Applied numerical methods, Assignment 3

Group 13
maaxen@kth.se
hfag@kth.se

November 8, 2022

Problem description

In problem 1 we are presented with a metallic rod with length L [m], (here the brackets [ ] denote which unit it is) which has the initial temperature temperature $T = 0$ [C]. At time $t = 0$, a pulse of heat comes with peak temperature $T = T_0$ and the duration of said pulse is t_p, [s] hits the left end, i.e $y = 0$. Moreover the rod at the right end, $y = L$ is insulated, the following PDE is given for the problem

$$\rho C_p \frac{\partial T}{\partial t} = k \frac{\partial^2 T}{\partial y^2}, \quad t > 0, \quad 0 < y < L. \tag{1}$$

With BC's,

$$T(0,t) = \begin{cases} T_0 \sin(\frac{\pi t}{t_p}), & 0 \leq t \leq t_p \\ 0, & t > t_p \end{cases} \qquad \frac{\partial T}{\partial y}(L,t) = 0 \tag{2}$$

and the initial condition $T(y,0) = 0$. Lastly we know the following units of the various variables, ρ has the unit $[kg/m^3]$, C_p has $[J/kg{\cdot}C]$ and k has $[J/m{\cdot}s{\cdot}C]$.

Problem 1

In the first problem we rescale it with the following variables,

$$T = T_0 u, \quad y = Lx, \quad t = t_p \tau.$$

We wish to reformulate the equation. Note

$$T = T_0 u, \quad \frac{\partial T}{\partial t} = T_0 \frac{\partial u}{\partial t}$$

but we wish to have the derivative RHS in the form, $\frac{\partial u}{\partial \tau}$, since $\tau = t_p t$, or more intuitively, $t = \tau / t_p$ we note that we must compensate with a linear factor of t_p and so we arrive at,

$$\frac{\partial u}{\partial t} = \frac{\partial u}{\partial \tau} \frac{1}{t_p}.$$

Similar argument can be made for the case, $y = Lx$,

$$\frac{\partial u^2}{\partial y^2} = \frac{\partial u^2}{\partial x^2} \frac{1}{L^2}$$

A more rigorous mathematical explanation can be seen below,

$$u(x,\tau) = \frac{1}{T_0}T\left(Lx, t_p\tau\right) \Rightarrow \frac{\partial^2 u}{\partial x^2} = \frac{1}{T_0}\frac{\partial}{\partial y}\left(\frac{\partial T}{\partial y}\left(Lx, t_p\tau\right)\right) = \frac{1}{T_0}\frac{\partial}{\partial y}\left(L \cdot T_y\left(Lx, t_pT\right)\right)$$
$$= \frac{L}{T_0}\frac{\partial}{\partial y}\left(T_y\left(Lx, t_p\tau\right)\right) = \frac{L}{T_0}T_{yy}\left(Lx, t_pT\right) \cdot L = \frac{L^2}{T_0}\frac{\partial^2 T}{\partial y^2}$$

. Moreover,

$$\frac{\partial u}{\partial t} = \frac{1}{T_0}\left(\frac{\partial T}{\partial \tau}(Lx, t_p\tau)\right) = \frac{t_p}{T_0}\frac{\partial T}{\partial \tau}$$

So rewriting (1) we have that

$$\rho C_p \frac{\partial T}{\partial t} = k\frac{\partial^2 T}{\partial y^2}$$

$$T_0\rho C_p \frac{\partial u}{\partial \tau}\frac{1}{t_p} = kT_0\frac{\partial^2 u}{\partial x^2}\frac{1}{L^2}$$

$$\frac{\partial u}{\partial \tau} = \frac{kt_p}{L^2\rho C_P}\frac{\partial^2 u}{\partial x^2}$$

we let a be,

$$a = \frac{kt_p}{L^2\rho C_p}$$

Note now the units of a, it is

$$\frac{J}{m \cdot s \cdot C} \cdot \frac{m^3}{kg} \cdot \frac{s}{1} \cdot \frac{1}{m^2} \cdot \frac{kg \cdot C}{J}$$

which can be seen to cancel out making a without unit.

