Assignment 2

02689 — Advanced Numerical Methods for Differential Equations

- Jeppe Klitgaard <s250250@dtu.dk>
- Tymoteusz Barcinski <s221937@dtu.dk>
- Pernille Christie <s204249@dtu.dk>

1) Boundary Value Problems

1.a) Legendre Methods

We are given the second-order boundary value problem on the domain $x \in [a, b]$ where $a = 0, b = 1$:

$$-\epsilon \frac{\mathrm{d}^2}{\mathrm{d}x^2} u - \frac{\mathrm{d}}{\mathrm{d}x} u = 1 \qquad u(a) = u(b) = 0 \tag{1}$$

Where $\epsilon > 0$ is a small parameter characterising the boundary layer width. We are given an exact solution to Equation 1:

$$u(x) = \frac{e^{-\frac{x}{\epsilon}} + (x - 1) - e^{-1/\varepsilon} x}{e^{-1/\epsilon} - 1} \tag{2}$$

1.a.i) Confirmation of Exact Solution

We confirm that Equation 2 is a solution to Equation 1 by computing the derivatives:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}x} u &= \frac{1}{e^{-1/\epsilon} - 1} \left[-\frac{1}{\epsilon} e^{-x/\epsilon} + 1 - e^{-1/\epsilon} \right] \\ \frac{\mathrm{d}^2}{\mathrm{d}x^2} u &= \frac{1}{e^{-1/\epsilon} - 1} \left[\frac{1}{\epsilon^2} e^{-x/\epsilon} \right] \end{aligned} \tag{3}$$

Plugging this into Equation 1:

$$-\epsilon \frac{\mathrm{d}^2}{\mathrm{d}x^2} u - \frac{\mathrm{d}}{\mathrm{d}x} u = \frac{1}{e^{-1/\epsilon} - 1} \left[-\epsilon \frac{1}{\epsilon^2} e^{-x/\epsilon} + \frac{1}{\epsilon} e^{-x/\epsilon} - 1 + e^{-1/\epsilon} \right] = 1 \tag{4}$$

We observe that the proposed solution indeed satisfies Equation 1.

1.a.ii) Coordinate Transformation

The implementation of of numerical schemes such as the Spectral Legendre Tau Method (LTM) and the Legendre Collocation Method (LCM) are typically undertaken in the canonical domain $z \in [-1, 1]$, which differs from our problem domain $x \in [0, 1]$. This can be remedied by either performing a coordination transformation of *differential operator* of the problem into the canonical domain, or alternatively by transforming the *differentiation matrix* of the method.

In either case, we consider the coordinate transformation at hand, with associated differentials:

$$x = \frac{b - a}{2} z + \frac{a + b}{2}, \quad \mathrm{d}x = \frac{b - a}{2} \, \mathrm{d}z \tag{5}$$

From which we can compute the change of basis using the chain rule:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}x} &= \frac{\mathrm{d}z}{\mathrm{d}x} \frac{\mathrm{d}}{\mathrm{d}z} = \frac{2}{b - a} \frac{\mathrm{d}}{\mathrm{d}z} \\ \frac{\mathrm{d}^2}{\mathrm{d}x^2} &= \frac{\mathrm{d}^2 z}{\mathrm{d}x^2} \frac{\mathrm{d}}{\mathrm{d}z} + \left(\frac{\mathrm{d}z}{\mathrm{d}x} \right)^2 \frac{\mathrm{d}^2}{\mathrm{d}z^2} = \frac{4}{(b - a)^2} \frac{\mathrm{d}^2}{\mathrm{d}z^2} \end{aligned} \tag{6}$$

Opting to transform our problem into the canonical domain results in the following modification of Equation 1 for problem domain $x \in [a, b] = [0, 1]$:

$$-4\epsilon\frac{\mathrm{d}^2}{\mathrm{d}z^2}u(z) - 2\frac{\mathrm{d}}{\mathrm{d}z}u(z) = 1 \qquad u(-1) = u(1) = 0 \tag{7}$$

With associated analytical solution based on Equation 2:

$$u(z) = \frac{e^{-\frac{z+1}{2\epsilon}} + \frac{z-1}{2} - e^{-1/\epsilon}\frac{z+1}{2}}{e^{-1/\epsilon} - 1} \tag{8}$$

1.a.iii) Spectral Legendre Tau Method (LTM)

The *Spectral Legendre Tau Method* may be formulated as a *Method of Weighted Residuals* (MWR) wherein the residual of the Nth order approximation of a function $u(z)$ is given by:

$$\mathcal{R}_N(z) = u(z) - u_N(z) = \mathcal{L}u_N(z) - f(z) \tag{9}$$

Where $\mathcal{L}$ and $f(z)$ are the *differential operator* and *right-hand side function* respectively, while $u_N(z)$ is the Nth order approximation as given by:

$$u_N(z) = \sum_{n=0}^{N} \hat{u}_n \phi_n(z) \tag{10}$$

With $\{\phi_n(x)\}_{n=0}^{N}$ being orthogonal basis functions defined by their inner product and weight $w(z)$:

$$\left(\phi_i, \phi_j\right)_w = \gamma_i \delta_{ij} \tag{11}$$

The MWR family of methods are characterised by the projection of the residual $\mathcal{R}_N$ onto a set of *test functions* $\{\psi_n\}_{n=0}^{N}$ vanishing, where the the choice of test functions determine whether the method is the Collocation Method, Tau Method, or some other method.

For the Tau Method, the test functions are given by the basis functions themselves, such that $\psi_n(z) = \phi_n(z)$, which by the MWR condition leads to:

$$\left(\mathcal{R}_N, \phi_i\right)_w = \int_a^b \left(u - \sum_{n=0}^{N} \hat{u}_n \phi_n\right) \phi_i w \,\mathrm{d}x = 0 \qquad i = 0, 1, ..., N \tag{12}$$

And orthogonality leads us to an expression for the expansion coefficients $\hat{u}_n$:

$$\hat{u}_n = \frac{\left(u, \phi_n\right)_w}{\left(\phi_n, \phi_n\right)_w} \qquad n = 0, 1, ..., N \tag{13}$$

However, unlike the *Galerkin Method*, the Tau Method relaxes the requirement that the basis functions $\{\phi_n(x)\}_{n=0}^{N}$ satisfy the boundary conditions. As such, we must *spend* two of the $N+1$ independent equations we were granted by Equation 10 to abide by the boundary conditions, leaving us with $N-1$ remaining equations to constrain using the projection of the residual onto the basis functions:

$$\sum_{n=0}^{N} \hat{u}_n \left(\mathcal{L}\phi_n, \phi_i\right)_w = \left(f, \phi_i\right)_w = \hat{f}_i \qquad i = 0, 1, ..., N-2 \tag{14}$$

Where the number of indices i is determined by the remaining number of equations.

The boundary conditions are imposed using the remaining two equations:

$$\mathcal{B}_- u_N(\alpha) = g_- \qquad \mathcal{B}_+ u_N(\beta) = g_+ \tag{15}$$

Where $\mathcal{B}_\pm$ are the differential operators associated with the conditions (most often *Dirichlet* or *Neumann*) while α, β are the boundary points and $g_\pm$ are the boundary values.

From this, we may construct a linear system of equations, $\mathcal{L}_N$:

$$\mathcal{L}_N \hat{\boldsymbol{u}} = \hat{\boldsymbol{f}} \tag{16}$$

Which gives modal solution:

$$\begin{aligned} \hat{\boldsymbol{u}} &= (\mathcal{L}_N)^{-1}\hat{\boldsymbol{f}} \\ u_N(x) &= \sum_{n=0}^{N} \hat{u}_n \phi_n(x) \end{aligned} \tag{17}$$

In order to find the discretisation given by Equation 16, we may obtain a sparse matrix A by finding a recurrence relation dictating the coefficients of the algebraic equations.

For this exercise, we use the Legendre basis functions, i.e.

$$u(z) = \sum_{n=0}^{\infty} \hat{u}_n L_n(z) \qquad \frac{\mathrm{d}^q}{\mathrm{d}z^q} u(z) = \sum_{n=0}^{\infty} \hat{u}_n^{(q)} L_{n(z)}, \tag{18}$$

where the coefficients for the derivatives are given by the recurrence

$$\hat{u}_n^{(q-1)} = \frac{\hat{u}_{n-1}^{(q)}}{2n-1} - \frac{\hat{u}_{n+1}^{(q)}}{2n+3}, \quad n \geq 1. \tag{19}$$

Furthermore, the derivative coefficients are related to the solution coefficients through

$$\hat{u}_n^{(1)} = (2n+1) \sum_{\substack{p=n+1 \\ n+p \text{ odd}}}^{\infty} \hat{u}_p, \qquad \hat{u}_n^{(2)} = \left(n + \frac{1}{2}\right) \sum_{\substack{p=n+2 \\ n+p \text{ even}}}^{\infty} (p(p+1) - n(n+1))\hat{u}_p, \quad n \geq 0. \tag{20}$$

Using the recurrence, we can build the system matrix A.

Consider a general second-order equation and its rewritten version,

$$a\hat{u}_n^{(2)} + b\hat{u}_n^{(1)} + c\hat{u}_n = \hat{f}_n, \quad \hat{u}_n^{(2)} = -\frac{b}{a}\hat{u}_n^{(1)} - \frac{c}{a}\hat{u}_n + \frac{1}{a}\hat{f}_n. \tag{21}$$

Using the recursion for derivatives of Legendre-polynomials, Equation 19, we get the following for $q = 2$ and $q = 1$

$$\hat{u}_n^{(1)} = \frac{1}{2n-1}\hat{u}_{n-1}^{(2)} - \frac{1}{2n+3}\hat{u}_{n+1}^{(2)}, \quad \hat{u}_n = \frac{1}{2n-1}\hat{u}_{n-1}^{(1)} - \frac{1}{2n+3}\hat{u}_{n+1}^{(1)}. \tag{22}$$

We plug this into Equation 21

$$\hat{u}_n^{(1)} = \frac{1}{2n-1}\left(-\frac{b}{a}\hat{u}_{n-1}^{(1)} - \frac{c}{a}\hat{u}_{n-1} + \frac{1}{a}\hat{f}_{n-1}\right) - \frac{1}{2n+3}\left(-\frac{b}{a}\hat{u}_{n+1}^{(1)} - \frac{c}{a}\hat{u}_{n+1} + \frac{1}{a}\hat{f}_{n+1}\right), \tag{23}$$

and rearrange the terms,

$$\begin{aligned} \hat{u}_n^{(1)} = &-\frac{b}{a}\left(\frac{1}{2n-1}\hat{u}_{n-1}^{(1)} - \frac{1}{2n+3}\hat{u}_{n+1}^{(1)}\right) - \frac{c}{a}\left(\frac{1}{2n-1}\hat{u}_{n-1} - \frac{1}{2n+3}\hat{u}_{n+1}\right) \\ &+\frac{1}{a}\left(\frac{1}{2n-1}\hat{f}_{n-1} - \frac{1}{2n+3}\hat{f}_{n+1}\right) \end{aligned} \tag{24}$$

We can now use the recursion for $q = 1$ to simplify,

$$\hat{u}_n^{(1)} = -\frac{b}{a}\hat{u}_n - \frac{c}{a}\left(\frac{1}{2n-1}\hat{u}_{n-1} - \frac{1}{2n+3}\hat{u}_{n+1}\right) + \frac{1}{a}\left(\frac{1}{2n-1}\hat{f}_{n-1} - \frac{1}{2n+3}\hat{f}_{n+1}\right). \tag{25}$$

Consider now

$$\begin{aligned}\frac{1}{2n-1}\hat{u}_{n-1}^{(1)} &= \frac{1}{2n-1}\left[-\frac{b}{a}\hat{u}_{n-1} - \frac{c}{a}\left(\frac{1}{2n-3}\hat{u}_{n-2} - \frac{1}{2n+1}\hat{u}_n\right) + \frac{1}{a}\left(\frac{1}{2n-3}\hat{f}_{n-2} - \frac{1}{2n+1}\hat{f}_n\right)\right] \\ &= \left(-\frac{c}{a}\frac{1}{2n-1}\frac{1}{2n-3}\right)\hat{u}_{n-2} + \left(-\frac{b}{a}\frac{1}{2n-1}\right)\hat{u}_{n-1} + \left(\frac{c}{a}\frac{1}{2n-1}\frac{1}{2n+1}\right)\hat{u}_n \\ &\quad + \left(\frac{1}{a}\frac{1}{2n-1}\frac{1}{2n-3}\right)\hat{f}_{n-2} + \left(-\frac{1}{a}\frac{1}{2n-1}\frac{1}{2n+1}\right)\hat{f}_n,\end{aligned} \tag{26}$$

and likewise

$$\begin{aligned}\frac{1}{2n+3}\hat{u}_{n+1}^{(1)} &= \frac{1}{2n+3}\left(-\frac{b}{a}\hat{u}_{n+1} - \frac{c}{a}\left(\frac{1}{2n+1}\hat{u}_n - \frac{1}{2n+5}\hat{u}_{n+2}\right) + \frac{1}{a}\left(\frac{1}{2n+1}\hat{f}_n - \frac{1}{2n+5}\hat{f}_{n+2}\right)\right) \\ &= \left(-\frac{c}{a}\frac{1}{2n+3}\frac{1}{2n+1}\right)\hat{u}_n + \left(-\frac{b}{a}\frac{1}{2n+3}\right)\hat{u}_{n+1} + \left(\frac{c}{a}\frac{1}{2n+3}\frac{1}{2n+5}\right)\hat{u}_{n+2} \\ &\quad + \left(\frac{1}{a}\frac{1}{2n+3}\frac{1}{2n+1}\right)\hat{f}_n + \left(-\frac{1}{a}\frac{1}{2n+3}\frac{1}{2n+5}\right)\hat{f}_{n+2}.\end{aligned} \tag{27}$$

