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

UNIVERSITY OF
BATH

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.

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

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 and 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 $ne = 4$.
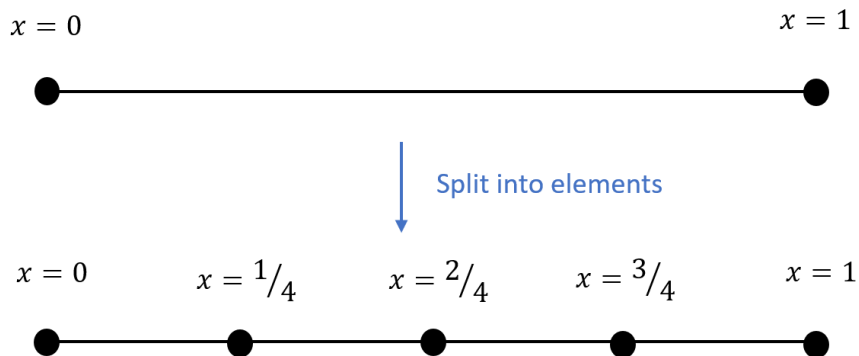
Figure 1: Splitting Domain Into Equispaced Elements

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.

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

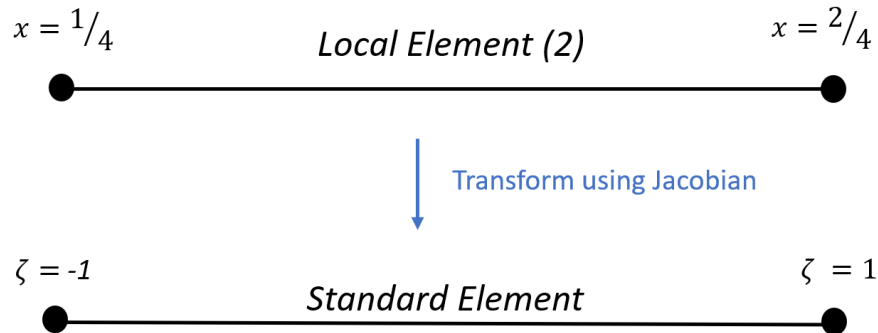It is necessary to evaluate the derivatives $\frac{\partial c}{\partial x}$ and $\frac{\partial v}{\partial x}$ which can be obtained by applying the chain rule to the definitions of c and v given by equations 5a and 6. The results are given by equations 11 and 12.

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

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

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

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

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.

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

Equation 14 gives two equations which, when written in full, are 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

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

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

Int_{00}

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

Int_{01}

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

$\underline{Int_{10}}$

$$Int_{01} = \int_{-1}^{1} D\frac{d\psi_1}{d\zeta}\frac{d\psi_0}{d\zeta}d\zeta$$
$$= \int_{-1}^{1} D.(\frac{1}{2}).(-\frac{1}{2})d\zeta$$
$$= \left[-\frac{D}{4}\zeta\right]_{-1}^{1} \tag{20}$$
$$= \left[(-\frac{D}{4}.1) - (-\frac{D}{4}.-1)\right]$$
$$= -\frac{D}{2}$$
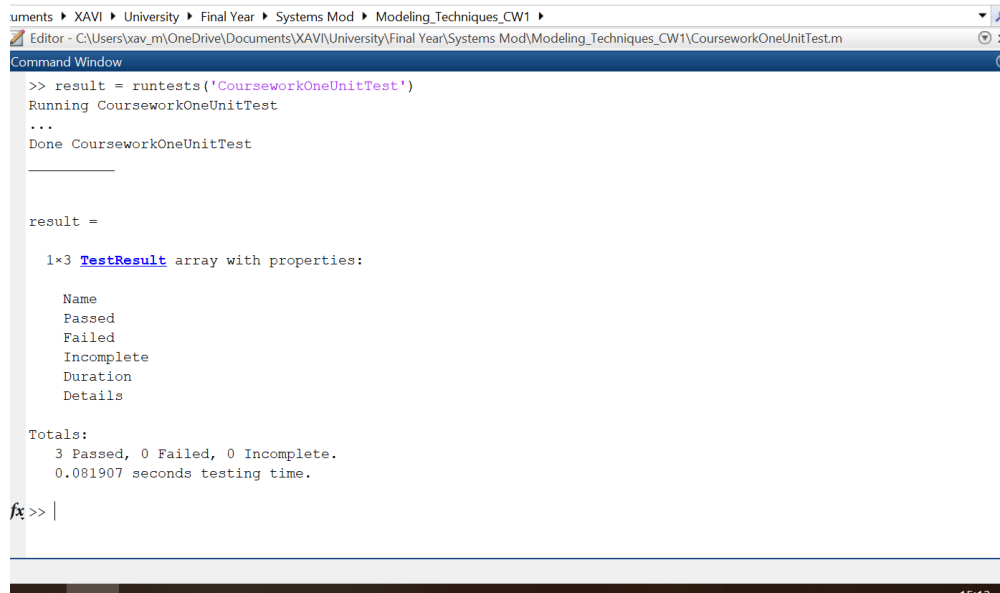
$\underline{Int_{11}}$

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

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 B. 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 C, 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.

6

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

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

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

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.

$\underline{Int_{00}}$

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

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

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

$\underline{Int_{11}}$

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

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 D.

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

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 G

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 E

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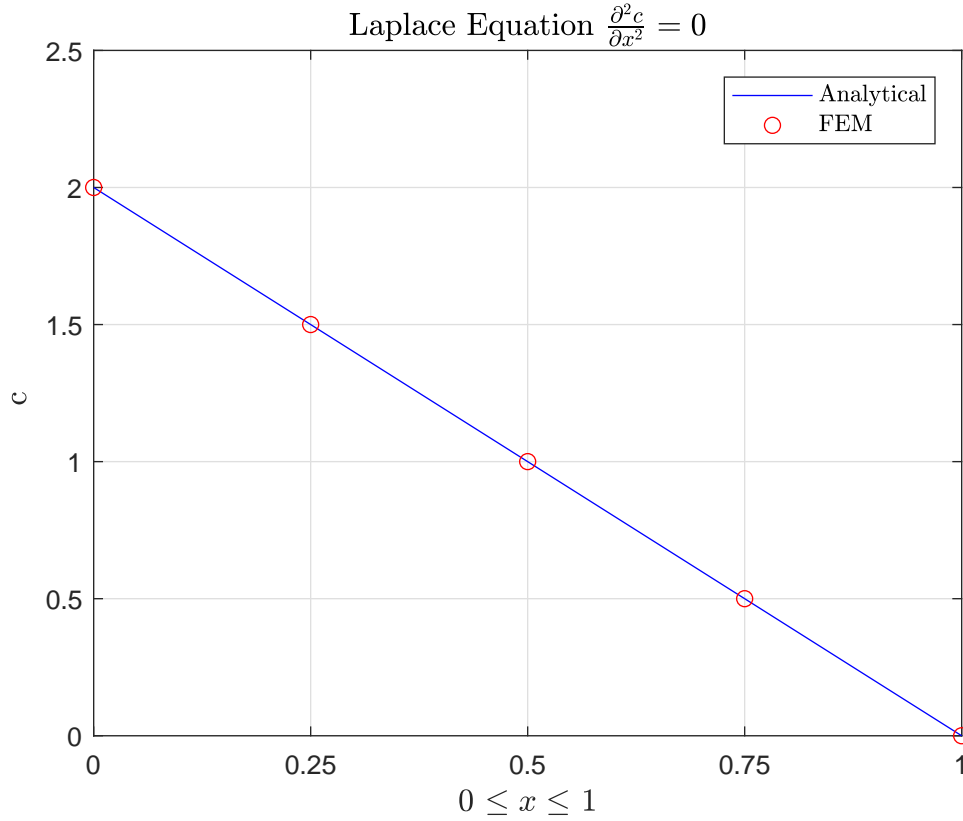
Figure 5: Comparison of Analytical and Finite Element Solutions of Laplace's Equation

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.

