Imperial College London
Department of Mathematics
Dr Igor Shevchenko

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 01724711

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).

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

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

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

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:

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

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 Solution of Ordinary Differential Equations
Coursework 4

CID:01724711

December 2022

2

a)

The implicit two stage RK method can be written as

$$\begin{aligned} k_1 &= f(t_n + c_1 h, x_n + (a_{11}k_1 + a_{12}k_2)h) \\ k_2 &= f(t_n + c_2 h, x_n + (a_{21}k_1 + a_{22}k_2)h) \\ x_{n+1} &= x_n + h(b_1 k_1 + b_2 k_2) \end{aligned} \tag{1}$$

From lecture results we already have obtained up to third order conditions for implicit 2 stage RK methods.

$$\begin{aligned} b_1 + b_2 &= 1 \\ b_1 c_1 + b_2 c_2 &= \frac{1}{2} \\ b_1 c_1^2 + b_2 c_2^2 = \frac{1}{3}, b_1(a_{11}c_1 + a_{12}c_2) + b_2(a_{21}c_1 + a_{22}c_2) &= \frac{1}{6} \end{aligned} \tag{2}$$

Fourth order conditions are the third order conditions plus the the condition for the method taylor expansion fourth order coefficients to be equal to the taylor expansion of the true solution. We taylor expand k_1, k_2 up to $\mathcal{O}(h^4)$.

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

Then we substitute the Taylor expansions of k_1 and k_2 up to order $\mathcal{O}(h^2)$ into the second term of k_1 and k_2, and the Taylor expansions of k_1 and k_2 up to order $\mathcal{O}(h)$ into the third term of k_1 and k_2. After this we again repeat this idea until we obtain k_1, k_2 without any k_1, k_2 in them up to the third order. After substitution of k_1, k_2 into

$x_{n+1} = x_n + h(b_1k_1 + b_2k_2)$ and then we match up coefficients of each term to

$$
\begin{aligned}
x(t_{n+1}) &= x(t_n) + hx^{'} + \frac{h^2}{2}x^{''} + \frac{h^3}{6}x^{'''} + \frac{h^4}{24}x^{''''} + \mathcal{O}(h^5) \\
x^{'} &= f \\
x^{''} &= f_t + ff_x \\
x^{'''} &= f_{tt} + 2f_{xt}f + f_{xx}f^2 + f_x(ft + ff_x) \\
x^{''''} &= f_{ttx}f + f_{ttt} + 2f(f_{xtx}f + f_{xtt}) + 2f_x(f_t + f_xf) + f^2(f_{xxx}f + f_{xxt}) + 2f_{xx}f(f_t + f_xf) \\
&\quad + f_x(f_{tx}f + f_{tt}) + f_t(f_{xx}f + f_{xt}) + f_x^2(f_t + f_xf)_2ff_x(f_{xx} + f_{xt})
\end{aligned}
\tag{4}
$$

We have a large number of simplifications due to the equality rule of mixed partial derivatives. We end up with the following as the order conditions for the general 2-stage Implicit Runge–Kutta (IRK) method of order 4.

$$
\begin{aligned}
b_1 + b_2 &= 1 \\
b_1c_1 + b_2c_2 &= \frac{1}{2} \\
b_1c_1^2 + b_2c_2^2 &= \frac{1}{3} \\
b_1(a_{11}c_1 + a_{12}c_2) + b_2(a_{21}c_1 + a_{22}c_2) &= \frac{1}{6} \\
b_1c_1^3 + b_2c_2^3 &= \frac{1}{4} \\
b_1(a_{11}c_1^2 + a_{12}c_2^2) + b_2(a_{21}c_1^2 + a_{22}c_2^2) &= \frac{1}{12} \\
b_1c_1(a_{11}c_1 + a_{12}c_2) + b_2c_2(a_{21}c_1 + a22c_2) &= \frac{1}{8} \\
b_1(a_{11}^2c_1 + a_{11}a_{12}c_2 + a_{12}a_{21}c_1 + a_{12}a_{22}c_2) + b_2(a_{21}a_{11}c_1 + a_{21}a_{12}c_2 + a_{22}a_{21}c_1 + a_{22}^2c_2) &= \frac{1}{24}
\end{aligned}
\tag{5}
$$

b)

Using sympy we solve for possible IRK(4,2) coefficients using the order conditions above. We obtain 4 possible sets of coefficients.

a_{11}	a_{12}	a_{21}	a_{22}	b_1	b_2	c_1	c_2
0.183013	0.788675	0.0283122	0	0.5	0.5	0.788675	0.211325
-0.683013	0.211325	1.47169	0	0.5	0.5	0.211325	0.788675
0.461325	-0.25	-0.25	1.03868	0.5	0.5	0.788675	0.211325
1.03868	-0.25	-0.25	0.461325	0.5	0.5	0.211325	0.788675

We arbitrarily decide to choose the first set of coefficients as the ones for my IRK(4,2) method.