Moreover $\tau > 0$ since $t > 0$ and $t = \tau t_p$ making $\frac{t}{t_p} > 0$ seeing as $t_p > 0$ as it represents some delay in time.

Further, seeing as $y = Lx$ and $0 < y < L$ then putting in $x = y/L$ then, $0 < y < L$ becomes $0 < x < \frac{L}{L} = 1$.

Lastly let's rewrite (2), starting with the first expression, seeing as $u = \frac{T}{T_0}$, and $u(x,\tau)$, then using,

$$T(0,t) = T_0 \sin\left(\frac{\pi t}{t_p}\right)$$

we have that u becomes,

$$u(0,\tau) = \sin(\pi\tau), \quad 0 \leq \tau \leq 1$$

seeing as, $t = t_p\tau$, then $0 \leq t \leq t_p$ becomes $0 \leq \tau \leq \frac{t_p}{t_p} = 1$.

The 0 case is readily seen, and as $t > t_p$ then $t_p\tau > t_p$ is the same as $\tau > 1$.

Lastly $\frac{\partial T}{\partial y}(L,t) = 0$, becomes $\frac{\partial u}{\partial x}(1,\tau) = 0$ by the transformation, since, $0 < y < L$ and $0 < x < 1$. Similarly, $T(0,y) = 0$ becomes $u(x,0) = 0$.

It is mentioned in the task that a wished value of a is one fourth and to better gain a estimate of various values we should consider the rod as made of copper. From [1] and [2] we find the following values $\rho = 8963kg/m^3$, $k = 401J/m \cdot s \cdot C$ and $C_p = 385J/kg \cdot C$.

Note again $a = \frac{kt_p}{L^2\rho C_p}$, by putting in our values we get roughly

$$a = \frac{t_p}{L^2} 0.0001$$

and with $a = 1/4$, we get the relation

$$L^2 \approx 0.00046 \cdot t_p.$$

We note here that having a larger rod takes the pulse longer time for this value of a which seems plausible as it has to heat more and transfer longer.

Say for a given rod length, $L = \frac{1}{2}$ then we get a rough estimate of $t_p = 543$ s.

Problem 2

The equations from problem 1 after rescaling are summarized here as,

$$\frac{\partial u}{\partial z} = a\frac{\partial^2 u}{\partial x^2}, \quad \tau > 0, \quad 0 < x < 1, \quad a = \frac{ktr}{\rho C_p L^2} = \frac{1}{4} \tag{3}$$

$$u(0,\tau) = \alpha(\tau) = \begin{cases} \sin(\pi\tau), & 0 \le \tau \le 1, \\ 0, & \tau > 1, \end{cases} \qquad \frac{\partial u}{\partial x}(1,\tau) = 0 \tag{4}$$
$$u(x,0) = 0.$$

The rescaled equation (3) are discretized with method of lines (MoL) with,

$$x_j = j\Delta x, \quad \Delta x = \frac{b-a}{n+1} = \frac{1-0}{n+1} = \frac{1}{n+1}, \quad j = 0,\ldots,n+1$$

and u_{xx} are then approximated with central difference as followed,

$$u_{xx}(x_j,\tau) = \frac{u_{j-1}(\tau) - 2u_j(\tau) + u_{j-1}(\tau)}{\Delta x^2} \tag{5}$$

We can therefore obtain the following equations,

$$\begin{aligned} &\tfrac{\partial u_1(\tau)}{\partial \tau} = \tfrac{u_2(\tau)-2u_1(\tau)+\alpha(\tau)}{\Delta x^2}\cdot a \\ &\tfrac{\partial u_j(\tau}{\partial \tau} = \tfrac{u_{j+1}(\tau)-2u_j(\tau)+u_{j-1}(\tau)}{\Delta x^2}\cdot a, \quad j = 2,...,n-1 \\ &\tfrac{\partial u_n(\tau)}{\partial \tau} = \tfrac{\beta(\tau)-2u_n(\tau)+u_{n-1}(\tau)}{\Delta x^2}\cdot a \\ &u_j(0) = 0. \end{aligned}$$

