18.335 Midterm Solutions, Fall 2013

Problem 1: GMRES (20 points)

(a) We assume A is nonsingular, in which case $A^n b \neq 0$ (except in the trivial case $b = 0$, for which we already have an exact solution $x = 0$ to $Ax = b$). Now, we are told to suppose $A^n b \in \mathscr{K}_n \implies A^n b = \sum_{k<n} c_k A^k b$ for some coefficients c_k. Let c_ℓ denote the first nonzero coefficient (i.e. $c_\ell \neq 0$ and $c_k = 0$ for $k < \ell$; in most cases $\ell = 0$). Then $A^n b = \sum_{k=\ell}^{n-1} c_k A^k b$ implies that we can solve for the $A^\ell b$ term as:

$$A^\ell b = \frac{1}{c_\ell}\left(A^n b - \sum_{\ell<k<n} c_k A^k b\right) \implies b = A\left[\frac{1}{c_\ell}\left(A^{n-\ell-1}b - \sum_{\ell<k<n} c_k A^{k-\ell-1} b\right)\right].$$

But we are solving $b = Ax$, and by inspection

$$x = \frac{1}{c_\ell}\left(A^{n-\ell-1}b - \sum_{\ell<k<n} c_k A^{k-\ell-1} b\right) \in \mathscr{K}_n,$$

since $0 \le k - \ell - 1 < n$ (since $\ell < k < n$) and $0 \le n - \ell - 1 < n$ (since $0 \le \ell < n$). But if the exact solution $x \in \mathscr{K}_n$, then we can obtain the exact solution from Q_n and we don't *need* to compute q_{n+1}. We are done, so breakdown is a good thing.

(b) Suppose b is a linear combination of n of the eigenvectors of A. Then $A^k b$ will still be a linear combination of those eigenvectors for all k, and hence the Krylov space can never have dimension $> n$. So, we must break down on the n-th step, when breakdown must occur if we try to compute q_{n+1}.

Technically, we must find n eigenvectors with distinct eigenvalues in order for $A^k b$ to be linearly independent for $k < n$, i.e. for it to break down in exactly n steps.

Problem 2: Conditioning (20 points)

The following parts can be solved *independently*.

(a) We want to avoid squareing the condition number of A. So, we compute the reduced QR factorization $A = \hat{Q}\hat{R}$ (Householder is the most efficient stable way), in which case $A^*A = \hat{R}^*\hat{R}$. Since A is full-rank, $\hat{R}$ is a nonsingular $n \times n$ matrix. Hence $C = (\hat{R}^{-1})^*\hat{R}^{-1}$, and $C_{ij} = e_i^* C e_j = (\hat{R}^{-1}e_i)^*(\hat{R}^{-1}e_j)$. Since $\hat{R}$ is upper-triangular, we can compute $x_i = \hat{R}^{-1}e_i$ efficiently by solving $\hat{R}x_i = e_i$ via backsubstitution.

(b) From class, the condition number of $f(x) = Ax$ is simply $\kappa(x) = \|A\|_2\|x\|_2/\|Ax\|_2$, since the Jacobian is A. (I told you to use $\|A\|_F$ for the norm of A, but that really applies when you have a *choice* of norms, i.e. in the second part; in the condition-number formula you *must* use the induced norm of the Jacobian matrix. However, the question was a bit confusing here.)

To get the condition number of $f(A) = Ax$, we first need to to get the Jacobian. Let'a define the input A as a "1d" vector a of length mn:

$$a = \begin{pmatrix} (A_{1,:})^T \\ (A_{2,:})^T \\ \vdots \\ (A_{m,:})^T \end{pmatrix},$$

i.e. a consists of the rows of A (transposed to column vectors), one after the other, in sequence. (i.e., row-major storage of A.) There are m outputs f_i of Ax, each one of which dots one row of A with x.

Hence, in terms of a, the $m \times (mn)$ Jacobian matrix looks like

$$J = \begin{pmatrix} x^T & & & \\ & x^T & & \\ & & \ddots & \\ & & & x^T \end{pmatrix}.$$

