Numerics background for Pseudo-Spectral Methods

Marek Stastna

Introduction

- IN the previous slide deck we saw the important physical concept of dispersive waves.

- This set off slides is NOT meant to replace a proper course in numerical PDEs or spectral methods.

- It is meant to provide an intuitive background letting you to either work with a spectral model like SPINS with some confidence or to allow you to explore pseudospectral methods using existing code in Matlab or Python.

- Pseudospectral methods also provide a nice intuitive point of view on numerics in contrast to many introductory courses that muddy the waters with a host of (in practice) unimportant issues.

- Let’s start by reviewing the Fourier transform.

- The Fourier transform is defined as: $\mathcal{F}(u) = \bar{u}(k) = \int_{-\infty}^{\infty} u(x)\exp(-ikx)dx$

- This is a linear operation so that the transform of a sum is the sum of transforms.

- The most useful property of Fourier transforms is how the transform of a derivative works out:

$$\mathcal{F}(u'(x)) = \int_{-\infty}^{\infty} u'(x)\exp(-ikx)dx = u(x)\exp(-ikx)\,|_{-\infty}^{\infty} + ik\int_{-\infty}^{\infty} u(x)\exp(-ikx)dx$$

- For mild technical requirements on u(x) the first term vanishes and we get the important property: $\mathcal{F}(u'(x)) = ik\mathcal{F}(u(x))$.

- There is a better way to prove this fact but one needs to build up considerably more properties (I can provide a reference).

- The Fourier transform: $\mathscr{F}(u) = \bar{u}(k) = \int_{-\infty}^{\infty} u(x)\exp(-ikx)dx$ allows for a continuous dependence on the wavenumber k.

- It can be thought of as the limit of a Fourier series as we allow for a large and larger domain.

- This is a very difficult pure mathematical limit, however, and we do not attempt to prove it.

- But an engineer might argue that they can synthesize any signal too acceptable accurate out of a relatively small number of constituent sines and cosines.

- So a practical point of view might consider the discretized Fourier transform, or DFT and this is precisely what I want to use in what follows.

Numerics: Philosophy

- The basic idea of the school of numerics I subscribe to is that what is a partial derivative in the text book is thought of as the action of a function on the computer.
- The easiest (but not most memory efficient) way to think of that is as a matrix acting on vectors.
- The easiest example of a PDE would be Poisson's equation: $\nabla^2 \phi = f$
- We want to solve for ϕ given f. If we build a matrix that approximates the Laplacian we would want to solve the matrix problem: $\mathscr{L} \vec{\phi} = \vec{f}$
- The vector here means we've evaluated f on a grid, and want to find the values of phi on the same grid.
- You can imagine that the properties of the matrix will influence how easy/hard this is.

- Poisson's equation is an example of an elliptic problem. These have no time dependence so are ideal play grounds for those looking to delve into the numerical linear algebra behind a method.

- They can also show the power of the Fourier technique (next slide).

- Several famous numerical linear algebra methods have their roots in solving elliptic problems (tridiagonal solvers, SOR, multi-grid, etc).

- Matlab is basically a giant linear algebra package (originally called LAPACK) and you can explore a lot of linear algebra using Matlab and our tutorial codes:

https://wiki.math.uwaterloo.ca/sheets/matlab/html/intro_linearsystems.html

Fourier Method for Poisson's Equation

- Consider Poisson's equation with a change of name for the known RHS: $\nabla^2\phi = g$
- If we consider a doubly periodic domain with a doubly periodic g(x,y) the double Fourier transform gives: $-(k^2 + l^2)\bar{\phi} = \bar{g}(k,l)$ where the overbar denotes a doubly Fourier transformed variable and (k,l) is the wave number.
- Transforming has turned the PDE into an algebraic equation and solving we get $\bar{\phi} = -\dfrac{\bar{g}(k,l)}{k^2 + l^2}$. Note something special has to be done for $(k,l) = (0,0)$.
- In analytical work we would be limited by the fact that the RHS rarely has an analytical inverse Fourier transform but numerically we can simply write: $\phi(x,y) = -\mathcal{F}_x^{-1}\mathcal{F}_y^{-1}\left(\dfrac{\bar{g}(k,l)}{k^2 + l^2}\right)$
- In practice, the inverse Fourier transforms would be carried out with the FFT algorithm.

Notice the k=l=0 part of the solution is not determined, but this is also clear from the PDE because k=l=0 is a constant.

