Chapter 1

Electrostatics, finite element method

1.1 Outline of the method

The *finite element method* is a technique for solving partial differential equations that is widely used in engineering to perform stress-strain analysis of mechanical structures, to compute heat flow, magnetic fields etc. Here, we apply the technique to compute an approximation of the electrostatic potential and field in nearly arbitrary assemblies of conductors, dielectrics and near vacuum. The technique is applicable to both 2- and 3-dimensional problems.

Since finite element methods are widely used, programs that perform these calculations are marketed by numerous specialised companies. Garfield has interfaces which read the solutions computed by the Maxwell program (Ansoft), Tosca (Vector Fields), FEMlab, QuickField and Ansys. Other interfaces can be added, should the need arise.

The calculation of a potential using finite element methods proceeds broadly speaking as follows:

- The domain in which the field is to be computed, is broken up in a number of *elements*. The collection of elements filling the domain is called the *mesh*. In 2 dimensions, the elements would usually be triangles or quadrilaterals, while in 3 dimensions typical elements include the tetrahedron and hexahedron. Elements never cross material boundaries: elements do no penetrate from e.g. a gas into a conductor, hence the dielectric constant will never vary within an element.

- A potential value is computed for each of the *nodes* of each element, respecting the boundary conditions and trying to best satisfy the Poisson equation. As a rule, the vertices of an element are nodes of the element. But it is common that various other points, e.g. the half-way points between the vertices, are also nodes. The nodes of one element are often also nodes of neighbouring elements, the potential values in such nodes are the same whether the node is treated as belonging to one element or to its neighbour. Nodes located on the surface of a conductor that is kept at a fixed potential, will have the potential of the conductor - the potential of such conductors is treated as a boundary condition.

- The potential at any point inside an element is interpolated using the *shape functions*. These shape functions usually are low-order polynomials.
- The potential thus obtained will, as a rule, not satisfy the Poisson equation, if only because 2nd order polynomials are in general not solutions of the Poisson equation. One can improve the situation by sub-dividing those elements where the discrepancy is largest and then recomputing the potentials at the nodes. This process is repeated until the discrepancy drops below a threshold value.

In the following, we'll have a closer look into the various concepts that appeared in the outline above.

1.2 Choice of elements

1.2.1 Maxwell

Maxwell 2D users second order triangles and Maxwell 3D uses second order tetrahedra - no choice is offered to the user.

Second order triangles resp. tetrahedra, are elements which have the vertices and one point between each pair of vertices as nodes. Second order triangles have 6 nodes while second order tetrahedra have 10 nodes. In Maxwell, the intermediate points are precisely half-way between the vertices. The edges and faces are therefore straight, both for triangles and for tetrahedra.

1.2.2 Tosca

Tosca offers a wider choice of elements, but currently there is only an interface for first order hexahedra. These hexahedra have 8 nodes.

Tosca uses, if appropriate, degenerate hexahedra in which 2 pairs of 2 vertices are collapsed, thus forming a pentahedron.

1.2.3 Ansys

Ansys has a vast choice of elements. Currently, Garfield has interfaces for the 10-node tetrahedron with curved sides (SOLID123) and the 8-node "serendipity" quadrilateral (PLANE121). Other elements can be added if needed.

