Assignment 1

02689 — Advanced Numerical Methods for Differential Equations

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

1) Fourier Spectral Methods

1.a) Truncated Fourier Expansion

We are given the objective function

$$\tilde{u}(x) \coloneqq \frac{1}{2 - \cos(\pi x)} \qquad x \in [0, 2] \tag{1}$$

Which we manipulate to match the conventional domain $[0, 2\pi]$ of the complex Fourier coefficients c_n in order to obtain

$$u(x) \coloneqq \frac{1}{2 - \cos(x)} \qquad x \in [0, 2\pi] \tag{2}$$

1.a.1) Closed-Form Analytical Fourier Coefficients

In order to determine a closed-form solution to the Fourier coefficients of a Fourier series expansion of Equation 2, we first recall the complex Fourier coefficients as:

$$\begin{aligned} c_n &= \frac{1}{2\pi} \int_0^{2\pi} u(x) e^{-inx} \, \mathrm{d}x \qquad n \in \mathbb{Z} \\ &= \frac{1}{2\pi} \int_0^{2\pi} \frac{1}{2 - \cos(x)} e^{-inx} \, \mathrm{d}x \end{aligned} \tag{3}$$

Inspection of Equation 2 immediately reveals that $u(x)$ is an *even* and *real* function, which in turn implies that the sine component of the integrand must vanish. This can be shown by using Euler's formula to express Equation 3 as:

$$\begin{aligned} c_n &= \frac{1}{2\pi} \left[\int_0^{2\pi} \frac{\cos(nx)}{2 - \cos(x)} \, \mathrm{d}x + \cancel{\int_0^{2\pi} \frac{i \sin(nx)}{2 - \cos(x)} \, \mathrm{d}x} \right] \\ &= \frac{1}{2\pi} \int_0^{2\pi} \frac{\cos(nx)}{2 - \cos(x)} \, \mathrm{d}x \end{aligned} \tag{4}$$

Where the cancellation becomes obvious upon realising that the integrand is odd centered around π (as it is a quotient of an even and an odd function) and thus integrates to zero over the interval $[0, 2\pi]$.

We seek to find a closed-form, analytical solution of Equation 4 by means of induction by considering:

$$c_{n+1} = \frac{1}{2\pi} \int_0^{2\pi} \frac{\cos((n+1)x)}{2 - \cos(x)} \, \mathrm{d}x = \frac{2}{2\pi} \int_0^{2\pi} \frac{\cos(x)\cos(nx)}{2 - \cos(x)} \, \mathrm{d}x - \frac{1}{2\pi} \int_0^{2\pi} \frac{\cos((n-1)x)}{2 - \cos(x)} \, \mathrm{d}x \tag{5}$$

Where the latter expression arises from the use of the trigonometric identities

$$\begin{gathered} \cos(\alpha \pm \beta) = \cos(\alpha)\cos(\beta) \mp \sin(\alpha)\sin(\beta) \\ \Updownarrow \\ \cos(nx + x) + \cos(nx - x) = 2\cos(x)\cos(nx) \end{gathered} \tag{6}$$

We consider the following integral:

$$\begin{aligned}\int_0^{2\pi} \frac{\cos(x)\cos(nx)}{2-\cos(x)} &= \int_0^{2\pi} \frac{(2-(2-\cos(x)))\cos(nx)}{2-\cos(x)}\,\mathrm{d}x \\ &= 2\int_0^{2\pi} \frac{\cos(nx)}{2-\cos(x)}\,\mathrm{d}x - \int_0^{2\pi} \cos(nx)\,\mathrm{d}x \\ &= 4\pi c_n - 2\pi\delta_{0n}\end{aligned} \tag{7}$$

Such that the inductive step from Equation 5 becomes:

$$c_{n+1} = 4c_n - 2\delta_{0n} - c_{n-1} \tag{8}$$

We consider the case $n > 0$ such that Equation 8 becomes:

$$c_{n+1} = 4c_n - c_{n-1} \tag{9}$$

Yielding the characteristic polynomial:

$$\alpha^2 - 4\alpha + 1 = 0 \tag{10}$$

With solutions

$$\alpha_{1,2} = 2 \pm \sqrt{3} \tag{11}$$

Leading to coefficients c_n of the form:

$$c_{n+1} = A_1\alpha_1^n + A_2\alpha_2^n = A_1\left(2+\sqrt{3}\right)^n + A_2\left(2-\sqrt{3}\right)^n \tag{12}$$

Where A_1, A_2 may be determined by considering cases c_0, c_1:

$$\begin{aligned} c_0 = A_1 + A_2 \qquad &= \frac{1}{2\pi}\int_0^{2\pi} \frac{\cos(0)}{2-\cos(x)}\,\mathrm{d}x = \frac{1}{\sqrt{3}} \\ c_1 = A_1\left(2+\sqrt{3}\right) + A_2\left(2-\sqrt{3}\right) &= \frac{1}{2\pi}\int_0^{2\pi} \frac{\cos(x)}{2-\cos(x)}\,\mathrm{d}x = \frac{1}{\sqrt{3}\left(2+\sqrt{3}\right)} \end{aligned} \tag{13}$$

$$\Updownarrow$$

$$A_1 = 0 \qquad \wedge \qquad A_2 = \frac{1}{\sqrt{3}} \tag{14}$$

Thus providing a closed-form solution for the coefficients c_n for the case $n > 0$:

$$c_n = \frac{1}{\sqrt{3}}\left(2-\sqrt{3}\right)^n \tag{15}$$

We may generalise this solution to the $n = 0$ case by simply observing that $c_0 = \frac{1}{\sqrt{3}}\left(2-\sqrt{3}\right)^0 = \frac{1}{\sqrt{3}}$ as in Equation 13.

The case $n < 0$ may be elucidated by symmetry arguments using the fact that the function $u(x) = \frac{1}{2-\cos(x)}$ is even, which implies that the Fourier coefficients themselves will be even: $c_n = c_{-n}$. This allows the complex Fourier coefficients c_n to be given as a single closed-form solution:

$$c_n = \frac{1}{\sqrt{3}}\left(2-\sqrt{3}\right)^{|n|} = \frac{1}{\sqrt{3}\left(2+\sqrt{3}\right)^{|n|}} \qquad n \in \mathbb{Z} \tag{16}$$

We thus find the asymptotic decay rate of the continuous Fourier coefficients to be exponential and of order $\mathcal{O}\left(\left(2+\sqrt{3}\right)^{-|n|}\right)$.

1.a.2) Convergence Behaviour of Truncated Fourier Expansion

We may then investigate the truncation error $\|u(x) - \mathcal{P}_N u(x)\|_2$ where $\mathcal{P}_N$ denotes a truncated Fourier expansion with N modes.

We simply evaluate the exact function and the truncated Fourier expansion on a fine grid over $x \in [0, 2\pi]$ for $N \in \{1, ..., 50\}$ to obtain the graph shown in Figure 1.

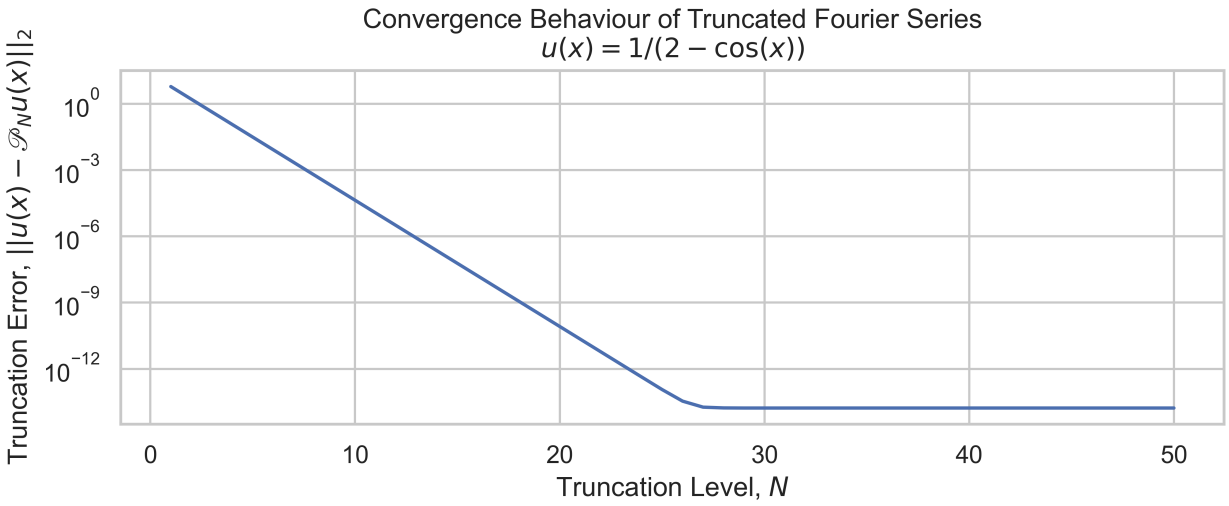

Figure 1: Truncation Error $\|u(x) - \mathcal{P}_N u(x)\|_2$ for $u(x) = \frac{1}{2-\cos(x)}$ over a range of truncation levels, N

We observe that we reach machine precision at $N \approx 27$, after which the truncation error is not improved by employing more modes in the expansion.

1.b) Convergence Behaviour of Discrete Fourier Coefficients

After having found a closed-form expression for the analytical Fourier coefficients in Section 1.a, we would like to compare this with the discrete Fourier coefficients, $\tilde{c}_n$, which may be obtained using a Fast Fourier Transform (FFT).

The discrete Fourier coefficients, $\tilde{c}_n$, are derived below in Section 1.c.1 as part of the even nodal expansion, and may be represented as:

$$\tilde{c}_n = \frac{1}{N} \sum_{j=0}^{N-1} f(x_j) e^{-inx_j} \tag{17}$$

These may be computed using the *Discrete Fourier Transform Method* by leveraging the FFT routines found in modern scientific computing packages such as `numpy` and `scipy`. Care needs to be taken around the normalisation and index conventions, which is handled using `norm="forward"` which scales the transform by $1/N$, and the appropriate `fftshift` operations.

We seek to investigate the effect of aliasing and the errors it introduces in the coefficients. Aliasing refers to the phenomenon where high-frequency modes in a continuous signal are indistinguishable from lower-frequency modes when sampled discretely. Effectively, the higher-frequency modes are 'folded' or 'aliased' into the lower-frequency components, and the resulting discrete coefficient $\tilde{c}_n$ becomes the sum of the true coefficient c_n and all of its higher-frequency aliases $c_{k \pm mN}$. We should be able to observe this effect as a deviation between the discrete and analytical coefficients, particularly at higher wavenumbers. Aliasing effects are often likened to wheels going in reverse in video recordings.

We compute the discrete Fourier coefficients with different levels of discretisation, $N \in \{4, 8, 16, 32, 64\}$ and compare them against the analytical coefficients c_n for which we obtained an analytical expression in Section 1.a. As seen in Figure 2, we observe that the coefficients of $\tilde{c}_n$ decay exponentially with respect to $|n|$ as predicted by theory. The effect of aliasing is greatest at the higher frequency modes, as $n \to |\frac{N}{2}|$.

While this may initially be concerning, we note that the coefficients c_n decay faster for the higher-frequency modes for any sufficiently smooth function, as seen here for $u(x)$.