Time Dependent Problems

- There are two very different “basic” time dependent problems.
- They differ in the basic physics they are meant to model, the manner in which this is reflected in their mathematics and the numerical methods used to solve them.
- The first is the advection equation: $A_t = -u_0 A_x$
- The advection equation is basically a conveyor belt running left to right with a speed u_0
- The second is the diffusion equation: $C_t = \kappa C_{xx}$
- The diffusion equation smears information, or sharp interfaces.
- For both problems it is typically true that the time step Δt is much smaller than the grid spacing Δx
- This might not be true in a field like nanoscience, but it is true in environmental problems of all sorts.

- To solve the advection equation: $A_t = -u_0 A_x$ we want to preserve as much as possible its “left to right nature”, recall D’Alembert’s solution: $A(x,t) = F(x - u_0 t)$

- This means the space derivative is usually evaluated at the “now” time step.

- Low order methods, like finite difference and finite volume also want to account for the “left to right nature of the problem” by using what is called upwinding.

- Higher order methods don’t bother since they are often based on Fourier methods which aim to represent the dispersion relations exactly.

Here is the dispersive wave concept in a different concept!

- Using the plane wave solution we have $A = \exp(i(kx - \sigma t)) \implies -i\sigma = -iu_0 k \implies \sigma = u0k$

- And a Fourier spectral method would choose a discrete number of k, say k_n and evolve these and then rebuild the solution out of them.

- If you think of the numerical Fourier transform as $\mathscr{F}$, in practice the FFT, then the PDE becomes an ODE if we Fourier transform: $\bar{A}_t = -u_0 ik\bar{A} \implies \bar{A}(k,t) = \bar{A}(k,0)\exp(-iu_0 kt)$ where the argument in the complex exponential is just like the sigma in the dispersion relation!

- The only problem with Fourier methods is that they assume periodicity, but for wave problems this is often OK since you want to know what the waves (as opposed to the boundary) are doing.

- In fact Fourier problems can be used for the diffusion equation, too.

- The key here is to note that one of the mathematical properties of diffusion is that it spreads information “infinitely fast” (yes it does in fact violate relativity).

- Another way to put this is that even either a step initial, or even a delta initial condition, both become infinitely differentiable at any time, t>0.

- This means in practice we want to evaluate the spatial derivative at the “future” time step or implicitly.

- If we use superscripts for time steps: $(C^{(n+1)} - C^{(n)})/\Delta t = \kappa C_{xx}^{(n+1)}$

- In a periodic domain we can again use the Fourier transform to get:
  $(\bar{C}^{(n+1)} - \bar{C}^{(n)})/\Delta t = -\kappa k^2 \bar{C}^{(n+1)}$

- Rearrange to get an algebraic equation: $(1 + k^2\kappa\Delta t)\bar{C}^{(n+1)} = \bar{C}^{(n)}$ which must be solved for all the discrete values of k (super easy in Matlab). Look back to the Poisson problem and note the similarity.

- Fourier methods for wave equations are a great way to test new code environments.
- This is because wave problems typically require explicit methods, which you can think of as a recipe:
- You have data at time step n, and a way (may be a complicated way) to update this data to get the function(s) you want at time step (n+1).
- In contrast diffusion problems may require a bit more work (especially when problems are coupled) since you want to evaluate the diffusion at the “new” time step (or implicitly).
- Let’s see an example of that, the advection diffusion equation: $C_t = -u_0 C_x + \kappa C_{xx}$
- We use the simplest Euler discretization on the LHS and the Fourier transform: $(\bar{C}^{(n+1)} - \bar{C}^{(n)})/\Delta t = -u_0 ik \bar{C}^{(n)} - \kappa k^2 \bar{C}^{(n+1)}$
- Notice how the advection part is evaluated “now” (explicitly) and the diffusion is evaluated in the “future” (implicitly).

- We use the simplest Euler discretization on the LHS and the Fourier transform: $(\bar{C}^{(n+1)} - \bar{C}^{(n)})/\Delta t = -u_0 ik\bar{C}^{(n)} - \kappa k^2 \bar{C}^{(n+1)}$
- Rearranging and factoring $(1 + \kappa k^2 \Delta t)\bar{C}^{(n+1)} = (1 - \Delta t u_0 ik)\bar{C}^{(n)}$
- We can now get a recipe in Fourier space: $\bar{C}^{(n+1)} = \dfrac{(1 - \Delta t u_0 ik)\bar{C}^{(n)}}{1 + \kappa k^2 \Delta t}$
- And the solution in physical space can be found by using the FFT to inverse Fourier transform numerically.

Notice for linear problems we could just solve in Fourier space and only transform back to physical space when plotting.