1.3 Coordinates and Shape functions

1.3.1 Natural coordinates

Although one can in principle use Cartesian coordinates throughout, it proves in practice easier to use natural coordinates inside the elements. Natural coordinates take on values like 0, 1 or -1 at the vertices. For triangles one uses 3 coordinates (ξ_1, ξ_2, ξ_3) which are such that $\xi_1 = 1$ in node 1 and $\xi_1 = 0$ in nodes 2 and 3, $\xi_2 = 1$ in node 2 and $\xi_2 = 0$ in nodes 1 and 3, $\xi_3 = 1$ in node 3 and $\xi_3 = 0$ in nodes 1 and 2. The coordinates are normalised by the condition that $\xi_1 + \xi_2 + \xi_3 = 1$ everywhere. The definition of the 4 natural coordinates $(\xi_1, \xi_2, \xi_3, \xi_4)$ for tetrahedra is similar: $\xi_1 = 1$ in node 1 and $\xi_1 = 0$ in nodes 2, 3 and 4, $\xi_2 = 1$ in node 2 and $\xi_2 = 0$ in nodes 1, 3 and 4, $\xi_3 = 1$ in node 3 and $\xi_3 = 0$ in nodes 1, 3 and 4, $\xi_4 = 1$ in node 4 and $\xi_4 = 0$ in nodes 1, 2 and 3 and $\xi_1 + \xi_2 + \xi_3 + \xi_4 = 1$ everywhere. In non-degenerate hexahedra, one uses only 3 coordinates (ξ_1, ξ_2, ξ_3) each of which equals -1 on one face and $+1$ on the opposite face. Degenerate hexahedra use 4 coordinates: 3 are triangular coordinates while the 4th has the hexahedral behaviour between the 2 non-touching faces.

These boundary conditions do not pinpoint the behaviour of the coordinates between the vertices. A simple solution for triangles and tetrahedra is to make the coordinates constant on lines resp. planes parallel with the base facing the vertex they are associated with, and to make them vary linearly between the vertex and the base. This approach is used for triangles and tetrahedra of first order. It also works for higher order elements, provided the edges are straight. For more complicated elements the behaviour is less trivial. In the case of hexahedra for instance, the coordinates vary smoothly between 2 non-parallel planes, something which can be achieved in a number of ways, none of which is a priori to be preferred.

To resolve this, one resorts to the use of isoparametric coordinates. These rely on the same shape functions which are used to interpolate the potential, and we'll therefore first introduce the shape functions. Interestingly, shape functions are themselves defined in terms of natural coordinates and one may therefore fear a cyclic argument.

1.3.2 Shape functions

Each node of an element has its own shape function. By definition, the shape function of a node equals 1 at the node and 0 at all other nodes. Between nodes, the shape function is in our applications a polynomial. The order of the polynomial depends on the structure of the element.

Triangle

In a first order triangle, a little-used element, the shape functions N_i are simply:

$$\begin{aligned} N_1 &= \xi_1 \\ N_2 &= \xi_2 \\ N_3 &= \xi_3 \end{aligned}$$

For second order triangles, there are 2 types of shape functions: those for the vertices and those for the intermediate points. The shape functions for the vertices resp. the intermediate points read:

$$\begin{aligned} N_1 &= \xi_1(2\xi_1 - 1) & N_4 &= 4\xi_2\xi_3 \\ N_2 &= \xi_2(2\xi_2 - 1) & N_5 &= 4\xi_1\xi_3 \\ N_3 &= \xi_3(2\xi_3 - 1) & N_6 &= 4\xi_1\xi_2 \end{aligned}$$

These shape functions make no assumptions on the shape of the elements: the same shape functions are used for elements with curved and with straight edges. It is the coordinate grid which becomes curved.

Quadrilateral

Quadrilaterals are 4-sided plane elements which can be curved. Since these are distorted squares, it seems logical to describe internal locations with 2 coordinates.

Shape functions that are quadratic in both coordinates call for 9 nodes, e.g. the 4 corner nodes, the 4 mid-side points and 1 node in the body. This element is called the 9-node bi-quadratic quadrilateral. Its shape functions are straightforward.

