18.335 Problem Set 4

Due Friday, 26 October 2012.

Problem 1: Hessenberg

In class, we described an algorithm to find the Hessenberg factorization $A = QHQ^*$ of an arbitrary matrix A, where H is upper-triangular plus nonzero elements just below the diagonal, and H has the same eigenvalues as A. Suppose A is Hermitian, in which case H is Hermitian and tridiagonal. Given the Hessenberg factorization H, we mentioned in class that many things become much easier, e.g. we can evaluate $p(z) = \det(A - zI) = \det(H - zI)$ in $O(m)$ operations for a given z.

(a) Let B be an arbitrary $m \times m$ tridiagonal matrix. Argue that $\det B = B_{m,m} \det B_{1:m-1,1:m-1} - B_{m-1,m} B_{m,m-1} \det B_{1:m-2,1:m-2}$.. Use this recurrence relation to write a Matlab function `evalpoly.m` that evaluates $p(z)$ in $O(m)$ time, given the tridiagonal H and z as arguments. Check that your function works by comparing it to computing $\det(H - zI)$ directly with the Matlab `det` function. (Hint: look up formulas for determinants in terms of cofactors or minors.)

(b) Explain how, given the tridiagonal H, we can compute also the derivative $p'(z)$ for a given z in $O(m)$ operations. (*Not* the coefficients of the polynomial p', just its value at z!). Modify your evalpoly.m routine to return both $p(z)$ and its derivative $p'(z)$. That is, your function should look like:

```
[p,pderiv] = evalpoly(H,z)
......compute p, pderiv.....
```

Check that your function works by comparing your $p'(z)$ to $[p(z + \Delta z) - p(z - \Delta z)]/2\Delta z$ for various z and small Δz.

(c) Using your function evalpoly, implement Newton's method to compute some eigenvalues of a random real-symmetric matrix, and compare them to those returned by Matlab's eig function—how many significant digits of agreement do you get?

That is, to get a random real-symmetric A, compute `X=rand(m); A=X'*X;` .... then, compute `H=hess(A);` to get H. Then compute the eigenvalues with `eig(A)`, and apply Newton's method starting at a few different points to converge to some different eigenvalues.

(If you make your matrix too big, you might encounter overflow problems in which the determinant is bigger than the largest representable value, and you get $\pm\infty$. This is easily fixed by modifying your `evalpoly` routine to scale its result, but you needn't bother. Just use a smaller matrix, say 100×100.)

Problem 2: Q's 'R us

(a) Trefethen, problem 27.5

(b) Trefethen, problem 28.2

Reminder: final project proposals

A half-page final-project proposal is **due on October 26** (same day as the pset!), outlining the goal and scope of your proposed paper—this is mainly so that I can give you feedback on whether your project is reasonable. Problems motivated by your research are perfectly fine, although you shouldn't simply recycle something you've already done. The only restriction is that, since PDEs are covered in 18.336 and other courses, I don't want projects where the primary focus is how to discretize the PDE (e.g. no projects on discontinuous Galerkin methods or stable timestepping, please). It is fine to take a discretized PDE as *input*, however, and then work on solvers, preconditioning, optimization, etcetera. Methods for ODEs are also fair game (especially recent developments that go beyond classic Runge-Kutta). One source of ideas might be to thumb through a copy of *Numerical Recipes* or a similar book and find a topic that interests you. Then go read some recent review papers on that topic (overview books like *Numerical Recipes* are not always trustworthy guides to a specific field).

You should **email** your final-project proposal to me; include *18.335 final project proposal* in the **subject** of the email. You can email the proposal to me early if you want! Your proposal should cite a couple of references that you will use as starting points.

See also the 18.335 web page, which answers some common questions about final projects.