Appendix C

Recapitulation of Numerical Schemes

ABSTRACT
This appendix gathers the most common numerical schemes for comparison and in order to facilitate their implementation.

C.1 THE TRIDIAGONAL SYSTEM SOLVER

As a special case of the general **LU** decomposition (e.g., Riley, Hobson & Bence, 1977), an efficient tridiagonal system solver, based on the so-called Thomas algorithm, can be constructed. We begin by assuming that there exists a decomposition for which the lower (**L**) and upper (**U**) matrices possess the same bandwidth of 2, that is, nonzero elements exist only along two diagonals:

$$\begin{pmatrix} a_1 & c_1 & 0 & 0 & \cdots & 0 \\ b_2 & a_2 & c_2 & 0 & \cdots & 0 \\ 0 & b_3 & a_3 & c_3 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & b_{m-1} & a_{m-1} & c_{m-1} \\ 0 & 0 & \cdots & 0 & b_m & a_m \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ \beta_2 & 1 & 0 & 0 & \cdots & 0 \\ 0 & \beta_3 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \beta_{m-1} & 1 & 0 \\ 0 & 0 & \cdots & 0 & \beta_m & 1 \end{pmatrix}$$
$$\times \begin{pmatrix} \alpha_1 & \gamma_1 & 0 & 0 & \cdots & 0 \\ 0 & \alpha_2 & \gamma_2 & 0 & \cdots & 0 \\ 0 & 0 & \alpha_3 & \gamma_3 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \alpha_{m-1} & \gamma_{m-1} \\ 0 & 0 & \cdots & 0 & 0 & \alpha_m \end{pmatrix} \tag{C.1}$$

where the first matrix is the original tridiagonal matrix to be decomposed (i.e., elements a_1 etc. are known), the second matrix is **L** with one line of nonzero elements below the diagonal, and the last matrix is **U** with one line of nonzero elements above the diagonal.

Introduction to Geophysical Fluid Dynamics, Volume 101, ISSN 0074-6142, DOI: 10.1016/B978-0-12-088759-0.00025-0

Performing the product of matrices, we identify elements $(k, k-1)$, (k, k), and $(k, k+1)$ of the product as

$$b_k = \beta_k \alpha_{k-1} \tag{C.2a}$$

$$a_k = \beta_k \gamma_{k-1} + \alpha_k \tag{C.2b}$$

$$c_k = \gamma_k. \tag{C.2c}$$

These relations can be solved for the components of $\mathbf{L}$ and $\mathbf{U}$ by observing that $\gamma_k = c_k$ for all k. Since the first row demands $a_1 = \alpha_1$, subsequent rows provide α_k and β_k recursively from

$$\beta_k = \frac{b_k}{\alpha_{k-1}}, \quad \alpha_k = a_k - \beta_k c_{k-1}, \quad k = 2, \ldots, m \tag{C.3}$$

provided that no α_k is zero, otherwise the matrix is singular and cannot be decomposed. Note that there is no β_1.

The tridiagonal matrix $\mathbf{A}$ has now been decomposed into the product of lower and upper triangular matrices. The solution of $\mathbf{Ax} = \mathbf{LUx} = \mathbf{f}$ is then obtained by first solving $\mathbf{Ly} = \mathbf{f}$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ \beta_2 & 1 & 0 & 0 & \cdots & 0 \\ 0 & \beta_3 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \beta_{m-1} & 1 & 0 \\ 0 & 0 & \cdots & 0 & \beta_m & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{m-1} \\ y_m \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_{m-1} \\ f_m \end{pmatrix} \tag{C.4}$$

by proceeding from the first row downward and then solving $\mathbf{Ux} = \mathbf{y}$,

$$\begin{pmatrix} \alpha_1 & \gamma_1 & 0 & 0 & \cdots & 0 \\ 0 & \alpha_2 & \gamma_2 & 0 & \cdots & 0 \\ 0 & 0 & \alpha_3 & \gamma_3 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \alpha_{m-1} & \gamma_{m-1} \\ 0 & 0 & \cdots & 0 & 0 & \alpha_m \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{m-1} \\ x_m \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{m-1} \\ y_m \end{pmatrix}, \tag{C.5}$$

