# 11 Pseudospectral Methods for Two-Point BVPs

Another class of very accurate numerical methods for BVPs (as well as many time-dependent PDEs) are the so-called *spectral* or *pseudospectral methods*. The basic idea is similar to the collocation method described above. However, now we use other basis functions. The following discussion closely follows the first few chapters of Nick Trefethen's book "Numerical Methods in Matlab".

Before we go into any details we present an example.

**Example** Consider the simple linear 2-pt BVP

$$y''(t) = e^{4t}, \quad t \in (-1, 1)$$

with boundary conditions $y(-1) = y(1) = 0$. The analytic solution of this problem is given by

$$y(t) = \left[e^{4t} - t\sinh(4) - \cosh(4)\right]/16.$$

In the Matlab program `PSBVPDemo.m` we compare the new pseudospectral approach with the finite difference approach.

The high accuracy of the pseudospectral method is impressive, and we use this as our motivation to take a closer look at this method.

As with all the other numerical methods, we require some sort of *discretization*. For pseudospectral methods we do the same as for finite difference methods and the RBF collocation methods, i.e., we introduce a set of grid points $t_1, t_2, \ldots, t_N$ in the interval of interest.

## 11.1 Differentiation Matrices

The main ingredient for pseudospectral methods is the concept of a *differentiation matrix* $D$. This matrix will map a vector of function values $\boldsymbol{y} = [y(t_1), \ldots, y(t_N)]^T = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N]^T$ at the grid points to a vector $\boldsymbol{y}'$ of derivative values, i.e.,

$$\boldsymbol{y}' = D\boldsymbol{y}.$$

What does such a differentiation matrix look like? Let's assume that the grid points are uniformly spaced with spacing $t_{j+1} - t_j = h$ for all $j$, and that the vector of function values $\boldsymbol{y}$ comes from a periodic function so that we can add the two auxiliary values $\boldsymbol{y}_0 = \boldsymbol{y}_N$ and $\boldsymbol{y}_{N+1} = \boldsymbol{y}_1$.

In order to approximate the derivative $y'(t_j)$ we start with another look at the finite difference approach. We use the symmetric (second-order) finite difference approximation

$$y'(t_j) \approx \boldsymbol{y}'_j = \frac{\boldsymbol{y}_{j+1} - \boldsymbol{y}_{j-1}}{2h}, \qquad j = 1, \ldots, N.$$

Note that this formula also holds at both ends ($j = 1$ and $j = N$) since we are assuming periodicity of the data.

These equations can be collected in matrix-vector form:

$$\boldsymbol{y}' = D\boldsymbol{y}$$

with $\boldsymbol{y}$ and $\boldsymbol{y}'$ as above and

$$D = \frac{1}{h}\begin{bmatrix} 0 & \frac{1}{2} & & & -\frac{1}{2} \\ -\frac{1}{2} & 0 & \ddots & & \\ & & \ddots & & \\ & & \ddots & 0 & \frac{1}{2} \\ \frac{1}{2} & & & -\frac{1}{2} & 0 \end{bmatrix}.$$

**Remark** This matrix has a very special structure. It is both *Toeplitz* and *circulant*. In a Toeplitz matrix the entries in each diagonal are constant, while a circulant matrix is generated by a single row vector whose entries are shifted by one (in a circulant manner) each time a new row is generated. As we will see later, the fast Fourier transform (FFT) can deal with such matrices in a particularly efficient manner.

As we saw earlier, there is a close connection between finite difference approximations of derivatives and polynomial interpolation. For example, the symmetric 2nd-order approximation used above can also be obtained by differentiating the interpolating polynomial $p$ of degree 2 to the data $\{(t_{j-1}, \boldsymbol{y}_{j-1}), (t_j, \boldsymbol{y}_j), (t_{j+1}, \boldsymbol{y}_{j+1})\}$, and then evaluating at $t = t_j$.

We can also use a degree 4 polynomial to interpolate the 5 (symmetric) pieces of data $\{(t_{j-2}, \boldsymbol{y}_{j-2}), (t_{j-1}, \boldsymbol{y}_{j-1}), (t_j, \boldsymbol{y}_j), (t_{j+1}, \boldsymbol{y}_{j+1}), (t_{j+2}, \boldsymbol{y}_{j+2})\}$. This leads to (e.g., modifying the code in the Maple worksheet `478578_DerivativeEstimates.mws`)

$$y'(t_j) \approx \boldsymbol{y}'_j = -\frac{\boldsymbol{y}_{j+2} - 8\boldsymbol{y}_{j+1} + 8\boldsymbol{y}_{j-1} - \boldsymbol{y}_{j-2}}{12h}, \qquad j = 1, \ldots, N,$$

so that we get the differentiation matrix

$$D = \frac{1}{h}\begin{bmatrix} 0 & \frac{2}{3} & -\frac{1}{12} & & & \frac{1}{12} & -\frac{2}{3} \\ -\frac{2}{3} & 0 & \frac{2}{3} & -\frac{1}{12} & & & \frac{1}{12} \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ -\frac{1}{12} & & & \frac{1}{12} & -\frac{2}{3} & 0 & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{12} & & & \frac{1}{12} & -\frac{2}{3} & 0 \end{bmatrix}.$$

Note that this matrix is again a circulant Toeplitz matrix (since the data is assumed to be periodic). However, now there are 5 diagonals, instead of the 3 for the second-order example above.

**Example** The fourth-order convergence of the finite-difference approximation above is illustrated in the Matlab script `FD4Demo.m`.