We can now use the recursion for $q = 1$ once again,

$$\frac{1}{2n-1}\hat{u}_{n-1}^{(1)} - \frac{1}{2n+3}\hat{u}_{n+1}^{(1)} - \hat{u}_n = 0 \tag{28}$$

Using Equation 26 and Equation 27 in Equation 28, we get an equation on the form

$$a_{n,n-2}\hat{u}_{n-2} + a_{n,n-1}\hat{u}_{n-1} + a_{n,n}\hat{u}_n + a_{n,n+1}\hat{u}_{n+1} + a_{n,n+2}\hat{u}_{n+2} = g_{n-2}\hat{f}_{n-2} + g_n\hat{f}_n + g_{n+2}\hat{f}_{n+2}. \tag{29}$$

Clearly, there are overlapping terms for $\hat{u}_n$ and $\hat{f}_n$. For the $\hat{u}_n$ terms, we get

$$\begin{aligned}a_{n,n}\hat{u}_n &= \left(\frac{c}{a}\frac{1}{2n-1}\frac{1}{2n+1}\right)\hat{u}_n - \left(-\frac{c}{a}\frac{1}{2n+3}\frac{1}{2n+1}\right)\hat{u}_n - \hat{u}_n \\ &= \left(\frac{c}{a}\frac{1}{2n-1}\frac{1}{2n+1} + \frac{c}{a}\frac{1}{2n+3}\frac{1}{2n+1} - 1\right)\hat{u}_n \\ &= \left(\frac{c}{a}\frac{2}{(2n-1)(2n+3)} - 1\right)\hat{u}_n,\end{aligned} \tag{30}$$

and for the $\hat{f}_n$ terms,

$$\begin{aligned}g_n\hat{f}_n &= \left(-\frac{1}{a}\frac{1}{2n-1}\frac{1}{2n+1}\right)\hat{f}_n - \left(\frac{1}{a}\frac{1}{2n+3}\frac{1}{2n+1}\right)\hat{f}_n \\ &= -\left(\frac{1}{a}\frac{1}{2n-1}\frac{1}{2n+1} + \frac{1}{a}\frac{1}{2n+3}\frac{1}{2n+1}\right)\hat{f}_n \\ &= -\frac{1}{a}\frac{2}{(2n-1)(2n+3)}\hat{f}_n.\end{aligned} \tag{31}$$

Thus, the coefficients for the linear system are the following

$$
\begin{aligned}
a_{n,n-2} &= -\frac{c}{a}\frac{1}{2n-1}\frac{1}{2n-3}, \\
a_{n,n-1} &= -\frac{b}{a}\frac{1}{2n-1}, \\
a_{n,n} &= \frac{c}{a}\frac{2}{(2n-1)(2n+3)} - 1, \\
a_{n,n+1} &= \frac{b}{a}\frac{1}{2n+3}, \\
a_{n,n+2} &= -\frac{c}{a}\frac{1}{2n+3}\frac{1}{2n+5}, \\
g_{n-2} &= -\frac{1}{a}\frac{1}{2n-1}\frac{1}{2n-3}, \\
g_{n} &= \frac{1}{a}\frac{2}{(2n-1)(2n+3)}, \\
g_{n+2} &= -\frac{1}{a}\frac{1}{2n+3}\frac{1}{2n+5}.
\end{aligned}
\tag{32}
$$

These coefficients are of course only valid for $2 \leq n < N-2$, and we need N equations to solve the system uniquely. To get two more equations, we utilize Equation 20. We are of course not able to calculate the infinite series, so we use a truncated version for $n = 0, 1$,

$$
\begin{aligned}
\frac{1}{2}a \sum_{\substack{p=2 \\ p \text{ even}}}^{N-1} p(p+1)\hat{u}_p + b \sum_{\substack{p=1 \\ p \text{ odd}}}^{N-1} \hat{u}_p + c\hat{u}_0 &= \hat{f}_0, \\
\frac{3}{2}a \sum_{\substack{p=3 \\ p+1 \text{ even}}}^{N-1} (p(p+1)-2)\hat{u}_p + 3b \sum_{\substack{p=2 \\ p+1 \text{ odd}}}^{N-1} \hat{u}_p + c\hat{u}_1 &= \hat{f}_1.
\end{aligned}
\tag{33}
$$

For the remaining two needed equations, we impose the boundary conditions, namely

$$
\begin{aligned}
\sum_{n=0}^{N-1} \hat{u}_n L_n(-1) &= 0, \\
\sum_{n=0}^{N-1} \hat{u}_n L_n(1) &= 0.
\end{aligned}
\tag{34}
$$

With these equations, we set up a new system, $\boldsymbol{\mathcal{A}}\hat{\boldsymbol{u}} = \hat{\boldsymbol{g}}$, where the coefficients are given by Equation 32 for $2 \leq n < N-2$ and

$$
\begin{aligned}
a_{0,n} &= \begin{cases} c & , \ n = 0 \\ \frac{1}{2}an(n+1) & , \ n > 0 \text{ and even} \\ b & , \ n > 0 \text{ and odd} \end{cases} \\
a_{1,n} &= \begin{cases} 0 & , \ n = 0 \\ c & , \ n = 1 \\ \frac{3}{2}a(n(n+1)-2) & , \ n > 1 \text{ and odd} \\ b & , \ n > 1 \text{ and even} \end{cases} \\
a_{N-2,n} &= L_n(-1) \\
a_{N-1,n} &= L_n(1) \\
\hat{g}_0 &= \hat{f}_0 \\
\hat{g}_1 &= \hat{f}_1 \\
\hat{g}_{N-2} &= 0 \\
\hat{g}_{N-1} &= 0
\end{aligned}
\tag{35}
$$

1.a.iv) Legendre Collocation Method (LCM)

The Legendre Collocation Method is somewhat simpler to implement and derive.

As it is a collocation method, we employ the *Dirac Delta* function as our test functions rather than the basis functions themselves as was done for the Tau Method:

$$
\psi_i(x) = \delta(x - x_i) \tag{36}
$$

By the MWR condition, we require the residual $\mathcal{R}_N$ to disappear at every nodal point x_i:

$$
\begin{aligned}
\left(\mathcal{R}_N, \delta(x - x_i)\right)_w &= \int_a^b \mathcal{R}_N(x)\delta(x - x_i)w(x)\,\mathrm{d}x = 0 \\
&= \mathcal{R}_N(x_i) = \sum_{n=0}^{N} \hat{u}_n\phi_n(x_i) - u(x_i)
\end{aligned}
\tag{37}
$$

Where we have used $w(x) = (1-x)^\alpha(1+x)^\beta = 1$ for the Legendre polynomials characterised by $\alpha = \beta = 0$. This yields the collocation condition:

$$
u(x_i) = \sum_{n=0}^{N} \hat{u}_n\phi_n(x_i) \qquad i = 0, 1, ..., N \tag{38}
$$

Where we may determine the expansion coefficients $\{\hat{u}_n\}_{n=0}^N$ by orthogonality:

$$
\hat{u}_N = \frac{\left(u, \phi_n\right)_{w,N}}{\left(\phi_n, \phi_n\right)_{w,N}} = \frac{1}{\gamma_n}\int_a^b u_N(x)\phi_n(x)\,\mathrm{d}x \qquad n = 0, 1, ..., N \tag{39}
$$

Where the integral will usually be computed using a quadrature. If we are free to pick our collocation points, x_i, and quadrature rule, we may pick the Legendre-Gauss-Lobatto points and a Jacobi Gauss-type quadrature, which is exact up to a polynomial degree of $2N + 1$ [1, p. 33].

Recalling the construction of the *Vandermonde matrix*, $\mathcal{V}$ from the first assignment, we find that it corresponds exactly to the conversion between the nodal and modal bases outlined above, thus allowing the expression of Equation 39 as:

$$
u_N = \mathcal{V}\hat{u}_N \tag{40}
$$

Where:

$$\mathcal{V} = \begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) & \dots & \phi_N(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \dots & \phi_N(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(x_N) & \phi_2(x_N) & \dots & \phi_N(x_N) \end{bmatrix} \tag{41}$$

This in turn enables the construction of the *differentiation matrix* with respect to the canonical coordinate z, $\mathcal{D}_z$:

$$\mathcal{D}_z = \mathcal{V}_z \mathcal{V}^{-1} \tag{42}$$

Where $\mathcal{V}_z$ is constructed in similar fashion to $\mathcal{V}$, though using the derivatives of the basis functions with respect to z.

Using the differentiation matrix $\mathcal{D}_z$ we are able to construct a discrete operator for the differential equation given in Equation 7:

$$\mathcal{L}_N u(z) = -4\epsilon \mathcal{D}_z^2 u(z) - 2\mathcal{D}_z u(z) = f(z) = 1 \tag{43}$$

Lastly, the boundary conditions are imposed by replacing the first and last rows of the operator $\mathcal{L}_N$, corresponding to the boundary points $z = -1$ and $z = 1$, with a mask that selects the corresponding element of u_N while updating the right-hand side to the boundary values accordingly.

1.a.v) Results

To check out solvers, we first try to solve it and compare it to the exact solution. The plots for the Legendre Tau Method can be seen in Figure 1, Figure 2 and Figure 3.

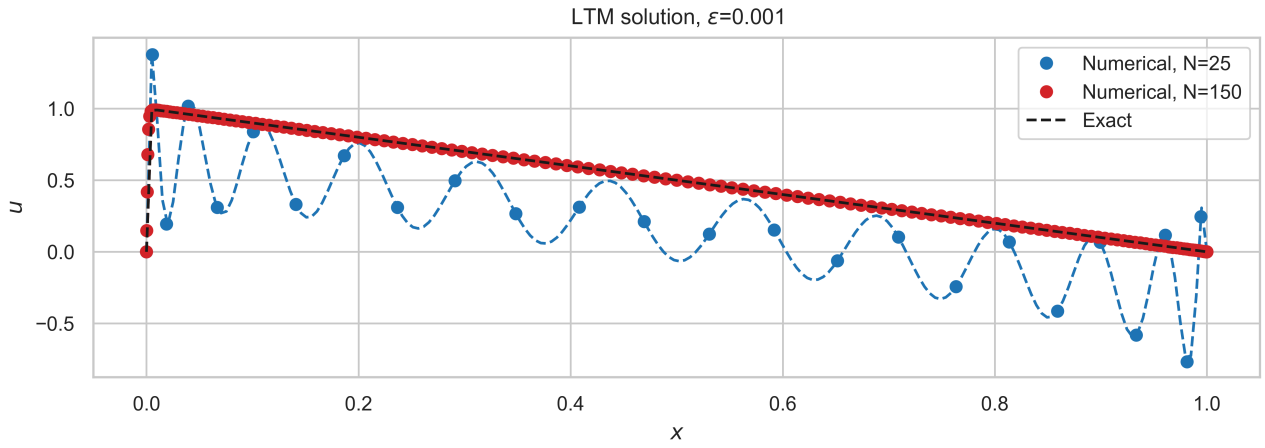

Figure 1: Computed solutions using LTM for different values of N to the problem with $\varepsilon = 0.001$

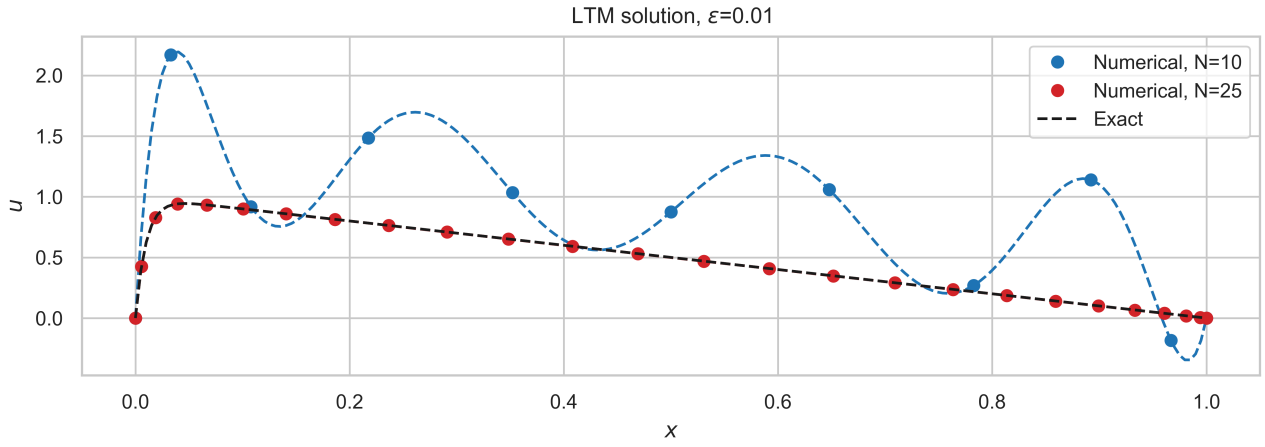

Figure 2: Computed solutions using LTM for different values of N to the problem with $\varepsilon = 0.01$

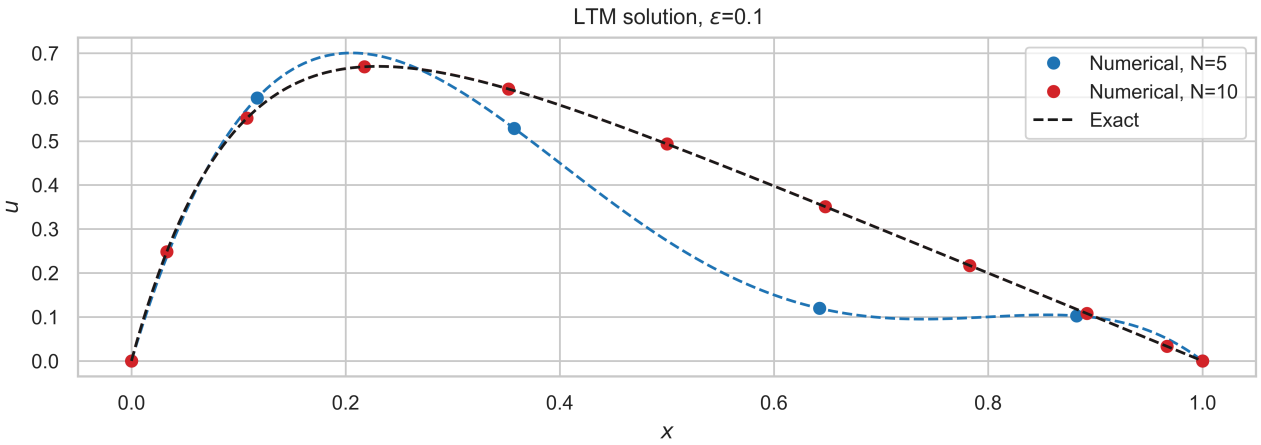

Figure 3: Computed solutions using LTM for different values of N to the problem with $\varepsilon = 0.1$

Likewise, he plots for the Legendre Collocation Method can be seen in Figure 4, Figure 5 and Figure 6.

