

Spectral Methods

1 Problem

Many problems can be solved more easily or more accurately by changing the basis of representation of the data, usually in terms of sets of orthogonal basis functions. How can we write efficient parallel implementations of codes based on these methods?

2 Context

“Spectral Methods” are a class of spatial discretizations for differential equations: they provide a way of translating an equation expressed in continuous space and time into a discrete equation which can be solved numerically. There are three primary benefits of Spectral Methods over alternative approaches, such as Finite Elements or Finite Differences. Spectral discretizations of differential equations, based for example on Fourier or Chebyshev series, provide very low-error approximations. In many cases, these approaches can be “exponentially convergent” – for a length N expansion, the difference of an analytical solution and the numerical solution can be $O(\frac{1}{N}^N)$. Second, since the numerical accuracy of spectral methods is so high the number of grid points required to achieve the desired precision can be very low, thus a spectral method may require less memory than alternative methods. This “memory-minimizing” property can be crucial, as the memory footprint of an algorithm has a first-order effect on its runtime on modern parallel processors. Finally, there exist high-performance implementations of the algorithms required to transform bases for most Spectral Methods, and the developer of a Spectral Method code need not implement these codes. For example, nearly all numerical libraries include a Fast Fourier Transform.

The purpose of changing the representation need not be solely to minimize the error of a discretization. Analysis of the spectrum of a signal is the most natural and powerful approach for solving many signal processing problems. Thus, while signal processing deals only with discrete data and the discussion of discretization does not apply, spectral methods are appropriate. Additionally, the spectrum of a dataset may be useful for visualization and other data inspection. Thus it is not necessary that data be transformed in both directions: the goal may only be the transformation into spectral components.

The natural basis of representation for a problem is usually in terms of variables with immediate physical significance: for example, spatial coordinates, time, density, temperature, and velocity. Methods exist that solve problems directly in this representation. For example, Finite Differences approximate derivatives via differences of elements within a discretized grid. The Finite Element Method approximates functions appearing in a differential equation with piecewise continuous functions. Spectral methods, however, represent functions

via sums of global, infinitely differentiable, orthogonal functions (e.g. Fourier Series).

A complete discussion of the different classes of Spectral Methods and their virtues is out of the scope of this pattern. The cited texts provide a comprehensive coverage of these topics. The goal of this pattern is to identify the potential for parallelism in codes developed based on Spectral Methods, and identify other pertinent patterns in the Pattern Language.

3 Solution

The crux of this pattern is to realize that, after the change of basis, the opportunities for parallelism are the same as those discussed in the *Structured Grid* and *Dense Linear Algebra*, or the *Unstructured Grid* and *Sparse Linear Algebra* patterns. Which pairing is most useful depends on the structure of the underlying problem. For clarity and pedagogical value, this pattern will focus on problems which map to dense, structured grids. As such, most of the content of this pattern resides in the Examples section. The goal of the examples is to illustrate the presence of the aforementioned patterns of an equation discretized via a Spectral Method.

The following simplified example illustrates this point well. Consider, for example, a periodic function of two variables $g(x, y) = g(x+2\pi, y) = g(x, y+2\pi)$. Suppose that we wish to find a function $f(x, y)$ that is a solution to Laplace's Equation:

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) f(x, y) = g(x, y) \quad (1)$$

If we represent the functions f and g via their Fourier Series, we can obtain a numerical algorithm for this problem. First, we obtain the Fourier Coefficients for our functions. That is, we approximate f and g by sums of trigonometric polynomials. ke

$$\begin{aligned} f &= \sum_{j,k} \mathbf{f}_{j,k} e^{2\pi i(jx+ky)} \\ g &= \sum_{j,k} \mathbf{g}_{j,k} e^{2\pi i(jx+ky)} \end{aligned} \quad (2)$$

With $\iota = \sqrt{-1}$ The coefficients $\mathbf{g}_{j,k}$ can be computed via a Discrete Fourier Transform (DFT), fast implementations of which are available for nearly any platform. Solving for the $\mathbf{f}_{j,k}$ and performing an Inverse DFT (IDFT) will provide us a numerical approximation to f . The x and y 2nd-order partial derivatives translate into analytical differentiation of the exponentials:

$$\begin{aligned} \frac{\partial^2}{\partial x^2} f(x, y) &= \sum_{j,k} \mathbf{f}_{j,k} (2\pi \iota j)^2 e^{2\pi i(jx+ky)} \\ \frac{\partial^2}{\partial y^2} f(x, y) &= \sum_{j,k} \mathbf{f}_{j,k} (2\pi \iota k)^2 e^{2\pi i(jx+ky)} \end{aligned}$$