We obtain an ODE-system on the form from the previous equation,

$$\frac{d\mathbf{u}}{d\tau} = A\mathbf{u} + \mathbf{b}(\tau), \quad \mathbf{u}(0) = \mathbf{g} \tag{6}$$

where,

$$A = \frac{a}{\Delta x^2}\begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 2 & -2 \end{pmatrix}, \quad \mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \\ u_{n+1} \end{pmatrix},$$

$$\mathbf{b}(\tau) = \begin{pmatrix} f(x_1,t) + \frac{\alpha(t)}{\Delta x^2} \\ f(x_2,t) \\ \vdots \\ f(x_n,t) \\ f(x_{n+1},t) + \frac{\beta(t)}{\Delta x^2} \end{pmatrix} = \begin{pmatrix} \frac{\alpha(\tau)}{\Delta x^2} \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{g} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}$$

The boundary condition at the end of the rod taken from equation (4) is given by,

$$\frac{\partial u}{\partial x}(1,\tau) = 0, \tag{7}$$

and is taken into consideration using ghost point method. The last point, n+1, is approximated with central difference where $O(\Delta x^2)$ is ignored seeing as the central difference scheme has a second order accuracy. Thus, obtaining,

$$\frac{u_{n+2} - u_n}{2\Delta x} = 0, \Rightarrow u_{n+2} = u_n \tag{8}$$

This is the reason for the 2 in the last row of A, similar to the previous assignment. In this problem we are using a rod of length 1.

a.

The system, equation (6), can now be solved using Explicit Euler method with $\tau \in (0,\ 2)$ using a constant timestep Δt and start values $\mathbf{g}$. The equation used for this is given by,

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \Delta t\left(A\boldsymbol{u}^{n+1} + \boldsymbol{b}(\tau_{n+1})\right) \tag{9}$$

Solving this for all τ with $\Delta x = \frac{1}{n+1}$, $n = 10$, and $\Delta t = \frac{1}{10}\Delta x$ gives us the following plots (reasoning for picking $\Delta t = \frac{1}{10}\Delta x$ is just to satisfy the stability condition described later),

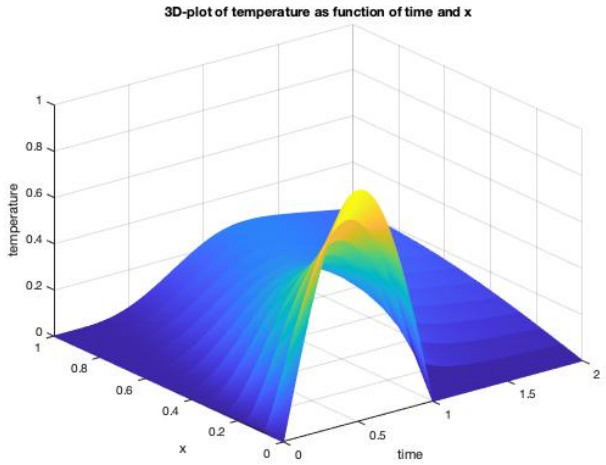

Figure 1: Stable Solution

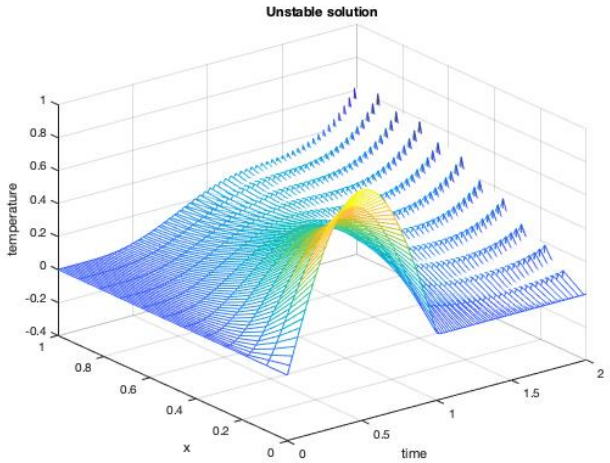

Figure 2: Unstable solution

The values used for the two cases are as followed:

