Numerical Methods II, Courant Institute, Spring 2014

http://www.math.nyu.edu/faculty/goodman/teaching/NumMethII2014/index.html

Always check the `classes` message board before doing any work on the assignment.

Assignment 5, due end of finals week

Corrections: (none yet)

1. (*An example for Krylov space methods*) Let A be the tridiagonal matrix with 2 on the diagonal and -1 on the off diagonals. Consider the $d \times d$ matrix with $d = 2M+1$ (an odd number, for simplicity only). Consider $\mathbf{x} \in \mathbb{R}^d$, with components labelled $(x_{-M}, \ldots, x_0, \ldots, x_M)$. Consider the Krylov space $K_n(A, \mathbf{x}_0)$, with $\mathbf{x}_0$ having components $\mathbf{x}_{0,j} = 0$ if $j \neq 0$, and $\mathbf{x}_{0,0} = 1$. Find a formula for

$$\lambda_{1,n} = \min_{\mathbf{x} \in K_n(A,\mathbf{x}_0)} \frac{\mathbf{x}^t A \mathbf{x}}{\mathbf{x}^t \mathbf{x}} \; .$$

   Conclude that $\lambda_{1,n}$ is not a good estimate of the true lowest eigenvalue $\lambda_1(A)$ if n is not close to $M \approx d/2$, for large d.

2. (*An example of ill conditioning*) Consider the gradient descent line search algorithm for solving $Ax = b$ with positive definite symmetric A. The problem is equivalent to minimizing $F(x) = \frac{1}{2}x^t Ax - x^t b$, over x of course. Let x_n be the current iterate, and $p_n = -\nabla F(x_n)$ be the (negative) gradient. Line search means taking $x_{n+1} = x_n + t_n p_n$. Consider an optimized line search strategy would take the *step size*, t_n, to minimize $F(x_n + tp_n)$. The minimization is a one dimensional minimization over t, with x_n and p_n fixed.

   (a) Is resulting iteration is linear? That means: the error at stage n is $e_n = x_n - x$. Is there a matrix B with $e_{n+1} = Bx_n$?

   (b) Show that this algorithm has the orthogonality property $p_{n+1}^t p_n = 0$.

   (c) Consider the problem in two dimension $d = 2$ with $Q(x) = \frac{1}{2}x^t Ax$ and $Q_1(x) = x_1^2 + \epsilon x_2^2$. Give a quantitative description of the convergence rate of this method (a description involving some power of ϵ) as $\epsilon \to 0$. You may find the orthogonality property useful.

3. (The A-stable second order BDF formula)

   (a) Find a difference approximation of the form

$$\partial_x u(x) = \frac{1}{h}\left[c_2 u(x-2h) + c_1 u(x-h) + c_0 u(x)\right] + O(h^2) \; .$$

(b) Use this formula to construct a second order implicit ODE time stepping method of the form

$$x_n = a_1 x_{n-1} + a_2 x_{n-2} + \Delta t f(x_n) \ .$$

Second order accuracy (as we have seen) is equivalent to third order local truncation error, which means that the exact solution to $\dot{(x)} = f(x)$ satisfies

$$x(t) = a_1 x(t - \Delta t) + a_2 x(t - 2\Delta t) + \Delta t f(x(t)) + R(t, \Delta t) \ ,$$

where the local truncation error satisfies $R(t, \Delta t) = O(\Delta t^3)$. Show that the coefficients a_1 and a_2 are completely determined by this order of accuracy.

(c) Show that this method is A-stable, which means that it is stable when applied to $\dot{x} = \lambda x$ for any $\Delta t > 0$ as long as $\mathrm{Re}(\lambda) \leq 0$.

4. (*adaptive method*) This exercise asks you to create a more reliable version of the ODE solver you wrote in the last assignment.

(a) Upgrade the solution algorithm from three stage Runge-Kutta to a four stage fourth order accurate Runge Kutta method. You can get the coefficients for the fourth order method from a book or from Wikipedia. That's what you would do in practice. Use the unit test procedures you wrote for the previous assignment to verify the fifth order local truncation error and fourth order overall error. If you wrote good code for the previous assignment, this should be easy. I hope you are impressed with the difference good code and unit testing can make.

(b) Time step adaptivity chooses time step Δt_n in order to keep the overall error within a specified tolerance. The solution times are $t_{n+1} = t_n + \Delta t_n$, and the corresponding approximation values are $x_n \approx x(t_n)$. Let $S(x, \Delta t)$ be the approximate nonlinear solution "operator" constructed by the one step (and four stages) of the Runge Kutta method. If $\dot{x} = f(x)$, then the local truncation error is $R(x, \Delta t) = S(x, \Delta t) - x(t + \Delta t)$. You should already have a procedure that takes one Runge Kutta time step with given Δt. In the present notation, this means evaluating $S(x, \Delta t)$ for a given x and Δt. The result of taking two time steps of size $\Delta t/2$ is

$$S_2(x, \Delta t) = S(S(x, \Delta t/2), \Delta t/2) \ .$$

Show that

$$S_2(x, \Delta t) - S(x, \Delta t) = C\, R(x, \Delta t) + O(\Delta t^{p+2}) \ ,$$

for a Runge Kutta method of overall order p. The constant C depends on p, but not on the specific ODE. Interpret this as saying that $S_2(x, \Delta t) - S(x, \Delta t)$ is an accurate error estimator.

(c) Write an adaptive ODE solver using the above error estimator. At each time step, make sure that Δt_n is chosen so that

$$\|R(x_n, \Delta t_n)\| \leq\approx \epsilon \Delta t_n \ .$$

The symbol $\leq\approx$ means "approximately less than or equal", so the left side could be a little larger than the right side, but not much. One way to do this is: start time step n with a guess $\Delta t_n = \Delta t_{n-1}$. Estimate the truncation error. If it is too large, reduce Δt_n by a factor of 2. If it is too small, increase Δt_n by a factor of 2. Make sure you don't try to both increase Δt and decrease it in the same time step, because that can lead to an infinite loop. One way to do this is to write a procedure that calls the time stepper three times and returns the size of the estimated error.

(d) Use this fancy adaptive method on your interacting body simulation with up to, say, ten bodies. Try to make some visualization that shows the time step dynamically growing and shrinking as bodies separate and come close to each other. With luck, you should not see the false ejections many of you got with fixed time step methods. Make a movie, but be careful that movie time should be physical time, not iteration number. Email the instructor and the grader a tarball of the code you used.