RESEARCH PROPOSAL (TASK 1)

TESTING OF FINITE ELEMENT JAVA CLASSES USED IN SOLVING TWO DIMENSIONAL HEAT CONDUCTION PROBLEMS

SUBMITTED TO:

DEPARTMENT OF MECHANICAL ENGINEERING

SUBMITTED BY:

DADA, AKINDOLU OLUWAKANYINSOLA

STUDENT NUMBER: 1050279

MSC IN ADVANCED MECHANICAL ENGINEERING

FEBRUARY 2011

AUTHOR'S DECLARATION

I declare that this work was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Taught Postgraduate Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, this work is my own work. Work done in collaboration with, or with the assistance of others, is indicated as such. I have identified all material in this dissertation which is not my own work through appropriate referencing and acknowledgement. Where I have quoted from the work of others, I have included the source in the references/bibliography. Any views expressed in the literature are those of the author.

SIGNED: ………………………………………………………………. DATE: ……………..

(Signature of student/candidate)

TABLE OF CONTENTS

Contents

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

This report discusses the tests carried out on the “simpleElements” finite element routine developed by Dr Mike Tierney. The program is used to solve heat conduction problems by employing the finite elements method; it uses the meshing function in the PDE Toolbox in MATLAB and then solves the conduction problem using the mesh data generated. The simple elements program calls java classes which it uses to carry out the computation.

The purposes of the activities reported are:

- To create a MATLAB script that would call and use the existing java classes and objects.
- To carry out simple analysis whose results would be checked against those obtained using the PDE Toolbox in MATLAB and analytical solutions.
- To determine errors in the program and its sources.
- To determine ways of avoiding such errors, or compensating for them.

The simple elements program calls the java classes whose source code is hidden, this is done in line with the object oriented paradigm which encourages abstraction, encapsulation, polymorphism and inheritance(Tierney 2011). The object oriented paradigm encourages the code reuse; it does so by creating classes based on existing classes, such that the new classes inherit the behavior of the parent class(Dietel and Dietel 2002).

These tests were carried out to verify the correct operation of the existing classes thereby ensuring that other classes which inherit the behavior of these classes would function correctly.

Seven tests are reported in this write-up, the first five considers results of the solution gotten from the simpleElements program and compares it with the result obtained from the PDE Toolbox of MATLAB, and also the analytical results.

The series of tests are carried out in order to determine if there are any errors in the code, trace the sources of the errors in a situation where they exist, and seek ways of avoiding them, eliminating them or compensating for such errors as much as possible.

TESTS

The tests carried out are documented in this section, the main purpose is to ensure that the program solution obtained using the objects created from the classes are accurate

TEST 1: TEST WITH TEMPERATURES

PROBLEM DESCRIPTION

This involves subjecting the north face of the rectangular element shown in Figure 1 below to a temperature of 70^oC and the south face to a temperature of 40^oC.

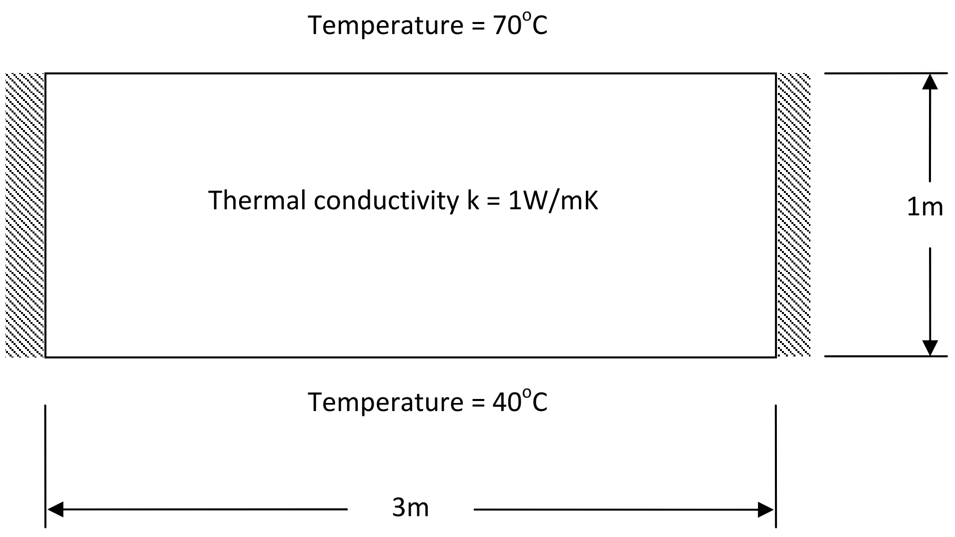


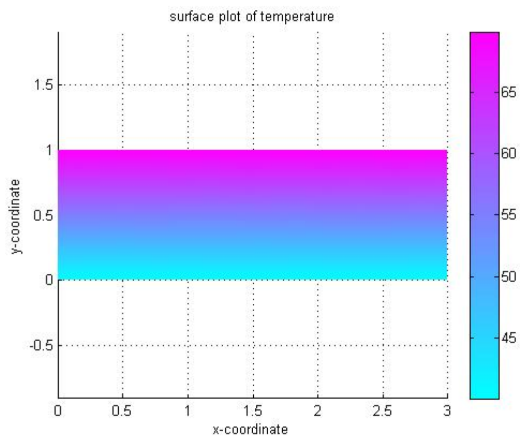
Figure 1: Problem description for Test 1

PROCEDURE

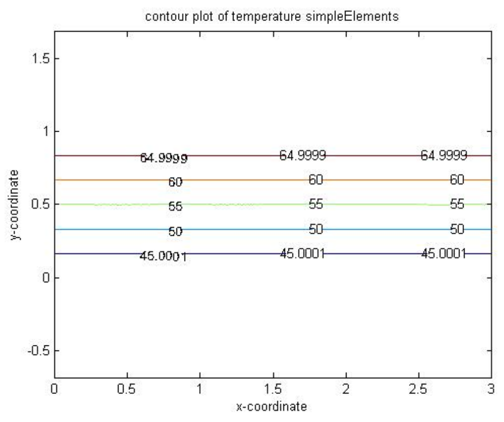
1. The problem was setup using the PDE Toolbox as shown in Figure 1 above, specifying the correct boundary conditions and the pde parameters in the PDE Toolbox
2. The mesh to be used for the solution was initialized
3. The problem was then solved using the PDE Toolbox
4. The boundary condition, geometry, point, edge, triangle and solution matrices were then exported to the workspace in MATLAB.
5. The exported matrices were saved with a unique name.
6. The MATLAB script shown in Appendix A was run to determine the temperatures at the various nodes of the mesh generated by the PDE Toolbox, produce surface plots of the temperature distribution over the entire surface, produce a plot of temperature against the y-coordinate of the various nodes and finally to determine the errors between the result obtained from the PDE Toolbox, analytical solution and that obtained using the simpleElements program.

RESULTS

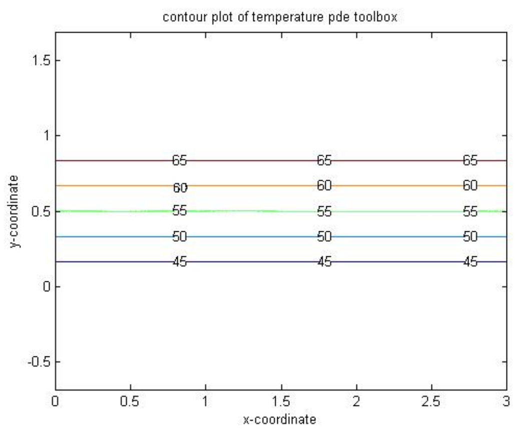
The results obtained using the simple elements program are shown below:



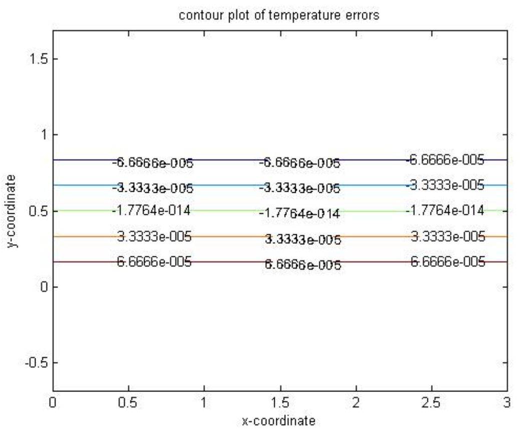
(a)



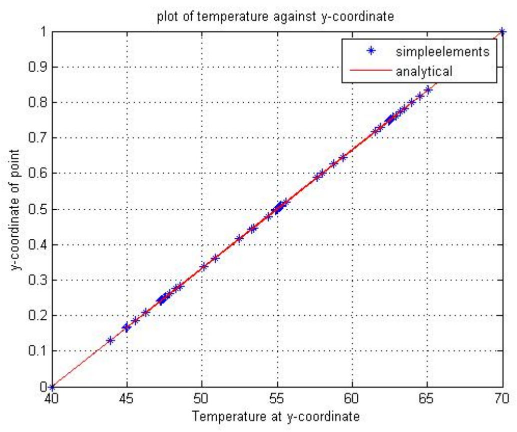
(b)



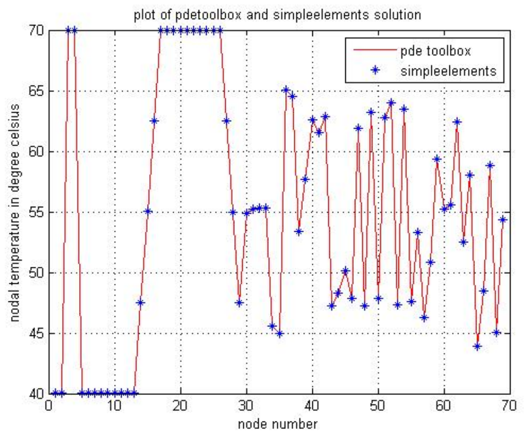
(c)



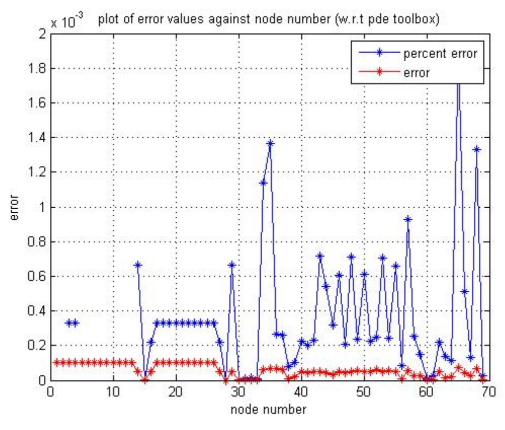
(d)

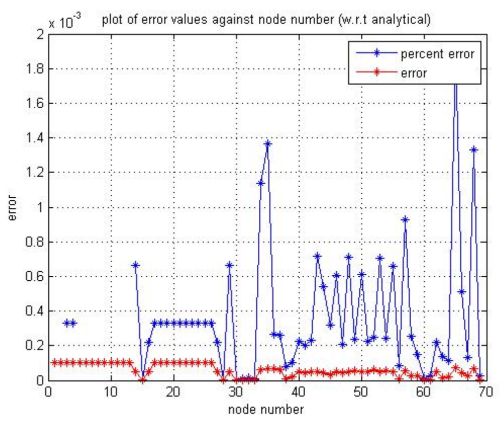


(e)



