Continuous Assessment 1

MSc in Computer and Mathematical Engineering

Exercise 1

Solve the following heat equation inital-boundary-value problem:

$$\boldsymbol{u_t = 2u_{xx}}$$

$$\boldsymbol{u(x,0) = -\sin 3\pi x + \frac{1}{4}\sin 6\pi x}$$

$$\boldsymbol{u(x,t) = u(1,t) = 0}$$

In this exercise we are asked to solve the above-mentioned initial-boundary-value problem numerically. For this purpose, we explain the approach we have taken to design the numerical solution algorithm and comment the results obtained. The algorithm or code used for the resolution of this exercise is available in Ex_1.m, a Matlab script contained in the folder. Even though we understood the example code of the teacher's example, we provided our own version which is easier for the author to understand.

One can see the problem uses the one-dimensional diffusion or heat equation, and because of the conditions being $u(x,t) = u(1,t) = 0$, then we can clearly see that the problem uses Dirichlet boundary conditions. The initial condition is the expression for $u(x,0)$, which can be interpreted as the initial heat distribution. Moreover, the heat equation is a parabolic PDE.

If we map this problem to that canonical problem of heating a rod, we can see that the length of the rod would be $L = 1$ because of the boundary condition $\boldsymbol{u(1,t) = 0}$, while we can choose some final time T to create the mesh of points. For visualization purposes, we choose $T = 0.1$ to use a relatively low number of points and be able to visualize the colour of the different surface levels.

Given the settings in this problem, one can substitute the derivatives for the finite differences and use an explicit finite difference method for the heat equation. This results in the following equation:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = 2\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{(\Delta x)^2} \Rightarrow u_j^{n+1} = u_j^n + 2\frac{\Delta t}{(\Delta x)^2}\left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right)$$

$$for\ each\ j = 0,1,2,\ldots\ and\ n = 0,1,2,\ldots$$

This method is easy to implement and requires fewer computations that other implicit methods normally used for initial-boundary-value problems involving the heat equation.

The only condition needed for this method is to comply with the stability condition, which requires

$$\rho = k\frac{\Delta t}{(\Delta x)^2} \leq \frac{1}{2}$$

Hence, we can choose a spacing Δx and Δt such that

$$k\frac{\Delta t}{(\Delta x)^2} \leq \frac{1}{2} \Rightarrow \Delta t \leq \frac{(\Delta x)^2}{2k}$$

By choosing $\Delta x = 0.05$, we obtain $\Delta t = 0.006$, $N_x = 20$ and $N_t = 160$ if we make the equality hold. Therefore, there is no need to check stability as it is guaranteed by enforcing it in this way.

The solution surface $u(x,t)$ is plotted on Figure 1. As we can observe, the initial sinusoidal distribution from the initial condition is right at the beginning (at $t = 0$) and we can see how temperature rapidly converges to 0 when time goes by, which is a characteristic of parabolic PDEs like this one (the initial distribution converges fast). Through this 3D plot we can follow the evolution of temperature along time and space, which is better for understanding than 2D contour plots, but we can also have a sense of contours by looking at the horizontal and vertical lines shown.