Stability 1

- The time stepping formula in Fourier space naturally introduces the notion of stability $\bar{C}^{(n+1)} = \frac{(1 - \Delta t u_0 ik)\bar{C}^{(n)}}{1 + \kappa k^2 \Delta t}$

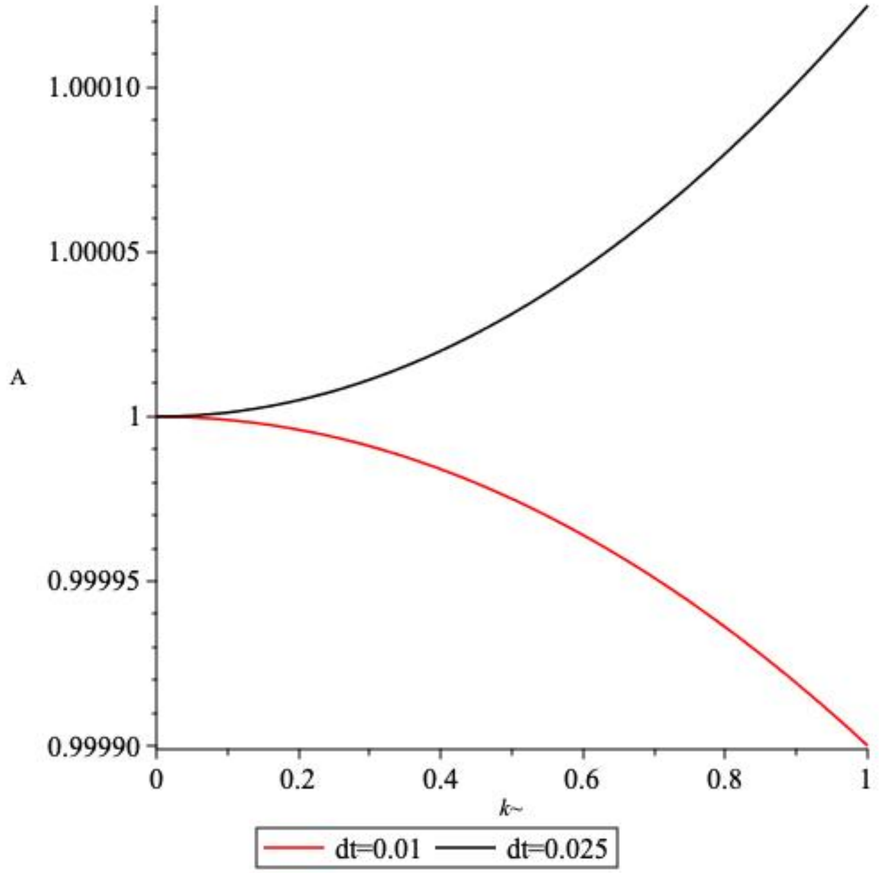
- Since $\bar{C}$ is a complex number we can take the complex modulus of both sides and find $|\bar{C}^{(n+1)}| = |\frac{(1 - \Delta t u_0 ik)}{1 + \kappa k^2 \Delta t}||\bar{C}^{(n)}| = A|\bar{C}^{(n)}|$ and stability comes down to whether A is larger or smaller than 1.

- $|\frac{(1 - \Delta t u_0 ik)}{1 + \kappa k^2 \Delta t}| = \frac{\sqrt{1 + \Delta t^2 u_0^2 k^2}}{1 + \kappa k^2 \Delta t}$ a couple of sample curves are plotted on the next slide

Stability 2

- $|\frac{(1-\Delta t u_0 ik)}{1+\kappa k^2 \Delta t}| = \frac{\sqrt{1+\Delta t^2 u_0^2 k^2}}{1+\kappa k^2 \Delta t}$ Choose $u_0 = 1$, $\kappa = 0.01$.

- The curves on the right show that there is a stability bound, but in practice we can just decrease Δt to get stability.



Nonlinearity

- OK, so far so good. But all the problems we have dealt with have been linear. So now let's consider the so called viscous Burgers equation.

- Our first nonlinear problem: $B_t + BB_x = \nu B_{xx}$

- We move the nonlinear term to the RHS,use the simplest Euler discretization on the LHS to discretize: $(B^{(n+1)} - B^{(n)})/\Delta t = -(BB_x)^{(n)} + (\nu B_{xx})^{(n+1)} = -\frac{1}{2}(B^2)_x^{(n)} + (\nu B_{xx})^{(n+1)}$

- Again we decided to treat the nonlinear term in an explicit manner (they are like advection but slightly more complicated) and the diffusion in an implicit manner. We also used a trick with the chain rule to rewrite the nonlinear term

- Next take the FFT and rearrange: $(1 + \nu k^2 \Delta t)\bar{B}^{(n+1)} = \bar{B}^{(n)} - ik\Delta t(\bar{B^2})^{(n)}$

- And again we can get a closed form expression for $\bar{B}^{(n+1)}$ which we can use the FFT to bring to physical space.