0.788675	0.183013	0.788675
0.211325	0.0283122	0
	0.5	0.5

Now we move onto formulating an automatic step size scheme.
Let x_1, x_2 represent approximate solutions that correspond to step size of h and $\frac{h}{2}$. Now let us see what happens when we do one step with size h versus two steps of size $\frac{h}{2}$.

$$
\begin{aligned}
\tilde{x}_{n+1} &= x_1 + Ch^{p+1} + \mathcal{O}(h^{p+2}) \\
\tilde{x}_{n+1} &= x_2 + 2C(\frac{h}{2})^{p+1} + \mathcal{O}(h^{p+2})
\end{aligned}
\tag{6}
$$

This gives

$$C = \frac{|x_1 - x_2|}{(1 - 2^{-p})h^{p+1}}$$

Note: We often use h and Δt interchangably here.
When subbing into the second estimate for the true solution we get

$$\tilde{x}_{n+1} = x_2 + \epsilon + \mathcal{O}(h^{p+2})$$

where

$$\epsilon = \frac{|x_1 - x_2|}{2^p - 1}$$

This ϵ can be used as an indication of the truncation error. So for our example of fourth order we have

$$\tilde{x}_{n+1} = x_2 + \frac{|x_1 - x_2|}{15} + \mathcal{O}(h^6)$$

where

$$|x_1 - x_2| = \frac{1}{2}hx_n^{'} + \frac{3}{8}h^2x_n^{''} + \frac{7}{48}h^3x_n^{'''} + \frac{15}{48}h^4x_n^{''''}$$

so

$$\epsilon = \frac{1}{15}(\frac{1}{2}hx_n^{'} + \frac{3}{8}h^2x_n^{''} + \frac{7}{48}h^3x_n^{'''} + \frac{15}{48}h^4x_n^{''''})$$

and

$$\epsilon_0 = \frac{1}{15}(\frac{1}{2}hx_0^{'} + \frac{3}{8}h^2x_0^{''} + \frac{7}{48}h^3x_0^{'''} + \frac{15}{48}h^4x_0^{''''})$$

Repeated differentiation of the IVP cannot be used to estimate the local truncation error, as this would negate the benefits of using LMMs. New technique is required to estimate higher order derivatives. In finding ϵ and ϵ_0 we decide to use the central difference second order formulae to approximate all the derivatives of x.

$$\begin{aligned}
x_n^{'} &= \frac{1}{2\Delta t}(x_{n+1} - x_{n-1}) + \mathcal{O}(h^2) \\
x_n^{''} &= \frac{1}{(\Delta t)^2}(x_{n+1} - 2x_n + x_{n-1}) + \mathcal{O}(h^2) \\
x_n^{'''} &= \frac{1}{2(\Delta t)^3}(x_{n+2} - 2x_{n+1} + 2x_{n-1} - x_{n-2}) + \mathcal{O}(h^2) \\
x_n^{''''} &= \frac{1}{(\Delta t)^4}(x_{n+2} - 4x_{n+1} + 6x_n - 4x_{n-1} + x_{n-2}) + \mathcal{O}(h^2)
\end{aligned} \tag{7}$$

$$\begin{aligned}
x_n^{'} &\approx \frac{1}{2\Delta t}(x_{n+1} - x_{n-1}) \\
x_n^{''} &\approx \frac{1}{(\Delta t)^2}(x_{n+1} - 2x_n + x_{n-1}) \\
x_n^{'''} &\approx \frac{1}{2(\Delta t)^3}(x_{n+2} - 2x_{n+1} + 2x_{n-1} - x_{n-2}) \\
x_n^{''''} &\approx \frac{1}{(\Delta t)^4}(x_{n+2} - 4x_{n+1} + 6x_n - 4x_{n-1} + x_{n-2})
\end{aligned} \tag{8}$$