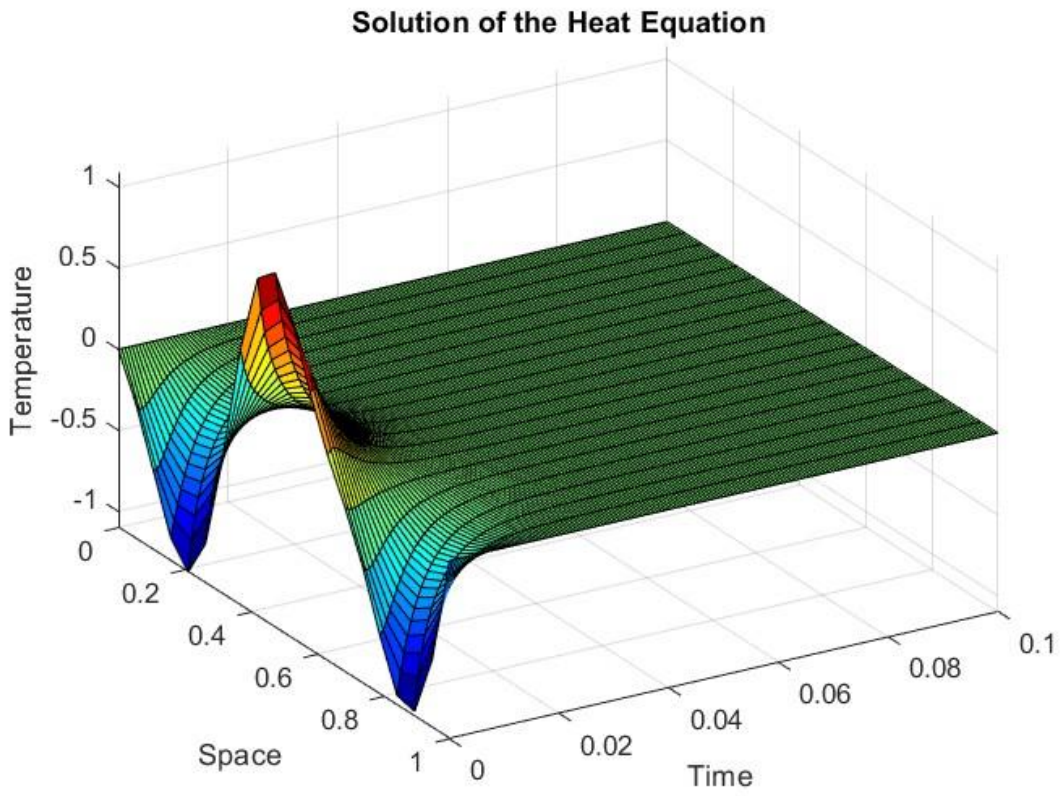

Figure 1. *3D plot for the numerically estimated $u(x,t)$ surface*

Additionally, because we know the analytical solution of this problem can be expressed through the first mode approximation

$$u(x,t) = -\sin(3\pi x)\, e^{-2(3\pi)^2 t} + \frac{1}{4}\sin(6\pi x)\, e^{-2(6\pi)^2 t}$$

given the initial and boundary conditions of this problem, we can provide estimations for the error for the different points in time t and for each iteration. In this case, we just show two tables for the 40th and the 80th iterations in Figures 2 and 3.

Iteration 40 (t = 0.0242)

Point	Estimated U(x,t)	Analytical U(x,t)	Error
1	0.00000	0.00000	0.00000
2	-0.00329	-0.00644	0.00315
3	-0.00578	-0.01132	0.00554
4	-0.00688	-0.01348	0.00660
5	-0.00632	-0.01238	0.00606
6	-0.00424	-0.00831	0.00407
7	-0.00114	-0.00223	0.00109
8	0.00224	0.00439	0.00215
9	0.00508	0.00995	0.00487
10	0.00669	0.01311	0.00642
11	0.00669	0.01311	0.00642
12	0.00508	0.00995	0.00487
13	0.00224	0.00439	0.00215
14	-0.00114	-0.00223	0.00109
15	-0.00424	-0.00831	0.00407
16	-0.00632	-0.01238	0.00606
17	-0.00688	-0.01348	0.00660
18	-0.00578	-0.01132	0.00554
19	-0.00329	-0.00644	0.00315
20	0.00000	-0.00000	0.00000

Figure 2. *Table of errors for the 40th iteration.*

Iteration 80 (t = 0.0491)

Point	Estimated U(x,t)	Analytical U(x,t)	Error
1	0.00000	0.00000	0.00000
2	-0.00002	-0.00008	0.00006
3	-0.00004	-0.00014	0.00010
4	-0.00004	-0.00016	0.00012
5	-0.00004	-0.00015	0.00011
6	-0.00003	-0.00010	0.00007
7	-0.00001	-0.00003	0.00002
8	0.00001	0.00005	0.00004
9	0.00003	0.00012	0.00009
10	0.00004	0.00016	0.00012
11	0.00004	0.00016	0.00012
12	0.00003	0.00012	0.00009
13	0.00001	0.00005	0.00004
14	-0.00001	-0.00003	0.00002
15	-0.00003	-0.00010	0.00007
16	-0.00004	-0.00015	0.00011
17	-0.00004	-0.00016	0.00012
18	-0.00004	-0.00014	0.00010
19	-0.00002	-0.00008	0.00006
20	0.00000	-0.00000	0.00000

Figure 3. *Table of errors for the 80th iteration.*

One can see that the error decreases as we advance few iterations, which gives some evidence of the speed of convergence of this method and its good precision. The tables for the last iterations printed have errors approximately null, so that we can state that the method converges and the solution is quite precise.

