1 Various Forms of DFT

1.1 The most general form of DFT

$$x_j = \frac{2\pi}{N} j, \quad j \in \mathbb{Z} \tag{1}$$

$$u(x_{j+N}) = u(x_j) \tag{2}$$

$$\tilde{u}_k = A \sum_{j=j_0}^{j_0+N-1} u(x_j) e^{-ikx_j}, \quad k \in \mathbb{Z} \tag{3}$$

$$\tilde{u}_{k+N} = \tilde{u}_k \tag{4}$$

$$u(x_j) = B \sum_{k=k_0}^{k_0+N-1} \tilde{u}_k e^{ikx_j}, \quad j \in \mathbb{Z} \tag{5}$$

$$AB = \frac{1}{N} \tag{6}$$

1.2 David's Implementing Spectral Methods for PDEs

$$x_j = \frac{2\pi}{N} j, \quad j = 0, 1, \ldots, N-1 \tag{7}$$

$$\tilde{u}_k = \frac{1}{N} \sum_{j=0}^{N-1} u(x_j) e^{-ikx_j}, \quad k = 0, 1, \ldots, N-1 \tag{8}$$

$$u(x_j) = \sum_{k=0}^{N-1} \tilde{u}_k e^{ikx_j}, \quad j = 0, 1, \ldots, N-1 \tag{9}$$

1.3 Scipy

$$x_j = \frac{2\pi}{N} j, \quad j = 0, 1, \ldots, N-1 \tag{10}$$

$$\tilde{u}_k = \sum_{j=0}^{N-1} u(x_j) e^{-ikx_j}, \quad k = 0, 1, \ldots, N-1 \tag{11}$$

$$u(x_j) = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{u}_k e^{ikx_j}, \quad j = 0, 1, \ldots, N-1 \tag{12}$$

1.4 Hussaini, Spectral Methods in Fluid Dynamics

$$x_j = \frac{2\pi}{N} j, \quad j = 0, 1, \ldots, N-1 \tag{13}$$

$$\tilde{u}_k = \frac{1}{N} \sum_{j=0}^{N-1} u(x_j) e^{-ikx_j}, \quad k = -\frac{N}{2}, \ldots, \frac{N}{2} - 1 \tag{14}$$

$$u(x_j) = \sum_{k=-N/2}^{N/2-1} \tilde{u}_k e^{ikx_j}, \quad j = 0, 1, \ldots, N-1 \tag{15}$$

1.5 Trefethen, Spectral Methods in Matlab

$$x_j = \frac{2\pi}{N} j, \quad j = 1, 2, \ldots, N \tag{16}$$

$$\tilde{u}_k = \frac{2\pi}{N} \sum_{j=1}^{N} u(x_j) e^{-ikx_j}, \quad k = -\frac{N}{2} + 1, \ldots, \frac{N}{2} \tag{17}$$

$$u(x_j) = \frac{1}{2\pi} \sum_{k=-N/2+1}^{N/2} \tilde{u}_k e^{ikx_j}, \quad j = 1, \ldots, N \tag{18}$$

1.6 Matlab

$$Y(k) = \sum_{j=1}^{n} X(j) e^{-i(k-1)x_{j-1}}, \quad k = 1, 2, \ldots, N \tag{19}$$

$$X(j) = \frac{1}{N} \sum_{k=1}^{n} Y(k) e^{i(k-1)x_{j-1}}, \quad j = 1, 2, \ldots, N \tag{20}$$

$$Y(k) = \tilde{u}_{k-1}, \quad k = 1, 2, \ldots, N \tag{21}$$

$$X(j) = u(x_{j-1}) \quad j = 1, 2, \ldots, N \tag{22}$$

1.7 DFT differentiation

Fourier interpolation:

$$I_N f(x) = \sum_{k=-N/2}^{N/2} \frac{\tilde{f}_k}{\bar{c}_k} e^{ikx} \tag{23}$$

Derivative approximation:

$$f^{(m)}(x) \approx (I_N f)^{(m)}(x) = \sum_{k=-N/2}^{N/2} \frac{(ik)^m \tilde{f}_k}{\bar{c}_k} e^{ikx} \tag{24}$$

