MAT 228B Notes

Sam Fleischer

February 6, 2016

1 Implicit-Time methods in Multi-D

$u_t = b\nabla^2 u$. Crank-Nicolson is

$$\left(I - \frac{\Delta t b}{2}L\right)u^{n+1} = \left(I + \frac{\Delta t b}{2}L\right)u^n$$

Backward Euler is

$$(I - \Delta t b L)u^{n+1} = u^n$$

BDF-2 is

$$\left(I - \frac{2}{3}\Delta t L\right)u^{n+1} = \text{stuff}$$

so the inversion is pretty similar for BDF-1,2,3,etc. We have to solve

$$(I - \beta\Delta t L)u^{n+1} = r$$

every timestep. Direct-solve (Gaussian elimination) is expensive. We can use iterative methods like SOR, Multigrid, PCG. Or we can use some specialized direct methods like Block-Cyclic Reduction or FFT method (both of these require structure and constant coefficients). Asymptotically, MG and FFT are the best. MG has more overhead, but is more general.

1.1 How well do the iterative methods work?

The condition number of L is $\mathcal{O}(1)\Delta x^2$. This is pretty bad.. the bigger the condition number, the slower the convergence (PCG). Let $A = I - \beta\Delta t L$. We can really just look at $\beta\Delta t L$. Then $\kappa(A) = \mathcal{O}\left(\frac{\Delta t}{\Delta x^2}\right)$ (remember $\kappa(A)$ is notation for the condition number of A). If $\Delta t = \mathcal{O}(\Delta x)$, then $\kappa(A) = \mathcal{O}\left(\frac{1}{\Delta x}\right)$.

So, all of these methods will work one order of magnitude faster than they do for the Poisson equation.

The actual convergence rate depends on the size of $\frac{\beta\Delta t}{\Delta x^2}$. Two extreme cases:

- $\frac{\beta\Delta t}{\Delta x^2} \to 0$
- $\frac{\beta\Delta t}{\Delta x^2} \to \infty$

For Backward Euler ($\beta = b$),

$$(I - \Delta t b L)u^{n+1} = u^n + \Delta t f^{n+1}.$$

If f contains boundary conditions, then we expect $f = \mathcal{O}\left(\frac{1}{\Delta x^2}\right)$.

- If $\frac{\beta\Delta t}{\Delta x^2} \to 0$, then $A \to I$ and the iterative methods converge very fast.
- If $\frac{\beta\Delta t}{\Delta t^2} \to \infty$, then $A \to -\beta\Delta t L$. In that limit, the equations look like

  $$-\Delta t b L u^{n+1} = \Delta t f^{n+1}$$

  This is a discrete Poisson equation. So the worst case scenario is that the iterative methods work better (converge faster) than they do for the Poisson equation.

A Multigrid solver on a 64^2 periodic domain has convergence factor for the Poisson equation is $\rho \approx 0.16$. Then the number of iterations per digit accuracy is $-\frac{1}{\log_{10}\rho} \approx 1.26$.

1.2 Results for Various βs

$(I - \Delta t \beta L)$:

$\beta = 1$	$\rho \approx 0.11$	$-\frac{1}{\log_{10}\rho} \approx 1.04$	20% fewer iterations
$\beta = 0.1$	$\rho \approx 0.05$	$-\frac{1}{\log_{10}\rho} \approx 0.77$	40% fewer iterations

and this is with $\Delta t = \Delta x$. Then $\frac{\beta \Delta t}{\Delta x^2} = \frac{\beta}{\Delta x} = \frac{\beta}{64}$ if the grid is 64^2.

We actually might do better than this since time-dependent problems give us an initial guess for each step. We'll take $\left(u^{n+1}\right)^0 = u^n$, i.e. the initial guess on the $(n+1)$th timestep is the nth timestep.

2 Exploiting the Time-Dependencies of the Heat Equation

There is another way to solve the equation, which was not available for the Poisson equation because it was time-independent.

- ADI scheme (Alternating Direction, Implicit)
- LOD scheme (Locally One-Dimensional), which is good for structured grids only.. not often used in practice.

Exploiting the Laplacian in 2D, $\nabla^2 u = u_{xx} + u_{yy}$, i.e. $L = L_x + L_y$. Intuitively, we diffuse in each dimension sequentially.

Crank-Nicolson is

$$\left(I - \frac{b\Delta t}{2} L_x - \frac{d\Delta t}{2} L_y\right) u^{n+1} = \left(I + \frac{b\Delta t}{2} L_x + \frac{d\Delta t}{2} L_y\right) u^n$$

So, sequentially, the LOD scheme is

$$\left(I - \frac{b\Delta t}{2} L_x\right) u^* = \left(I + \frac{b\Delta t}{2} L_x\right) u^n, \qquad \text{followed by} \qquad \left(I - \frac{b\Delta t}{2} L_y\right) u^{n+1} = \left(I + \frac{b\Delta t}{2} L_y\right) u^*$$

It turns out this is pretty reasonable. The LOD scheme looks like a fractional stepping method. The "half" step is in the x direction. The "full" step is that, followed by the step in the y direction. In fractional stepping methods, we ignore part of the ODE in each fractional step.. not like RK.