Al-Khwarizmi Engineering Journal

Al-Khwarizmi Engineering Journal, Vol. 6, No. 4, PP 37 - 51 (2010)

Finite Element Based Solution of Laplace's Equation Applied to Electrical Activity of the Human Body

Zainab T. Baqer

Department of Electrical Engineering/ College of Engineering/ University of Baghdad

Email: zainab_alissa@yahoo.com

(Received 8 November 2009; Accepted 4 November 2010)

Abstract

Computer models are used in the study of electrocardiography to provide insight into physiological phenomena that are difficult to measure in the lab or in a clinical environment.

The electrocardiogram is an important tool for the clinician in that it changes characteristically in a number of pathological conditions. Many illnesses can be detected by this measurement. By simulating the electrical activity of the heart one obtains a quantitative relationship between the electrocardiogram and different anomalies.

Because of the inhomogeneous fibrous structure of the heart and the irregular geometries of the body, finite element method is used for studying the electrical properties of the heart.

This work describes the implementation of the Conjugate Gradient iterative method for the solution of large linear equation systems resulting from the finite element method. A diagonal Jacobi preconditioner is used in order to accelerate the convergence. Gaussian elimination is also implemented and compared with the Precondition Conjugate Gradient (PCG) method and with the iterative method. Different types of matrix storage schemes are implemented such as the Compressed Sparse Row (CSR) to achieve better performance. In order to demonstrate the validity of the finite element analysis, the technique is adopted to solve Laplace's equation that describes the electrical activity of the human body with Dirichlet and Neumann boundary conditions. An automatic mesh generator is built using C^{++} programming language. Initially a complete finite element program is built to solve Laplace's equation. The same accuracy is obtained using these methods. The results show that the CSR format reduces computation time compared to the order format. The PCG method is better for the solution of large linear system (sparse matrices) than the Gaussian Elimination and back substitution method, while Gaussian elimination is better than iterative method.

Keywords: *Finite element, ECG, PCG, volume conductor and GE.*

1. Introduction

Bioelectric field problems can be found in a wide variety of biomedical applications which range from single cells, to organs, up to models which incorporate partial to full humane structure. In this work a class of direct and inverse volume conductor problems which arise in electrocardiography is studied.

The solutions to these problems have applications to defibrillation studies, detection and location of arrhythmias, impedance imaging techniques, and localization and analysis of spontaneous brain activity (in case of electroencephalography) in epileptic patients. Furthermore, they can, in general, be used to estimate the electrical activity inside a volume conductor, either from potential measurements at an outer surface, or directly from the interior bioelectric sources [1].

2. The Conduction System of the Heart

The human heart basically consists of the left and right atria, and the left and right ventricles. Each of these parts includes myocardial tissue surrounding a cavity. The right atrium collects and stores deoxygenated blood from the body. The right atrium adjoins to the right ventricle through the tricuspid valve. The blood of the right ventricle is pumped through the pulmonary arteries passing the pulmonary valve and finally reaches the lungs where it is oxygenated. The left atrium collects and stores the oxygenated blood from the lungs. The border between the left atrium and the left ventricle is the mitral valve. The blood of the left ventricle is pumped through the aorta into the body passing the aortic valves. The left and right ventricles are separated by the inter-ventricular septum, the left and right atria by the interatrial septum**.**

Figure 1 illustrates the conduction system of the heart that controls these cardiac contractions. The figure shows:

a) the sino-atrial (S-A) node in which the normal rhythmic self excitatory impulse is generated,
b) the internal pathways that conduct the impulse from the S-A node to the atrio-ventricular (A-V) node,
c) the A-V node in which the impulse from the atria is delayed before passing into the ventricles,
d) the A-V bundle, which conducts the impulse from the atria into the ventricles, and
e) the left and right bundles of Purkinje fibers, which conduct the cardiac impulse to all parts of the ventricles.

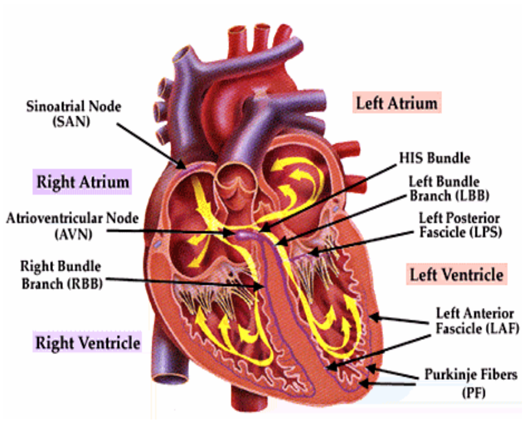


Fig.1. The Conduction System of the Heart [2].

The SA node creates an impulse of electrical excitation that spreads across the right and left atria. This impulse initiates the depolarization of the nerves and muscles of both atria, causing the atria to contract and pump blood into the ventricles. Repolarization of the atria follows. The impulse then passes into the atrioventricular (A-V) node, which initiates the depolarization of the right and left ventricles, causing them to contract and force blood into the pulmonary and general circulations. The ventricle nerves and muscles then repolarize and the sequence begins again.

Ion movement in heart muscle constitutes a current flow, which results in potential difference in the tissue outside the fibers and on the surface of the body .These potential differences could be measured by placing electrodes on the surface of the body (Fig.2) and then displaying the result as an ECG [3].

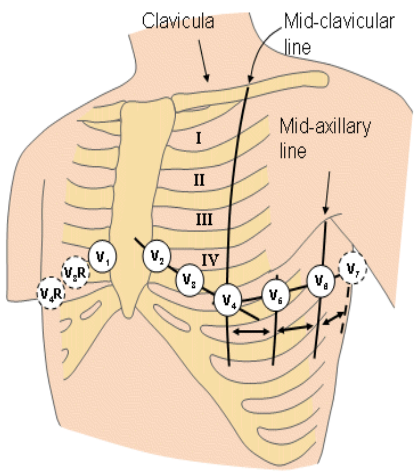


Fig.2. The Chest Leads [4].

3. Bioelectric Volume Conductor

A general volume conductor can be defined as a region of volume, Ω, which has conductivity, σ, and permittivity, ε, in which resides a source current , I_v, where the ($_v$) signifies per-unit volume. Solving a volume conductor problem means finding expressions for electrical field, *E*, the potential, Φ, everywhere within the volume, Ω , and/or on one of the bounded surfaces, Γ_i .

The more general formulation in terms of the primary current sources within the heart described by Poisson’s equation for electrical conduction [1]:

$$\nabla . \sigma \nabla \Phi = -I_v \quad \text{in} \quad \Omega \qquad \ldots(1)$$

One can define a surface bounding the region which includes the sources and recast the formulation in terms of information on that surface, yielding Laplac's equation because the distributions of voltages on the surface are solved instead of current sources within a volume [1]:

$$\nabla.\sigma\nabla\Phi = 0 \quad \text{in} \quad \Omega \qquad \ldots(2)$$

with Neumann boundary condition:

$$\Phi = \Phi_0 \quad \text{on} \quad \Sigma \subseteq \Gamma_T \qquad \ldots(3)$$

and Dirichlet boundary condition:

$$\sigma\nabla\Phi.n = 0 \ \text{on} \ \Gamma_T \qquad \ldots(4)$$

This particular formulation is known as the Cauchy problem for Laplace's equation.

