I pledge that the work submitted for this coursework, both the report and the MATLAB code, is my own unassisted work unless stated otherwise.

CID........................ 01730921

Are you a Year 4 student?.. yes

Coursework 4

Fill in your CID and include the problem sheet in the coursework. Before you start working on the coursework, read the coursework guidelines. Any marks received for this coursework are only indicative and may be subject to moderation and scaling. *The mastery component is marked with a star.*

Exercise 1 is for Year-3 students only.

Exercise 1 (Explicit Runge–Kutta methods) % of course mark: /5.0

a) Derive all order conditions for the general 4-stage Explicit Runge–Kutta (ERK) method of order 4.
b) Develop your own ERK method of order 4 with automatic step size control.
c) Find its region and interval of absolute stability.

Exercise 2 (Implicit Runge–Kutta methods) % of course mark: /5.0*

a) Derive all order conditions for the general 2-stage Implicit Runge–Kutta (IRK) method of order 4.
b) Develop your own IRK method of order 4 with automatic step size control.
c) Find its region and interval of absolute stability.

Exercise 3 is for Year-3 students only.

Exercise 3 (Numerical solution of linear PDEs) % of course mark: /10.0

a) Solve the differential equation in the periodic domain:

$$u_t = \varepsilon u_{xx}\,, \quad x \in [0, L], \quad t = (0, 10], \quad L = 10, \quad u(0, x) = \mathrm{e}^{-L^2(x-L/2)^2}, \quad \varepsilon = 0.1 \tag{1}$$

with the RK method developed in Exercise 1. Use the second order approximation for u_{xx}.

b) Present the numerical solution as a waterfall graph (*waterfall* function in MATLAB).

Exercise 4 (Numerical solution of BVP for PDEs) % of course mark: /10.0

a) Solve the boundary value problem:

$$u_t + vu_x = \varepsilon u_{xx}\,, \quad u_x(t, 0) = u_x(t, L) = 0\,, \quad x \in [0, L], \quad t = (0, 1], \tag{2}$$

$$u(0, x) = \cos(2\pi x/L), \quad v = 1.0, \quad L = 10, \quad \varepsilon = 0.1\,,$$

with the RK method developed in Exercise 1/2 (for Year-3/Year-4 students). Use the second order approximation for u_x and u_{xx}.

b) Present the numerical solution as a waterfall graph (*waterfall* function in MATLAB).

In hand date: 29 Nov 2022 **Due date: 13 Dec 2022**

Exercise 5 (Numerical solution of nonlinear PDEs) **% of course mark:** **/16.0★**

a) Solve the Camassa–Holm equation in the periodic domain:

$$m_t + mu_x + (mu)_x = 0, \quad x \in [0, L], \quad t = (0, 400], \quad u(0, x) = \frac{1}{10}\mathrm{e}^{-5(x-L\frac{1}{3})^2} - \frac{1}{10}\mathrm{e}^{-5(x-L\frac{2}{3})^2} \tag{3}$$

with an implicit method of your choice; $m = u - u_{xx}$, $L = 100$. Use the Newton method to solve the system of nonlinear equations. For the space approximation use the following approximation:

$$m_t^n + m^n Du^n + D(m^n u^n) = 0, \quad m^n = u^n - D^- D^+ u^n,$$

where D^+ and D^- denote the forward and backward difference operators, and $D = (D^+ + D^-)/2$.

b) Present the numerical solution as a waterfall graph (*waterfall* function in MATLAB).

Coursework mark: **% of course mark**

Coursework Guidelines

Below is a set of guidelines to help you understand what coursework is and how to improve it.

Coursework

- The coursework requires more than just following what has been done in the lectures, some amount of individual work is expected.
- The coursework report should describe in a concise, clear, and coherent way of what you did, how you did it, and what results you have.
- The report should be understandable to the reader with the mathematical background, but unfamiliar with your current work.
- Do not bloat the report by paraphrasing or presenting the results in different forms.
- Use high-quality and carefully constructed figures with captions and annotated axis, put figures where they belong.
- All numerical solutions should be presented as graphs.
- Use tables only if they are more explanatory than figures. The maximum table length is a half page.
- All figures and tables should be embedded in the report. The report should contain all discussions and explanations of the methods and algorithms, and interpretations of your results and further conclusions.
- The report should be typeset in LaTeX or Word Editor and submitted as a single pdf-file.
- The maximum length of the report is ten A4-pages (additional 3 pages is allowed for Year 4 students); the problem sheet is not included in these ten pages.
- Do not include any codes in the report.
- Marks are not based solely on correctness. The results must be described and interpreted. The presentation and discussion is as important as the correctness of the results.

Codes

- You cannot use third party numerical software in the coursework.
- The code you developed should be well-structured and organised, as well as properly commented to allow the reader to understand what the code does and how it works.
- All codes should run out of the box and require no modification to generate the results presented in the report.

Submission

- The coursework submission must be made via Turnitin on your Blackboard page. You must complete and submit the coursework anonymously, **the deadline is 1pm on the date of submission** (unless stated otherwise). The coursework should be submitted via two separate Turnitin drop boxes as a pdf-file of the report and a zip-file containing MATLAB (m-files only) or Python (py-files only) code. The code should be in the directory named `CID_Coursework#`. The report and the zip-file should be named as `CID_Coursework#.pdf` and `CID_Coursework#.zip` , respectively. The executable MATLAB (or Python) scripts for the exercises should be named as follows: exercise1.m, exercise2.m, etc.

Numerical odes - Coursework 4

CID: 01730921

1 Question 1 - Master student

1.1 Derivation of order conditions for 2-stage IRK scheme

In this first section, we concern ourselves with deriving the order conditions for the general 2-stage Implicit Runge-Kutta (IRK) method of order 4. Presented as a Butcher tablea, this has the from

$$\begin{array}{c|cc} c_1 & a_{11} & a_{12} \\ c_2 & a_{21} & a_{22} \\ \hline & b_1 & b_2 \end{array} \quad \Longrightarrow \quad \begin{aligned} k_1 &= f(t_n + hc_1, x_n + h(a_{11}k_1 + a_{12}k_2)), \\ k_2 &= f(t_n + hc_2, x_n + h(a_{21}k_1 + a_{22}k_2)), \\ x_{n+1} &= x_n + h(b_1k_1 + b_2k_2) \end{aligned} \tag{1}$$

with $c_1 = a_{11} + a_{12}$ and $c_2 = a_{21} + a_{22}$ by definition.

