RESEARCH PROPOSAL (TASK 2)

INVESTIGATION OF SOLUTION METHODS FOR SIMULATION OF ADSORPTION PROCESSES IN ADSORPTION CHILLERS

SUBMITTED TO:

DEPARTMENT OF MECHANICAL ENGINEERING

SUBMITTED BY:

DADA, AKINDOLU OLUWAKANYINSOLA

STUDENT NUMBER: 1050279

MSC IN ADVANCED MECHANICAL ENGINEERING

APRIL 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)

EXECUTIVE SUMMARY

This proposal discusses the results of the tests carried out on some codes which are intended to be used in the major research work; two sets of codes were investigated because of the flexibility they offer in modifying the system parameters as required by the research problem.

The methodology to be employed and the schedule of the activities to be carried out are also given the report is then concluded with a brief section on the resources required and the cost implications of the research work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

This proposal discusses the approach which would be employed in carrying out the major research project, and the major activities which would be carried out in the research. It also discusses the schedule and of carrying out these activities, and the required resources and the cost which would be incurred during the research activity.

Most importantly it discusses the results obtained from the investigations carried out to determine suitable tools which would be employed in the main research work. These tools were investigated in order to obtain a suitable tool which would be used to solve the transient heat conduction problem with heat generation, which would be investigated for a pellet of silica gel, in the final research work.

Some of the tools investigated are the finite element java classes created by Tierney, M.J[1] and the PDE Toolbox in Matlab, the results obtained from these tools were compared against the analytical solutions.

1.1. BACKGROUND OF STUDY

Based on the literature survey carried out by Dada, A.O [2], it can be inferred that adsorption process occurring in adsorbent of a solar refrigerator is not easy to model by empirical means, hence there is a need to create numerical models which CAN be used to study the system and possibly provide some insight into the working of the adsorption process. It was also observed that the heat transfer process which takes place in the adsorbent involves constantly varying properties which may result in the system being non-linear (when the properties of the system vary with temperature). The system may also contain spatially varying parameters like thermal conductivity or heat generation, or time varying ones.

The varying properties of the system require a solution method with the capability of varying the system properties in each time step, this possibility would be investigated for the proposed solution methods.

1.2. RESEARCH QUESTIONS

The following research questions are intended to be answered by the major research

- Does temperature affect the adsorption process in an adsorbent?
- If temperature does affect the adsorption process, what is the exact relationship between temperature and adsorption? Does this agree with prior research work carried out in this area?
- Is the adsorption process affected significantly by the geometry of the system?
- If the adsorption process is affected by the geometry of the system, are there geometries that outperform others?
- Is there an optimal operating condition for the adsorption process?
- If there is an optimal operating condition; what is it, and how can this be used to improve the efficiency of solar refrigerators?

1.3. OBJECTIVES

Based on the research questions above, the objectives of the research would be the following:

- To develop a finite element approach that can be used to study the adsorption problem in solar refrigerators.
- To use the finite element approach developed to study the adsorption problem, using silica gel – water working pair. The result obtained would be compared against available experimental data, and analytical results, where available, this would be done to ensure the solution method solves the problem with a reasonable degree of accuracy.

- To carry out an optimization to determine the optimal operating condition of a given system.
- To investigate the possibility of extending the same solution method to other adsorber-adsorbate working pairs.

2. METHODOLOGY

The research questions above will be answered by developing mathematical models of the problem which would then be solved numerically. The numerical solution would be obtained by using the PDE Toolbox of Matlab or the finite element java classes [1]. The numerical approach would be employed because of the difficulty involved in solving the heat equation analytically (in some instances analytical solutions do not exist for some problems) and the increasing processing power of computers which enables us to obtain more accurate numerical solutions faster than we would have been able to a couple of years ago.

The numerical solutions would be achieved by creating programs in Matlab to solve the problem and also present the results in graphical form, these programs would make use of some of the existing functions in Matlab and the PDE Toolbox, and also the finite element java classes mentioned above. In order to do this the capability of the existing functions and classes to solve the problem would have to be investigated, and the results obtained would be compared with existing analytical solution. Once it has been confirmed that the functions, or classes, are capable of solving the problem, it can then be integrated fully to form a solution method which can be used to study the behavior of the system.

Preliminary test results of some Matlab functions and Java classes are reported in section 2.1. and 2.2. These programs have the potential of being used in the final research. The codes are desired to have the capability of setting different system properties in each time step of a transient analysis, these two posses this capability. The complete integration of the codes into the solution approach would be carried out in the major research work.

By making use of the solution method obtained above, the geometry of the adsorber would be optimized by employing a multi-objective optimization routine (for example the Pareto front) which can be used to optimize for several parameter at the same time.

2.1. PRELIMINARY TESTS OF CODES FOR RADIAL PROBLEMS

These tests were carried out to determine the accuracy of the ElementRad class, the problems were set-up and then analyzed using the ElementRad class, and the obtained results were compared with those obtained from analytical results and the PDE Toolbox, this was done to determine the most suitable method to be employed in solving radial heat transfer problems which is related to the pellet problem to be solved in the major research project.

2.1.1. TEST 1: STEADY STATE RADIAL HEAT CONDUCTION WITHOUT HEAT GENERATION

This is set-up to test possibility of using the Java classes and the Matlab PDE Toolbox to solve the one dimensional steady state heat conduction problem without heat generation in radial coordinates. The problem is set-up as shown in Fig. 1 below, the top and bottom face of the rectangular domain is insulated to ensure a one dimensional heat transfer problem in which heat only flows in the radial direction. The domain was then meshed as shown in Fig. 2(b) below.

The temperature at the west face is 0^{o}C and that at the east face is 100^{o}C (i.e. Dirichlet boundary conditions), while the north and south faces are insulated (i.e. an adiabatic boundary condition). The problem is then solved using the ElementRad java class and Matlab's PDE Toolbox; the obtained results are compared with those obtained from the analytical solution of a one dimensional radial heat transfer problem, derived as shown below in equation (1). The comparison is done by computing the percentage normalized root-mean-squared error over all the nodes in the mesh; this is obtained from equation (2).

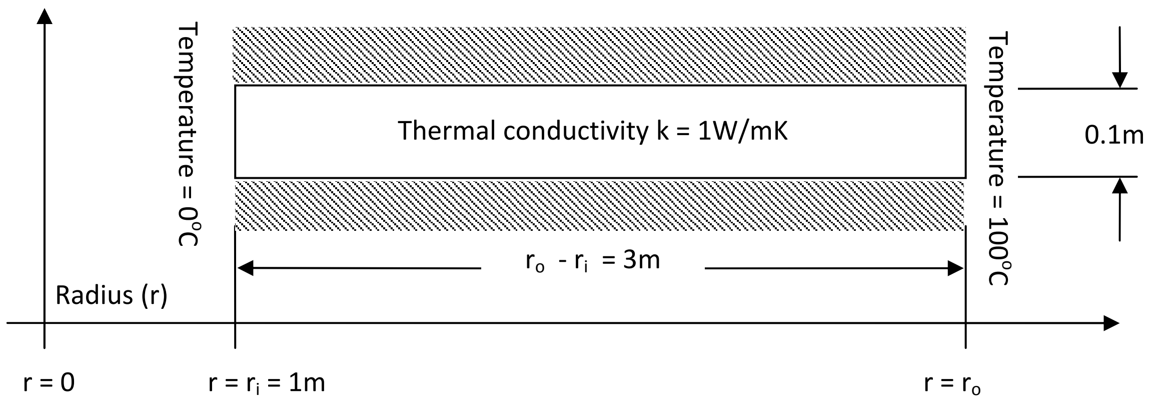


Fig. 1 Setup of steady state radial test without heat generation

The steady state radial heat equation with heat generation is given as

$$\frac{d}{dr}\left(r\frac{dT}{dr}\right) = 0$$

where the boundary conditions are given as

$$T(r_i) = T_i; \qquad T(r_o) = T_o$$

The solution of the steady state radial heat equation without heat generation is

$$T(r) = \frac{(T_o - T_i)}{ln\left(\frac{r_o}{r_i}\right)} lnr + \left[T_a - \frac{(T_o - T_i)}{ln\left(\frac{r_o}{r_i}\right)} lnr_i\right] \ldots\ldots (1)$$

The root mean squared error (RMSE)and normalized RMSE(NRMSE) are given by

$$RMSE = \sqrt{\frac{\sum_1^{N_{nodes}} (T_n - T_a)^2}{N_{nodes}}} \ldots\ldots (2.a)$$

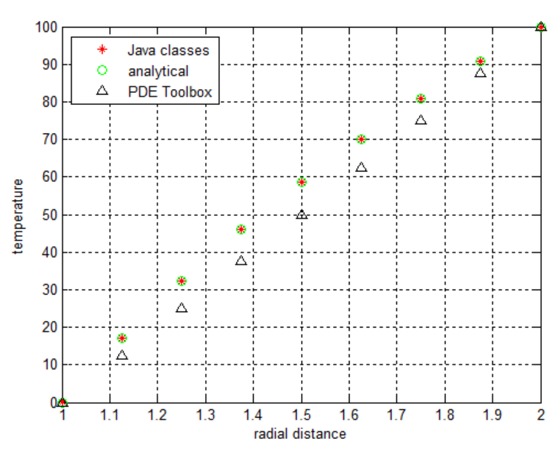
$$NRMSE(\%) = \frac{RMSE}{\max(T_a) - \min\,(T_a)} \times 100\% \ldots\ldots (2.b)$$

Where N_{nodes} = *number of nodes*; T_n = *temperature obtained from numerical method*;

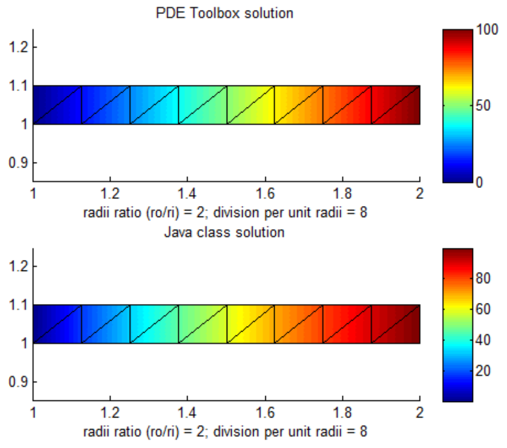
T_a = *temperature obtained from analytical solution*

RESULTS

The results of the problem were observed for different mesh sizes (or divisions per unit radii) and radii ratio (r_o/r_i). The results obtained for this test are shown below in Fig. 2 to Fig. 4 and Table 1 to Table 4.



