for figure:
    set(fig,'Color',get(0,'defaultUiicontrolBackgroundColor'));

    % Generate a structure of handles to pass to callbacks, and store it.
    handles = guihandles(fig);
    guidata(fig, handles);

    if nargin > 0

        varargout{1} = fig;
    end

elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
    try
        [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
    catch
        disp(lasterr);
    end

end

% -----
function varargout = edit1_Callback(h, eventdata, handles, varargin)
%-----Saves the filepath in a created file-----
savepath=pwd;
fid = fopen([savepath,'\filepath','.txt'],'wt');
filepath=get(h,'String');
fprintf(fid,filepath);
fclose(fid);
% -----
function varargout = edit2_Callback(h, eventdata, handles, varargin)
%-----Saves the filename in a created file-----
savepath=pwd;
fid = fopen([savepath,'\filename','.txt'],'wt');
filename=get(h,'String');
fprintf(fid,filename);
fclose(fid);
% -----
function varargout = popupmenu1_Callback(h, eventdata, handles, varargin)
%-----Saves the strategy in a created file-----
savepath=pwd;
fid = fopen([savepath,'\strategy','.txt'],'wt');
strategy=popupstr(h);
fprintf(fid,strategy);
fclose(fid);
% -----
function varargout = edit3_Callback(h, eventdata, handles, varargin)
%-----Saves the error in a created file-----
savepath=pwd;
fid = fopen([savepath,'\error','.txt'],'wt');
error=get(h,'String');
fprintf(fid,error);
fclose(fid);
% -----
function varargout = pushbutton4_Callback(h, eventdata, handles, varargin)
close(usermenu);

```

Connectmenu.m

```

function [filepath,filename,eta,strategy]=connectmenu(elem_ref);
%-----
%-----Reads the parameters from the files created by the function usermenu-----
%-----and returns these to AMG v1.0-----
%-----
savepath=pwd;
%-----Reads the filepath if it exists otherwise the default path is used-----
if exist([savepath,'\filepath.txt'])==2
    fid = fopen([savepath,'\filepath','.txt']);
    filepath=fscanf(fid,'%s');
    fclose(fid);
    delete filepath.txt
else
    filepath='c:\bamse\';
end
%-----Reads the filename if it exists otherwise the default name is used-----
if exist([savepath,'\filename.txt'])==2
    fid = fopen([savepath,'\filename','.txt']);
    filename=fscanf(fid,'%s');
    fclose(fid);
    delete filename.txt
else
    filename='errortest2';
end
%-----Reads the strategy if it exists otherwise the default strategy (B) is used--
if exist([savepath,'\strategy.txt'])==2
    fid = fopen([savepath,'\strategy','.txt']);
    strategy=fscanf(fid,'%s');
    fclose(fid);
    delete strategy.txt
else
    strategy='StrategyB';
end
%-----Reads the error if it exists otherwise the default error (10%) is used-----
if exist([savepath,'\error.txt'])==2
    fid = fopen([savepath,'\error','.txt']);
    eta=fscanf(fid,'%f');
    fclose(fid);
    delete error.txt
else
    eta=10;
end
eta=eta/100;

```

Writebat.m

```

function writebat(fullpath);
%-----
%-----The function writes 2 batch files for GiD to use; one if the condition is--
%-----fulfilled the other if it is not-----
%-----The batch files are created in the directory c: -----
%-----
%-----Writes the batch file which makes GiD read the background mesh and-----
%-----calculates the problem-----
fid = fopen('c:\bat.bat','wt');
fprintf(fid,'escape escape escape Meshing reset \n');
fprintf(fid,'Yes \n');
fprintf(fid,'escape escape escape Meshing AssignSizes BackgMesh \n');
fprintf(fid,'c:\mesh.txt');
fprintf(fid,'\nescape escape escape Meshing Generate \n');
fprintf(fid,'Yes \n');
fprintf(fid,'1000 \n');
fprintf(fid,'escape escape escape Meshing MeshView \n');
fprintf(fid,'escape escape escape Utilities Calculate \n');
fprintf(fid,'escape escape escape utilities variables DefaultFileNameInCalcFile 1
escape \n');
fprintf(fid,'escape escape escape files WriteCalcFile \n');
fprintf(fid,'escape escape escape utilities variables DefaultFileNameInCalcFile 0
escape \n');
fclose(fid);
%-----Writes the batch file which makes GiD read the background mesh and show the

```