	Stable	*Unstable*
Δx	0.0909	0.0909
Δt	$8.2645 \cdot 10^{-4}$	0.0248
$\frac{\Delta t}{\Delta x^2}$	0.1	3

Regarding the stability condition that need be fulfilled for us to have a stable solution we refer back to the lecture notes and see the following, for the central difference scheme,

$$A = \frac{1}{4\Delta x^2}\begin{pmatrix} -2 & 1 & & \\ 1 & -2 & 1 & \\ & \ddots & \ddots & \ddots \end{pmatrix}, \quad \lambda_k = -\frac{1}{\Delta x^2}\sin^2\left(\frac{k\pi}{2(n+1)}\right).$$

Note moreover that we have a factor of 4 here coming from the $a = \frac{1}{4}$. Thus the eigenvalues lie in the interval,

$$\lambda_k \in \left(\frac{-1}{\Delta x^2}, \quad 0\right).$$

Lastly for our explicit Euler with central difference we have that,

$$-2 < \frac{-\Delta t}{\Delta x^2} < 0 \Rightarrow \frac{\Delta t}{\Delta x^2} < 2.$$

The stable solution of the temperature distribution over the whole rod is plotted for a fixed $\tau = 1$, i.e. $u(x, 1)$,

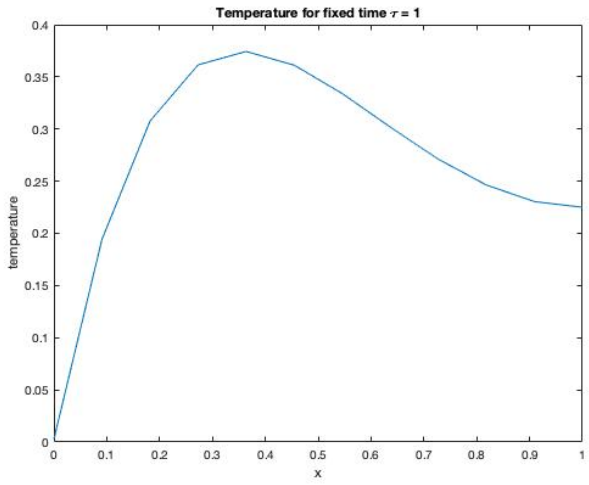

Figure 3: Fixed time $\tau = 1$

Lastly, the stable solution for the temperature distribution at the endpoints of the rod as a function of time, i.e. $u(0, \tau)$ and $u(1, \tau)$, is demonstrated with

the following plot,

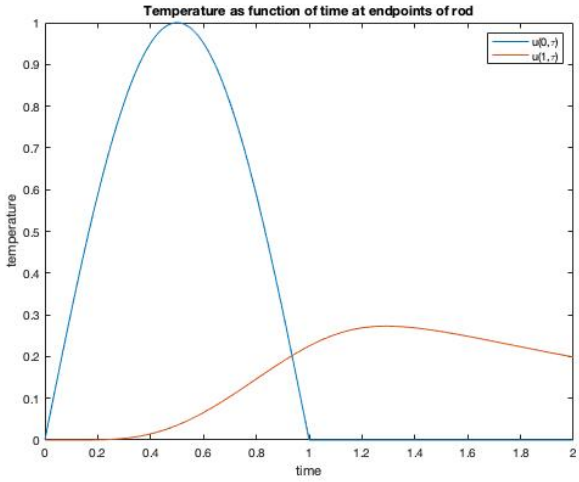

Figure 4: Temperature over time at endpoints x = 0 and x = 1

b.

The system from the equation (6) is here solved using MATLAB's built-in ODE-solvers: ODE23 (explicit method), ODE23s (implicit method), and ODE23sJ (implicit method) with the option Jacobian = A. The three methods are used for solving the same problem under similar conditions. The methods are compared with regard to number of timesteps for the solution to reach $\tau = 2$ and the methods computational time. The number of timesteps are collected from the length of the time vector that the built-in function produces and the computational time can be measured with MATLAB's **tic/toc**. The methods are compared for three different values of n: n = 100, n = 200, n = 400, and is presented in the following table,