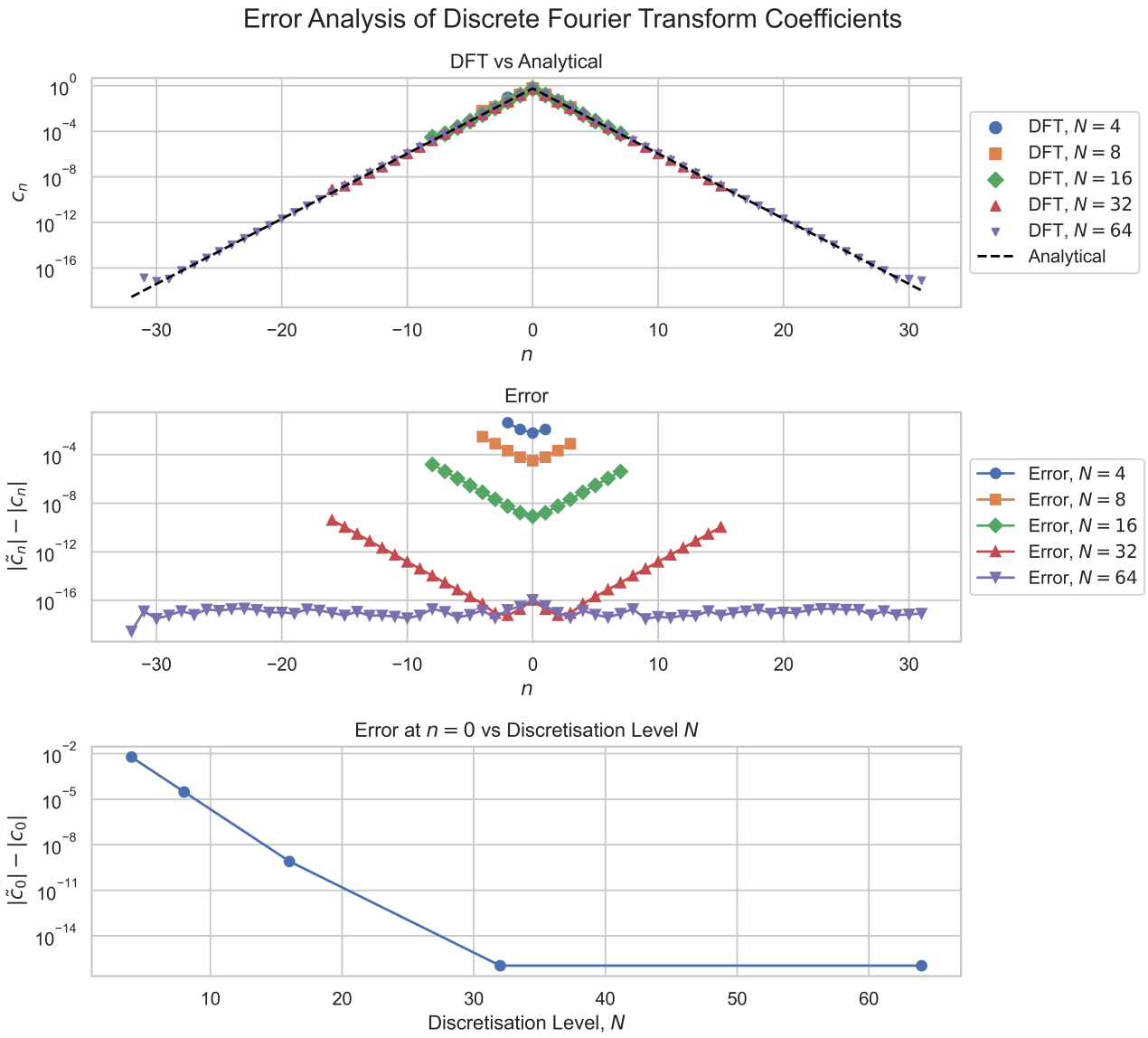

Figure 2: Convergence and errors associated with aliasing effects in discretisation of Fourier coefficients for $u(x) = \frac{1}{2-\cos(x)}$.

From Figure 2 it becomes clear that N must be chosen sufficiently high to capture the significant modes of the underlying function, which matches our expectation that the smoothness of the function sets a lower bound on N required to achieve a given tolerance. Any frequencies above the $N/2^{\text{th}}$ mode, corresponding to the *Nyquist Frequency* are not captured in the expansion and instead enter in the lower frequency modes as an error. As such, we can summarise the smoothness condition as: *sufficient smoothness is implied by the signal having no significant contributions above the Nyquist Frequency.*

In the lower panel of Figure 2 we find spectral convergence for a chosen mode, c_0, reaching machine precision at a discretisation level of $N = 32$.

As such, we find that the trigonometric polynomials form a good basis in which to discretise a signal for any sufficiently smooth, periodic function and can be used to faithfully represent such a signal provided it is without significant contributions above the *Nyquist Frequency*, which is determined by the chosen number of modes, N.

1.c) Nodal Expansion

We seek to represent the discrete trigonometric polynomial in terms of a nodal expansion over the interval $[0, 2\pi)$ using $N = 2k : k \in \mathbb{N}$ points, $x_j = \frac{2\pi}{N} j$ for $j = 0, 1, ..., N-1$. Note the open end of the interval

indicating that the last point, x_{N-1} is not placed at the boundary but rather one grid spacing $h = \frac{2\pi}{N}$ inside the domain.

To aide the reader through the treacherous ambiguity of how the nodal points are placed within the domain, see Figure 3.

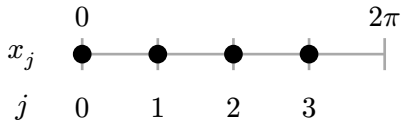

Figure 3: Nodal Expansion. Note that last point x_3 is not located at the boundary, as this point is already given by x_0 due to periodicity of the trigonometric polynomials.

As a consequence of the periodic nature of the Fourier expansion, we find that any reconstruction in such a basis will have the property $u(0) = u(2\pi)$.

1.c.1) Closed-Form Lagrange Polynomial Derivation

We seek to derive a closed-form solution for the Lagrange polynomials using a nodal expansion of the discrete trigonometric polynomial by first recalling the Fourier series:

$$f(x) = \sum_{-\infty}^{\infty} c_n e^{inx} \qquad x \in [0, 2\pi] \tag{18}$$

Where the complex coefficients c_n are given by:

$$c_n = \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-inx} \, \mathrm{d}x \tag{19}$$

We seek to find c_n by evaluating the integral in Equation 19 by quadrature using the trapezoidal rule:

$$\int_0^{2\pi} f(x) e^{-inx} \approx h \sum_{j=0}^{N-1} \left(\frac{f\big(x_{j+1} e^{-inx_{j+1}}\big) + f\big(x_j e^{-inx_j}\big)}{2} \right) \tag{20}$$

Noting that the element $j = N - 1$ of the sum contains the term $f(x_N)e^{-inx_N} = f(x_0)e^{-inx_0}$ by the aforementioned periodicity (which also holds for the factor e^{-inx} by inspection) we may rewrite the sum as follows upon realising that each term enters exactly twice, cancelling out the factor of $\frac{1}{2}$:

$$\int_0^{2\pi} f(x) e^{-inx} \approx h \sum_{j=0}^{N-1} f\big(x_j\big) e^{-inx_j} \tag{21}$$

Which enables the evaluation of the discretised complex coefficients $\tilde{c}_n$ as:

$$\begin{aligned} \tilde{c}_n &= \frac{1}{2\pi} \frac{2\pi}{N} \sum_{j=0}^{N-1} f\big(x_j\big) e^{-inx_j} \\ &= \frac{1}{N} \sum_{j=0}^{N-1} f\big(x_j\big) e^{-inx_j} \end{aligned} \tag{22}$$

Such that an objective function $f(x)$ may be approximated by expansion into a truncated Fourier series using the discrete coefficients in Equation 22 as:

$$
\begin{aligned}
f(x) &\approx \sum_{n=-\frac{N}{2}}^{\frac{N}{2}} \alpha_n \tilde{c}_n e^{inx} \\
&\approx \sum_{n=-\frac{N}{2}}^{\frac{N}{2}} \left(\frac{1}{N} \sum_{j=0}^{N-1} f(x_j) e^{-inx_j} \right) e^{inx} \\
&= \sum_{j=0}^{N-1} f(x_j) \underbrace{\frac{1}{N} \sum_{n=-\frac{N}{2}}^{\frac{N}{2}} e^{-inx_j} e^{inx}}_{h_j(x)}
\end{aligned}
\tag{23}
$$

Where

$$
\alpha_n := \begin{cases} \frac{1}{2} & |n| = \frac{N}{2} \\ 1 & |n| < \frac{N}{2} \end{cases}
\tag{24}
$$

is introduced to handle the overcounting arising from the fact that $e^{i\frac{N}{2}x} = e^{-i\frac{N}{2}x}\ \forall x = x_i$.

$h_j(x)$ is understood to scale the contributions of each node x_j and may be expressed as:

$$
\begin{aligned}
h_j(x) = \frac{\alpha_n}{N} \sum_{n=-\frac{N}{2}}^{\frac{N}{2}} e^{-inx_j} e^{inx} &= \frac{\alpha_n}{N} \sum_{n=-\frac{N}{2}}^{\frac{N}{2}} e^{in\theta} \qquad\qquad \theta := x - x_j \\
&= \frac{1}{N} \left[\frac{e^{i\frac{N}{2}\theta} + e^{-i\frac{N}{2}\theta}}{2} + \sum_{n=-\frac{N}{2}+1}^{\frac{N}{2}-1} e^{in\theta} \right] \\
&= \frac{1}{N} \left[\sum_{n=-\frac{N}{2}}^{\frac{N}{2}} e^{in\theta} - \frac{e^{i\frac{N}{2}\theta} + e^{-i\frac{N}{2}\theta}}{2} \right] \qquad = \frac{1}{N} \left[\sum_{n=-\frac{N}{2}}^{\frac{N}{2}} e^{in\theta} - \cos\left(\frac{N}{2}\theta\right) \right]
\end{aligned}
\tag{25}
$$

Where the sum may be rewritten to closed form using the identity for a finite geometric series,

$$
\sum_{k=0}^{M} ar^k = a\frac{1-r^M}{1-r},
\tag{26}
$$

with initial value $a = e^{-i\frac{N}{2}\theta}$ and common ratio $r = e^{i\theta}$ with $M = N + 1$ elements:

$$
\begin{aligned}
\sum_{n=-\frac{N}{2}}^{\frac{N}{2}} e^{in\theta} &= e^{-i\frac{N}{2}\theta} \cdot \frac{1 - e^{i(N+1)\theta}}{1 - e^{i\theta}} \\
&= e^{-i\frac{N}{2}\theta} \cdot \frac{e^{i\frac{\theta}{2}}\left(e^{-i\frac{\theta}{2}} - e^{i(N+\frac{1}{2})\theta}\right)}{e^{i\frac{\theta}{2}}\left(e^{-i\frac{\theta}{2}} - e^{i\frac{\theta}{2}}\right)} \\
&= \frac{e^{-i(\frac{1}{2}+\frac{N}{2})\theta} - e^{i(\frac{1}{2}+\frac{N}{2})\theta}}{e^{-i\frac{\theta}{2}} - e^{i\frac{\theta}{2}}} \\
&= \frac{2i \cdot \sin\left(\frac{N+1}{2}\theta\right)}{2i \cdot \sin\left(\frac{\theta}{2}\right)} = \frac{\sin\left(\frac{N+1}{2}\theta\right)}{\sin\left(\frac{\theta}{2}\right)}
\end{aligned}
\tag{27}
$$

Which in turn enables a final rewrite of $h_j(x)$ using the identity $\sin(\alpha \pm \beta) = \sin(\alpha)\cos(\beta) \pm \cos(\alpha)\sin(\beta)$:

$$
\begin{aligned}
h_j(x) &= \frac{1}{N}\left[\frac{\sin\left(\frac{N+1}{2}\theta\right)}{\sin\left(\frac{\theta}{2}\right)} - \cos\left(\frac{N}{2}\theta\right)\right] \\
&= \frac{1}{N}\left[\frac{\sin\left(\frac{N}{2}\theta\right)\cos\left(\frac{\theta}{2}\right) + \cos\left(\frac{N}{2}\theta\right)\sin\left(\frac{\theta}{2}\right)}{\sin\left(\frac{\theta}{2}\right)} - \cos\left(\frac{N}{2}\theta\right)\right] \\
&= \frac{1}{N}\left[\sin\left(\frac{N}{2}\theta\right)\cot\left(\frac{\theta}{2}\right) + \cos\left(\frac{N}{2}\theta\right) - \cos\left(\frac{N}{2}\theta\right)\right] \\
&= \frac{1}{N}\sin\left(\frac{N}{2}(x-x_j)\right)\cot\left(\frac{x-x_j}{2}\right) \qquad \blacksquare
\end{aligned} \tag{28}
$$

Inspecting Equation 28 easily reveals that the polynomials take the form of the Dirac delta function over the nodes, $h_j(x_i) = \delta_{ij} \quad \forall i, j \in \{0, ..., N-1\}$.