Derivatives at the Fourier nodes:

$$(I_N f)^{(m)}(x_j) = \begin{cases} \sum\limits_{k=-N/2}^{N/2-1} (ik)^m \tilde{f}_k e^{ikx_j}, & m \text{ even} \\ \\ \sum\limits_{k=-N/2+1}^{N/2-1} (ik)^m \tilde{f}_k e^{ikx_j}, & m \text{ odd} \end{cases} \tag{25}$$

1.8 Real DFT

N even, $\{f_j : j = 0, 1, \ldots, N-1\}$ real
David's version:

$$\begin{aligned} f_j &= \frac{a_0}{2} + \sum_{k=1}^{N/2-1} [a_k \cos(kx_j) + b_k \sin(kx_j)] + \frac{(-1)^j a_{N/2}}{2} \\ &= \frac{a_0}{2} + \sum_{k=1}^{N/2-1} [a_k \cos(k\frac{2\pi j}{N}) + b_k \sin(k\frac{2\pi j}{N})] + \frac{(-1)^j a_{N/2}}{2}, \quad j = 0, 1, \ldots, N-1 \end{aligned} \tag{26}$$

This is actually the inverse real DFT. Real DFT refers to formulas of a_k and b_k.. We don't need these formulas since we can use DFT to get a_k and b_k.
DFT:

$$\begin{aligned} f_j &= \sum_{k=0}^{N-1} \tilde{f}_k e^{ikx_j} \\ &= \tilde{f}_0 + 2 \sum_{k=1}^{N/2-1} (\Re(\tilde{f}_k) \cos(kx_j) - \Im(\tilde{f}_k) \sin(kx_j)) + (-1)^j \tilde{f}_{N/2} \end{aligned} \tag{27}$$

Relation:

$$a_k = 2\Re(\tilde{f}_k), \quad b_k = -2\Im(\tilde{f}_k), \quad k = 0, 1, \ldots, N/2 \tag{28}$$

In fact, $b_0 = b_{N/2} = 0$ since $\tilde{f}_0, \tilde{f}_{N/2} \in \mathbb{R}$. To do the inverse real DFT, we again use DFT. We just need to construct:

$$\tilde{f}_k = \frac{a_k}{2} - i\frac{b_k}{2}, \quad k = 0, 1, ..., \frac{N}{2}, \tag{29}$$

$$\tilde{f}_k = F^*_{N-k}, \quad k = \frac{N}{2} + 1, ..., N-1 \tag{30}$$

2 Discrete Cosine Transform (DCT)

Gauss-Lobatto points underlie the DCT and the Chebyshev transform:

$$x_j = \cos\frac{j\pi}{N}, \quad j = 0, 1, \ldots, N \tag{31}$$

Input $\{f_j : j = 0, 1, \ldots, N\}$

$$f_j = \sum_{k=0}^{N} \frac{a_k}{\bar{c}_k} \cos\left(k\frac{\pi j}{N}\right) = \sum_{k=0}^{N} \frac{a_k}{\bar{c}_k} \cos(kx_j), \tag{32}$$

$$a_k = \frac{2}{N}\sum_{j=0}^{N} \frac{f_j}{\bar{c}_j} \cos\left(k\frac{\pi j}{N}\right) = \frac{2}{N}\sum_{j=0}^{N} \frac{f_j}{\bar{c}_j} \cos(kx_j), \tag{33}$$

$$\bar{c}_k = \begin{cases} 2, & k = 0, N \\ 1, & k = 1, 2, \ldots, N-1 \end{cases} \tag{34}$$

To reduce computation cost, assume N is even and let

$$e_j = \frac{1}{2}(f_j + f_{N-j}) - (f_j - f_{N-j})\sin\frac{j\pi}{N}, \quad j = 0, 1, \ldots, N-1. \tag{35}$$

Plugging the inverse DCT for f_j gives

$$e_j = \frac{a_0}{2} + \sum_{k=1}^{N/2-1} [\cos(2k\frac{\pi}{N}j)a_{2k} + \sin(2k\frac{\pi}{N}j)(a_{2k+1} - a_{2k-1})] + \frac{a_N}{2}(-1)^j. \tag{36}$$

Real DFT of $\{e_j\}$:

$$e_j = \frac{\bar{a}_0}{2} + \sum_{k=1}^{N/2-1} [\cos(2k\frac{\pi}{N}j)\bar{a}_k + \sin(2k\frac{\pi}{N}j)\bar{b}_k] + \frac{a_{N/2}}{2}(-1)^j. \tag{37}$$

