A 3-D geometrically realistic finite element mesh of the human heart
Baijens, L.

Published: 01/01/2001

Document Version
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 27. Dec. 2018
A 3-D geometrically realistic finite element mesh of the human heart

L. Baijens

Reportnr. 2001.04
## Contents

1. Introduction  

2. Methods  
   2.1 Introduction  
   2.2 Basic mesh  
   2.3 Deformation  
   2.4 Choice of geometry  

3. Results  
   3.1 Introduction  
   3.2 Basic mesh  
   3.3 Deformation  

4. Discussion and Conclusion  

Bibliography  

A. Mesh generating programs  

B. Mesh deformation programs
Chapter 1

Introduction

The Eindhoven University of Technology, the University of Maastricht and Medtronic work together on a project "Dynamic shape changes of the heart". The aim of this project is to predict the lifetime of the pacemaker-leads and the hindrance that pacemaker-leads undergo due to heart contractions. Their approach is to make a numerical model of the cardiac mechanics using the finite element method.

This research was a part of the cooperation project. The aim of this research project was to make a realistic 3D-model of the whole heart, left and right ventricle. The 3D-model was created in the finite element package Sepran. First a basic model was made that closely fits the shape of the heart. Next this basic mesh was deformed to the actual heart shape.

Chapter 2 discusses the methods that are used to generate the basic mesh and to deform this basic mesh. The results of the mesh generation and the mesh deformation are presented in chapter 3. The discussion and conclusions are dealt with in chapter 4.
Chapter 2

Methods

2.1 Introduction

In this chapter the mesh generation of the whole heart will be discussed. A basic mesh will be generated and then this mesh will be deformed to the realistic heart shape. This approach is chosen instead of directly generating a realistic geometry because it is too difficult to get the nodes at the exact positions of the real heart shape and to distribute the elements evenly over an irregular shape. First the real heart shape and then the creation of a basic mesh will be considered. After that the deformation of the basic mesh will be discussed. At last it is explained why the geometry of the deformed mesh is chosen.

2.2 Basic mesh

First a basic mesh is generated that resembles the human heart shape. The human heart has two atria and two ventricles and the heart shape is illustrated in figure 2.1. Only the ventricles are modeled in this case because they pump the blood out of the heart and because they are thick-walled structures therefore they are easier to model than the thin-walled atria.

Figure 2.1a

Figure 2.1b

Figure 2.1: The geometric shape of a human heart. 2.1a: Transversal slice of the human heart. RV is the right ventricle and LV is the left ventricle. 2.1b: Longitudinal slice of the human heart. RA is the right atrium and LA is the left atrium.
The finite element package Sepran was chosen for the mesh generation because this package makes it possible to use self-made routines and to adjust standard routines. The mesh is created in the pre-processing part of Sepran. The order of basis functions of the elements used is linear and the element type is bricks. Linear bricks are chosen because they are used for simulations of electrical propagation and computations to test the mesh are solved faster using these elements.

In the method for generating a basic mesh of the ventricles the diameter is proportional to the height. This is done so the mesh closely resembles the geometric heart mesh. The created mesh has the shape of a solid cone with one round hole and one half-moon shaped hole in the wall. The round hole is the left ventricular cavity and the half-moon shaped hole is the right ventricular cavity (fig 2.2). The mesh has no distorted elements and the element corners are not deformed.

*Figure 2.2: Top view of the basic mesh. The left ventricular cavity is represented by the hole in the middle with radius radiusss and the right ventricular cavity is represented by the half-moon shaped hole. The radius of the outer surface of the heart is radiusl.*

The radii and the heights displayed in figures 2.2 and 2.3 define the main dimensions of this basic shape. In figure 2.2 the radii are given. The radii on the top of the mesh differ from the radii at the base of the mesh because the mesh becomes smaller at the base. This is done so the basic mesh closely resembles the 'real' mesh of the heart. This is illustrated in figure 2.3. In figure 2.3 the longitudinal cross-section of the basic mesh is given with the different heights and radii. In this figure the XY-plane is shown and the height above it (arrow 1). The arrows 2 and 5 indicate the upper and lower outer radii and arrows 3 and 4 the upper and lower inner radii. Arrow 6 gives the wall thickness. The arrows 7 and 8 are the various heights that are needed for the mesh deformation. The values for the given heights in figure 2.3 are chosen in a way that they closely resemble the data presented in literature (Bovendeerd 1990).
2.3 Deformation

The basic mesh must be deformed to get a good representation of the ventricles. During the deformation process the initial exterior contour moves towards the desired contour and the internal nodes are reallocated, so that a regular element division is accomplished. The deformation is reached by imposing nodal displacements for each node of the exterior contour of the basic mesh. There are two methods of deforming the mesh, which will be explained further.

The first method uses Poisson equations for the deformation. The nodal displacements of the basic mesh are determined by solving three separate 1-D-Poisson equations (one for each coordinate direction) on the 3-D domain of the initial mesh. The equations are given by:

$$\frac{\partial^2 u_i}{\partial x^2} + \frac{\partial^2 u_i}{\partial y^2} + \frac{\partial^2 u_i}{\partial z^2} = f_i \quad i = 1, 2, 3 \quad (2.1)$$

with:
- $u_1$ = nodal displacement in x-direction
- $u_2$ = nodal displacement in y-direction
- $u_3$ = nodal displacement in z-direction
- $f_i$ = source term

Choosing $f_i = 0$ in equation (2.1) ensures a regular division of the nodes on the domain, as there are no source terms that impose a different division of the displacements.
The boundary conditions that hold for equation (2.1) are
\[
\vec{u} = u_0 \quad \text{at the border } \Gamma_0
\] (2.2)
and
\[
\vec{n} \cdot \nabla \vec{u} = 0 \quad \text{at the border } \Gamma_1
\] (2.3)
where \( u_0 \) are the prescribed displacements and \( n \) is the normal.