Exercise 2

Solve the following Laplace boundary value problem:

$$\boldsymbol{u_{xx} + u_{yy} = 0}$$

$$\boldsymbol{u(x, 0) = \sin 3\pi x}$$

$$\boldsymbol{u(x, 1) = \sin \pi x}$$

$$\boldsymbol{u(0, y) = u(1, y) = 0}$$

In this exercise we are asked to solve the above-mentioned boundary-value problem numerically. For this purpose, we explain the approach we have taken to design the numerical solution algorithm and comment the results obtained. The algorithm or code used for the resolution of this exercise is available in Ex_2.m, a Matlab script contained in the folder.

One can see the problem uses the two-dimensional Laplace equation, whose boundary conditions are $u(x, 0) = \sin 3\pi x$ and $u(x, 1) = \sin \pi x$ for x and $u(0, y) = u(1, y) = 0$

for y (we do not have initial conditions here because time has no effect). These conditions define the temperature values along the edges of the rectangular region considered.

If we map this problem to that canonical problem of the rectangular plate, we can see that the dimensions of the plate are 1×1, so that $x, y \in [0,1]$. This will make us consider a similar number of observations for each variable, as then the mesh will be more regular.

Given the settings in this problem, one can substitute the derivatives for the finite differences and use a finite difference method for the Laplace equation. This results in the following equation:

$$\frac{u_{j+1}^i - 2u_j^i + u_{j-1}^i}{(\Delta x)^2} + \frac{u_{j+1}^i - 2u_j^i + u_{j-1}^i}{(\Delta y)^2} = 0$$

$$\Rightarrow\ u_j^i = \frac{1}{2((\Delta x)^{-2} + (\Delta y)^{-2})}\left(\frac{u_j^{i+1} - u_j^{i-1}}{(\Delta x)^2} + \frac{u_{j+1}^i - u_{j-1}^i}{(\Delta y)^2}\right)$$

$$for\ each\ \ i = 0,1,2, \ldots\ \ and\ \ j = 0,1,2, \ldots$$

Given that this is an equilibrium state problem, as the PDE is elliptic, we can use the Gauss-Seidel iterative method to numerically solve the problem. This method is well-suited for elliptic PDE problems and converges faster than other methods such as the Jacobi. For this algorithm, we try with a spacing of $\Delta x = \Delta y = 0.05$, just as with the previous problem (to see whether it is enough for a fast convergence) and we specify a tolerance number of 1×10^{-8} to make sure errors are not big and therefore state convergence. Finally, we consider first 10000 iterations, but we expect a much lower number of necessary iterations.

The solution surface $u(x, y)$ is plotted on Figure 4. As we can observe, the initial sinusoidal distributions from the boundary conditions are placed at the extremes of the "horizontal" edges of the plate, while the 0 conditions are seen in the "vertical" edges, but not the interior points (as expected from an elliptic PDE as this one).

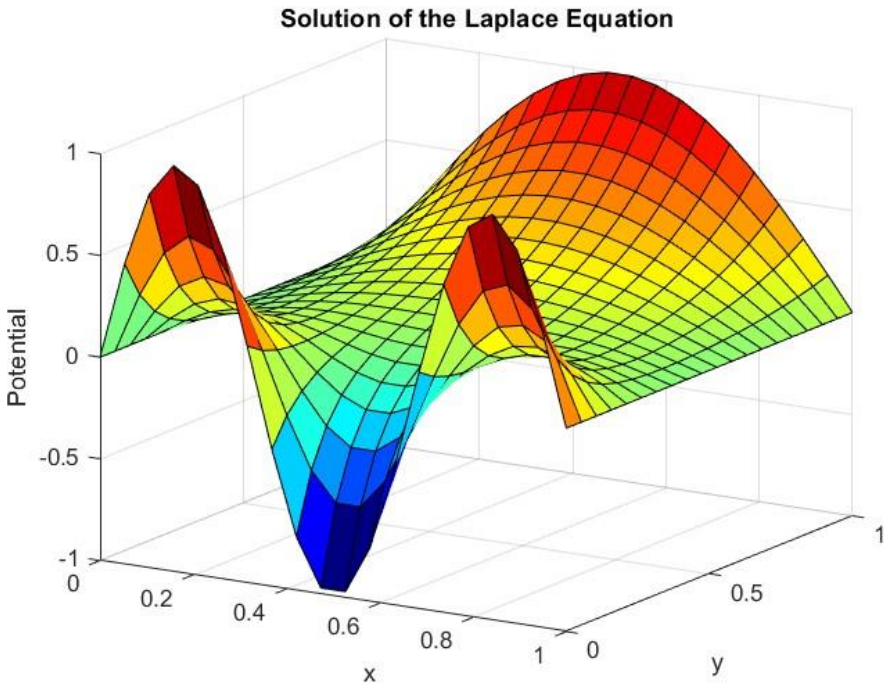

Figure 4. *3D plot for the numerically estimated $\boldsymbol{u(x,y)}$ surface*

Additionally, because we know the analytical solution of this problem can be expressed through the first mode approximation