Notice that $\bar{B}^2 \neq \bar{B^2}$ so that here we have to take the known function $B^{(n)}$ square it, and then take its FFT for the 2nd piece of the RHS

Sample Code

```
impop=1./(1+dt*nu*ks2);

for ii=1:numouts
    for jj=1:numstps
     t=t+dt;
% implicit time stepping for diffusion
     u1 = real(ifft(impop.*(fft(u1) -
dt*i*0.5*ks.*fft(u1.*u1))));
% explicit time stepping for diffusion (for comparison only)
     u2 = real(ifft((fft(u2) - dt*i*0.5*ks.*fft(u2.*u2))
+expop.*fft(u2)));
    end
end
```

Because the above transforms back to physical space at each time step it is called a **pseudospectral method**.
A pure **spectral method** would do all computations in Fourier space.

Comparison to finite differences 1

- Spectral methods are tidy, and easy to implement. This would be a good argument for their adoption, but they are also much more accurate than low order methods.

- The classical comparison is for the advection equation: $A_t = -u_0 A_x$

- Using 2nd order, centred differencing in space, $A_x \approx \dfrac{A_{i+1} - A_{i-1}}{2\Delta x}$, leads to an algorithm that is unstable. Finite differences thus try to mimic the exact solution, $A(x,t) = F(x - u_0 t)$ for which information propagates from left to right.

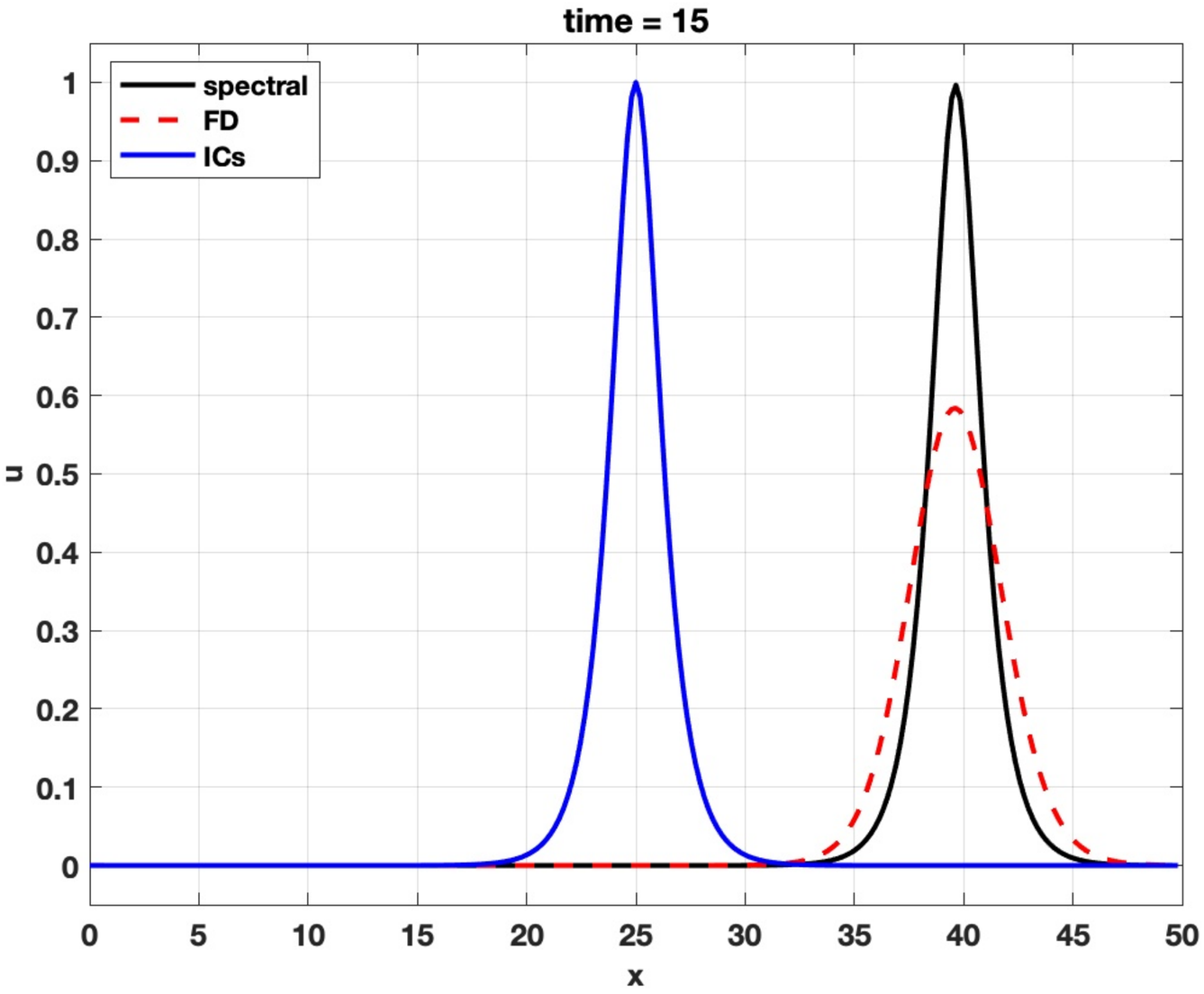
- The so-called upwind derivative is $A_x \approx \dfrac{A_i - A_{i-1}}{\Delta x}$ and is only first order.