(a)



(b)

Fig. 2 Results for radii ratio of 2 and 8 divisions per unit radii (a) Plot showing nodal temperature against radial distance for Java classes, analytical and PDE Toolbox solution (b) Surface plot showing result for PDE Toolbox and Java classes

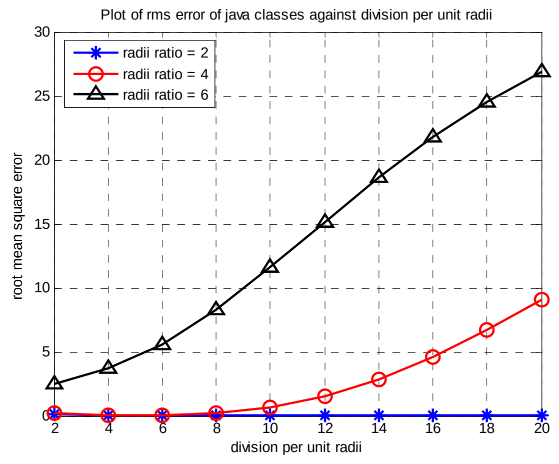
VARIATION OF DIVISION PER UNIT RADII

Number of divisions per unit radii	RMS Error (radii ratio of 2)	RMS Error (radii ratio of 4)	RMS Error (radii ratio of 6)
2	0.1586	0.2071	2.5100
4	0.0509	0.0354	3.7010
6	0.0250	0.0570	5.6115
8	0.0154	0.2428	8.3168
10	0.0108	0.6766	11.6142
12	0.0083	1.5089	15.1616
14	0.0068	2.8227	18.6281
16	0.0058	4.5964	21.7905
18	0.0051	6.7246	24.5520
20	0.0045	9.0654	26.9101

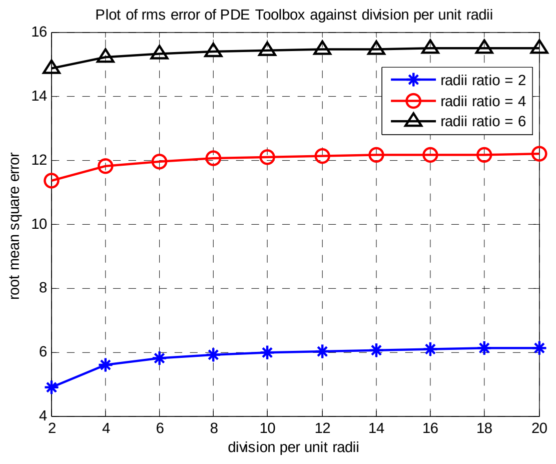
Table 1 Percentage normalized RMS error of Java classes when the division per unit radii is varied for radii ratios of 2, 4 and 6

Number of divisions per unit radii	RMS Error (radii ratio of 2)	RMS Error (radii ratio of 4)	RMS Error (radii ratio of 6)
2	4.9053	11.3574	14.8499
4	5.6003	11.8024	15.2049
6	5.8105	11.9576	15.3273
8	5.9196	12.0373	15.3895
10	5.9871	12.0857	15.4272
12	6.0332	12.1184	15.4525
14	6.0668	12.1418	15.4706
16	6.0922	12.1595	15.4843
18	6.1123	12.1733	15.4949
20	6.1284	12.1844	15.5034

Table 2 Percentage normalized RMS error of PDE Toolbox solution when the division per unit radii is varied for radii ratios of 2, 4 and 6



(a)



(b)

Fig. 3 (a) Plot of percentage normalized RMS error against division per unit radii for Java classes solution
(b) Plot of RMS error against division per unit radii for PDE Toolbox solution.

VARIATION OF RADII RATIO

The accuracy was observed for a fixed number of divisions per unit domain length (3), and with different domain lengths.

Radii ratio (ro/ri)	RMS Error (2 divisions per unit radii)	RMS Error (8 divisions per unit radii)	RMS Error (14 divisions per unit radii)
2	0.1586	0.0154	0.0068
4	0.2071	0.2428	2.8227
6	2.5100	8.3168	18.6281
8	11.6062	20.8994	30.2956
10	21.2825	29.9310	36.9204

Table 3 Percentage normalized RMS error of Java classes' solution when the radii ratio is varied for divisions per unit radii of 2, 8 and 14

Radii ratio (ro/ri)	RMS Error (2 divisions per unit radii)	RMS Error (8 divisions per unit radii)	RMS Error (14 divisions per unit radii)
2	4.9053	5.9196	6.0668
4	11.3574	12.0373	12.1418
6	14.8499	15.3895	15.4706
8	17.1919	17.6416	17.7083
10	18.9236	19.3104	19.3673

Table 4 Percentage normalized RMS error of PDE Toolbox solution when the radii ratio is varied for divisions per unit radii of 2, 8 and 14

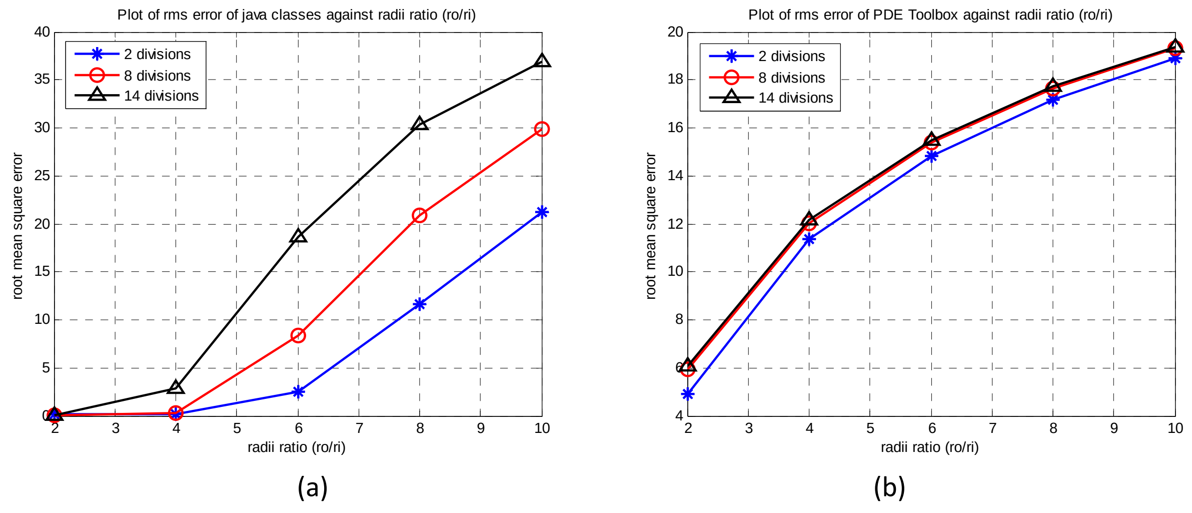


Fig. 4 (a) Plot of percentage normalized RMS error against radii ratio for Java classes solution (b) Plot of percentage normalized RMS error against radii ratio for PDE Toolbox solution.

DISCUSSION

From the results shown above, it is observed that the errors obtained for the PDE Toolbox solution are somewhat larger than those obtained for the Java classes. This is expected because the PDE Toolbox does not provide an explicit way to specify a problem as a radial problem; therefore it solves the problem above as a Cartesian one dimensional problem, whereas the problem was solved as a radial problem with the Java classes. This results in a linear temperature profile for the PDE Toolbox solution, while that of the Java classes is a logarithmic graph as can be observed from Fig. 2(a).

From the results displayed in Table 1 to Table 4, and Fig. 3 and Fig. 4; it was observed that the error in the Java classes' solution increased with an increase in radii ratio and number of divisions per unit radii.

Fig. 3(a) and Fig. 4(a) showed that errors below 1% were obtained for radii ratios less than 4 and divisions less than 8, whereas for radii ratios of 6, the lowest error value was about 2.5%.The lowest error values were seen to occur when the radii ratio was 2, at this value the errors reduced for all increase in number of divisions per domain length; this behavior is what we would expect from any numerical solution. For domain lengths greater than1m, the error first decreased and then increased as the division per domain length increased, while in other instances the error just increased with increase in division per domain length, this should not be so, as the accuracy should not reduce when a finer mesh is used.

Very little comment can be made on the errors obtained from the PDE Toolbox solution, because it solves a different problem (a Cartesian instead of radial heat conduction) and very little comparison should be made between its result and the analytical result. Some observations worth noting is that the error in its solution did not vary much with changes in division per domain length but varied greatly with increase in domain length. These two behaviours are expected because the temperature profile of the analytical solution should remain constant regardless of the number of divisions used, and that of the numerical solution also is expected to converge to a particular value; therefore the errors between these two should remain relatively constant after some time regardless of further increase in division per unit radii. On the other hand the errors should increase as the radii ratio is increased because the temperature gradient obtained from the PDE Toolbox remains linear, while that of radial analytical solution changes shape (the curvature increases) as the radii ratio is increased, therefore the error is also expected to increase. The rate of increase of the error in the PDE Toolbox decreases as the radii ratio or division per unit radii is increased and the error would be expected to level-off after some time and then remain constant.

