University of Bath
Faculty of Engineering & Design

Word count: 1584

November 9, 2018

Systems Modelling & Simulation Coursework 1

Supervisor
A. Cookson

Assessor

Author's Candidate Number
10838

Contents

List of Figures

1 Introduction

This paper is based on solving the static diffusion-reaction equation given by equation 1.

$$D\frac{\partial^2 c}{\partial x^2} + \lambda c + f = 0 \tag{1}$$

The weak form of this equation after integration by parts will be the starting point for derivations and is given by equation 2.

$$\int D\frac{\partial c}{\partial x}\frac{\partial v}{\partial x}dx - \int \lambda c v dx = \int v f dx + \left[vD\frac{\partial c}{\partial x}\right] \tag{2}$$

2 Part 1: Software Verification & Analytical Testing

2.1 Question 1a

2.1.1 Derivation of Diffusion Element Matrix

The 2-by-2 element matrix for a diffusion operator for an arbitrary element e_n between the points x_0 x_1 shall be derived. The derivation shall start from the weak form version of the static diffusion-reaction equation, after performing integration by parts, which is given by equation 2. To obtain the diffusion element matrix only the diffusion integral needs to be evaluated which will be done over the domain $x = 0$ to $x = 1$. The resulting integral to be evaluated is given by equation 3

$$\int_0^1 D\frac{\partial c}{\partial x}\frac{\partial v}{\partial x}dx \tag{3}$$

The domain from $x = 0$ to $x = 1$ can be split into ne number of elements. This is shown pictorially by Figure 1 for the case $ne = 4$.

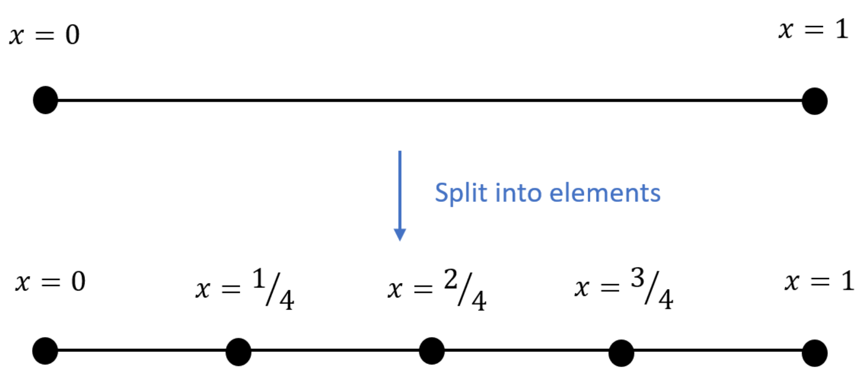


Figure 1: Splitting Domain Into Equispaced Elements

Now the integral from $x = 0$ to $x = 1$ can be split into the sum of the integrals of the individual elements as demonstrated by equation 4 for the $ne = 4$ case.

$$\int_0^1 dx = \int_0^{\frac{1}{4}} dx + \int_{\frac{1}{4}}^{\frac{2}{4}} dx + \int_{\frac{2}{4}}^{\frac{3}{4}} dx + \int_{\frac{3}{4}}^{1} dx \tag{4}$$

To solve the integrals the Galerkin formulation will be used were c and x are represented by linear Lagrange nodal functions as shown by equations 5a and 5b. As per the Galerkin assumption the test function v is set equal to the basis function ψ in order to minimise the residual error.

$$c = c_0\psi_0(\zeta) + c_1\psi_1(\zeta) \tag{5a}$$

$$x = x_0\psi_0(\zeta) + x_1\psi_1(\zeta) \tag{5b}$$

$$v = \psi_0, \psi_1 \tag{6}$$

where,

$$\psi_0 = \frac{1-\zeta}{2} \quad , \quad \psi_1 = \frac{1+\zeta}{2} \tag{7}$$

and

$$\zeta = 2\left(\frac{x - x_0}{x_1 - x_0}\right) - 1 \tag{8}$$

for x in that element between x_0 and x_1.

The local element shall be mapped to a standard element using the Jacobian transform J, given by equation 9, to map from the x to the ζ coordinate system. The transform is shown pictorially in Figure 2 for the second element of a four element equispaced mesh.

$$J = \left|\frac{dx}{d\zeta}\right| \tag{9}$$

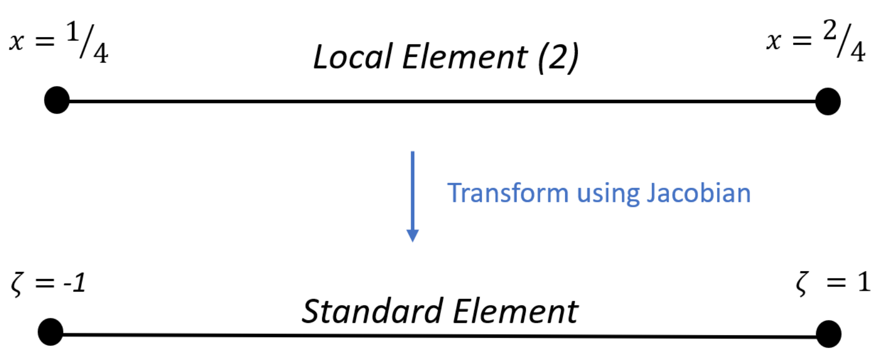


Figure 2: Mapping to Standard Element

The diffusion integral for a single element can now be mapped to the integral of a standard element using $dx = Jd\zeta$ which is a rearrangement of equation 9 and equation 8 to transform the limits of integration. The result is given by equation 10.

$$\int_{x_0}^{x_1} D\frac{\partial c}{\partial x}\frac{\partial v}{\partial x}dx = \int_{-1}^{1} D\frac{\partial c}{\partial x}\frac{\partial v}{\partial x}Jd\zeta \tag{10}$$

It is necessary to evaluate the derivatives $\frac{\partial c}{\partial x}$ and $\frac{\partial v}{\partial x}$ which can be obtain by applying the chain rule to the definitions of c and v given by equations 5a and 6. The results are given by equations 11 and 12.

$$
\begin{aligned}
\frac{dc}{dx} &= c_0 \frac{d\psi_0}{d\zeta}\frac{d\zeta}{dx} + c_1 \frac{d\psi_1}{d\zeta}\frac{d\zeta}{dx} \\
&= c_n \frac{d\psi_n}{d\zeta}\frac{d\zeta}{dx} \quad \text{for } n = 0, 1
\end{aligned} \tag{11}
$$

$$
\frac{dv}{dx} = \frac{d\psi_m}{d\zeta}\frac{d\zeta}{dx} \quad \text{for } m = 0, 1 \tag{12}
$$

The right hand side of equation 10 can now be rewritten as equation 13, recognising c_n is independent of x and therefore ζ.

$$
c_n \int_{-1}^{1} D \frac{d\psi_n}{d\zeta}\frac{d\zeta}{dx}\frac{d\psi_m}{d\zeta}\frac{d\zeta}{dx} J d\zeta \tag{13}
$$

Knowing that $\frac{d\zeta}{dx} = J^{-1}$ (for $x_1 > x_0$) from equation 9 and that for a given element J is constant, equation 10 can be rewritten as equation 14.

$$
c_n J^{-1} \int_{-1}^{1} D \frac{d\psi_n}{d\zeta}\frac{d\psi_m}{d\zeta} d\zeta \quad \text{for} \quad n = 0, 1 \ \& \ m = 0, 1 \tag{14}
$$

Equation 14 gives two equations which, when written in full, is clearly suitable for matrix representation.

$$
J^{-1}\left[c_0 \int_{-1}^{1} D \frac{d\psi_0}{d\zeta}\frac{d\psi_0}{d\zeta} d\zeta + c_1 \int_{-1}^{1} D \frac{d\psi_1}{d\zeta}\frac{d\psi_0}{d\zeta} d\zeta)\right] \tag{15a}
$$

$$
J^{-1}\left[c_0 \int_{-1}^{1} D \frac{d\psi_1}{d\zeta}\frac{d\psi_0}{d\zeta} d\zeta + c_1 \int_{-1}^{1} D \frac{d\psi_1}{d\zeta}\frac{d\psi_1}{d\zeta} d\zeta)\right] \tag{15b}
$$

The matrix representation is as follows where I_{nm} represents the individual integrals in the above equations 15a and 15b.

$$
J^{-1} \begin{bmatrix} Int_{00} & Int_{01} \\ Int_{10} & Int_{11} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} \tag{16}
$$