It should now be clear that — in order to increase the accuracy of the finite-difference derivative approximation to spectral order — we want to keep on increasing the polynomial degree so that more and more grid points are being used, and the differentiation matrix becomes a dense matrix. Thus, we can think of pseudospectral

methods as finite difference methods based on *global* polynomial interpolants instead of local ones.

For an *infinite interval* with *infinitely many* grid points spaced a distance $h$ apart one can show that the resulting differentiation matrix is given by the circulant Toeplitz matrix

$$D = \frac{1}{h} \begin{bmatrix} & \vdots & \\ \ddots & \frac{1}{3} & \\ \ddots & -\frac{1}{2} & \\ \ddots & 1 & \\ & 0 & \\ & -1 & \ddots \\ & \frac{1}{2} & \ddots \\ & -\frac{1}{3} & \ddots \\ & \vdots & \end{bmatrix}. \tag{98}$$

For a finite (even) $N$ and periodic data we will show later that the differentiation matrix is given by

$$D_N = \begin{bmatrix} & \vdots & \\ \ddots & \frac{1}{2}\cot\frac{3h}{2} & \\ \ddots & -\frac{1}{2}\cot\frac{2h}{2} & \\ \ddots & \frac{1}{2}\cot\frac{1h}{2} & \\ & 0 & \\ & -\frac{1}{2}\cot\frac{1h}{2} & \ddots \\ & \frac{1}{2}\cot\frac{2h}{2} & \ddots \\ & -\frac{1}{2}\cot\frac{3h}{2} & \ddots \\ & \vdots & \end{bmatrix}. \tag{99}$$

**Example** If $N = 4$, then we have

$$D_4 = \begin{bmatrix} 0 & \frac{1}{2}\cot\frac{1h}{2} & \frac{1}{2}\cot\frac{2h}{2} & -\frac{1}{2}\cot\frac{1h}{2} \\ -\frac{1}{2}\cot\frac{1h}{2} & 0 & \frac{1}{2}\cot\frac{1h}{2} & \frac{1}{2}\cot\frac{2h}{2} \\ \frac{1}{2}\cot\frac{2h}{2} & -\frac{1}{2}\cot\frac{1h}{2} & 0 & \frac{1}{2}\cot\frac{1h}{2} \\ \frac{1}{2}\cot\frac{1h}{2} & \frac{1}{2}\cot\frac{2h}{2} & -\frac{1}{2}\cot\frac{1h}{2} & 0 \end{bmatrix}.$$

The Matlab script `PSDemo.m` illustrates the spectral convergence obtained with the matrix $D_N$ for various values of $N$. The output should be compared with that of the previous example `FD4Demo.m`.

## 11.2 Unbounded Grids and the Semi-Discrete Fourier Transform

We now consider an *infinite* uniform grid $h\mathbb{Z}$ with grid points $t_j = jh$ for all integers $j$. While this case is not useful for practical computation, it is important for our understanding of problems on bounded intervals.

First we recall the definition of the *Fourier transform* $\hat{y}$ of a function $y$ that is square-integrable on $\mathbb{R}$:

$$\hat{y}(\omega) = \int_{-\infty}^{\infty} e^{-i\omega t} y(t) dt, \qquad \omega \in \mathbb{R}. \tag{100}$$

Conversely, the *inverse Fourier transform* lets us reconstruct $y$ from its Fourier transform $\hat{y}$:

$$y(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{i\omega t} \hat{y}(\omega) d\omega, \qquad t \in \mathbb{R}. \tag{101}$$

**Example** Consider the function

$$y(t) = \begin{cases} 1, & \text{if } -1/2 \le t \le 1/2 \\ 0, & \text{otherwise,} \end{cases}$$

and compute its Fourier transform.

By the definition of the Fourier transform, the definition of $y$ and Euler's formula we have

$$\begin{aligned} \hat{y}(\omega) &= \int_{-\infty}^{\infty} e^{-i\omega t} y(t) dt \\ &= \int_{-1/2}^{1/2} e^{-i\omega t} dt \\ &= \int_{-1/2}^{1/2} \left[\cos(\omega t) - i \sin(\omega t)\right] dt \\ &= 2 \int_{0}^{1/2} \cos(\omega t) dt \\ &= 2 \frac{\sin(\omega t)}{\omega} \Big|_0^{1/2} = \frac{\sin \omega/2}{\omega/2}. \end{aligned}$$

These functions play an important role in many applications (e.g., signal processing). The function $y$ is known as a *square pulse* or *characteristic function of the interval* $[-1/2, 1/2]$, and its Fourier transform $\hat{y}$ is known as the *sinc function.*

If we restrict our attention to a *discrete* (unbounded) physical space, i.e., the function $y$ is now given by the (infinite) vector $\boldsymbol{y} = [\ldots, \boldsymbol{y}_{-1}, \boldsymbol{y}_0, \boldsymbol{y}_1, \ldots]^T$ of discrete values, then the formulas change. In fact, the *semidiscrete Fourier transform* of $\boldsymbol{y}$ is given by the (continuous) function

$$\hat{y}(\omega) = h \sum_{j=-\infty}^{\infty} e^{-i\omega t_j} \boldsymbol{y}_j, \qquad \omega \in [-\pi/h, \pi/h], \tag{102}$$

and the *inverse semidiscrete Fourier transform* is given by the (discrete infinite) vector $\boldsymbol{y}$ whose components are of the form

$$\boldsymbol{y}_j = \frac{1}{2\pi} \int_{-\pi/h}^{\pi/h} e^{i\omega t_j} \hat{y}(\omega) d\omega, \qquad j \in \mathbb{Z}. \tag{103}$$

**Remark** Note that the notion of a semidiscrete Fourier transform is just a different name for a *Fourier series* based on the complex exponentials $e^{-i\omega t_j}$ with *Fourier coefficients* $\boldsymbol{y}_j$.