The equation for the border is
\[
\Gamma = \Gamma_0 + \Gamma_1
\] (2.4)
\( \Gamma_0 \) is total outer surface of the heart and the inner surface of the left ventricle. \( \Gamma_1 \) is the inner surface of the right ventricle. \( \Gamma_0 \) and \( \Gamma_1 \) are illustrated in figure 2.4.

\[\text{Figure 2.4: The border of the basic mesh. } \Gamma_0 \text{ is illustrated with dots and } \Gamma_1 \text{ with stripes. The border is defined in equation 2.4. } \Gamma_0 \text{ is total outer surface of the heart and the inner surface of the left ventricle. } \Gamma_1 \text{ is the inner surface of the right ventricle.}\]

The second method uses mechanical linear elastic elements. In this case the problem is stationary and there are no body forces so the equation that need to be solved for linear elastic problems is
\[
\nabla \cdot \sigma = 0
\] (2.5)
where \( \sigma \) is the stress tensor.

The boundary conditions that hold for equation (2.5) are
\[
\vec{u} = u_0 \quad \text{at border } \Gamma_0
\] (2.6)
and
\[
\sigma \cdot \vec{n} = 0 \quad \text{at border } \Gamma_1
\] (2.7)
where \( \vec{n} \) is the normal.

The stress-strain relations are defined by constitutive equations, which for the linear elastic case are given by a direct linear relation:
\[
\sigma = D \varepsilon
\] (2.8)
where \( D \) denotes the elasticity matrix and \( \varepsilon \) given by strain-displacement relation
\[
\varepsilon = \frac{1}{2} \left[ \nabla \vec{u} + \left( \nabla \vec{u} \right)^T \right]
\] (2.9)
The elasticity matrix $\mathbf{D}$ is dependent on the type of material properties. For this case it is assumed that the material is 3D isotropic, so $\mathbf{D}$ becomes

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

(2.10)

In this case a linear solver will solve the equations. When the displacements are not too large, no overshoot, the elements will not be inside out. The elastic modulus $\mathbf{E}$ and the Poisson’s ratio $\nu$ must be defined to solve the equation. The Poisson’s ratio $\nu$ has a value between 0 and 0.5. When $\nu$ is 0 the material is very compressible and when $\nu$ is 0.5 the material is incompressible. In this case $\nu$ is 0.1 since the material is compressible. The value for the elastic modulus $\mathbf{E}$ is chosen to be 10 (MPa).

### 2.4 Choice of geometry

The real geometry of the human heart is shown in figure 2.1. This geometry should be reached by imposing the displacements on the epicardium and endocardium of both the left and right ventricle. Since the displacements are different for every node and there is no equation defining the displacements, it is very time consuming to define the displacements for each node individually. Therefore an equation that is an approximation of this real geometry is imposed on the surfaces to generate a mesh that closely resemble the real geometry of the heart.

In this case the geometry of the ventricles will be approximated by ellipsoids. The outer and inner contours of the basic mesh will be deformed to the geometry of the left ventricle, thus ignoring the right ventricular cavity. This is done because there was not enough time to deform the heart more realistically. The choice of the value of the parameters defining the geometry is based upon data presented in literature (Bovendeerd 1990).

Bovendeerd (1990) presented the value of the parameters of the ellipsoid axes using experimental data of dogs (fig 2.5 and table 2.1). In figure 2.5 the wall thickness can be computed by subtracting $Z_i$ from $Z_o$. The value for the wall thickness is 7.2 (mm) and this is the same value as the value for the wall thickness of the basic mesh.
Figure 2.5: Geometry of the left ventricle showing the endocardial radius $R_i$, the epicardial radius $R_o$, the endocardial height $Z_i$, the epicardial height $Z_o$ and height above the equator $h$.

Table 2.1: Geometric parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_i$</td>
<td>16.3</td>
</tr>
<tr>
<td>$R_o$</td>
<td>31.3</td>
</tr>
<tr>
<td>$Z_i$</td>
<td>46.0</td>
</tr>
<tr>
<td>$Z_o$</td>
<td>53.2</td>
</tr>
<tr>
<td>$h$</td>
<td>24.8</td>
</tr>
</tbody>
</table>

The equation of the ellipsoid that describes the endocardium of the left ventricle is

$$\frac{(x-x_0)^2}{R_i^2} + \frac{(y-y_0)^2}{R_i^2} + \frac{(z-z_0)^2}{Z_i^2} = 1$$

(2.11)

and the equation of the ellipsoid describing the epicardium of the left ventricle is

$$\frac{(x-x_0)^2}{R_o^2} + \frac{(y-y_0)^2}{R_o^2} + \frac{(z-z_0)^2}{Z_o^2} = 1$$

(2.12)

$R_i$, $Z_i$, $R_o$ and $Z_o$ are illustrated in figure 2.5 and their values are given in table 2.1. In both equations $x_0$, $y_0$ and $z_0$ are zero. In figure 2.6 the surfaces of the basic mesh equation (2.11) holds for are dotted and the surfaces equation (2.12) holds for are line-dot. There are no restrictions imposed on the ventricular endocardium of the right ventricle so it can deform freely. The upper surface of the mesh is fixed in the $Z$-direction so the mesh can not displace in this direction.
Figure 2.6: Surfaces equations (2.11) and (2.12) are imposed on. For the dotted circle equation (2.11) holds and for the line-dot circle equation (2.12) holds.