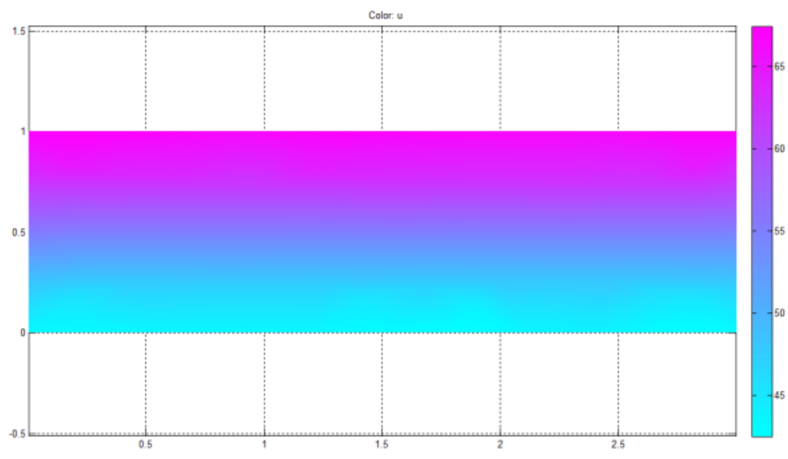
(f)



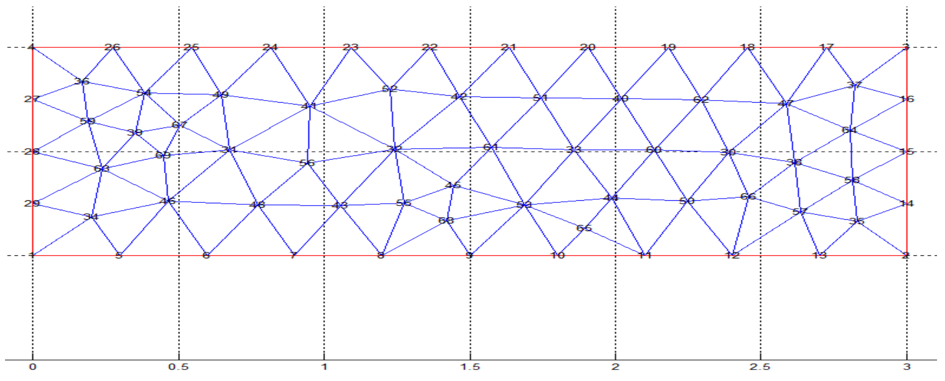


(g) (h)

Figure 2: Test 1 results



(a)



(b)

Figure 3: PDE Toolbox result and mesh

The analytical solution can be obtained by solving the one-dimensional heat conduction equation with no internal heat generation; this is done over the prescribed boundary conditions as shown in Figure 1 above.

$$k\frac{d^2T}{dy^2} = 0 \quad boundary\ conditions\colon\ T = T_0 \quad at\ y = 0; \quad and \quad T = T_l \quad at\ y = L$$

The solution of this equation gives the equation for the temperature profile over the x-direction

$$T = (T_l - T_0)y + T_0$$

$$for\ this\ test\ \ T_0 = 40^oC\ ;\ \ T_l = 70^oC \quad and\ L = 1m \quad \therefore \quad T = 30x + 40$$

Property	Value
Number of elements	107
Number of nodes	69
Number of edges	29
Level of mesh refinement	None
Number of iterations used in solver	200
Maximum error (with reference to PDE Toolbox)	9.9999e-005
Maximum percentage error (with reference to PDE Toolbox)	0.0019
Root mean squared error (with reference to PDE Toolbox)	6.7882e-005
Maximum error (with reference to analytical solution)	9.9999e-005
Maximum percentage error (with reference to analytical solution)	0.0019
Root mean squared error (with reference to analytical solution)	6.7882e-005

Table 1: Test 1 results

DISCUSSION

The results obtained using the simple elements program showed close correlation with the results obtained analytically and those obtained using the PDE Toolbox in MATLAB. The maximum percentage error was 0.00025% for both cases.

A temperature gradient of 29.9998 was obtained when a linear fit was applied to the plot of temperature against the y-coordinate of all the nodes, this value is in very close correlation to what is expected when the Fourier's law is applied to the problem. When the Fourier's law is applied a temperature gradient of 30 is expected.

The root mean squared error gotten with respect to the PDE Toolbox result was 6.7882e-5 while that obtained with respect to the analytical result had the same value.

The contour plot in Figure 2(b) above also shows that the temperature is constant in the x-direction while it varies in the y-direction.

TEST 2: TEST WITH HEAT FLUX AND TEMPERATURE

PROBLEM DESCRIPTION

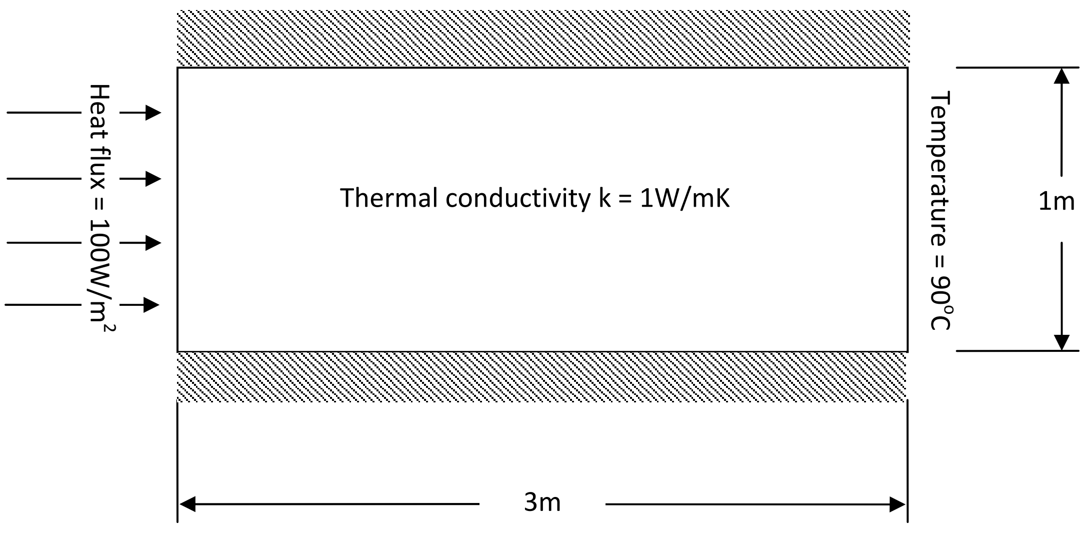
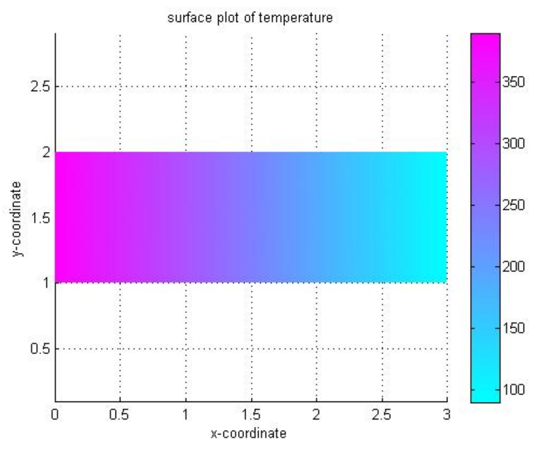


Figure 4: Problem description for Test 2

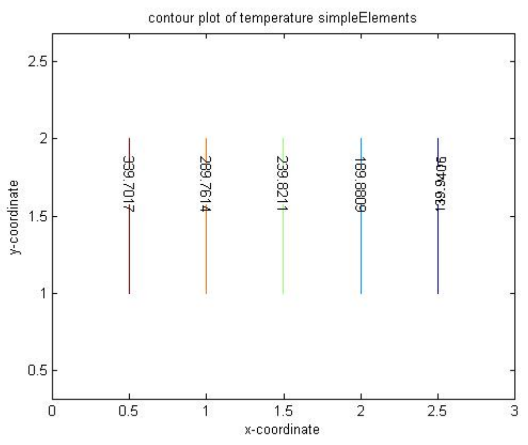
PROCEDURE

1. The procedure here is similar to that followed in Test 1.

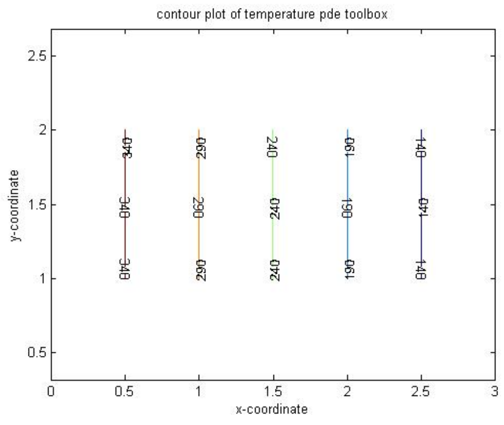
RESULTS

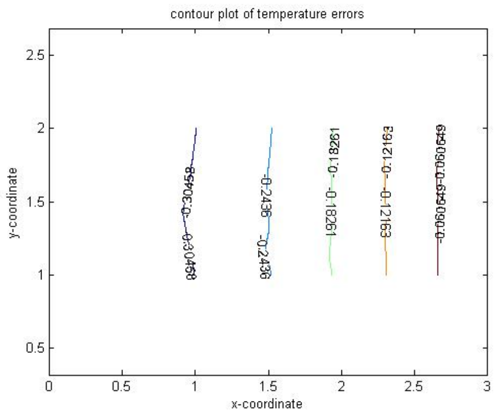


(a)



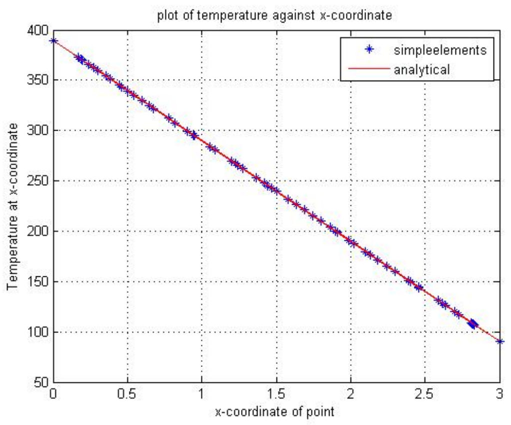
(b)

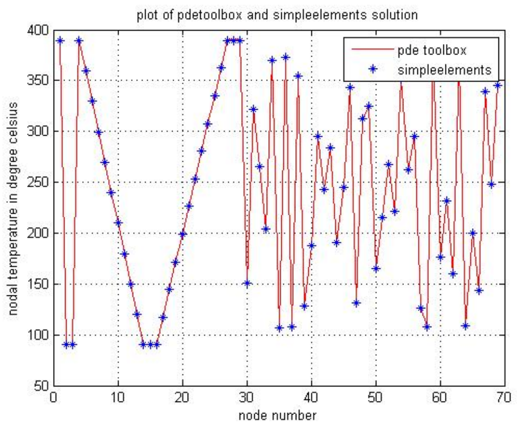




(c)