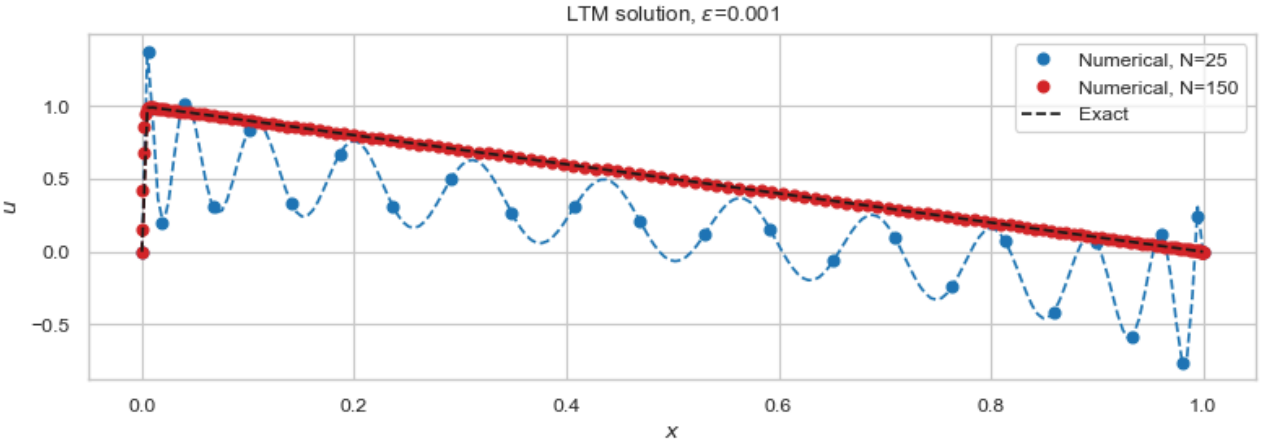

Figure 4: Computed solutions using LCM for different values of N to the problem with $\varepsilon = 0.001$

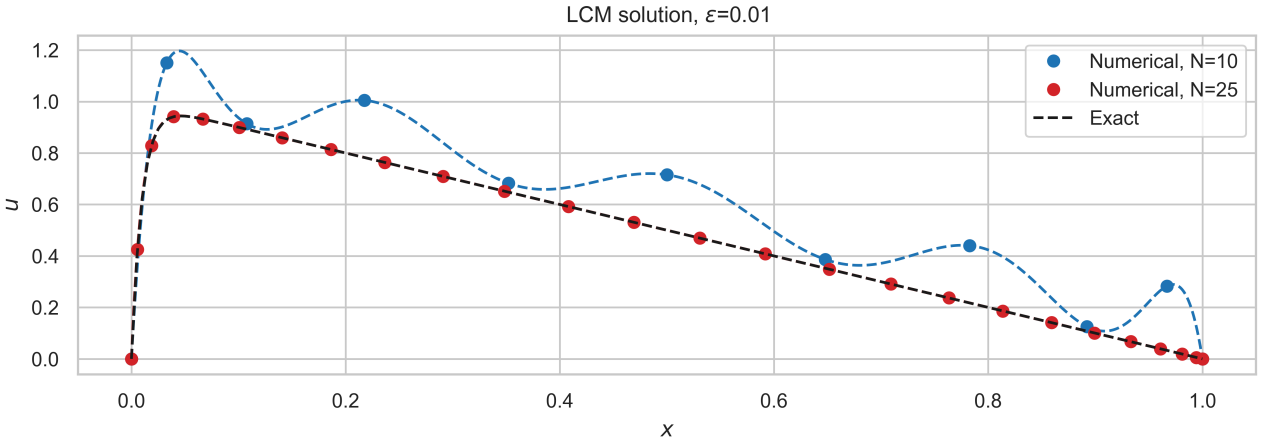

Figure 5: Computed solutions using LCM for different values of N to the problem with $\varepsilon = 0.01$

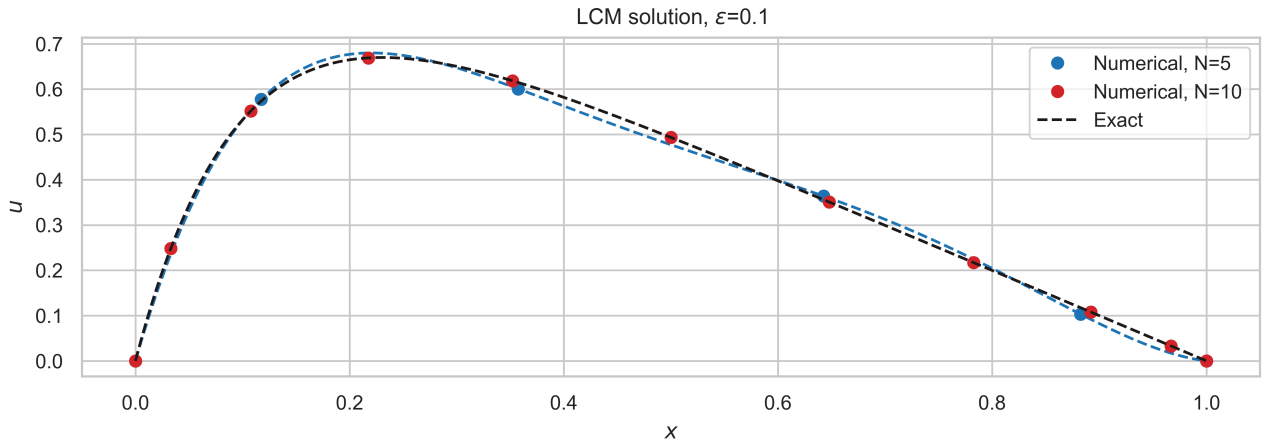

Figure 6: Computed solutions using LCM for different values of N to the problem with $\varepsilon = 0.1$

For both methods, we see that we need quite different grid refinements to achieve results that mimic the solution. To further investigate this, we consider the errors for each of the methods.

1.a.vi) Discussion of Methods

We may evaluate the performance of our models above by considering the relative error against the known solution given by Equation 2 by again using a quadrature to compute a highly accurate estimate of the L^2 norm over the domain $[0, 1]$:

$$\|u_N(x) - u(x)\|_{L^2_{[0,1]}} \approx \sqrt{\sum_{j=0}^{N} \left(u_N(x_j) - u(x_j)\right)^2 J w_j} \tag{44}$$

Where $J = \frac{b-a}{2} = \frac{1}{2}$ is the Jacobian from the coordinate transformation given in Equation 5 and w_j are the weights from the Jacobi-Gauss quadrature that was used to determine the abscissas x_j as detailed in [1, eq. 1.131].

This allows us to investigate the convergence of the methods by plotting the relative error against N for varying values of ε, see Figure 7 and Figure 8.

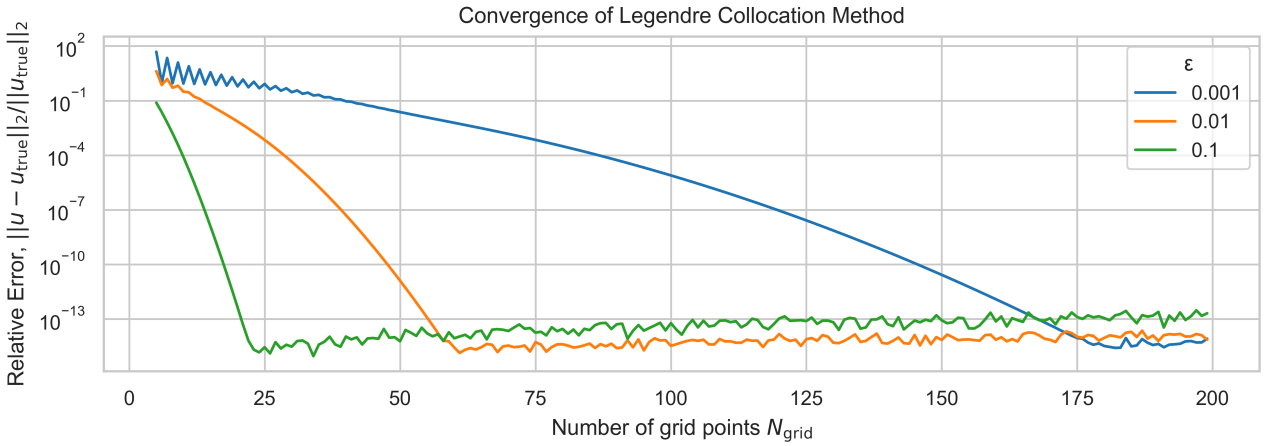

Figure 7: Plot of the relative error for a solution using LCM for varying N.

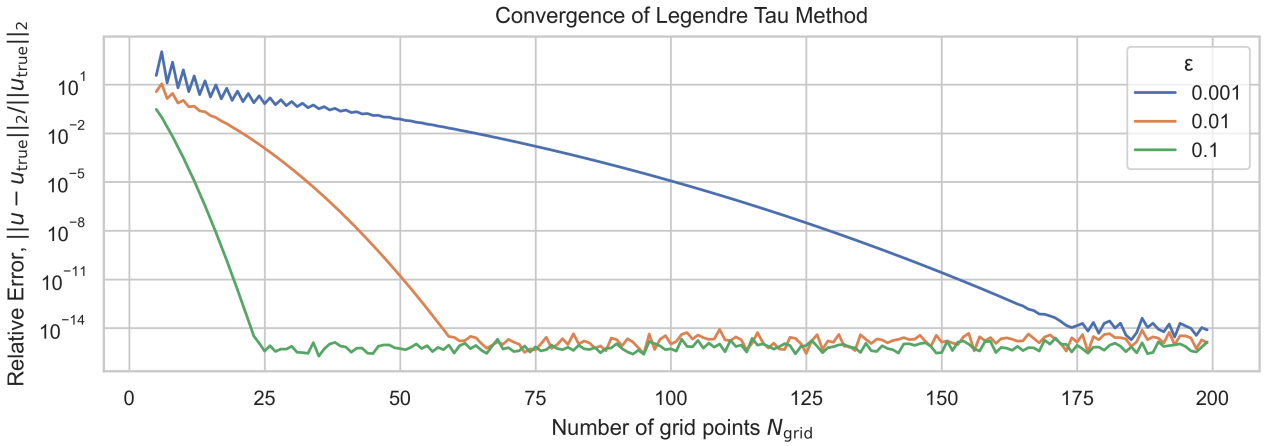

Figure 8: Plot of the relative error for a solution using LTM for varying N.

The plots show that the convergence is quite similar for the two methods and that they perform quite equally. As ε grows, the convergence is much faster, getting to machine-precision around $N = 25$ for $\varepsilon = 0.01$ compared to $N = 175$ for $\varepsilon = 0.001$. To explain this difference, we look at the exact solutions, as plotted, see Figures 1-6. Clearly the solution is much steeper close to $x = 0$ for smaller values of ε. This means that the solver will need a much finer grid to achieve enough precision in the estimation of the derivative close to $x = 0$, explaining the much slower convergence rate.

A disadvantage of both methods is that they have even spacing, which means that there is no other way to accommodate for steeper gradients than simply increasing the grid refinement N. Consequently, we get the slow convergence rate for small ε that we just saw.

The Legendre Tau Method results in a sparse and banded system matrix (excluding the boundary conditions and first 2 rows from Equation 20) which means that it is more memory efficient and the system of equations can be solved in linear time with respect to N. This is of course a large advantage, when we consider problems, such as the problem presented here for $\varepsilon = 0.001$, where a large number of grid points is needed to get precise solutions. Note that the sparsity of the system matrix is obtained by using the recursions and implementing the system matrix only with Equation 20 results in dense upper triangular matrix.

1.b) Irrotational Flow Around a Cylinder

We are given the Laplace's Equation applied to the scalar velocity potential $\phi(x, y)$ in 2 dimensions expressed in polar coordinates:

$$\nabla^2\phi = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial\phi}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2\phi}{\partial\theta^2} = 0, \qquad (r,\theta) \in [r_1, \infty[\times[0, 2\pi], \qquad r_1 > 0 \tag{45}$$

With the exact solution:

$$\phi(r,\theta) = V_\infty\left(r + \frac{r_1^2}{r}\right)\cos(\theta) \tag{46}$$

Where $V_\infty = 1$ is the far-field flow speed and r_1 is the radius of the obstructing cylinder.