And, after some simplification, equation (1) translates into:

$$-4\pi^2 \sum_{j,k} \mathbf{f}_{j,k} (j^2 + k^2) e^{2\pi i(jx+ky)} = \sum_{j,k} \mathbf{g}_{j,k} e^{2\pi i(jx+ky)} \quad (3)$$

This equation is now entirely algebraic. Using the fact that the Fourier basis functions are orthogonal, we can solve individually for the coefficients $\mathbf{f}_{j,k}$:

$$\mathbf{f}_{j,k} = \frac{\mathbf{g}_{j,k}}{-4\pi^2(j^2 + k^2)} \quad (4)$$

The corresponding function f can be computed via an IDFT – i.e. Equation (2). Note that the computation of Equation (4) is an element-wise operation on a dense grid, and thus the discussions in the *Structured Grid* and *Geometric Decomposition* patterns apply. The software implementation would involve a 2-Dimensional loop over the indices j and k , and thus the *Loop Parallelism* and *Data Parallelism* patterns pertain.

The computation of the transforms (in this example, the Discrete Fourier Transforms) can also be parallelized. In fact, the existence of the Fast Fourier Transform (FFT) algorithm and the abundance of high-performance implementations are crucial to the efficacy of Spectral Methods. The most prudent approach is to seek out a pre-existing parallel implementation of the desired transform. Platform library developers have invested large amounts of effort into ensuring high performance of FFT libraries, and usually achieve near-peak performance. We will not discuss the performance issues of transformation algorithms further.

4 Examples

We will discuss two different differential equations discretized via a Fourier-Galerkin method. This is only one example of a spectral method: we cannot hope to cover all methods of interest, as that is the domain of the cited texts.

Both of our examples are evolution equations of the form:

$$\frac{\partial u}{\partial t} - \mathcal{M}(u) = 0 \quad (5)$$

That is, the first-order time derivative of our function u is equal to some linear differential operator \mathcal{M} applied to u . As is common practice, we will use Spectral approximations for the spatial derivatives only, and use a different discretization method for the time variable. For both of our examples we will assume periodic boundary conditions, and thus we will use the trigonometric polynomials (i.e. Fourier Series) as our basis functions. $\phi_k(x)$ are the basis functions used to approximate our function u , and $\psi_k(x)$ are the basis functions used to evaluate our result:

$$\phi_k(x) = e^{ikx}$$

$$\psi_k(x) = \frac{1}{2\pi} e^{-ikx}$$

Thus we construct an N^{th} order approximation u^N :

$$u^N(x, 1) = \sum_{k=-N/2}^{N/2} a_k(t) \phi_k(x) \quad (6)$$

We will construct our solution equations by requiring that the integration of the *residual* $\frac{\partial u^N}{\partial t} - \mathcal{M}(u^N)$ against our test functions ψ_k be zero:

$$\int_0^{2\pi} \left[\frac{\partial u^N}{\partial t} - \mathcal{M}(u^N) \right] \psi_k(x) dx = 0 \quad (7)$$

4.1 Linear Hyperbolic Equation

$$\frac{\partial u}{\partial t} - \frac{\partial u}{\partial x} = 0$$

The constraint posed by (7) become:

$$\frac{1}{2\pi} \int_0^{2\pi} \left[\left(\frac{\partial}{\partial t} - \frac{\partial}{\partial x} \right) \sum_{l=-N/2}^{N/2} a_l(t) e^{ilx} \right] e^{-ikx} dx = 0$$

Where the a_l are the coefficients from (6). Taking the spatial derivative of the trial functions $\phi_k(x) = e^{ikx}$:

$$\frac{1}{2\pi} \int_0^{2\pi} \left[\sum_{l=-N/2}^{N/2} \left(\frac{da_l}{dt} - ila_l \right) e^{ilx} \right] e^{-ikx} dx = 0 \quad (8)$$