The interesting difference between the continuous and semidiscrete setting is marked by the *bounded* Fourier space in the semidiscrete setting. This can be explained by the phenomenon of *aliasing*. Aliasing arises when a continuous function is sampled on a discrete set. In particular, the two complex exponential functions $f(t) = e^{i\omega_1 t}$ and $g(t) = e^{i\omega_2 t}$ differ from each other on the real line as long as $\omega_1 \neq \omega_2$. However, if we sample the two functions on the grid $h\mathbb{Z}$, then we get the vectors $\boldsymbol{f}$ and $\boldsymbol{g}$ with values $\boldsymbol{f}_j = e^{i\omega_1 t_j}$ and $\boldsymbol{g}_j = e^{i\omega_2 t_j}$. Now, if $\omega_2 = \omega_1 + 2k\pi/h$ for some integer $k$, then $\boldsymbol{f}_j = \boldsymbol{g}_j$ for all $j$, and the two (different) continuous functions $f$ and $g$ appear identical in their discrete representations $\boldsymbol{f}$ and $\boldsymbol{g}$. Thus, any complex exponential $e^{i\omega t}$ is matched on the grid $h\mathbb{Z}$ by infinitely many other complex exponentials (its *aliases*). Therefore we can limit the representation of the Fourier variable $\omega$ to an interval of length $2\pi/h$. For reasons of symmetry we use $[-\pi/h, \pi/h]$.

### 11.2.1 Spectral Differentiation

To get the interpolant of the $\boldsymbol{y}_j$ values we can now use an extension of the inverse semidiscrete Fourier transform, i.e., we define the interpolant to be the function

$$p(t) = \frac{1}{2\pi} \int_{-\pi/h}^{\pi/h} e^{i\omega t} \hat{y}(\omega) d\omega, \qquad t \in \mathbb{R}. \tag{104}$$

It is obvious (cf. (103)) from this definition that $p$ interpolates the data, i.e., $p(t_j) = \boldsymbol{y}_j$, for any $j \in \mathbb{Z}$.

Moreover, the Fourier transform of the function $p$ turns out to be

$$\hat{p}(\omega) = \begin{cases} \hat{y}(\omega), & \omega \in [\pi/h, \pi, h] \\ 0, & \text{otherwise} \end{cases}$$

This kind of function is known as a *band-limited function*, and $p$ is called the *band-limited interpolant* of $\boldsymbol{y}$.

The spectral derivative vector $\boldsymbol{y}'$ of $\boldsymbol{y}$ can now be obtained by one of the following two procedures we are about to present. First,

1. Sample the function $y$ at the (infinite set of) discrete points $t_j \in h\mathbb{Z}$ to obtain the data vector $\boldsymbol{y}$ with components $\boldsymbol{y}_j$.

2. Compute the semidiscrete Fourier transform of the data via (102):

$$\hat{y}(\omega) = h \sum_{j=-\infty}^{\infty} e^{-i\omega t_j} \boldsymbol{y}_j, \qquad \omega \in [-\pi/h, \pi/h].$$

3. Find the band-limited interpolant $p$ of the data $\boldsymbol{y}_j$ via (104).

4. Differentiate $p$ and evaluate at the $t_j$.

However, from a computational point of view it is better to deal with this problem in the Fourier domain. We begin by noting that the Fourier transform of the derivative $y'$ is given by

$$\widehat{y'}(\omega) = \int_{-\infty}^{\infty} e^{-i\omega t} y'(t) dt.$$

Applying integration by parts we get

$$\widehat{y'}(\omega) = e^{-i\omega t} y(t)\big|_{-\infty}^{\infty} + i\omega \int_{-\infty}^{\infty} e^{-i\omega t} y(t) dt.$$

If $y(t)$ tends to zero for $t \to \pm\infty$ (which it has to for the Fourier transform of $y$ to exist) then we see that

$$\widehat{y'}(\omega) = i\omega \hat{y}(\omega). \tag{105}$$

Therefore, we obtain the spectral derivative $\boldsymbol{y'}$ by the following alternate procedure:

1. Sample the function $y$ at the (infinite set of) discrete points $t_j \in h\mathbb{Z}$ to obtain the data vector $\boldsymbol{y}$ with components $\boldsymbol{y}_j$.

2. Compute the semidiscrete Fourier transform of the data via (102):

$$\hat{y}(\omega) = h \sum_{j=-\infty}^{\infty} e^{-i\omega t_j} \boldsymbol{y}_j, \qquad \omega \in [-\pi/h, \pi/h].$$

3. Compute the Fourier transform of the derivative via (105):

$$\widehat{y'}(\omega) = i\omega \hat{y}(\omega).$$

4. Find the derivative vector via inverse semidiscrete Fourier transform (see (103)), i.e.,

$$\boldsymbol{y}'_j = \frac{1}{2\pi} \int_{-\pi/h}^{\pi/h} e^{i\omega t_j} \widehat{y'}(\omega) d\omega, \qquad j \in \mathbb{Z}.$$

Now we need to find out how we can obtain the entries of the differentiation matrix $D$ from the preceding discussion. We follow the first procedure above.