Comparison to finite differences 2

- We implement the spectral and finite difference codes and show the results with 256 points.

- You can see that both algorithms capture the propagation speed of the disturbance, but the FD scheme damps and broadens the signal by a rather large amount.

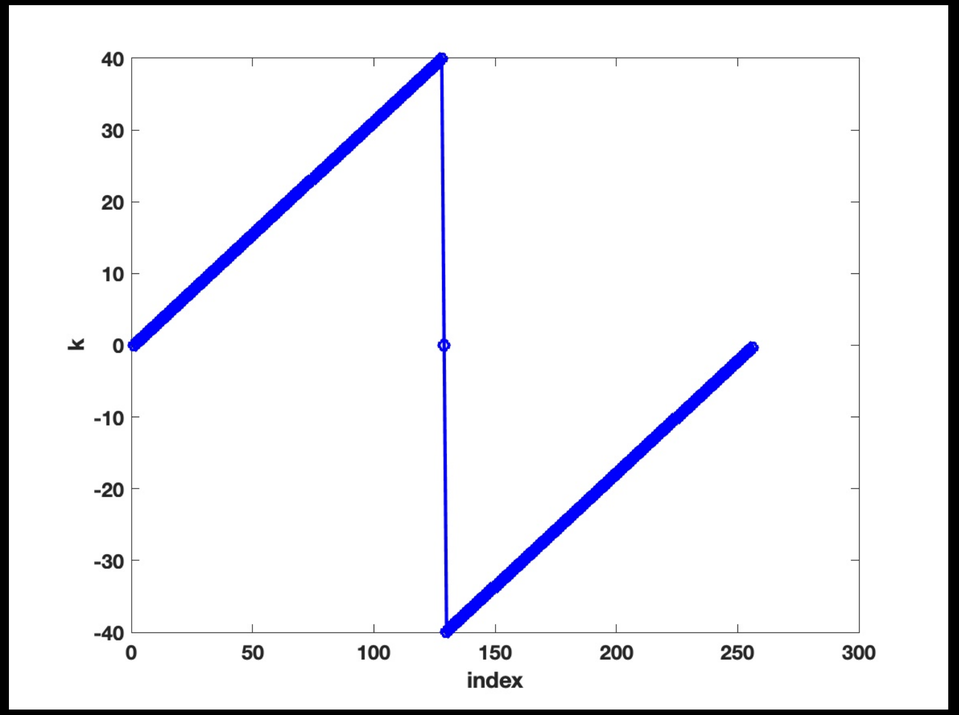
- There are FD schemes that can do better but in practice the case for spectral methods is quite clear.



