Understanding residuals produced by Zinc

John Blackburn

Contents

[Introduction 1](#_Toc460537481)

[Linear dielectric example 1](#_Toc460537482)

[Linear piezoelectric example 6](#_Toc460537483)

[Non-linear electro-thermal problem 10](#_Toc460537484)

[Fuel cell problem 24](#_Toc460537485)

[Theoretical background 26](#_Toc460537486)

[Activation of physics in different regions 31](#_Toc460537487)

[Zinc matrices 34](#_Toc460537488)

[Formulation in terms of mass fractions 36](#_Toc460537489)

[Results 39](#_Toc460537490)

[Conclusions 43](#_Toc460537491)

[Appendix: Example of residuals for a specific matrix 43](#_Toc460537492)

# Introduction

During the convergence process, Zinc attempts to estimate how close it is to a solution by outputting *residuals*. If the residual is zero an exact solution has been found, but this never happens in practice! So it is important to understand “how small is small?” For non-linear problems, there is another issue in that Zinc solves the non-linear equations by repeatedly solving linear equations. These intermediate linear equations have their own residuals which must be considered separately from the overall non-linear residual which dictates how close we are to the solution of the entire non-linear problem. Finally residuals can be defined in several ways: as a vector or a scalar quantity and also in a diagonally scaled version which allows physical quantities with different units to be considered on an equal footing.

To illustrate we go through some of the Zinc tutorial examples, starting with a linear problem and moving on to Multiphysics and then non-linear problems. The most advanced problem is a fully coupled fuel cell with strong non-linearity and 8 dependent variables. Some of the simulations presented may be of interest in their own right, as they show new examples of the sort of problems Zinc can solve. The final fuel cell problem is more advanced than the fuel cell tutorial example in the tutorial manual.

# Linear dielectric example

This is a simple, linear problem with only one dependent variable: the voltage V. The input file looks like this:

solver=UMFPACK

nvar=1

ndim=3

removeFixed=YES

itmax=10000

tol=1e-12

omega=1.5 ! only used for SOR

labels = V

region 1 elements C

V V = [eps1, 0, 0, &

0, eps1, 0, &

0, 0, eps1]

end

region 2 elements C

1 1 = [eps2, 0, 0, &

0, eps2, 0, &

0, 0, eps2]

end

region 3 nodes

V = 1.0

end

region 4 nodes

V = 0.0

end

init

V = 0.0

Zinc produces the following output:

Starting UMFPACK (direct solver)

Set to column format

Symbolic calculation

Numeric calculation

Symbolic= 3059984

Numeric= 4132568

Solve

**Direct residual:**

**0.52469E-15, 0.16639E-10 ?= 0.16639E-10**

**Normalised residual:**

**0.45887E-15, 0.40857E+01 ?= 0.40857E+01**

NL step iterations error error index

1 -1 -0.10000E+01 -1

Final Energy = 9.2689276655527343E-012

Final energy from Q= -7.6173905399177685E-011

Writing final state to .zou

Opening file:

dielectric.zou

Zinc completed successfully

Simulation: dielectric

Here we use the UMFPACK solver which is a direct solver explicitly solving the equations using Gaussian elimination. Since it is not an iterative solver, UMFPACK does not advance towards the solution like other solvers and so does not output a convergence table showing residuals as a function of iteration number (it also does not use the initial guess supplied in the “init” section). However, when the UMFPACK solver completes, Zinc outputs the final residuals (shown bold) so we can check if the solution was accurate.

Zinc solves the following equation

where u contains the solution to the problem and is a vector equal in size to the number of degrees of freedom in the problem. In the case of the above simulation there is 1 variable and 11x11x11 nodes to a total of 1331 degrees of freedom. Thus **u** is a vector of length 1331. The vector **R** is of the same length and contains the Dirichlet boundaries on the problem and terms within the f-matrix (these are zero here but, in non-zero, would correspond to charge density in a dielectric problem). The **Q** matrix maps to the Zinc C-matrix (containing permittivities in our dielectric problem) and is of size 1331x1331 (stored sparse: only non-zero entries are stored)

We define the residual as

or in component terms

Note that the residual is a vector quantity. All its entries are zero if we get an exact solution to the problem, but this never happens in practice. To get a feel for how large **r** is we use its l2-norm, but we do not plot this directly as this parameter would increase with the size of the vector **u** and would be hard to interpret. To avoid this we introduce a new parameter,

This quantity is always bounded in the range [0,1] (see Appendix). This is the quantity output by Zinc, r’=**0.52469E-15** in the above simulation.This number is very small compared to 1.0 so we can be sure the problem solved accurately. The other two numbers printed are

These numbers represent the left and right hand sides of the equations to be solved: they should be equal and here we see they are equal up to the decimal places output. Note that these quantities are not dimensionless so might be naturally small or large compared to 1. In practice getting the “left hand side” and “right hand side” to agree in this way is not difficult and the problem may not have converged despite apparent agreement of the left and right hand side. It is better to consult the more sensitive parameter, . The other benefit of dividing by the left hand side to obtain r’ is to give a dimensionless quantity. The Zinc Q matrix is in units of the (C matrix units) length. In our problem, the C-matrix contains permittivities (unit F/m) so and []=F.V (Farads times Volts). In our particular problem we have free space permittivities ( F/m) and Voltages and distances of order unity so will be of order F/m. The above values of and confirm this. So simply looking at itself might give a false impression of good convergence even when this has not been achieved. The scaled residual magnitude avoids all these problems by providing a number between 0 and 1. The Appendix shows a worked example of residuals for a particular, simplified, matrix Q giving evidence that is indeed a good parameter to estimate convergence.

Let us now consider the second set of parameters output by Zinc, the *diagonally-scaled residual* (and corresponding left- and right-hand-sides). This residual has the form,

where **D** is a diagonal matrix consisting of the diagonal elements of **Q**,

so that

We can write the residual in component form as

The parameter Zinc writes out is

In our case this parameter is 0.45887e-15 (dimensionless and comparable to 1). This parameter may be a better estimate of convergence than r’ especially in multiphysics problems or problems with large variation of the Zinc C matrix across the system. Note that the magnitude is not dimensionless but in units of u. For Multiphysics problems multiplying by the matrix is *not* the same thing as reducing the problem to a dimensionless one. To accomplish the latter, we would have to introduce auxiliary dimensionless variables, e.g., for a problem where we solve for voltages we might introduce then solve for V’ and recover V at the end of the simulation. Zinc does not do this automatically.

The procedure of pre-multiplying both sides of the equation by is used by many of Zinc’s internal solvers and is called diagonal scaling. The “energy minimisation” solvers: Generalised Mean Residual (GMRES) and BiConjugate Gradient (BCG) can optionally use diagonal scaling to precondition the matrices to be solved. That is they solve the equations

where and The matrix **Q’** often has a lower (better) condition number than **Q**. The other preconditioning available to Zinc is incomplete LU factorisation, so there are 4 solvers available called: ***BCG\_DIAG, BCG\_LU, GMRES\_DIAG, GMRES\_LU***. Again, the diagonal preconditioning strategy is not the same as reducing the original PDEs to dimensionless form and it might be worth users doing this before solving a problem in Zinc.

We now solve the same problem using the Successive Over Relaxation (SOR) solver. The input file is the same except we use the command

SOLVER=SOR

The output is:

Starting SOR solver

iter resid lhs rhs residDiag lhsDiag rhsDiag

0 Infinity 0.00000E+00 0.16639E-10 Infinity 0.00000E+00 0.40857E+01

10 0.42896E-01 0.16712E-10 0.16639E-10 0.37120E-01 0.41021E+01 0.40857E+01

20 0.45162E-02 0.16645E-10 0.16639E-10 0.39472E-02 0.40871E+01 0.40857E+01

30 0.47128E-03 0.16640E-10 0.16639E-10 0.41179E-03 0.40858E+01 0.40857E+01

40 0.49160E-04 0.16639E-10 0.16639E-10 0.42953E-04 0.40857E+01 0.40857E+01

50 0.51278E-05 0.16639E-10 0.16639E-10 0.44804E-05 0.40857E+01 0.40857E+01

60 0.53487E-06 0.16639E-10 0.16639E-10 0.46734E-06 0.40857E+01 0.40857E+01

70 0.55791E-07 0.16639E-10 0.16639E-10 0.48747E-07 0.40857E+01 0.40857E+01

80 0.58194E-08 0.16639E-10 0.16639E-10 0.50847E-08 0.40857E+01 0.40857E+01

90 0.60701E-09 0.16639E-10 0.16639E-10 0.53037E-09 0.40857E+01 0.40857E+01

100 0.63315E-10 0.16639E-10 0.16639E-10 0.55322E-10 0.40857E+01 0.40857E+01

110 0.66043E-11 0.16639E-10 0.16639E-10 0.57705E-11 0.40857E+01 0.40857E+01

120 0.68889E-12 0.16639E-10 0.16639E-10 0.60190E-12 0.40857E+01 0.40857E+01

**Direct residual:**

**0.68889E-12 0.16639E-10 0.16639E-10**

**Normalised residual:**

**0.60190E-12 0.40857E+01 0.40857E+01**

NL step iterations error error index

1 120 0.68889E-12 0

Final Energy = 9.2689276655527198E-012

Final energy from Q= -7.6173905399165872E-011

Writing final state to .zou

Opening file:

dielectric.zou

Zinc completed successfully

Simulation: dielectric

The SOR solver solves the problem iteratively starting from the initial condition. As the system converges it outputs residuals, in this case every 10 iteration steps. “resid” shows the r’ parameter and “residDiag” shows the parameter already discussed (and corresponding left- and right-hand-sides). The iteration process continues until tol which was set to 1e-12 in the input file (a maximum of ***itmax*** iterations is performed). The final residual is thus 0.6889e-12 and the normalised (diagonal scaled) residual is 0.6019e-12.

Note that Zinc outputs a summary of the linear solutions discovered under “NL step”. In this case we have a linear problem so only 1 step is used, but more steps are used for non-linear problems. In this case the final error is simply the final value of r’ discovered during the iteration process. The number of iterations needed is also output, 120 in this case.