$$u(x,y) = \sin(3\pi x)\cos(\pi y)$$

given the boundary conditions of this problem, we can provide estimations for the error for the different iterations. To have a notion of the dynamics of our finite differences explicit scheme, we print a table for 20 different equispaced points of the grid, even though we just print the 100th iteration and the 500th for brevity (the algorithm converges at the 554th iteration), shown in Figures 5 and 6.

Iteration 100

Point	Estimated U(x,y)	Analytical U(x,y)	Error
1	0.00000	0.00000	0.00000
24	0.51188	0.79906	0.28718
47	0.38715	0.90451	0.51736
70	0.09950	0.27534	0.17583
94	-0.10061	-0.72084	0.62023
117	-0.03484	-0.63004	0.59520
140	0.05317	-0.09195	0.14512
163	0.09159	0.32102	0.22943
186	0.07597	0.30521	0.22924
209	0.02914	0.07102	0.04188
233	0.03467	-0.07102	0.10569
256	0.11664	-0.30521	0.42185
279	0.21441	-0.32102	0.53543
302	0.32205	0.09195	0.23010
325	0.42640	0.63004	0.20363
348	0.50805	0.72084	0.21279
372	0.49415	-0.27534	0.76949
395	0.42433	-0.90451	1.32884
418	0.26285	-0.79906	1.06191
441	0.00000	-0.00000	0.00000

Max error at iteration 100: 1.53884177

Figure 5. *Table of errors for the 100th iteration.*

Iteration 500

Point	Estimated U(x,y)	Analytical U(x,y)	Error
1	0.00000	0.00000	0.00000
24	0.51356	0.79906	0.28550
47	0.39326	0.90451	0.51125
70	0.11150	0.27534	0.16384
94	-0.08238	-0.72084	0.63846
117	-0.01352	-0.63004	0.61651
140	0.07455	-0.09195	0.16650
163	0.10975	0.32102	0.21127
186	0.08805	0.30521	0.21716
209	0.03333	0.07102	0.03769
233	0.03981	-0.07102	0.11083
256	0.13060	-0.30521	0.43581
279	0.23416	-0.32102	0.55518
302	0.34396	0.09195	0.25201
325	0.44695	0.63004	0.18308
348	0.52456	0.72084	0.19627
372	0.50411	-0.27534	0.77945
395	0.42908	-0.90451	1.33359
418	0.26407	-0.79906	1.06313
441	0.00000	-0.00000	0.00000

Max error at iteration 500: 1.53884177

Figure 6. *Table of errors for the 500th iteration.*

From the figures and the results of the algorithm (which prints this table for each iteration), we can see that the convergence of the solution to a stable solution is fast. This tells us that our approach has rapidly stabilized, but we can see that there would be differences regarding the analytical solution and the precision of our approximation might not be the desired (it is 1.5 ºC off approximately). Because of our algorithm using Gauss-Seidel algorithm, we just stop when the differences between solutions have not changed within some tolerance, but this does not mean the approximation error must be below this tolerance level.

Exercise 3

A rectangular plate of dimensions $L_x = 2$ and $L_y = 1$ is heated initially at a uniform temperature of $T(x, y) = T_0 = 100$ ºC.

a) Compute the evolution of the temperature distribution on the plate when all the borders are kept at a constant temperature of $T = 0$ ºC

In this exercise we are asked to solve the above-mentioned boundary-value problem numerically. For this purpose, we explain the approach we have taken to design the numerical solution algorithm and comment the results obtained. The algorithm or code used for the resolution of this exercise is available in Ex_3a.m, a Matlab script contained in the folder. The code is based on the teacher's example for an explicit method for a 2D parabolic PDE problem.