If the field is determined from the source and conductor (heart), the problem is called a direct problem. If the source (potential of the heart) is determined from the known field and conductor (potential of the part of the surface of the body), the problem is called an inverse problem [1].

4. Main Program

The flow chart of the main program can be shown in figure 3. The following sections describe the details of each part.

4.1. Mesh Generation

For simplicity, the proposed model is defined on a simple two dimensional domain (fig.4) which shows a cross section of the heart.
To find the potential distribution for this model, the region is divided into a number of finite elements (meshed) as illustrated in **Fig.5**. A mesh could be created either inside the program or generated elsewhere using third party software. In this work the two methods are adopted.

1) Automatic Mesh Generation is built from scratch.
2) Automatic Mesh Generation using ANSYS package

4.2. Automatic Mesh Generation

This program performs a mesh generation of an arbitrary solution domain. A few points are given to determine the general configuration of the region. Then the program automatically generates triangular or quadrilateral elements. Triangular elements are chosen. The subroutine Input accepts the data which defines the solution region outline and the material zones.

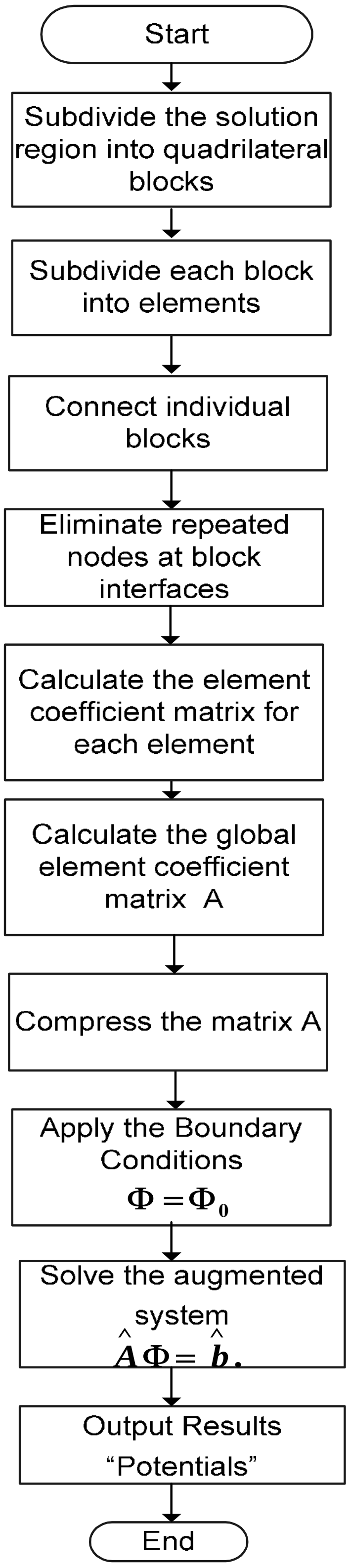

Start
Subdivide the solution region into quadrilateral blocks
Subdivide each block into elements
Connect individual blocks
Eliminate repeated nodes at block interfaces
Calculate the element coefficient matrix for each element
Calculate the global element coefficient matrix A
Compress the matrix A
Apply the Boundary Conditions $\Phi = \Phi_0$
Solve the augmented system $\hat{A}\Phi = \hat{b}$.
Output Results "Potentials"
End

Fig.3. The Flow Chart of the Main Program.

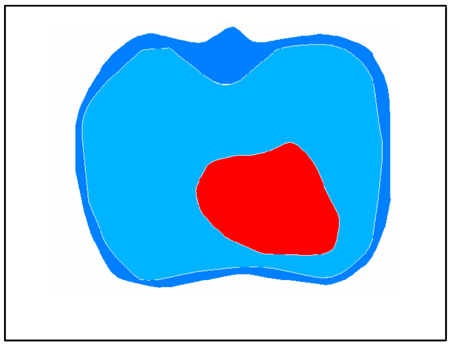


Fig.4. The Proposed Model (a Cross-Section of the Heart).

.

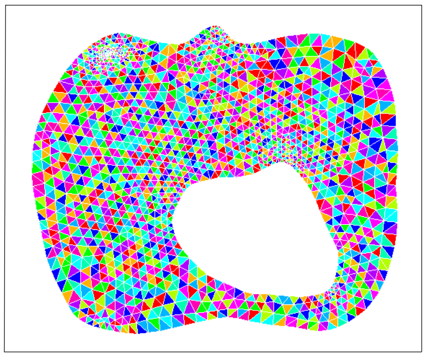


Fig.5. Mesh Configuration.

The basic steps involved in a mesh generation are as follows:

- Subdivide the solution region into few quadrilateral blocks,
- Separately subdivide each block into elements,
- Connect individual blocks

Each step is explained as follows:

(A) Definition of Blocks

The solution region is subdivided into quadrilateral blocks (Fig.6). Sub-domains with different parameters must be represented by separate blocks. As input data, block topologies and the coordinates at eight points describing each block is specified. Each block is represented by an eight-node quadratic isoperimetric element (Fig.7), i.e. for example:

Block number *one*, its eight nodes (1, 2, 3, 8, 7, 6, 5 and 4)
Block number two, its eight nodes (5, 6, 7, 16, 15, 17, 9 and 8)
Block number three, its eight nodes (9, 17, 15, 14, 13, 12, 11and 10)
. . . and so on.
With natural coordinate system (ζ,η), the *x* and *y* coordinates are represented as [6]:

$$x(\zeta,\eta)=\sum_{i=1}^{8}\alpha_i(\zeta,\eta)x_i \qquad \ldots(5)$$

$$y(\zeta,\eta)=\sum_{i=1}^{8}\alpha_i(\zeta,\eta)y_i \qquad \ldots(6)$$

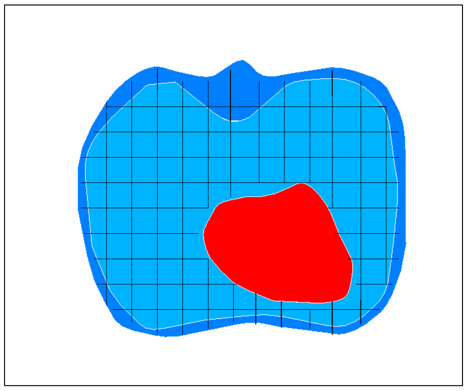


Fig.6. Subdivision of the Solution Region Into Quadrilateral Blocks.

