Solving non-linear PDEs using the finite element method

# Introduction

This document shows how non-linear finite element problems can be solved using a Newton-Raphson method. We first of all show the technique using a simple system: a 1D electrostatic problem in which the permittivity depends on the electrostatic potential and/or its spatial derivative. We then extend the analysis to allow 3D multiphysics coupled simulations such as those solved by Zinc.

# One dimensional electrostatic calculation

x=xmin

x=xmax

4=imax

element 4

x

3

0

1

2

*Figure 1: Discretisation of electrostatic potential using linear shape functions, shown as thin lines. The function is shown as a bold line.*

Considering figure 1, we write the solution ϕ(x) as

Where Ni(x) is a shape functions centred on node i. The electrostatic differential equation we wish to solve is

With boundary condition,

Which holds at the boundaries x=xmax, xmin. The unit normal, n, takes the value +1 at x=xmax and at x=xmin.

Substituting (1) into (2) we get

Using the Galerkin method, we multiply both sides of this equation by the shape function Nk(x) which acts as a test function to give

Integrating over the simulation range [xmin,xmax] gives

Which is a set of linear equations in the as required. However, since we have used linear shape functions, we will need to replace the second derivative terms with a product of first derivatives. Using integration by parts gives

The term [..] which is evaluated at the simulation boundary gives a type of boundary condition. It is now possible to simplify this term using , and (3) to give

The first term will disappear if we choose the “natural” boundary condition with g=0 in (3). We can write this term using a surface integral notation,

Where the surface integral just means “take the value at either end of the 1D system and subtract”. This notation indicates that this term will become a true surface integral in 3D. These equations can be written in the form

Where the notation indicates that depends on the set of all ϕi, that is, the solution at each node. The parameters and are defined as

Where the second form (in each case) indicates that the global integration can be replaced with an integral over only elements indicated due to the local nature of the shape functions. Thus, indicates a sum only over elements (e) which connect nodes i and k and indicates sum over elements which connect to node k. This immediately implies will be a sparse matrix: in fact, in 1D, it will be triple diagonal. In 2D or 3D it is triple diagonal with additional “side bands”.

We could evaluate (5) directly from its definition, we would loop over k,i (considering only k,i which are connected by an element, else ) and then sum over all elements connected to both k and i. However, an alternative method is to instead loop over elements (e), consider every possible pair of nodes i,k connected to (e) and make a contribution to Qki from the element integral. This gives an element-based summation,

With initially. This is the usual method of forming the matrix Q especially in unstructured meshed where it is difficult to discover all elements connected to a node pair i,k and much easier to loop over elements.

(As an aside, in 3D, a similar analysis would give the analogous results, namely,

Assuming isotropic permittivity ϵ. If permittivity is anisotropic, the equations must be slightly generalised. The surface integral now takes its normal meaning.)

# Jacobian calculation

x=xmin

x=xmax

4=imax

element 4

x

3

0

1

2

*Figure 2: Interpretation of where k=1 in this case. The result of perturbing is to alter in the vicinity of (red). Consequently the function is changed and the relative difference is (green) which is non-zero only in elements connected to k=1.*

Assuming we have prepared the matrices Q, R, to solve the set of non-linear equations (4), we can use the Newton-Raphson method which is based on Taylor’s series. We can write the residual F as

Using Taylor’s series, we can write

If is a better guess at the solution than we require so that

where is the Jacobian matrix defined so that

We can solve equation (7) for and thus get a better guess to the solution. This gives an iteration scheme based on an initial guess .

It is clear from (6) and (8) that

From the definition, (5), we may expand this as

so that

where the sum over j should include only nodes connected to both nodes i and k, else there will be no contribution to . Also, unless i is connected to k by an element. Recalling that , the function may be understood as the relative difference in the function when the solution at node k is perturbed as shown in Figure 2. Owing to the discretisation scheme (1), this “difference function” will be at a maximum at node k and reduce to zero on neighbouring nodes. It is zero on all nodes more distant than nearest neighbour just like the shape functions themselves (and their derivatives). A similar result holds for .

From (1), the term may be written as

Where we assume and . In practice, would be functions supplied by the user. We obtain from (9),

Considering the locality of the shape functions, this can again be simplified by writing in terms of integrals over elements neighbouring nodes i and k:

In the first sum, we only consider elements (e) connected to i, j and k; in the second we consider elements connected to I and k. For the second and third terms, it is again possible to sum over elements (e) and then loop over ijk or ik to obtain contributions to ,

# Numerical check

To prove that Q and J can be calculated as we have described, these matrices were prepared using a simple 1D finite element system. We used

and

The numerical results were as follows, with equation numbers illustrating how the matrix was calculated in each case:

Q Matrix by elements (equation 5a):

7.330 -7.330 0.000 0.000

-7.330 16.663 -9.332 0.000

0.000 -9.332 34.645 -25.313

0.000 0.000 -25.313 25.313

J Matrix by perturbation (equation 8, the definition of J):

3.991 -11.993 0.000 0.000

-3.991 23.993 -7.000 0.000

0.000 -12.000 13.939 -51.944

0.000 0.000 -6.940 51.944

J Matrix by elements (equation 12):

3.990 -11.990 0.000 0.000

-3.990 23.988 -6.997 0.000

0.000 -11.998 13.935 -51.938

0.000 0.000 -6.938 51.938

Q Matrix full integral: (equation 5, first integral)