In this case, we can see that the problem is the canonical heat equation problem for parabolic PDEs, so that the initial-boundary-value problem, given the conditions mentioned, is the following:

$$u_t = k(u_{xx} + u_{yy})$$

$$u(x, y, 0) = 100$$

$$u(x, 0, t) = u(x, 1, t) = 0$$

$$u(0, y, t) = u(2, y, t) = 0$$

We can see that this problem has Dirichlet boundary conditions and an initial condition. Because we are not given any k nor T, a normal assumption is to fix $T = k = 1$. In this case, just as with the first exercise, we can use an explicit finite differences scheme in order to solve this problem. This scheme would be the following, omitting the derivation for the sake of brevity:

$$u_{i,j}^{n+1} = u_{i,j}^{n} + \Delta t \left(\frac{u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}}{(\Delta x)^2} + \frac{u_{i,j-1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}}{(\Delta y)^2} \right)$$

All of the mentioned in the first problem can be generalized for this 2-dimensional problem. In this case, the stability condition is now:

$$\frac{\Delta t[(\Delta x)^2+(\Delta y)^2]}{(\Delta x)^2(\Delta y)^2} \leq \frac{1}{2} \;\Rightarrow\; \Delta t \leq \frac{(\Delta x)^2(\Delta y)^2}{2[(\Delta x)^2+(\Delta y)^2]}$$

At first we chose $\Delta x = \Delta y = 0.01$ and $\Delta t = 0.000001$, so that $N_x = 100$, $N_y = 200$ and $N_t = 1.000.000$, but the final result gave us a high error. Because we had the time to, we also tried $\Delta x = \Delta y = 0.01$ and $\Delta t = 0.0000001$ as it allowed to obtain a lower error. Hence, $N_x = 100$, $N_y = 200$ and $N_t = 10.000.000$, and we obtain more accurate results (closer to the real analytical solution). We, however, noticed that it took a long time, and that users might be willing to give up some precision to obtain faster results.

The solution surfaces $u(x, y, t)$ for 4 different t values are plotted on Figure 7. We need to plot more than one surface because, when we go from 1D to 2D, we would be dealing with 4 different dimensions (u, x, y, t), and hence we cannot graph it on a single plot. Results show what is expected: at $t = 0$, the whole plate has a temperature of 100 ºC, but because of the boundary conditions, heat is flowing from the center to the edges of the rectangular plate, which can be seen as the temperature of the inner nodes or inner points decreases as we advance in time.

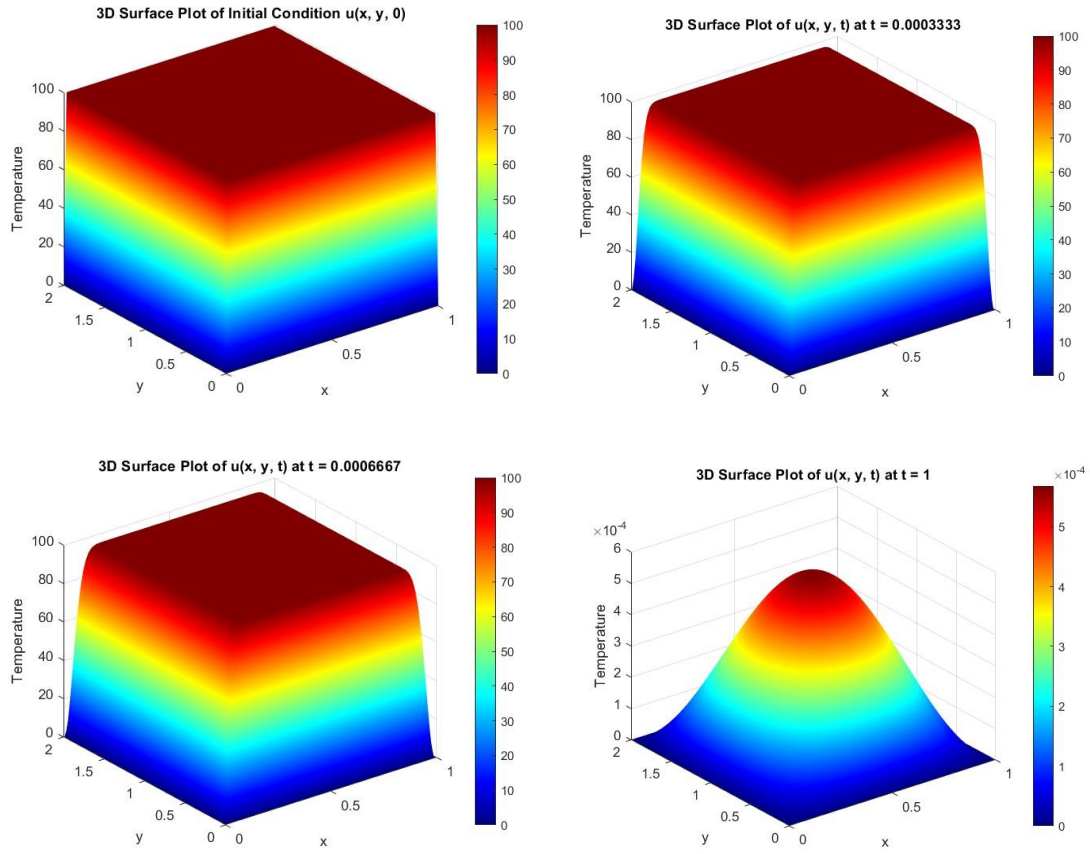

Figure 7. *3D plots for the numerically estimated $u(x, y)$ surface at times 0, 0.000333, 0.000666, and 1.*