We will use ϵ as an estimate for the step size control. We want to find the largest possible step size h_{new} at each iteration of our numerical solution so that the truncation error after the step with this step size is below a given accuracy that we desire, ϵ_0.

$$Ch_{new}^{p+1} \leq \epsilon_0 \iff (\frac{h_{new}}{h})^{p+1}\frac{|x_1 - x_2|}{1 - 2^{-p}} \leq \epsilon_0$$

So

$$h_{new} = h(\frac{\epsilon_0}{\epsilon})^{\frac{1}{p+1}}$$

The idea is that if the two answers are close to one another then the approximation is accepted but if $\epsilon > \epsilon_0$ then we decrease the step size and if $\epsilon < \epsilon_0$ then we increase the step size.
Because we are using an estimate of the local truncation error (since the real one is too complicated to have exactly due to the complexity of our method) we decide to incorporate a 'safety factor' of β into the automatic step size control. We will choose $\beta = 0.9$. So

$$h_{new} = \beta h(\frac{\epsilon_0}{\epsilon})^{\frac{1}{p+1}}$$

c)

To find the region and interval of stability we apply the method to $x^{'} = \lambda x$ giving $f(x) = \lambda x$. This gives

$$\begin{aligned} k_1 &= \lambda(x_n + h(a_{11}k_1 + a_{12}k_2)) \\ k_2 &= \lambda(x_n + h(a_{21}k_1 + a_{22}k_2)) \end{aligned} \tag{9}$$

which we solve to obtain

$$\begin{aligned} k_1 &= \frac{(a_{12} - a_{22})h\lambda^2 x_n + n}{(a_{11}a_{22} - a_{12}a_{21})h^2\lambda^2 - (a_{11} + a_{22})h\lambda + 1} \\ k_2 &= \frac{(a_{21} - a_{11})h\lambda^2 x_n + n}{(a_{11}a_{22} - a_{12}a_{21})h^2\lambda^2 - (a_{11} + a_{22})h\lambda + 1} \end{aligned} \tag{10}$$

Subbing this into the method yields

$$\begin{aligned} x_{n+1} &= x_n + h(b_1k_1 + b_2k_2) \\ &= x_n(1 + \frac{[b_1(a_{12} + a_{21}) + b_2(a_{21} - a_{11})]\hat{h}^2 + \hat{h}(b_1 + b_2)}{det(A)\hat{h}^2 - tr(A)\hat{h} + 1}) \end{aligned} \tag{11}$$

where

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

We can then form the stability polynomial

$$p(r) = r - (1 + \frac{[b_1(a_{12} + a_{21}) + b_2(a_{21} - a_{11})]\hat{h}^2 + \hat{h}(b_1 + b_2)}{det(A)\hat{h}^2 - tr(A)\hat{h} + 1})$$

Using Sympy we obtain solutions for $\hat{h}$. We apply the boundary locus method by letting $r = e^{is}, s \in [0, 2\pi)$ and plotting the $\hat{h}_1, \hat{h}_2$ values for these r.

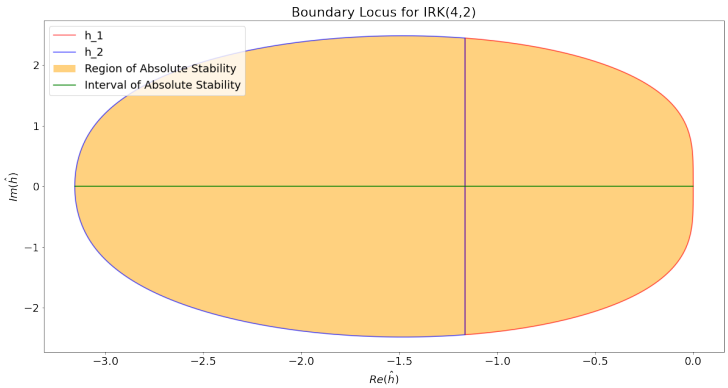


Figure 1: Plot of Boundary Locus for IRK(4,2)

To find the region and interval of absolute stability we try values of $\hat{h}$ along the real axis in each region shown on the boundary (here we try $\hat{h} \in (-4, -2, -1, 1)$) locus plugged into the solutions for r of the stability polynomial and test the three roots for the strict root condition. The Interval and Region of Absolute stability is the part corresponding to the tried $\hat{h}$ that gives r roots satisfying the strict root condition. The Interval of Absolute Stability was obtained to be

$$\hat{h} \in (-3.155, 0)$$

.

