


Universidad del País Vasco

Facultad de Ciencia y Tecnología

NSDE: Practices 5 and 6

Group 2

Nerea Andia

Ander Cano

Mikel Lamela

Aitor Larrinoa

March 2021

Computer practice 5

Problem 1

Statement

(a) Write a program in *Mathematica* that solves (3) with

$$f(x) = (3x + x^2)\,\exp(x) \tag{1}$$

and

$$f(x) = \sin(x)\,\exp(x). \tag{2}$$

for instance for $n = 20$.

Use the *Mathematica* command `Solve` in order to solve the linear system (3).

In a plot compare the obtained solution with the results we got in the first practice where we have been using the `NDSolve` command in order to solve numerically the ODE (3) with the $f(x)$ function given by (1) and (2).

Solution

We are asked to solve the following simple boundary problem:

$$\begin{cases} -u''(x) = f(x), & 0 < x < 1, \\ u(0) = u(1) = 0 \end{cases} \tag{3}$$

where f(x) is going to be defined later. We will solve it by two different methods. On the one hand we will solve it making use of the FD method, and, on the other hand, we will solve the problem making use of the *Mathematica* command NDSolve.

The structure we are going to follow is the next one: We will first solve the problem for each one of the f(x) defined in the statement. Then, we are going to look for the solution with FD. After that, we will solve it with the *Mathematica* command NDSolve, and finally we will compare both results. Then, let's start with the first function.

$\mathbf{f_1(x) = (3x+x^2)}\exp x$

Before starting to solve the problem, it is important to define the grid we are going to use. In order to define it, different variables are necessary, such as the number of interior points or the A matrix defined in theory classes. All this stuff has been defined in the *Mathematica*

notebook. The problem asks us to take 20 interior points, then let's solve this problem for N = 20.

We have printed the tridiagonal A matrix in order to see more or less the structure of the system we have to solve. The matrix is the next one:

Out[ ]//MatrixForm=

$$\begin{pmatrix}
882 & -441 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-441 & 882 & -441 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -441 & 882 & -441 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -441 & 882 & -441 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -441 & 882 & -441 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -441 & 882 & -441 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -441 & 882 & -441 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -441 & 882 & -441 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -441 & 882 & -441 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -441 & 882 & -441 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -441 & 882 & -441 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -441 & 882 & -441 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -441 & 882 & -441 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -441 & 882 & -441 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -441 & 882 & -441 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -441 & 882 & -441 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -441 & 882 & -441 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -441 & 882 & -441 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -441 & 882 & -441 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -441 & 882
\end{pmatrix}$$

Figure 1: A matrix for f_1 and N = 20

We also have to define the vertical vector BB. This vector has as components the values of the function $f_1(x)$ evaluated in the grid points. It is the inhomogeneous term of the problem Au = B. Considering the vector BB we have just mentioned and the matrix A we have just showed, we form a system which is going to give us the solution to the problem. Then, the system to solve is:

$$A \cdot x = BB \tag{4}$$

Where A and BB are the matrix and the vertical vector we have just defined. Making use of *Mathematica*, we have solved the system in two different ways. On the one hand we have solved the system with the *Mathematica* command Solve, and on the other hand, we have solved the system with the *Mathematica* command LinearSolve[A,BB] that solves the system Ax = BB. In both cases the result we have obtained has been equal.

If we plot the solution of the problem with N = 20 interior points, we get the next graphic:

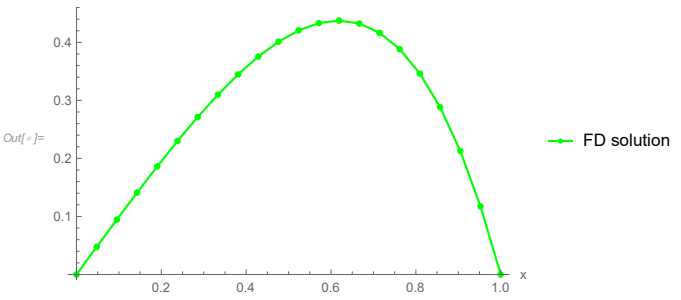

Figure 2: Solution of the problem with $f_1(x)$ and N = 20

We can observe that both boundary conditions are satisfied, in fact, values at u(1) and at u(0) are 0 as the graphic shows.

On the other hand, if we solve the same problem (3) making use of the *Mathematica* NDSolve command, we get almost the same result. In fact, let's plot the result and later we will compare both results.

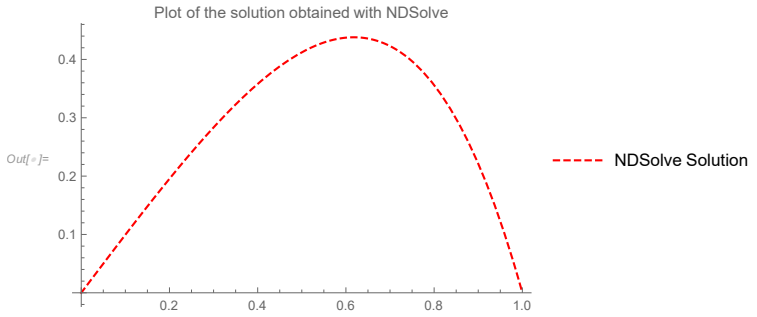

Figure 3: Solution of the problem with NDSolve and $f_1(x)$

If we consider the CPU time in both cases, we can observe that the result obtained using the *Mathematica* command NDSolve is faster, it is just 0.062 seconds whereas the other result takes 0.266 seconds. If we plot both solutions, the result is the next one:

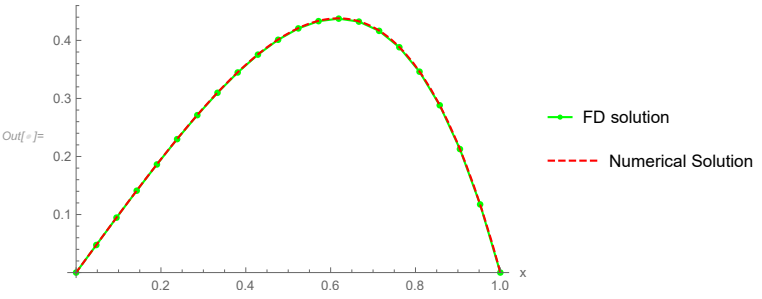

Figure 4: Comparison between NDSolve and FD for $f_1(x)$

We have also analyzed the difference between the errors of each method. The maximum value of the errors we have obtained is 0.000491072, which can be considered a good error. This means that both approximations are very close one to each other and both methods are almost equal.