In order to carry out numerical analysis, we require the domain to be finite, to which end we introduce $r_2 \in \mathbb{R}$ such that $0 < r_1 < r_2$.

For our discretisation, we choose to keep the original θ coordinate, which we solve using the spectral Fourier collocation method in order to exploit the intrinsic periodicity of the angular coordinate.

For the r coordinate, we employ the spectral Legendre collocation method which has canonical domain $z_r \in [-1, 1]$, which we may transform into our physical domain using the transformation with associated Jacobian:

$$r(z_r) = \frac{r_2 - r_1}{2} z_r + \frac{r_1 + r_2}{2}, \quad J_r = \frac{r_2 - r_1}{2} \tag{47}$$

Thus allowing us to re-express the original differential equation, Equation 45, restricted to a transformed finite radial domain $[-1, 1]$ as:

$$\begin{aligned} \nabla^2 \phi &= \frac{1}{r(z_r)} J_r^{-1} \frac{\partial}{\partial z_r} \left(r(z_r) J_r^{-1} \frac{\partial \phi}{\partial z_r} \right) + \frac{1}{r(z_r)^2} \frac{\partial^2 \phi}{\partial \theta^2} \\ &= \left(J_r^{-1}\right)^2 \frac{\partial^2 \phi}{\partial z_r^2} + \frac{1}{r(z_r)} J_r^{-1} \frac{\partial \phi}{\partial z_r} + \frac{1}{r(z_r)^2} \frac{\partial^2 \phi}{\partial \theta^2} = 0 \end{aligned} \tag{48}$$

Where the domain is given by $(z_r, \theta) \in [-1, 1] \times [0, 2\pi]$.

Construction of the differentiation matrices is similar to the treatment in Section 1.a.iv, though with the use of the discrete Fourier series as the basis functions, $\psi(x)$:

$$\frac{\mathrm{d}^k \psi(x)}{\mathrm{d}x^k} = \left(\frac{i 2\pi n}{L} \right)^k e^{\frac{i 2\pi n}{L} x} \tag{49}$$

With this, we are able to implement the differentiation matrices for the z_r and θ coordinates and use the standard technique of applying the Kronecker product to produce a single system of equations that capture both coordinates:

$$\begin{aligned} \mathcal{D}_{z_r,\text{full}} &= \mathcal{D}_{z_r} \otimes I_\theta \\ \mathcal{D}_{\theta,\text{full}} &= I_{z_r} \otimes \mathcal{D}_\theta \end{aligned} \tag{50}$$

Where $I_{|\cdot|}$ are the identity matrices associated with the respective coordinates.

With this, we are able to construct the following discretisation of the problem, implicitly letting $r = r(z_r)$ and using the 'full' subscript to denote the construction that matches the multi-dimensional differentiation operators:

$$\mathcal{L}_N \phi = \left(J^{-1}\right)^2 \mathcal{D}^2_{z_r,\text{full}} \phi + \frac{1}{r_{\text{full}}} J^{-1} \mathcal{D}_{z_r,\text{full}} \phi + \frac{1}{r^2} \mathcal{D}^2_{\theta,\text{full}} \phi \tag{51}$$

After which the solution ϕ may be computed as:

$$\phi_N = \left(\tilde{\mathcal{L}}_N\right)^{-1} \tilde{f}_N \tag{52}$$

Where the overhead tildes denote that the incorporation of boundary conditions on the obstructing cylinder and the domain boundary as given by $\phi_{\text{cylinder}(\theta)} = \phi(r_1, \theta)$ and $\phi_{\text{boundary}(\theta)} = \phi(r_2, \theta)$ respectively.

Solving this yields a solution for the scalar velocity potential $\phi(r, \theta)$, which in turn can be used to compute a velocity field $(u, v) = \nabla \phi$, which may be represented alongside the velocity potential as seen below in Figure 9, in which we observe the expected motion of the fluid around obstructing cylinder.

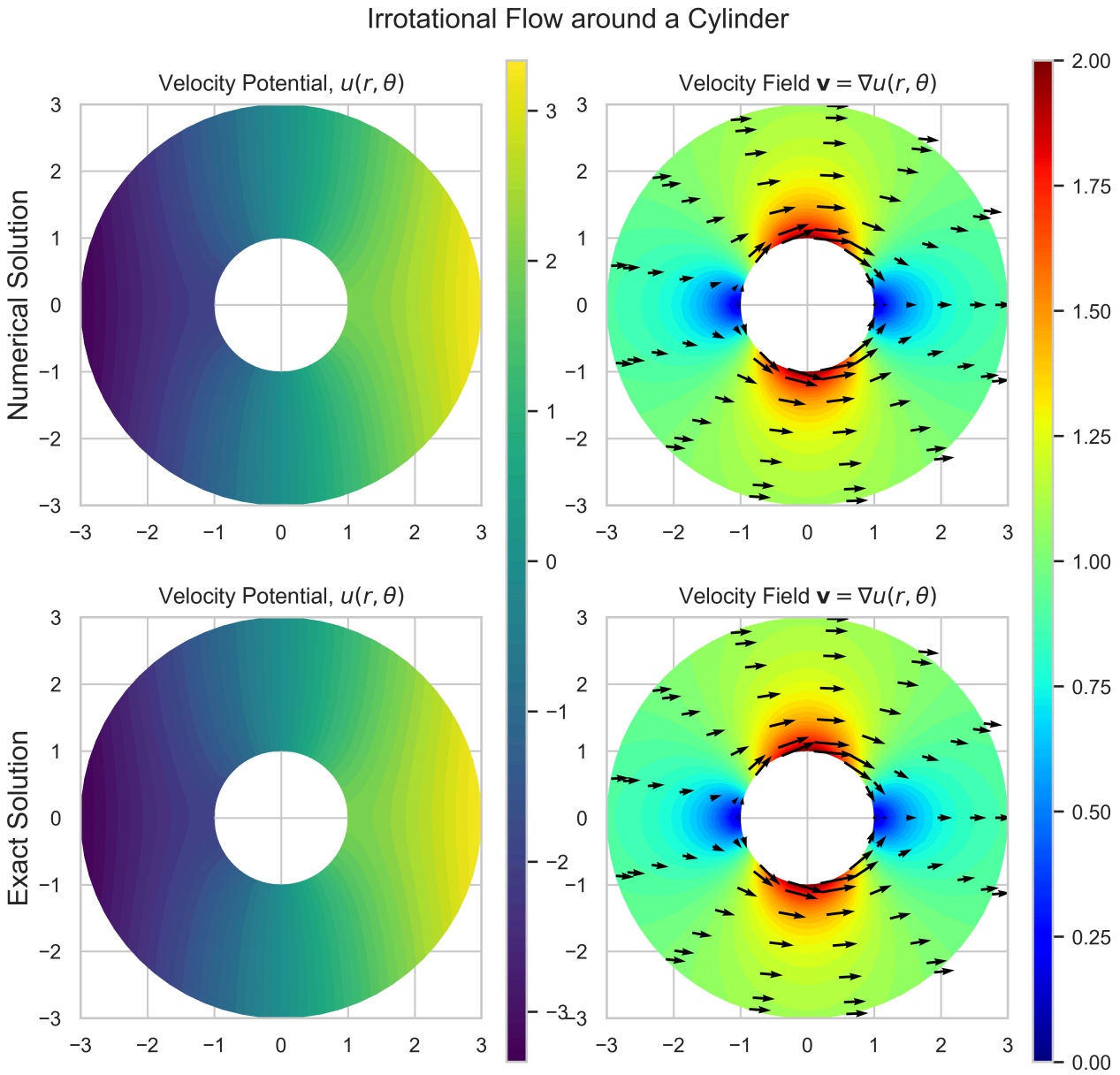

Figure 9: Solution to irrotational flow around an obstructing cylinder with both exact and numerical solutions shown.

We then investigate the convergence behaviour of our implementation as shown in Figure 10. Notably, we find that the solver is *exact* in the angular direction for angular discretisation $N_\theta > 1$, which may be understood by referring to the exact solution from Equation 46:

$$\phi(r, \theta) = V_\infty \left(r + \frac{r_1^2}{r} \right) \cos(\theta) \tag{53}$$

Clearly the angular dynamics will be captured exactly as soon as the first order trigonometric polynomial, $e^{i\theta}$ is included, which exactly captures the $\cos(\theta)$ factor that constitutes the entire angular behaviour of the solution.

We may understand the behaviour for the $N_\theta = 1$ case by asserting that the error is dominated by the angular discretisation in this case. Importantly, N_θ denotes the *discretisation order*, which is one higher than the polynomial order, $P_\theta = N_\theta - 1$. As such, the $N_\theta = 1$ case corresponds to a single mode capturing the signal mean only.

For $N_\theta > 1$ the error is dominated by the radial discretisation and we observe the expected spectral convergence, which achieves machine precision at around $N_r = 20$.

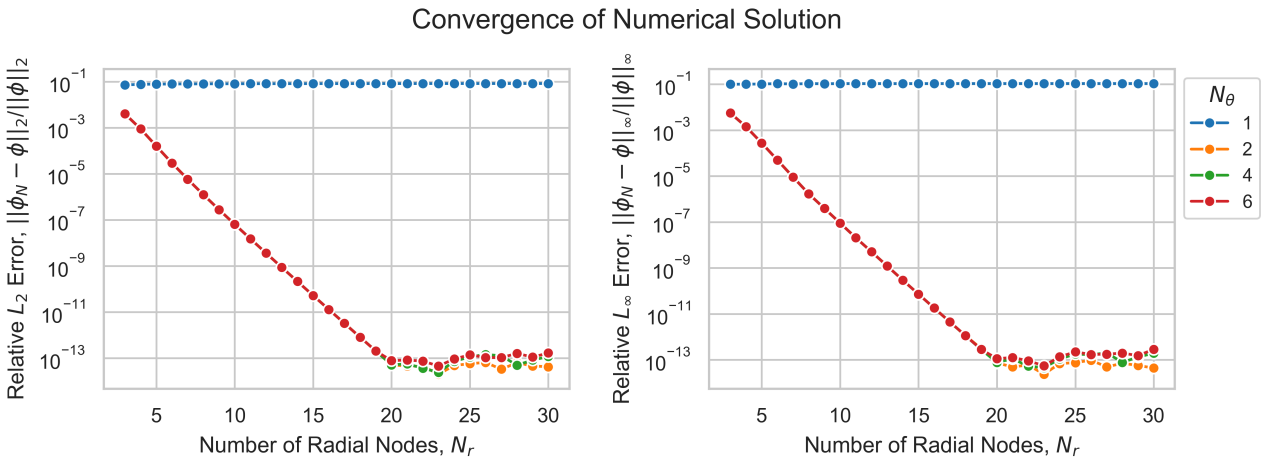

Figure 10: Convergence plot for the irrotational flow around a cylinder solved using Legendre polynomials in the radial direction and trigonometric polynomials in the angular direction.

2) Time-Dependent Problems

2.c) Spectral Fourier Solver for the KdV Equation

We are given the following non-linear initial value problem

$$\frac{\partial}{\partial t}u + 6u\frac{\partial}{\partial x}u + \frac{\partial^3}{\partial x^3}u = 0, \quad -\infty < x < \infty, \quad t > 0 \tag{54}$$

with an exact solution given as

$$u(x,t) = f(x-ct), \quad f(x) = \frac{1}{2}c\,\mathrm{sech}^2\left(\frac{1}{2}\sqrt{c}(x-x_0)\right) \tag{55}$$

where x_0 is the center of the soliton at $t = 0$ and $c > 0$ is the speed of the soliton as it travels to the right. With this knowledge of a solution, we can calculate an initial condition, to be used in our solver,

$$u(x,0) = f(x). \tag{56}$$

To solve this problem, we will treat space and time separately using Method of Lines. For the spatial discretisation, we use the spectral Fourier Collocation method. Along the spectral dimension x we have

$$\lim_{x\to\infty} u(x,t) = 0, \tag{57}$$

for finite t. Thus, if we go out far enough, we may approximate the function by the periodic Fourier basis, since the left and right boundaries are approximately equal for all times (since they will be approximately 0). We will therefore consider x on a finite interval $[-\alpha, \alpha]$ for some large α. Since the Fourier basis functions are defined on $[0, 2\pi]$, we have to be careful when evaluating. However, if we simply scale the frequencies by $\frac{2\pi}{L}$, where L is the length of the interval, when evaluating the Fourier functions, everything will work.

Using the aforementioned Fourier basis with scaled frequencies, we can write an expression for the approximated solution

$$u_N(x,t) = \sum_{k=-N/2}^{N/2} \hat{u}_k(t)\phi_k(x), \quad \phi_k(x) = e^{i\frac{2\pi k}{2\alpha}x} = e^{i\frac{\pi k}{\alpha}x} \tag{58}$$

We use the Fourier differentiation matrix D constructed from the Vandermonde matrices with the Fourier basis functions. We use the spectral Fourier Collocation Method, and we therefore want the residual to be zero at each collocation node,

$$\left\{(u_N)_t + 6u_N(u_N)_x + (u_N)_{xxx}\right\}\Big|_{x_j} = 0, \qquad j = 0, 1, ..., N \tag{59}$$

We look at the derivatives of u_N,

$$\partial_x u_N(x_j,t) = \sum_{k=-N/2}^{N/2} \hat{u}_k(t) i \frac{\pi k}{\alpha} \varphi_k(x_j), \quad \partial_{xxx} u_N(x_j,t) = -\sum_{k=-\frac{N}{2}}^{\frac{N}{2}} \hat{u}_k(t) i \left(\frac{\pi k}{\alpha}\right)^3 \varphi_k(x_j), \tag{60}$$

and insert this into the equation. Defining $\boldsymbol{u}_N(t) = [\hat{u}_0(t), \hat{u}_1(t), ..., \hat{u}_N(t)]$, we get a nonlinear operator

$$L_N \boldsymbol{u}_N(t) = -6 \ \mathrm{diag}(\boldsymbol{u}_N(t)) D \boldsymbol{u}_N(t) - D^3 \boldsymbol{u}_N(t) \tag{61}$$

We obtain a system of ordinary differential equations with initial condition

$$\frac{\partial}{\partial t} \boldsymbol{u}_N(t) = L_N \boldsymbol{u}_N(t), \quad \boldsymbol{u}_N(0) = [f(x_0), f(x_1), ..., f(x_N)]. \tag{62}$$

To solve the system of ordinary differential equations we use `solve_ivp` form SciPy's `integrate` module with it's standard numerical solver `RK45`, which is an explicit Runge-Kutta method of order 5(4), [2].