1.c.2) Visualisation of Lagrange Polynomials

Having now obtained an expression for the Lagrange Polynomials, $h_j(x)$ as given in Equation 28, we visualise the polynomials for the $N = 6$ case as shown in Figure 4.

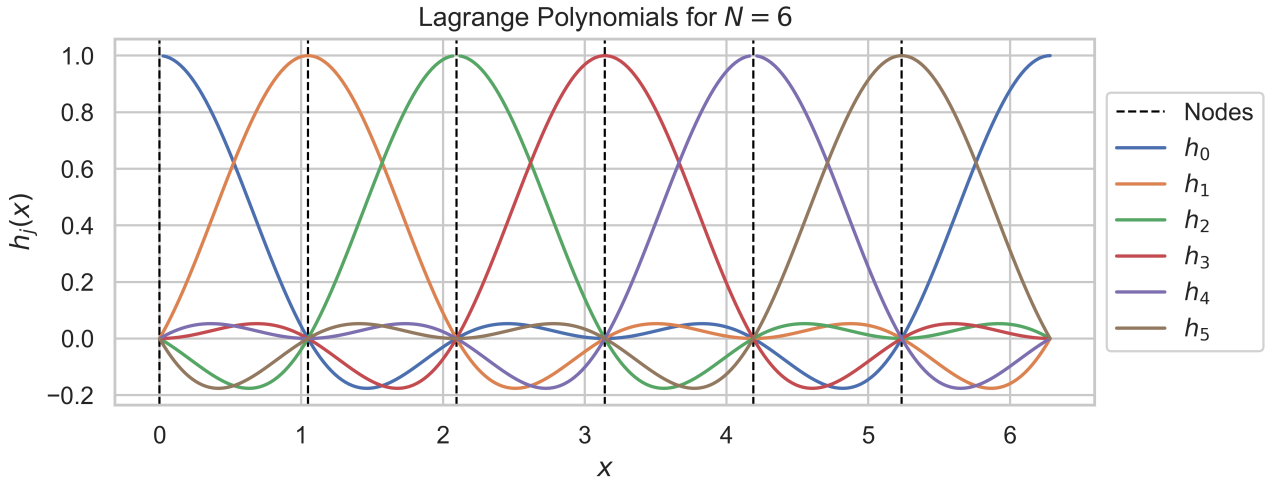

Figure 4: Lagrange Polynomials $h_j(x)$ for $N = 6$. Note that $h_j(x_i) = \delta_{ij}$ where x_i are demarked by the dashed vertical lines.

1.c.3) Fourier Differentiation Matrix

To derive the first-order Fourier Differentiation matrix D, we differentiate the Fourier Interpolation function

$$
I_N u(x) = \sum_{j=0}^{N-1} u(x_j) h_j(x) \text{ where } h_j(x) = \frac{1}{N}\sin\left(\frac{N}{2}(x-x_j)\right)\cot\left(\frac{1}{2}(x-x_j)\right) \tag{29}
$$

We seek the following expansion

$$
\frac{d}{dx} I_N u(x) = \sum_{j=0}^{N-1} u(x_j) h_j'(x) \tag{30}
$$

The derivative of Lagrange polynomial located at point $x_j = \frac{2\pi}{N} j$ is

$$
h_j'(x) = \frac{1}{N}\left(\frac{N}{2}\cos\left(\frac{N}{2}(x-x_j)\right)\cot\left(\frac{1}{2}(x-x_j)\right) - \frac{1}{N}\frac{\sin\left(\frac{N}{2}(x-x_j)\right)}{\sin\left(\frac{1}{2}(x-x_j)\right)^2}\right) \tag{31}
$$

We consider the equidistant grid of the same points that was used to construct Lagrange polynomials, namely $x_k = \frac{2\pi}{N} k$, resulting in $x_k - x_j = \frac{2\pi}{N}(k-j)$. Inserting into the above expression leads to the following simplification

$$h_j'(x_k) = \frac{1}{N}\left(\frac{N}{2}\cos(\pi(k-j))\cot\left(\frac{\pi}{N}(k-j)\right)\right) - \frac{1}{N}\frac{\sin(\pi(k-j))}{\sin\left(\frac{\pi}{N}(k-j)\right)^2} = \frac{1}{2}(-1)^{k-j}\cot\left(\frac{\pi}{N}(k-j)\right) - 0 = \frac{1}{2}(-1)^{k-j}\frac{\cos\left(\frac{\pi}{N}(k-j)\right)}{\sin\left(\frac{\pi}{N}(k-j)\right)}. \tag{32}$$

Note that $(-1)^{k-j} = (-1)^{k+j}$. Thus, the entries of the Fourier Differentiation matrix D are

$$D_{kj} = h_j'(x_k) = \begin{cases} \frac{1}{2}(-1)^{k+j}\cot\left(\frac{\pi}{N}(k-j)\right) & \text{if } j \neq k \\ 0 & \text{if } j = k \end{cases} \tag{33}$$

1.d) Fourier Differentiation Routine

We consider the function

$$v(x) = \exp(\sin(x)) \quad x \in [0, 2\pi] \tag{34}$$

where we changed the argument to match the interval of the Fourier Differentiation matrix D. The derivative of v is

$$v'(x) = \exp(\sin(x))\cos(x) \quad x \in [0, 2\pi] \tag{35}$$

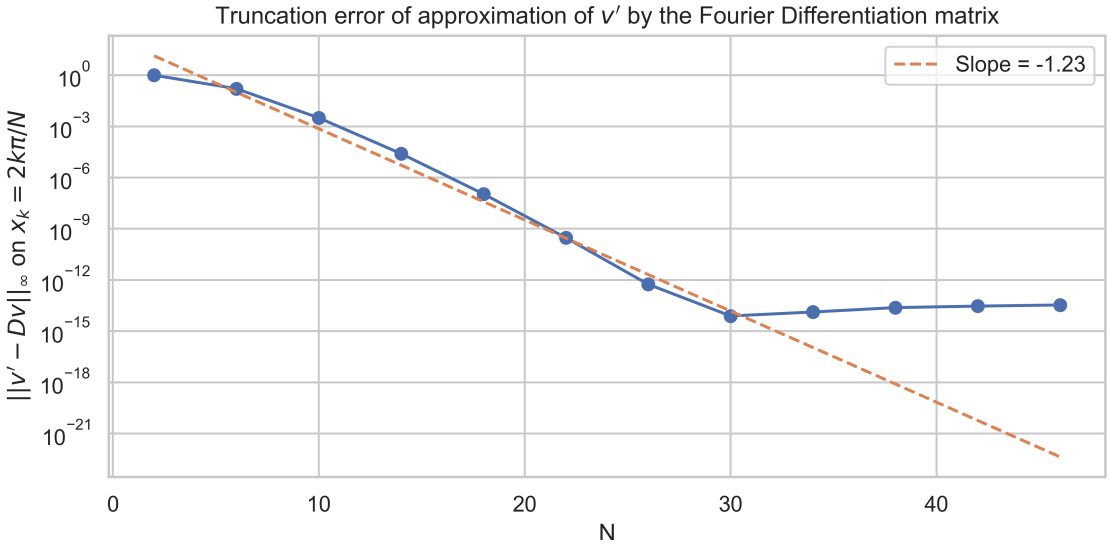

Figure 5: Truncation error of $\max_{j\in[0,N-1]}\{|v'(x_j) - Dv(x_j)|\}$ across different N in the semi-log plot.

Figure 5 presents the convergence test for increasing N. We observe spectral convergence which is justified by the fact that $v \in C^\infty([0, 2\pi])$, which implies that function v is smooth and hence Fourier Series converges rapidly.

1.e) Convergence in the L^2-norm

We consider the sequence of functions $w^i(x)$ with changed argument such that the domain of each function is $[-2\pi, 2\pi]$. Consider the analytical integration

$$w^{i+1}(x) = \int_{-2\pi}^{2\pi} w^i(x)dx \quad \text{such that} \quad w^{i+1} \in C^i([-2\pi, 2\pi]) \quad \text{where} \quad w^i(x) = \frac{dw^{i+1}}{dx}(x) \tag{36}$$

We obtain the following functions

$$w^0(x) = \begin{cases} -\cos(x) & \text{if } x \in [-2\pi, 0) \\ \cos(x) & \text{if } x \in [0, 2\pi] \end{cases} \quad w^1(x) = \begin{cases} -\sin(x) & \text{if } x \in [-2\pi, 0) \\ \sin(x) & \text{if } x \in [0, 2\pi] \end{cases} \tag{37}$$

$$w^2(x) = \begin{cases} \cos(x) - 1 & \text{if } x \in [-2\pi, 0) \\ -(\cos(x) - 1) & \text{if } x \in [0, 2\pi] \end{cases} \quad w^3(x) = \begin{cases} \sin(x) - x & \text{if } x \in [-2\pi, 0) \\ -(\sin(x) - x) & \text{if } x \in [0, 2\pi] \end{cases} \tag{38}$$

Figure 6 shows the functions $w^i(x)$ for $x \in [-2\pi, 2\pi]$. Note that the function w^0 is not periodic on the domain and has a discontinuity at $x = 0$, whereas the function w^i for $i \geq 1$ are periodic on $[-2\pi, 2\pi]$.