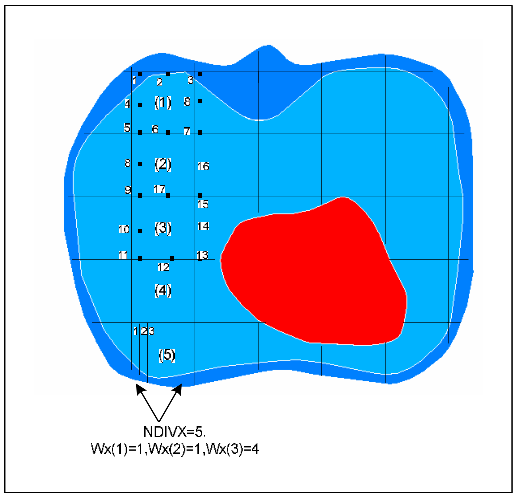


Fig.7. Eight-Node Quadratic Block.

where $\alpha_i(\zeta,\eta)$ is a shape function associated with node *i*, and (x_i,y_i) are the coordinates of node *i* defining the boundary of the quadrilateral block as shown in figure 8. The shape functions are expressed in terms of the quadrilateral or parabolic isoparametric element shown in figure 9. They are given by [6]:

for corner nodes,

$$\alpha_i = \frac{1}{4}(1+\zeta\zeta_i)(1+\eta\eta_i)(\zeta\zeta_i+\eta\eta_i-1) \qquad i=1,3,5,7$$

…(7)

for midside nodes

$$\alpha_i = \frac{1}{2}\zeta_i^2(1+\zeta\zeta_i)(1-\eta^2)+\frac{1}{2}\eta_i^2(2+\eta\eta_i)(1-\zeta^2)$$

$$i = 2,4,6,8$$

…(8)

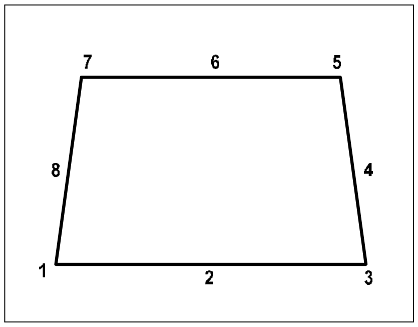


Fig.8. Typical Quadrilateral Block [6].

From the above equations any node can be created inside or on the boundary of each block using the shape function and the eight nodes defining the boundary of the quadrilateral block. The properties of the shape functions are [6]:

1. They satisfy the conditions

$$\sum_{i=1}^{n}\alpha_i(\zeta,\eta)=1 \qquad \text{…(9)}$$

$$\alpha_i(\zeta,\eta)=\begin{cases}1, & i=j\\ 0, & i\neq j\end{cases} \qquad \text{…(10)}$$

2. They became quadratic along element edges

$$(\zeta=\pm1, \eta=\pm1).$$

The subroutine INPUT(Algorithm 1) reads the number of points defining the mesh NPOIN, the number of blocks NBLOCK, the element type NTYPE (triangular or quadrilateral element), the number of coordinate dimension NDIME (2-D or 3-D), the nodes defining each block (as described previously), and the coordinates of each node in the mesh. Usually these data are read only once and stored in files. It is impracticable to read these data whenever the program is run.

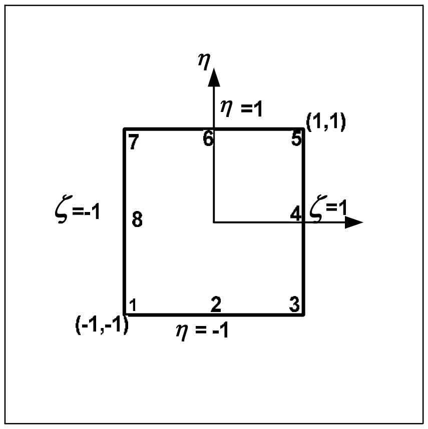


Fig.9. Eight-node Serendipity Element [6].

INPUT Algorithm

```
1.Read NP ,Number of coordinate points defining the solution region
2.Read NBLOCK, Number of blocks or zones
3.Read NTYPE, The type of elements into which the structure is to be subdivided
4.Read NDIM, The number of coordinate dimensions, NDIM=2
5.Do IELEM=1,. . ., NBLOCK
6.Do I=1,. . .,8
7.Read NL(IELEM,I),NL is the connectivity matrix
8.end Do (I)
9.end Do (IELEM)
10.Do J=1,. . ., NP
11.Do I=1,. . ., NDIM
12.Read COARD(J,I),coordinate of each node
13. end Do (I)
14. end Do (J)
15. Read NDIVX, NDIVY, the number of element subdivisions to be made in the $\zeta$ and $\eta$ directions, respectively for all the blocks
16.Read weighting factors $(W_\zeta)_i$ and $(W_\eta)_i$ for all the blocks
17.end
```

Algorithm 1 The INPUT Algorithm

The following is an algorithm that estimates the shape function (using equations 7 and 8).

Evaluation of the shape function

```
//*************************************
//SHAPEF(ζ,η)
//*************************************
1.SHAPEE[1]=0.25*(1-ζ)*(1-η)*(-ζ-η-1);
2.SHAPEE[2]=0.5*(1-ζ*ζ)*(1-η);
3.SHAPEE[3]=0.25*(1+ζ)*(1-η)*(ζ-η-1);
4.SHAPEE[4]=0.5*(1-η*η)*(1+ζ);
5.SHAPEE[5]=0.25*(1+ζ)*(1+η)*(ζ+η-1);
6.SHAPEE[6]=0.5*(1-ζ*ζ)*(1+η);
7.SHAPEE[7]=0.25*(1-ζ)*(1+η)*(-ζ+η-1);
8.SHAPEE[8]=0.5*(1-η*η)*(1-ζ);
9.End
```

Algorithm 2 Evaluation of the Shape Function.

(B) Subdivision of Each Block

For each block, NDIVX and NDIVY, the number of element subdivisions to be made in the ζ *and* η directions, respectively are specified.

Also the weighting factors $(W_\zeta)_i$ *and* $(W_\eta)_i$ are specified allowing for graded mesh within a block. As an example if block five is desired to be divided into three divisions in the x-axis direction with different weighting (fig 6), NDIVX have to be equal to three and the weighting factor as follows:

W(1)=1, W(2)=1, W(3)=4.

In specifying, NDIVX, NDIVY, $(W_\zeta)_i$ *and* $(W_\eta)_i$ care must be taken to ensure that the subdivision along block interfaces (for adjacent blocks) is compatible. ζ *and* η is initialized to -1 so that the natural coordinates are incremented according to [6]:

$$\zeta_i = \zeta_i + \frac{2(W_\zeta)_i}{W_\zeta^T . F} \qquad \ldots(11)$$

$$\eta_i = \eta_i + \frac{2(W_\eta)_i}{W_\eta^T . F} \qquad \ldots(12)$$

where

$$W_\zeta^T = \sum_{j=1}^{NDIVX} (W\zeta)_j \qquad \ldots(13)$$

$$W_\eta^T = \sum_{j=1}^{NDIVY} (W_\eta)_j \qquad \ldots(14)$$

$$F = \begin{cases} 1, & \text{for linear elements} \\ 2, & \text{for quadratic elements} \end{cases} \qquad \ldots(15)$$

Three types of elements are permitted:

a) Linear four-node quadrilateral elements,
b) Linear three-node triangular elements,
c) Quadratic eight-node isoparimetric elements.

(C) Connection of individual Blocks

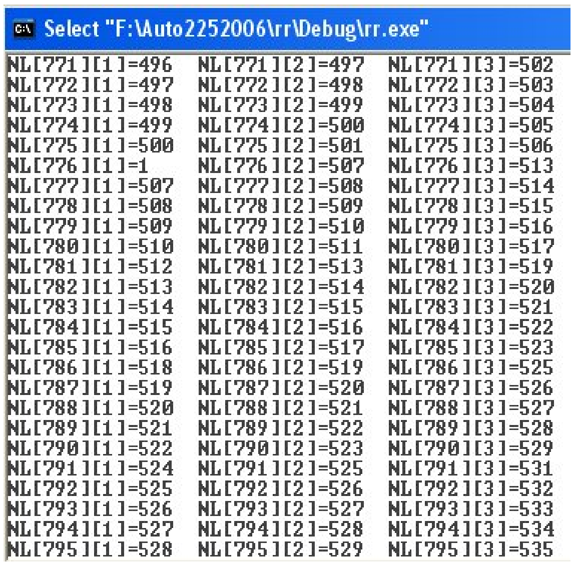
After subdividing each block and numbering its nodal points separately, it is necessary to connect the blocks and have each node numbered uniquely. This is accomplished by comparing the coordinates of all nodal points and assigning the same number to all nodes having identical coordinates. That is the coordinates of node 1 is compared with all other nodes, and then node 2 with other nodes, etc., until all repeated nodes are eliminated.

