SSSU 108
ISSN 0140 3818

# UNIVERSITY OF SOUTHAMPTON

DEPARTMENT OF SHIP SCIENCE

FACULTY OF ENGINEERING AND APPLIED SCIENCE

# TECHNIQUES FOR ASSESSING THE FLOW AND SPATIAL QUALITY OF ARBITRARY 3D COMPUTATIONAL MESHES

A.M.Wright and S.R.Turnock

Ship Science Report 108

March 1999

# Techniques for assessing the flow and spatial quality of arbitrary 3D computational meshes

A. M. Wright
S. R. Turnock
Department of Ship Science, University of Southampton

March 31, 1999

**Abstract**

The solution accuracy of any model using computational fluid dynamics is dependent upon the quality of the discretised mesh. This report details the range of spatial and flow properties affecting the accuracy and stability of any solution and comments upon their individual effects. Methods of calculating these properties upon an arbitrary 3D mesh are defined and statistical analysis provides a unique quantative value upon any given mesh. As a result volumes of poor grid quality can be identified without user interpretation, as well as the most suitable form of rectification.

Ship Science Report 108

# Contents

# List of Figures

# List of Tables

# 1 Introduction

The computational solution of any mathematical problem requires the discretisation of the physical domain into distinct sections, be it for a finite difference, finite element or finite volume methodology. The arrangement of these discrete points throughout the flow field is simply called a *grid.* The manner of grid generation is a significant consideration in CFD; the type of grid chosen for a given problem strongly influences whether a solution is possible, the amount of computational effort required, and the spatial accuracy of the solution.

Due to the dependency of the solution upon the grid the techniques of grid generation are an important aspect of research developments in CFD; there are numerous publications, panels and conferences devoted to improving and easing the ability to create grids.

Although this generation deals entirely with exact numerical procedures, presently the production of suitable grids easily and quickly is still somewhat of an art form. Experience and expert knowledge of the numerical techniques involved can aid the engineers ability to create a suitable grid, as can a priori knowledge of the flow's physical behaviour. It is however extremely beneficial to qualify and quantify the grid in terms of its ability to model the flow.

To clarify the problem the definition of quality must first of all be stated. "Quality" is the placement of the object within acceptable levels. In this case the object is the discrete mesh, and the acceptable levels are the accuracy of the final flow solution and the resources required (be they time, computing facilities or technical ability) to gain that solution. Thus a linking of the grids physical properties and the flows properties must be created. From this a direct relationship between the grids physical characteristics and the flows accuracy can be derived.

The objective of this report is to obtain a quantative method of measuring grid quality upon a grid of unstructured arbitrary polyhedrals; for a description of grid generation techniques see Rycroft and Turnock [1]. Chapter 2 reviews the current literature published upon the topic of defining grid quality, while Chapters 3 and 4 define the geometrical entities required by a mesh, governing mathematical laws and key issues in their definition. Geometrical and flow properties are defined in Chapters 5 and 6, with their effects noted in the succeding two chapters. Chapter 9 discusses the statistical properties and methods required to manipulate the data generated and Chapter 10 defines the algorithms used in the implementation of the scheme. Methods of regulating and improving grid quality are detailed in Chapter 11. Finally three test cases are listed in Chapter 12 and future developments and conclusions reached are listed in Chapter 13.

# 2 Literature review

The majority of research concerning the topic of grid quality has tended to be of a qualitative nature, rather than a quantitative nature, and in total the topic has been given little theoretical definition.

One of the earliest papers found that deals directly with geometrical grid quality is that by Haimes, Connell and Vermeersch [2]. The emphasis of the paper is the accurate and easy visualisation of overall grid quality, but also highlights the requirement that "the user have the ability to construct a grid quality measure that corresponds to the accuracy and stability of the flow solver's spatial and temporal discretisation". An unstructured tetrahedral grid is analysed, with cell aspect ratio and skew being used to measure the grids quality, and grid adaptation via flow entropy averaging is considered. There are no direct links drawn between specific grid quality levels and solution accuracy.

Ilinca et al [3], Oliger & Zhu [4] and Rassineux [5] all incorporate some form of numerical calculation of geometrical error to define the mesh quality in the presence of adaptive meshing. Ilinca et al introduce the explicit adherence to the Global Conservation Laws as being of vital importance, and states that the "role of the error estimator is to provide a measure of the grid quality ... and to provide information for mesh adaptation". The concept of the optimal grid for the numerical solution being to equalise the local error for all of the computational cells is also reiterated. Oliger & Zhu [4] developed the concept of linking stability and convergence with error estimation and meshes that are unsuitable for the given flow situation. Rassineux [5] defines mesh quality by control volume shape and size via -

$$\mathbf{Q}(tetrahedra) = \frac{\alpha\rho}{h} \tag{1}$$

where h is the longest edge length; $\rho$ is the ratio of the radius of the sphere inscribed in the element; and $\alpha$ is a coefficient to set the highest value (for an equilateral element ) at 1.

Individual qualities of the mesh are not defined. A size limiter is placed using the average mesh density. A number of examples are detailed, with optimised meshes. The user specifies the lowest allowable quality value (**Q**) as well as a maximum to minimum edge ratio, and the algorithm defines a suitable mesh.

# 3 Grid Definition

The discretisation of the physical domain into a grid is a geometrictransformation. The methods used and the solutions obtained are entirely geometrical entities, thus any measurement of this grid must be in terms of geometric definitions. The base entities of a mesh, in order of hierarchy, are nodes, edges, faces and cells. That is, the most basic entity is a node. A series of nodes together can create edges; a series of edges defines a face; and a series of faces defines a cell. Figure 1 shows these four entities Solutions are limited to a finite volume of space whose extents are defined by specified bounding surfaces. It is within these surfaces that the grid needs to be created.

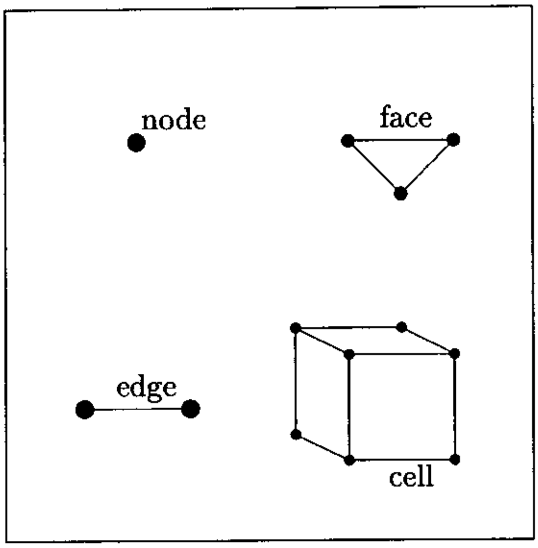

Figure 1: The basic geometrical entities

The majority of CFD solvers operate from a Eulerian point of view, but a number operate within a Lagrangian framework or a joint Lagrangian-Eulerian scheme[1]. The result of such schemes is a velocity associated with the mesh entities. It is important to note that for a numerical solution to be obtained the mesh must adhere to the Global Conservation Law (GCL), which states that -

$$\frac{d}{dt}\int_V dV = \int_S W_S \; dS \tag{2}$$

V - cell volume W - boundary velocity S - cell face area

This can be broken into two separate aspects, volume conservation and surface area conservation. The volume conservation law (VCL) requires that the volumetric increment of a moving cell must be equal to the sum of the changes along the surfaces

[1] termed ALE - Arbitrary Lagrangian Eulerian

that enclose the volume, while the surface conservation law (SCL) states that the cell volume must be closed by its surfaces. Contravention of the surface conservation law will lead to a misrepresentation of the convective velocities and numerical violation of the volume conservation will produce extra sources or sinks in the physical media (Hindman [6], Demirdzic & Perić [7]). As a result, the violation of the GCL places severe restrictions on numerical solvers.

# 4 Discretised Mesh

Discrete meshes are used in a large number of computational solutions to complex problems, via a wide variety of methods. The background, and numerical examples, used in this report are from a finite volume Computational Fluid Dynamics (CFD) point of view but are equally applicable to finite element methods and other fields of study. The most basic concept that must be remembered at all times is that the continuous domain has been *discretised*. This results in a lack of smoothness in the final solution; obviously as the interval between discretisations is decreased then these interface steps will decrease in size, but will be present at all times.

The other key property of the mesh that must be established, besides its characteristic control volume size, is where solution information is stored. This can be at node points or in cells. This is key in areas of high gradients, such as the highly curved surface of the nose of an aerofoil. Because of the high pressure gradients, the domain properties will be noticeably different at the node point on the surface from the adjoining cell. Therefore the clear benefit of node centred meshes is a more accurate representation of the domain at its boundaries, but they tend to have the disadvantages of requiring more complex numerical algorithms and often the requirement for greater and more complex information to uniquely define the grid.

Whether a node or cell centred mesh is utilised, it will have areas of poor definition when modelling a realistic geometry. These areas tend to be singularities, areas of high boundary curvature, or the intersection of two or more bodies. A mathematical singularity is defined as an occurrence of division by zero, which manifests itself in grid generation as one (or more) of the edges or faces of a cell having zero physical size. This would occur for example at the tip of a three dimensional foil. The result of singularities, unless carefully monitored, can be sources and/or sinks which imply a non-zero divergence value for the domain and hence a non-physical situation. High boundary curvature, as already stated, leads to incorrect interpolation and extrapolation of results, hence lowering the accuracy of the solution. The intersection of multiple geometries often leads to unstable solutions. The presence of sources/sinks tend also to cause the problem to become numerically unstable. The mathematical reasons for these forms of behaviour shall be detailed later.

The final topic of primary interest in a discretised mesh is whether the mesh uses a structured or unstructured format. A structured mesh uses control volumes of a regular shape in a set pattern such that there are rows of geometrical entities with constant co-ordinate values $\xi, \eta and \zeta$, and - more importantly - a given family of entities do not intersect. That is, lines of constant $\zeta$ do not cross, lines of constant $\xi$ do not cross etc. An unstructured mesh does not adhere to these rules. The finite element and finite volume methods do not demand a uniform grid; control volumes can be of an arbitrary shape, with geometrical entities not corresponding to a constant co-ordinate value. This allows for maximum flexibility in matching mesh cells with the boundary surfaces and for putting cells where they are required. It should be noted that tetrahedral control volumes are normally associated with unstructured meshes; while such volumes do require an unstructured approach, the reciprocal statement is not true - unstructured meshes can as easily consist of hexahedral volumes. The only requisit of unstructured-ness is an arbitray (but unique) identification scheme.

While there can be many reasons for the numerical failure of a solution, the cause of problems can often be traced back to the geometrical mesh. There are precise geometrical properties associated with any mesh and they can, in a converse manner, be used to predict whether problems in the numerical algorithm are to be expected.

# 5 Geometrical Grid Properties

## 5.1 Face properties

A face can be specified to be an enclosed surface, in three dimensional space, defined by an arbitrary number of edges, which each are in turn the shortest vectors between two unique nodes.

Due to the arbitrary number of edges, and their arbitrary direction, they are not necessarily of the same length, nor do they necessarily lie within the same plane.

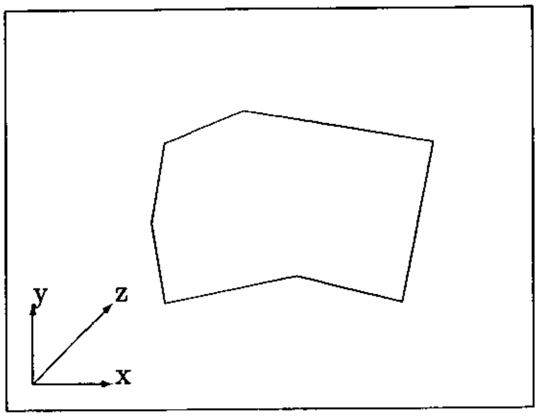

Figure 2: An arbitrary interface, defined by edges

The only stipulations required by the GCL are that the edges fully enclose the surface, and that the edges do not intersect except at their limit (that is they do not twist over each other).

Therefore there can be deviations in the planarity and the orthogonality of the edges, as well as a variation of surface area and aspect ratio. All four of these properties have a bearing upon the numerical solution gained from the numerical mesh.

**Face area** is the integration of the faces width along its length.

**Face aspect ratio** is the ratio of one face dimension to another in an orthogonal framework.

**Face deviation from planarity** is relevant only in non-triangular faces. A plane can be uniquely defined by two vectors, which, if originating from a unique point in space, also uniquely define a triangle ( the third side joining the free ends of the vectors). A surface with an arbitrary number of sides can be split into a set number of triangles, $n$, using a regular stencil of nodes defining the face . As a result $n$ planes can be defined, not necessarily identical. An averaged value of the orientation of these planes can be utilised to define the plane of the surface, and the variation of the triangular sub-element normals' used to define the deviation.

**Face skew** is the deviation of the surface from a symmetrical shape. The easiest example is that of a quadrilateral face; due to the arbitrary nature of the edge vectors, adjoining edges are not necessarily orthogonal to each other (whatever the plane). A more generalised approach to this geometrical problem, that is applicable to an arbitrarily edged face, uses the edges' normal and the face centroid. If the normal (or negative of the normal, whichever points towards the face centroid) is taken from the edge centre, the skew is the scalar distance by which the projected normal misses the face centroid by. In the case of an orthogonal quadrilateral the distance would be zero, as would be the case for an equilateral triangle, and indeed any symmetrical face. The variation in this distance can be used as a measure of asymmetry of the face.

Methods of defining these properties, and exact mathematical definitions, are listed in Chapter 10.

## 5.2 Cell properties

A cell, or a control volume (CV) as they are more commonly referred to in mesh generation, is a three dimensional body enclosed by an arbitrary number of faces.

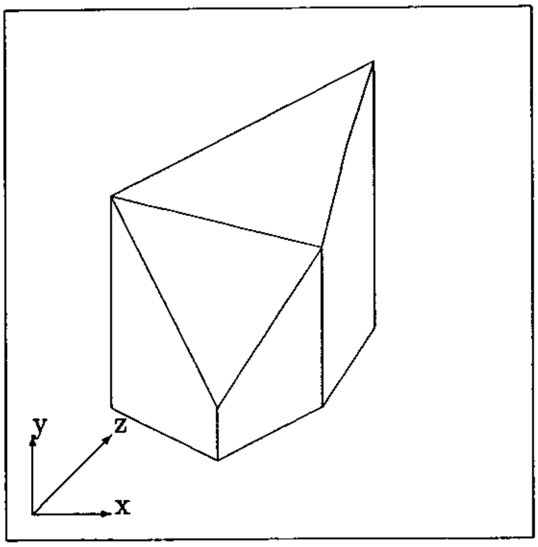

Figure 3: An arbitrary cell, defined by faces

The Global Conservation Law (2), in particular the SCL, must be adhered to. Each and every face defining a side of the volume will have the properties defined in the last section. Another key point worth noting is that the faces are directional. Because the face has vector defined edges it has a uniquely defined plane of existence, which has a unique normal vector. This face normal defines a 'positive' and 'negative' side to the face.

**Cell volume** is the integration of the cross sectional area along the length of the three dimensional entity.

**Ratio of adjacent cell volumes** is merely the arithmetical ratio of the enclosed volumes of adjacent control entities. For each cell the mean of all the ratios is taken.

**Cell twist** is akin to the skew of a face. For a hexahedral cell it is the variation of the cells' edge's orthogonality from a localised framework, while more generally it can be thought of as the scalar distance by which the projected normal of a face (from its centroid) misses the cells volumetric centroid.

**Solid angles** are the segments of a sphere subtended by the joining of two or more faces of the volume. Thus the joining of three mutually orthogonal faces will create a solid angle of $\frac{1}{6}\pi r^3$, and if a unitary radius is assumed then the value is $\frac{\pi}{6}$.

As with the facial properties, methods of defining these properties are listed in Chapter 10.

# 6 Flow Solution Properties

## 6.1 Fluid primative properties

The fluid dynamics of any given situation are defined by the three fundemental governing equations -

- Mass is conserved
- Newton's second Law, $\mathbf{F} = \mathbf{ma}$
- Energy is conserved

That is, the continuity, momentum and energy equations. These conservative equations in turn can be defined by the fluids primative variables [1] -

- Density, $\rho$
- Pressure, P
- Velocity, $\vec{V}$

**Pressure** is the magnitude of the force exerted on a surface due to particles striking it. This force is equal to the time rate of change of the momentum of the rebounding molecules per unit area.

**Density** of a fluid at a point in space is the mass of the fluid contained in an incremental volume surrounding the point.

**Velocity** is the rate of change of distance of a particle with respect to time in Cartesian co-ordinate space.

The Euler equations are -

$$\frac{\partial \mathbf{U}}{\partial t} + \vec{\nabla} \cdot \vec{\mathbf{F}} = \mathbf{Q} \tag{3}$$

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{bmatrix} \quad \mathbf{f} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ \rho uH \end{bmatrix} \quad \mathbf{g} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ \rho vH \end{bmatrix} \quad \mathbf{h} = \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ \rho wH \end{bmatrix}$$

$$\mathbf{Q} = \begin{bmatrix} 0 \\ \rho\vec{F}_x \\ \rho\vec{F}_y \\ \rho\vec{F}_z \\ \rho W_f + q_H \end{bmatrix}$$

Where $F_{\vec{x}}$ is the external force, $q_H$ the heat source and $W_f$ the work performed by the external force ($W_f = \rho F_{\vec{x}} \cdot \vec{v}$ ).

[1] Please note that Euler flow is assumed; i.e. all viscous and heat conducting terms removed

## 6.2 Derived Properties

The primative properties of a fluid will vary throughout the domain of interest, in varying quantities and in a unique manner. There is some cross coupling (for example as velocity increases, pressure usually decreases), but in the main the properties are independent of each other, or non-linearly coupled.

The **gradient** of any property in a given direction is merely the rate of change of the property with respect to distance, and can be uniquely defiend for any point with in the domain.

The **divergence** is the extent to which the vector field is expanding or compressing around a point. For relatively low speed flows (i.e. incompressible flows), the continuity equation reduces to the requirement for zero divergence. Physically the divergence of a flow is the time rate of change of the volume of a moving fluid element, per unit volume [8].

The *circulation* is defined as the line integral of the velocity around any closed curve. Thus the circulation is given by -

$$-\Gamma = \oint_C \vec{V} \cdot \vec{dr} \tag{4}$$

By Stokes theorem it is clear that if the circulation of a flow is zero, then it has no rotationality.**Rotationality** (or vorticity) is the extent to which the flow is rotating about a given point [8,9].

## 6.3 Vector field definition

If the domain is thought of in terms of being a vector field, then all three of the flow properties discussed (gradients, divergence, and rotationality) can be defined by cartesian co-ordinates.

$$\begin{aligned} gradient\ of\ p &= gradp \\ &= \nabla p \\ &= \hat{\imath}\frac{\partial p}{\partial x} + \hat{\jmath}\frac{\partial p}{\partial y} + \hat{k}\frac{\partial p}{\partial z} \end{aligned} \tag{5}$$

$$\begin{aligned} field\ divergence &= div\vec{V} \\ &= \nabla \cdot \vec{V} \\ &= \left(\hat{\imath}\frac{\partial}{\partial x} + \hat{\jmath}\frac{\partial}{\partial y} + \hat{k}\frac{\partial}{\partial z}\right) \cdot \left(\hat{\imath}u + \hat{\jmath}v + \hat{k}w\right) \end{aligned} \tag{6}$$

$$
\begin{aligned}
vorticity &= curl\vec{V} \\
&= \nabla \times \vec{V} \\
&= \hat{\imath}\left(\frac{\partial w}{\partial y} - \frac{\partial v}{\partial z}\right) - \hat{\jmath}\left(\frac{\partial w}{\partial x} - \frac{\partial u}{\partial z}\right) + \hat{k}\left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right) \qquad (7)
\end{aligned}
$$

# 7 Effects Of Geometrical Properties

## 7.1 Numerical algorithm

To understand correctly the effect a given mesh will have upon a certain computational code, the nature of the underlying numerical algorithm must also be known. In partial differential equations there are obviously two or more forms of discretisation; the norm in engineering problems, both structural and fluid dynamics, is spatial and time differentiation. The solution to a system of quasi-linear partial differential equations of the first order, if hyperbolic, will be associated with propagating waves and properties of the physical system will be dominated by wave-like phenomena.

The finite differentiation used to solve these equations can be

- implicit or explicit algorithm
- joint or separate in time and space
- central, forward backward or multistage in nature
- first, second or higher order accurate

The various different schemes all, of course, have different advantages and disadvantages. A simple algorithm, known as the Lax Wendroff scheme, utilises joint spatial and time differentiation with central differencing in space and forward differencing in time. This method is robust, but is inaccurate at discontinuities, such as a backwards facing step or a sonic shock wave.

Explicit schemes are simple to create, but are always conditionally stable, i.e. the timestep size is very important. The timestep size is set to allow no more than one numerical wave form of the solution to pass through any given cell during the step and is hence closely linked to the geometrical mesh. Implicit schemes on the other hand are more stable (often unconditionally stable), but computationally more expensive in time and resources. Useful intermediate forms of time discretisation are multi-stage implicit methods, such as Jameson's method [10] or semi-implicit schemes. These allow much larger time steps than simple explicit schemes but do still require some form of stability criterion, and hence the time step and stability are dependent upon the mesh geometry.

Space discretisation can be present in two basic forms; central differencing and upwinding. Central differencing does not pay attention to the direction of propagation of information as specified by the solutions eigenvalues, whereas upwind schemes, which utilise forwards and backwards differencing, do. The result of this is that upwinding schemes capture shocks and discontinuities better. This gives better accuracy to the solution, and a more robust approach to sharp changes in the geometry in the physical domain.

The other basic aspect of the space discretisation is the 'order' of the scheme. That is, using Taylor expansion, what level of accuracy in the discretisation is used.

Higher order schemes are more accurate and are better able to cope with high curvature, but tend to be less stable.

The method of calculating the accuracy and stability of a given scheme can be seen in greater detail elsewhere, such as Hirsch [11].

## 7.2 Effects of facial geometry

Decreasing face area has the obvious effect of increasing accuracy; as noted in Section 4, there will always be a step wise linearity to the solution, but decreasing face area will nominally result in smaller control volumes and smaller 'steps' between adjacent volumes. Smaller face areas also result in smaller interface fluxes - of primary interest to Finite Volume Method (FVM) calculations. These smaller fluxes cause smaller residuals between time steps, and therefore smoother change to the domain flow.

It is vitally important to note that it is not merely the face size that is of importance; reducing the face size alone may not produce more accurate results. Indeed, it can induce the exact opposite. The aspect ratio of the face is important to consider, particularly when concentrating control volumes around a body of interest or areas of high gradients. The focusing of discretisation in one body fitted dimension only produces very elongated cells - so while the face areas have been reduced, the aspect ratio of approximately half the faces will have increased sharply. This is a major problem in node centred meshes for those control volumes adjacent to the boundaries. High aspect ratio faces (and therefore volumes) are of use in viscous flow calculations in areas where large shear forces are present, but in non-viscous applications they can cause localised 'chequer-board' effects and increased errors, particularly in lower order schemes.

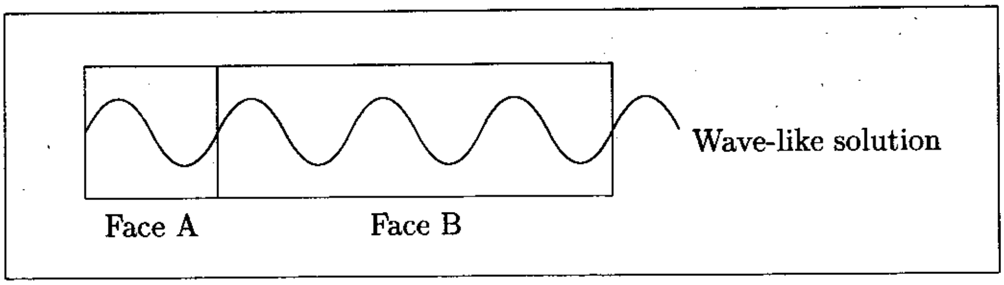

Figure 4: Wave-like solution passing across interfaces

Figure 4 diagramatically shows a wave-like solution passing through two faces, one with an aspect ratio of approximately 1.0 (face A) and one with a much higher aspect ratio (face B).

While only one wave form (approximately) can be present in the cell projected onto face A, several can be present in the cell projected onto face B. Because the timestep is commonly calculated from a cell's volume, with an averaging of the cells facial sizing, the extreme dimensioning of face B would probably not be recognised.

As a result the stability criterion for the numerical algorithm would be broken and local instabilities would be incurred, resulting to the observer as a chequer-board effect of alternatively 'hot' and 'cold' patches in the flow. Another effect of a long face (and cell) would obviously be a much higher gradient across the face, which can lead to a stretching of the underlying finite differencing scheme beyond its limits, thus lowering accuracy.

The skew of a face also destabilises a numerical scheme by producing an asymmetry to the face. This results in a variation in the distance travelled by an entity crossing the face, when travelling in a straight line. That is, the distance a wave-like form would spend tangentially crossing the face would vary. The variation in distance can be specified in wavelengths; if the number of wavelengths incurred is greater than the CFL [3] number then the scheme will become unstable. Therefore a highly skewed face will produce more extreme distances, which are unlikely to be registered in the time-step calculation, and hence more likely to cause instabilities.

Deviation from planarity usually causes a loss of accuracy, and in extreme cases will result in serious problems such as sources and sinks within the physical domain of the model. This is simply due to the fact that a non planar face will not have the same area and normal direction as one that is planar; area calculations are required to be fully developed surface definitions for domain volume calculations, yet projected planar for flux calculations. It is difficult to accurately calculate both during the flow solution process. What is a greater problem is the definition of the face normal. A planar face will have a constant normal direction across its entirety, but a non-planar surface will have a varying normal direction. The problem is in the choosing of a representative normal direction for the face as an entity. This will cause geometrical errors and hence a loss of accuracy.

## 7.3 Effects of cellular geometry

The largest single influence upon solution accuracy, from a geometric point of view, is the volume of the control volumes. The coarser the mesh, the lower the accuracy. Indeed, entire flow features can be omitted if the mesh is not fine enough in the required area. It should also be stressed at this point that a fine mesh throughout the domain is generally a very inefficient method of gaining the desired accuracy. The key issue is the relative size of the flow gradients and the cellular volume size. Stability can also be affected via the extrapolation of properties between CV's beyond that suitable for the method utilised.

The ratio of adjacent volumes can alter the final flow solution due to the common use of interpolation and extrapolation routines. When adjacent volumes are widely different checkerboard errors may form from either over- or under-prediction of properties. This is also closely linked to the type of numerical solver applied. Central differencing is unstable in most cases, so numerical diffusion is used, as it is also for higher order upwinding schemes (there termed a limiter); second and fourth

[3] Courant-Friedrichs-Levy condition defines the maximum stable timestep size

order terms are common, hence implicitly involving adjacent volumes. As with the actual volume specification, the key aspect is the ratio of flow property gradients. Ideally these values should be as close to unity as possible through out the domain.

Control volume twist tends to de-stabilise a system by allowing misrepresentations of cellular geometry to occur in the time step calculation. Another danger of highly twisted control volumes is miscalculation of interface fluxes due to unrecognised misalignment to the flow; the face normal (a weighted average of its planar components) may be orthogonally aligned to the flow, but its extremities will not be, causing numerical errors.

The final geometrical property of a discrete volume to significantly affect the flow solution, solid angles, tend to influence accuracy and numerical error. This is due to flux calculation error normally. Very acute and/or obtuse angles result in high aspect ratio segments of control volume between the face and the control volume storage point. This in turn produces an imbalance to the numerical differencing scheme. In addition the very high aspect ratio nature of the control volume segment, as well as the highly acute/obtuse nature of the angle itself, tend to approach the limits of floating point precision, hence introducing a greater numerical error.

# 8 Effects Of Flow Solution Properties

## 8.1 Effects of extreme gradients

In areas of very high flow property change the main effect will be a loss of accuracy due to small scale variations in the flow. As a result of these high gradients another problem - particularly in higher order solutions - is 'overshooting', where the high gradient causes ripple-like oscillations. This clearly leads to localised instabilites.

Lower than average gradients on the otherhand do not cause any detrimental effects to either the accuracy or the stability of the solution, but are however a major waste of resources (CPU time and memory requiremnts in particular). These resources could be much more advantageously utilised elsewhere in the solution, such as the volumes of higher than average gradients.

This general principle of minimalising the range of gradients applies to *all* the primiative variables.

## 8.2 Flow divergence

In incompressible flows the presence of flow divergence is a sign of a major instability in the solution, but is also a serious problem in compressible flows. If divergence occurs then the mass fluxes through the mesh cell-boundaries will not cancel and hence the numerical scheme will not keep the total mass constant [11]. This type of problem can often appear at singularities within the mesh, and at the conjunction of interfaces of vastly different size.

## 8.3 Rotationality

As stated in Section 7 the ideal situation is to have all flow movement either normal or tangential to the interfaces, thus reducing the numerical error due to rounding errors. Therefore a highly rotational flow would be more suited to using a cylindrical or spherical mesh (depending on the exact nature of the flow) rather than a cartesian based mesh. That is, a 'O' type mesh would have much greater accuracy than an 'H' mesh in volumes of large vorticity. Thus the variation of rotationality, when taken in conjunction with the formation of the mesh present, will have a marked effect upon the conservation of properties during the run-time of the solver as well as the accuracy of the final solution. One possible manner of utilising the rotationality of local flow is to align the interfaces and CV's in either a predominantly spherical or cartesian manner, depending upon the level of rotationality.

# 9 Measurement of Properties

The key problem in assimilating and tabulating the geometrical grid quality data into an easily digested format is its shear quantity. A small, two dimensional numerical grid can have in the order of a few thousand control volumes, and a similar number of interfaces, while research level grids used in the solution of three dimensional problems are presently comprised of hundreds of thousands of control volumes. A vast quantity of data can be generated, without any clear conclusions being drawn.

Data visualisation has become a prominent tool in this field, highlighting 'hot' spots of poor quality. Such methods are however still very much using an absolute approach, with no concern of the relative nature of one section of a grid to another. It is now a generally accepted rule that the optimal grid for the computation of a numerical solution is the one for which the local error is the same for all the computational cells [3]. Thus it is not only areas of poor grid quality that worthy of attention; areas of high quality will be ineffective, and often result in the flow solver being inefficient. As a result, both areas of low and high geometrical grid quality should be altered for the most efficient solution.

As stated in Chapter 7, the smaller the interface areas and control volumes, the greater the accuracy of the solution. There is however a physical limit on this refinement, namely the memory capacity of the computer being utilised. This results in a cap upon the ultimate resolution available. The accuracy in areas of interest can however be improved by intelligent manipulation of the distribution of CV's, increasing their size in areas of low interest, and as a result allowing greater clustering in areas of higher interest.

Therefore the required form of measurement must be able to specify a set level of quality and mark extreme values. The obvious choice here is some form of statistical measure, incorporating averaging and deviation from the norm. Such a method will obviously be able to cope with extremely large quantities of data, spotting trends and marking areas for rectification. With the assumption of a standard form of distribution, the bandwidth of the grid quality can be easily set.

The other problem in rating a face or a cell is that each entity has a number of properties, as defined in Chapter 5, all of which affect the quality to varying extents yet the entity has to be rated by one numerical value. The solution to this has been to give the properties different weightings, and to sum these weighted properties to provide one scalar number, defining the quality of that given entity (be it a cell or an interface).