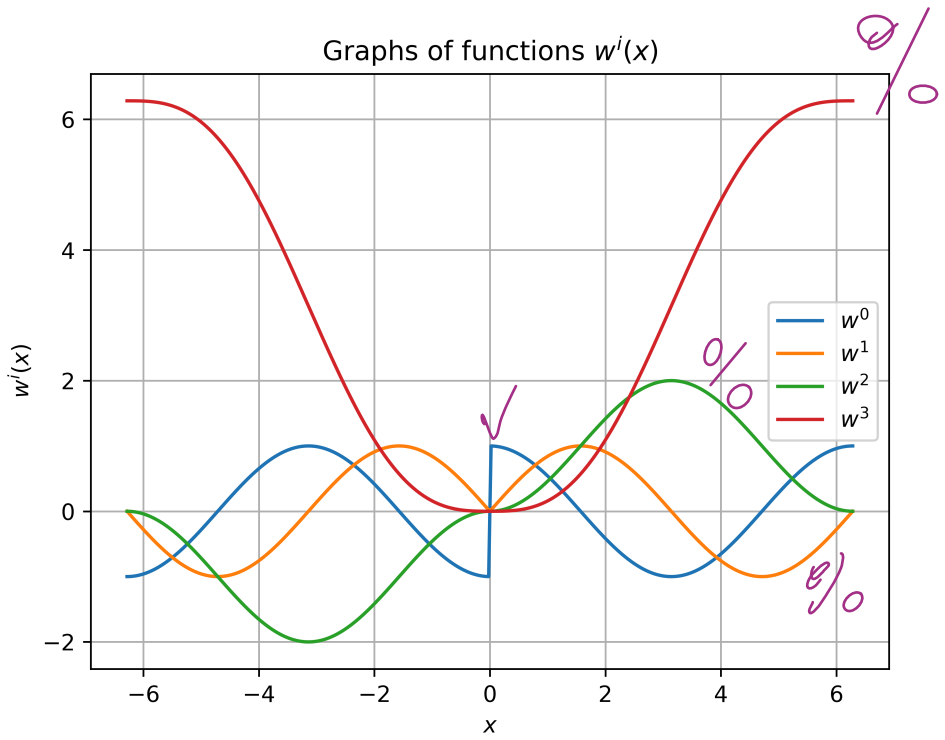

Figure 6: Functions $w^i(x)$ for $x \in [-2\pi, 2\pi]$

Consider the derivative of the interpolating Lagrange polynomial

$$I_N(w^i)'(x) = \sum_{j=0}^{N} w^i(x_j) h_j'(x) \tag{39}$$

We can write the expression

$$w^{i-1}(x) = (w^i)'(x) = \sum_{j=0}^{N} w^i(x_j) h_j'(x) + \tau_N(x) \tag{40}$$

Consider the L^2 norm of the truncation error

$$\|\tau_N\|_{L^2}^2 = \int_{-2\pi}^{2\pi} (\tau_N(x))^2 dx = \int_{-2\pi}^{2\pi} \left(w^{i-1}(x) - \sum_{j=0}^{N} w^i(x_j) h_j'(x) \right)^2 dx \tag{41}$$

Consider the estimate of the L^2 norm by the discrete L^2 norm obtained via a quadrature rule with M points

$$\|\tau_N\|_{L^2}^2 = \int_{-2\pi}^{2\pi} (\tau_N(x))^2 dx = \sum_{m=0}^{M} (\tau_N(x_m))^2 w_m + \eta_M \tag{42}$$

where η_M is the truncation error of the quadrature rule, and we use the formula for quadrature rule for periodic functions. We use the quadrature routine from `scipy` with the number of points M such that the truncation error η_M is below machine precision. This allows us to observe only the truncation error τ_N for different N when computing the discrete L^2 norm on the convergence test. Note that this is a generalization compared to the Fourier Differentiation matrix D, which constructs the Lagrange polynomials on a specific uniform grid and evaluates the derivative on the same uniform grid. In the above formulation, the interpolating Lagrange polynomial is constructed on the uniform grid with N points, but the derivative is evaluated on the grid of size M.

Figure 7 presents the convergence test for discrete first derivative of w^i. We have the following

$$w^1 \in C^0([-2\pi, 2\pi]) \quad w^2 \in C^1([-2\pi, 2\pi]) \quad w^3 \in C^2([-2\pi, 2\pi]) \tag{43}$$

Therefore the upper bound for the algebraic convergence is 1 for w^1, 2 for w^2, and 3 for w^3. We observe that for derivatives of w^2, w^3 the convergence rate is higher. Note, that since the function $w^0 = (w^1)'$ does not have a continuous first derivative and is not periodic the interpolating Lagrange polynomials do not converge point wise for points $x \in \{-2\pi, 0, 2\pi\}$ and the neighboring points, and the Gibbs oscillations can be observed. However the convergence in the L^2 norm is achieved although only linear based on the convergence test.

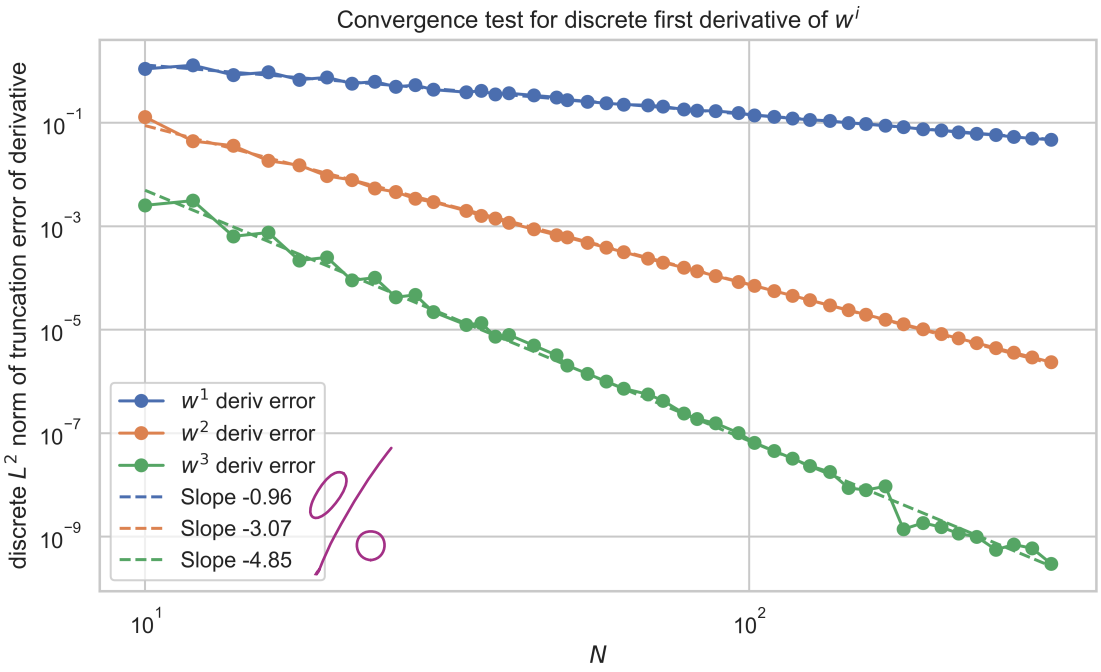

Figure 7: Convergence test in the log-log plot for approximating the derivative of w^i with Lagrange polynomials in a loglog plot.

1.f) Numerical Differentiation using Fast Fourier Transform

The derivative of function v can be computed via FFT by the following formula

$$\frac{d}{dx} P_N v(x) = \sum_{n=0}^{N-1} c_n (in) e^{inx} \tag{44}$$

Therefore we proceed by obtaining the coefficients $\tilde{c}_n$ by the FFT across the equidistant grid of N points, and subsequently we compute the inverse FFT of the set of coefficients $\{inc_n\}_{n=0}^{N-1}$. The comparison of CPU time between the computation of derivative of function v via the Fourier Differentiation matrix D and the FFT are presented in Figure 8. The break point beyond which the FFT is faster than Fourier Differentiation matrix D was found to be $N = 10^3$ on MacBook M4 Pro. The asymptotic convergence rate for Fourier Differentiation matrix D was found to be $O(N^2)$ and the asymptotic convergence rate for FFT was found to be $O(N \log(N))$ for $N > 10^2$ which is consistent with the theory.

Note that for the comparison the function $v \in C^\infty([0, 2\pi])$ was used and spectral convergence is observed on the right plot of Figure 8 - The level of machine precision is achieved with less than $N = 50$. Therefore for this function the differentiation via the Fourier Differentiation matrix D should be preferred. Functions which have only up to specific number of continuous derivatives converge algebraically and would require higher N to achieve machine precision.

Generally, the results imply that the choice between Fourier Differentiation matrix D and FFT for derivative computation should be based on the expected number of required N, while the number of

required N to achieve desired precision should be based on the smoothness of the function to be differentiated.

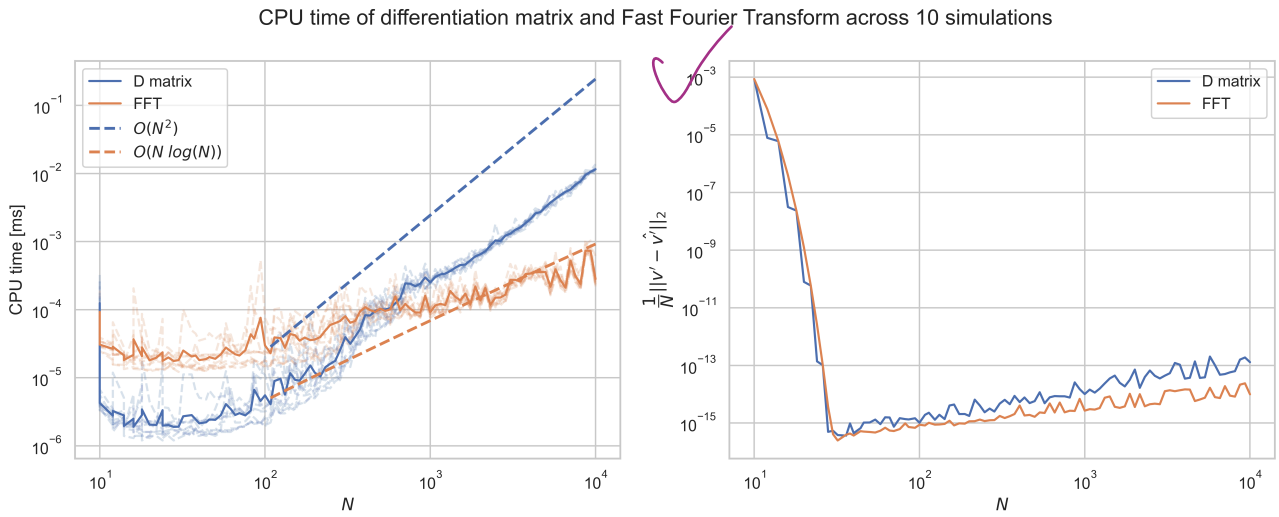

Figure 8: Comparison of CPU time of Fourier Differentiation matrix D and the FFT for derivative computation of function v for increasing N in a loglog plot.

2) Polynomial Methods

2.h) Jacobi Polynomials

The Jacobi polynomials are given as a recursion,

$$\begin{aligned} P_0^{(\alpha,\beta)}(x) &= 1 \\ P_1^{(\alpha,\beta)}(x) &= \frac{1}{2}(\alpha - \beta + (\alpha + \beta + 2)x) \\ P_n^{(\alpha,\beta)}(x) &= \frac{1}{a_{n,n-1}^{(\alpha,\beta)}}\left(\left(a_{n-1,n-1}^{(\alpha,\beta)} + x\right)P_{n-1}^{(\alpha,\beta)} - a_{n-2,n-1}^{(\alpha,\beta)}P_{n-2}^{(\alpha,\beta)}\right), \end{aligned} \tag{45}$$

where

$$\begin{aligned} a_{n-1,n}^{(\alpha,\beta)} &= \frac{2(n+\alpha)(n+\beta)}{(2n+\alpha+\beta+1)(2n+\alpha+\beta)} \\ a_{n,n}^{(\alpha,\beta)} &= \frac{\alpha^2 - \beta^2}{(2n+\alpha+\beta+2)(2n+\alpha+\beta)} \\ a_{n+1,n}^{(\alpha,\beta)} &= \frac{2(n+1)(n+\alpha+\beta+1)}{(2n+\alpha+\beta+2)(2n+\alpha+\beta+1)} \end{aligned} \tag{46}$$

for $n \geq 0$, and $a_{-1,0}^{(\alpha,\beta)} = 0$.