Each Int_{nm} term shall now be evaluated individually. In order to evaluate the integrals first the derivatives of ψ_0 and ψ_1 with respect to ζ shall be calculated using the definition of the test function given by equation 7. The results are given by equations

$$
\frac{d\psi_0}{d\zeta} = \frac{d}{d\zeta}(\frac{1-\zeta}{2}) = -\frac{1}{2} \tag{17a}
$$

$$
\frac{d\psi_1}{d\zeta} = \frac{d}{d\zeta}(\frac{1+\zeta}{2}) = \frac{1}{2} \tag{17b}
$$

$\underline{Int_{00}}$

$$\begin{aligned} Int_{00} &= \int_{-1}^{1} D\frac{d\psi_0}{d\zeta}\frac{d\psi_0}{d\zeta}d\zeta \\ &= \int_{-1}^{1} D.(-\frac{1}{2}).(-\frac{1}{2})d\zeta \\ &= \left[\frac{D}{4}\zeta\right]_{-1}^{1} \\ &= \left[(\frac{D}{4}.1) - (\frac{D}{4}. -1)\right] \\ &= \frac{D}{2} \end{aligned} \tag{18}$$

$\underline{Int_{01}}$

$$\begin{aligned} Int_{01} &= \int_{-1}^{1} D\frac{d\psi_0}{d\zeta}\frac{d\psi_1}{d\zeta}d\zeta \\ &= \int_{-1}^{1} D.(-\frac{1}{2}).(\frac{1}{2})d\zeta \\ &= \left[-\frac{D}{4}\zeta\right]_{-1}^{1} \\ &= \left[(-\frac{D}{4}.1) - (-\frac{D}{4}. -1)\right] \\ &= -\frac{D}{2} \end{aligned} \tag{19}$$

$\underline{Int_{10}}$

$$\begin{aligned} Int_{01} &= \int_{-1}^{1} D\frac{d\psi_1}{d\zeta}\frac{d\psi_0}{d\zeta}d\zeta \\ &= \int_{-1}^{1} D.(\frac{1}{2}).(-\frac{1}{2})d\zeta \\ &= \left[-\frac{D}{4}\zeta\right]_{-1}^{1} \\ &= \left[(-\frac{D}{4}.1) - (-\frac{D}{4}. -1)\right] \\ &= -\frac{D}{2} \end{aligned} \tag{20}$$

$\underline{Int_{11}}$

$$
\begin{aligned}
Int_{11} &= \int_{-1}^{1} D \frac{d\psi_1}{d\zeta} \frac{d\psi_1}{d\zeta} d\zeta \\
&= \int_{-1}^{1} D.(\frac{1}{2}).(\frac{1}{2}) d\zeta \\
&= \left[\frac{D}{4} \zeta \right]_{-1}^{1} \\
&= \left[(\frac{D}{4}.1) - (\frac{D}{4}.-1) \right] \\
&= \frac{D}{2}
\end{aligned} \tag{21}
$$

The local element matrix for the diffusion integral can now be assembled (not including the c term matrix). This is the form used in the code for LaplaceElemMatrix.m function available in Appendix A. Where J and D are scalars (we have assumed D to be constant).

$$
J^{-1} D \begin{bmatrix} 0.5 & -0.5 \\ -0.5 & 0.5 \end{bmatrix} \tag{22}
$$

2.1.2 Passes Unit Tests

Figure 3 shows the function LaplaceElemMatrix.m passes the unit tests defined in CourseworkOne-UnitTest.m, available in Appendix B, with no errors.

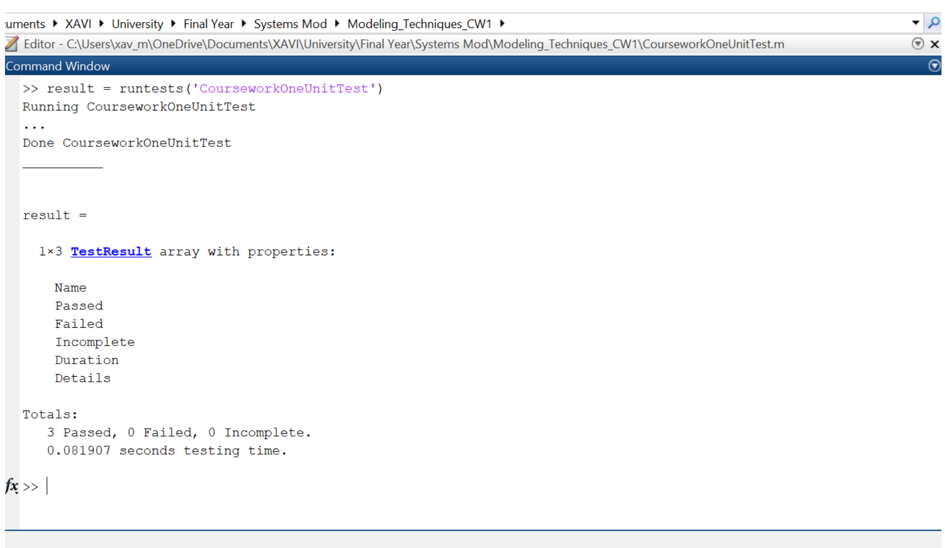


Figure 3: Screenshot of LaplaceElemMatrix.m Function Passing Unit Tests

2.2 Question 1b

2.2.1 Derivation of Reaction Element Matrix

The local reaction element matrix shall now be derived. This is found by evaluating the reaction integral, equation 23, from the starting point for derivations equation 2.

$$\int_{x_0}^{x_1} \lambda c v dx \tag{23}$$

As with the diffusion derivation the Jacobi is applied to map to the ζ domain. This gives equation 24.

$$\int_{-1}^{1} \lambda c v J d\zeta \tag{24}$$

The basis function for c defined by equation 5a and the Galerkin assumption for v given by equation 6 shall be used. Equation 24 can then be written as set of two equations, similar to equations 15a and 15b. Assuming λ to be independent of x gives equations 25a and 25a. These equations can also be written in the form of a matrix as shown by equation 26.

$$J\lambda \left[c_0 \int_{-1}^{1} \psi_0 \psi_0 d\zeta + c_1 \int_{-1}^{1} \psi_1 \psi_0 d\zeta) \right] \tag{25a}$$

$$J\lambda \left[c_0 \int_{-1}^{1} \psi_0 \psi_1 d\zeta + c_1 \int_{-1}^{1} \psi_1 \psi_1 d\zeta) \right] \tag{25b}$$

$$J\lambda \begin{bmatrix} Int_{00} & Int_{01} \\ Int_{10} & Int_{11} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} \tag{26}$$

The Int_{nm} integrals shall be evaluated to derive the matrix. This requires substituting in the definitions of ψ_0 and ψ_1 given by equation 7.

$\underline{Int_{00}}$

$$\begin{aligned} Int_{00} &= \int_{-1}^{1} \psi_0 \psi_0 \ d\zeta \\ &= \int_{-1}^{1} \Big(\frac{1-\zeta}{2}\Big)^2 d\zeta \\ &= \left[\frac{1}{3} \Big(\frac{1-\zeta}{2}\Big)^3 (-2) \right]_{-1}^{1} \\ &= \frac{2}{3} \end{aligned} \tag{27}$$

$\underline{Int_{01} = Int_{10}}$

$$
\begin{aligned}
Int_{00} &= \int_{-1}^{1} \psi_0 \psi_1 \ d\zeta \\
&= \int_{-1}^{1} \Big(\frac{1-\zeta}{2}\Big)\Big(\frac{1+\zeta}{2}\Big) d\zeta \\
&= \Big[\frac{\zeta}{4} - \frac{\zeta^3}{12}\Big]_{-1}^{1} \\
&= \Big[\frac{1}{6} - \Big(-\frac{1}{4} + \frac{1}{12}\Big)\Big]_{-1}^{1} \\
&= \frac{1}{3}
\end{aligned} \tag{28}
$$

<u>Int_{11}</u>

$$
\begin{aligned}
Int_{00} &= \int_{-1}^{1} \psi_1 \psi_1 \ d\zeta \\
&= \int_{-1}^{1} \Big(\frac{1+\zeta}{2}\Big)^2 d\zeta \\
&= \Big[\frac{1}{3}\Big(\frac{1+\zeta}{2}\Big)^3 .2\Big]_{-1}^{1} \\
&= \frac{2}{3}
\end{aligned} \tag{29}
$$

Putting the results of the integrals into the matrix from equation 26 gives the result shown by equation 30. This result is used by the LinearReactionElemMatrix.m function available in Appendix C.

$$
J\lambda \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} \end{bmatrix} \tag{30}
$$

As per equation 2 this result needs to be subtracted from the local diffusion element matrix result given by equation 22 in order to get the overall local element matrix which can then assemble into an global matrix. The global matrix is assembled in this way in the function StaticReactDiffSolver.m available in Appendix F

2.2.2 Linear Reaction Element Matrix Unit Test

A test script with four unit tests was made to check that the Linear Reaction Element Matrix function was working correctly. The tests were as follows