Since this is block-diagonal, it is easy to figure out $\sup_{z \neq 0} \frac{\|Jz\|}{\|z\|}$. Let's write z as

$$z = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix}$$

in terms of vectors $x_k \in \mathbb{C}^n$. Note that, under the Frobenius norm, $\|A\|_F = \|z\|_2$. Then

$$\|J\|_2 = \frac{\|Jz\|_2}{\|z\|_2} = \sqrt{\frac{|x^T x_1|^2 + |x^T x_2|^2 + \cdots + |x^T x_m|^2}{x_1^* x_1 + x_2^* x_2 + \cdots + x_m^* x_m}},$$

which is clearly maximized when $x_k = \alpha_k \bar{x}$ (to maximize the dot products $x^T x_k \to |\alpha_k|^2 \|x\|_2^2$ over all vectors x_k of a given length) for some scalar $\alpha_k \in \mathbb{C}$, giving $\|J\|_2 = \|x\|_2 \sqrt{\frac{\sum |\alpha_k|^2}{\sum |\alpha_k|^2}} = \|x\|_2$. Hence, the condition number is $\kappa(A) = \frac{\|J\|}{\|Ax\|/\|A\|} = \frac{\|x\|_2 \|A\|_F}{\|Ax\|_2}$, which is almost exactly the same the condition number for $f(x) = Ax$ above, except that we substitute $\|A\|_F$ for $\|A\|_2$. Due to the equivalence of norms, however, this means that the condition numbers differ only by at most a constant factor independent of A or x.

(I asked you to use the Frobenius norm largely because it made it eaasier to compute the induced norm $\|J\|_2$ in the second part. Otherwise you would have had to convert z back to a matrix and used the induced L_2 norm of *that* matrix for $\|z\|$ in the *denominator* of the $\|J\|_2$ formula. It's possible to work this out, but it seemed like an unnecessary amout of complexity to get something that differs only by a constant factor. The usual principle here is that, because of the equivalence of norms, we pick whatever norm is most convenient when we are discussing conditioning.)

Problem 3: QR updating (20 points).

Suppose you are given the QR factorization $A = QR$ of an $m \times n$ matrix A (rank $n < m$). Describe an efficient $O(m^2 + n^2) = O(m^2)$ algorithm to compute the QR factorization of a rank-1 update to A, that is to factorize $A + uv^* = Q'R'$ for some vectors $u \in \mathbb{C}^m$ and $v \in \mathbb{C}^n$, following these steps:

(a) (Note that this applies to the full QR factorization, where Q is an $m \times m$ unitary matrix, not to the reduced QR factorization with an $m \times n$ $\hat{Q}$!) $Q'R' = A + uv^* = QR + uv^* = Q(R + zv^*)$ implies that $Qz = u$ or $z = Q^* u$. Multiplying an $m \times m$ matrix $Q^*$by the vector u costs $\Theta(m^2)$ operations.

(b) The $R+zv^*$ matrix looks like this:

$$R+zv^* = \begin{pmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \\ & & & & \times \\ \\ \\ \\ \end{pmatrix} + \begin{pmatrix} z\overline{v_1} & z\overline{v_2} & \cdots & z\overline{v_n} \end{pmatrix}.$$

This means that, *below the diagonal* of R, the entries in *every column* are multiples of the *same* vector z. So, if we perform Givens rotations from the bottom up to introduce zeros into the *first* column, this rotation will also introduce zeros in *all* the columns *until the diagonal is reached*.

More specifically, you were asked to apply the Givens rotations that rotate z into a multiple of e_1, from the bottom up. As explained above, this will introduce zeros into each column of $R+zv^*$ until the diagonal is reached. In column k, this means it will introduce zeros until you get to the point of rotating rows k and $k+1$. Because the (k,k) entry contains $R_{k,k}$, the Givens rotation designed for z will no longer work, and will leave *both* of these rows nonzero (and similarly for rows $< k$). Hence, the resulting matrix is **upper Hessenberg** as desired (one nonzero below each diagonal).

There is $\Theta(m)$ work required to rotate z via $m-1$ Givens rotations, and $\Theta(n^2)$ work required to apply these rotations to $R+zv^*$ starting from the diagonal rows. And hence $\Theta(m+n^2)$ work overall; I don't mind if you ignore the $\Theta(m)$ term since it is negligible compared to the $\Theta(m^2)$ term from part (a). (Naively, these Givens rotations to introduce zeros into the first column will require $\Theta(mn)$ work because of the cost of applying them to the other columns, but you don't actually have to perform the rotations of the other columns for rows $> n$ since we know *a priori* that this will just introduce zeros.)

(c) Given upper-Hessenberg form, we just need to apply one Givens rotation to each column (to rows k and $k+1$ for column k) to restore tridiagonal form. There are n columns, and on average $\Theta(n)$ work per rotation (since the rotation has to apply to all the columns $\geq k$), for $\Theta(n^2)$ work overall.

(You solved a very similar problem for homework, in the context of the upper-Hessenberg GMRES least-squares problem.)