To calculate the Jacobi polynomials, we have to calculate the recursion with the correct a-values. The code, `jacobi_p`, is implemented in such a way that the polynomials are calculated "from the bottom up", to prevent the costs of recursion. To check our implementation, we plot the first six Lagrange and Chebyshev polynomials, see Figure 9. For the Chebyshev polynomials, we note that they are scaled Jacobi polynomials,

$$T_n^{(-\frac{1}{2},-\frac{1}{2})}(x) = \frac{\Gamma(n+1)\Gamma\left(\frac{1}{2}\right)}{\Gamma\left(n+\frac{1}{2}\right)}P_n^{(-\frac{1}{2},-\frac{1}{2})}(x). \tag{47}$$

The plots align with the plots given in lectures, so we assume that our correction is correct.

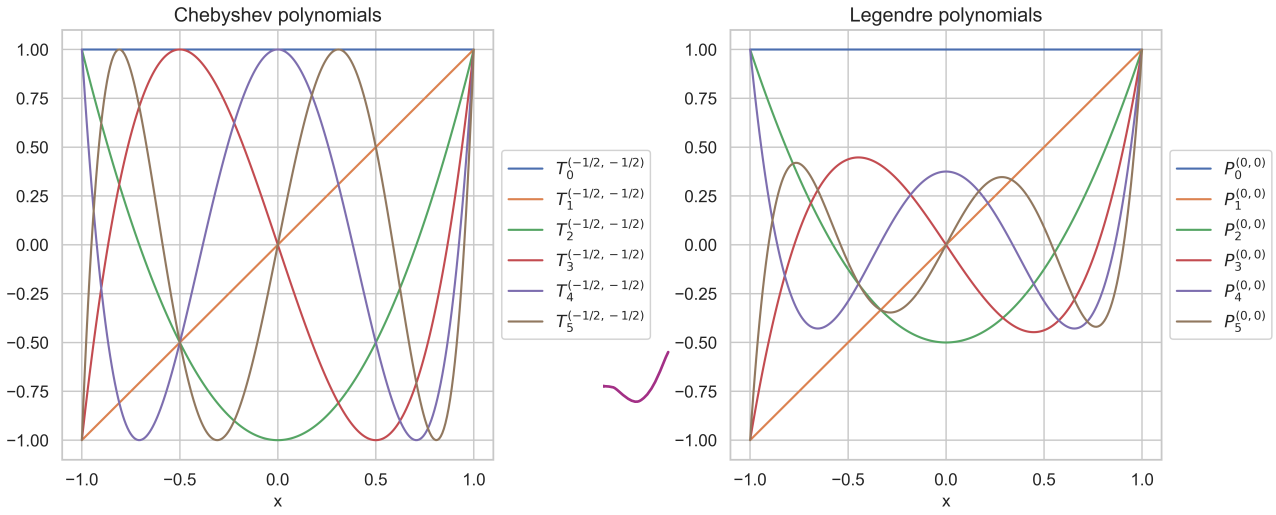

Figure 9: Chebyshev polynomials $T_n^{(-\frac{1}{2},-\frac{1}{2})}(x)$ and Legendre polynomials $P_n^{(0,0)}(x)$ for $n \in \{0, ..., 5\}$ calculated by the `jacobi_p` routine. Note that for the Chebyshev polynomial $P_n^{(-\frac{1}{2},-\frac{1}{2})}(x)$ has been scaled.

2.i) Numerical Experiments

We calculate the abscissas x_j using translated versions of the functions JacobiQP and JacobiGL, given on Learn. We then calculate the first $K = 200$ coefficients as follows,

$$\tilde{u}_k = \frac{1}{\gamma_k} \sum_{j=0}^{N} u(x_j)\varphi_k(x_j)w_j, \qquad \text{for } k = 0, ..., K, \tag{48}$$

where in our case $u(x) = \frac{1}{(2-\cos(\pi(x+1))}$, $x \in [-1, 1]$ and we use the Legendre polynomials, i.e. $\varphi_k(x) = P_k^{0,0}(x)$. Notice that we have redefined u such that it covers the correct interval. A plot of the calculated coefficients can be seen in Figure 10.

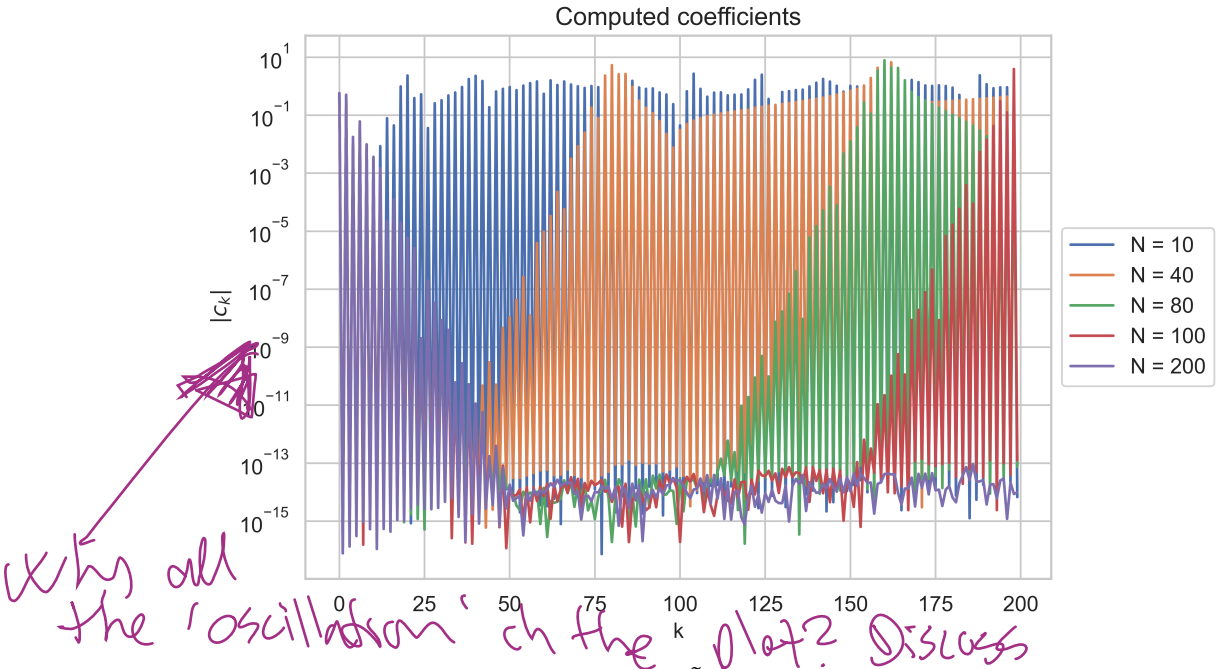

Figure 10: Plot of calculated coefficients $|c_k| = |\tilde{f}_k|$ for $k = 0, ..., K$ and varying N in a semi-log plot.

We see that for $N = K$ the coefficients behave as expected, where they decay towards 0 as k grows. For $N < K$, we see that the coefficients follows the expected pattern until the error grows again. We calculate the interpolant as

$$I_N u(x) = \sum_{k=0}^{K} \hat{u}_k \, \varphi_{k(x)}, \tag{49}$$

which allows us to examine the interpolation error. Our plot of the error, see Figure 11, shows the error we expect from the coefficients, namely that we only get satisfactory results when $N = K$.

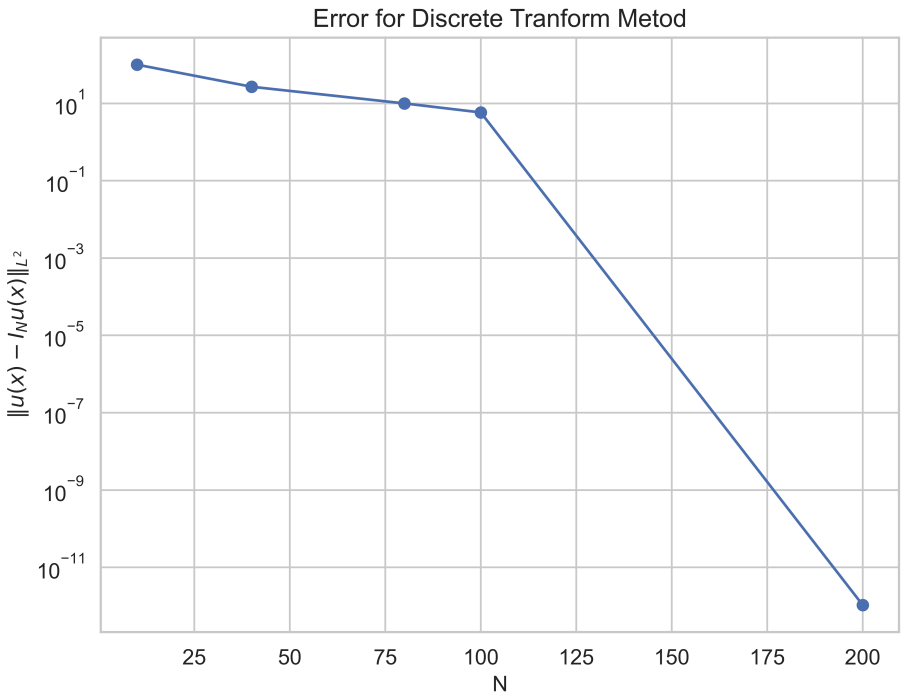

Figure 11: Truncation error $\|u(x) - I_N u(x)\|_2$ of the Discrete Polynomial Transform for the function $\tilde{u}(x) = \frac{1}{(2-\cos(\pi(x+1))}$ using 200 coefficients and for varying grid sizes in a semi-log plot.

To explain this error, we look at how we calculate the coefficients,

$$\begin{aligned}
\tilde{f}_k &= \frac{(f, \varphi_k)_N}{\|\varphi_k\|_N^2} = \frac{\left(\sum_{m=0}^{\infty} \hat{f}_m \varphi_m, \varphi_k\right)_N}{\|\varphi_k\|_N^2} \\
&= \frac{\left(\sum_{m=0}^{N} \hat{f}_m \varphi_m, \varphi_k\right)_N}{\|\varphi_k\|_N^2} + \frac{\left(\sum_{m=N+1}^{\infty} \hat{f}_m \varphi_m, \varphi_k\right)_N}{\|\varphi_k\|_N^2} \\
&= \frac{1}{\|\varphi_k\|_N^2} \sum_{m=0}^{N} \hat{f}_m (\varphi_m, \varphi_k)_N + \frac{1}{\|\varphi_k\|_N^2} \sum_{m=N+1}^{\infty} \hat{f}_m (\varphi_m, \varphi_k)_N
\end{aligned} \tag{50}$$

For $k \leq N$, we get

$$\tilde{f}_k = \hat{f}_k + \frac{1}{\|\varphi_k\|_N^2} \sum_{m=N+1}^{\infty} \hat{f}_m (\varphi_m, \varphi_k)_N, \tag{51}$$

due to the orthogonality of the φ-basis. This means that we have the exact coefficient plus some error. We know that $(\varphi_m, \varphi_k)_N \in \mathbb{P}^{m+k}$ and that the quadrature is exact for any $f \in \mathbb{P}^{2N-1}$, so we get errors from the sum term when $m + k > 2N - 1$. This is also true for $k > N$, but in this case, we no longer have the exact term and more terms will have a too high degree, resulting in large errors. This matches the image we see in Figure 10.

2.j) Generalized Vandermonde Matrix

To construct the generalized Vandermonde matrix, we find polynomials and abscissas as in the previous exercise and gather in a matrix

$$\mathcal{V} = \begin{bmatrix} \varphi_1(x_1) & \varphi_2(x_1) & \dots & \varphi_N(x_1) \\ \varphi_1(x_2) & \varphi_2(x_2) & \dots & \varphi_N(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_1(x_N) & \varphi_2(x_N) & \dots & \varphi_N(x_N) \end{bmatrix} \tag{52}$$

To calculate the Lagrange polynomials on a uniform grid, $\tilde{x}_i$ for $i = 0, \dots, 99$, we solve the equation

$$\mathcal{V}^\top \boldsymbol{h}(\tilde{x}_i) = \boldsymbol{\varphi}(\tilde{x}_j) \tag{53}$$

for each $i = 0, \dots, 99$ in the uniform grid. The resulting polynomials can be seen in Figure 12. We see that the Lagrange polynomials evaluated at the abscissas take the expected values of 1 or 0, specifically that $h_j(x_i) = \delta_{ij}$, indicating that these are in fact the correct polynomials.