When it comes to the comparison between the analytical results and the estimation, we can see that the first mode approximation would be

$$u(x, y, t) = 100 \sin\left(\frac{\pi x}{2}\right) \sin(\pi y)\, e^{-\frac{5}{2}\pi^2 t}$$

given the boundary conditions of this problem. We can therefore evaluate at the results at equispaced different points and get a sense of the approximation errors we have to have a notion of the dynamics of our finite differences explicit scheme. The error table for the 500.000th and the 9.001.000th iterations are shown in Figure 8 and 9.

```
Iteration 500000 (t = 0.0500)
Point  Estimated U(x,y,t)   Analytical U(x,y,t)  Error
-----------------------------------------------------------
1      0.00000              0.00000              0.00000
1069   11.79485             3.88575              7.90909
2138   23.82718             -6.33233             30.15951
3206   8.00243              -3.89290             11.89532
4275   43.48634             17.40235             26.08399
5343   56.30440             -10.32482            66.62922
6412   29.00197             -14.84242            43.84439
7480   57.76893             28.69674             29.07219
8548   74.14976             -5.17407             79.32383
9617   50.12341             -30.03836            80.16177
10685  51.18387             33.01173             18.17214
11754  73.58236             6.67037              66.91199
12822  56.25876             -43.68662            99.94538
13890  29.70554             28.03245             1.67308
14959  54.70796             23.85926             30.84870
16027  41.44726             -50.82811            92.27537
17096  8.31876              16.21508             7.89632
18164  21.56244             39.98074             18.41830
19233  9.41670              -49.37318            58.78987
20301  0.00000              -0.00000             0.00000
Max error at iteration 500000: 109.91775
```

Figure 8. *Table of errors for the 500.000th iteration.*

```
Iteration 9500000 (t = 0.9500)
Point  Estimated U(x,y,t)   Analytical U(x,y,t)  Error
-------------------------------------------------------
1      0.00000              0.00000              0.00000
1069   0.00014              0.00006              0.00008
2138   0.00030              -0.00010             0.00040
3206   0.00007              -0.00006             0.00013
4275   0.00049              0.00026              0.00023
5343   0.00076              -0.00016             0.00091
6412   0.00026              -0.00022             0.00049
7480   0.00060              0.00043              0.00017
8548   0.00103              -0.00008             0.00111
9617   0.00048              -0.00045             0.00094
10685  0.00050              0.00050              0.00000
11754  0.00102              0.00010              0.00092
12822  0.00058              -0.00066             0.00124
13890  0.00027              0.00042              0.00015
14959  0.00074              0.00036              0.00038
16027  0.00046              -0.00077             0.00123
17096  0.00007              0.00024              0.00017
18164  0.00027              0.00060              0.00033
19233  0.00011              -0.00074             0.00086
20301  0.00000              -0.00000             0.00000
Max error at iteration 9500000: 0.00144
```

Figure 9. *Table of errors for the 9.001.000th iteration.*

From the figures and the results of the algorithm, we can see that the convergence is slow but the resulting error becomes small enough to consider the approximation valid, and by executing the code until the very end one could see the maximum error has an order of 10^{-4}. Because the algorithm prints the table every 10.000 iterations, one could see that the algorithm has an almost smooth but slow convergence, reducing the maximum error (from all points) at every iteration most of the time.

b) Repeat the computations, but now assume that the upper border is kept completely insulated, i.e. the gradient of temperature is zero, while the rest of the borders are maintained at a constant temperature of $T = 0$ ºC

In this exercise we are asked to solve the above-mentioned boundary-value problem numerically. For this purpose, we explain the approach we have taken to design the numerical solution algorithm and comment the results obtained. The algorithm or code used for the resolution of this exercise is available in Ex_3b.m, a Matlab script contained in the folder. The code is based on the teacher's example for an explicit method for a 2D parabolic PDE problem.