By comparison

$$a_0 = \bar{a}_0, \quad a_N = \bar{a}_{N/2} \tag{38}$$
$$a_{2k} = \bar{a}_k, \tag{39}$$
$$a_{2k+1} - a_{2k-1} = \bar{b}_k \quad k = 1, \ldots, N/2-1 \tag{40}$$
$$\tag{41}$$

a_1 is calculated directly:

$$a_1 = \frac{2}{N}\sum_{j=0}^{N} \frac{f_j}{\bar{c}_j}\cos\left(\frac{\pi j}{N}\right). \tag{42}$$

The DCT and its inverse follow the same pattern:

$$DCT: \quad output_k = \frac{2}{N}\sum_{j=0}^{N} \frac{input_j}{\bar{c}_j}\cos\left(\frac{\pi kj}{N}\right), \tag{43}$$

$$invDCT: \quad output_k = \sum_{j=0}^{N} \frac{input_j}{\bar{c}_j}\cos\left(\frac{\pi kj}{N}\right), \tag{44}$$

where in the second equation we have swapped j and k in the original formula. Now we can see that both relation are the same up to a constant factor $2/N$. Therefore, once we get a subroutine to do DCT, we can use the same program and multiply the result by $N/2$ to get the inverse DCT.

In scipy, the type-1 DCT is

$$\begin{aligned} a_k &= 2\sum_{j=0}^{N} \frac{f_j}{\bar{c}_j}\cos\left(\frac{\pi kj}{N}\right) \\ &= f_0 + 2\sum_{j=0}^{N-1} f_j \cos\left(\frac{\pi kj}{N}\right) + (-1)^k f_N, \end{aligned} \tag{45}$$

which is N times our DCT.

3 Chebyshev Transformation (ChebT)

3.1 The transformations

Target function:

$$f(x), x \in [-1, 1] \tag{46}$$

Gauss-Lobatto points:

$$x_j = \cos\frac{j\pi}{N}, \quad j = 0, 1, \dots, N \tag{47}$$

Chebyshev interpolation:

$$F(x) = \sum_{k=0}^{N} \tilde{f}_k T_k(x), \tag{48}$$

$$F(x_j) = f(x_j), \quad j = 0, 1, \dots, N \tag{49}$$

$$f_j = f(x_j) = F(x_j) = \sum_{k=0}^{N} \tilde{f}_k T_k(x_j) = \sum_{k=0}^{N} \tilde{f}_k \cos(k\frac{j\pi}{N}). \tag{50}$$

Comparing with DCT:

$$f_j = \sum_{k=0}^{N} \frac{a_k}{\bar{c}_k} \cos\left(k\frac{\pi j}{N}\right) \tag{51}$$

Thus,

$$\tilde{f}_k = \frac{a_k}{\bar{c}_k}, \qquad k = 0, 1, \dots, N \tag{52}$$

Therefore, to get the Chebyshev coefficients $\tilde{f}_k$, we only need to do DCT, and divide the first and the last coefficients by 2.

To do the inverse Chebyshev transform, again we compare it with the inverse DCT:

$$f_j = \sum_{k=0}^{N} \tilde{f}_k T_k(x_j) = \sum_{k=0}^{N} \tilde{f}_k \cos(k\frac{j\pi}{N}), \tag{53}$$

$$f_j = \sum_{k=0}^{N} \frac{a_k}{\bar{c}_k} \cos\left(k\frac{\pi j}{N}\right). \tag{54}$$

Thus, we multiply $\tilde{f}_0$ and $\tilde{f}_N$ by 2, and then use the inverse DCT.

3.2 Derivatives

$$f^{(1)}(x) \approx \frac{\mathrm{d}}{\mathrm{d}x} \sum_{k=0}^{N} \tilde{f}_k T_k(x) = \sum_{k=0}^{N} \tilde{f}_k^{(1)} T_k(x). \tag{55}$$

Since the interpolating polynomial's degree is reduced by 1 by differentiation, we have $\tilde{f}_N^{(1)} = 0$. Since no terms for higher order polynomials exist, we have $\tilde{f}_{N+1}^{(1)} = 0$. $\tilde{f}_k^{(1)}$ is calculated recursively, from $\tilde{f}_{N-1}^{(1)} = 0$ to $\tilde{f}_0^{(1)} = 0$:

$$\tilde{f}_k^{(1)} = \frac{1}{c_k}(\tilde{f}_{k+2}^{(1)} + 2(k+1)\tilde{f}_{k+1}). \tag{56}$$