$\mathbf{f_2(x)} = \sin x \ \exp x$

Our goal now is to get the solution to the problem (3) knowing that $f(x) = f_2(x)$. We will act the same way as we have done for $f_1(x)$ in the previous section.

First, in order to solve the problem (3), we need to prepare all the ingredients to define the grid we are going to use. All these variables are shown in the *Mathematica* notebook. The system to solve is the same as we have seen in (4) where the A matrix is the same we have seen in figure 1, but the BB vertical vector is different. The difference is due to the fact that we have changed the function, and its values are different.

The solution has been calculated with the *Mathematica* commands Solve and LinearSolve and in both cases the result is the same. It is shown in the *Mathematica* notebook.

Let's plot now the solution with N = 20 interior points as the problem tells us.

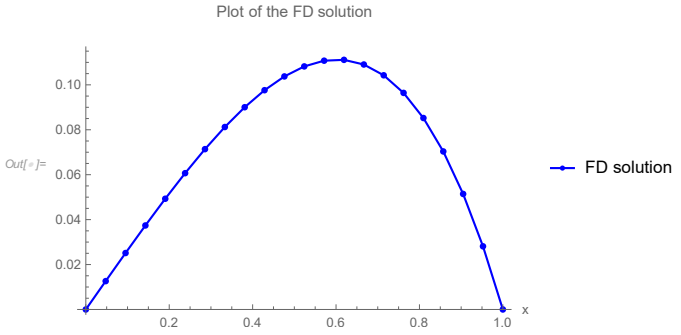

Figure 5: Solution of the problem with $f_2(x)$ and N = 20

We can observe that the boundary conditions are satisfied, in fact we have that u(0) = u(1) = 0 which can be seen graphically.

Now let's solve the problem with the command NDSolve. The result we get is the same if we attend to both plots. The plot of the solution related to NDSolve is the next one:

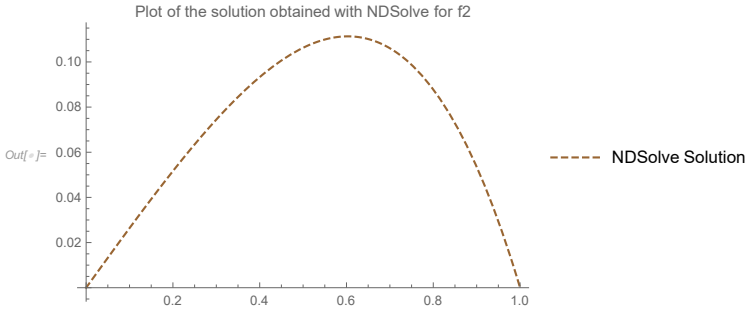

Figure 6: Solution of the problem with NDSolve for $f_2(x)$

The result seems to be the same. In order to be sure that both results are almost equal, we are going to plot both plots together and then we will attend to the maximum of the difference of the errors. Both plots together:

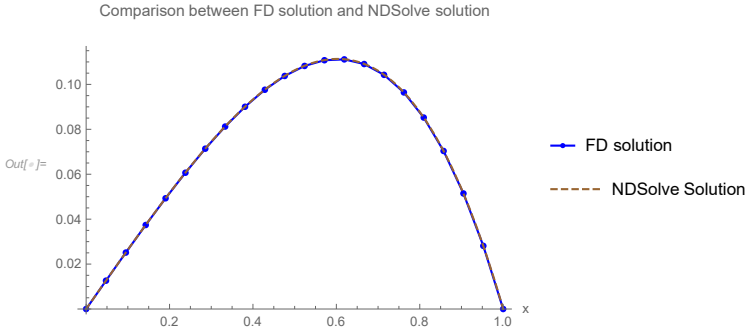

Figure 7: Solution of the problem with NDSolve for $f_2(x)$

Referring to errors, we have printed the maximum of the difference between both errors and the value has been 0.0000668168. Thus, we can conclude that both solutions are almost equal.

Problem 2

Statement

It is possible to generalize the Poisson's equation such that

$$\begin{cases} -u''(x) = f(x), & 0 < x < 1, \\ u(0) = u_0, \qquad u(1) = u_1 \end{cases} \tag{5}$$

(b) Write a program in *Mathematica* that solves (5), for $n = 20$, with $f(x)$ as in (**??**) and with

$$u(0) = 2, \quad u(1) = 1,$$

and with $f(x)$ as in (**??**) and with

$$u(0) = 5, \quad u(1) = -1.$$

In a plot compare these new solutions with the ones obtained in the first practice where we have been using the `NDSolve` command in order to solve numerically the ODE (5) with the $f(x)$ function given by (**??**) and (**??**), and the corresponding BC.

Solution

This problem has boundary conditions different from 0, and we will have to be cautious when analyzing the solution because we have to see if the boundary conditions are satisfied.

We will separate the problem in two different cases because the statement asks us to solve it for two different situations. Let's start then, with the solution.

$\mathbf{u(0) = 2}$, $\mathbf{u(1) = 1}$ and $\mathbf{f_1(x) = (3x + x^2)}\exp x$

We will solve the problem with FD first, and then we will solve the same problem with the *Mathemtaica* command NDSolve, finishing with a comparison of both solutions.

In order to complete the solution referring to FD, we need to define the grid. The grid for this case has been defined the same way as we have defined in problem 1, coinciding in everything except in the boundary conditions and the vertical BB vector. Thus, the A matrix is going to be the same as in figure 1. The system to solve is (4) and as we have said, A matrix is the same as the one in problem 1 and the vertical BB vector is different.

Let's see the plot of the solution in order to see how the solution behaves.

