ShapeNotes

1 One Dimensional Considerations

Pseudospectral methods on non-periodic domains are based on polynomial interpolation on non-equispaced points. Typically, Chebyshev points $\{x_j\}$ are chosen as collocation points on $[-1, 1]$, which are defined as:

$$x_j = \cos\left(\frac{j\pi}{N}\right), \quad j = 0, 1, ..., N, \tag{1}$$

see [1]. These points are clustered at the endpoints of the interval, and sparse around 0. Using this approach, the points are distributed from 1 to -1, which is counter-intuitive. Therefore, in the code library [2], which is used in producing the results of this report, the Chebyshev points are automatically flipped back to run from -1 to 1. All computations in 2DChebClass are done on this computational domain $[-1, 1]$. This can then be mapped onto a physical domain $[a, b]$ of interest vial a linear map.

The function of interest, f, is evaluated at the Chebyshev points $\{x_j\}$ and a grid function, $f_j := f(x_j)$, is defined. There exists a unique polynomial of degree $\leq N$ that can be used to interpolate a function f on the grid points x_j. The polynomial p_N satisfies, by definition, the following relationship

$$p_N(x_j) = f_j, \tag{2}$$

so that the residual $p_N(x_j) - f_j$ is zero at these points. Therefore, this method is called a collocation method, see [3]. Interpolation on the Chebyshev grid is done using barycentric Lagrange interpolation, derived in [4]. The barycentric formula is

$$p_N(x) = \frac{\displaystyle\sum_{k=0}^{N} \frac{\tilde{w}_j}{x - x_j} f(x_j)}{\displaystyle\sum_{j=0}^{N} \frac{\tilde{w}_j}{x - x_j}},$$

where the weights are defined as

$$\tilde{w}_j = (-1)^j d_j, \quad d_j = \begin{cases} \frac{1}{2} \text{ for } j = 0,\ j = N, \\ 1 \text{ otherwise}, \end{cases}$$

see [4] and [2]. In the code library 2DChebClass, this is implemented as a matrix-vector product, interpolating from the set of points $\{x_j\}$, onto another set of points $\{x_i\}$, where $i = 0, ..., M$

and $j = 0, ..., N$. The interpolation matrix is of the form

$$\text{Interp}_{ij} = \frac{1}{\omega_i}\left(\frac{\tilde{w}_j}{x_i - x_j}\right), \qquad \omega_i = \sum_{k=0}^{N} \frac{\tilde{w}_k}{x_i - x_k}.$$

Generally, $\{x_j\}$ lies in computational space $[-1, 1]$, while the second set of points lies in computational space by default, but can be customised by the user, to be a set of M points in $[a, b]$. There is another function in 2DChebClass, which takes a set $\{x_j\}$ in physical space instead, but with N points still. This set of points is then first mapped onto the computational domain, using the linear map, before the interpolation matrix is computed, as described above. This method can then be applied to interpolating an arbitrary set of values $\{f_j\}$ onto the new set of points $\{f_i\}$.

The derivation of the Chebyshev differentiation matrices is described below, following the presentation in [1]. Given a polynomial p_N (++definitely same as above??++), where condition (2) holds, it can be differentiated so that $f_j' = p'(x_j)$, which can be rewritten as a multiplication of f_j by a $(N+1) \times (N+1)$ matrix, denoted by D, as follows

$$f_j' = Df_j,$$

using (2). A $(N+1) \times (N+1)$ differentiation matrix D has the following entries, compare with [1]

$$\begin{aligned}
(D)_{00} &= \frac{2N^2+1}{6}, \\
(D)_{NN} &= -\frac{2N^2+1}{6}, \\
(D)_{jj} &= -\frac{x_j}{2(1-x_j^2)}, \quad j = 1, ..., N-1, \\
(D)_{ij} &= \frac{c_i}{c_j}\frac{(-1)^{i+j}}{(x_i - x_j)}, \quad i \neq j, \quad i, j = 0, ..., N,
\end{aligned}$$

where

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

The second derivative is represented by the second differentiation matrix D_2, which can be found by squaring the first differentiation matrix; $D_2 = D^2$, and more generally the j^{th} differentiation matrix is found as follows

$$D_j = D^j.$$

However, in [2], the exact coefficients, derived in a similar way as above for D, are used to compute D_2, since it is more accurate than squaring D. Furthermore, all differentiation matrices are derived in computational space and then mapped to physical space using a Jacobian transformation. (++ details?++)

In order to evaluate integrals in a similar way, the so-called Clenshaw–Curtis quadrature is used, which is derived in [5]. This is, for the integral over a smooth function f:

$$\int_{-1}^{1} f(x)dx = \sum_{k=0}^{N} w_k f(x_k), \tag{3}$$

where the weights are defined as:

$$w_j = \frac{2d_j}{N} \begin{cases} 1 - \displaystyle\sum_{k=1}^{(N-2)/2} \frac{2\cos(2kt_j)}{4k^2-1} - \frac{\cos(\pi j)}{N^2-1} & \text{for } N \text{ even,} \\ 1 - \displaystyle\sum_{k=1}^{(N-2)/2} \frac{2\cos(2kt_j)}{4k^2-1} & \text{for } N \text{ odd,} \end{cases}$$

see [2]. In 2DChebClass this is implemented as a vector, such that $(\text{Int})_k = w_k$. Again, this is done in computational space, so that a Jacobian transformation has to be taken to map this onto a desired physical domain. A dot product can be taken between a vector with entries f_j and the resulting integration vector.