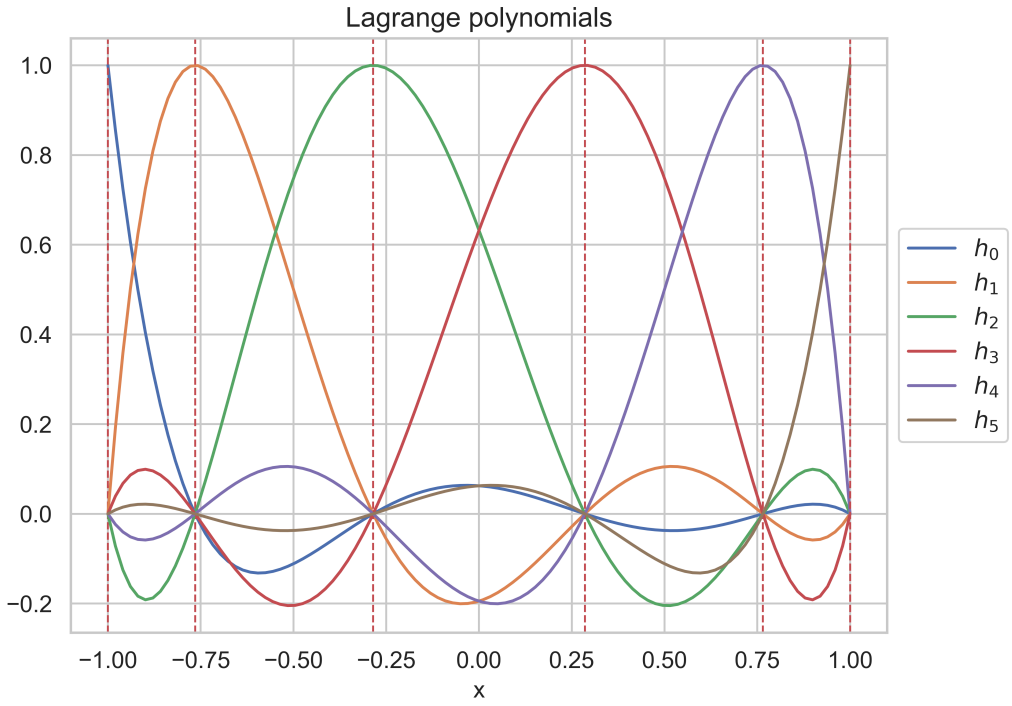

Figure 12: Lagrange Polynomials $h_j(x)$ for $N = 6$. Note that $h_j(x_i) = \delta_{ij}$ where the abscisses x_i are demarked by the dashed vertical lines.

We then consider the duality between the nodal and the modal expansions for some function f,

$$\boldsymbol{f} = \mathcal{V}\hat{\boldsymbol{f}} \tag{54}$$

where $\boldsymbol{f} = (f(\tilde{x}_0), f(\tilde{x}_1), \dots, f(\tilde{x}_{99}))^\top$ are the nodal values, and $\hat{\boldsymbol{f}}$ denotes the modal values, see Equation 48. We can now approximate any function using the nodal values and the calculated Lagrange polynomials as our basis.

2.j.1) Approximating a Function Using Nodal Expansion

We do this for the function $v(x) = \sin(\pi x)$, $x \in [-1, 1]$. We first calculate the modal values as in h) and the Lagrange polynomials as above. Then we can calculate the interpolant,

$$I_N v(x) = \sum_{j=0}^{N-1} (\mathcal{V}\hat{\boldsymbol{v}})_j \, h_j(x). \tag{55}$$

To inspect the result, we calculate the interpolant for varying N and plot both the interpolant and the errors, see Figure 13. Looking at the plots of the approximations, we see that we quite quickly get satisfying results (in the eyeball-norm). This is confirmed by the error plot, where we see that we reach errors of the size of the machine precision using as little as 21 grid points. We see a convergence that looks like the spectral convergence, we would expect for the DPT method, which makes sense, as the two expansions are equal.

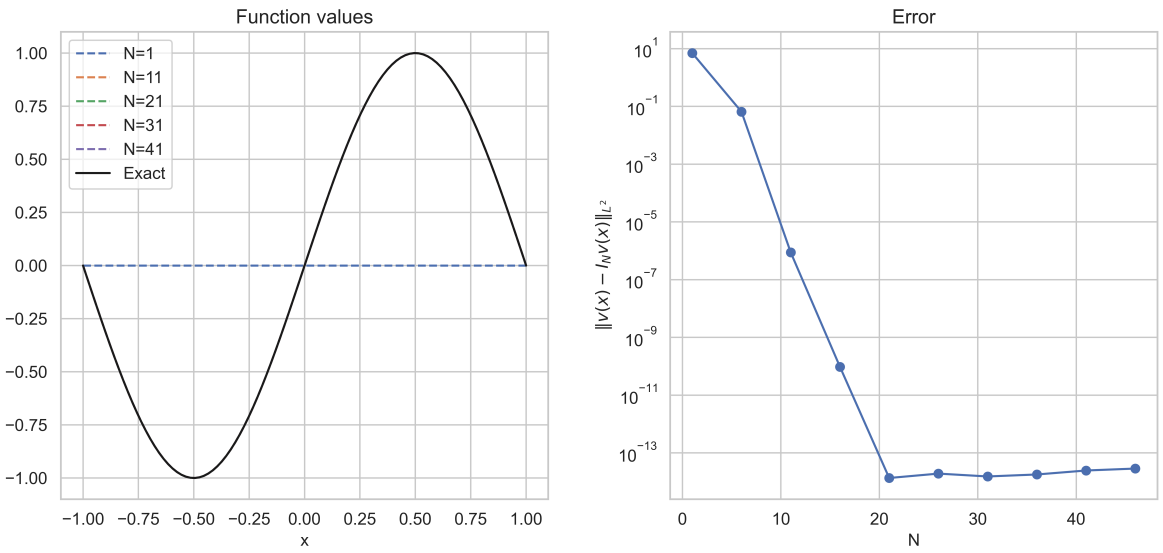

Figure 13: Plot of the approximation of $v(x)$ for $x \in [-1, 1]$ for a selection of values of N together with a plot of the truncation error $\|v(x) - I_N v(x)\|_2$ in a semi-log plot.

2.j.2) Moving Outside Interval

If we consider the expansion outside of the interval of our Jacobi polynomials, e.g. $x \in [-1.5, 1.5]$, the picture changes. We define the abscissas as before and do everything in the exact same way, except that we now do it for 150 uniform grid points $\tilde{x}_i \in [-1.5, 1.5]$. We change the number of grid points to ensure that the distance between each pair is the same as before.

We inspect the results like before, see Figure 14. Considering the plots of the interpolant, we see that errors occur outside $[-1, 1]$ and that the interpolant to grow when moving further away. Looking at the plot of the error, we see that the error indeed grows rapidly for growing N after $N = 21$.

By definition, the Jacobi polynomials only solve the Sturm-Liouville problem for $x \in [-1, 1]$ and since the abscissas, coefficients and Lagrange polynomials are based on these, it makes sense that the interpolant fails to be exact outside the interval.

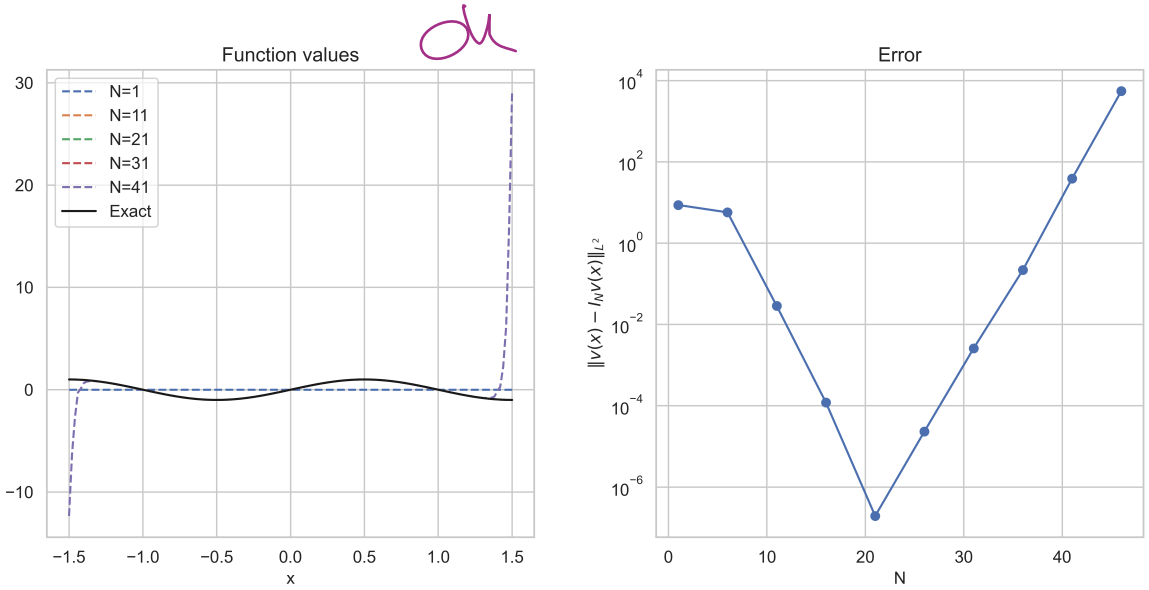

Figure 14: Plot of the approximation of $v(x)$ for $x \in [-1.5, 1.5]$ for a selection of values of N together with a plot of the truncation error $\|v(x) - I_N v(x)\|_2$ in a semi-log plot.

2.k) Derivative of Jacobi Polynomials

In order to further leverage the power of the Vandermonde matrices, we employ a routine for finding the first derivatives of the Jacobi polynomials, which are given by [1, S.15]:

$$\frac{\mathrm{d}}{\mathrm{d}x} P_n^{(\alpha,\beta)}(x) = \frac{1}{2}(\alpha + \beta + n + 1) P_{n-1}^{(\alpha+1,\beta+1)}(x) \tag{56}$$

Where $P_n^{(\alpha,\beta)}$ are the Jacobi Polynomials as given in Equation 45. We implement a single routine which returns an $(n+1) \times m$ matrix where n is the order of the highest-order polynomial and m is the number of nodes in $[-1, 1] \ni x$ at which the derivatives are evaluated such. As such, each column represents a Jacobi polynomial evaluated at the nodes x_i.

This is implemented in Python as:

```python
def grad_jacobi_p(x: npt.NDArray, alpha: float, beta: float, n: int) -> npt.NDArray:
    """
    Computes the gradient of the first `n+1` Jacobi polynomials at nodes `x`.
    Reflects L2, slide 15.

    Arguments:
        x: Points at which to evaluate the gradients, shape (m,)
        alpha: Jacobi parameter, $α > -1$
        beta: Jacobi parameter, $β > -1$
        n: Highest order polynomial to compute (must be positive)
    Returns: Array of shape (m, n+1) where each column corresponds to the gradient of a Jacobi polynomial
    """
    grad_p = np.empty((len(x), n + 1))

    for i in range(n + 1):
        if i == 0:
            p_i = np.zeros_like(x)
        else:
            p_i = jacobi_p(x, alpha + 1, beta + 1, i - 1)[:, i - 1]

        coeff = 1 / 2 * (alpha + beta + i + 1)
        grad_p[:, i] = coeff * p_i

    return grad_p
```

Where the definition of the function `jacobi_p` has been relegated to Appendix A.1.