by proceeding from the bottom row upward. The solution is

$$y_1 = f_1, \quad y_k = f_k - \beta_k y_{k-1}, \quad k = 2, \ldots, m \tag{C.6}$$

$$x_m = \frac{y_m}{\alpha_m}, \quad x_k = \frac{y_k - \gamma_k x_{k+1}}{\alpha_k}, \quad k = m-1, \ldots, 1. \tag{C.7}$$

In practice, the α values can be stored in vector $\mathbf{a}$ initially holding the values a_k, since once α_k is known, a_k is no longer needed. Similarly, β values can be stored in vector $\mathbf{b}$ initially holding b_k and γ values in a vector $\mathbf{c}$. Also

the y and f values can share the same vector $\mathbf{f}$ as the f_k value is no longer needed once the y_k value has been computed. With the additional vector $\mathbf{x}$ for the solution, only five vectors are required, and the solution is obtained with only three loops over m points. This demands approximately $5m$ floating-point operations instead of the m^3 operations that a brutal matrix inversion would have required. The algorithm is implemented in MATLAB™ file `thomas.m`.

C.2 1D FINITE-DIFFERENCE SCHEMES OF VARIOUS ORDERS

TABLE C.1 Standard Finite-difference Operators for Uniform Grids

Forward Difference $\mathcal{O}(\Delta t)$

	u^n	u^{n+1}	u^{n+2}	u^{n+3}	u^{n+4}
$\Delta t \frac{\partial u}{\partial t}$	−1	1			
$\Delta t^2 \frac{\partial^2 u}{\partial t^2}$	1	−2	1		
$\Delta t^3 \frac{\partial^3 u}{\partial t^3}$	−1	3	−3	1	
$\Delta t^4 \frac{\partial^4 u}{\partial t^4}$	1	−4	6	−4	1

Forward Difference $\mathcal{O}(\Delta t^2)$

	u^n	u^{n+1}	u^{n+2}	u^{n+3}	u^{n+4}	u^{n+5}
$2\Delta t \frac{\partial u}{\partial t}$	−3	4	−1			
$\Delta t^2 \frac{\partial^2 u}{\partial t^2}$	2	−5	4	−1		
$2\Delta t^3 \frac{\partial^3 u}{\partial t^3}$	−5	18	−24	14	−3	
$\Delta t^4 \frac{\partial^4 u}{\partial t^4}$	3	−14	26	−24	11	−2

Backward Difference $\mathcal{O}(\Delta t)$

	u^{n-4}	u^{n-3}	u^{n-2}	u^{n-1}	u^n
$\Delta t \frac{\partial u}{\partial t}$				−1	1
$\Delta t^2 \frac{\partial^2 u}{\partial t^2}$			1	−2	1
$\Delta t^3 \frac{\partial^3 u}{\partial t^3}$		−1	3	−3	1
$\Delta t^4 \frac{\partial^4 u}{\partial t^4}$	1	−4	6	−4	1

Backward Difference $\mathcal{O}(\Delta t^2)$

	u^{n-5}	u^{n-4}	u^{n-3}	u^{n-2}	u^{n-1}	u^n
$2\Delta t \frac{\partial u}{\partial t}$				1	−4	3
$\Delta t^2 \frac{\partial^2 u}{\partial t^2}$			−1	4	−5	2
$2\Delta t^3 \frac{\partial^3 u}{\partial t^3}$		3	−14	24	−18	5
$\Delta t^4 \frac{\partial^4 u}{\partial t^4}$	−2	11	−24	26	−14	3