To ensure that we get a stable solver, we have to control the time stepping. Specifically, we have to control the size of each time step, Δt. For a linear system of equations of the form

$$\frac{dy}{dt} = Ay, \quad y \in R^m, \quad A \in R^{m \times m} \tag{63}$$

the stability condition is the following [3, slide 40]

$$\Delta t \leq \frac{s}{CN^p}, \quad \lambda_{\max}(A) \leq CN^p. \tag{64}$$

where the parameter s is derived from the the absolute stability region of the ODE solver. We have found that for Runge-Kutta 4 $s = 2.7853$ [4].

Since our system matrix is nonlinear, see Equation 61, we consider its linearization such that we could apply the above result. To this end, we apply the technique of frozen coefficients from [5, slide 36], where we approximate the system matrix as

$$\bar{L}_N(t=0) \approx -6 \left(\max_{i \in \{0,...,N\}} |u_N(x_i, t=0)| \right) D - D^3, \tag{65}$$

which results in a linear operator $\bar{L}_N(0)$ and allows us to find the its eigenvalues and relate them to the absolute stability region of the Runge-Kutta solver and the condition Equation 64.

We note that for the considered KdV problem we have that the $\max_{x \in [0,2\pi]} |u(x,t)| = \frac{c}{2}$ is constant over time and therefore

$$\max_{i \in \{0,...,N\}} |u_N(x_i, 0)| \approx \max_{i \in \{0,...,N\}} |u_N(x_i, t)| \quad \Rightarrow \bar{L}_N(0) \approx \bar{L}_N(t) \quad \forall t \tag{66}$$

where the approximation is a result of the discretization. This implies that the eigenvalues $\lambda\left(\bar{L}_N(t)\right)$ should not change with time. If it was not the case, then the eigenvalues of the linear operator $\lambda\left(\bar{L}_N(t)\right)$ should be computed during the time integration of the ODE solver and the stability condition should be updated accordingly.

Consider again Equation 64. For the p value, we have $|\lambda_{\max}| = \max|i| = N$ for D, [5, slide 37]. In this case, we have D^3 in our system, so we must have $p = 3$. To find a value for C and confirm the value for p, we plot $\max_{i \in \{0,...,N\}} |\lambda\left(\bar{L}_N(0)\right)|$, see Figure 11.

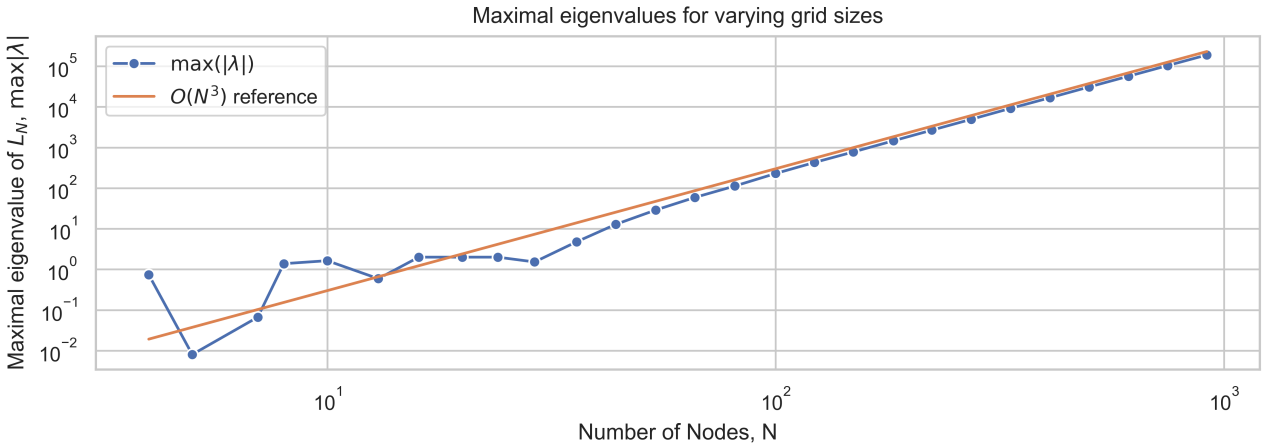

Figure 11: Plot of $\max_N |\lambda|$ for varying values of N. The equation for the reference line is CN^p with $C = 3 \cdot 10^{-4}$ and $p = 3$.

The plot confirms that $p = 3$ and furthermore, we see that choosing $C = 3 \cdot 10^{-4}$ ensures $\lambda_{\max}\left(\bar{L}_N(0)\right) \leq CN^p$ for all large values of N as we want.

We can now check the performance of our solver by plotting the computed solutions and compare them to the exact solution for varying t. This can be seen in Figure 12.

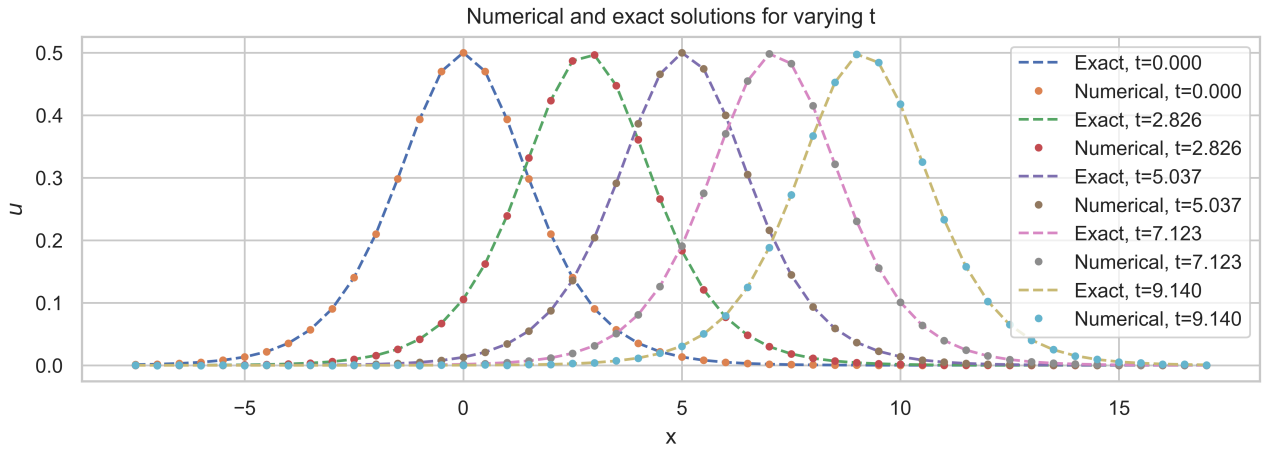

Figure 12: Plot of the numerical solutions and the exact solutions for varying t.

The solver seems to work as wanted. However, we need to do more analysis of the error, and we will do this in the following questions.

2.d) Testing the Solver

To further test the solver, we calculate the estimated errors $\|u - \mathcal{I}_N u\|_2$ and $\|u - \mathcal{I}_N u\|_\infty$ for varying values of the soliton speed, c, and number of nodes, N. We start by doing a plot for $\alpha = 25$, see Figure 13.

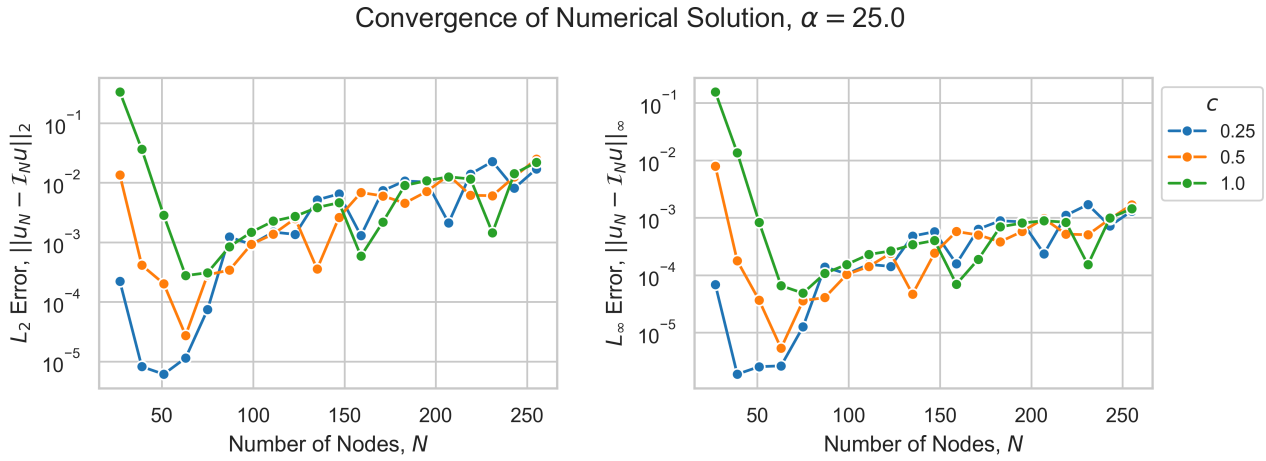

Figure 13: The estimated L_2- and L_∞-errors with $\alpha = 25$ and for different values of soliton speed, c, and varying number of nodes, N.

None of the solutions seem to converge, on the contrary the error grows after $N = 50$. This might stem from the fact that we consider the solution in a too small finite interval, which means that our periodic boundary conditions are not satisfied for the exact solution, we are seeking. We therefore try to solve the problem with $\alpha = 50$, see Figure 14.

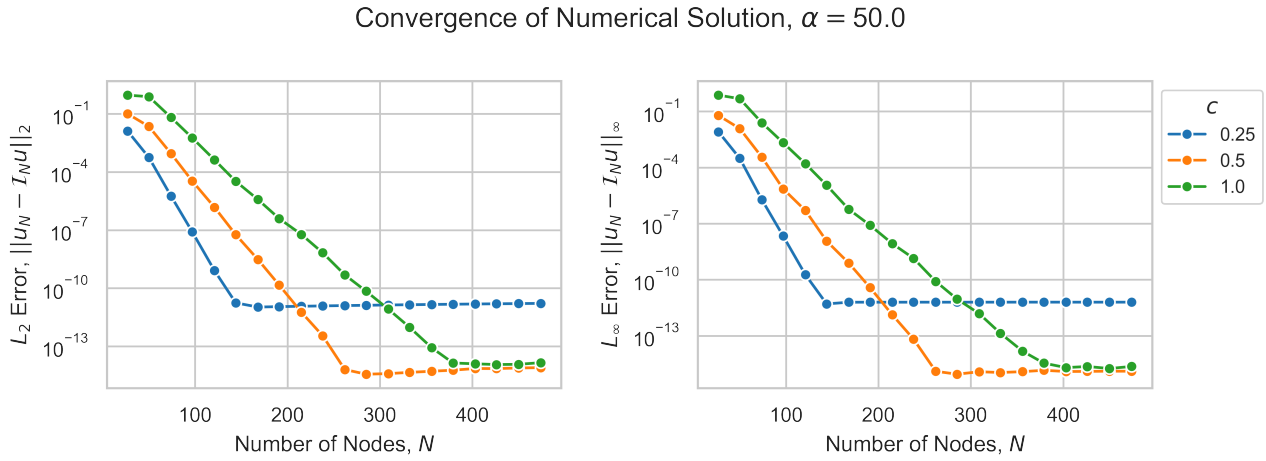

Figure 14: The estimated L_2- and L_∞-errors with $\alpha = 50$ and for different values of soliton speed, c, and varying number of nodes, N.

In this case, we achieve convergence for all values of c. While we reach machine precision for $c = 0.5$ and $c = 1.0$, we are only able to reach approximately 10^{-11} for both of the estimated errors for $c = 0.25$.

The higher floor for a lower c must stem from the fact that the soliton will be more tail-heavy for lower c. Thus we need a larger interval to ensure that the periodic boundary conditions are upheld. This means that it might be worth to decide α depending on c to ensure that we will always be able to find a numerical solution.

Another way to validate the solver is to check that the three fundamental quantities mass (M), momentum (V) and energy (E) are all invariant with respect to time. To approximate the mass, momentum and energy, we only consider the interval $[-\alpha, \alpha]$ to allow us to estimate the integral using the trapezoidal rule as follows

$$M = \int_{-\infty}^{\infty} u(x)\,\mathrm{d}x \approx \int_{-\alpha}^{\alpha} u(x)\,\mathrm{d}x \approx \sum_{i=1}^{N} \frac{u(x_{i-1}) + u(x_i)}{2} \Delta x_i = \widetilde{M}, \tag{67}$$

and likewise for momentum and energy,

$$
\begin{aligned}
V &= \int_{-\infty}^{\infty} u^2(x)\,\mathrm{d}x \approx \sum_{i=1}^{N} \frac{(u(x_{i-1}))^2 + (u(x_i))^2}{2}\Delta x_i = \tilde{V}, \\
E &= \int_{-\infty}^{\infty} \left(\frac{1}{2}u_x^2(x) + u^3(x)\right)\mathrm{d}x \\
&\approx \sum_{i=1}^{N} \frac{\left(\frac{1}{2}(u_x(x_{i-1}))^2 - (u(x_{i-1}))^3\right) + \left(\left(\frac{1}{2}(u_x(x_i))^2 - (u(x_i))^3\right)\right)}{2}\Delta x_i = \tilde{E}.
\end{aligned}
\tag{68}
$$

In the code, we use the function `trapezoidal` from SciPy's `integrate` module. The approximated quantities over time can be seen in Figure 15 for varying N.