1. Check outputted matrix is symmetrical
2. Two outputs for two elements in an equispaced mesh are the same
3. Check against known solution from (from tutorial 3 q2c solution).
4. Element 11 is double the value of element 21 for random mesh size and λ

The test confirms the output matrix has all the properties of the matrix derived in equation 30. This includes symmetry, and the lead diagonal symmetric pair being double the anti-diagonal symmetric pair. The function will be also be tested to assert two different elements of an equispaced mesh are identical which confirms the function works over the entire domain and not just the first element. It is also necessary to check the values are correct as well as the form so the function has been tested against a known result in Test 3.
As the tests confirm the form is correct across the domain and test the values are also correct they provide sufficient confidence that the function is correct. However for added assurance test 4 has also been conducted for a random mesh size and a random value of λ .
The function LinearReactionElemMatrix.m passes the unit tests as shown by Figure 4. The test code is available in Appendix D

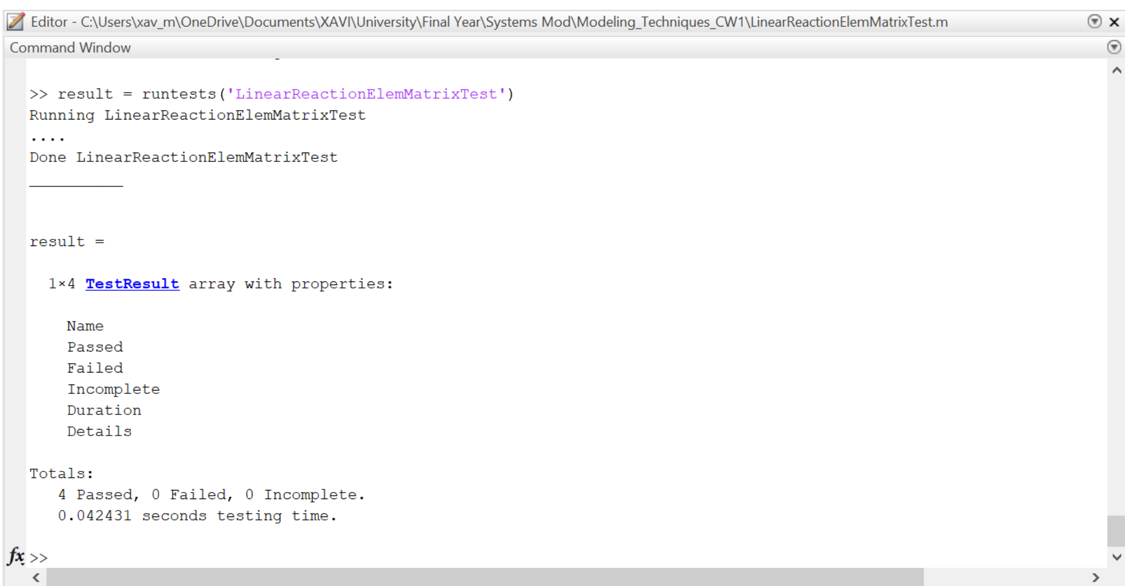


Figure 4: Screenshot of LinearReationElemMatrix.m Function Passing Unit Tests

2.3 Question 1c

2.3.1 Solving Laplace With Dirichlet Boundaries

The finite element solver StaticReactDiffSolver.m was used to solve

$$\frac{\partial^2 c}{\partial x^2} = 0 \tag{31}$$

over the domain $x = 0$ to $x = 1$ with the Dirichlet boundary conditions:

$$c = 2 \quad at \quad x = 0 \tag{32a}$$
$$c = 0 \quad at \quad x = 1 \tag{32b}$$

The analytical solution is given by equation 33.

$$c = 2(1 - x) \tag{33}$$

The result of the analytical solution has been plotted in Figure 5 with the FEM results overlaid. The FEM solution is very accurate here because linear approximations have been used as our basis functions and the analytical solution is also linear. This allowed good results even with a low resolution 4 element mesh.

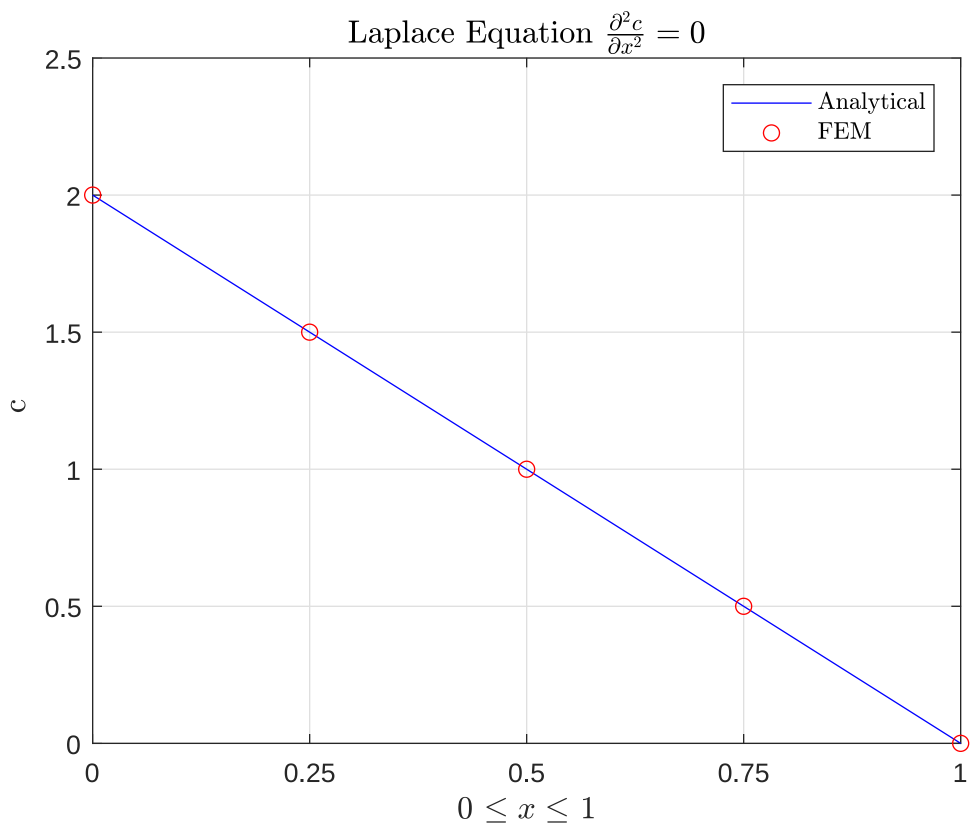


Figure 5: Comparison of Analytical and Finite Element Solutions of Laplace's Equation

2.3.2 Add a Neumann Boundary

The initial boundary condition shall be changed to a Neumann boundary, the conditions are given by equations 34a and 34b.

$$\frac{dc}{dx} = 2 \quad at \quad x = 0 \tag{34a}$$

$$c = 0 \quad at \quad x = 1 \tag{34b}$$

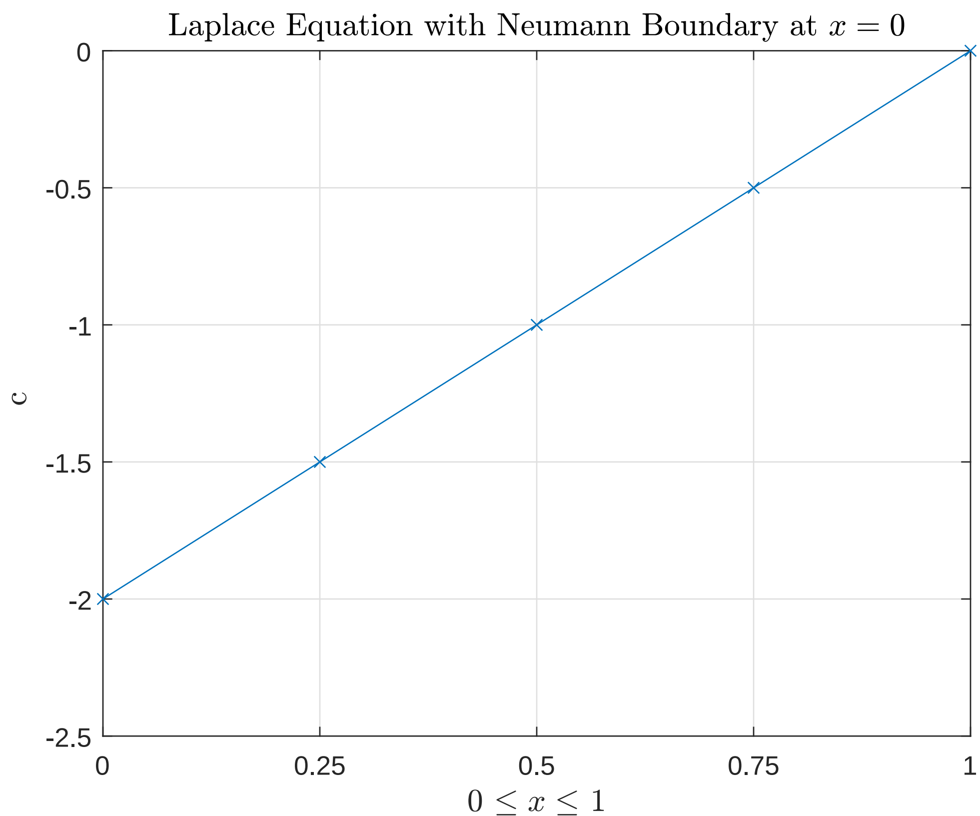


Figure 6: Using the FEM to solve Laplace's Equation with an initial Neumann Boundary