4

a)

We have

$$\begin{aligned} u_x(t_n, x_j) &\approx \frac{u(t_n, x_{j+1}) - u(t_n, x_{j-1})}{2\Delta x} \\ u_{xx}(t_n, x_j) &\approx \frac{u(t_n, x_{j+1}) - 2u(t_n, x_j) + u(t_n, x_{j-1})}{(\Delta x)^2} \end{aligned} \tag{12}$$

Using these central difference approximations we can then begin to form the Finite Difference Equation

$$\begin{bmatrix} \frac{\epsilon}{(\Delta x)^2} + \frac{v}{2\Delta x} & -\frac{2\epsilon}{(\Delta x)^2} & \frac{\epsilon}{(\Delta x)^2} - \frac{v}{2\Delta x} & 0 & & \\ 0 & \frac{\epsilon}{(\Delta x)^2} + \frac{v}{2\Delta x} & -\frac{2\epsilon}{(\Delta x)^2} & \frac{\epsilon}{(\Delta x)^2} - \frac{v}{2\Delta x} & 0 & \\ & 0 & \frac{\epsilon}{(\Delta x)^2} + \frac{v}{2\Delta x} & -\frac{2\epsilon}{(\Delta x)^2} & \frac{\epsilon}{(\Delta x)^2} - \frac{v}{2\Delta x} & \\ & \ddots & \ddots & \ddots & \ddots & 0 \\ & & 0 & \frac{\epsilon}{(\Delta x)^2} + \frac{v}{2\Delta x} & -\frac{2\epsilon}{(\Delta x)^2} & \frac{\epsilon}{(\Delta x)^2} - \frac{v}{2\Delta x} \end{bmatrix} \begin{bmatrix} u(t, x_0) \\ u(t, x_1) \\ u(t, x_2) \\ \vdots \\ u(t, x_{M-1}) \\ u(t, x_M) \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_{M-2} \\ w_{M-1} \end{bmatrix}$$

Where

$$w_i \approx u_t(t, x_i)$$

and so for given t

$$w(x) \approx u_t(t, x)$$

We now seek to use the IRK(4,2) method on this BVP. We require to find k_1, k_2 for this.

$$\begin{aligned} k_1(t, x_i) &= w(x_i + h(a_{11}k_1(t, x_i) + a_{12}k_2(t, x_i))) \\ k_2(t, x_i) &= w(x_i + h(a_{21}k_1(t, x_i) + a_{22}k_2(t, x_i))) \end{aligned} \tag{13}$$

Where

$$a_{11}, a_{12}, a_{21}, a_{22} = 0.183013, 0.788675, 0.0283122, 0$$

Since they are defined implicitly we will use an algorithm to approximately find their values for each time step. We do this using fixed point iteration simultaneously on k_1, k_2.

$$\begin{aligned} k_1^{j+1}(t, x_i) &= w(x_i + h(a_{11}k_1^j(t, x_i) + a_{12}k_2^j(t, x_i))) \\ k_2^{j+1}(t, x_i) &= w(x_i + h(a_{21}k_1^j(t, x_i) + a_{22}k_2^j(t, x_i))) \end{aligned} \tag{14}$$

However we notice that we do not have the continuous form for $w(x) \approx u_t(t, x)$ but we do have a sampling of values for $w(x)$ namely what we get in the RHS of the finite difference equation. So to implement the fixed point iteration to obtain k_1, k_2 we interpolate using w_i to get an approximate $w(x)$. Since the spacings between each x_i, x_{i+1} pair are so minute and that we do not have an inclination to the shape of $w(x)$ we proceed with linear interpolation.

$$w(x) = \frac{w_i(x_{i+1} - x) + w_{i+1}(x - x_i)}{x_{i+1} - x_i} \quad x \in (x_i, x_{i+1})$$