The basic blocks of the automatic mesh generator is illustrated in algorithm (3). The *OUTPUT* function prints out the coordinates of the nodes and the element topologies (Figs.10, 11).

Automatic Mesh Generator

```
1.INPUT
2.Generate
2.1 Subdivide the blocks into quadrilateral
        element
2.2 Eliminate the repeated nodes at block
        interfaces
3. Triangle //divides each four-node
   quadrilateral element into two triangular
   elements. The subdivision is done across
   the shorter diagonal
4. OUTPUT// provide the coordinates of the
     nodes, element topologies and material
     property of the generated mesh.
5.end
```

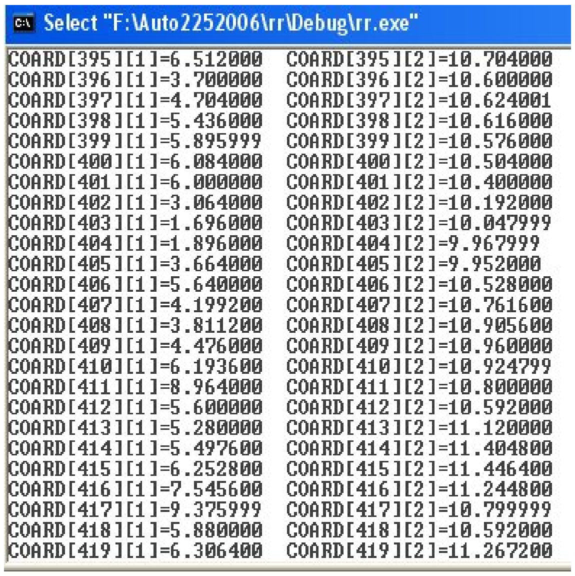
Algorithm 3 Main Blocks of the Automatic Mesh Generator.



Select "F:\Auto2252006\rr\Debug\rr.exe"

```
NL[771][1]=496   NL[771][2]=497   NL[771][3]=502
NL[772][1]=497   NL[772][2]=498   NL[772][3]=503
NL[773][1]=498   NL[773][2]=499   NL[773][3]=504
NL[774][1]=499   NL[774][2]=500   NL[774][3]=505
NL[775][1]=500   NL[775][2]=501   NL[775][3]=506
NL[776][1]=1     NL[776][2]=507   NL[776][3]=513
NL[777][1]=507   NL[777][2]=508   NL[777][3]=514
NL[778][1]=508   NL[778][2]=509   NL[778][3]=515
NL[779][1]=509   NL[779][2]=510   NL[779][3]=516
NL[780][1]=510   NL[780][2]=511   NL[780][3]=517
NL[781][1]=512   NL[781][2]=513   NL[781][3]=519
NL[782][1]=513   NL[782][2]=514   NL[782][3]=520
NL[783][1]=514   NL[783][2]=515   NL[783][3]=521
NL[784][1]=515   NL[784][2]=516   NL[784][3]=522
NL[785][1]=516   NL[785][2]=517   NL[785][3]=523
NL[786][1]=518   NL[786][2]=519   NL[786][3]=525
NL[787][1]=519   NL[787][2]=520   NL[787][3]=526
NL[788][1]=520   NL[788][2]=521   NL[788][3]=527
NL[789][1]=521   NL[789][2]=522   NL[789][3]=528
NL[790][1]=522   NL[790][2]=523   NL[790][3]=529
NL[791][1]=524   NL[791][2]=525   NL[791][3]=531
NL[792][1]=525   NL[792][2]=526   NL[792][3]=532
NL[793][1]=526   NL[793][2]=527   NL[793][3]=533
NL[794][1]=527   NL[794][2]=528   NL[794][3]=534
NL[795][1]=528   NL[795][2]=529   NL[795][3]=535
```

Fig.10. Element Topologies.



Select "F:\Auto2252006\rr\Debug\rr.exe"

```
COARD[395][1]=6.512000  COARD[395][2]=10.704000
COARD[396][1]=3.700000  COARD[396][2]=10.600000
COARD[397][1]=4.704000  COARD[397][2]=10.624001
COARD[398][1]=5.436000  COARD[398][2]=10.616000
COARD[399][1]=5.895999  COARD[399][2]=10.576000
COARD[400][1]=6.084000  COARD[400][2]=10.504000
COARD[401][1]=6.000000  COARD[401][2]=10.400000
COARD[402][1]=3.064000  COARD[402][2]=10.192000
COARD[403][1]=1.696000  COARD[403][2]=10.047999
COARD[404][1]=1.896000  COARD[404][2]=9.967999
COARD[405][1]=3.664000  COARD[405][2]=9.952000
COARD[406][1]=5.640000  COARD[406][2]=10.528000
COARD[407][1]=4.199200  COARD[407][2]=10.761600
COARD[408][1]=3.811200  COARD[408][2]=10.905600
COARD[409][1]=4.476000  COARD[409][2]=10.960000
COARD[410][1]=6.193600  COARD[410][2]=10.924799
COARD[411][1]=8.964000  COARD[411][2]=10.800000
COARD[412][1]=5.600000  COARD[412][2]=10.592000
COARD[413][1]=5.280000  COARD[413][2]=11.120000
COARD[414][1]=5.497600  COARD[414][2]=11.404800
COARD[415][1]=6.252800  COARD[415][2]=11.446400
COARD[416][1]=7.545600  COARD[416][2]=11.244800
COARD[417][1]=9.375999  COARD[417][2]=10.799999
COARD[418][1]=5.880000  COARD[418][2]=10.592000
COARD[419][1]=6.306400  COARD[419][2]=11.267200
```

Fig.11. Coordinates of the Nodes.

4.3. ANSYS Mesh Generator

The ANSYS package can be used for mesh generation (fig 5). The 6-node triangular element type is selected for the proposed model. The coordinates of each node and the connectivity matrix (the matrix that describe each element and its *6*-nodes) could be taken. A procedure is implemented to compress the *6*-nodes to *3*-node triangular element to fit the model's requirement (algorithm 4). This procedure also creates a new connectivity matrix (now each element is described by only 3-node).

Compression Algorithm

```
//****************************************
//Renumber the 6-node triangular elements to 3-
    node elements
//****************************************
1.Construct a Matrix,"flag_node" to contain the new
    numbering of the nodes
2. N=0,the new number
3.Do IELEM =1,...,NELEM;loop for all elements
4.Do INODE =1,2,3
5.Check if flag_node not utilized do
6.Store the node number in flag_node;
7.Store the new number in the connectivity matrix
8.Save the Coordinates of the node
9.Increase the number
10.end of for INODE
11.end of for IELEM
12.end
```

Algorithm 4 Compression Algorithm.