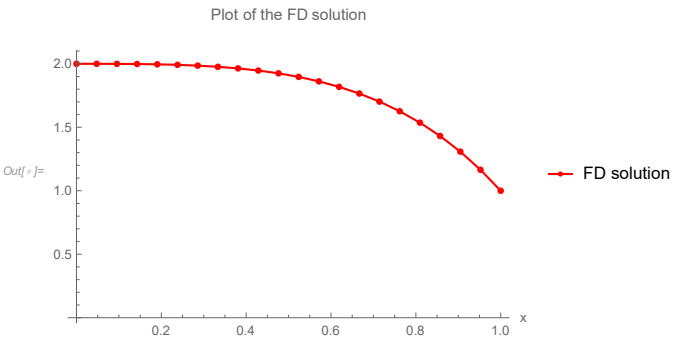

Figure 8: Solution of the problem with $f_1(x)$ and BC different from 0

As we can observe, u(0) = 2, but we can't see in a clear way that u(1) = 1, then, with the aim to guarantee that u(1) = 1 let's make use of *Mathematica*. *Mathematica* tells us the next one:

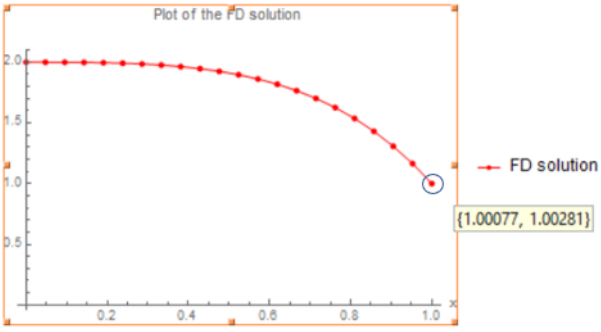

Figure 9: Exact point more or less of the boundary condition

We don't get an exactly accurate point, but we can see more or less that the boundary condition is satisfied.

On the other hand, we are solving the problem making use of the *Mathematica* NDSolve command, and the result has been the next one:

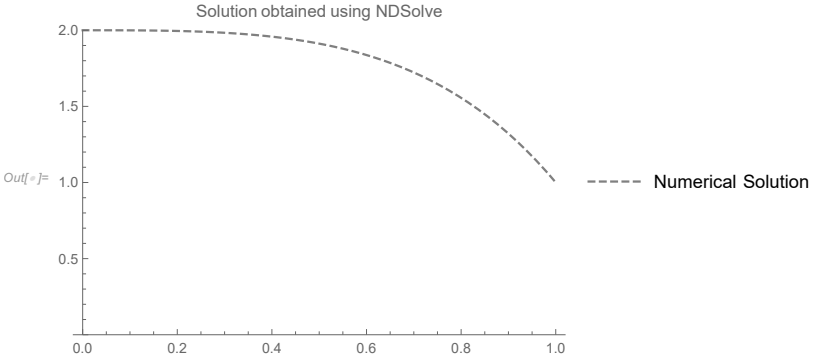

Figure 10: Solution obtained using NDSolve for $f_1(x)$ and BC different from 0

Apparently, both results are nearly the same. However, in order to be sure of it, we will first plot both plots together, and then, we will attend to the difference of the error of both methods. The plot where we have the two plots together is:

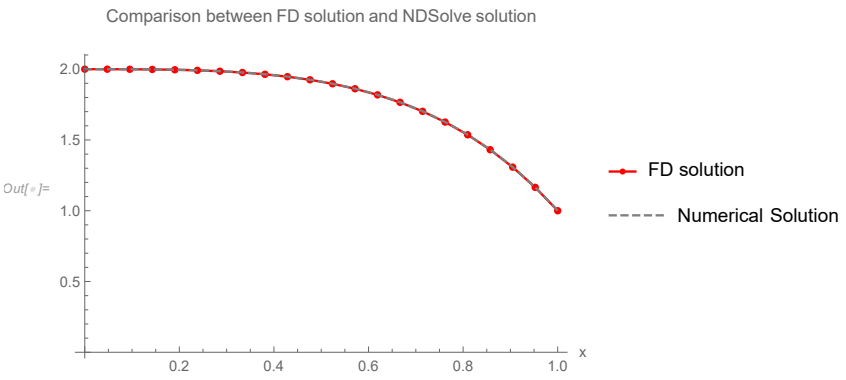

Figure 11: Comparison between both solutions for f1 and BC different from 0

As we can see, both solutions are equal, although it is important to attend at the error, which is 0.000491061. Thus, we have seen that both solutions are equal, although it is important to say that the result obtained with the *Mathematica* command NDSolve is faster attending to the CPU time. In fact, the time required for NDSolve solution is 0.234 whereas the time required for FD solution is 0.047, that confirms our statement.

$\mathbf{u(0) = 5,\ u(1) = -1}$ and $\mathbf{f_2(x)} = \sin x \exp x$

As always, we will start solving the problem with FD. In order to do this, the first thing we have to do is to define the grid. The A matrix is going to be the same as in the previous case and the vertical BB vector is going to be different. Now, let's see the plot of the FD solution:

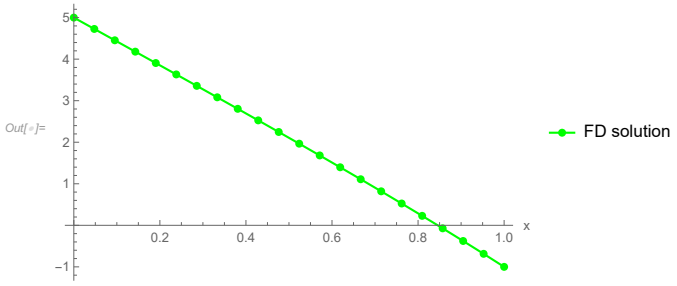

Figure 12: FD solution for $f_2(x)$ and BC different from zero

In this plot we have to see that the BC conditions are satisfied. In fact, we can easily observe that u(0) = 5. However, the boundary condition u(1) = −1 can not be seen clearly and in order to be sure that it is satisfied, we are going to act as we did in figure 9.