```

%-----final mesh-----
fid = fopen('c:\bat2.bat','wt');
fprintf(fid,'escape escape escape Meshing reset \n');
fprintf(fid,'Yes \n');
fprintf(fid,'escape escape escape Meshing AssignSizes BackgMesh \n');
fprintf(fid,'c:\mesh.txt');
fprintf(fid,'\nescape escape escape Meshing Generate \n');
fprintf(fid,'Yes \n');
fprintf(fid,'1000 \n');
fprintf(fid,'escape escape escape Meshing MeshView \n');
fclose(fid);

```

Read_gid.m

```

function
[edof,coord,ex,ey,disp,nelem,nnode,sigma_smooth_gid,D,selfweight,t,an_type,el_type]=read
_gid(fullpath)
%-----
%-----The function reads the connectivity, coordinates, displacements and-----
%-----nodal stress values from the GiD files *.cal and *.flavia.res-----
%-----
%-----Reads the number of nodes and elements-----
fid=fopen([fullpath,'.cal']);
ident='Identifier';
d=size(ident);
g=1;
while g>-1
    g=fgetl(fid);
    if size(g)==d
        if ident=='Identifier'
            n=fscanf(fid,'%f');
            nelem=n(1);nnode=n(2);E=n(3);v=n(4);
            selfweight=n(5);t=n(6);an_type=n(7);el_type=n(8);
        end
    end
end
fclose(fid);
%-----Calculates the constitutive matrix D for plane stress-----
if an_type==1
    D=E/(1-v^2)*[1, v, 0; v, 1, 0; 0, 0, 0.5*(1-v)];
elseif an_type==2
%-----Calculates the constitutive matrix D for plane strain-----
    D=E/((1+v)*(1-2*v))*[1-v, v, 0; v, 1-v, 0; 0, 0, 0.5*(1-2*v)];
else
    error('Check analysis type, an_type=1 for plane stress and 2 for plane strain');
end
%-----Specify were to read in the files-----
range_a=[50 0 49+nelem*5 0];
range_b=[52+nelem 0 51+nelem+nnode*3 0];
range_c=[4 0 3+nnode*3 0];
range_d=[9+nnode 0 8+nnode+nnode*5 0];
%-----Reads the connectivity-----
a=dlmread([fullpath,'.cal'],'\t',range_a);
%-----Reads the nodal coordinates-----
b=dlmread([fullpath,'.cal'],'\t',range_b);
%-----Reads the displacements-----
c=dlmread([fullpath,'.flavia.res'],'\t',range_c);
%-----Reads the smoothed stresses-----
d=dlmread([fullpath,'.flavia.res'],'\t',range_d);

%-----Collects the connectivity into edof-----

for i=1:nelem
    edof(i,1)=a(5*i-4);
    edof(i,2)=a(5*i-2);
    edof(i,3)=a(5*i-1);
    edof(i,4)=a(5*i);
end
%-----Collects the coordinates into coord-----
for i=1:nnode
    coord(i,1)=b(3*i-1);
    coord(i,2)=b(3*i);
end
%-----Collects the displacements into disp-----
for i=1:nnode
    disp(i,1)=c(3*i-1);

```

```

        disp(i,2)=c(3*i);
    end
    %-----Collects the nodal smoothed stress values into sigma_smooth_gid-----
    for i=1:nnode
        sigma_smooth_gid(i,1)=d(5*i-3);
        sigma_smooth_gid(i,2)=d(5*i-2);
        sigma_smooth_gid(i,3)=d(5*i-1);
    end
    %-----Calls coordxtr which collects the nodal coordinates and displacements-----
    %-----elementwise-----
    dof=transpose([1:1:nnode]);
    [ex,ey]=coordxtr(edof,coord,dof,3);
    elnum=edof(:,1);
    [dispx,dispy]=coordxtr(edof,disp,dof,3);
    clear disp;
    %-----Collects the displacements elementwise in the format-----
    %-----[ux1 uy1 ux2 uy2 ux3 uy3 ux4 uy4]-----
    for i=1:nelem
        for j=1:3
            disp(i,j*2-1)=dispx(i,j);
            disp(i,j*2)=dispy(i,j);
        end
    end
end

```

Coordxtr.m (based on Calfem file)

```

function [Ex,Ey,Ez]=coordxtr(Edof,Coord,Dof,nen)
%[Ex,Ey,Ez]=coordxtr(Edof,Coord,Dof,nen)
%-----
% PURPOSE
%   Extract nodal coordinate data from the global coordinate
%   matrix for a number of elements with equal number of
%   element nodes and dof's.
%
% INPUT:  Edof : topology matrix , dim(t)= nie x ned+1
%          nie= number of identical elements
%          ned= number of element dof's
%
%          Coord: global coordinate matrix
%
%          Dof:   global nodal dof matrix
%
%          nen:   number of element nodes
%
% OUTPUT: Ex,Ey,Ez : element coordinate matrices
%          Ex=[x1 x2 ...xnen;   one row for each element
%             ...   ... ;
%             nel   ... ]
%          dim= nel x nen ;   nel:number of elemnts
%-----
% LAST MODIFIED: Johan Olausson & Joel Gustafsson 2001-11-20
%-----
    [nel,dum]=size(Edof);
    ned=dum-1;
    [n,nsd]=size(Coord);
    [n,nd]=size(Dof);
    nend=ned/nen;