In order to be able to compute the semidiscrete Fourier transform of an arbitrary data vector $\boldsymbol{y}$ we represent its components in terms of *shifts of* (discrete) *delta functions*, i.e.,

$$\boldsymbol{y}_j = \sum_{k=-\infty}^{\infty} \boldsymbol{y}_k \delta_{j-k}, \tag{106}$$

where the *Kronecker delta function* is defined by

$$\delta_j = \begin{cases} 1 & j = 0 \\ 0 & \text{otherwise.} \end{cases}$$

We use this approach since the semidiscrete Fourier transform of the delta function can be computed easily. In fact, according to (102)

$$\begin{aligned}\hat{\delta}(\omega) &= h\sum_{j=-\infty}^{\infty} e^{-i\omega t_j}\delta_j \\ &= he^{-i\omega t_0} = h\end{aligned}$$

for all $\omega \in [-\pi/h, \pi/h]$. Then the band-limited interpolant of $\delta$ is of the form (see (104))

$$\begin{aligned}p(t) &= \frac{1}{2\pi}\int_{-\pi/h}^{\pi/h} e^{i\omega t}\hat{\delta}(\omega)d\omega \\ &= \frac{1}{2\pi}\int_{-\pi/h}^{\pi/h} e^{i\omega t}h d\omega \\ &= \frac{h}{\pi}\int_{0}^{\pi/h} \cos(\omega t)d\omega \\ &= \frac{h}{\pi}\frac{\sin(\omega t)}{t}\Big|_0^{\pi/h} \\ &= \frac{h}{\pi}\frac{\sin(\pi t/h)}{t} = \frac{\sin(\pi t/h)}{\pi t/h} = \mathrm{sinc}(\pi t/h).\end{aligned}$$

Therefore, the band-limited interpolant of an arbitrary data vector $\boldsymbol{y}$ is given by

$$\begin{aligned}p(t) &= \frac{1}{2\pi}\int_{-\pi/h}^{\pi/h} e^{i\omega t}\hat{y}(\omega)d\omega \\ &= \frac{1}{2\pi}\int_{-\pi/h}^{\pi/h} e^{i\omega t}\left[h\sum_{j=-\infty}^{\infty} e^{-i\omega t_j}\boldsymbol{y}_j\right]d\omega \\ &= \frac{1}{2\pi}\int_{-\pi/h}^{\pi/h} e^{i\omega t}\left[h\sum_{j=-\infty}^{\infty} e^{-i\omega t_j}\sum_{k=-\infty}^{\infty}\boldsymbol{y}_k\delta_{j-k}\right]d\omega.\end{aligned}$$

Thus far we have used the definition of the band-limited interpolant (104), the definition of the semidiscrete Fourier transform of $y$ (102), and the representation (106). Interchanging the summation, and then using the definition of the delta function and the same calculation as for the band-limited interpolant of the delta function above we

obtain the final form of the band-limited interpolant of an arbitrary data vector $\boldsymbol{y}$ as

$$\begin{aligned} p(t) &= \frac{1}{2\pi}\int_{-\pi/h}^{\pi/h} e^{i\omega t}\left[h\sum_{k=-\infty}^{\infty}\boldsymbol{y}_k\sum_{j=-\infty}^{\infty}e^{-i\omega t_j}\delta_{j-k}\right]d\omega \\ &= \frac{1}{2\pi}\int_{-\pi/h}^{\pi/h} e^{i\omega t}h\sum_{k=-\infty}^{\infty}\boldsymbol{y}_k e^{-i\omega t_k}d\omega \\ &= \sum_{k=-\infty}^{\infty}\boldsymbol{y}_k\frac{1}{2\pi}\int_{-\pi/h}^{\pi/h} e^{i\omega(t-t_k)}hd\omega \\ &= \sum_{k=-\infty}^{\infty}\boldsymbol{y}_k\mathrm{sinc}\frac{(t-t_k)\pi}{h}. \end{aligned}$$

**Example** Band-limited interpolation for the functions

$$y_1(t) = \begin{cases} 1, & t = 0 \\ 0, & \text{otherwise,} \end{cases}$$

$$y_2(t) = \begin{cases} 1, & |t| \le 3 \\ 0, & \text{otherwise,} \end{cases}$$

and

$$y_3(t) = (1 - |t|/3)_+.$$

is illustrated in the Matlab script `BandLimitedDemo.m`. Note that the accuracy of the reproduction is not very high. Note, in particular, the *Gibbs phenomenon* that arises for $h \to 0$. This is due to the low smoothness of the data functions.

In order to get the components of the derivative vector $\boldsymbol{y}'$ we need to differentiate the band-limited interpolant and evaluate at the grid points. By linearity this leads to

$$\boldsymbol{y}'_j = p'(t_j) = \sum_{k=-\infty}^{\infty}\boldsymbol{y}_k\frac{d}{dt}\left[\mathrm{sinc}\frac{(t-t_k)\pi}{h}\right]_{t=t_j},$$

or in (infinite) matrix form

$$\boldsymbol{y}' = D\boldsymbol{y}$$

with the entries of $D$ given by

$$D_{jk} = \frac{d}{dt}\left[\mathrm{sinc}\frac{(t-t_k)\pi}{h}\right]_{t=t_j}, \qquad j,k = -\infty,\ldots,\infty.$$

The entries in the $k = 0$ column of $D$ are of the form

$$D_{j0} = \frac{d}{dt}\left[\mathrm{sinc}\frac{t\pi}{h}\right]_{t=t_j=jh} = \begin{cases} 0, & j = 0 \\ \frac{(-1)^j}{jh}, & \text{otherwise,} \end{cases}$$

The remaining columns are shifts of this column since the matrix is a Toeplitz matrix. This is exactly of the form (98). The explicit formula for the derivative of the sinc function above is obtained using elementary calculations:

$$\frac{d}{dt}\left[\operatorname{sinc}\frac{t\pi}{h}\right] = \frac{1}{t}\cos\left(\frac{t\pi}{h}\right) - \frac{h}{t^2\pi}\sin\left(\frac{t\pi}{h}\right),$$

so that

$$\frac{d}{dt}\left[\operatorname{sinc}\frac{t\pi}{h}\right]_{t=t_j=jh} = \frac{1}{jh}\cos(j\pi) - \frac{1}{j^2h\pi}\sin(j\pi).$$

### 11.3 Periodic Grids: The DFT and FFT

We now consider the case of a *bounded* grid with periodic data, i.e., we will now explain how to find the entries in the matrix $D_N$ of (99).

