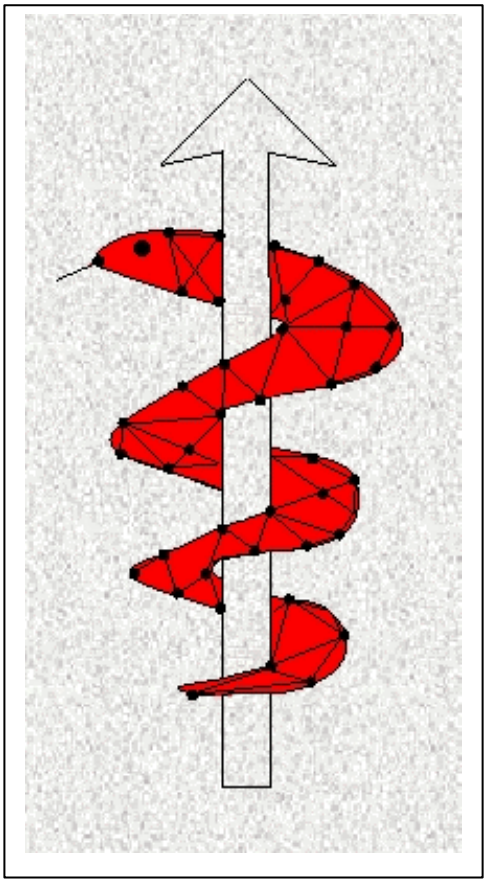
The IST Programme
Project No. 10378

SimBio

SimBio - A Generic Environment for Bio-numerical Simulation

http://www.simbio.de



Deliverable D1.2b
Mesh Generation Release Notes

Status:	final
Version:	1.0
Security:	Public
Responsible:	NEC
Authoring Partners:	NEC, MPI, USFD, ESI

Release History

Version	Date
0.1	2001-04-17
1.0	2001-05-04

The SimBio Consortium :

NEC Europe Ltd. – UK
A.N.T. Software – The Netherlands
CNRS-DR18 – France
ESI Group – France
MPI of Cognitive Neuroscience – Germany
Biomagnetisches Zentrum Jena – Germany
Sheffield University – UK
Smith & Nephew - UK

1. Introduction

This document outlines the status of implemetation of the algorithms designed for the SimBio mesh generation tasks. It should be emphasised that this is a technical report that does not contain the new results achieved in the past months. As described in the deliverable D1.2a there are two techniques for generating finite element meshes that are developed within the SimBio project:

- The VGrid meshing tool
- The Mesh Template Approach

These release notes not only describe the current state of the software but present as well guidelines for accessing, installing and using the tools. The options to be specified by the SimBio user will be thoroughly explained. Questions arising when porting the software to different computer platforms will be covered as well. In the final chapter a roadmap towards completion of both software tools will be given by sketching the technical development plan.

2. Status of the Implementation

2.1 The VGrid Mesher

The main part of the algorithms described in D3a have been implemented in the C++ language using the VISTA library for image data handling and the generation of graphical output.
It is assumed that a 3D segmented MRI dataset is input for our mesh generator. Voxels with the label „zero“ are interpreted as image background and the creation of background elements is suppressed. In our problem domain, the segmentation labels not equal to zero correspond to different objects (i.e. soft tissue, hard tissue, etc.) and/or regions with specific material properties (i.e. electrical conductivity, stiffness).

2.1.1 The Basic Algorithm

Currently the basic version of VGrid is **completely** implemented and runs stable on a variety of computer platforms (Silicon Graphics, LINUX and WINDOWS PC). The basic version of our algorithm simplifies the task of meshing by exploiting the discretisation of 3D space that is a given for any scanned medical dataset. Introducing cells of different size into the 3D dataset yields anisotropic meshes with a significantly reduced number of nodes. In the final step the cells are sequentially processed to generate a tetrahedral mesh. The basic algorithm evolves in three steps:

- Isotropic subdivision of the input image into bricks.
- Collection of bricks to form cells.
- Generation of tetrahedral or hybrid meshes.