An alternative quadrilateral element, the so-called 8-node "serendipity" quadrilateral, leaves out the node in the body. For this element, one uses cubic shape functions:

$$
\begin{array}{ll}
N_1 = -(1-\xi_1)(1-\xi_2)(1+\xi_1+\xi_2)/4 & N_5 = (1-\xi_1)(1+\xi_1)(1-\xi_2)/2 \\
N_2 = -(1+\xi_1)(1-\xi_2)(1-\xi_1+\xi_2)/4 & N_6 = (1+\xi_1)(1+\xi_2)(1-\xi_2)/2 \\
N_3 = -(1+\xi_1)(1+\xi_2)(1-\xi_1-\xi_2)/4 & N_7 = (1-\xi_1)(1+\xi_1)(1+\xi_2)/2 \\
N_4 = -(1-\xi_1)(1+\xi_2)(1+\xi_1-\xi_2)/4 & N_8 = (1-\xi_1)(1+\xi_2)(1-\xi_2)/2
\end{array}
$$

Tetrahedra

The definitions for tetrahedra are similar to those used for triangles, the only difference is that there are 4 coordinates instead of 3.

$$
\begin{array}{ll}
N_1 = \xi_1(2\xi_1 - 1) & N_5 = 4\xi_1\xi_2 \\
N_2 = \xi_2(2\xi_2 - 1) & N_6 = 4\xi_1\xi_3 \\
N_3 = \xi_3(2\xi_3 - 1) & N_7 = 4\xi_1\xi_4 \\
N_4 = \xi_4(2\xi_4 - 1) & N_8 = 4\xi_2\xi_3 \\
 & N_9 = 4\xi_2\xi_4 \\
 & N_{10} = 4\xi_3\xi_4
\end{array}
$$

Pentrahedra and Hexahedra

Bearing in mind that the natural coordinates for hexahedra range from -1 to $+1$, their shape functions are as follows:

$$
\begin{array}{l}
N_1 = (1-\xi_1)(1-\xi_2)(1-\xi_3)/8 \\
N_2 = (1+\xi_1)(1-\xi_2)(1-\xi_3)/8 \\
N_3 = (1+\xi_1)(1+\xi_2)(1-\xi_3)/8 \\
N_4 = (1-\xi_1)(1+\xi_2)(1-\xi_3)/8 \\
N_5 = (1-\xi_1)(1-\xi_2)(1+\xi_3)/8 \\
N_6 = (1+\xi_1)(1-\xi_2)(1+\xi_3)/8 \\
N_7 = (1+\xi_1)(1+\xi_2)(1+\xi_3)/8 \\
N_8 = (1-\xi_1)(1+\xi_2)(1+\xi_3)/8
\end{array}
$$

1.3.3 Isoparametric coordinates

Definition

Given the simplicity of the shape functions, it appears natural to use them not only for the interpolation of the potential, but also for defining the coordinates. We do this by requiring that the Cartesian coordinates of any point can be expressed in terms of the Cartesian coordinates of the nodes by the equations

$$\begin{aligned} x &= \textstyle\sum_i x_i N_i(\xi) \\ y &= \textstyle\sum_i y_i N_i(\xi) \\ z &= \textstyle\sum_i z_i N_i(\xi) \end{aligned} \tag{1.1}$$

(the latter equation only in the 3-dimensional case). These equations are automatically satisfied at the nodes by virtue of the definition of the shape functions. If the natural coordinates also satisfy these equations in other points, then they are called isoparametric coordinates.

One can readily verify that the natural coordinates described above for triangles and tetrahedra are in fact isoparametric. This implies for second order triangles and tetrahedra that the sides do not have to be straight: second order shape functions are parabolic. Ansys uses this possibility, Maxwell does not.

Isoparametric coordinates eliminate the ambiguity of the behaviour of the natural coordinates between nodes - which does not mean that they are the only viable choice of coordinates. Apart from the elegance of the formulation, the main benefit for our purposes is that isoparametric coordinates greatly simplify the calculation of derivatives of the potential.

The downside of using isoparametric coordinates is that the relation between the Cartesian coordinates for which we need to compute the fields, and the natural coordinates that we use in the actual interpolation, is not particularly simple for some elements. For triangles and tetrahedra with straight sides, Eq. (1.1) can be solved exactly, but this is not the case for elements with curved sides nor for hexahedra and pentahedra. Instead, for hexahedra, pentahedra and curved tetrahedra, an iterative Newton-Raphson solution is used.