    for i = 1:nel
        nodnum=zeros(1,nen);
        for j = 1:nen
            check=Dof(:,1:nend)-ones(n,1)*Edof(i,(j-1)*nend+2:j*nend+1);
            [indx,dum]=find(check==0);
            nodnum(j)=indx(1);
        end
    end

    Ex(i,:)=Coord(nodnum,1)';
    if nsd>1
        Ey(i,:)=Coord(nodnum,2)';
    end
    if nsd>2
        Ez(i,:)=Coord(nodnum,3)';
    end
end
%-----end-----

```

Jacobtriangle.m

```

function [B,detjacob]=jacobtriangle(ex,ey)
%-----
%-----Calculates the Jacobian, the jacobian matrix and the Operator B for a-----
%-----three node triangular element-----
x1=ex(1);y1=ey(1);x2=ex(2);y2=ey(2);x3=ex(3);y3=ey(3);
coord=[x1 y1;x2 y2;x3 y3];
%-----Derivation of the natural coordinates-----
dNksi1=-1;
dNksi2=1;
dNksi3=0;
dNeta1=-1;
dNeta2=0;
dNeta3=1;

dN=[dNksi1,dNeta1;dNksi2,dNeta2;dNksi3,dNeta3];
%-----Calculates the Jacobian matrix-----
jacoben=[x2-x1, y2-y1; x3-x1, y3-y1];
detjacob=det(jacoben);
%-----Numerical approximation of dxdksi,dxdeta,dydksi and dydeta-----
dxdksi=dN(1,1)*coord(1,1)+dN(2,1)*coord(2,1)+dN(3,1)*coord(3,1);
dxdeta=dN(1,2)*coord(1,1)+dN(2,2)*coord(2,1)+dN(3,2)*coord(3,1);
dydksi=dN(1,1)*coord(1,2)+dN(2,1)*coord(2,2)+dN(3,1)*coord(3,2);
dydeta=dN(1,2)*coord(1,2)+dN(2,2)*coord(2,2)+dN(3,2)*coord(3,2);
%-----Establish the operator B-----
b1=dydeta*dNksi1-dydksi*dNeta1;
c1=dxdksi*dNeta1-dxdeta*dNksi1;
b2=dydeta*dNksi2-dydksi*dNeta2;
c2=dxdksi*dNeta2-dxdeta*dNksi2;
b3=dydeta*dNksi3-dydksi*dNeta3;
c3=dxdksi*dNeta3-dxdeta*dNksi3;
B=1/det(jacoben)*[ b1 0 b2 0 b3 0;
                  0 c1 0 c2 0 c3;
                  c1 b1 c2 b2 c3 b3];

```

Backgroundsize.m

```

function
[backgroundfile,err_glob,err_loc,elem_ref]=backgroundsize(e2,u2,esum,usum,eta,nelem,ex,e
y,el_type,detjacobian,strategy)
%-----Function which calculates the new element sizes depending on which strategy--
%-----is used in the problem-----
err_glob=esum/usum/eta;

m=1;d=2;

%-----Calculates the elements mean sidelength-----
%-----to obtain the element height for the different strategies-----
if strategy=='StrategyB'
    for i=1:nelem
        h(i)=(sqrt((ex(i,2)-ex(i,1))^2+(ey(i,2)-ey(i,1))^2)+...
            sqrt((ex(i,3)-ex(i,2))^2+(ey(i,3)-ey(i,2))^2)+...
            sqrt((ex(i,3)-ex(i,1))^2+(ey(i,3)-ey(i,1))^2))/3*sqrt(3)/2;
        err_loc(i)=sqrt(e2(i))/(esum*nelem^(-0.5));
        elem_ref(i)=sqrt(e2(i))/(eta*usum*nelem^(-0.5));
        beta(i)=err_glob^(1/m)*err_loc(i)^(2/(2*m+d));
        h_new(i)=h(i)/beta(i);
    end
elseif strategy=='StrategyC'
    jacob_tot=sum(detjacobian);
    for i=1:nelem
        h(i)=(sqrt((ex(i,2)-ex(i,1))^2+(ey(i,2)-ey(i,1))^2)+...
            sqrt((ex(i,3)-ex(i,2))^2+(ey(i,3)-ey(i,2))^2)+...
            sqrt((ex(i,3)-ex(i,1))^2+(ey(i,3)-ey(i,1))^2))/3*sqrt(3)/2;
        err_loc(i)=sqrt(e2(i))/esum*sqrt(jacob_tot/detjacobian(i));
        elem_ref(i)=err_loc(i)*err_glob;
        beta(i)=(elem_ref(i))^(1/m);
        h_new(i)=h(i)/beta(i);
    end
end
end
%-----Sidelength is set to avoid elements of extreme sizes-----
for i=1:nelem

```