The basic algorithm is very fast by exploiting an optimised cache memory accessing scheme. Generating a brick mesh of the femur consisting of approximately 600000 nodes takes about one minute on an SGI machine with a MIPS12000 processor.

2.1.2 Node Shifting

The boundaries of digital or brick meshes can be smoothed by shifting nodes of the FE mesh. The user of VGrid now has the possibility to switch on nodeshifting as a postprocessing step. Furthermore the user governs the degree by which the nodes are shifted by specifying a so called *shift parameter* (see Section 3.1.6).

Node shifting works as follows: For each node of a finite element brick mesh the material types (e.g. skull, brain, meniscus, bone) of all eight elements sharing that node are determined. If a material type is represented by one, two or three of the eight elements, the elements of this material type are

designated as the *minority* elements. Relative to the central node, the spatial position of the centroid of the minority elements is calculated and assigned the coordinates (x, y, z).
It should be noted again that this option is **only available for meshes entirely consisting of hexahedral elements**. If it is switched on with other mesh types (i.e. tetrahedral or hybrid) the user gets a message and the programme quits.

Shifting nodes does not produce a lot of overhead in terms of computation time. It increased the total meshing time by less than 5% for some tested anatomical meshes. Up to now node shifting has passed all tests without failure. In a first SimBio application (related to source localisation) this approach has already shown its usefulness: The accuracy of a numerical result has increased by several percent when compared with the results obtained with a mesh of non-shifted nodes (see Section 4.3.2).

2.1.3 The Marching Tetrahedra Approach

For generating smooth mesh surfaces **for tetrahedral and hybrid meshes** the Marching Tetrahedra Algorithm as described in D1.2a is well suited. The price one has to pay to achieve smoothed surfaces is a higher number of mesh nodes (compared to the unsmoothed mesh) and likewise a longer computation time for simulations based on these meshes. The mesh generation process itself is still very fast despite the increased computational demands. Due to the intelligent cache memory management it takes about half a minute on the abovementioned machine to create a smoothed tetrahedral mesh of the human femur that consists of about 70000 mesh nodes.

The Marching Tetrahedra algorithm has been completely implemented during the last months. Currently this technique enables the creation of smoothed external surfaces and the generation of smoothed internal surfaces presuming that not more than **two** different materials are present within the transistion zone. At present the Marching Tetrahedra module is passing its test phase.

Furthermore an operator has been added to this module that extracts the triangular surface of a tetrahedral mesh object. Such surfaces are useful for visualisation purposes and the SimBio evaluation application related to electric source localisation in the brain.

2.1.4 VGrid Meshes in VISTA Format

The Vista format has been accepted as SimBio internal file format for medical images as well as for finite element meshes. For the latter the Vista graph type has been chosen. More information about the Vista format is to be found on the SimBio webpage. For the last software release two types of mesh formats have been provided depending on the application desired by the use. One format was designed for sequential applications and a second for applications running in parallel. In contrast to the last release for the current version of the meshing software a unique mesh format has been designed for sequential and parallel execution. Furthermore, the new software is featuring a single file approach meaning that all necessary information for the simulation is stored in one mesh file in contrast to earlier versions where a large number of files (e.g. one for each processor) had to be handled. The above statement (only one file for all the information) has to be restricted in the sense that real material parameters (Young's modulus, conductivity tensors. etc.) are not stored within the mesh file. Either this information is read by the simulation code or it is provided by a standalone module that adds the information to the mesh file. This design has been chosen due to practicability and compatibility reasons.

An FE mesh in Vista format consists of two graph structures. A graph is a linked list of elements. The elements of the first graph representing a mesh are the nodal points of a mesh. This graph is referred to as the vertex graph. Vista offers six different types of nodes. In the simplest case only the Cartesian coordinates of the FE node are stored. All the other node types offer the possibility to save additional node related information (e.g. normals, curvature, colour).
Other node related information (e.g. partition information, applied forces) can be stored in additional one-dimensional images that are included to the Vista mesh file.