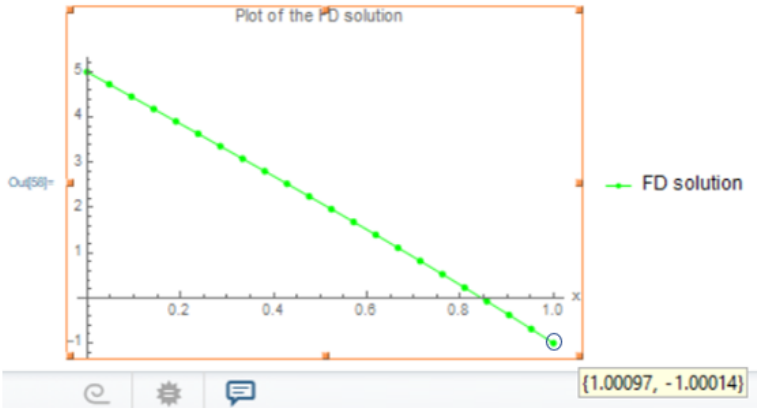

Figure 13: exact point at u(1) = −1

It is not exact but it can be seen more or less that the boundary condition is satisfied, thus u(1) = −1. Therefore, we have seen that the two boundary conditions are satisfied.

On the other hand, from the numerical point of view, we use the NDSolve command to solve the problem and we get the following plot:

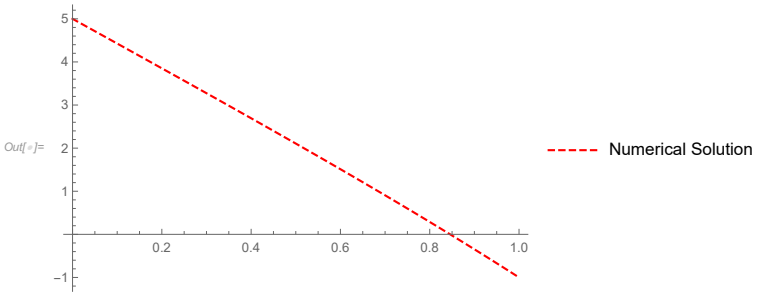

Figure 14: Solution obtained using NDSolve for $f_2(x)$ and BC different from zero

Seeing this graphic we deduce that both the FD solution and the numerical solution are nearly the same. With the purpose of verifying it, let's plot the two solutions together and calculate the error of both methods:

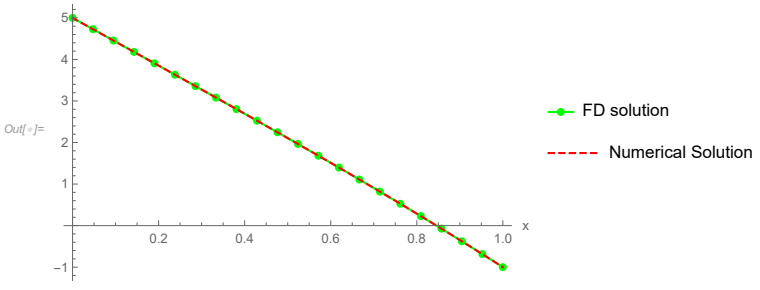

Figure 15: Comparison between both solutions for f_2 and BC different from 0

The maximum of the differences of the errors is 0.0000667681. Thus, it is a very good approximation of the solution. With respect to CPU time, the NDSolve is faster. It only needed 0.047 seconds, whereas the FD command needed 0.186 seconds.

Computer practice 6

Problem 1

Statement

(a) Write a program in *Mathematica* that solves (6) using FD , in the case $\nu = 1/4$, $L = \pi$, with the IC $f(x)$ as in (1)

$$f(x) = Exp[-30(x-1)^2] + \frac{Exp[-20(x-2)^2]}{2} + \frac{Exp[-40(x-2.5)^2]}{4}$$

and BC

$$g_1(t) = \sin^2(2\,\pi\,t), \qquad g_2(t) = t.$$

- Plot the 3D graphic of the solution for $0 \leq x \leq \pi$ and $0 \leq t \leq 2$.....
- Play around with different values of the Courant number ρ (for instance try $\rho = 0.3, 0.4, 0.6$), keeping fixed n, e.g. $n = 50$.

Solution

This time we are solving the next problem.

$$\begin{cases} u_t = \nu\, u_{xx}, & \nu > 0, \quad 0 < x < L, \quad t > 0 \\ u(t,0) = u(t,L) = 0, & t > 0 \\ u(0,x) = u_0(x), & 0 < x < L. \end{cases} \tag{6}$$

where ν, L and the initial and boundary conditions are fixed, but we are asked to play around different values of ρ. Then, we will solve the problem and we will be showing different results based on the value of ρ. We will comment different conclusions, making references to results seen in theory classes.

We will start defining the grid in order to solve the problem. This means that we have to define different variables such as the number of interior points, the Courant number or the Δx and Δt for example. Everything has been defined in the *Mathematica* notebook with little comments that facilitate the understanding. Once the grid is correctly defined, we proceed with the solution of the problem. Each Courant number that we are going to use has associated a different grid.

First of all, we will plot just the initial condition. This plot is the same for every value of ρ we are going to consider.

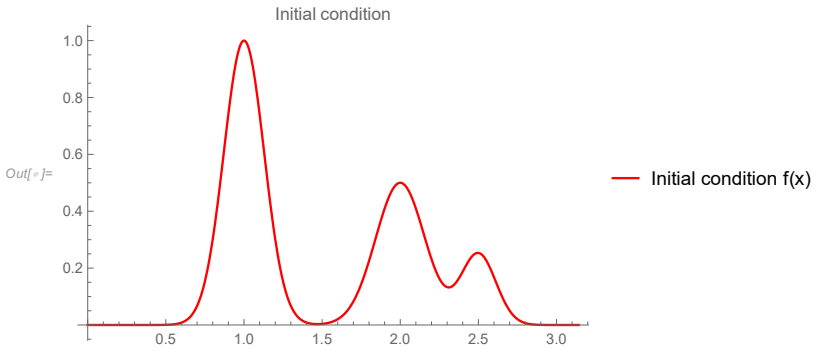

Figure 16: Initial condition f(x)

Making use of the FD method we can approximate the solution. Taking the grid mentioned before and defining the explicit FD approximation in a correct way, we can get a good approximation to the result. This stuff can be found in the *Mathematica* notebook.