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

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

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 $x = 0$.
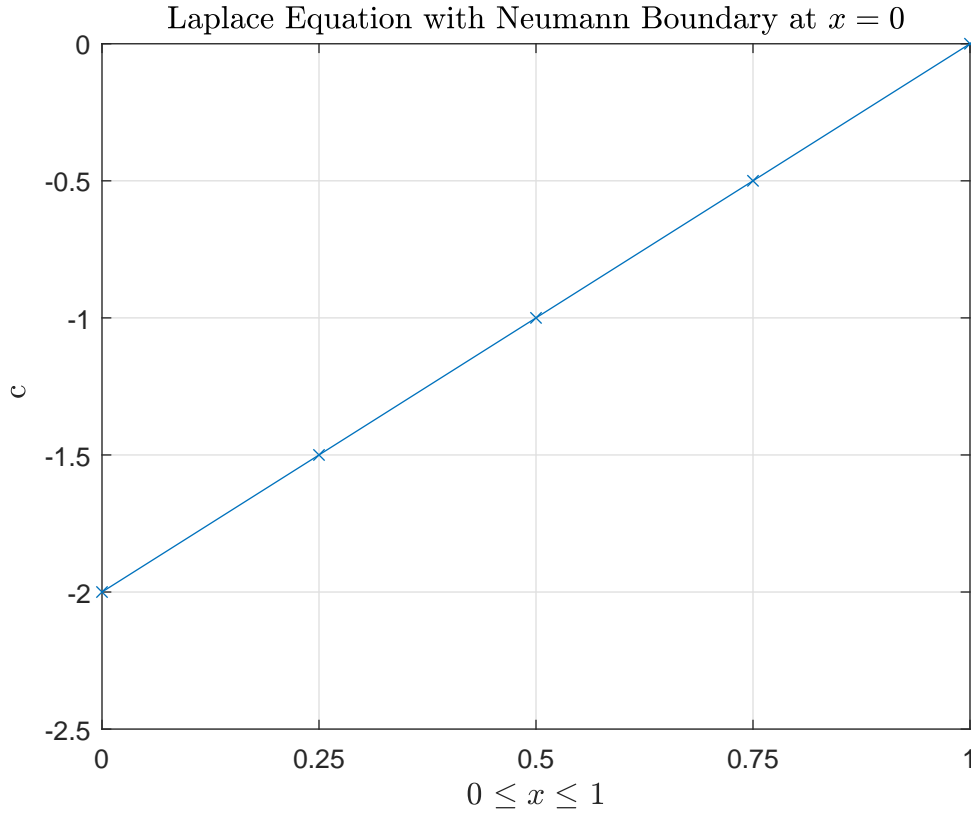
Figure 6: Using the FEM to solve Laplace's Equation with an initial Neumann Boundary

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:

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

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

11

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%.

$$c(x) = \frac{e^3}{e^6 - 1}(3e^{3x} - 3e^{-3x}) \tag{35}$$
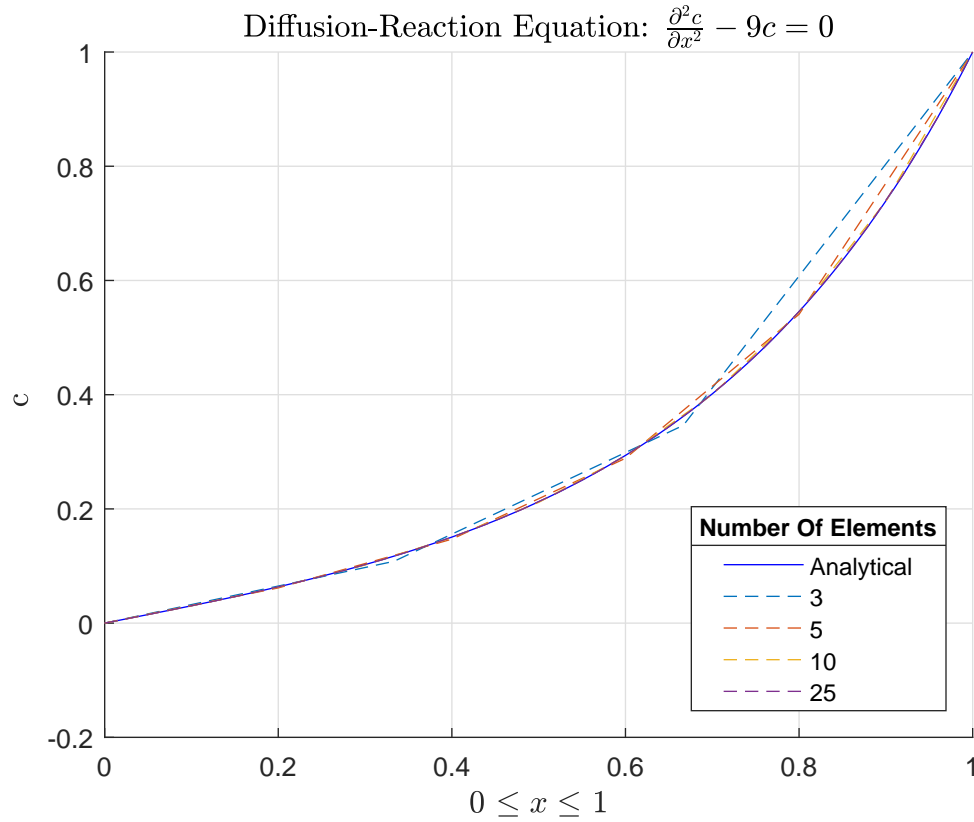
Figure 7: Using the FEM to solve the Static Diffusion-Reaction Equation with Dirichlet Boundary Conditions

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.
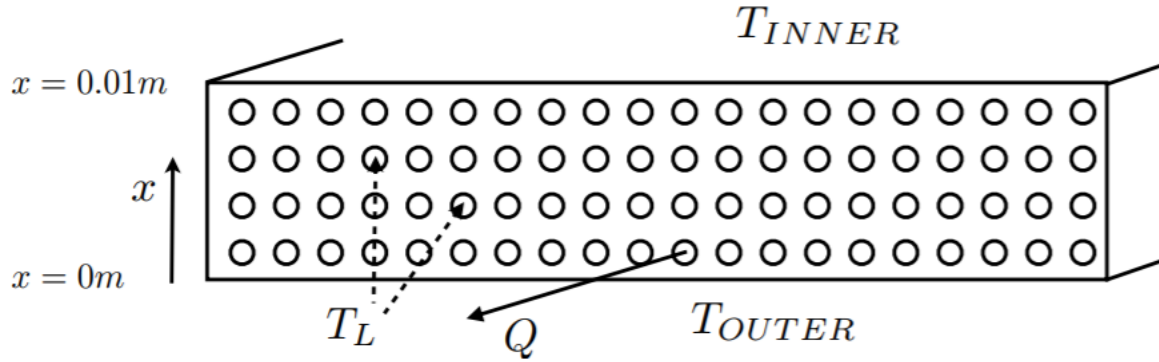
Figure 8: Cross Section of Material.

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

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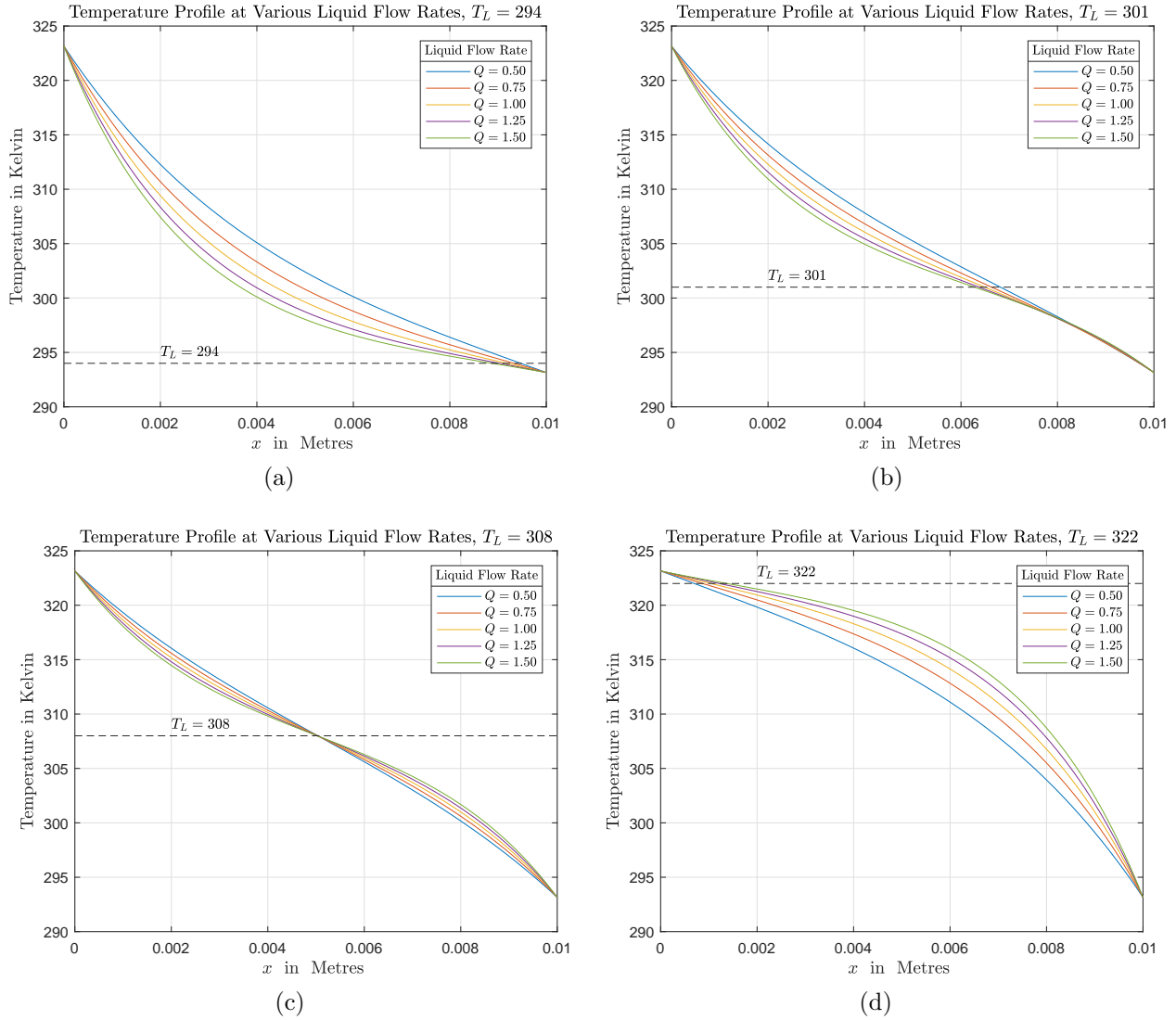
Figure 9: The Effect of Varying the Liquid Flow Rate on Temperature Distribution and Gradient