The second graph is called the primitives graph. Its nodes are the finite elements, i.e. tetrahedra or hexahedra. Finite elements are represented by a set of integer numbers. The links between nodes of the primitive graph can be used to express neighbourhood relationships.
Each Vista file starts with a list of attributes that is human readable (ASCII). Besides compulsory attributes self-defined attributes can be added to the list. Please have a look at the following ASCII header of a VGrid mesh file. The attributes that are explained in the remainder of this section are printed in bold face whilst the others had been clarified in earlier SimBio documents being downloadable via the project webpage.

```
V-data 2 {
    graph: graph {                          #information on the mesh node graph
        data: 0
        length: 4713584
        useWeights: 0
        nnodes: 86481
        nfields: 4
        repn: float
        component_interp: vertex
        partition(s): 1                     #number of partitions
        nverts: 67750                       #number of mesh nodes
        srcnodes: image {                   #a 1D attribute image for the  mesh nodes
            data: 4713584
            length: 8469
            nbands: 1
            nframes: 1
            nrows: 1
            ncolumns: 67750
            repn: bit
        }
        partnode: image {                   #a 1D attribute image for the mesh nodes
            data: 4722053
            length: 67750
            nbands: 1
            nframes: 1
            nrows: 1
            ncolumns: 67750
            repn: ubyte
        }
    }

    graph: graph {                          #information on the mesh node graph
        data: 4789803
        length: 8590512
        useWeights: 0
        nnodes: 437806
        nfields: 5
        repn: long
        component_interp: primitive
        primitive_interp: volume
        nelems: 306804                      #number of mesh nodes
        matprops: image {                   #a 1D attribute image for the mesh elements
            data: 13380315
            length: 306804
            nbands: 1
            nframes: 1
            nrows: 1
```

```
            ncolumns: 306804
            repn: ubyte
        }
        partelem: image {                    #a 1D attribute image for the mesh elements
            data: 13687119
            length: 306804
            nbands: 1
            nframes: 1
            nrows: 1
            ncolumns: 306804
            repn: ubyte
        }
    }
}
```

Comments on the vertex graph attributes
The first additional attribute has the name

partition(s).

Its value specifies the number of processors the SimBio application using this mesh will run on. Typical values for this attribute within the SimBio project range from 1-32.
The next new attribute that is called

nverts

gives the number of nodes in the finite element mesh. Please note that this value usually differs from the number of vertices in the vertex graph.

The next nine lines provide information on a one-dimensional attribute image storing information at which nodes there are possible sources of electrical activity in the human brain.

This information is exploited to create the right-hand-side of the system of linear equations that have to be solved for SimBio applications. The number of columns (*ncolumns*) has to be equal to the number of mesh nodes (*nverts*). As we deal with a one-dimensional image the values of *nrows* and *nbands* have to be set to one. The last line within the curled brackets specifies the data type of the image elements. Here, the only values allowed are 0 and 1 as the image elements are of the bit data type. One can define for instance that at a certain node an electrical source is present by simply switching the corresponding entry in the 1D image on (i.e. setting the entry to one).

The next nine lines provide information on a one-dimensional attribute image storing information on the partitioning of the mesh nodes. By default VGrid generates an initial partitioning for the mesh nodes. The algorithm is most simple: VGrid creates a uniform distribution of the mesh nodes that are spread amongst the processors. Firstly, the total number of mesh nodes (*nverts*) is divided by the number of processors (*npartitions*). Thus, VGrid knows how many nodes will reside on each processor (*nperproc*). Then VGrid starts assigning the „processor ID zero“ to the first *nperproc* nodes within the vertex graph. The following *nperproc* nodes get the „processor ID one“ and so on …
Let us assume we have a mesh of 16 nodes. The application will run on four processors. Then there will be four nodes on each processor. The first four nodes that are in the vertex graph are assigned the value zero (they „live“ on the processor with the ID equal to zero). The assignment is realised by storing the chosen processor ID for the mesh node in the corresponding entry of the 1D partnode image (the global node number of the node is used as index for this attribute image). After the first four nodes have been processed the next four nodes are assigned the ID one and so on.
The last line within the curled brackets specifies the data type of the image elements. Here, values between 0 and 255 are allowed as the image elements are of the unsigned byte data type.