Central Difference $\mathcal{O}(\Delta t^2)$

	u^{n-2}	u^{n-1}	u^n	u^{n+1}	u^{n+2}
$2\Delta t \frac{\partial u}{\partial t}$		−1	0	1	
$\Delta t^2 \frac{\partial^2 u}{\partial t^2}$		1	−2	1	
$2\Delta t^3 \frac{\partial^3 u}{\partial t^3}$	−1	2	0	2	1
$\Delta t^4 \frac{\partial^4 u}{\partial t^4}$	1	−4	6	−4	1

Central Difference $\mathcal{O}(\Delta t^4)$

	u^{n-3}	u^{n-2}	u^{n-1}	u^n	u^{n+1}	u^{n+2}	u^{n+3}
$12\Delta t \frac{\partial u}{\partial t}$		1	−8	0	8	−1	
$12\Delta t^2 \frac{\partial^2 u}{\partial t^2}$		−1	16	−30	16	−1	
$8\Delta t^3 \frac{\partial^3 u}{\partial t^3}$	1	−8	13	0	−13	8	−1
$6\Delta t^4 \frac{\partial^4 u}{\partial t^4}$	−1	12	−39	56	−39	12	−1

Adapted from Chung (2002)

C.3 TIME-STEPPING ALGORITHMS

TABLE C.2 Standard Time-stepping Methods for $\mathrm{d}u/\mathrm{d}t = Q(t,u)$

Euler Methods

	Scheme	*Order*
Explicit	$\tilde{u}^{n+1} = \tilde{u}^n + \Delta t Q^n$	Δt
Implicit	$\tilde{u}^{n+1} = \tilde{u}^n + \Delta t Q^{n+1}$	Δt
Trapezoidal	$\tilde{u}^{n+1} = \tilde{u}^n + \frac{\Delta t}{2}\left(Q^n + Q^{n+1}\right)$	Δt^2
General	$\tilde{u}^{n+1} = \tilde{u}^n + \Delta t\left((1-\alpha)Q^n + \alpha Q^{n+1}\right)$	Δt

Multistage Methods

	Scheme	*Order*
Runge-Kutta	$\tilde{u}^{n+1/2} = \tilde{u}^n + \frac{\Delta t}{2} Q\left(t^n, \tilde{u}^n\right)$ $\tilde{u}^{n+1} = \tilde{u}^n + \Delta t Q\left(t^{n+1/2}, \tilde{u}^{n+1/2}\right)$	Δt^2
Runge-Kutta	$\tilde{u}_a^{n+1/2} = \tilde{u}^n + \frac{\Delta t}{2}\ Q\left(t^n, \tilde{u}^n\right)$ $\tilde{u}_b^{n+1/2} = \tilde{u}^n + \frac{\Delta t}{2}\ Q\left(t^{n+1/2}, \tilde{u}_a^{n+1/2}\right)$ $\tilde{u}^\star = \tilde{u}^n + \Delta t\ Q\left(t^{n+1/2}, \tilde{u}_b^{n+1/2}\right)$ $\tilde{u}^{n+1} = \tilde{u}^n + \Delta t\left(\frac{1}{6}Q\left(t^n, \tilde{u}^n\right) + \frac{2}{6}Q\left(t^{n+1/2}, \tilde{u}_a^{n+1/2}\right)\right.$ $\left. + \frac{2}{6}Q\left(t^{n+1/2}, \tilde{u}_b^{n+1/2}\right) + \frac{1}{6}Q\left(t^{n+1}, \tilde{u}^\star\right)\right)$	Δt^4

Multistep Methods

	Scheme	*Truncation Order*
Leapfrog	$\tilde{u}^{n+1} = \tilde{u}^{n-1} + 2\Delta t Q^n$	Δt^2
Adams-Bashforth	$\tilde{u}^{n+1} = \tilde{u}^n + \frac{\Delta t}{2}\left(-Q^{n-1} + 3Q^n\right)$	Δt^2
Adams-Moulton	$\tilde{u}^{n+1} = \tilde{u}^n + \frac{\Delta t}{12}\left(-Q^{n-1} + 8Q^n + 5Q^{n+1}\right)$	Δt^3
Adams-Bashforth	$\tilde{u}^{n+1} = \tilde{u}^n + \frac{\Delta t}{12}\left(5Q^{n-2} - 16Q^{n-1} + 23Q^{n+1}\right)$	Δt^3

