AM119: HW4 and more OpenFOAM tutorials

Prof. Trask

April 4, 2016

1 Assignment 5: Implicit schemes and iterative linear solvers (due Apr. 11)

In class this week we learned where the stability restrictions come from, and saw that they can be removed by switching from an explicit to an implicit scheme. This shift requires some increased complexity, as implicit schemes require the solution of a linear system of equations. Luckily, the linear system in the finite difference and finite volume methods is sparse and there are efficient linear solvers. For this homework assignment, we will be altering our 1D finite difference code from the first homework assignment to use the backward Euler scheme, and we will use Jacobi's method to invert the resulting system. There are substantially faster algorithms for solving this problem than Jacobi, but Jacobi is extremely simple to code up. After this, we'll alter our advection diffusion solver in OpenFoam to handle diffusion implicitly.

2 1D code

The first component of this assignment is straightforward - just code up the scheme presented in class. For simplicity, we'll just solve the unsteady diffusion equation and treat the advection term explicitly.

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + a\frac{u_i^n - u_{i-1}^n}{\Delta x} - \nu\frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{\Delta x^2} = f_i$$

which can be rewritten:

$$(-\frac{\nu\Delta t}{\Delta x^2})u_{i+1}^{n+1} + (1 + \frac{2\nu\Delta t}{\Delta x^2})u_i^{n+1} + (-\frac{\nu\Delta t}{\Delta x^2})u_{i-1}^{n+1} = u_i^n + \Delta t f_i - \frac{a\Delta t}{\Delta x}(u_i^n - u_{i-1}^n)$$

This will require a rewrite of your update function to implement Jacobi's method. At this point we're all C++ experts so I'll let you decide what data structures you'll need for this. Do **not** store the tridiagonal matrix $\mathbf{A}$ - it is only faster to use this method if you exploit the sparsity of the matrix by implementing a function to compute the sparse matrix vector product $\mathbf{A}x$. If you need help getting started with this assignment please make an appointment to see me after class. The wikipedia page for Jacobi is pretty concise if you'd like a reference to follow`https://en.wikipedia.org/wiki/Jacobi_method`, otherwise this can be found in any computational linear algebra textbooks and some introduction to scientific computing texts.

Assignment part 1:

- Study the problem from HW1 by implementing the implicit scheme with a tolerance of $||r||_2/||b||_2 \leq 0.01$. For resolution of $N = 10, 100, 1000, 10000$, $t_{final} = 1$, and $a = 0, \nu = 1$, generate a table comparing the total runtime for the implicit method and the explicit method from HW1. For the implicit solver, use a timestep of $\Delta t = t_{final}/N$ - for the explicit solver use the stability restriction set up in HW1.
- For a single timestep and for the case $N = 1000$, generate a log-log plot showing the number of iterations required to reach convergence for a given value of $\nu\Delta t/\Delta x^2$.

Assignment part 2:

- Make sure that you understand how the linear solvers were set up in the OpenFoam tutorial today. Since the first part of this assignment may be a bit programming intensive, there is no formal assignment for this. We will be using linear solvers extensively next week however so make sure you drop by office hours if you had any trouble altering advectionDiffusionFoam to make it implicit.