Wave numbers and their ordering

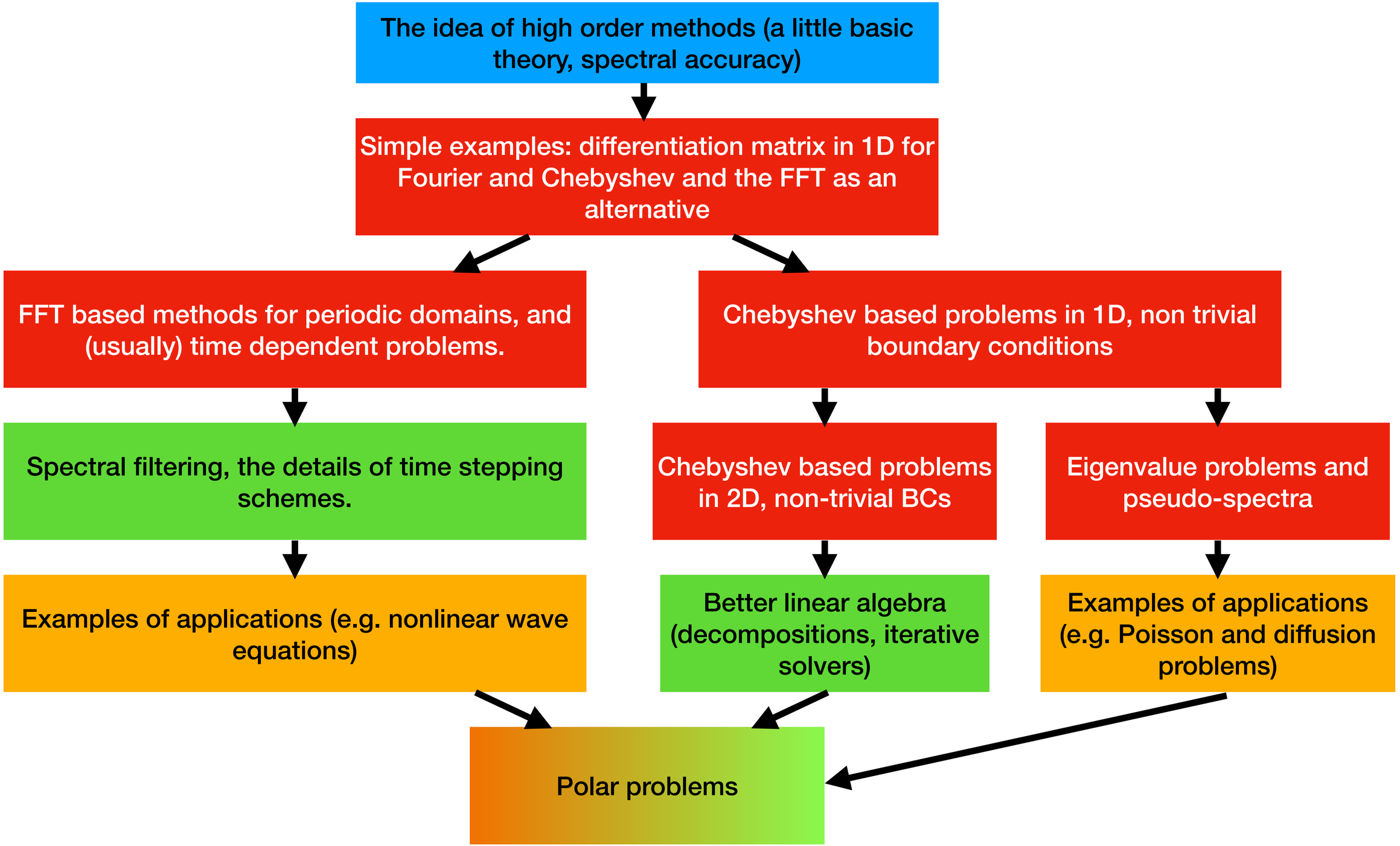
Wavenumber ordering

- There is a common way to store wavenumbers that FFT implementations often use.
- The idea is to store $k = 0$, then the positive wavenumbers, then the negative wavenumbers.
- If you only want to look at the first bit of the spectrum you do not need to build all of the k vector. You could just write myks=(0:numks)*kmin to get the average and the first numbs k values.




```
%make wave numbers
nyquist_freq = 2*pi/(xmax-xmin);
ks=[0:N/2-1 0 -N/2+1:-1]*nyquist_freq;
```

Flow chart for a more complete view of pseudospectral methods


The idea of high order methods (a little basic theory, spectral accuracy)
Simple examples: differentiation matrix in 1D for Fourier and Chebyshev and the FFT as an alternative
FFT based methods for periodic domains, and (usually) time dependent problems.
Chebyshev based problems in 1D, non trivial boundary conditions
Spectral filtering, the details of time stepping schemes.
Chebyshev based problems in 2D, non-trivial BCs
Eigenvalue problems and pseudo-spectra
Examples of applications (e.g. nonlinear wave equations)
Better linear algebra (decompositions, iterative solvers)
Examples of applications (e.g. Poisson and diffusion problems)
Polar problems

Appendix Slides for those reading Trefethen or wanting more information about matrix solves during time stepping

- Chapter 2 of Trefethen discusses the background of Fourier based spectral methods in a fair bit of detail.

- Trefethen not only shows how discretizing “x” affects k, he shows a form of the uncertainty principle, more commonly known as aliasing (his figure 2.1)

- Finally he shows how one can construct a “basis” for an interplant for periodic function on a periodic grid.

- The theory is very, very pretty, with deep links with complex analysis.

- In practice however, one does not need to form a matrix, as Trefethen does.