We now run the same simulation using the BCG\_DIAG solver (BiConjugate Gradient with DIAGonal scaling). We use the command SOLVER=BCG\_DIAG with all other settings the same, and the result is

Starting Biconjugate Gradient (Diag scaled)

Direct residual:

0.98577E-12 0.16639E-10 0.16639E-10

Normalised residual:

0.96436E-12 0.40857E+01 0.40857E+01

NL step iterations sol error error index

1 37 0.98575E-12 0

Final Energy = 9.2689276655527214E-012

Information on the convergence is given in the ZLS file output by Zinc (dielectric.zls in this case). In this case the file shows

Preconditioned BiConjugate Gradient for N, ITOL = 1089 1

ITER Error Estimate Alpha Beta

0 0.1000000D+01

1 0.5006425D+00 0.1348720D+01 NaN

2 0.3425275D+00 0.1758974D+01 0.2513077D+00

3 0.2730975D+00 0.1884244D+01 0.4364351D+00

4 0.2239430D+00 0.2166525D+01 0.5313100D+00

5 0.1972140D+00 0.2155250D+01 0.6785087D+00

6 0.1697221D+00 0.2179794D+01 0.7358273D+00

7 0.1617974D+00 0.2430604D+01 0.7732186D+00

8 0.1383788D+00 0.2271453D+01 0.8959031D+00

9 0.9495873D-01 0.2222180D+01 0.7837806D+00

10 0.2426812D-01 0.9800804D+00 0.4155416D+00

11 0.8727081D-02 0.1344207D+01 0.6785328D-01

12 0.4379299D-02 0.1250810D+01 0.1460045D+00

13 0.2430045D-02 0.1464078D+01 0.2123514D+00

14 0.9642100D-03 0.1410619D+01 0.3046389D+00

15 0.3457117D-03 0.1210950D+01 0.1656403D+00

16 0.1458094D-03 0.1294455D+01 0.1157578D+00

17 0.8723556D-04 0.1808022D+01 0.2122891D+00

18 0.4746891D-04 0.1572721D+01 0.3879863D+00

19 0.3435091D-04 0.1619070D+01 0.2777174D+00

20 0.1814273D-04 0.1941417D+01 0.4618009D+00

21 0.7708649D-05 0.1316531D+01 0.2910037D+00

22 0.3581072D-05 0.1372767D+01 0.1899908D+00

23 0.1770119D-05 0.1686682D+01 0.2641224D+00

24 0.5311619D-06 0.1253382D+01 0.2122924D+00

25 0.1568120D-06 0.1118589D+01 0.9642678D-01

26 0.5074887D-07 0.1130810D+01 0.8087880D-01

27 0.1411504D-07 0.1172043D+01 0.1026291D+00

28 0.5845532D-08 0.1237902D+01 0.9401053D-01

29 0.1751103D-08 0.1270611D+01 0.1609754D+00

30 0.4941417D-09 0.1062875D+01 0.8976068D-01

31 0.1243504D-09 0.1114526D+01 0.7767594D-01

32 0.2652198D-10 0.1032048D+01 0.6640058D-01

33 0.7056037D-11 0.1045633D+01 0.4472956D-01

34 0.4385714D-11 0.1342485D+01 0.6799652D-01

35 0.5182258D-11 0.3364429D+01 0.4095597D+00

36 0.2709131D-11 0.2237315D+01 0.1333116D+01

37 0.9857508D-12 0.1241773D+01 0.2727991D+00

This output is generated directly by the packaged library routine ***dsdbcg*** (part of the open source SLATEC library and available from Netlib). The error estimate is clearly r’ as derived previously. In this case 37 iterations were needed to achieve the tolerance of ***tol=1e-12*** specified in the input file.

# Linear piezoelectric example

We now consider the Zinc piezoelectric tutorial example. This is still a linear problem but it is Multiphysics as we are solving for both voltage and elastic displacement: 3 components of elastic displacement, and the voltage, V. Here is the beginning of the Zinc piezo.zin file:

solver=UMFPACK

removeFixed=YES

nvar=4

ndim=3

scale=0.01

itmax=10000

tol=1e-8

nstride=100

omega=1.0

residual=pivots

labels= ux uy uz V

The result is

Starting UMFPACK (direct solver)

Set to column format

Symbolic calculation

Numeric calculation

Symbolic= 3047256

Numeric= 4127704

Solve

Direct residual:

0.22750E-14 0.45257E+00 0.45257E+00

Normalised residual:

0.25552E-14 0.57730E+01 0.57730E+01

NL step iterations error error index

1 -1 -0.10000E+01 -1

Final Energy = -1.3307226899883171E-010

Final energy from Q= 1.9331922924023067E-009

Writing final state to .zou

Opening file:

piezo.zou

Opening file:

piezo.resid

Zinc completed successfully

Simulation: piezo

So the final residual is 0.22750e-14 which is an excellent result. The parameter is similar at 0.25552E-14. As well as residual totals, Zinc can also output a map of residuals by using the “residual” command in the ZIN file. Zinc then outputs an additional file, piezo.resid, which can be read into Zmesh. The result is shown in Fig 1. At each node we output the residual corresponding to that node,

The residual magnitude discussed above is

Note that this residual value can be positive or negative and is not normalised unlike . Node residuals are separated depending on which variable they correspond to. For example Fig 1 corresponds to the residual of the V variable, shown as “V.resid”. The residual is not normalised so very small values can be obtained.



Figure 1: Plot of residual distribution for V in a piezoelectric problem.

We can also plot the left and right hand sides of the matrix. In our case we chose resid=pivot which means

(ie, only the pivot value appears in the left-hand-side, , and the remaining unknowns are shown in the right hand side). These results are shown in Figs 2 and 3. Note that LHS and RHS are virtually identical and both are much larger than the residual indicating the equations have been solved to good accuracy. Similar plots can be obtained for variables.

As well as the ***resid=pivots*** option, ***resid=unknowns*** is also available. The plot of residuals is identical but we change the meaning of the LHS and RHS at each point so that,

If neither option is selected, no residuals are output.



Figure 2: LHS of Voltage equations in a piezoelectric problem



Figure 3: RHS of Voltage equations in a piezoelectric problem

Let us now solve the problem using SOR. We keep the input file the same but use the command,

solver=SOR

The result is:

Starting SOR solver

iter resid lhs rhs residDiag lhsDiag rhsDiag

0 Infinity 0.00000E+00 0.45257E+00 Infinity 0.00000E+00 0.57730E+01

100 0.26158E-01 0.45271E+00 0.45257E+00 0.43377E-02 0.57727E+01 0.57730E+01

200 0.69760E-02 0.45254E+00 0.45257E+00 0.29447E-03 0.57730E+01 0.57730E+01

300 0.95241E-03 0.45256E+00 0.45257E+00 0.95511E-04 0.57730E+01 0.57730E+01

400 0.94825E-04 0.45257E+00 0.45257E+00 0.13825E-04 0.57730E+01 0.57730E+01

500 0.22831E-04 0.45257E+00 0.45257E+00 0.15353E-05 0.57730E+01 0.57730E+01

600 0.15307E-04 0.45257E+00 0.45257E+00 0.46111E-06 0.57730E+01 0.57730E+01

700 0.10420E-04 0.45257E+00 0.45257E+00 0.30832E-06 0.57730E+01 0.57730E+01

800 0.71012E-05 0.45257E+00 0.45257E+00 0.21079E-06 0.57730E+01 0.57730E+01

900 0.48432E-05 0.45257E+00 0.45257E+00 0.14391E-06 0.57730E+01 0.57730E+01

1000 0.33039E-05 0.45257E+00 0.45257E+00 0.98193E-07 0.57730E+01 0.57730E+01

1100 0.22539E-05 0.45257E+00 0.45257E+00 0.66989E-07 0.57730E+01 0.57730E+01

1200 0.15376E-05 0.45257E+00 0.45257E+00 0.45700E-07 0.57730E+01 0.57730E+01

1300 0.10489E-05 0.45257E+00 0.45257E+00 0.31176E-07 0.57730E+01 0.57730E+01

1400 0.71558E-06 0.45257E+00 0.45257E+00 0.21268E-07 0.57730E+01 0.57730E+01

1500 0.48816E-06 0.45257E+00 0.45257E+00 0.14509E-07 0.57730E+01 0.57730E+01

1600 0.33302E-06 0.45257E+00 0.45257E+00 0.98982E-08 0.57730E+01 0.57730E+01

1700 0.22719E-06 0.45257E+00 0.45257E+00 0.67525E-08 0.57730E+01 0.57730E+01

1800 0.15499E-06 0.45257E+00 0.45257E+00 0.46065E-08 0.57730E+01 0.57730E+01

1900 0.10573E-06 0.45257E+00 0.45257E+00 0.31426E-08 0.57730E+01 0.57730E+01

2000 0.72129E-07 0.45257E+00 0.45257E+00 0.21438E-08 0.57730E+01 0.57730E+01

2100 0.49206E-07 0.45257E+00 0.45257E+00 0.14625E-08 0.57730E+01 0.57730E+01

2200 0.33568E-07 0.45257E+00 0.45257E+00 0.99773E-09 0.57730E+01 0.57730E+01

2300 0.22900E-07 0.45257E+00 0.45257E+00 0.68065E-09 0.57730E+01 0.57730E+01

2400 0.15622E-07 0.45257E+00 0.45257E+00 0.46433E-09 0.57730E+01 0.57730E+01

2500 0.10658E-07 0.45257E+00 0.45257E+00 0.31677E-09 0.57730E+01 0.57730E+01

2600 0.72705E-08 0.45257E+00 0.45257E+00 0.21610E-09 0.57730E+01 0.57730E+01

Direct residual:

0.72705E-08 0.45257E+00 0.45257E+00

Normalised residual:

0.21610E-09 0.57730E+01 0.57730E+01

NL step iterations error error index

1 2600 0.72705E-08 0

Final Energy = -1.3307226899882747E-010

Final energy from Q= 1.9331922924011243E-009

Writing final state to .zou

Opening file:

piezo.zou

Opening file:

piezo.resid

Zinc completed successfully

Simulation: piezo

Although given a fairly modest tolerance of tol=1e-8, the SOR solver takes some time to find the solution, 2600 iterations (we allowed a maximum of tmax=10,000). The final result is 0.72705E-08 and 0.21610E-09. The residual plot for V is shown in Fig 4 and we can see the much greater residuals than in Fig 1. This shows the direct solver, UMFPACK is very good at solving problems to high accuracy.

# Non-linear electro-thermal problem

We now consider a non-linear, Multiphysics problem, with coupling between temperature and voltage. The geometry is a simple cuboid 0.1x0.1x0.2 m (x,y,z-directions) with , at z=zmax and V=0 at z=zmin. All other boundaries are open, Neumann boundaries. The material has a non-linear, temperature-dependent electrical conductivity, and a linear thermal conductivity, k. Due to electric currents, heat is generated as which acts as a thermal source. Aside from these non-linear issues, standard Poisson equations are used throughout.



Figure 4: Residual plot for voltage in piezoelectric problem using SOR.