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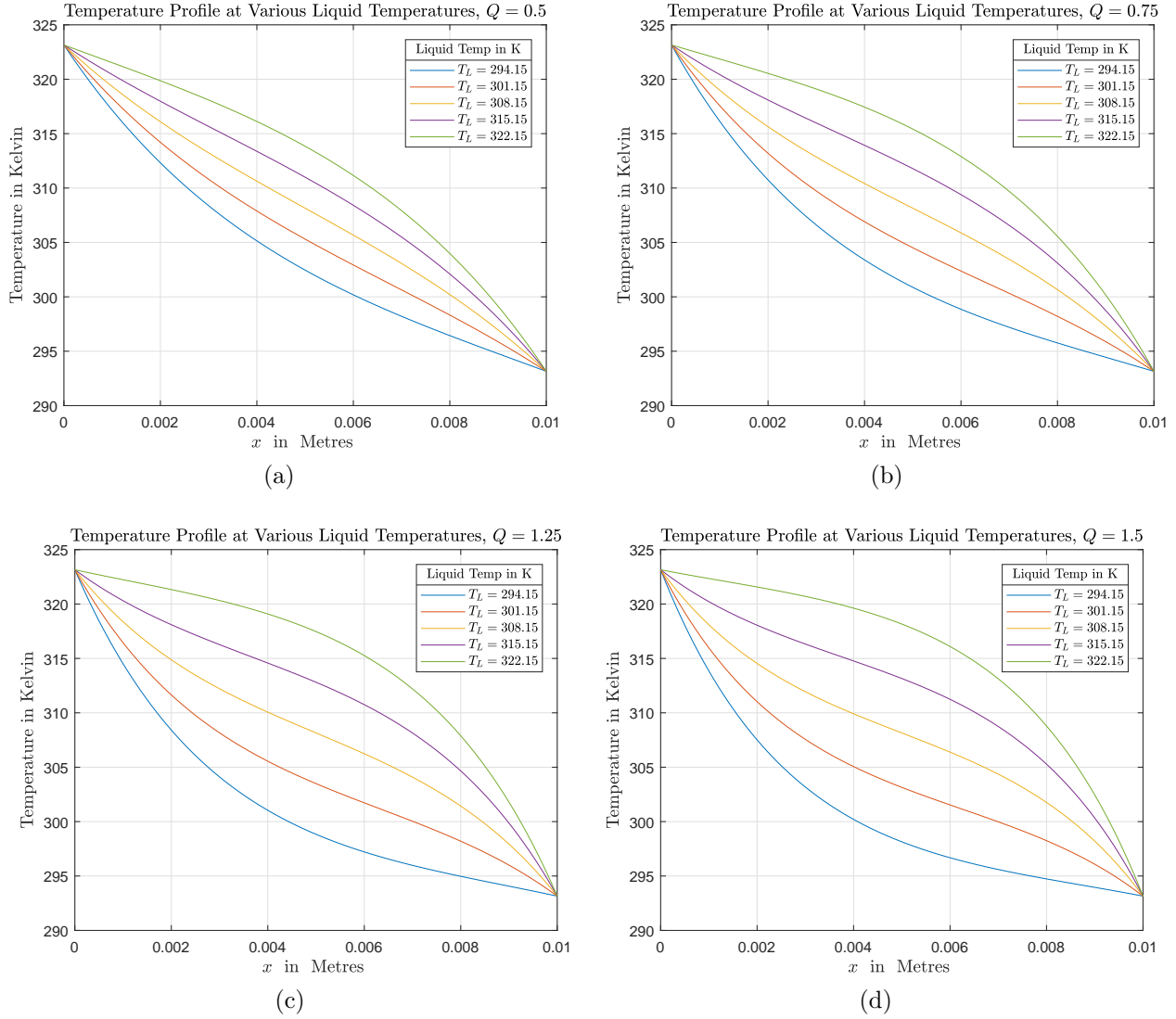
Figure 10: The Effect of Varying the Liquid Temperature on Temperature Distribution and Gradient

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.

15

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.
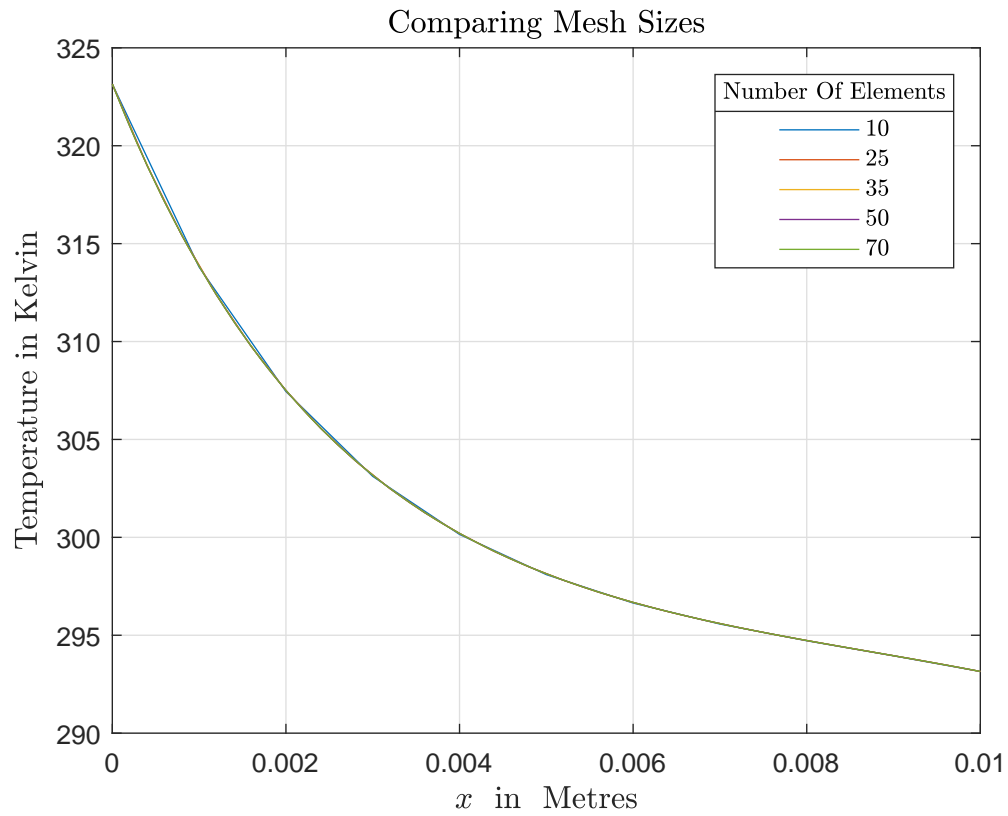
Figure 11: Cross Section of Material
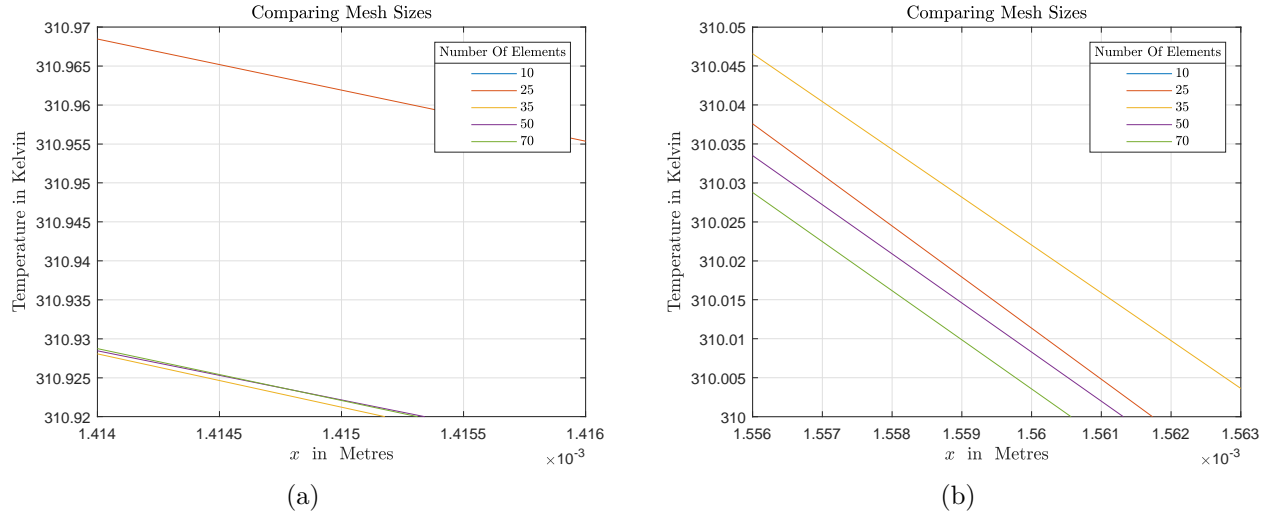
Figure 12: The Effect of Mesh Size on Temperature Resolution

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 39.