- One can instead use the FFT algorithm to compute the DFT.

- This is the subject of his Chapter 3, with the practical result being his program 6.

- I have taken this code, and added comments to form the basis of all my wave codes.

- Fourier methods are definitely not the first things taught in numerics class (they probably should be…)

- If we think of a general situation we just want to replace the derivative by a matrix: $\partial_x \to \mathbf{D}$

- If we use superscripts for time steps: $(C^{(n+1)} - C^{(n)})/\Delta t = \kappa \mathbf{D^2} C^{(n+1)}$

- Rearrange to get an algebraic equation: $(\mathbf{I} - \kappa \Delta t \mathbf{D^2}) C^{(n+1)} = C^{(n)}$ which must be solved for all the discrete values of k (super easy in Matlab).

- Because the diffusion coefficient and time step are both typically fairly small the left hand side is a “small perturbation to the identity” so in practice the matrix problem is easy to solve.

- It takes a bit of work to take care of boundary conditions (this is why many models avoid “exact” treatments of boundary conditions).

- Since we have effectively replaced the derivative by a matrix $\partial_x \rightarrow \mathbf{D}$, we can impose both Dirichlet and Neumann conditions by modifying our operator appropriately

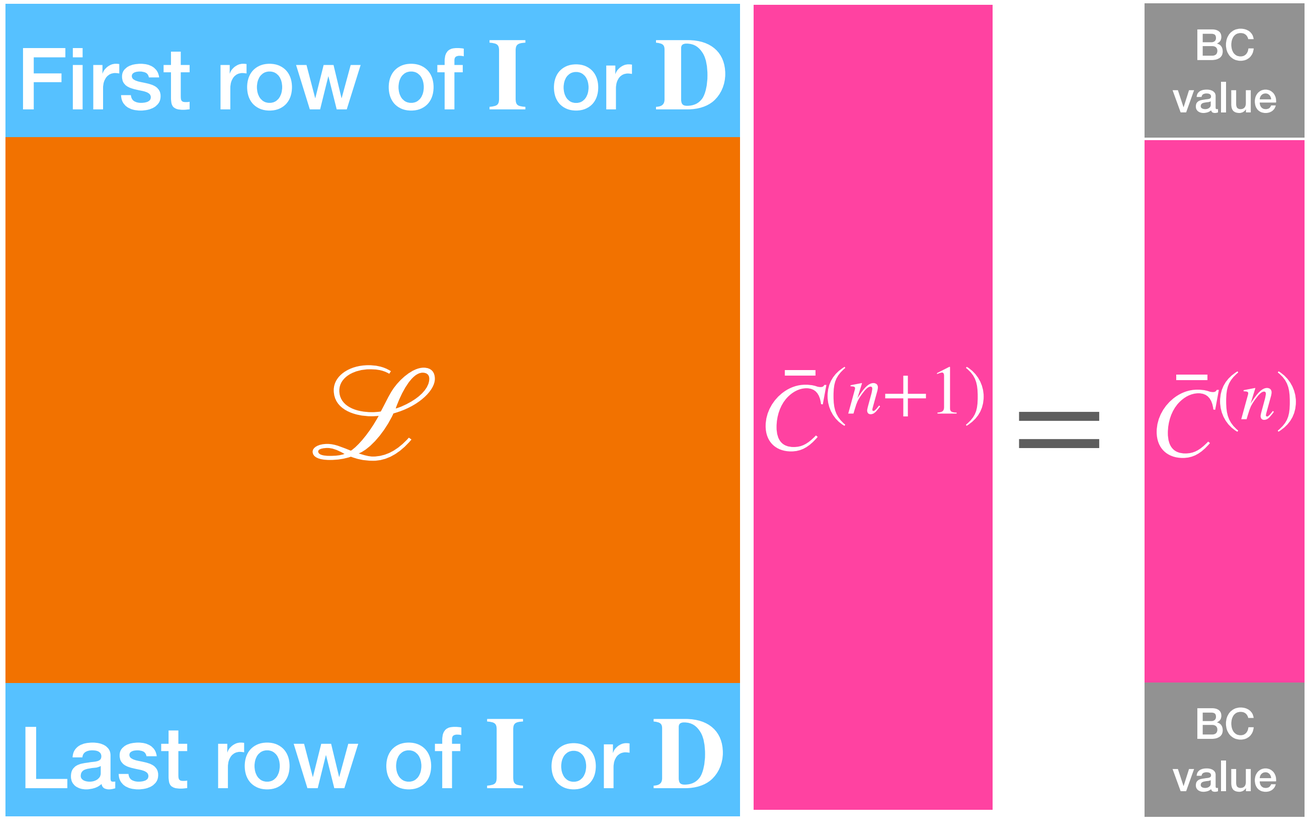
- The operator $\mathscr{L} = (\mathbf{I} - \kappa \Delta t \mathbf{D^2})$ is an NxN matrix. The first (last) row acts on the concentration vector at the new time to give the value at the left (right) boundary.