Predictor-Corrector Methods

	Scheme	*Order*
Heun	$\tilde{u}^\star = \tilde{u}^n + \Delta t Q\left(t^n, \tilde{u}^n\right)$ $\tilde{u}^{n+1} = \tilde{u}^n + \frac{\Delta t}{2}\left(Q\left(t^n, \tilde{u}^n\right) + Q\left(t^{n+1}, \tilde{u}^\star\right)\right)$	Δt^2
Leapfrog-Trapezoidal	$\tilde{u}^\star = \tilde{u}^{n-1} + 2\Delta t Q^n$ $\tilde{u}^{n+1} = \tilde{u}^n + \frac{\Delta t}{2}\left(Q^n + 5Q\left(t^{n+1}, \tilde{u}^\star\right)\right)$	Δt^2
ABM	$\tilde{u}^\star = \tilde{u}^n + \frac{\Delta t}{2}\left(-Q^{n-1} + 3Q^n\right)$ $\tilde{u}^{n+1} = \tilde{u}^n + \frac{\Delta t}{12}\left(-Q^{n-1} + 8Q^n + 5Q\left(t^{n+1}, \tilde{u}^\star\right)\right)$	Δt^3

C.4 PARTIAL-DERIVATIVES FINITE DIFFERENCES

On a regular grid $x = x_0 + i\Delta x$, $y = y_0 + j\Delta y$, the following expressions are of second order

- Jacobian $J(a,b) = \dfrac{\partial a}{\partial x}\dfrac{\partial b}{\partial y} - \dfrac{\partial b}{\partial x}\dfrac{\partial a}{\partial y}$

$$J^{++}_{i,j} = \frac{(a_{i+1,j} - a_{i-1,j})(b_{i,j+1} - b_{i,j-1}) - (b_{i+1,j} - b_{i-1,j})(a_{i,j+1} - a_{i,j-1})}{4\Delta x \Delta y}$$

$$J^{+\times}_{i,j} = \frac{\left[a_{i+1,j}(b_{i+1,j+1} - b_{i+1,j-1}) - a_{i-1,j}(b_{i-1,j+1} - b_{i-1,j-1})\right]}{4\Delta x \Delta y} - \frac{\left[a_{i,j+1}(b_{i+1,j+1} - b_{i-1,j+1}) - a_{i,j-1}(b_{i+1,j-1} - b_{i-1,j-1})\right]}{4\Delta x \Delta y}$$

$$J^{\times+}_{i,j} = \frac{\left[b_{i,j+1}(a_{i+1,j+1} - a_{i-1,j+1}) - b_{i,j-1}(a_{i+1,j-1} - a_{i-1,j-1})\right]}{4\Delta x \Delta y} - \frac{\left[b_{i+1,j}(a_{i+1,j+1} - a_{i+1,j-1}) - b_{i-1,j}(a_{i-1,j+1} - a_{i-1,j-1})\right]}{4\Delta x \Delta y}$$

- Cross derivatives

$$\left.\frac{\partial^2 u}{\partial x \partial y}\right|_{i+1/2,j+1/2} \simeq \frac{u_{i+1,j+1} - u_{i+1,j} + u_{i,j} - u_{i,j+1}}{\Delta x \Delta y}$$

$$\left.\frac{\partial^2 u}{\partial x \partial y}\right|_{i,j} \simeq \frac{u_{i+1,j+1} - u_{i+1,j-1} + u_{i-1,j-1} - u_{i-1,j+1}}{4\Delta x \Delta y}$$

- Laplacian

$$\left.\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right|_{i,j} \simeq \frac{u_{i+1,j} + u_{i-1,j} - 2u_{i,j}}{\Delta x^2} + \frac{u_{i,j+1} + u_{i,j-1} - 2u_{i,j}}{\Delta y^2}$$