The ZIN input file is like this:

solver=UMFPACK

newton=NO

nvar=2

ndim=3

itmax=1000

tol=1e-12

omega=1.5

scale=1

labels = V T

nstep=5

key\_q=1

restart=tv\_std.zou

region 1 elements C

V V = [$SIG,0,0, &

0,$SIG,0, &

0,0,$SIG]

T T = [k,0,0, &

0,k,0, &

0,0,k]

end

region 1 elements F

V = 0

T = $Q

end

region 2 nodes

V = Vtot

T = T0

end

region 3 nodes

V=0

end

init

V=(z+0.1)/0.2\*Vtot

T=T0

where the use of $Q and $SIG define non-linearity properties in Zinc. The meaning of these “token” strings are specified in functions within a DLL file provided by the user. The corresponding source code is shown next:

function cfun(token,x,y,z,ur,dur,nvar,istep,ireg,iregup,rnode,vec,imax,jmax,kmax)

implicit none

character(\*) token

integer nvar,istep,imax,jmax,kmax,ireg(\*),iregup(\*)

double precision cfun,x,y,z,ur(2),dur(2,3),rnode(\*),vec(\*)

double precision sig0,sig1,T0,T

sig0=5.97e7

sig1=0.1\*sig0

T0=298

T=ur(2)

if (token=='$SIG') then

cfun=sig0+sig1\*(T-T0)

else

print \*,'CFUN: Illegal token'

stop

endif

end function cfun

function ffun(token,x,y,z,ur,dur,nvar,istep,ireg,iregup,rnode,vec,imax,jmax,kmax)

implicit none

character(\*) token

integer nvar,istep,imax,jmax,kmax

double precision ffun,x,y,z,ur(2),dur(2,3)

integer ireg(0:imax,0:jmax,0:kmax),iregup(0:imax,0:jmax,0:kmax)

double precision rnode(0:imax,0:jmax,0:kmax,\*),vec(\*)

double precision sig0,sig1,Vx,Vy,Vz,T0

sig0=5.97e7

sig1=0.1\*sig0

T0=298

Vx=dur(1,1) ! -dV/dx

Vy=dur(1,2)

Vz=dur(1,3)

if (token=='$Q') then

ffun=sig0\*(Vx\*\*2+Vy\*\*2+Vz\*\*2)

else

print \*,'Illegal token'

stop

endif

end function ffun

which implement the non-linear terms mentioned above. Zinc currently has two methods of solving non-linear problems depending on the keyword ***newton*** in the ZIN file. If ***newton=NO*** (the default), Zinc follows a simple iterative method to solve the non-linear problem:

So for a given initial guess, , Zinc sets up the matrices **Q** and **R** based on this parameter. It then solves for which is taken as the next guess at the solution and used to form the next **Q** and **R** matrices and so on. This technique is very simple and does not require derivative quantities, but convergence can be poor and it is hard to understand the meaning of residuals as we will see.

The other technique with ***newton=YES*** is the Newton-Raphson method. Here we solve the problem

where **J** is the Jacobian matrix

and clearly involves knowledge of derivatives: the derivatives in **Q** map to derivates in Zinc’s **C, f** etc matrices. Note that the right hand side of this equation is just our familiar residual vector, **r**. We then solve for which may be regarded as a displacement in phase space which takes us closer to the non-linear solution:

which gives us a new guess at the solution from which new version of **Q, R** and **J** can be formed. The parameter is a damping factor which is modified to prevent us moving too far (see below). In terms of residuals we must distinguish between two: the residual that tells us if the intermediate equation (1) has been solved to accurately give :

and also the familiar, overall residual, which tells us if the entire non-linear problem has been solved:

The procedure for optimising s is as follows. Suppose we solve (1) for . We then start with which gives the candidate new solution, . We then check that is smaller than the previous residual discovered. If it is larger (specifically if r’ is larger) we scale s down by a factor (typically 2) and try again (ie form ). This process is repeated until a reduction in the residual has been discovered, possibly with only a modest change in **u**. The procedure is called *backtracking* and is advisable to prevent instability. Note that backtracking involves reforming the **Q** and **R** matrices for candidate vectors so can be time consuming (but we don’t need to recalculate the Jacobian during backtracking).

To implement the Newton-Raphson method, Zinc needs derivatives of the Zinc, C, a, f etc matrices. In fact we need derivatives wrt fields, etc and also derivatives wrt field *spatial derivatives*, etc. These are provided using additional callback subroutines within the DLL file called dcfun\_du, dcfun\_ddu, dffun\_du, dffun\_ddu. We only need to supply these subroutines since the problem does not have non-linearity in the Zinc **a, g, q** matrices. These subroutines are shown next:

subroutine dcfun\_du(token,x,y,z,ur,dur,nvar,istep, &

ireg,iregup,rnode,vec,imax,jmax,kmax,dCdu)

implicit none

character(\*) token

double precision dCdu(2),x,y,z,ur(2),dur(2,3)

integer nvar,istep,imax,jmax,kmax

integer ireg(0:imax,0:jmax,0:kmax),iregup(0:imax,0:jmax,0:kmax)

double precision rnode(0:imax,0:jmax,0:kmax,\*),vec(\*)

double precision sig0,sig1,T0

sig0=5.97e7

sig1=0.1\*sig0

T0=298

if (token=='$SIG') then

dCdu(1)=0

dCdu(2)=sig1

else

print \*,'DCFUN\_DU: Illegal token'

stop

endif

end subroutine dcfun\_du

subroutine dcfun\_ddu(token,x,y,z,ur,dur,nvar,istep, &

ireg,iregup,rnode,vec,imax,jmax,kmax,dCddu)

implicit none

character(\*) token

integer nvar,istep,imax,jmax,kmax

double precision dCddu(2,3),x,y,z,ur(2),dur(2,3)

integer ireg(0:imax,0:jmax,0:kmax),iregup(0:imax,0:jmax,0:kmax)

double precision rnode(0:imax,0:jmax,0:kmax,\*),vec(\*)

if (token=='$SIG') then

dCddu=0

else

print \*,'DCFUN\_DDU: Illegal token'

stop

endif

end subroutine dcfun\_ddu

subroutine dffun\_du(token,x,y,z,ur,dur,nvar,istep, &

ireg,iregup,rnode,vec,imax,jmax,kmax,dfdu)

implicit none

character(\*) token

double precision dfdu(2),x,y,z,ur(2),dur(2,3)

integer nvar,istep,imax,jmax,kmax

integer ireg(0:imax,0:jmax,0:kmax),iregup(0:imax,0:jmax,0:kmax)

double precision rnode(0:imax,0:jmax,0:kmax,\*),vec(\*)

if (token=='$Q') then

dfdu=0

else

print \*,'Illegal token'

stop

endif

end subroutine dffun\_du

subroutine dffun\_ddu(token,x,y,z,ur,dur,nvar,istep, &

ireg,iregup,rnode,vec,imax,jmax,kmax,dfddu)

implicit none

character(\*) token

integer nvar,istep,imax,jmax,kmax

double precision dfddu(2,3),x,y,z,ur(nvar),dur(2,3)

integer ireg(0:imax,0:jmax,0:kmax),iregup(0:imax,0:jmax,0:kmax)

double precision rnode(0:imax,0:jmax,0:kmax,\*),vec(\*)

double precision sig0,sig1,T0,Vx,Vy,Vz

sig0=5.97e7

sig1=0.1\*sig0

T0=298

Vx=dur(1,1) ! -dV/dx

Vy=dur(1,2)

Vz=dur(1,3)

if (token=='$Q') then

dfddu(1,1)=2\*sig0\*Vx

dfddu(1,2)=2\*sig0\*Vy

dfddu(1,3)=2\*sig0\*Vz

dfddu(2,:)=0

else

print \*,'Illegal token'

stop

endif

end subroutine dffun\_ddu

First we consider a simple non-linear system without derivatives, newton=NO. We will begin with the UMFPACK solver. The output, in the ZLS file, is as follows (the single table is split into two parts):

NL step iterations sol. error error index

1 -1 -0.10000E+01 -1

2 -1 -0.10000E+01 -1

3 -1 -0.10000E+01 -1

4 -1 -0.10000E+01 -1

5 -1 -0.10000E+01 -1

resid lhs rhs residDiag lhsDiag rhsDiag

0.66819E-15 0.11400E+06 0.11400E+06 0.11874E-14 0.12299E+04 0.12299E+04

0.10376E-14 0.13039E+06 0.13039E+06 0.11937E-14 0.12299E+04 0.12299E+04

0.10332E-14 0.13039E+06 0.13039E+06 0.11532E-14 0.12319E+04 0.12319E+04

0.98371E-15 0.13177E+06 0.13177E+06 0.11922E-14 0.12319E+04 0.12319E+04

0.96343E-15 0.13177E+06 0.13177E+06 0.11787E-14 0.12312E+04 0.12312E+04

In this case we set nstep=5 which means that 5 non-linear iterations were performed. Note that Zinc does not have a stopping criterion for the final tolerance of the non-linear residual: so all 5 iterations are always carried out. The residuals shown are the result of solving multiple linearised equations sets: we start with

so the first residual is

we then update Q according to this and then solve

so the next residual output is

etc. So the residual printed is always comparing the *next* solution to the *previous* values of the **Q** and **R** matrices. These intermediate residuals do not necessarily decrease as the non-linear solution progresses but the solution being generated should eventually stop changing and converge to a final value equal to the true solution. This implies the sequence should converge to final answer but this will not be zero. This is a problem with the simple non-linear calculation method which is why the Newton-Raphson method, discussed below, is superior despite the greater complexity involved in calculating derivatives.

The residuals output by Zinc represents the residual of the intermediate linear equation set that Zinc actually solves but it does not give an idea of how close we are to the actual non-linear solution. Such a residual can be achieved by using the new value of u *after* we have formed the new matrices **Q, R**,

This residual is not currently output by Zinc. However, if the solution converges it is clear that as since if the solution sequence converges.

So far we have used the UMFPACK direct solver. Since UMFPACK does not depend on the initial guess at the solution (and is not an iterative method) the time taken to converge each intermediate linear solution is likely to be the same. If we solved a problem that happened to be linear, the intermediate residuals would be identical.