Once we get $\{\tilde{f}_k^{(1)}\}$, we use the backward ChebyT to find approximation to $\{f^{(1)}(x_j) : j = 0, 1, \dots, N\}$

3.3 Derivatives, a method of Trefethen

$$f(x): \text{ target function} \tag{57}$$

$$x_j = \cos\frac{\pi}{N}j, \quad j = 0, 1, \dots, N \tag{58}$$

$$f_j = f(x_j), \tag{59}$$

$$\text{Goal: Find } \frac{\mathrm{d}f}{\mathrm{d}x}(x_j) \tag{60}$$

Change of variable:

$$x = \cos(\theta), \tag{61}$$

$$\theta_j = \frac{\pi}{N} j, \quad j = 0, 1, \ldots, N \tag{62}$$

$$f(x) \to f(\theta) \tag{63}$$

$\{\theta_j : j = 0, 1, \ldots, N\}$ are part of the Fourier nodes, so we make an extension:

$$\theta_j = \frac{\pi}{N} j, \quad j = 0, 1, \ldots, 2N - 1. \tag{64}$$

Then we make an even extension of $\{f_j : j = 0, 1, \ldots, N\}$:

$$fj = f_{2N-j}, \tag{65}$$

$$f_j : j = 0, 1, \ldots, 2N - 1. \tag{66}$$

Fourier interpolation of $f(\theta)$:

$$I_{2N} f(\theta) = \sum_{k=-N}^{N} \frac{\tilde{f}_k}{\bar{c}_k} e^{ik\theta_j}, \tag{67}$$

where

$$\bar{c}_k = \begin{cases} 2, & k = \pm N \\ 1, & \text{otherwise} \end{cases} \tag{68}$$

Derivatives at Fourier nodes:

$$I_{2N} f^{(1)}(\theta_j) = \sum_{k=-N+1}^{N-1} ik \tilde{f}_k e^{ik\theta_j}. \tag{69}$$

Derivatives wrt x:

$$\frac{\mathrm{d}f}{\mathrm{d}x}(x_j) = \frac{\frac{\mathrm{d}f}{\mathrm{d}\theta}(\theta_j)}{\frac{\mathrm{d}x}{\mathrm{d}\theta}\theta_j} \approx \frac{I_{2N} f^{(1)}(\theta_j)}{-\sin(\theta_j)}, \quad j = 1, \ldots, N - 1 \tag{70}$$

This formula does not apply to θ_0 and θ_N since the denominator is zero at these nodes. The derivatives at these nodes require special treatment.

$$f_j \in \mathbb{R}, f_j = f_{2N-j} \Rightarrow \tilde{f}_k \in \mathbb{R}, \tilde{f}_k = \tilde{f}_{-k}. \tag{71}$$

This implies

$$I_{2N} f(\theta) = \tilde{f}_0 + \sum_{k=1}^{N-1} 2\tilde{f}_k \cos(k\theta) + \tilde{f}_N \cos(N\theta) = \sum_{k=0}^{N} a_k \cos(k\theta) = \sum_{k=0}^{N} a_k T_k(x) \tag{72}$$

$$a_0 = \tilde{f}_0, a_N = \tilde{f}_N, a_k = 2\tilde{f}_k, k = 1, \ldots, N - 1 \tag{73}$$

This implies that the Fourier interpolation is essentially the same as a Chebyshev interpolation. They give the same approximation of the derivatives. Finally, we have special treatment at θ_0 and θ_N:

$$\frac{\mathrm{d}f}{\mathrm{d}x}(x_0) = \lim_{x \to x_0} \frac{\mathrm{d}f}{\mathrm{d}x}(x) \approx \lim_{\theta \to \theta_0} \frac{\frac{\mathrm{d}\sum_{k=0}^{N} a_k \cos(k\theta)}{\mathrm{d}\theta}}{-\sin(\theta)} = \sum_{k=0}^{N} a_k k^2, \tag{74}$$

$$\frac{\mathrm{d}f}{\mathrm{d}x}(x_N) = -\sum_{k=0}^{N} a_k (-1)^k k^2 \tag{75}$$