2.1.2. TEST 2: STEADY STATE RADIAL HEAT CONDUCTION WITH HEAT GENERATION

This test is set up as shown in the Fig. 5 Below, the system has a uniform heat source term in the entire domain, the system is considered for a domain length of 3m, the result obtained using the java classes and the pde toolbox is compared with the analytical solution given in equation (3)

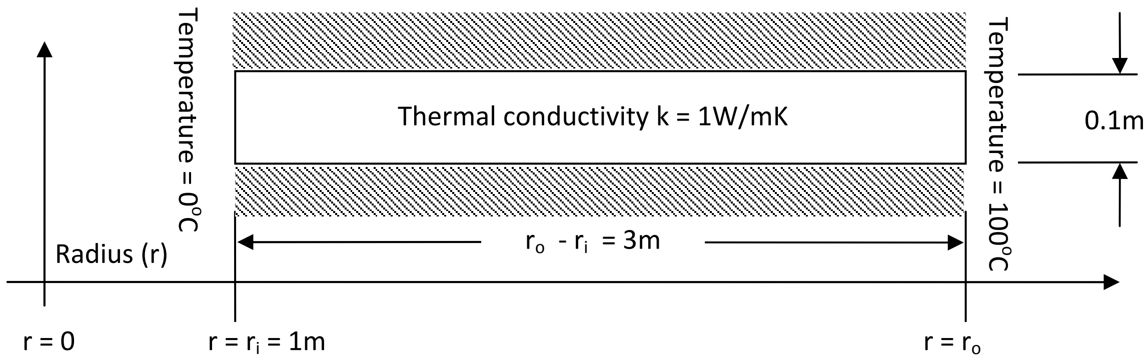


Fig. 5 Setup of steady state radial test with heat generation

The steady state radial heat equation with heat generation is given as

$$\frac{d}{dr}\left(r\frac{dT}{dr}\right) = \frac{-u'''r}{k}$$

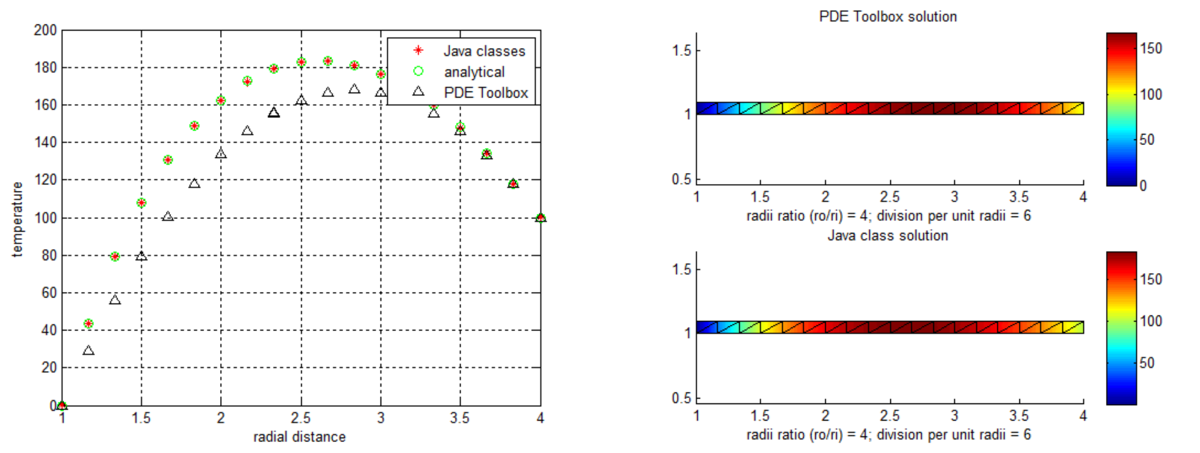
where the boundary conditions are given as

$$T(r_a) = T_a; \qquad T(r_b) = T_b$$

The solution to the steady state radial heat equation with heat generation is

$$T(r) = \frac{-u'''r^2}{4k} + \left[\frac{(T_a - T_b) + \frac{u'''(r_b^2 - r_a^2)}{4k}}{ln\left(\frac{r_b}{r_a}\right)}\right] lnr + \left[T_a + \frac{u'''r^2}{4k} - \left[\frac{(T_a - T_b) + \frac{u'''(r_b^2 - r_a^2)}{4k}}{ln\left(\frac{r_b}{r_a}\right)}\right] lnr_a\right] \text{.................................}(3)$$

RESULTS



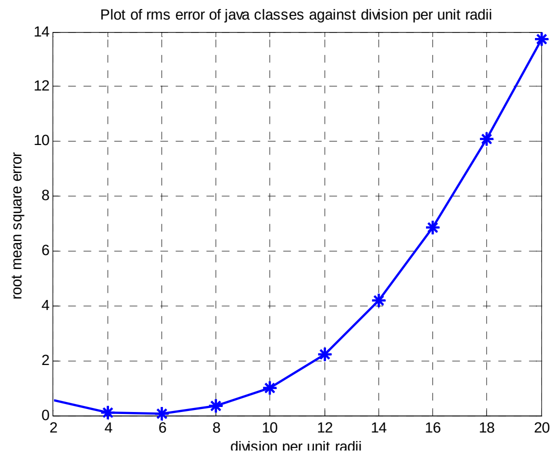
(a) (b)

Fig. 6 (a) Plot showing nodal temperature against radial distance for Java classes, analytical and PDE Toolbox solution (b) Surface plot showing result for PDE Toolbox and Java classes' solution

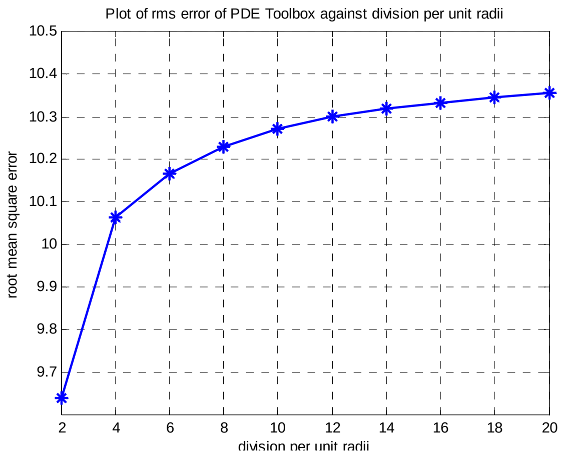
VARIATION OF DIVISION PER UNIT RADII

Number of divisions per unit radii	JAVA Classes RMS Error (radii ratio of 4)	PDE Toolbox RMS Error (radii ratio of 4)
2	0.5501	9.6409
4	0.1183	10.0650
6	0.0575	10.1676
8	0.3406	10.2293
10	0.9854	10.2714
12	2.2221	10.3017
14	4.1869	10.3200
16	6.8635	10.3335
18	10.1101	10.3452
20	13.7242	10.3554

Table 5 Percentage normalized RMS error of Java classes and PDE Toolbox solution when the division per unit radii is varied for a radii ratio of 4



(a)



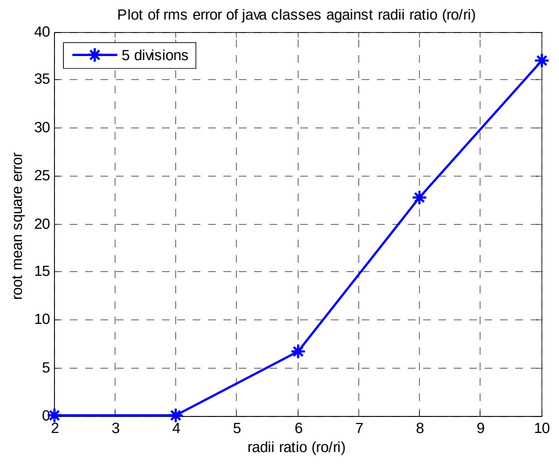
(b)

Fig. 7 (a) Plot of percentage normalized RMS error against division per unit radii for Java classes solution (b) Plot of percentage normalized RMS error against division per unit radii for PDE Toolbox solution.

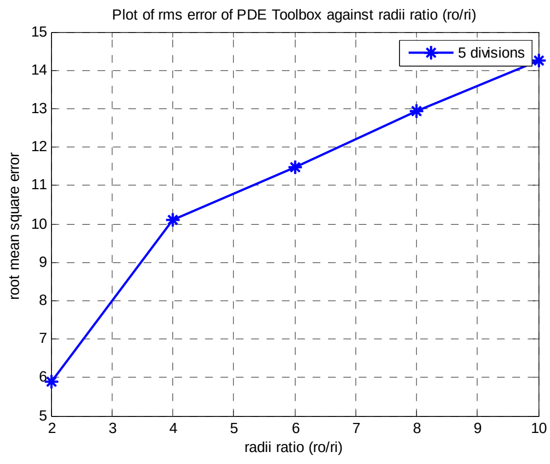
VARIATION OF RADII RATIO

Radii ratio (ro/ri)	JAVA Classes RMS Error (5 Divisions per unit radii)	PDE Toolbox RMS Error (5 Divisions per unit radii)
2	0.0606	5.8822
4	0.0401	10.1079
6	6.7687	11.4856
8	22.7899	12.9417
10	37.0030	14.2654

Table 6 Percentage normalized RMS error of Java classes and PDE Toolbox solution when radii ratio is varied for 5 divisions per unit radii



(a)



(b)

Fig. 8 (a) Plot of percentage normalized RMS error against radii ratio for Java classes solution (b) Plot of percentage normalized RMS error against radii ratio for PDE Toolbox solution.

DISCUSSION

The errors obtained in this test it somewhat similar to those obtained in the test discussed in section 2.1.1. The errors for the PDE Toolbox solution are generally larger than those for the Java classes because of the same reason stated in section 2.1.1.

The error is less than 1% for radii ratios less than or equal to 4, and with 10 or less divisions per unit length. The error values were observed to decrease then increase after 6 divisions per unit length.

The rate of increase of the error in the Java classes increased while that in the PDE Toolbox decreased, this resulted in the error value of the Java classes being greater than that of the PDE Toolbox after some time, this should not be so because the PDE Toolbox solution is not the solution of the radial problem (or a very bad approximation of the radial problem), therefore it's error values is expected to be greater than that of the Java classes solution at all times.

From the results obtained it can be inferred that; to obtain a good result from the Java classes solution, the radii ratio and division per unit radii has to be kept as small as possible, as the error seems to grow exponentially.

2.2. PRELIMINARY TEST OF CODES TO SOLVE TRANSIENT PROBLEMS

The next set of tests carried out are transient tests, these are carried out to investigate the accuracy of the various tools when applied to transient heat transfer problems, the tools would also be investigated to determine if they possess the following capabilities:

- The possibility of varying some (or all) of the properties of the system during the trasient process, for instance the value of the heat source term, as this capability would be required in the final research work
- The ability to change the boundary condition of the system in the transient process
- The ability to set spatially varying system properties like, thermal conductivity and heat source.

The two tools to be tested are the finite element Java classes (developed by Dr. Mike Tierney)[1] and the PDE Toolbox of Matlab. The results obtained from these tests are compared against the analytical solutions.

The “parabolic” function of the PDE Toolbox of Matlab was used with some slight modification to the function call; these modifications were done to give us the flexibility of setting different properties for the system in each time step (see Appendix A.1 for detailed discussion on this).

2.2.1. TEST 1: TRANSIENT TEST WITHOUT HEAT GENERATION

This test was carried out in order to compare the performance of the two codes when used in solving simple transient problems without heat generation, in which the heat transfer is one-dimensional and the system is subjected to Dirichlet boundary conditions.