In this case, we can see that the problem is the canonical heat equation problem for parabolic PDEs, so that the initial-boundary-value problem, given the conditions mentioned, is the following:

$$u_t = k(u_{xx} + u_{yy})$$

$$u(x, y, 0) = 100$$

$$u(x,0,t) = 0 \qquad \frac{\partial}{dx}u(x,1,t) = 0$$

$$u(0,y,t) = u(2,y,t) = 0$$

We can see that this problem has Dirichlet and Von Neumann boundary conditions and an initial condition. Because we are not given any k nor T, a normal assumption is to fix $T = k = 1$. In this case, just as with the first exercise, we can use an explicit finite differences scheme in order to solve this problem. This scheme would be the following, omitting the derivation for the sake of brevity:

$$u_{i,j}^{n+1} = u_{i,j}^{n} + \Delta t\left(\frac{u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}}{(\Delta x)^2} + \frac{u_{i,j-1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}}{(\Delta y)^2}\right)$$

In this case, because we introduce a Von Neumman condition which says that the rate of change of the top edge must be zero, we only have to change the corresponding boundary condition to

$$u(ny,j) = u(ny-1,j) \quad where \quad ny = nº\ of\ total\ y\ points$$

This small difference is in fact very important because it speeds our algorithm through a reflective effect, where the heat reflects back into the domain and does not allow heat to escape, making us reach a steady state faster. All of the mentioned in the first problem can be generalized for this 2-dimensional problem. In this case, the stability condition is still:

$$\frac{\Delta t[(\Delta x)^2 + (\Delta y)^2]}{(\Delta x)^2(\Delta y)^2} \leq \frac{1}{2} \;\Rightarrow\; \Delta t \leq \frac{(\Delta x)^2(\Delta y)^2}{2[(\Delta x)^2 + (\Delta y)^2]}$$

For this problem we tried again $\Delta x = \Delta y = 0.01$ and $\Delta t = 0.0000001$ as it allowed to obtain a lower error. Hence, $N_x = 100$, $N_y = 200$ and $N_t = 10.000.000$, and we obtain more accurate results (closer to the real analytical solution). As mentioned, now time is not a constraint.

The solution surfaces $u(x,y,t)$ for 4 different t values are plotted on Figure 10. We need to plot more than one surface because, when we go from 1D to 2D, we would be dealing with 4 different dimensions (u,x,y,t), and hence we cannot graph it on a single plot. Results show what is expected: at $t = 0$, the whole plate has a temperature of 100 ºC, but because of the boundary conditions, heat is flowing from the center to the edges of the rectangular plate, which can be seen as the temperature of the inner nodes or inner points decreases as we advance in time.