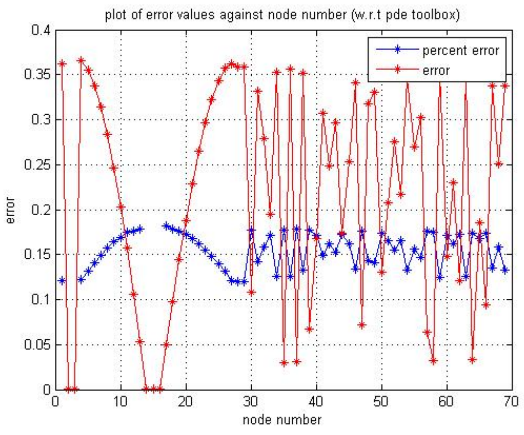
(d)

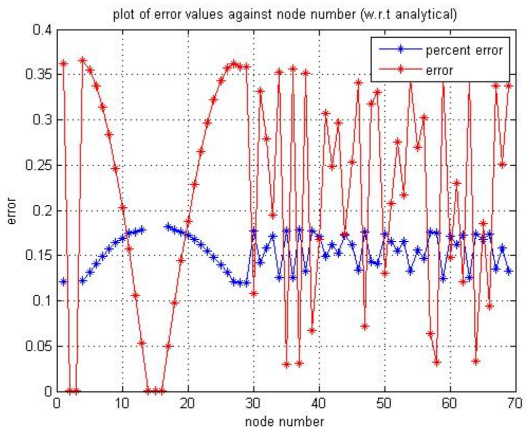




(e)

(f)





(g)

(h)

Figure 5: Test 2 results

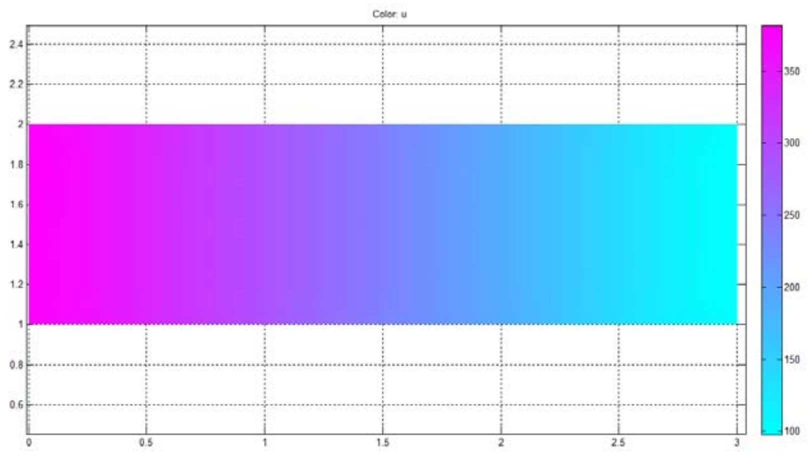


Figure 6: PDE Toolbox result for Test 2

The analytical solution can be obtained by solving the one-dimensional heat conduction equation with no internal heat generation; this is done over the prescribed boundary conditions as shown in Figure 4 above.

$$k\frac{d^2T}{dx^2} = 0 \quad boundary\ conditions\colon\ \frac{dT}{dx} = -q \quad at\ x = 0; \quad and \quad T = T_l \quad at\ x = L$$

The solution of this equation gives the equation for the temperature profile over the x-direction

$$T = -qx + T_l + qL$$

$$for\ this\ test\ \ q = 100W/m^2\ ;\ \ T_l = 90^oC \quad and\ L = 3m \quad \therefore \quad T = -100x + 390$$

Property	Value
Number of elements	107
Number of nodes	69
Number of edges	29
Level of mesh refinement	None
Number of iterations used in solver	200
Maximum error (with reference to PDE Toolbox)	0.3656
Maximum percentage error (with reference to PDE Toolbox)	0.1813
Root mean squared error (with reference to PDE Toolbox)	0.2496
Maximum error (with reference to analytical solution)	0.3656
Maximum percentage error (with reference to analytical solution)	0.1813
Root mean squared error (with reference to analytical solution)	0.2496

Table 2: Test 2 results

DISCUSSION

In this case the error (between the simpleElements program and PDE Toolbox result) was higher than that obtained in the test with temperatures (Test 1), the maximum percentage error obtained was 0.1813%. Although higher than the previous test this value still seems reasonably low.

The errors are seen to be greatest near the heat flux boundary as seen from the contour lines of the error contour plot. This suggests that greater error values are obtained for problems with a heat flux boundary, and could also explain why the error values in this test are greater than those obtained in the previous test (Test 1) which only has fixed temperature boundaries. The important thing to be determined now is why this situation occurs.

A temperature gradient was also obtained in this case by applying a linear fit to the plot of temperature against the x-coordinates of all the points. The temperature gradient obtained was -99.8765. This value is also in close approximation to the temperature gradient expected when the Fourier's law is applied (or from the analytical solution i.e. a gradient of -100).

TEST 3: TEST WITH INTERNAL HEAT GENERATION

PROBLEM DESCRIPTION

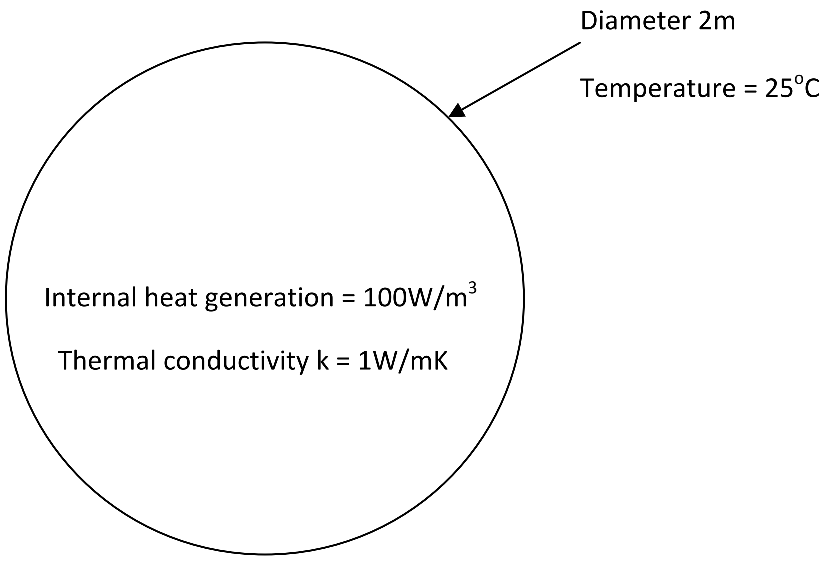
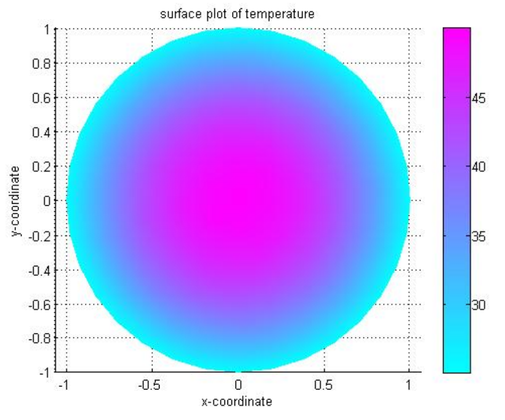

Diameter 2m
Temperature = 25^oC
Internal heat generation = 100W/m^3
Thermal conductivity k = 1W/mK

Figure 7: Problem description for Test 3

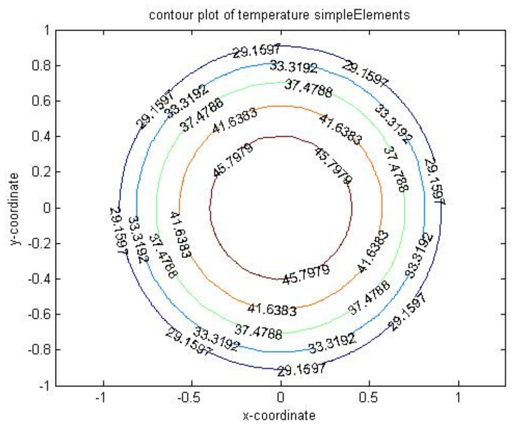
PROCEDURE

1. The procedure here is similar to that followed in Test 1.

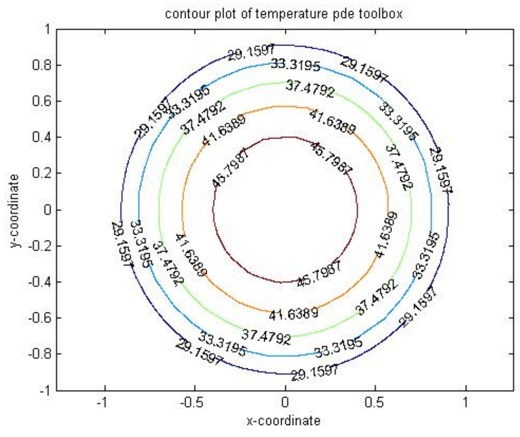
RESULTS



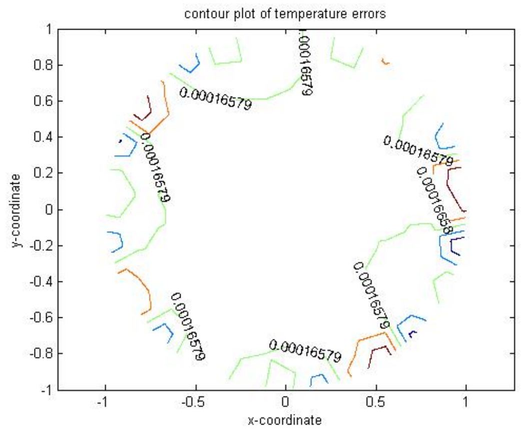
(a)



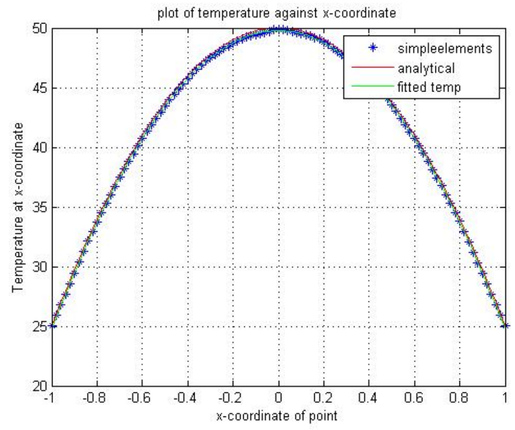
(b)