	# of time steps			*computational time*		
n	$ODE23$	$ODE23s$	$ODE23sJ$	$ODE23$	$ODE23s$	$ODE23sJ$
100	8118	138	140	0.273	0.457	0.043
200	32154	153	150	0.966	0.749	0.061
400	127985	153	159	4.508	1.937	0.096

The main difference between explicit and implicit methods is that explicit methods calculate the state of a system at a later time by using the present state of the system, while calculating the state with implicit methods involve both the present and the later state. Implicit methods are by this reason more stable for stiff problems and can therefore use a larger timestep to calculate the states of the system compared to explicit methods. This is the reason for the differences in number of timesteps between ODE23 and ODE23s. The number of timesteps will be approximately the same for ODE23s as ODE23sJ, seeing as the only difference between the two methods is that ODE23sJ already has the Jacobian pre-computed.

As for the relation between Δx and Δt is going to be $\Delta x \sim \Delta t^2$, thus increasing a factor of 2 in Δx means that, $2\Delta x \sim (2\Delta t)^2 = 4\Delta x^2$ and so the timesteps will increase by a factor of 4.

An implicit method will require more complex computations for each state compared to an explicit method. This is the reason why ODE23s takes a longer time to compute a solution than ODE23 for a small n (n=100), i.e. when the number of timesteps for ODE23 is relatively small. When n grows (n=200, n=400), the number of timesteps for ODE23 increases with a factor of 4, imply-

ing that the number of computations needed also increases approximately with a factor 4. ODE23s needs to compute roughly the same amount of computations, causing the time to increase more linear with n. This is reason why the total time to compute a solution for large n with ODE23 is much longer than ODE23s.

The input of our Jacobian specifies which Jacobian is used in the method, whilst without the specification the ODE method must for each timestep compute it. Thus specifying the Jacobian makes it go vastly faster.

The matrix A is the correct input for the equation since we have rewritten the equation to a first degree differential problem (with explicit Euler), that is in the form of,

$$\frac{d\mathbf{u}}{d\tau} = A\mathbf{u} + \mathbf{b}(\tau)$$

then the Jacobian to solve said equation is obviously the matrix A which is our input.

So Finally, we will conclude that it is beneficial to take the implicit method with the Jacobian pre-calculated, seeing as we avoid the problem of stiffness (and take fewer time steps) and additionally compute it faster seeing as the Jacobian is already calculated.

Problem 3

Problem description

In this section we consider the previous 2D problem in the assignment before, but solve it here with Crank-Nicolson in time. To give a brief description, we have a metal block with dimensions, $\Omega = [0 < x < 5, 0 < y < 2]$ we let $u(x, y, \tau)$ denote temperature at x, y at a time τ, the temperature is kept constant at the edges, namely $u = 40$ at $x = 0$ and $u = 400$ at $x = 5$. At $\tau = 0$ we start to heat it by the external heat source f. The following equations are given,

$$\frac{\partial f}{\partial \tau} = \Delta u + f, \quad f(x, y) = 6000 \exp(-5(x-1)^2 - 10(y-1.5)^2) \tag{10}$$

for the $x, y \in \Omega$ and $\tau > 0$ we have BC's as,

$$u(0, y) = 40, \quad u(5, y) = 400 \quad \forall y \in (0, 2), \quad \frac{\partial u}{\partial y}(x, 0) = \frac{\partial u}{\partial y}(x, 2) = 0 \quad \forall x \in (0, 5).$$

Lastly the initial data is given by,

$$u(x, y, 0) = u_0(x, y) = 40 + 72x.$$

Solution

We first solve the PDE in (10) via the Crank-Nicolson method. Firstly we discretize the grid of our 2D rectangle, as in the previous assignment with $N = 49$ and $M = 21$ where N denotes the amount of grid divisions in x, and M the divisions in y as in the picture seen below.