Let us now use an iterative method to solve the intermediate linear equations. Here we use the SOR solver, with **solver=SOR**. The output in the ZLS file is

iter resid lhs rhs

0 0.21828E-03 0.11400E+06 0.11400E+06 0.22342E-02 0.12293E+04 0.12299E+04

100 0.51922E-04 0.11400E+06 0.11400E+06 0.71683E-03 0.12299E+04 0.12299E+04

200 0.12811E-04 0.11400E+06 0.11400E+06 0.17687E-03 0.12299E+04 0.12299E+04

300 0.31610E-05 0.11400E+06 0.11400E+06 0.43641E-04 0.12299E+04 0.12299E+04

400 0.77993E-06 0.11400E+06 0.11400E+06 0.10768E-04 0.12299E+04 0.12299E+04

500 0.19244E-06 0.11400E+06 0.11400E+06 0.26568E-05 0.12299E+04 0.12299E+04

600 0.47482E-07 0.11400E+06 0.11400E+06 0.65554E-06 0.12299E+04 0.12299E+04

700 0.11716E-07 0.11400E+06 0.11400E+06 0.16175E-06 0.12299E+04 0.12299E+04

800 0.28907E-08 0.11400E+06 0.11400E+06 0.39909E-07 0.12299E+04 0.12299E+04

900 0.71323E-09 0.11400E+06 0.11400E+06 0.98469E-08 0.12299E+04 0.12299E+04

1000 0.17598E-09 0.11400E+06 0.11400E+06 0.24296E-08 0.12299E+04 0.12299E+04

NL step iterations sol. error error index resid lhs rhs residDiag lhsDiag rhsDiag

1 1001 0.17598E-09 1 0.17354E-09 0.11400E+06 0.11400E+06 0.23958E-08 0.12299E+04 0.12299E+04

iter resid lhs rhs

0 0.31780E-01 0.13205E+06 0.13039E+06 0.12791E-05 0.12299E+04 0.12299E+04

100 0.26544E-03 0.13039E+06 0.13039E+06 0.62476E-08 0.12299E+04 0.12299E+04

200 0.19229E-05 0.13039E+06 0.13039E+06 0.15266E-09 0.12299E+04 0.12299E+04

300 0.13930E-07 0.13039E+06 0.13039E+06 0.35990E-10 0.12299E+04 0.12299E+04

400 0.10092E-09 0.13039E+06 0.13039E+06 0.88797E-11 0.12299E+04 0.12299E+04

500 0.74410E-12 0.13039E+06 0.13039E+06 0.21909E-11 0.12299E+04 0.12299E+04

600 0.34645E-13 0.13039E+06 0.13039E+06 0.54061E-12 0.12299E+04 0.12299E+04

NL step iterations sol. error error index resid lhs rhs residDiag lhsDiag rhsDiag

2 600 0.34645E-13 0 0.34645E-13 0.13039E+06 0.13039E+06 0.54061E-12 0.12299E+04 0.12299E+04

iter resid lhs rhs

0 0.18848E-03 0.13039E+06 0.13039E+06 0.21775E-02 0.12299E+04 0.12319E+04

100 0.13032E-04 0.13039E+06 0.13039E+06 0.20546E-03 0.12319E+04 0.12319E+04

200 0.32155E-05 0.13039E+06 0.13039E+06 0.50694E-04 0.12319E+04 0.12319E+04

300 0.79337E-06 0.13039E+06 0.13039E+06 0.12508E-04 0.12319E+04 0.12319E+04

400 0.19576E-06 0.13039E+06 0.13039E+06 0.30862E-05 0.12319E+04 0.12319E+04

500 0.48300E-07 0.13039E+06 0.13039E+06 0.76148E-06 0.12319E+04 0.12319E+04

600 0.11917E-07 0.13039E+06 0.13039E+06 0.18789E-06 0.12319E+04 0.12319E+04

700 0.29405E-08 0.13039E+06 0.13039E+06 0.46358E-07 0.12319E+04 0.12319E+04

800 0.72553E-09 0.13039E+06 0.13039E+06 0.11438E-07 0.12319E+04 0.12319E+04

900 0.17901E-09 0.13039E+06 0.13039E+06 0.28223E-08 0.12319E+04 0.12319E+04

1000 0.44169E-10 0.13039E+06 0.13039E+06 0.69636E-09 0.12319E+04 0.12319E+04

NL step iterations sol. error error index resid lhs rhs residDiag lhsDiag rhsDiag

3 1001 0.44169E-10 1 0.43556E-10 0.13039E+06 0.13039E+06 0.68668E-09 0.12319E+04 0.12319E+04

iter resid lhs rhs

0 0.82376E-02 0.13174E+06 0.13177E+06 0.31561E-06 0.12319E+04 0.12319E+04

100 0.60935E-04 0.13177E+06 0.13177E+06 0.18368E-08 0.12319E+04 0.12319E+04

200 0.40537E-06 0.13177E+06 0.13177E+06 0.61770E-10 0.12319E+04 0.12319E+04

300 0.26967E-08 0.13177E+06 0.13177E+06 0.14945E-10 0.12319E+04 0.12319E+04

400 0.17941E-10 0.13177E+06 0.13177E+06 0.36875E-11 0.12319E+04 0.12319E+04

500 0.13230E-12 0.13177E+06 0.13177E+06 0.90988E-12 0.12319E+04 0.12319E+04

NL step iterations sol. error error index resid lhs rhs residDiag lhsDiag rhsDiag

4 500 0.13230E-12 0 0.13230E-12 0.13177E+06 0.13177E+06 0.90988E-12 0.12319E+04 0.12319E+04

iter resid lhs rhs

0 0.64108E-04 0.13177E+06 0.13177E+06 0.74540E-03 0.12319E+04 0.12312E+04

100 0.29785E-05 0.13177E+06 0.13177E+06 0.47481E-04 0.12312E+04 0.12312E+04

200 0.73490E-06 0.13177E+06 0.13177E+06 0.11715E-04 0.12312E+04 0.12312E+04

300 0.18133E-06 0.13177E+06 0.13177E+06 0.28906E-05 0.12312E+04 0.12312E+04

400 0.44740E-07 0.13177E+06 0.13177E+06 0.71321E-06 0.12312E+04 0.12312E+04

500 0.11039E-07 0.13177E+06 0.13177E+06 0.17598E-06 0.12312E+04 0.12312E+04

600 0.27238E-08 0.13177E+06 0.13177E+06 0.43420E-07 0.12312E+04 0.12312E+04

700 0.67206E-09 0.13177E+06 0.13177E+06 0.10713E-07 0.12312E+04 0.12312E+04

800 0.16582E-09 0.13177E+06 0.13177E+06 0.26434E-08 0.12312E+04 0.12312E+04

900 0.40914E-10 0.13177E+06 0.13177E+06 0.65222E-09 0.12312E+04 0.12312E+04

1000 0.10095E-10 0.13177E+06 0.13177E+06 0.16093E-09 0.12312E+04 0.12312E+04

NL step iterations sol. error error index resid lhs rhs residDiag lhsDiag rhsDiag

5 1001 0.10095E-10 1 0.99548E-11 0.13177E+06 0.13177E+06 0.15869E-09 0.12312E+04 0.12312E+04

Note that the residual decreases in each iteration phase but then jumps up again as we reset the Q, R matrices based on the new solution. Fig 5 shows a plot of the residuals; we see a characteristic “saw tooth” behaviour. Note also that itmax=1000 in this simulation. During some of the linear solutions, the required convergence tol=1e-12 was achieved within this iteration limit, in others all 1000 iterations were used and the solver returns an “error index” of 1 indicating the system did not converge to the correct tolerance. In most cases this is not a problem during a non-linear simulation. Indeed it may be advisable to set tol=0 for non-linear problems as this will ensure all itmax iterations are carried out at each non-linear step. If itmax is then held at a modest value it prevents the simulation from wasting too much time on one intermediate solution. Note that the solution discovered at one non-linear iteration is used as the initial guess for the next iteration, ie, when solving

the initial guess for is .

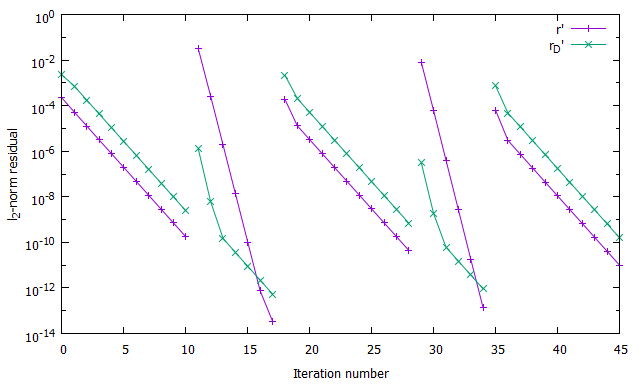


Figure 5: Convergence for standard non-linear problem without derivatives.

Next, we consider the BiConjugate Gradient method solver=BCG\_DIAG which is a packaged library, part of SLATEC:

Preconditioned BiConjugate Gradient for N, ITOL = 4719 2

ITER Error Estimate Alpha Beta

0 0.2233113D-02

1 0.9490731D-02 0.5200000D+02 NaN

2 0.8991574D-02 0.2529915D+01 0.1850000D+02

3 0.8492439D-02 0.2522523D+01 0.8974359D+00

4 0.7993329D-02 0.2514286D+01 0.8918919D+00

5 0.7494250D-02 0.2505049D+01 0.8857143D+00

6 0.6995207D-02 0.2494625D+01 0.8787879D+00

7 0.6496210D-02 0.2482759D+01 0.8709677D+00

8 0.5997271D-02 0.2469136D+01 0.8620690D+00

9 0.5498403D-02 0.2453333D+01 0.8518519D+00

10 0.4999631D-02 0.2434785D+01 0.8399999D+00

11 0.4500984D-02 0.2412697D+01 0.8260870D+00

12 0.4002509D-02 0.2385965D+01 0.8095239D+00

13 0.3504281D-02 0.2352941D+01 0.7894736D+00

14 0.3006421D-02 0.2311111D+01 0.7647059D+00

15 0.2509149D-02 0.2256411D+01 0.7333333D+00

16 0.2012901D-02 0.2181817D+01 0.6923077D+00

17 0.1518681D-02 0.2074074D+01 0.6363636D+00

18 0.1029414D-02 0.1904762D+01 0.5555556D+00

19 0.5582783D-03 0.1600000D+01 0.4285714D+00

20 0.2612767D-09 0.8888889D+00 0.2000000D+00

21 0.9590135D-10 0.1095964D+01 0.3476664D-12

22 0.6451956D-10 0.2016150D+01 0.1391868D+00

23 0.4931368D-10 0.1770792D+01 0.4584954D+00

24 0.5421550D-10 0.2027001D+01 0.5302297D+00

25 0.4689674D-10 0.4436352D+01 0.8355029D+00

26 0.2568557D-10 0.1518882D+01 0.1174964D+01

27 0.2007639D-10 0.2210885D+01 0.2703631D+00

28 0.1521307D-10 0.1879050D+01 0.6459013D+00

29 0.9290711D-11 0.2115763D+01 0.4870605D+00

30 0.4795731D-11 0.1512262D+01 0.4125372D+00

31 0.3649369D-11 0.2082006D+01 0.2821703D+00

32 0.2401951D-11 0.1790819D+01 0.5988427D+00

33 0.1449409D-11 0.1902119D+01 0.4097709D+00

34 0.8528473D-12 0.1582362D+01 0.3713232D+00

NL step iterations sol. error error index resid lhs rhs residDiag lhsDiag rhsDiag