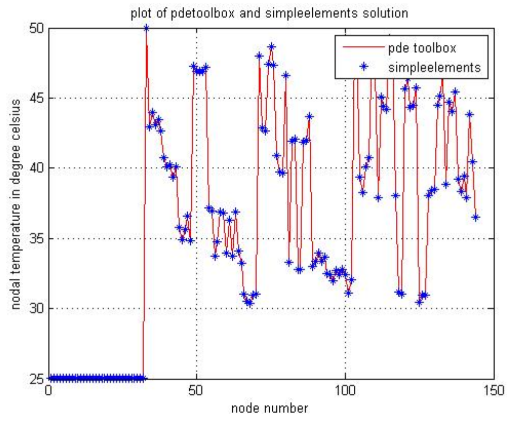
(c)



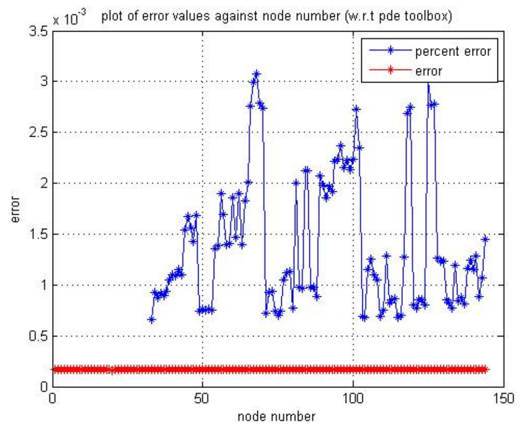
(d)



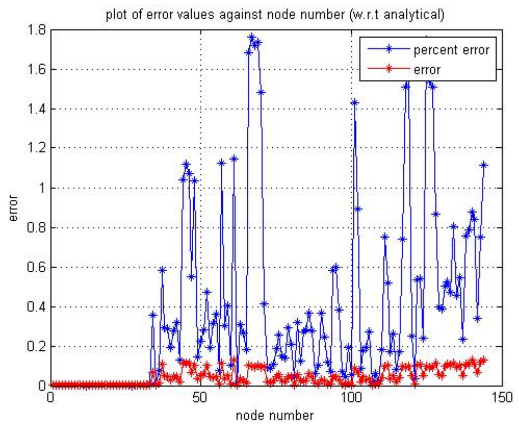
(e)



(f)

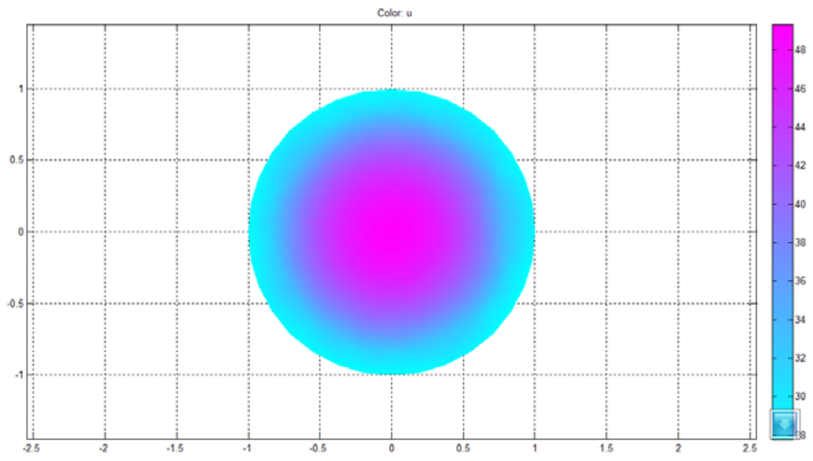


(g)

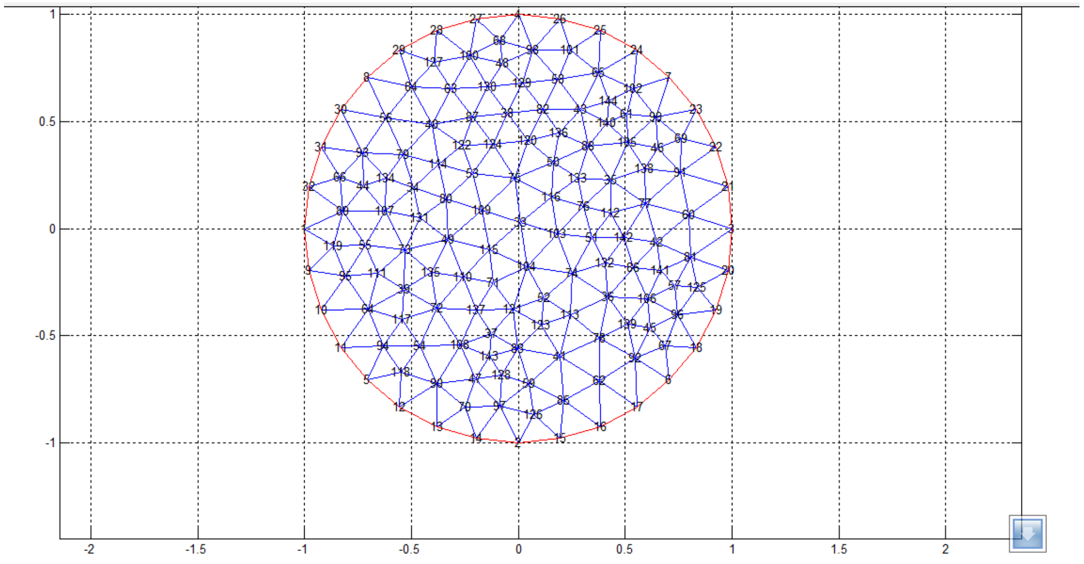


(h)

Figure 8: Test 3 results



(a)



(b)

Figure 9: PDE Toolbox result and mesh for Test 3

The analytical solution can be obtained by solving the radial one-dimensional heat conduction equation over the prescribed boundary conditions,

$$k\frac{d^2T}{dr^2}+\frac{k}{r}\frac{dT}{dr}=-G \quad boundary\ conditions: \ \frac{dT}{dr}= -0 \quad at\ r=0; \quad and \quad T=T_R \quad at\ r=R$$

The solution of this equation gives the equation for the temperature profile in the x-direction

$$T=\frac{-Gr^2}{4k}+T_R+\frac{GR^2}{4k}$$

$$for\ this\ test \quad G=100W/m^3; \quad T_R=25^oC; \quad k=1W/mK \quad and\ R=1m \quad \therefore \quad T=-25x^2+50$$

Property	Value
Number of elements	254
Number of nodes	144
Number of edges	32
Level of mesh refinement	None
Number of iterations used in solver	200
Maximum error (with reference to PDE Toolbox)	1.6818e-004
Maximum percentage error (with reference to PDE Toolbox)	0.0031
Root mean squared error (with reference to PDE Toolbox)	1.6584e-004
Maximum error (with reference to analytical solution)	0.1309
Maximum percentage error (with reference to analytical solution)	1.7601
Root mean squared error (with reference to analytical solution)	0.0600

Table 3: Test 3 result

DISCUSSION

The maximum percentage error obtained between when the simpleElements result is considered with reference to the PDE Toolbox is 0.0031% and when considered with reference to the analytical solution is 1.7601%, this is difference is due to computational errors in the numerical solutions of both the simpleElements and the PDE Toolbox solution.

The error in the simpleElements solution is greatest near the boundary of the circular area, this can be attributed to the approximation of the elements there to perfect triangles (having three straight sides), but in the real sense they are not perfect rectangles (because one of the three sides is an arc), this in effect introduces some error into the solution.

A finer mesh would reduce the errors near the boundary, because as the elements get smaller, they approximate the perfect triangular shape better. The mesh refinement can be done by refining the mesh globally or applying a mesh bias (where the application supports this) which produces a finer mesh near the boundary and a more coarse mesh in the inner part of the circle.

The RMS error has a value of 0.0600^{o}C, which represents a percentage of about 0.24% when considered with reference to the overall temperature range of 25^{o}C.

TEST 4: TEST WITH HEAT FLUX AND INTERNAL HEAT GENERATION

PROBLEM DESCRIPTION

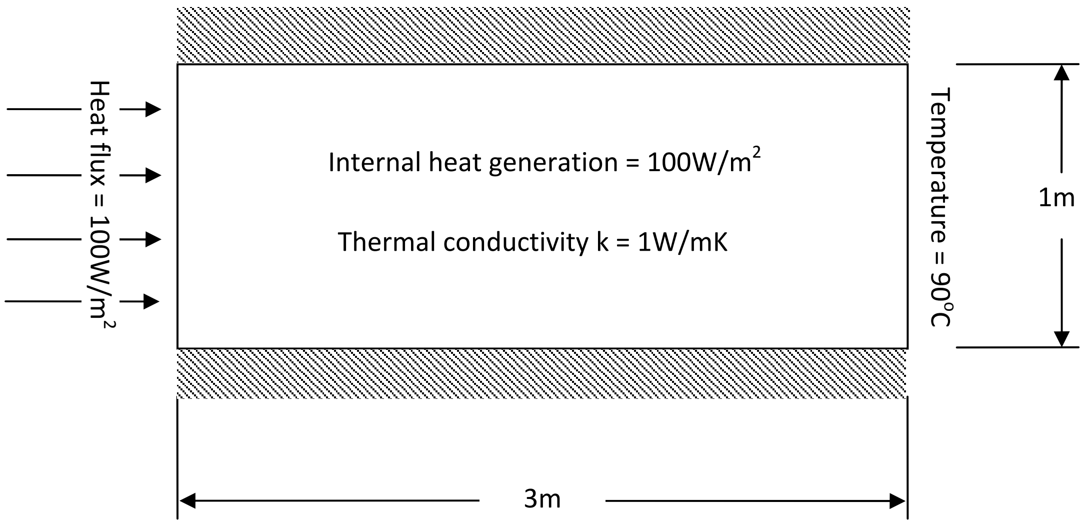
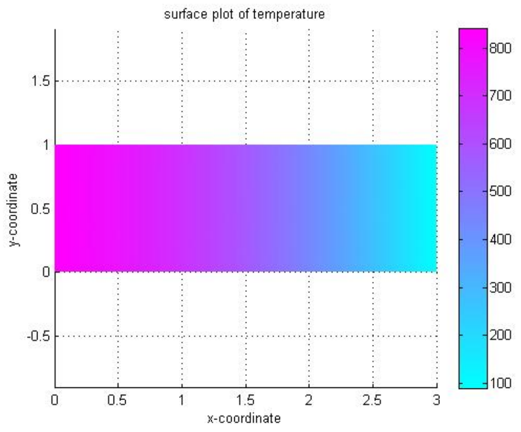

Heat flux = 100W/m^2
Internal heat generation = 100W/m^2
Thermal conductivity k = 1W/mK
Temperature = 90^{o}C
1m
3m