In order to gauge the correctness of our implementation, we plot the derivatives of the first 6 Legendre polynomials as shown in Figure 15.

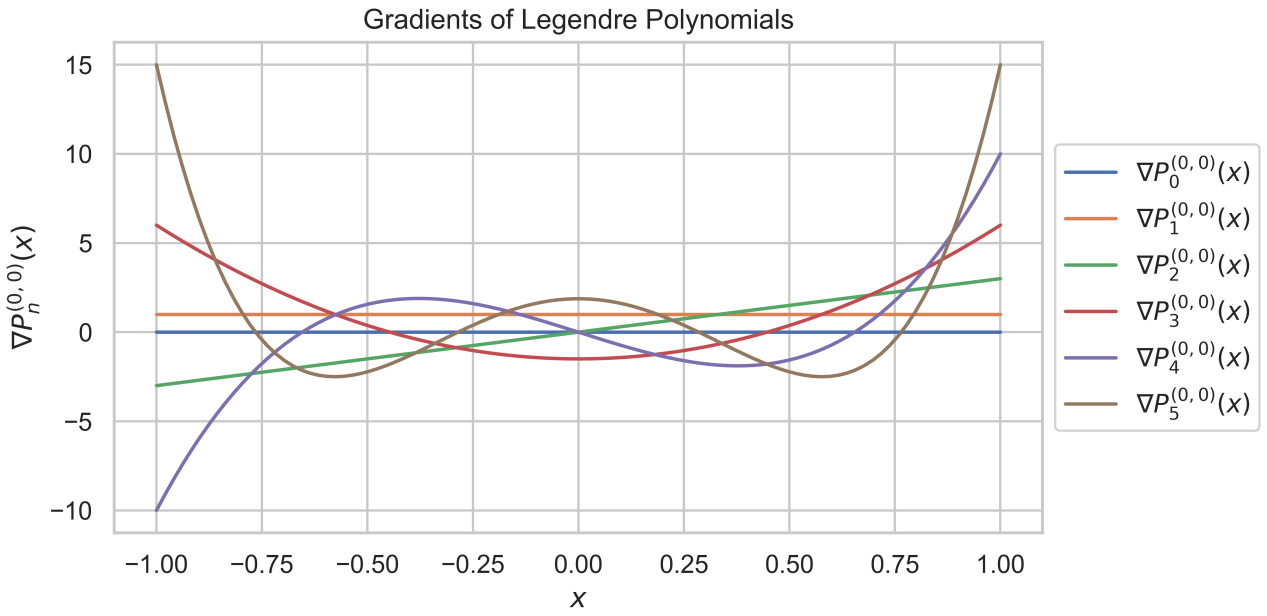

Figure 15: Gradients of the first 6 Legendre polynomials evaluated over $x \in [-1, 1]$ using the `grad_jacobi_p` routine.

We may use this function to construct a generalized Vandermonde matrix $\mathcal{V}_x$ enabling the transformation between the nodal and modal coefficients, given by f and $\hat{f}$ respectively, of a polynomial expansion of the first derivative of a function.

Using the same definition as in Section 2.j, we let $\mathcal{V}_x$ be given by:

$$\mathcal{V}_x := \begin{bmatrix} \varphi_1'(x_1) & \varphi_2'(x_1) & \dots & \varphi_N'(x_1) \\ \varphi_1'(x_2) & \varphi_2'(x_2) & \dots & \varphi_N'(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_1'(x_N) & \varphi_2'(x_N) & \dots & \varphi_N'(x_N) \end{bmatrix} \qquad \mathcal{V}_{x,ij} = \left.\frac{\mathrm{d}\varphi_j}{\mathrm{d}x}\right|_{x=x_i} \tag{57}$$

We begin by considering the N-nodal expansion of the first derivative of $f(x)$ in the Lagrange basis, where $\boldsymbol{f} = f(x_i)$ for $i \in 0, ..., N$:

$$\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}x} \approx \frac{\mathrm{d}}{\mathrm{d}x}\left(\sum_{j=0}^{N} f_j h_j(x_i)\right) = \sum_{j=0}^{N} f_j \left.\frac{\mathrm{d}h_j}{\mathrm{d}x}\right|_{x=x_i} \tag{58}$$

Where we introduce a *differentiation matrix*, $\mathcal{D}$ characterised by its operation on the nodal coefficients f_j:

$$\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}x}(x_i) = \mathcal{D}\boldsymbol{f} \tag{59}$$

$$\mathcal{D}_{ij} := \left.\frac{\mathrm{d}h_j}{\mathrm{d}x}\right|_{x=x_i} \tag{60}$$

Problematically, the Lagrange polynomial construction is *global* and as such the evaluation of $\frac{\mathrm{d}h_j}{\mathrm{d}x}$ at nodes x_i requires information about all nodes, which may be computationally expensive. Instead, we would like to evaluate the derivative in the *modal* basis, which is *local* in the sense that a good modal representation will depend only on nearby 'neighbours', as seen through the recurrence relation of the Jacobi polynomials in Equation 45.

As such, we repeat the treatment above for the N-modal representation of the first derivative in a polynomial basis given by $\varphi_j(x)$:

$$\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}x} \approx \frac{\mathrm{d}}{\mathrm{d}x}\left(\sum_{j=0}^{N} \hat{\boldsymbol{f}}_j \varphi_j(x_i)\right) = \sum_{j=0}^{N} \hat{\boldsymbol{f}}_j \left.\frac{\mathrm{d}\varphi_j}{\mathrm{d}x}\right|_{x=x_i} = \mathcal{V}_x \hat{\boldsymbol{f}} \tag{61}$$

As such, by recalling $\boldsymbol{f} = \mathcal{V}\hat{\boldsymbol{f}}$ and using Equation 60 and Equation 61 we find the relation:

$$\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}x}\mathcal{D}\boldsymbol{f} = \mathcal{D}\mathcal{V}\hat{\boldsymbol{f}} = \mathcal{V}_x\hat{\boldsymbol{f}} \tag{62}$$

$$\mathcal{D} = \mathcal{V}_x\mathcal{V}^{-1} \tag{63}$$

Where $\mathcal{V}_x$ may be evaluated numerically using the function `grad_jacobi_p`.

We exercise our approach by solving the problem of evaluating the first derivative of function $v(x)$ given by:

$$v(x) = \exp(\sin(\pi x)) \tag{64}$$

Where the domain of the function is given as $x \in [-1, 1]$. The analytical derivative of Equation 64 becomes:

$$\frac{\mathrm{d}v}{\mathrm{d}x} = \pi\cos(\pi x)\exp(\sin(\pi x)) \tag{65}$$

Using $N =$ nodes/modes we obtain the following following:

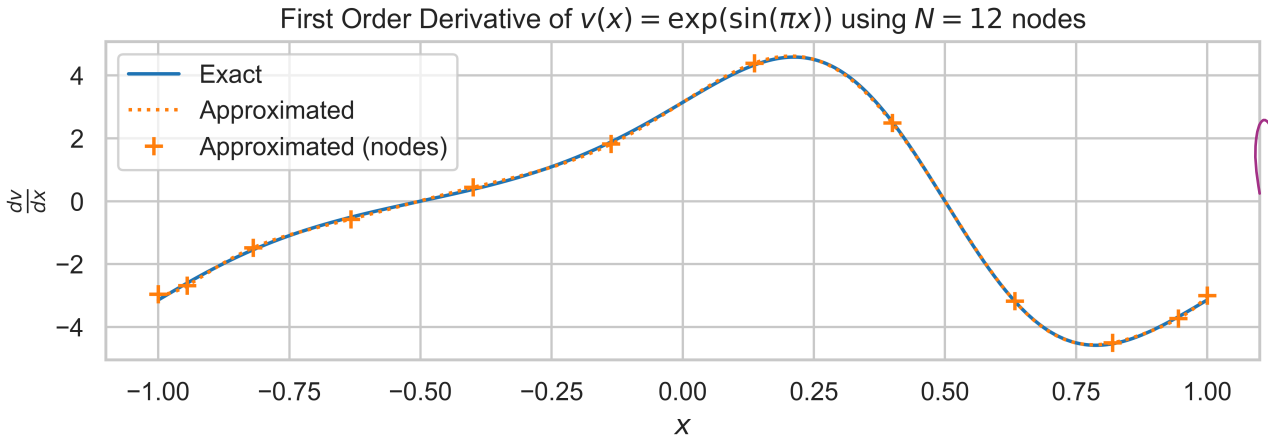

Figure 16: First order derivative of $v(x) = \sin(\pi x)$ evaluated on $x \in [-1, 1]$ using the Vandermonde-derived differentiation matrix $\mathcal{D}$ with $N = 12$ nodes.

Lastly, we seek to evaluate the convergence of this method by computing the L^2-norm of the defect $\frac{\mathrm{d}v}{\mathrm{d}x}(x_i) - \mathcal{D}\boldsymbol{f}$ with $\boldsymbol{f}$ being the nodal coefficients of $v(x_i)$.

This may be done by considering the definition of the L^2 norm:

$$\begin{aligned}
\|f\|^2 &= \int_\Omega |f(x)|^2 \,\mathrm{d}x \\
&\approx \int_\Omega \boldsymbol{f}^\mathsf{T}\boldsymbol{h}(x)(\boldsymbol{f}^\mathsf{T}\boldsymbol{h}(x))^\mathsf{T} \,\mathrm{d}x \\
&= \int_\Omega \boldsymbol{f}^\mathsf{T}\boldsymbol{h}(x)\boldsymbol{h}^\mathsf{T}(x)\boldsymbol{f} \,\mathrm{d}x \\
&= \boldsymbol{f}^\mathsf{T}\left(\int_\Omega (\mathcal{V}^\mathsf{T})^{-1}\hat{\boldsymbol{\varphi}}(x)\hat{\boldsymbol{\varphi}}^\mathsf{T}(x)\mathcal{V}^{-1} \,\mathrm{d}x\right)\boldsymbol{f} \\
&= \boldsymbol{f}^\mathsf{T}(\mathcal{V}^\mathsf{T})^{-1}\left(\int_\Omega \hat{\boldsymbol{\varphi}}(x)\hat{\boldsymbol{\varphi}}^\mathsf{T}(x) \,\mathrm{d}x\right)\mathcal{V}^{-1}\boldsymbol{f} \\
&= \boldsymbol{f}^\mathsf{T}(\mathcal{V}\mathcal{V}^\mathsf{T})^{-1}\boldsymbol{f}
\end{aligned} \tag{66}$$

Where Ω is the domain of the basis functions and the inner product of the basis functions evaluates to unity under the condition of orthonormality.

We may then define a *mass matrix*, $\mathcal{M}$ as:

$$\mathcal{M} := \left(\mathcal{V}\mathcal{V}^{\mathsf{T}}\right)^{-1} \tag{67}$$

Such that the L^2 norm may be evaluated as:

$$\|f\| \approx \sqrt{\boldsymbol{f}^{\mathsf{T}}\mathcal{M}\boldsymbol{f}} \tag{68}$$

We note that the function `jacobi_p` in Appendix A.1 is not normalized, which is handled by implementing `jacobi_p_normalisation_const` and `jacobi_p_normalised` which may be found in Appendix A.2.