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

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

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

This can be split into the sum of the constant and linear term integrals as shown in equation 40.

$$\int_0^1 v\left[QT_L + 4QT_L x\right] dx = \int_0^1 v(QT_L)dx + \int_0^1 v(4QT_L x)dx \tag{40}$$

Dealing with just the linear integral and splitting into summation of elements gives the following.

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

Applying the Jacobi to map to the ζ domain and taking constants out of the integral gives equation 41.

$$\int_{x_0}^{x_1} v(4QT_L x)dx = 4QT_L J \int_{-1}^1 vx d\zeta \tag{41}$$

To solve this extra term it is necessary to use the basis function for x and v given by equations 5b and 6 respectively and repeated below.

$$x = x_0\psi_0(\zeta) + x_1\psi_1(\zeta)$$
$$v = \psi_0, \psi_1$$

As before this results a set of two equations given by equations 42a and 42b. These can be represented in the matrix form of equation 43.

$$4QT_L J \left[x_0 \int_{-1}^{1} \psi_0 \psi_0 d\zeta + x_1 \int_{-1}^{1} \psi_1 \psi_0 d\zeta \right) \right] \tag{42a}$$

$$4QT_L J \left[x_0 \int_{-1}^{1} \psi_0 \psi_1 d\zeta + x_1 \int_{-1}^{1} \psi_1 \psi_1 d\zeta \right) \right] \tag{42b}$$

$$4QT_L J \begin{bmatrix} Int_{00} & Int_{01} \\ Int_{10} & Int_{11} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} \tag{43}$$

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.

Int_{00}

$$\begin{aligned} Int_{00} &= \int_{-1}^{1} \psi_0 \psi_0 \ d\zeta \\ &= \int_{-1}^{1} \left(\frac{1-\zeta}{2} \right)^2 d\zeta \\ &= \left[\frac{1}{3} \left(\frac{1-\zeta}{2} \right)^3 (-2) \right]_{-1}^{1} \\ &= \frac{2}{3} \end{aligned} \tag{44}$$

$Int_{01} = Int_{10}$

$$\begin{aligned} Int_{00} &= \int_{-1}^{1} \psi_0 \psi_1 \ d\zeta \\ &= \int_{-1}^{1} \left(\frac{1-\zeta}{2} \right) \left(\frac{1+\zeta}{2} \right) d\zeta \\ &= \left[\frac{\zeta}{4} - \frac{\zeta^3}{12} \right]_{-1}^{1} \\ &= \left[\frac{1}{6} - \left(-\frac{1}{4} + \frac{1}{12} \right) \right]_{-1}^{1} \\ &= \frac{1}{3} \end{aligned} \tag{45}$$

$\underline{Int_{11}}$

$$
\begin{aligned}
Int_{00} &= \int_{-1}^{1} \psi_1 \psi_1 \ d\zeta \\
&= \int_{-1}^{1} \left(\frac{1+\zeta}{2} \right)^2 d\zeta \\
&= \left[\frac{1}{3} \left(\frac{1+\zeta}{2} \right)^3 .2 \right]_{-1}^{1} \\
&= \frac{2}{3}
\end{aligned}
\tag{46}
$$

Now substituting these results into the matrix form to get equation 47 for the local element vector linear source terms, f_L, at the local element nodes 0 and 1. This equation has been implemented in LinearSourceVector.m function available in Appendix I.2.

$$
\begin{bmatrix} f_{L_0} \\ f_{L_1} \end{bmatrix} = 4QT_L J \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}
\tag{47}
$$

As per equation 40 the local linear source vector must be added to the local constant source nodes created for the QT_L term, thus deriving the global source matrix needed for the FEM solver. In SourceVectorGen.m the constant source term and local linear elements are calculated and added to the global source vector in the correct place.

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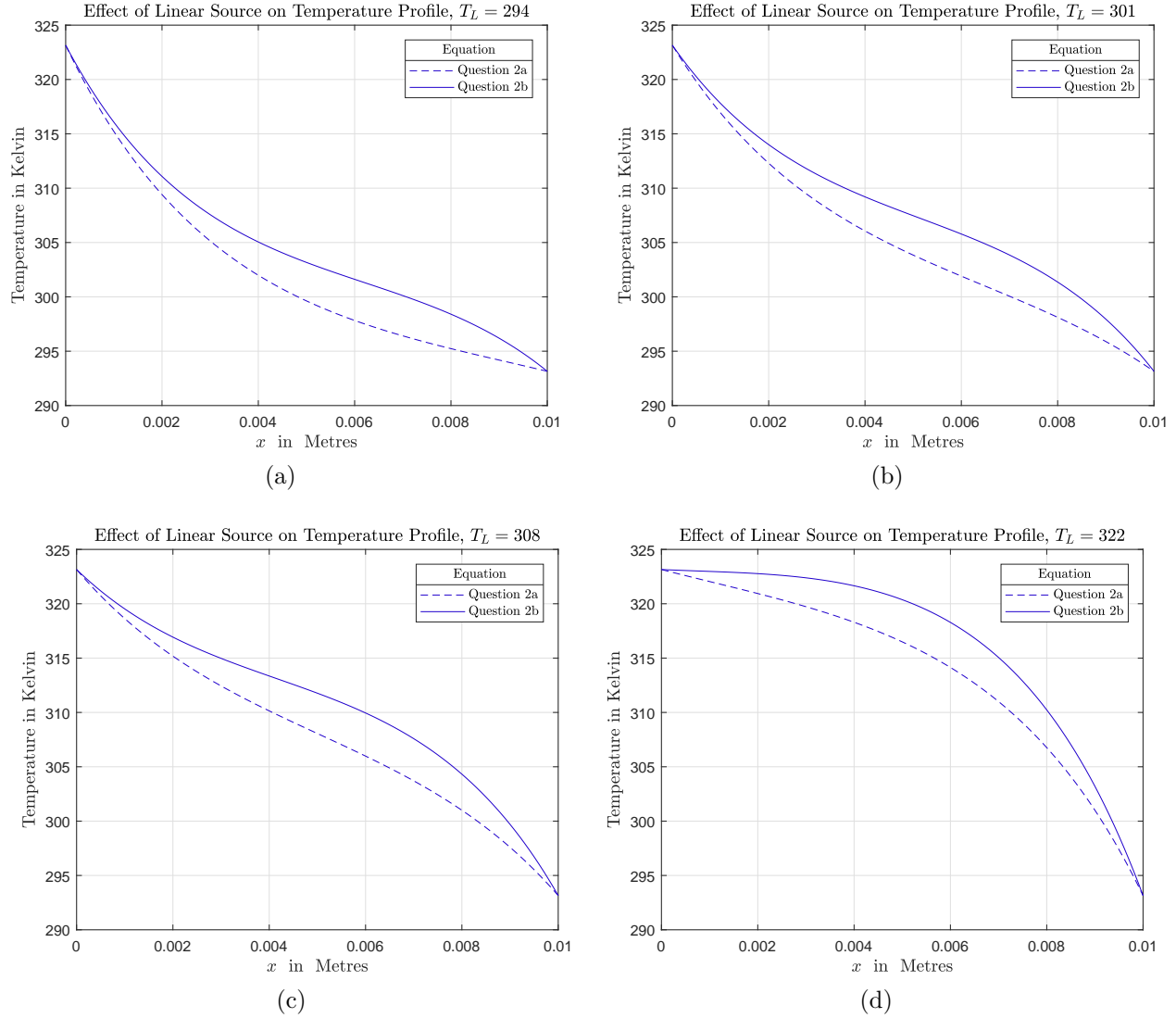
Figure 13: The Effect of Varying the Liquid Temperature on Temperature Distribution and Gradient ($Q = 1$)