Figure 10: Problem description for Test 4

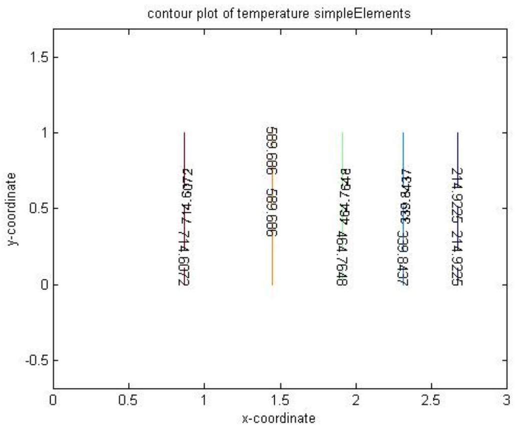
PROCEDURE

1. The procedure here is similar to that followed in Test 1.

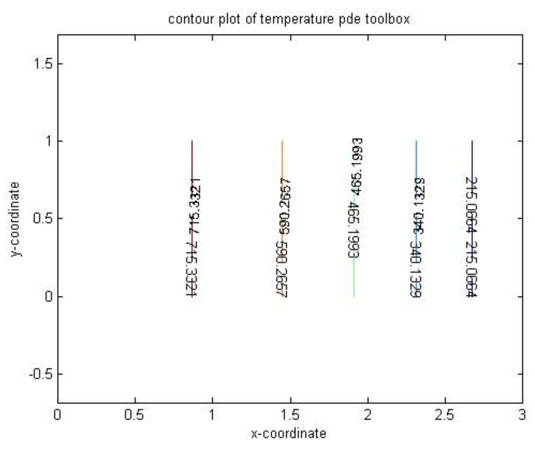
RESULTS



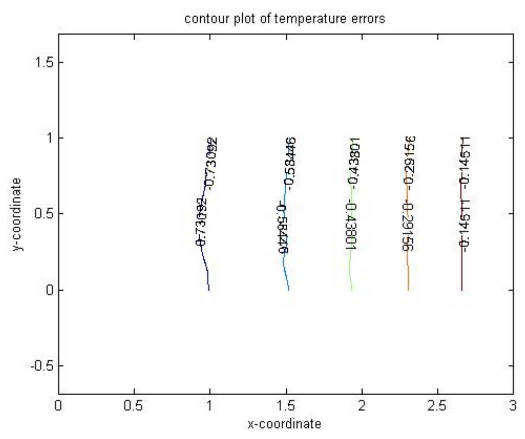
(a)



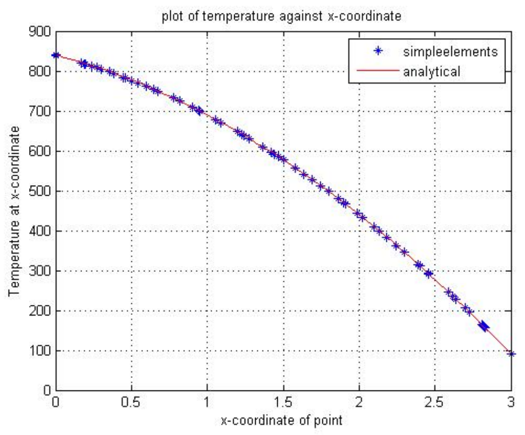
(b)



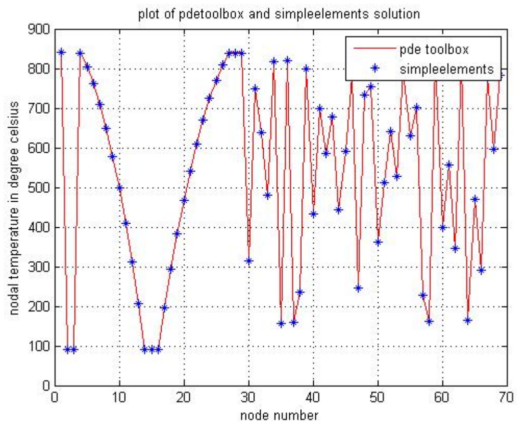
(c)



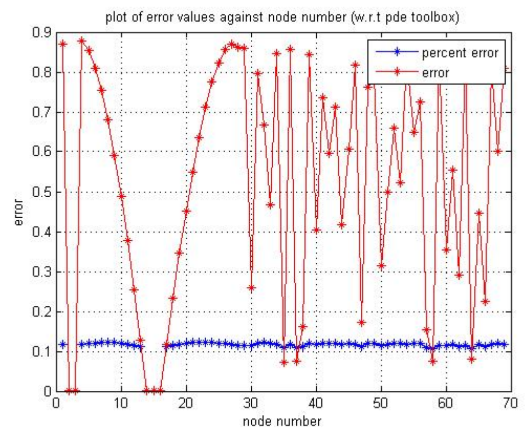
(d)



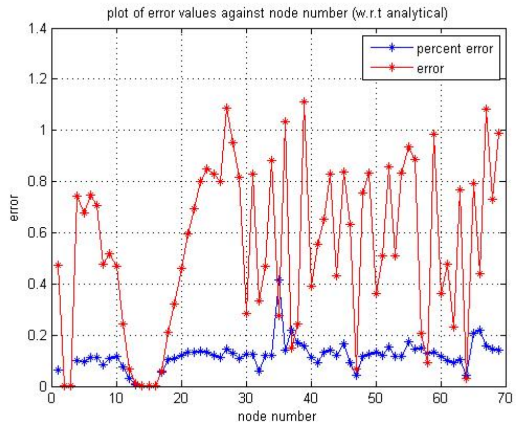
(e)



(f)



(g)



(h)

Figure 11: Test 4 results

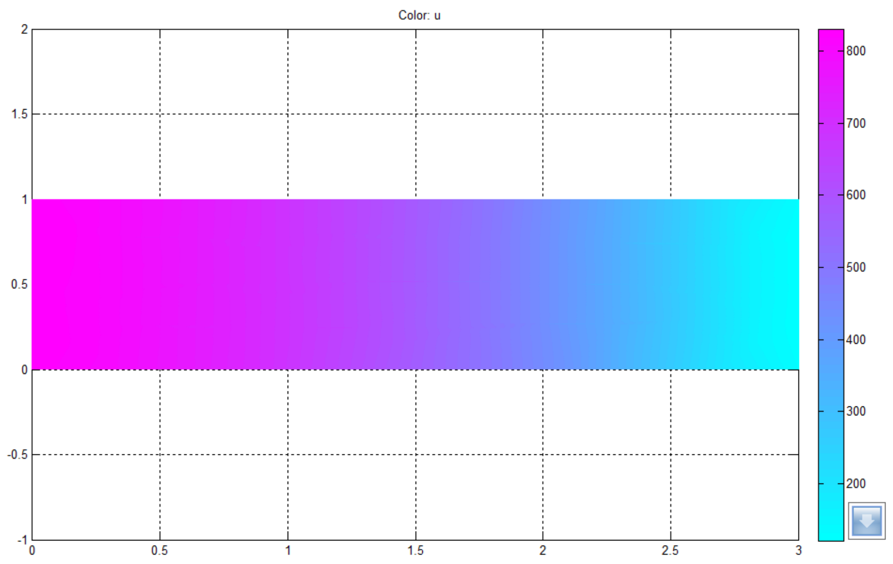


Figure 12: PDE Toolbox resut for Test 4

The analytical solution can be obtained by solving the one-dimensional heat conduction equation over the prescribed boundary conditions,

$$k\frac{d^2T}{dx^2} = -G \quad boundary\ conditions: \quad \frac{dT}{dx} = -q \quad at\ x = 0; \quad and \quad T = T_l \quad at\ x = L$$

The solution of this equation gives the equation for the temperature profile over the x-direction

$$T = \frac{-Gx^2}{2k} - qx + T_l + \frac{GL^2}{2k} + qL$$

$$for\ this\ test \quad G = 100W/m^3; \quad q = 100W/m^2; \quad T_R = 25^oC; \quad k = 1W/mK \quad and\ R = 1m$$

$$\therefore \quad T = -50x^2 - qx + 840$$

Property	Value
Number of elements	107
Number of nodes	69
Number of edges	29
Level of mesh refinement	None
Number of iterations used in solver	200
Maximum error (with reference to PDE Toolbox)	0.8774
Maximum percentage error (with reference to PDE Toolbox)	0.1225
Root mean squared error (with reference to PDE Toolbox)	0.5989
Maximum error (with reference to analytical solution)	1.1103
Maximum percentage error (with reference to analytical solution)	0.4166
Root mean squared error (with reference to analytical solution)	0.6307

Table 4: Test 4 results

DISCUSSION
The result here shows a non-linear temperature gradient which is due to the internal heat generation in the boundary, this effect is depicted by the contour plot of temperatures, as they are not evenly spaced but closer together near the fixed temperature boundary, suggesting a steeper temperature gradient at this end.

The correlation between the results obtained from the simpleElements program and the analytical and PDE Toolbox results is quite high, with a maximum error of 1.1103^{o}C when comared with the analytical result (this occurs over a temperature range of around 750^{o}C, therefore it is quite small compared to the temperature distribution in the region being considered), and a maximum percentage error of 0.4166%.

The errors are also greater near the heat flux boundary of the problem as suggested by the error values in the contour plot.

TEST 5: WITH HEAT FLUX AND INTERNAL HEAT GENERATION (2-DIMENSIONAL CASE)

PROBLEM DESCRIPTION

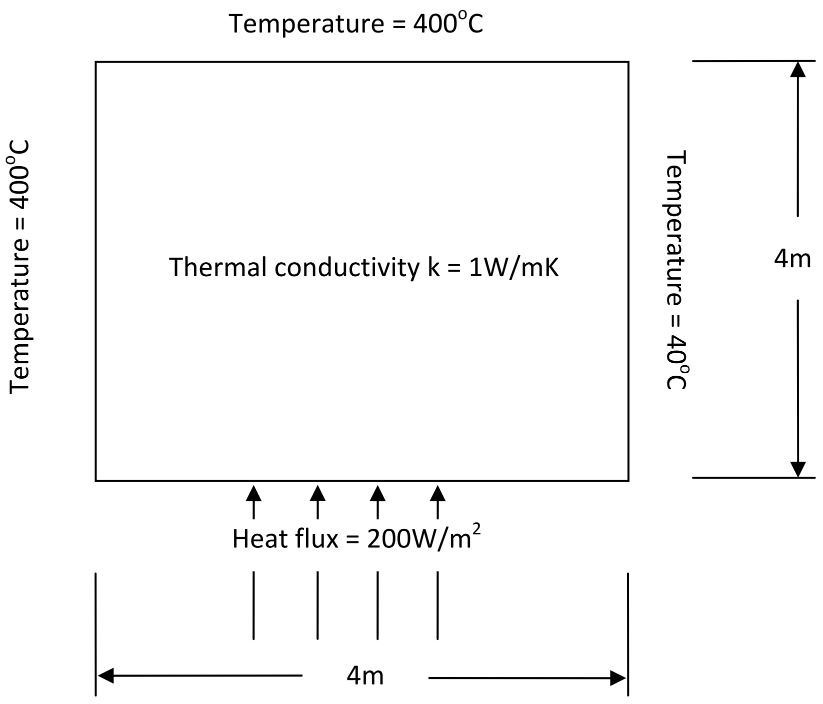
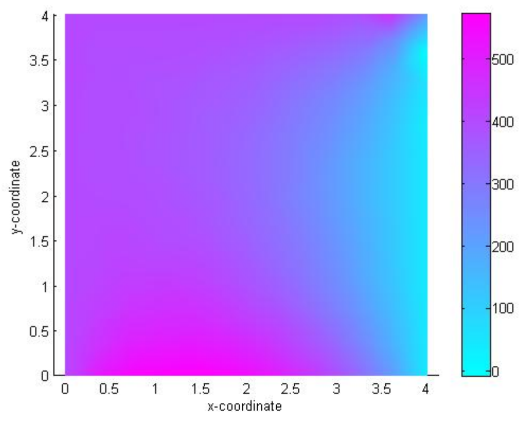


