

Lecture 7

Spatial discretisation : Spectral and finite element techniques

D. Dickinson
d.dickinson@york.ac.uk

York Plasma Institute, School of PET, University of York

Semester 1

Overview of course

This course provides a brief overview of concepts relating to numerical methods for solving differential equations.

Topics to be covered include:

- Integrating ODEs
 - Explicit techniques
 - Implicit techniques
 - Using Scipy to integrate ODEs
- Spatial discretisation
 - Finite differencing
 - Spectral methods
 - Finite elements
- Particle In Cell (PIC) approaches
- Continuum techniques

Notes available on the VLE.

Overview this lecture

This lecture will look at

- Spectral techniques
- Fourier Transforms
- Finite elements

Spectral techniques

We've just seen how we can solve two point boundary problems using finite differences and sparse matrices. These are relatively generic techniques that can be widely applied.

In this lecture we'll look at a different approach known as **spectral**. With this approach we express our solution in terms of known basis functions

$$f(x) = \sum_i \hat{f}_i \alpha_i(x)$$

where α_i are the basis functions

which allows us to write

$$\frac{df}{dx} = \sum_i \hat{f}_i \frac{d\alpha_i}{dx}$$

If you make good guess for functions, you only need to use very few of them.

Here the basis functions $\alpha_i(x)$ cover the entire domain which means the derivative calculation can involve the entire domain. This tends to lead to dense matrices, unlike the sparse ones arising in finite differences. As such spectral methods can be much more computationally demanding, however they can offer **extremely fast convergence**.

Periodic domains

A common situation that arises in plasma physics is the study of waves in an **infinite system**, i.e. where we **have no boundaries**. To represent this in our numerical simulation we make our domain **periodic**, such that anything leaving one side of the box re-enters at the other side. This allows waves/particles to carry on moving without encountering a boundary.



Figure: Figure showing periodic grid of length L and spacing δ_x .

There is one main difference between a truly infinite domain and a periodic one; in an infinite domain we can capture waves with any wavelength whilst in a periodic domain we can only capture waves which will fit in one period. In other words, if the periodic box has length L we can capture waves with wavenumber¹ $k = 2\pi n/L$ with n an integer. **integer numbers of waves**

¹Or with wavelength $\lambda = L/n$.

Periodic domains : Fourier transforms

In a periodic domain a good choice of basis functions are the sine and cosine waves, which we'll write as

$$\alpha_j(x) = \exp\left(\frac{i2\pi jx}{L}\right) = \exp(ik_j x)$$

Our spectral representation is then

(spectral decomposition) $f(x) = \sum_j \hat{f}_j \exp(ik_j x)$

which you may recognise as a [discrete Fourier transform²](#) (DFT). We'll now have a look at how we can use this Fourier representation to solve a spatial differential equation.

²Ignoring any factors of 2π .

Periodic domains : Fourier transform

Let's consider Poisson's equation in 1D

$$\frac{d\epsilon}{dx} - \frac{d^2\phi}{dx^2} = -\frac{\rho(x)}{\epsilon_0}$$

First let's look at the Fourier transform of ϕ

$$\phi(x) = \sum_j \hat{\phi}_j \exp(ik_j x)$$

If we try to differentiate this wrt x we get

$$\frac{d\phi}{dx} = \sum_j \hat{\phi}_j \frac{d \exp(ik_j x)}{dx} = \sum_j ik_j \hat{\phi}_j \exp(ik_j x)$$

and similarly the second derivative gives us

$$\frac{d^2\phi}{dx^2} = \sum_j -k_j^2 \hat{\phi}_j \exp(ik_j x)$$

need to know weightings, phi_j hat
- orthonormal

Periodic domains : Fourier transform

Substituting this into Poisson's equation we get

$$\epsilon_0 \sum_j k_j^2 \hat{\phi}_j \exp(ik_j x) = \sum_j \hat{\rho}_j \exp(ik_j x)$$

want some justification for why we can ignore the sums and equate the terms inside

Now we can take advantage of the fact that each $\exp(ik_j x)$ is orthogonal to others, i.e.

$$\int \exp(ik_j x) \exp(ik_l x) dx = \delta_{k_j, k_l}$$

to equate each individual term in the sum

because they must be ($j=l$) or it would be zero

$$\epsilon_0 k_j^2 \hat{\phi}_j = \hat{\rho}_j, \Rightarrow \hat{\phi}_j = \frac{\hat{\rho}_j}{k_j^2 \epsilon_0} = -E$$

put into spectral representation to sum and find phi

Now we know each $\hat{\phi}_j$ in terms of $\hat{\rho}_j$ we can inverse Fourier transform to find $\phi(x)$.

inverse fourier transform
just means evaluating
spectral representation

fourier method (code) can tell us how much of each basis vector to put in to get a good approximation of our overall function phi - fits the rho_j_hats

Periodic domains : Fourier transform

We've seen how calculating derivatives of (or integrating) functions can be much simpler using Fourier transforms, becoming simple multiplication in Fourier space. This simplicity comes at the cost of having to perform a Fourier transform and its inverse on the data. Thankfully fast algorithms have been developed for this, known as [FFTs](#), which operate in $\mathcal{O}(N \log N)$ time, where N is the number of grid points. Unfortunately FFTs don't tend to work that well in parallel as they require information from the whole domain. [Not good for high-performance computing then](#)

A final consideration to make when using Fourier techniques is something known as [aliasing](#). Essentially any high frequency components not properly resolved will appear as low frequency components.

SciPy provides a FFT submodule, and there's an example on the VLE of using this to integrate a simple function.

[But uses dense matrix,](#)

Nonperiodic domains

Changing basis functions

Whilst sine and cosine waves were a good choice for the periodic domain we were just considering, if the problem has boundaries then there are different choices of basis function which may be better. These include a range of different orthogonal polynomials including Legendre, Laguerre and Chebyshev.

When using different basis functions the Fourier transform is replaced by what can be crudely thought of as a fitting procedure³. When fitting data with enforced boundary conditions, uniform grids are not usually the best choice and can lead to large oscillations near the edge, something known as [Runge's phenomena](#). Instead grids with points clustered near the boundaries often lead to much better performance. One way to achieve this is using [Chebyshev points](#).

For an illustration of this check the notes and the `cheby.py` example on the VLE.

Go from spectral space to real space

If we can't take advantage of Fourier transforms

³In fact we project our function onto the basis functions, $\hat{\rho}_j = \int \rho(x) F_j(x) dx$ where $F_j(x)$ is the j^{th} basis function.

Finite elements

Chebyshev works well with only few grid points

One of the main downsides of the spectral techniques we've looked at so far is that they couple the entire domain **making associated matrices dense** and posing problems for parallelisation.

Finite element methods take a similar approach to the spectral techniques we've seen so far except the basis functions only span a small part of the domain. This allows much better parallelisation opportunities and results in sparse matrices whilst retaining benefits of a spectral approach.

^ Can split domain into regions to enable parallel processing (no need for communication)

These finite element methods are very popular in a large range of applications and a number of libraries exist to assist in implementing these.