This test considers a transient case set up as shown below in Fig. 9; in this test the east and west faces are subjected to Dirichlet boundary conditions which are set at 100°C and 0°C respectively. The system has no internal heat generation. The problem is that of a semi-infinite domain in which the entire system is at an initial temperature T_o = 0°C, after which the temperature of the west face is increased to 100°C and kept at this temperature while that of the east face is maintained at 0°C. This problem is that of a semi infinite domain, and the solution as obtained by Cengel[3] is shown in equation (4).

The problem was solved using the Java classes and PDE Toolbox of Matlab, the results were then compared with the analytical solution obtained using the complementary error function solution; equation (4).

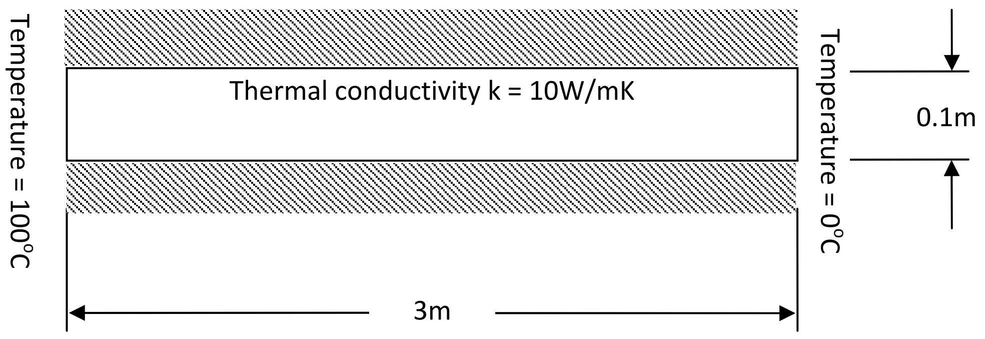


Fig. 9 Setup of the trasnsient test without heat generation

The solution for this transient problem as obtained by Cengel [3] is given as

$$\frac{T(x,t)-T_i}{T_s-T_i} = erfc\left(\frac{x}{2\sqrt{(\alpha t)}}\right) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

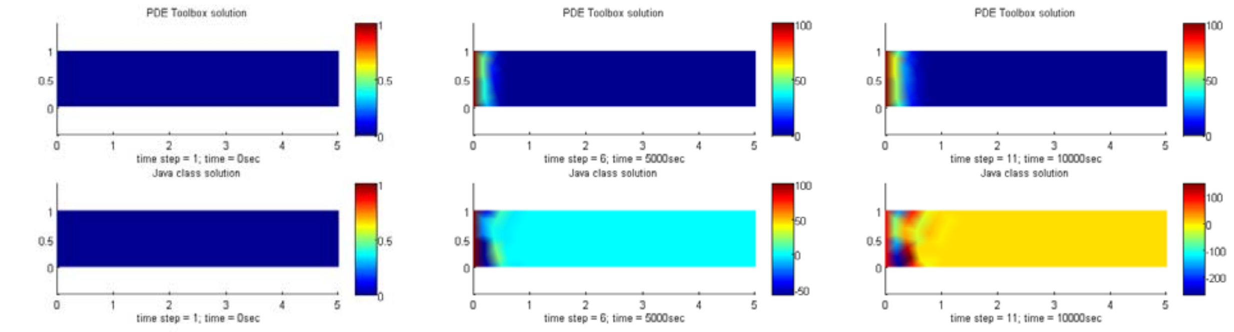
where; T_i *is the initial temperature which is also the temperature on the east face;*

T_s *is the surface temperature which is the temperature on the west face;*

$$erfc(\xi) = 1 - \frac{2}{\sqrt{\pi}}\int_0^{\xi} e^{-u^2}du\,; \qquad \xi = \frac{x}{2\sqrt{(\alpha t)}}\,; \quad and \quad \alpha = \frac{k}{\rho c}$$

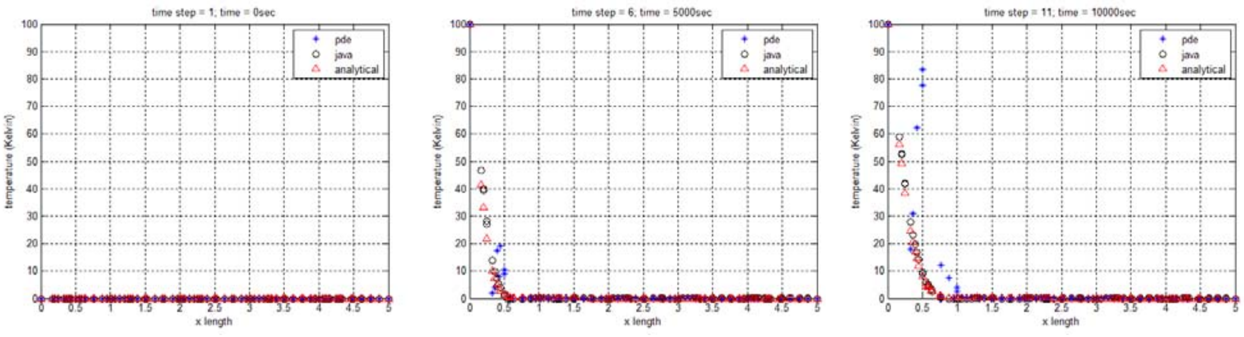
RESULTS

Results for time step = 1,000 seconds (Fourier number = 1.6461e-004)



(a) (b) (c)

Fig. 10 (a) Surface plot of temperature at time = 0sec (b) Surface plot of temperature at time = 5,000sec (c) Surface plot of temperature at time = 10,000sec



(a) (b) (c)

Fig. 11 (a) Plot of nodal temperatures against x length at time = 0sec (b) plot of nodal temperature against x length at time = 5,000sec (c) Plot of nodal temperature against x length for time = 10,000sec

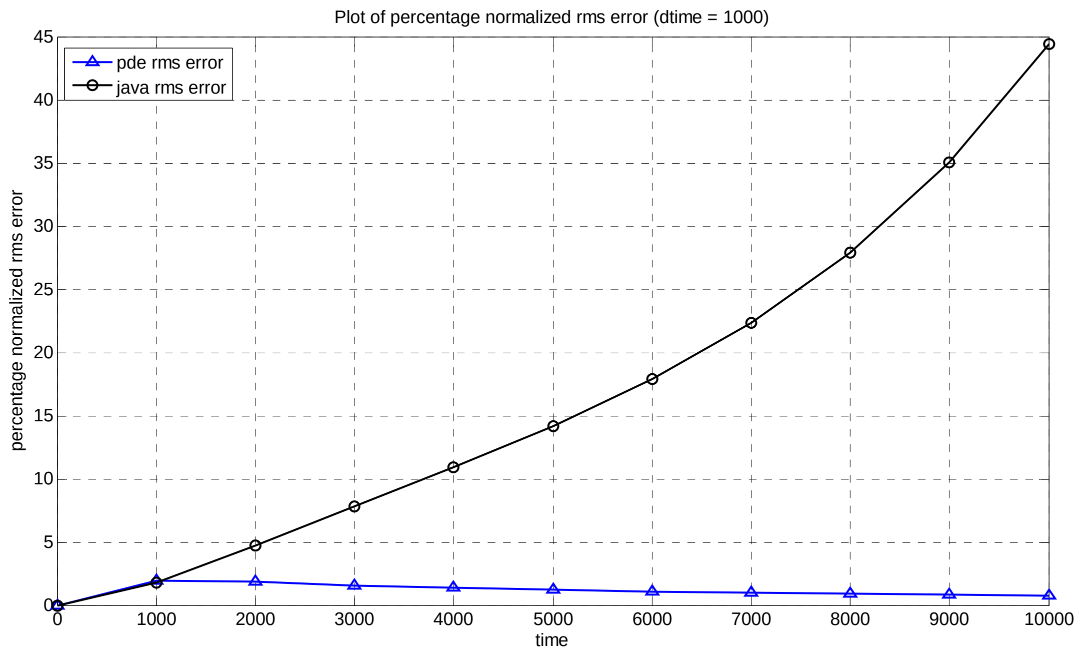
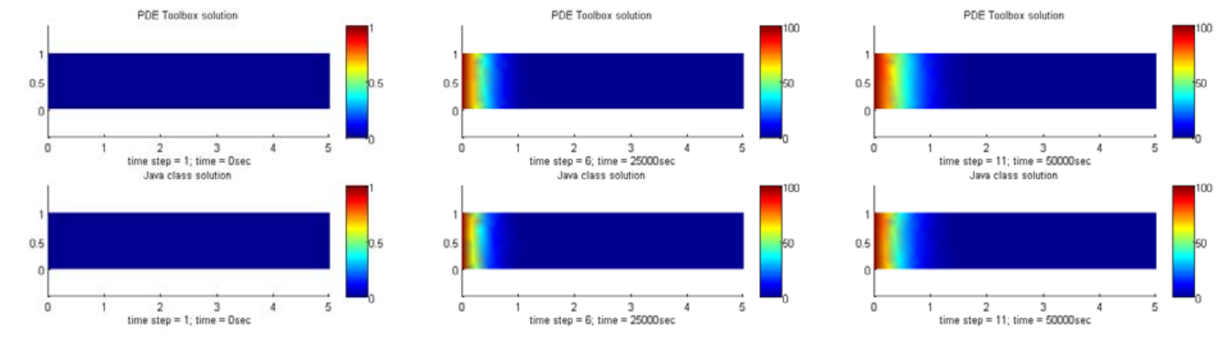


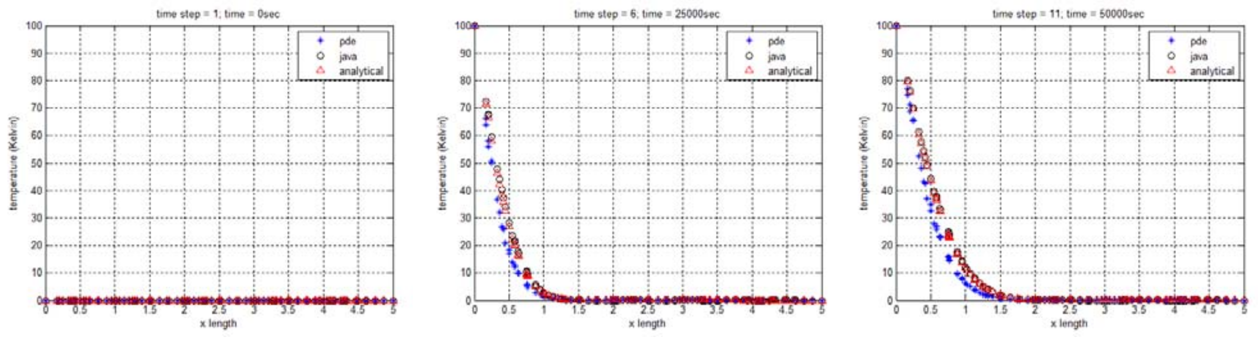
Fig. 12 Plot of percentage normalized RMS error against time

Results for time step = 5,000 seconds (Fourier number = 8.2305e-004)



(a) (b) (c)

Fig. 13 (a) Surface plot of temperature at time = 0sec (b) Surface plot of temperature at time = 25,000sec (c) Surface plot of temperature at time = 50,000sec.