Using the approximation of the realistic heart shape in the second method of deformation, the mechanical deformation, the basic mesh is deformed to ellipsoids describing the left ventricle. The prescribed displacements from the basic mesh to the deformed mesh \((u, v, w)\) are calculated from the known coordinates \((x, y, z)\) of the basic mesh (fig. 2.7).

Figure 2.7: Deformation of the smaller basic mesh to the ellipsoid. The two arrows are intersecting lines from the top of the mesh through the basic mesh to the ellipsoid. The deformations \((u, v, w)\) refer to the displacement for \((x_1, y_1, z_1)\) to \((x_n, y_n, z_n)\).
Figure 2.7 shows that the point of intersection of the ellipsoid with the line that goes from the origin through the basic mesh must be calculated. Any point \( \vec{v} \) on a line is given by the following equation:

\[
\vec{v} = \lambda \vec{a} + \vec{b} \tag{2.13}
\]

where \( \vec{a} \) represents the direction of the line and \( \vec{b} \) an arbitrary point on the line. In this case \( \vec{a} \) represents the coordinates of the basic mesh and \( \vec{b} \) is point \((0, 0, 24.8)\) illustrated in figure 2.7. So in this case equation (2.13) gives the coordinates of the deformed mesh. When equations (2.11) and (2.12) are combined with equation (2.13) \( \lambda \) can be computed and after that the displacements and the coordinates of the deformed mesh.

When equation (2.13) is combined with equation (2.11) intersection \( \lambda \) can be calculated:

\[
\lambda = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{2.14}
\]

with

\[
a = \frac{x_1^2}{R_1^2} + \frac{y_1^2}{R_1^2} + \frac{(z_1 - z_v)^2}{Z_i^2} \tag{2.15}
\]

\[
b = \frac{2z_v (z - z_v)}{Z_i^2} \tag{2.16}
\]

\[
c = \frac{z_v^2}{Z_i^2} - 1 \tag{2.17}
\]

where \( z_v \) is the height above the XY-plane and \((x_1, y_1, z_1)\) are the coordinates of the ellipsoids. If equation (2.13) is combined with equation (2.12) \( R_0 \) and \( Z_0 \) are used instead of \( R_i \) and \( Z_i \). When \( \lambda \) is calculated then the displacements \((u, v, w)\) can be calculated using the equation

\[
\vec{v} - \vec{a} = (\lambda - 1)\vec{a} + \vec{b} \tag{2.18}
\]

The new coordinates of the deformed mesh can be calculated by substituting \( \lambda \) into equation (2.13).
Chapter 3

Results

3.1 Introduction

In this chapter the results of the mesh generation of the whole heart will be discussed. First the results of the generation of a basic mesh will be considered. Then the results of two alternatives of the basic mesh deformation will be discussed.

3.2 Basic mesh

The method for the generation of the basic mesh generates the shape of a solid cone with one round hole, left ventricular cavity, and a half moon shape hole, right ventricular cavity. This method distorts no elements. The programs for the mesh generation are given in appendix A and the resulting basic mesh is given in figure 3.1 and figure 3.2.

![Figure 3.1: Top view of the basic mesh where the half-moon shaped hole represents the right ventricular cavity and the round hole in the middle represents the left ventricular cavity.](image)
3.3 Deformation

There are two methods to deform the mesh: deformation that uses Poisson equations and mechanical deformation using linear elastic elements. The results of these two deformation methods will be discussed.

There is a problem using the deformation that uses Poisson equations. When the z-displacements are imposed the nodes on the outer contour displace as imposed by the Poisson equation but the internal nodes reallocate in the wrong way. In this case the internal nodes have a greater displacement than the nodes on the external contour so the mesh will be inside out (fig. 3.3). Too large displacements, the overshoot, and a too rough mesh are responsible for the distorted mesh. This distorted mesh is also caused because this deformation is a mechanical problem and it is solved using Poisson equations where $u_1$, $u_2$ and $u_3$ are chosen as nodal displacements. Therefore it is possible that the Poisson equations reallocate the nodes in the wrong way.

Figure 3.2: Side-view of the basic mesh. Node A is the node at the corner where the apex is attached to the solid cone. The angle of the node A must not be too large in order to get a good the deformation of the basic mesh.
Figure 3.3: The deformation is reached by solving a Poisson problem. 3.3a: Displacement of the nodes of the whole mesh. 3.3b: Enlargement of the displacements of the nodes of the apex; 3.3b shows that the displacement of the internal nodes exceeds the displacement of the contour nodes. This causes the mesh to turn inside out.

When using the mechanical deformation there is no overshoot causing the mesh to turn inside out, because overshoot would cause high stresses and they are not possible in a mechanical deformation. The displacements are small and in the resulting mesh the internal nodes are not displaced more than the nodes on the external contour. Figure 3.4 shows the top view of the new mesh and figure 3.5 the side-view. As can be seen the outer contour is an ellipsoid so the displacements are correct. The programs used for the mechanical deformation are given in appendix B.
Figure 3.5: Side-view of the new mesh using mechanical linear elastic elements for the deformation. Node A is the node at the corner where the apex is attached to the solid cone. The angle of the node is acceptable for this deformation, but it is quite large. Adjusting the angle of node A in the basic mesh can solve this problem. By making the angle of node A smaller in the basic mesh the angle in the new mesh will not be so large.