4 Differentiation Matrix

4.1 Fourier Differentiation Matrix

$$x_j = \frac{2\pi}{N} j, \quad j = 0, 1, \ldots, N-1, \tag{76}$$

$$f_j = f(x_j), \tag{77}$$

$$I_N f(x) = \sum_{k=-N/2}^{N/2} \frac{\tilde{f}_k}{\bar{c}_k} e^{ikx}, \tag{78}$$

$$\tilde{f}_k = \frac{1}{N} \sum_{j=0}^{N-1} f_j e^{-ikx_j}. \tag{79}$$

Lagrange Form:

$$I_N f(x) = \sum_{j=0}^{N-1} f_j h_j(x), \tag{80}$$

$$h_j(x) = \frac{1}{N} \sum_{k=-N/2}^{N/2} \frac{1}{\bar{c}_k} e^{ik(x-x_j)}. \tag{81}$$

Real form of $h_j(x)$:

$$h_j(x) = \frac{\sin(\frac{N}{2}(x - x_j))}{N \tan(\frac{1}{2}(x - x_j))}. \tag{82}$$

When the demonimator is 0 or ∞, the value of the form is understood in the limit sense.

$$D_{nj} = h_j'(x_n) = \begin{cases} 0, & j = n \\ \frac{1}{2}(-1)^{n-j} \cot(\frac{(n-j)\pi}{N}), & j \neq n \end{cases} \tag{83}$$

Negative Sum Trick:

$$\sum_{j=0}^{N-1} D_{nj} = 0, \tag{84}$$

$$\Rightarrow D_{nn} = - \sum_{j=0, j\neq n}^{N-1} D_{nj} \tag{85}$$

4.2 Differentiation of Lagrange Form Polynomials

The target function and the interpolation nodes:

$$f(x), \tag{86}$$

$$x_j, j = 0, 1, \ldots, N \tag{87}$$

$$f_j = f(x_j) \tag{88}$$

The Lagrange form interpolating polynomial:

$$p_N(x) = \sum_{j=0}^{N} f_j l_j(x), \tag{89}$$

$$l_j = \frac{(x - x_0)\cdots(x - x_{j-1})(x - x_{j+1})\cdots(x - x_N)}{(x_j - x_0)\cdots(x_j - x_{j-1})(x_j - x_{j+1})\cdots(x_j - x_N)} = w_j \prod_{m=0, m\neq j}^{N} (x - x_m), \tag{90}$$

$$w_j = \frac{1}{\prod_{m=0, m\neq j}^{N} (x_j - x_m)} \tag{91}$$

Derivatives:

$$p'_N(x) = \sum_{j=0}^{N} f_j l'_j(x), \tag{92}$$

$$l'_j(x) = w_j \sum_{\substack{k=0\\k\neq j}}^{N} \prod_{\substack{m=0\\m\neq j\\m\neq k}}^{N} (x - x_m), \tag{93}$$

$$l'_j(x_i) = w_j \prod_{\substack{m=0\\m\neq j\\m\neq i}}^{N} (x_i - x_m) = \frac{w_j}{w_i}\frac{1}{x_i - x_j}, \quad i \neq j, \tag{94}$$

$$l'_j(x_j) = w_j \sum_{\substack{k=0\\k\neq j}}^{N} \prod_{\substack{m=0\\m\neq j\\m\neq k}}^{N} (x_j - x_m) = \sum_{\substack{k=0\\k\neq j}}^{N} \frac{1}{x_j - x_m}, \tag{95}$$

$$p'_N(x_i) = \sum_{\substack{j=0\\j\neq i}}^{N} (f_i + \frac{f_j w_j}{w_i})\frac{1}{x_i - x_j}. \tag{96}$$

Trick:

$$f(x) = c \Rightarrow f_j = c \Rightarrow \sum_{j=0}^{N} l_j(x) = 1 \Rightarrow \sum_{j=0}^{N} l'_j(x) = 0, \tag{97}$$

$$\Rightarrow \sum_{j=0}^{N} l'_j(x_i) = 0 \Rightarrow l'_i(x_i) = -\sum_{\substack{j=0\\j\neq i}}^{N} l'_j(x_i), \tag{98}$$

$$p'_N(x_i) = \sum_{\substack{j=0\\j\neq i}}^{N} f_j l'_j(x_i) + f_i l'_i(x_i) \tag{99}$$

$$= \sum_{\substack{j=0\\j\neq i}}^{N} f_j l'_j(x_i) - f_i \sum_{\substack{j=0\\j\neq i}}^{N} l'_j(x_i) \tag{100}$$

$$= \sum_{\substack{j=0\\j\neq i}}^{N} (f_j - f_i) l'_j(x_i) \tag{101}$$

$$= \sum_{\substack{j=0\\j\neq i}}^{N} (f_j - f_i)\frac{w_j}{w_i}\frac{1}{x_i - x_j} \tag{102}$$

$$= -\frac{1}{w_i}\sum_{\substack{j=0\\j\neq i}}^{N} w_j \frac{f_i - f_j}{x_i - x_j}. \tag{103}$$