C.5 DISCRETE FOURIER TRANSFORM AND FAST FOURIER TRANSFORM

In a periodic domain in which x varies between 0 and L, a complex function $u(x)$ may be expanded in Fourier modes according to

$$u(x) = \sum_{n=-\infty}^{+\infty} a_n \, \mathrm{e}^{\mathrm{i}\, n \frac{2\pi x}{L}}. \tag{C.8}$$

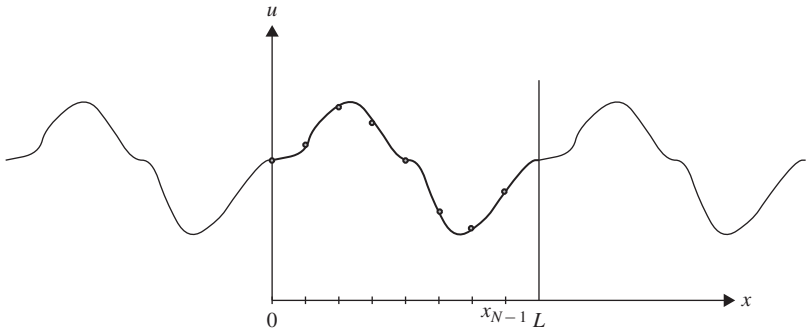

FIGURE C.1 Periodic signal sampled with N evenly spaced points x_0 to x_{N-1}. Note that no x_N is needed since by periodicity the value of the function at x_N is the same as at x_0.

From the orthogonality among Fourier modes,[1]

$$\frac{1}{L}\int_0^L \mathrm{e}^{\mathrm{i}\,(n-m)\frac{2\pi x}{L}}\,\mathrm{d}x = \delta_{nm}, \tag{C.9}$$

the complex coefficients a_n are readily obtained by multiplying (C.8) by $\exp(-\mathrm{i}\,m2\pi x/L)$ and integrating over the interval:

$$a_m = \frac{1}{L}\int_0^L u(x)\,\mathrm{e}^{-\mathrm{i}\,m\frac{2\pi x}{L}}\,\mathrm{d}x. \tag{C.10}$$

Note that we could have defined $a_n = b_n/L$ such that b_n were the integral without the factor $1/L$. Different authors use different notations.

To be an exact representation of the periodic function, the sum must cover an infinity of Fourier modes, but this is not possible on finite computers. The discrete Fourier transform (DFT) simply truncates the infinite series by limiting it to its first N terms. The procedure begins with the sampling of the function $u(x)$ at N equidistant points: $u_j = u(x_j)$, with $x_j = j\Delta x$, $j = 0, \ldots, N-1$ ($\Delta x = L/N$). The Fourier coefficients are then calculated from

$$a_n = \frac{1}{N}\sum_{j=0}^{N-1} u_j\,\mathrm{e}^{-\mathrm{i}\,nj\frac{2\pi}{N}}, \tag{C.11}$$

[1] The symbol δ_{ij} is called the Kronecker delta. Its value is 1 if $i = j$, and 0 if i differs from j.

and the sampled function can be reconstructed by summing over the first N modes[2]:

$$\tilde{u}(x) = \sum_{n=0}^{N-1} a_n \, \mathrm{e}^{\mathrm{i}\, n \frac{2\pi x}{L}}. \tag{C.12}$$