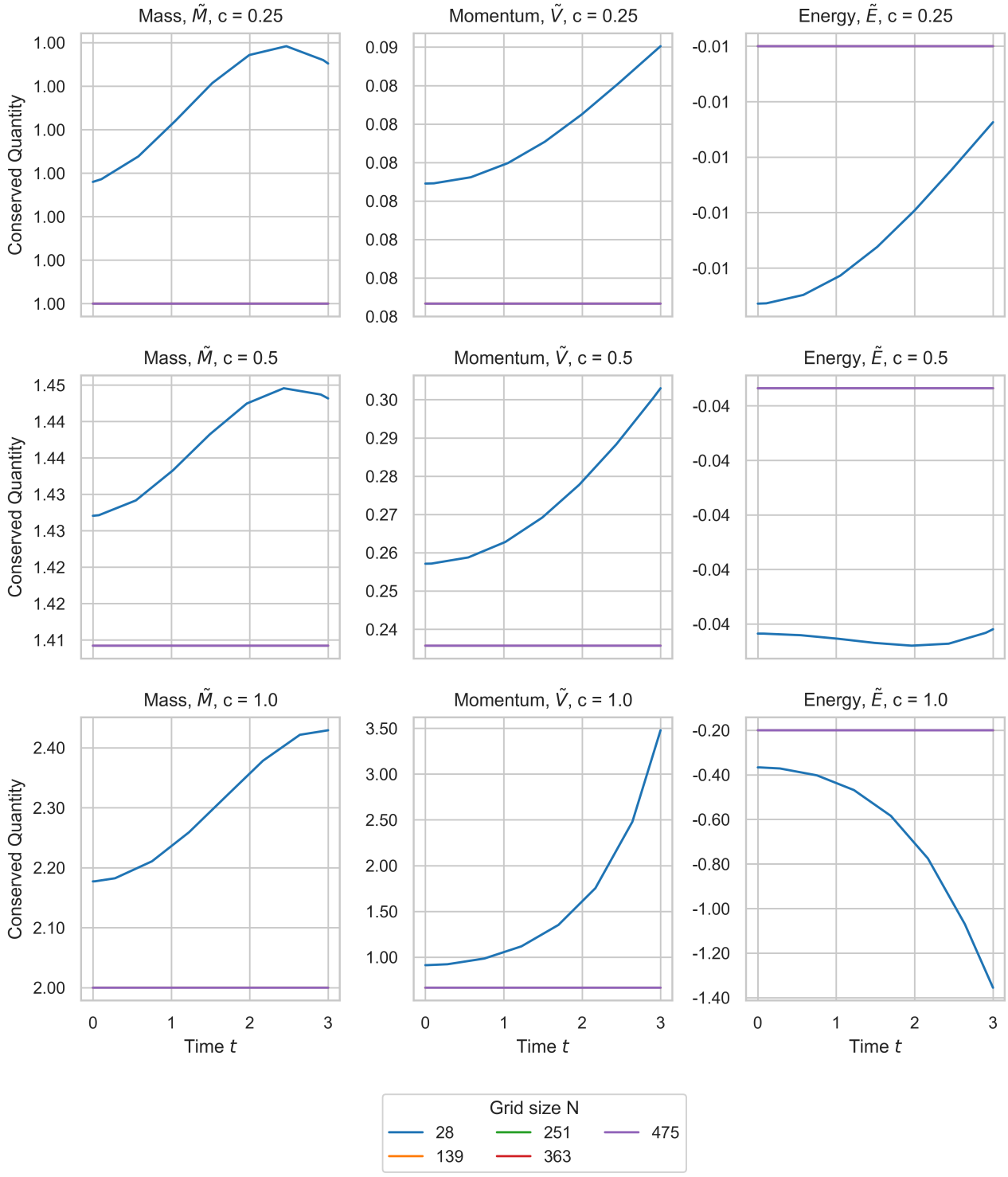

Figure 15: Overview of the three quantities mass, momentum and energy over time for varying grid sizes, N with $\alpha = 50$.

For small grid sizes, we see that the quantities are clearly not conserved for any value of c, which we attribute the phenomenon of *aliasing*, which is discussed further below in Section 2.e. However, when we move up in grid size, we see that the estimated quantities flatten and become constants. This is what we expect from a solution to the KdV solution, and this further validates the convergence of our solver.

2.e) Aliasing Errors

For non-linear terms in a modelled differential equation, here taken to be the product $w(x)$ of two functions $u(x), v(x)$, we get products in the nodal domain:

$$w(x) = u(x)v(x) \tag{69}$$

Each of the functions $u(x), v(x)$ may be considered as an expansion in the modal basis, here taken to be the trigonometric polynomials:

$$u(x) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{u}_k e^{ikx}, \quad v(x) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{v}_k e^{ikx}, \quad w(x) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{w}_k e^{ikx} \tag{70}$$

Where the coefficients for $\beta \in \{u, v, w\}$ are given as:

$$\tilde{\beta}_k = \frac{1}{N} \sum_{j=0}^{N-1} \beta_j e^{-ikx_j} \tag{71}$$

With which we consider the modal coefficients of the product:

$$w_k = \frac{1}{N} \sum_{j=0}^{N-1} \omega_j e^{-ikx_j} = \sum_{m=-\frac{N}{2}}^{\frac{N}{2}-1} \sum_{l=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{v}_m \hat{u}_l \overbrace{\frac{1}{N} \sum_{j=0}^{N-1} e^{i(m+l-k)x_j}}^{\delta_{m+l,k+nN}, \quad n\in\{-1,0,1\}} = \hat{\omega}_k + \underbrace{\sum_{m+l=\pm N} \hat{v}_m \hat{u}_l}_{\text{aliasing errors}} \tag{72}$$

Which tersely outlines the treatment in [5, S.14-15] and reveals by the orthogonality condition that modes at $k \pm N$ are indistinguishable from the kth mode, resulting in *aliasing*.

As such, the element-wise multiplication in the nodal space becomes a convolutional sum in the modal space, which by construction will lead to the creation of higher-order modes, which are *aliased* when restricted to the original lower-order discretisation. This aliasing effect pollutes the low frequency modes.

In order to visualise this effect we investigate the the temporal evolution of the spectral coefficients, as shown in Figure 16. By inspection of the problem, which features a travelling wave, we would expect the frequency spectrum to remain constant over time due to the constant shape of the wave – the movement of the wave is represented by the argument of the spectral coefficients, but their amplitude should remain constant.

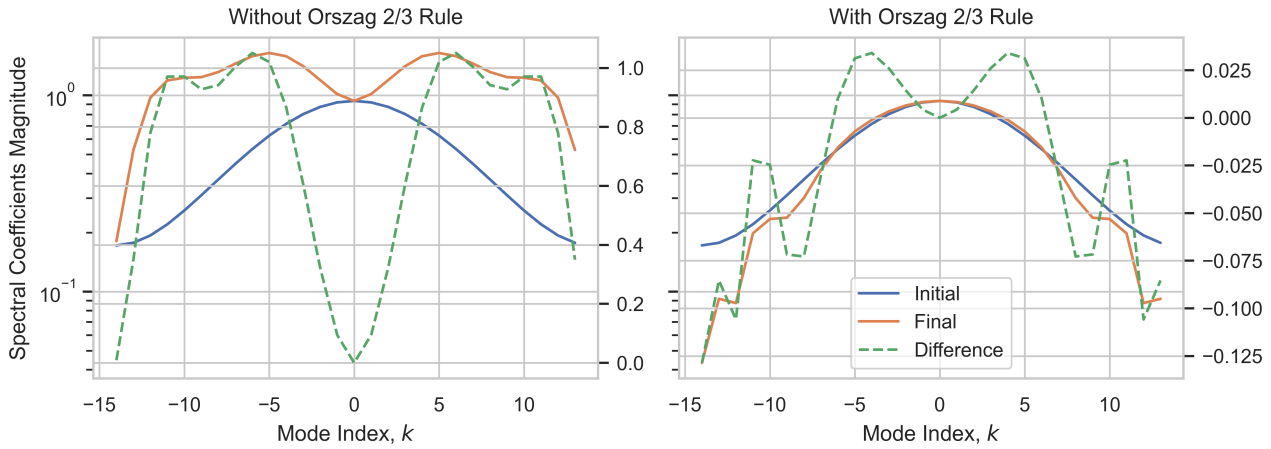

Figure 16: Temporal evolution of Fourier coefficients for the KdV equation with parameters $x \in [-30, 30], c = 1, t \in [0, 3]$ with $N_{\text{grid}} = 28$. Note that the green lines denote the difference between the initial and final spectra and are to be read off of the right-hand ordinate.

Inspection of the left panel in Figure 16 shows that without any mitigation, we *create* energy at the higher order modes as we evolve the system in time, which leads to significant aliasing errors, especially for lower spatial discretisations. This matches the results found in Section 2.d, as seen in Figure 15.

In order to mitigate against this effect, we may perform the convolutional sum in a space containing higher frequency basis functions, as described by [5, S. 11—22] where aliasing effects and *Orszag's 3/2 de-aliasing rule*, which states that the product be carried out in a spectral space of order $M \geq \frac{3}{2}N$ in order to achieve alias-free convolution operations in the spectral space.

In practice, this is by transforming the factors $v(x), u(x)$ into the Fourier space and padding them with an additional $\frac{N}{2}$ higher-frequency modes the associated coefficients being zero, then inverse transforming back into the physical space to carry out the spectral convolution as an element-wise multiplication, after which the product is once more transformed into the frequency space where the padded modes are removed alongside their coefficients, which now are polluted by aliasing, after which the final inverse transformation back into the physical space is undertaken. An outline of the this method implemented in Python may be found in Section 3.a.

Using the Orszag's rule implementation, we once more simulate our system to find the evolution shown in the right panel of Figure 16, which now shows significantly reduced aliasing. The remaining change in the spectrum are attributed as artifacts due to the small boundary condition violation described the sections above. Note that the scale of the right axis corresponding to the difference is different in the two panels.

2.e.i) Reanalysis of conserved quantities using Orszag's de-aliasing rule

Having now implemented Orszag's 3/2 de-aliasing rule, we repeat the experiment described in Section 2.d to produce Figure 17.

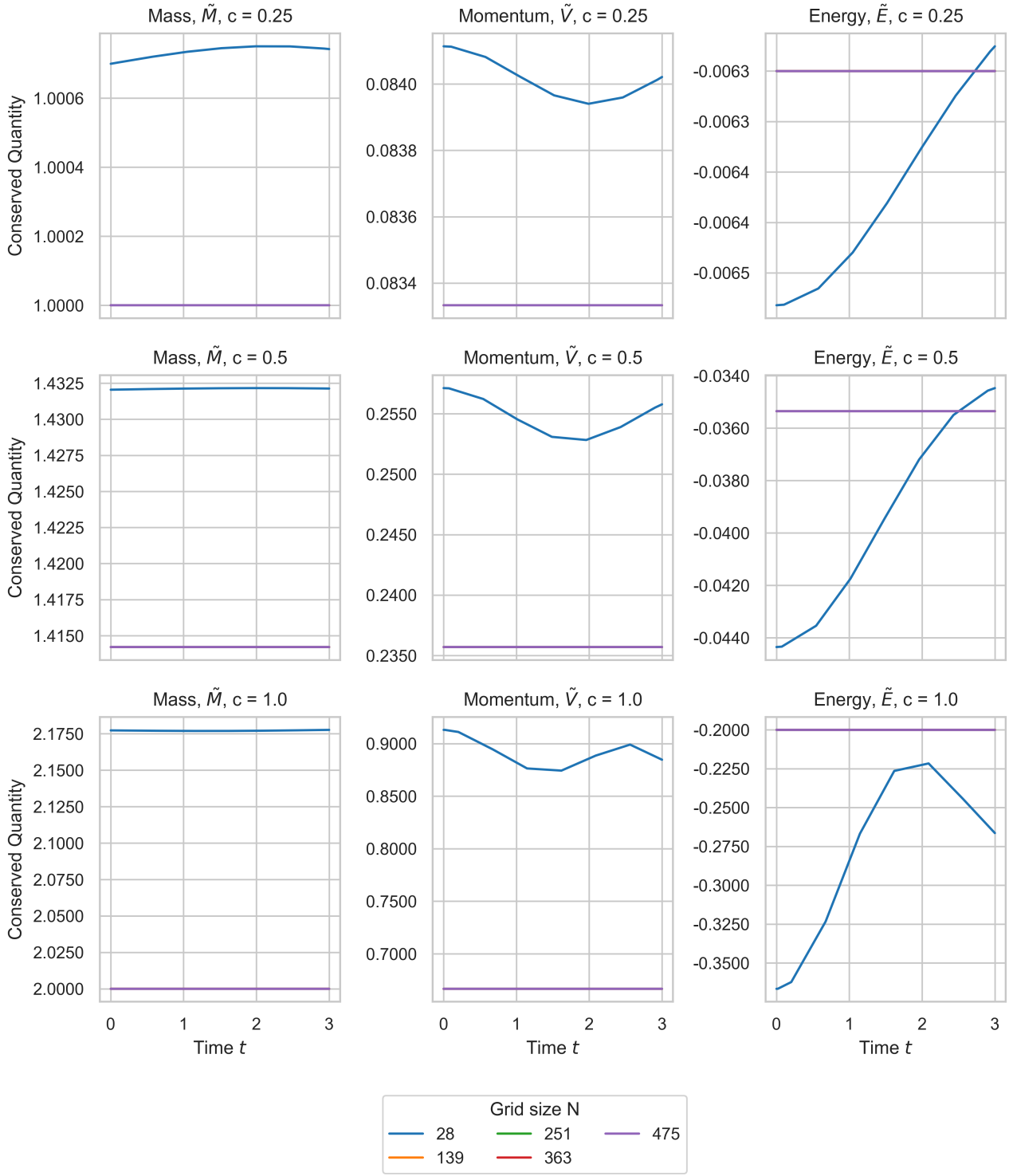

Figure 17: Overview of conserved quantities simulated with the Orszag 3/2 de-aliasing rule implemented.