- For Dirichlet boundary conditions, which specific concentration value, we must replace the row of $\mathscr{L}$ by the appropriate row of the identity matrix, and the right hand side by the value we want the concentration to take at the boundary.

- For Neumann boundary conditions, which specify flux, we replace the row of $\mathscr{L}$ by the appropriate row of $\mathbf{D}$, and the right hand side by the value we want the flux to take at the boundary.

- In summary if the pure discretization reads $\mathscr{L}C^{(n+1)} = C^{(n)}$ then the discretization with BCs looks like:

Notice that the structure of the problem is the same regardless of what we use for a spatial discretization

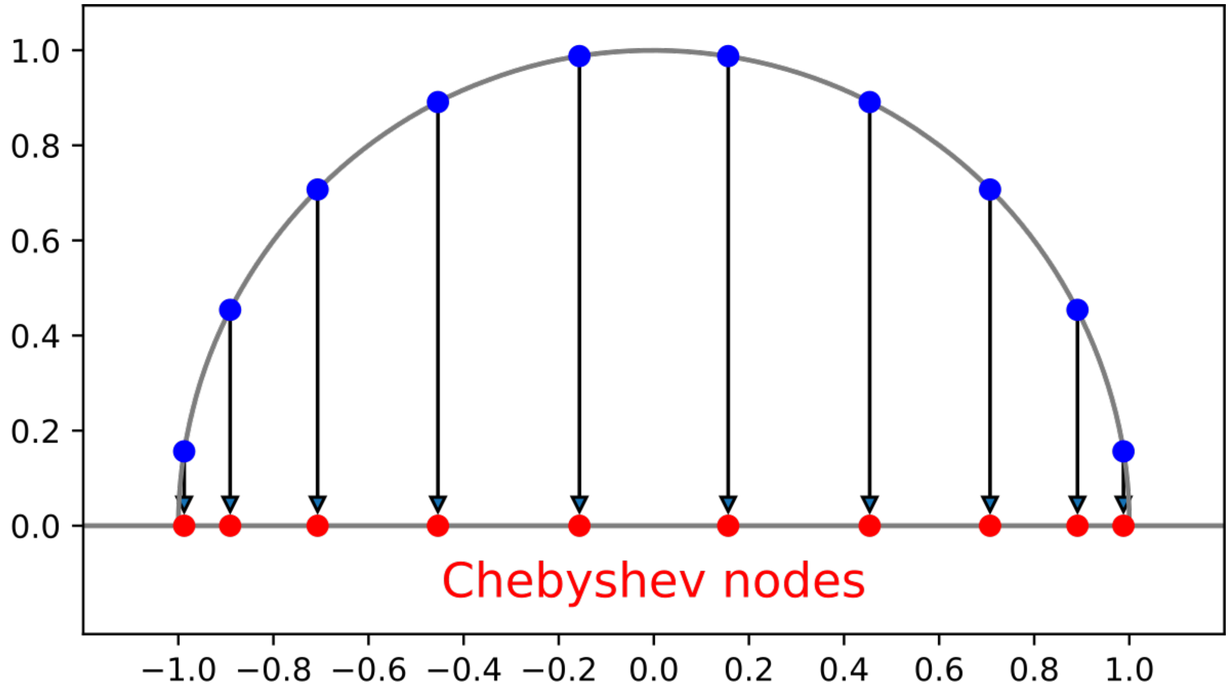


- In practice this is often OK. However a worry may be that the drastic modification of the metric on the LHS may make it hard to solve (i.e. large condition number).

- Chebyshev differentiation starts with a pretty simple question: “Given N discrete points and the values of a function on them, can you give me the best possible approximation of the function’s derivative?”

- It turns out that if the points are regularly spaced on the real line, the problem is impossible (e.g. the Runge phenomenon).

- But if they are regularly spaced on a circle this gives the Chebyshev points and the problem leads to the Chebyshev differentiation matrix (Trefethen’s Spectral Methods in Matlab book has details and we can pick up the story there).



- Once we have the grid, the idea is to build the derivative matrix $\mathbf{D}$.
- From then on the rest follows the conceptual outline of the previous few slides.
- As the number of points increases the grid does get rather fine near the end points.
- This could be useful (for boundary layer problems, for example) but will definitely restrict the time step.