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 Flow Chart of FEM Solver
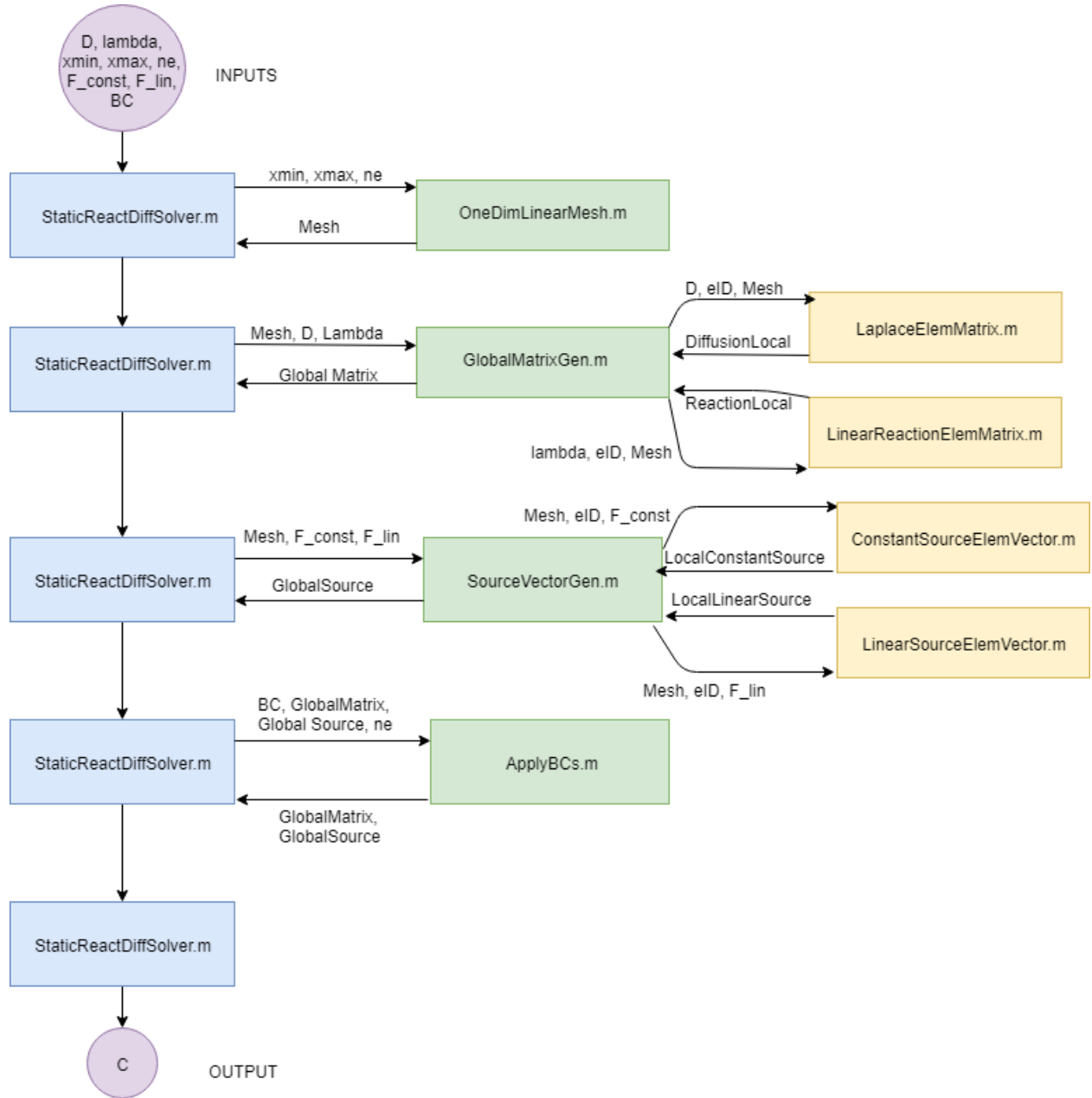
Figure 14: Flow Chart Showing The Process Followed by The FEM solver StaticReactDiffSolver.m

B   Laplace Element Matrix Code

```matlab
1  function [SqMatrix] = LaplaceElemMatrix(D, eID, msh)
2
3  %Returns the local 2x2 element matrix of a given element for a given diffusion
4  %coefficient
5  %
6  % Inputs:
7  % D - Coefficient of Diffusion
8  % eID - Index of element within mesh structure
9  % msh - Mesh which contains local elements within it's structure
10
11 %% Form of Laplace Elem matrix for J=1, D=1
12
13 SqMatrix = [0.5, -0.5; ...
14             -0.5, 0.5];
15
16 %% Multiply  by (1/J) and D to get solution for the particular element
17
18 J = msh.elem(eID).J;  %Get Jacobi for the element
19
20 SqMatrix = (1/J) * D * SqMatrix;   %Local element matrix for element eID
21 end
```

C CourseworkOneUnitTest.m

```matlab
1  %% Test 1: test symmetry of the matrix
2  % Test that this matrix is symmetric
3  tol = 1e-14;
4  D = 2; %diffusion coefficient
5  eID=1; %element ID
6  msh = OneDimLinearMeshGen(0,1,10);
7
8  elemat = LaplaceElemMatrix(D,eID,msh); %THIS IS THE FUNCTION YOU MUST WRITE
9
10 assert(abs(elemat(1,2) - elemat(2,1)) ≤ tol)
11
12 %% Test 2: test 2 different elements of the same size produce same matrix
13 % % Test that for two elements of an equispaced mesh, as described in the
14 % % lectures, the element matrices calculated are the same
15 tol = 1e-14;
16 D = 5; %diffusion coefficient
17 eID=1; %element ID
18 msh = OneDimLinearMeshGen(0,1,10);
19
20 elemat1 = LaplaceElemMatrix(D,eID,msh);%THIS IS THE FUNCTION YOU MUST WRITE
21
22 eID=2; %element ID
23
24 elemat2 = LaplaceElemMatrix(D,eID,msh);%THIS IS THE FUNCTION YOU MUST WRITE
25
26 diff = elemat1 - elemat2;
27 diffnorm = sum(sum(diff.*diff));
28 assert(abs(diffnorm) ≤ tol)
29
30 %% Test 3: test that one matrix is eveluted correctly
31 % % Test that element 1 of the three element mesh problem described in the lectures
32 % % the element matrix is evaluated correctly
33 tol = 1e-14;
34 D = 1; %diffusion coefficient
35 eID=1; %element ID
36 msh = OneDimLinearMeshGen(0,1,3);
37
38 elemat1 = LaplaceElemMatrix(D,eID,msh);%THIS IS THE FUNCTION YOU MUST WRITE
39
40 elemat2 = [ 3 -3; -3 3];
41 diff = elemat1 - elemat2; %calculate the difference between the two matrices
42 diffnorm = sum(sum(diff.*diff)); %calculates the total squared error between ...
       the matrices
43 assert(abs(diffnorm) ≤ tol)
```

D   Linear Reaction Element Matrix Code

```matlab
1  function [SqMatrix] = LinearReactionElemMatrix(lambda, eID, msh)
2
3  %Returns a local 2x2 element matrix of a given element for a diffusion
4  %operator
5  % Inputs:
6  % lambda - Scalar Coefficient of Diffusion
7  % eID - Index of element within mesh structure
8  % msh - Mesh which contains local elements within it's structure
9
10 %% Form of Linear Reaction local element matrix for J = 1 and lambda = 1
11 SqMatrix = [(2/3), (1/3); ...
12             (1/3), (2/3)];
13
14 %% Multiply by J and lambda to get solution for the particular element
15
16 J = msh.elem(eID).J;  %Get Jacobi for the element
17
18 SqMatrix = J * lambda .* SqMatrix;   %Local Linear Reaction matrix for element eID
19 end
```

E   Linear Reaction Element Matrix Test

```
1  %Testing LinearReactionElemMatrix.m creates local reation element vector correctly
2
3  %% Test 1: test symmetry of the matrix
4  % Test that this matrix is symmetric
5  tol = 1e-14;
6  lambda = 2; %diffusion coefficient
7  eID=1; %element ID
8  msh = OneDimLinearMeshGen(0,1,10);  %generate mesh using given funciton
9
10 elemat = LinearReactionElemMatrix(lambda,eID,msh); %Calculate Linear Element Matrix
11
12 assert(abs(elemat(1,2) - elemat(2,1)) ≤ tol && abs(elemat(1,1) - elemat(2,2))  ≤...
         tol, ...
13     'Local element matrix is not symmetric!')
14 %% Test 2: test 2 different elements of the same size produce same matrix
15 %Test that for two elements of an equispaced mesh, as described in the
16 %lectures, the element matrices calculated are the same
17 tol = 1e-14;
18 lambda = 5; %diffusion coefficient
19 eID=1; %element ID
20 msh = OneDimLinearMeshGen(0,1,10);
21
22 elemat1 = LinearReactionElemMatrix(lambda,eID,msh); %calculate element 1 matrix
23
24 eID=2; %element ID
25
26 elemat2 = LinearReactionElemMatrix(lambda,eID,msh); %calculate element 2 matrix
27
28 diff = elemat1 - elemat2;
29 diffnorm = sum(sum(diff.*diff));
30 assert(abs(diffnorm) ≤ tol)
31
32 %% Test 3: test that one matrix is evaluted correctly
33 % % Test that element 1 of the three element mesh problem described
34 % in Tutorial 3 Question 2c has the element matrix evaluated correctly
35 tol = 1e-14;
36 lambda = 1; %diffusion coefficient
37 eID=1; %element ID
38 msh = OneDimLinearMeshGen(0,1,6);
39
40 elemat1 = LinearReactionElemMatrix(lambda,eID,msh); %calculate element 1 matrix
41
42 elemat2 = [ (1/18), (1/36); (1/36), (1/18)];    % This is the known result
43 diff = elemat1 - elemat2; %calculate the difference between the two matrices
44 SummedDiff = sum(sum(diff)); %calculates the sum of the elements in the diff matrix
45 assert(abs(SummedDiff) ≤ tol) %Checks the error of the summed differences is ...
        below tol
46
47 %% Test 4: test main diagonal values are double antidiagonal values
48 % using random inputs for number of elemnts and lambda
49 tol = 1e-14;
```