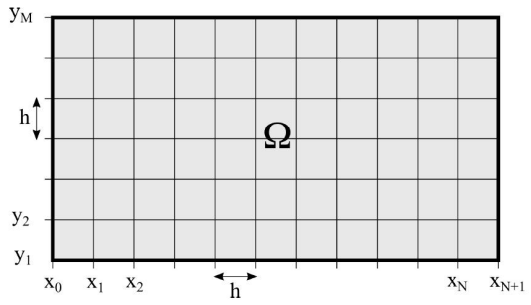

Figure 5: Grid discretisation

The selection of $N = 49$ and $M = 21$ is the same as before, the logic behind this is that, the bound of the 2D array is $x = 5$ and $y = 2$ making the N and M proportional to just these. Similar to before we construct the block matrix as following, with the central difference scheme.

$$A = \frac{1}{\Delta x^2}\begin{pmatrix} T & -I & & \\ -I & T & -I & \\ & \ddots & \ddots & \ddots \\ & & -I & T \end{pmatrix} \in \mathbb{R}^{MN\times MN}, \quad T = \begin{pmatrix} 4 & -1 & & \\ -1 & 4 & -1 & \\ & \ddots & \ddots & \ddots \\ & & -1 & 4 \end{pmatrix} \in \mathbb{R}^{N\times N}$$

Where I denotes the identity matrix. Note further by the usage of the boundary condition,

$$\frac{\partial u}{\partial y}(x,0) = \frac{\partial u}{\partial y}(x,2) = 0 \quad \forall x \in (0,5).$$

We get for the central difference scheme that on the boundary,

$$\frac{u_{k,l+1} - u_{k,l-1}}{2\Delta x} = 0 \Rightarrow u_{k,l+1} = u_{k,l-1}$$

same for the other side of the 2D grid,

$$\frac{u_{k,1} - u_{k,-1}}{2\Delta x} = 0 \Rightarrow u_{k,1} = u_{k,-1}$$

So we have to adjust the A matrix as in the previous assignment to,

$$A = \frac{1}{\Delta x^2}\begin{pmatrix} T & -2I & & \\ -I & T & -I & \\ & \ddots & \ddots & \ddots \\ & & -2I & T \end{pmatrix} \in \mathbb{R}^{MN\times MN}$$

To then use our Crank-Nicolson, that is given by,

$$\left(I - \frac{1}{2}\Delta t A\right) u^{n+1} = \left(I + \frac{1}{2}\Delta t A\right) u^n + \frac{1}{2}\Delta t \left(b\left(t_n\right) + b\left(t_{n+1}\right)\right) \tag{11}$$

where we have clearly gotten an expression for our A matrix, we only need to form the $b(t_n), b(t_{n+1})$. It is formed via the boundary conditions and the function, f, (10), note further since these functions are not based on time but only the grid points we can disregard the time aspect.

We create the vector $b(t_n)$ in the following manner, let $L_x = 5$ and $L_y = 2$, and further let,

$$b(t_n) = \begin{pmatrix} f_{11} \\ \vdots \\ f_{N1} \\ \hline f_{22} \\ \vdots \\ f_{N2} \\ \hline \vdots \\ \hline f_{1N} \\ \vdots \\ f_{NN} \end{pmatrix} + \frac{1}{\Delta x^2} \begin{pmatrix} g(x_1, 0) \\ \vdots \\ g(x_N, 0) \\ \hline 0 \\ \vdots \\ 0 \\ \hline 0 \\ \vdots \\ 0 \\ \hline g(x_1, L_y) \\ \vdots \\ g(x_N, L_y) \end{pmatrix} + \frac{1}{\Delta x^2} \begin{pmatrix} g(0, y_1) \\ 0 \\ \vdots \\ 0 \\ g(L_x, y_1) \\ \hline g(0, y_2) \\ 0 \\ \vdots \\ 0 \\ g(L_x, y_2) \\ \hline \vdots \\ \hline g(0, y_m) \\ 0 \\ \vdots \\ 0 \\ g(L_x, y_m) \end{pmatrix}$$

Here the middle vector denotes the BC's at $y = 0$ and $y = L_y$ and the right most vector denotes the BC's at $x = 0$ and $x = L_x$. Where the various f_{NM} denotes the values of f at $f(x_n, y_m)$ and the boundaries of g, such as, $g(0, y) = 40$ and $g(5, y) = 400$ for the right most vector.