(a) (b) (c)

Fig. 14 (a) Plot of nodal temperatures against x length at time = 0sec (b) plot of nodal temperature against x length at time = 25,000sec (c) Plot of nodal temperature against x length for time = 50,000sec.

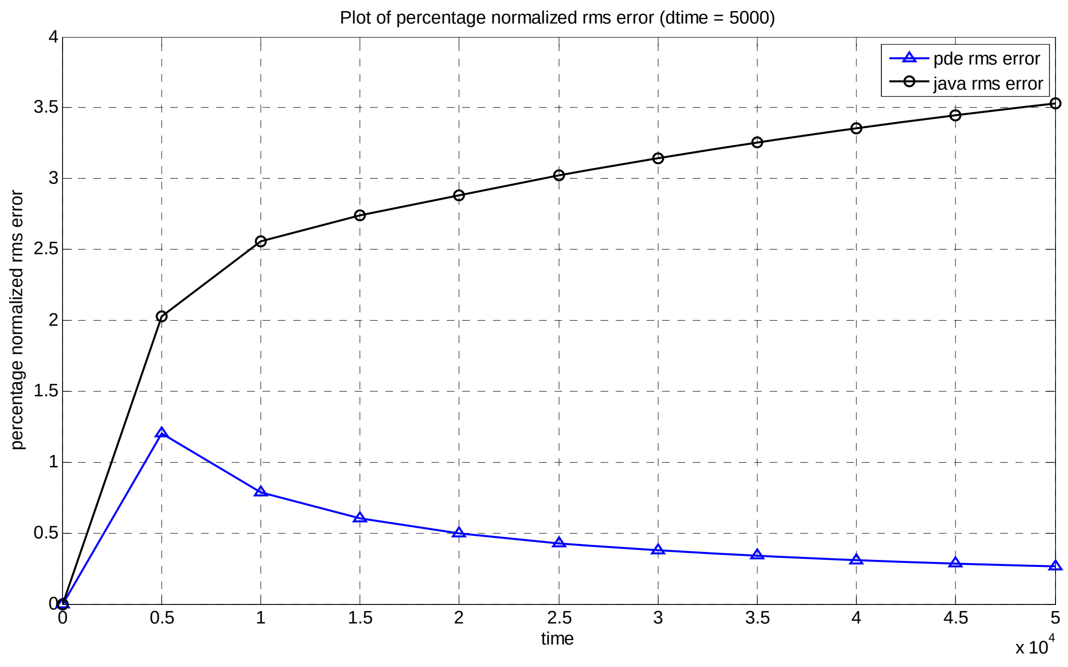
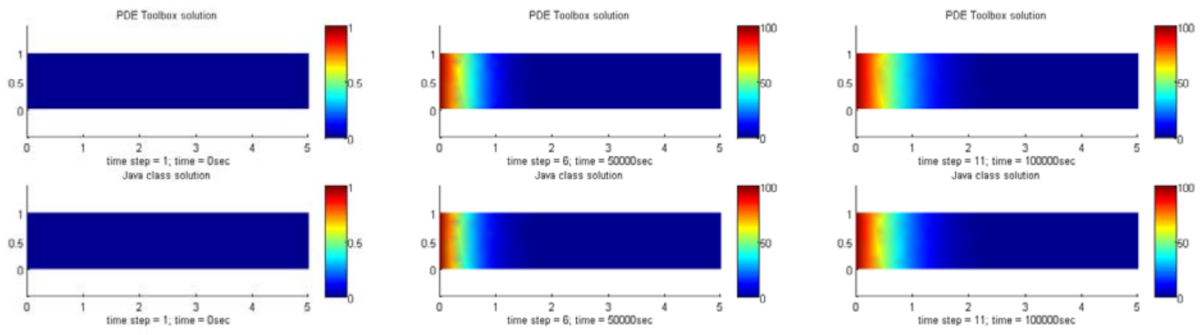


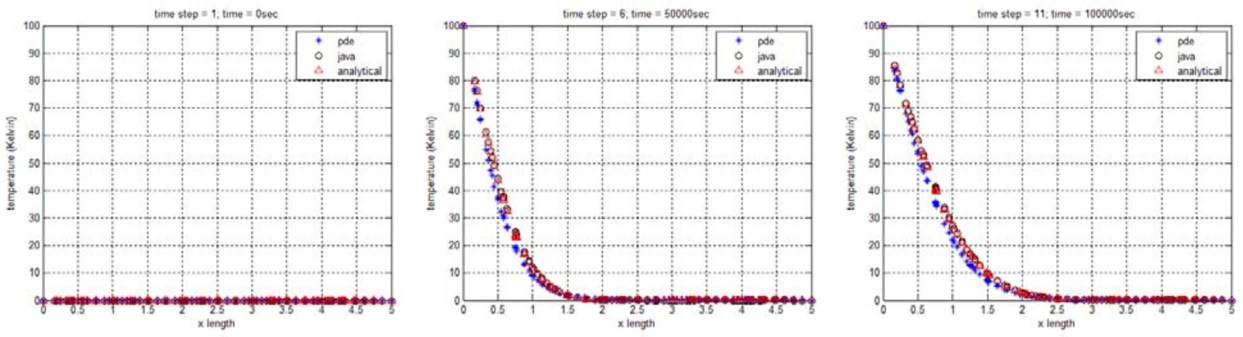
Fig. 15 Plot of percentage normalized RMS error against time.

Results for time step = 10,000 seconds (Fourier number = 0.0016)



(a) (b) (c)

Fig. 16 (a) Surface plot of temperature at time = 0sec (b) Surface plot of temperature at time = 50,000sec (c) Surface plot of temperature at time = 100,000sec.



(a) (b) (c)

Fig. 17 (a) Plot of nodal temperatures against x length at time = 0sec (b) plot of nodal temperature against x length at time = 50,000sec (c) Plot of nodal temperature against x length for time = 100,000sec.

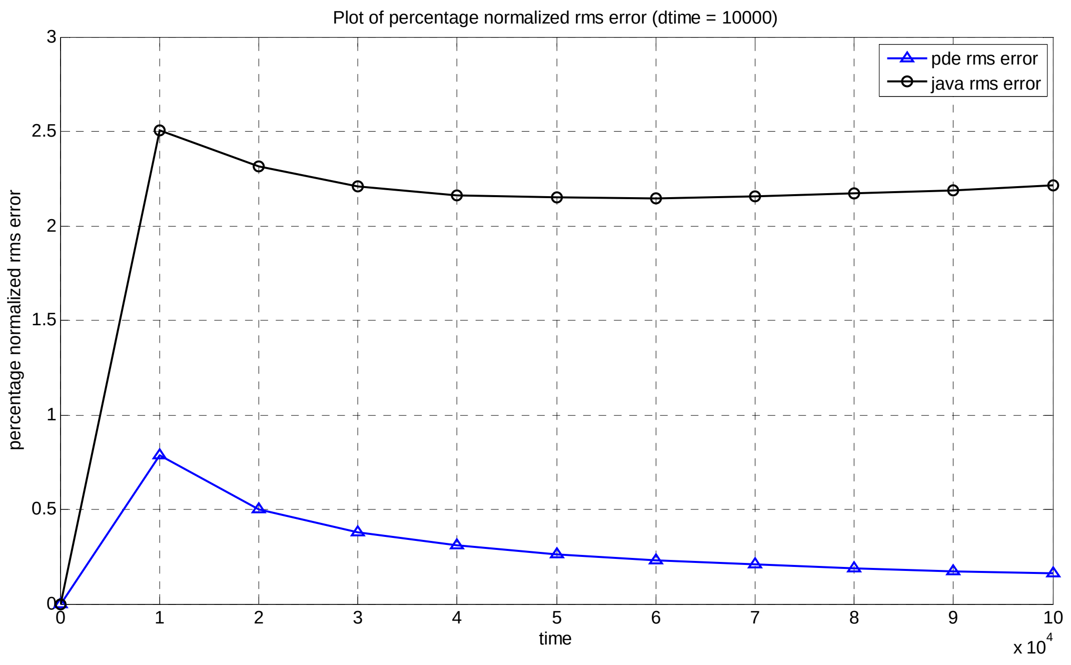


Fig. 18 Plot of percentage normalized RMS error against time (time step = 10,000sec and thermal conductivity = 10W/mK; Fourier Number = 0.0016).

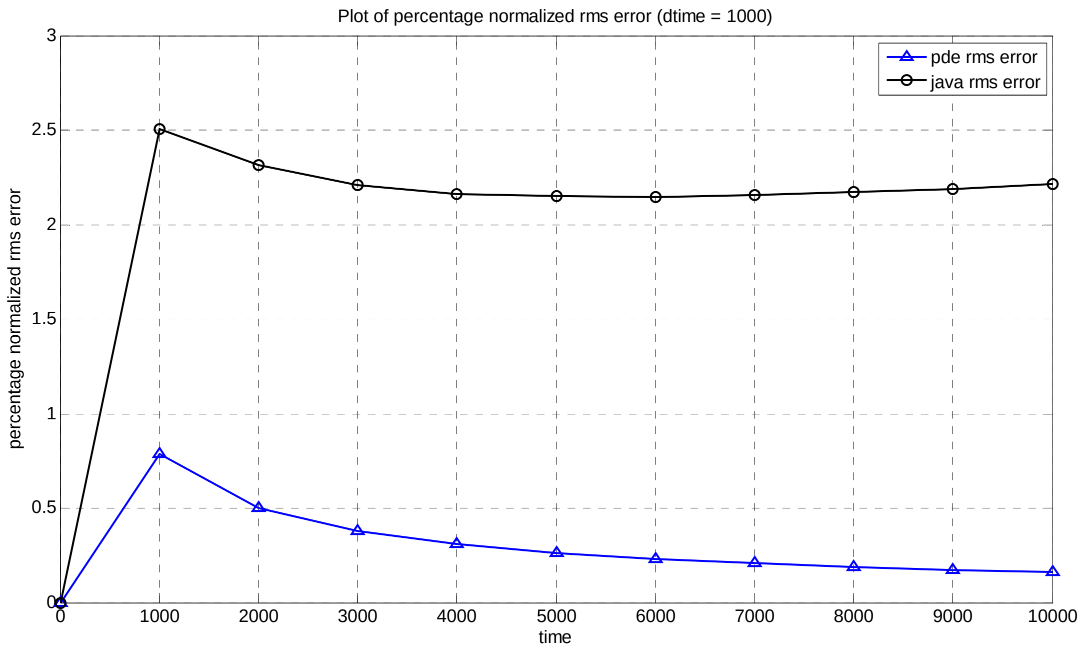


Fig. 19 Plot of percentage normalized RMS error against time (time step = 1,000sec and thermal conductivity = 100W/mK; Fourier Number = 0.0016)

DISCUSSION

From Fig. 12, Fig. 15 and Fig. 18; it is observed that the error increases sharply from zero at time = 0sec, this is because at time = 0sec, the system is at the initial state and no computation has been done at that time therefore the error is zero.