For this derivation, we notice the simmetries in form between k_1 and k_2. For now, we thus focus on Taylor expansion of k_1 about (t_n, x_n) up to order $\mathcal{O}(h^4)$ as for k_2's it can be deduce consequently. The Taylor expansion of $f(t_n + \alpha h, x_n + \beta h)$ about (t_n, x_n) is given by definition by:

$$\begin{aligned} f(t_n + \alpha h, x_n + \beta h) = f(t_n, x_n) + h(\alpha f_t + \beta f_x) + \frac{h^2}{2}(\alpha^2 f_{tt} + 2f_{tx}\alpha\beta + \beta^2 f_{xx}) \\ + \frac{h^3}{6}(\alpha^3 f_{ttt} + 3f_{ttx}\alpha^2\beta + 3f_{txx}\alpha\beta^2 + \beta^3 f_{xxx}) + \mathcal{O}(h^4). \end{aligned} \tag{2}$$

In our case, we have:

$$\begin{aligned} k_1 = f + h(c_1 f_t + (a_{11}k_1 + a_{12}k_2)f_x) + \frac{h^2}{2}(c_1^2 f_{tt} + 2c_1(a_{11}k_1 + a_{12}k_2)f_{tx} + (a_{11}k_1 + a_{12}k_2)^2 f_{xx}) \\ + \frac{h^3}{6}(c_1^3 f_{ttt} + 3c_1^2(a_{11}k_1 + a_{12}k_2)f_{ttx} + 3c_1(a_{11}k_1 + a_{12}k_2)^2 f_{txx} + (a_{11}k_1 + a_{12}k_2)^3 f_{xxx}) + \mathcal{O}(h^4). \end{aligned} \tag{3}$$

Similarly, for k_2:

$$\begin{aligned} k_2 = f + h(c_2 f_t + (a_{21}k_1 + a_{22}k_2)f_x) + \frac{h^2}{2}(c_2^2 f_{tt} + 2c_2(a_{21}k_1 + a_{22}k_2)f_{tx} + (a_{21}k_1 + a_{12}k_2)^2 f_{xx}) \\ + \frac{h^3}{6}(c_2^3 f_{ttt} + 3c_2^2(a_{21}k_1 + a_{22}k_2)f_{ttx} + 3c_2(a_{21}k_1 + a_{22}k_2)^2 f_{txx} + (a_{21}k_1 + a_{22}k_2)^3 f_{xxx}) + \mathcal{O}(h^4). \end{aligned} \tag{4}$$

We plug in (3) the Taylor expansion of k_1 and k_2 given in (3) and (4), such that the remainder of k_1 keeps being $O(h^4)$. The aim of this nested plugging of Taylor expansions is that we get to a point where terms greater than $O(h^4)$ do not have k_1 or k_2 inside. Then,

$$\begin{aligned}
k_1 = f + h\Bigg[&c_1 f_t + \Bigg(a_{11}\Big(f + h(c_1 f_t + (a_{11}k_1 + a_{12}k_2)f_x) + \frac{h^2}{2}(c_1^2 f_{tt} + 2c_1(a_{11}k_1 + a_{12}k_2)f_{tx} + (a_{11}k_1 + a_{12}k_2)^2 f_{xx})\Big) \\
&+ a_{12}\Big(f + h(c_2 f_t + (a_{21}k_1 + a_{22}k_2)f_x) + \frac{h^2}{2}(c_2^2 f_{tt} + 2c_2(a_{21}k_1 + a_{22}k_2)f_{tx} + (a_{21}k_1 + a_{22}k_2)^2 f_{xx})\Big)\Bigg)f_x\Bigg] \\
&+ \frac{h^2}{2}\Bigg[c_1^2 f_{tt} + 2c_1\Bigg(a_{11}\Big(f + h(c_1 f_t + (a_{11}k_1 + a_{12}k_2)f_x)\Big) + a_{12}\Big(f + h(c_2 f_t + (a_{21}k_1 + a_{22}k_2)f_x)\Big)\Bigg)f_{tx} \\
&+ \Bigg(a_{11}\Big(f + h(c_1 f_t + (a_{11}k_1 + a_{12}k_2)f_x)\Big) + a_{12}\Big(f + h(c_2 f_t + (a_{21}k_1 + a_{22}k_2)f_x)\Big)\Bigg)^2 f_{xx}\Bigg] \\
&+ \frac{h^3}{6}\Bigg[c_1^3 f_{ttt} + 3c_1^2(a_{11} + a_{12})f f_{ttx} + 3c_1(a_{11} + a_{12})^2 f^2 f_{txx} + (a_{11} + a_{12})^3 f^3 f_{xxx}\Bigg] + \mathcal{O}(h^4).
\end{aligned}$$

Taylor expanding again:

$$\begin{aligned}
k_1 = f + h\Bigg[&c_1 f_t + \Bigg(a_{11}\Bigg(f + h\Bigg(c_1 f_t + \Bigg(a_{11}\Big(f + h(c_1 f_t + (a_{11}k_1 + a_{12}k_2)f_x)\Big) \\
&\quad + a_{12}\Big(f + h(c_2 f_t + (a_{21}k_1 + a_{22}k_2)f_x)\Big)\Bigg)f_x\Bigg) + \frac{(c_1 h)^2}{2}(f_{tt} + 2f f_{tx} + f^2 f_{xx})\Bigg) \\
&\quad + a_{12}\Bigg(f + h\Bigg(c_2 f_t + \Bigg(a_{21}\Big(f + h(c_1 f_t + (a_{11}k_1 + a_{12}k_2)f_x)\Big) + a_{22}\Big(f + h(c_2 f_t + (a_{21}k_1 + a_{22}k_2)f_x)\Big)\Bigg)f_x\Bigg) \\
&\quad + \frac{(c_2 h)^2}{2}(f_{tt} + 2f f_{tx} + f^2 f_{xx})\Bigg)\Bigg)f_x\Bigg] \\
&\quad + \frac{h^2}{2}\Bigg[c_1^2 f_{tt} + 2c_1\Bigg(a_{11}\Big(f + c_1 h(f_t + f f_x)\Big) + a_{12}\Big(f + c_2 h(f_t + f f_x)\Big)\Bigg)f_{tx} \\
&\quad + \Bigg(a_{11}\Big(f + c_1 h(f_t + f f_x)\Big) + a_{12}\Big(f + c_2 h(f_t + f f_x)\Big)\Bigg)^2 f_{xx}\Bigg] \\
&\quad + \frac{(c_1 h)^3}{6}\Bigg[f_{ttt} + 3f f_{ttx} + 3f^2 f_{txx} + f^3 f_{xxx}\Bigg] + \mathcal{O}(h^4)
\end{aligned}$$