Derivative at x that is not a node:

$$p_N(x) = \frac{\sum_{j=0}^{N} f_j \frac{w_j}{x - x_j}}{\sum_{j=0}^{N} \frac{w_j}{x - x_j}}, \tag{104}$$

$$p'_N(x) = \frac{\sum_{j=0}^{N} \frac{[p_N(x) - f_j] w_j}{(x - x_j)^2}}{\sum_{j=0}^{N} \frac{w_j}{x - x_j}} \tag{105}$$

4.3 General Differentiation Matrix

$$f'(x_i) \approx (I_N f(x_i))' = \sum_{j=0}^{N} f_j l_j'(x_i) = \sum_{j=0}^{N} f_j D_{ij}, \tag{106}$$

$$D_{ij} = l_j'(x_i) = \frac{w_j}{w_i} \frac{1}{x_i - x_j}, \quad i \neq j \tag{107}$$

$$l_i'(x_i) = -\sum_{\substack{j=0 \\ j \neq i}}^{N} l_j'(x_i) \Rightarrow D_{ii} = -\sum_{\substack{j=0 \\ j \neq i}}^{N} D_{ij} \tag{108}$$

For Chebyshev nodes:

$$x_j = \cos(\frac{\pi}{N} j), \quad j = 0, 1, \ldots, N \tag{109}$$

$$D_{ij} = \frac{\bar{c}_i}{\bar{c}_j} \frac{(-1)^{i+j}}{x_i - x_j}, \quad i \neq j \tag{110}$$

5 The Fourier Collocation Method

5.1 Outline

Problem:

$$u_t + u_x = \nu u_{xx}, \quad 0 < x < 2\pi, t > 0 \tag{111}$$
$$u(x,0) = u_0(x), \quad 0 \le x \le 2\pi, \tag{112}$$
$$u(0,t) = u(2\pi, t), \quad t \ge 0. \tag{113}$$

Interpolation:

$$u(x,t) \approx p(x,t) = \sum_{n=0}^{N-1} u_n(t) h_n(x), \text{ (Fourier interpolation in Lagrange form)} \tag{114}$$
$$h_n(x) = \frac{1}{N} \sum_{k=-N/2}^{N/2} \frac{e^{ik(x-x_j)}}{\bar{c}_k}, \tag{115}$$
$$x_n = \frac{2\pi}{N} n, \quad n = 0, 1, \ldots, N-1, \tag{116}$$
$$\bar{c}_k = \begin{cases} 2, & k = \pm N/2, \\ 1, & k = -N/2+1, \ldots, k = N/2-1 \end{cases} \tag{117}$$

Collocation:

$$p_t + p_x - \nu p_{xx}|_{x=x_j} = 0. \tag{118}$$

About $p_t(x_j)$:

$$p_t(x) = \sum_{n=0}^{N-1} \dot{u}_n(t) h_n(x), \tag{119}$$
$$h_n(x_j) = \delta_{nj}, \tag{120}$$
$$p_t(x_j) = \dot{u}_n(t), \tag{121}$$

$p_x(x_j)$ given by the matrix multiplication

$$p_x(x_j) = \sum_{n=0}^{N-1} D_{jn} u_n. \tag{122}$$

$p_{xx}(x_j)$ given by the matrix multiplication:

$$p_{xx}(x_j) = \sum_{n=0}^{N-1} D_{jn}^{(2)} u_n, \tag{123}$$
$$[p_{xx}] = D^{(2)}[u_n] \approx D^2[u_n] \text{ (Matrix form)} \tag{124}$$

The resulting ODE system:

$$[\dot{u}_n(t)] = \nu D^2[u_n] - D[u_n] = D(\nu D[u_n] - [u_n]), \tag{125}$$
$$u_n(0) = u_0(x_n), \quad n = 0, 1, \ldots, N-1 \tag{126}$$

$[p_{xx}]$ and $[p_x]$ can also be calculated by DFT.