This means at each t we use the finite difference equation to find the w_i and then proceed with fixed point iteration to find k_1, k_2 at which point we can then use the IRK(4,2) method for each x_i

$$u(t_{n+1}, x_i) = u(t_n, x_i) + h(b_1k_1(t_n, x_i) + b_2k_2(t_n, x_i))$$

where $b_1, b_2 = \frac{1}{2}$ To initialise the IRK method we use the boundary condition

$$u(t_0, x_i) = cos(\frac{2\pi x_i}{L})$$

We apply the automatic step size control at each time step t_{n+1} from t_n. We remember from 2b) that

$$h_{new} = \beta h(\frac{\epsilon_0}{\epsilon})$$

where we used $\beta = 0.9$

$$\epsilon = \frac{1}{15}(\frac{1}{2}hu_t(t_n, x_i) + \frac{3}{8}h^2u_{tt}(t_n, x_i) + \frac{7}{48}h^3u_{ttt}(t_n, x_i) + \frac{15}{48}h^4u_{tttt}(t_n, x_i)$$

and

$$\epsilon_0 = \frac{1}{15}(\frac{1}{2}hu_t(t_0, x_i) + \frac{3}{8}h^2u_{tt}(t_0, x_i) + \frac{7}{48}h^3u_{ttt}(t_0, x_i) + \frac{15}{48}h^4u_{tttt}(t_0, x_i)$$

where

$$\begin{aligned}
u_t(t_n, x_i) &\approx \frac{1}{2h}(u(t_{n+1}, x_i) - u(t_{n-1}, x_i)) \\
u_{tt}(t_n, x_i) &\approx \frac{1}{h^2}(u(t_{n+1}, x_i) - 2u(t_n, x_i) + u(t_{n-1}, x_i)) \\
u_{ttt}(t_n, x_i) &\approx \frac{1}{2h^3}(u(t_{n+2}, x_i) - 2u(t_{n+1}, x_i) + 2u(t_{n-1}, x_i) - u(t_{n-2}, x_i)) \\
u_{tttt}(t_n, x_i) &\approx \frac{1}{h^4}(u(t_{n+2}, x_i) - 4u(t_{n+1}, x_i) + 6u(t_n, x_i) - 4u(t_{n-1}, x_i) + u(t_{n-2}, x_i))
\end{aligned} \tag{15}$$

b)

We implement the above idea and output the solution. There is no exact 'waterfall' plot in Python so we produce a 3D surface (colormap) plot of our solution to the BVP.

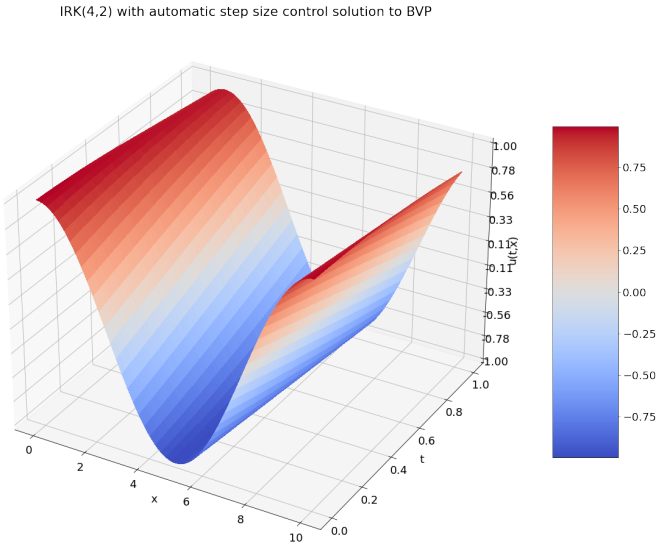


Figure 2: Plot of IRK(4,2) with automatic step size control solution to BVP

5

a)

The Camassa-Holm equation is a nonlinear integrable shallow water wave equation. It is an extension of the Korteweg-de Vries equation, which is a model for unidirectional shallow water waves in a canal or channel of constant cross-sectional area.

In the periodic domain, the Camassa-Holm equation takes the form:

$$m_t + mu_x + (mu)_x = 0,$$

where $m = u - u_{xx}$ is the "momentum" of the system, u is the height of the water above a reference level, x is the horizontal coordinate, and t is time.

