An Introduction to Spectral Methods for Partial Differential Equations

With applications to fluid flow

Tom Stopford

August 21, 2022

1 Introduction

This guide intends to teach the reader the basics of spectral methods as a means for solving partial differential equations (PDEs). The main application that will be taught is that of fluid flow, but this is far from the limit of this powerful technique.

My intention is that this guide can be read and understood by anybody who knows multivariable calculus. However, familiarity with the Navier-Stokes equations, basic numerical ODE solving methods and the Fourier transform will definitely add to the reader's understanding. If you ever see a mistake, find something unclear, or feel that a part of these notes is just not up to scratch, please send me a message via email (tstopford17@gmail.com) and I will try my best to fix it.

With all that being said, I hope this guide helps you.

1.1 Structure of the guide

Through this guide, we will work up to solving the incompressible Navier-Stokes equations in 2-D using spectral methods. These are given by

$$\mathbf{u}_t + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \mathbf{g} + \nu \nabla^2 \mathbf{u} \tag{1}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{2}$$

where $\mathbf{u}(x, y, t)$ denotes the velocity of an eulerian fluid element located at position (x, y) at time t, $p = p(x, y, t)$ is the pressure field, and $\mathbf{g} = \mathbf{g}(x, y, t)$ is the force field. We refer to 1 as *momentum balance* and 2 as *mass balance* or *incompressibility* In this guide, we will not use vector calculus notation in this guide though for the sake of clarity. Expanding out the vector calculus

notation, we get the equations in this form

$$\begin{aligned} u_t + uu_x + vu_y &= -p_x + g_x + \nu\left(u_{xx} + u_{yy}\right) \\ v_t + uv_x + vv_y &= -p_y + g_y + \nu\left(v_{xx} + v_{yy}\right) \\ u_x + v_y &= 0 \end{aligned}$$

Note the subscripts x and y denote derivatives except for g, where they are components. We will be solving these with periodic boundary conditions. This may be used to approximate static or uniform flow at infinity. In general, spectral methods may be used for problems that are not periodic, but that is not yet covered here.

The first chapter will very quickly cover the Fourier transform. The fact that periodic functions may be written as a series of exponentials, whose derivative is easy to compute, is a key fact exploited by spectral methods and a big part of why they are so powerful.

The second chaper will cover some basic methods for numerically solving ODEs. This is important because writing a PDE in terms of Fourier series leaves us with many ODEs which we have to solve by discretising time.

At this stage we will be ready to solve our first PDE. This will be the 1-D heat equation

$$u_t = \nu u_{xx}$$

We will then move on to the 1-D Burgher equation

$$u_t + uu_x = \nu u_{xx}$$

after this we will solve the 2-D Burgher equation

$$\begin{aligned} u_t + uu_x + vu_y &= \nu\left(u_{xx} + u_{yy}\right) \\ v_t + uv_x + vv_y &= \nu\left(v_{xx} + v_{yy}\right) \end{aligned}$$

and then we will cover the projection method to enforce incompressibility. At this point we will have everything we need to simulate fluids in 2-D.

2 A quick introduction to the Fourier transform

Here we will quickly brush over a bit of the maths needed to use spectral methods. The rigour of this is too deep to get into in this guide, but an interested reader should look for resources on Functional analysis and Fourier analysis. Maybe one day I will make something covering these topics, maybe I won't, maybe I already have. That's the magic of the version of me writing these notes and you occupying different points in time.

2.1 Fourier Series

All functions we are interested in approximating will have the following properties:

- Their spatial domain will be the closed interval $[a, b]$ or the closed rectangle $[a, b] \times [c, d]$ and their time domain will be $[0, T]$.
- They are periodic in their spatial domain. That is $f(a, t) = f(b, t)$ for all $t \in [0, T]$ for a function of a 1-D spatial domain, or $f(a, y, t) = f(b, y, t)$ and $f(x, c, t) = f(x, d, t)$ for all $y \in [c, d]$, $x \in [a, b]$ and $t \in [0, T]$ for a function of a 2-D spatial domain.
- They are Lebesgue square-intgrable. In our context we don't need to worry about this, but what it essentially means is that it doesn't have too many discontinuities.