Comments on the primitive graph attributes
The explanations made for the vertex graph attributes are easily transferable to the primitive graph. The first additional attribute introduced for the latest release is the

nelems

attribute that specifies the actual number of finite elements in the mesh. The following nine lines provide information on the material labels that have been assigned to the finite elements during the mesh generation process. These labels are again stored in a one-dimensional image whose elements are of the unsigned byte data type. Obviously, the possible number of entries of the image must correspond to the number of finite elements. The global number of a finite element serves as index to retrieve the material label from the *matprop* image. The following table provides a relation between material labels and tissue types. As mentioned earlier the „real" material properties are finally read by the simulation code. The material labels will be used to enter a material table that contains all the material properties needed by the code.

The *partelem* image is the last attribute image of the example header file. It is very similar to the *partnode* attribute image of the vertex graph. It contains an initial partition for the elements. For advanced partitioning demands the VGrid initial partition serves as input for the DRAMA tool (see D3b).

1	Skull
2	Soft Tissue (as seen on CT)
3	Scalp
4	Muscle
5	Brain
6	White Matter
7	Grey Matter
8	Internal CSF
9	External CSF
10	Dura
11	Hypointense Lesion (i.e., cerebral infarct)
12	Hyperintense Lesion (i.e., tumour)
13	Femur
	to be completed

A table providing the relationship between labels and tissues.

Comments on the faces graph attributes
In the above example there is no faces graph present. We will make use of such a graph type for storing surface information. As mentioned earlier, VGrid offers the option to create triangular surfaces of selected anatomical objects. The triangles making up the surface are organised in a graph structure similar to that of the finite elements. As far as we can see now there is no need for further attributes besides the standard ones. Whatsoever, it would not pose any problem to introduce additional attribute images for faces whenever the demand appears.

2.1.5 The Sheffield Bug Report

The first release of VGrid was made for the Irix OS from SGI. It compiled without error. However, when tested initially it failed to create tetrahedral meshes. The failure was caused by a bug in grid generation, which was fixed promptly by NEC. A further bug was found later in the grid generation code that could cause illegal cells to be entered into the grid. A modification was made by MPI to fix this. The second release of the WP1 MG tool was for the Linux OS that led to various inconsistencies

in behaviour between different versions of Linux and their compilers. In particular, VGrid failed to compile without error using our Linux release, namely Redhat Linux V7 (2.2.16-22 kernel), using the gcc/g++ version 2.96 compiler. Following several discussions with MPI, various modifications were made to the VGrid related files to fix these errors and VGrid now compiles and runs correctly under Linux at USFD.

2.2 The Mesh Templates Approach

2.1.0 Introduction

As an alternative approach for meshing the knee, in the event that VGrid proves unsuitable, USFD is exploring a method of mesh generation that combines the process of mesh generation with image segmentation. It has been termed the Mesh Template (MT) Approach. The technical details of the segmentation process are not covered in this document, as they will form an important part of delayed Deliverable 1.1b, which is not due until Project Month 18. However, the principal features and preliminary results were described in Section 2 of D1.1a, which outlined the functions *vreglocal3d* and *vsegmap3*.

In brief, a fundamental step in achieving image segmentation is to create parameterised meshed models, *or mesh templates*, for each component of the knee. One of the problems in developing automatic segmentation of complex medical data is the level of expert knowledge and manual intervention required to create a "sensible" segmented anatomical structure. This is especially true for MR images of synovial joints because different tissue structures exhibit similar intensity values, thus making intensity based segmentation problematic. By creating an exemplar model for each knee component it will be possible to capture the expert knowledge in the model and use this to steer segmentation. The process of exemplar mesh development itself is time-consuming, however its advantage is that it only needs to be done once, and can be re-used to aid segmentation and to decrease the time required for future mesh generation. A high-quality generic mesh for each component of the knee will exist at the outset of the mesh generation process for a patient. During the non-linear image registration process undertaken during segmentation a mapping function is calculated. This mapping will be used to "morph" the generic mesh to form a patient-specific mesh.