To solve this equation numerically, we can use the backward Euler method, which is an implicit finite difference method that is first-order accurate in time. Backward Euler is given by

$$y(t_{n+1}, x_i) = y(t_n, x_i) + hf(t_{n+1}, x_i)$$

We know will derive the finite difference equation to which we will apply the backward Euler method.

$$Du^n = \frac{D^+ + D^-}{2}u^n = \frac{1}{2}(u^{n+1} - u^n + u^n - u^{n-1}) = \frac{1}{2}(u^{n+1} - u^{n-1})$$

$$D(m^n u^n) = \frac{1}{2}(m^{n+1}u^{n+1} - m^{n-1}u^{n-1})$$

$$D^-D^+u^n = D^-(u^{n+1} - u^n) = u^{n+1} - 2u^n + u^{n-1}$$

$$\Rightarrow m^n = -(u^{n+1} - 3u^n + u^{n-1})$$

Now apply the given space approximation in the question and using the backward Euler method we obtain

$$\begin{aligned}
&\frac{(u_{i+1}^{n+1} - 3u_i^{n+1} + u_{i-1}^{n+1}) - (u_{i+1}^n - 3u_i^n + u_{i-1}^n)}{\Delta t} + \frac{1}{2}(u_{i+1}^{n+1} - 3u_i^{n+1} + u_{i-1}^{n+1})(u_{i+1}^{n+1} - u_{i-1}^{n+1}) \\
&\qquad + \frac{1}{2}(u_{i+1}^{n+1}(u_{i+2}^{n+1} - 3u_{i+1}^{n+1} + u_i^{n+1}) - u_{i-1}^{n+1}(u_i^{n+1} - 3u_{i-1}^{n+1} + u_{i-2}^{n+1}) = 0 \\
&\frac{(u_{i+1}^{n+1} - 3u_i^{n+1} + u_{i-1}^{n+1}) - (u_{i+1}^n - 3u_i^n + u_{i-1}^n)}{\Delta t} + \frac{1}{2}u_{i+1}^{n+1}(u_{i+2}^{n+1} - 2u_{i+1}^{n+1} - 2u_i^{n+1}) \\
&\qquad + \frac{1}{2}u_{i-1}^{n+1}(2u_i^{n+1} + 2u_{i-1}^{n+1} - u_{i-2}^{n+1}) = 0
\end{aligned} \tag{16}$$

Now importantly in the formulation of $F(x) = 0$ to apply the Newton Method we have to observe that the domain is periodic and so

$$u_{-1}^{n+1} = u_{M-1}^{n+1}$$

$$u_{-2}^{n+1} = u_{M-2}^{n+1}$$

$$u_{M+1}^{n+1} = u_1^{n+1}$$

$$u_{M+2}^{n+1} = u_2^{n+1}$$

$$u_0^{n+1} = u_M^{n+1}$$

where M is the index of the last grid point in the space direction. This allows us to have M equations of (16) for $i = 1, \ldots, M$ in M unknowns. This will give us a Jacobian $M \times M$

matrix.

$$F\begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_{M-2}^{n+1} \\ u_{M-1}^{n+1} \\ u_M^{n+1} \end{pmatrix} = \begin{bmatrix} \frac{(u_2^{n+1}-3u_1^{n+1}+u_M^{n+1})-(u_2^n-3u_1^n+u_M^n)}{\Delta t} & +\frac{1}{2}u_2^{n+1}(u_3^{n+1}-2u_2^{n+1}-2u_1^{n+1}) \\ & +\frac{1}{2}u_M^{n+1}(2u_1^{n+1}+2u_M^{n+1}-u_{M-1}^{n+1}) \\ \frac{(u_3^{n+1}-3u_2^{n+1}+u_1^{n+1})-(u_3^n-3u_2^n+u_1^n)}{\Delta t} & +\frac{1}{2}u_3^{n+1}(u_4^{n+1}-2u_3^{n+1}-2u_2^{n+1}) \\ & +\frac{1}{2}u_1^{n+1}(2u_2^{n+1}+2u_1^{n+1}-u_M^{n+1}) \\ \vdots & \\ \frac{(u_M^{n+1}-3u_{M-1}^{n+1}+u_{M-2}^{n+1})-(u_M^n-3u_{M-1}^n+u_{M-2}^n)}{\Delta t} & +\frac{1}{2}u_M^{n+1}(u_1^{n+1}-2u_M^{n+1}-2u_{M-1}^{n+1}) \\ & +\frac{1}{2}u_{M-2}^{n+1}(2u_{M-1}^{n+1}+2u_{M-2}^{n+1}-u_{M-3}^{n+1}) \\ \frac{(u_1^{n+1}-3u_M^{n+1}+u_{M-1}^{n+1})-(u_1^n-3u_M^n+u_{M-1}^n)}{\Delta t} & +\frac{1}{2}u_1^{n+1}(u_2^{n+1}-2u_1^{n+1}-2u_M^{n+1}) \\ & +\frac{1}{2}u_{M-1}^{n+1}(2u_M^{n+1}+2u_{M-1}^{n+1}-u_{M-2}^{n+1}) \end{bmatrix} = 0$$