Note. Another way of expressing the third point is that the functions are elements of $L^2(A)$ where $A = [a, b]$ or $[a, b] \times [c, d]$. This is defined to be the space of functions that are integrable and $\int_a |f|^2 < \infty$.

This is a *Hilbert space* and has the following inner product

$$\langle f, g \rangle = \int_A f\overline{g}$$

where $\overline{g}$ is the complex conjugate of g. Typically, $L^2(A)$ denotes the space of such functions that are real valued, while $L^2(A, \mathbb{C})$ denotes the space of such functions that are complex valued.

To read more about Lebesgue spaces and integrability, the reader should look for resources on the topic of measure theory.

We will assume that these properties holds for all functions that we refer to from now on. The following are some key facts about these types of functions that will be instrumental later on. For simplicity, these facts will be stated in the context of functions with one spatial dimension. Very similar versions of them hold for functions with 2 spatial dimensions, but we will deal with them when we get to it.

Key Fact 2.1. All such functions $f = f(x)$ where x is our spatial variable may be uniquely written in the form $f(x) = \sum_{n=-\infty}^{\infty} \widehat{f}_n e^{\frac{i2\pi nx}{L}}$, where $L = b - a$.

This is known as the *Fourier expansion* of f, and the $\widehat{f}_n \in \mathbb{C}$ are known as the *Fourier coefficients* of f. We may use notation $k = \frac{2\pi}{L}$ or $k_n = \frac{2\pi n}{L}$ for convenience.

An extension of this is that for a function that varies with time we may write $f(x, t) = \sum_{n=-\infty}^{\infty} \widehat{f}_n(t) e^{\frac{i2\pi nx}{L}}$ where the $\widehat{f}_n$ are functions of time only.

It is this fact that permits the power of spectral methods. Writing functions this way makes it very easy to write down their time and spatial derivatives.

Key Fact 2.2. All of these coefficients are given by the formula $\widehat{f}_n = \int_a^b f(x)e^{-iknx}\mathrm{d}x$. Note that this is the inner product of f and e^{iknx} on $L^2([a,b],\mathbb{C})$.

In other words, our Fourier expansion is given by

$$f = \sum_{n=-\infty}^{\infty} \langle f, e^{iknx}\rangle e^{iknx}$$

.

Note. The process of obtaining the Fourier coefficients of f is known as the *Fourier transform.*

Since we will be solving for these functions computationally, we could never store the values of all of these Fourier coefficients. This is where the following fact is useul.

Key Fact 2.3. If we can only use a finite number of Fourier coefficients, say, for some even N we can only use the coefficients $-N/2,\ldots,N/2$-th coefficients, then the best approximation of f in this basis is given by truncating the full Fourier series. That is, we should approximate $f = \sum_{n=-\infty}^{\infty} \widehat{f}_n e^{iknx}$ by $\tilde{f} = \sum_{n=-N/2}^{N/2} \widehat{f}_n e^{iknx}$.

Note. For real valued functions, the imaginary parts of the Fourier coefficients must cancel out. The only way that this can happen is if $\widehat{f}_n = \overline{\widehat{f}_{-n}}$ for all positive n. This fact is known as the *hermitian symmetry* and may be used by a Fourier transform library to save memory and processing time.

2.2 The discrete Fourier transform

Again, since we are working computationally, the domain of our functions is not a continuum but in fact a discrete grid. Again we will look at things in the context of a function with a 1-D spatial domain.

We will always assume a uniform grid, $(a = x_1, x_2, \ldots, x_N = b)$ with spacing Δx between gridpoints. It is worth noting that the word *grid* refers to the division of our spacial domain but does not include the time domain.

In this context, we cannot evaluate a function f on all of $[a,b]$ but instead only sample it at the gridpoints. We will write $f_i = f(x_i)$. To approximate the integral of a function f on $[a,b]$ using this grid, we use the approximation $\int_a^b f(x)\mathrm{d}x \approx \sum_{i=1}^N f_i$.