```

    if h_new(i)<0.01
        h_new(i)=0.01;
        elem_ref(i)=1;
    end
    if h_new(i)>10
        h_new(i)=10;
        elem_ref(i)=1;
    end
end
%-----Writes a matrix with the first column being the element number and the-----
%-----second being the element size-----
backgroundfile=zeros(nelem,2);
for i=1:nelem
    backgroundfile(i,1)=i;
    backgroundfile(i,2)=h_new(i);
end
end

```

Write_mesh.m

```

%-----
%-----The program writes Backgroundmeshfile under the c: directory-----
%-----The file is called mesh.txt and is readable by GiD-----
%-----
coordn=zeros(nnode,3);
for i=1:nnode
    coordn(i,1)=i;
    coordn(i,2)=coord(i,1);
    coordn(i,3)=coord(i,2);
end
fid = fopen('c:\mesh.txt','wt');
fprintf(fid,'BackgroundMesh V 1.0 MESH \n');
fprintf(fid,'MESH dimension 3 ElemType Triangle Nnode 3 \n');
fprintf(fid,'Coordinates \n');
fprintf(fid,'%1.0f %12.8f %12.8f\n',coordn);
fprintf(fid,'end coordinates \n');
fprintf(fid,'\n');
fprintf(fid,'Elements \n');
fprintf(fid,'%1.0f %1.0f %1.0f %1.0f\n',edof');
fprintf(fid,'\n');
fprintf(fid,'end elements \n');
fprintf(fid,'\n');
fprintf(fid,'DesiredSize Elements \n');
fprintf(fid,'%1.0f %12.8f \n',backgroundfile');
fprintf(fid,'\n');
fprintf(fid,'End DesiredSize \n');
fclose(fid);

```

Quality.m

