**Coursework 2**

**Solution 1a**

Apply the Laplace transform with respect to time t to the heat conduction equation:

Insert initial conditions into the transformed equation to get:

Now rearrange to get inhomogeneity on the RHS and we have:

(1)

We can now solve this ODE by finding the characteristic polynomial:

Which calculates the roots. For large positive values of x and shift by 0.5π to better handle boundary condition, the complementary function becomes:

Now, since the RHS consist of cosine function, a trial solution for the particular integral is:

Where P and Q are must be determined. We find the second derivative of Up with respect to x and substitute this and Up into Equation 1 above:

Rearranging above equation in terms of sine and cosine to get:

We get the following equations to solve for P and Q:

⇒

⇒

Thus the general solution U(x,s) is:

Now we can apply boundary condition to find the functions *A(s)* and *B(s)*. The equivalent Laplace transform boundary conditions are and. We apply this to general solution above:

For boundary we get:

And for we get:

We can now derive the expression for A(s) and B(s):

⇒

⇒

Thus the Laplace domain solution is

**Solution 2b**

Using Table 2 we apply the Laplace transform to boundary conditions to get

Applying and and get Laplace domain solution in required form

Now we use the given formula for

Which has the inverse

For the time domain solution is

We use MATLAB to plot Figure 1 showing contour and Figure 2 showing surface of the analytical solution for 20 terms of the series



Figure 2

Figure 1

From the series solution we can see that the magnitude decrease as 1/n3. This suggests a high rate convergence and therefore large number of terms are not required. We investigate the convergence for numbers of term N from 1 to 50 for the series. We do this by calculating the difference between solutions for each term used with a step of 5. For simplicity we use generated analytical solution values in MATLAB for a single grid point and calculate difference for each N term. For tolerance of 0.05% we calculate that it is sufficient to use N≥10.

**Solution 1c**

We choose to solve the problem numerically using FTCS scheme. Although it is unconditionally stable, in contrast to Crank-Nicholson, the scheme will provide an accurate solution of the heat equation after a stability test.

First we must define the grid in both the x and t direction and their respective mesh spacing. We define the mesh as following

Before we start the code of these two functions, we need to write down all the given parameters, boundary conditions and the initial condition as shown in part a. Then we need setup the ETM function which is an explicit time marching method, so we put time increment of dt and time stop into this function. For the FTCS part, which it is a forward time-centered space function, both of them are the functions to solve transient partial derivatives functions. Inside the code, we need to call the FTCS function inside the ETM function, but we need to write these functions in two separate codes. The idea of the ETM function is to first replace each derivative by Finite Difference stencils, then use the second order central difference approximation.

As we write down all the parameters in ETM code, setup a function of time loop to show the initial condition in ETM function. We set an array to record all the value we gain inside function, then we call the FTCS inside the loop. So, we also need a separate FTCS code to make it much clearly, the FTCS function is the forward time scheme centered space for the one-dimension diffusion equation so that we only need to alter the Rx which shows Rx = kappa\*dt/(dx^2). The kappa is 0.1 as shown.

As we get the FTCS code, back to the ETM function because there is FTCS function in the loop. In the ETM code, when we got the FTCS code and type the boundary condition inside the loop, we can eventually get the total of T(bigT).

Finally, back to the main code, we put all the same parameter and initial condition again which is similar with part b. However, we found that if we still use the dt=0.04 from part b substituting in numerical part, the figure of numerical part does not make sense, which means the time increment here is too big. We change the dt to 0.02 which get the similar figure comparing with the analytical solution. The same problem occurs in Nx part. The previous Nx part is 50 which makes the grid have too less nodes and dx too big, if we want to get the accurate numerical part, we need to decrease the Nx as well. So, we set Nx is equal to 25 to make the grids contains 101 nodes and the dx=0.016 to get the decent figure of numerical part. As we setup all the parameters and initial condition, we then create a time loop similar to part b and add the ETM function parts and FTCS function parts after the analytical part.

Then we plot all the figures of analytical and numerical in surface figure and contour figure. Comparing both of them, we find that it looks same and hard to tell the error between them, which can prove that the method we use in both analytical and numerical solution is accurate.