2.2.1 Development of Exemplar Meshes

Initial parameterised exemplar meshes of the menisci were created in Matlab™ as was discussed in Section 3.6 of D1.2a. The exemplar meshes are now being built using the pre-processor of ANSYS™ 5.6. Using ANSYS™ as a pre-processor in the early stages of development has several advantages over using Matlab; it is compatible with the PAM SYSTEM input file interface, PAM GENERIS™, and as such will permit ANSYS input files (collections of nodes and connectivity) to be loaded and converted into a form suitable for PAM-SAFE™. In addition the mesh quality of the initial mesh templates can be checked in ANSYS™. Progress on the creation of the mesh templates is well underway and the template bones of the knee i.e. femur, tibia, fibula and patella have been created and scaled to represent a typical size of knee joint (Figure 1). For comparison purposes, Figure 2 shows a surface model of "live" volunteer knee, scanned as part of Sub Task 7.3 and segmented using 3rd party software 'SurfDriver 3.5" (which required manual intervention). The meshes have been generated using tetrahedral finite elements, as the bones will be assumed to be rigid.

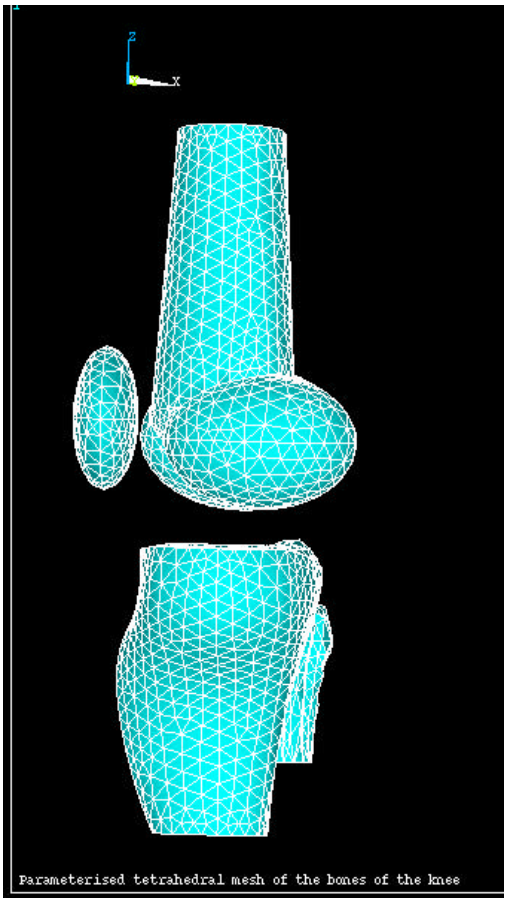


Figure 1 The parameterised femur and tibia meshed with tetrahedral elements

Figure 2 Segmented bones from a "live" knee showing surfaces smoothed using SURFDRIVER,

2.2.2 Developing the Mesh Template Model of the Knee

Other soft tissue components, i.e. the menisci and the articular cartilage, will be meshed using hexahedral elements due to anisotropy issues as discussed in Section 3.5 of D1.2a. USFD intends that the completed mesh template of the knee (i.e. including soft tissues) will be available for the Review meeting in June 2001.

2.2.3 VISTA File Format

USFD intends to utilise the SIMBIO agreed file format, namely VISTA for the mesh template approach. USFD will support this format for the WP1 tools that it will release for segmentation and for mesh template generation.

2.2.4 Target Platform/OS

USFD intends to be compatible with the SimBio WP1 software released already and thus will release for Linux. In addition, for the convenience of USFD and for our industrial partners at Smith and Nephew we will release the MT tools for the SGI platform running under the Irix 6.5 OS.