Figure 3.1 and 3.4 differ from each other. It seems that figure 3.4 has more elements but the view is distorted because the outside of the equator of the ellipsoid is larger than the outside of the top so the top view shows also a part of the outside at the equator. In figure 3.1 the outside of the top is larger than the outside of rest of the cone so no extra elements are pictured. The elements in the left ventricular cavity also have a different division in both figures. In figure 3.1 more elements on the inside are visible because the radius cone narrows near the apex. In figure 3.4 there are less elements visible because the inside of the cone has the shape of an ellipsoid and therefore the cone has almost the same radius at different heights. There is also a difference between the right ventricular cavity of the figures. The cavity is smaller in figure 3.4 because of the deformation. The right ventricular cavity has no imposed displacements and it can move freely, therefore it is possible that it becomes smaller.

Figure 3.2 and 3.5 are also different. Figure 3.5 is an ellipsoid and figure 3.2 is an approximation of this ellipsoid. When the nodes on the x=0-axis from both figures are plotted, figure 3.6, it can be seen that there is no overshoot but the deformation is not good. Figure 3.6.b shows that the nodes in the apex do not have a regular distribution and therefore the elements also do not have a regular division. The top element is a lot smaller than the other three elements.
Figure 3.6a

Figure 3.6: The deformation is reached using mechanical linear elastic elements. 3.6a: Displacement of the whole mesh. 3.6b: Displacements of the apex; 3.6b shows that the displacement of the internal nodes does not exceed the displacement of the contour nodes. Figure 3.6b also shows that the elements do not have a regular division. The top element is larger than the other three elements beneath it.
Chapter 4

Discussion and Conclusion

The aim of this research project was to make a realistic 3D-mesh of the whole heart, left and right ventricle, which can be used in finite element computations of cardiac mechanics. First a basic mesh was made and later this mesh was deformed to the heart shape. This approach was chosen instead of directly generating a mesh with the heart shape because it was too difficult to directly get the nodes at the exact positions of the real heart shape and to distribute the elements evenly over an irregular shape.

The method for the creation of the basic mesh generates the shape of a solid cone with one round hole, left ventricular cavity, and a half moon shape hole, right ventricular cavity. This method doesn't cause large deformations and therefore distorts no elements. There are some minor complications using this method that need further discussion. The angles of the elements at the corners are just large enough (fig3.2). It would be better to make them larger so the basic mesh resembles the heart mesh even better.

There are two methods of deformation. The first uses Poisson equations to deform the basic mesh. This method can not be applied because the deformation is a mechanical problem but Poisson equations are used to solve the deformation problem and therefore it is possible that the nodes are reallocated in the wrong way. The internal nodes displace more than the nodes on the contour causing an overshoot (fig3.3b). Too large displacements, the overshoot, and a too rough mesh are responsible for the distorted mesh.

Using the second method of mechanical deformation there is no overshoot because a mechanical problem is solved. The whole mesh (fig 3.5) has the correct form of the imposed ellipsoid. The overall deformation (fig 3.6a) goes well only the apex has an irregular element division that can be observed better in figure 3.6b. Figure 3.6b shows that the top element is very small and the elements under it are large. The division of the nodes of the deformed mesh in this figure is not even. Adjusting $v$ could solve this problem.

When the value of $v$ is made larger then the material is less compressible and the elements should be distributed more evenly since the change in volume will be minimized. The elasticity modulus $E$ is not relevant in this deformation process and $E$ can be any value except zero. When $E$ is zero, the deformation can not take place.

Another way to distribute the elements more evenly is when the relation between the stress and the strain is chosen as non-linear (fig4.1). In this case the beginning of the stress-strain curve is important because the stresses and strains must be as small as possible. When the stress-strain relation is non-linear, and the strain is not large the stress is very small (fig 4.1b) compared to the stress by the same strain in the linear
case (fig 4.1a). Due to these small stresses in the nonlinear case the mesh is less
compressible and the volume can not change easy so the elements will be distributed
more evenly.

When the mesh is deformed a kink is visible at the corners in the new mesh (node A
in fig.3.5). This kink at the corners is caused by the small angle of the basic mesh that
is discussed previously. This small angle in the basic mesh leads to a very large angle
in the mesh after deformation. By adjusting the angle of the basic mesh the mesh after
deformation will be smoother.

The geometry of the whole heart is different from the geometry of figure 3.5 because
the outer and inner contours of the basic mesh are deformed to the geometry of the
left ventricle. The right ventricular cavity is ignored in this method. It is also assumed
that the geometry of the left ventricle can be described by ellipsoids ignoring the real
gemetry of the heart. This is done because there wasn’t enough time to deform the
basic mesh to the real heart geometry because each node has to be displaced by
another equation.

The data used to obtain the geometric parameters of the left ventricle are of a dog’s
heart instead of a human heart because this is the first test of the mesh and it is not
relevant what equation is imposed when deforming the mesh. Also the data of the
human heart were not available at the time.

To make the geometry of the mesh more realistic the data of a human heart should be
imported in Matlab and the contour of the whole heart should be defined. This
geometry should then be implemented in the deformation program and should be
imposed on the right surfaces of the mesh in order to make the mesh more realistic.

In short the basic mesh is generated using the method which generates the shape of a
solid cone with one round hole, left ventricular cavity, and a half moon shape hole,
right ventricular cavity. Because the deformation using the Poisson equations causes an
overshoot this method can not be used. So the mechanical method is used to deform
the mesh using linear elastic equations. The data of the left ventricle of a dog’s heart
are used to deform the whole mesh ignoring the right ventricular cavity. The ultimate generated heart mesh is illustrated in figure 3.4 and 3.5.