This leads us to the discrete Fourier transform, which is what Fourier transform libraries actually perform. The Fourier coefficients of a function

defined by a grid are given by

$$\widehat{f}_n = \sum_{j=1}^{N} f_i e^{-iknx_j} \Delta x$$

It is worth noting that the process of obtaining the gridpoint values in physical space is known as the *inverse (discrete) Fourier transform* and is given by

$$\widehat{f}_j = \sum_{n=-N/2}^{N/2} \widehat{f}_n e^{iknx_j} \Delta x$$

Most fast Fourier transform (FFT) libraries will store the same amount of Fourier coefficients as there are spacial gridpoints (or +1) by default. It doesn't make sense to store any more than this since we don't have enough grid data for the coefficients to have any real meaning beyond this amount.

3 Simple numerical solvers of ODEs

4 1-D problems

Now that we know enough about the fourier transform and the discrete Fourier transform, we can do our first example of using spectral methods.

4.1 The heat equation

The 1-D heat equation is given by

$$u_t = \nu u_{xx}$$

where $u : [a, b] \times [0, T] \to \mathbb{R}$ satisfies $u(a) = u(b)$ and $u(x, t = 0) = u_0(x)$.

To solve this, we use the fact that $u(x, t) = \sum_{n=-\infty}^{\infty} \widehat{u}_n(t) e^{iknx}$ to see that

$$\begin{aligned} u_t(x,t) = \frac{\partial}{\partial t}\left(\sum_{n=-\infty}^{\infty} \widehat{u}_n(t) e^{iknx}\right) &= \sum_{n=-\infty}^{\infty} \frac{\partial}{\partial t}\left(\widehat{u}_n(t) e^{iknx}\right) \\ &= \sum_{n=-\infty}^{\infty} \widehat{u}_n'(t) e^{iknx} \end{aligned}$$

where $f' = \frac{\mathrm{d}f}{\mathrm{d}t}$, and

$$\begin{aligned} u_{xx} = \frac{\partial^2}{\partial x^2}\left(\sum_{n=-\infty}^{\infty} \widehat{u}_n(t) e^{iknx}\right) &= \sum_{n=-\infty}^{\infty} \frac{\partial^2}{\partial x^2}\left(\widehat{u}_n(t) e^{iknx}\right) \\ &= \sum_{n=-\infty}^{\infty} -(kn)^2 \widehat{u}_n(t) e^{iknx} \end{aligned}$$

so our 1-D heat equation becomes

$$\sum_{n=-\infty}^{\infty} \left(\widehat{u}_n'(t) + (kn)^2 \widehat{u}_n(t)\right) e^{iknx} = 0$$

with initial conditions

$$\sum_{n=-\infty}^{\infty} \left(\widehat{u}_n(0) - \widehat{u_0}_n\right) e^{iknx} = 0$$

Observe that this means the LHS of both expressions is the fourier expansion of 0. Since the fourier expansion is unique, this must mean that each term in the fourier series in the LHS is equal to 0.

That is, for each $n \in \mathbb{Z}$ we have $\widehat{u}_n' + (kn)^2 \widehat{u}_n(t) = 0$ with initial conditions $\widehat{u}_n(0) = \widehat{u_0}_n$. We have turned our PDE into a series of ODEs. In general, ODEs are much simpler to solve than PDEs, this simplification is the essence of spectral methods.

In general, we would have to solve this ODE numerically, but in this case we can solve it analytically. Doing this we obtain

$$\widehat{u}_n(t) = \widehat{u_0}_n e^{-(kn)^2 t}$$

So we obtain the final solution

$$u(x,t) = \sum_{n=-\infty}^{\infty} \left(\widehat{u_0}_n e^{-(kn)^2 t}\right) e^{iknx}$$

4.2 General structure of a spectral solver

The simple example of the heat equation gives us a general idea of how to use spectral methods. Suppose we are solving for a function u that satisfies $u(x, t = 0) = u_0(x)$ and $u = f$ where A is some function of u and its derivatives. Typically an algorithm will follow the following steps.