3. How to Access, Install and Use the Software

3.1 The VGrid Tool

3.1.0 Introduction

The VGrid mesher is part of the Workpackage 1 software. The entire WP1 package is downloadable via the protected part of the SimBio webpage. Two releases are to be found on the webpage: the first on from 2nd of August 2000 and the second dating from the 17th of January 2001.

The following steps have to be followed in order to properly install the software:

1. Unpack the release: tar xzf wp1.tar.gz
2. Change to the directory wp1: cd wp1
3. Revise settings in the configuration file config/site.def and config/linux.cf
4. Make the toplevel makefile: make Makefile
5. Make all other makefiles: make Makefiles
6. Clean: make clean
7. Depend: make depend
8. Build the system: make
9. Install programs, library and documentation: make install (you may need to login as 'root')

For installing the software on a Silicon Graphics machine it is necessary to rename the file **linux.cf** (to be found in the directory *./config*) to **sgi.cf**. The compilation of a standalone version of VGrid using Microsoft Visual C++ did not pose any problem. The files to be found in the *./lib* directory of the directory tree have to be added to the project together with the file vgrid.C residing in the *./pgms/vgrid/* directory.

To get an idea of the options avalaible in the current release one can simlply type

./vgrid

and as a result one obtains a list of command line switches together with their possible parameters. The following lines show the output of the ./vgrid command:

-help
Prints usage information.

-in <string>
Input file. Default: (none)

-out <string>
Output file. Default: (none)

-elem cube | tetra5 | tetra6a | tetra6b
Primitive type. Default: cube

-min <integer>
Minimal grid resolution. Default: 4

-max <integer>
Maximal grid resolution. Default: 4

-smooth no | shift | marching

Smooth boundaries. Default: no

-shift <number>
Degree of node shifting (range 0-0.49). Default: 0.49

-surface [true | false]
Surface mesh output. Default: false

-np <integer>
# of partitions. Default: 1

3.1.1 The –in Option

The -in option specifies the input file for the VGrid mesher. The input file can be any segmented dataset in the SimBio VISTA format. One can also feed MRI raw data into the programme but a sensible material property assignment is critical in this case.

3.1.2 The –out option

The -out option specifies the name of the VGrid VISTA output file. The format of VGrid output files has been discussed in detail in Section 2.1.4. VGrid output files can be visualised with the SimBio VM tool (see D5b).

3.1.3 The –elem option

The –elem option specifies the type of finite elements. Currently it can be chosen amongst the following types:

cube	hexahedral elements
tetra5	tetrahedral elements
tetra6a	tetrahedral elements
tetra6b	tetrahedral elements

The tetrahedral meshes differ from each other in the scheme used to to create tetrahedra by subdividing cubical elements (see D1.2a). In the next release an additional option will be the

hybrid	mixed mesh type

switch. Choosing this options yields mixed meshes consisting of tetrahedra and hexahedra. By default this switch is set to cube as element type.

3.1.4 The –min and –max option

By specifying the min/max options the VGrid user defines the spatial resolution of the mesh. Both options give the minimal and maximal edge length of the cell mesh (phase 2 of the algorithm). The parameters thus govern the number of elements produced. If small structures are present in the image the user is advised to choose a small value (2 or4) for min. Please note that the parameters for min/max can only be **integer number**s as the edge length is measured in the number of voxels the cell is covering. By default min and max are set to the value 4.

3.1.5 The –smooth option

As explained in D1.2a the VGrid user can choose between different strategies to smooth mesh surfaces.

no	no smoothing
shift	nodeshifting is switched on
marching	the marching tetrahedra algorithm is used.

By default *no smoothing* is chosen.

3.1.6 The –shift option

If the –smooth option has been to *shift* (see above) the value of this option becomes relevant. The user gets the chance to influence the degree of nodeshifting. The maximal permitted value for shift is 0.49. A value of 0.5 would cause the creation of degenerate hexahedra. By default the value of –shift option is set to 0.49.