1 34 0.85285E-12 0 0.81597E-13 0.11400E+06 0.11400E+06 0.85282E-12 0.12299E+04 0.12299E+04

Preconditioned BiConjugate Gradient for N, ITOL = 4719 2

ITER Error Estimate Alpha Beta

0 0.1279110D-05

1 0.2977192D-05 0.1105867D+02 NaN

2 0.2626076D-05 0.2583047D+01 0.4148319D+01

3 0.2304408D-05 0.2520827D+01 0.9300682D+00

4 0.2018337D-05 0.2484373D+01 0.8889472D+00

5 0.1763715D-05 0.2456252D+01 0.8634239D+00

6 0.1535548D-05 0.2430716D+01 0.8431359D+00

7 0.1329625D-05 0.2405140D+01 0.8244609D+00

8 0.1142678D-05 0.2377930D+01 0.8056672D+00

9 0.9722472D-06 0.2347803D+01 0.7856644D+00

10 0.8165338D-06 0.2313470D+01 0.7635608D+00

11 0.6742717D-06 0.2273404D+01 0.7384582D+00

12 0.5446465D-06 0.2225652D+01 0.7093063D+00

13 0.4272401D-06 0.2167487D+01 0.6747702D+00

14 0.3220043D-06 0.2094963D+01 0.6330425D+00

15 0.2292580D-06 0.2002062D+01 0.5815779D+00

16 0.1497044D-06 0.1879132D+01 0.5166697D+00

17 0.8446206D-07 0.1709724D+01 0.4327960D+00

18 0.3506406D-07 0.1463703D+01 0.3217974D+00

19 0.9364222D-13 0.1081138D+01 0.1735651D+00

NL step iterations sol. error error index resid lhs rhs residDiag lhsDiag rhsDiag

2 19 0.93642E-13 0 0.76158E-09 0.13039E+06 0.13039E+06 0.93745E-13 0.12299E+04 0.12299E+04

Preconditioned BiConjugate Gradient for N, ITOL = 4719 2

ITER Error Estimate Alpha Beta

0 0.2173911D-02

1 0.2386226D-02 0.3591748D+01 NaN

2 0.2168983D-02 0.2414033D+01 0.1206733D+01

3 0.2021931D-02 0.2471657D+01 0.8251721D+00

4 0.1931439D-02 0.2533540D+01 0.8681965D+00

5 0.1882866D-02 0.2588364D+01 0.9120877D+00

6 0.1863548D-02 0.2631757D+01 0.9504042D+00

7 0.1862260D-02 0.2661338D+01 0.9801047D+00

8 0.1868923D-02 0.2676495D+01 0.9995033D+00

9 0.1874909D-02 0.2678482D+01 0.1008311D+01

10 0.1873927D-02 0.2670374D+01 0.1007726D+01

11 0.1863224D-02 0.2656868D+01 0.1000366D+01

12 0.1844816D-02 0.2644152D+01 0.9900690D+00

13 0.1826506D-02 0.2639796D+01 0.9817408D+00

14 0.1822824D-02 0.2652953D+01 0.9813017D+00

15 0.1856460D-02 0.2694531D+01 0.9958880D+00

16 0.1960680D-02 0.2775849D+01 0.1034007D+01

17 0.2178246D-02 0.2897927D+01 0.1103854D+01

18 0.2511171D-02 0.2990199D+01 0.1199914D+01

19 0.2551756D-02 0.2635928D+01 0.1224032D+01

20 0.1082112D-08 0.1043818D+01 0.6807626D+00

21 0.2109793D-09 0.9350475D+00 0.2545738D-12

22 0.1369769D-09 0.1648595D+01 0.3806508D-01

23 0.7135678D-10 0.1487761D+01 0.3789902D+00

24 0.4249068D-10 0.1816125D+01 0.3117139D+00

25 0.2810072D-10 0.1854589D+01 0.3848209D+00

26 0.1886263D-10 0.1874017D+01 0.4315305D+00

27 0.1329988D-10 0.1901046D+01 0.4557755D+00

28 0.1008810D-10 0.2110771D+01 0.4963993D+00

29 0.8046301D-11 0.2073870D+01 0.5820996D+00

30 0.6360932D-11 0.2288936D+01 0.6225197D+00

31 0.5170117D-11 0.2165992D+01 0.6391864D+00

32 0.4087789D-11 0.2128975D+01 0.6581140D+00

33 0.2954221D-11 0.2113346D+01 0.6094203D+00

34 0.2121660D-11 0.2036125D+01 0.5486583D+00

35 0.1518518D-11 0.1944632D+01 0.5331905D+00

36 0.1000390D-11 0.1862276D+01 0.5002158D+00

37 0.6270692D-12 0.1782158D+01 0.4329288D+00

NL step iterations sol. error error index resid lhs rhs residDiag lhsDiag rhsDiag

3 37 0.62707E-12 0 0.13193E-11 0.13039E+06 0.13039E+06 0.62698E-12 0.12319E+04 0.12319E+04

Preconditioned BiConjugate Gradient for N, ITOL = 4719 2

ITER Error Estimate Alpha Beta

0 0.3156100D-06

1 0.1200305D-05 0.2088224D+02 NaN

2 0.6010416D-06 0.1871468D+01 0.1028523D+02

3 0.5094653D-06 0.1865973D+01 0.3278583D+00

4 0.6285782D-06 0.3606383D+01 0.7137348D+00

5 0.4620873D-06 0.2421506D+01 0.1623754D+01

6 0.3650475D-06 0.1947947D+01 0.6265984D+00

7 0.3910228D-06 0.2824395D+01 0.6249143D+00

8 0.3181886D-06 0.2537427D+01 0.1161214D+01

9 0.2366189D-06 0.1969025D+01 0.7320333D+00

10 0.2231205D-06 0.2399143D+01 0.5620310D+00

11 0.1821544D-06 0.2446644D+01 0.8740712D+00

12 0.1228982D-06 0.1912655D+01 0.7106736D+00

13 0.9510520D-07 0.1996699D+01 0.4700685D+00

14 0.6892010D-07 0.2117304D+01 0.5821478D+00

15 0.3688628D-07 0.1700867D+01 0.5373588D+00

16 0.1688050D-07 0.1454214D+01 0.2986519D+00

17 0.6456483D-08 0.1328917D+01 0.2090500D+00

18 0.2423020D-08 0.1139835D+01 0.1377125D+00

19 0.1382004D-12 0.2475616D+01 0.1224107D+00

NL step iterations sol. error error index resid lhs rhs residDiag lhsDiag rhsDiag

4 19 0.13820E-12 0 0.17326E-09 0.13177E+06 0.13177E+06 0.13826E-12 0.12319E+04 0.12319E+04

Preconditioned BiConjugate Gradient for N, ITOL = 4719 2

ITER Error Estimate Alpha Beta

0 0.7458250D-03

1 0.8022823D-03 0.3739507D+01 NaN

2 0.6922990D-03 0.2297322D+01 0.1158006D+01

3 0.6118718D-03 0.2350855D+01 0.7435702D+00

4 0.5527296D-03 0.2400999D+01 0.7800115D+00

5 0.5092189D-03 0.2448137D+01 0.8149091D+00

6 0.4771850D-03 0.2490766D+01 0.8477879D+00

7 0.4534301D-03 0.2527239D+01 0.8774487D+00

8 0.4354014D-03 0.2556107D+01 0.9026017D+00

9 0.4210711D-03 0.2576719D+01 0.9221935D+00

10 0.4089748D-03 0.2589857D+01 0.9358800D+00

11 0.3983875D-03 0.2598314D+01 0.9445142D+00

12 0.3896299D-03 0.2607575D+01 0.9505913D+00

13 0.3845178D-03 0.2626604D+01 0.9587598D+00

14 0.3870815D-03 0.2669213D+01 0.9765082D+00

15 0.4049384D-03 0.2756045D+01 0.1015389D+01

16 0.4522067D-03 0.2916117D+01 0.1092955D+01

17 0.5546070D-03 0.3177342D+01 0.1234078D+01

18 0.7384807D-03 0.3463200D+01 0.1456976D+01

19 0.8024023D-03 0.2996415D+01 0.1623594D+01

20 0.3449203D-09 0.1030417D+01 0.7951821D+00

21 0.7219887D-10 0.9456638D+00 0.2564500D-12

22 0.4691262D-10 0.1728911D+01 0.4382887D-01

23 0.2514137D-10 0.1481878D+01 0.3909738D+00

24 0.1469890D-10 0.1807337D+01 0.3189138D+00

25 0.9492633D-11 0.1835421D+01 0.3778551D+00

26 0.6139931D-11 0.1821778D+01 0.4119543D+00

27 0.4148561D-11 0.1833522D+01 0.4216128D+00

28 0.2905457D-11 0.1988561D+01 0.4505633D+00

29 0.2147072D-11 0.1938167D+01 0.4983196D+00

30 0.1611853D-11 0.2153607D+01 0.5351552D+00

31 0.1292698D-11 0.2131975D+01 0.5719929D+00

32 0.1056943D-11 0.2166582D+01 0.6453101D+00

33 0.8155953D-12 0.2225844D+01 0.6516269D+00

NL step iterations sol. error error index resid lhs rhs residDiag lhsDiag rhsDiag

5 33 0.81560E-12 0 0.92378E-12 0.13177E+06 0.13177E+06 0.81566E-12 0.12312E+04 0.12312E+04

Here, we are using the diagonal scaled residual, (residDiag) as the error estimate. The situation is similar to the SOR simulation with the residual decreasing during each linear solution phase then jumping up again as we reformulate Q and R.

Let us now consider the Newton-Raphson method for solving non-linear problems. We set newton=YES in the input file and revert to using the UMFPACK method. The result is shown in Fig 6 for 5 non-linear steps.