The solution found using the FEM method with a 4 element mesh is plotted in Figure 6. The solution is still linear which is expected but the effect of change the initial boundary condition from $c = 2$ to $\frac{\partial c}{\partial x} = 2$ has meant $c = -2$ at $x = 0$. This is to be expected as the function is linear and therefore has a constant gradient over the domain which is defined by the Neumann condition. As the Dirichlet boundary is fixed at $c = 0$ at $x = 1$ in order to achieve the gradient $\frac{\partial c}{\partial x} = 2$ the only solution is $c = -2$ at $x = 0$. The same result could be achieved with a Dirichlet boundary of $c = -2$ at $x = 0$.

2.4 Question 1d

A reaction term will now be introduced to solve the static diffusion-reaction equation:

$$D\frac{\partial^2 c}{\partial x^2} + \lambda c = 0$$

with the following parameters:

$$D = 1, \quad \lambda = -9$$

and the Dirichlet boundary conditions:

$$c = 0 \quad at \quad x = 0$$
$$c = 1 \quad at \quad x = 1.$$

The analytical solution is given by equation 35. This has been plotted on Figure 7 along with the Finite Element Method solution for a range of mesh sizes. It can be seen how the FEM converges on the analytical solution as the mesh size is increased. For a mesh size of 3 elements there is a clear deviation from the analytical solution. This divergence is clearer towards $x = 1$ where the gradient of the analytical solution changes sharply and the linear approximation is least valid. However once the a mesh size is increased to 10 elements the plot is difficult to distinguish from the analytical solution and by 25 elements the error is less than 1%.

$$c(x) = \frac{e^3}{e^6 - 1}(3e^{3x} - 3e^{-3x}) \tag{35}$$

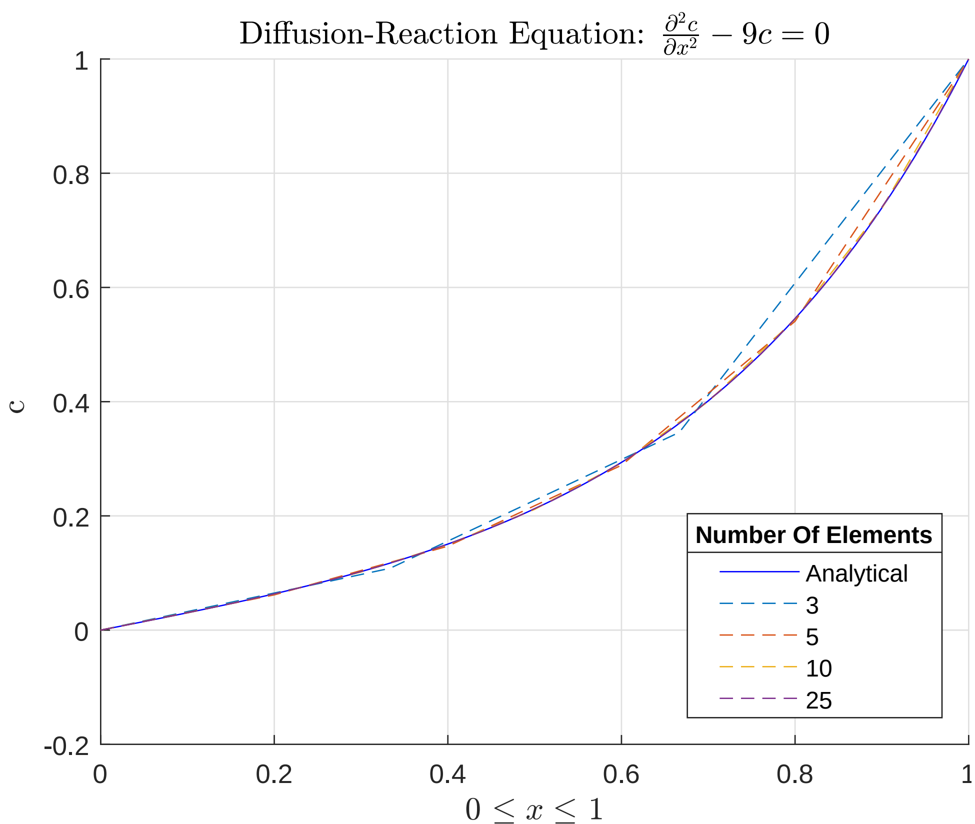


Figure 7: Using the FEM to solve the Static Diffusion-Reaction Equation with Dirichlet Boundary Conditions

3 Part 2

3.1 Question 2a

3.1.1 Setting The Equation

The FEM method to find the temperature profile through a material filled with small diameter heating channels. The cross section of the material is shown in Figure 8 and approximates to a 1D heat transfer problem given by equation 36.

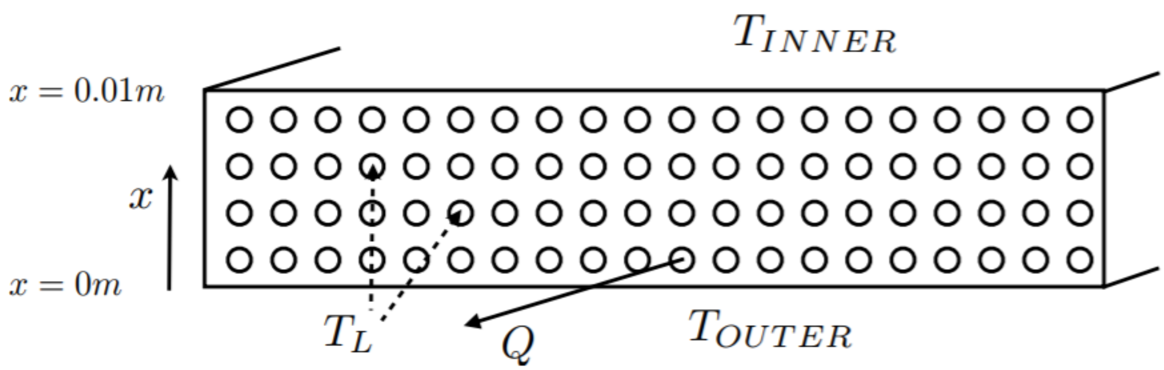


Figure 8: Cross Section of Material.

$$k\frac{\partial^2 T}{\partial x^2} + Q(T_L - T) = 0 \tag{36}$$

Rewriting this equation into the general diffusion-reaction equation form given by equation 1 gives us 37b.

$$D\frac{\partial^2 c}{\partial x^2} + \lambda c + f = 0 \tag{37a}$$

$$D\frac{\partial^2 T}{\partial x^2} + (-Q)T + QT_L = 0 \tag{37b}$$

The range of liquid flow rates, Q, and liquid temperatures, T_L is given below.

$$Q = 0.5 \text{ to } 1.5$$
$$T_L = 294.15K \text{ to } 322.15K$$

3.1.2 Effect of Varying Liquid Flow Rate

The equation was solved for a range of values of Q and the results plotted. This was repeated for four values of T_L and the results are shown in Figure 9.