Taylor expanding again one last time up to order $\mathcal{O}(h)$ the remaining k-s, and using $f' = f_t + f_x f$:

$$\begin{aligned}
k_1 = f + h\Bigg[&c_1 f' + \Bigg(a_{11}\Big(h(c_1 f_t + (c_1 f + (a_{11}c_1 + a_{12}c_2)hf')f_x) + \frac{(c_1 h)^2}{2}(f_{tt} + 2f f_{tx} + f^2 f_{xx})\Big) \\
&+ a_{12}\Big(h(c_2 f_t + (c_2 f + (a_{21}c_1 + a_{22}c_2)hf')f_x) + \frac{(c_2 h)^2}{2}(f_{tt} + 2f f_{tx} + f^2 f_{xx})\Big)\Bigg)f_x\Bigg] \\
&+ \frac{h^2}{2}\Bigg[c_1^2 f_{tt} + 2c_1(c_1 f + (a_{11}c_1 + a_{12}c_2)hf')f_{tx} + (c_1 f + (a_{11}c_1 + a_{12}c_2)hf')^2 f_{xx}\Bigg] \\
&+ \frac{(c_1 h)^3}{6}\Bigg[f_{ttt} + 3f f_{ttx} + 3f^2 f_{txx} + f^3 f_{xxx}\Bigg] + \mathcal{O}(h^4).
\end{aligned}$$

To simplify things, we denote as $\eta_1 = a_{11}c_1 + a_{12}c_2$, $\eta_2 = a_{21}c_1 + a_{22}c_2$. Hence:

$$
\begin{aligned}
k_1 &= f + h\Big[c_1 f' + h\Big(\eta_1 f' + (a_{11}\eta_1 + a_{12}\eta_2)hf'f_x + (a_{11}c_1^2 + a_{12}c_2^2)\frac{h}{2}(f_{tt} + 2ff_{tx} + f^2 f_{xx})\Big)f_x\Big] \\
&\quad + \frac{h^2}{2}\Big(c_1^2(f_{tt} + 2ff_{tx} + f^2 f_{xx}) + 2hc_1\eta_1 f'(f_{tx} + ff_{xx})\Big) + \frac{(c_1 h)^3}{6}(f_{ttt} + 3ff_{ttx} + 3f^2 f_{txx} + f^3 f_{xxx}) + \mathcal{O}(h^4), \\
k_2 &= f + h\Big[c_2 f' + h\Big(\eta_2 f' + (a_{21}\eta_1 + a_{22}\eta_2)hf'f_x + (a_{21}c_1^2 + a_{22}c_2^2)\frac{h}{2}(f_{tt} + 2ff_{tx} + f^2 f_{xx})\Big)f_x\Big] \\
&\quad + \frac{h^2}{2}\Big(c_2^2(f_{tt} + 2ff_{tx} + f^2 f_{xx}) + 2hc_2\eta_2 f'(f_{tx} + ff_{xx})\Big) + \frac{(c_2 h)^3}{6}(f_{ttt} + 3ff_{ttx} + 3f^2 f_{txx} + f^3 f_{xxx}) + \mathcal{O}(h^4),
\end{aligned}
$$

Deducing by symmetry arguments the expansion of k_2 we substitute these into the last equation of the general form of the IRK2, to get:

$$
\begin{aligned}
x_{n+1} = x_n &+ h(b_1 + b_2)f + h^2(b_1c_1 + b_2c_2)f' + h^3\Big((b_1\eta_1 + b_2\eta_2)f'f_x + \frac{b_1c_1^2 + b_2c_2^2}{2}(f_{tt} + 2ff_{tx} + f^2 f_{xx})\Big) \\
&+ h^4\Big[b_1\Big((a_{11}\eta_1 + a_{12}\eta_2)f'f_x + \frac{a_{11}c_1^2 + a_{12}c_2^2}{2}(f_{tt} + 2ff_{tx} + f^2 f_{xx})\Big)f_x \\
&+ b_2\Big((a_{21}\eta_1 + a_{22}\eta_2)f'f_x + \frac{a_{21}c_1^2 + a_{22}c_2^2}{2}(f_{tt} + 2ff_{tx} + f^2 f_{xx})\Big)f_x \\
&+ (b_1c_1\eta_1 + b_2c_2\eta_2)f'(f_{tx} + ff_{xx}) + \frac{b_1c_1^3 + b_2c_2^3}{6}(f_{ttt} + 3ff_{ttx} + 3f^2 f_{txx} + f^3 f_{xxx})\Big] + \mathcal{O}(h^5)
\end{aligned}
$$

The Taylor expansion of the exact solution is given by

$$
\begin{aligned}
&x(t_{n+1}) = x(t_n) + hf + \frac{h^2}{2}(f_t + ff_x) + \frac{h^3}{6}(f_{tt} + 2f_{tx}f + f_{xx}f^2 + f_x(f_t + f_x f)) \\
&+ \frac{h^4}{24}\Big((f_{ttt} + 3ff_{ttx} + 3f^2 f_{txx} + f^3 f_{xxx}) + (f_{tt} + 2ff_{tx} + f^2 f_{xx})f_x + f'f_x^2 + 3f'(f_{tx} + ff_{xx})\Big) + \mathcal{O}(h^5), \quad (5)
\end{aligned}
$$

where we used the fact that

$$
f^{(n)}(x) = \frac{df^{(n-1)}}{dt} = \frac{\partial f^{(n-1)}}{\partial t}\frac{dt}{dt} + \frac{\partial f^{(n-1)}}{\partial x}\frac{dx}{dt} = \frac{\partial f^{(n-1)}}{\partial t} + f\frac{\partial f^{(n-1)}}{\partial x}. \quad (6)
$$

Hence, the local truncation error given by $x(t_{n+1}) - x_{n+1}$, where $x_{n+1} = x_n + h(b_1k_1 + b_2k_2)$ is

$$
\begin{aligned}
LTE &:= x(t_{n+1}) - x_{n+1} = (1 - (b_1 + b_2))hf + \Big(\frac{1}{2} - (b_1c_1 + b_2c_2)\Big)h^2 f' \\
&+ h^3\Big[\Big(\frac{1}{6} - (b_1\eta_1 + b_2\eta_2)\Big)f'f_x + \Big(\frac{1}{6} - \frac{b_1c_1^2 + b_2c_2^2}{2}\Big)(f_{tt} + 2ff_{tx} + f^2 f_{xx})\Big] \\
&+ h^4\Big[\Big(\frac{1}{24} - b_1(a_{11}\eta_1 + a_{12}\eta_2) - b_2(a_{21}\eta_1 + a_{22}\eta_2)\Big)f'f_x^2 \\
&+ \Big(\frac{1}{24} - \frac{b_1(a_{11}c_1^2 + a_{12}c_2^2) + b_2(a_{21}c_1^2 + a_{22}c_2^2)}{2}\Big)(f_{tt} + 2ff_{tx} + f^2 f_{xx})f_x \\
&+ \Big(\frac{1}{8} - (b_1c_1\eta_1 + b_2c_2\eta_2)\Big)f'(f_{tx} + ff_{xx}) + \Big(\frac{1}{24} - \frac{b_1c_1^3 + b_2c_2^3}{6}\Big)(f_{ttt} + 3ff_{ttx} + 3f^2 f_{txx} + f^3 f_{xxx})\Big] + \mathcal{O}(h^5).
\end{aligned}
$$