Figure 13: Problem description for Test 5

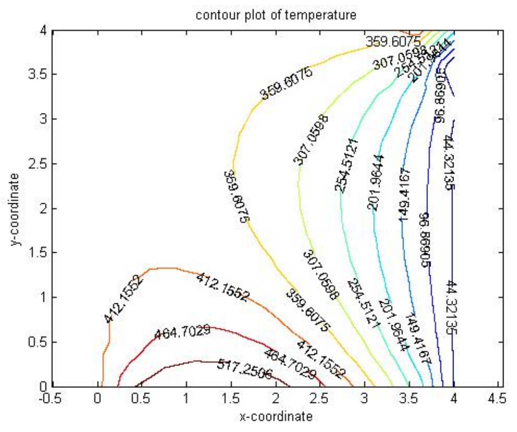
PROCEDURE

1. The procedure here is similar to those followed in Test 1.

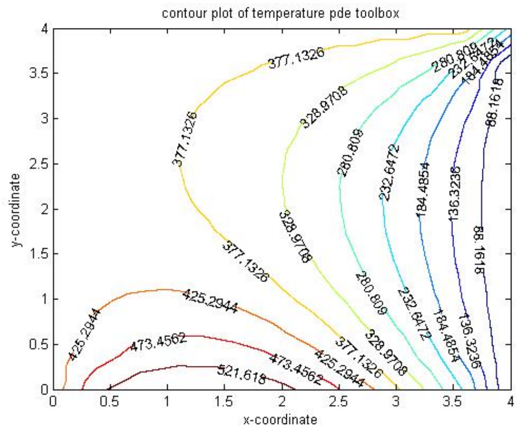
RESULTS



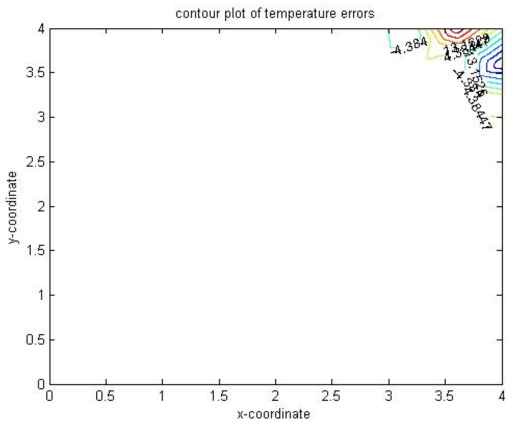
(a)



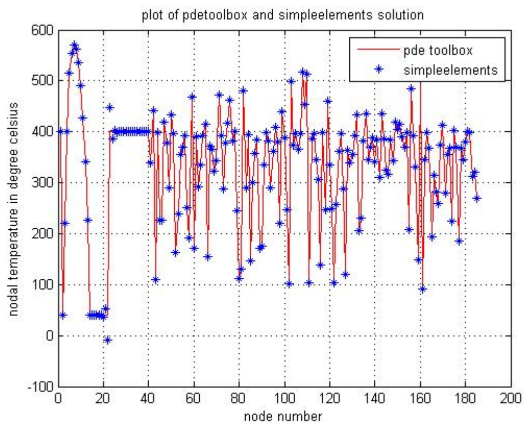
(b)



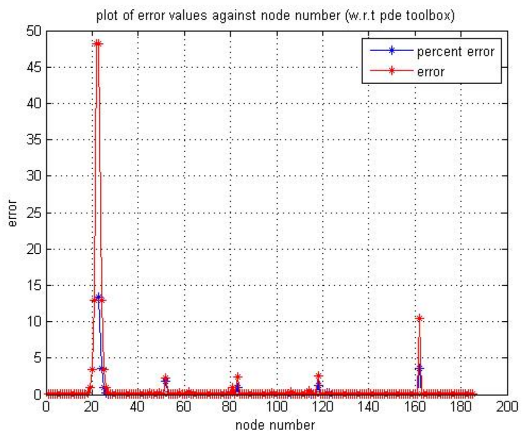
(c)



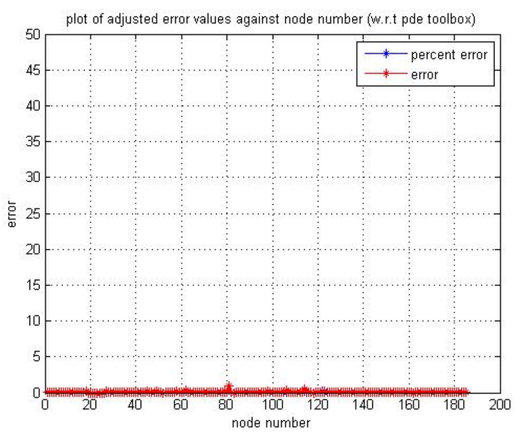
(d)



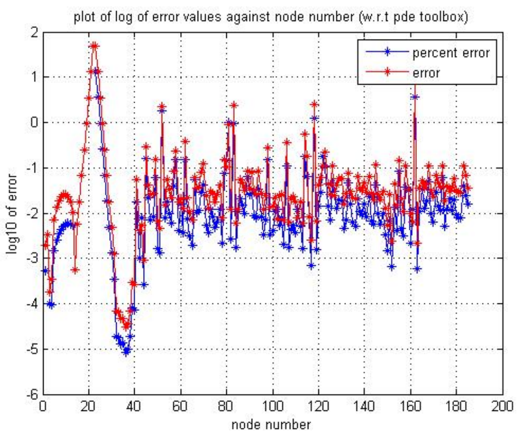
(e)



(f)

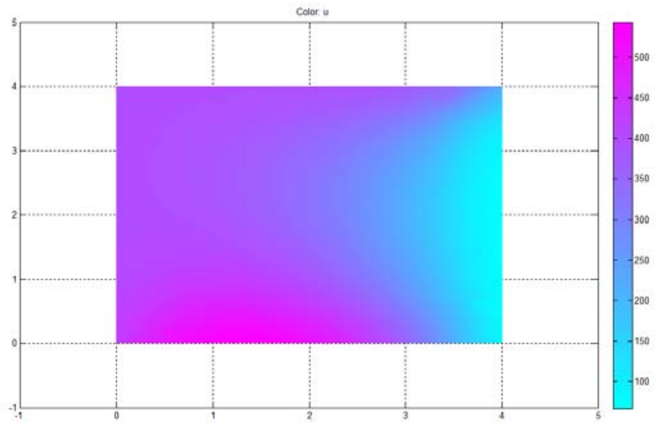


(g)

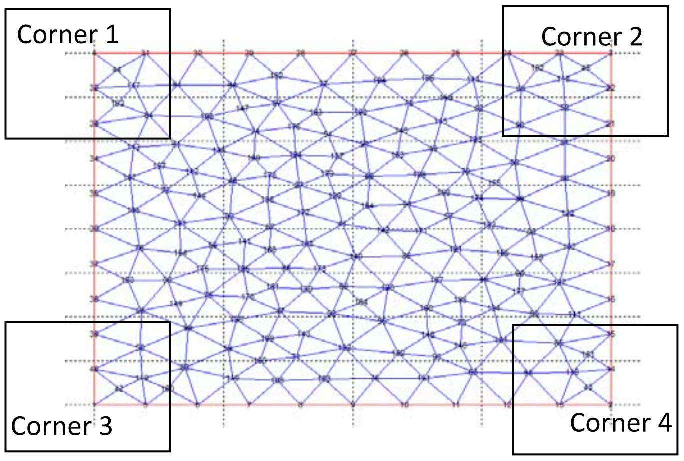


(h)

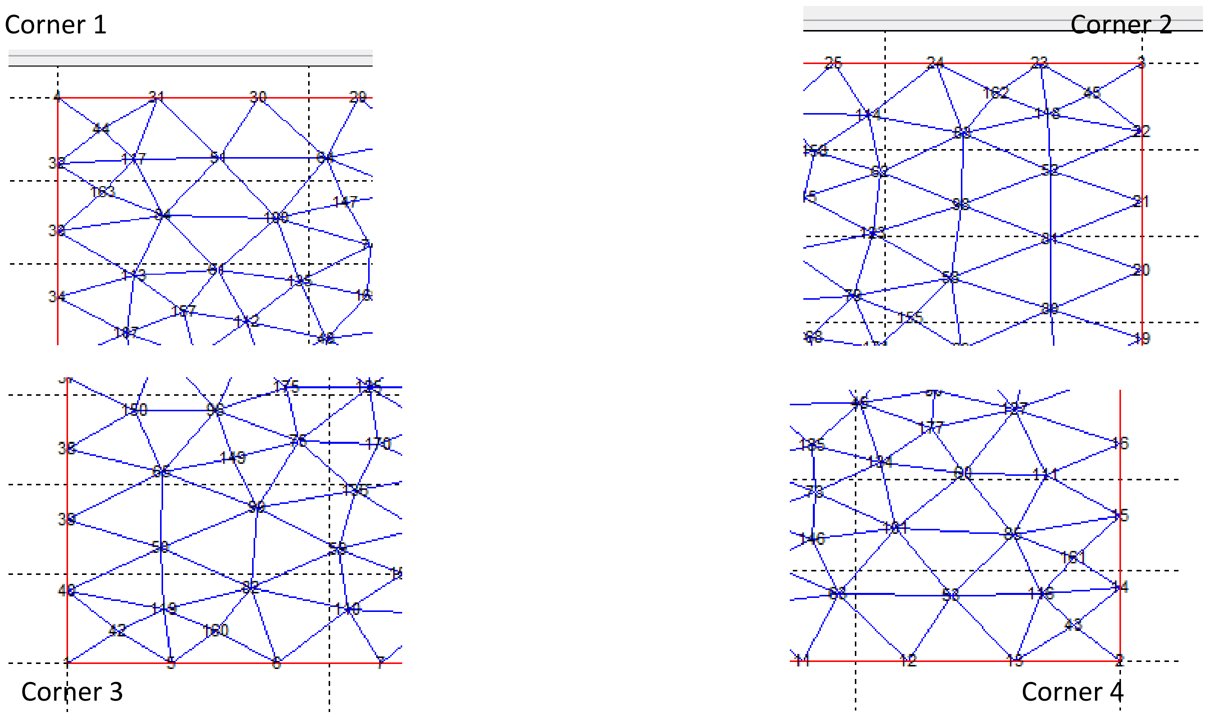
Figure 14: Test 5 results



(a)



(b)



(c)

Figure 15: PDE Toolbox solution and mesh for Test 5

Property	Value
Number of elements	328
Number of nodes	185
Number of edges	40
Level of mesh refinement	None
Number of iterations used in solver	200
Maximum error (with reference to PDE Toolbox)	48.2268
Maximum percentage error (with reference to PDE Toolbox)	13.3963
Root mean squared error (with reference to PDE Toolbox)	5.2709

Table 5: Test 5 results

DISCUSSION

The maximum error in this test occurred at the nodes near the upper right corner of the domain, this is the corner where conflicting temperatures exist, with the north face having a temperature of 400^{o}C while the east face has a temperature of 40^{o}C. This can be deduced from the contour plot of errors, the plot of errors against node number.