5.2 An ODE solver

An ODE system:

$$\dot{u} = F(u,t), \tag{127}$$
$$\Delta t, \tag{128}$$
$$t_n = n\Delta t, \tag{129}$$
$$u^n \approx u(t_n) \tag{130}$$

A 3rd-order Runge-Kutta solver:

Initializing:

$$u^{(0)} = u^n, \tag{131}$$
$$g^{(0)} = 0, \tag{132}$$
$$\tag{133}$$

First update:

$$g^{(1)} = g^{(0)} + F(u^{(0)}, t_n), \tag{134}$$
$$u^{(1)} = u^{(0)} + \frac{1}{3}\Delta t g^{(1)}, \tag{135}$$

Second update:

$$g^{(2)} = -\frac{5}{9}g^{(1)} + F(u^{(1)}, t_n + \frac{1}{3}\Delta t), \tag{136}$$
$$u^{(2)} = u^{(1)} + \frac{15}{16}\Delta t g^{(2)}, \tag{137}$$

Third update:

$$g^{(3)} = -\frac{153}{128}g^{(2)} + F(u^{(2)}, t_n + \frac{3}{4}\Delta t), \tag{138}$$
$$u^{(3)} = u^{(2)} + \frac{8}{15}\Delta t g^{(3)}, \tag{139}$$

General form:

$$g^{(i)} = cg[i]g^{(i-1)} + F(u^{(i-1)}, t_n + ct[i]\Delta t), \tag{140}$$
$$u^{(i)} = u^{(i-1)} + cu[i]\Delta t g^{(i)}, \tag{141}$$

Implementation:

$$g \leftarrow cg[i]\, g + F(u, t_n + ct[i]\Delta t), \tag{142}$$
$$u \leftarrow u + cu[i]\, \Delta t\, g \tag{143}$$

$$\begin{array}{c|c|c|c} i & cg & ct & cu \\ \hline 1 & 0 & 0 & 1/3 \\ 2 & -5/9 & 1/3 & 15/16 \\ 3 & -153/128 & 3/4 & 8/15 \end{array} \tag{144}$$

6 The Fourier Galerkin Method

6.1 Outline

Problem:

$$u_t + u_x = \nu u_{xx}, \quad 0 < x < 2\pi, t > 0 \tag{145}$$
$$u(x,0) = u_0(x), \quad 0 \leq x \leq 2\pi, \tag{146}$$
$$u(0,t) = u(2\pi, t), \quad t \geq 0. \tag{147}$$

Weak form:

$$\int_0^{2\pi} (u_t + u_x - \nu u_{xx})\phi^* \mathrm{d}x = 0, \tag{148}$$
$$\Rightarrow \int_0^{2\pi} u_t \phi^* \mathrm{d}x + \int_0^{2\pi} u_x \phi^* \mathrm{d}x = \nu u_x \phi^* |_0^{2\pi} - \int_0^{2\pi} u_x \phi_x^* \mathrm{d}x \tag{149}$$

The second line is the weak form since it only contains the first-order derivatives. The weak form is obtained by integration by parts.
Galerkin:

$$\text{Fourier interpolation: } u(x,t) = \sum_{n=-N/2}^{N/2} \tilde{u}_n(t) e^{inx}, \tag{150}$$
$$\text{Test functions identical to the basis functions: } \phi = e^{inx}, \tag{151}$$
$$\frac{\mathrm{d}\tilde{u}_n(t)}{\mathrm{d}t} + in\tilde{u}_n(t) = -\nu n^2 \tilde{u}_n(t), \quad n = -N/2, \ldots, N/2. \tag{152}$$

IC:

$$\sum_{n=-N/2}^{N/2} \tilde{u}_n(0) e^{inx} \approx u_0(x) \approx P_N u_0(x) = \sum_{n=-N/2}^{N/2} \hat{u}_{0,n} e^{inx}, \tag{153}$$
$$\Rightarrow \tilde{u}_n(0) = \hat{u}_{0,n} = \frac{1}{2\pi} \int_0^{2\pi} u_0(x) e^{-inx} \mathrm{d}x \tag{154}$$

Note that the initial coefficients are set to be the coefficients of the truncated series, not the coefficients of the Fourier interpolation of $u_0(x)$. The two sets of coefficients are not the same.