Obviously, this amounts to a discrete and finite version of the set (C.8) and (C.10). The interesting point is that the coefficients a_n are exact in the sense that if we evaluate $\tilde{u}(x_j)$ with those coefficients, we recover the sampled values at grid points $u(x_j)$. The proof that $\tilde{u}(x_j) = u(x_j)$, which is not trivial, begins by evaluating $\tilde{u}(x_j)$ of (C.12) with the a_n coefficients given by (C.11). We obtain successively

$$\begin{aligned} \tilde{u}(x_j) &= \sum_{n=0}^{N-1} a_n \, \mathrm{e}^{\mathrm{i}\, n \frac{2\pi x_j}{L}} \qquad\qquad (\text{C.13}) \\ &= \sum_{n=0}^{N-1} \frac{1}{N} \sum_{m=0}^{N-1} u_m \, \mathrm{e}^{-\mathrm{i}\, nm \frac{2\pi}{N}} \, \mathrm{e}^{\mathrm{i}\, nj \frac{2\pi}{N}} \\ &= \frac{1}{N} \sum_{m=0}^{N-1} u_m \left[\sum_{n=0}^{N-1} \mathrm{e}^{\mathrm{i}\,(j-m) n \frac{2\pi}{N}} \right] \\ &= \frac{1}{N} \sum_{m=0}^{N-1} u_m \left[\sum_{n=0}^{N-1} \rho^n \right] \qquad\qquad (\text{C.14}) \end{aligned}$$

where

$$\rho = \mathrm{e}^{\mathrm{i}\,(j-m)\frac{2\pi}{N}}. \tag{C.15}$$

For $j \neq m$, $\rho \neq 1$, and the geometric sum takes the value

$$\sum_{n=0}^{N-1} \rho^n = \frac{1-\rho^N}{1-\rho}, \tag{C.16}$$

which turns out to vanish because $\rho^N = 1$ with $j-m$ being an integer. When $j = m$, the sum between brackets is simply the sum of ones and is equal to N, which leads to $\tilde{u}(x_j) = u(x_j)$, the desired result.

Transformation (C.11) is called the direct discrete Fourier transform (DFT), whereas (C.13) is called the inverse discrete Fourier transform (IDFT). Note that the direct and inverse transforms are very similar in form, with two essential

[2]The truncation from 0 to $N-1$ is quite arbitrary. Sometimes a series with mode number running from $-N/2$ to $N/2$ is preferred.

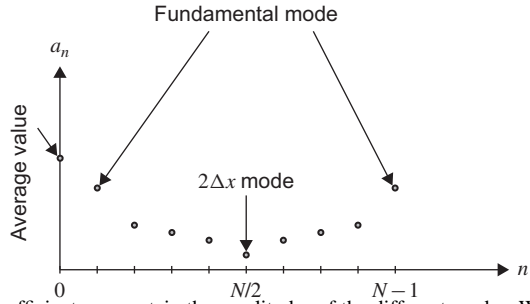

FIGURE C.2 Coefficients a_n contain the amplitudes of the different modes. With the summation running from 0 to $N-1$, $a_{N/2}$ contains the amplitude of the shortest wave.

differences, the sign in the exponential and the factor $1/N$. As for the infinite Fourier series, the scaling factor (then $1/L$, now $1/N$) is occasionally placed in front of the inverse transform rather than in front of the direct transform. It is a matter of choice.

Once the Fourier coefficients are known, Eq. (C.12) effectively provides a continuous and differentiable interpolation of the function $u(x)$ known only from its sampled values u_j. This interpolation may be used to calculate the value of the function at any intermediate location and also to evaluate derivatives to any order. The first derivative is

$$\frac{\mathrm{d}\tilde{u}}{\mathrm{d}x} = \sum_{n=0}^{N-1} \mathrm{i}\, k_n a_n \, \mathrm{e}^{\mathrm{i}\, k_n x}, \quad k_n = \frac{2\pi n}{L}. \tag{C.17}$$

Thus, all we have to do to calculate the first derivative is to create a set of Fourier coefficients b_n that are simply the function's Fourier coefficients a_n multiplied by a factor: $b_n = \mathrm{i}\, k_n a_n$. This simplicity is the key to the usefulness of the Fourier transform. Schematically, we have

$$u(x) \xrightarrow{\text{sampling}} u(x_j) \xrightarrow{\mathrm{DFT}(u_j)} a_n \xrightarrow{\text{derivative}} b_n = \mathrm{i}\, k_n a_n \xrightarrow{\mathrm{IDFT}(b_n)} \left.\frac{\mathrm{d}\tilde{u}}{\mathrm{d}x}\right|_{x_j}.$$