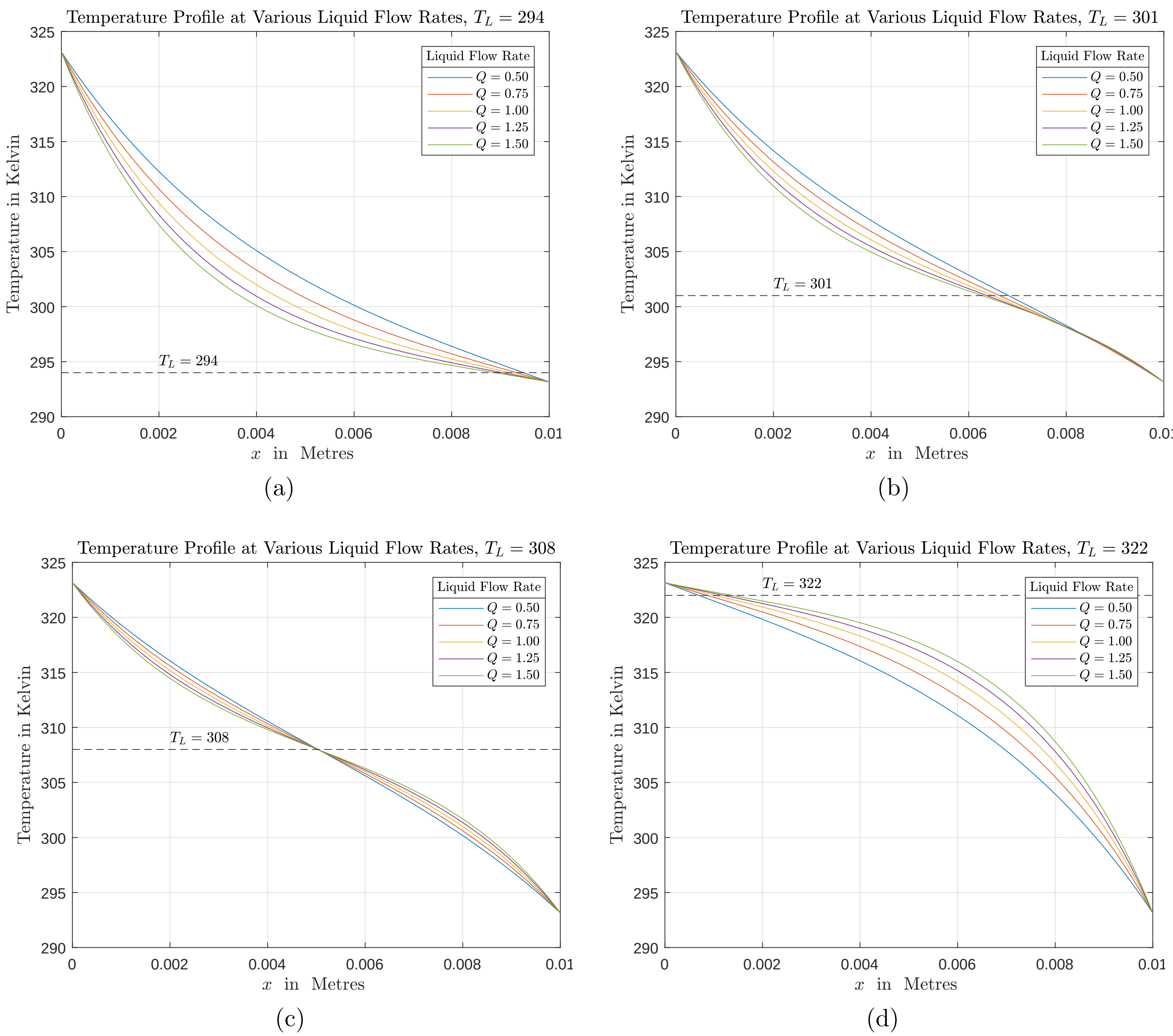


Figure 9: The Effect of Varying the Liquid Flow Rate on Temperature Distribution and Gradient

The effect of increasing the liquid flow rate is to bring the temperature profile towards the liquid temperature. This is most evident when their is a large differential temperature between the liquid and a boundary condition as a large temperature differential means a lot of heat can be transferred. For example in Figure 10a the liquid temperature is much cooler than the left hand boundary and so is heated by the material. As the thermal energy is transferred to the liquid its temperature rises reducing the differential temperature which reduces the rate of heat transfer. With higher flow rates the liquid temperature does not rise as much and so there is more heat transfer and steeper temperature gradient at and near the LHS boundary compared to the lower flows.

3.1.3 Effect of Varying Liquid Temperature

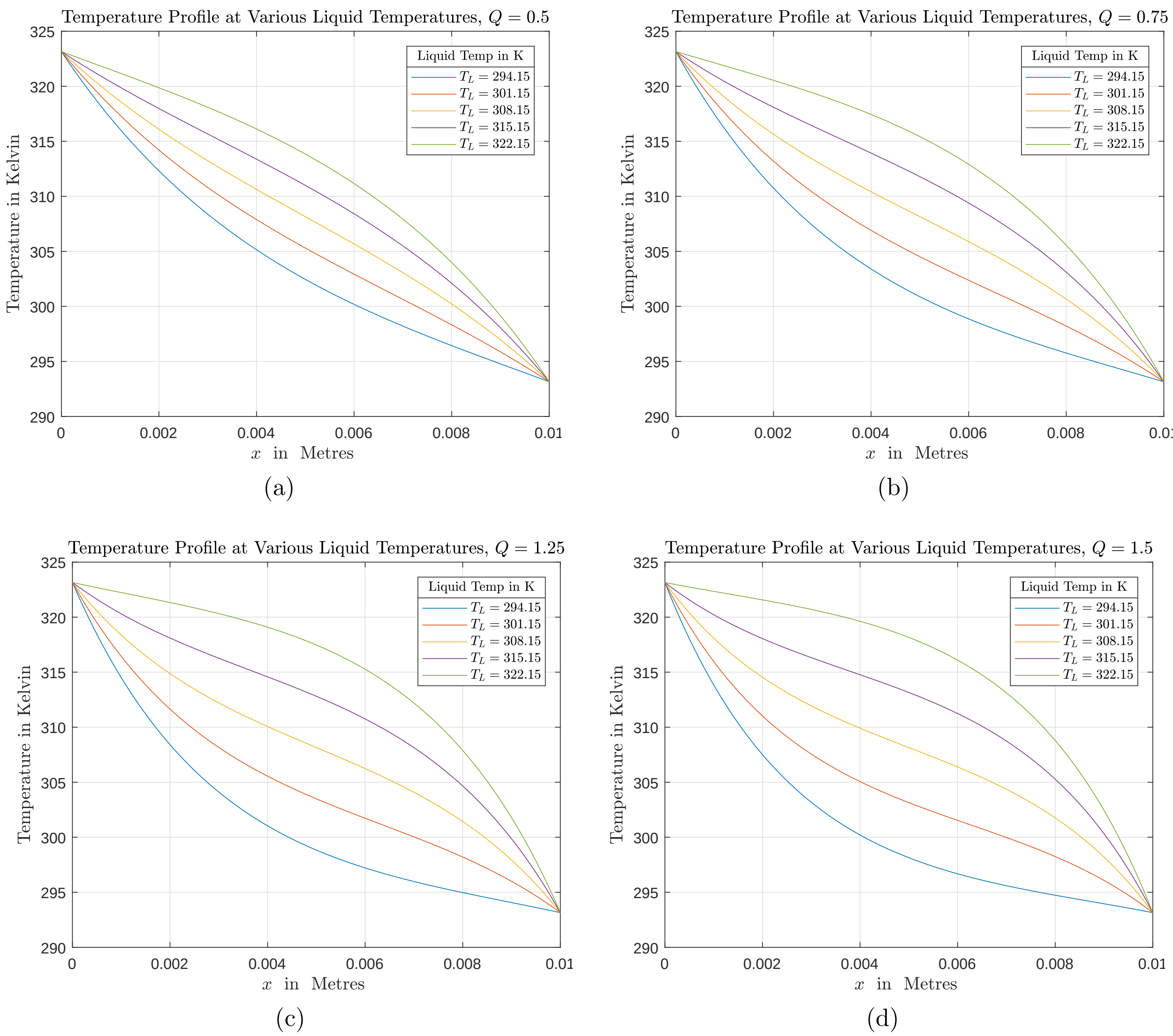


Figure 10: The Effect of Varying the Liquid Temperature on Temperature Distribution and Gradient

Figure 10 show how the liquid temperature determines the curvature of the temperature profile. When the liquid temperature is at 308.15k the temperature profile is relatively linear as this temperature is the mid-point between the two boundary conditions. When the liquid temperature increases the profile becomes more parabolic and convex. Similarly as the temperature decreases the profile becomes more parabolic and concave. Again this effect is more pronounced with higher liquid flows rates.

3.1.4 Effect of Mesh Size

To run the solver efficiently minimum mesh size which provides sufficient resolution is needed. To find the appropriate mesh size the least linear solution needs to be used which is the highest liquid flow rate $Q = 1.5$ and minimum liquid temperature $T_L = 294.15$. The solution for several mesh sizes is shown in Figure 11.