4.4. Calculation of the Element Coefficient Matrix

The element coefficient matrix is constructed for each element using geometry information, element connectivity and nodal coordinates. The element a_{ij}^{e} of the coefficient matrix may be regarded as the coupling between nodes *i* and *j* for element *e*, xl_i , yl_i is the coordinate of node *i* its value can be calculated as shown in algorithm 5:

Calculate the Element Coefficient Matrix Algorithm

```
1.Read from files the local coordinates for the
three nodes of the element e (XL, YL)
2.P1[1]=YL[2]-YL[3],Construct P and Q vector
3.P1[2]=YL[3]-YL[1]
4.P1[3]=YL[1]-YL[2]
5.Q[1]=XL[3]-XL[2]
6.Q[2]=XL[1]-XL[3]
7.Q[3]=XL[2]-XL[1]
//Compute the area of the element.
8.AREA=0.5*ABS(P1[2]*Q[3]-Q[2]*P1[3]);
9. $a_{ij}^{e} = \frac{P_i P_j + Q_i Q_j}{4 * Area}$    [6]
10.end
```

Algorithm 5 Element Coefficient Matrix.

Methods of calculating the coefficient matrix can be subdivided into two broad categories [7]: ASSEMBLY and Element by Element.

1. Assembly Method

In assembly methods, sparse global element matrices are built using the element coefficient matrices (algorithm 6). Element a_{ij} is the coupling between node *i* and *j*. It can be obtained using the fact that the potential distribution must be continuous across inter-element boundaries. The contribution to the *i*, *j* position in $[a]$ comes from all elements containing nodes *i* and *j*. For example if elements 1 and 2 have node 1 in common then: $a_{11} = a_{11}^1 + a_{11}^2$
For another example if node *4* belongs to element *1*, *2* and *3*; and it represent the second node for element *1* and third node for element *2* and element *3* hence,

$$a_{44} = a_{22}^1 + a_{33}^2 + a_{33}^3 \quad \text{and so on.}$$

```
Calculate the Global Element Coefficient Matrix Algorithm

1. Do I=1,..,NELEM;loop all elements
2.Read from files the local coordinates(XL, YL) for the three nodes of the element I
3.P[1]=YL[2]-YL[3],Construct P and Q vector
4.P[2]=YL[3]-YL[1]
5.P[3]=YL[1]-YL[2]
6.Q[1]=XL[3]-XL[2]
7.Q[2]=XL[1]-XL[3]
8.Q[3]=XL[2]-XL[1]
//Compute the area of the element.
9.AREA=0.5*ABS(P1[2]*Q[3]-Q[2]*P1[3]);
10. $a_{ij}^e = \frac{P_i P_j + Q_i Q_j}{4 * Area}$
11.Save $a^e{}_{ij}$ in a proper place in a global matrix according to the node number
12.end
```

Algorithm 6 Global Element Coefficient Matrix Algorithm.

2. Element by Element Method

In element by element methods [7], a global system matrix is never created (algorithm 7). Explicit or iterative solvers are used to solve the equations. There are many different iterative solvers and the choice of which one to use depends on the type of problem being solved. Element by element methods lead naturally to a parallel solution strategy that may be applied to all the general problem types.

Whether to use assembly or to use element by element methods depends on performance. One should not be surprised if the preferred method changes from machine to machine, from problem to problem or from time to time as hardware characteristics change. Although some researchers disagree about which method is faster. Two methods are adopted in this work (assembly and element by element method).

```
Element-by-Element Method

1.Do I=1,..,NELEMI; loop all elements
2.P1[1]=YL[2]-YL[3]
3.P1[2]=YL[3]-YL[1]
4.P1[3]=YL[1]-YL[2]
5.Q[1]=XL[3]-XL[2]
6.Q[2]=XL[1]-XL[3]
7.Q[3]=XL[2]-XL[1]
8.AREA=0.5*ABS(P1[2]*Q[3]-Q[2]*P1[3])
/Determine coefficient matrix for Element I
9.Do i=1..3
10.Do j=1..3
11.a[i][j]=(P1[i]*P1[j]+Q[i]*Q[j])/(4*AREA)
12.Find row and column number (IR,IC) of each
      node
13.If(a[i][j]!=0)Do
14.Store this value in proper place in a
      comp-matrix and store its row number in
      row matrix
15.end if
16.end for j
17.end for i
18.end for I
19.end
```

Algorithm 7 Element-by-Element Method.

4.5. Matrix Storage Scheme

Any finite element problem for which the element ‘stiffness’ matrices fit in cache can be computed in a fast and efficient manner. As soon as the cache memory size is exceeded, performance will quickly degrade and be dominated by the speed of the slower main memory. So, what is the largest problem size or the maximum number of finite elements *Nelsc* that can theoretically reside in cache memory? For an approximate calculation, only three quantities are required. These are the size of the cache *Cs* , the number of floating point values required for each finite element (*FPel*) and the number of bytes required to represent the floating point precision (*Bfpp*). These are simply related in the expression:

$$Nels = \frac{C_s}{FP_{el} * B_{fpp}} \qquad \ldots(16)$$

To reduce storage requirements, various strategies have been developed. Typically, two dimensional problems can be fairly large before memory size becomes an issue. However with three dimensional problems, memory quickly becomes a problem "to perform analyses with thousands, if not millions of elements" and in such cases, it may be preferable to use element by element methods]. In this work the model is two-dimension but a general program is implemented.

4.6. Matrix Compression Algorithm

For this purpose "reduce storage" and since element coefficient matrix A[N][N] is a symmetric positive definite sparse matrix Let K_i the number of non zero elements in the i-th column and let K_{max} maximum(K_i) i=0,…N-1 .

In our implementation matrix A is stored using two K_{max}* N matrices AC and ROWS where AC is a compressed version of the matrix A and ROWS contains the indices of the elements of A stored in AC (algorithm 8). In figure (12) an example demonstrates how this algorithm acts.

As an example if the dimension of the element coefficient matrix is 10000*10000 and the maximum number of none zero elements in column is equal to 50 so instead of 10000*10000=100,000,000 storage element required 1000*50*2=100,000 storage elements are utilized.

```
Matrix Compression Algorithm

//Compress A[N][N] to  AC[Kmax][N]
1.Do j=1,..,N
2. Do i=1,..,N
3.If(A[i][j]!=0)Do
4.Store row number, i in proper place in matrix ROW
5.Save A[i][j] in proper place in AC matrix
6.end of if statement in step 3
7.end for i loop
8.end for j loop
9.end of the algorithm
```

Algorithm 8 Matrix Compression Algorithm.

$$A[5][5] = \begin{bmatrix} 3 & 0 & 1 & 0 & 0 \\ 0 & 4 & 2 & 0 & 1 \\ 2 & 0 & 0 & 1 & 0 \\ 7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \end{bmatrix} , \quad AC[3][5] = \begin{bmatrix} 3 & 4 & 1 & 0 & 1 \\ 2 & 0 & 2 & 1 & 0 \\ 7 & 0 & 3 & 0 & 0 \end{bmatrix},$$

$$ROWS[3][5] = \begin{bmatrix} 1 & 2 & 1 & 0 & 2 \\ 3 & 0 & 2 & 3 & 0 \\ 4 & 0 & 5 & 0 & 0 \end{bmatrix}$$

Fig.12. The Compressed Version of Matrix *A*.