Setting the LTE equal to 0 up to order 4 included, we get the first four order conditions as follows:

1. **First and second order conditions:**

$$1^{st}: \; b_1 + b_2 = 1, \qquad 2^{nd}: \; b_1c_1 + b_2c_2 = \frac{1}{2} \tag{7}$$

2. **Third order conditions:**

$$b_1c_1^2 + b_2c_2^2 = \frac{1}{3}, \quad b_1(a_{11}c_1 + a_{12}c_2) + b_2(a_{21}c_1 + a_{22}c_2) = \frac{1}{6} \tag{8}$$

3. **Fourth order conditions:**

$$\begin{aligned} &b_1(a_{11}(a_{11}c_1 + a_{12}c_2) + a_{12}(a_{21}c_1 + a_{22}c_2)) + b_2(a_{21}(a_{11}c_1 + a_{12}c_2) + a_{22}(a_{21}c_1 + a_{22}c_2)) = \frac{1}{24}, \\ &b_1(a_{11}c_1^2 + a_{12}c_2^2) + b_2(a_{21}c_1^2 + a_{22}c_2^2) = \frac{1}{12}, \qquad b_1c_1^3 + b_2c_2^3 = \frac{1}{4} \\ &b_1c_1(a_{11}c_1 + a_{12}c_2) + b_2c_2(a_{21}c_1 + a_{22}c_2) = \frac{1}{8} \end{aligned} \tag{9}$$

1.2 IRK(4) and automatic step size

To develop a fourth-order two-stage implicit method we have to find the values of the a-s, b-s and c-s such that all the four order conditions are satisfied. This is a non trivial problem as the system of equation is nonlinear. Exploiting symmetry and linearity of the equations in b_1, b_2 we choose $b_1 = b_2 = 1/2$. This leads to

$$\begin{cases} b_1c_1 + b_2c_2 = 1/2 \\ b_1c_1^2 + b_2c_2^2 = 1/3 \\ b_1c_1^3 + b_2c_2^3 = 1/4 \end{cases} \implies \begin{cases} c_1 + c_2 = 1 \\ c_1^2 + c_2^2 = 2/3 \\ c_1^3 + c_2^3 = 1/2 \end{cases} \implies c_{1,2} = \frac{3 \pm \sqrt{3}}{6} \tag{10}$$

We are now left with 4 equations, 3 of which are linear in a_{11}, a_{12}, a_{21}, a_{22},. Trying to enforce symmetry once again by setting $a_{11} = a_{22}$, it leads us to:

$$a_{11} = a_{22} = \frac{1}{4}, \qquad a_{12,21} = \frac{3 \pm 2\sqrt{3}}{12} \tag{11}$$

Automatic step size

To implement automatic step size we need two numerical schemes that have order p and $p-1$. This could be either done by implementing an RK(5) method to extract the new time step, or by implementing an RK method of order 3 that uses the aforementioned RK(4) to extract h_{new}. Based on Dr. Shevchenko's instructions, we choose to implement the latter strategy. To keep computations as simple as possible we select an explicit

Runge-Kutta of order 3, namely the *Heun's method*, that is mentioned in the lecture notes. Hence:

$$\begin{aligned}
&\text{RK}(3): && \text{RK}(4): \\
&k_1 = f(t_n, x_n) && \tilde{k}_1 = f(t_n + h_n c_1, x_n + h_n(a_{11}\tilde{k}_1 + a_{12}\tilde{k}_2)) \\
&k_2 = f(t_n + h_n/3, x_n + h_n k_1/3) && \tilde{k}_2 = f(t_n + h_n c_2, x_n + h_n(a_{21}\tilde{k}_1 + a_{22}\tilde{k}_2)) \\
&k_3 = f(t_n + 2h_n/3, x_n + 2h_n k_2/3) && x_{n+1}^{<4>} = x_n + h_n(\tilde{k}_1 + \tilde{k}_2)/2 \\
&x_{n+1}^{<3>} = x_n + h_n(k_1 + 3k_3)/4 && a_{11,22} = 1/4,\ a_{12,21} = (3 \pm 2\sqrt{3})/12,\ c_{1,2} = (3 \pm \sqrt{3})/6
\end{aligned} \tag{12}$$

where h_n is the step size at time n. The error of each method with respect to the true solution is

$$e_n^{<3>} = x(t_n) - x_n^{<3>} = \mathcal{O}(h^3), \qquad e_n^{<4>} = x(t_n) - x_n^{<4>} = \mathcal{O}(h^4) \tag{13}$$

Rearranging we can express $x(t_n)$ as $x(t_n) = x_n^{<4>} + \mathcal{O}(h^4)$. Hence $e_n^{<3>} = x_n^{<4>} - x_n^{<3>} + \mathcal{O}(h^4)$. The local truncation error of our RK(3) method can thus be estimated as

$$R_{n+1} = x_{n+1}^{<4>} - x_{n+1}^{<3>} = h_n\left(\frac{\tilde{k}_1 + \tilde{k}_2}{2} - \frac{k_1 + 3k_3}{4}\right) \tag{14}$$

Moreover, LTE can be written as

$$R_{n+1}(h_n) = H(t_n)h_n^4 + \mathcal{O}(h_n^5) \implies R_{n+1}(h_n) \approx H(t_n)h_n^4 \implies H(t_n) = \frac{R_{n+1}(h_n)}{h_n^4}. \tag{15}$$

Since we want $R_{n+1}(h_{\text{new}}) = \epsilon$, where $\epsilon > 0$ is the tolerance, we have

$$\epsilon \approx |H(t_n)|h_{\text{new}}^4 \implies h_{\text{new}}^4 = \frac{\epsilon}{|H(t_n)|} \implies h_{\text{new}}^4 = h_n^4\left|\frac{\epsilon}{R_{n+1}(h_n)}\right| \tag{16}$$

So that the step size control formula becomes

$$h_{\text{new}} = h_n\left|\frac{\epsilon}{R_{n+1}}\right|^{\frac{1}{4}} \tag{17}$$

and this is initiated by $h_0 = \epsilon$ and R_{n+1} is given in (14).