We observe a trend in the errors obtained for the PDE Toolbox solution; the error starts from about 2% when the time step is 1,000sec and then decreases after every successive time step, the same behavior is observed for the other two cases, i.e. when the time step is 5,000sec and 10,000sec, except that the error starts from lower values for greater time steps; for a time step of 5,000sec the error starts from about 1.2% while for 10,000sec it starts from about 0.75%.

The Java solution errors increased for every time step when the time step was 1000sec, and 5,000sec, but the rate at which the error increased (i.e. the gradient of the error curve) increased for a time step of 1,000sec, while it decreased for a time step of 5,000sec. As the time step was further increased to 10,000sec the errors decreased and then increased, but the slope of the error was more or less horizontal suggesting that the error values probably not go above the initial value obtained after the first time step.

The errors are related to the Fourier number, similar problems with the same Fourier number produce similar error curves, as observed from Fig. 18 and Fig. 19. The graph of Fig. 18 is for a time step of 10,000sec with a thermal conductivity of 10W/mK, while the graph of Fig. 19 is for a time step of 1,000sec with a thermal conductivity of 100W/mK, but both cases have the same Fourier number of 0.0016.

2.2.2. TEST 2: TRANSIENT WITH HEAT GENERATION

This test was carried out to compare the performance of the two codes when used in solving transient heat conduction problem with internal heat generation, similar to the one which we might encounter when dealing with the silica-gel pellet problem, the result obtained from the two codes are compared against the analytical result obtained from equation (5).

This problem involves a body with internal heat generation (similar to a nuclear fuel rod or an ohmic heating device), the body is assumed to be infinitely long in two directions; therefore one dimensional heat conduction is assumed. The whole body is initially at a temperature T_∞ before the internal heat generation comes into play. After the internal heat generation is turned on, the temperature of the faces in the conduction direction are maintained at the initial temperature T_∞ , this gives rise to a symmetrical problem which can be idealized, by considering half of the domain of the problem with the plane of symmetry acting as an adiabatic boundary, this is shown in Fig. 20.

The problem is set-up and solved as shown in Fig. 20, the solution obtained is then compared with analytical results obtained using equation (5); with $n = 500$.

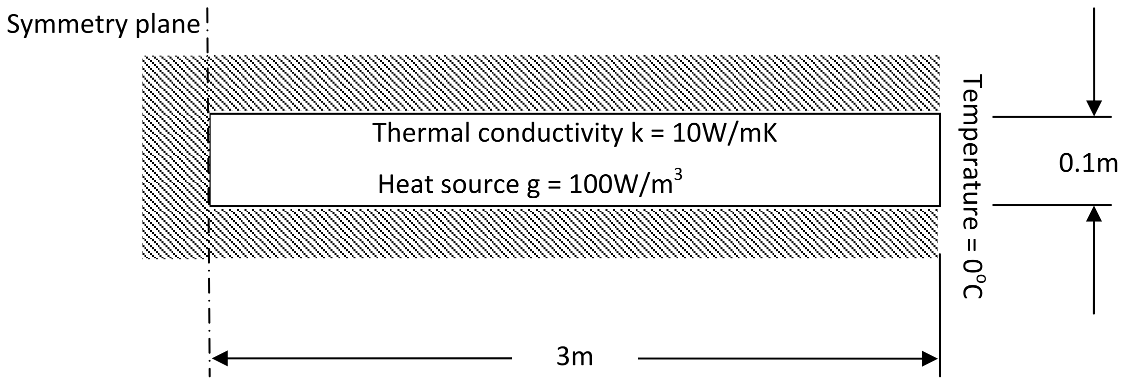


Fig. 20 Setup of transient test with heat generation

the heat equation with heat generation term given as

$$\frac{\partial\theta}{\partial t} = a\frac{\partial^2\theta}{\partial x^2} + \frac{u'''}{\rho c}$$

where the initial and boundary conditions aregiven as

$$\theta(x,0) = 0; \qquad \frac{\partial\theta}{\partial t}(0,t) = 0; \qquad \theta(L,t) = 0$$

$$where\ \ a = \frac{k}{\rho c}; \qquad \theta = T - T_\infty;$$

According to Arpaci [4] solution of the heat equation obtained by inversion theorem gives:

$$\theta(x,t) = \left(\frac{u'''L^2}{k}\right)\left[\frac{1}{2}\left[1-\left(\frac{x}{L}\right)^2\right] - 2\left[\sum_{n=0}^{\infty}\frac{(-1)^n}{(\lambda_n L)^3}e^{-a\lambda_n{}^2 t}cos(\lambda_n x)\right]\right] \text{..........................} (5)$$

$$where\ \lambda_n = \frac{1}{L}\left[(2n+1)\frac{\pi}{2}\right]$$

RESULTS

Results for time step = 1000 sec (Fourier number = 1.6461e-004)

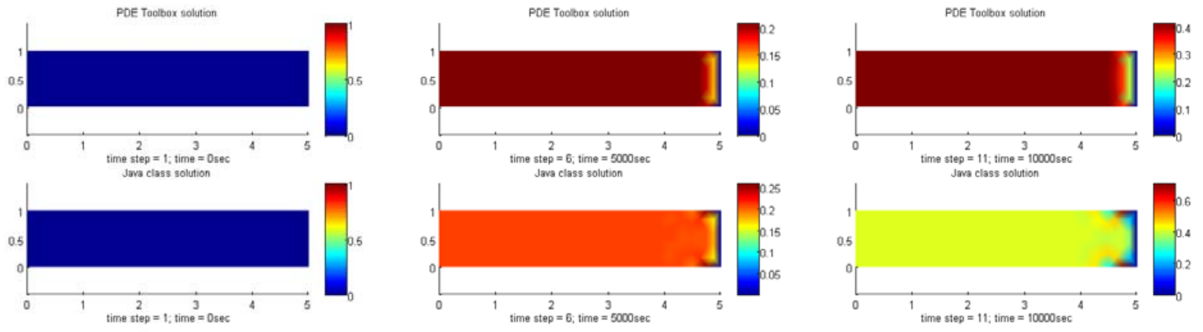


Fig. 21 (a) Surface plot of temperature at time = 0sec (b) Surface plot of temperature at time = 5000sec (c) Surface plot of temperature at time = 10,000sec

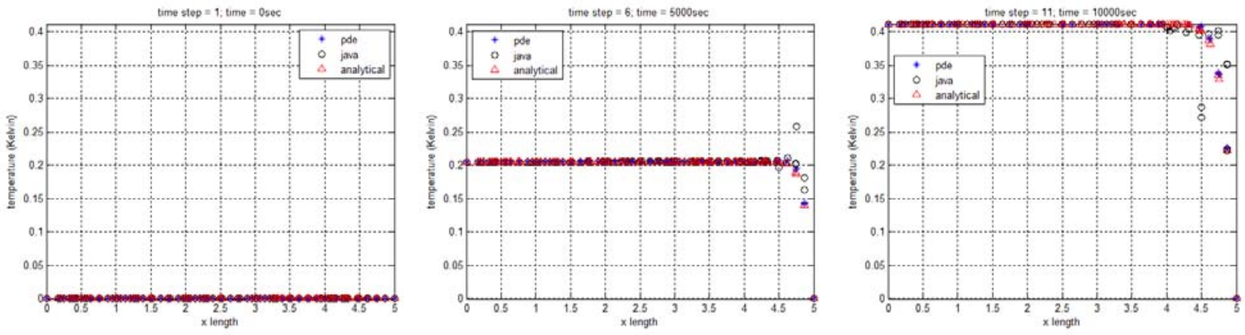


Fig. 22 (a) Plot of nodal temperatures against x length at time = 0sec (b) plot of nodal temperature against x length at time = 5,000sec (c) Plot of nodal temperature against x length for time = 10,000sec.

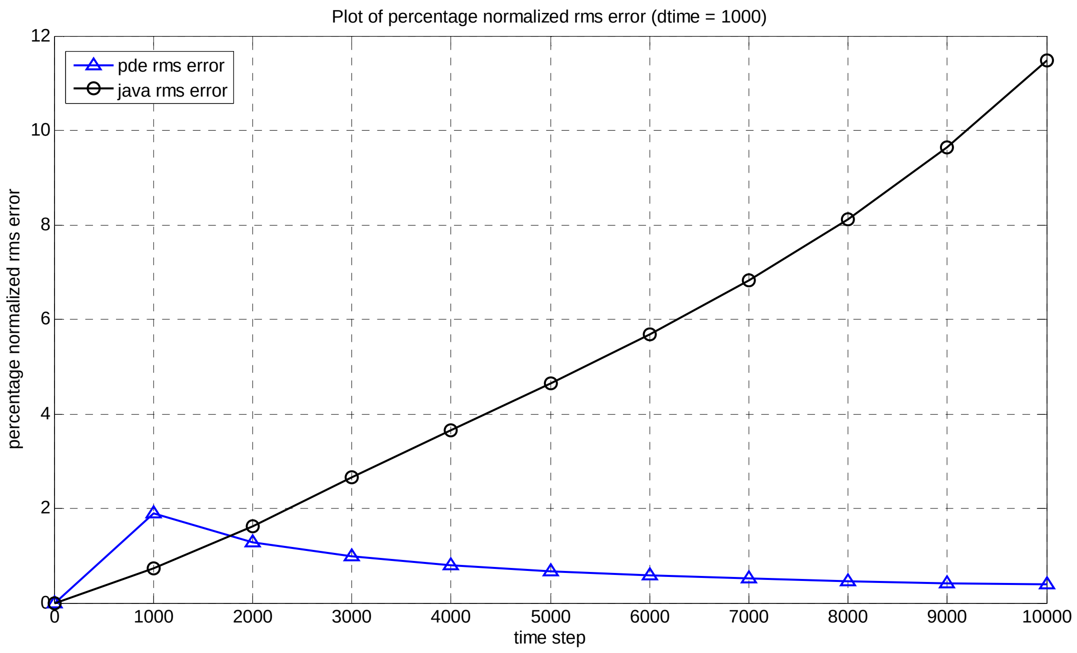


Fig. 23 Plot of percentage normalized RMS error against time.