They correspond to the normalized Jacobi polynomials, $\hat{P}_n^{(\alpha,\beta)}$ [1, S.19]:

$$\hat{P}_n^{(\alpha,\beta)} = \frac{1}{\sqrt{\gamma_n^{(\alpha,\beta)}}} P_n^{(\alpha,\beta)} \tag{69}$$

And associated normalisation constant $\gamma_n^{(\alpha,\beta)}$ [1, S.11] given by:

$$\gamma_n^{(\alpha,\beta)} = 2^{\alpha+\beta+1} \frac{\Gamma(n+\alpha+1)\Gamma(n+\beta+1)}{n!(2n+\alpha+\beta+1)\Gamma(n+\alpha+\beta+1)} \tag{70}$$

Using these normalised polynomials to construct $\mathcal{V}$ and subsequently $\mathcal{M}$ allows us to easily evaluate the L^2-norm of the defects as a function of the number of modes, N, as shown in Figure 17.

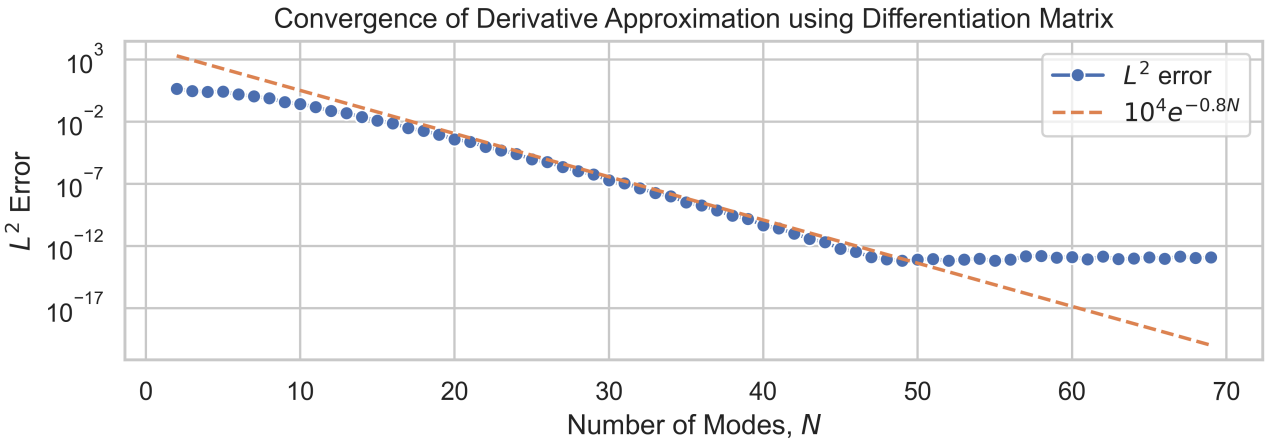

Figure 17: Convergence for approximation of first derivative of $v(x) = \sin(\pi x)$ using differentiation matrix with Legendre polynomial basis in a semi-log plot.

The expected convergence using a polynomial expansion will depend on how well-suited the polynomial series is to the modelled function, where particularly the *smoothness* of the function may set a bound on the rate of convergence.

The Legendre polynomials leveraged here belong to the class of *ultraspherical polynomials*, $P_n^{(\alpha,\alpha)}$, while the function $v(x)$ belongs in the smoothness class $C^\infty([-1,1])$ by inspection. As such, borrowing the result from Hesthaven, Gottlieb & Gottlieb (2007) presented in [1, S.23-24] we find that we would expect to find *spectral convergence*, that is, the defect $d = \frac{\mathrm{d}v}{\mathrm{d}x}(x_1) - \mathcal{D}\boldsymbol{f}$, should follow:

$$\|d\| \leq Ae^{-BN} \tag{71}$$

Where $A, B \in \mathbb{R}_+$ are some constants dependant on the problem. This should form a straight line in a semi-log plot, as observed in Figure 17, where we find a decent agreement with $B = 0.8$ as indicated by the dashed orange line.

We note that machine precision ($\approx 10^{-13}$) is achieved with just $N = 49$ modes.

We may conclude that the outlined method performs exceedingly well for sufficiently smooth functions, as evidenced by its performance on the sine function.

2.l) Mass Matrix

In order to extend the domain Ω of the norm $\|f\|$ from $[-1, 1]$ to an arbitrary interval $[a, b]$ where $a < b < \infty$, we simply introduce an affine coordinate transformation into our construction of the mass matrix $\mathcal{M}$ which scales and centers the domain:

$$\bar{x}(x) = \frac{b-a}{2}x + \frac{a+b}{2} \tag{72}$$

Where x is are the coordinates in the domain $\Omega = [-1, 1]$ and $\bar{x}$ the coordinates in the physical domain $\bar{\Omega} = [a, b]$.

We compute the differentials:

$$\mathrm{d}\bar{x} = \frac{b-a}{2}\,\mathrm{d}x \tag{73}$$

And again consider the definition of the L^2 inner product as done previously in Equation 66 and introduce a change of variables:

$$\begin{aligned} \|f\|^2 &= \int_{\bar{\Omega}} |f(\bar{x})|^2 \,\mathrm{d}\bar{x} \\ &= \int_{\Omega} |f(x)|^2 \,\frac{b-a}{2}\,\mathrm{d}x \\ &= \frac{b-a}{2}\int_{\Omega} |f(x)|^2 \,\mathrm{d}x \\ &= \frac{b-a}{2}\boldsymbol{f}^{\mathsf{T}}(\mathcal{V}\mathcal{V}^{\mathsf{T}})^{-1}\boldsymbol{f} \end{aligned} \tag{74}$$

As such, we simply need to transform the coordinates obtained from our root-finding routine into the physical domain using Equation 72 and scale the mass matrix $\mathcal{M}$ by a factor of $\frac{b-a}{2}$ in order to enable the L^2-norm to be found for any arbitrary domain $\bar{\Omega} = [a, b]$.

We employ this technique on the interval $\bar{\Omega} = [0, 2]$ and the following functions:

$$\begin{aligned} u_1(x) &:= 1 \\ u_2(x) &:= \sin(x) \end{aligned} \tag{75}$$

Whose squares we additionally integrate analytically to validate the correctness of our implementation:

$$\int_a^b |u_1(x)|^2 \,\mathrm{d}x = \int_a^b 1 \,\mathrm{d}x = b - a \tag{76}$$

$$\int_a^b |u_2(x)|^2 \,\mathrm{d}x = \int_a^b \sin(x) \,\mathrm{d}x = \left[\frac{x}{2} - \frac{\sin(2x)}{4}\right]_a^b \tag{77}$$

We again use the orthonormal Jacobi polynomials, $\hat{P}_n^{(\alpha,\beta)}$, to determine the Vandermonde matrix to ensure the correct scaling of the mass matrix.

Using N nodes we find the following L^2-norms of $u_1(x)$ and $u_2(x)$ over $\bar{\Omega} = [0, 2]$:

Function	N	Numerical	Analytical	Error
$u_1(x) = 1$	5	1.4142135623730951	1.4142135623730951	$< 10^{-16}$
$u_2(x) = \sin(x)$	5	1.0905052416391159	1.0905047564439974	4.9×10^{-7}
$u_2(x) = \sin(x)$	9	1.0905047564439974	1.0905047564439974	5.1×10^{-15}

Table 1: Comparison of numerical approximations of the L^2 norms of two functions, $u_1(x)$, $u_2(x)$

We note that in the definition of the mass matrix above and the derivation of its use in determining the L^2 norm, we introduce an approximation only in the second step of Equation 66:

$$\begin{aligned} \|f\|^2 &= \int_\Omega |f(x)|^2 \, \mathrm{d}x \\ &\approx \int_\Omega \boldsymbol{f}^\mathsf{T} \boldsymbol{h}(x) (\boldsymbol{f}^\mathsf{T} \boldsymbol{h}(x))^\mathsf{T} \, \mathrm{d}x \end{aligned} \tag{78}$$

As such, our matrix-based integration using the mass matrix is exact under the assumption that the interpolating polynomials $\boldsymbol{h}(x)$ (and equivalently, $\hat{\boldsymbol{\varphi}}(x)$) are perfect representations of the modelled function. This will be the case if and only if the modelled function is itself a polynomial of order $q < p$ where $p = N - 1$ is the highest order of the interpolating polynomials.

As such, we find that the L^2-norm of $u_1(x) = 1$ will be determined exactly using the numerical integration for interpolants with any number of nodes N, whereas $u_2(x) = \sin(x)$ is only exact in the limit $N \to \infty$, since the sine function is the sum of an infinite series of polynomials:

$$\sin(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)!} \tag{79}$$

Thankfully, as observed in Table 1, we obtain machine precision at just $N = 9$ nodes for $u_2(x)$.

A Appendix

A.1 Jacobi Polynomial Function

Python

```python
def jacobi_p(x: npt.NDArray, alpha: float, beta: float, n: int) ->
npt.NDArray:
    """
    Evaluates first `n+1` Jacobi polynomials at points `x` with parameters `alpha`
    and `beta`.
    Reflects L2, slide 12.

    Arguments:
        x: Points at which to evaluate the polynomials, shape (m,)
        alpha: Jacobi parameter, $α > -1$
        beta: Jacobi parameter, $β > -1$
        n: Highest order polynomial to compute (must be positive)

    Returns: Array of shape (m, n+1) where each column corresponds to a Jacobi
    polynomial
    """
    assert n >= 0, "n must be non-negative"

    P = np.empty((len(x), n + 1))

    P[:, 0] = 1.0
```

```
    if n == 0:
        return P

    P[:, 1] = 1 / 2 * (alpha - beta + (alpha + beta + 2) * x)
    if n == 1:
        return P

    for k in range(1, n):
        a_nm1_n = (
            2
            * (k + alpha)
            * (k + beta)
            / ((2 * k + alpha + beta + 1) * (2 * k + alpha + beta))
        )
        a_n_n = (alpha**2 - beta**2) / (
            (2 * k + alpha + beta + 2) * (2 * k + alpha + beta)
        )
        a_np1_n = (
            2
            * (k + 1)
            * (k + alpha + beta + 1)
            / ((2 * k + alpha + beta + 2) * (2 * k + alpha + beta + 1))
        )

        P[:, k + 1] = ((a_n_n + x) * P[:, k] - a_nm1_n * P[:, k - 1]) / a_np1_n

    return P
```

A.2 Normalized Jacobi Polynomial Function

Python

```
def jacobi_p_normalisation_const(
    alpha: float, beta: float, n: int | npt.NDArray
) -> int | npt.NDArray:
    """
    Computes the normalisation constant for Jacobi polynomials.
    Reflects $γ_n^(α,β)$ from L2, slide 11.

    Arguments:
        alpha: Jacobi parameter, $α > -1$
        beta: Jacobi parameter, $β > -1$
        n: npt.ArrayLike

    Returns: normalisation constant(s) $γ$ for Jacobi polynomials $P_n$
    """
    return (
        2 ** (alpha + beta + 1)
        * (gamma(n + alpha + 1) * gamma(n + beta + 1))
        / (factorial(n) * (2 * n + alpha + beta + 1) * gamma(n + alpha + beta + 1))
    )

```

```python

def jacobi_p_normalised(
    x: npt.NDArray, alpha: float, beta: float, n: int
) -> npt.NDArray:
    """
    Convenience function to get normalized Jacobi polynomials using `jacobi_p` and
    `jacobi_p_normalisation_const`.
    """
    P = jacobi_p(x, alpha, beta, n)
    norm_const = jacobi_p_normalisation_const(alpha, beta, np.arange(n + 1))
    return P / np.sqrt(norm_const)
```

Bibliography

[1] A. P. Engsig-Karup, "02689 Advanced Numerical Methods, Lecture 2: Polynomial Methods." 2023.