As apparent when comparing Figure 17 which is produced with the de-aliasing rule enabled and Figure 15 in which aliasing mitigation was undertaken, we observe significantly improved stability in the system seen from the perspective of conservation of energy, momentum, and mass.

This confirms out previous suspicion that aliasing effects were to blame for the variation in these quantities as the system was evolved.

2.f) Collision of Two Waves

We leverage the implementation for the KdV equation outlined in Section 2.c alongside Orszag's 2/3rds dealiasing rule rule described in Section 2.e to model the collision of two solitions constructed using the initial condition corresponding to the superposition of two solitons:

$$u_0(x) = \frac{1}{2}[c_1 \operatorname{sech}^2(\frac{1}{2}\sqrt{c_1}(x - x_{0,1}) + c_2 \operatorname{sech}^2\left(\frac{1}{2}\sqrt{c_2}(x - x_{0,2})\right] \tag{73}$$

Over the domain $x \in [-L_x, L_x]$ with $L_x = 75$, $(x_{0,1}, c_1) = (-40, 0.5)$ and $(x_{0,2}, c_2) = (-15, 0.25)$ over the time interval $t \in [0, 120]$

We visualise the solution using a ridge plot, though notably without the smoothing kernel that is often applied in such plots, in order to obtain Figure 18.

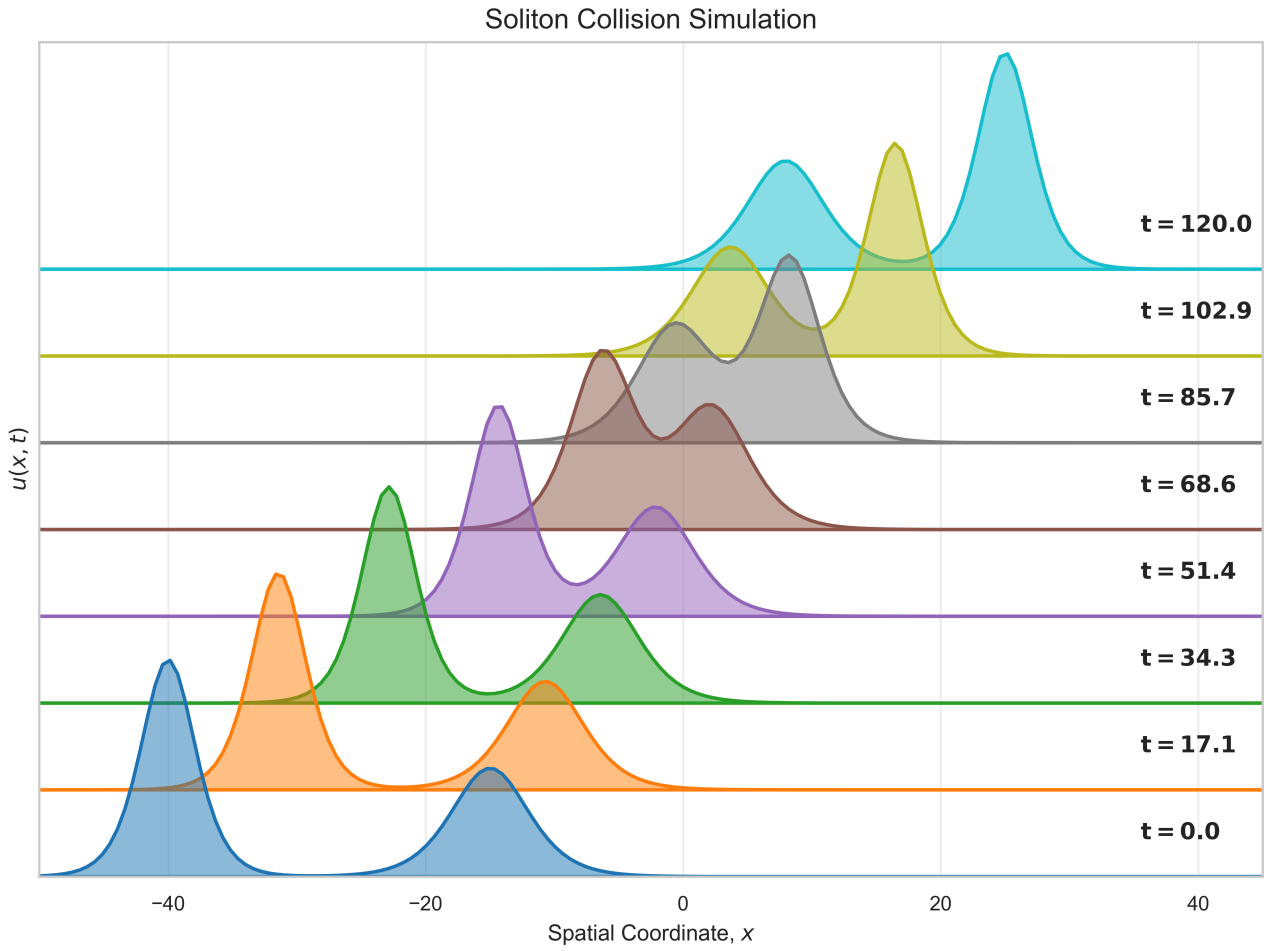

Figure 18: Unsmoothed ridge plot of two solitons colliding using the KdV equation and a Fourier collocation method for the spatial discretisation and the RK4 time-stepping algorithm. Note that the plotted domain is slightly smaller than the simulated domain to aid visibility.

It should be noted that an exact solution may not be produced using simply the superposition of solitons due to the non-linearity of the KdV equation, which enables interaction between the colliding solitons. If the dynamics of the system had instead been described by simple advection, the evolution of the system could have been described by simple superposition of the solitons.

It was identified that the maximum of the resulting solution do not change with time and hence the stability condition from Equation 64 could be calculated with eigenvalue of the linearized operator $\bar{L}_N$ for $t = 0$.

2.g) Scalability Analysis

To get a better understanding the scalability of our solver implementation, we benchmark a version where the differentiation is done by carrying out a Fourier transform such that the convolutional sum of the differentiation may be undertaken in $\mathcal{O}(N)$ time as a element-wise multiplication in the modal domain, after which it is returned to the nodal domain with an inverse Fourier transformation.

Each transformation is performed using a Fast Fourier Transform algorithm, which has $N\log(N)$ scaling and as such will outperform the differentiation matrix based approach, which has N^2 scaling for dense matrices as is the case here.

We instrument the solver to capture the time taken across the time-stepping algorithm, which solves the spatially discretised system at each time step using a Fourier collocation method, which is then divided by to number of evaluations of the right-hand side function, which in this case is the collocation method.

This approach approach yields the scaling seen in Figure 19, which notably does *not* follow the expected scaling. This is attributed to significant and variable overhead in the time-stepping algorithm, as well as the analysis being carried out at relatively low N where linear or constant time operations may still dominate the performance.

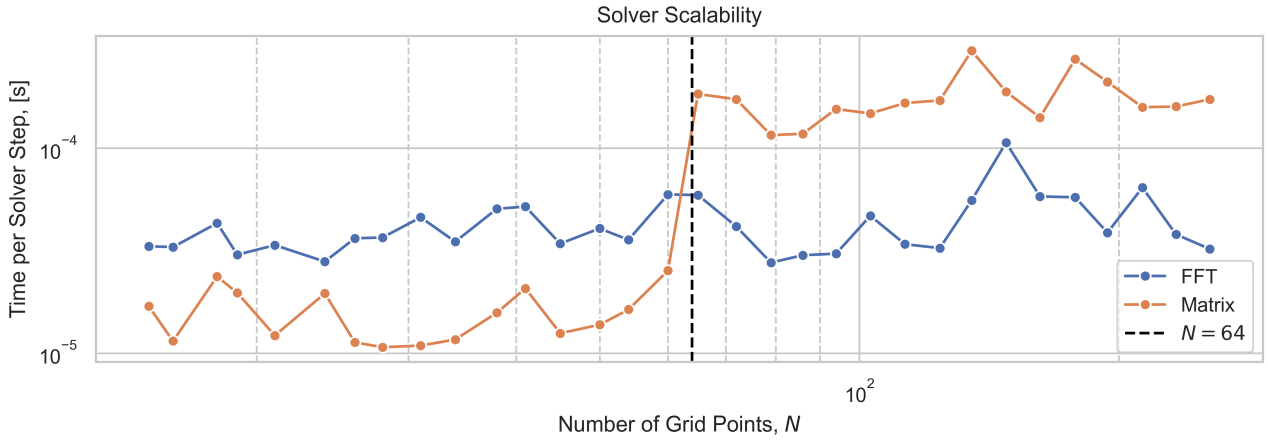

Figure 19: Timing of solver steps collected using *internal* timing method.

To mitigate against this and properly benchmark the solver, the Fourier collocation solver is used to solve only the spatial problem:

$$-6u\partial_x u - \partial_{xxx} u = 0 \tag{74}$$

This process does not involve the time-stepping algorithm at all and is able to more directly sample the performance as it would be per time step. The problem is solved $N_{\text{repeat}} = 1000$ times for each grid size $N \in [32, 4096] \in \mathbb{N}$ following $N_{\text{warmup}} = 100$ solutions that are discarded to prevent burn-in effects, with the result shown in Figure 20.

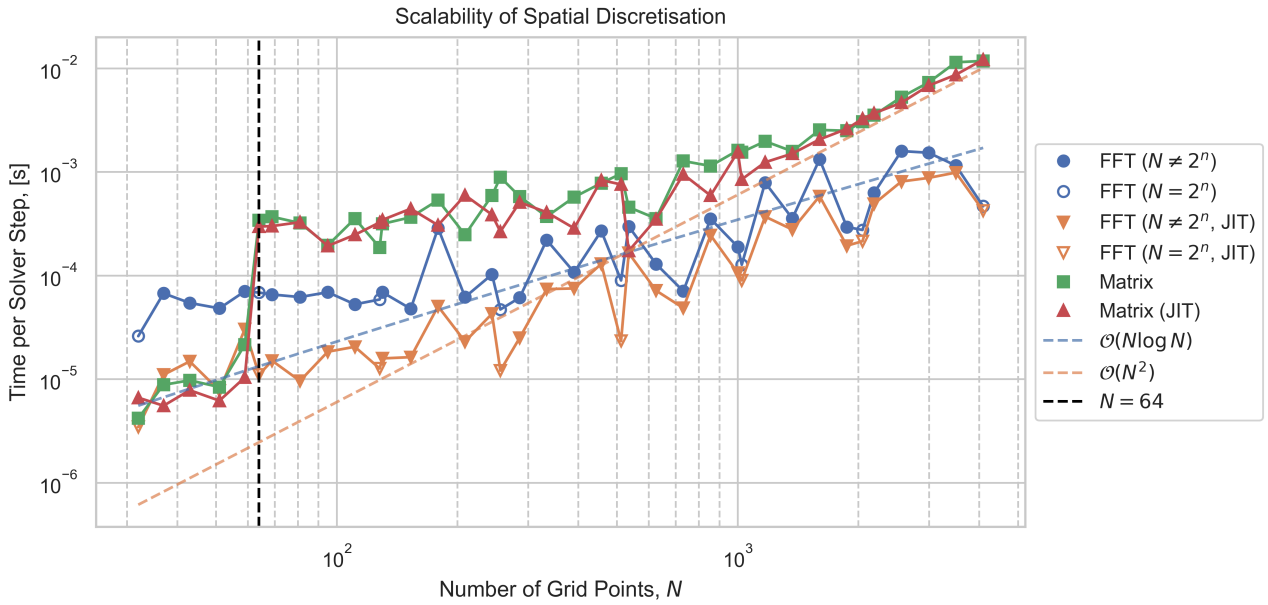

Figure 20: Timing of solver steps collected using *external* timing method.

We note that in both cases the matrix-based solver has a step in performance at $N = 64$, which we theorise to arise from exceeding a cache-level or no longer fitting in a SIMD operation. This also highlights the importance of other operations with below quadratic scaling at low N, which causes the matrix-based implementation to outperform the FFT-based implementation for $N \leq 64$.

As we go to higher grid discretisations we observe the expected scaling characteristics where the matrix-based implementation tends to $\mathcal{O}(N^2)$ in the asymptotic limit while the FFT-based differentiation tends to $\mathcal{O}(N \log(N))$.

As hinted at by the generally higher performance of the $N = 2^n$ sizes, there are certain discretisations that perform very favourably in FFT algorithms. A deeper discussion of these is beyond the scope of this report, but as shown in Figure 20 it may often be beneficial to use an $N = 2^n, n \in \mathbb{N}_1$ discretisation to achieve the best performance.