# 10 Implementation of Scheme

## 10.1 Geometrical entity quality algorithms

### 10.1.1 Interface area

An interface can be defined by a unique number of points, numbering three (3) or more. By defining a centroid of these points (henceforth denoted as the interface centroid), a series of triangular elements can be declared, joining adjacent vertices to the centroid. If these triangular elements together fully define the interface, then the sum of their areas equals the full surface area of the interface. These triangular elements are by definition planar, hence the plane of the interface can be taken to be the area weighted average of the planes of the elements. The face centroid is defined to be the average position of the nodes defining the face.

$$\begin{aligned} A_{(\Delta i)} &= \frac{1}{2}|\vec{F_cV_i} \times F_c\vec{V_{(i+1)}}| \\ A_{FACE} &= \sum_{i=0}^{n} A_{(\Delta i)} \end{aligned} \tag{8}$$

$$\vec{n}_{FACE} = \sum_{i=0}^{n} \frac{A_{(\Delta i)}}{A_{FACE}} \vec{n}_{(\Delta i)} \tag{9}$$

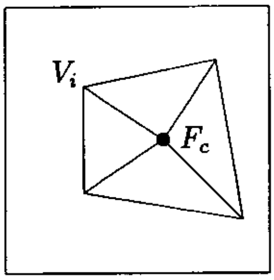

Figure 5: Mathematical definition of facial area

### 10.1.2 Interface aspect ratio

The first stage in defining the aspect ratio was to find the longest distance between nodes of the face. An orthogonal framework was defined from this edge and the facial normal, the third direction being obtained from the cross product of the other two. The maximum distance between nodes of the face in this direction was then found. The aspect ratio was defiend to be the ratio of this two distances. Figure 6 and Equation (10).

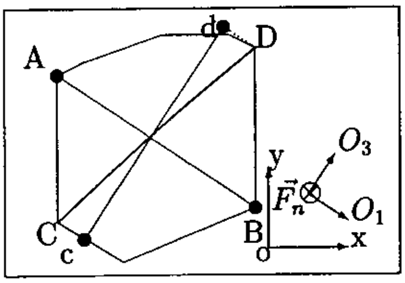

Figure 6: Definition of facial aspect ratio

$$
\begin{aligned}
\vec{O_3} &= \vec{AB} \times \vec{F_n} \\
cd &= \vec{O_3} \cdot \vec{CD} \\
AR &= \frac{|\vec{AB}|}{cd}
\end{aligned}
\tag{10}
$$

#### 10.1.3 Interface skew

Interface skew is defined as the distance by which the normal of any of the face sides, from its centre, misses the face centroid. Hence the value taken for the interface as a whole is deemed to be the average of the values for each of the sides. The face side is take to be a vector, from which the normal can be defined, and another vector is defined from the side centroid to the face centroid. The skew is defined via vector maths and simple trigonometry.

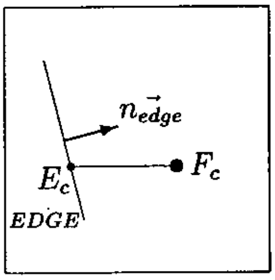

Figure 7: Mathematical definition of facial skew

$$
\begin{aligned}
\cos(\theta) &= \frac{\vec{E_cF_c} \cdot \vec{n_{edge}}}{|\vec{E_cF_c}||\vec{n_{edge}}|} \\
SKEW &= |\vec{E_cF_c}| \sin(\theta)
\end{aligned}
\tag{11}
$$

$$\textbf{DEV. PLANARITY} = \frac{1}{n}\sum_{i=0}^{n} n_{\vec{face}} \cdot \vec{F_cV_i} \qquad (12)$$

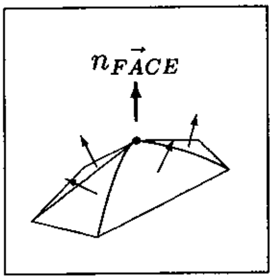

Figure 8: Definition of facial planarity

### 10.1.4 Interface deviation from planarity

The deviation from planarity of a given interface is taken as being the average of the dot product of the face normal with the vector from the face centroid to each of the face vertices.

### 10.1.5 Interface normal misalignment

This property is taken as being the dot product of the interface normal with the vector between the control volume centroid's on either side of the face. Thus an ideal value would be unity and as the value drops from this towards zero so also does the alignment of the face with the edge.

$$\vec{\mathbf{n}}\ \textbf{MISALIGNMENT} = n_{\vec{FACE}} \cdot n_{\vec{EDGE}} \qquad (13)$$

### 10.1.6 CV volume

The volume calculations are an extension of the face area calculations. Each triangular 'sub-area' is extended into a tetrahedron, with the apex being the control volume centroid, as defined by equation 14 and shown in Figure 9.

$$\begin{aligned}
\mathbf{V_{CV}} &= \sum_{i=0}^{n} V_{TETRA} \\
V_{TETRA} &= \frac{1}{3} height \cdot A_{BASE} \\
A_{BASE} &= \frac{1}{2}|\vec{AB} \times \vec{AC}| \\
height &= \vec{AD}_n \cdot n_{\mathbf{A}} \qquad (14)
\end{aligned}$$

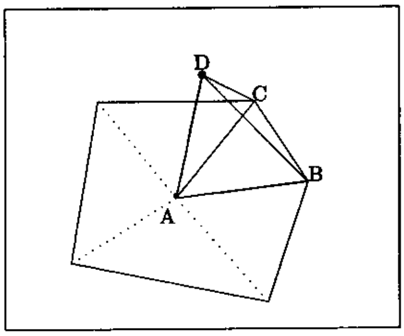

Figure 9: Definition of cellular volume

### 10.1.7 CV adjacent volume ratio

This is simply taken as the average of the ratio of the enclosed volume of the cell in question with all of those adjacent to it (this may be greater or less than unity).

### 10.1.8 CV twist

The twist of a cell is taken as being the average of the scalar distance by which the projected normal, from the centroid, of the face's miss the cells' volumetric centroid. Each face has a unique normal, and the mathematics are exactly the same as for interface skew, transposing faces for edges and volumetric centroid for area centroid.

$$\begin{aligned} \cos(\theta)_n &= \frac{\left(F_c\vec{C}V_c\right)_n \cdot (n_{F\vec{A}CE})_n}{|F_c\vec{C}V_c|_n |n_{F\vec{A}CE}|_n} \\ \textbf{TWIST} &= \frac{1}{n}\sum_{i=0}^{n} F_c\vec{C}V_{cn}\sin(\theta)_n \end{aligned} \qquad (15)$$

### 10.1.9 CV solid angles

A corner of any cell is deemed to be a place where three (3) or more faces meet. This solid angle is mathematically defined as the segment of a sphere subtended by the joining of the faces. Thus the joining of three mutually orthogonal faces will create a solid angle of $\frac{1}{6}\pi r^3$, and if a unitary radius is assumed, then the value is $\frac{1}{6}\pi$

This is mathematically attained by normalising each of the edges that meet at the corner in question. Two of these edge vectors are used to define a triangular area. The third vector edge is then used to complete a tetrahedron. This volume is then normalised with respect to a sphere; if the three edges are mutually orthogonal then the volume would be $\frac{1}{6}$. With the volume of a unitary radius sphere being $\frac{4}{3}\pi$, this means the volume between three mutually orthogonal edges is $\frac{1}{6}\pi$

$$\begin{aligned}\mathbf{A} &= \frac{1}{2}|\vec{OA}_n \times \vec{OB}_n| \\ \mathbf{V} &= \frac{1}{3}\mathbf{A}\left(\vec{OC}_n \cdot n_{\mathbf{A}}\right) \\ \textbf{SOLID ANGLE} &= \pi\mathbf{V}\end{aligned} \qquad (16)$$

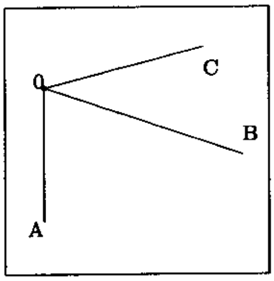

Figure 10: Definition of a solid angle

## 10.2 Flow quality assessment algorithms

Each control volume will have a uniquely defined set of properties (mass, momentum and energy), and a flow gradient can be defined for a given data storage point by averaging the gradient between the point and the adjacent data storage points for each of the required properties, as shown in Figure 11

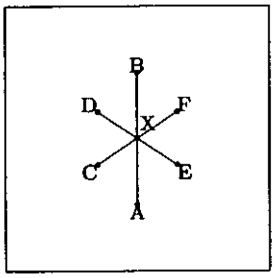

Figure 11: Flow gradient across a control volume

$$\nabla \rho_{\mathbf{x}} = \frac{1}{n_x} \sum_{i=0}^{n_x} \frac{\rho_x - \rho_i}{|\vec{xi}|} \qquad (17)$$

## 10.3 Derivation of statistical grid quality

Due to the vast quantity of data that is produced by such analysis of even a relatively small mesh, statistical properties and definitions are utalised.

### 10.3.1 Maxima and minima

To gain a concept of the range of any given variable the maxima, minima and geometric mean values were calculated for each property. Each individual data value is also stored in a 7 membered array to define the distribution of the data points for graphical review.

$$\bar{x} = \frac{1}{n}\sum_{i=0}^{n} x_i \tag{18}$$

### 10.3.2 Standard deviation

To allow the computational algorithm to define whither any given face or control volume was outwith allowable parameters the standard deviation was used.

$$\begin{aligned} & \textit{standard deviation} \\ = \; & \sigma \\ = \; & \sqrt{\frac{1}{n-1}\left(\sum_{i}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i}^{n} x_i\right)^2\right)} \end{aligned} \tag{19}$$

A given data value was deemed to be unsuitable if it was more than 2.365 $\sigma$ from $\bar{x}$. This value corresponds to a 95% population with a normal distribution curve.

| | |
|---|---|
| Deviation from planarity | 2.0 |
| Skew | 1.0 |
| Aspect ratio | 1.5 |
| Area | 0.85 |
| Misalignment of normal | 1.0 |
| Volume | 0.75 |
| Twist | 1.0 |
| Solid angles | 1.0 |
| Ratio of adjacent cells | 1.5 |
| Pressure gradient | 0.75 |
| Velocity$_x$ gradient | 0.95 |
| Velocity$_y$ gradient | 0.95 |
| Velocity$_z$ gradient | 0.95 |

Table 1: Spatial and flow quality weightings

## 10.4 Definition of scalar grid quality

The previous defined properties and methods of analysis were combined to provide an overall quality for any given mesh, as well as the ability to declare whither any given control volume is within acceptable quality limits for the given mesh.

The geometric and/or the flow properties were calculated for each face or cell and stored (the facial propties were transferred to the control volumes by averaging over the number of faces attached to the volume). Each of these values were multiplied by the weightings provided in Table 1, and then summed together to provide a total quality scalar value for the control volume. The quantifiable level of these weights cannot as yet be fully justified, but numerical experiments carried out in two dimensional flow have provided their relative qualitative levels. The total quality value for the control volumes was then statistically analysied, providing a mean, maximum, minimum and standard deviation. The mean cellular total quality value is taken as the quality of the entire grid and the standard deviation is used to determine those cells that are deemed to be unfit.

An ideal grid quality value would be zero, and as quality decreases the scalar number increases.

# 11 Methods of regulating grid quality

To improve the accuracy and/or stability of a solution by altering the geometrical mesh there are a number of methods available. The three basic areas of manipulation are

- Cellular fission/fusion
- Face manipulation
- Node movement

Each of these will affect the mesh quality, not always in an advantageous manner.

## 11.1 Cell fission/fusion

The individual control volumes can be either split or joined together with adjacent control volumes. This will clearly have a direct increase or decrease in the volume enclosed and facial areas respectively. It will also have a major influence, through judicial selection of the direction of splitting or joining, upon the aspect ratio of individual faces as well as the ratio of adjacent cellular volumes.

The major reason for cell creation and/or destruction is however to control the flow property gradients.

## 11.2 Face manipulation

Face manipulation can be in the form of centroid movement or corner node movement. Both can reduce a faces deviation from planarity but corner node movement (usually in an elliptical refinement manner) is more effective at reducing skew and twist. Side effects of such alteration will be changes in area, volume, aspect ratio and mis-alignment of the face normal with the edge normal. It is often the case that as one properties quality increases another decreases.

## 11.3 Node movement

The movement of nodes (or cell centroids in a cell centered scheme) will have the effect of radically and quickly altering the alignment of the face normals with the edge normal. It will also alter the flow property gradients, althought not as effectively as cellular creation and destruction.

All of these methods of grid adaptation, and others, are detailed in Ship Science Report 109, [12].

# 12 Test cases

To test the computational algorithm devised three typical cases were utilised. The meshs were all created using the in-house mesh generator *FLEXIMESH*, detailed in Ship Science Report 101, [1].

## 12.1 Flow over a circular arc hump

### 12.1.1 Geometric definition

The first test case is a three dimensional representation of a two dimensional problem. The domain is a simple channel of unitary height with a circular arc obstruction on its lower surface, centrally placed along the length. The arc's height is 10% of its chord, with the free stream channel height equalling the chord. Figure 12 shows a wire frame model of the domain.

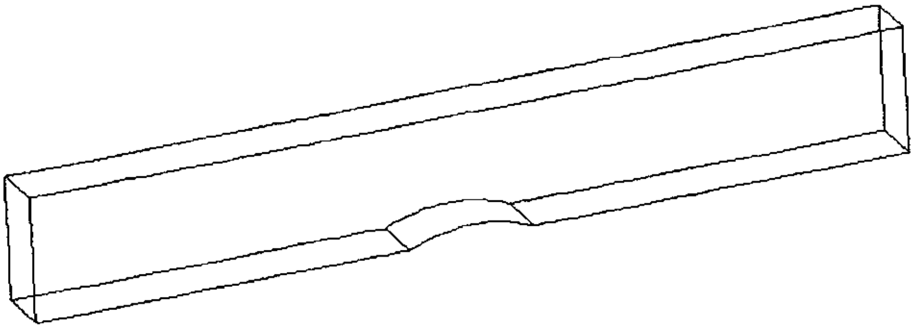

Figure 12: Wire frame of hump mesh

### 12.1.2 Variations

The first variation tested was a simple increment in the number of geometrical entities present. All block sizes and shapes were left unaltered, as were edge weightings and facial sources. Three different increments were tested, which resulted in the number of nodes defined in Table 2. The resulting meshes can be seen in Figure 13.

The second variation for the hump mesh was to take the second mesh (H2), and alter the shape of the blocks and the edge distributions to create a fourth mesh (mesh H2z). The overall number of nodes was left constant. Figure 14 shows the alterations.

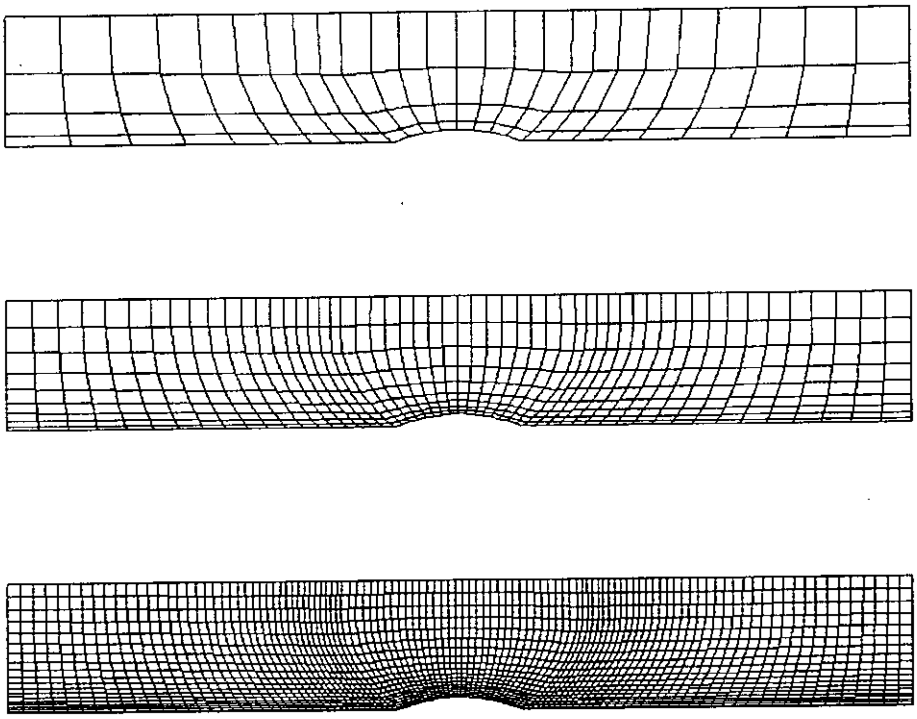

Figure 13: Hump meshes utalised to test effects of increasing number of nodes; H1, H2, & H3 respectively

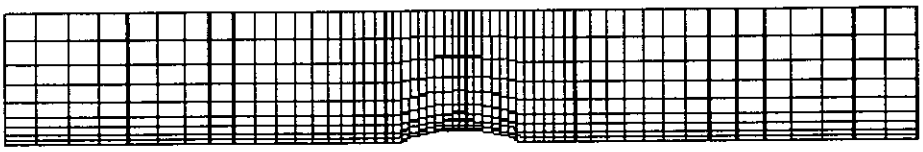

Figure 14: Mesh H2z, derived from the mesh H2, showing effects of edge distribution

| Grid | Number of nodes |
|---|---|
| mesh H1 | 405 |
| mesh H2 | 1650 |
| mesh H2z | 1650 |
| mesh H3 | 8960 |

Table 2: Number of nodes in hump mesh

### 12.1.3 Results

All four meshes were analysed in the same manner. The CPU time on a Departmental SUN SPARC Ultra 10 for the analysis is shown in Table 3, and the overall quality value (and deviation) for each of these grids is defined in Table 4. Figure 15 visually defines the variation in quality through the grid for meshes H2 and H2z. Detailed results are listed in the Appendix, with relevant graphs.

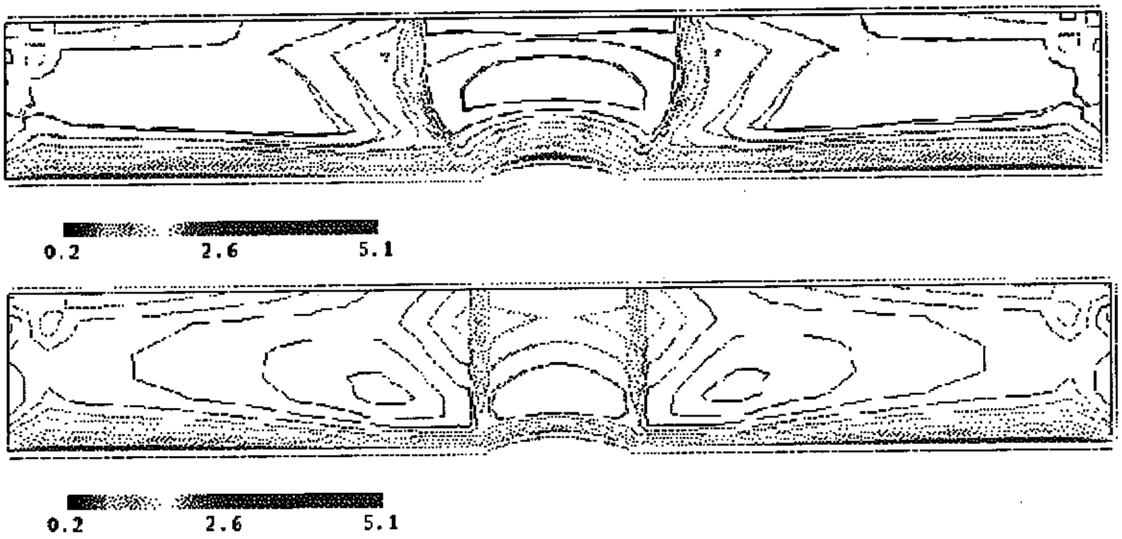

Figure 15: Variation in grid quality through the mesh (mesh H2 and H2z respectively)

As can be seen by comparision of the quality value for meshes H1, H2, and H3, the overall quality does not nessecarily increase with an increased number of control volumes, particularly in a structured grid . Any abberations are highlighted and extenuated. A case in point is the aspect ratio of faces and cells; with a given edge distribution, an increase in the number of sub-divisions merely emphasises the difference in size of the largest and smallest segment of the line, resulting in a larger spread of results and hence a larger standard deviation.

A comparision of the results between meshes H2 and H2z however does demonstrate the benificial effect of adapting face shapes and edge distributions to the overall topology. Due to the greater orthogonality of the vertical faces next to, and the faces on the hump, the overall quality is improved, and the range of values is reduced. This dooes not mean that each and every property has been improved - indeed skew has been degraded - but that the sum of the quality of the individual properties has been. This *generalised* improvement should result in a more stable and accurate solution.

## 12.2 Wigley hull

### 12.2.1 Geometric definition

The second example choosen was of the mathematical three dimensional Wigley hull form. The hull form is defined by equation 20.

$$y = \pm\frac{B}{2}\left\{1 - \left(\frac{x}{L}\right)^2\right\}\left\{1 - \left(\frac{z}{T}\right)^2\right\} \tag{20}$$

For this model $\frac{B}{L}$ was taken to be 0.1 and $\frac{T}{L}$ was defined as 0.0625. With a unitary half ships length (L), the domain taken for the model extends from $x = -2L$ to $x = 2L$, $y = 0$ to $y = 0.75L$ and from $z = 0$ to $z = 0.75L$. Figure 16 shows a wireframe of the domain.

### 12.2.2 Variations

The only variation implemented for this model was to place the model in a wave train, the individual waves having a length of 0.2 L and an amplitude of 0.01 L. Figure 17 shows the two meshes analysed. The distorted free surface can clearly be seen on the W2 mesh. Both meshes had a total of 11448 nodes defining the domain.

### 12.2.3 Results

Both meshes were analysed in the same manner as before. The CPU time on a Departmental SUN SPARC Ultra 10 for the analysis is shown in Table 3, and the overall quality value (and deviation) for each of these grids is defined in Table 4. Figure 18 visually defines the variation in quality through the grid for mesh W2. Detailed results are listed in the Appendix, with relevant graphs.

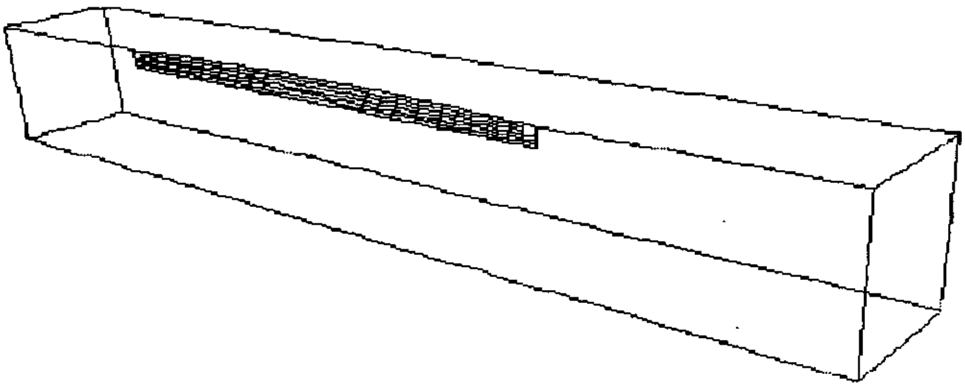

Figure 16: Wire frame of mesh around a Wigley hull form

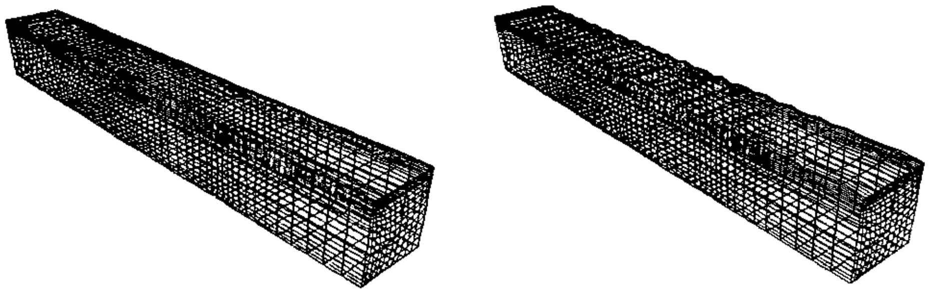

Figure 17: Meshes defined around the Wigley hullform;W1 & W2 respectively

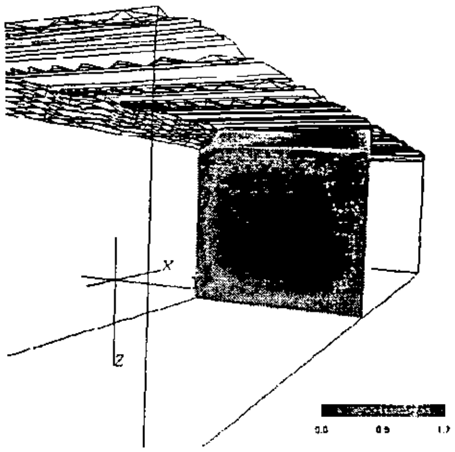

Figure 18: 2D slice of grid quality, at amidships for mesh W2

## 12.3 Truncated cylinder

### 12.3.1 Geometric definition

The third and final case study is of a truncated cylinder in a cubic domain. The cylinder, of diameter 1m, extends for a depth of 2m. The domain extends from (0,0,0) to (8,4,4) with the cylinder placed vertically at x=4, y=2. This type of geometry is typical of a SPAR bouy, as used in offshore oil extraction processes. A wireframe model can be seen in Figure 19.

### 12.3.2 Variations

As with the wigley hull mesh, only one variation was implemented, due to constraints on time and computational resources. As can be seen from Figure 20, there is a high density of node points around the cylinder, hence to show the effect of poor clustering and non-smooth variations across block boundaries the variation implemented has been in the edge weightings. Both meshes have a total of 48 355 nodes - a realistic number for a modern computational model, but mesh S1 has a trigonometrical weighting (see [1]) of weight 1.85 at a position of 1.9 for the edges in blocks in the out-lying domain. Mesh S2 has a uniform distribution on all edges.

### 12.3.3 Results

After the standard analysis, the following results were obtained. The mesh S1 has a geometrical grid quality of , while the S2 mesh, with the smoothing of cell size ratios removed, has its grid quality defined as . The lowering of the quality is also reflected in the standard deviation of the meshes, as listed in Table 4. The CPU time on a Departmental SUN SPARC Ultra 10 for the analysis is shown in Table 3, alongside the times for the other meshes. A plot of the variation of the time required to analyse a mesh against the number of nodes (i.e. the number of control volumes, as all calculations were carried out upon node centred meshes) can be seen in Figure 23. Figure 21 visually defines the volumes of unacceptable quality for mesh S1. Detailed results are listed in the Appendix, with relevant graphs.

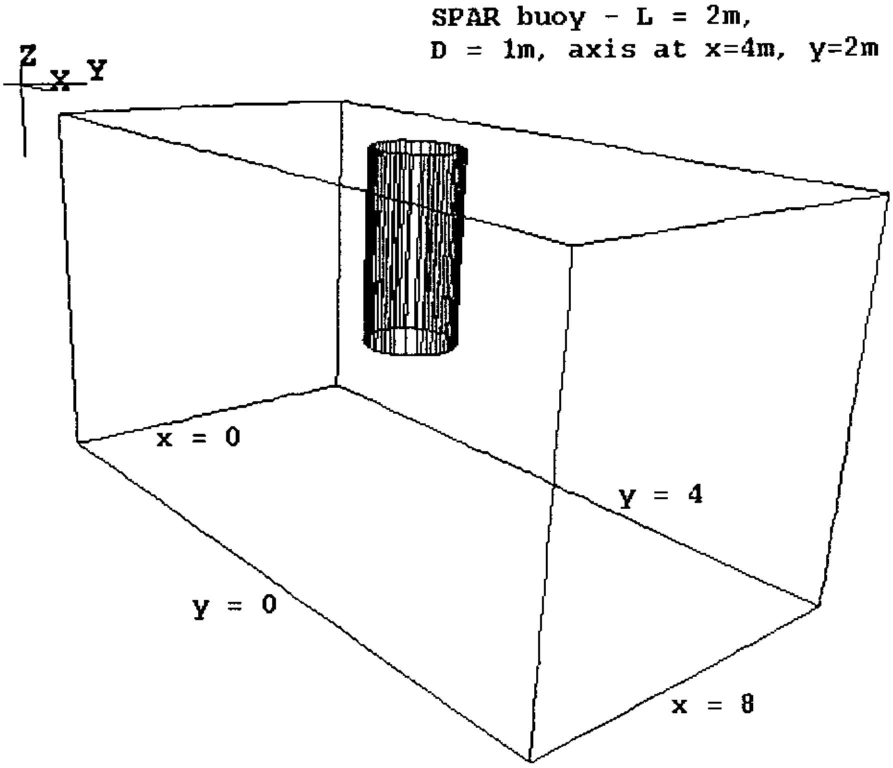

Figure 19: Wire frame of mesh around a truncated cylinder

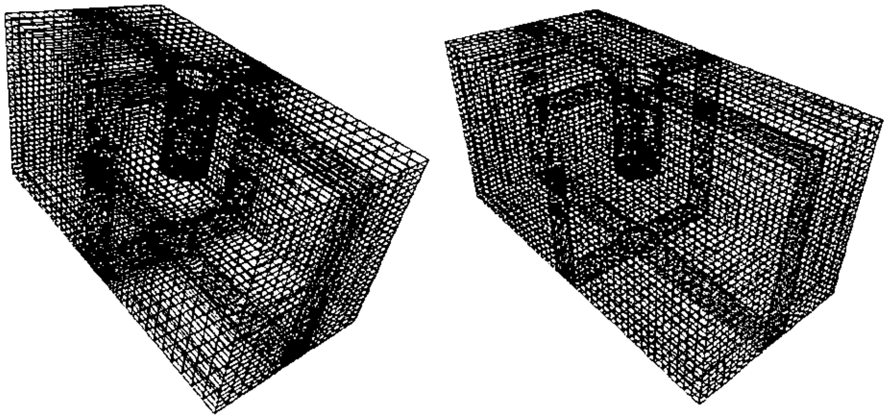

Figure 20: Meshes defined around the truncated cylinder; S1 & S2 respectively

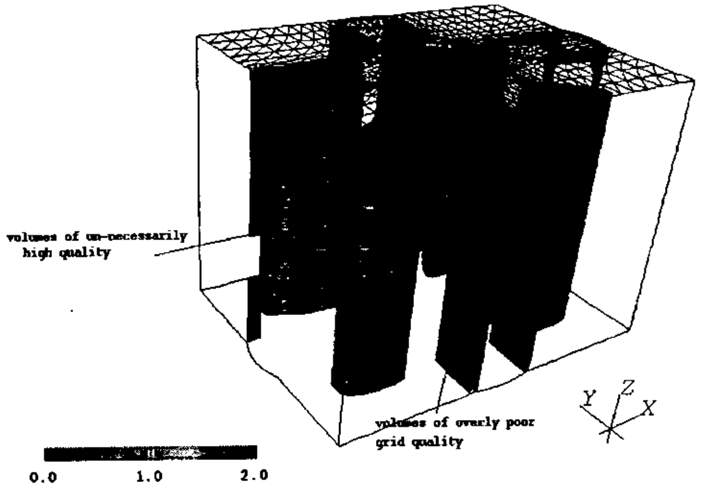
volumes of un-necessarily high quality
volumes of overly poor grid quality
0.0
1.0
2.0
Y
Z
X

Figure 21: Visual definition of volumes with unacceptable quality in mesh S1

| Grid | CPU time (secs) |
|---|---|
| mesh H1 | 0.77 |
| mesh H2 | 3.01 |
| mesh H2z | 2.99 |
| mesh H3 | 17.03 |
| mesh W1 | 22.91 |
| mesh W2 | 22.88 |
| mesh S1 | 42.47 |
| mesh S2 | 42.03 |

Table 3: CPU requirements for geometrical analysis of meshes

| Grid | Quality value | Std Deviation |
|---|---|---|
| mesh H1 | 0.962536 | 0.494514 |
| mesh H2 | 0.970718 | 0.840020 |
| mesh H2z | 0.973656 | 0.868222 |
| mesh H3 | 1.159126 | 0.974868 |
| mesh W1 | 1.196925 | 1.002978 |
| mesh W2 | 1.299051 | 1.009270 |
| mesh S1 | 0.564961 | 0.329237 |
| mesh S2 | 0.849084 | 0.446683 |