1.3 Region and interval of absolute stability

Definition 1 (**Absolute Stability**). *An numerical scheme is said to be absolutely stable if all roots of the stability polynomial, $p(r)$, are such that $|r_i|_{i\in\{1,\dots,m\}} < 1$ or if any root $|r_k| = 1$ it has to be simple; i.e. the stability polynomial must satisfy the strict root condition.*

Definition 2 (**Region of Absolute Stability**). *The set of values in the complex $\hat{h}$ plane for which the* *numerical scheme is absolutely stable is called the region of absolute stability of the method.*

Definition 3 (**Interval of Absolute Stability**)**.** *The interval $(Re(\hat{h}), 0)$, $Re(\hat{h}) < 0$ for which the numerical scheme is absolutely stable is called the interval of absolute stability of the method. This is simply the intersection of the region of absolute stability with the negative real $\hat{h}$ axis.*

To find the stability polynomial we apply the IRK to $x' = \lambda x$, $\mathrm{Re}(\lambda) < 0$. The k-s become:

$$\begin{aligned} k_1 &= \lambda(x_n + ha_{11}k_1 + ha_{12}k_2) \\ k_2 &= \lambda(x_n + ha_{12}k_1 + ha_{22}k_2) \end{aligned} \tag{18}$$

Solving the above system of equations and substituting the values we have found in (**??**) for the parameters, we get, setting $\hat{h} := \lambda h$:

$$k_1 = \frac{2\hat{h}\sqrt{3} + 12}{\hat{h}^2 - 6\hat{h} + 12}\lambda x_n, \qquad k_2 = -\frac{2\hat{h}\sqrt{3} - 12}{\hat{h}^2 - 6\hat{h} + 12}\lambda x_n$$

$$\implies \quad x_{n+1} = x_n + \frac{12\hat{h}}{\hat{h}^2 - 6\hat{h} + 12}x_n \tag{19}$$

Therefore the stability function of RK(4) reads:

$$R(\hat{h}) = 1 + \frac{12\hat{h}}{(\hat{h}^2 - 6\hat{h} + 12)} = \frac{\hat{h}^2 + 6\hat{h} + 12}{\hat{h}^2 - 6\hat{h} + 12} \tag{20}$$

Setting its absolute value smaller than 1, it can be shown that:

$$\left|\frac{\hat{h}^2 + 6\hat{h} + 12}{\hat{h}^2 - 6\hat{h} + 12}\right| < 1 \quad \implies \quad \mathrm{Re}(\hat{h}) < 0 \tag{21}$$

Therefore the region of absolute stability is the left-half of the complex plane (no plot needed to visualise it). Hence, the interval of absolute stability is the real negative axis i.e. $\hat{h} < 0$. This implies that the method we have found is *A-stable*. When it comes to RK3, from the lecture notes we have that its stability polynomial is

$$p_3(r) = r - \left(1 + \hat{h} + \frac{\hat{h}^2}{2} + \frac{\hat{h}^3}{6}\right) \tag{22}$$

and its interval of absolute stability is given by

$$-12 < \hat{h}(6 + 3\hat{h} + \hat{h}^2) < 0 \tag{23}$$

2 Question 2

Consider the following BVP:

$$u_t = \varepsilon u_{xx} - v u_x, \quad u_x(t,0) = u_x(t,L) = 0, \quad x \in [0,L], \quad t = [0,10] \tag{24}$$
$$u(0,x) = \cos(2\pi x/L), \quad v = 1.0, \quad L = 10, \quad \varepsilon = 0.1$$

We start linearly discretizing the space domain into $M+1$ points where $x_i - x_{i-1} = \Delta x = h, \;\; h = L/(M+1)$.

Throughout this question we make use of the notation $u_i^n = u(t_n, x_i)$ where t_n is the n-th point of the time span and x_i is the i-th point of the space span.

Furthermore, we consider the second order central finite difference approximation in space for u_x and u_{xx}, implying that $\forall i \in 1, 2, \ldots, M-1$

$$(u_i^n)_t \;=\; \varepsilon \, \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{h^2} - v \, \frac{u_{i+1}^n - u_{i-1}^n}{2h} \tag{25}$$

Boundary conditions tell us that $\forall n, \;\; (u_0^n)_x = (u_M^n)_x = 0$; hence,

$$0 \approx \frac{u_1^n - u_{-1}^n}{2h} \implies u_{-1}^n \approx u_1^n; \qquad 0 \approx \frac{u_{M+1}^n - u_{M-1}^n}{2h} \implies u_{M+1}^n \approx -u_{M-1}^n \tag{26}$$

All together it becomes a linear system of ODEs *at each time-step*, where the variables are the u-s at the $M+1$ points of the space span. As it is linear it can be written in matrix form:

$$(\mathbf{u}^n)_t \;=\; A\mathbf{u}^n \tag{27}$$

where $\mathbf{u}^n \;=\; [u_0^n, u_1^n, \ldots, u_M^n]^T$ and

$$A \;=\; \begin{bmatrix} \frac{-2\varepsilon}{h^2} & \frac{2\varepsilon}{h^2} & 0 & 0 & \cdots & 0 & 0 & 0 \\ \frac{\varepsilon}{h^2} + \frac{v}{2h} & \frac{-2\varepsilon}{h^2} & \frac{\varepsilon}{h^2} - \frac{v}{2h} & 0 & \cdots & 0 & 0 & 0 \\ 0 & \frac{\varepsilon}{h^2} + \frac{v}{2h} & \frac{-2\varepsilon}{h^2} & \frac{\varepsilon}{h^2} - \frac{v}{2h} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \frac{\varepsilon}{h^2} + \frac{v}{2h} & \frac{-2\varepsilon}{h^2} & \frac{\varepsilon}{h^2} - \frac{v}{2h} \\ 0 & 0 & 0 & 0 & \cdots & 0 & \frac{2\varepsilon}{h^2} & \frac{-2\varepsilon}{h^2} \end{bmatrix} \tag{28}$$

We denote $f(\mathbf{u}^n) := A\mathbf{u}^n$, $\forall n$ according to (27). Notice that here $f : \mathbb{R}^{M+1} \to \mathbb{R}^{M+1}$ (since it is t-independent).