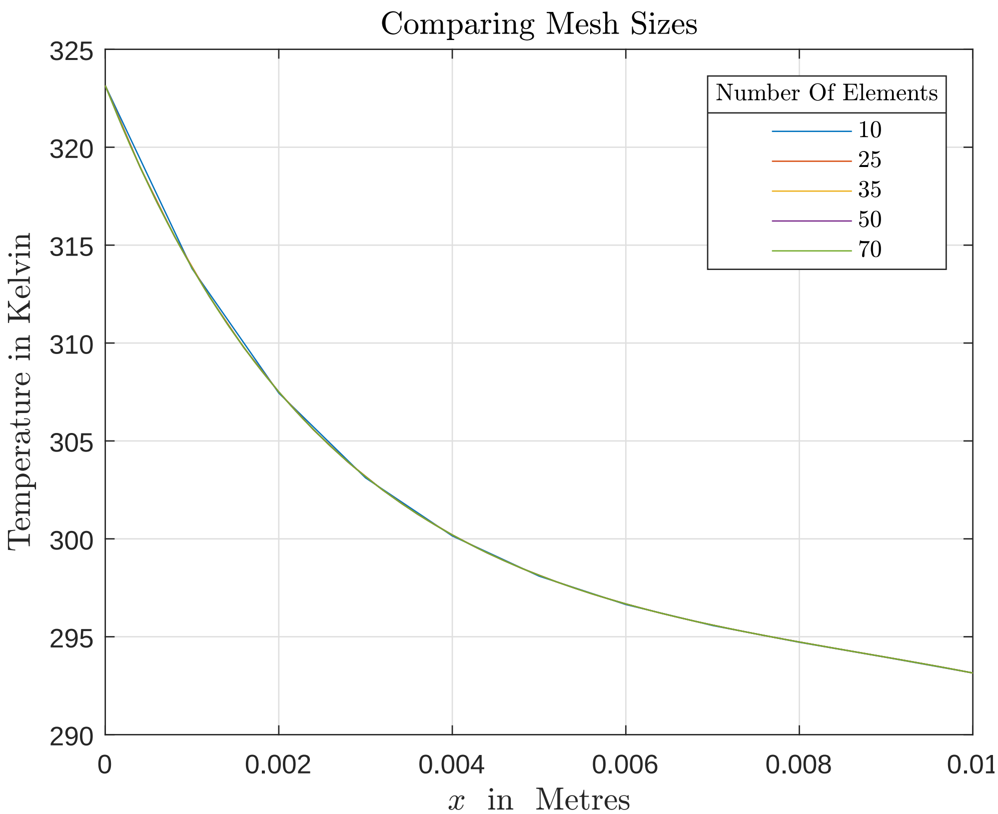


Figure 11: Cross Section of Material

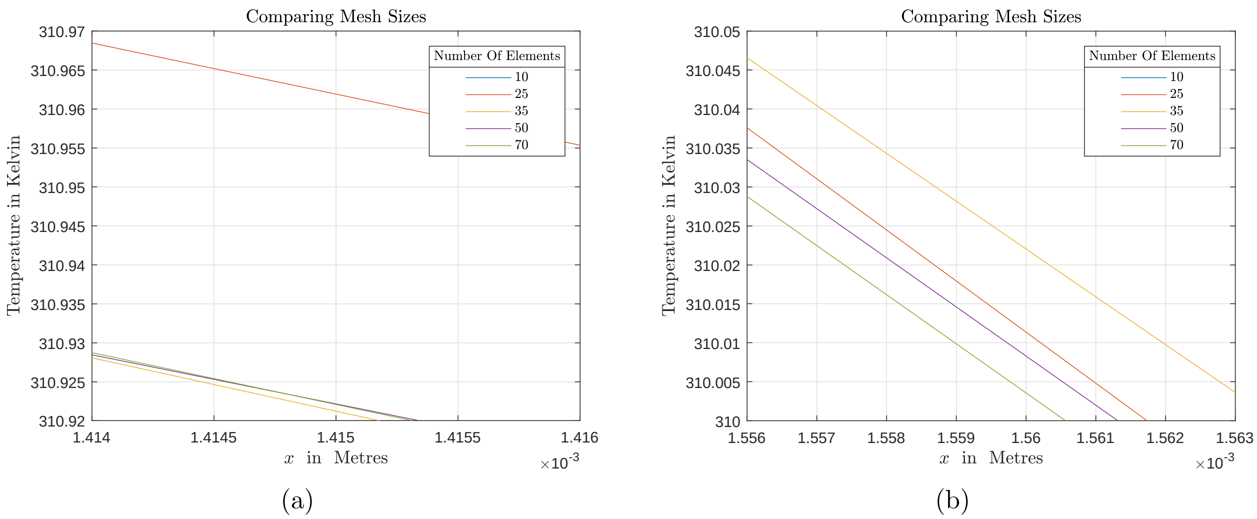


Figure 12: The Effect of Mesh Size on Temperature Resolution

An acceptable resolution for temperature is .01K as only very specialised and expensive temperature sensors have greater accuracy than this. In Figure 12a it appears as though the 35, 50 and 70 element meshes have all converged to well within 0.01k. Figure 12b, which is at the same temperature scale as Figure 12a, shows that this convergence does not hold over the entire domain. The 35 element mesh is now approximately .02k above the 70 element mesh however the 50 and 70 element meshes are still within .01K. Figure 12b is at one of the points in Figure 11 with the greatest divergence in solutions and so it can be said that a 70 element mesh is provides the required resolution.

3.2 Question 2b

3.2.1 Derivation of Linear Source Term

The temperature of the liquid is changed to be a function of x resulting in a new governing equation described by equation 38.

$$k\frac{\partial^2 T}{\partial x^2} + Q(T_L(1+4x) - T) = 0 \tag{38}$$

The equation can be rearranged to the standard form as given by equation 37a which gives us equation **??**.

$$k\frac{\partial^2 T}{\partial x^2} + (-Q)T + [QT_L + 4QT_Lx] = 0 \tag{39}$$

This is the same as equation 37b but with an extra source term $4QT_Lx$. To derive the source term vector the following integral from equation 1 needs to be solved.

$$\int_0^1 vf dx = \int_0^1 v\left[QT_L + 4QT_Lx\right] dx$$

This can be split into the sum of two separate integrals as follows.

$$\int_0^1 v\left[QT_L + 4QT_Lx\right] dx = \int_0^1 vQT_L dx + 4QT_L \int_0^1 vx dx$$

Split this integration into a sum of elements as shown below for four elements.

$$\int_0^1 4QT_Lx dx = \int_0^{\frac{1}{4}} 4QT_Lx dx + \int_{\frac{1}{4}}^{\frac{2}{4}} 4QT_Lx dx + \int_{\frac{2}{4}}^{\frac{3}{4}} 4QT_Lx dx + \int_{\frac{3}{4}}^1 4QT_Lx dx$$

Applying the Jacobi to map to the ζ domain and taking constants out of the integral gives the following.

$$\int_{x_0}^{x_1} = 4QT_L \int_{-1}^1 vxJd\zeta \tag{40}$$

To solve this extra term it is necessary to use the basis function for x given by equation 5b and shown below as well as the basis function for v.

$$\begin{aligned} x &= x_0\psi_0(\zeta) + x_1\psi_1(\zeta) \\ v &= \psi_0, \psi_1 \end{aligned}$$

As before this gives a set of two integrals which can be written in matrix form as shown below.

$$4QT_LJ\left[x_0\int_{-1}^1 \psi_0\psi_0 d\zeta + x_1\int_{-1}^1 \psi_1\psi_0 d\zeta)\right] \tag{41a}$$

$$4QT_LJ\left[x_0\int_{-1}^1 \psi_0\psi_1 d\zeta + x_1\int_{-1}^1 \psi_1\psi_1 d\zeta)\right] \tag{41b}$$

$$4QT_LJ\begin{bmatrix}Int_{00} & Int_{01}\\ Int_{10} & Int_{11}\end{bmatrix}\begin{bmatrix}x_0\\ x_1\end{bmatrix} \tag{42}$$

We shall now evaluate the Int_{nm} integrals to derive the matrix. It can be noted that these are the same Int_{nm} terms that were solved to derive the reaction element matrix and so the same result should be found.