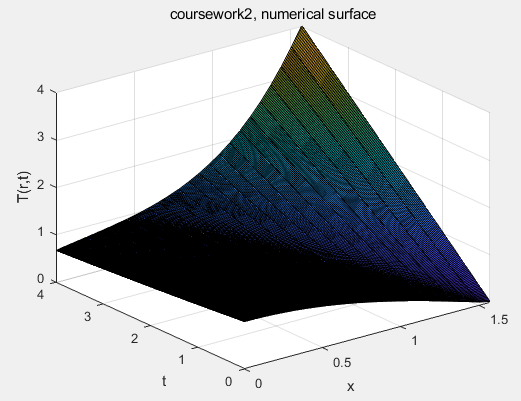
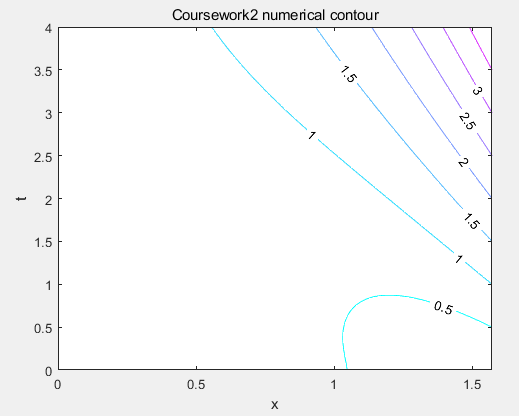
 

Figure 4 Figure 5

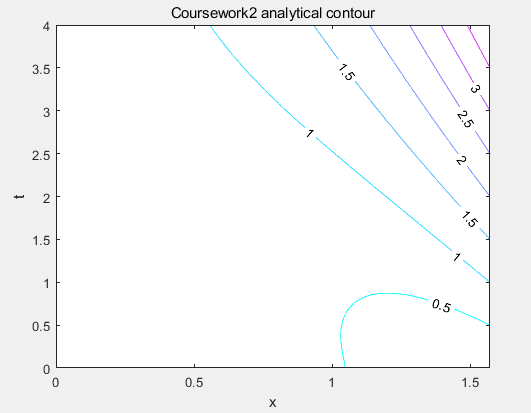
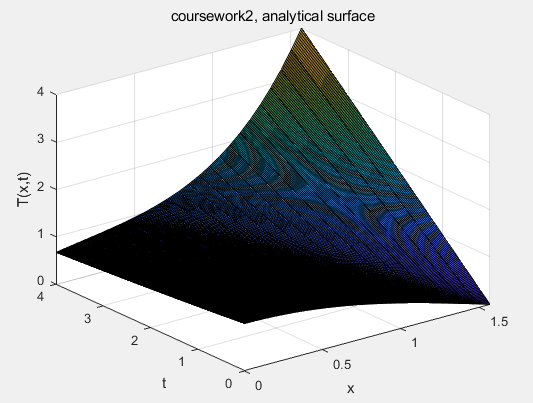


Figure 6 Figure 7

At last, we need to find the difference between the analytical solution and numerical solution because it is difficult to find error from the above figures. Through the surf and contour code, we type the value of analytical solution minus value of numerical solution in z-axis. In surface difference of figure 8, we can tell the error range is roughly 0.005 to -0.02. In contour figure 9, we can tell the maximum error is 0.018.

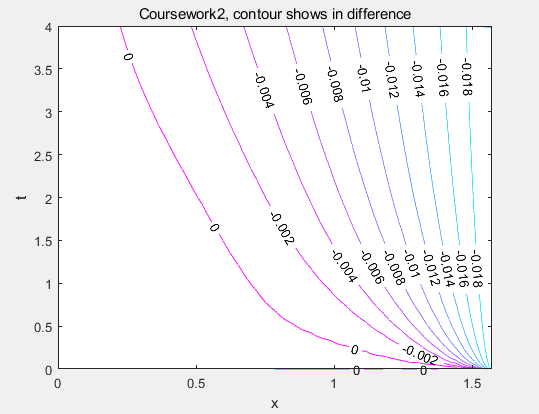
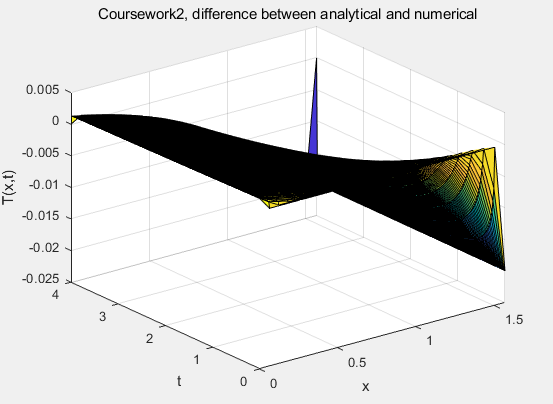


Figure 8 Figure 9