Our implementation features standard techniques for improved performance including vectorisation of operations where possible. Notably, for the matrix-based implementation the differentiation matrices are precomputed to prevent superfluous computation at every evaluation of the spatial problem. Just-in-time optimisation using `numba` was also undertaken, but as evidenced by Figure 20, we see little performance gains from this, which may be understood by realising that the vectorised `numpy` operations are highly optimised already and leave little on the table performance-wise. A last optimisation that is undertaken is preallocation of the memory for the arrays, which saves slightly on memory allocation and deallocation. This trick is usually very significant for GPU-based setups, where the memory is passed back on forth between devices, but does also provide some speedup in this case.

2.h) Solving an IVP with Space-Time Discretisation

We consider the linear advection equation which is an Initial Value Problem

$$\begin{aligned} \frac{\partial \varphi}{\partial t} + a\frac{\partial \varphi}{\partial x} &= 0, \quad x \in]0, 2\pi[,\, t > 0 \\ \varphi(x, 0) &= \varphi_0(x), \quad x \in [0, 2\pi] \\ \varphi(0, t) &= g_l(x), \quad t \geq 0 \end{aligned} \tag{75}$$

We choose to consider the soliton function from Section 2.c as the solution to the linear advection equation, restricted to the domain $[0, 2\pi]$ and we further derive from it the boundary conditions φ_0, g_l. The function is the following

$$\varphi(x, t) = f(x - ct), \quad f(x) = \frac{1}{2} c \operatorname{sech}^2\left(\frac{1}{2}\sqrt{c}(x - x_0)\right), \quad x \in [0, 2\pi]. \tag{76}$$

We wish to solve the Initial Value Problem as a Boundary Value Problem with the spectral Legendre Collocation Method. We consider the formulation of the linear advection equation as the Boundary Value Problem

$$\begin{aligned} \frac{\partial \varphi}{\partial t} + a\frac{\partial \varphi}{\partial x} &= 0, \quad (x, t) \in]0, 2\pi[\times]0, T_{\max}] \\ \varphi(x, 0) &= \varphi_0(x), \quad x \in [0, 2\pi] \\ \varphi(0, t) &= g_l(x), \quad t \in [0, T_{\max}] \end{aligned} \tag{77}$$

where we denoted by $T_{\max}$ the temporal end of the domain.

We consider the following coordinate transformations to map the two dimensional domain of the BVP problem to the $(z_x, z_t) \in [-1, 1]^2$ domain

$$\begin{aligned} t = \frac{T_{\max}}{2} z_t + \frac{T_{\max}}{2}, \quad \frac{\partial}{\partial t} = \frac{2}{T_{\max}} \frac{\partial}{\partial z_t} \\ x = \pi z_x + \pi, \quad \frac{\partial}{\partial z} = \frac{1}{\pi} \frac{\partial}{\partial z_x}. \end{aligned} \tag{78}$$

After coordinate transformation we obtain the following BVP

$$\begin{aligned} \frac{2}{T_{\max}}\frac{\partial \varphi}{\partial z_t}(z_x, z_t) + a\frac{1}{\pi}\frac{\partial \varphi}{\partial z_x}(z_x, z_t) = 0, \quad (z_x, z_t) \in [-1,1]^2 \\ \varphi(z_x, 0) = \varphi_0(z_x), \quad z_x \in [-1,1] \\ \varphi(0, z_t) = g_l(z_t), \quad z_t \in [-1,1]. \end{aligned} \tag{79}$$

We consider the modal and nodal expansion

$$\varphi_N(z_x, z_t) = \sum_{i=0}^{N_x}\sum_{j=0}^{N_t} \hat{\varphi}_{ij} P_i(z_x) P_j(z_t) = \sum_{i=0}^{N_x}\sum_{j=0}^{N_t} \varphi_{ij} h_i(z_x) h_j(z_t) \tag{80}$$

where the grid points are the Jacobi Gauss-Lobatto grid points, and the P are the Legendre polynomials. We use the spectral Legendre Collocation Method, and we therefore want the residual to be zero at each collocation node,

$$\left\{ \frac{2}{T_{\max}}(\varphi_N)_{z_t} + \frac{a}{\pi}(\varphi_N)_{z_x} \right\}\Bigg|_{\left(z_x^i, z_t^j\right)} = 0, \qquad i = 0, 1, ..., N_x, \quad j = 0, 1, ..., N_t \tag{81}$$

we furhter impose appropriate boundary conditions. We construct the partial derivatives of φ_N by using the Differentiation matrices

$$\frac{\partial \varphi_N}{\partial z_x}(z_x, z_t) = D_{z_x}\boldsymbol{\varphi}, \quad \frac{\partial \varphi_N}{\partial z_t}(z_x, z_t) = D_{z_t}\boldsymbol{\varphi}, \quad D_{z_x} \in \mathbb{R}^{N_x \times N_x}, \quad D_{z_t} \in \mathbb{R}^{N_t \times N_t} \tag{82}$$

We construct the global operator by using the Kronecker products with the Identity matrices

$$L_N = \left(\left(\frac{2}{T_{\max}} D_{z_t} \right) \otimes I_{z_x} \right) + \left(I_{z_t} \otimes \left(\frac{a}{\pi} D_{z_x} \right) \right) \tag{83}$$

after incorporating the boundary conditions the system matrix becomes $\tilde{L}_N$, and the right hand side of the system equation becomes $\tilde{f}$. The system of equations which yields the nodal solution φ_N at collocation points becomes

$$\tilde{L}_N \varphi_N = \tilde{f} \tag{84}$$

We solve the problem with the following parameters $T_{\max} = 1, c = 10, a = 1, x_0 = \pi$. The convergence plots are presented in Figure 21.

We note the relationship between the temporal discretisation N_t and the spatial discretisation N_x. Based on the plots it could be estimated that we should ensure $N_x > \gamma N_t$ where $\gamma \approx 4$, when the gird is refined. It was verified that γ does not depend on the problem parameters. Consider the convergence curve for fixed N_t and as a function of N_x. When $N_x < \gamma N_t$ the spatial error dominates the overall error, since the temporal dimension have been discretised enough, and we get the spectral convergence based on the semilog plot. Once $N_x > \gamma N_t$, further spatial refinement do not result in smaller norm of the error, which indicates that the temporal error dominates. This statement is further supported by the fact that with higher fixed N_t the plateau in the convergence plot is seen on lower values of the norm of the error.

The snapshots of the solution are presented in Figure 22.

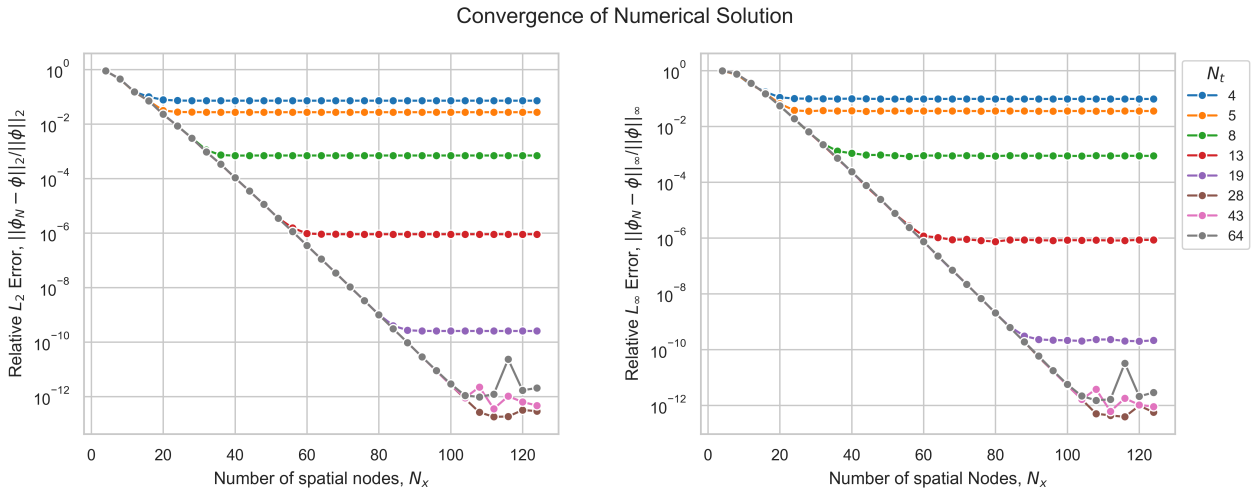

Figure 21: Convergence plot for the linear advection equation solved as the BVP with the Spectral Legendre Collocation method.

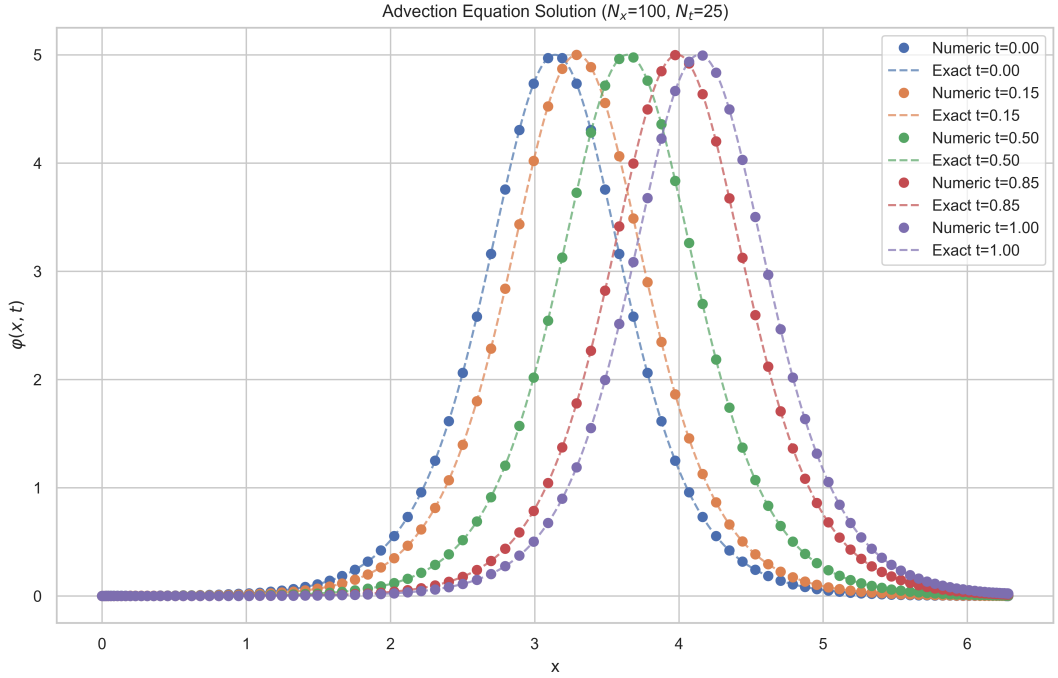

Figure 22: Snapshots of the solutions for the linear advection equation in the original domain $[0, 2\pi]$.

3) Appendix

3.a) Outline of Orszag's 3/2 de-aliasing rule

Python

```
# Compute wave numbers and differentiation operators
k = 2 * np.pi * np.fft.fftfreq(N_grid, d=(L / N_grid))  # Wave numbers
D_hat = 1j * k
D3_hat = (1j * k) ** 3

# Pad
N_size = u_hat.size
M_size = int(np.ceil(N_size * 3 / 2))  # Pad size
if N_size % 2 == 0:
    N_pos = N_size // 2 + 1  # Includes 0, positive freqs, Nyquist freq
```

```
    N_neg = (
        N_size // 2 - 1
    )  # Includes negative freqs excluding Nyquist freq
else:
    N_pos = (N_size + 1) // 2  # Includes 0, positive freqs
    N_neg = N_pos - 1  # Includes negative freqs

u_hat_padded = np.zeros(M_size, dtype=np.complex128)

# Add in original coefficients, note new higher frequencies are at center
u_hat_padded[:N_pos] = u_hat[:N_pos]
u_hat_padded[M_size - N_neg :] = u_hat[N_size - N_neg :]

# Differentiate in spectral space
k_padded = 2 * np.pi * np.fft.fftfreq(M_size, d=(L / M_size))
D_hat_padded = 1j * k_padded
u_x_hat_padded = D_hat_padded * u_hat_padded

# Transform back to physical space
u_padded = np.fft.ifft(u_hat_padded)
u_x_padded = np.fft.ifft(u_x_hat_padded)

# Compute product in physical space
uu_x_padded = u_padded * u_x_padded

# Now remove padding in spectral space
uu_x_hat_padded = np.fft.fft(uu_x_padded)
uu_x_hat = np.zeros(N_size, dtype=np.complex128)
uu_x_hat[:N_pos] = uu_x_hat_padded[:N_pos]
uu_x_hat[N_size - N_neg :] = uu_x_hat_padded[M_size - N_neg :]

# Fix normalisation gain made by FFT computation
uu_x_hat *= M_size / N_size  # Should be ≈ 3/2

u_xxx_hat = D3_hat * u_hat

u_alias_free = np.fft.ifft(-6 * uu_x_hat - u_xxx_hat)
```

Bibliography

[1] D. A. Kopriva, *Implementing Spectral Methods for Partial Differential Equations: Algorithms for Scientists and Engineers*. in Scientific Computation. Dordrecht: Springer Netherlands, 2009. doi: 10.1007/978-90-481-2261-5.

[2] "solve_ivp." Accessed: Oct. 31, 2025. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.html

[3] A. P. Engsig-Karup, "02689 Advanced Numerical Methods, Lecture 6: Initial Value Problems." 2025.

[4] L. R. Hellevik, "Numerical Methods for Engineers." Accessed: Oct. 29, 2025. [Online]. Available: https://leifh.folk.ntnu.no/teaching/tkt4140/._main025.html

[5] A. P. Engsig-Karup, “02689 Advanced Numerical Methods, Lecture 7: Nonlinear Differential Equations.” 2025.