$\underline{Int_{00}}$

$$\begin{aligned}Int_{00} &= \int_{-1}^{1} \psi_0\psi_0 \ d\zeta\\ &= \int_{-1}^{1} \Big(\frac{1-\zeta}{2}\Big)^2 d\zeta\\ &= \Big[\frac{1}{3}\Big(\frac{1-\zeta}{2}\Big)^3(-2)\Big]_{-1}^{1}\\ &= \frac{2}{3}\end{aligned} \tag{43}$$

$\underline{Int_{01} = Int_{10}}$

$$\begin{aligned}Int_{00} &= \int_{-1}^{1} \psi_0\psi_1 \ d\zeta\\ &= \int_{-1}^{1} \Big(\frac{1-\zeta}{2}\Big)\Big(\frac{1+\zeta}{2}\Big)d\zeta\\ &= \Big[\frac{\zeta}{4} - \frac{\zeta^3}{12}\Big]_{-1}^{1}\\ &= \Big[\frac{1}{6} - \Big(-\frac{1}{4} + \frac{1}{12}\Big)\Big]_{-1}^{1}\\ &= \frac{1}{3}\end{aligned} \tag{44}$$

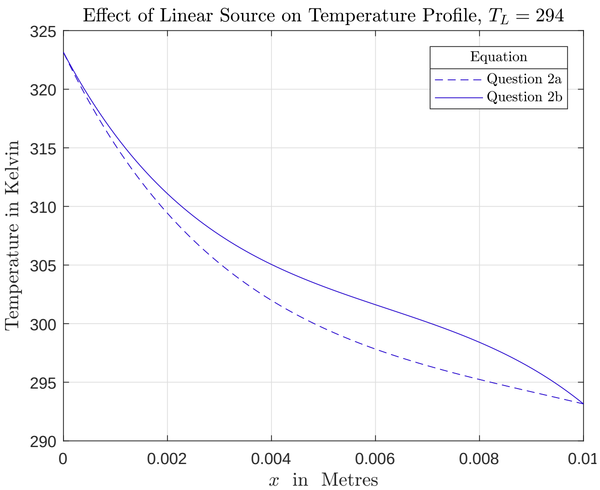
$\underline{Int_{11}}$

$$\begin{aligned}Int_{00} &= \int_{-1}^{1} \psi_1\psi_1 \ d\zeta\\ &= \int_{-1}^{1} \Big(\frac{1+\zeta}{2}\Big)^2 d\zeta\\ &= \Big[\frac{1}{3}\Big(\frac{1+\zeta}{2}\Big)^3 .2\Big]_{-1}^{1}\\ &= \frac{2}{3}\end{aligned} \tag{45}$$

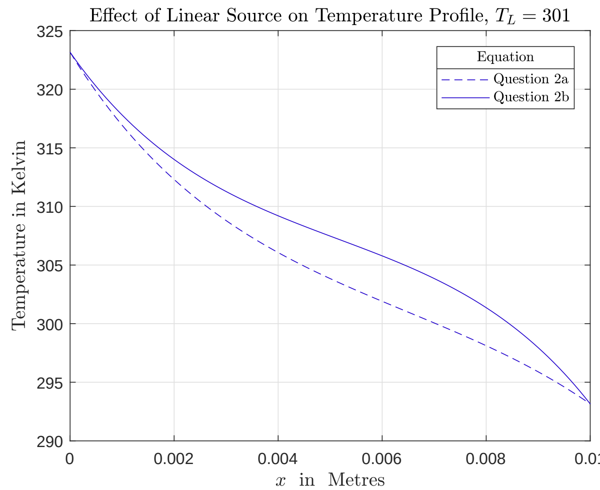
Now putting substituting these results into the matrix form to get the following solution for the local element vector linear source terms, f_L at the local element nodes 0 and 1. These local linear source nodes can be added to the local 'scalar' source nodes created for the QT_L term, thus deriving the global source matrix needed for the FEM solver.

$$\begin{bmatrix} f_{L_0} \\ f_{L_1} \end{bmatrix} = 4QT_LJ \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} \tag{46}$$

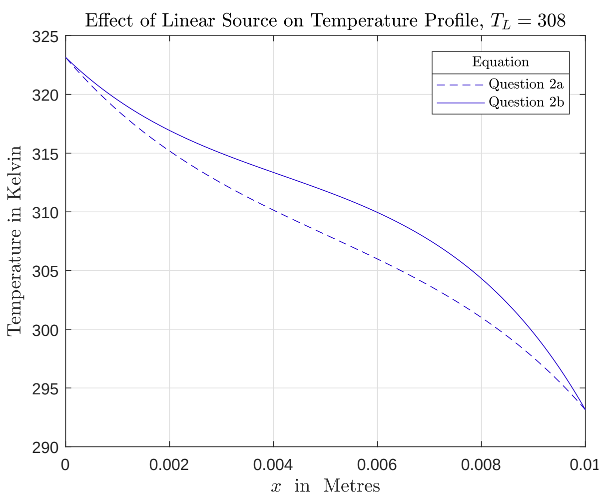
3.2.2 Linear Source Results



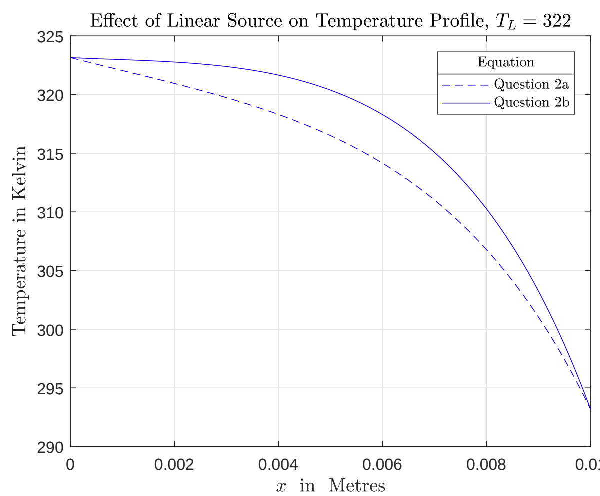
(a)



(b)



(c)



(d)

Figure 13: The Effect of Varying the Liquid Temperature on Temperature Distribution and Gradient ($Q = 1$)

Figure 13 shows how the linear source term, which represents the liquid temperature rising as x increases, raises the temperature profile at the right hand boundary compared to the constant liquid temperature case.

Appendices

A Laplace Element Matrix Code

```
function [SqMatrix] = LaplaceElemMatrix(D, eID, msh)

%Returns the local 2x2 element matrix of a given element for a given diffusion
%coefficient
%
% Inputs:
% D - Coefficient of Diffusion
% eID - Index of element within mesh structure
% msh - Mesh which contains local elements within it's structure

%% Form of Laplace Elem matrix for J=1, D=1

SqMatrix = [0.5, -0.5; ...
            -0.5, 0.5];

%% Multiply  by (1/J) and D to get solution for the particular element

J = msh.elem(eID).J;  %Get Jacobi for the element

SqMatrix = (1/J) * D * SqMatrix;   %Local element matrix for element eID
end
```

B CourseworkOneUnitTest.m

```
%% Test 1: test symmetry of the matrix
% Test that this matrix is symmetric
tol = 1e-14;
D = 2; %diffusion coefficient
eID=1; %element ID
msh = OneDimLinearMeshGen(0,1,10);

elemat = LaplaceElemMatrix(D,eID,msh); %THIS IS THE FUNCTION YOU MUST WRITE

assert(abs(elemat(1,2) - elemat(2,1)) ≤ tol)

%% Test 2: test 2 different elements of the same size produce same matrix
% % Test that for two elements of an equispaced mesh, as described in the
% % lectures, the element matrices calculated are the same
tol = 1e-14;
D = 5; %diffusion coefficient
eID=1; %element ID
msh = OneDimLinearMeshGen(0,1,10);

elemat1 = LaplaceElemMatrix(D,eID,msh);%THIS IS THE FUNCTION YOU MUST WRITE

eID=2; %element ID

elemat2 = LaplaceElemMatrix(D,eID,msh);%THIS IS THE FUNCTION YOU MUST WRITE

diff = elemat1 - elemat2;
diffnorm = sum(sum(diff.*diff));
assert(abs(diffnorm) ≤ tol)

%% Test 3: test that one matrix is evaluted correctly
% % Test that element 1 of the three element mesh problem described in the lectures
% % the element matrix is evaluated correctly
tol = 1e-14;
D = 1; %diffusion coefficient
eID=1; %element ID
msh = OneDimLinearMeshGen(0,1,3);

elemat1 = LaplaceElemMatrix(D,eID,msh);%THIS IS THE FUNCTION YOU MUST WRITE

elemat2 = [ 3 -3; -3 3];
diff = elemat1 - elemat2; %calculate the difference between the two matrices
diffnorm = sum(sum(diff.*diff)); %calculates the total squared error between ...
    the matrices
assert(abs(diffnorm) ≤ tol)
```

C Linear Reaction Element Matrix Code

```
function [SqMatrix] = LinearReactionElemMatrix(lambda, eID, msh)

%Returns a local 2x2 element matrix of a given element for a diffusion
%operator
% Inputs:
% lambda - Scalar Coefficient of Diffusion
% eID - Index of element within mesh structure
% msh - Mesh which contains local elements within it's structure

%% Form of Linear Reaction local element matrix for J = 1 and lambda = 1
SqMatrix = [(2/3), (1/3); ...
            (1/3), (2/3)];

%% Multiply by J and lambda to get solution for the particular element

J = msh.elem(eID).J;  %Get Jacobi for the element

SqMatrix = J * lambda .* SqMatrix;   %Local Linear Reaction matrix for element eID
end
```