As described earlier, there are two residuals to consider, the residual due to solving the linear equation which yields the step and the overall residual, **r**. There are therefore 2 rows in the table for each iteration step, the first shows the overall residual and the second the residual. The former residual does not show a systematic decrease from one step to the next. The overall residual does decrease at a steady rate showing good convergence to the overall solution. Considering the overall residual, the standard residual resid= first increases and then decreases while the diagonally scalled residual, residDiag= decreases monotonically. This is likely because the fixed degrees of freedom (Dirichlet boundaries) are not explicitly fixed in the Newton-Raphson method and any drift of these values from their fixed values will have a huge effect on r’ owing to the different units involved. The diagonally-scaled residual cancels out these different units so that errors at Dirichlet boundaries do not have an excessive effect on the residual. (see Appendix for further discussion)

Figure 6: ZLS file output for Newton-Raphson non-linear solution

NL step iterations sol. error error index resid lhs rhs residDiag lhsDiag rhsDiag

OVERALL RESIDUAL 0.75911E-02 0.32780E+04 0.32781E+04 0.83784E-03 0.32780E+04 0.32780E+04

1 0 -0.10000E+01 -1 0.79103E-12 0.24884E+02 0.24884E+02 0.25666E-13 0.27464E+01 0.27464E+01

OVERALL RESIDUAL 0.50125E+00 0.37884E+04 0.32782E+04 0.22674E-03 0.32780E+04 0.32780E+04

2 0 -0.10000E+01 -1 0.82805E-14 0.18989E+04 0.18989E+04 0.27504E-13 0.74326E+00 0.74326E+00

OVERALL RESIDUAL 0.48140E-01 0.32819E+04 0.32781E+04 0.19150E-04 0.32780E+04 0.32780E+04

3 0 -0.10000E+01 -1 0.10920E-13 0.15799E+03 0.15799E+03 0.15186E-13 0.62774E-01 0.62774E-01

OVERALL RESIDUAL 0.30148E-03 0.32782E+04 0.32782E+04 0.16483E-06 0.32780E+04 0.32780E+04

4 0 -0.10000E+01 -1 0.45219E-14 0.98831E+00 0.98831E+00 0.38202E-13 0.54030E-03 0.54030E-03

OVERALL RESIDUAL 0.23845E-07 0.32782E+04 0.32782E+04 0.24674E-11 0.32780E+04 0.32780E+04

5 0 -0.10000E+01 -1 0.68864E-14 0.78169E-04 0.78169E-04 0.16367E-13 0.80881E-08 0.80881E-08

It should be noted that in the Newton-Raphson method, Zinc represents Dirichlet boundaries as explicit equations which appear in the FE matrix **Q**. For “standard” non-linearity and linear problems, these fixed degrees of freedom are eliminated from the matrix when it is formed if removeFixed=YES (the default). But this feature is not currently supported in the Newton-Raphson method as it makes formation of the Jacobian complicated (removeFixed is ignored for newton=YES). In future we hope to eliminate fixed degrees of freedom when using Newton-Raphson non-linear solutions also.

Figures 7 and 8 show the simulation result after 5 non-linear steps for both the standard and Newton-Raphson method. A Comsol simulation is shown for comparison and we see that the Newton-Raphson method gives a more accurate answer.

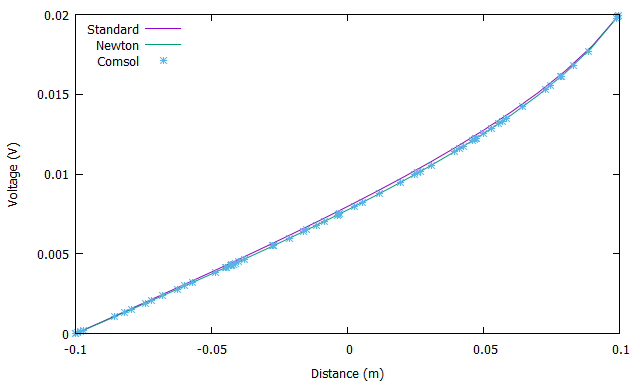


Figure 7: Simulation result. Voltage variation across the sample.

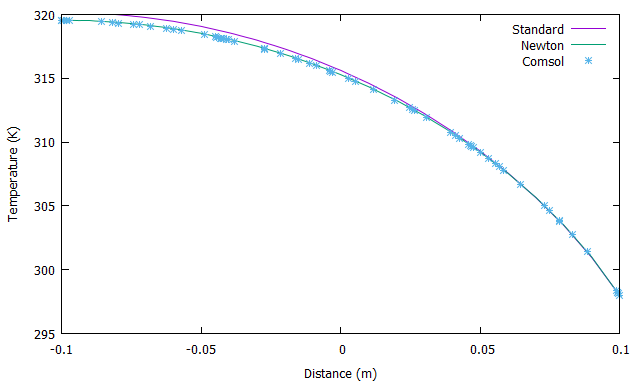
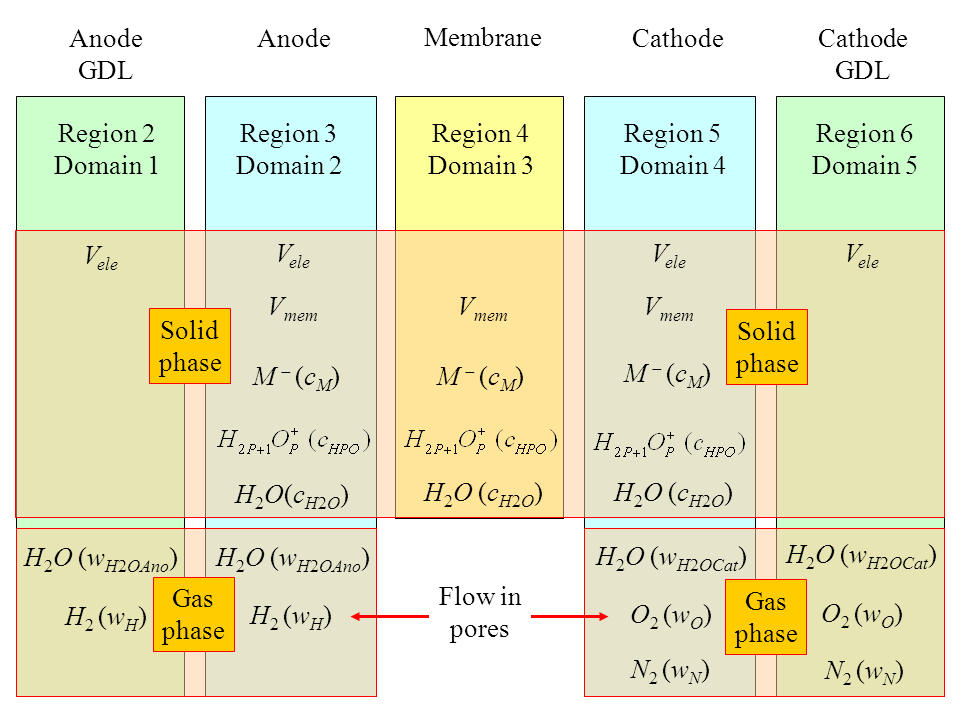


Figure 8: Simulation result. Temperature variation across sample.

# Fuel cell problem



p

p

p

p

Figure 9: Schematic diagram showing various species (with notation) that need to be modelled, depending on the phase and region within the fuel cell. We will use the terms “Domain” and “Region” interchangeably in the following text.



Figure 10. Fuel cell as meshed in Zinc. Region numbers are as in Fig 9 “Domains”. Regions 2 (anode) and 4 (cathode) are very thin on either side of Region 3. Regions 6 and 7 are “void” regions (see text). The black line is the scan line through the simulation plotted in Figures 15-17.



fixed

fixed

fixed

fixed

Figure 11: Distorted view (3x magnified in the x-direction). Regions 1 and 5 are not shown. Regions 2 and 4 are easier to see in this figure. External Dirichlet boundaries (fuel cell “ports”) are indictated as follows:- **blue**: fixed; **green**: fixed; **yellow**: fixed; **red:** fixed.

## Theoretical background

We now consider a more realistic, complex simulation: a fuel cell problem shown schematically in Fig 9. This system involves multiple dependent variables: two voltage fields, pressure and material densities (the latter sometimes expressed as molar or mass fractions). The problem is also highly non-linear with much of the complexity arising from the “right hand side” source terms based on an agglomerate model as will be described below. In terms of residuals, however, the situation is very similar to the previous section. We will use the UMFPACK solver and the NEWTON non-linear method (since the problem is highly non-linear).

Figure 10 shows the simulation as meshed in Zinc. The simulation is really 2D, but Zinc currently only handles 3D so a single layer of finite elements is simulated (nothing happens in the z-direction). Figure 11 is the same image but distorted with some nodes shown colour coded to indicate Dirichlet boundary conditions that have been applied: at the 4 simulation ports, concentration of various species and pressure is fixed; at the left and right of the simulation the voltage is fixed, as shown.

Note that the fuel cell has an irregular outer perimeter. It is in the form of a letter H on its side. However, all Zinc simulations must have an outer boundary which is a cuboid. This is the case because Zinc uses a structured mesh. To overcome this problem we introduce additional “void” regions 6 and 7 which have materials properties (ie C, A, F matrices) set to zero. Note that these void regions have elements in them which are wasted, (not used in the simulation). This is unfortunate, but it is a consequence of having a structured mesh, which has other advantages.

The dependent variables to be solved in the problem are:

 Electrical potential in solid conducting phase

 Electrical potential in membrane phase

 Molar concentration of negative ions in membrane material

 Molar concentration of complex positive ions (hydronium) in membrane material. In practice P=3 so that the molecule is H3O

 Molar concentration of water in membrane material

Mass fraction of hydrogen molecules in the gas phase at anode

Mass fraction of water in the gas phase at anode

Mass fraction of oxygen molecules in the gas phase at the cathode

Mass fraction of water in the gas phase at cathode

Mass fraction of nitrogen molecules in the gas phase at the cathode

p pressure

The mass fractions must add up to 1 on each side of the fuel cell (anode and cathode sides). Therefore some of these variables can be eliminated as follows:

Furthermore, the negative ion concentration can be eliminated using the principle of charge neutrality:



Where and are the charge of the membrane and positive ions complex expressed in units of electron charge. This leave 8 dependent variables to solve:

The mass fractions described above can be converted to molar fractions and densities as follows:

 (2)

Where are the molar masses, are the molar fractions, are the partial densities, m is the effective mass (defined thus) and is the total density. Note that the total density cannot be calculated if we only know the full set of molar or mass fractions and must be provided separately. In the simulation, this is calculated from the pressure as follows:

 (anode)

 (cathode)

In regions 1, 2, the variation of mass fractions is given by the Maxwell-Stephan equation:

 with

Where are diffusion coefficients described below and **u** is the velocity field given by ( are the porosity and viscocity). In regions 4, 5 (cathode) the Maxwell-Stephan equations are





With



The source terms, R, will be described below.

In region 1,2,4 and 5, Darcy’s law is specifies the variation of pressure, p while Poissons equation specifies the voltage variation, :





Where are the porosity and viscosity and is the conductivity (all in SI units). The source terms are described below. In the membrane (regions 2,3,4) the Nernst Planck equation is used to specify the concentrations of hydronium and water:

, k=1,2,3

Where k=1,2,3 map to respectively, T is the temperature, R is the gas constant, F is Faraday’s constant (see below for source term ).