The RK methods we have developed in Question 1 applied to our problem where f is defined above are:

$$\begin{aligned}
&RK(3): && RK(4): \\
&\mathbf{k}_1 = A[\mathbf{u}^n] && \tilde{\mathbf{k}}_1 = A[\mathbf{u}^n + \tau_n(a_{11}\tilde{\mathbf{k}}_1 + a_{12}\tilde{\mathbf{k}}_2)] \\
&\mathbf{k}_2 = A[\mathbf{u}^n + \tau_n\mathbf{k}_1/3] && \tilde{\mathbf{k}}_2 = A[\mathbf{u}^n + \tau_n(a_{21}\tilde{\mathbf{k}}_1 + a_{22}\tilde{\mathbf{k}}_2)] \\
&\mathbf{k}_3 = A[\mathbf{u}^n + 2\tau_n\mathbf{k}_2/3] && \tilde{\mathbf{u}}^{n+1} = \mathbf{u}^n + \tau_n(b_1\tilde{\mathbf{k}}_1 + b_2\tilde{\mathbf{k}}_2) \\
&\mathbf{u}^{n+1} = \mathbf{u}^n + \tau_n(\mathbf{k}_1 + 3\mathbf{k}_3)/4 && b_{1,2} = 1/2,\ a_{11,22} = 1/4,\ a_{12,21} = (3 \pm 2\sqrt{3})/12
\end{aligned} \tag{29}$$

where $\mathbf{k}_1$, $\mathbf{k}_2$, $\tilde{\mathbf{k}}_1$ and $\tilde{\mathbf{k}}_2$ are vectors of length $M+1$ and $\tau_n = \Delta t = t_{n+1} - t_n$.

The solution is thus given by applying the RK(3) method to (27), whereas the RK(4) method is used to perform automatic step size selection.

The implementation of RK(3) throughout the computation of $\mathbf{k}_1$, $\mathbf{k}_2$ and $\mathbf{k}_3$ is straightforward as the method is explicit, thus we will focus on the implementation of RK(4).

Since it is an implicit method we have to isolate $\tilde{\mathbf{k}}_1$ and $\tilde{\mathbf{k}}_2$ solving the system of equations:

$$\begin{aligned}
\tilde{\mathbf{k}}_1 &= A(\mathbf{u}^n + \tau_n a_{11}\tilde{\mathbf{k}}_1 + \tau_n a_{12}\tilde{\mathbf{k}}_2) \\
\tilde{\mathbf{k}}_2 &= A(\mathbf{u}^n + \tau_n a_{21}\tilde{\mathbf{k}}_1 + \tau_n a_{22}\tilde{\mathbf{k}}_2)
\end{aligned}$$

The solution at each n is given by:

$$\tilde{\mathbf{k}}_2 = ((\mathrm{Id} - a_{11}\tau_n A)(\mathrm{Id} - a_{22}\tau_n A) - a_{12}a_{21}\tau_n^2 A^2)^{-1}(a_{21}\tau_n A + (\mathrm{Id} - a_{11}\tau_n A))A\mathbf{u}^n \tag{30}$$

$$\tilde{\mathbf{k}}_1 = (\mathrm{Id} - a_{11}\tau_n A)^{-1}(A\mathbf{u}^n + \tau a_{12}\tilde{\mathbf{k}}_2) \tag{31}$$

where Id is the identity matrix of dimension $(M+1)\times(M+1)$ and the coefficients are given in (29).

At each time step n, the numerical solution using RK(4) is thus given by

$$\tilde{\mathbf{u}}^{n+1} = \mathbf{u}^n + \tau_n(\tilde{\mathbf{k}}_1 + \tilde{\mathbf{k}}_2)/2 \tag{32}$$

where $\tilde{\mathbf{k}}_1$ and $\tilde{\mathbf{k}}_2$ are given in (30)-(31) and $\mathbf{u}^0$ is the initial condition in (24).

We use this solution to compute τ_{n+1} according to (17). I.e.

$$\tau_{n+1} = \tau_n \left|\frac{\eta}{R_{n+1}}\right|^{\frac{1}{4}}, \qquad R_{n+1} = \tilde{\mathbf{u}}^{n+1} - \mathbf{u}^{n+1} = \tau_n\left(\frac{\tilde{\mathbf{k}}_1 + \tilde{\mathbf{k}}_2}{2} - \frac{\mathbf{k}_1 + 3\mathbf{k}_3}{4}\right) \tag{33}$$

where we initialize $\tau_0 = \eta$.

2.1 Algorithm

The algorithm is set up as follows:

1. Implementation of the matrix A as given by (28).

2. Implementation of RK(3) method (29) that takes as input $\mathbf{u}^n$, τ_n and outputs $\mathbf{u}^{n+1}$.

3. Implementation of RK(4) method (29) that takes as input $\mathbf{u}^n$, τ_n and, by computing $\tilde{\mathbf{k}}_1$, $\tilde{\mathbf{k}}_2$ as in (30)-(31), outputs $\tilde{\mathbf{u}}^{n+1}$.

4. Implementation of *get_next_timestep* function that takes as input the solutions from the two RK solvers $\tilde{\mathbf{u}}^{n+1}$ and $\mathbf{u}^{n+1}$, and outputs τ_{n+1} according to (33).

5. Implementation of the main while loop where the entry condition is that $t_n = \sum_{i \leq n} \tau_i \leq t_{max} = 100$. Here we use the τ_n computed at the previous iteration to compute $\mathbf{u}^{n+1}$ and $\tilde{\mathbf{u}}^{n+1}$ using both of the RK solvers. The solution given by the RK(3) method is stacked into the solution matrix. Then, the computation of the time step to be used in the next iteration is performed. At the end of the while loop we choose the last time-step size such that the ending point of the numerical scheme is t_{max}.

2.2 Numerical Results

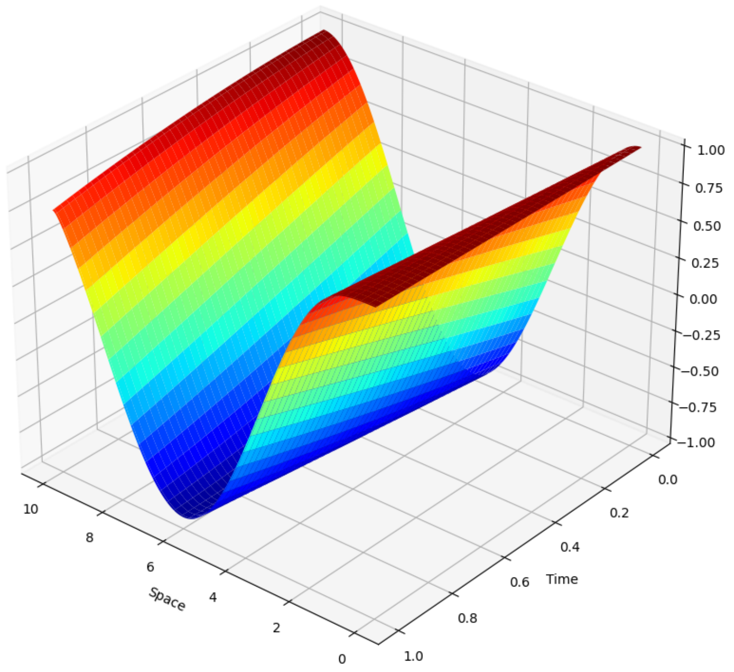

Figure 1: In the figure above the numerical solution to linear BVP given in (24). Here, $M = 400$ (i.e. $h = 0.025$) and $\eta = 10^{-5}$. From the graph we can appreciate $u_x(t, 0) = u_x(t, L) = 0$ and the propagation of the initial condition through time. (Waterfall graph is not available on Python)