Table 4: Quantative geometrical analysis of all meshes

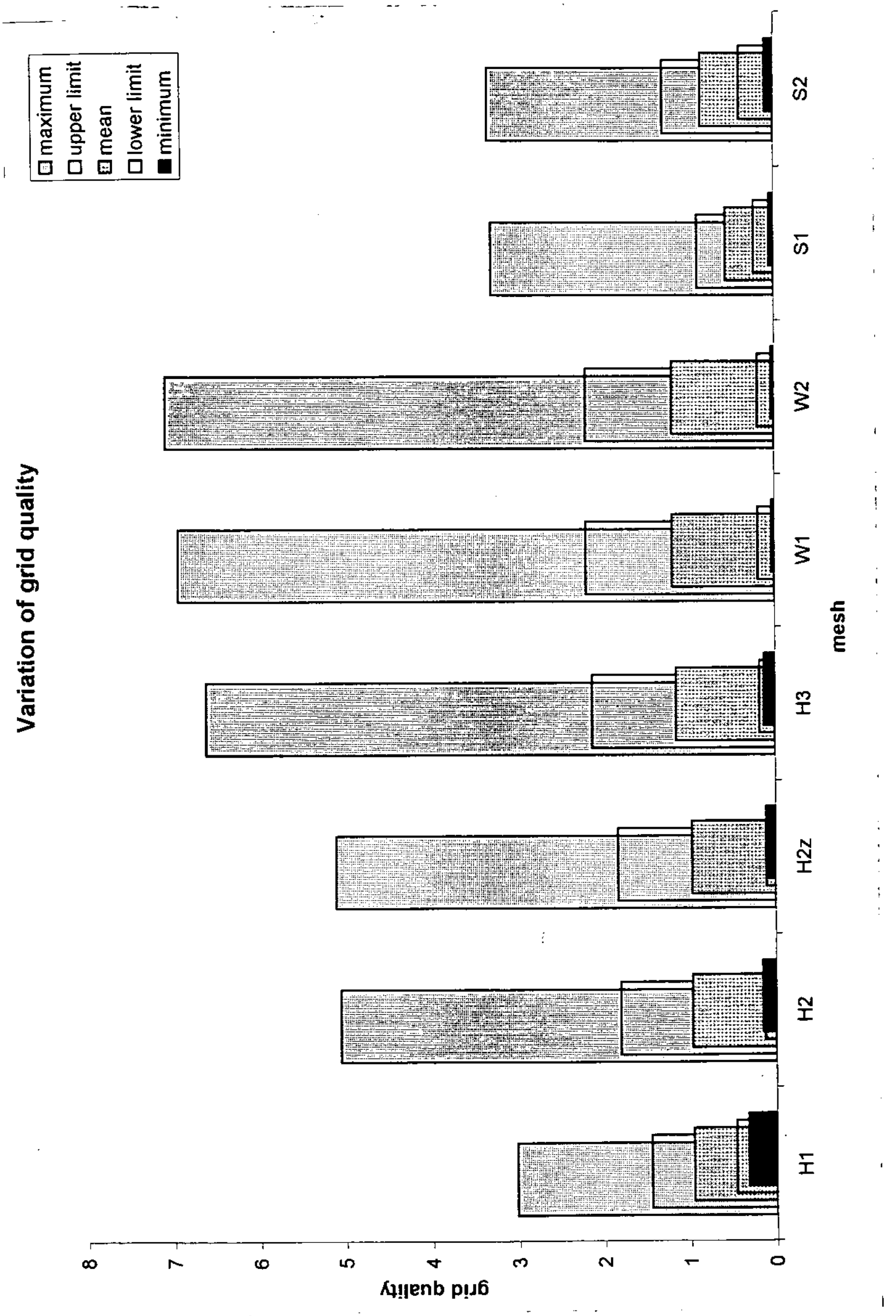

Figure 22: Relative quality of all the meshes created and analysied

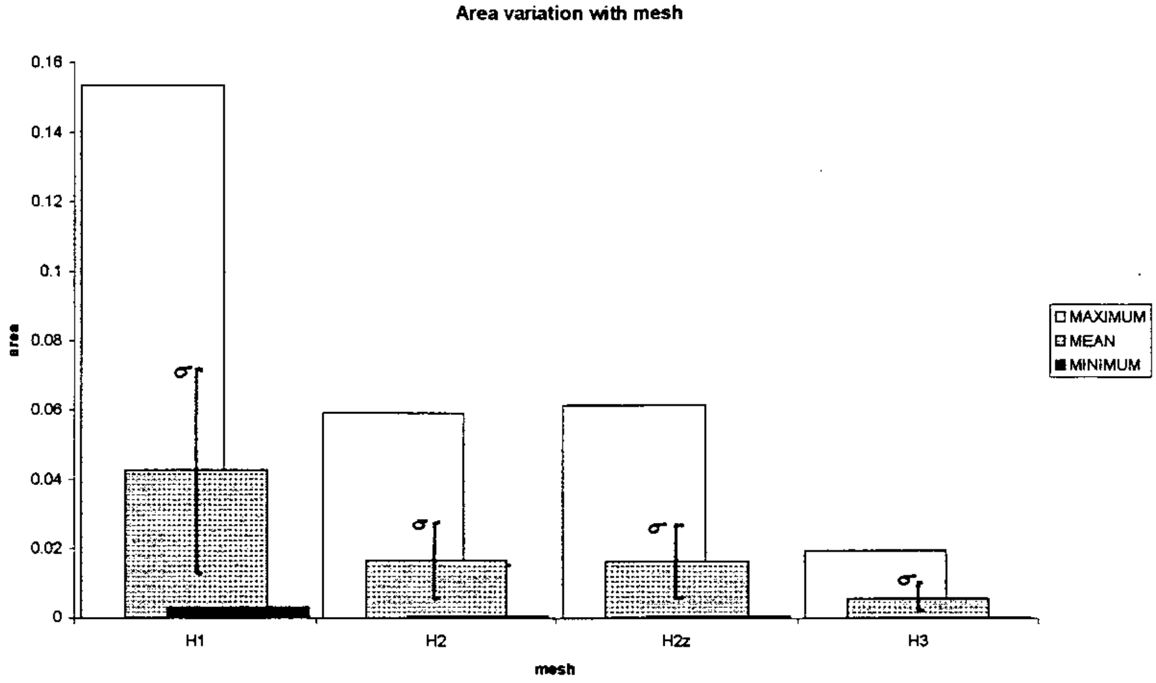

Figure 24: Variation of facial area for hump meshes

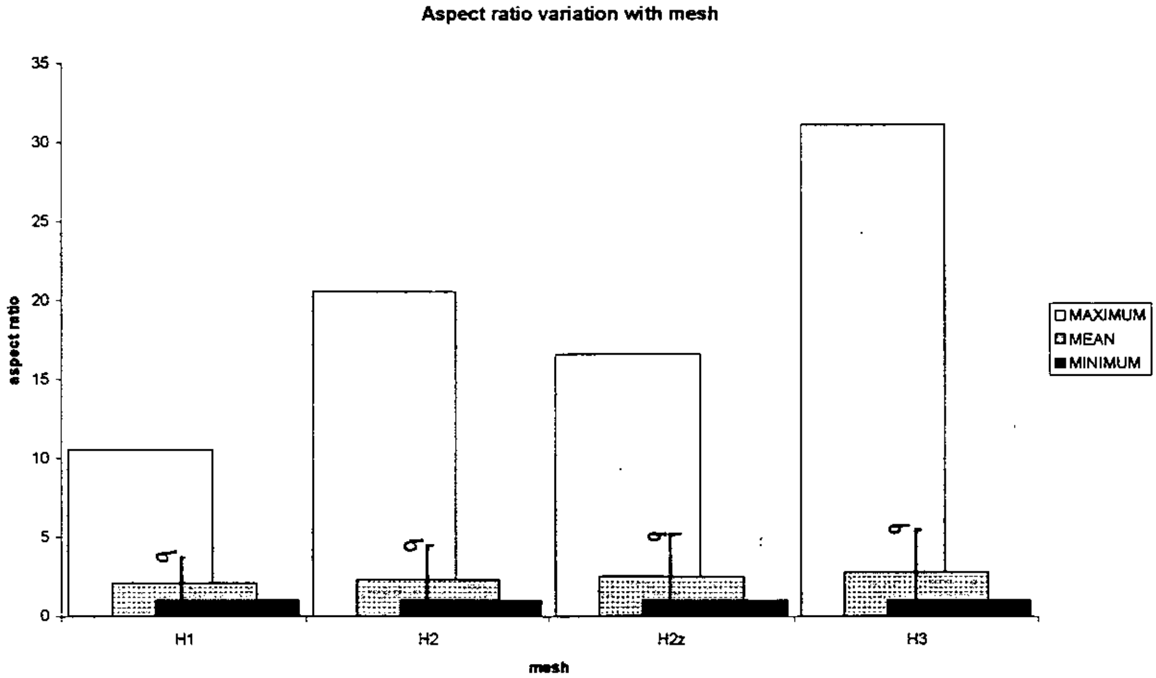

Figure 25: Variation of aspect ratio for hump meshes

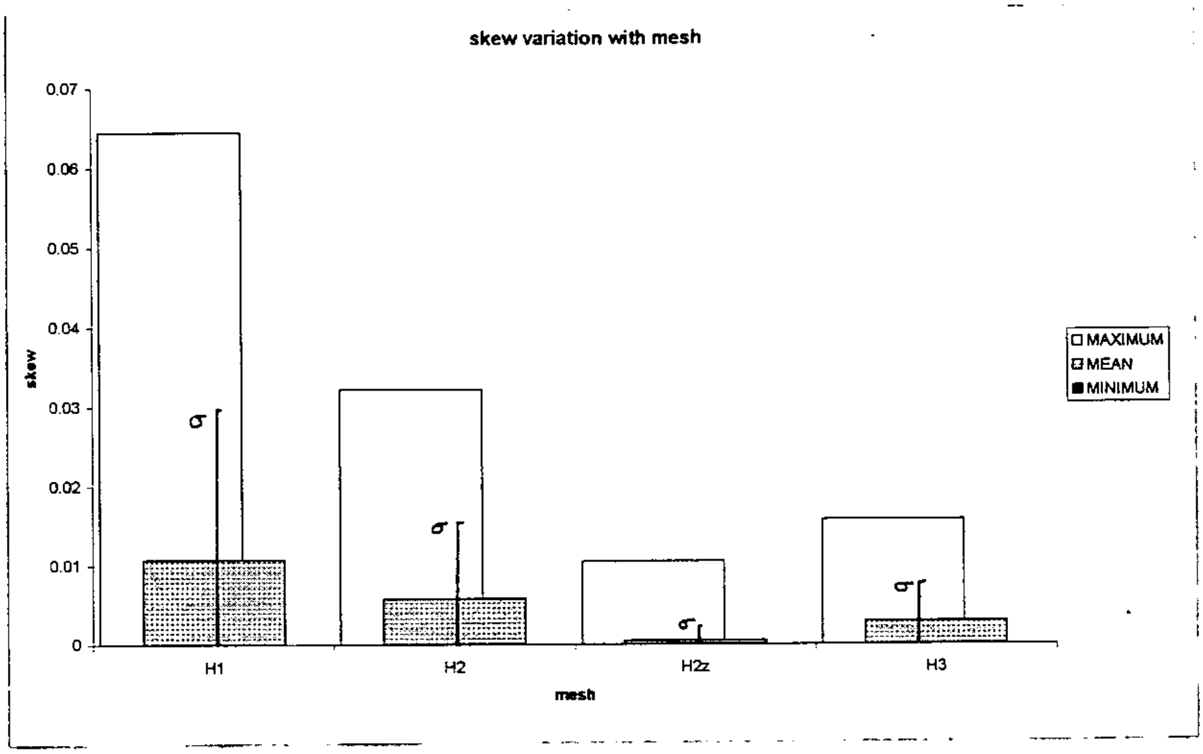

Figure 26: Variation of facial skew for hump meshes

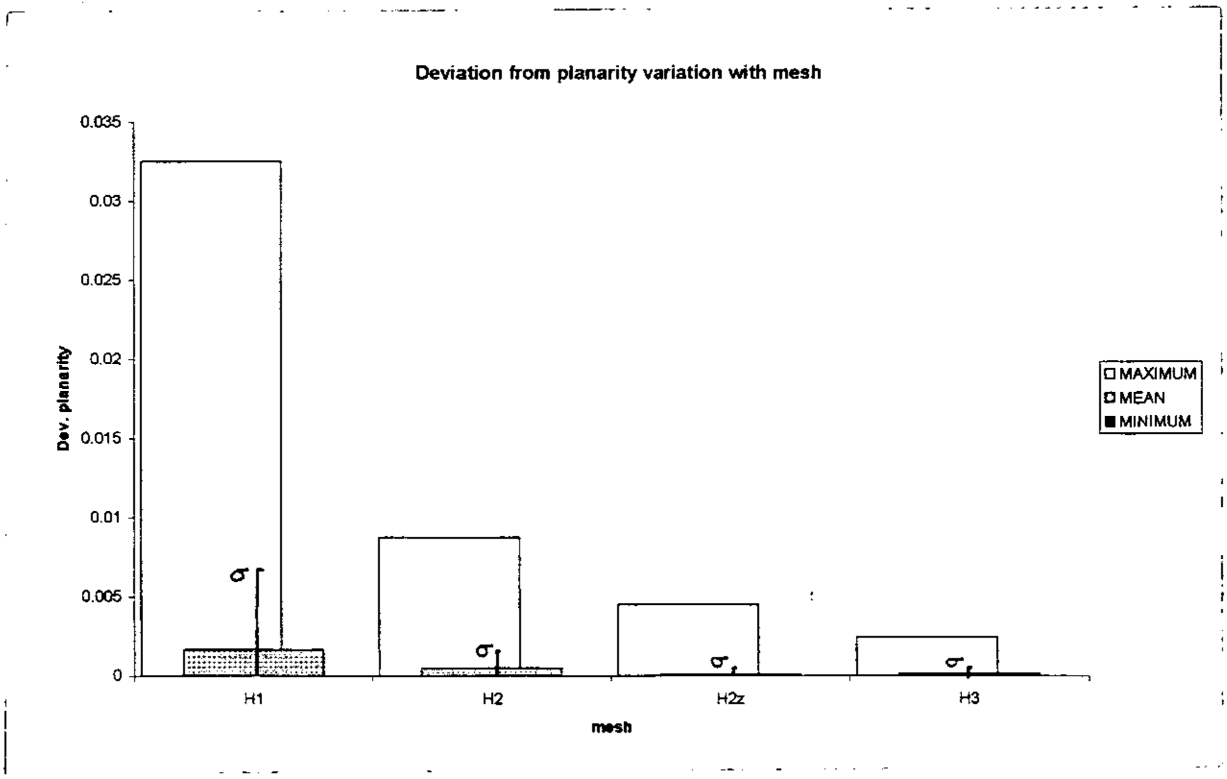

Figure 27: Variation of deviation from planarity for hump meshes

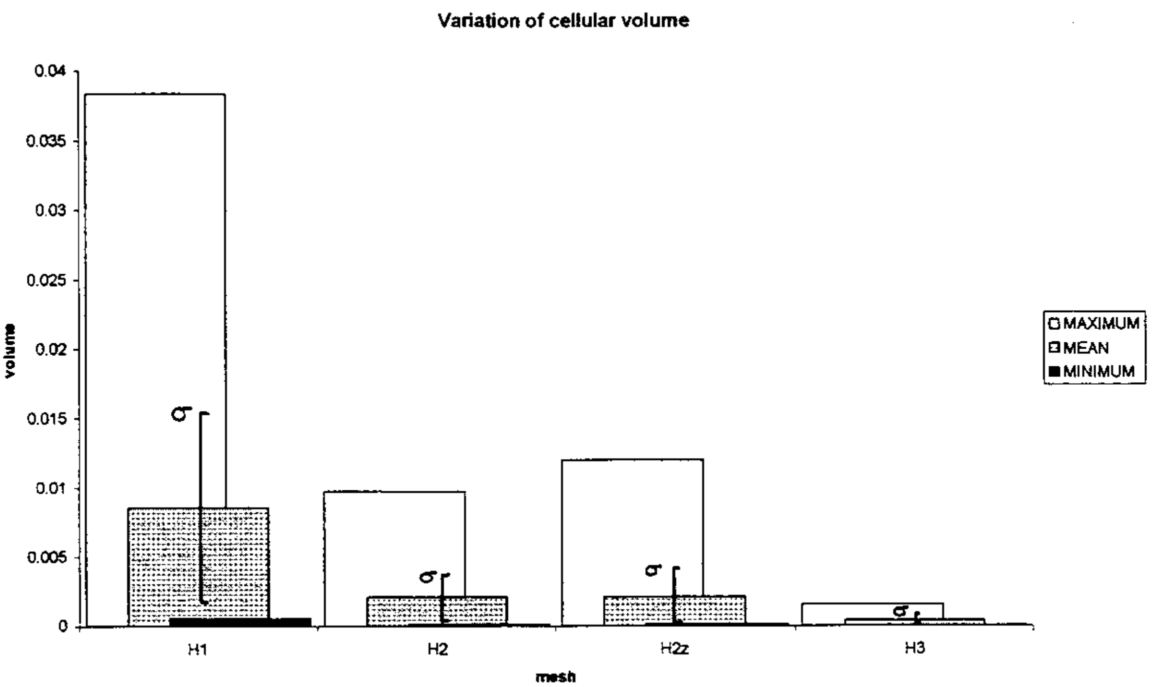

Figure 28: Variation of cellular volume for hump meshes

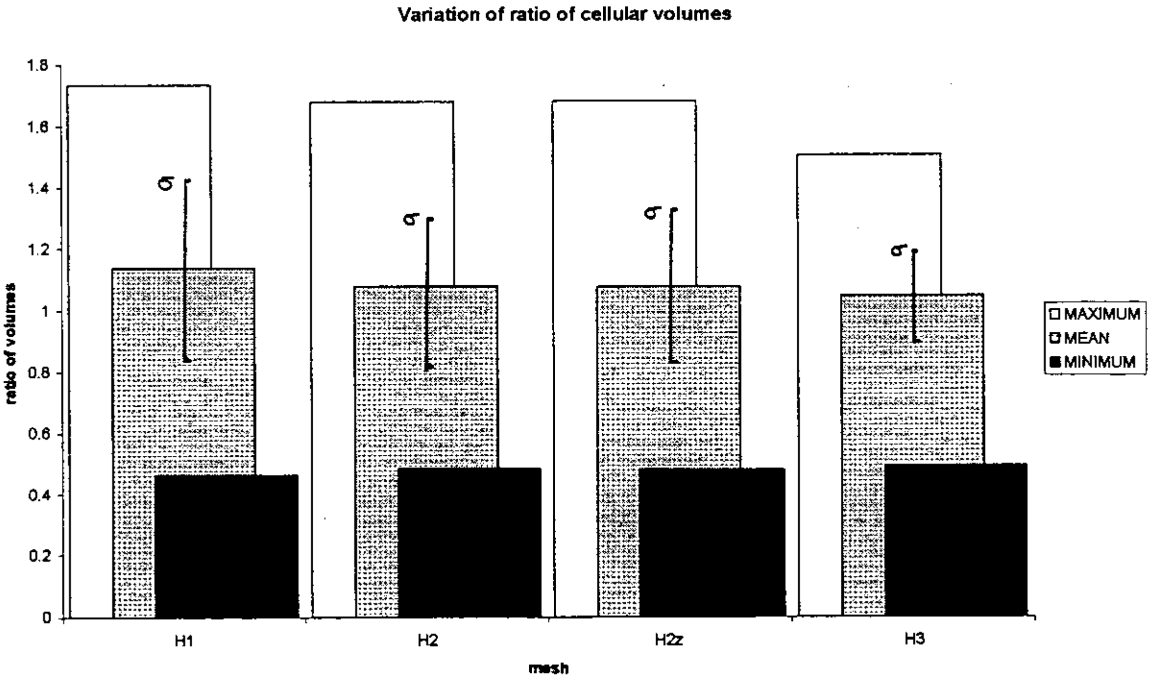

Figure 29: Variation of ratio of cellular volumes for hump meshes

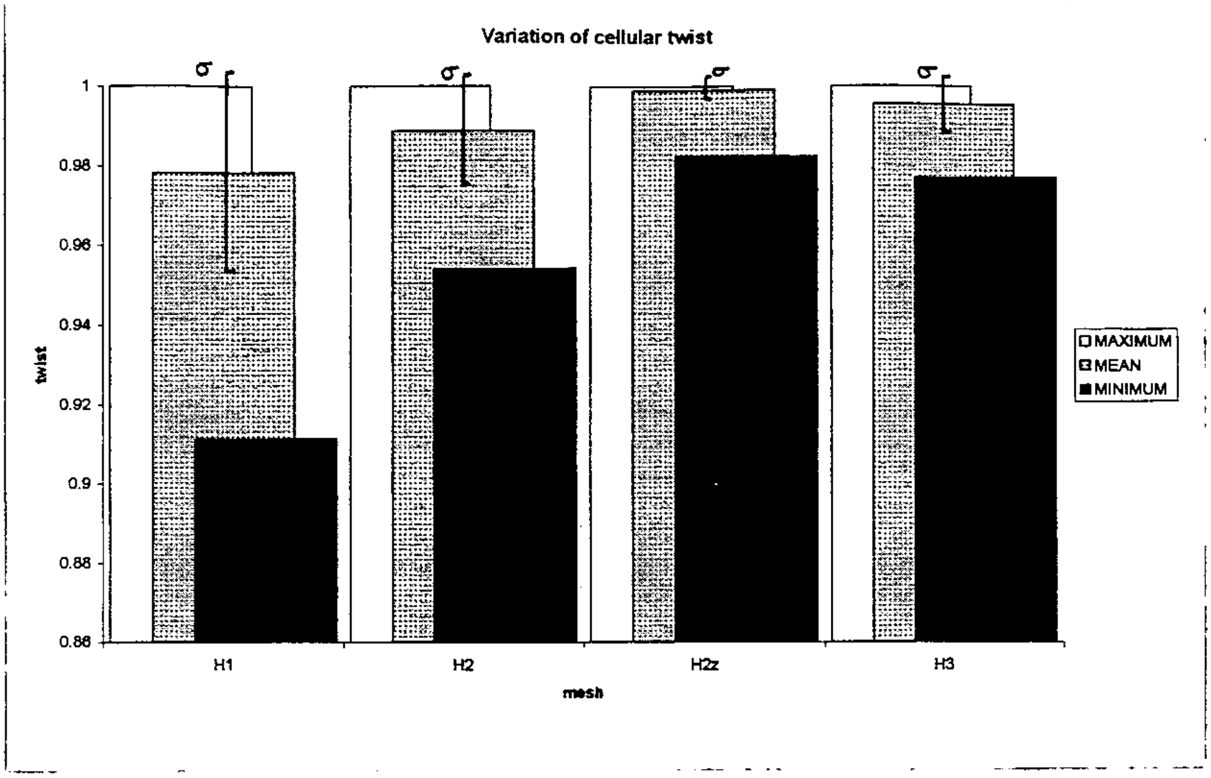

Figure 30: Variation of cellular twist for hump meshes

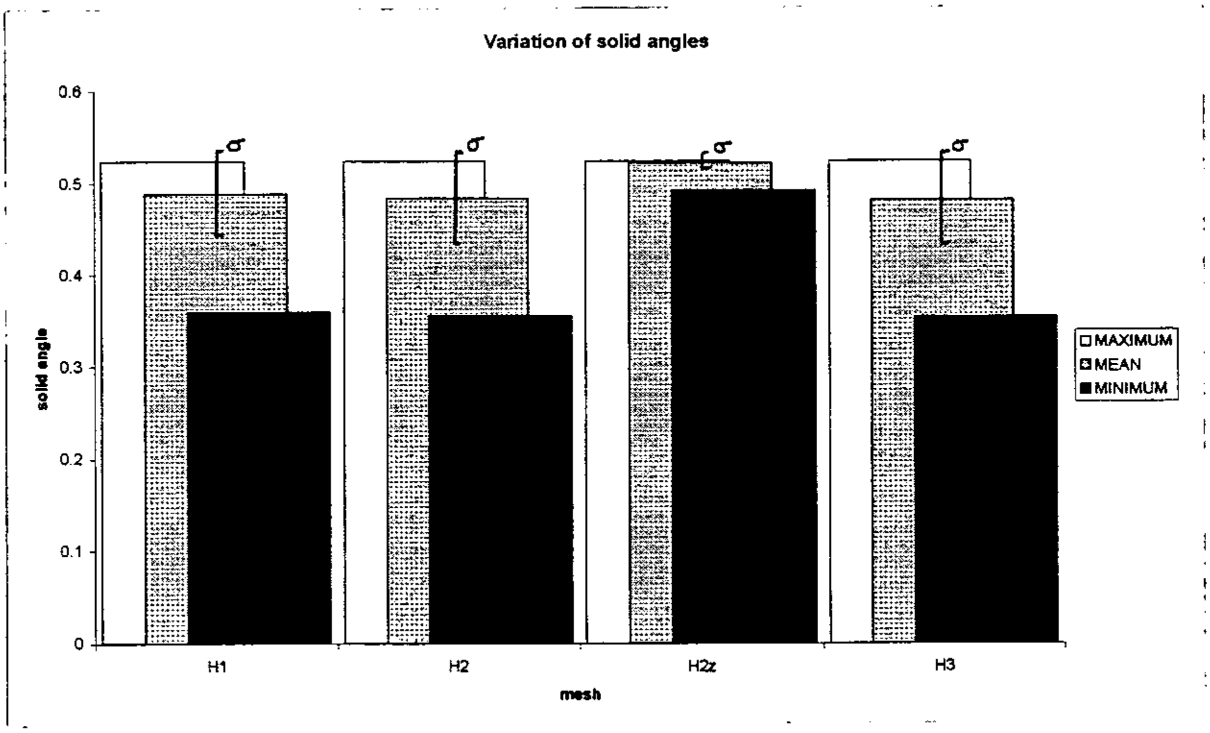

Figure 31: Variation of solid angles for hump meshes

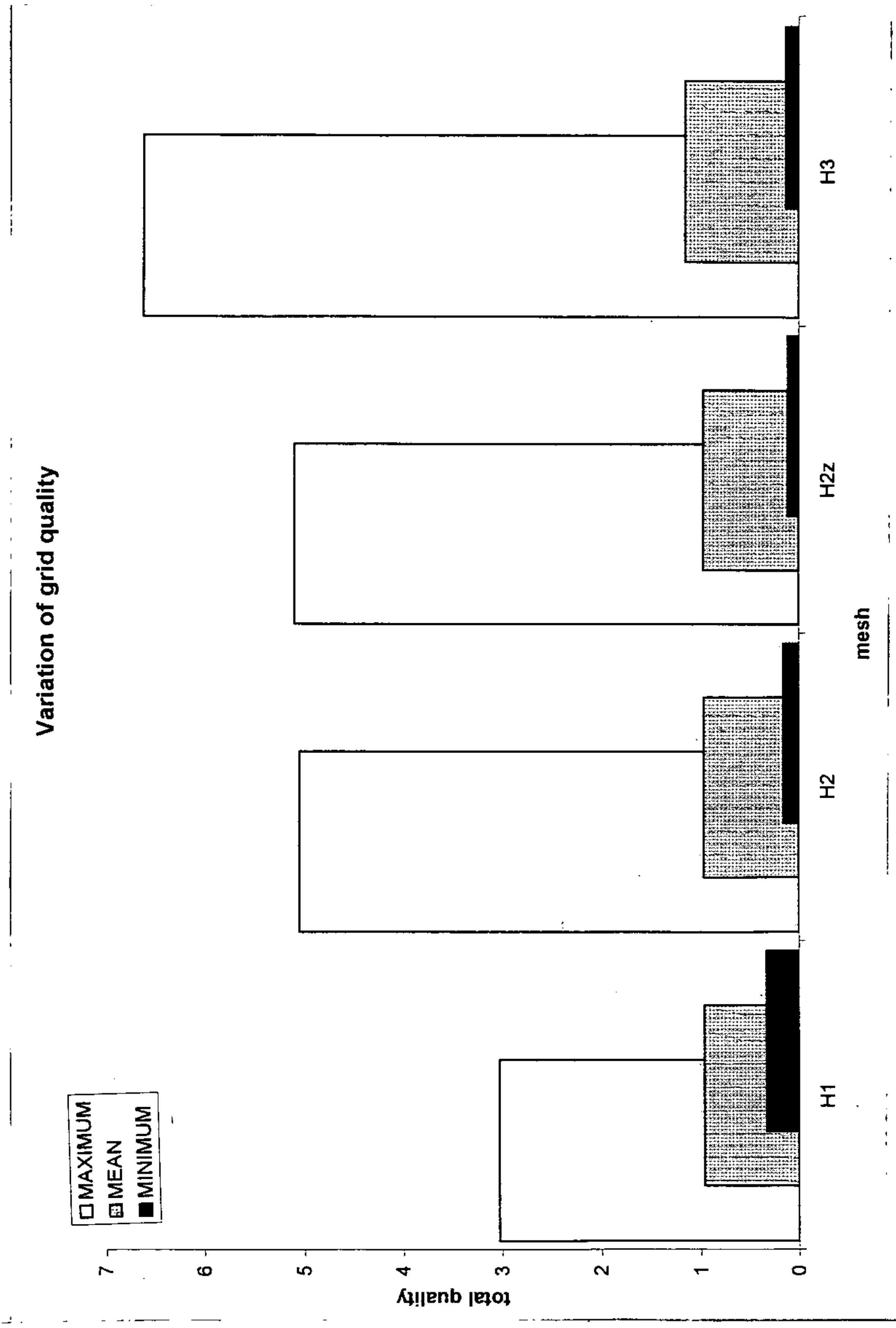

Figure 32: Variation of hump grid quality

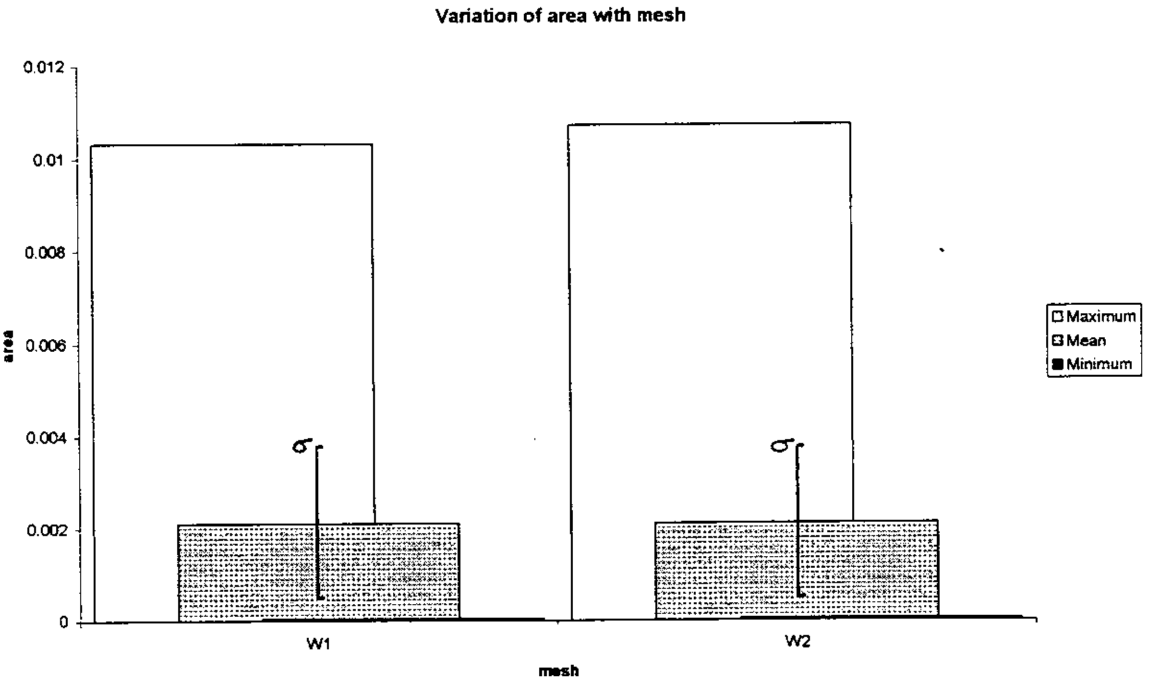

Figure 33: Variation of facial area for Wigley hullform meshes

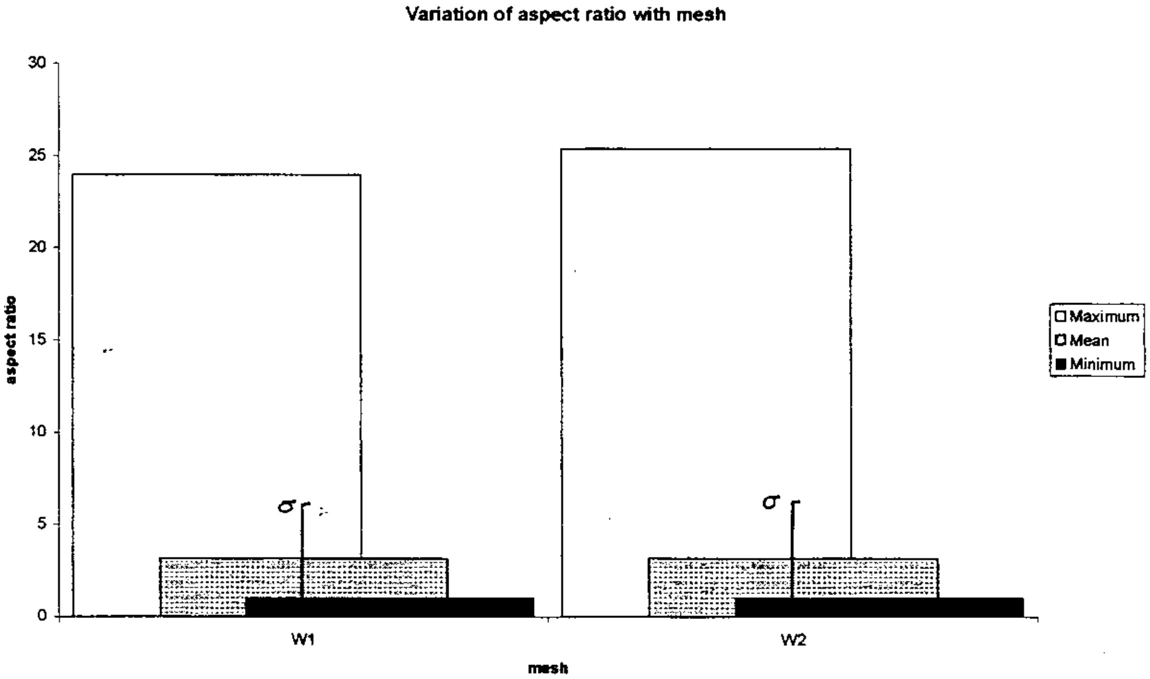

Figure 34: Variation of aspect ratio for Wigley hullform meshes

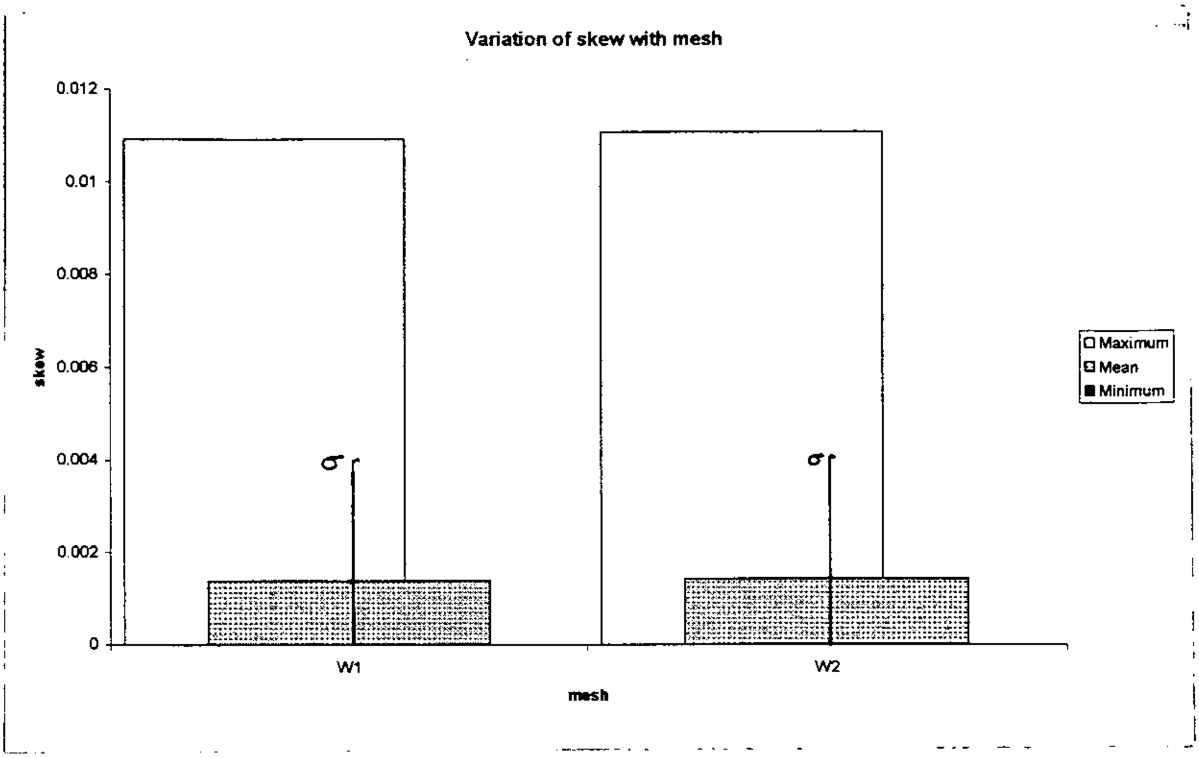

Figure 35: Variation of facial skew for Wigley hullform meshes

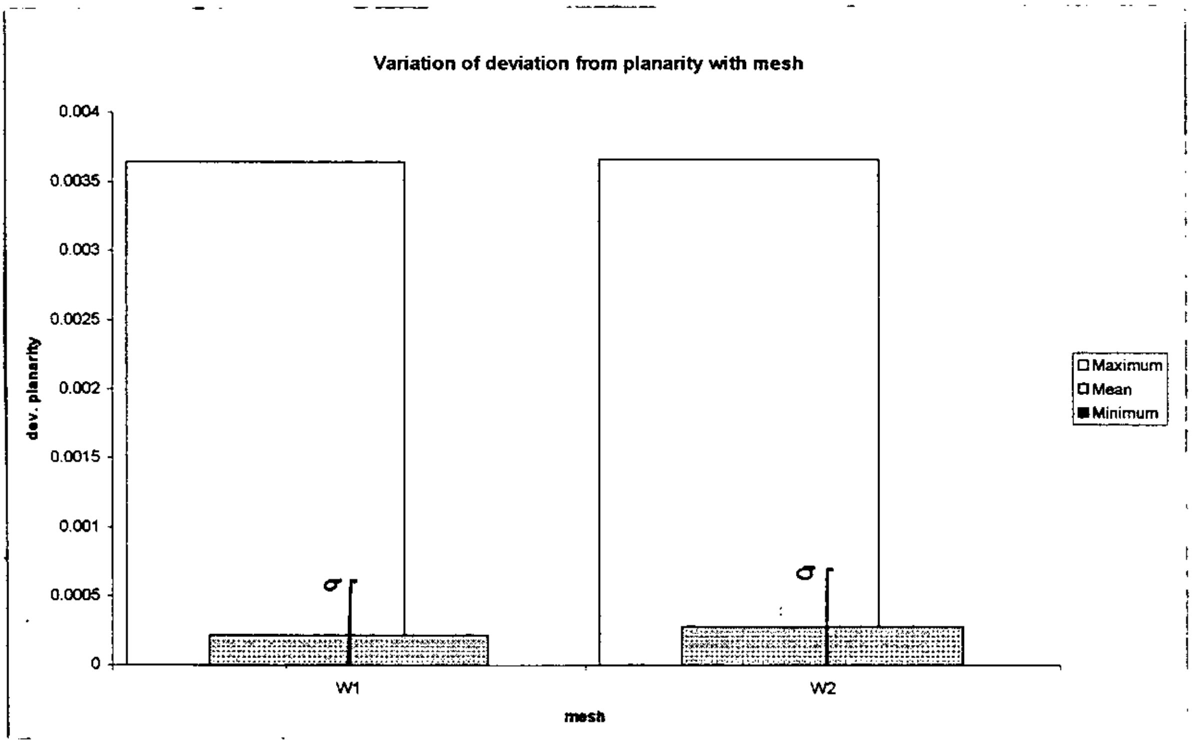

Figure 36: Variation of deviation from planarity for Wigley hullform meshes

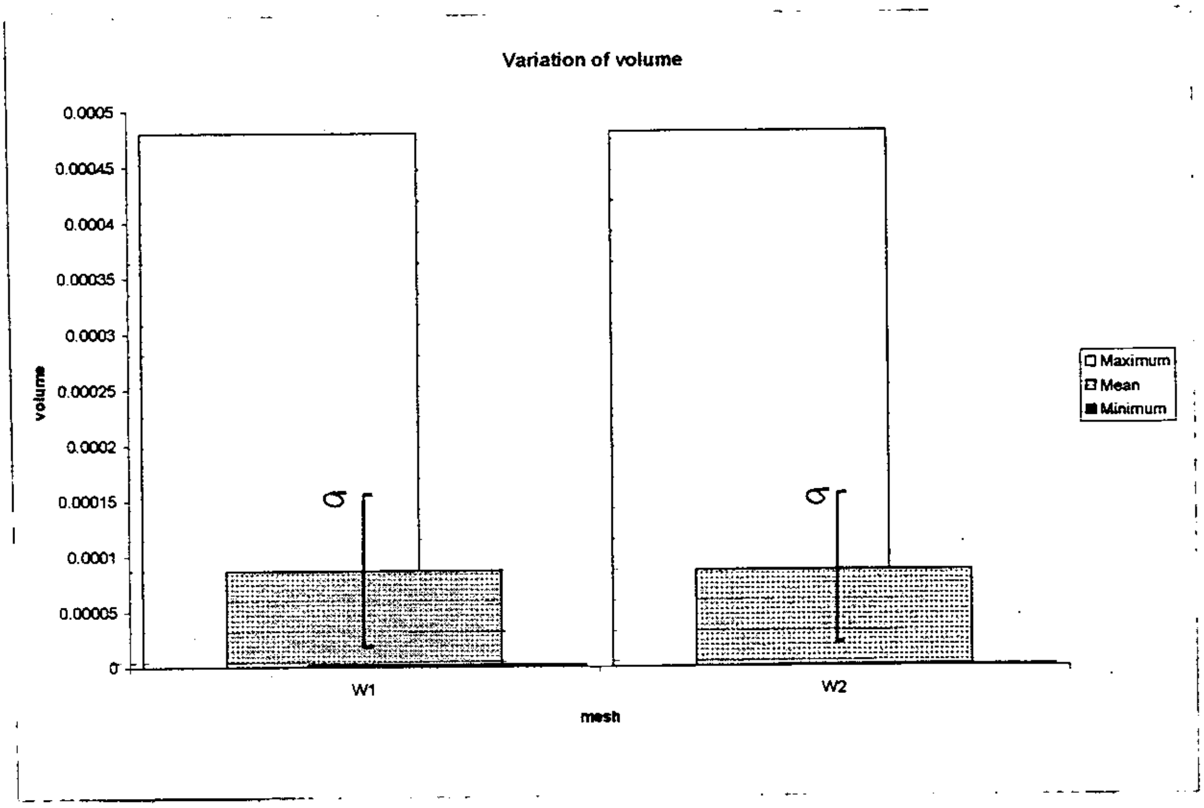

Figure 37: Variation of cellular volume for Wigley hullform meshes

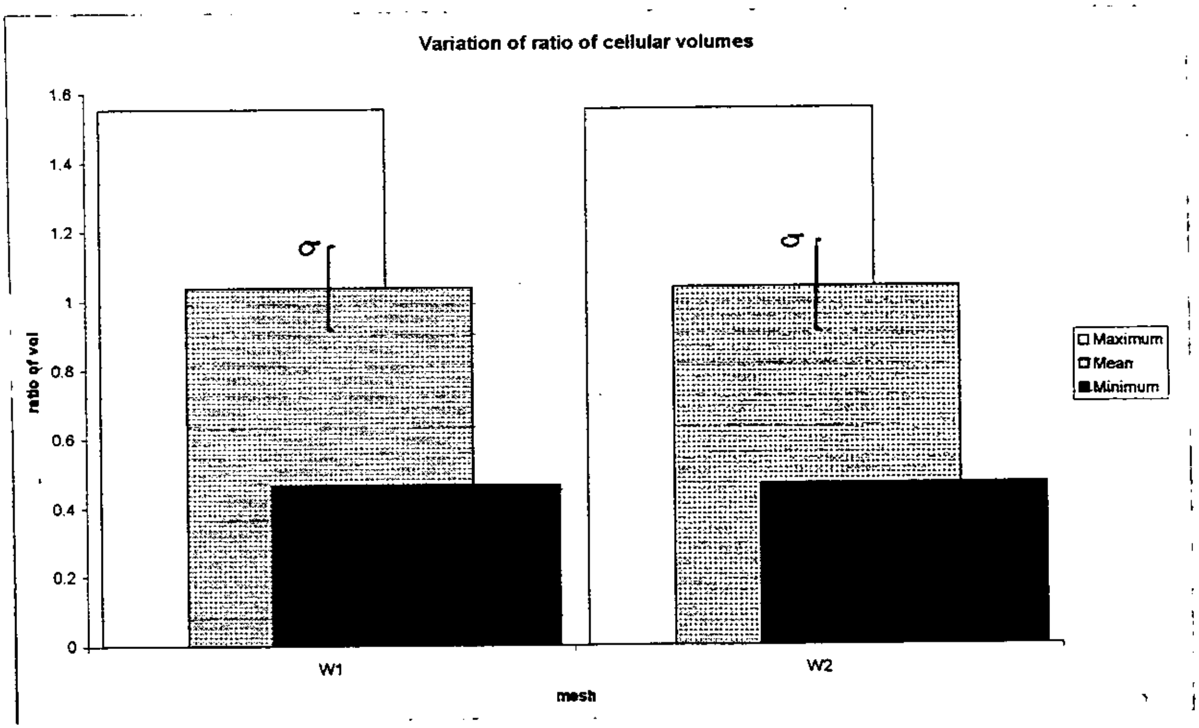

Figure 38: Variation of ratio of cellular volumes for Wigley hullform meshes

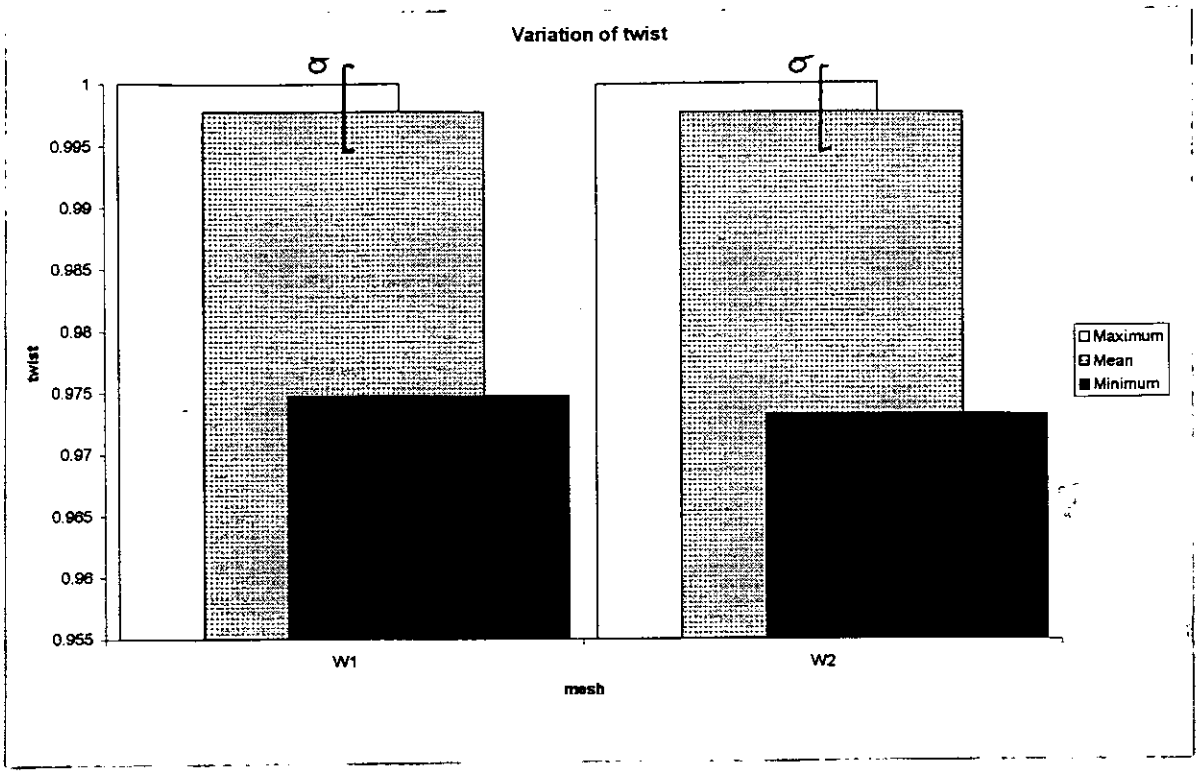

Figure 39: Variation of cellular twist for Wigley hullform meshes

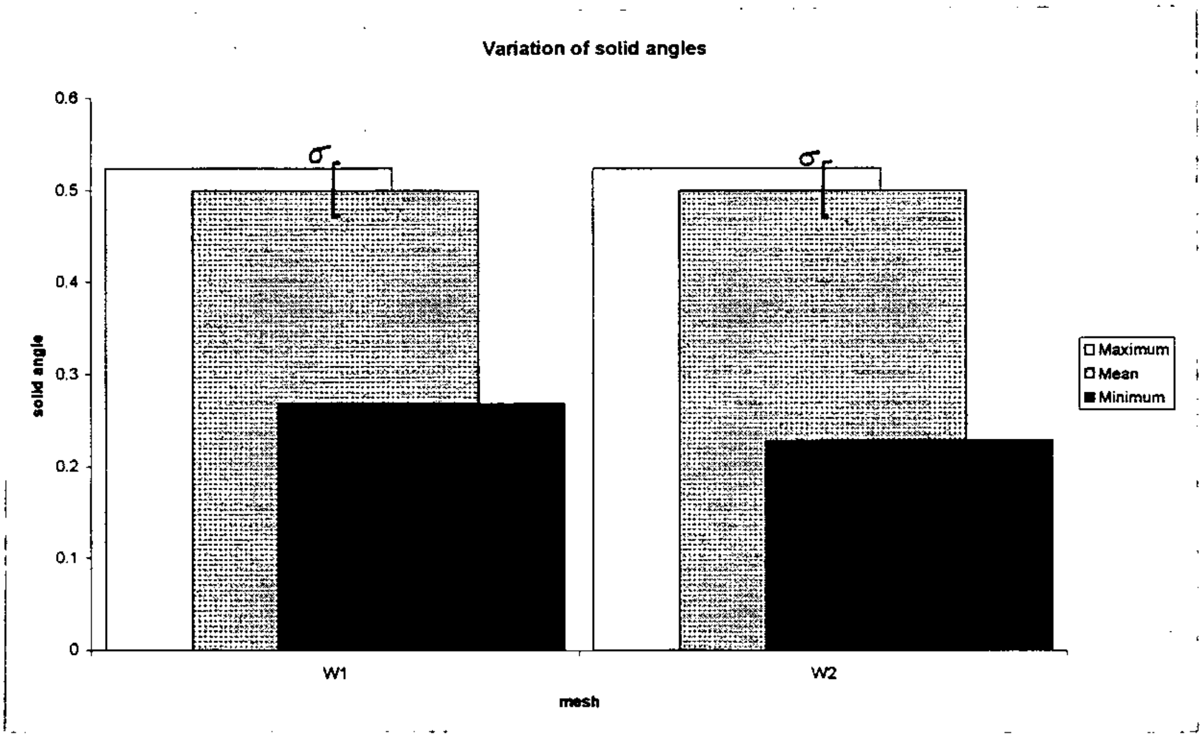

Figure 40: Variation of solid angles for Wigley hullform meshes

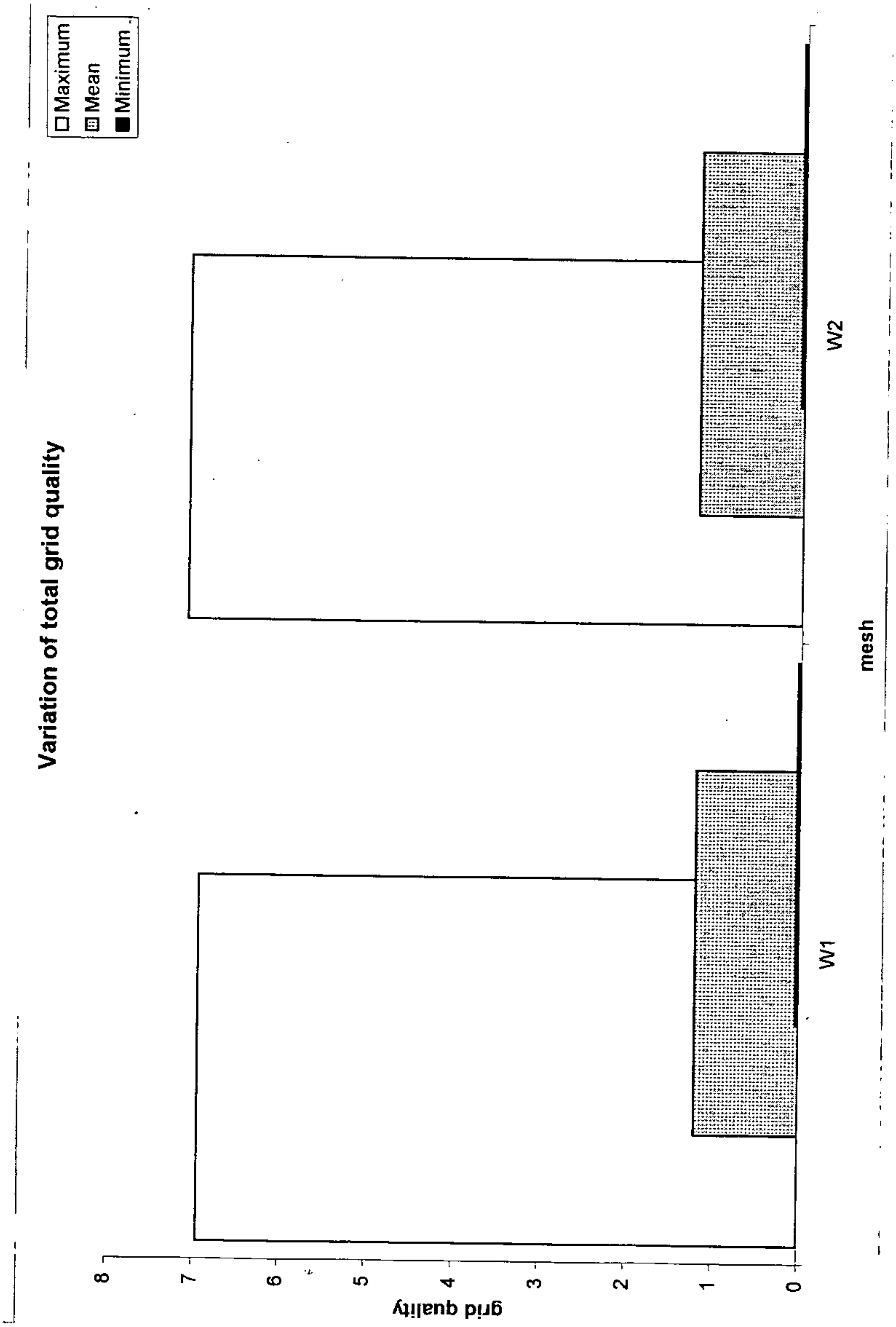

Figure 41: Variation of Wigley hullform grid quality

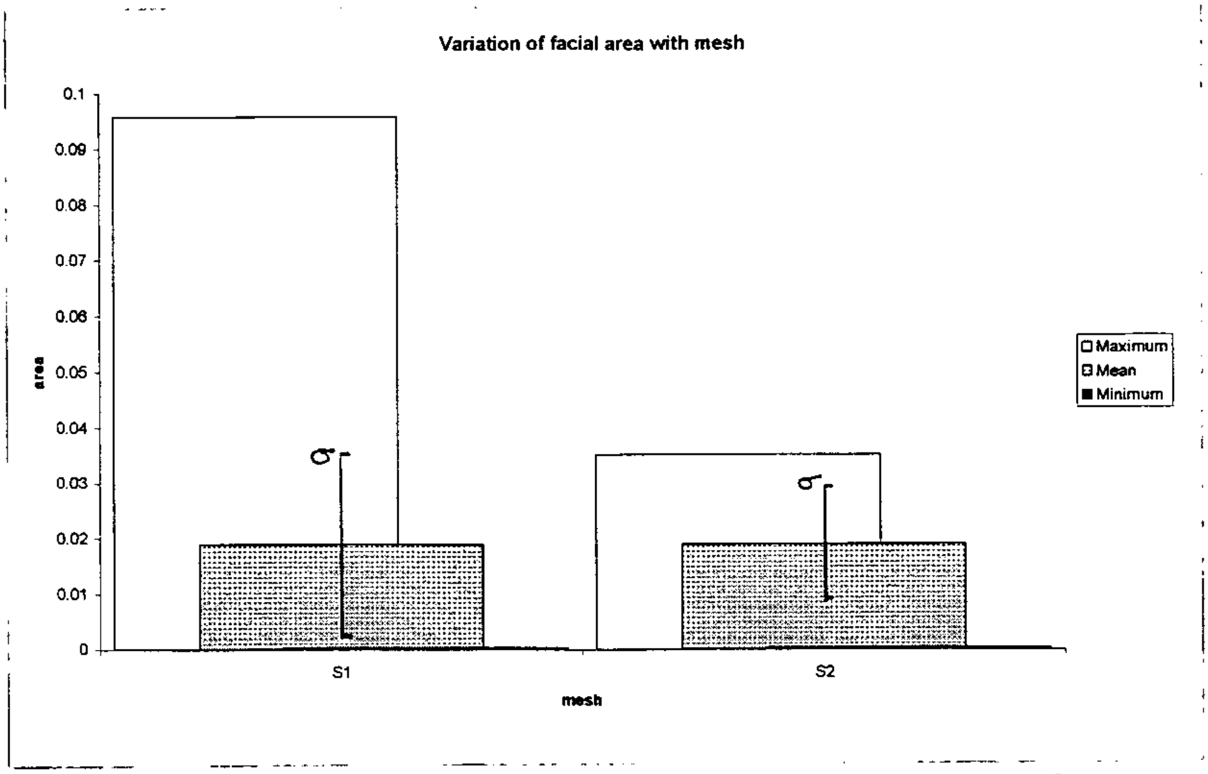

Figure 42: Variation of facial area for truncated cylinder meshes

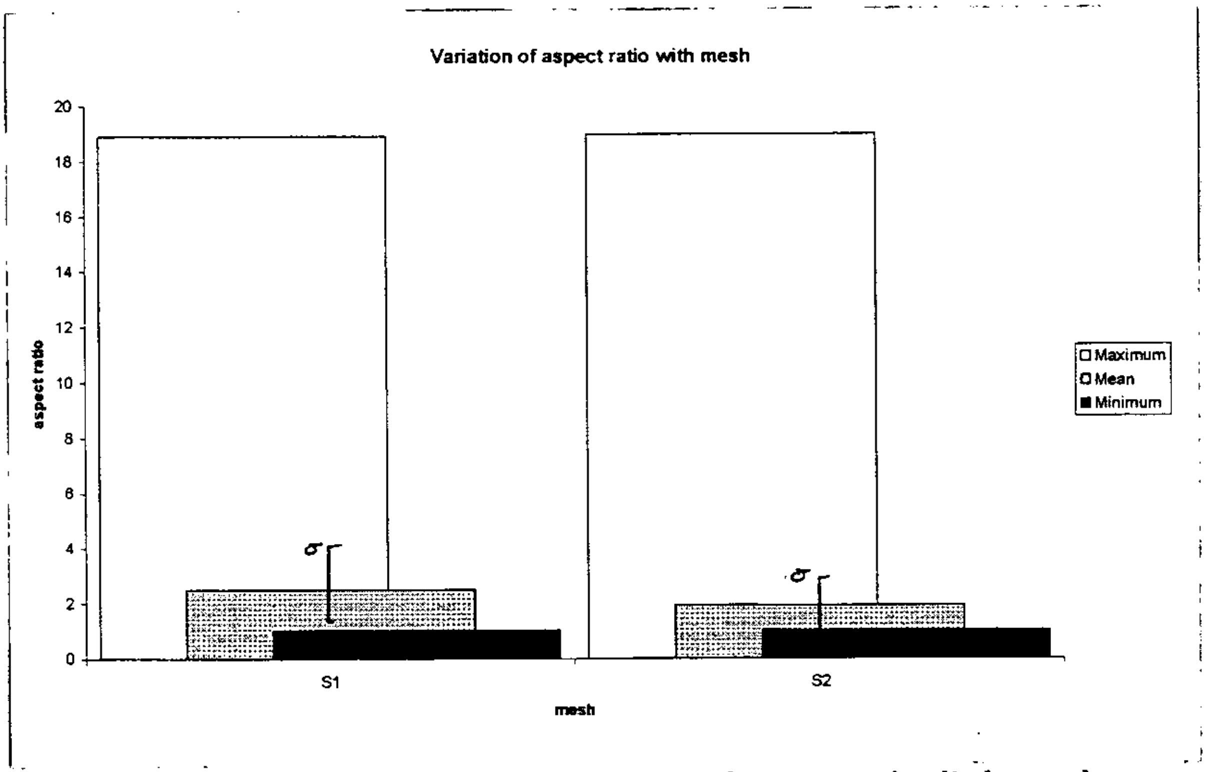

Figure 43: Variation of aspect ratio for truncated cylinder meshes

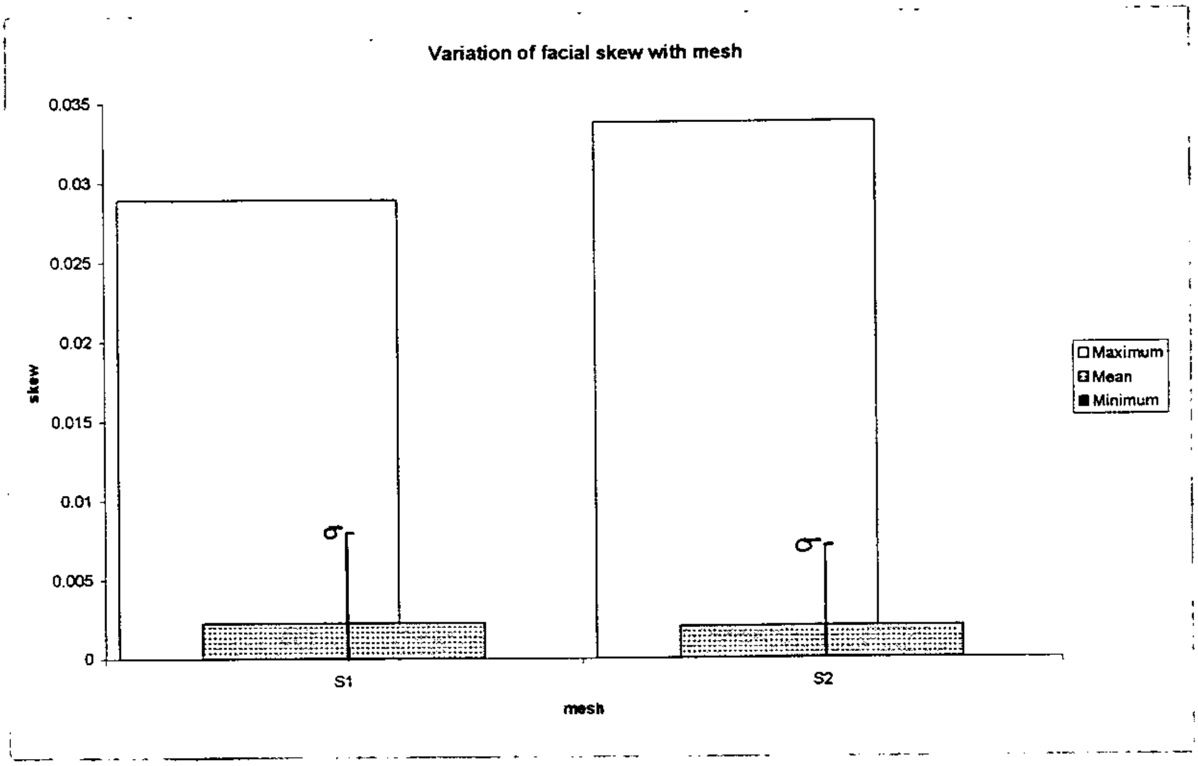