1. Fill the values of u in physical space using the initial conditions.
2. Apply the Fourier transform to u to obtain its Fourier coefficients.
3. Perform some operations on the Fourier coefficients to calculate their valeues at the next timestep.
4. Apply the inverse Fourier transform to obtain the values of the function in physical space.
5. Go back to step 2 and repeat for the next timestep.

This is the most simple such algorithm. We may end up wishing to compute some values before applying the Fourier transform or even move back and forth between physical space and Fourier space multiple times.

4.3 Burgher's equation

Burgher's equation is given by

$$u_t + uu_x = \nu u_{xx}$$

This is not as simple to solve as the heat equation, since simply writing the Fourier expansion of u does not give us an easy expression for uu_x - if you don't belive me, try to find the Fourier expansion for this term.

Instead, we can use the Fourier expansion for this term more directly. The first thing to do is compute the value of u_x. Observe that each Fourier coefficient of u_x is given by $(\widehat{u_x})_n = ikn\widehat{u}_n$. This gives us an easy way to compute u_x.

Once we have this, we can use the Fourier expansion for uu_x directly, that is, instead of trying to multiply the Fourier expansions of u and u_x, just compute the Fourier expansion of uu_x.

Using this method, our PDE becomes

$$\sum_{n=-\infty}^{\infty} \left(\widehat{u}_n'(t) + (\widehat{uu_x})_n(t) + \nu(kn)^2\widehat{u}_n(t)\right) e^{iknx} = 0$$

by the same reasoning as before, this is the same as having every Fourier coefficient equal to 0, so for each $n \in \mathbb{Z}$

$$\widehat{u}_n'(t) + (\widehat{uu_x})_n(t) + \nu(kn)^2\widehat{u}_n(t)$$

We cannot solve this analytically as we could before, so we must discretify time. Lets divide our time domain into equally spaced points $0 = t_0, t_1, \ldots, t_N = T$, where the spacing is Δt. We will write u^j for our approximation of $u(t_j)$. Now our ODE becomes

$$\frac{\widehat{u}_n^{j+1} - \widehat{u}_n^j}{\Delta t} + (uu_x)_n^j + \nu(kn)^2\widehat{u}_n^{j+1} = 0$$

Rearranging for $\widehat{u}_n^{j+1}$ we get our time evolution scheme for each Fourier coefficient. Note that we use the uu_x term explicitely but the u_{xx} term implicitely.

4.4 A brief note on aliasing and de-aliasing

It is now an appropriate time to bring up the fact that the numerical solution to Burgher's equation you just did is actually wrong! Probbaly not by much but defninitely a little. This is due to a mathematical phenomenon know as *aliasing*, which arises when we don't have enough gridpoints to represent the functions we would like to.

5 2-D problems

The important thing to note about moving to 2-D is that a function of 2 spacial variables $u = u(x, y)$ may be written as

$$u(x,y) = \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} \widehat{u}_{mn} e^{ik_x mx + ik_y ny}$$

where $k_x = 2\pi/L_x$ and $k_y = 2\pi/L_y$.

For notational convenience, we will often omit the mn subscript and write $\widehat{u}$ for $\widehat{u}_{mn}$, along with omitting the x and y subscripts for k. We infer which k is being referred to by whether it is next to m or n. In many cases these will be the same anyway.

I will add in the 2-D Burgher's equation soon, but it is a simple enough expansion onto what we already know that it can be treated as an exercise for the reader.

6 Solving Navier-Stokes

Now that we have solved the 2-D Burgher equation, Navier-Stokes is not far away. Notice that momentum balance of Navier-Stokes, given by

$$u_t + uu_x + vu_y = -p_x + g_x + \nu\left(u_{xx} + u_{yy}\right)$$
$$v_t + uv_x + vv_y = -p_y + g_y + \nu\left(v_{xx} + v_{yy}\right)$$

is very similar to the 2-D Burgher equation. Only with the added pressure gradient and force terms. Timestepping forward with these equations is very similar to what we have already done.