Note that our trial functions $\phi_l = e^{ilx}$ and test functions $\psi_k = \frac{1}{2\pi} e^{-ikx}$ are orthogonal:

$$\int_0^{2\pi} \phi_l(x) \psi_k(x) dx = \delta_{l,k} \quad (9)$$

Where δ_{kl} is 1 iff $k = l$, and 0 otherwise. Thus the integral (8) simplifies into the dynamical equations:

$$\frac{da_k}{dt} - ik a_k = 0$$

For $k = -N/2 \dots N/2$. The initial conditions for the a_k are the coefficients for the expansion of the initial condition of $u(x)$. For the Galerkin approximation:

$$a_k(0) = \int_0^{2\pi} u(x, 0) \psi_k(x) dx \quad (10)$$

There are a variety of approaches for time discretization. The simplest is the *Forward Euler* (FE) method, which provides a linear approximation of the time derivative. Given a time-discretization setp Δt we have the equation:

$$a_k(t + \Delta t) = a_k(t) + \Delta t k a_k(t) \quad (11)$$

The program to implement this would first perform a DFT to find the coefficients $a_k(0)$ specified by (10), evolve the coefficients over the desired length of time as specified by (11), and perform an IDFT on a_k fo retrieve the approximation to u . The code to be written by the developer is only the evolution (11), which consists of a doubly-nested loop over t and k .

4.2 Burgers Equation

It is not always a simple matter of “DFT, evolve, IDFT”. In particular, when the equation involves nonlinearities, efficiently computing the evolution may require transformation of particular terms. The nonlinear Burgers Equation is a simplified model of Navier-Stokes turbulence:

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} - \nu \frac{\partial^2 u}{\partial x^2} = 0$$

We will again construct an approximation function u^N via the truncated Fourier series, i.e. $\phi_k = e^{ikx}$:

$$u^N(x, t) = \sum_{k=-N/2}^{N/2-1} \hat{u}_k(t) e^{ikx} \quad (12)$$

And as before, integrate against test functions $\psi_k(x) = \frac{1}{2\pi} e^{-ikx}$:

$$\frac{1}{2\pi} \int_0^{2\pi} \left(\frac{\partial u^N}{\partial t} + u^N \frac{\partial u^N}{\partial x} - \nu \frac{\partial^2 u^N}{\partial x^2} \right) e^{-ikx} dx = 0 \quad (13)$$

For $k = -\frac{N}{2}, \dots, \frac{N}{2} - 1$. Due to the orthogonality of the test and trial functions per Eqn. 9, the integral (13) simplifies to a set of ODEs for the ODEs for the \hat{u}_k :

$$\frac{d\hat{u}_k}{dt} + \left(u^N \frac{\partial u^N}{\partial x} \right)_k + k^2 \nu \hat{u}_k = 0$$

Where the fourier coefficients of the nonlinear product (*advection term*) $u \frac{\partial u}{\partial x}$ is represented by:

$$\left(u^N \frac{\partial u^N}{\partial x}\right)_k^\wedge = \frac{1}{2\pi} \int_0^{2\pi} u^N \frac{\partial u^N}{\partial x} e^{-ikx} dx \quad (14)$$

The difficulty in implementing (14) arises in its expression in terms of the Fourier coefficients of u and $\frac{\partial u}{\partial x}$. The product $u \frac{\partial u}{\partial x}$ in the original equation becomes a *convolution sum* when expressed in terms of the spectral components. If we let $v = \frac{\partial u}{\partial x}$, with Fourier coefficients \hat{v} , (14) becomes:

$$(uv)_k^\wedge = \sum_{p+q=k} \hat{u}_p \hat{v}_q$$

a straightforward implementation of which requires $O(N^2)$ operations. Moreover, such an implementation does *not* fall under the standard definition of *Structured Grid* patterns, as operations are not local and element-wise. Thus an efficient solver for the Burgers equation would need to perform IDFTs on its representations of u and v at each time step, perform the element-wise multiplications in the spatial domain, and then transform the resulting product back into the spectral domain per Eqn. (14). The resulting code again consists solely of calls to the FFT and IFFT library routines, as well as the element-wise operations performed on u and v .

5 Related Patterns

As mentioned above, the *Structured Grid* and *Unstructured Grid* patterns

6 References

C. Canuto, M.Y. Hussaini, A. Quarteroni, T.A. Zang, *Spectral Methods: Fundamentals in Single Domains*, Springer 2006, ISBN: 987-3-540-30725-9 <http://www.dimat.polito.it/chqz/>

John P. Boyd, *Chebyshev and Fourier Spectral Methods*, Springer 2000. Available online.

http://www-personal.umich.edu/~jpboyd/BOOK_Spectral2000.html

7 Author

Mark Murphy