Figure 44: Variation of facial skew for truncated cylinder meshes

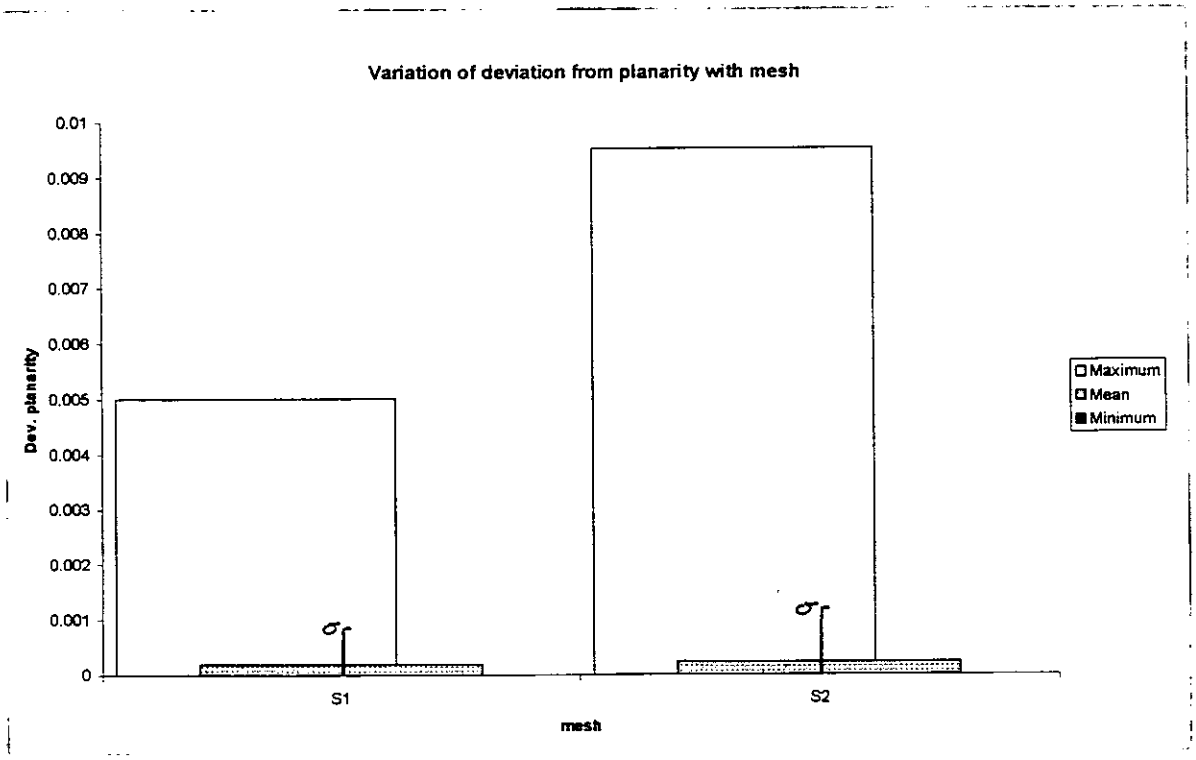

Figure 45: Variation of deviation from planarity for truncated cylinder meshes

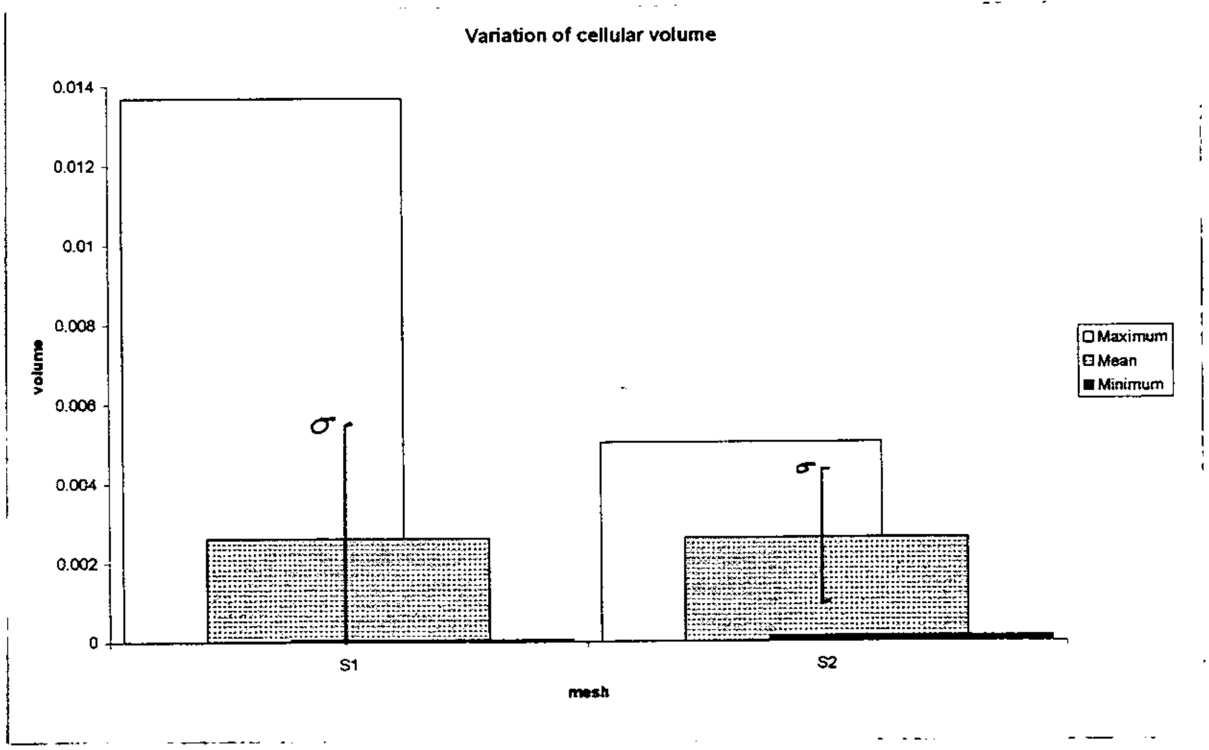

Figure 46: Variation of cellular volume for truncated cylinder meshes

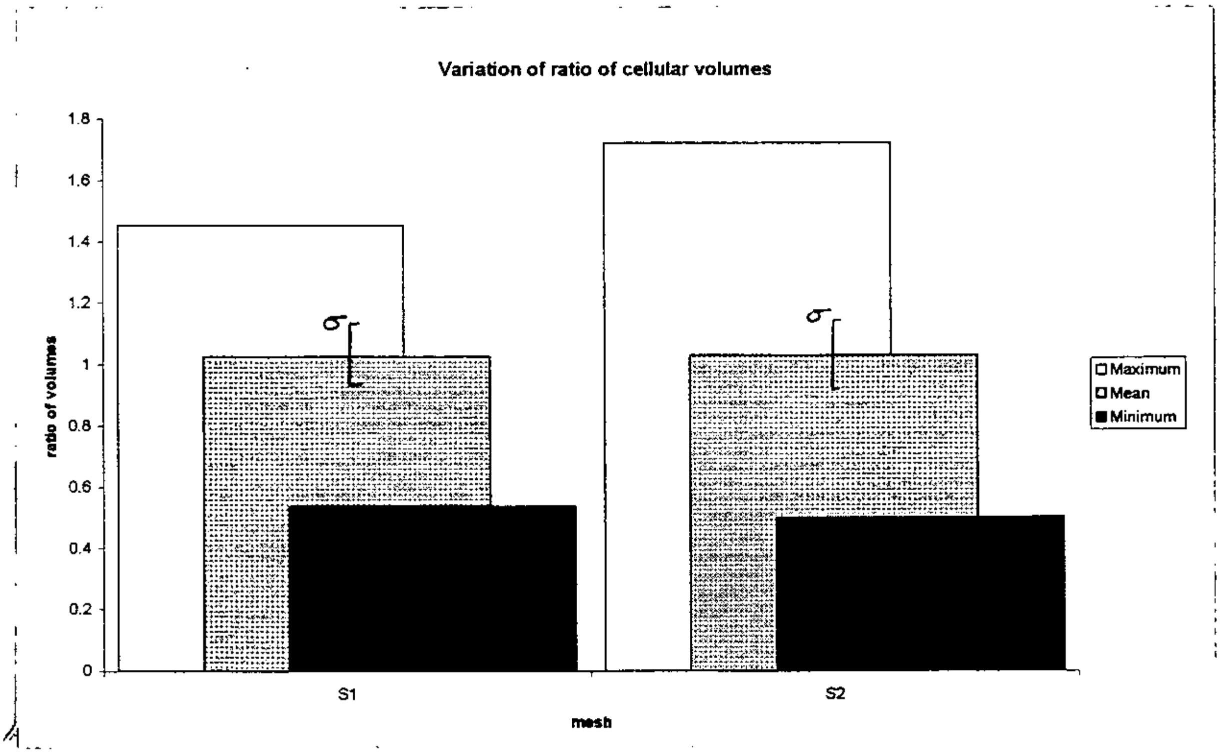

Figure 47: Variation of ratio of cellular volumes for truncated cylinder meshes

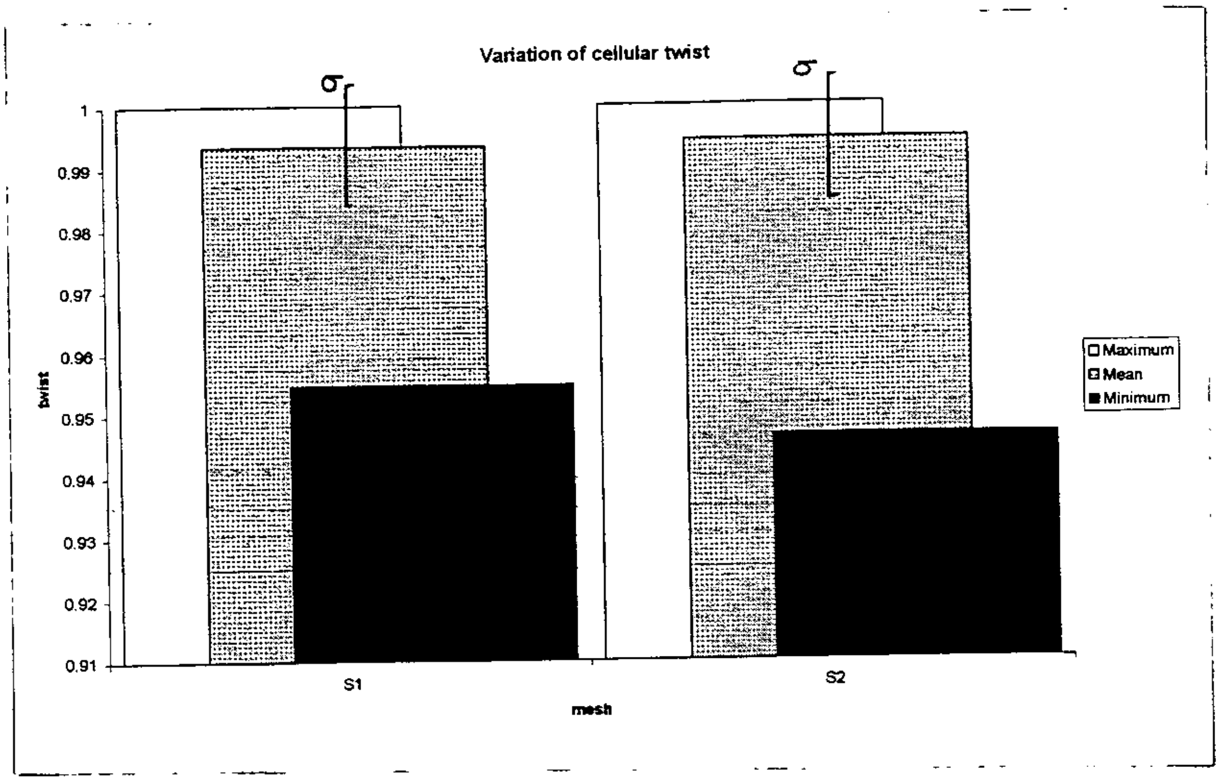

Figure 48: Variation of cellular twist for truncated cylinder meshes

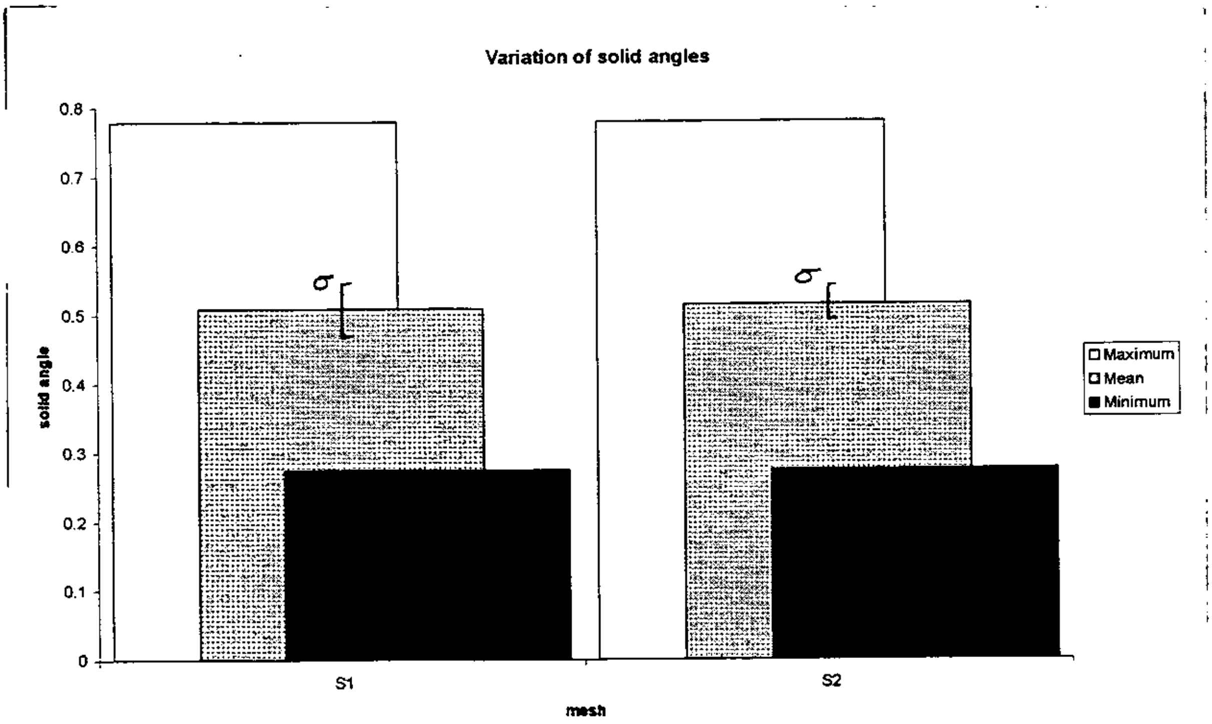

Figure 49: Variation of solid angles for truncated cylinder meshes

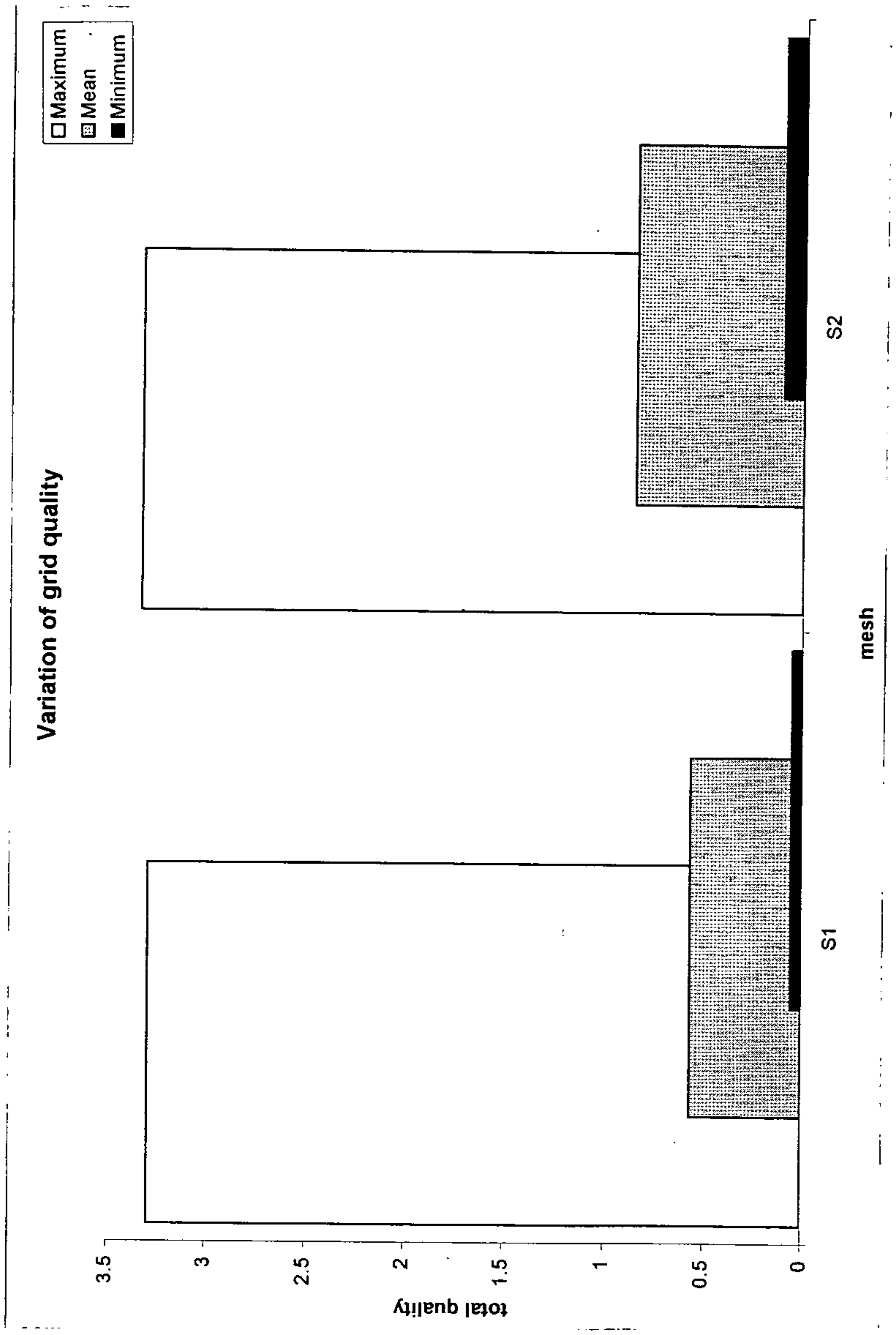

Figure 50: Variation of truncated cylinder grid quality

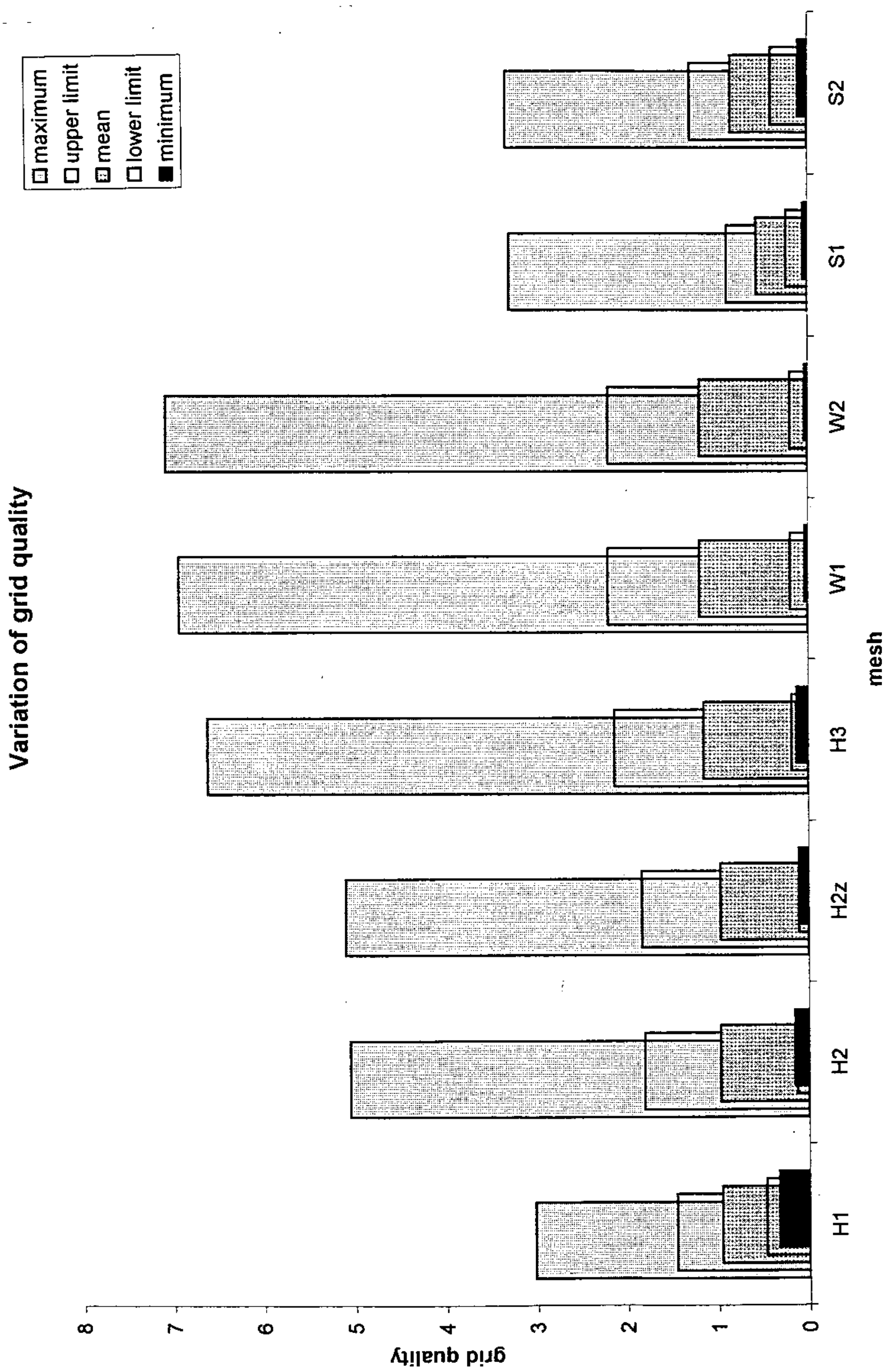

Figure 22: Relative quality of all the meshes created and analysied

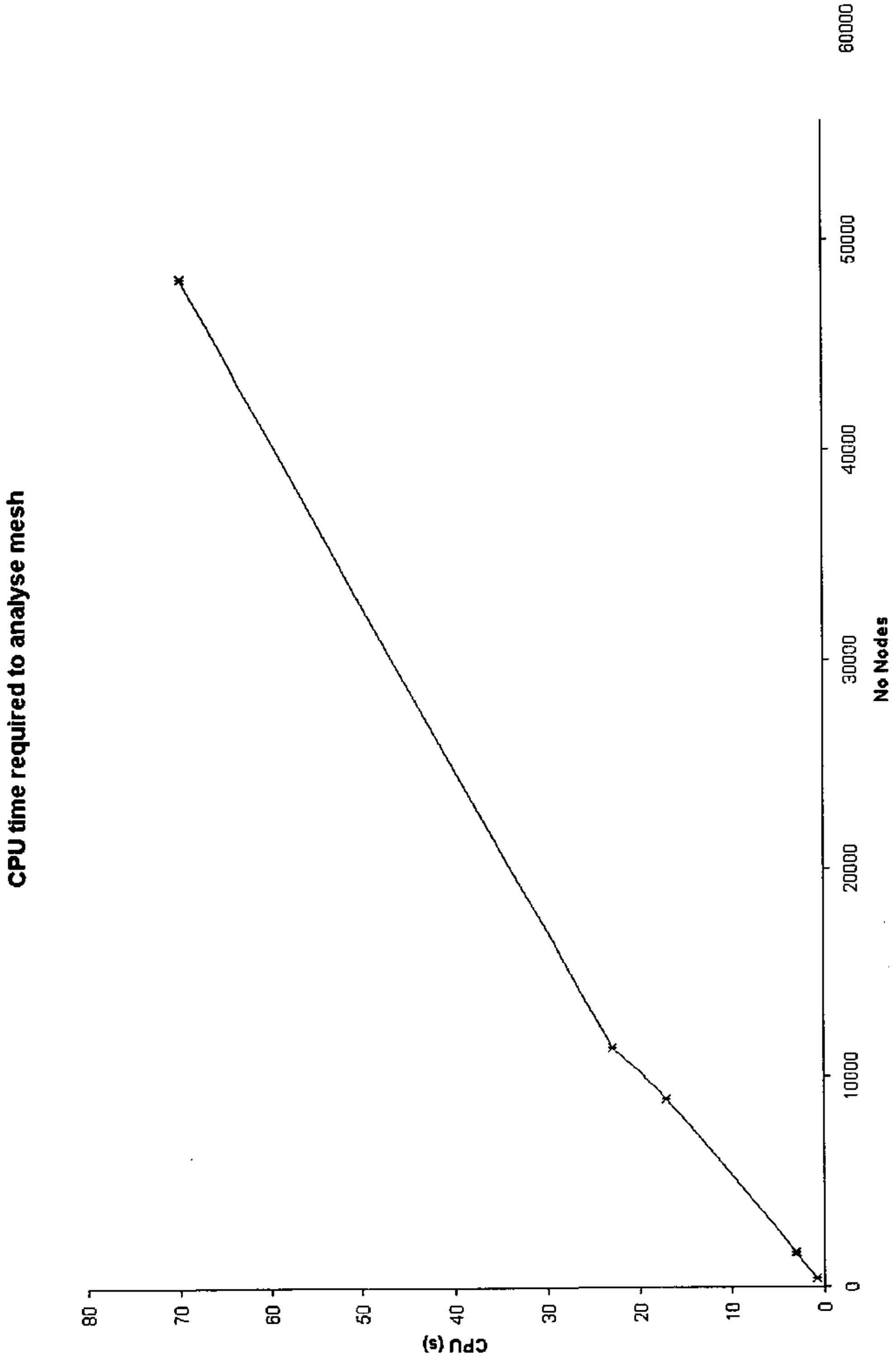

Figure 23: CPU time required to analyse the meshes

# 13 Future Developments

A method of measuring grid quality in a quantative manner as well as a qualitative manner has been successfully created. Differet meshing of the same domain can be compared against each other, as well as against meshs of different domains, thus providing an absolute quality control tool.

The relative influence of various geometrical and flow properties are known, and have been incorporated in the scheme, as shown in Chapther 10 and in the weightings detailed in Table 1.

The presentation of the calculated data can be seen visually via the AVS [4] software package, as well as tabulated maxima, mean and standard deviations.

There are two areas in which future advancements are expected to be achieved.

**Visualisation** Because of the unstructured nature of the data being dealt with, it is increasingly difficult to allow for all possible forms of geometrical structure while maintaining an efficient storeage of information in the structured format that is expected by packages such as AVS. Thus it is expected that an in-house dedicated visualisation system will be developed, allowing the geometrical structure and its quality to be visualised simultaneously in an efficient manner, as well as allowing easy identification of problematic areas of meshing.