If we only cared about momentum balance and had no pressure gradient, our time-stepping scheme would be

$$\frac{\widehat{u}^{j+1} - \widehat{u}^j}{\Delta t} + \widehat{uu_x}^j + \widehat{vu_y}^j = ik_m \widehat{g}^j - \nu(k_m^2 + k_n^2)\widehat{u}^{j+1}$$
$$\frac{\widehat{v}^{j+1} - \widehat{v}^j}{\Delta t} + \widehat{uv_x}^j + \widehat{vv_y}^j = ik_n g^j - \nu(k_m^2 + k_n^2)\widehat{v}^{j+1}$$

However, we cannot guarantee using only this that incompressibility is satified. This is where the role of the pressure comes in.

6.1 The Projection Method

Instead of the timestepping scheme above, lets do the same update method, but instead instead of u^{j+1} and v^{j+1}, we have the variables $\bar{u}$ and $\bar{v}$. The method we use to ensure that the resulting field is incompressible is known as the *projection method*. The idea is to project $\bar{u}$ onto a divergence free field. The method relies on the following result

Theorem 6.1. *(Helmholtz Decomposition) Let* $\bar{\mathbf{u}} = (\bar{u}, \bar{v})$ *(or* $(\bar{u}, \bar{v}, \bar{w})$*) be a field defined on a simply connected domain. Then* $\mathbf{u}$ *may be uniquely decomposed into a divergence-free part and a vorticity-free part. In other words*

$$\bar{\mathbf{u}} = \mathbf{u} + \nabla\phi$$

where $\nabla \cdot \mathbf{u} = 0$ *and* ϕ *is a scalar field.*

This fact allows us to enforce incompressibility in a simple manner. We can find the field ϕ in the following way.

Enforcing incompressibility on our new field

$$\begin{aligned} 0 = \nabla \cdot \mathbf{u} &= \nabla \cdot (\bar{\mathbf{u}} - \nabla\phi) \\ &= \nabla \cdot \bar{\mathbf{u}} - \nabla^2\phi \\ &= \sum_{n=-\infty}^{\infty} \left(ik_n \widehat{\bar{u}}_{nm} + ik_m \widehat{\bar{v}}_{nm} + \left(k_n^2 + k_m^2\right) \phi_{nm} \right) e^{i(k_n x + k_m y)} \end{aligned}$$

so each Fourier coefficient of ϕ satisfies

$$ik_n\bar{u} + ik_m\bar{v} + \left(k_n^2 + k_m^2\right)\phi = 0$$

Once we have found ϕ, we may simply take its gradient away from $\bar{\mathbf{u}}$ to obtain our divergence free field $\mathbf{u}$.

7 Details on implementation

This guide is intended to be as general as possible, so most language specific implementation details will be neglected. It is recommended that the reader uses an external FFT (Fast Fourier Transform) library rather than attempting to make their own. Examples of these include FFTW if C or C++ is being used, or the FFT functionality of Numpy if Python is the reader's language of choice. I haven't used the FFT in Numpy, so I can only give information about FFTW in C.

7.1 Notes on FFTW

The documentation of FFTW is reasonably good, but there are a couple of things worth noting. Firstly, it stores the Fourier coefficients in a slightly weird way. Suppose we are doing a 1-D Fourier transform using c2c.

If we have N+1 Fourier coefficients for some even N, we would expect for them to be stored in this way:

$$[u_{-N/2}, u_{-N/2+1}, \ldots, u_{N/2}]$$

But instead, FFTW stores them like this:

$$[u_0, u_1, \ldots, u_{N/2}, u_{-N/2}, \ldots, u_{-1}]$$

This isnt a problem if you use r2c in 1-D (which you should be for everything in this resource), since it makes use of Hermitian symmetry of the coefficients, so it stores 1-D coefficients in the following way

$$[u_0, u_1, \ldots, u_{N/2}]$$

But using r2c in 2-D does lead to these same complications. In general, if your code doesn't work and you don't know why, check that you are accessing the correct Fourier coefficients.

8 Further Reading