Now we are able to find the Jacobian by differentiating with respect to u_i^{n+1} for $i = 1, \ldots, M$. We notice that due to the structure present in F, having only u_j terms in F_i for $j = -1, 2, 0, 1, 2$ we will obtain a wrapped quintdiagonal matrix for the Jacobian.

$$\begin{aligned} A_i &= \frac{\partial}{\partial u_{i-2}^{n+1}} = -\frac{1}{2}u_{i-1}^{n+1} \\ B_i &= \frac{\partial}{\partial u_{i-1}^{n+1}} = \frac{1}{\Delta t} + \frac{1}{2}(2u_i^{n+1} + 4u_{i-1}^{n+1} - u_{i-2}^{n+1}) \\ C_i &= \frac{\partial}{\partial u_i^{n+1}} = -\frac{3}{\Delta t} - u_{i+1}^{n+1} + u_{i-1}^{n+1} \\ D_i &= \frac{\partial}{\partial u_{i+1}^{n+1}} = \frac{1}{\Delta t} + \frac{1}{2}(u_{i+2}^{n+1} - 4u_{i+1}^{n+1} - 2u_i^{n+1}) \\ E_i &= \frac{\partial}{\partial u_{i+2}^{n+1}} = \frac{1}{2}u_{i+1}^{n+1} \\ \frac{\partial}{\partial u_j^{n+1}} &= 0 \quad j \neq -2, -1, 0, 1, 2 \end{aligned} \tag{17}$$

$$F'\begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_{M-2}^{n+1} \\ u_{M-1}^{n+1} \\ u_M^{n+1} \end{pmatrix} = \begin{pmatrix} C_0 & D_0 & E_0 & 0 & 0 & \ldots & 0 & A_0 & B_0 \\ B_1 & C_1 & D_1 & E_1 & 0 & \ldots & 0 & 0 & A_1 \\ A_2 & B_2 & C_2 & D_2 & E_2 & \ddots & 0 & 0 & 0 \\ 0 & A_3 & B_3 & C_3 & D_3 & \ddots & & & \\ & \ddots & \ddots & \ddots & \ddots & & & & \\ 0 & 0 & \ldots & & A_{M-2} & B_{M-2} & C_{M-2} & D_{M-2} & E_{M-2} \\ E_{M-1} & 0 & 0 & \ldots & & A_{M-1} & B_{M-1} & C_{M-1} & D_{M-1} \\ D_M & E_M & 0 & 0 & \ldots & & A_M & B_M & C_M \end{pmatrix}$$

So at each time step we apply the Newton Method to find the u values at each x_i.

$$\begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_{M-2}^{n+1} \\ u_{M-1}^{n+1} \\ u_M^{n+1} \end{pmatrix}^{s+1} = \begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_{M-2}^{n+1} \\ u_{M-1}^{n+1} \\ u_M^{n+1} \end{pmatrix}^{s} - inv(F'\begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_{M-2}^{n+1} \\ u_{M-1}^{n+1} \\ u_M^{n+1} \end{pmatrix}^{s})F\begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_{M-2}^{n+1} \\ u_{M-1}^{n+1} \\ u_M^{n+1} \end{pmatrix}^{s}$$