**Absolute weightings of properties** As has been noted above , the relative weightings of the various properties have been defined, but the absolute values have not as yet. This could be achieved by creating meshs about well documented geometries with parametric variation of the properties. The solution accuracy can then be directly related to the mesh quality, and more accurate values for the weightings gained.

[4] Advanced Visualisation System

# References

[1] N. C. Rycroft and S. R. Turnock. "3-D Multiblock Grid Generator; FLEXIMESH". Ship Science Report 101, Department of Ship Science, University of Southampton, 1997.

[2] R. Haimes, S. D. Connell, and S. A. Vermeersch. "Visual grid quality assesment for 3D unstructured meshes". AIAA 1993; internet.

[3] A. Ilinca, R. Camarero, J. Y. Trepanier, and M. Reggio. "Error estimator and adaptive moving grids for finite volumes schemes". *AIAA Journal*, 33(11):2058–2065, November 1995.

[4] J. Oliger and X. Zhu. "Stability and error estimation for component adaptive grid methods". *Applied Numerical Mathematics*, 20:407–426, 1996.

[5] A. Rassineux. "Generation and optimisation of tetrahedral meshes by advancing front technique". *Int. Journal for numerical Methods in Engineering*, 41:651–674, 1998.

[6] R. G. Hindman. "Generalized coordinate forms of governing fluid equations and associated geometrically induced errors". *AIAA Journal*, 20:1359, 1982.

[7] I. Demirdzic and M. Perić. "Space conservation law in finite volume calculations of fluid flow". *Int. Journal of Numerical Methods in Fluids*, 8:1037, 1988.

[8] J.D. Anderson. *"Computational Fluid Dynamics - The basics with applications"*. McGraw Hill Inc., 1995. ISBN 0-07-113210-4.

[9] J.J. Bertin and M.L. Smith. *"Aerodynamics for engineers"*. Prentice Hall, 1989. ISBN 0-13-018243-5.

[10] A. Jameson, W. Schmidt, and E. Turkel. "Numerical solution of the euler equations by finite volume methods using Runge Kutta time stepping schemes". AIAA Paper 81-1259, 1981.

[11] C. Hirsch. *"Numerical Computation Of Internal And External Flows"*, volume 1. J. Wiley and sons, 1988. ISBN 0-471-92385-0.

[12] A. M. Wright and S. R. Turnock. "Efficient multi-level adaptation methods for unstructured polyhedral computational meshes". Ship Science Report 109, Department of Ship Science, University of Southampton, 1999.