Results for time step = 5,000 seconds (Fourier number = 8.2305e-004)

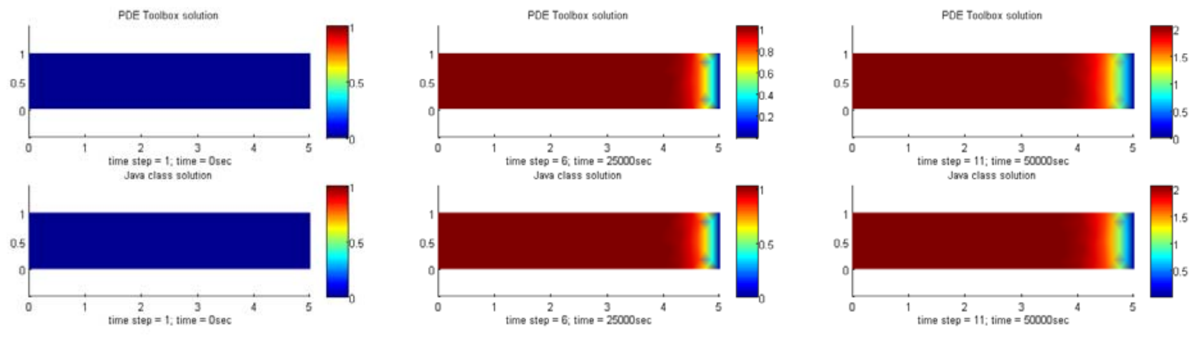


Fig. 24 (a) Surface plot of temperature at time = 0sec (b) Surface plot of temperature at time = 25,000sec (c) Surface plot of temperature at time = 50,000sec

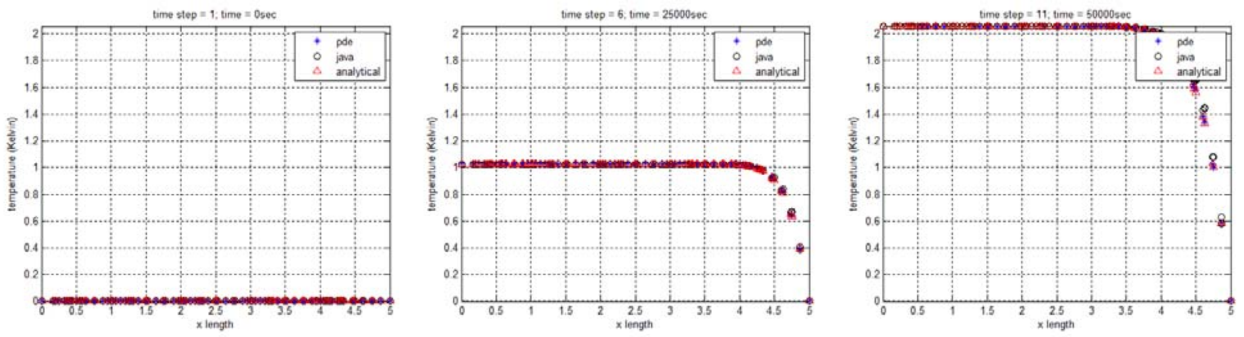


Fig. 25 (a) Plot of nodal temperatures against x length at time = 0sec (b) plot of nodal temperature against x length at time = 25,000sec (c) Plot of nodal temperature against x length for time = 50,000sec.

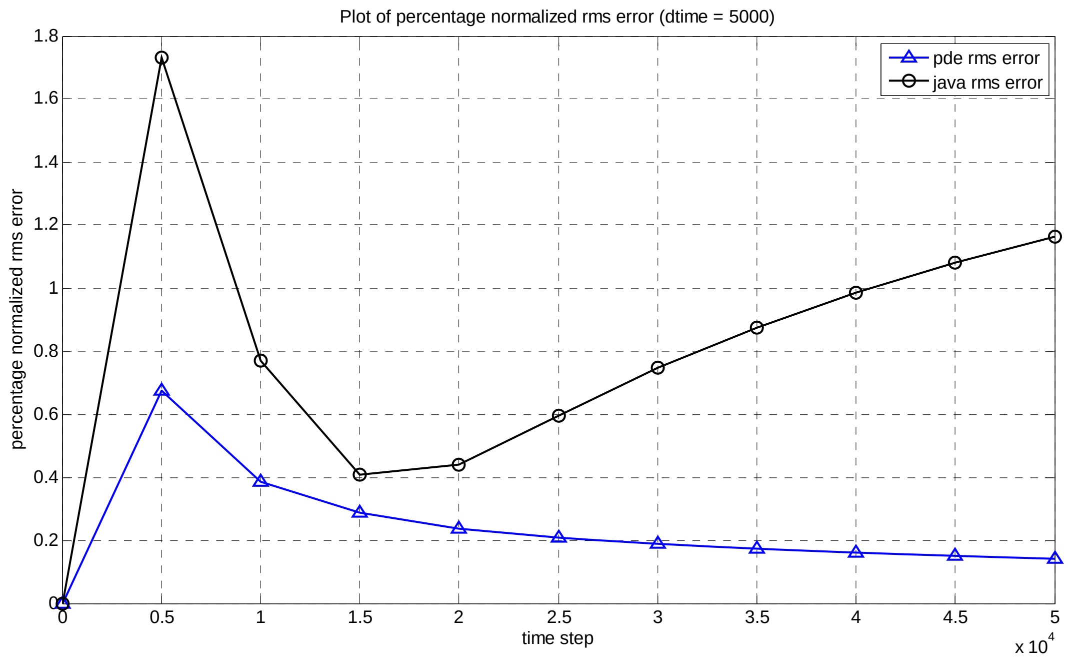


Fig. 26 Plot of percentage normalized RMS error against time.

Results for time step = 10000 seconds (Fourier number = 0.0016)

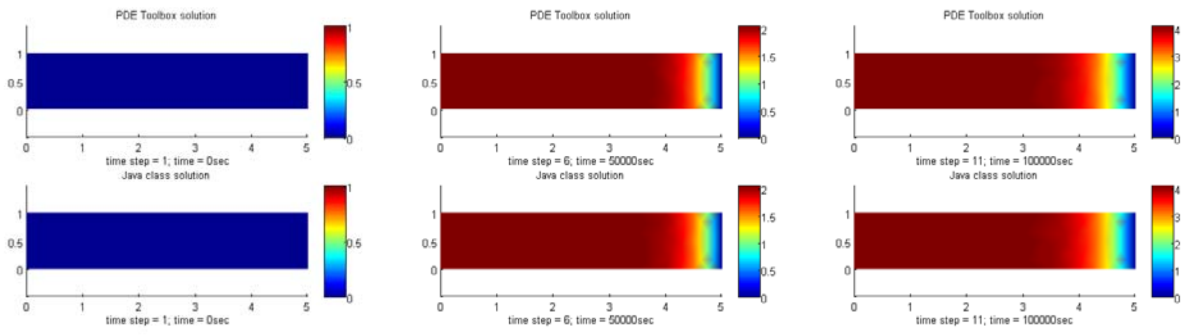


Fig. 27 (a) Surface plot of temperature at time = 0sec (b) Surface plot of temperature at time = 50,000sec (c) Surface plot of temperature at time = 100,000sec

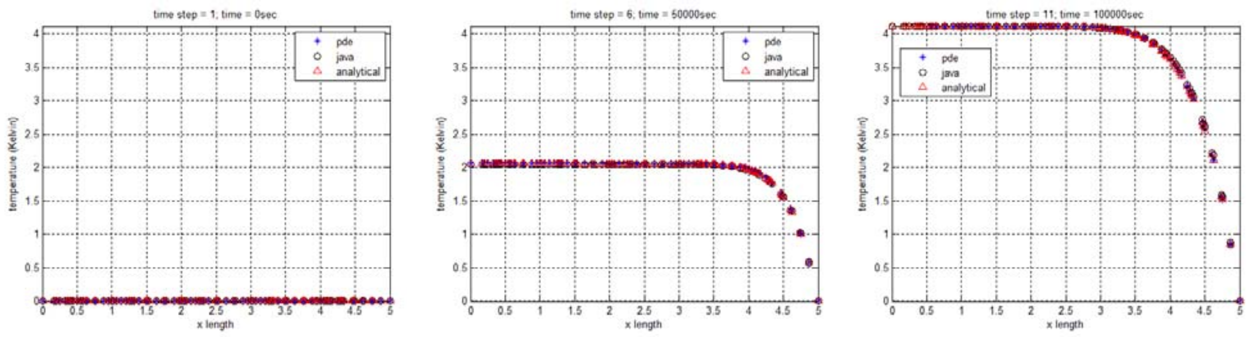


Fig. 28 (a) Plot of nodal temperatures against x length at time = 0sec (b) plot of nodal temperature against x length at time = 50,000sec (c) Plot of nodal temperature against x length for time = 100,000 sec.