```

%-----
%-----Calculates and plots the element distorsion for-----
%-----a mesh with element coordinates ex and ey-----
%-----
for i=1:nelem
    clear a b c s C1 K
    a=sqrt((ex(i,2)-ex(i,1))^2+(ey(i,2)-ey(i,1))^2);
    b=sqrt((ex(i,3)-ex(i,2))^2+(ey(i,3)-ey(i,2))^2);
    c=sqrt((ex(i,3)-ex(i,1))^2+(ey(i,3)-ey(i,1))^2);
    s=(a+b+c)/2;
    C1=[1 ex(i,1) ey(i,1);1 ex(i,2) ey(i,2);1 ex(i,3) ey(i,3)];
    K=det(C1)/2;
    H(i)=a*b*c/(4*K);
    R(i)=K/s;
    qual(i)=H(i)/R(i);
end
mesh_qual=figure;
plot([1:nelem],qual(1:nelem));grid on;hold on;axis([1 nelem 1 20]);
title('Mesh Quality');xlabel('Element number');ylabel('Distortion');
plot([1:nelem],2*ones(1,nelem),'k');plot([1:nelem],5*ones(1,nelem),'r');
legend('Distortion','Optimal Element','Distortion Limit')

```

Appendix B

Example1

Results from 3 node triangle elements

#CALSEF 2001 by Dr. Francisco Zarate and Ing. Daniel Di Capua

Result "DISPLACEMENTS" "Load Analys" 1 Vector OnNodes

ComponentNames "Disp-X" "Disp-Y" "Disp-Z"

Values

1	0.0000000E+00	0.0000000E+00	0.0000000E+00
2	0.0000000E+00	0.0000000E+00	0.0000000E+00
3	-0.1156482E-17	-0.1000000E-01	0.0000000E+00
4	-0.1734723E-17	-0.1000000E-01	0.0000000E+00

End Values

Result "Tensiones_SET1" " " "Load Analys" 1 PlainDeformationMatrix OnNodes

ComponentNames "Tens-X" "Tens-Y" "Tens-Z" "Tens-XY"

Values

1	0.0000000E+00	-0.1000000E+05	-0.5782412E-12	0.0000000E+00
2	-0.2891206E-12	-0.1000000E+05	-0.2891206E-12	0.0000000E+00
3	-0.2891206E-12	-0.1000000E+05	-0.2891206E-12	0.0000000E+00
4	-0.5782412E-12	-0.1000000E+05	0.0000000E+00	0.0000000E+00

End Values

Results from 6 node triangle elements

#CALSEF 2001 by Dr. Francisco Zarate and Ing. Daniel Di Capua

Result "DISPLACEMENTS" "Load Analys" 1 Vector OnNodes

ComponentNames "Disp-X" "Disp-Y" "Disp-Z"

Values

1	0.0000000E+00	0.0000000E+00	0.0000000E+00
2	-0.7589415E-18	-0.5000000E-02	0.0000000E+00
3	-0.1264903E-17	0.0000000E+00	0.0000000E+00
4	-0.2059984E-17	-0.5000000E-02	0.0000000E+00
5	-0.4445229E-17	-0.1000000E-01	0.0000000E+00
6	-0.5637851E-17	0.0000000E+00	0.0000000E+00
7	-0.9757820E-18	-0.5000000E-02	0.0000000E+00
8	-0.7589415E-18	-0.1000000E-01	0.0000000E+00
9	0.1301043E-17	-0.1000000E-01	0.0000000E+00

End Values

Result "Tensiones_SET1" " " "Load Analys" 1 PlainDeformationMatrix OnNodes

ComponentNames "Tens-X" "Tens-Y" "Tens-Z" "Tens-XY"

Values

1	-0.1517883E-11	-0.1000000E+05	0.2501110E-11	0.0000000E+00
2	-0.1554023E-11	-0.1000000E+05	0.4888534E-11	0.0000000E+00
3	-0.4625929E-11	-0.1000000E+05	0.2387424E-11	0.0000000E+00
4	-0.5421011E-12	-0.1000000E+05	0.9094947E-12	0.0000000E+00
5	0.2294895E-11	-0.1000000E+05	0.4320100E-11	0.0000000E+00
6	-0.3379097E-11	-0.1000000E+05	-0.2501110E-11	0.0000000E+00
7	0.1951564E-11	-0.1000000E+05	-0.5229595E-11	0.0000000E+00
8	0.4553649E-11	-0.1000000E+05	-0.9094947E-12	0.0000000E+00
9	0.2927346E-11	-0.1000000E+05	-0.3183231E-11	0.0000000E+00

End Values

Results from 4 node rectangular elements

#CALSEF 2001 by Dr. Francisco Zarate and Ing. Daniel Di Capua

Result "DISPLACEMENTS" "Load Analys" 1 Vector OnNodes

ComponentNames "Disp-X" "Disp-Y" "Disp-Z"

Values

1	0.0000000E+00	0.0000000E+00	0.0000000E+00
2	-0.4165186E-33	0.0000000E+00	0.0000000E+00
3	-0.2602085E-17	-0.1000000E-01	0.0000000E+00
4	-0.2602085E-17	-0.1000000E-01	0.0000000E+00

End Values

Result "Tensiones_SET1" " " "Load Analys" 1 PlainDeformationMatrix OnNodes

ComponentNames "Tens-X" "Tens-Y" "Tens-Z" "Tens-XY"

Values

1	-0.3029226E-27	-0.1000000E+05	-0.9094947E-12	0.0000000E+00
2	0.0000000E+00	-0.1000000E+05	0.4547474E-12	0.0000000E+00
3	0.0000000E+00	-0.1000000E+05	0.4547474E-12	0.0000000E+00
4	-0.3029226E-27	-0.1000000E+05	-0.9094947E-12	0.0000000E+00

End Values

Results from 8 node rectangular elements

#CALSEF 2001 by Dr. Francisco Zarate and Ing. Daniel Di Capua

Result "DISPLACEMENTS" "Load Analys" 1 Vector OnNodes

ComponentNames "Disp-X" "Disp-Y" "Disp-Z"

Values

1	0.0000000E+00	0.0000000E+00	0.0000000E+00
2	-0.6523688E-17	0.0000000E+00	0.0000000E+00
3	-0.1127570E-16	-0.5000000E-02	0.0000000E+00
4	-0.9889268E-17	0.0000000E+00	0.0000000E+00