D Linear Reaction Element Matrix Test

```
%%%THIS SCRIPT TESTS LinearReactionElemMatrix.m WORKS CORRECTLY

%% Test 1: test symmetry of the matrix
% Test that this matrix is symmetric
tol = 1e-14;
lambda = 2; %diffusion coefficient
eID=1; %element ID
msh = OneDimLinearMeshGen(0,1,10);  %generate mesh using given funciton

elemat = LinearReactionElemMatrix(lambda,eID,msh); %Calculate Linear Element Matrix

assert(abs(elemat(1,2) - elemat(2,1)) ≤ tol && abs(elemat(1,1) - elemat(2,2))  ≤...
     tol, ...
    'Local element matrix is not symmetric!')
%% Test 2: test 2 different elements of the same size produce same matrix
%Test that for two elements of an equispaced mesh, as described in the
%lectures, the element matrices calculated are the same
tol = 1e-14;
lambda = 5; %diffusion coefficient
eID=1; %element ID
msh = OneDimLinearMeshGen(0,1,10);

elemat1 = LinearReactionElemMatrix(lambda,eID,msh); %calculate element 1 matrix

eID=2; %element ID

elemat2 = LinearReactionElemMatrix(lambda,eID,msh); %calculate element 2 matrix

diff = elemat1 - elemat2;
diffnorm = sum(sum(diff.*diff));
assert(abs(diffnorm) ≤ tol)

%% Test 3: test that one matrix is evaluted correctly
% % Test that element 1 of the three element mesh problem described
% in Tutorial 3 Question 2c has the element matrix evaluated correctly
tol = 1e-14;
lambda = 1; %diffusion coefficient
eID=1; %element ID
msh = OneDimLinearMeshGen(0,1,6);

elemat1 = LinearReactionElemMatrix(lambda,eID,msh); %calculate element 1 matrix

elemat2 = [ (1/18), (1/36); (1/36), (1/18)];    % This is the known result
diff = elemat1 - elemat2; %calculate the difference between the two matrices
SummedDiff = sum(sum(diff)); %calculates the sum of the elements in the diff matrix
assert(abs(SummedDiff) ≤ tol) %Checks the error of the summed differences is ...
    below tol

%% Test 4: test main diagonal values are double antidiagonal values
% using random inputs for number of elemnts and lambda
tol = 1e-14;
```

```
lambda = 5*abs(rand(1,1)); %returns random diffusion coefficient between 0 and 4
eID=1; %element ID
ne = randi([4 100], 1,1);     %sets number of elements to random integer ...
    between 4 and 100
msh = OneDimLinearMeshGen(0,1,ne);

elemat1 = LinearReactionElemMatrix(lambda,eID,msh);
elem = elemat1(1,1)/elemat1(2,1); %Gets ratio between 1,1 and 2,1 elements
assert(abs(elem - 2) ≤ tol)     %Checks ratio is 2 within the tolerance
```

E Apply Boundary Condition Code

```
function [GlobalMatrix, F] = ApplyBCs(BC, GlobalMatrix, F, ne)
% Applies boundary conditions to F and Global Matrix as appropriate
% for the specified boundary condition
%
%Inputs:
%BC - Structure which contains the following:
%       BC(1) - Holds data for minimum x boundary
%       BC(2) - Holds data for maximum x boundary
%       BC().type - Type of boundary condition: "neumann", "dirichlet" or
%                   "none" (must be a lower case string)
%       BC().value - Value of the boundary condition (float or int)
%
%GlobalMatrix - formation of local element matrices (NxN matrix)
%F - Source Vector (size Nx1 vector)

%% Solve xmin Boundary Condition
if BC(1).type == "none" %No action needed if no boundary condition
    % pass

elseif BC(1).type == "dirichlet"  %solve for a dirichlet BC

    %set all first row elements to 0 except first element
    GlobalMatrix(1,:) = [1, zeros(1,ne)];
    %set first element of the source vector to the xmin BC value
    F(1) = BC(1).value;

elseif  BC(1).type == "nuemann"    %solve for a nuemann BC

    %subtract the xmin BC value from the first element of the source vector
    F(1) = F(1) - BC(1).value;
end

%% Solve xmax Boundary Condition
if BC(2).type == "none" %No action needed if no boundary condition
    % pass

elseif BC(2).type == "dirichlet"  %solve for dirichlet BC

    %set all first row elements to 0 except last element
    GlobalMatrix(end,:) = [zeros(1,ne), 1];
    %set last element of the source vector to the xmax BC value
    F(end) = BC(2).value;

elseif BC(2).type == "nuemann"    %solve nuemann BC

    %subtract the xmin BC value from the last element of the source vector
    F(end) = F(end) + BC(2).value;
end
end
```

F Static Diffusion-Reaction Solver Code

```
function [mesh] = StaticReactDiffSolver(lambda, D, xmin, xmax, ne,f_scalar, ...
    f_linear,  BC)
%%% Solves the static reaction diffusion equation
% Inputs:
% lambda - Coefficient for Reaction (float ot int)
% D - Coefficient of Diffusion (float or int)
% xmin - Minimum value for x, usually 0(float or int)
% xmax - Maximum value for x, usually 1(float or int)
% ne - Number of Elements in Mesh (float)
% f_scalar - Constant Source Term (float or int)
% f_linear - Linear Source Term (float or int)
%BC - Structure which contains the following:
%        BC(1) - Holds data for minimum x boundary
%        BC(2) - Holds data for maximum x boundary
%        BC().type - Type of boundary condition: "neumann", "dirichlet" or
%                    "none" (must be a lower case string)
%        BC().value - Value of the boundary condition (float or int)

mesh = OneDimLinearMeshGen(xmin, xmax, ne); %Generate Mesh

GlobalMatrix = zeros((ne+1),(ne+1));  %Initiate Global Matrix

%% Generate the source the vector
F = SourceVectorGen(mesh, f_scalar, f_linear);

%% Assemble local element matrices into global matrix
for eID = 1:ne

    %Calculate Local Element Matrix for Diffusion Term
    DiffusionLocal = LaplaceElemMatrix(D, eID, mesh);

    %Calculate Local Element Matrix for Linear Reaction Term
    ReactionLocal = LinearReactionElemMatrix(lambda, eID, mesh);

    %Local Element Matrix is the Diffusion subtract the Linear Reation
    LocalElementMatrix = DiffusionLocal - ReactionLocal;

    %Add Local Element Matrix into the correct location within the Global Matrix
    GlobalMatrix(eID:(eID+1),eID:(eID+1)) = ...
        GlobalMatrix(eID:(eID+1),eID:(eID+1))...
                                              + LocalElementMatrix;
end

%Apply Boundary Conditions
[GlobalMatrix, F] = ApplyBCs(BC, GlobalMatrix, F, ne);

%% Solve for C
C = GlobalMatrix \ F;

%Add values of C into the mesh structure
for i = 1:length(C)
```

```
    mesh.c(i) = C(i);
end

end
```

G Source Vector Generator

```
function [F] = SourceVectorGen(mesh, f_scalar, f_linear)
%Generates source vector
%Inputs:
%mesh - Mesh contains local elements and other data in it's structure
%f_scalar - Constant source term
%f_linear - Linear source term

ne = mesh.ne;    %Number of elements
J = mesh.elem(1).J; %Jacobi (assumed equally spaced mesh)

%% Initalise Scalar Part of Source Vector F and add in Scalar Part
F = ones((ne+1),1);
F = (2*f_scalar*J) .* F;
F(1) = f_scalar*J;
F(end) = f_scalar*J;

%% Add in the Linear Element Matrices
for eID = 1:ne
    %Caluculat Linear Source Local Element Matrix
    LinearSourceLocal = LinearSourceElemMatrix(mesh, eID, f_linear);
    %AddLocal Linear Source into Source Vector at correct location
    F(eID:(eID+1)) = F(eID:(eID+1)) + LinearSourceLocal;
end
```

H Linear Source Vector

```
function [ElemVector] = LinearSourceElemVector(mesh, eID, f_linear)
% Returns the Local Linear Source Vector for an element
%
% Inputs:
% mesh - Mesh which contains local elements within it's structure and
%            related variables
% eID - index for the element within the mesh
% f_linear - Linear source term multiplier

%% Extract element variables from mesh
J =mesh.elem(1).J;      %Get Jacobi for the element
x0 = mesh.elem(eID).x(1);   %Get he x0 value for the element
x1 = mesh.elem(eID).x(2);   %Get he x1 value for the element
xVector = [x0;x1];          %Create a vector of the x values

%% Local Source Vector with J =1 and f_linear = 1
StandardVector = [(2/3), (1/3);...
                  (1/3), (2/3)];

%% Multiply by J, f_linear and the x vector to get Local Linear Source
%Vector for the specific element
ElemVector = J * f_linear *  StandardVector * xVector;
end
```