Tetrahedra

Tetrahedra are described with 4 isoparametric coordinates in 3-dimensional space. The isoparametric coordinates are solved under the normalisation constraint:

$$s = \xi_1 + \xi_2 + \xi_3 + \xi_4, s = 1$$

An initial estimate of the isoparametric coordinates is obtained using the expressions for the tetrahedron with straight sides. This works well in practice since finite element programs take care not to curve the sides of the elements excessively.

The isoparametric coordinates are refined using Newton-Raphson iteration:

$$\begin{pmatrix} 1 \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \xi_1 + \xi_2 + \xi_3 + \xi_4 \\ \sum_i x_i N_i(\xi_1 + \epsilon_1, \xi_2 + \epsilon_2, \xi_3 + \epsilon_3, \xi_4 + \epsilon_4) \\ \sum_i y_i N_i(\xi_1 + \epsilon_1, \xi_2 + \epsilon_2, \xi_3 + \epsilon_3, \xi_4 + \epsilon_4) \\ \sum_i z_i N_i(\xi_1 + \epsilon_1, \xi_2 + \epsilon_2, \xi_3 + \epsilon_3, \xi_4 + \epsilon_4) \end{pmatrix} \approx \begin{pmatrix} \xi_1 + \xi_2 + \xi_3 + \xi_4 \\ \sum_i x_i N_i(\xi_1, \xi_2, \xi_3, \xi_4) \\ \sum_i y_i N_i(\xi_1, \xi_2, \xi_3, \xi_4) \\ \sum_i z_i N_i(\xi_1, \xi_2, \xi_3, \xi_4) \end{pmatrix} + J \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \end{pmatrix}$$

where the Jacobian has the form

$$J = \frac{\partial(s,x,y,z)}{\partial(\xi_1,\xi_3,\xi_3,\xi_4)} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ \sum_i x_i \frac{\partial N_i}{\partial \xi_1} & \sum_i x_i \frac{\partial N_i}{\partial \xi_2} & \sum_i x_i \frac{\partial N_i}{\partial \xi_3} & \sum_i x_i \frac{\partial N_i}{\partial \xi_4} \\ \sum_i y_i \frac{\partial N_i}{\partial \xi_1} & \sum_i y_i \frac{\partial N_i}{\partial \xi_2} & \sum_i y_i \frac{\partial N_i}{\partial \xi_3} & \sum_i y_i \frac{\partial N_i}{\partial \xi_4} \\ \sum_i z_i \frac{\partial N_i}{\partial \xi_1} & \sum_i z_i \frac{\partial N_i}{\partial \xi_2} & \sum_i z_i \frac{\partial N_i}{\partial \xi_3} & \sum_i z_i \frac{\partial N_i}{\partial \xi_4} \end{pmatrix} \tag{1.2}$$

The 4×4 Jacobian has been inverted with Mathematica, which saves the overhead of numerically solving the equations at each iteration. The Jacobian is subsequently used when computing the electric field.

Quadrilaterals, hexahedra and pentahedra

Hexahedra and pentahedra are 3-dimensional elements for which we use 3 isoparametric coordinates (ξ_1, ξ_2, ξ_3). Quadrilaterals are the 2-dimensional equivalent of the hexahedral element, with 2 instead of 3 isoparametric coordinates. Apart from the number of dimensions, the methods used are identical. An initial estimate for the isoparametric coordinates is obtained from the ratio of distances to the planes with which the coordinates are linked:

$$\xi = \frac{d_1 - d_2}{d_1 + d_2}$$

Next, one iteratively refines the estimate with ϵ update vectors computed from:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \sum_i x_i N_i(\xi_1+\epsilon_1, \xi_2+\epsilon_2, \xi_3+\epsilon_3) \\ \sum_i y_i N_i(\xi_1+\epsilon_1, \xi_2+\epsilon_2, \xi_3+\epsilon_3) \\ \sum_i z_i N_i(\xi_1+\epsilon_1, \xi_2+\epsilon_2, \xi_3+\epsilon_3) \end{pmatrix} \approx \begin{pmatrix} \sum_i x_i N_i(\xi_1, \xi_2, \xi_3) \\ \sum_i y_i N_i(\xi_1, \xi_2, \xi_3) \\ \sum_i z_i N_i(\xi_1, \xi_2, \xi_3) \end{pmatrix} + J \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{pmatrix}$$

The Jacobian J is given by:

$$J = \frac{\partial(x,y,z)}{\partial(\xi_1,\xi_3,\xi_3)} = \begin{pmatrix} \sum_i x_i \frac{\partial N_i}{\partial \xi_1} & \sum_i x_i \frac{\partial N_i}{\partial \xi_2} & \sum_i x_i \frac{\partial N_i}{\partial \xi_3} \\ \sum_i y_i \frac{\partial N_i}{\partial \xi_1} & \sum_i y_i \frac{\partial N_i}{\partial \xi_2} & \sum_i y_i \frac{\partial N_i}{\partial \xi_3} \\ \sum_i z_i \frac{\partial N_i}{\partial \xi_1} & \sum_i z_i \frac{\partial N_i}{\partial \xi_2} & \sum_i z_i \frac{\partial N_i}{\partial \xi_3} \end{pmatrix} \tag{1.3}$$

Since one needs the inverse of the Jacobian during this iteration, the method looks numerically expensive. Fortunately, the initial estimate is usually excellent because finite element method programs avoid elements with strongly curved sides. As a result, only a few iterations are needed in general. Further, the Jacobian is only 3×3 so that it can easily be inverted analytically. In addition, we reuse the inverse Jacobian when computing the gradient of the potential.

1.4 Potential and the Field interpolation

1.4.1 Interpolation formulae

The potential is the prime target of the finite element program and the potentials at the nodes are output in the form of a field map. This field map is interpolated with the shape functions using the natural coordinates:

$$V = \sum_i v_i N_i(\xi) \tag{1.4}$$

Some finite element programs, such as Maxwell, output a field vector for each element to which a node belongs. If this is the case, then one can interpolate the field the same way one interpolates the potential:

$$\begin{aligned} E_\mathrm{x} &= \textstyle\sum_i e_{\mathrm{x}_i} N_i(\xi) \\ E_\mathrm{y} &= \textstyle\sum_i e_{\mathrm{y}_i} N_i(\xi) \\ E_\mathrm{z} &= \textstyle\sum_i e_{\mathrm{z}_i} N_i(\xi) \end{aligned} \tag{1.5}$$

Other programs provide only one field vector per node, even if the node belongs to several elements. This poses a problem on interfaces between materials with different dielectric constants because the perpendicular component of the field is discontinuous across such borders (the tangential component is continuous). Hence, the field values at the nodes on the interface that should be used when interpolating in elements on one side of the interface, are different from the values to be used for interpolation on the other side.

Fortunately, computing the gradient in elements is simple:

$$\begin{pmatrix} E_\mathrm{x} \\ E_\mathrm{y} \\ E_\mathrm{z} \end{pmatrix} = \begin{pmatrix} \frac{\partial \xi_1}{\partial x} & \frac{\partial \xi_2}{\partial x} & \frac{\partial \xi_3}{\partial x} \\ \frac{\partial \xi_1}{\partial y} & \frac{\partial \xi_2}{\partial y} & \frac{\partial \xi_3}{\partial y} \\ \frac{\partial \xi_1}{\partial z} & \frac{\partial \xi_2}{\partial z} & \frac{\partial \xi_3}{\partial z} \end{pmatrix} \begin{pmatrix} \sum_i v_i \frac{\partial N_i}{\partial \xi_1} \\ \sum_i v_i \frac{\partial N_i}{\partial \xi_2} \\ \sum_i v_i \frac{\partial N_i}{\partial \xi_3} \end{pmatrix} \tag{1.6}$$

The matrix in this equation is readily identified with the inverse Jacobian J^{-1} of Eq. (1.3).