27

```
50  lambda = 5*abs(rand(1,1)); %returns random diffusion coefficient between 0 and 4
51  eID=1; %element ID
52  ne = randi([4 100], 1,1);      %sets number of elements to random integer ...
        between 4 and 100
53  msh = OneDimLinearMeshGen(0,1,ne);
54
55  elemat1 = LinearReactionElemMatrix(lambda,eID,msh);
56  elem = elemat1(1,1)/elemat1(2,1); %Gets ratio between 1,1 and 2,1 elements
57  assert(abs(elem - 2) <= tol)      %Checks ratio is 2 within the tolerance
```

F Apply Boundary Condition Code

```matlab
1  function [GlobalMatrix, F] = ApplyBCs(BC, GlobalMatrix, F, ne)
2  % Applies boundary conditions to F and Global Matrix as appropriate
3  % for the specified boundary condition
4  %
5  %Inputs:
6  %BC - Structure which contains the following:
7  %       BC(1) - Holds data for minimum x boundary
8  %       BC(2) - Holds data for maximum x boundary
9  %       BC().type - Type of boundary condition: "neumann", "dirichlet" or
10 %                   "none" (must be a lower case string)
11 %       BC().value - Value of the boundary condition (float or int)
12 %
13 %GlobalMatrix - formation of local element matrices (NxN matrix)
14 %F - Source Vector (size Nx1 vector)
15
16 %% If user inputted character arrays change these to strings
17 BC(1).type = string(BC(1).type);
18 BC(2).type = string(BC(2).type);
19
20 %% Solve xmin Boundary Condition
21 if BC(1).type == "none" %No action needed if no boundary condition
22     % pass
23
24 elseif BC(1).type == "dirichlet"  %solve for a dirichlet BC
25
26     %set all first row elements to 0 except first element
27     GlobalMatrix(1,:) = [1, zeros(1,ne)];
28     %set first element of the source vector to the xmin BC value
29     F(1) = BC(1).value;
30
31 elseif  BC(1).type == "nuemann"    %solve for a nuemann BC
32
33     %subtract the xmin BC value from the first element of the source vector
34     F(1) = F(1) - BC(1).value;
35 end
36
37 %% Solve xmax Boundary Condition
38 if BC(2).type == "none" %No action needed if no boundary condition
39     % pass
40
41 elseif BC(2).type == "dirichlet"  %solve for dirichlet BC
42
43     %set all first row elements to 0 except last element
44     GlobalMatrix(end,:) = [zeros(1,ne), 1];
45     %set last element of the source vector to the xmax BC value
46     F(end) = BC(2).value;
47
48 elseif BC(2).type == "nuemann"    %solve nuemann BC
49
50     %subtract the xmin BC value from the last element of the source vector
51     F(end) = F(end) + BC(2).value;
```

```
52  end
53  end
```

G   Static Diffusion-Reaction Solver Code

```matlab
1  function [mesh] = StaticReactDiffSolver(D, lambda, xmin, xmax, ne,f_constant, ...
       f_linear,  BC)
2  %%% Solves the static diffusion-reaction equation
3  % Inputs:
4  % lambda - Coefficient for Reaction (float ot int)
5  % D - Coefficient of Diffusion (float or int)
6  % xmin - Minimum value for x, usually 0(float or int)
7  % xmax - Maximum value for x, usually 1(float or int)
8  % ne - Number of Elements in Mesh (float)
9  % f_constant - Constant Source Term (float or int)
10 % f_linear - Linear Source Term (float or int)
11 %BC - Structure which contains the following:
12 %       BC(1) - Holds data for minimum x boundary
13 %       BC(2) - Holds data for maximum x boundary
14 %       BC().type - Type of boundary condition: "neumann", "dirichlet" or
15 %                   "none" (must be a lower case string)
16 %       BC().value - Value of the boundary condition (float or int)
17
18 mesh = OneDimLinearMeshGen(xmin, xmax, ne); %Generate Mesh
19
20 %% Generate the Global Matrix for Diffusion and Reaction terms
21 GlobalMatrix = GlobalMatrixGen(mesh, D, lambda);
22
23 %% Generate the Global source the vector
24 F = SourceVectorGen(mesh, f_constant, f_linear);
25
26 %% Apply Boundary Conditions to the Global Matrix and Global Source Vector
27 [GlobalMatrix, F] = ApplyBCs(BC, GlobalMatrix, F, ne);
28
29 %% Solve for C
30 C = GlobalMatrix \ F;        %Returns a vector of C values
31
32 %Add values of C into the mesh structure
33 mesh.c = C;
34
35 end
```

H Global Matrix Generator Code

```matlab
 1  function [GlobalMatrix] = GlobalMatrixGen(mesh, D, lambda)
 2  %Returns Global Matrix of Difusion and Reaction terms
 3  % Inputs:
 4  % lambda - Coefficient for Reaction (float ot int)
 5  % D - Coefficient of Diffusion (float or int)
 6
 7
 8  %% Initiate Global Matrix
 9  ne = mesh.ne;        %Number of Elements
10  GlobalMatrix = zeros((ne+1),(ne+1));
11  %% Loop over Elements and Assemble Local Element Matrices into Global Matrix
12  for eID = 1:ne
13
14      %Calculate Local Element Matrix for Diffusion Term
15      DiffusionLocal = LaplaceElemMatrix(D, eID, mesh);
16
17      %Calculate Local Element Matrix for Linear Reaction Term
18      ReactionLocal = LinearReactionElemMatrix(lambda, eID, mesh);
19
20      %Local Element Matrix is the Diffusion subtract the Linear Reation
21      LocalElementMatrix = DiffusionLocal - ReactionLocal;
22
23      %Add Local Element Matrix into the correct location within the Global Matrix
24      GlobalMatrix(eID:(eID+1),eID:(eID+1)) = ...
              GlobalMatrix(eID:(eID+1),eID:(eID+1))...
25                                          + LocalElementMatrix;
26  end
```

I Source Vector Generator Code

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

ne = mesh.ne;   %Number of elements

%% Initalise Gloabal Source Vector F
F = zeros((ne+1),1);      %Initialise Source Vector

%% Add in the Constant and Linear Element Vectors and Add to F
for eID = 1:ne

    %Caluculate Constant Source Local Element Vectors
    LocalConstantSource = ConstantSourceElemVector(mesh, eID, f_constant);
    %Caluculate Linear Source Local Element Vectors
    LocalLinearSource = LinearSourceElemVector(mesh, eID, f_linear);

    %Add Linear and Constant Vectors to get Local Source Vector
    LocalSourceVector = LocalLinearSource + LocalConstantSource;

    %Add Local Source Vector into Global Source Vector at correct location
    F(eID:(eID+1)) = F(eID:(eID+1)) + LocalSourceVector;
end
```

I.1 Constant Source Element Vector Code

```matlab
function [ElemVector] = ConstantSourceElemVector(mesh, eID, f_constant)
% Returns the Local Constant 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_const - Constant source term

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

ElemVector = [f_constant*J; f_constant*J];  %each term is fJ

end
```

I.2   Linear Source Element Vector Code

```matlab
1  function [ElemVector] = LinearSourceElemVector(mesh, eID, f_linear)
2  % Returns the Local Linear Source Vector for an element
3  %
4  % Inputs:
5  % mesh - Mesh which contains local elements within it's structure and
6  %           related variables
7  % eID - index for the element within the mesh
8  % f_linear - Linear source term multiplier
9
10 %% Extract element variables from mesh
11 J =mesh.elem(1).J;       %Get Jacobi for the element
12 x0 = mesh.elem(eID).x(1);   %Get he x0 value for the element
13 x1 = mesh.elem(eID).x(2);   %Get he x1 value for the element
14 xVector = [x0;x1];          %Create a vector of the x values
15
16 %% Local Source Vector with J =1 and f_linear = 1
17 StandardMatrix = [(2/3), (1/3);...
18                   (1/3), (2/3)];
19
20 %% Multiply by J, f_linear and the x vector to get Local Linear Source
21 %Vector for the specific element
22 ElemVector = J * f_linear *  StandardMatrix * xVector;
23 end
```

J Additional Software Verification

J.1 Global Matrix Test