The source terms for Maxwell Stephan equations are

, (region 2)

, (region 4)

, (region 4)

(see below for the S-terms). The source terms (and porosity, viscosity) for the Darcy law are



The source terms for Poisson’s law are:

(region 2)

(region 4)

The source term for Nernst-Planck are

The S-parameters above are given by

So that all the source terms are ultimately specified in terms of and which are given by:





With





where





Various concentration expressions above are given by simple formulae:





Where the K factors are Henry’s constants which are functions only of temperature. The remaining parameters above ( and equivalent for cat) are either constants or simple functions of temperature. Since temperature is constant and uniform in the simulation (T=353 K) these also evaluate to contants. We refer the reader to the simulation for specification of these parameters (see in particular fullfuelmod.f90). In fact over 100 parameters are needed to specify the fuel cell simulation!

Finally, the diffusion expressions in the Maxwell-Stephan equations are given by:

 (anode)



 , (cathode)

 ,

 ,

 , (cathode)



Where the values are functions only of temperature. Since this is constant, they evaluate to constants.

## Activation of physics in different regions

The dependent variables,

are not all active in each region. The regions where each is active is shown in Fig 9. For example is active in regions 1, 2, 4, 5 but not in region 3 and is active in regions 2, 3, 4 only. The variable is active only in regions 1 and 2. To achieve this in Zinc, it is necessary to clamp the redundant degrees of freedom to zero at nodes in the corresponding regions. So, for example, in region 5, must be set to zero, otherwise the simulation will not converge as Zinc has no equation for in region 5. If you look at the fullfuel.zin file, you will see many “region..node” commands clamping redundant degrees of freedom in this way. In addition “region..node” commands are used to implement the Dirichlet boundary conditions described in Fig 11.

Region 2

Region 3

Figure 12: Introducing internal boundary conditions between region 2 and 3 for which is deactivated in region 3. Nodes which have (ie clamped) are shown black; is free to vary in white nodes. In (a) a Dirichlet boundary condition is implied at the thick line; in (b) a Neumann BC is applied.

Since some of the degrees of freedom are deactivated in certain regions we need to introduce *internal* boundary conditions. These can either be Dirichlet or Neumann internal boundaries and the precise arrangement of fixed nodes will determine which. Consider Fig 12 which is a zoom in of regions 2 and 3 from Fig 11. Variable is active in region 2 (white) but deactivated in region 3 (black). This means that nodes *fully within* region 3 must be clamped (black), otherwise their values will be undefined. It’s conventional to fix them at zero, but any value will do. But that still leaves the nodes *on the interface* between regions 2 and 3: we can either clamp these or leave them free (as far as is concerned). In Fig 12(a), we clamp on the interface nodes which implies a Dirichlet boundary will be applied at the interface. So as we scan across region 2, will decrease to zero as we approach the interface. The other option is shown in Fig 12(b): the interface nodes are not clamped but the material property corresponding to , namely , is set to zero in region 3. In this case Zinc will set the value of on the interface nodes so that at the boundary so therefore region 3 becomes a *void region* with material property zero.

Note that in Fig 12(a), it doesn’t matter what value we set in region 3, as these elements are surrounded by clamped nodes so the material properties cannot affect the solution. In fact, region 3 in Fig 12(a) is acting as an *electrode* with fixed potential of 0 Volts.

What we actually want in the fuel cell model is the setup of Fig 12(b) so it is important not to clamp the nodes at the Region 2-3 interface (but conductivity must be set to zero in region 3). All of this applies only to the degree of freedom and other variables might be clamped/not clamped on the same nodes. Thus, the fuel cell simulation involves a complex pattern of fixed nodes, due to the deactivation of degrees of freedom in certain regions (we can say that some physics equations apply only to certain regions). Setting up this pattern can be difficult to get right. Fortunately Zinc provides a facility to show the nature of nodes in a diagram. If you set nodecheck=YES in the ZIN file, instead of writing the output solution, Zinc writes a ZOU file where each degree of freedom is given a value specifying the type of boundary condition or constraint at that node. The values are: 1 deactivated (ie fixed at zero); 2 fixed at a non-zero value; 3 external Neumann boundary; 4 internal Neumann boundary (bordering a void region); 5 continuity (between two active regions); 6 variable (in the bulk). This file can be read into Zmesh in the usual way and plotted per degree of freedom. Note that the actual simulation solution is not output so you should run for zero iteration steps.

Figure 13 shows the node check result for the variable. Nodes are numbered 1 to 6 with the above-mentioned scheme (see also caption). The nodes which are at the interface between regions 2 and 3 are designated “internal Neumann boundary” indicating a boundary condition will be applied there as required. On the other hand, the nodes shown light blue on the right of region 5 are fixed at (Dirichlet boundary). So these nodes act as an electrode with Volts. Another electrode exists at the left of Region 1 with . The fuel cell “ports” (as indicated in the Figure) have Dirichlet boundaries .

Figure 14 shows a similar nodecheck picture for the pressure variable. The result is similar to Fig 13 but note the nodes at the right border of region 5 (magenta squares) are green which indicates there, so a Neumann boundary is applied rather than the Dirichlet one for at the same place. But the nodes for the ports of the fuel cell (shown as “port”) are now shown light blue indicating a Dirichlet boundary p=const is applied there.



port

port

port

port

Figure 13: Using nodecheck=yes to indicate the nature of each node. Here we show the specification of nodes for variable . Dark blue: (Dirichlet); light blue: (Dirichlet); orange: is free to vary and elements on either side have different material properties (continuity node); red: is free to vary and all elements around the node have the same material property (bulk); green: free to vary and elements on one side have zero material properties (bordering void region; Neumann boundary).



port

port

port

port

Figure 14: Node check for pressure. Dark blue: (Dirichlet); light blue: (Dirichlet); orange: is free to vary and elements on either side have different material properties (continuity node); red: is free to vary and all elements around the node have the same material property (bulk); green: free to vary and elements on one side have zero material properties (bordering void region; Neumann boundary).

## Zinc matrices

Let us now consider the formulation of the Zinc, C, F matrices (the A matrix is zero throughout). First it should be noted that the Maxwell-Stephan equations:



(here shown for the anode) depend on both the molar fractions and the mass fractions . The two sets of variables can be related to each other according to (2). We must eliminate either the molar or mass fractions to end up with just one set of variables. As stated above we eventually end up setting the equations in terms the mass fractions However, in an earlier simulation, we used the molar fractions, x, instead. Since the molar fractions are easier to work with, we will start with these and then move to mass fractions. Considering molar fractions, then, the dependent variables can be grouped into a vector **u** as:



We wish to express the PDEs required in the form

.

The c-matrix has the form



with







where *I* is the 3x3 identity matrix (the entire simulation is isotropic). The numbers in brackets after each equation show which regions the component is active in (else it must be zero as discussed). In other domains, the matrix component should be set to zero. *Note, in the variables DM and uM above are referred as D0=1e-8 and u0=0 in the model*. Also in the model terminology, the DH3O and DH2O parameters are given suffices \_2, \_3, \_4 depending on region and uM=DH3O/RT. We use zH3O=1 and zM= -1 throughout. The *f*-vector is given by



where again the numbers in brackets indicate the domains where the component applies. (Only the grad(ω) term applies in regions marked \*). Outside of these domains, the component is zero. We have

The parameters appear in the model as SH3O\_2, SH3O\_4 and SH2O\_2, SH2O\_4.

## Formulation in terms of mass fractions

In the above description, we have used molar fractions (*x*) as the dependent variable within the Maxwell-Stefan equations. This was done because terms in the gradient of molar fractions appear directly in the equations. However, the existing COMSOL model uses mass fractions, *ω*, in the formulation and this may give rise to different Neumann boundary conditions so that the two formulations are not equivalent. We now prepare a PDE formulation in terms of mass fractions.

Molar fractions, *x*, can be written in terms of mass fractions, ω, as follows. Firstly, on the anode side we have:



On the cathode side we have:



It is clear that xH(ωH) while xO(ωO, ωH2O) and xH2O(ωO, ωH2O). Note that ωH2O and xH2O refer to the water mass/molar fractions on the *cathode* side. In the COMSOL/Zinc model these are referred to as wH2OCat and xH2OCat respectively. We can now define the gradients of xH, xO and xH2O as:

 (3)

with



which can be expressed in the following more compact form



with



Let us write out the Maxwell-Stefan equations explicitly:



Eliminating xH2O, ωH2O (anode) and xN, ωN (cathode) gives



(4)

Substituting from (3) gives



and we obtain the c-matrix



Note that the remaining components in this matrix are unchanged with the substitution . Eg etc. This completes the formulation of the fuel cell problem in Zinc. The main simulation file is called fullfuel.zin. It links to a DLL fullfuel.dll which is created from two source code files, fullfuel.f90 and fullfuelmod.f90 (use compile.bat to recreate the DLL). These source files contain the cfun and ffun functions as specified above. Since the Newton option is used, derivatives of the C and F matrices are also also provided using subroutines dcfun\_du, dcfun\_ddu, dffun\_du and dffun\_ddu. The fuel cell simulation is available in the Zinc source code repository at (<https://svn.npl.co.uk/repos/MaterialsModelling/zinc>).

## Results

The simulation was run for 5 non-linear time steps in all using the Newton method. The formation of the Jacobian was very slow so several simulations were run which followed from each other using the Zinc RESTART command. In future it will be necessary to make the formation of the Jacobian in Zinc more efficient, perhaps by taking advantage of symmetry or isotropy (note that this entire problem is isotropic). However, the results are very accurate: figures 15-17 show a comparison between Zinc and Comsol results for the mass fractions of oxygen and water, and variation of pressure on the cathode side of the fuel cell. The line scan was done along the line shown in Figure 10. Note that the molar fractions of oxygen and water are essentially fixed (Dirichlet) at y=2mm but free to vary at y=0 mm (Neumann), see Figure 11. The pressure is fixed at both ends and thus easier to simulate accurately.

Figure 18 shows the decrease in residuals as the simulation progresses. As before, residDiag is the most reliable measure of residual and this decreases monotomically from one Newton step to another. The decrease is roughly one order of magnitude per Newton step which is very satisfactory.

Finally Figures 19-26 show the residual map across the system for each of the dependent variables. In each case the residual is much less than the right and left-hand side maps (not shown) which indicates good convergence.

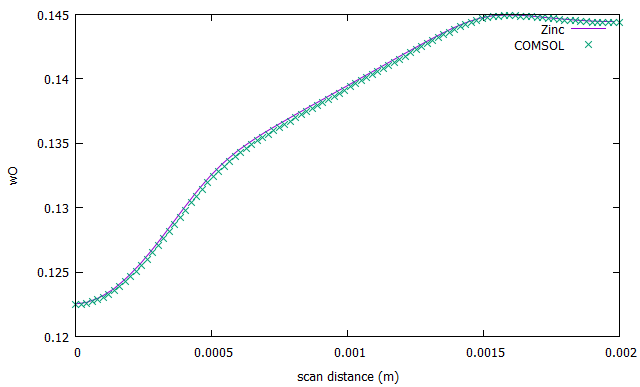


Figure 15: Variation of oxygen mass fraction in the cathode side of the fuel cell.