# 14 Appendices

## 14.1 Algorithms for property calculation

### 14.1.1 Control volume twist

```
double CV_twist
(N_Grid *grid, Full_N_list *CV_corners,
 CC_info *CV_faces,FILE *logfile)
{int i, j;
 int no_f_corners, *face_corner;
 double offset, twist, theta;
 coord3D CV_centroid, F_centroid;
 Vector normal, A;
 CC_info *currentPTR;
 NodePTR_list *current_NA_PTR;
twist = 0.0;

// calculate the CV centroid.
CV_centroid.x = CV_centroid.y = CV_centroid.z = 0.0;

current_NA_PTR = CV_corners->nextPTR;
for (i = 0; i < CV_corners->number; i ++){

CV_centroid.x += current_NA_PTR->address->coords.x;
CV_centroid.y += current_NA_PTR->address->coords.y;
CV_centroid.z += current_NA_PTR->address->coords.z;

current_NA_PTR = current_NA_PTR->nextPTR;
}

CV_centroid.x *= (1.0/CV_corners->number);
CV_centroid.y *= (1.0/CV_corners->number);
CV_centroid.z *= (1.0/CV_corners->number);

currentPTR = CV_faces->nextPTR;
for (i = 0; i < CV_faces->number; i ++){

// 2. find the face centroid
F_centroid.x = F_centroid.y = F_centroid.z = 0.0;
find_face_corners
(&face_corner, grid->face[currentPTR->number], no_f_corners);

for (j = 0; j < no_f_corners; j ++){
```

```
number = currentPTR->number;
F_centroid.x +=
grid->face[number]->node[face_corner[j]]->coords.x;
F_centroid.y +=
 grid->face[number]->node[face_corner[j]]->coords.y;
F_centroid.z +=
 grid->face[number]->node[ face_corner[j]]->coords.z;
}
F_centroid.x *= (1.0/no_f_corners);
F_centroid.y *= (1.0/no_f_corners);
F_centroid.z *= (1.0/no_f_corners);

// 3. and the face normal
normal =
calc_face_normal(grid->face[ currentPTR->number ], logfile);

// 4. distance between the face and CV centroids'
A = Dist_2_point(F_centroid, CV_centroid);

// 5. calc the angle between the normal and the direct line
theta = acos( ( Dot_product(normal, A)/ A.magnitude() ) );

// 6. from which the orthogonal distance can be found
offset = A.magnitude()*sin(theta);

twist += offset;

delete []face_corner;
}

// 7. the CV twist is the average for all the faces
twist *= (1.0/CV_faces->number);
twist = 1.0 - twist;

return(twist);
}
```

### 14.1.2 Solid angle of CV corner

```
/* Solid angles are taken to be the spherical volume
 inclosed by the vertices
 defining the corner of a volume. Thus three
mutually orthogonal vertices will give a solid
 angle of 0.25*{(4/3)PI}
 (with unitary radii)*/

e connections point towards the CV corner, put it in the array
if (currentPTR->number == i){
vertices[count] = current_NA_PTR->address->coords;
if (count < connections->number)
currentPTR = currentPTR->nextPTR;
else
flagB = 1;
count ++;
}

// cycle onto the next node in the list of CV corners
i ++;
if (i < corner_list->number)
current_NA_PTR = current_NA_PTR->nextPTR;
else{
current_NA_PTR = corner_list->nextPTR;
i = 0;
}

} while ( (flagA == 0) || (flagB == 0) );

n_tetra = connections->number - 2;
total_angle = 0.0;
for (i = 0; i < n_tetra; i ++){
// make these edges into vectors
OA = Dist_2_point(vertices[0], vertices[1]);
OB = Dist_2_point(vertices[0], vertices[i + 2]);
OC = Dist_2_point(vertices[0], vertices[i + 3]);

// normalise these vectors
OA.normalise();
OB.normalise();
OC.normalise();

// calc. area of 'base' face
cross = Cross_product(OB, OC);
area = 0.5*cross.magnitude();
```

```
// The normal to the face is set to unitary magnitude
normal.set_vector(
(cross.normal()).x,
(cross.normal()).y,
(cross.normal()).z );

// height of pyramid is projection of normal onto third vector
height = Dot_product(OA, normal);
height = fabs(height);

// volume = 1/3 * area * height
solid_angle = (1.0/3.0)*area*height;

/* normalise w.r.t spherical volume => vol of a
 pyramid with unitary sides is 1/6 there are
 eight such pyramids in a sphere & the vol. of a unitary
 sphere is (4/3)*PI As a result a pyramid of 1/6
 would result in a spherical section of 1/6 PI i.e.
  1/8 of 4/3 of PI
=> 4/8 of 1/3 of PI
=> 1/2 of 1/3 of PI
=> 1/6 */
// of PI
solid_angle *= PI;

total_angle += solid_angle;
}

/* this angle could be any multiple of PI/6,
from a planar surface to  volume
of a sphere. hence manipulate
the answer to be interms of PI/6 */
total_angle /= SOLID_RIGHT_ANGLE;
// this will leave a value of the type 1.21
temp = (int) (total_angle/1);
// this gives the integer number of SOLID_RIGHT_ANGLES
deviation = total_angle - (float)temp;
if (fabs(deviation) > 0.5)
solid_angle = fabs(deviation);
// then reset the total_answer to be a varient from 1.0
else
solid_angle = 1.0 + deviation;
solid_angle *= SOLID_RIGHT_ANGLE;

return(solid_angle);
}
```

### 14.1.3 Facial deviation from planarity

```
double calc_dev_planarity
(Face *face, Vector normal, FILE *logfile)
{int i, count;
 double deviation, dev_plane;
 coord3D base_point;
 Vector delta;

deviation = 0.0;
count = 0;

if (0 == face->boundaryID){
base_point = face->node[2]->coords;

for (i = 3; i < (face->noNodes-1); i ++){
delta = Dist_2_point(
base_point,
 face->node[i]->coords);
deviation += fabs(Dot_product(delta, normal));
count ++;
}
}
else{
base_point = face->node[0]->coords;

for (i = 2; i < (face->noNodes-1); i ++){
delta = Dist_2_point(base_point, face->node[i]->coords);
deviation += fabs(Dot_product(delta, normal));
count ++;
}
}

dev_plane = deviation/((float)count);

return (dev_plane);
}
```

### 14.1.4 Facial aspect ratio

```
double calc_face_AR(Face *face, double area, int *corner)
{double AR;
 Vector span;


span = Dist_2_point(
face->node[ corner[0] ]->coords,
face->node[ corner[1] ]->coords);

AR = area/SQR(span.magnitude());

if (AR < 1)
AR = 1.0/AR;

return(AR);
}
```

### 14.1.5 Facial skew

```
double calc_face_skew
(Face *face, Vector face_normal, int *corner, int no_corners)
{int i;
 double theta, offset, answer;
 coord3D *edge_centroid, centroid;
 Vector side, temp;
 Vector *edge_N;
 Vector EC_FC;

centroid.x = centroid.y = centroid.z = 0.0;
edge_N = new Vector[no_corners];
edge_centroid = new coord3D[no_corners];
answer = 0.0;

for (i = 0; i < no_corners; i ++){
centroid.x += face->node[ corner[i] ]->coords.x;
centroid.y += face->node[ corner[i] ]->coords.y;
centroid.z += face->node[ corner[i] ]->coords.z;

if (i == (no_corners-1))
side = Dist_2_point(
face->node[ corner[i] ]->coords,
face->node[ corner[0] ]->coords);
else
side = Dist_2_point(
face->node[ corner[i] ]->coords,
face->node[ corner[i+1] ]->coords);

temp = scale_vector(0.5, side);

edge_centroid[i] =
Add_a_vector(face->node[ corner[i] ]->coords, temp);

edge_N[i] = Cross_product(face_normal, side);
edge_N[i].normalise();

}

centroid.x *= 1.0/no_corners;
centroid.y *= 1.0/no_corners;
centroid.z *= 1.0/no_corners;

for (i = 0; i < no_corners; i ++){
// create a vector from edge centroid to face centroid
EC_FC = Dist_2_point(edge_centroid[i], centroid);

/* in case normal points in the opposite
 direction from the face centroid */
if (Dot_product(EC_FC, edge_N[i]) < 0)
edge_N[i] = scale_vector(-1.0, edge_N[i]);

// angle between EC_FC and the edge normal
theta = Dot_product(
EC_FC,
edge_N[i])/( EC_FC.magnitude()*edge_N[i].magnitude() );
theta = acos(theta);

/* and finally the distance by which the normal
 misses the face centroid */
offset = EC_FC.magnitude()*sin(theta);
```

```
answer += offset;
}

answer *= 1.0/no_corners;

delete []edge_N;
delete []edge_centroid;

return(answer);
}
```

### 14.1.6 Facial area

```
double calc_face_area
(Face *face, Vector &face_normal, FILE *logfile)
{double area;

if (face->boundaryID == 0)
area = calc_CCN_internal_face(face, face_normal, logfile);
else
area = calc_CCN_boundary_face(face, face_normal, logfile);

return(area);
}


double calc_CCN_internal_face
(Face *face, Vector &face_normal, FILE *logfile)
{int i, j;
 int no_subfaces;
 double temp_A, area, projected_A;
 coord3D temp_K;
 coord3D base_point;
 Vector temp_N;
 Vector OA, OB;
 Vector vector_A[10];


face_normal.set_vector(0.0, 0.0, 0.0);
```

```
/* calc the area of each of the faces surrounding the node,
 as well as the normal of the face */
no_subfaces = (face->noNodes - 4);

base_point = face->node[2]->coords;
for (i = 0; i < no_subfaces; i ++){

// set up the 2 face edge vectors
OA = Dist_2_point(base_point, face->node[3 + i]->coords);
OB = Dist_2_point(base_point, face->node[4 + i]->coords);

// calc the area and sub face normal
vector_A[i] = Cross_product(OA, OB);
vector_A[i] = scale_vector(0.5, vector_A[i] );

/* and calculate the face normal as a weighted
 average of the sub-face areas */
face_normal = Vect_sum(face_normal, vector_A[i]);

}

face_normal.normalise();


//now set the area, taking only the normal component
area = 0.0;
for (i = 0; i < no_subfaces; i ++){
projected_A = fabs( Dot_product(face_normal, vector_A[i] ) );

area = area + projected_A;
}

 return(area);
}



double calc_CCN_boundary_face
(Face *face, Vector &face_normal, FILE *logfile)
{int i;
 int no_subfaces;
 double temp_A, area, projected_A;
 coord3D base_point;
 Vector temp_N;
 Vector OA, OB;
 Vector vector_A[10];
/* this was previously dynamically set, but 10
should be more than twice the nessecary */


face_normal.set_vector(0.0, 0.0, 0.0);

/*calc the area of each of the faces surrounding the node,
 as well as the normal of the face */
no_subfaces = face->noNodes - 3;

base_point = face->node[0]->coords;


for (i = 0; i < no_subfaces; i ++){

// get the principle sides of the triangular sub-faces
OA = Dist_2_point(base_point, face->node[2 + i]->coords);
OB = Dist_2_point(base_point, face->node[3 + i]->coords);

// calc the area and sub face normal
vector_A[i] = Cross_product(OA, OB);
vector_A[i] = scale_vector(0.5, vector_A[i] );

/* and calculate the face normal as a weighted
 average of the sub-face areas */
face_normal = Vect_sum(face_normal, vector_A[i]);

}

face_normal.normalise();


//now set the area, taking only the normal component
area = 0.0;
for (i = 0; i < no_subfaces; i ++){
projected_A = fabs( Dot_product(face_normal, vector_A[i] ) );

area = area + projected_A;
}
```

```
 return(area);
}
```

### 14.1.7 Cellular volume

```
void Set_control_vol(N_Grid *grid, FILE *logfile, int flag)
/* This routine calculates the control volume (CV) for each
 node by calculating the volume of the pyramid from
each of its faces to the node itself. Thus each
(sub)face will have 2 volumes associated with it, 1
 for the node on either side. */
{int i, j, k;
 int no_subfaces;
 int count;
 double face_area, volume;
 double height, total_volume;
 Vector diagonal[2], cross;
 Vector face_normal, ridge_line;
 NodePTR node_to_use;
 NodePTR alive[256];

// initialise the volume for each node
for (i = 0; i < grid->noNodes; i ++){
count = 0;
mark_alive_CVs(grid->node[i], alive, count);
for (j = 0; j < count; j ++)
alive[j]->volume = 0.0;
}

for (i = 0; i < grid->noFaces; i ++)
if (grid->face[i]->boundaryID == 0){

// split into sub-faces
no_subfaces = (grid->face[i]->noNodes - 4)/2;
for (j = 0; j < no_subfaces;  j ++){

diagonal[0] = Dist_2_point (
grid->face[i]->node[2]->coords,
 grid->face[i]->node[(2*j) + 4]->coords);
diagonal[1] = Dist_2_point (
grid->face[i]->node[(2*j) + 3]->coords,
grid->face[i]->node[(2*j) + 5]->coords);

/* Cross multiply the vectors, take the magnitude and
 multiply by a half => AREA */
cross = Cross_product(diagonal[0], diagonal[1]);
face_area = 0.5*cross.magnitude();


// The normal to the face is set to unitary magnitude
face_normal.set_vector(
(cross.normal()).x,
(cross.normal()).y,
(cross.normal()).z );

for ( k = 0; k < 2; k ++){

node_to_use = alive_branch_member_of( grid->face[i]->node[k] );

/* The height of the pyramid is the component of
 one of the vertices in the direction of the normal
multiplied by that vertices length */

ridge_line = Dist_2_point(
grid->face[i]->node[2]->coords,
 node_to_use->coords);

height = Dot_product(face_normal, ridge_line);
height = fabs(height);
volume = ( (1.0/3.0) * height * face_area );

node_to_use->volume += volume;
} // end of loop between face primary nodes
} // end of subface loop
}

total_volume = 0.0;
for (i = 0; i < grid->noNodes; i ++){
count = 0;
mark_alive_CVs(grid->node[i], alive, count);
for (j = 0; j < count; j ++)
total_volume += alive[j]->volume;
```

```
}

if (flag == 1)
fprintf(stderr,"\nTOTAL DOMAIN VOLUME - %f\n", total_volume);
fprintf(logfile,"\nTOTAL DOMAIN VOLUME - %f\n", total_volume);

return;
}
```

## 14.2 Test case results

### 14.2.1 Flow over a 10% circular arc hump

| Mesh | Minimum | Mean | Maximum | Std Deviation |
|---|---|---|---|---|
| H1 | 0.003140 | 0.042786 | 0.153562 | 0.028745 |
| H2 | 0.000589 | 0.016609 | 0.059166 | 0.011189 |
| H2z | 0.000499 | 0.016316 | 0.061415 | 0.012302 |
| H3 | 0.000127 | 0.005623 | 0.019496 | 0.004115 |

Table 5: Facial area for hump meshes

| Mesh | Minimum | Mean | Maximum | Std Deviation |
|---|---|---|---|---|
| H1 | 1.000271 | 2.098188 | 10.514190 | 1.355819 |
| H2 | 1.000099 | 2.288447 | 20.525227 | 2.072530 |
| H2z | 1.001254 | 2.502964 | 16.579067 | 2.227619 |
| H3 | 1.000024 | 2.773980 | 31.077102 | 2.636949 |

Table 6: Facial aspect ratio for hump meshes

| Mesh | Minimum | Mean | Maximum | Std Deviation |
|---|---|---|---|---|
| H1 | 0.000000 | 0.010722 | 0.064438 | 0.018915 |
| H2 | 0.000000 | 0.005817 | 0.032172 | 0.009875 |
| H2z | 0.000000 | 0.000509 | 0.010489 | 0.001720 |
| H3 | 0.000000 | 0.002873 | 0.015694 | 0.004882 |

Table 7: Facial skew for hump meshes

| Mesh | Minimum | Mean | Maximum | Std Deviation |
|---|---|---|---|---|
| H1 | 0.000000 | 0.001649 | 0.032501 | 0.005025 |
| H2 | 0.000000 | 0.000493 | 0.008679 | 0.001328 |
| H2z | 0.000000 | 0.000059 | 0.004479 | 0.000333 |
| H3 | 0.000000 | 0.000129 | 0.002412 | 0.000336 |

Table 8: Facial deviation from planarity for hump meshes

| Mesh | Minimum | Mean | Maximum | Std Deviation |
|---|---|---|---|---|
| H1 | 0.000548 | 0.008560 | 0.038390 | 0.006813 |
| H2 | 0.000099 | 0.002101 | 0.009708 | 0.001711 |
| H2z | 0.000121 | 0.002101 | 0.011990 | 0.001967 |
| H3 | 0.000014 | 0.000387 | 0.001565 | 0.000310 |

Table 9: Cellular volume for hump meshes

| Mesh | Minimum | Mean | Maximum | Std Deviation |
|---|---|---|---|---|
| H1 | 0.461290 | 1.136110 | 1.732582 | 0.293967 |
| H2 | 0.483097 | 1.077069 | 1.675458 | 0.241628 |
| H2z | 0.479866 | 1.074717 | 1.678212 | 0.241614 |
| H3 | 0.492376 | 1.045688 | 1.501801 | 0.155663 |

Table 10: Ratio of adjacent volume sizes for hump meshes

| Mesh | Minimum | Mean | Maximum | Std Deviation |
|---|---|---|---|---|
| H1 | 0.359168 | 0.487897 | 0.523599 | 0.045202 |
| H2 | 0.355142 | 0.482961 | 0.523599 | 0.048108 |
| H2z | 0.491944 | 0.521720 | 0.523599 | 0.005068 |
| H3 | 0.353964 | 0.481437 | 0.523599 | 0.048650 |

Table 11: Control volume solid angles for hump meshes

| Mesh | Minimum | Mean | Maximum | Std Deviation |
|---|---|---|---|---|
| H1 | 0.911261 | 0.978307 | 1.000000 | 0.025188 |
| H2 | 0.954185 | 0.988980 | 1.000000 | 0.013294 |
| H2z | 0.982461 | 0.999110 | 1.000000 | 0.002325 |
| H3 | 0.977219 | 0.995619 | 1.000000 | 0.006404 |

Table 12: Control volume twist for hump meshes

| Mesh | Minimum | Mean | Maximum | Std Deviation |
|---|---|---|---|---|
| H1 | 0.330282 | 0.962536 | 3.026080 | 0.494514 |
| H2 | 0.155699 | 0.970718 | 5.053871 | 0.840020 |
| H2z | 0.110605 | 0.973656 | 5.103807 | 0.868222 |
| H3 | 0.131348 | 1.159126 | 6.625059 | 0.974868 |