It is worth noting that the wavenumber k_{N-n} associated with mode $N-n$ in Eq. (C.12) corresponds to the same wavelength as wavenumber k_n of mode n. This is because the two exponentials differ by the factor $\exp(\mathrm{i}\, 2\pi N x/L) = \exp(\mathrm{i}\, 2\pi x/\Delta x)$, which is equal to 1 at every sampling point. From this follows that the shortest wavelength resolved by the truncated series corresponds to $n = N/2$ and has wavenumber $k = N\pi/L$ or wavelength $2L/N = 2\Delta x$. Indeed, the shortest wave in numerical terms is the simple oscillation $(+1, -1, +1, -1,$ etc.) that repeats every $2\Delta x$ interval. The coefficient a_0 corresponds to the zero-wavenumber component and thus holds the average value; the pair (a_1, a_{N-1}) contains information on wavelength L (amplitude and phase), (a_2, a_{N-2}) on

wavelength $L/2$, and so on up to $a_{N/2}$ for which there is no phase information.[3] We can see why some presentations of the DFT define the series with an index n running between $-N/2$ and $N/2$ instead of running from 0 to N. The information content, or $N+1$ numbers, is the same.

Up to now, the functions we considered were complex functions. For a real-valued function u, there is a redundancy among the coefficients a_n: a_{N-n} is equal to the complex conjugate of a_n. This is easily seen by equating the Fourier transforms of the function and of its complex conjugate, which must be the same because a real number is always its own complex conjugate.

Some algorithms exploit this redundancy. Cosine Fourier transforms (CFT) and sine Fourier transforms (SFT) perform in a way similar to DFT by using only cosine or sine functions, and they are particularly useful if the solution of a problem is known to satisfy particular boundary conditions. For homogeneous Neumann conditions at $x=0$ and $x=L$ (i.e., derivative of function vanishes at both ends), CFT is the method of choice, whereas for homogeneous Dirichlet conditions (i.e., function itself vanishes at both ends), expansion in terms of sine functions guarantees automatic satisfaction of the boundary conditions.

The annoying aspect about the discrete Fourier transform is that a straightforward implementation as a sum of exponentials demands a data array of length N, and for each coefficient a_n the calculation of N exponentials, necessitating a total of N^2 calculations of exponential functions (or sine or cosine functions). Recognizing that this is prohibitively expensive even for a modicum of spatial resolution, Cooley and Tukey (1965) introduced a clever method that probably ranks among the most celebrated numerical algorithms of all times. It reduces the computational cost of the DFT from N^2 to $N\log_2 N$ operations. Interestingly enough, the original idea of the method, now called Fast Fourier Transform, goes back to Gauss in 1805 (see collected reprints Gauss, 1866), long before computational methods could exploit it.

The Fast Fourier Transform (FFT) is a practical calculation of the discrete Fourier transform that starts with the observation that if N is even, the series can be split between its even and odd terms in j:

$$\begin{aligned} Na_n &= \sum_{\substack{j=0 \\ j\,\text{even}}}^{N-1} u_j \mathrm{e}^{-\mathrm{i}nj\frac{2\pi}{N}} + \sum_{\substack{j=0 \\ j\,\text{odd}}}^{N-1} u_j \mathrm{e}^{-\mathrm{i}nj\frac{2\pi}{N}} \\ &= \sum_{m=0}^{N/2-1} u_{2m} \mathrm{e}^{-\mathrm{i}nm\frac{2\pi}{(N/2)}} + \mathrm{e}^{-\mathrm{i}n\frac{2\pi}{N}} \sum_{m=0}^{N/2-1} u_{2m+1} \mathrm{e}^{-\mathrm{i}nm\frac{2\pi}{(N/2)}} \end{aligned} \tag{C.18}$$

in which we set $j=2m$ in the first sum where j is even and $j=2m+1$ in the second sum where j is odd. Now, if N is divisible by 2, the new exponentials are