4.7. Equation Solution

The finite element approximation of equation (2) can equivalently be expressed as a system of *N* equations with *N* unknowns $\Phi_1, \ldots\ldots\ldots\ldots, \Phi_N$ (the electrostatic potentials, for example). In matrix form, the above system can be written as: AΦ=b where $A = (a_{ij})$ is the global coefficient (stiffness) matrix

For volume conductor problems, *A* contains all of the geometry and conductivity information of the model [1]. Researchers have different views for the conductivity. Some of them consider the conductivity varying with the dimension (x and y axis in the two dimension model) and with the type of the tissue [8]. As an example the conductivity of the bone differs from the conductivity of the fat and so on. Others for simplicity consider unit conductivity [9] and this conductivity is considered in this work.

4.8. Boundary Conditions

For the finite element method, it turns out that the Neumann condition is very easy to apply; while the Dirichlet boundary condition ($\Phi = \Phi_0$ on $\Sigma \subseteq \Gamma_T$, equation 3) takes a little extra effort. The Nemann boundary condition ($\sigma \nabla \Phi . n = 0$ on Γ_T, equation 4) is satisfied automatically within the Galerkin and variational formulations.

To apply the Dirichlet boundary condition directly, the (a_{ij}) matrix is modified and the b_i vector by implementing the following steps such that the ith value of Φ_i is known (algorithm 9).

```
Applying-Dirichlet-Boundary Condition
1. Subtract from the $ith$ member of the r.h.s. the product of $a_{ij}$ and the known value of $\Phi_i$ (call it $\bar{\Phi}_i$; this yields the new right hand side, $\hat{b}_i = b_i - a_{ij}\bar{\Phi}$.
2. Zero the $ith$ row and column of A: $\hat{a}_{ij} = \hat{a}_{ji} = 0$.
3. Assign $\hat{a}_{ii} = 1$.
4. Set the $jth$ member of the r.h.s. equal to $\bar{\Phi}_i$
5. Continue for each Dirichlet condition.
6. Solve the augmented system, $\hat{A}\Phi = \hat{b}$.
7.end
```

Algorithm 9 Applying - Dirichlet - Boundary Condition.

For example, the following *3*3* system,

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

where Φ_3 is known, $\overline{\Phi}_3$. Following the algorithm described above, the following augmented system is obtained,

$$\begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \end{bmatrix} = \begin{bmatrix} b_1 - a_{13}\overline{\Phi}_3 \\ b_2 - a_{23}\overline{\Phi}_3 \\ \overline{\Phi}_3 \end{bmatrix}$$

Solving this system yields the solution set, (Φ_1, Φ_2, Φ_3). In the following sections different methods are implemented to solve this system.

In the direct problem equation (2) is solved using the measured voltages on the surface of the heart to calculate the voltages at the surface of the torso. The inverse problems are formulated as using measurements on the surface of the torso and calculating the voltages on the surface of the heart.

Three methods are used to solve these equations: Precondition Conjugate Gradient method (PCG), Gaussian Elimination (GE) and Back Substitution (BS) method and iteration method. In the following sections a brief description, implementation and results of each method would be shown.

4.9. Precondition Conjugate Gradient Method

The PCG algorithm is used for solving systems of linear equations in the form $Ax=b$ where A is an N*N (symmetric positive definite sparse matrix) [10]. In this work a diagonal PCG algorithm is considered. The preconditioner M is a vector simply obtained in the following way: M[i]=A[i][i]. The implementation of precondition conjugate gradient method is shown in algorithm 10.

In the description of the algorithm, k defines the iteration count, $\underline{r}$ specifies the step length, and *beta* denotes the correction factor. In the implementation the norm of the residual, d is used for the convergence check.

```
Precondition Conjugate Gradient Algorithm
// Solve Ax=b
1.Choose $\underline{x}_0$,intial guess
2.$\underline{r}_0 = \underline{b} - A\underline{x}_0$
3. Solve $M\underline{z}_0 = \underline{r}_0$
4.$\underline{P}_0 = \underline{z}_0$
5.   $d_0$ = vv_product($\underline{r}_0$,$\underline{r}_0$)
  Do k=0,1,. . .Kmax
6.$\underline{q}_k$ = mv_product(A,$\underline{P}_k$)
7. alpha = vv_product($\underline{P}_k$,$\underline{q}_k$)
8. alpha = $\underline{d}_k$ / alpha
9. $\underline{x}_{k+1} = \underline{x}_k$ + alpha $\underline{P}_k$
10. $\underline{r}_{k+1} = \underline{r}_k$ – alpha $\underline{q}_k$
11. Solve $M\underline{z}_{k+1} = \underline{r}_{k+1}$
11. $\underline{d}_{k+1}$ = vv_product($\underline{z}_{k+1}$,$\underline{r}_{k+1}$)
12. If (SQRT($\underline{d}_{k+1}$) < Tolerance)Exist
13. beta = $\underline{d}_{k+1}$ / $\underline{d}_k$
14. $\underline{P}_{k+1} = \underline{z}_{k+1}$ + beta $\underline{P}_k$
15 end Do
16 end the algorithm
```

Algorithm 10 Precondition Conjugate Gradient

The elapsed time for this algorithm is measured as shown in Fig. 13. Increasing the size of the matrix increases the consumed time. The number of iterations required to reach convergence is also increased with increasing the size of the matrix. It can be shown that the number of iterations to reach convergence is less than the size of the matrix.

4.10. Gaussian Elimination and Back Substitution Method

The Gaussian elimination algorithm has three nested loops. Several versions of the algorithm exist, depending on the order in which the loops are arranged. Algorithm 11 shows one version of Gaussian elimination, which is implemented. It converts a system of linear equations Ax=b to a unit upper-triangular system Ux=y. The matrix U shares storage with A and overwrites the upper-triangular portion of A. The element A[k,j] computed on line 6 is actually U[k,j]. Similarly the element A[k,k] equal to 1 on line 8 is U[k,k]. This algorithm leads to the LU factorization of A as a product L*U. After the termination of the procedure, L is stored in the lower-triangular part of A, and U occupies the locations above the principle diagonal.

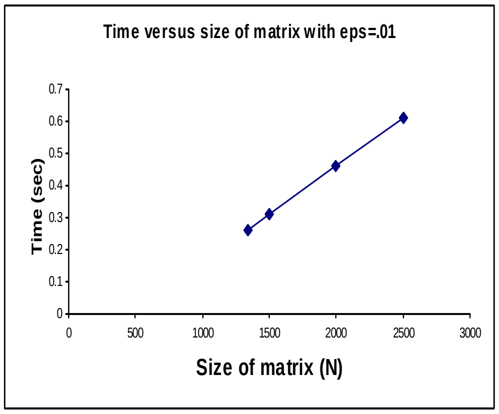


Fig.13. Time Consumed by PCG Method With Different Size of a Matrix.

For k varying from 0 to n-1, the Gaussian elimination procedure systematically eliminates variable x[k] from equation k+1 to n-1 so that the matrix of coefficients becomes upper-triangular. In the kth iteration of the outer loop, an appropriate multiple of the kth equation is subtracted from each of the equations k+1 to n-1. The multiples of the kth equation are selected such that the kth coefficient becomes zero in equations k+1 to n-1 eliminating x[k] from these equations. A typical computation of the Gaussian elimination procedure in the kth iteration of the outer loop is shown in Fig.14. The kth iteration does not involve any computation on rows 1 to k-1 or columns 1 to k-1. Thus at this stage, only the lower-right (n-k)*(n-k) sub-matrix of A (the shaded portion in Fig.14) is computationally active. The sequential run time of the procedure for large n is approximately $2n^3/3$ [11].

```
Gaussian Elimination Algorithm

//convert a matrix A to an upper triangular
matrix
***********************************
1. begin
2. Do k=0,..,n-1
3. Do j = k+1,.., n-1
4. A[k,j] = A[k,j] / A[k,k]
5. y[k] = b[k] / A[k][k]
6.  A[k,k]=1
7. Do i = k + 1,..,n-1
8. Do j = k + 1,.., n-1
9. A[i,j] = A[i,j] – A[i,k] *A[k,j];
10. b[i] = b[i] –A[i,k] * y[k]
11. A[i,k]=0
12. end j loop
13. end i loop
14. end k loop
15.end Gaussian_Elimination
```

Algorithm 11 Gaussian Elimination.