To keep the discussion simple we will consider the interval $[0, 2\pi]$ only, and assume that we are given $N$ (with $N$ even) uniformly spaced grid points $t_j = jh$, $j = 1, \ldots, N$, with $h = 2\pi/N$.

**Remark** Formulas for odd $N$ also exist, but are slightly different. For the sake of clarity, we focus only on the even case here.

As in the previous subsection we now look at the Fourier transform of the discrete and periodic data $\boldsymbol{y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N]^T$ with $\boldsymbol{y}_j = y(jh) = y(2j\pi/N)$, $j = 1, \ldots, N$. For the same reason of aliasing the Fourier domain will again be bounded. Moreover, the periodicity of the data implies that the Fourier domain is also discrete (since only waves $e^{ikt}$ with integer wavenumber $k$ have period $2\pi$).

Thus, the *discrete Fourier transform* (DFT) is given by

$$\hat{\boldsymbol{y}}_k = h\sum_{j=1}^{N} e^{-ikt_j}\boldsymbol{y}_j, \qquad k = -\frac{N}{2}+1, \ldots, \frac{N}{2}. \tag{107}$$

Note that the (continuous) Fourier domain $[\pi/h, \pi/h]$ used earlier now translates to the discrete domain noted in (107) since $h = 2\pi/N$ is equivalent to $\pi/h = N/2$.

The formula for the *inverse discrete Fourier transform* (inverse DFT) is given by

$$\boldsymbol{y}_j = \frac{1}{2\pi}\sum_{k=-N/2+1}^{N/2} e^{ikt_j}\hat{\boldsymbol{y}}_k, \qquad j = 1, \ldots, N. \tag{108}$$

We obtain the spectral derivative of the finite vector data by exactly the same procedure as in the previous subsection. First, we need the band-limited interpolant of the data. It is given by the formula

$$p(t) = \frac{1}{2\pi}\sum_{k=-N/2}^{N/2\,\prime} e^{ikt}\hat{\boldsymbol{y}}_k, \qquad t \in [0, 2\pi]. \tag{109}$$

Here we *define* $\hat{\boldsymbol{y}}_{-N/2} = \hat{\boldsymbol{y}}_{N/2}$, and the prime on the sum indicates that we add the first and last summands only with weight $1/2$. This modification is required for the band-limited interpolant to work properly.

**Remark** The band-limited interpolant is actually a *trigonometric polynomial* of degree $N/2$, i.e., $p(t)$ can be written as a linear combination of the trigonometric functions $1, \sin t, \cos t, \sin 2t, \cos 2t, \ldots, \sin Nt/2, \cos Nt/2$. We will come back to this fact when we discuss non-periodic data.

Next, we want to represent an arbitrary periodic data vector $\boldsymbol{y}$ as a linear combination of shifts of periodic delta functions. We omit the details here (they can be found in the Trefethen book) and give only the formula for the band-limited interpolant of the periodic delta function:

$$p(t) = S_N(t) = \frac{\sin(\pi t/h)}{(2\pi/h)\tan(t/2)},$$

which is known as the *periodic sinc function* $S_N$.

Now, just as in the previous subsection, the band-limited interpolant for an arbitrary data function can be written as

$$p(t) = \sum_{k=1}^{N} \boldsymbol{y}_k S_N(t - t_k).$$

Finally, using the same arguments and similar elementary calculations as earlier, we get

$$S'_N(t_j) = \begin{cases} 0, & j \equiv 0 \pmod{N}, \\ \frac{1}{2}(-1)^j \cot(jh/2), & j \not\equiv 0 \pmod{N}. \end{cases}$$

These are the entries of the $N$-th column of the Toeplitz matrix (99).

**Example** The Matlab script `SpectralDiffDemo.m` illustrates the use of spectral differentiation for the not so smooth hat function and for the infinitely smooth function $y(t) = e^{\sin t}$.

### 11.3.1 Implementation via FFT

The most efficient computational approach is to view spectral differentiation in the Fourier domain (the alternate approach earlier) and then implement the DFT via the fast Fourier transform (FFT). The general outline is as follows:

1. Sample the function $y$ at the (finite set of) discrete points $t_j$, $j = 1, \ldots, N$ to obtain the data vector $\boldsymbol{y}$ with components $\boldsymbol{y}_j$.

2. Compute the discrete Fourier transform of the (finite) data vector via (107):

$$\hat{\boldsymbol{y}}_k = h \sum_{j=1}^{N} e^{-ikt_j} \boldsymbol{y}_j, \qquad k = -\frac{N}{2} + 1, \ldots, \frac{N}{2}.$$

3. Compute the Fourier transform of the derivative based on (105), i.e.,

$$\widehat{\boldsymbol{y}'}_k = \begin{cases} 0, & k = N/2, \\ ik\hat{\boldsymbol{y}}_k, & \text{otherwise.} \end{cases}$$

4. Find the derivative vector via inverse discrete Fourier transform (see (108)), i.e.,

$$\boldsymbol{y}'_j = \frac{1}{2\pi} \sum_{k=-N/2+1}^{N/2} e^{ikt_j} \widehat{\boldsymbol{y}'}_k, \qquad j = 1, \ldots, N.$$

**Remark** Cooley and Tukey (1965) are usually given credit for discovering the FFT. However, the same algorithm was already known to Gauss (even before Fourier completed his work on what is known today as the Fourier transform). A detailed discussion of this algorithm goes beyond the scope of this course. We simply use the Matlab implementations `fft` and `ifft`. These implementations are based on the current state-of-the-art FFTW algorithm (the "fastest Fourier transform in the West") developed at MIT by Matteo Frigo and Steven G. Johnson.