```
1  %Testing GlobalMatrixGen.m creates global matrix correctly
2
3  %% Test 1: test correct number of elements are non zero
4  % formula for number of non zeros is 3(n+1) - 2
5
6  ne = 24;     %number of mesh elements
7  mesh = OneDimLinearMeshGen(0,1,ne); %Generate 20 element mesh
8
9  %Calculate Analytical number of zeros
10 AnNumNonZeros = 3*(ne+1) - 2;
11 %Set Arbitary values for D and lambda
12 D = 3;
13 lambda = 5.4;
14 GlobalMatrix = GlobalMatrixGen(mesh, D, lambda);
15 NumNonZeros = nnz(GlobalMatrix); %sum index to get number of zero elements
16
17 assert( NumNonZeros == AnNumNonZeros);
18
19 %% Test 2: Check first two elements of the first row are the same as the
20 % last two elements of the last row
21 tol = 1e-14;
22 mesh = OneDimLinearMeshGen(0,1,10); %Generate 10 element mesh
23 %Set Arbitary values for D and lambda
24 D = 3;
25 lambda = 5.4;
26 %Generate global matrix
27 GlobalMatrix = GlobalMatrixGen(mesh, D, lambda);
28
29 %Get first row first two elements
30 FirstRowValues = GlobalMatrix(1,1:2);
31 %Get last row last two elements
32 LastRowValues = GlobalMatrix(end,(end-1):end);
33 %Subrtact elements and sum to get total difference
34 diff = sum(FirstRowValues - LastRowValues);
35
36 assert(abs(diff) < tol)
37
38 %% Test 3: Check non zero values of any two rows are the same as the
39 % excluding first and last row
40 tol = 1e-14;
41 mesh = OneDimLinearMeshGen(0,1,10); %Generate 10 element mesh
42 %Set Arbitary values for D and lambda
43 D = 3;
44 lambda = 5.4;
45 %Generate global matrix
46 GlobalMatrix = GlobalMatrixGen(mesh, D, lambda);
47
48 %Get third row non zero elements
49 ThirdRowValues = GlobalMatrix(3,3:6);
```

```
50  %Get fith row non zero elements
51  FithRowValues = GlobalMatrix(5,5:8);
52  %Subrtact elements and sum to get total difference
53  diff = sum(ThirdRowValues - FithRowValues);
54
55  assert(abs(diff) < tol)
```

J.2 Application of Boundary Conditions Test

```
1   %Testing ApplyBCs.m applies boundary conditions correctly
2
3   %% Test 1: Test Dirichlet boundary at xmin sets all but first element of
4   % global matrix to zero
5
6   D = 1;
7   lambda = 0;
8   ne = 6;
9   f_constant = 0;
10  xmin = 0;
11  xmax = 1;
12  %Create Boundary Condition Stucture
13  BC(1).type = "dirichlet";
14  BC(1).value = 2;
15  BC(2).type = "none";
16  f_linear = 0;
17  %Generate Mesh
18  mesh = OneDimLinearMeshGen(xmin,xmax,ne);
19  %Generate Global Matrix
20  GlobalMatrix = GlobalMatrixGen(mesh, D, lambda);
21  %Generate source vector
22  F = SourceVectorGen(mesh, f_constant, f_linear);
23
24  %Apply Boundary Condition
25  [GlobalMatrix, F] = ApplyBCs(BC, GlobalMatrix, F, ne);
26
27  assert(GlobalMatrix(1,1) ≠ 0 && sum(GlobalMatrix(1, 2:end)) ==0)
28
29  %% Test 2: Test for parameters in Test 1 the first source term is equal to
30  % Boundary Condition value and all other elements are zero
31
32  %Check number of non zero elements in F is 1
33  NumNonZeroF = nnz(F);% Gets number of non zero values in F
34
35  assert(F(1) == BC(1).value && NumNonZeroF == 1)
36
37  %% Test 3: Test that global matrix is unchanged for two nuemann BCs
38  D = 2;
39  lambda = 0;
40  ne = 8;
41  f_constant = 1;
42  xmin = 0;
43  xmax = 1;
```

```matlab
44  %Create Boundary Condition Stucture
45  BC(1).type = "neumann";
46  BC(1).value = 2;
47  BC(2).type = "nuemann";
48  f_linear = 0;
49
50  %Generate Mesh
51  mesh = OneDimLinearMeshGen(xmin,xmax,ne);
52  %Generate Global Matrix
53  GlobalMatrixIn = GlobalMatrixGen(mesh, D, lambda);
54  %Generate source vector
55  F = SourceVectorGen(mesh, f_constant, f_linear);
56
57  %Apply the BCs
58  [GlobalMatrixOut, F] = ApplyBCs(BC, GlobalMatrixIn, F, ne);
59
60  assert(isequal(GlobalMatrixOut, GlobalMatrixIn))
```

K Part 1 Results

```matlab
1  %%% Generate Results for Part 1 Qustions
2  clear
3  clc
4
5  %% Question 1c - Solving Laplaces Equation:
6  %Set Parameter from the question
7  D = 1;      %Coefficient of Diffusion
8  lambda = 0; %Coefficient of Reaction
9  ne = 4;     %Number of elements
10 f_constant = 0; %No source
11 f_linear = 0;   %No source
12 xmin = 0;       %Set x domain from x=0 to x=1
13 xmax = 1;
14 %Create Boundary Condition Stucture
15 BC(1).type = "dirichlet";
16 BC(1).value = 2;
17 BC(2).type = "dirichlet";
18 BC(2).value = 0;
19
20 %Use the FEM to obtain a solution for C, solution stored as mesh.c
21 mesh = StaticReactDiffSolver(D, lambda, xmin, xmax, ne, f_linear, f_constant, BC);
22
23 %Create plot points for analytical solution
24 xvec = 0:0.25:1;        %Same number of nodes/points as FEM method
25 c = zeros(length(xvec),1);   %Initialise C
26 % Use the equation given in the question to calculate c values
27 for i = 1:length(xvec)
28     c(i) = 2*(1-xvec(i));
29 end
30
31 %Plot the results
32 figure()
33 plot(xvec, c, '-xb',mesh.nvec, mesh.c, 'ro');
34 grid on
35 title('Laplace Equation $\frac{\partial^2 c}{\partial x^2} = 0$', 'interpreter' ...
       ,'latex', 'FontSize', 12)
36 xlim([0 1]);
37 ylim([0 2.5]);
38 ylabel('c ', 'interpreter','latex', 'FontSize', 12);
39 xlabel('0 $\leq x  \leq$ 1','interpreter','latex', 'FontSize', 12);
40 legend({'Analytical','FEM'}, 'Interpreter', 'latex')
41 xticks(0:.25:1)
42 print -depsc '\Users\xav_m\OneDrive\Documents\XAVI\University\Final ...
       Year\Systems Mod\Modeling_Techniques_CW1\Report\Figures\epsLaplaceFig1'
43
44
45 %Now solve with Neumann at x = 0:
46 %Nuemann Boundary
47 BC(1).type = "nuemann";
48 BC(1).value = 2;
49 %Dirichlet Boundary
```

```matlab
50  BC(2).type = "dirichlet";
51  BC(2).value = 0;
52  %Solve for new boundary conditions
53  mesh = StaticReactDiffSolver(D, lambda, xmin, xmax, ne, f_linear, f_constant, BC);
54
55  %Plot the result
56  figure()
57  plot(mesh.nvec, mesh.c, '-x');
58  title('Laplace Equation with Neumann Boundary at $x = 0$', 'interpreter' ...
         ,'latex', 'FontSize', 12)
59  grid on
60  xticks(0:.25:1)
61  xlabel('0 $\leq x  \leq$ 1','interpreter','latex', 'FontSize', 12);
62  ylabel('c ', 'interpreter','latex', 'FontSize', 12);
63  ylim([-2.5 0]);
64  print -depsc '\Users\xav_m\OneDrive\Documents\XAVI\University\Final ...
         Year\Systems Mod\Modeling_Techniques_CW1\Report\Figures\epsLaplaceFig2'
65
66
67  %% QUESTION 1d - Check Reaction Terms are Solved Correctly
68  %Set up Parameters from the question
69  D = 1;
70  lambda = -9;
71  BC(1).type = "dirichlet";
72  BC(1).value = 0;
73  BC(2).type = "dirichlet";
74  BC(2).value = 1;
75
76  %Analytical solution
77  xvec = 0:(1/100):1;
78  for i = 1:length(xvec)
79      c(i) = (exp(3)/(exp(6)-1)) * (exp(3*xvec(i)) - exp(-3*xvec(i)));
80  end
81
82  %Plot the results of FEM and analytical
83  figure()
84  grid on
85  plot(xvec, c, '-b', 'LineWidth', .5)    %Plot analytical solution
86  hold on
87
88  %Now solve using FEM method with several mesh sizes
89  elem_range = [3,5,10,25];   %Mesh sizes to be used
90  %Solve for each mesh size and plot
91  for i = 1:length(elem_range)
92      ne = elem_range(i);
93      mesh = StaticReactDiffSolver(D, lambda, xmin, xmax, ne, f_linear, ...
             f_constant, BC);
94      plot(mesh.nvec, mesh.c, '--')
95      hold on
96  end
97  %Format the result
98  title('Diffusion-Reaction Equation: $\frac{\partial^2 c}{\partial x^2} -9c = ...
         0$', 'interpreter' ,'latex', 'FontSize', 12)
99  lgd = legend('Analytical', num2str(elem_range(1)), num2str(elem_range(2)), ...
         num2str(elem_range(3)) , num2str(elem_range(4)), 'Location', 'southeast');
```

```matlab
100  lgd.Title.String = 'Number Of Elements';
101  xlabel('0 $\leq x  \leq$ 1','interpreter','latex', 'FontSize', 12);
102  ylabel('c ', 'interpreter','latex', 'FontSize', 12);
103  %Save the Figure
104  print -depsc '\Users\xav_m\OneDrive\Documents\XAVI\University\Final ...
         Year\Systems Mod\Modeling_Techniques_CW1\Report\Figures\epsReactDiff1'
105  hold off
```

L Part 2 Results

L.1 Question 2a Results Script