3.1.7 The –surface option

Here the user decides whether he wants a surface mesh or a volumetric mesh as output. This is boolean parameter that can only take on the values *true* or *false*. The latter is set by default which means that volumetric meshes are generated if nothing is specified.

3.1.8 The –np option

The user has to specify already at this early stage on how many processors his application will run. This switch is important for creating an initial partitioning that is optionally improved by the DRAMA tool. The parameter for –np must be an integer number. By default it is set to 1.

4. Future Planning

4.1 Completion of Marching Tetrahedra Algorithm

As explained in this document the Marching Tetrahedra Algorithm has been fully implemented and is currently passing its test phase. Thus, within the next months work on this subject will continue.

4.2 Mesh Quality Module

Smoothing is likely to be of particular importance for the knee where the curvature of the femoral condyles and tibial surface will affect the dynamic behaviour of the knee. It is not yet clear whether the smoothing offered by node shifting will be adequate for the knee. However, this will be tested as part of the validation task undertaken in ST7.3 and should be clearer by **Month 18**. A solution has been proposed that will generate a hybrid mesh of hexahedral and tetrahedral elements, where tetrahedral elements will "fill in" gaps between hexahedral elements to smooth at structural boundaries. The tetrahedral meshes created using VGrid produce meshes that create some subjective concern over the mesh quality, particularly regarding element distortion. An assessment system is being developed as part of ST1.2 to test mesh quality and thus resolve this concern

For the sake of stability and accuracy of any FE simulation carried out with tetrahedral elements the tetrahedra have to satisfy certain quality criteria. As a rule of thumb, angles below 10 degrees are more likely to cause stability problems when solving the equation system. For the tetrahedra resulting from a brick decomposition this requirement is fulfilled in any case. We only find 3 different angles (all greater than 10 degrees) and 3 different edge lengths. For more complex versions of the algorithm such theoretical considerations concerning element shape cannot easily be undertaken. Therefore it is very useful to implement a software module that checks the angles of tetrahedral elements. Such a module automatically invokes simple refinement strategies (e.g. placing an additional node into the centre of the tetrahedron) whenever an angle falls under a certain limit. The implementation of the quality module is planned for the next months.

At this phase of the project we cannot foresee which degree of mesh smoothness will be required by the SimBio evaluation applications. If the approaches described in this design report do not fulfil the requirements imposed by the applications, we will further explore the option of locally smoothing the internal and external mesh boundaries. Promising techniques have been developed to generate C1 smooth surfaces that we could implement as a postprocessing step for the meshes obtained with the algorithms described above.

4.3 Analytical Testing of Mesh Quality

The influence of the nodeshifting procedure is to be validated. This is only possible for simple geometries for which analytical formulae exist (ss D3a and D3b).

4.3.1 Theoretical Aspects

For the forward problem in EEG/MEG-source reconstruction, the potential distribution in the human head for a given dipolar source in the brain is simulated using models for the source and the head. The source is usually modelled as a current dipole, i.e. a source and a sink which are infinitely close together in the human cortical layer. This point-like equivalent current dipole has proven to be an adequate model for the synchronous polarisation of a cortical surface of about 30mm^2 surface. The point-like source leads directly to a singularity in the related potential that has to be treated numerically. One possibility is to introduce the "blurred dipole" i.e. current monopoles are placed at neighbouring FE-mesh nodes around the dipole location such that the resultant moment matches that of the mathematical dipole. An alternative is the subtraction method where the "singularity-potential" for a mathematical dipole in an unbounded homogeneous conductor is calculated analytically and the correction is carried out numerically on the realistic geometry. The correction is calculated with the FE method.