**Example** The Matlab script `SpectralDiffFFTDemo.m` is an FFT version of the earlier script `SpectralDiffDemo.m`. The FFT implementation is considerably faster than the implementation based on differentiation matrices (see Computer Assignment 5).

## 11.4 Smoothness and Spectral Accuracy

Without getting into any details (see Chapter 4 of Trefethen's book) we will simply illustrate with a few examples the basic behavior of spectral differentiation:

> The smoother the data, the more accurate the spectral derivative.

**Example** In the Matlab script `SpectralAccuracyDemo.m` we expand on the earlier script `SpectralDiffDemo.m` and illustrate the dependence of the convergence rate of spectral differentiation on the smoothness of the data more clearly for the four periodic functions on $[0, 2\pi]$

$$\begin{aligned} y_1(t) &= |\sin t|^3, \\ y_2(t) &= \exp(-\sin^{-2}(t/2)), \\ y_3(t) &= \frac{1}{1+\sin^2(t/2)}, \\ y_4(t) &= \sin(10t). \end{aligned}$$

These functions are arranged according to their (increasing) smoothness. The function $y_1$ has a third derivative of bounded variation, $y_2$ is infinitely differentiable (but not analytic), $y_3$ is analytic in the strip $|\mathrm{Im}(t)| < 2\ln(1+\sqrt{2})$ in the complex plane, and $y_4$ is band-limited.
**Note:** A continuous function $y$ is of bounded variation if

$$\sup_{t_0<t_1<\cdots<t_N} \sum_{j=1}^{N} |y(t_j) - y(t_{j-1})|$$

is bounded for all choices of $t_0, t_1, \ldots, t_N$. Plainly said, a function of bounded variation cannot "wiggle around too much". For example, on the interval $[0, 1/2]$ the function $y(t) = t^2 \sin(1/t)$ is of bounded variation while $y(t) = t \sin(1/t)$ is not.

## 11.5 Polynomial Interpolation and Clustered Grids

We already saw in the Matlab script `BandLimitedDemo.m` that a spectral interpolant performs very poorly for non-smooth functions. Thus, if we just went ahead and treated a problem on a bounded domain as a periodic problem via periodic extension, then the resulting jumps that may arise at the endpoints of the original interval would lead to Gibbs phenomena and a significant degradation of accuracy. Therefore, we do not use the trigonometric polynomials (discrete Fourier transforms) but algebraic polynomials instead.

For interpolation with algebraic polynomials we saw at the very beginning of this course (in the Matlab script `PolynomialInterpolationDemo.m`) the effect that different distributions of the interpolation nodes in a bounded interval have on the accuracy of the interpolant (the so-called *Runge phenomenon*). Clearly, the accuracy is much improved if the points are clustered near the endpoints of the interval. In fact, the so-called *Chebyshev points*

$$t_j = \cos(j\pi/N), \qquad j = 0, 1, \ldots, N$$

yield a set of such clustered interpolation nodes on the standard interval $[-1, 1]$. These points can easily be mapped by a linear transformation to any other interval $[a, b]$ (see Assignment 8). Chebyshev points arise often in numerical analysis. They are the extremal points of the so-called *Chebyshev polynomials* (a certain type of *orthogonal polynomial*). In fact, Chebyshev points are *equally spaced on the unit circle*, and therefore one can observe a nice connection between spectral differentiation on bounded intervals with Chebyshev points and periodic problems on bounded intervals as described earlier. It turns out that (contrary to our expectations) the FFT can also be used for the Chebyshev case. However, we will only consider Chebyshev differentiation matrices below.

## 11.6 Chebyshev Differentiation Matrices

Our last step in our preparation for the solution of general boundary value problems is to determine the entries of the differentiation matrices to be used for problems on bounded intervals (with non-periodic data).

As before, we follow our well-established approach for spectral differentiation:

1. Discretize the interval $[-1, 1]$ using the Chebyshev points

   $$t_j = \cos(j\pi/N), \qquad j = 0, 1, \ldots, N,$$

   and sample the function $y$ at those points to obtain the data vector $\boldsymbol{y} = [y(t_0), y(t_1), \ldots, y(t_N)]^T$.

2. Find the (algebraic) polynomial $p$ of degree at most $N$ that interpolates the data, i.e., s.t.

   $$p(t_i) = \boldsymbol{y}_i, \qquad i = 0, 1, \ldots, N.$$

3. Obtain the spectral derivative vector $\boldsymbol{y}'$ by differentiating $p$ and evaluating at the grid points:

   $$\boldsymbol{y}'_i = p'(t_i), \qquad i = 0, 1, \ldots, N.$$

This procedure (implicitly) defines the differentiation matrix $D_N$ that gives us

$$\boldsymbol{y}' = D_N \boldsymbol{y}.$$

Before we look at the general formula for the entries of $D_N$ we consider some simple examples.