Recommendations for future research are implementing the data of the left and right ventricle of a human heart in the deformation program to make a more realistic mesh and adjusting \( v \) and the angle in the basic mesh to obtain a more evenly distributed element division.

There is also an alternative method for the generation of the basic mesh. In this method the existing mesh of the left ventricle generated by Bovendeerd (1990) is extended with the mesh of the right ventricle (fig 4.2).

![Figure 4.2: Alternative basic mesh where the right ventricle, left ring, is coupled with the existing left ventricle, right ring. The elements at the connection are not distorted and it will be easier to deform them to the real heart shape then the elements in the basic mesh generated in chapter 2 and 3.](image)

The whole ring represents the existing left ventricle and the half-ring represents the right ventricle. This alternative method for the generation of the basic mesh is better than the basic mesh generated with the method described in chapter 2 and 3 because the deformation of the basic mesh of chapter 2 and 3 to the real human heart shape will distort elements. The elements at point A and B in the basic mesh of chapter 2 and will be distorted (fig. 4.3). This will not happen with the elements of the alternative basic mesh.

![Figure 4.3: The basic mesh generated in chapter 2 and 3. When deforming this basic mesh to the real heart shape the elements containing node A and B will be distorted. Therefore another method for generating the basic mesh is necessary.](image)
Bibliography

Appendix A

Mesh generating programs

The main program is heart.msh:

```
#heart.msh
constants
integers
nelmh=6  # elementen in de hoogte
nelml=4  # elementen in de lengterichting van de arc
nelmwin=2 # elementen in de binnenste radiale richting
nelmwout=2 # elementen in de buitenste radiale richting
nelslw
reals
afstandrv=4   # afstand tussen de 2 middelpunten
radiussso=16  # straal van de bovenste kleinste cirkel
radiussso=24  # straal van de onderste kleinste cirkel
radiussso=24  # straal van de 2de middelste cirkel
radiussso=32  # straal van de 3de bovenste cirkel
radiuslo     # straal van de 3de onderste cirkel
radiusrvs    # straal vanaf het middelpunt
straalblin   # straal voor binnenste flappen
straalmid    # straal voor middelste flappen
straalbui    # straal voor buitenste flappen
height = 24.8 # hoogte van het hart
height_one = -50 # hoogte punten buitenste cirkel
height_two = -40 # hoogte punten middelste cirkel(-49.6)
height_three = -40 # hoogte punten binnenste cirkel
height_four  # hoogte punten van buitenste flappen
height_five  # hoogte punten van middelste flappen
height_six   # hoogte punten van binnenste flappen
angles=90     # 90 graden hoek
angles=180    # 180 graden hoek
anglel=270    # 270 graden hoek
hoekss        # hoek die de helft is van angles
hoekl         # hoek die de helft is van angless
hoekxl        # hoek die de helft is van de laatste kwart van het hart
end

mesh3d
points
# punten van de eerste cirkel
p1=(0.0,$height_two) # centroid1
pd2=($radiusssbo,0,$height_three)
```

20
pd3=($radiusss,$angless,$height-three)
pd4=($radiusss,$angless,$height-three)
pd5=($radiusss,$anglel,$height-three)
p6=(0,0,$height)
pd7=($radiusssb,0,$height)
pd8=($radiusssb,$angless,$height)
pd9=($radiusssb,$angless,$height)
pd10=($radiusssb,$anglel,$height)

#punten van de tweede cirkel
pd11=($radiusso,0,$height_two)
pd12=($radiusso,$angless,$height_two)
pd13=($radiusso,$angless,$height_two)
pd14=($radiusso,0,$height)
pd15=($radiusso,$angless,$height)
pd16=($radiusso,$angless,$height)

#extra middelpunt grondvlak
p17 = (0,0,$height_one)

#punten van de derde cirkel
pd18=($radiuslo,0,$height_one)
pd19=($radiuslo,$angless,$height_one)
pd20=($radiuslo,$angless,$height_one)
pd21=($radiuslo,$anglel,$height_one)
pd22=($radiuslb,0,$height)
pd23=($radiuslb,$angless,$height)
pd24=($radiuslb,$angless,$height)
pd25=($radiuslb,$anglel,$height)

#punten rechterevenrikel
p26=(0,$afstandrv,$height-two) # centroid2 onder
p27=(0,$afstandrv,$height)  # centroid2 boven

#punten voor creeren van de flappen
pd29=($straallb,$hoekss,$height_five)
pd30=($straallb,$hoekss,$height_five)
pd31=($straallb,$hoekss,$height_five)
pd32=($straallb,$hoekss,$height_five)
pd33=($straalle,$hoekss,$height_four)
pd34=($straalle,$hoekss,$height_four)
pd35=($straalle,$hoekss,$height_four)
pd36=($straalle,$hoekss,$height_four)
pd37=($straalmid,$hoekss,$height_six)
pd38=($straalmid,$hoekss,$height_six)