From the plot of errors and percentage errors against node number the sharp rises in errors were seen to occur at the nodes 19, 20, 21, 22, 23, 24, 25, 26, 52, 83, 118 and 162, all these nodes exist in the same vicinity; near the corner with the conflicting temperature value.

The errors at these nodes all have a value of 0.9^{o}C and above, while all other errors are below this value. If all these errors are ignored (that is all errors above 0.9), then the resulting RMS error would be 0.1071^{o}C, while the maximum error would be 0.8919^{o}C and the maximum percentage error would be 0.9965%. This means that the errors at this corner alone make up about 98% of the total

RMS error. The plot of the error values when the error at the upper right corner was ignored is shown in Figure 14(g).

Since the error at this node accounts for a greater portion of the overall error, it would be necessary to devise a method of either correcting or compensating for it.

TEST 6: EFFECT OF MESH REFINEMENT ON SOLUTION ACCURACY

PROBLEM DESCRIPTION

The problem used in this test is the same as that used in Test 3.

PROCEDURE

1. The problem described in Test 3, is replicated in the PDE Toolbox in MATLAB
2. The mesh, boundary condition and solution was exported to the MATLAB workspace and saved with a unique name (e.g. "`simple_elements_8_1_data.mat`")
3. The mesh is refined and the solution recalculated, the resulting boundary condition, mesh and solution is exported to MATLAB and saved with a different name (e.g. "`simple_elements_8_2_data.mat`").
4. Step 3 above is repeated two more times to create two more data sets (e.g. "`simple_elements_8_3_data.mat, simple_elements_8_4_data.mat`"), this gives one data set for the initial mesh and three other data sets for the three successive mesh refinements.
5. The four data sets are then solved using the simpleElements program ("`simpleElements_test_8.m`"), by loading the four data sets into it. The number iteration used in the solver is varied for each data set, this gives twelve sets of results. The results are shown in Tables 6, 7 and 8.

RESULTS

Mesh refinement	None	1	2	3
Number of elements	254	1016	4064	16256
Number of nodes	144	541	2097	8257
Number of edges	32	64	128	256
Level of mesh refinement	None	1	2	3
Number of iterations used in solver	200	200	200	200
Maximum error (with reference to PDE Toolbox)	1.6818e-004	1.6717e-004	0.6390	12.8038
Maximum percentage error (with reference to PDE Toolbox)	0.0031	0.0058	2.6402	52.5458
Root mean squared error (with reference to PDE Toolbox)	1.6584e-004	1.6575e-004	0.3415	6.9671
Maximum error (with reference to analytical solution)	0.1309	0.0401	0.6392	12.8038
Maximum percentage error (with reference to analytical solution)	1.7601	0.5595	2.6600	52.5471
Root mean squared error (with reference to analytical solution)	0.0600	0.0140	0.3437	6.9677

Table 6: Test 6 results using 200 iterations in the solver

Mesh refinement	None	1	2	3
Number of elements	254	1016	4064	16256
Number of nodes	144	541	2097	8257
Number of edges	32	64	128	256
Level of mesh refinement	None	1	2	3
Number of iterations used in solver	500	500	500	500
Maximum error (with reference to PDE Toolbox)	1.6818e-004	1.6717e-004	0.0014	2.9103
Maximum percentage error (with reference to PDE Toolbox)	0.0031	0.0058	0.0069	11.9959
Root mean squared error (with reference to PDE Toolbox)	1.6584e-004	1.6646e-004	7.0248e-004	1.5788
Maximum error (with reference to analytical solution)	0.1309	0.0401	0.0124	2.9104
Maximum percentage error (with reference to analytical solution)	1.7601	0.5595	0.1749	11.9983
Root mean squared error (with reference to analytical solution)	0.0600	0.014	0.0039	1.5794

Table 7: Test 6 results using 500 iterations in the solver

Mesh refinement	None	1	2	3
Number of elements	254	1016	4064	16256
Number of nodes	144	541	2097	8257
Number of edges	32	64	128	256
Level of mesh refinement	None	1	2	3
Number of iterations used in solver	1000	1000	1000	1000
Maximum error (with reference to PDE Toolbox)	1.6818e-004	1.6717e-004	1.6687e-004	0.2451
Maximum percentage error (with reference to PDE Toolbox)	0.0031	0.0058	0.0113	1.0107
Root mean squared error (with reference to PDE Toolbox)	1.6584e-004	1.6646e-004	1.6658e-004	0.1329
Maximum error (with reference to analytical solution)	0.1309	0.0401	0.0116	0.2451
Maximum percentage error (with reference to analytical solution)	1.7601	0.5595	0.1698	1.0187
Root mean squared error (with reference to analytical solution)	0.0600	0.0140	0.0033	0.1335

Table 8: Test 6 results using 1000 iterations in the solver

DISCUSSION

The discussion below in most part considers the 200 iterations result.

The accuracy of the solution increased with increase in the number of elements, but as the number of elements continued to increase the accuracy began to drop. The best solution accuracy was gotten with one level of refinement (having 1016 elements).

It was also observed that with the first mesh refinement (for the result with 200 iterations) the error of the simpleElements solution with respect to the PDE Toolbox solution did not show any noticeable decrease (just about 0.054% reduction in the RMS error) while that with respect to the analytical solution showed a considerable decrease (from an RMS error of 0.0600 to 0.0140, approximately a reduction of about 76.67% in the RMS error). This effect is due to the fact that both solutions (i.e. simpleElements and PDE Toolbox) converge to the analytical solution, therefore the change in the error with reference to PDE Toolbox depends on the difference in the convergence rate of the two solutions, whereas the change in the error with reference to the analytical solution is absolute, as mesh refinements do not alter the analytical solution.

It is also necessary to point out at this stage that, the difference in the error values obtained with respect to the PDE Toolbox and the analytical solution while using the initial mesh and the first mesh refinements was quite large (was about two orders of magnitude for the first mesh refinement). This suggests that at this mesh refinement level there exists a significant difference between the PDE Toolbox results and the analytical results, although the actual values of these errors are only fractions of a degree Kelvin, but when the errors are considered relative to each other the difference is significant.

With further mesh refinements the errors with respect to PDE Toolbox and analytical increased and these errors also now had a closer value, suggesting two things:

1. The accuracy of the simple elements program decreased as the number of elements increased beyond a threshold value (this threshold can be said to depend on the number of iterations used in the solver).
2. The accuracy of the PDE Toolbox increased with an increase in the number of elements, thus resulting in similar error values since both the PDE Toolbox results and the analytical are compared with the same values (i.e. the simpleElements solution).

Hence, It can be inferred that the solution accuracy of the simpleElements program increases as the number of elements increases, but there is a peak (which depends on the number of iterations used in the solver), after this peak the solution accuracy begins to drop. The solution accuracy of the PDE Toolbox on the other hand continues to increase as the number of elements increases with no visible peak after which the solution accuracy drops, this can be inferred from the fact that the error values with respect to the PDE Toolbox for the third refinement is closer to the error values with respect to the analytical than it was for the second mesh refinement.

TEST7: EFFECT OF INCREASING THE NUMBER OF ITERATIONS ON SOLUTION ACCURACY

PROBLEM DESCRIPTION

The problem used in this test is the same as that used in Test 5.

PROCEDURE

1. The procedure here is similar to that followed in Test 6.

RESULTS

Mesh refinement	None	1	2	3
Number of elements	328	1312	5248	20992
Number of nodes	185	697	2705	10657
Number of edges	40	80	160	320
Level of mesh refinement	None	1	2	3
Number of iterations used in solver	200	200	200	200
Maximum error (with reference to PDE Toolbox)	48.2268	48.2223	68.4896	297.2076
Maximum percentage error (with reference to PDE Toolbox)	13.3963	16.4191	23.3650	93.3002
Root mean squared error (with reference to PDE Toolbox)	5.2709	2.7676	36.7530	169.5573

Table 9: Test 7 results using 200 iterations in the solver

Mesh Refinement	None	1	2	3
Number of elements	328	1312	5248	20992
Number of nodes	185	697	2705	10657
Number of edges	40	80	160	320
Level of mesh refinement	None	1	2	3
Number of iterations used in solver	500	500	500	500
Maximum error (with reference to PDE Toolbox)	48.2268	48.2223	48.2136	143.8986
Maximum percentage error (with reference to PDE Toolbox)	13.3963	16.4174	16.3627	49.3471
Root mean squared error (with reference to PDE Toolbox)	5.2709	2.7667	2.2520	79.4071

Table 10: Test 7 results using 500 iterations in the solver

Mesh Refinement	None	1	2	3
Number of elements	328	1312	5248	20992
Number of nodes	185	697	2705	10657
Number of edges	40	80	160	320
Level of mesh refinement	None	1	2	3
Number of iterations used in solver	1000	1000	1000	1000
Maximum error (with reference to PDE Toolbox)	48.2268	48.2223	48.2136	48.1964
Maximum percentage error (with reference to PDE Toolbox)	13.3963	16.4174	16.3522	16.3969
Root mean squared error (with reference to PDE Toolbox)	5.2709	2.7667	1.4004	22.5967

Table 11: Test 7 results using 1000 iterations in the solver

DISCUSSION

For this test the simpleElements solution was only compared with the PDE Toolbox solution due to the following reasons:

1. The analytical solution for two-dimensional solutions is very difficult to obtain and in most cases one would normally resort to a numerical solution (of which the PDE Toolbox is one).
2. It has been shown in Test 6 that the solution of the PDE Toolbox converges to the analytical solution as the mesh is refined, hence it can be trusted.

When 200 iterations were used in the solver, it was observed that the solution accuracy of the simpleElements program increased with the first mesh refinement but then decreased on subsequent mesh refinements.

The solution accuracy increased with the second mesh refinement when the number of iterations was increased to 500, but decreased on the third refinement.

For a number of iterations of 1000, the solution accuracy did not increase in the third refinement, but rather the solution accuracy in the second refinement was better than that gotten when the number of iterations was five hundred.

When the effect a greater number of iterations has on the solution at each level of mesh refinement was considered, it was observed that an increase in the number of iterations brings about an increase in the solution accuracy. The solution accuracy for the third mesh refinement increase with increase in number of iterations (the RMS error reduced from 169.5573 when two hundred iterations were used to 79.4071 for five hundred iterations, and then 22.5967 for one thousand iterations).