**Example** For $N = 1$ we have the two points $t_0 = 1$ and $t_1 = -1$, and the interpolant is given by

$$\begin{aligned} p(t) &= \frac{t - t_1}{t_0 - t_1}\boldsymbol{y}_0 + \frac{t_0 - t}{t_0 - t_1}\boldsymbol{y}_1 \\ &= \frac{t+1}{2}\boldsymbol{y}_0 + \frac{1-t}{2}\boldsymbol{y}_1. \end{aligned}$$

The derivative of $p$ is (the constant)

$$p'(t) = \frac{1}{2}\boldsymbol{y}_0 - \frac{1}{2}\boldsymbol{y}_1,$$

so that we have

$$\boldsymbol{y}' = \begin{bmatrix} \frac{1}{2}\boldsymbol{y}_0 - \frac{1}{2}\boldsymbol{y}_1 \\ \frac{1}{2}\boldsymbol{y}_0 - \frac{1}{2}\boldsymbol{y}_1 \end{bmatrix}$$

and the differentiation matrix is given by

$$D_1 = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \end{bmatrix}.$$

**Example** For $N = 2$ we start with the three Chebyshev points $t_0 = 1$, $t_1 = 0$, and $t_2 = -1$. The quadratice interpolating polynomial (in Lagrange form) is given by

$$\begin{aligned} p(t) &= \frac{(t-t_1)(t-t_2)}{(t_0-t_1)(t_0-t_2)}\boldsymbol{y}_0 + \frac{(t-t_0)(t-t_2)}{(t_1-t_0)(t_1-t_2)}\boldsymbol{y}_1 + \frac{(t-t_0)(t-t_1)}{(t_2-t_0)(t_2-t_1)}\boldsymbol{y}_2 \\ &= \frac{t(t+1)}{2}\boldsymbol{y}_0 - (t-1)(t+1)\boldsymbol{y}_1 + \frac{(t-1)t}{2}\boldsymbol{y}_2. \end{aligned}$$

Now the derivative of $p$ is a linear polynomial

$$p'(t) = \left(t + \frac{1}{2}\right)\boldsymbol{y}_0 - 2t\boldsymbol{y}_1 + \left(t - \frac{1}{2}\right)\boldsymbol{y}_2,$$

so that – evaluating at the nodes – we have

$$\boldsymbol{y}' = \begin{bmatrix} \frac{3}{2}\boldsymbol{y}_0 - 2\boldsymbol{y}_1 + \frac{1}{2}\boldsymbol{y}_2 \\ \frac{1}{2}\boldsymbol{y}_0 - \frac{1}{2}\boldsymbol{y}_2 \\ -\frac{1}{2}\boldsymbol{y}_0 + 2\boldsymbol{y}_1 - \frac{3}{2}\boldsymbol{y}_2 \end{bmatrix}$$

and the differentiation matrix is given by

$$D_2 = \begin{bmatrix} \frac{3}{2} & -2 & \frac{1}{2} \\ \frac{1}{2} & 0 & -\frac{1}{2} \\ -\frac{1}{2} & 2 & -\frac{3}{2} \end{bmatrix}.$$

We note that the differentiation matrices no longer are Toeplitz or circulant. Instead, the entries satisfy (also in the general case below)

$$(D_N)_{ij} = -(D_N)_{N-i,N-j}.$$

For general $N$ one can prove

**Theorem 11.1** *For each $N \geq 1$, let the rows and columns of the $(N+1) \times (N+1)$ Chebyshev spectral differentiation matrix $D_N$ be indexed from 0 to $N$. The entries of this matrix are*

$$\begin{aligned}
(D_N)_{00} &= \frac{2N^2+1}{6}, \qquad (D_N)_{NN} = -\frac{2N^2+1}{6}, \\
(D_N)_{jj} &= \frac{-t_j}{2(1-t_j^2)}, \qquad j = 1, \ldots, N-1, \\
(D_N)_{ij} &= \frac{c_i}{c_j}\frac{(-1)^{i+j}}{(t_i - t_j)}, \qquad i \neq j, \quad i,j = 0,1,\ldots,N,
\end{aligned}$$

*where*

$$c_i = \begin{cases} 2, & i = 0 \text{ or } N, \\ 1, & \text{otherwise.} \end{cases}$$

This matrix is implemented in the Matlab script `cheb.m` that was already used in the Matlab function `PSBVP.m` that we used in our motivational example `PSBVPDemo.m` at the beginning of this chapter. Note that only the off-diagonal entries are computed via the formulas given in the theorem. For the diagonal entries the formula

$$(D_N)_{ii} = -\sum_{\substack{j=0 \\ j \neq i}}^{N} (D_N)_{ij}$$

was used.

**Example** The spectral accuracy of Chebyshev differentiation matrices is illustrated in the Matlab script `ChebyshevAccuracyDemo.m`. One should compare this to the earlier script `SpectralAccuracyDemo.m` in the periodic case.

The functions used for the Chebyshev example are

$$\begin{aligned}
y_1(t) &= |t|^3, \\
y_2(t) &= \exp(-t^{-2}), \\
y_3(t) &= \frac{1}{1+t^2}, \\
y_4(t) &= t^{10}.
\end{aligned}$$

These functions are again arranged according to their (increasing) smoothness. The function $y_1$ has a third derivative of bounded variation, $y_2$ is infinitely differentiable (but not analytic), $y_3$ is analytic in $[-1,1]$, and $y_4$ is a polynomial (which corresponds to the band-limited case earlier).

Note that the error for the derivative of the function $y_2$ dips to zero for $N = 2$ since the true derivative is given by

$$y_2'(t) = 2\frac{\exp(-t^{-2})}{t^3},$$

and the values at $t_0 = 1$, $t_1 = 0$, and $t_2 = -1$ are $2/e$, 0, and $-2/e$, respectively. These all lie on a line (the linear derivative of the quadratic interpolating polynomial).