3 Question 3

Consider the following nonlinear BVP

$$m_t = -mu_x - (mu)_x, \quad m = u - u_{xx}, \quad u_x(t,0) = u_x(t,L), \tag{34}$$

$$x \in [0, L], \quad t = [0, 400], \quad u(0,x) = [\exp(-5(x - L/3)^2) - \exp(-5(x - 2L/3)^2)]/10 \tag{35}$$

The space domain is linearly discretised into $M+1$ points where $x_i - x_{i-1} = \Delta x = h,\ h = L/(M+1)$ whereas the time domain is linearly discretised into $M+1$ points where $x_i - x_{i-1} = \Delta t = \tau,\ h = 400/(N+1)$.

Throughout this question we make use of the notation $u_i^n = u(t_n, x_i)$ where t_n is the n-th point of the time span and x_i is the i-th point of the space span.

The boundaries implies that the solution is periodic spatially and so we use the fact that $\forall n$:

$$u_{-i}^n = u_{M-i}^n, \qquad u_{M+i}^n = u_i^n \tag{36}$$

We approximate the derivatives in space using:

$$m^n = u^n - D^- D^+ u^n, \qquad (m^n)_t + m^n D u^n + D(m^n u^n) = 0 \tag{37}$$

where D^+ and D^- are respectively the forward and backward difference operator and $D = (D^+ + D^-)/2$.

First we consider the discretisation given by $m^n = u^n - D^- D^+ u^n$. Taking the derivative on both sides with respect to t and exploiting the periodicity of the domain, it leads to:

$$m_t = \left(I - D^- D^+\right) u_t = \begin{bmatrix} 1 + \frac{2}{h^2} & -\frac{1}{h^2} & 0 & 0 & \cdots & -\frac{1}{h^2} & 0 \\ -\frac{1}{h^2} & 1 + \frac{2}{h^2} & -\frac{1}{h^2} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & -\frac{1}{h^2} & 0 & 0 & \cdots & -\frac{1}{h^2} & 1 + \frac{2}{h^2} \end{bmatrix} u_t = A u_t \tag{38}$$

As $m_t = Au_t$ is a linear relationship, (notice that $D^- D^+$ is the second order central difference) we can write $u_t = A^{-1} m_t$. Thus, isolating u_t requires that we take the inverse of the $M+1$ by $M+1$ matrix, A. This is constant for the whole procedure and so only needs to be computed once.

Taking into account the second equation in (37), we get that for all time steps n we have

$$\begin{aligned}(m_i^n)_t = & -\left(u_i^n - \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{h^2}\right)\left(\frac{u_{i+1}^n - u_{i-1}^n}{2h}\right) - \frac{(u_{i+1}^n)^2 - (u_{i-1}^n)^2}{2h} \\ & + \frac{u_{i+1}^n(u_{i+2}^n - 2u_{i+1}^n + u_i^n) - u_{i-1}^n(u_i^n - 2u_{i-1}^n + u_{i-2}^n)}{2h^3}\end{aligned} \tag{39}$$

Denoting the above system of equations as $(\mathbf{m}^n)_t = \hat{f}(\mathbf{u}^n)$, using (38) we find that

$$(\mathbf{u}^n)_t = A^{-1}\hat{f}(\mathbf{u}^n) = f(\mathbf{u}^n), \qquad f := A^{-1}\hat{f} \tag{40}$$

To tackle this ODEs system we use a two-stage Runge-Kutta method that is diagonally implicit, i.e. $a_{12} = 0$ in the Butcher table, so that the Newton method has to be applied only once. The IRK method being used here is Cranck-Nicholson scheme, with constant step size τ, given by:

$$\mathbf{k}_1 = f(t_n, \mathbf{u}^n) \tag{41}$$

$$\mathbf{k}_2 = f(t_n + \tau, \mathbf{u}^n + \frac{1}{2}\tau(\mathbf{k}_1 + \mathbf{k}_2)) \tag{42}$$

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \frac{1}{2}\tau(\mathbf{k}_1 + \mathbf{k}_2) \tag{43}$$

This second-order method can be derived setting $b_1 = b_2 = 1/2$, $a_{11} = 1$ and $a_{21} = a_{22}$ in order conditions (7).

In order to compute $\mathbf{k}_2$, we will make use of the Newton method, that aims to solve a non linear system of equation written in the form $\mathbf{F}(\mathbf{x}) = 0$ by iterating

$$\mathbf{x}^{i+1} = \mathbf{x}^i - (\mathbf{F}'(\mathbf{x}^i))^{-1}\mathbf{F}(\mathbf{x}^i) \tag{44}$$

i-times until a stopping criterion is met, which here has been chosen as $\|\mathbf{x}^{i+1} - \mathbf{x}^i\|_2 = \|\mathbf{x}^{i+1}\|_2 < \epsilon$, where ϵ is a given tolerance. In our situation, $\mathbf{F}(\mathbf{k}_2) = \mathbf{k}_2 - f\left(\mathbf{u}^n + \tau(\mathbf{k}_1 + \mathbf{k}_2)/2\right)$.

$$\implies \quad \mathbf{F}'(\mathbf{k}_2) = \frac{\partial[\mathbf{k}_2 - f\left(\mathbf{u}^n + \tau(\mathbf{k}_1 + \mathbf{k}_2)/2\right)]}{\partial \mathbf{k}_2} = \mathrm{Id}_{M+1} - \frac{\tau}{2}A^{-1}\mathbf{G}'(\mathbf{u}^n + \tau(\mathbf{k}_1 + \mathbf{k}_2)/2) \tag{45}$$

where $\mathbf{G}' = \partial\hat{f}(\mathbf{u})/\partial\mathbf{u}$ as $f = A^{-1}\hat{f}$.