curves
# lijnen van de eerste cirkel
c1 =arc1(p2,p3,p29,nelm=$nelml)
c2 =arc1(p3,p4,p30,nelm=$nelml)
c3 =arc1(p4,p5,p31,nelm=$nelml)
c4 =arc1(p5,p2,p32,nelm=$nelml)
c5 =arc1(p7,p8,p6,nelm=$nelml)
c6 =arc1(p8,p9,p6,nelm=$nelml)
c7 =arc1(p9,p10,p6,nelm=$nelml)
c8 =arc1(p10,p7,p6,nelm=$nelml)
c9 =line1(p2,p7,nelm=$nelmh)
c10=line1(p3,p8,nelm=$nelmh)
c11=line1(p4,p9,nelm=$nelmh)
c12=line1(p5,p10,nelm=$nelmh)
# lijnen van de tweede cirkel
cl3=arcl(p12,p37,nelm=$nelml)
cl4=arcl(p12,p13,p38,nelm=$nelml)
cl5=arcl(p14,p28,p27,nelm=$nelml)
cl6=arcl(p28,p16,p27,nelm=$nelml)
cl7=arcl(p11,p14,nelm=$nelmh)
c15=arcl(p14,p15,p6,nelm=$nelml)
c18=arcl(p15,p16,p6,nelm=$nelml)
c19=arcl(p11,p14,nelm=$nelmh)
c20=arcl(p12,p15,nelm=$nelmh)
c21=arcl(p13,p16,nelm=$nelmh)
# spaken van de tweede cirkel
c22=arcl(p2,p11,nelm=$nelmwin)
c23=arcl(p3,p12,nelm=$nelmwin)
c24=arcl(p4,p13,nelm=$nelmwin)
c25=arcl(p7,p14,nelm=$nelmwin)
c26=arcl(p8,p28,nelm=$nelmwin)
c27=arcl(p9,p16,nelm=$nelmwin)
# lijnen van de derde cirkel
c28=arcl(p18,p19,p33,nelm=$nelml)
c29=arcl(p19,p20,p34,nelm=$nelml)
c30=arcl(p20,p21,p35,nelm=$nelml)
c31=arcl(p21,p18,p36,nelm=$nelml)
c32=arcl(p22,p23,p6,nelm=$nelml)
c33=arcl(p23,p24,p6,nelm=$nelml)
c34=arcl(p24,p25,p6,nelm=$nelml)
c35=arcl(p25,p22,p6,nelm=$nelml)
# spaken van de derde cirkel
c36=arcl(p11,p18,nelm=$nelmwout)
c37=arcl(p12,p19,nelm=$nelmwout)
c38=arcl(p13,p20,nelm=$nelmwout)
c39=arcl(p5,p21,nelm=$nelmwl)
c40=arcl(p14,p22,nelm=$nelmwout)
c41=arcl(p15,p23,nelm=$nelmwout)
c42=arcl(p16,p24,nelm=$nelmwout)
c43=arcl(p10,p25,nelm=$nelmwl)
# extra rechterventrikel
c44=arcl(p12,p28,nelm=$nelmh)
# lijnen in derde cirkel
cl45=arcl(p18,p22,nelm=$nelmh)
c46=arcl(p19,p23,nelm=$nelmh)
c47=arcl(p20,p24,nelm=$nelmh)
c48=arcl(p21,p25,nelm=$nelmh)
# extra curves voor oppervlakten
c49=curves(c22,c36)
c50=curves(c24,c38)
c51=curves(c25,c40)
c52=curves(c27,c42)
c53=curves(c23,c37)
surfaces
# eerste deel tussen ventrikels
s1=coons5(c1,c10,-c5,-c9)
s2=coons5(-c13,c19,c15,-c44)
s3=coons5(c1,c23,-c13,-c22)
s4=coons5(c5,c26,-c15,-c25)
s5=rectangle5(-c23,c10,c26,-c44)
s6=rectangle5(-c22,c9,c25,-c19)  # achtervlak

tweede deel tussen de ventrikels
s7 =coons5(c2,c11,-c6,-c10)  # binnenvlak
s8 =coons5(-c14,c44,c16,-c21)  # buitenvlak
s9 =coons5(c2,c24,-c14,-c23)  # ondervlak
s10=rectangle5(c6,c27,c16,-c26)  # bovenvlak
s11=rectangle5(-c24,c11,c27,-c21)  # voorvlak

eerste deel buitenwand
s12=coons5(c3,c12,-c7,-c11)  # binnenvlak
s13=coons5(-c30,c47,c34,-c48)  # buitenvlak
s14=coons5(c3,c39,-c30,-c50)  # ondervlak
s15=coons5(c7,c43,-c34,-c52)  # bovenvlak
s16=rectangle5(-c50,c11,c32,-c47)  # achtervlak
s17=rectangle5(-c39,c12,c43,-c48)  # voorvlak

tweede deel buitenwand
s18=coons5(c4,c9,-c8,-c12)  # binnenvlak
s19=coons5(-c31,c48,c35,-c45)  # buitenvlak
s20=coons5(c4,c49,-c31,-c39)  # ondervlak
s21=coons5(c8,c51,-c35,-c43)  # bovenvlak
s22=rectangle5(-c49,c11,c52,-c45)  # achtervlak

eerste helft buitenwand rechterventrikel
s23=coons5(c13,c20,-c17,-c19)  # binnenvlak
s24=coons5(-c28,c45,c32,-c46)  # buitenvlak
s25=coons5(c13,c37,-c28,-c36)  # ondervlak
s26=coons5(c17,c41,-c32,-c40)  # bovenvlak
s27=rectangle5(-c37,c20,c41,-c46)  # voorvlak
s28=rectangle5(-c36,c19,c40,-c45)  # achtervlak

tweede helft buitenwand rechterventrikel
s29=coons5(c14,c21,-c18,-c20)  # binnenvlak
s30=coons5(-c29,c46,c33,-c47)  # buitenvlak
s31=coons5(c14,c38,-c29,-c37)  # ondervlak
s32=coons5(c18,c42,-c33,-c41)  # bovenvlak
s33=rectangle5(-c38,c21,c42,-c47)  # voorvlak