DISCUSSION

From the tests carried out above it can be inferred that there are three major sources of errors in the simpleElements program;

- The first source of error is due to the finite element discretization, and this can be reduced by using a finer mesh, but there’s a threshold above which the error would no longer decrease but begin to increase.
- The second one is in the computational scheme used to solve the problem, this can be reduced by increasing the number of iterations and so far it has been observed that this always reduces the error in the computational scheme, until the reduction in this error is negligible compared to the error due to the discretization, at this point no considerable reduction is noticed with further increase in the number of iterations.
- The last source of error is due to the problem definition this usually occurs at a corner node subjected to conflicting boundary conditions, this error is observed to propagate throughout the solution regardless of the increase in the number of iterations or number of elements in the mesh. It is usually expressed as the maximum error or maximum percentage error. This error has a constant value for each mesh refinement level and does not change with an increase or decrease in the number of iterations.

Hence we can infer that to get a good solution accuracy from the simpleElements program we need to select a good combination of mesh refinement and number of iterations, a very high value of both does not necessarily guarantee a very accurate solution as can be seen in Test 7. A higher accuracy was gotten when a mesh with 328 elements was used with 200 iterations than when a mesh with 20992 elements was used with 1000 iterations.

RECOMMENDATIONS FOR FURTHER WORK

Some of the errors noted during these tests would need to be addressed later on to ensure that the classes function as required and produce better result.

1. The method of assigning the boundary conditions at nodes with conflicting boundary conditions should be modified.
2. It would also be necessary to determine why greater errors occur when a heat flux boundary condition is present.
3. Finally, the reason for the reduction in accuracy when a finer mesh is used without increasing the number of iterations would need to be investigated.

GLOSSARY

JAVA CLASSES CALLED BY SIMPLE ELEMENTS

The simple elements program calls some java classes which are used to instantiate Java objects with predefined methods and properties, the methods and properties of these objects are then used to solve the problem.

The Java classes called by the simple elements program are discussed below

Node(Tierney 2010): This instantiates the node object.

The syntax for doing this is: Node (id, t, x, y, z)

"id" is the index of the node

"t" stands for the boundary type

"x" stands for the x-coordinate of the node

"y" stands for the y-coordinate of the node

"z" stands for the z-coordinate of the node

Element2DC03N(Tierney 2010): this instantiates the element object.

The syntax for doing this is: Element2DC03N (id, t, n1, n2, n3)

"id" is the element serial number

"t" is the material type

"n1": is the first node

"n2": is the second node

"n3": is the third node

Face2DC03N(Tierney 2010): this instantiates the face object.

The syntax for doing this is: Face2DC03N (id, t, n1, n2)

"Id" is the face serial number

"t" is the face boundary type

"n1" is the first node of the face

"n2" is the second node of the face

StiffMatrix(Tierney 2010): this instantiates the stiffness matrix object.

The syntax for doing this is:	StiffMatrix (ntot, bwidth)
	“Ntot” is the total number of nodes
	“Bwidth”: is the bandwidth of the band matrix

LoadMatrix(Tierney 2010): this is used to instantiate the load vector object.

The syntax for doing this is:	LoadMatrix (ntot)
	“Ntot” is the total number of element

REFERENCES

Dietel, H. M. and P. J. Dietel (2002). Java, How to program, Prentice Hall.

Tierney, M. J. (2010). Element2DC03N.java. Bristol, U.K, Tireney, M.J.

Tierney, M. J. (2010). Face2DC03N.java. Bristol, U.K, Tierney, M.J.: Java class to instantiate face objects.

Tierney, M. J. (2010). LoadMatrix.java. Bristol, U.K, Tierney, M.J: Java class to instantiate element node matrix object.

Tierney, M. J. (2010). Node.java. Bristol, U.K, Tierney, M.J.: Java class to instantiate node objects.

Tierney, M. J. (2010). StiffMatrix.java. Bristol, U.K, Tierney, M.J.: Java class to instantiate element stiffness matrix.

Tierney, M. J. (2011). Discussions with personal tutor. A. O. Dada. Bristol, U.K.

Tierney, M. J. and A. O. Dada (2010). simpleElements.m. Bristol: Matlab script to call java classes.

APPENDIX

This appendix contains a sample of the MATLAB script used to test the classes, there are slight variations in the scripts used for the seven tests, but the functionality is similar for all.

simpleElements.m (Tierney and Dada 2010)

```
clear % clears all data in the workspace
close all % closes all open figure windows
clc % clears the command line

tic % starts the timer

% load the data file containing the mesh data produced in PDE Toolbox
load('simple_elements_7_data')

javaclasspath myFile.jar ; % Allows dynamic use of all Java classes in
working directory.


% create variables which indicate the number of elements, triangles and
% edges
% Number of elements is equal to columns_p
% Number of triangles is equal to columns_t
% Number of edges is equal to columns_e

[rows_p, columns_p] = size(p);
[rows_t, columns_t] = size(t);
[rows_e, columns_e] = size(e);
cond = 1; % thermal conductivity
source = 100; % volumetric heat generation

% set up nodes
for ii = 1: columns_p;
    n(ii) = Node (ii,   0,  p(1, ii), p(2, ii), 0);
end

% set up elements
for jj = 1: columns_t;
    el(jj) = Element2DC03N(1, 0, n(t(1,jj)), n(t(2,jj)), n(t(3,jj)));
end

% set up faces
for kk = 1:columns_e;
    face(kk) = Face2DC03N(1, 0,  n(e(1,kk)), n(e(2,kk)));
end

% calculate element stiffness
for els = 1: columns_t
    el(els).calculateStiff(cond,  source); % element stiffness matrices.
end

% set face boundary conditions
for fcs = 1:columns_e;
    bound_type = e(5, fcs);
    if bound_type == 1
        a_value = 0; b_value = 0;
    elseif bound_type == 2
        a_value = 300000*90; b_value = 300000;
    elseif bound_type == 3
        a_value = 0; b_value = 0;
    elseif bound_type == 4
        a_value = 100; b_value = 0;
    end
    face(fcs).calculateStiff(a_value, b_value);
end







% Assemble global matrices and vectors
ntot = columns_p; % total number of notes.
bwidth = 30; % bandwith
% PART 2.3.3 Assemble global stiffness matrix
        s1 = StiffMatrix (ntot, bwidth);
        l1 = LoadMatrix (ntot);
        for els =1:columns_t
            s1.addLocalStiffness(el(els).getNodes(), el(els).getStiff);
            l1.addLocalLoad(el(els).getNodes, el(els).getLoad);
        end

        for faces = 1:columns_e  %
            s1.addLocalStiffness(face(faces).getNodes,
face(faces).getStiff);
            l1.addLocalLoad(face(faces).getNodes, face(faces).getLoad);
        end

% solve problem using assembled global matrices and vectors
niter = 200; % 200 iterations.
T = 20*ones(1,columns_p);

% Solve by successive over relaxation
display ('new temperatures')
TT = s1.sor (l1, Matrix (ntot, 1, T'),  niter, 1.6).cnv();


%% calculate errors

error_pde = u-TT;
max_error_pde = max(abs(error_pde))
percent_error_pde = abs(error_pde)./(u-min(u))*100;

% this sets the percent errors greater than 10e5 to zero, as these are
% caused by zero division or division by very small values

percent_error_pde = percent_error_pde.*(percent_error_pde<=10e5);
max_percent_error_pde = max(abs(percent_error_pde))
rms_error_pde = ((sum(error_pde.^2))/length(error_pde)).^0.5

analytical_temp = (-source/(2*cond)*(p(1,:)).^2)-(100*p(1,:))+...
    (90+(source*(max(p(1,:))^2)/(2*cond))+source*max(p(1,:)));
error_analytical = analytical_temp'-TT;
max_error_analytical = max(abs(error_analytical))
```

```
percent_error_analytical = abs(error_analytical)./(analytical_temp-
min(analytical_temp))'*100;

% this sets the percent errors greater than 10e5 to zero, as these are
% caused by zero division or division by very small values

percent_error_analytical =
percent_error_analytical.*(percent_error_analytical<=10e5);
max_percent_error_analytical = max(abs(percent_error_analytical))
rms_error_analytical =
((sum(error_analytical.^2))/length(error_analytical)).^0.5


toc

%% plot graphs

pdesurf(p,t, TT)
xlabel('x-coordinate');
ylabel('y-coordinate');

% this produces the surface plot of temperature
title('surface plot of temperature')
colorbar
view([0 0 1])
grid on
axis equal

% this plots the temperature at the nodes against the x-coordinte of the
% node
figure
x_values = min(p(1,:)):max(p(1,:))/50:max(p(1,:));
analytical_temp2 = (-source/(2*cond)*(x_values).^2)-(100*x_values)+...
    (90+(source*(max(p(1,:))^2)/(2*cond))+source*max(p(1,:)));
plot(p(1,:), TT,'*',x_values,analytical_temp2,'r');
xlabel('x-coordinate of point');
ylabel('Temperature at x-coordinate');
title('plot of temperature against x-coordinate');
legend('simpleelements','analytical')
grid on

% creates a contour plot of the temperatures obtained from the
% simpleElements program
figure
x = 0:0.01:max(p(1,:));
y = 0:0.01:max(p(2,:));
[c h] = contour (x,y,tri2grid (p,t, TT, x, y),5);
clabel(c, h)
xlabel('x-coordinate');
ylabel('y-coordinate');
title('contour plot of temperature simpleElements')
axis equal

% creates a contour plot of the temperatures obtained from the
% PDE Toolbox
figure
x = min(p(1,:)):0.001:max(p(1,:));
y = min(p(2,:)):0.01:max(p(2,:));
[c h] = contour (x,y,tri2grid (p,t, u, x, y),5);
```

```
clabel(c, h)
xlabel('x-coordinate');
ylabel('y-coordinate');
title('contour plot of temperature pde toolbox')
axis equal

% creates a contour plot of the errors in temperatures between the results
% obtained from the simpleElements program and PDE Toolbox
figure
x = min(p(1,:)):0.001:max(p(1,:));
y = min(p(2,:)):0.01:max(p(2,:));
[c h] = contour (x,y,tri2grid (p,t, (TT-u), x, y),5);
clabel(c, h)
xlabel('x-coordinate');
ylabel('y-coordinate');
title('contour plot of temperature errors')
axis equal

% this plots the nodal temperatures against the node numbers
figure
plot((1:columns_p), u,'r',(1:columns_p),TT,'b*');
legend('pde toolbox','simpleelements');
xlabel('node number');
ylabel('nodal temperature in degree celsius');
title('plot of pdetoolbox and simpleelements solution');
grid on

% this plots the error and percentage error values against the node numbers
% for the simpleElements solution and PDE Toolbox
figure
plot((1:columns_p), percent_error_pde,'-*b',(1:columns_p),
abs(error_pde),'-*r');
legend('percent error','error');
xlabel('node number');
ylabel('error');
title('plot of error values against node number (w.r.t pde toolbox)');
grid on

% this plots the error and percentage error values against the node numbers
% for the simpleElements solution and analytical solution
figure
plot((1:columns_p), percent_error_analytical,'-*b',(1:columns_p),...
    abs(error_analytical),'-*r');
legend('percent error','error');
xlabel('node number');
ylabel('error');
title('plot of error values against node number (w.r.t analytical)');
grid on
```