Table 13: Control volume quality for hump meshes

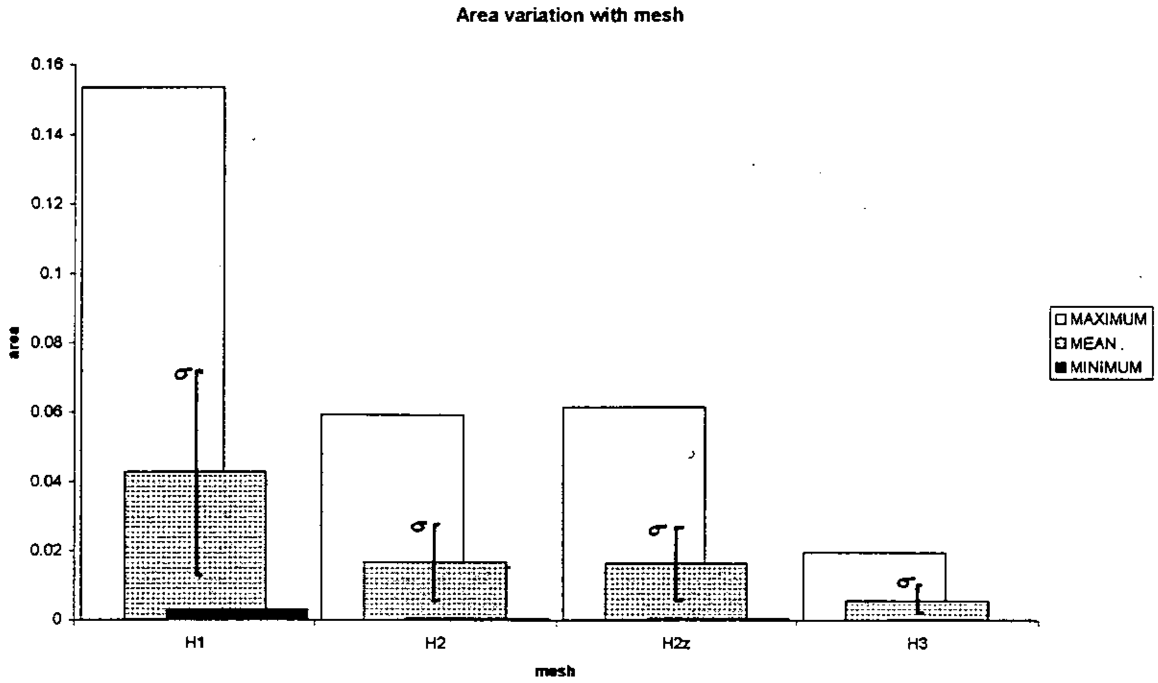

Figure 24: Variation of facial area for hump meshes

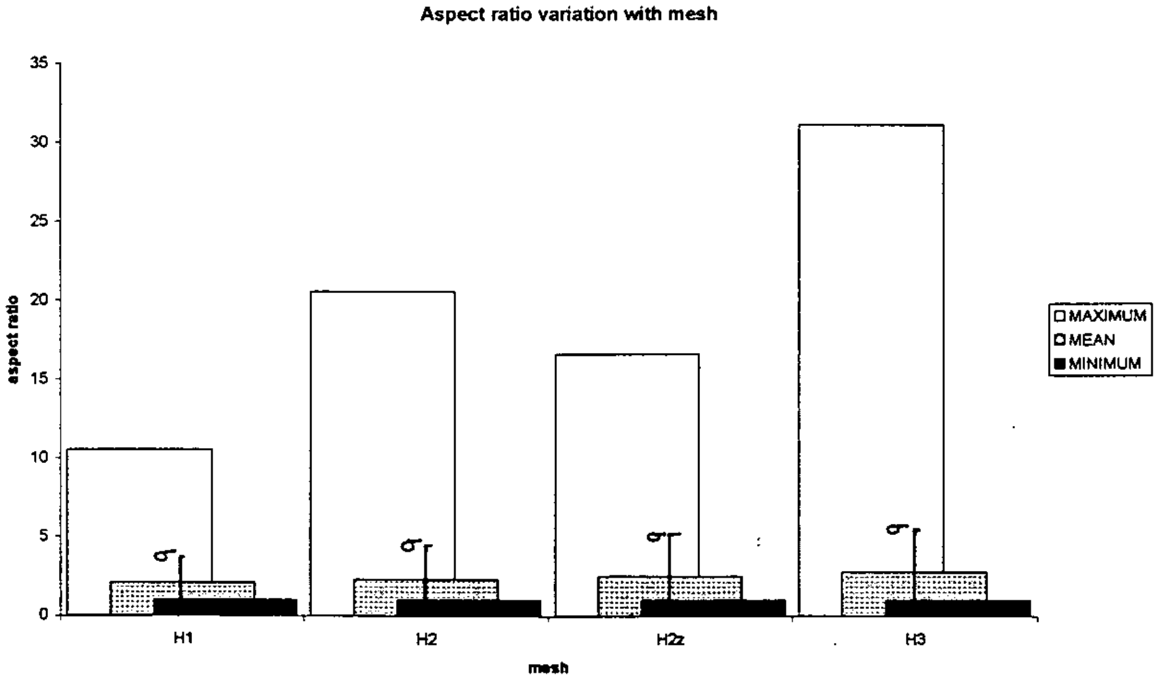

Figure 25: Variation of aspect ratio for hump meshes

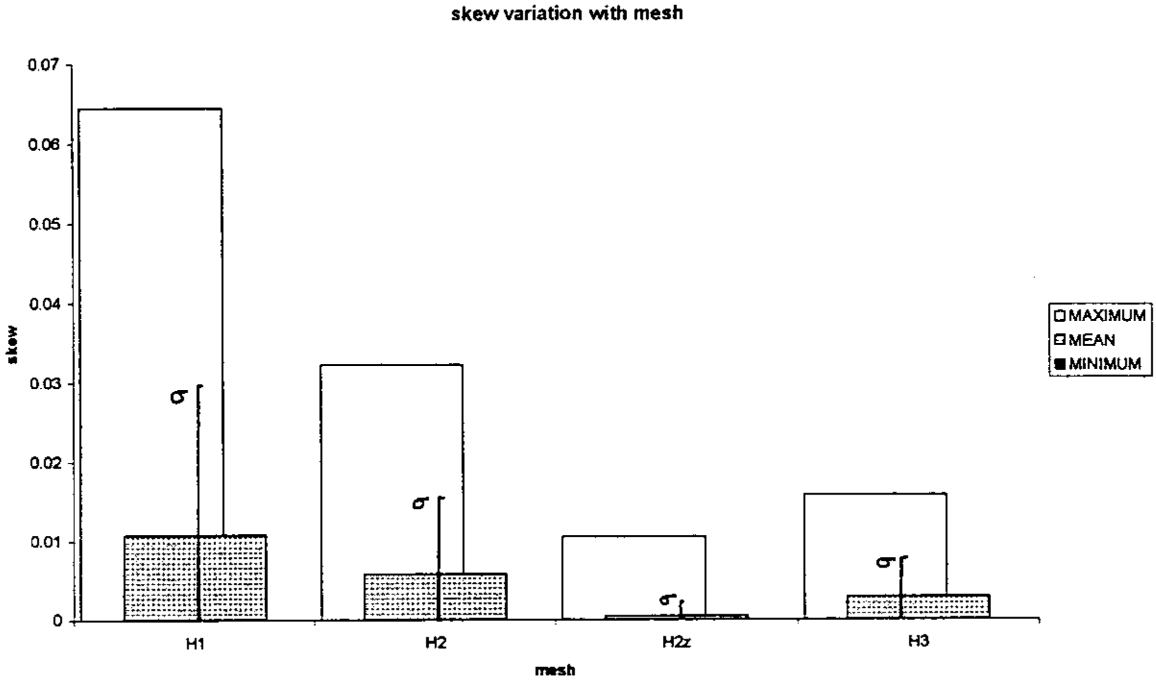

Figure 26: Variation of facial skew for hump meshes

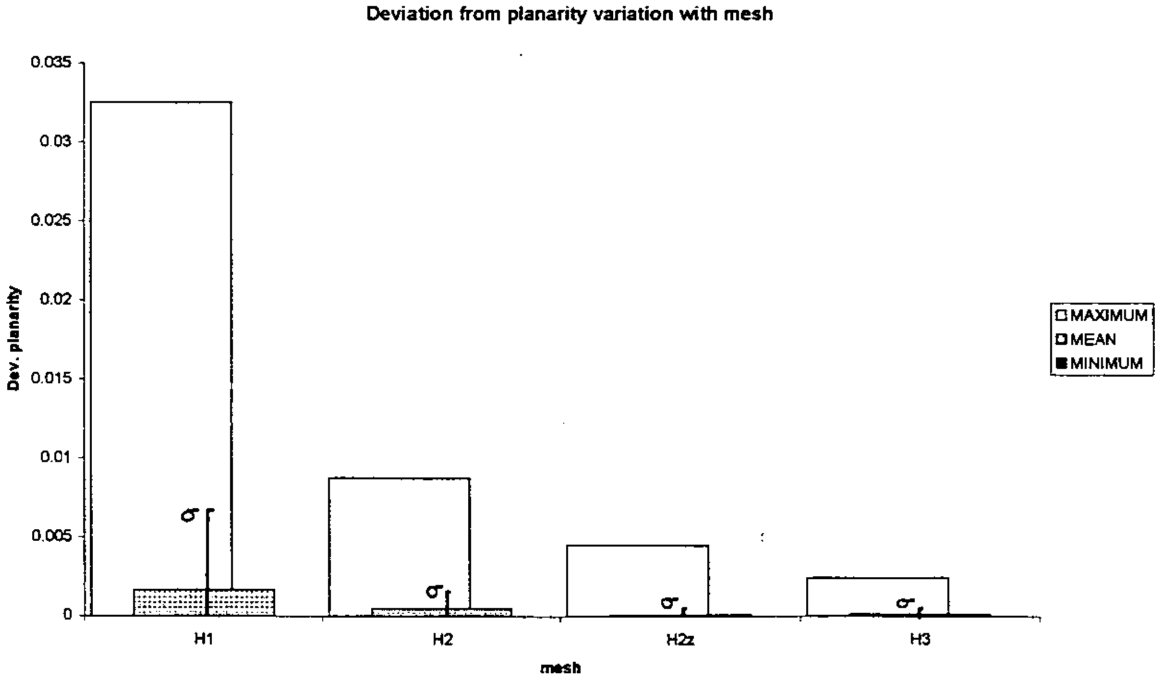

Figure 27: Variation of deviation from planarity for hump meshes

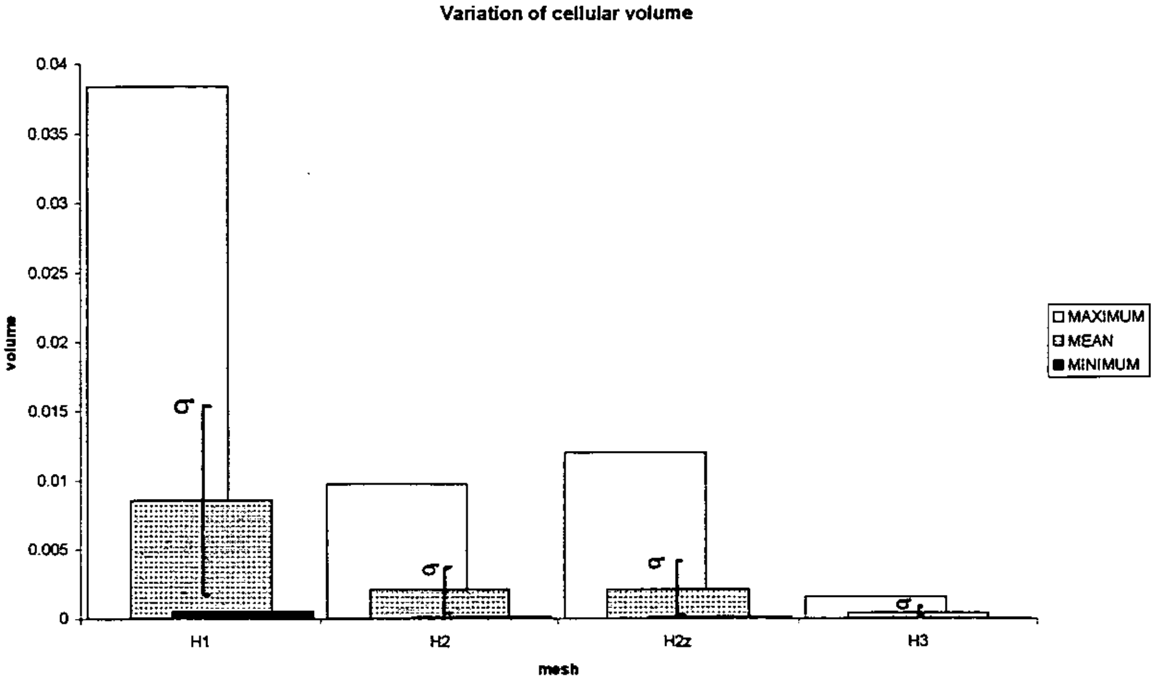

Figure 28: Variation of cellular volume for hump meshes

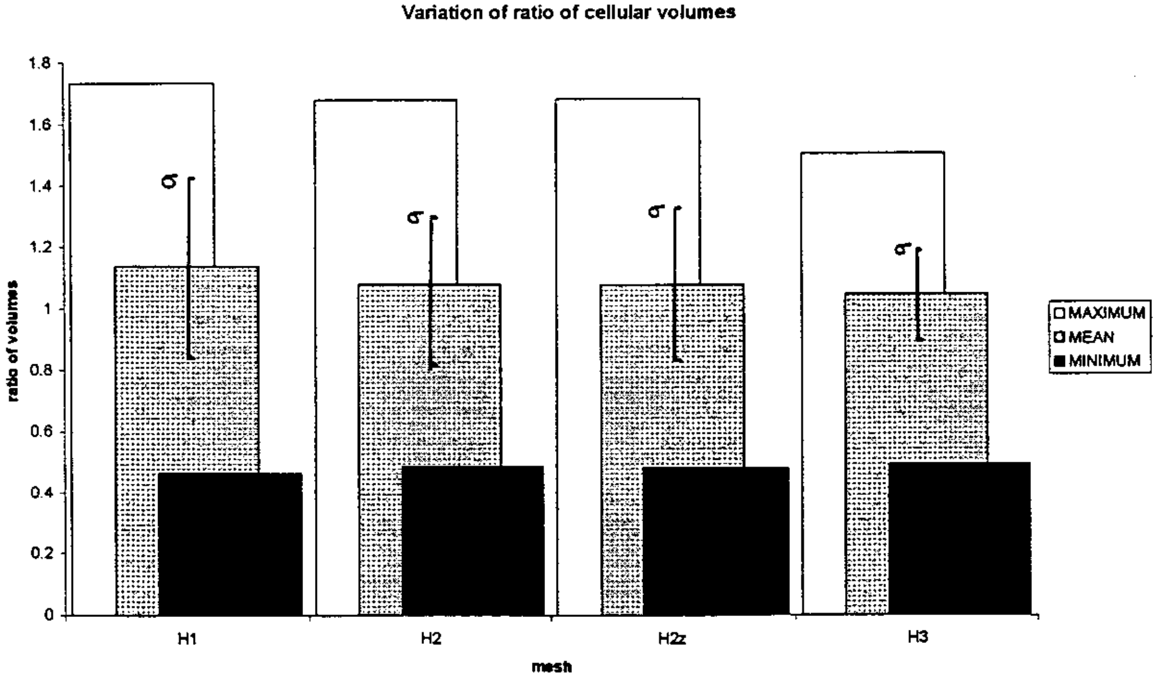

Figure 29: Variation of ratio of cellular volumes for hump meshes

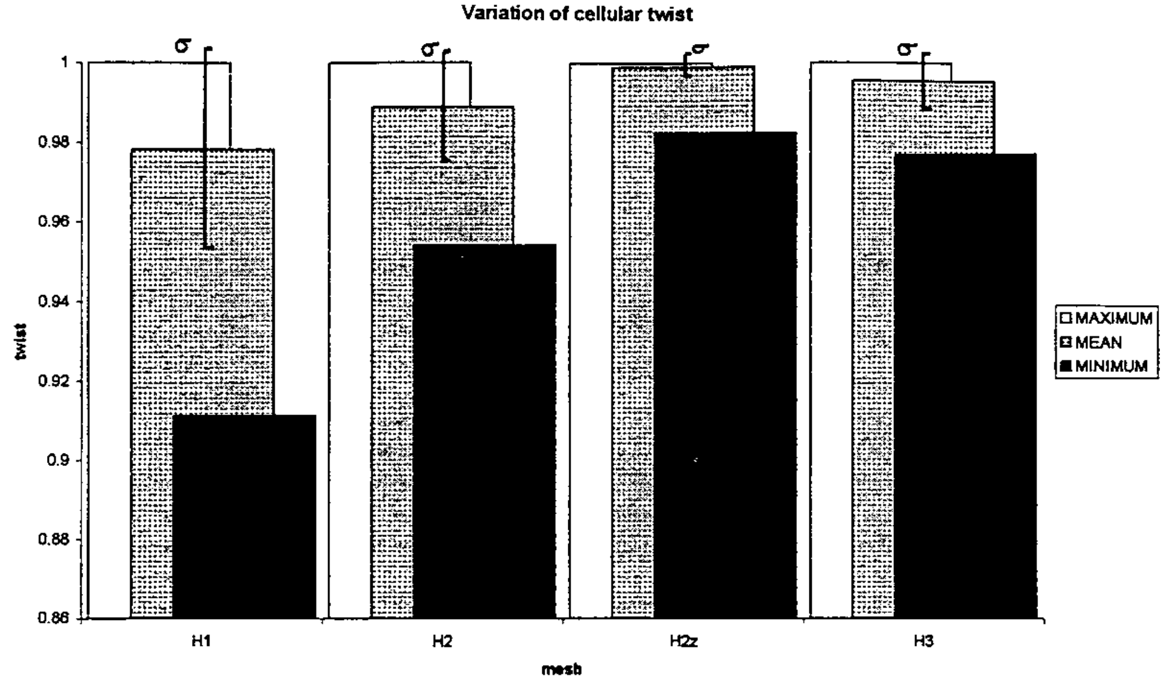

Figure 30: Variation of cellular twist for hump meshes

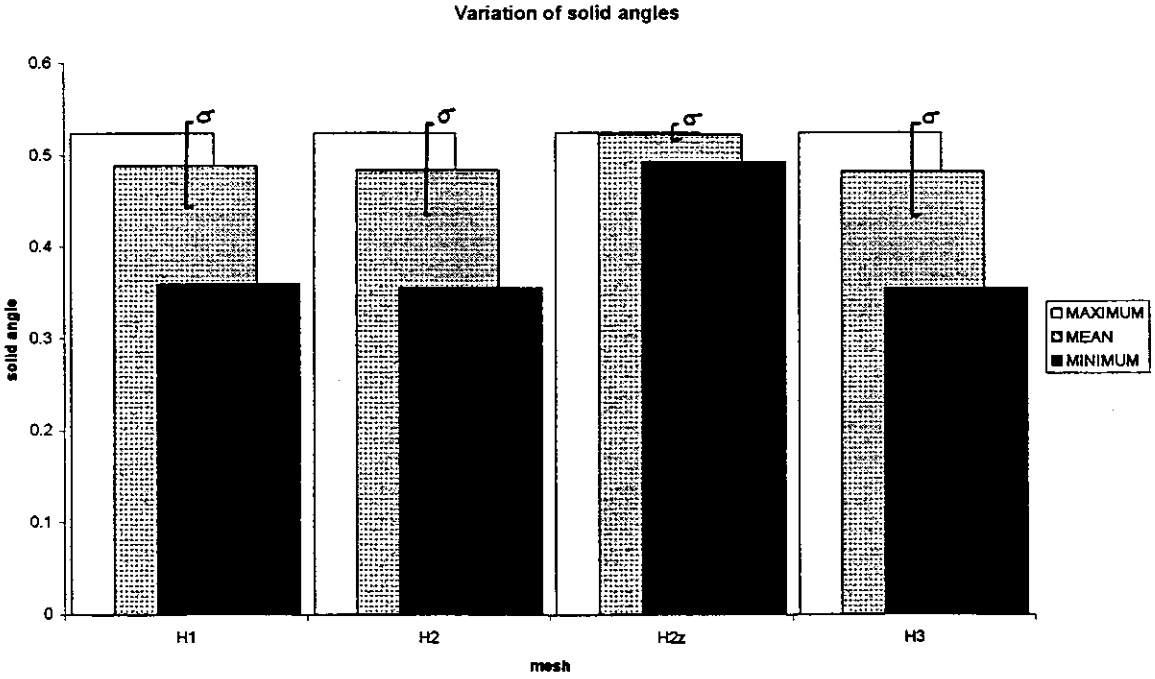

Figure 31: Variation of solid angles for hump meshes

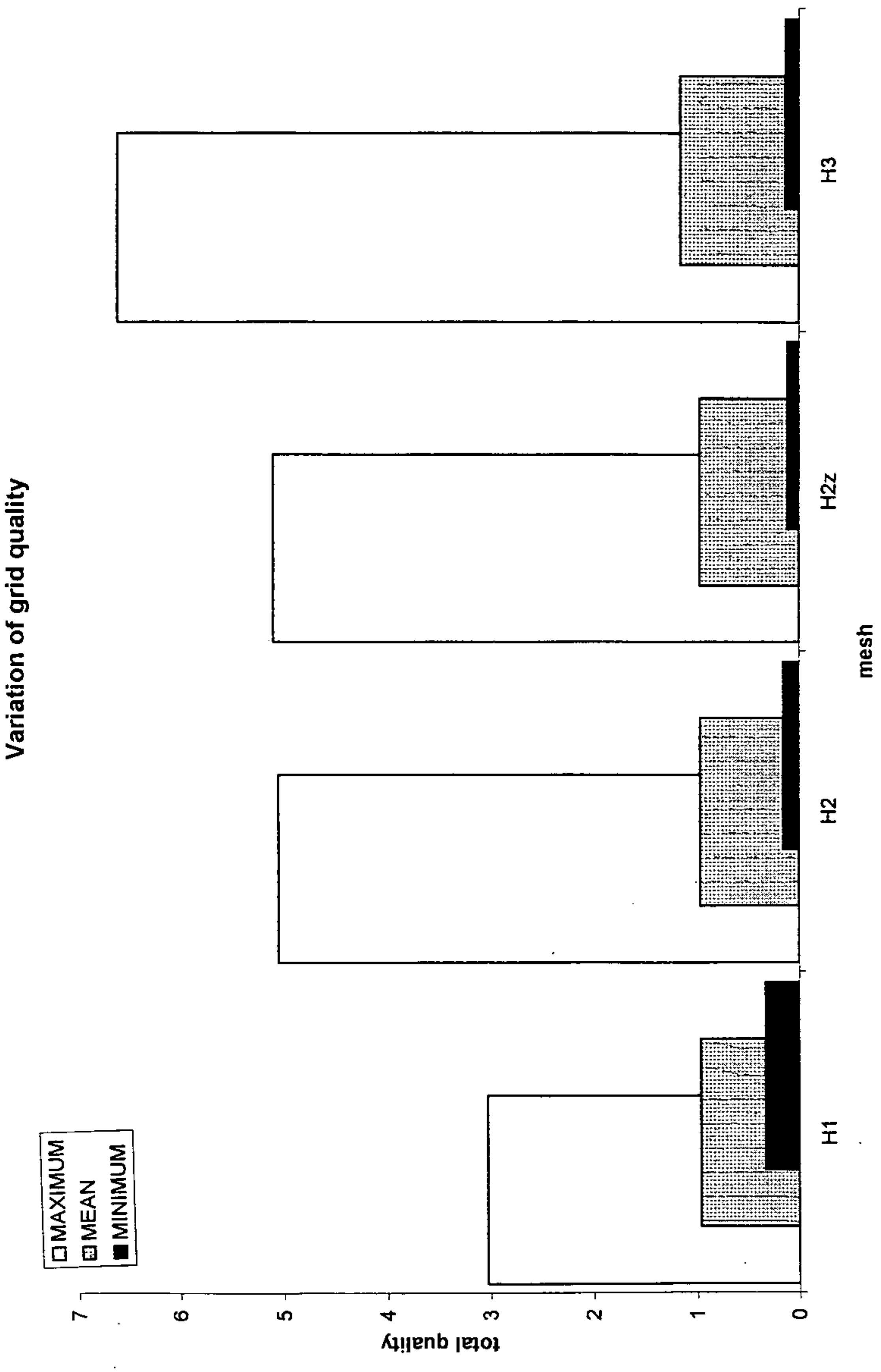

Figure 32: Variation of hump grid quality

### 14.2.2 Wigley hullform

| Mesh | Minimum | Mean | Maximum | Std Deviation |
|---|---|---|---|---|
| W1 | 0.000034 | 0.002110 | 0.010323 | 0.001611 |
| W2 | 0.000031 | 0.002110 | 0.010721 | 0.001614 |

Table 14: Facial area for Wigley hullform meshes

| Mesh | Minimum | Mean | Maximum | Std Deviation |
|---|---|---|---|---|
| W1 | 1.000120 | 3.130264 | 23.953729 | 2.938011 |
| W2 | 1.000004 | 3.132992 | 25.395043 | 2.944306 |

Table 15: Facial aspect ratio for Wigley hullform meshes

| Mesh | Minimum | Mean | Maximum | Std Deviation |
|---|---|---|---|---|
| W1 | 0.000000 | 0.001382 | 0.010917 | 0.002544 |
| W2 | 0.000000 | 0.001424 | 0.011066 | 0.002522 |

Table 16: Facial skew for Wigley hullform meshes

| Mesh | Minimum | Mean | Maximum | Std Deviation |
|---|---|---|---|---|
| W1 | 0.000000 | 0.000216 | 0.003641 | 0.000398 |
| W2 | 0.000000 | 0.000273 | 0.003663 | 0.000406 |

Table 17: Facial deviation from planarity for Wigley hullform meshes

| Mesh | Minimum | Mean | Maximum | Std Deviation |
|---|---|---|---|---|
| W1 | 0.000002 | 0.000087 | 0.000480 | 0.000069 |
| W2 | 0.000001 | 0.000087 | 0.000481 | 0.000069 |

Table 18: Cellular volume for Wigley hullform meshes

| Mesh | Minimum | Mean | Maximum | Std Deviation |
|---|---|---|---|---|
| W1 | 0.465637 | 1.035882 | 1.551666 | 0.123179 |
| W2 | 0.465708 | 1.036160 | 1.551769 | 0.123327 |

Table 19: Ratio of adjacent volume sizes for Wigley hullform meshes

| Mesh | Minimum | Mean | Maximum | Std Deviation |
|---|---|---|---|---|
| W1 | 0.268047 | 0.499223 | 0.523599 | 0.028750 |
| W2 | 0.228116 | 0.499174 | 0.523599 | 0.028865 |

Table 20: Control volume solid angles for Wigley hullform meshes

| Mesh | Minimum | Mean | Maximum | Std Deviation |
|---|---|---|---|---|
| W1 | 0.974743 | 0.997776 | 1.000000 | 0.003419 |
| W2 | 0.973257 | 0.997672 | 1.000000 | 0.003363 |

Table 21: Control volume twist for Wigley hullform meshes

| Mesh | Minimum | Mean | Maximum | Std Deviation |
|---|---|---|---|---|
| W1 | 0.034614 | 1.196925 | 6.946052 | 1.002978 |
| W2 | 0.033186 | 1.199051 | 7.094290 | 1.005270 |

Table 22: Control volume quality for Wigley hullform meshes

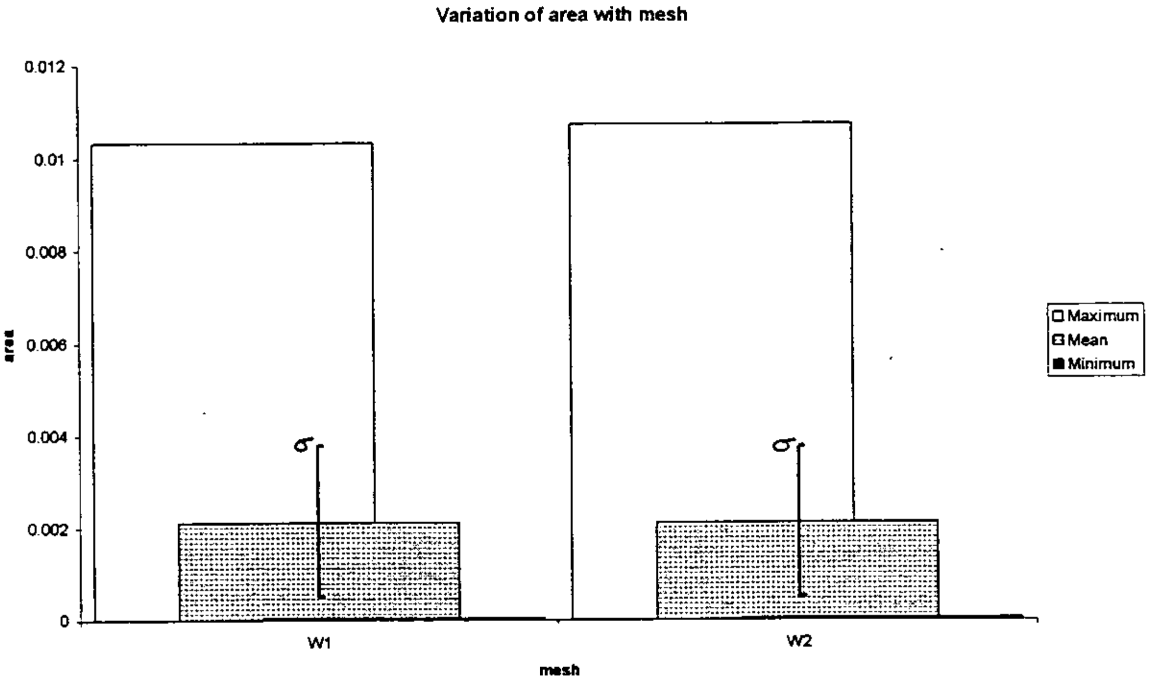

Figure 33: Variation of facial area for Wigley hullform meshes

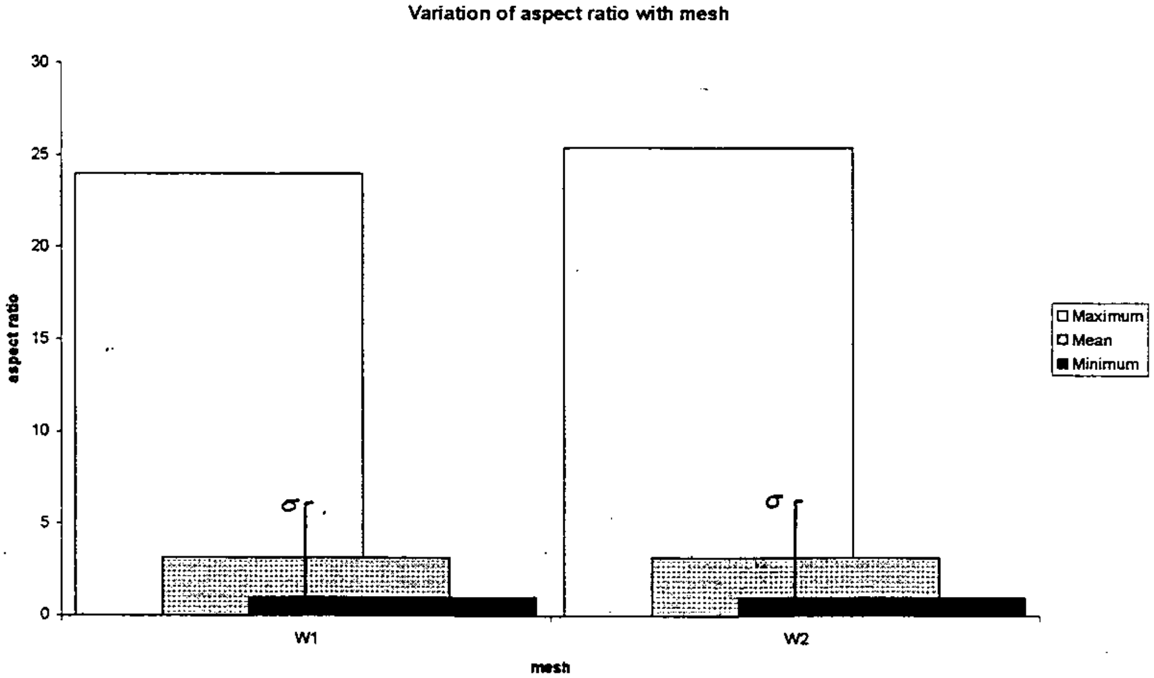

Figure 34: Variation of aspect ratio for Wigley hullform meshes

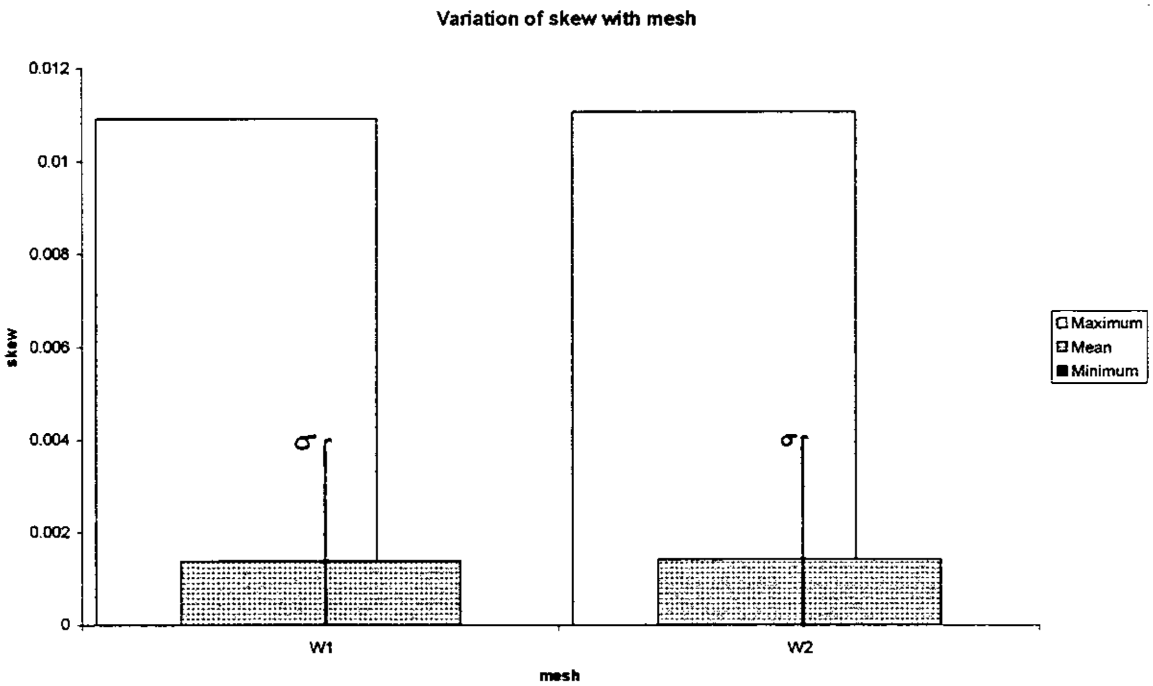

Figure 35: Variation of facial skew for Wigley hullform meshes

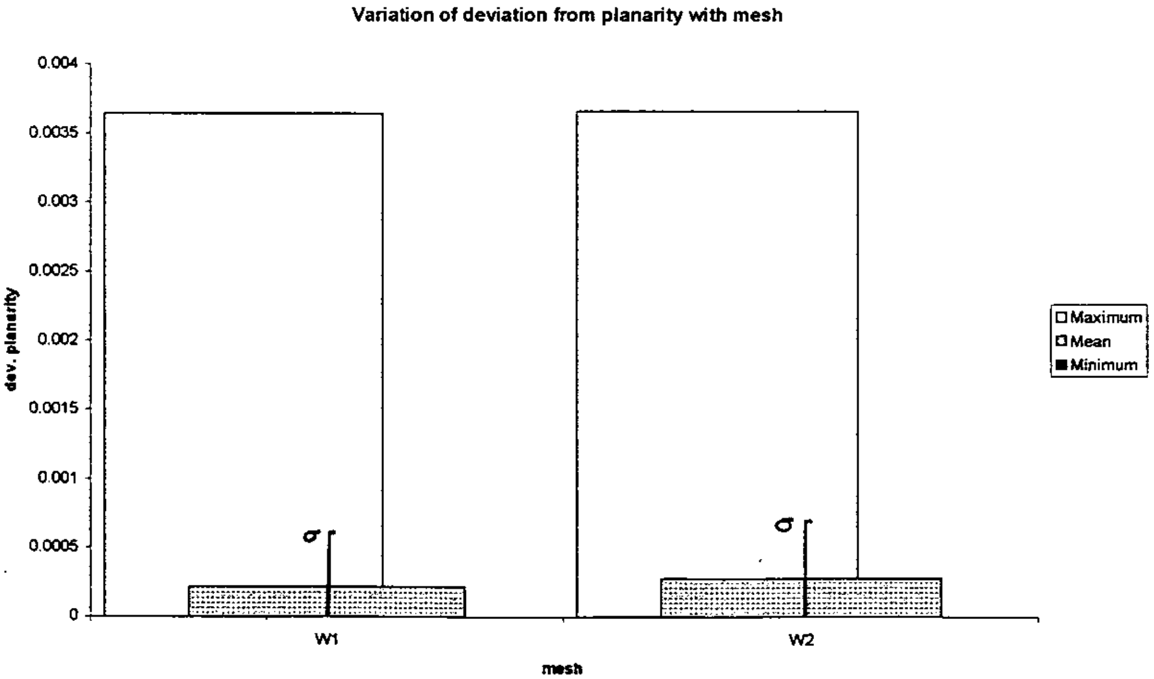

Figure 36: Variation of deviation from planarity for Wigley hullform meshes

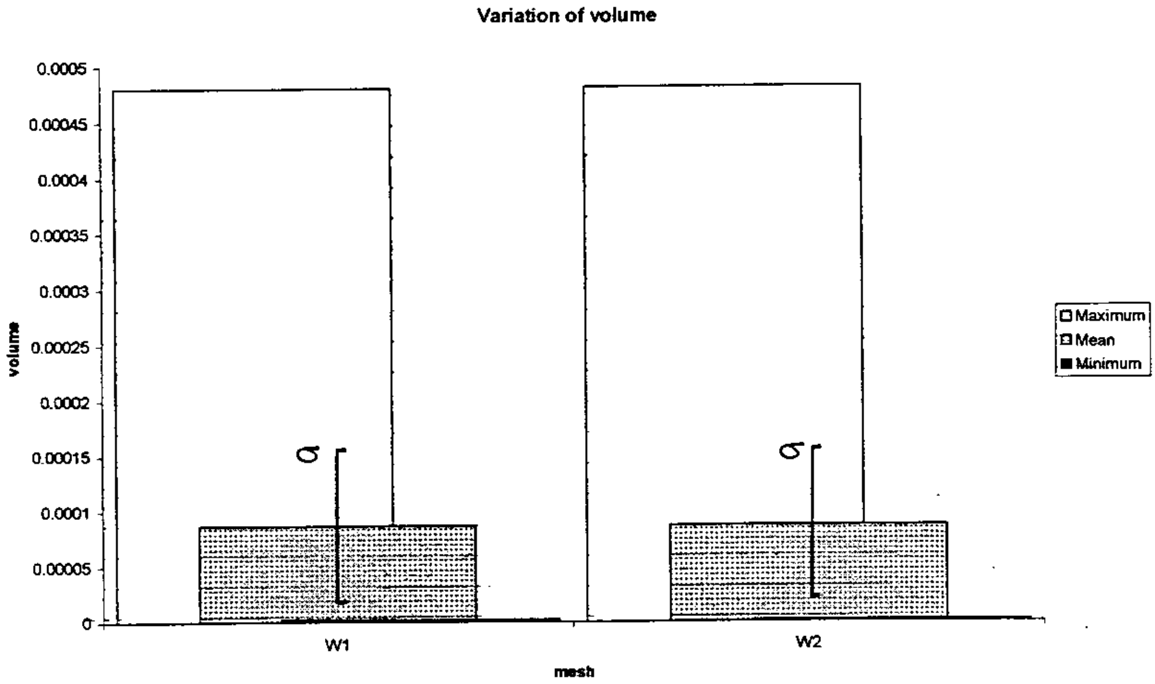

Figure 37: Variation of cellular volume for Wigley hullform meshes

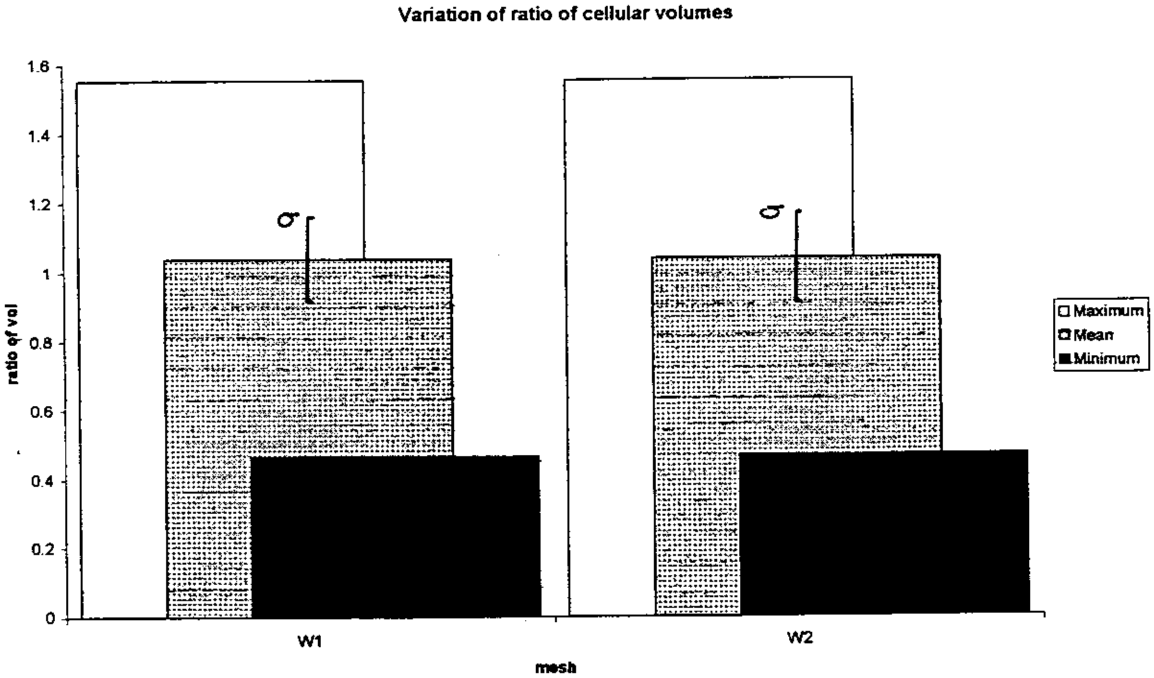

Figure 38: Variation of ratio of cellular volumes for Wigley hullform meshes

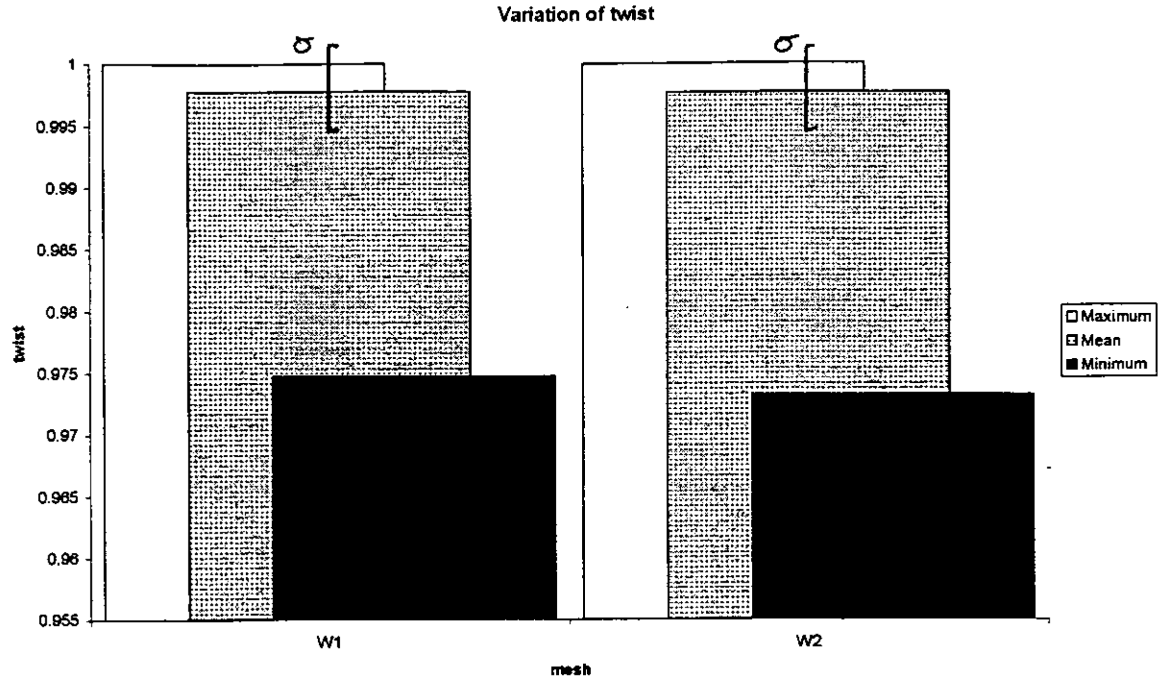

Figure 39: Variation of cellular twist for Wigley hullform meshes

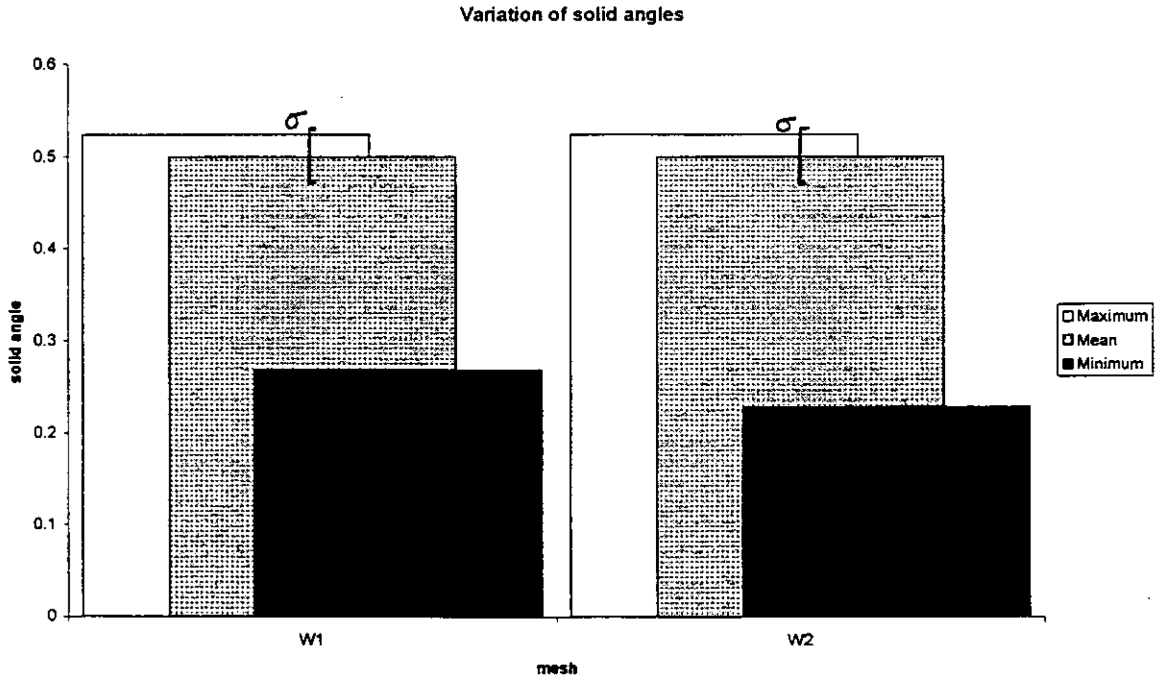

Figure 40: Variation of solid angles for Wigley hullform meshes

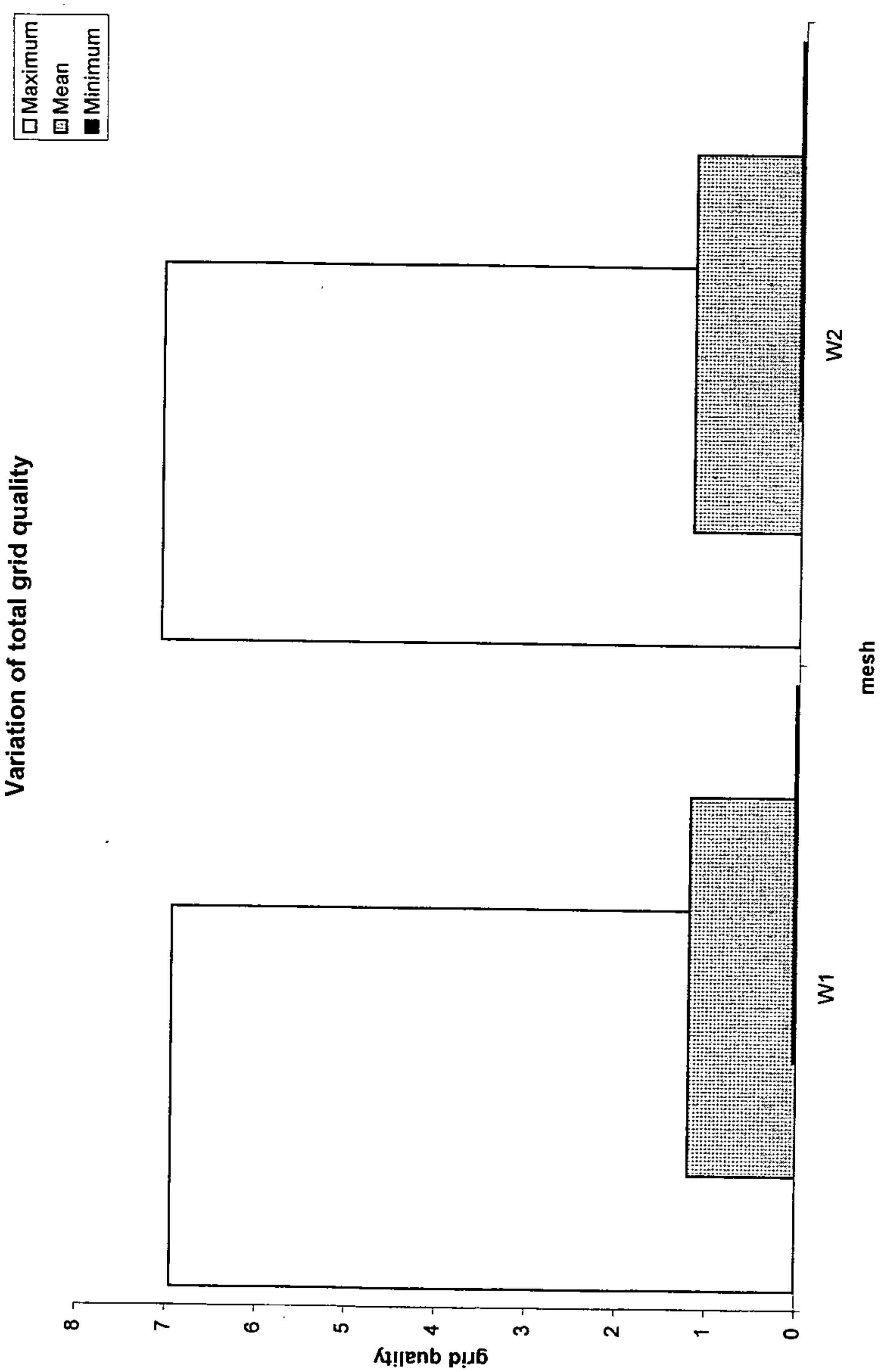

Figure 41: Variation of Wigley hullform grid quality

### 14.2.3 Truncated cylinder

| Mesh | Minimum | Mean | Maximum | Std Deviation |
|---|---|---|---|---|
| S1 | 0.000272 | 0.019033 | 0.095725 | 0.016308 |
| S2 | 0.000300 | 0.019011 | 0.035095 | 0.010136 |