4.3.2 Evaluation Studies

The nodeshifting approach mesh generation approach has been validated in a 4-layer sphere model for which a spherical harmonics series expansion of the dipole potential can be derived. To validate the method we assumed the following isotropic conductivities in the 4 layer model: 0.33, 0.0049, 1.0 and 0.34 S/m for the layers skin, skull, CSF and brain, respectively. Nodeshifting decreased the magnification error (optimum 1) from 1.115 (model without nodeshifting) to 1.053 and the relative difference measure (optimum 0) from 0.027 to 0.023 for six electrodes at all extreme sphere surface positions.
Future studies will be carried out taking more electrodes and different source positions into account.

4.4 Local Mesh Smoothing

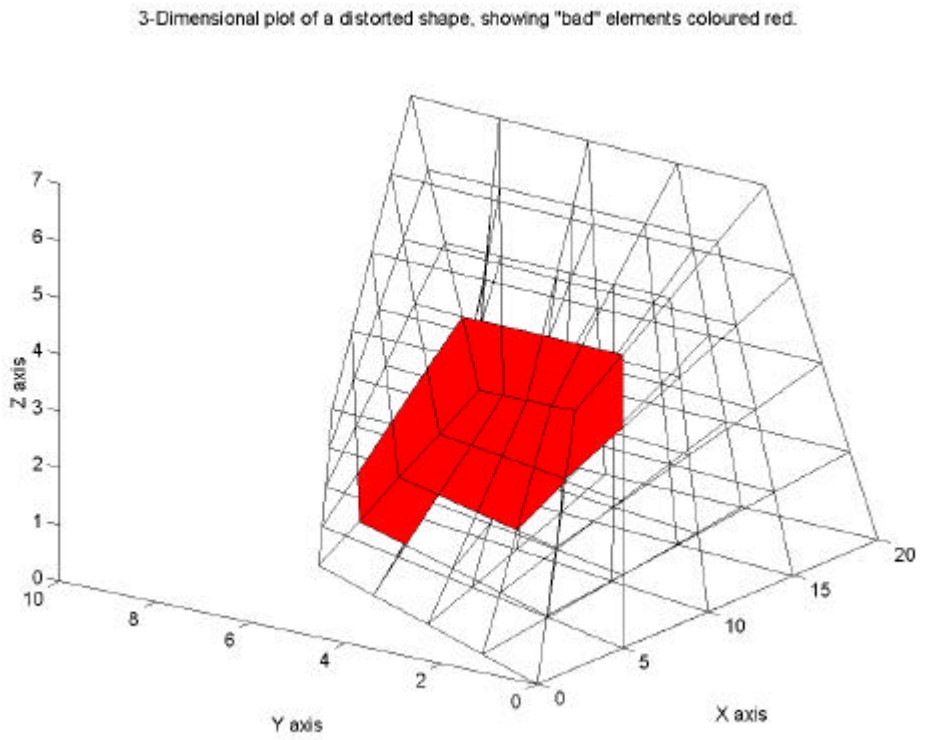


Figure 3 MatlabÔ 3D plot highlighting "bad" elements (based on Jacobian ratios) in red.

At this phase of the project we cannot foresee which degree of smoothness is required by the SimBio evaluation applications. If the implemented techniques will not fulfil the demands imposed by the applications, we will further explore the option of locally smoothing the internal and external mesh boundaries. Promising methods have been developed to generate C1 smooth surfaces that we could implement as a postprocessing step.

4.5 Future Manipulation of MT meshes

The use of ANSYS™ as a pre-processor will not restrict the later development/manipulation of meshes using Matlab™ because the elements of the mesh can be written out by ANSYS™ in a form that can be read by Matlab™. Indeed, the use of Matlab™ will be necessary to undertake quality checking of the morphed meshes and to create pseudo-images from the mesh template to permit template-to-patient image registration. Preliminary work to address the issue of mesh quality for the mesh templates was presented at the 9 Month progress meeting in York. Matlab code has been written to calculate the Jacobian of an element at 8 vertices. Figure 3 shows a plot from a test case demonstrating (in red) distorted elements based on Jacobian ratios. Work will continue up to Month 18 to implement mesh quality checking for the Mesh Template Approach.