Since we have three different values for ρ, we are going to plot three different graphics. These three graphics are the approximated solution when we are at time t = 0. Each graphic is related with its value of ρ.

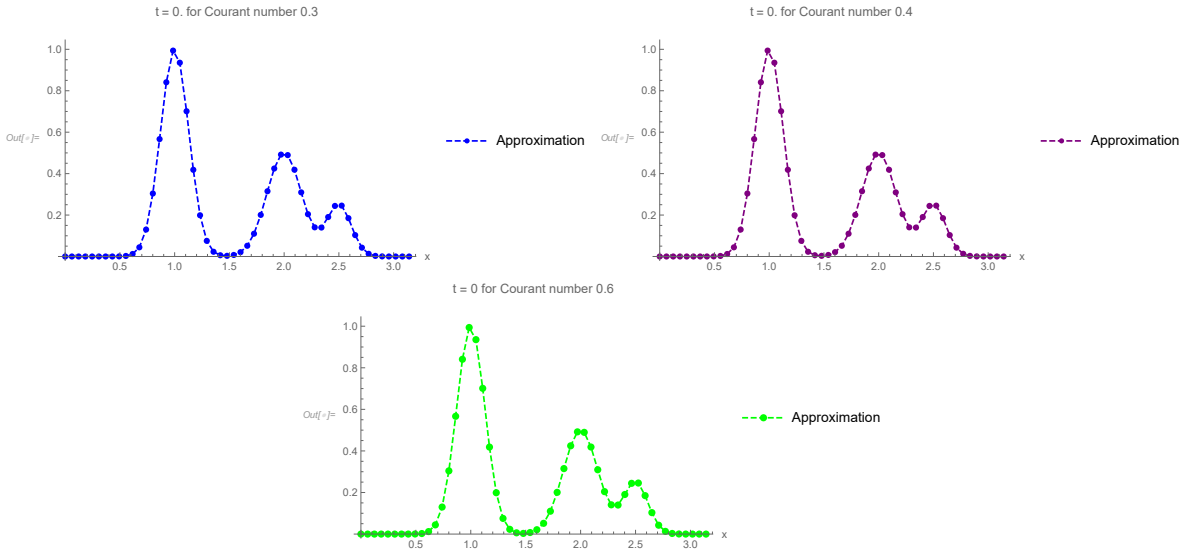

Figure 17: Approximated solution when t = 0

Clearly the Initial Conditions are satisfied, since the approximation at t = 0 is similar to the figure 16 which refers to the plot of the initial condition.

Now, before commenting the results, let's see the different 3D plots we obtain with *Classic Explicit FD approximation*.

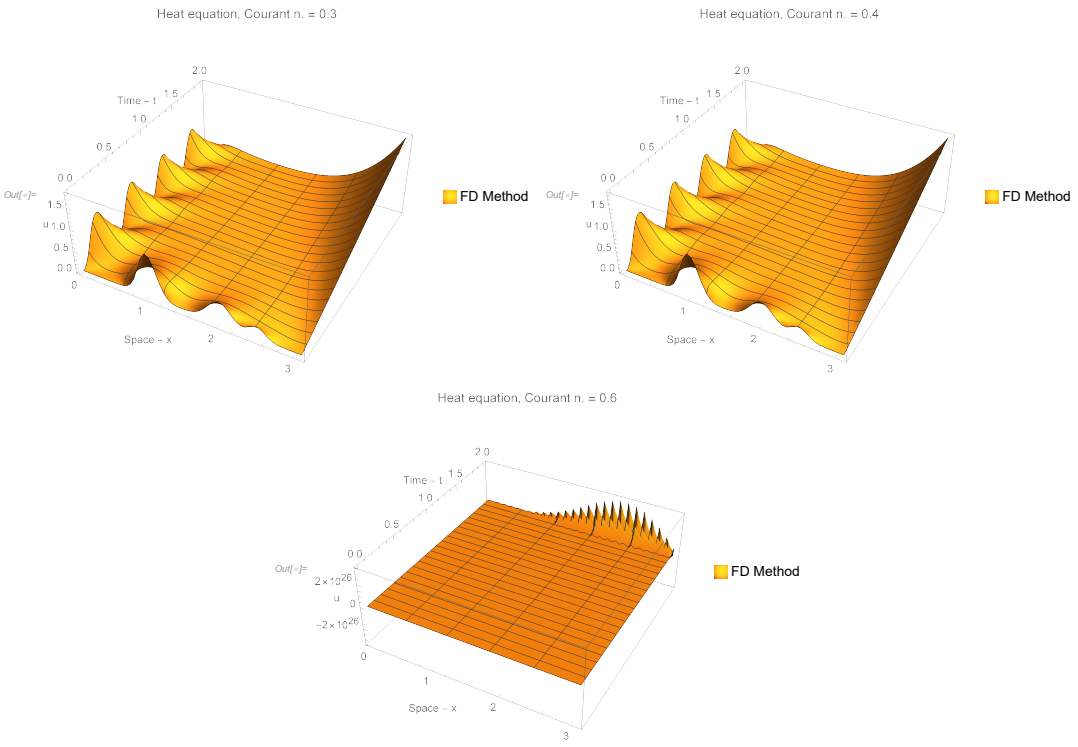

Figure 18: 3D Plots

It is obvious that with Courant numbers 0.3 and 0.4 there are not inconveniences, the approximation is quite good and boundary conditions are satisfied. Nevertheless, for $\rho = 0.6$ the method does not work properly. We know that *Classic Explicit FD approximation* requires a Courant number in the interval (0,0.5] to have stability. So, this case is a clear example of what happens when there is not stability guaranteed. This result we have just mentioned coincides with what we have seen in theory classes.

To have a better idea of the approximation, let's compare the obtained results with MOL. In order to get a good and appropriate approximated solution, we have taken n = 201 interior points in the grid. The solution can be seen in the *Mathematica* notebook.

Next plots are the ones that will help us to compare both results. Let's plot both results together and let's see what differences are between both.