The entries of the Jacobian matrix $\mathbf{G}'$ at a generic u are:

$$(\mathbf{G}'(\mathbf{u}))_{i,j} = \mathbf{G}'_{i,j} = \frac{\partial\hat{f}(u_i)}{\partial u_j} \qquad i, j \in \{0, 1, \ldots, M\} \tag{46}$$

Given $\hat{f}$ in (39), we evaluate that:

- for $i = j \,\cap\, i \in \{0, \ldots, M\}$:

$$\mathbf{G}'_{i,i} \,=\, \partial \hat{f}(u_i)/\partial u_i \,=\, -\frac{\left(h^2+1\right)\left(u_{i+1}-u_{i-1}\right)}{2h^3} \tag{47}$$

- for $j = i+1 \,\cap\, i \in \{0, \ldots, M-1\}$:

$$\mathbf{G}'_{i,i+1} \,=\, \partial \hat{f}(u_i)/\partial u_{i+1} \,=\, \frac{-2h^2 u_{i+1} - h^2 u_i - 2u_{i+1} + u_{i+2} - u_i}{2h^3} \tag{48}$$

- for $j = i-1 \,\cap\, i \in \{1, \ldots, M\}$:

$$\mathbf{G}'_{i,i-1} \,=\, \partial \hat{f}(u_i)/\partial u_{i-1} \,=\, \frac{2h^2 u_{i-1} + h^2 u_i + 2u_{i-1} - u_{i-2} + u_i}{2h^3} \tag{49}$$

- for $j = i+2 \,\cap\, i \in \{0, \ldots, M-2\}$ and $j = i-2 \,\cap\, i \in \{2, \ldots, M+1\}$ respectively:

$$\mathbf{G}'_{i,i+2} \,=\, \partial \hat{f}(u_i)/\partial u_{i+2} \,=\, u_{i+1}/(2h^3) \qquad \mathbf{G}'_{i,i-2} \,=\, \partial \hat{f}(u_i)/\partial u_{i-2} \,=\, -u_{i-1}/(2h^3) \tag{50}$$

Exploiting the periodicity of the domain (36) we are able to fill the entries $(0, M-1)$, $(0, M-2)$, $(1, M-1)$, $(M, 1)$, $(M, 2)$, $(M-1, 1)$. All the other entries of the Jacobian are of course equal to 0. Putting (45) and (47)-(50) together we finally determine that Newton method is given, *at each time step*, by:

$$\mathbf{k}_2^{i+1} \,=\, \mathbf{k}_2^i - \left[\mathrm{Id}_{M+1} - \frac{\tau}{2} A^{-1} \mathbf{G}'\left(\mathbf{u}^n + \frac{\tau(\mathbf{k}_1 + \mathbf{k}_2^i)}{2}\right)\right]^{-1} \left[\mathbf{k}_2^i - f\left(\mathbf{u}^n + \frac{\tau(\mathbf{k}_1 + \mathbf{k}_2^i)}{2}\right)\right] \tag{51}$$

Finally, after computing $\mathbf{k}_1$ and $\mathbf{k}_2$ the solution at each time step is obtained plugging the values into (43).

3.1 Algorithm

The algorithm is implemented according to the following steps:

1. Initialization of the parameters, the A matrix according to (38) and computation of its inverse that is going to be constant throughout the problem.
2. Implementation of f function according to (40).
3. Implementation of the *G_prime* function that takes as inputs a generic vector $\mathbf{v}$ and outputs $\mathbf{G}'(v)$ according to (46)-(50).
4. Implementation of a function that performs the Newton method; i.e. taking as inputs $\mathbf{k}_1$, $\mathbf{u}^n$ and τ and using the *G_prime* function, performs a while loop until a criterion is met, and outputs (51).

5. Implementation of Runge Kutta function that takes as inputs $\mathbf{u}^n$ and using the aforementioned f function and the newton method outputs $\mathbf{u}^{n+1}$ according to (43).

6. Implementation of the main for-loop that at each time step runs the Runge-Kutta solver to get $\mathbf{u}_{n+1}$.

3.2 Numerical Results

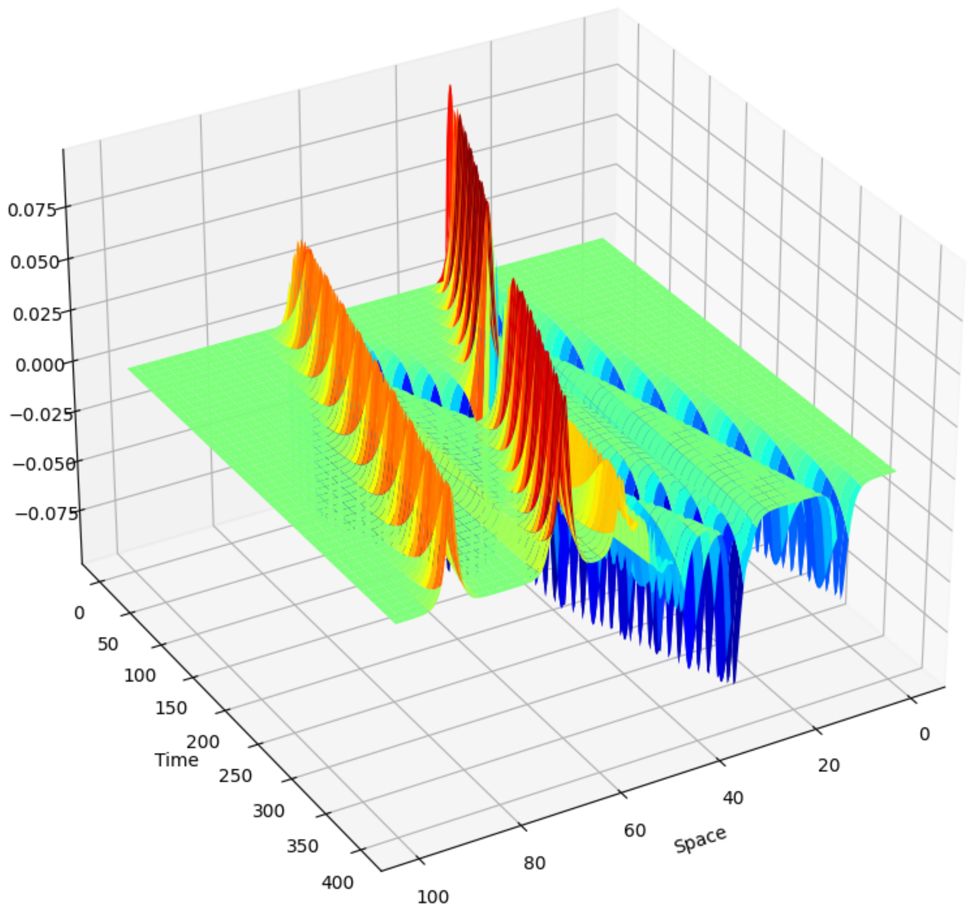

Figure 2: In the figure above the numerical solution to the Camassa-Holm PDE is shown. Here $M = 250$ and $N = 1000$, i.e. $h = \tau = 0.4$. The value for tolerance used in the Newton method is $\epsilon = 10^{-8}$. From the graph we can appreciate the travelling waves through time.