After the full matrix A has been reduced to an upper-triangular matrix U with ones along the principal diagonal, a back-substitution is performed to determine the vector x. Algorithm 12 shows a sequential back-substitution for solving an upper-triangular system of equations U x=y .

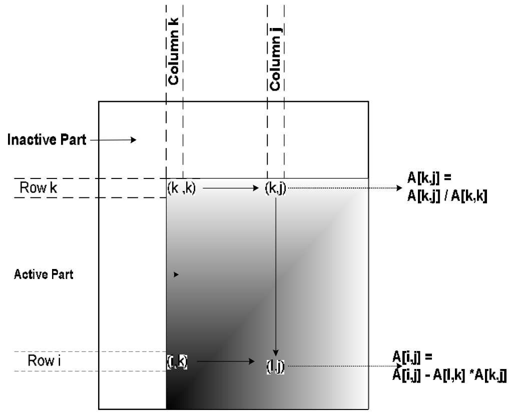


Fig.14. A Typical Computation in Gaussian Elimination.

```
Back Substitution Algorithm
*********************************
1.Do k = n-1,..,0
2.x[k] = y[k]
3.Do i = k-1,..,0
4.y[i] = y[i] – x[k]* U[i,k]
5.end i loop
6.end k loop
7.end Back_Sustitution
```

Algorithm 12 Back Substitution.

The consuming time by a Gaussian Elimination method to solve the system $AX = B$ is measured with different sizes of the matrix A as shown in Fig.15. It is obvious that the elapsed time by this algorithm is very large compared to the PCG's time (Fig.13); so for the suggested model PCG method is more efficient than the GE method. Using GE or PCG may not affect in a number of problems. The constructed matrix in this work is positive definite symmetric sparse matrix makes the PCG method more suitable.

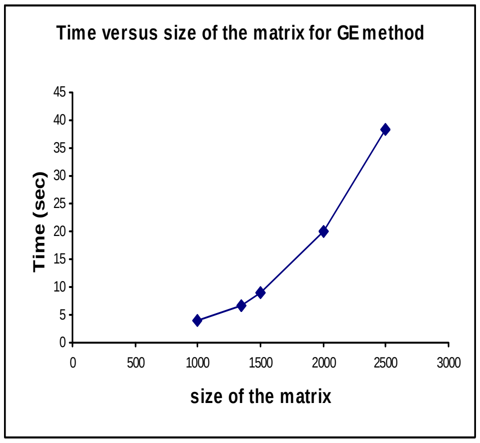


Fig.15. Time elapsed by the GE and Back Substitution Method.

4.11. Iteration Method

This is the third method used in this work for the solution, at a node k in a mesh with n nodes [6]

$$V_k = -\frac{1}{A_{kk}} \sum_{i=1, i \neq k}^{n} V_i A_{ki} \qquad \text{…(17)}$$

where node k is a free node, A is the element coefficient matrix and V_k is the potential at node k. This equation can be applied iteratively to all the free nodes (where the potentials are unknown). The iteration process begins by setting the potentials of fixed nodes (where the potentials are prescribed or known) to their prescribed values and the potentials at the free nodes are equated to zero or to the potential average. The accuracy in this method is changed with the number of iterations and there is a noticeable difference between the results when the number of iterations is changed. The time consuming is large (Fig.16) compared to the previous methods (Fig.17) so this method is not practical for such problems.

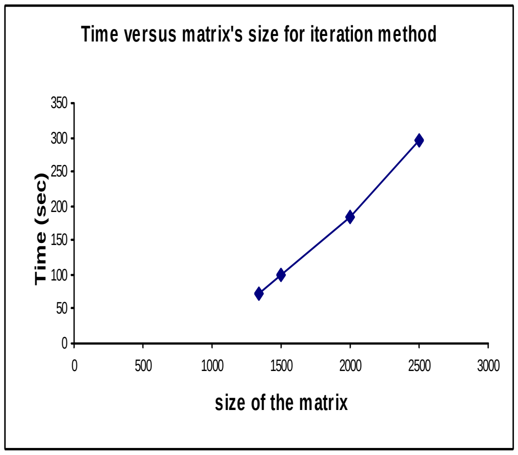


Fig.16. Elapsed Time by the Iteration Method with Different Sizes for the Matrix.

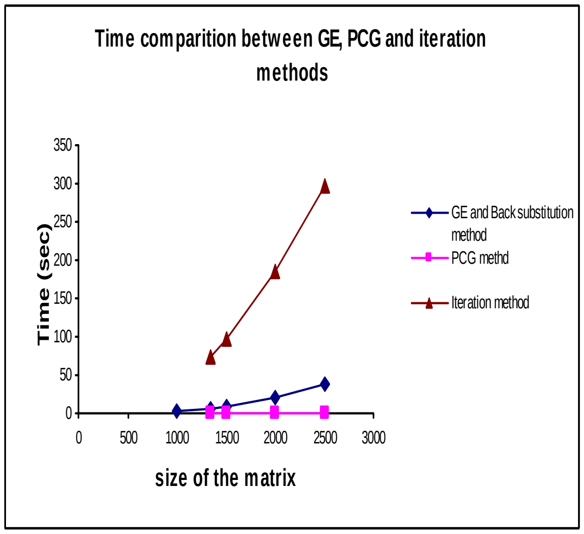


Fig.17. A comparison between the Elapsed Time of PCG, GE and Iteration Methods.

4.12. Direct and Reverse Solution

In a direct problem voltages are measured on the surface of the heart and used to calculate the voltages at the surface of the torso, as well as within the volume conductor of the thorax. This is implemented in this work by dividing the ECG cycle into steps for every *20 msec.* In the direct solution a measurement is obtained for the entire surface (fig.18).

There are many issues that affect these results. In the implementation a homogenous tissue is considered, a cross-section (2-D) of the heart is proposed. A simple model is suggested to approximate the electrical activity of the heart.
In the future the model will be improved (homogenous and 3D) so as to get more accurate results.

The inverse problems are formulated as using measurements on the surface of the torso and calculating the voltages on the surface of the heart. Better results are obtained by increasing the number of measurements on the surface of the torso.

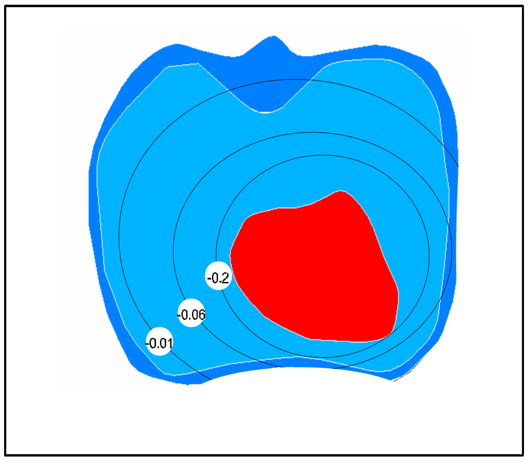


Fig.18. Potential Distribution in the Human Heart During the Direct Phase (Numbers Represent the Potential in mV).