The final aspect to be considered is the convolution matrix, which is needed to compute convolution integrals. The convolution integral is defined as:

$$(n \star \chi)(y) = \int \chi(y - \tilde{y}) n(\tilde{y}) d\tilde{y}.$$

As explained in [2] in detail, χ is evaluated for all points in the domain at $y - \tilde{y}$ and transformed into a matrix $\text{diag}(\chi)$. This is integrated, using the integration vector, to result in the convolution matrix, which can then be applied to a density n. This implies that the convolution matrix can be precomputed and applied to different n, which saves computational time.

2 Two Dimensional Considerations

In order to extend the definition of the differentiation matrices to two-dimensional grids, a so-called tensor product grid has to be defined. First, Chebyshev points x_j, for $j = 1, ..., n$, on the x-axis and another set of Chebyshev points y_i, for $i = 1, ..., m$ on the y-axis are taken, both between $[-1,1]$. Then the following two vectors are defined:

$$\begin{aligned}\mathbf{x}_K &= (x_1, x_1, ..., x_1, x_2, x_2, ..., x_2, ..., x_n, x_n, ..., x_n)^T,\\ \mathbf{y}_K &= (y_1, y_2, ..., y_m, y_1, y_2 ..., y_m, ..., y_1, y_2, ..., y_m)^T.\end{aligned}$$

In $\mathbf{x}_K$, each x_j is repeated m times, while $\mathbf{y}_K$, each sequence $y_1, y_2, ..., y_m$ is repeated n times. The total length of each vector is $n \times m$. These vectors are defined, so that the set $(\mathbf{x}_K, \mathbf{y}_K)$ is a full set of all Chebyshev points on the two-dimensional tensor grid. Note that the points are clustered around the boundary of the two-dimensional grid and sparse in the middle of the grid. These Kronecker vectors can be used to find the Chebyshev differentiation matrices for two-dimensional problems as follows, compare to [1]. For a first derivative D in the x direction, a Kronecker product is taken of the one-dimensional Chebyshev differentiation matrix with the identity, as demonstrated here with three points:

$$D_x = I \otimes D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{pmatrix}$$

$$= \begin{pmatrix} d_{11} & d_{12} & d_{13} & & & & & & \\ d_{21} & d_{22} & d_{23} & & & & & & \\ d_{31} & d_{32} & d_{33} & & & & & & \\ & & & d_{11} & d_{12} & d_{13} & & & \\ & & & d_{21} & d_{22} & d_{23} & & & \\ & & & d_{31} & d_{32} & d_{33} & & & \\ & & & & & & d_{11} & d_{12} & d_{13} \\ & & & & & & d_{21} & d_{22} & d_{23} \\ & & & & & & d_{31} & d_{32} & d_{33} \end{pmatrix},$$

where the block structure matches the repetition of each x_j in $\mathbf{x}_K$. The second derivative with respect to x, D_{xx} can be found by using D_2 instead of D in this calculation. The derivative with respect to y is found by taking the Kronecker product the other way around:

$$D_y = D \otimes I = \begin{pmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} d_{11} & & & d_{12} & & & d_{13} & & \\ & d_{11} & & & d_{12} & & & d_{13} & \\ & & d_{11} & & & d_{12} & & & d_{13} \\ d_{21} & & & d_{22} & & & d_{23} & & \\ & d_{21} & & & d_{22} & & & d_{23} & \\ & & d_{21} & & & d_{22} & & & d_{23} \\ d_{31} & & & d_{32} & & & d_{33} & & \\ & d_{31} & & & d_{32} & & & d_{33} & \\ & & d_{31} & & & d_{32} & & & d_{33} \end{pmatrix},$$

which now matches the repetition of each $y_1, ...y_m$ in $\mathbf{y}_K$. The Chebyshev differentiation of the Laplacian is given by: $L = I \otimes D_2 + D_2 \otimes I$.

The advantage of Spectral Methods is that, for smooth functions, the convergence is exponential, see [3]:

$$\text{Pseudospectral Error} \approxeq O\left[\left(\frac{1}{N}\right)^N\right].$$

A good overview on spectral methods is given in [1] and a more in depth discussion can be found in [3].

References

[1] Lloyd N. Trefethen. *Spectral Methods in Matlab*. SIAM, 2000.

[2] Andreas Nold, Benjamin D. Goddard, Peter Yatsyshin, Nikos Savva, and Serafim Kalliadasis. Pseudospectral methods for density functional theory in bounded and unbounded domains. *CoRR*, abs/1701.06182, 2017.

[3] John P. Boyd. *Chebyshev and Fourier Spectral Methods*. Dover Publications, Inc, 2000.

[4] Jean-Paul Berrut and Lloyd N. Trefethen. Barycentric Lagrange interpolation. *SIAM Review*, 46(3):501–517, 2004.

[5] Charles W. Clenshaw and A. R. Curtis. A method for numerical integration on an automatic computer. *Numerische Mathematik*, 2(1):197–205, 1960.