Lecture 11, Sci. Comp. for DPhil Students II

Nick Trefethen, Thursday 27.02.20

Last lecture

- V.11 Fourier, Laurent, and Chebyshev
- V.12 Chebyshev series and interpolants

Today

- V.13 Chebyshev spectral discretization

Handouts

- Assignment 4
- `m55_grayscott.m` - Gray-Scott movie
- `cheb.m`, from *Spectral Methods in MATLAB*
- `m56_waveeqcheb.m` - wave equation in 1D, Chebyshev spectral
- `m57_leapfrog2Dcheb.m` - 2D wave eq. via Chebyshev spectral method
- `m58_allencahn.m` - Allen-Cahn equation and metastability
- Allen-Cahn equation page from *PDE Coffee Table Book*
- "Ten-digit algorithms", Trefethen 2005

Katherine Johnson, 1918-2020

Next Tuesday's lecture, open to all: a talk on "Who invented the great algorithms?"

First, "Ten digit algorithms". This may look casual but it's very serious and it applies to YOU. Please read it once, then read it again. In particular, the "five seconds" rule is not just for other people less serious than yourself; it is for *you*.

The 58 codes of this course, while not all exactly TDAs, illustrate this spirit: if you work with codes that do a lot quickly and compactly, this is the best way to encourage exploration that helps ensure results are correct.

More fun with the Gray-Scott equations from Assignment 3. Here's a MATLAB movie version (Just Euler time-stepping with finite differences in space, so not quantitatively correct).

[`m55_grayscott.m` - Gray-Scott equations movie]

V.13 Chebyshev spectral discretization

Chebyshev spectral differentiation:

Given: data $v_0, \dots, v_N$ at Chebyshev points

Want: "derivative" $(w_0, \dots, w_N)$ at these points

Method:

(1) p = unique polynomial of degree $\le N$ with $p(x_j) = v_j$ for all j

(2) $w_j = p'(x_j)$ [and similarly for higher derivatives]

The process is linear, and we can write it in matrix form:

$$\begin{pmatrix} w_0 \\ \vdots \\ w_N \end{pmatrix} = D \begin{pmatrix} v_0 \\ \vdots \\ v_N \end{pmatrix}.$$

Example: $N = 2$

$$x_0 = 1, \quad x_1 = 0, \quad x_2 = -1.$$

$$p(x) = .5x(1+x)v_0 + (1+x)(1-x)v_1 + .5x(x-1)v_2$$

$$p'(x) = (.5+x)v_0 - 2xv_1 + (x-.5)v_2$$

In these three coefficients we see the three columns of D:

$$D = \begin{pmatrix} 1.5 & -2 & .5 \\ .5 & 0 & -.5 \\ -.5 & 2 & -1.5 \end{pmatrix}$$

For the general case there are formulas, whose derivation we won't go into (see *Spectral Methods in MATLAB*).

Strangely, nobody wrote down a formula for these Chebyshev differentiation matrices until Gottlieb, Hussaini and Orszag in 1984.

[`cheb.m`, from *Spectral Methods in MATLAB*]

In Chebfun, you can get these matrices with `diffmat`. To be precise, `diffmat(N+1)` corresponds to `cheb(N)`, except with some sign changes since Chebfun numbers Chebyshev grids from left to right whereas we've defined them from right to left.

Here's an example of using these matrices to solve a PDE.

$$u_{tt} = u_{xx} \quad \text{on } [-1,1] \quad \text{with } u(\pm 1) = 0.$$

Leap frog:

$$\frac{v^{n+1} - 2v^n + v^{n-1}}{k^2} = D^2 v^n$$

where D is the first-order spectral differentiation matrix.

[`m56_waveeqcheb.m`]

It's kind of amazing to watch the wave propagating so smoothly through a grid that's irregular.

What if we had a different BC? Here's an example:

$$u(-1) = 0, \quad u'(1) = 0.$$

We can approximate this by enforcing the first row of the system of equations

$$Dv^n = 0$$

at each step n. That is, $dv^n = 0$, where $d = D_0^{0:N}$, or equivalently, in MATLAB notation,

```
v(1)   =  -D(1,2:end) v(2:end) / D(1,1) .
```

(Draw sketch.)

[`m56neumann.m` = `m55_waveeqcheb.m` again with this line commented in]

Closely related techniques are at the heart of Chebfun. See Aurentz & T, "Block operators and spectral discretizations" (handout).

The next example code does much the same, but in 2D:

$$u_{tt} = u_{xx} + u_{yy}.$$

Here's the new wrinkle here. We store the grid data in a matrix, not a vector. Then we compute

$$u_{yy} \text{ by multiplying this matrix on the left by } D^2$$

and

$$u_{xx} \text{ by multiplying this matrix on the right by } (D^2)^T$$

[`m57_leapfrog2Dcheb.m`]

Like Fourier spectral differentiation, Chebyshev spectral differentiation can be carried out with the FFT. The details are a bit tricky and we won't go into this here.

In practice, one often mixes both. For example, one can discretize flow in a pipe like this:

```
      r :  Chebyshev (since bounded)

  theta :  Fourier (since periodic)

      z :  Fourier (assuming periodicity)
```

Now the point of all these spectral methods is high accuracy. Our simple wave equation examples aren't the best to illustrate this, since the accuracy is corrupted by merely 2nd-order time-stepping. For more complicated time-dependent problems, such as reaction-diffusion equations, one can use fourth-order time-stepping instead. With fairly small time steps, one can then get e.g. 10 digits of accuracy for a nonlinear PDE.

Here's a final example that uses only Euler time-stepping, and is qualitatively but not quantitatively accurate, the **Allen-Cahn equation**,

$$u_t = u_{xx} + u - u^3, \qquad u(-1) = -1, \quad u(1) = 1.$$

(Cf. earlier Fisher-KPP eq., with $u - u^2$: `m39_FisherKPP.m`)

[Allen-Cahn page from *PDE Coffee Table Book*]

[`m58_allencahn.m` - Allen-Cahn equation]

In Chebfun, you can explore the Allen-Cahn equation through the scalar PDE Chebgui demo by that name, and also via `spin('ac')`.

With the Allen-Cahn equation — as indeed also with the Gray-Scott equations we began with — you can see the effect of *metastability*: structures that have finite but very long lifetimes, often depending exponentially or super-exponentially on a parameter. See the table of chemical elements from *Nature* in January 2019. Half-lives differ in the table from more than one year to less than one millisecond.