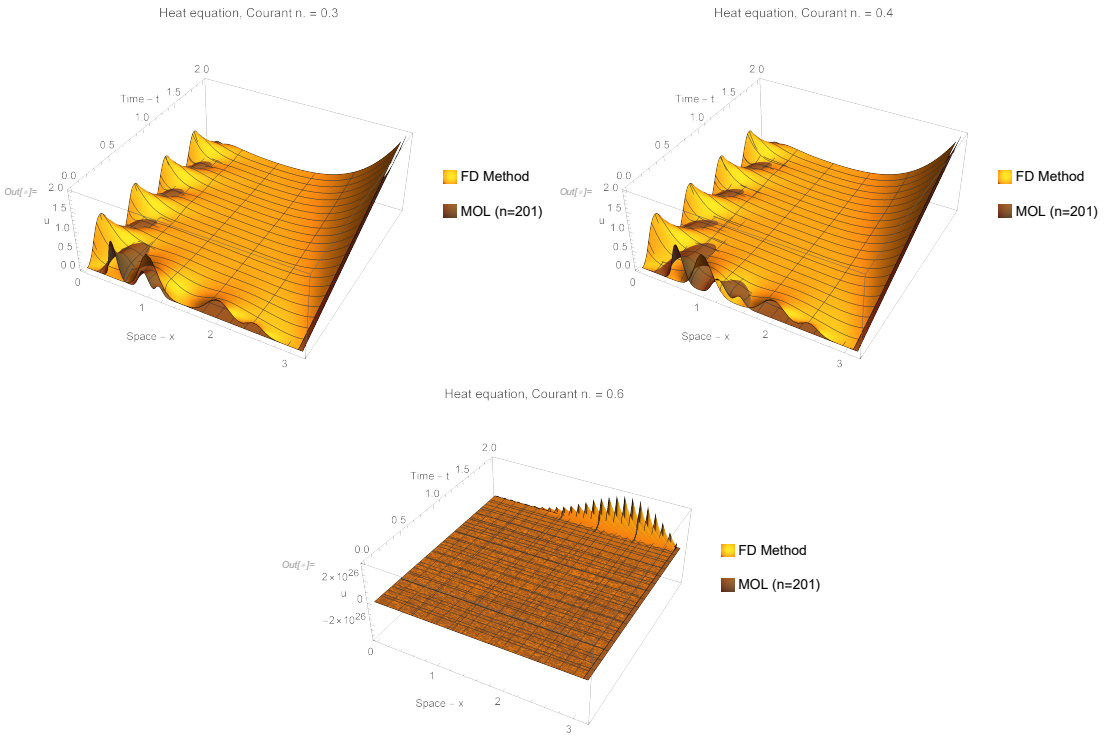

Figure 19: Results obtained with MOL

We appreciate that the approximation with MOL is slightly different, but it works quite well for the last case. In terms of computational time we can say that MOL method is faster as we can see in the *Mathematica* Notebook. The computational times can vary depending on the computer used in the execution.

Problem 2

Statement

The **Richardson Explicit approximation - or Leapfrog scheme**

$$u_s^{r+1} = u_s^{r-1} + 2\,\rho\left(u_{s+1}^r + u_{s-1}^r\right) - 4\,\rho\,u_s^r. \tag{7}$$

is $\mathcal{O}(h^2 + k^2)$, but it turns out to be unconditionally unstable. In any case,

(c) Write a program in *Mathematica* that solves (**??**) using FD (7), in the case $\nu = 1/4$, $L = 1$, $n = 20$, with the IC $f(x)$ as in (**??**) and BC

$$g_1(t) = \frac{t}{1+3\,t}, \qquad g_2(t) = 0. \tag{8}$$

Use, for example, $\rho = 0.3$ and $\Delta t = \rho\,\Delta x^2/\nu$, and try to reach $t_{final} = 0.1$. Then repeat for $\Delta t/100$. What is happening?

Solution

We will solve the problem with the Richardson Explicit approximation. We will consider two different cases, where we will compare the solution obtained when Δt is varied.

Firstly, we are going to verify that the initial conditions IC are satisfied. In fact, we can see this in the following plot:

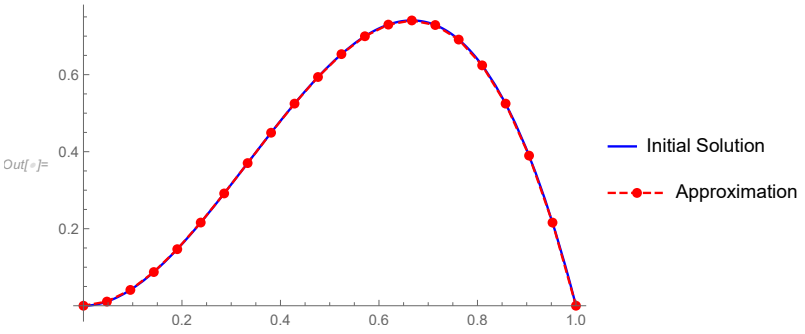

Figure 20: Initial plot

As we wanted the approximation in t = 0 is the function given, so we conclude that the IC are satisfied. Since boundary conditions are u(1) = u(0) = 0 when t = 0, we can say that also boundary conditions are satisfied as the graphic shows.

Now, let's see the results for $\Delta t = \rho\,\Delta x^2/\nu$. As we know by theory, the *Richardson Explicit approximation* is unconditionally unstable, then we expect that the solution is not going to be very good in terms of stability. If we plot the 3D solution, what we get is:

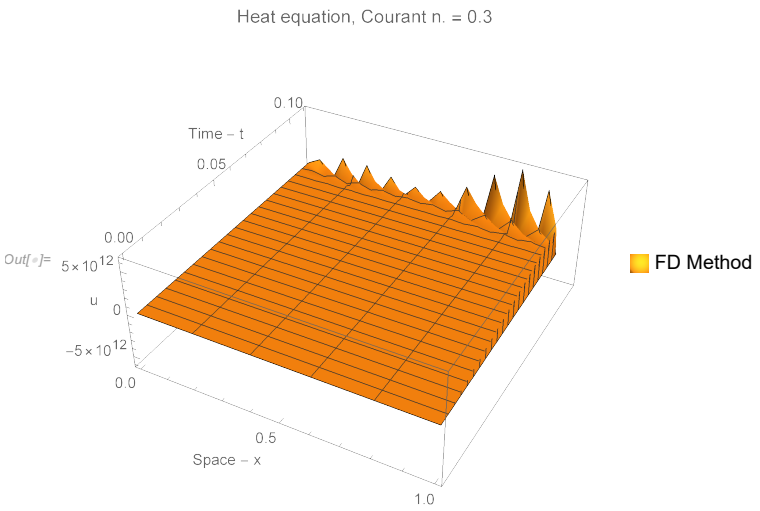

Figure 21: 3D plot of FD of the Richardson Explicit approx

As we expected the approximation is not good. We appreciate that around t = 0.1 there are some ripples.