To then solve the system we compute the various matrices of (11) and use the sparse command in MATLAB to to make the computation-speed more quick. The system is solved with a backslash.

Our matrix that has been solved for, contains for each timestep, t, a vector with all the combination of x, y values in our grid. To then plot the solutions for a given t we take out the corresponding column to that t and then reshape it, to then mesh it. Listed below are the various plots for the various t.

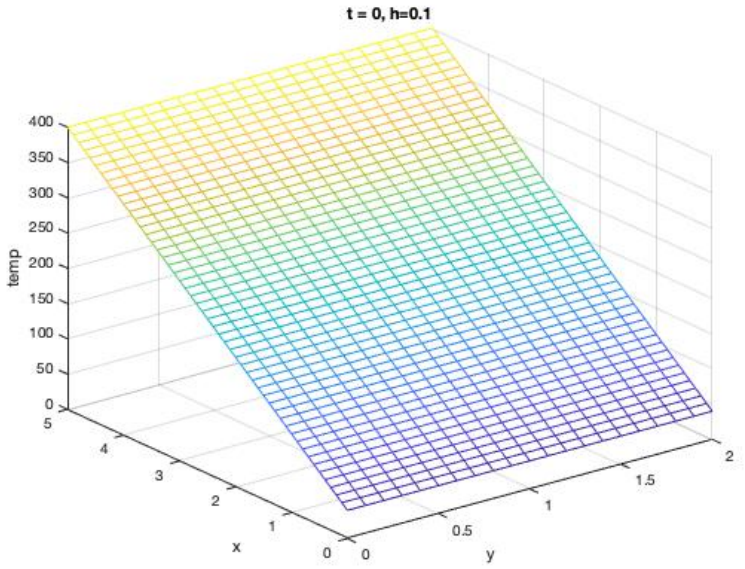

Figure 6: Mesh plot of solution t=0

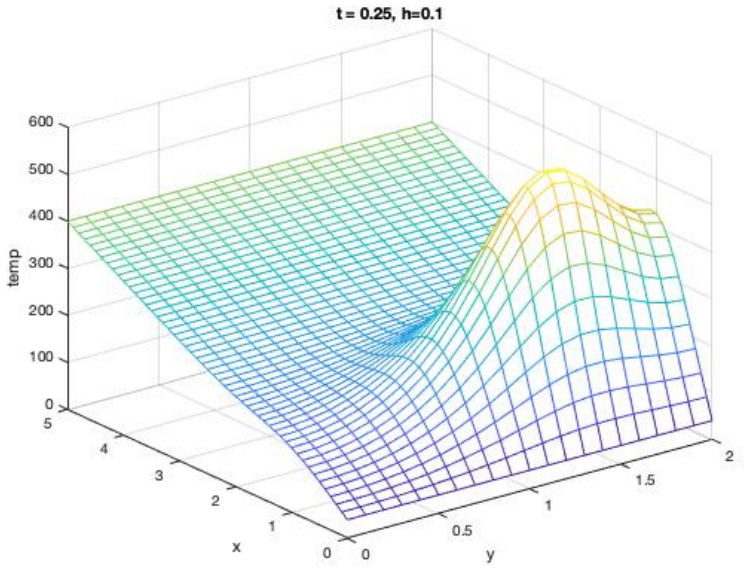

Figure 7: Mesh plot of solution t=0.25

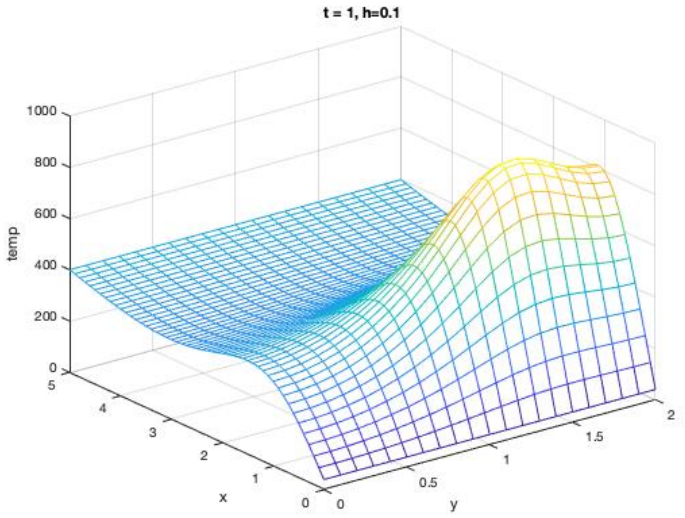

Figure 8: Mesh plot of solution t=1

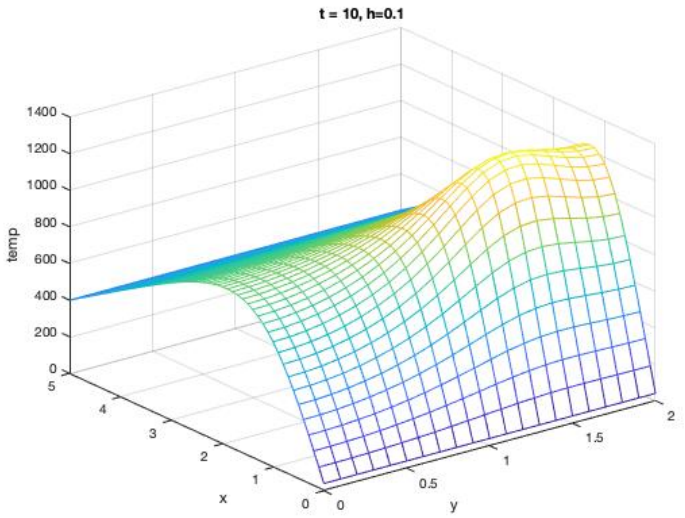

Figure 9: Mesh plot of solution t=10

Lastly let's plot the temperature at $(x, y) = (3, 1)$ as time progresses, this is done via taking out the corresponding row in our solution matrix that contains that specific time.

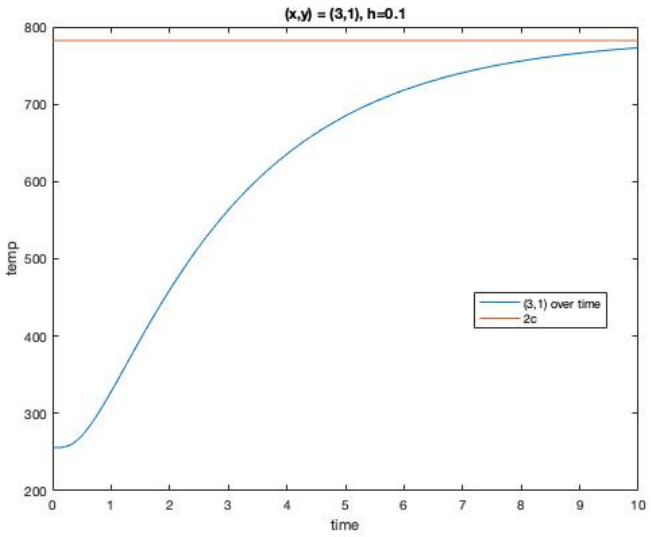

Figure 10: Plot of temperature change with t at $(x, y) = (3, 1)$

We can note that the first plot, $t = 0$, is just a visualisation of the initial condition $u(x, y, 0) = u_0(x, y) = 40 + 72x$. Moreover, we see that the more t increases the more it is starting to replicate our results in 2c in our previous assignment. A more illustrative representation of just this replication is that last plot, here we can see how our solution is approaching the temperature in 2c.

The choice of u_0 here is convenient as it satisfies the BC conditions given in the assignment for the temperature. We shall further note the criteria for explicit ode methods (methods where we have a criteria of),

$$\frac{\Delta T}{\Delta x^2} < C$$

seeing as we are here using our implicit method we can easier avoid this problem for the stiff equation. Further note too that we are aware that the central difference scheme has second order accuracy, as this is something which we wish to maintain our solution of the discretized equation should also be of the same order.

References

[1] https://en.wikipedia.org/wiki/Copper, 2021-11-01

[2] https://gchem.cm.utexas.edu/data/section2.php?target=heat-capacities.php , 2021-11-01