extra oppervlakten voor de apex
s34=coons5(c1,c53,-c28,-c49)  # zijvlak
s35=coons5(c2,c50,-c29,-c53)  # zijvlak
s36=coons5(c1,c2,c3,c4)  # bovenvlak
s37=coons5(c28,c29,c30,c31)  # ondervlak

volumes
v1=brick13(s3,s5,s1,s6,s2,s4,orientation = 716117)
v2=brick13(s9,s11,s7,s5,s8,s10,orientation = 716117)
v3=brick13(s14,s17,s12,s16,s13,s15,orientation = 716117)
v4=brick13(s20,s22,s18,s17,s19,s21,orientation = 716117)
v5=brick13(s25,s27,s23,s28,s24,s26,orientation = 716117)
v6=brick13(s31,s33,s29,s27,s30,s32,orientation = 716117)

het apexvolume
v7=brick13(s37,s34,s35,s14,s20,s36,orientation = 188331)

meshvolume
velml=(v1,v7)
renumber
plot, eyepoint=(12,-16,8)
end

# значит комментарий
A program that calculates the parameters defined in the main program is called compcons.f and is given below.

```fortran
subroutine compcons
  implicit none
  include '/cuscons.f'

  double precision radiusrv, verplaatsing, opschuiving_one, opschuiving_two,
               opschuiving_three, getal, verschil

  c elementen in lange radiele ricting

  nelmwl = nelmwin + nelmwout

  c berekenen van de hoogte van de middelste cirkel

  height_two = -((height_three - height_one)/2) + height_three

  c bereken van de straal van de middelpunten van de flappen

  getal = 10
  straalbin=radiusso + getal
  straalmid=radiusso + getal
  straalbui=radiuslo + getal

  c berekenen van de hoeken voor de flappen van de apex

  hoekss = angless / 2
  hoeks = angles - hoekss
  hoekl = angell - hoekss
  hoekxl = angell + hoekss

  c berekenen van de hoogte van de middelpunten van de flappen

  verschil = 7
  height_four = height_one + verschil
  height_five = height_three + verschil
  height_six = height_two + verschil
  write(*,*), height_four = ', height_four

  c berekeningen van de straal van de rechterventrikel
  c ten opzichte van het extra middelpunt

  radiusrv = sqrt((afstandrv*afstandrv) + (radiussb*radiussb))

  c berekening van de straal van rechterventrikel
  c t.o.v. het middelount

  radiusrvs = radiusrv-afstandrv

  c verplaatsing = de afstand die de onderste punten verschuiven
  c t.o.v. de bovenste punten

  verplaatsing = radiussb-radiusrvs
```

c berekening van de stralen van het ondervlak

opschuiving_one = ((verplaatsing/(height + height_one))*(height+height_one))*2
opschuiving_two = ((verplaatsing/(height + height_one))*(height+height_two))*2
opschuiving_three = ((verplaatsing)/(height + height_one))*(height+height_three))*2
radiuslo = radiuslb-opschuiving_one
radiusso = radiussb-opschuiving_two
radiussso = radiusssb-opschuiving_three

c opschrijven van de uitkomsten op het scherm
write(*,*)' radiusrv = ', radiusrv
write(*,*)' radiusrvs = ', radiusrvs
write(*,*)' radiuslo = ', radiuslo
write(*,*)' radiusso = ', radiusso
write(*,*)' radiussso = ', radiussso
write(*,*)' height_two = ', height_two
end
Appendix B

Mesh deformation programs

The main program for the mesh deformation is meshcreation.f.

```fortran
program createmesh3d
    implicit none
    call sepcom(0)
end
```

---

The subroutine `funcbc` is used to calculate the nodal displacements:

```fortran
function funcbc(ifunc,x,y,z)
    implicit none
    integer ifunc,t
    double precision funcbc,x,y,z,deltax,deltay,deltaz,
    +       xnew, ynew, znew, zv,
    +       Zi, Zo, Ri, Ro, xratio, yratio,
    +       zratio, lambda, a, b, c
```

- If `ifunc=1`, `funcbc` returns the displacement along the inner contour.
- If `ifunc=2`, `funcbc` returns the displacement along the outer contour.
- If `ifunc=3`, `funcbc` returns the displacement along the inner contour.
- If `ifunc=4`, `funcbc` returns the displacement along the inner contour.
- If `ifunc=5`, `funcbc` returns the displacement along the outer contour.
- If `ifunc=6`, `funcbc` returns the displacement along the outer contour.
include 'SPcommon/consta'