7.330 -7.330 0.000 0.000

-7.330 16.662 -9.332 0.000

0.000 -9.332 34.645 -25.313

0.000 0.000 -25.313 25.313

J Matrix full integral (equation 10):

3.990 -11.990 0.000 0.000

-3.990 23.988 -6.997 0.000

0.000 -11.998 13.935 -51.938

0.000 0.000 -6.938 51.938

J Matrix by node pairs (equation 11):

3.990 -11.990 0.000 0.000

-3.990 23.987 -6.997 0.000

0.000 -11.998 13.935 -51.938

0.000 0.000 -6.938 51.938

It is clear that the Q and J matrix calculations give the same answers, within numerical error, using all techniques. The “Q, J matrix by elements” techniques are the ones that would be normally used in practice.

# Extension to Zinc calculation

We now wish to extend these techniques to calculate the Jacobian matrix using Zinc which operates in 3D and allows anisotropy and multiple dependent variables. We will use an analogous calculation with the following replacements:

The PDEs we wish to solve have the form,

with implied summation over repeated indices. We will use the convention that i,k sum over 1,2,…N where N is the number of variables to solve for and j,l=1,2,3 and that . At the boundary, is the outward unit normal vector. In this section, we will assume that the a, q and g matrices are linear so give no contribution to the Jacobian. In Zinc, the Q and R matrices are given by

where α and β are unique node index numbers. The index pairs and are compressed into a single index to give the non-linear equations (4) and Jacobian (6). For the Jacobian, we obtain,

where we only need to consider α, γ node pairs connected by an element and the sum over β should only encompass nodes connected to both α and γ. From (13) and (14) we can write this as

The discretisation scheme has the form

so that

where . From (17), we can write this as

Substituting in (16), we obtain,

The integrals again only need to be evaluated over elements which connect the relevant nodes

Again, the most convenient way to evaluate the 2nd and 3rd terms is by summing over elements:

with initially. The derivative functions must be supplied by the user as follows

where and **du** such that are the inputs the functions and the return values are the matrices shown. In Zinc, the “token” string essentially encodes the values of (e) and i (elements are tagged as being in a particular region then individual components are given token strings in the Zinc input file. The user written functions ffun, dffun\_du, dffun\_ddu then refer to these tokens). A similar set of function are supplied for the C matrix: Cfun, dCfun\_du and dCfun\_ddu.

# Some practical considerations

Schemes for preparing the Jacobian such as (20) involve setting the Jacobian to the Q matrix initially then adding on new elements. This can be problematic when both matrices are stored in sparse format (as is essential for moderate or large FE problems) since adding two sparse matrices can be a problem if N is the number of non-zeros in the matrix. This is because we potentially need to scan all of Q to see if each element of J is present (or vice versa). There is also the issue of storage: it might not be practical to store all three terms of (20) as sparse matrices and then add the result into a 4th matrix. It is better to accumulate all terms into a single matrix but this can also be difficult. However, it may be that the Jacobian has the same sparsity pattern as the Q matrix. In this case it is easy to first set J equal to Q then add the extra terms onto J without appending any additional non-zero values. If the two matrices have the same sparsity pattern, the addition is a linear process of O(N). From the numerical experiments on 1D shown here, the Jacobian indeed seems to have the same form as Q, that is triple diagonal. It is not clear if this result is true with multiple variables and 3D, however. Currently, Zinc assumes the two matrices have the same sparsity pattern but will fail with an error if this is found not to be true.

Another issue is the integration in (20). Owing to the arbitrary user-supplied terms etc, the integrand could be non-polynomial or a high-order polynomial. The Gaussian quadrature process normally used to integrate over the elements assumes a low-order polynomial (specifically, if we evaluate at p points, the integration is exact if the integrand is a polynomial of order 2p-1 or less. In Zinc, p=3 is chosen since this is sufficient in the linear case). One option would be to allow the user to control the number of Gaussian quadrature evaluation points. Another might be to take the terms (etc) out of the integrals and then integrate only shape functions which have a guaranteed polynomial order depending on the type of the shape functions (in this document we assumed linear). This would imply evaluating at some representative point, perhaps the centre of the element and, of course, is less accurate. But it may prevent a catastrophic failure due to a non-polynomial function being supplied by the user and erroneously integrated with Gaussian quadrature. If we extract the material properties terms from the integrals we obtain, from (19),

where the material properties are assumed to be evaluated in the centres of the elements.

An issue we haven’t discussed is Dirichlet boundary conditions. These conditions imply some of the dependent variables will be fixed on some nodes. Thus, we should use (7) to update only non-fixed degrees of freedom, leaving fixed ones as set by the user. But it is not sufficient to avoid updating some degrees of freedom: the update for each node depends on the whole matrix J and this must be adjusted to account for the fixed nodes. ***This means that Jacobian rows corresponding to fixed degrees of freedom must be set all zeros except for a single “1” on the pivot***. In other words, the Jacobian matrix is equal to the FE Q-matrix for those rows. Since the J matrix is, in practice, generated from the Q-matrix it is sufficient to do nothing on a row corresponding to a fixed degree of freedom so the J matrix is left equal to the Q matrix for that row. This is how Zinc operates in practice.