## 11.7 Boundary Value Problems

We can now return to our introductory example, the 2-pt boundary value problem

$$y''(t) = e^{4t}, \quad t \in (-1, 1)$$

with boundary conditions $y(-1) = y(1) = 0$. Its analytic solution was given earlier as

$$y(t) = \left[e^{4t} - t\sinh(4) - \cosh(4)\right]/16.$$

How do we solve this problem in the Matlab programs `PSBVPDemo.m` and `PSBVP.m`?

First, we note that – for Chebyshev differentiation matrices – we can obtain higher derivatives by repeated application of the matrix $D_N$, i.e., if

$$\boldsymbol{y}' = D_N\boldsymbol{y},$$

then

$$\boldsymbol{y}'' = D_N\boldsymbol{y}' = D_N^2\boldsymbol{y}.$$

In other words, for Chebyshev differentiation matrices

$$D_N^{(k)} = D_N^k, \quad k = 1, \ldots, N,$$

and $D_N^{N+1} = 0$.

**Remark** We point out that this fact is true only for the Chebyshev case. For the Fourier differentiation matrices we established in the periodic case we in general have $D_N^k \neq D_N^{(k)}$ (see Assignment 8).

With the insight about higher-order Chebyshev differentiation matrices we can view the differential equation above as

$$D_N^2\boldsymbol{y} = \boldsymbol{f},$$

where the right-hand side vector $\boldsymbol{f} = \exp(4\boldsymbol{t})$, with $\boldsymbol{t} = [t_0, t_1, \ldots, t_N]^T$ the vector of Chebyshev points. This *linear system*, however, cannot be solved uniquely (one can show that the matrix $(N + 1) \times (N + 1)$ matrix $D_N^2$ has an $(N + 1)$-fold eigenvalue of zero). Of course, this is not a problem. In fact, it is reassuring, since we have not yet taken into account the boundary conditions, and the ordinary *differential equation* (without appropriate boundary conditions) also does not have a unique solution.

So the final question is, how do we deal with the boundary conditions?

We could follow either of two approaches. First, we can build the boundary conditions into the spectral interpolant, i.e.,

1. Take the *interior* Chebyshev points $t_1, \ldots, t_{N-1}$ and form the polynomial interpolant of degree at most $N$ that satisfies the boundary conditions $p(-1) = p(1) = 0$ and interpolates the data vector at the interior points, i.e., $p(t_j) = \boldsymbol{y}_j$, $j = 1, \ldots, N-1$.

2. Obtain the spectral derivative by differentiating $p$ and evaluating at the interior points, i.e.,
$$\boldsymbol{y}''_j = p''(t_j), \quad j = 1, \ldots, N-1.$$

3. Identify the $(N-1) \times (N-1)$ matrix $\widetilde{D}_N^2$ from the previous relation, and solve the linear system
$$\widetilde{D}_N^2 \boldsymbol{y}(1:N-1) = \exp(4\boldsymbol{t}(1:N-1)),$$
where we used Matlab-like notation.

The second approach is much simpler to implement, but not as straightforward to understand/derive. Since we already know the value of the solution at the boundary, i.e., $\boldsymbol{y}_0 = 0$ and $\boldsymbol{y}_N = 0$, we do not need to include these values in our computation. Moreover, the values of the derivative at the endpoints are of no interest to us. Therefore, we can simply solve the linear system
$$\widetilde{D}_N^2 \boldsymbol{y}(1:N-1) = \exp(4\boldsymbol{t}(1:N-1)),$$
where
$$\widetilde{D}_N^2 = D_N^2(1:N-1, 1:N-1).$$
This is exactly what was done in the Matlab program `PSBVP.m`.

**Remark** One can show that the eigenvalues of $\widetilde{D}_N^2$ are given by $\lambda_n = -\frac{\pi^2 n^2}{4}$, $n = 1, 2, \ldots, N-1$. Clearly, these values are all nonzero, and the problem has (as it should have) a unique solution.

We are now ready to deal with more complicated boundary value problems. They can be nonlinear, have non-homogeneous boundary conditions, or mixed-type boundary conditions with derivative values specified at the boundary. We give examples for each of these cases.

**Example** As for our initial value problems earlier, a nonlinear ODE-BVP will be solved by iteration (either fixed-point, or Newton).

Consider
$$y''(t) = e^{y(t)}, \quad t \in (-1, 1)$$
with boundary conditions $y(-1) = y(1) = 0$. In the Matlab program `NonlinearPSBVPDemo.m` we use fixed-point iteration to solve this problem.

**Example** Next, we consider a linear BVP with non-homogeneous boundary conditions:
$$y''(t) = e^{4t}, \quad t \in (-1, 1)$$
with boundary conditions $y(-1) = 0$, $y(1) = 1$. In the Matlab program `PSBVPNonHomoBCDemo.m` this is simply done by replacing the first and last rows of the differentiation matrix `D2` by corresponding rows of the identity matrix and then imposing the boundary values in the first and last entries of the right-hand side vector `f`.

**Example** For a linear BVP with mixed boundary conditions such as

$$y''(t) = e^{4t}, \quad t \in (-1, 1)$$

with boundary conditions $y'(-1) = y(1) = 0$ we can follow the same strategy as in the previous example. Now, however, we need to replace the row of `D2` that corresponds to the derivative boundary condition with a row from the first-order differentiation matrix `D`. This leads to the Matlab program `PSBVPMixedBCDemo.m`.