```

5 -0.2515349E-16 -0.1000000E-01 0.0000000E+00
6 -0.1360674E-16 -0.5000000E-02 0.0000000E+00
7 -0.2233456E-16 -0.1000000E-01 0.0000000E+00
8 -0.2168404E-16 -0.1000000E-01 0.0000000E+00
End Values
Result "Tensiones_SET1" "Load Analys" 1 PlainDeformationMatrix OnNodes
ComponentNames "Tens-X" "Tens-Y" "Tens-Z" "Tens-XY"
Values
1 -0.1199470E-10 -0.1000000E+05 -0.5911716E-11 0.0000000E+00
2 -0.8032168E-11 -0.1000000E+05 -0.3183231E-11 0.0000000E+00
3 -0.3097657E-11 -0.1000000E+05 0.0000000E+00 0.0000000E+00
4 -0.4069638E-11 -0.1000000E+05 -0.2160050E-11 0.0000000E+00
5 0.4744734E-11 -0.1000000E+05 0.5229595E-11 0.0000000E+00
6 -0.1564413E-11 -0.1000000E+05 -0.4547474E-11 0.0000000E+00
7 0.2315448E-11 -0.1000000E+05 -0.1250555E-11 0.0000000E+00
8 -0.1138382E-12 -0.1000000E+05 -0.1125500E-10 0.0000000E+00
End Values

```

Example 2**Results from 3 node triangle elements**

#CALSEF 2001 by Dr. Francisco Zarate and Ing. Daniel Di Capua

Result "DISPLACEMENTS" "Load Analys" 1 Vector OnNodes

ComponentNames "Disp-X" "Disp-Y" "Disp-Z"

Values

```

1 0.0000000E+00 0.0000000E+00 0.0000000E+00
2 -0.5000035E-07 -0.5000035E-07 0.0000000E+00
3 -0.7071450E-02 -0.7070850E-02 0.0000000E+00
4 -0.7071200E-02 -0.7070500E-02 0.0000000E+00

```

End Values

Result "Tensiones_SET1" "Load Analys" 1 PlainDeformationMatrix OnNodes

ComponentNames "Tens-X" "Tens-Y" "Tens-Z" "Tens-XY"

Values

```

1 -0.5000298E+04 -0.4999823E+04 -0.5000011E+04 0.0000000E+00
2 -0.5000142E+04 -0.4999823E+04 -0.4999922E+04 0.0000000E+00
3 -0.5000142E+04 -0.4999823E+04 -0.4999922E+04 0.0000000E+00
4 -0.4999985E+04 -0.4999823E+04 -0.4999834E+04 0.0000000E+00

```

End Values

Results from 6 node triangle elements

#CALSEF 2001 by Dr. Francisco Zarate and Ing. Daniel Di Capua

Result "DISPLACEMENTS" "Load Analys" 1 Vector OnNodes

ComponentNames "Disp-X" "Disp-Y" "Disp-Z"

Values

```

1 -0.7071337E-02 -0.7068275E-02 0.0000000E+00
2 -0.7072305E-02 -0.7069989E-02 0.0000000E+00
3 -0.3535548E-02 -0.3534658E-02 0.0000000E+00
4 -0.3535887E-02 -0.3535081E-02 0.0000000E+00
5 -0.7072450E-02 -0.7070763E-02 0.0000000E+00
6 -0.3751067E-07 -0.3751067E-07 0.0000000E+00
7 -0.3536133E-02 -0.3535430E-02 0.0000000E+00
8 -0.8906768E-07 -0.8906768E-07 0.0000000E+00
9 0.0000000E+00 0.0000000E+00 0.0000000E+00

```

End Values

Result "Tensiones_SET1" "Load Analys" 1 PlainDeformationMatrix OnNodes

ComponentNames "Tens-X" "Tens-Y" "Tens-Z" "Tens-XY"

Values

```

1 -0.5000004E+04 -0.4999761E+04 -0.4999111E+04 0.0000000E+00
2 -0.5000428E+04 -0.499973E+04 -0.4999513E+04 0.0000000E+00
3 -0.4999568E+04 -0.4999549E+04 -0.4999255E+04 0.0000000E+00
4 -0.5000009E+04 -0.4999785E+04 -0.4999758E+04 0.0000000E+00
5 -0.5000539E+04 -0.5000209E+04 -0.4999981E+04 0.0000000E+00
6 -0.4999480E+04 -0.4999360E+04 -0.4999536E+04 0.0000000E+00
7 -0.5000465E+04 -0.5000089E+04 -0.5000079E+04 0.0000000E+00
8 -0.5000264E+04 -0.4999665E+04 -0.4999891E+04 0.0000000E+00
9 -0.5000703E+04 -0.4999947E+04 -0.5000111E+04 0.0000000E+00

```

End Values

Results from 4 node rectangular element

#CALSEF 2001 by Dr. Francisco Zarate and Ing. Daniel Di Capua

Result "DISPLACEMENTS" "Load Analys" 1 Vector OnNodes

ComponentNames "Disp-X" "Disp-Y" "Disp-Z"

Values

```

1 0.0000000E+00 0.0000000E+00 0.0000000E+00
2 -0.1499986E-06 -0.1499986E-06 0.0000000E+00
3 -0.7071950E-02 -0.7070750E-02 0.0000000E+00
4 -0.7071200E-02 -0.7070100E-02 0.0000000E+00

```

End Values

Result "Tensiones_SET1" "Load Analys" 1 PlainDeformationMatrix OnNodes