Let's see how chaotic the solution is when we plot different times together:

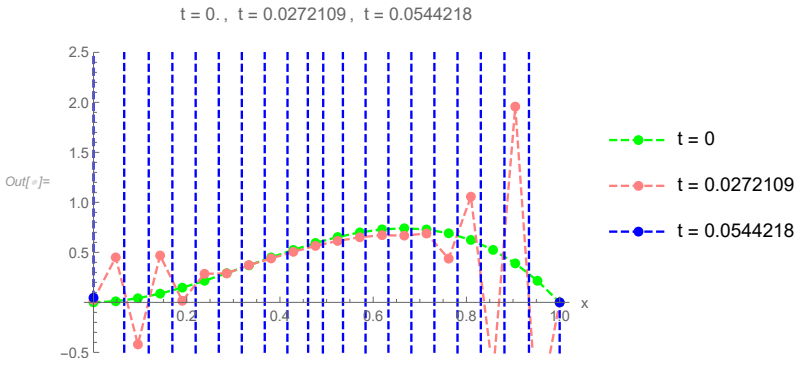

Figure 22: Solution for different times

Now, we will analyse the results for $\Delta t/100$. The IC are satisfied again, since we obtain practically the same plot as in figure 20.

Once again we expect a bad approximation of the solution due to the instability we have commented previously. So, the 3D plot obtained is:

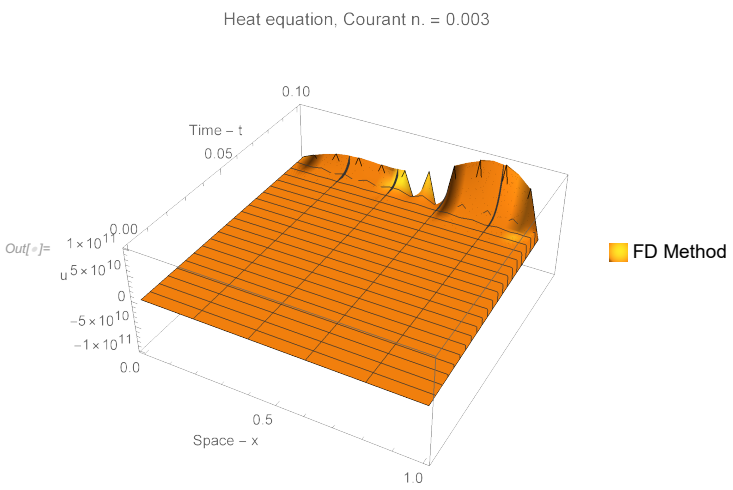

Figure 23: 3D plot

It is obvious that it is not an acceptable solution, since we observe strange things as we reach t = 0.1. Moreover, as Δt gets smaller, since we have more points, the computational time gets higher. Therefore, this relation is important to take into account when solving the problem.

Problem 4

Statement

The **Backwards Implicit approximation**

$$(1+2\,\rho)\,u_s^{r+1} - \rho\left(u_{s+1}^{r+1} + u_{s-1}^{r+1}\right) = u_s^r. \tag{9}$$

which is unconditionally stable.

(e) Write a program in *Mathematica* that solves (8) using FD (9), in the case $\nu = 1/4$, $L = 1$, $n = 20$, with the IC $f(x)$ as in (1) and BC as in (8). Use, for example, $\Delta t = 0.05$. Then repeat for $\Delta t/10$, $\Delta t/100$ and $10\,\Delta t$. Which are the values of the Courant numbers? What is happening?

Plot the 3D graphics for the obtained solutions for $0 \le x \le 1$ and $0 \le t \le 1$.

Solution

In this problem, we have to solve the problem using *Backwards Implicit approximation*. It is known from theory that this method is unconditionally stable, so the results will be good.

We will compare the solutions for four cases of Δt. But first, let's check that initial conditions are satisfied. The following plot shows that the function and the approximation at t = 0 are practically equal, that is the IC conditions are satisfied.

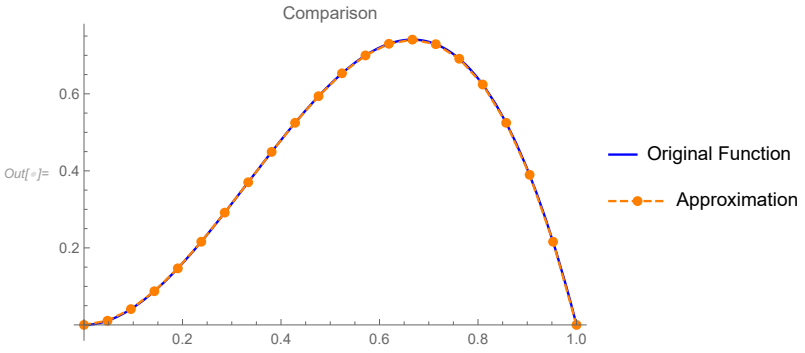

Figure 24: Initial plot

Attending to the boundary conditions, we can observe that u(0) = u(1) = 0, thus we can see that both conditions are satisfied since we are attending to the graphic at time t = 0.