[3] For the $2\Delta x$ wave, the only possible phase shift is the shift by Δx, which is accommodated by a sign change in amplitude.

those of a discrete Fourier transform with only $N/2$ points, that is, half as many points. In turn, if $N/2$ is itself divisible by 2, the coefficients can be obtained from a transform with yet half as many points, $N/4$. Quite obviously, the greatest advantage is obtained when N is repeatedly divisible by 2, that is, when it is a power of 2, but computational efficiency can also be achieved, to a lesser extent, with N values that have factors other than 2.

To estimate the computational cost $C(N)$ for a transformation with N data, we note that it consists of the cost of two transformations of size $N/2$ and a multiplication by $\exp(-\mathrm{i}\, n(2\pi/N))$ for each of the N coefficients a_n. Hence,

$$C(N) = 2C(N/2) + N. \tag{C.19}$$

For a single point ($N = 1$), only one operation is needed ($C(1) = 1$) so that we have $C(2) = 4$, $C(4) = 12$, $C(8) = 32$, $C(16) = 80$, etc. leading to an asymptotic behavior growing with N as

$$C(N) \sim N \log_2 N. \tag{C.20}$$

There is thus a substantial gain compared to the brute-force approach, and this is particularly interesting in the context of spectral methods (Section 18.4) when direct and inverse transforms are performed repeatedly.

Interpolation of a function knowing its Fourier series coefficients could be obtained by brute-force summation of many exponential functions (or sines and cosines) as prescribed by (C.12) at the various sampling locations. But we can do better. Suppose that we have at our disposal values of a function on a certain regular grid $x_j = (j/N)L$, $j = 0, ..., N-1$ and that we need to interpolate onto a finer regular grid $x_k = (k/M)L$, $k = 0, ..., M-1$ with $M > N$. One method of interpolation is to perform the discrete Fourier transform of the function on the original grid, add a string of zeros for the amplitudes of the higher modes permitted by the finer grid but absent from the original grid, and then take the inverse Fourier transform. This back and forth transformation in and out of spectral space may first appear as a wasteful detour, but given the computational efficiency of the Fast Fourier Transform, the procedure is actually advantageous. The procedure is called *padding*.

ANALYTICAL PROBLEMS

C.1. Find the truncation errors of the two Adams–Bashforth schemes by means of Taylor expansions.

C.2. Which of the two second-order approximations of the cross derivative has the lowest truncation error?

C.3. Generate a finite-difference approximation of a Jacobian at a corner point $i+1/2, j+1/2$, using only values from the nearest grid points.

C.4. Prove that the discrete Fourier transform is exact in the sense that the inverse transform of the transform returns exactly the initial set of values.

C.5. Prove that $C(N) \sim N \log_2 N$ is the asymptotic behavior of recursive relation $C(N) = 2\, C(N/2) + N$ for large N.

NUMERICAL EXERCISES

C.1. Compare the behavior of the second-order Adams–Bashforths method and leapfrog-trapezoidal method in the numerical simulation of an inertial oscillation.

C.2. Discretize the function $u(x, y) = \sin(2\pi x/L)\cos(2\pi y/L)$ on a regular grid in the plane (x, y). Calculate the numerical Jacobian between

- $\tilde{u}$ and $\tilde{u}$
- $\tilde{u}$ and $\tilde{u}^2$
- $\tilde{u}$ and $\tilde{u}^3$

and interpret your results.

C.3. Perform an FFT on $f(x) = \sin(2\pi x/L)$ between $x = 0$ and $x = L$ by sampling with 10, 20, or 40 points. Using the spectral coefficients obtained from the FFT, plot the Fourier series using very fine resolution in x (say, 200 points) and verify that you recover the initial function. Then repeat with the function $f(x) = x$. What do you observe?

C.4. Redo Numerical Exercise C.3 using the padding technique instead of the brute-force evaluation to plot the Fourier series expansion.