```

ComponentNames "Tens-X" "Tens-Y" "Tens-Z" "Tens-XY"
Values
  1 -0.5000493E+04 -0.4999753E+04 -0.5000020E+04 0.0000000E+00
  2 -0.5000141E+04 -0.4999777E+04 -0.5000083E+04 0.0000000E+00
  3 -0.4999789E+04 -0.4999753E+04 -0.4999795E+04 0.0000000E+00
  4 -0.5000141E+04 -0.4999528E+04 -0.4999731E+04 0.0000000E+00
End Values
Results from 4 node rectangular element
#CALSEF 2001 by Dr. Francisco Zarate and Ing. Daniel Di Capua
Result "DISPLACEMENTS" "Load Analsys" 1 Vector OnNodes
ComponentNames "Disp-X" "Disp-Y" "Disp-Z"
Values
  1 -0.1939914E-06 -0.1939914E-06 0.0000000E+00
  2 -0.3535444E-02 -0.3534497E-02 0.0000000E+00
  3 -0.1435821E-06 -0.1435821E-06 0.0000000E+00
  4 -0.7071559E-02 -0.7067782E-02 0.0000000E+00
  5 0.0000000E+00 0.0000000E+00 0.0000000E+00
  6 -0.3536302E-02 -0.3535248E-02 0.0000000E+00
  7 -0.7072522E-02 -0.7069907E-02 0.0000000E+00
  8 -0.7072240E-02 -0.7070701E-02 0.0000000E+00
End Values
Result "Tensiones_SET1" "Load Analsys" 1 PlainDeformationMatrix OnNodes
ComponentNames "Tens-X" "Tens-Y" "Tens-Z" "Tens-XY"
Values
  1 -0.4998647E+04 -0.4998984E+04 -0.4998968E+04 0.0000000E+00
  2 -0.4999052E+04 -0.4999299E+04 -0.4999276E+04 0.0000000E+00
  3 -0.5000075E+04 -0.4999492E+04 -0.4999818E+04 0.0000000E+00
  4 -0.5000267E+04 -0.4999961E+04 -0.4999005E+04 0.0000000E+00
  5 -0.5001200E+04 -0.4999605E+04 -0.5000320E+04 0.0000000E+00
  6 -0.5000590E+04 -0.4999823E+04 -0.5000257E+04 0.0000000E+00
  7 -0.5000680E+04 -0.5000372E+04 -0.4999485E+04 0.0000000E+00
  8 -0.5000791E+04 -0.5000388E+04 -0.4999616E+04 0.0000000E+00
End Values

```

Example 3

```

Results from 3 node triangular element
#CALSEF 2001 by Dr. Francisco Zarate and Ing. Daniel Di Capua
Result "DISPLACEMENTS" "Load Analsys" 1 Vector OnNodes
ComponentNames "Disp-X" "Disp-Y" "Disp-Z"
Values
  1 0.1000000E-01 0.1000000E-01 0.0000000E+00
  2 0.1001250E-01 -0.1000000E-01 0.0000000E+00
  3 0.3003000E-01 0.1000750E-01 0.0000000E+00
  4 0.3003750E-01 -0.1000750E-01 0.0000000E+00
End Values
Result "Tensiones_SET1" "Load Analsys" 1 PlainDeformationMatrix OnNodes
ComponentNames "Tens-X" "Tens-Y" "Tens-Z" "Tens-XY"
Values
  1 0.1000000E+02 0.0000000E+00 0.1000000E+02 0.0000000E+00
  2 0.1250000E+02 -0.7500000E+01 0.1250000E+02 0.0000000E+00
  3 0.7500000E+01 0.7500000E+01 0.7500000E+01 0.0000000E+00
  4 0.1000000E+02 0.0000000E+00 0.1000000E+02 0.0000000E+00
End Values

```