Table 23: Facial area for truncated cylinder meshes

| Mesh | Minimum | Mean | Maximum | Std Deviation |
|---|---|---|---|---|
| S1 | 1.000454 | 2.471489 | 18.913953 | 1.452423 |
| S2 | 1.000454 | 1.907578 | 18.959543 | 0.959447 |

Table 24: Facial aspect ratio for truncated cylinder meshes

| Mesh | Minimum | Mean | Maximum | Std Deviation |
|---|---|---|---|---|
| S1 | 0.000000 | 0.002262 | 0.028943 | 0.005716 |
| S2 | 0.000000 | 0.002012 | 0.033755 | 0.005562 |

Table 25: Facial skew for truncated cylinder meshes

| Mesh | Minimum | Mean | Maximum | Std Deviation |
|---|---|---|---|---|
| S1 | 0.000000 | 0.000195 | 0.005001 | 0.000602 |
| S2 | 0.000000 | 0.000224 | 0.009503 | 0.000932 |

Table 26: Facial deviation from planarity for truncated cylinder meshes

| Mesh | Minimum | Mean | Maximum | Std Deviation |
|---|---|---|---|---|
| S1 | 0.000052 | 0.002615 | 0.013675 | 0.002907 |
| S2 | 0.000130 | 0.002615 | 0.005014 | 0.001760 |

Table 27: Cellular volume for truncated cylinder meshes

| Mesh | Minimum | Mean | Maximum | Std Deviation |
|---|---|---|---|---|
| S1 | 0.506040 | 1.025233 | 1.503211 | 0.092767 |
| S2 | 0.499322 | 1.028810 | 1.721065 | 0.110749 |

Table 28: Ratio of adjacent volume sizes for truncated cylinder meshes

| Mesh | Minimum | Mean | Maximum | Std Deviation |
|---|---|---|---|---|
| S1 | 0.274577 | 0.507893 | 0.778343 | 0.030057 |
| S2 | 0.274577 | 0.513341 | 0.778343 | 0.024189 |

Table 29: Control volume solid angles for truncated cylinder meshes

| Mesh | Minimum | Mean | Maximum | Std Deviation |
|---|---|---|---|---|
| S1 | 0.954843 | 0.993523 | 1.000000 | 0.009660 |
| S2 | 0.946495 | 0.994330 | 1.000000 | 0.009649 |

Table 30: Control volume twist for truncated cylinder meshes

| Mesh | Minimum | Mean | Maximum | Std Deviation |
|---|---|---|---|---|
| S1 | 0.051998 | 0.849084 | 3.292059 | 0.446683 |
| S2 | 0.101700 | 0.564961 | 3.330699 | 0.329237 |

Table 31: Control volume quality for truncated cylinder meshes

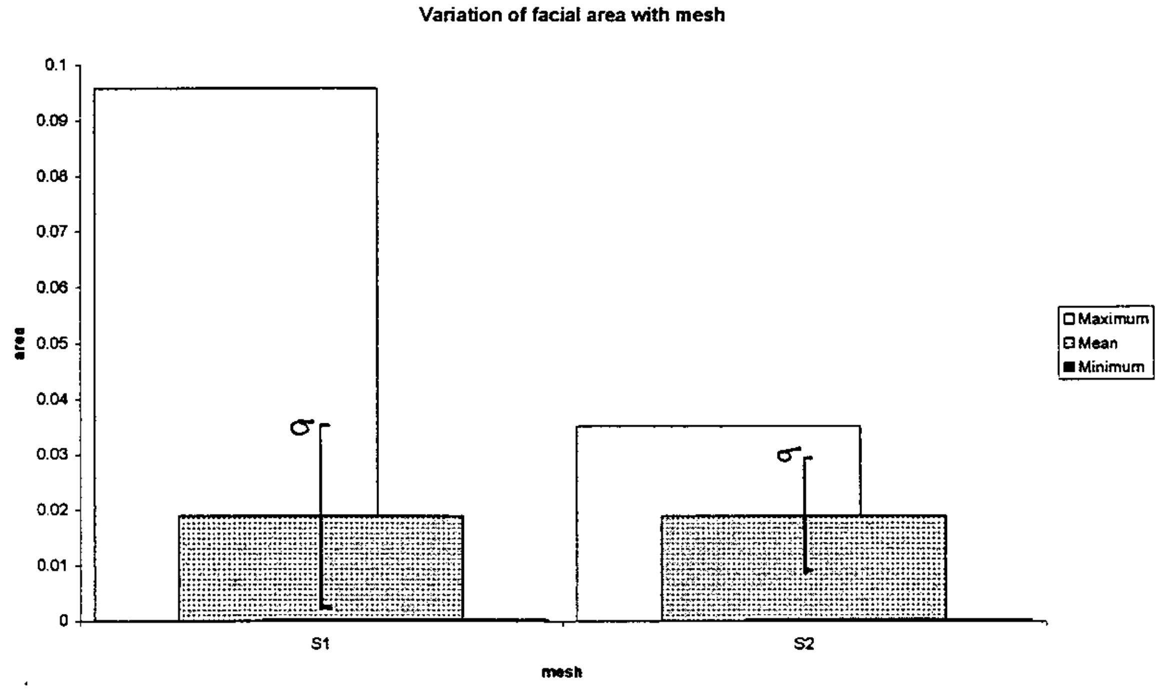

Figure 42: Variation of facial area for truncated cylinder meshes

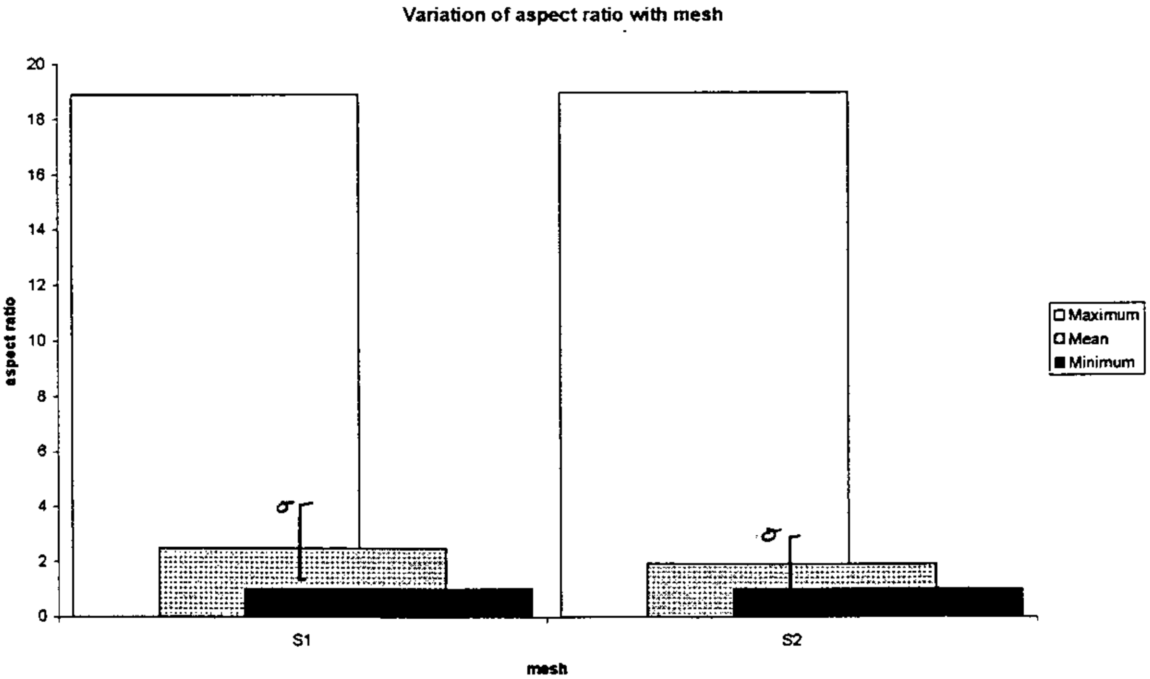

Figure 43: Variation of aspect ratio for truncated cylinder meshes

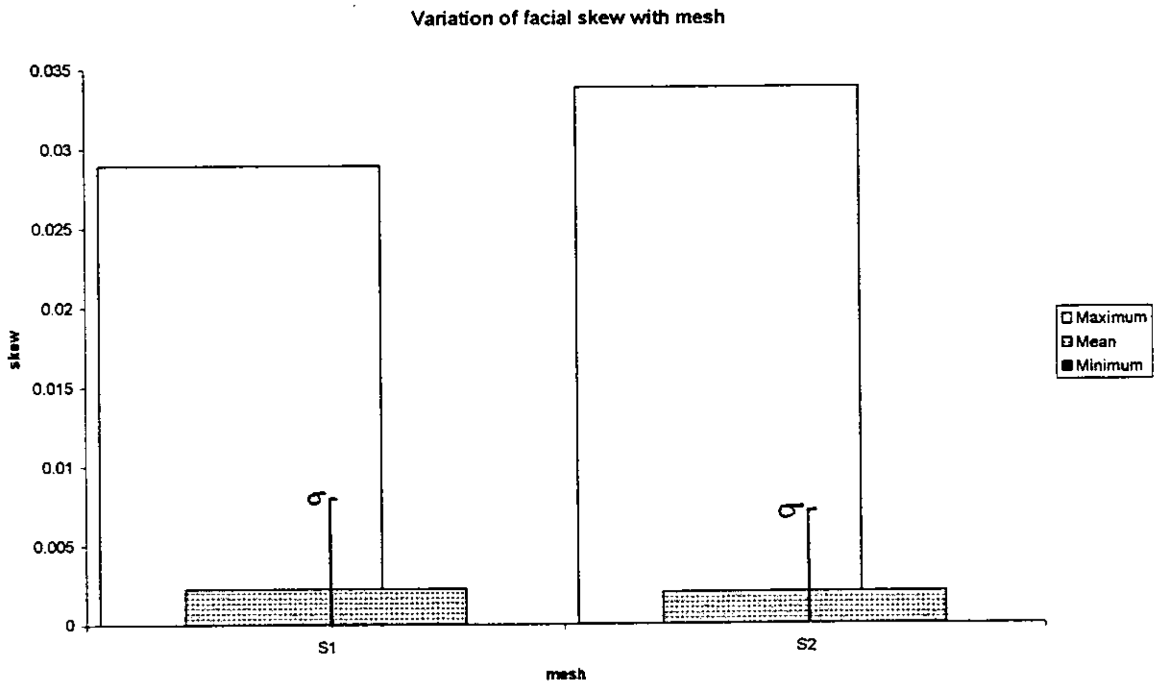

Figure 44: Variation of facial skew for truncated cylinder meshes

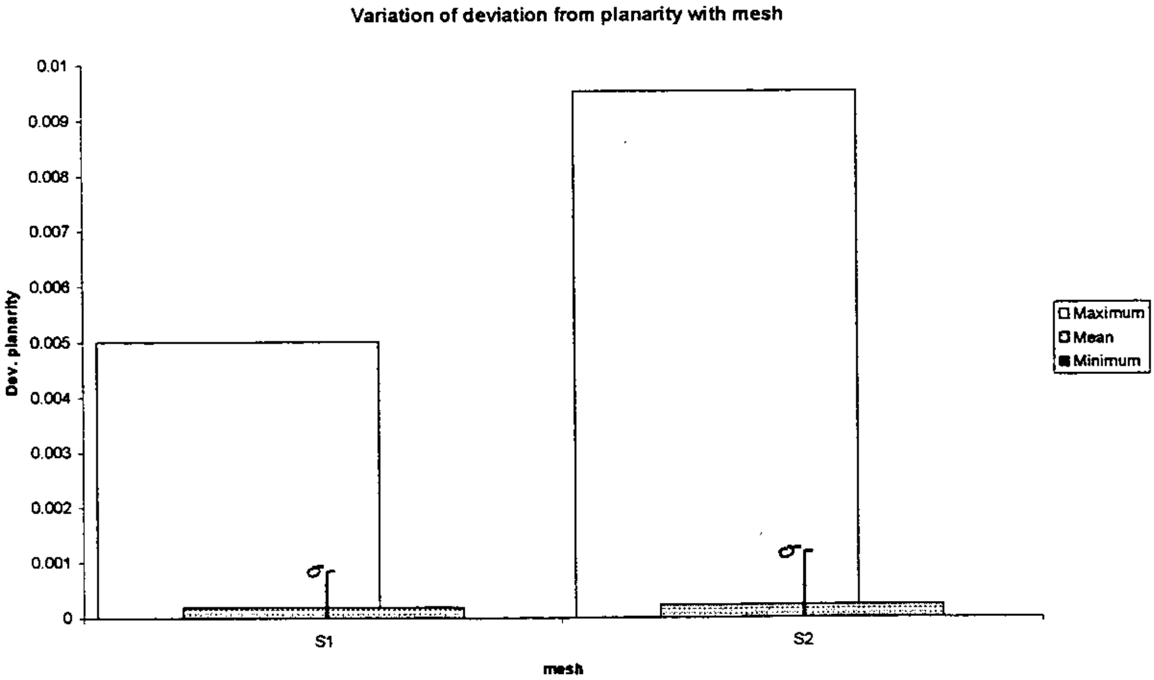

Figure 45: Variation of deviation from planarity for truncated cylinder meshes

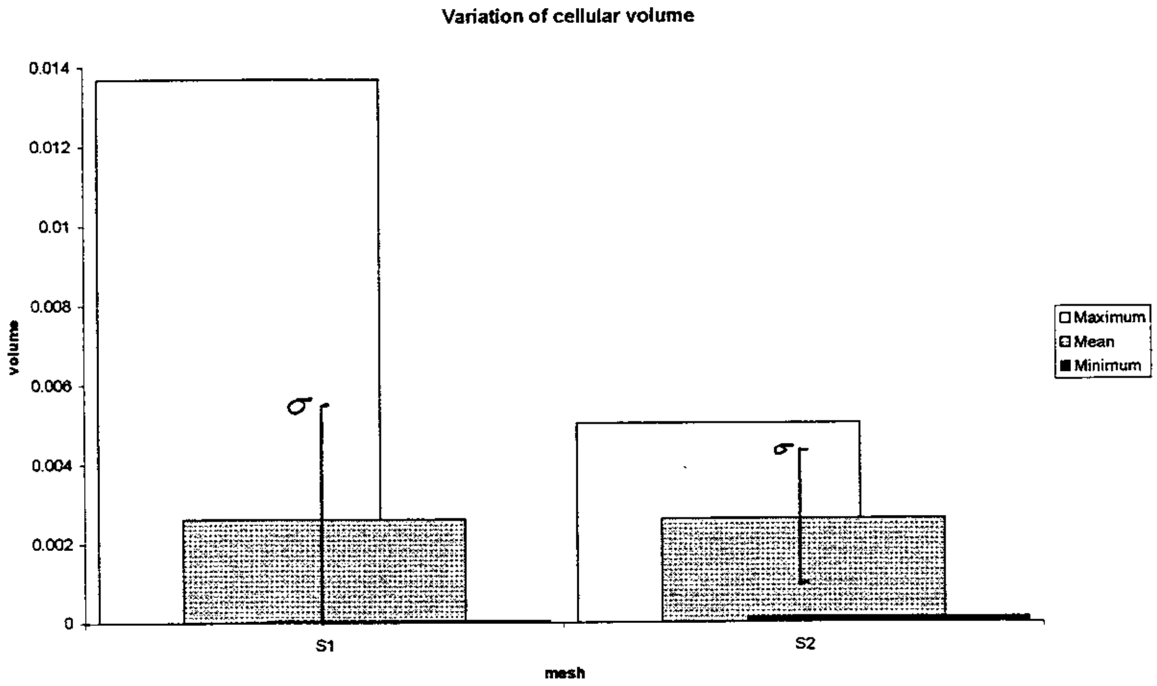

Figure 46: Variation of cellular volume for truncated cylinder meshes

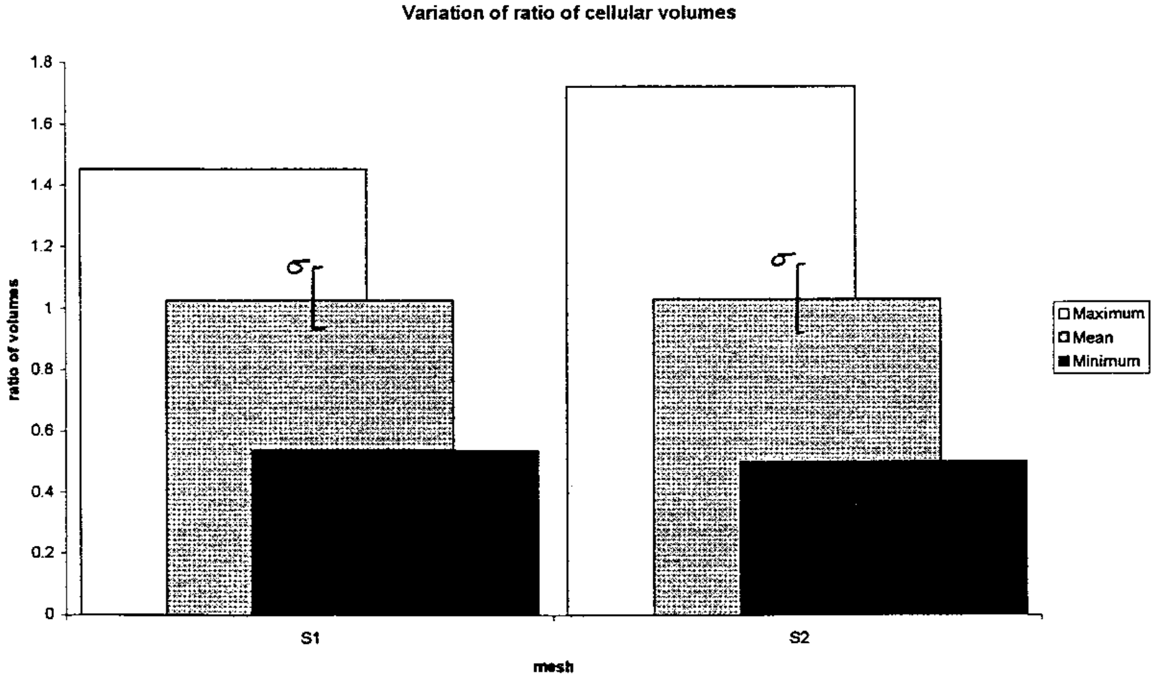

Figure 47: Variation of ratio of cellular volumes for truncated cylinder meshes

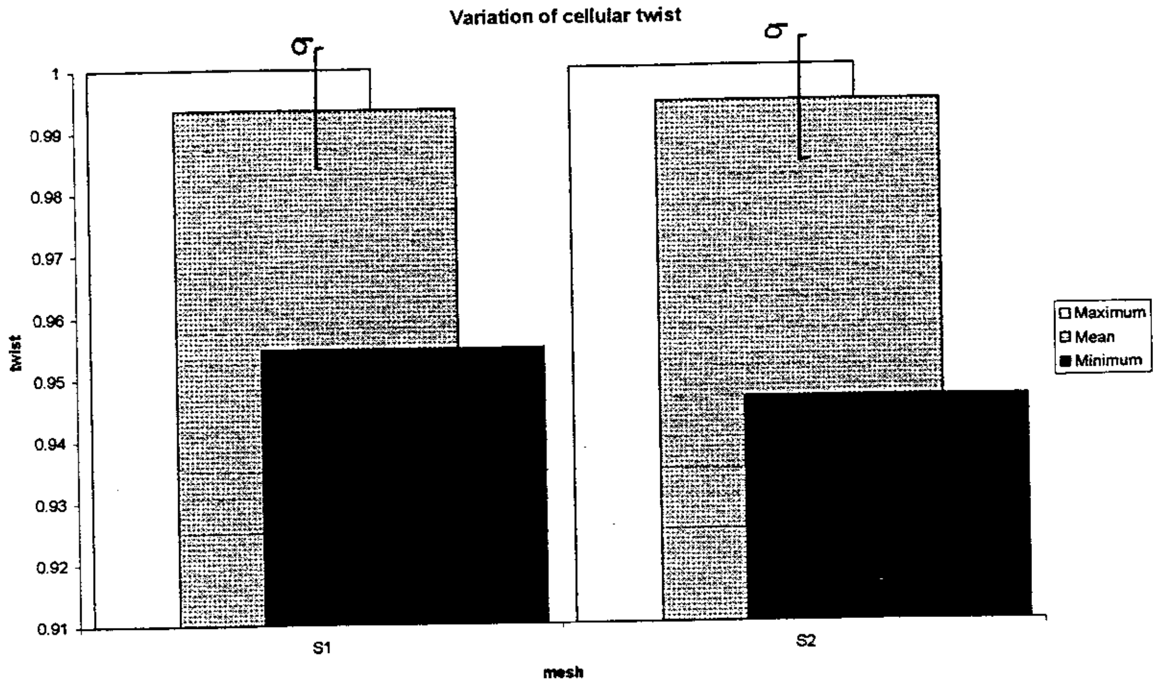

Figure 48: Variation of cellular twist for truncated cylinder meshes

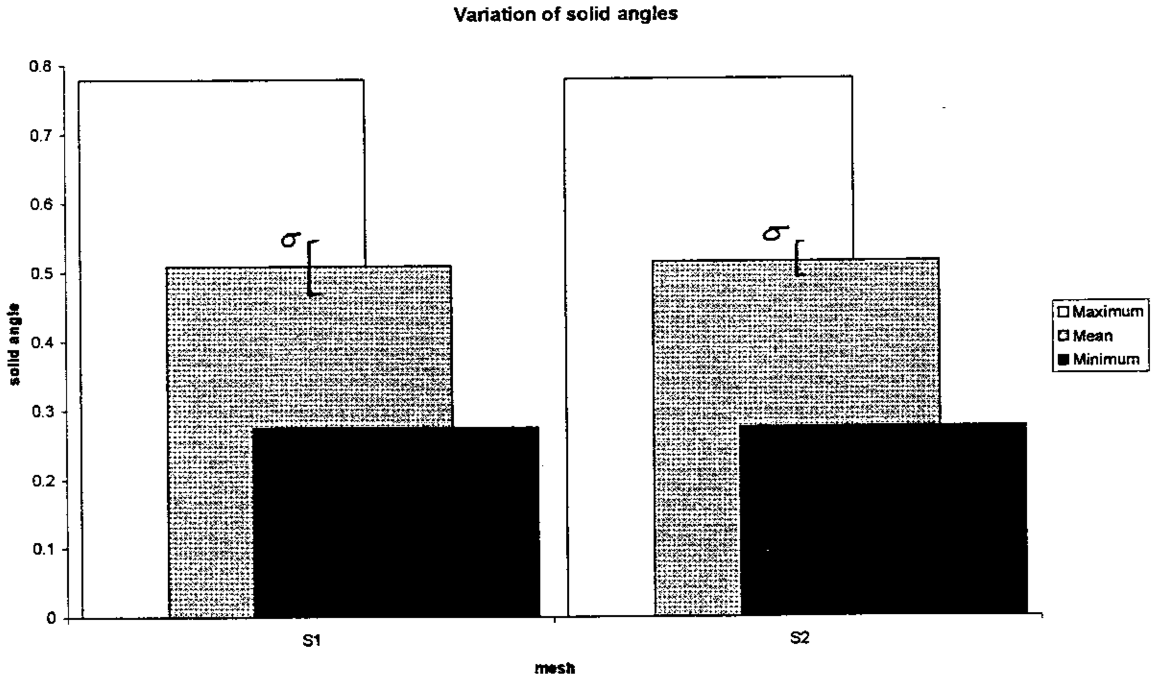

Figure 49: Variation of solid angles for truncated cylinder meshes

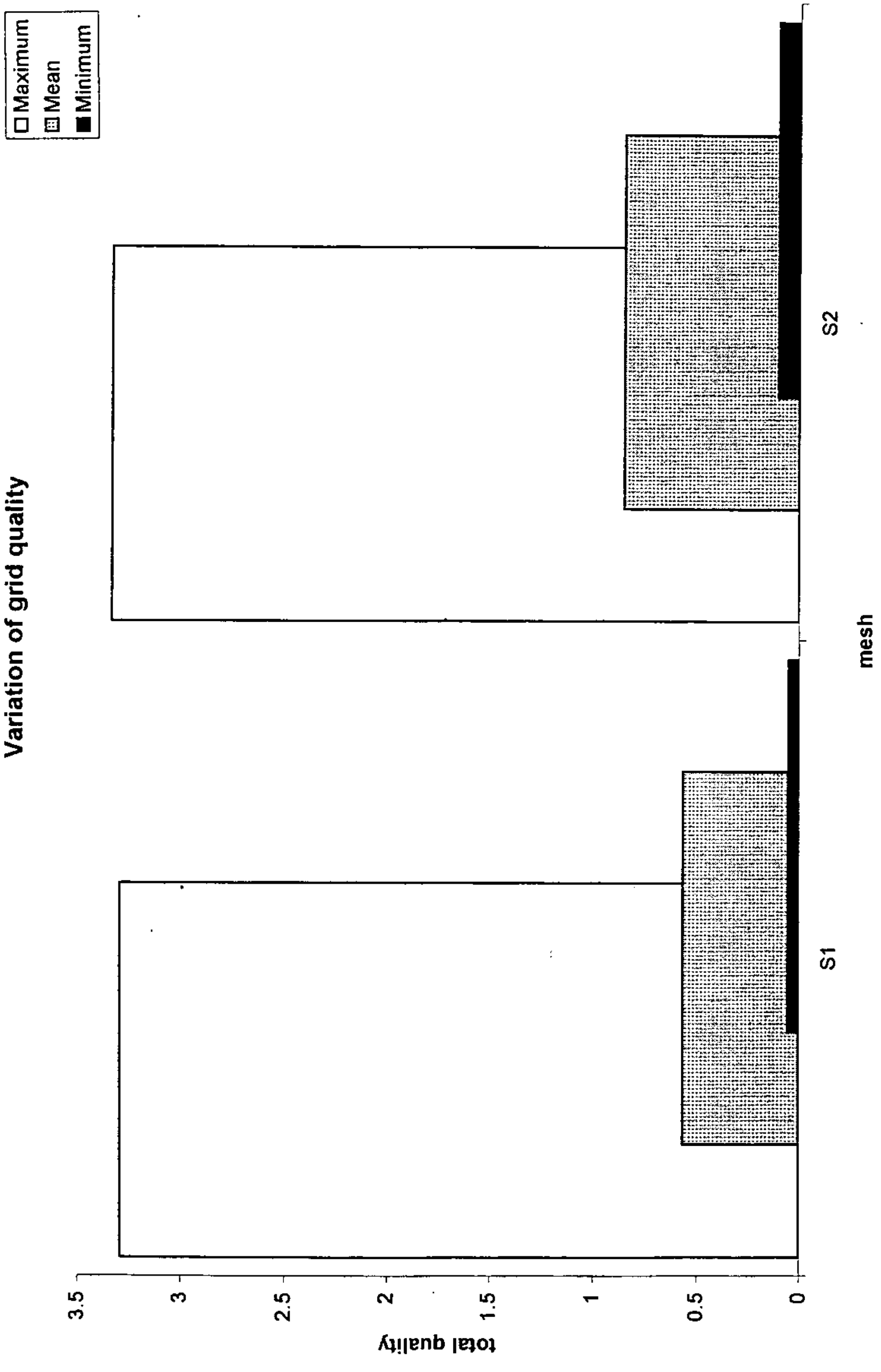

Figure 50: Variation of truncated cylinder grid quality