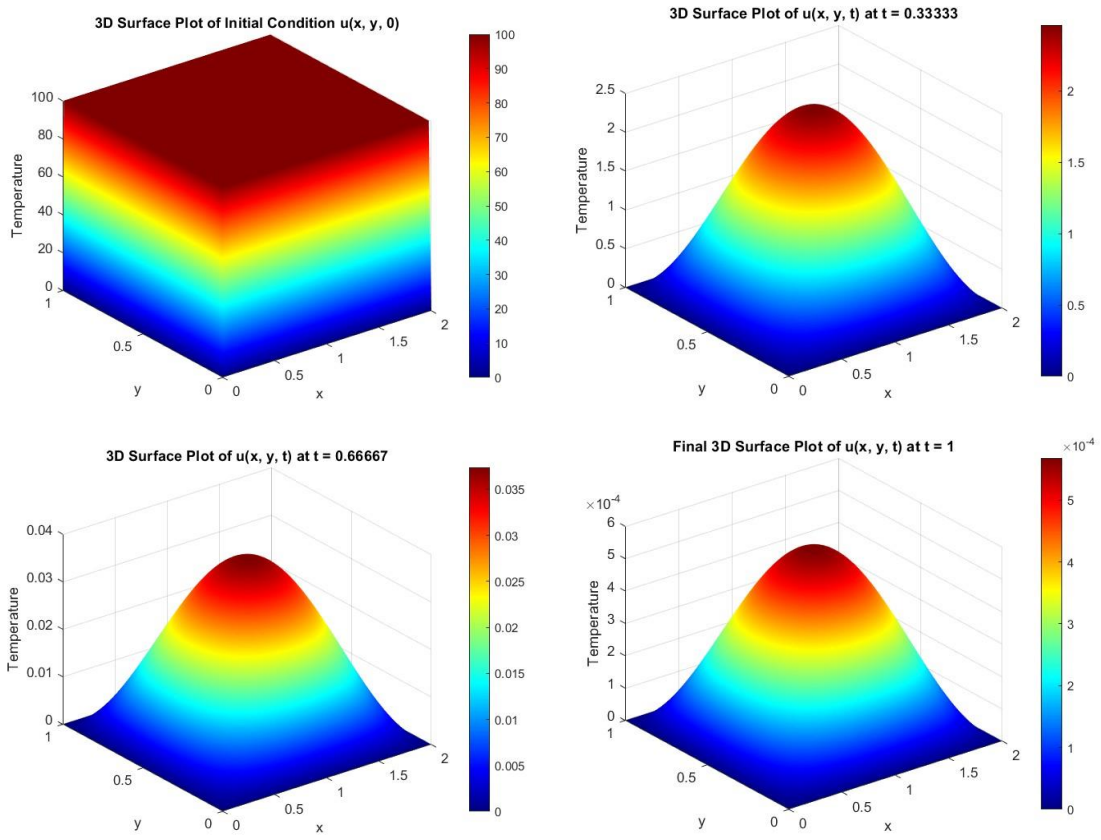

Figure 10. *3D plots for the numerically estimated $\boldsymbol{u(x,y)}$ surface at times 0, 0.333, 0.666, and 1*

When it comes to the comparison between the analytical results and the estimation, we can see that the first mode approximation would be

$$u(x,y,t) = 100 \sin\left(\frac{\pi x}{2}\right) \sin(\pi y)\, e^{-\frac{5}{2}\pi^2 t}$$

given the boundary conditions of this problem. We can therefore evaluate at the results at equispaced different points and get a sense of the approximation errors we have to have a notion of the dynamics of our finite differences explicit scheme. The error table for the 2.500.000th and the 10.000.000th iterations are shown in Figure 11 and 12.

```
Iteration 2503000 (t = 0.2503)
Point  Estimated U(x,y,t)   Analytical U(x,y,t)  Error
------------------------------------------------------------
1      0.00000              0.00000              0.00000
1069   1.06574              0.69086              0.37488
2138   1.11219              0.71151              0.40069
3206   2.35218              1.55528              0.79690
4275   3.66681              2.35953              1.30729
5343   1.44332              1.02709              0.41623
6412   5.85484              3.80336              2.05147
7480   1.01485              0.65460              0.36024
8548   6.15576              4.05306              2.10271
9617   4.29874              2.78595              1.51280
10685  4.11809              2.78595              1.33215
11754  6.21691              4.05306              2.16385
12822  0.80762              0.65460              0.15302
13890  5.78314              3.80336              1.97978
14959  1.57508              1.02709              0.54798
16027  3.52020              2.35953              1.16067
17096  2.35257              1.55528              0.79728
18164  1.00009              0.71151              0.28859
19233  0.96790              0.69086              0.27705
20301  0.00000              0.00000              0.00000
Max error at iteration 2503000: 2.41357
```

Figure 11. *Table of errors for the 2.500.000th iteration.*

```
Iteration 10000000 (t = 1.0000)
Point  Estimated U(x,y,t)   Analytical U(x,y,t)  Error
------------------------------------------------------------
1      0.00000              0.00000              0.00000
1069   0.00009              0.00007              0.00002
2138   0.00009              0.00007              0.00002
3206   0.00019              0.00015              0.00004
4275   0.00030              0.00023              0.00007
5343   0.00012              0.00010              0.00002
6412   0.00048              0.00037              0.00011
7480   0.00008              0.00006              0.00002
8548   0.00050              0.00039              0.00011
9617   0.00035              0.00027              0.00008
10685  0.00034              0.00027              0.00007
11754  0.00051              0.00039              0.00012
12822  0.00007              0.00006              0.00000
13890  0.00047              0.00037              0.00010
14959  0.00013              0.00010              0.00003
16027  0.00029              0.00023              0.00006
17096  0.00019              0.00015              0.00004
18164  0.00008              0.00007              0.00001
19233  0.00008              0.00007              0.00001
20301  0.00000              0.00000              0.00000
Max error at iteration 10000000: 0.00013
```

Figure 12. *Table of errors for the 10.000.000th iteration.*

From the figures and the results of the algorithm, we can see that the convergence is slow but the resulting error becomes small enough to consider the approximation valid, and by executing the code until the very end one could see the maximum error has an order of 10^{-4}. Because the algorithm prints the table every 10.000 iterations, one could

see that the algorithm has an almost smooth but slow convergence, reducing the maximum error (from all points) at every iteration most of the time.