```matlab
%%%% Part 2a: Modelling and Simultion Results
% Use FEM solver to solve k.(d^2T/dx^2) + Q(T_L - t) = 0
clear
clc
%% Set parameters given in the question
xmin = 0;    %Min x in metres
xmax = .01; % Max x in metres
ne = 70;     %Number of elements
k = 1.01e-5;    % Coefficient of diffusion
%Set Boundary Conditions:
BC(1).type = "dirichlet";
BC(1).value = 323.15;    %Temperature in Kelvin
BC(2).type = "dirichlet";
BC(2).value = 293.15;        %Temperature in Kelvin
%No linear source for part a:
f_linear = 0;

%Initialise counter
i = 1;

%% Create four plots at different T_L values showing the effect of Q
for T_L = 294.15:7:322.15

    figure(i)
    i = i+1;
    %Vary Q from 0.5 to 1.5 and plot the variation of results
    for Q = 0.5:0.25:1.5

        %Calculate f_scalar for boundary condition
        f_scalar = Q*T_L;
        %Solve temperature using FEM
        mesh = StaticReactDiffSolver(k, -Q, xmin, xmax, ne, f_scalar, f_linear, ...
            BC);
        %Plot result
        plot(mesh.nvec, mesh.c)
        hold on

    end
    %Format Figure:
    title(strcat('Temperature Profile at Various Liquid Flow Rates, $T_L = ', ...
        num2str(floor(T_L)), '$'), 'interpreter' ,'latex', 'FontSize', 12)
    lgd = legend({'$Q = 0.50$', '$Q = 0.75$', '$Q = 1.00$', '$Q = 1.25$' , '$Q ...
        = 1.50$'},'Location', 'northeast', 'interpreter', 'latex');
    lgd.Title.String = 'Liquid Flow Rate';
    xlabel('$x$ \ in \ Metres ','interpreter','latex', 'FontSize', 12);
    ylabel('Temperature in Kelvin', 'interpreter','latex', 'FontSize', 12);
    %Plot T_L to help with visualisation of the result:
    plot([0 0.01] , [T_L T_L], 'LineStyle','--', 'Color', [.17 .17 .17], ...
        'HandleVisibility','off')
```

```matlab
46      text(0.002, (T_L + 1), strcat('$T_L = ', num2str(T_L), '$'), 'interpreter', ...
            'latex')
47      grid on
48      %Save the figure:
49      str = strcat("\Users\xav_m\OneDrive\Documents\XAVI\University\Final ...
            Year\Systems Mod\Modeling_Techniques_CW1\Report\Figures\epsVaryQ", ...
            num2str(floor(T_L)));
50      print(str, '-depsc')
51      hold off
52  end
53
54  %% Now we vary T_L in the plots for 4 different Q values
55  i1=0; %new counter for indexing saved results
56  %Create a plot for each value of Q
57  for Q = 0.5:0.25:1.5
58
59      i1 = i1+1;
60      figure()
61      %Vary T_L and plot the variation of results
62      for T_L = 294.15:7:322.15
63
64          %Calculate f_scalar for boundary condition
65          f_scalar = Q*T_L;
66          disp(f_scalar)
67          %Solve temperature using FEM
68          mesh = StaticReactDiffSolver(k, -Q, xmin, xmax, ne, f_scalar, f_linear, ...
                BC);
69          %Plot the results
70          plot(mesh.nvec, mesh.c)
71          hold on
72
73      end
74      %Format the Figure:
75      title(strcat('Temperature Profile at Various Liquid Temperatures, $Q = ', ...
            num2str(Q), '$'), 'interpreter' ,'latex', 'FontSize', 12)
76      lgd = legend({'$T_L = 294.15$', '$T_L = 301.15$', '$T_L = 308.15$', '$T_L = ...
            315.15$' , '$T_L = 322.15$'},'Location', 'northeast', 'interpreter', ...
            'latex');
77      lgd.Title.String = 'Liquid Temp in K';
78      xlabel('$x$ \ in \ Metres ','interpreter','latex', 'FontSize', 12);
79      ylabel('Temperature in Kelvin', 'interpreter','latex', 'FontSize', 12);
80      grid on
81      %Save the Figure
82      str = strcat("\Users\xav_m\OneDrive\Documents\XAVI\University\Final ...
            Year\Systems Mod\Modeling_Techniques_CW1\Report\Figures\epsVaryTL", ...
            num2str(i1));
83      print(str, '-depsc')
84      hold off
85  end
86
87  %% Investigate the effect of Mesh size
88  figure()
89  ne = [10, 25, 35, 50, 70];  %Range of Mesh sizes to investigate
90
91  for i = 1:length(ne)
```

```matlab
92       Q = 1.5;     %Choose extreme Q value
93       T_L = 294.15;    %Choose extreme T_L value
94       %Calculate f_scalar for boundary condition
95       f_scalar = Q*T_L;
96       %Solve temperature using FEM
97       mesh = StaticReactDiffSolver(k, -Q, xmin, xmax, ne(i), f_scalar, f_linear, BC);
98       %Plot the result
99       fig1 = plot(mesh.nvec, mesh.c);
100      hold on
101
102  end
103  %Format the Figure
104  xlabel('$x$ \ in \ Metres','interpreter','latex', 'FontSize', 12);
105  ylabel('Temperature in Kelvin', 'interpreter','latex', 'FontSize', 12);
106  title('Comparing Mesh Sizes', 'interpreter' ,'latex', 'FontSize', 12)
107  lgd = legend({num2str(ne(1)), num2str(ne(2)), num2str(ne(3)) , num2str(ne(4)), ...
         num2str(ne(5))}, 'interpreter', 'latex', 'Location', 'northeast');
108  lgd.Title.String = 'Number Of Elements';
109  grid on
110  %Save the Figure
111  print -depsc '\Users\xav_m\OneDrive\Documents\XAVI\University\Final ...
         Year\Systems Mod\Modeling_Techniques_CW1\Report\Figures\epsMesh01'
112
113  %Zoom in on a point where meshes have converged
114  xlim([1.414e-3 1.416e-3]);
115  ylim([310.92 310.97]);
116  yticks(310.92:.005:310.97)
117  grid on
118  %Save the Figure
119  print -depsc '\Users\xav_m\OneDrive\Documents\XAVI\University\Final ...
         Year\Systems Mod\Modeling_Techniques_CW1\Report\Figures\epsMesh11'
120
121  %Zoom in on point where meshes are close to maximum diverence
122  xlim([1.556e-3 1.563e-3]);
123  ylim([310 310.05]);
124  yticks(310:.005:310.05);
125  grid on
126  %Save the Figure
127  print -depsc '\Users\xav_m\OneDrive\Documents\XAVI\University\Final ...
         Year\Systems Mod\Modeling_Techniques_CW1\Report\Figures\epsMesh12'
```

L.2 Question 2b Results Script

```matlab
1  %Part 2b, investigating the linear source term
2  clear
3  clc
4  %% Set parameters given in the question
5  xmin = 0;
6  xmax = .01;
7  ne = 100;
8  k = 1.01e-5;
9  BC(1).type = "dirichlet";
10 BC(1).value = 323.15;    %Temperature in Kelvin
11 BC(2).type = "dirichlet";
12 BC(2).value = 293.15;        %Temperature in Kelvin
13
14 %% Set Q = 1 as other values are exaggerations of the same shape
15 Q = 1;
16
17 %%Plot four graphs at different values of T_L
18 color = [0.127, 0.0, 0.8];  %set color for the lines
19 for T_L = 294.15:7:322.15     %
20
21     figure() %initiate next figure
22
23     %Calculate f_scalar for boundary condition
24     f_constant = Q*T_L;
25     f_linear = 4*Q*T_L;
26
27     %Solve temperature using FEM with no Linear Source
28     mesh = StaticReactDiffSolver(k, -Q, xmin, xmax, ne, f_constant, 0, BC);
29     plot(mesh.nvec, mesh.c, '--', 'Color', color)
30     hold on
31     %Now solve including the Linear Source
32     mesh = StaticReactDiffSolver(k, -Q, xmin, xmax, ne, f_constant, f_linear, BC);
33     plot(mesh.nvec, mesh.c, '-', 'Color', color)
34
35     %% Format the figure
36     title(strcat('Effect of Linear Source on Temperature Profile, $T_L = ', ...
37         num2str(T_L), 'K$'), 'interpreter' ,'latex', 'FontSize', 12)
37     lgd = legend({'Question 2a', 'Question 2b'},'Location', 'northeast', ...
           'interpreter', 'latex');
38     lgd.Title.String = 'Equation';
39     xlabel('$x$ \ in \ Metres','interpreter','latex', 'FontSize', 12);
40     ylabel('Temperature in Kelvin', 'interpreter','latex', 'FontSize', 12);
41     grid on
42     %% Saving Figure
43     str = strcat("\Users\xav_m\OneDrive\Documents\XAVI\University\Final ...
           Year\Systems Mod\Modeling_Techniques_CW1\Report\Figures\epsLinear", ...
           num2str(floor(T_L)));
44     %print(str, '-depsc')
45     %% Clear figure
46     hold off
47 end
```