c --- Ellipsoid properties
zv=24.8d0
Ri=16d0
Ro=32d0
Zi=46.0d0
Zo=53.2d0
c --- Computation of deltax and deltay
c
write (*,*) 'x=', x
write (*,*) 'y=', y
write (*,*) 'z=', z
c berekening van de verplaatsingen
if ((ifunc.eq.1) .or. (ifunc.eq.2) .or. (ifunc.eq.3)) then
  xratio=(x*x)/(Ri*Ri)
  yratio=(y*y)/(Ri*Ri)
  zratio=((z-zv)*(z-zv))/(Zi*Zi)
  a= xratio+yratio+zratio
  b= (2*zv*zv)/(Zi*Zi)
  c= ((zv*zv)/(Zi*Zi)) -1
  lambda=(-b + sqrt((b*b)-(4*a*c)))/(2*a)
  xnew=lambda*x
  ynew=lambda*y
  znew=(z-zv)*lambda+zv
  deltax=xnew-x
  deltay=ynew-y
  deltaz=znew-z
write(*,*) x,y,z,xnew,ynew,znew
if ( ifunc .eq. 1 ) then
  funcbc=deltax
  write(*,*) 'hij doet het '
endif
if ( ifunc .eq. 2 ) then
  funcbc=deltay
  write(*,*) 'hij doet het 2'
endif
if ( ifunc .eq. 3 ) then
  funcbc=deltaz
  write(*,*) 'hij doet het 3'
endif
elseif ((ifunc.eq.4) .or. (ifunc.eq.5) .or. (ifunc.eq.6)) then
  xratio=(x*x)/(Ro*Ro)
  yratio=(y*y)/(Ro*Ro)
  zratio=((z-zv)*(z-zv))/(Zo*Zo)
  a= xratio+yratio+zratio
  b= (2*zv*zv)/(Zo*Zo)
  c= ((zv*zv)/(Zo*Zo)) -1
endif
\[ \lambda = \frac{-b + \sqrt{b^2 - (4ac)}}{2a} \]
\[ x_{\text{new}} = \lambda x \]
\[ y_{\text{new}} = \lambda y \]
\[ z_{\text{new}} = (z - z_v) \lambda + z_v \]
\[ \Delta x = x_{\text{new}} - x \]
\[ \Delta y = y_{\text{new}} - y \]
\[ \Delta z = z_{\text{new}} - z \]
\[ \text{write}(\ast, \ast) \ x, y, z, x_{\text{new}}, y_{\text{new}}, z_{\text{new}} \]

if ( ifunc .eq. 4 ) then
  \[ \text{funcbc} = \Delta x \]
endif
if ( ifunc .eq. 5 ) then
  \[ \text{funcbc} = \Delta y \]
endif
if ( ifunc .eq. 6 ) then
  \[ \text{funcbc} = \Delta z \]
endif
end

c mean commentary.
The program that defines the surfaces the equations are imposed on is called defmesh.prb and is given below.

* Define problem definition 3D *

PROBLEM
TYPES
ELGRP1 = (TYPE = 250)
EBSBOUNCOND:
  degfd1=surfaces(s1)
  degfd1=surfaces(s7)
  degfd1=surfaces(s12)
  degfd1=surfaces(s18)
  degfd1=surfaces(s13)
  degfd1=surfaces(s19)
  degfd1=surfaces(s24)
  degfd1=surfaces(s30)
  degfd1=surfaces(s36)
  degfd1=surfaces(s37)
  degfd2=surfaces(s1)
  degfd2=surfaces(s7)
  degfd2=surfaces(s12)
  degfd2=surfaces(s18)
  degfd2=surfaces(s13)
  degfd2=surfaces(s19)
  degfd2=surfaces(s24)
  degfd2=surfaces(s30)
  degfd2=surfaces(s36)
  degfd2=surfaces(s37)
  degfd3=surfaces(s1)
  degfd3=surfaces(s7)
  degfd3=surfaces(s12)
  degfd3=surfaces(s18)
  degfd3=surfaces(s13)
  degfd3=surfaces(s19)
  degfd3=surfaces(s24)
  degfd3=surfaces(s30)
  degfd3=surfaces(s36)
  degfd3=surfaces(s37)
  degfd3=surfaces(s4)
  degfd3=surfaces(s10)
  degfd3=surfaces(s15)
  degfd3=surfaces(s21)
  degfd3=surfaces(s26)
  degfd3=surfaces(s32)
END

MATRX
METHOD = 1
ESSENTIAL BOUNDARY CONDITIONS

func=1,deqfd1,surfaces(s1)
func=1,deqfd1,surfaces(s7)
func=1,deqfd1,surfaces(s12)
func=1,deqfd1,surfaces(s18)
func=1,deqfd1,surfaces(s36)
func=4,deqfd1,surfaces(s13)
func=4,deqfd1,surfaces(s19)
func=4,deqfd1,surfaces(s24)
func=4,deqfd1,surfaces(s30)
func=4,deqfd1,surfaces(s37)

func=2,deqfd2,surfaces(s1)
func=2,deqfd2,surfaces(s7)
func=2,deqfd2,surfaces(s12)
func=2,deqfd2,surfaces(s18)
func=2,deqfd2,surfaces(s36)
func=5,deqfd2,surfaces(s13)
func=5,deqfd2,surfaces(s19)
func=5,deqfd2,surfaces(s24)
func=5,deqfd2,surfaces(s30)
func=5,deqfd2,surfaces(s37)

func=3,deqfd3,surfaces(s1)
func=3,deqfd3,surfaces(s7)
func=3,deqfd3,surfaces(s12)
func=3,deqfd3,surfaces(s18)
func=3,deqfd3,surfaces(s36)
func=6,deqfd3,surfaces(s13)
func=6,deqfd3,surfaces(s19)
func=6,deqfd3,surfaces(s24)
func=6,deqfd3,surfaces(s30)
func=6,deqfd3,surfaces(s37)

COEFFICIENTS

elgrp1 (nparm = 45) # element 250, LINEAR ELASTIC.
* coefficients nog veranderen
icoef2 = 0 # type of stress-strain relation: 3D stress
icoef4 = 0
coeff6 = 1d7 # E
coeff7 = 1d-1 # nu

END
*LINEAR_EQUATIONS, sequence_number = 1
* equation 1
* fill_coefficients 1
*END

END_OF_SEPRAN_INPUT