until either

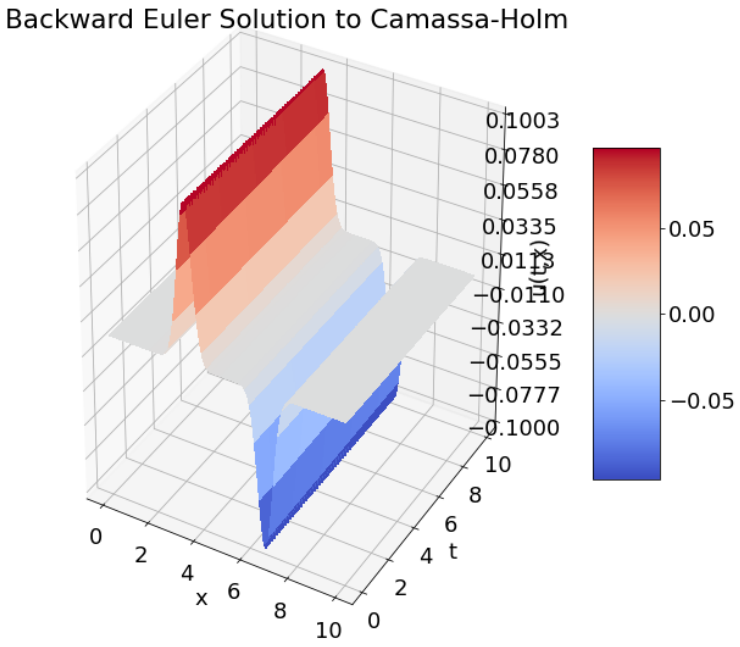
$$|| \begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_{M-2}^{n+1} \\ u_{M-1}^{n+1} \\ u_M^{n+1} \end{pmatrix}^{s+1} - \begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_{M-2}^{n+1} \\ u_{M-1}^{n+1} \\ u_M^{n+1} \end{pmatrix}^{s} ||_2 \div || \begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_{M-2}^{n+1} \\ u_{M-1}^{n+1} \\ u_M^{n+1} \end{pmatrix}^{s+1} ||_2 < \epsilon$$

where ϵ is the given stopping criteria tolerance or we hit maximum number of iterations. We will initialise the Newton Method with the solution for the Newton Method from the previous time step.
We are able to start the process to solve for u_i^1 at all spaces since we know the initial conditions for

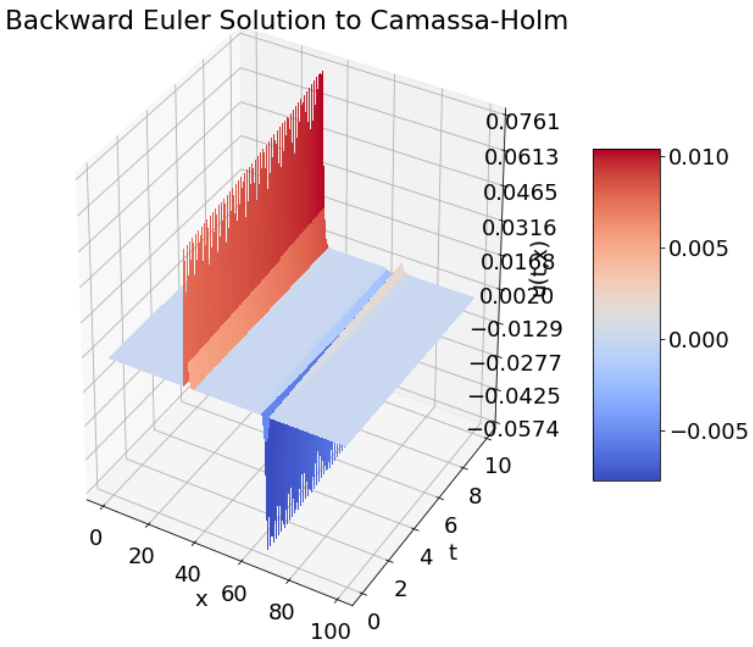
$$u_i^0 = \frac{1}{10}e^{-5(x_i - \frac{L}{3})^2} - \frac{1}{10}e^{-5(x_i - \frac{2L}{3})^2}$$

b)

Note: Due to computational limitations of available hardware we only implement the methods up to $t \in (0, 10)$
We implement the method above with stopping tolerance of 0.01 in the Newton Method and using $\Delta t = \Delta x = 0.1$. We showcase the solution by producing a 3D surface (colormap) plot of the BVP. First we plot for $L = 10$. Then we plot for $L = 100$. We observe that



as L increases we obtain smaller and less wide peaks. The Camassa–Holm equation has peakon solutions: solitons with a sharp peak, so with a discontinuity at the peak in the wave slope. As L increases this becomes more prolific.


Backward Euler Solution to Camassa-Holm
0.0761
0.0613
0.0465
0.0316
0.0168
0.0020
−0.0129
−0.0277
−0.0425
−0.0574
u(x,t)
0.010
0.005
0.000
−0.005
0
20
40
60
80
100
x
0
2
4
6
8
10
t