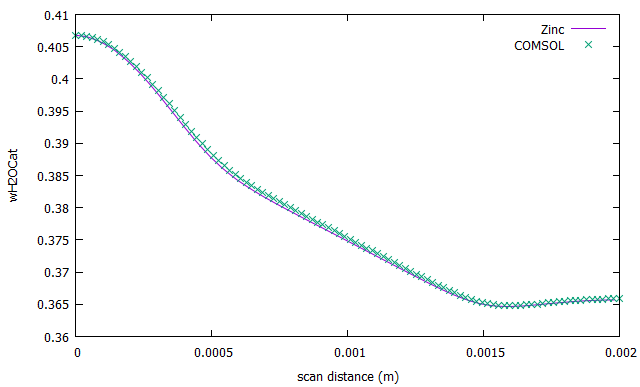


Figure 16: Variation of water mass fraction in the cathode side of the fuel cell.

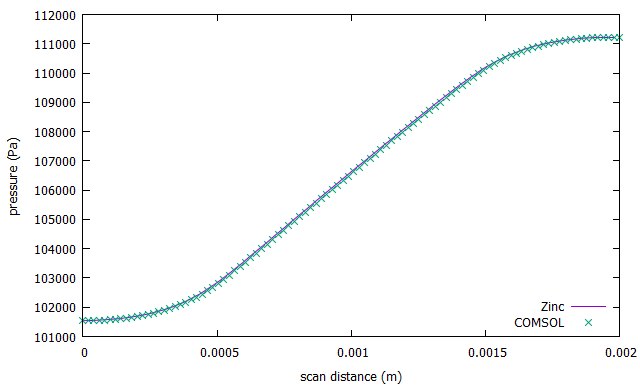


Figure 17: Variation of pressure in the cathode side of the fuel cell.

Figure 18: Residuals for fuel cell problem during non linear convergence

NL step iterations sol. error error index resid lhs rhs residDiag lhsDiag rhsDiag

OVERALL RESIDUAL 0.56439E-11 0.14127E+07 0.14127E+07 0.54898E-01 0.14148E+07 0.14127E+07

1 0 -0.10000E+01 -1 0.14211E-13 0.79730E-05 0.79730E-05 0.58683E-15 0.79037E+05 0.79037E+05

OVERALL RESIDUAL 0.36127E-11 0.14127E+07 0.14127E+07 0.27084E-02 0.14127E+07 0.14127E+07

2 0 -0.10000E+01 -1 0.18746E-13 0.51037E-05 0.51037E-05 0.56611E-15 0.38225E+04 0.38225E+04

OVERALL RESIDUAL 0.55727E-11 0.14127E+07 0.14127E+07 0.10549E-04 0.14127E+07 0.14127E+07

1 0 -0.10000E+01 -1 0.16462E-13 0.78725E-05 0.78725E-05 0.56573E-14 0.14359E+02 0.14359E+02

OVERALL RESIDUAL 0.50526E-11 0.14127E+07 0.14127E+07 0.61659E-05 0.14127E+07 0.14127E+07

2 0 -0.10000E+01 -1 0.12536E-13 0.71378E-05 0.71378E-05 0.10694E-13 0.78897E+01 0.78897E+01

OVERALL RESIDUAL 0.32285E-10 0.14127E+07 0.14127E+07 0.25482E-04 0.14127E+07 0.14127E+07

# 

Figure 19: Residual map for



Figure 20: Residual map for pressure.



Figure 21: Residual map of .



Figure 22: Residual map of



Figure 23: Residual map of .



Figure 24: Residual map for



Figure 25: Residual map for .



Figure 26: Residual map for

# Conclusions

A linear finite element problem can be reduced to a set of linear equations. The accuracy to which these equations is solved is given by the residual which is a vector quantity. To formulate this into a scalar quantity a simple *l*2-norm is taken with additional scaling so that the quantity is dimensionless and in the range 0 to 1. A further subtlety is to diagonally scale the residual which removes the distorting effect of large changes in material properties and fixed degrees of freedom in the residual. The diagonally scaled residual is therefore preferable to the regular one and seems to have no disadvantages so its use is recommended. Normally both types of residuals decrease in a similar way (ie both might decrease over 10 orders of magnitude as the simulation converges, although the absolute value of the residuals will be different).

For non-linear problems, Zinc supports two solution methods: a direct method and the Newton-Raphson method. The latter is more complex in that derivative quantities are required but, experience shows, tends to give better solutions. In terms of residuals the direct method has only one type of residual but this tends to cycle up and down according to a saw-tooth curve which is not very satisfactory in deciding whether a problem has converged (and often fails to converge anyway!). The Newton-Raphson method yields two residuals: the overall residual which dictates if the entire problem has been solved and an intermediate residual showing how well we have discovered a direction in phase space to traverse. The latter residual need not decrease systematically but the overall residual should (and does, in practice) decrease monotonically (taking the diagonally-scaled version) which is very satisfactory. Thus the Newton-Raphson method is superior in all ways which explains why it has become the industry standard in most FE packages including Comsol and Abacus.

In this report, several real life example problems have been shown of increasing complexity, building up to a fully coupled, non-linear fuel cell problem. In each case the residual and how it reduces as the simulation converges was shown along with residual variation maps across the simulation.

# Appendix: Example of residuals for a specific matrix

Consider the linear equations corresponding to a simple 1D electrostatic, finite difference problem with N=4 internal nodes:

where are the voltages at each internal node, ϵ is the constant permittivity, is the length of each element and is the fixed voltage at each end of the simulation (Dirichlet boundary condition). The space charge has been set to zero. We could, of course, divide each side of the equation by but for now we leave these factors in as this is how Zinc will solve the problem and in general these parameters vary in space. Of course the actual Zinc Q-matrix will be more complex than this, but this example gives an idea of magnitudes.

It is easy to see how this matrix would extend to the case : the matrix remains triple diagonal while the rhs vector has for the first and last elements and other elements are zero.

It is clear that the solution to this problem is,

If we use this solution, the residual will be clearly be zero indicating an exact solution. However, let us imagine we get a trial solution which is somewhat displaced from the exact value:

where is the error of the solution at each node, usually small compared to . In that case we obtain

and

so that (generalising to matrix size NxN),

We see that |r| increases with the number of nodes, N. In the limit it becomes

The problem with this parameter is it gets larger with increasing N. We would rather have a parameter which depends only on the error of the solution at each node, namely . Also, |r| is not dimensionless and, in this case, has dimensions of permittivity voltage length.

We therefore introduce

The norm of the left hand side is given by

So that

The denominator is clearly always greater than the numerator. When and when . For well-converged solutions we can interpret r’ as the average fractional error of the solution at each node (up to a dimensionless multiplicative factor). Also, r’ has the benefit of being dimensionless. Since r’ is comparable to 1 it is easy to see how well the solution has converged. Note that would not be a good convergence parameter as does not depend on the solution error per node, so this fraction is unbounded as and it is difficult to decide “how small is small”.

This proves that r’ is a good scalar measure of how well the solution has converged. Of course it does not tell us everything about the full residual vector, **r**, which might vary spatially, for example.

Let us now consider a Multiphysics case in which there are 2 variables to solve for at each node. We will consider a problem with electrostatic and elastic effects. The FE equations are still in the form but this time **u** is partitioned as follows:

where are the elastic displacements in metres and the are voltages are before (here we have N=4 but we will extend to arbitrary N later. The size of the vector is 2N). Note that the matrix elements are in different units, volts and metres.

We will assume the same finite difference problem as before for the electrostatic and elastic parts separately with no coupling between the two. In that case

where the block matrices and are given by

and

with C the stiffness matrix in . The right hand side vector is given by

with the applied voltage at each end of the system. It is obvious that the solution of these equations is given by

Let us again consider a perturbation on the exact result

We obtain the following residual, generalising to N nodes (2N degrees of freedom),

Note that the two terms in under the square root are in different units so adding them up in this way may be problematic. If the voltage is of order unity, say, and displacement is of order m then the errors δ and λ will be in these units also so . On the other hand the material properties ϵ and C are in different units as well so this may offset the issue to some extent. Nevertheless there is a danger that degrees of freedom in the residual will be over- or under-represented in the residual magnitude.

The problem arises because the individual equations

are in different units for the electrostatic and elastic halves of the problem. For the electrostatic, [u] is in V.m, [Q] is in F/m and [R] is in F.V. For the piezoelectric, [u] is in m, [Q] is in N/m and [R] is in N. So summing the residuals of all these equations gives a sum-of-squares containing a mixture of units. Normally in physics it is wrong to add up two terms with different units but the residual is not an observable quantity like momentum so it is allowed to do it – but not ideal!

In this case, the scaled residual is given by,

which as before is a dimensionless number between 0 and 1 which tends to 0 as . But this does not overcome the problem of different units within the summation in the numerator and denominator.

Finally, the diagonal-scaled residual magnitude, for the electro-elastic matrix is given by

The material properties ϵ and C are thus removed from the expression but note that the numerator and denominator terms are still in mixed units, and so even with diagonal scaling we are still summing quantities in different units. The diagonally scaled residual is of most use when the entire equation is multiplied by a factor. For instance,

:

Here is it clearly beneficial to divide both sides of the second equation by the pivot value -100 so that both equations have equal weight in the residual (it also helps to get a good solution). After division it is clear the unknowns all have similar orders of magnitude. However, the Multiphysics FE problems do not have this form. In the electro-elastic problem we have equations like this:

:

where are arbitrary constants to emphasise the problem would generally show coupling between the elastic and electric terms (above we have used, without loss of generality, ). In terms of dimensions, in transforming from the first equation to the second we have multiplied by on the left hand side but by on the right hand side: this is not equivalent to multiplying the second equation by a factor throughout. If we multiply by we get

Although the matrix is now dimensionless, the right hand sides clearly have different dimensions which implies that the solution values for will naturally be in different dimensions with different orders of magnitude. This shows why simple diagonal scaling does not remove dimensions from the problem and so will not always help in improving the condition number of the matrix or in establishing a meaningful residual.

Diagonal scaling is useful when the problem has Dirichlet boundaries, that is when various degrees of freedom are fixed. For example, returning to the dielectric problem,

If then we replace the second equation with this statement

One option now is to eliminate from the equations but if the equation is left as an explicit equation to be solved then there is a problem with dimensionality since the second row of Q is dimensionless while other rows are in dimensions of ϵ h. If we multiply both sides by , we obtain

where are equations are in the same dimensions. In practice Zinc will normally eliminate fixed variables from the equations. It will do so if removeFixed=YES (the default). However, this option does not currently work for the Newton-Raphson method for solving non-linear problems (newton=YES) so in that case diagonally-scaled residuals are advisable.