```

Results from 6 node triangular element
#CALSEF 2001 by Dr. Francisco Zarate and Ing. Daniel Di Capua
Result "DISPLACEMENTS" "Load Analsys" 1 Vector OnNodes
ComponentNames "Disp-X" "Disp-Y" "Disp-Z"
Values
  1 0.1000000E-01 0.1000000E-01 0.0000000E+00
  2 0.2003300E-01 0.1002062E-01 0.0000000E+00
  3 0.1001500E-01 -0.1125000E-05 0.0000000E+00
  4 0.2003525E-01 -0.7500000E-06 0.0000000E+00
  5 0.3006450E-01 0.1002850E-01 0.0000000E+00
  6 0.1002100E-01 -0.1000000E-01 0.0000000E+00
  7 0.2003750E-01 -0.1002213E-01 0.0000000E+00
  8 0.3007050E-01 -0.3750000E-06 0.0000000E+00
  9 0.3008550E-01 -0.1003150E-01 0.0000000E+00
End Values
Result "Tensiones_SET1" "Load Analsys" 1 PlainDeformationMatrix OnNodes
ComponentNames "Tens-X" "Tens-Y" "Tens-Z" "Tens-XY"
Values
  1 0.1175000E+02 0.2175000E+02 0.1175000E+02 0.0000000E+00
  2 0.2750000E+01 0.1925000E+02 0.1100000E+02 0.0000000E+00
  3 0.1550000E+02 -0.1100000E+02 0.7250000E+01 0.0000000E+00
  4 0.1175000E+02 0.6750000E+01 0.1175000E+02 0.0000000E+00

```

```

5      0.6500000E+01    0.6500000E+01    0.6500000E+01    0.0000000E+00
6      0.6500000E+01   -0.3350000E+02    0.6500000E+01    0.0000000E+00
7      0.2750000E+01   -0.2075000E+02    0.1100000E+02    0.0000000E+00
8      0.1550000E+02   -0.1000000E+01    0.7250000E+01    0.0000000E+00
9      0.1175000E+02   -0.8250000E+01    0.1175000E+02    0.0000000E+00
End Values

```

Results from 4 node rectangular element

```
#CALSEF 2001 by Dr. Francisco Zarate and Ing. Daniel Di Capua
```

```
Result "DISPLACEMENTS" "Load Analys" 1 Vector OnNodes
```

```
ComponentNames "Disp-X" "Disp-Y" "Disp-Z"
```

```
Values
```

```

1      0.1000000E-01    0.1000000E-01    0.0000000E+00
2      0.1001000E-01   -0.1000000E-01    0.0000000E+00
3      0.3004000E-01    0.1002000E-01    0.0000000E+00
4      0.3005000E-01   -0.1002000E-01    0.0000000E+00

```

```
End Values
```

```
Result "Tensiones_SET1" " " "Load Analys" 1 PlainDeformationMatrix OnNodes
```

```
ComponentNames "Tens-X" "Tens-Y" "Tens-Z" "Tens-XY"
```

```
Values
```

```

1      0.1000000E+02    0.1154701E+02    0.1577350E+02    0.0000000E+00
2      0.1000000E+02    0.1154701E+02    0.4226497E+01    0.0000000E+00
3      0.1000000E+02   -0.1154701E+02    0.4226497E+01    0.0000000E+00
4      0.1000000E+02   -0.1154701E+02    0.1577350E+02    0.0000000E+00

```

```
End Values
```

Results from 8 node rectangular element

```
#CALSEF 2001 by Dr. Francisco Zarate and Ing. Daniel Di Capua
```

```
Result "DISPLACEMENTS" "Load Analys" 1 Vector OnNodes
```

```
ComponentNames "Disp-X" "Disp-Y" "Disp-Z"
```

```
Values
```

```

1      0.1000000E-01    0.1000000E-01    0.0000000E+00
2      0.1002640E-01   -0.1562500E-04    0.0000000E+00
3      0.2004847E-01    0.1001140E-01    0.0000000E+00
4      0.1003917E-01   -0.1000000E-01    0.0000000E+00
5      0.3006818E-01    0.1000917E-01    0.0000000E+00
6      0.2004388E-01   -0.1003223E-01    0.0000000E+00
7      0.3008095E-01   -0.5208333E-05    0.0000000E+00
8      0.3010735E-01   -0.1005083E-01    0.0000000E+00

```

```
End Values
```

```
Result "Tensiones_SET1" " " "Load Analys" 1 PlainDeformationMatrix OnNodes
```

```
ComponentNames "Tens-X" "Tens-Y" "Tens-Z" "Tens-XY"
```

```
Values
```

```

1      0.3803030E+02    0.3126820E+02    0.2113636E+02    0.0000000E+00
2      0.2166667E+02    0.1041667E+02    0.9545455E+01    0.0000000E+00
3      -0.4583333E+01    0.1490457E+02    0.9545455E+01    0.0000000E+00
4      0.5303030E+01   -0.4793487E+02    0.6136364E+01    0.0000000E+00
5      0.5303030E+01   -0.1459070E+01    0.6136364E+01    0.0000000E+00
6      -0.4583333E+01   -0.3157123E+02    0.9545455E+01    0.0000000E+00
7      0.2166667E+02    0.1041667E+02    0.9545455E+01    0.0000000E+00
8      0.3803030E+02   -0.1520760E+02    0.2113636E+02    0.0000000E+00

```

```
End Values
```