Now, let's see the four different 3D plots obtained:

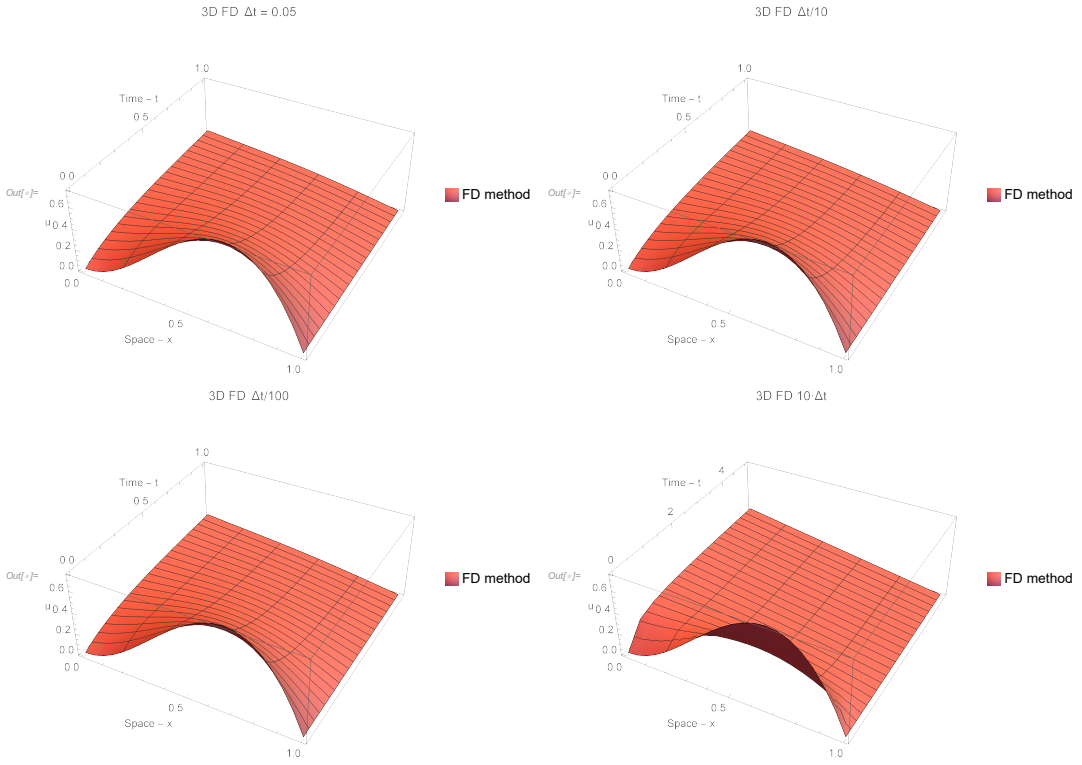

Figure 25: 3D plots for different values of Δt

We see that the solutions are very similar, except the last one where the heat dissipate faster than in the other cases. Moreover, there is not strange phenomenon as expected.

The Courant number in each case is:

- $\rho = 5.5125$
- $\rho = 0.55125$
- $\rho = 0.055125$
- $\rho = 55.125$

Now, let's compare this results with MOL. We will observe light differences on the corners for all the cases. But in general the solutions obtained with both methods are quite similar.

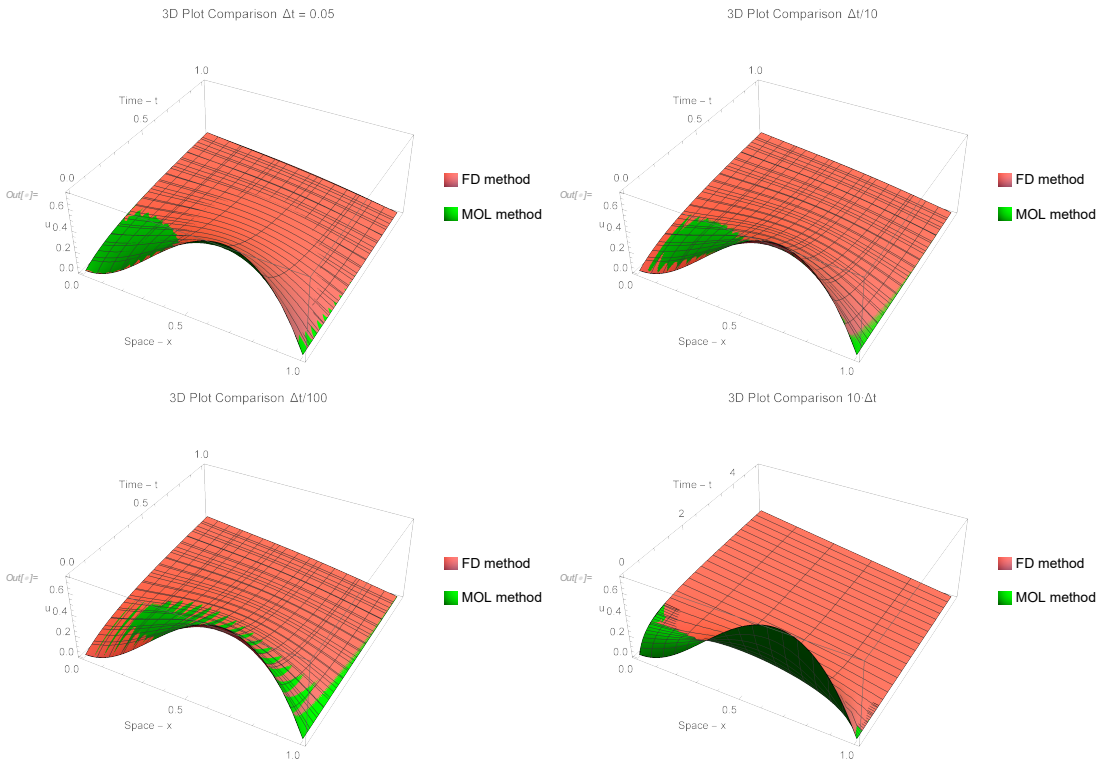

Figure 26: Comparison with MOL

In conclusion, by solving the problems we have seen the importance of the stability of the method. Furthermore, we also have experiment the relation between the Courant number ρ and Δt, have the same behaviour. It should be noted that, smaller Δt implies more computational time. Finally, if we compare the computational of both methods, we see that *Backwards Implicit approximation* is faster when Δt is quite big for example for $\Delta t = 0.05$ and $\Delta t = 0.5$, but in these cases we have state that the approximation is not good. However in the other cases where the approximation is acceptable the MOL is faster than the implementation of *Backwards Implicit approximation*.