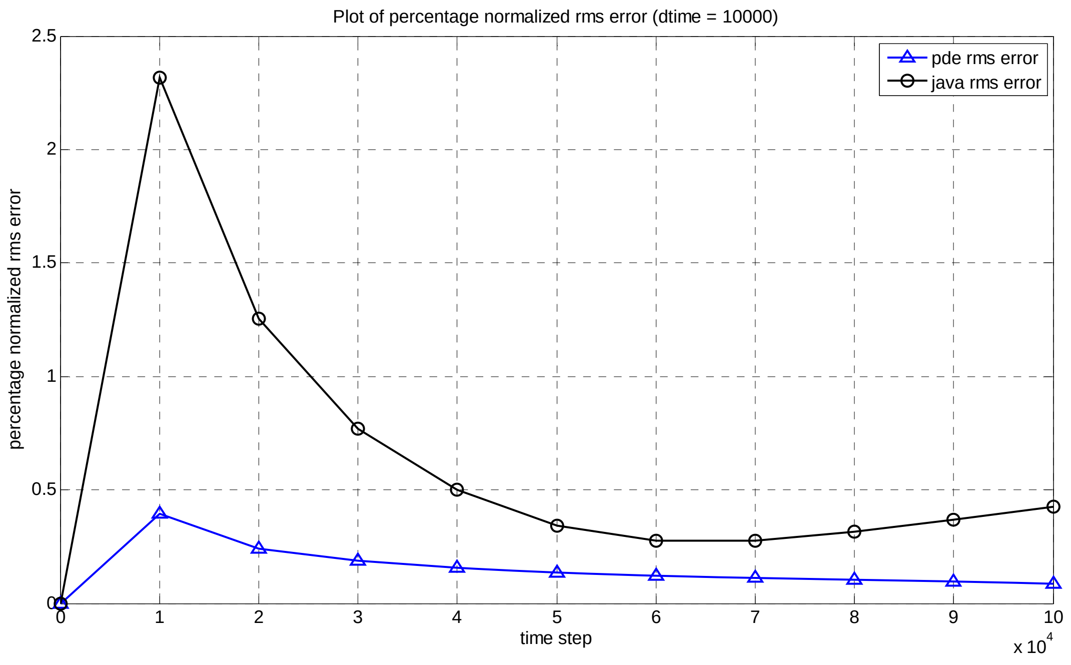


Fig. 29 Plot of percentage normalized RMS error against time (for time step = 10,000sec and 10 time steps).

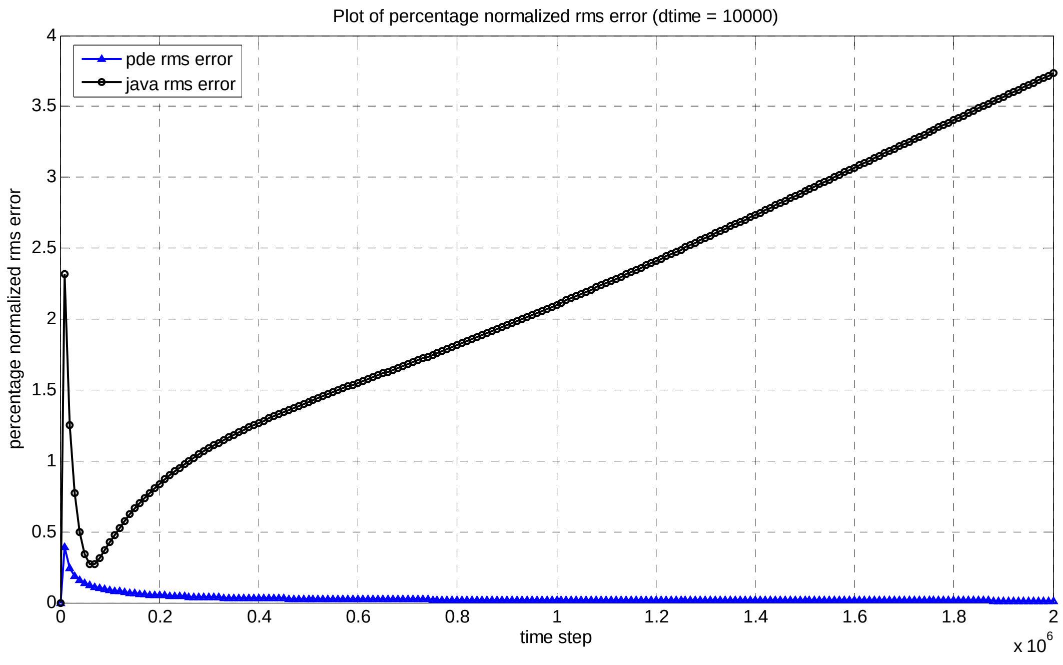


Fig. 30 Plot of percentage normalized RMS error against time(for time step = 10,000sec and 200 time steps).

DISCUSSION

The errors behave somewhat similar to the previous test discussed in section 2.2.1., for small values of time step (or Fourier number), the error form the Java classes solution increases with every successive time step. Whereas for the PDE Toolbox solution the error decreases with every successive time step regardless of the time step chosen.

For larger time steps, the error decreases and then increases after attaining a minimum value, as shown in Fig. 30, but the PDE Toolbox solution decreases continuously.

We can therefore conclude that in order to obtain an accurate result from the Java classes solution, the size of the time step should be made as large as possible and the number of time steps should also be kept close to the value at which the minimum error occurs. While the PDE Toolbox produces accurate results regardless of the size of the time step chosen or the number of time steps over which the solution is computed.

3. SCHEDULE OF WORK

The research would be carried out in four phases which are itemized below:

1. The first phase would involve the development of a suitable finite element solution approach which would be used in solving the adsorption problem; this phase would involve the creation of programs which could be used in solving transient heat transfer problems with variable internal heat generation.

   The PDE Toolbox of Matlab[5] and the Finite Element Java classes created by [1] would be used in this phase, this is done because it would give us two solution approaches to compare against each other.

2. The second phase would involve; applying the solution approach developed in the first phase to solve an adsorption problem, and then analyzing the result obtained to ensure that the solution approach developed functions properly and accurately for the problem it is intended for. In essence it involves testing the code to ensure that it is “fit for purpose”. This phase would also involve using the solution approach (and the computer model) to study the adsorption process in order to gain in-depth understanding of the process, the factors that inhibits it and how to optimize the system for best efficiency.
3. The third phase would involve; running an optimization scheme which tests various geometries and operating conditions to determine an optimal operating condition for given geometries, and also to draw relationships with the results obtained which might be used in designing a solar refrigerator to operate at its optimal conditions.
4. The last phase of the research would be to investigate the possibility of extending this solution approach to other adsorber-adsorbate working pairs.

The schedule with which I intend to carry out the research is shown below

Week (Start date)	Activity
Week 1 (June 13, 2011)	Develop the complete solution approach to be used for the analysis
Week 2 (June 20, 2011)	Carry out analysis, and study results
Week 3 (June 27, 2011)	Carryout analysis, and study results
Week 4 (July 4, 2011)	Investigate ways of optimizing problem
Week 5 (July 11, 2011)	Optimize problem
Week 6 (July 18, 2011)	Prepare mid-session report.
Week 7 (July 25, 2011)	Optimize problem
Week 8 (Aug 1, 2011)	Investigate the possibility of extending this solution approach to other adsorber-adsorbate working pairs.
Week 9 (Aug 8, 2011)	Prepare final draft of thesis, to be submitted to supervisor.
Week 10 (Aug 15, 2011)	Submit draft of final thesis to supervisor for discussion and inspection.
Week 11 (Aug 22, 2011)	Make necessary corrections and amendments to thesis.
Week 12 (Aug 29, 2011)	Print, bind and submit final thesis.

Table 7 Proposed schedule of research work

4. RESOURCE AND COST ESTIMATION

COST ESTIMATION

The project does not involve any significant cost implication as the major software to be used is Matlab, which is available on most of the faculty computers.

RESOURCES REQUIRED

- Matlab and PDE Toolbox: This is available on most faculty computers
- Two-Dimensional Heat Conduction Finite Element Java Classes: This is provided by Dr. Mike Tierney
- Java Development Kit and Runtime Environment: Installed on faculty computers or freely available on the Internet.

5. CONCLUSION

The test results above show that the radial case is better handled by the Java classes as PDE Toolbox tends to solve the problem in the Cartesian coordinates and not radial. This happens because the PDE Toolbox does not have a scheme for solving radial problems.

The solution obtained with the PDE Toolbox was more accurate than that obtained from the Java classes when a transient problem was considered. The difference in accuracy was more pronounced for small Fourier numbers and a large number of time steps, but otherwise both approaches performed reasonably well, but the PDE Toolbox gives us a better resolution in the time domain because it is able to handle smaller time steps.

Subsequent to these tests been carried out the time stepping algorithm of the Java classes has been corrected in line with Lewis et.al, but due to time constraints the corrected codes are yet to be tested.

The two approaches, the PDE Toolbox and the Java classes, offer us the flexibility required; therefore these two approaches would be used in the major research as this would give us two solutions which can then be compared with each other and with experimental data, where available.

REFERENCES

1. Tierney, M.J., *2-dimensional heat conduction finite element Java classes*. 2010, Tierney, M.J: Bristol, U.K. p. Java class to instantiate element node matrix object.
2. Dada, A.O., *Examples and research in the Galerkin Finite Element Method: From a viewpoint of application to adsorption-chillers*. 2010, University of Bristol: Bristol, UK.
3. Cengel, Y.A., *Heat transfer: A practical approach*. Second Edition ed. 2002: McGraw-Hill.
4. Arpaci, V.S., *Conduction heat transfer*. 1966: Addison-Wesley Publishing Company.
5. Mathworks Inc, *PDE Toolbox*. 2010, Mathworks, Inc.

APPENDIX

A DESCRIPTION OF MAIN MATLAB FUCTIONS USED

A.1 u=assempde(b,p,e,t,c,a,f)

The input variables of the function "b, p, e and t" are generated by the PDE Toolbox or a similar meshing algorithm; "b" is the boundary matrix, "p" is the point matrix, "e" is the edge matrix, and "t" is the triangle matrix. The other input variables "c, a and f" are parameters in the PDE specification; "c" is the thermal conductivity of the material, the term "a" does not exist in the heat equation, therefore it is set to zero, and "f" is the internal heat generation value.

A.2. u1=parabolic(u0,tlist,b,p,e,t,c,a,f,d)

The variables "b, p, e, t, c, a, and f" are the same as defined above in Appendix A.1. While the variable "u0" is the initial value of the temperature, "d" is the volumetric heat capacity, and "tlist" is an vector array of the time over which the temperatures would be calculated.

The implementation of the function requires that the time list be specified in the vector "tlist" for all time steps, if this function is implemented this way it would be impossible to set different properties for the system in each time step. In order to overcome the problem above, the function is implemented as shown below:

```
for time = 1:length(tlist)
    u1_temp = parabolic(u0,[0 dtime],b,p,e,t,c,a,f,d);
    u1(:, time+1) = u1_temp(:,2);
    u0 = u1_temp(:,2);
end
```

This involves passing an argument "tlist" which is a vector of two numbers; 0 and dtime (the time step or time increment), by doing this the function calculates the temperature over a single time step and then the answer obtained is set as the initial value for the next iteration in the loop.