1.4.2 Continuity

Every derivative of the potential inside an element is continuous because the shape functions have this property. The field is the gradient of the potential and is therefore continuous inside each element.

Continuity across the border between elements is less obvious. The finite element method guarantees that the potential of a node is the same whether the node is treated as belonging to one element or to another. Thus, on an interface between 2 elements, all nodes will have the same potential seen from both sides. Since moreover the potential on the interface is determined only by the nodes that are located on the interface, and not by the other nodes of the element, the potential on the interface will be the same whether calculated using the nodes of one element or the nodes of the other. Therefore, the potential is continuous across the interface.

With the shape functions discussed earlier, there is no guarantee that also the derivatives of the potential are continuous, and in practice, these derivatives, i.e. the field components, are indeed discontinuous across element interfaces. Elements have been devised which provide additional degrees of freedom which can be used to ensure continuity of the derivatives, but these are not available in the programs with which Garfield currently has an interface.

1.5 Mesh refinements and Virtual volumes

With only few exceptions, finite element programs include techniques which generate a mesh and which iteratively refine the mesh. Although these techniques are reliable, they are usually not written with the specific configurations in mind that occur in particle detectors.

Finite element programs are designed with engineering applications in mind, i.e. structures which have a reasonable homogeneity in dimensions. Suppliers of finite element programs may in fact warn the user that the program should not be used if the range of object dimensions exceeds e.g. 3 orders of magnitude. Particle detectors almost invariably have both tiny electrodes, e.g. the anodes, and large empty space, and they often do not fall within the stated range of applicability.

Simple meshing programs may generate elements with enormous aspect ratios if the range of object dimensions is large. Since the fields inside elements is as a rule a low order polynomial, while the field in the vicinity of small structures definitely is not polynomial, the approximation of the field will be particularly poor Better meshing programs will try and ensure that the aspect ratio of elements remains close to unity. But even then, it is generally worth the effort to closely monitor the mesh which is being generated.

In practice, it turns out that meshing can be assisted efficiently by enveloping tiny structures with a Russian-doll style set of larger and similarly shaped structures, all filled with the same medium. This technique, called *virtual elements*, works since the meshing programs do not generate elements that cross material boundaries, even if the dielectric constants happen to be the same. Well used, the technique leads to more accurate solutions with fewer elements, while the elements have smaller aspect ratios which improves the accuracy of, in particular, the fields.

1.6 Advantages and disadvantages

The main interest of using finite element methods is that dielectric media as well as nearly arbitrarily shaped electrodes can be handled. Thus, the technique lends itself to a range of detectors where dielectrics are an important ingredient, such as GEMs and MSGCs, for which the fields can not be computed analytically.

The principal drawbacks of the method are:

- The potential is usually a low order polynomial, and the electric field is therefore an even lower order polynomial. These low order polynomials are not well suited to approximate the $\log(r)$ and $1/r^2$ fields encountered in particle detectors. It is therefore very difficult to obtain accurate estimates of the gain, for which precise knowledge is required of the high fields in the immediate vicinity of the electrodes.

- Finite element methods compute the potential, which is irrelevant for physics processes, and unfortunately not the field which determines the behaviour of particles inside the detector. The finite element method guarantees that the potential is continuous but the electric field often is not, see Section 1.4.2, and e.g. tracing particles through a discontinuous electric field poses problems.

- Finite element methods tend to take significantly more time than analytic calculations in cases where both methods can be used.

Other techniques, such as the integral equation method, have been devised which do not suffer from these problems - but programs that perform these calculations are less commonly available.