5. Conclusions

In studying the electrical activity of the heart, a volume conductor approach is used to approximate the voltages on the surface of the heart using the voltages at the part of the surface of the torso. The results show that increasing the size of the surface (known values) will increase the accuracy of the results. The difference between the normal results and the measured results is due to the assumption of homogenous tissue and the simple 2-D model suggested approximating the thorax. There are hopes the studies in this field will continue to get more accurate results. The results will be improved if a 3D and homogenous model are suggested since this model is more close to the fact. An automatic mesh generator is built from scratch. By decreasing the size of the elements the result are more accurate. A comparison between different methods is performed. The results show that the PCG method is preferable for a fast system than the GE, BS and iteration methods, while the iteration method is the most time consuming method. In addition compression of sparse matrices increases the speed of the system.

Notations

AV	Atrio-Ventricular
CG	Conjugate Gradient
CSR	Compressed Sparse Row
COARD	Coordinate
E	Electrical Field
ECG	Electrocardiography
EEG	Electroencephalography
FEM	Finite Element Method
GE	Gaussian Elimination
ID	Identification
I_v	Current per unit volume
LU	Lower Upper
L	Lower
U	Upper
Matr	Matrix
mv_ prod	Matrix-Vector Product
N	Number of nodes
NDIVX	Number of Division in the- ζ Axis Direction
NDIVY	Number of Division in the η - Axis Direction
NBLOCK	Number of Blocks
NPP	Number of Prescribed (fixed) nodes
NELEM	Number of Elements
PCG	Precondition Conjugate Gradient
PVM	Parallel Virtual Machine
SA	Sino-Atrial
V	Potential
vv_prod	Vector-Vector Product
V_k	Potential at subspace *k*
Vect	Vector
Wx(i)	Weight of Division i in the ζ - Axis Direction

Wy(i)	Weight of Division i in the η - Axis Direction
1-D	One-Dimension
2-D	Two-Dimension

Greek letters

Ω	Volume
σ	Conductivity
ε	Permittivity
Φ	Potential
Φ_e	Potential at element ***e***
$\overline{\Phi}$	Test Function
Γ	Surface
Ψ_i	Basis function at node ***i***
α_i	Interpolation function
ζ, η	Intrinsic coordinates

6. References

[1] Christopher R. Johnson."Adaptive Finite Element and Local Regularization Methods for the Inverse ECG Problem". Biology Society 17th Annual International Conference, pages 209–210. IEEE Press. Center for Scientific Computing and Imaging, Department of Computer Science, University of Utah, Salt Lake City, Utah 84112 US, 1995.

[2] Julian Johnson, "Wireless Electrocardiogram"

[3] Wael Namat Al_Sahar, "The design and implementation of a PC based ECG recording and interpretation system", M.Sc Thesis, University of Technology, Electronic Engineering Dept, 1999.

[4] Pilkington TC, Plonsey R: "Engineering Contributions to Biophysical Electrocardiography", IEEE Press, John Wiley, New York, 1982.

[5] Tirupathi R. Chandrupatla and Ashok D. Belegundu "Introduction to Finite Elements in Engineering", copyright ©1991 by Prentice-Halls Inc.

[6] [Matthew N. O. Sadiku "Numerical Techniques in Electromagnetics"; 2nd edition, © 2001 by CRC Press LLC.

[7] C.O.Moretti, J.B. Cavalcante Neto, T.N. Bittencourt and L.F. Martha,"A Parallel Environment for Three-Dimensional Finite Element Method Analysis", 2000.

[8] O. Skipa, D. Farina, C. Kaltwasser, O.D¨ossel, W. R. Bauer "Fast Interaction ECG Simulation and Optimization-Based Reconstruction of Depolarization" Institute for Biomedical Technique, University of Karlsruhe (TH), Germany Institute for Biophysics, University at Wurzburg, Deutschland Oleg.Skipa@ibt.uni-karlsruhe.de, 2004.

[9] M.Okajima, K.Doniwa, T. Yamana, K. Ohta, N.Suzumura, A. Iwata and K. Kamiya. "Individualized Modeling of the Heart and Torso for Finite Element Method to be used in Forward Calculation of Body Surface Map", IEEE, University, Toyoake, Nagoya Area 470-11, Japan, 1989.

[10] Antonio d' Acierno, Giuseppe De Pietro, Antonio Giordano, "A Parallel Implementation of the Conjugate Gradient Method on the Meiko CS-2", IRSIP _ CNR, Napoli (Italy), 1996.

[11] Ananth Grama, Anshul Gupta, George Karypis and Vipin Kumar "Introduction to Parallel Computing" 2nd edition published 2003.

العنصر المحدد لحل معادلة لابلاس مطبقة على النشاط الكهربائي للجسم الأنساني

زينب توفيق باقر
قسم هندسة الكهرباء\ كلية الهندسة\ جامعة بغداد
البريد الألكتروني: zainab_alissa@yahoo.com

الخلاصة

نماذج الحاسبات تستخدم لدراسة التخطيط الكهربائي لعمل القلب للتَزويد برؤية إلى الظواهرِ الفسلجيةِ التي صعبة القيَاس في المختبرِ أو في بيئة سريرية. التخطيط الكهربائي لعمل القلب أداة مهمة للطبيب في انهاّ تَتغيّرُ على نحو مميز في عدد مِنْ الشروطِ الباثولوجيةِ . العديد من الأمراض ممك ن ان تكتش ف بهذا المقياس. بمحاكاة النشاطِ الكهربائي للقلبِ يعتمدُ على علاق ة كمّي ة ب ين التخط يط الكهرب ائي وأش ياء غي ر س وية مختلفة بسبب التركي ب الغي ر متج انس الليفي للقلب والجسم الهندسي الغير منتظم, طريقة العنصر المحدد استخدمت لدراسة الخواص الكهربائية للقلب.

يصف هذا العمل بناء طريقة التدرج التكرارية ل حـل مجموع ة المع ادلات الخطي ة الكبي رة الناتج ة م ن طريق ة العنص ر المح دد ان طريق ة الش رط الاولي لجاكوبي القطري (PCG) استخدمت لتسريع الاقتراب.طريقة الحذف الكاوسي ايض ا بني ت وقورن ت معه ا وم ع الطريق ة التكراري ة. ق د ت م اس تخدام انواع مختلفة من طرق خزن المصفوفات مثل طريقة الصف المتفرق المضغوط للحصول على افضل اداء. لغرض التحقق من دقة الحس ابات لتحلي ل العنص ر المحدود تقني تبني لحل معادل ة لاب لاس الت ي تص ف النش اط الكهرب ائي لقل ب الانسان ض من ش روط (العالم ان) د رش لت ونيوم ان ول د ش بكة ال ي ب ذ ي بأستخدام لغة برمجة C++ . في البداية برنامج العنصر المحدد الكامل بني لحل معادلة لابلاس. نفس الدقة حصل عليها من هذه الطرق. طريق ة الش رط الاول ي لجاكوبي القطري PCG هي الافضل في النظام الخطي الكبير من طريقة الحذف الكاوسي و طريقة الحذف الكاوسي افضل من الطريقة التكرارية.