18.335 Midterm Solutions

Problem 1: Schur, backsubstitution, complexity (20 points)

You are given matrices $A$ ($m \times m$), $B$ ($n \times n$), and $C$ ($m \times n$), and want to solve for an unknown matrix $X$ ($m \times n$) solving:

$$AX - XB = C.$$  

We will do this using the Schur decompositions of $A$ and $B$. (Recall that any square matrix $S$ can be factorized in the Schur form $S = QUQ^*$ for some unitary matrix $Q$ and some upper-triangular matrix $U$.)

(a) $AX - XB = C = QA^r_AQ^*_A X - XQBUBQ^*_B$, and hence (multiplying on the left by $Q^*_A$ and on the right by $Q_B$), we obtain

$$U_AQA^r_A XQBUBQ^*_B = U_A^r CQ_B,$$

so $A' = U_A, B' = U_B, C' = Q^*_A C Q_B$, and $X' = Q_X XQB$. To get $X$ from $X'$, we obtain $X = Q^*_A X'B$.  

(b) The last row of $A'X' - X'B' = C'$, since $A'$ is upper-triangular, is:

$$A'_{mm}X'_{mj} - X'_{mj}B' = C'_{mj} = X'_{mj}(A'_{mm}I - B'),$$

which is only in terms of the last row $X'_{mj}$ of $X'$. To find this last row, then, we merely need to solve the system of equations above—since $A'_{mm}I - B'$ is upper-triangular, we can do this by backsubstitution in $O(n^2)$ operations. Or, I guess, technically, this is “forward” substitution because you start with the first column of $B'$ and move right, but whatever—it’s the same thing under a permutation. [Although this is a row-vector problem, we can obviously transpose to get the familiar column-vector problem, in which case $(A'_{mm}I - B')^T$ is lower-triangular.]

(c) More generally, the $j$-th row of $A'X' - X'B' = C'$ can be written purely in terms of the $j$-th and later rows of $X'$, since $A'$ is upper-triangular:

$$A'_{jj}X'_{jj} + \sum_{i > j} A'_{ij}X'_{ij} - X'_{jj}B' = C'_{jj},$$

and hence

$$X'_{jj} (A'_{jj}I - B') = C'_{jj} - \sum_{i > j} A'_{ij}X'_{ij},$$

which is again an upper-triangular system of equations. It takes $2(m - j)n$ operations to construct the right-hand side, and $O(n^2)$ operations to solve by backsubstitution.

(d) We have to solve for $m$ rows. Each of them requires an $O(n^2)$ backsubstitution, for $O(mn^2)$ operations. There are also $\approx \sum_{j=1}^m 2(m - j)n = O(m^2 n)$ flops to compute the right-hand sides. Finally, to compute $X = Q_A X' Q_B^*$ requires two matrix multiplies, for $2m^2n + 2mn^2$ flops. So, the total complexity is $O(m^2 n) + O(mn^2)$, not including the $O(m^3) + O(n^3)$ time for the Schur factorizations.

Problem 2: Stability (20 points)

Since it is backwards stable (with respect to $A$ and/or $b$), we obtain an $x + \delta x$ such that $(A + \delta A)(x + \delta x) = b + \delta b \approx A(x + \delta x) + \delta A x$, where $||\delta A|| = O(\epsilon_{\text{machine}})||A||$ and $||\delta b|| = O(\epsilon_{\text{machine}})||b||$. That means that the residual, computed in exact arithmetic, would be $r = b - A(x + \delta x) = \delta A x - \delta b$. The norm of this is $\leq ||\delta A|| ||x|| + ||\delta b|| \leq ||\delta A|| ||x|| + ||\delta b|| = ||A|| ||x|| + ||b|| + O(\epsilon_{\text{machine}})$. But $||x|| = ||A^{-1}b|| \leq ||A^{-1}|| ||b||$, and so we obtain $||r|| \leq ||x(A) + 1 || ||b|| O(\epsilon_{\text{machine}})$. However, I didn’t specify whether the backwards stability was with respect to $A$ or $b$; if you only assumed the latter you wouldn’t have gotten the $x(A)$ term.

This is still not quite right, however, if the residual $r$ itself is computed in floating-point arithmetic. In particular, the computation of $b - Ay$ in floating-point for any $y$ is also backwards stable with respect
to y, so in computing $b - A(x + \delta x)$ we obtain $b - A(x + \delta x + \delta x')$ where $\|\delta x'\| = \|x\|O(\epsilon_{\text{machine}}) \leq \|A^{-1}\| b\|O(\epsilon_{\text{machine}})$. Hence, this gives us an additional term $A\delta x'$ in the residual, which has magnitude $\leq \|A\| \|\delta x'\| \leq \kappa(A)\|b\|O(\epsilon_{\text{machine}})$.

Adding these two sources of error, we obtain a residual whose magnitude proportional to $\kappa(A)\|b\|O(\epsilon_{\text{machine}})$.

**Problem 3: Conjugate gradient (20 points)**

(a) CG does not change the component of $x_n$ in the nullspace (the span of the zero-$\lambda$ eigenvectors).

Proof: If we expand $x_j = \sum_i \gamma_i^{(j)} q_i$ in the eigenvectors $q_i$ with some coefficients $\gamma_i^{(j)}$, we see immediately that $Ax_j = \sum_{i \geq k} \lambda_i \gamma_i^{(j)} q_i$ is in the span of the nonzero-$\lambda$ eigenvectors of $A$; equivalently, it is perpendicular to the nullspace. Hence, the residual $r_j = b - Ax_j$ (which we compute by recurrence in the CG algorithm) is also perpendicular to the nullspace. Since all the residuals are perpendicular to the nullspace, and since the directions $d_j$ are linear combinations of the residuals (via Gram-Schmidt), the directions $d_j$ are also perpendicular to the nullspace. Hence, when we compute $x_n = \lambda_{n-1} + a_{n-1}$, we do not change the components of $x$ in the nullspace, and $\gamma_i^{(n)} = \gamma_i^{(n-1)}$ for $i \leq k$.

(b) Because CG only changes $x_n$ in directions perpendicular to the nullspace, it is equivalent to doing CG on the nonsingular problem $Ax = b$ acting within the column space of $A$. Since $x_0 = 0$, it initially has no (nonzero) component in the nullspace and hence $x_n$ has no component in the nullspace. Hence, if $b = \sum_{i > k} \beta_i q_i$ for some coefficients $\beta_i$, it converges to $x_n \rightarrow \sum_{i > k} \frac{\beta_i}{\lambda_i} q_i$. The rate of convergence is determined by the square root of the condition number of $A$ within this subspace, i.e. at worst the convergence requires $O(\sqrt{\lambda_m/\lambda_{k+1}})$ iterations, assuming we have sorted the $\lambda_j$'s in increasing order. (Not including possible superlinear convergence depending on the eigenvalue distribution.)

(c) If we choose the initial guess $x_0 \neq 0$, it will still converge, but it may just converge to a different solution—the component of $x_0$ in the nullspace has no effect on CG at all, and the component in the column space is just a different starting guess for the nonsingular CG in the subspace. That is, since the component $\sum_{i \leq k} \gamma_i q_i$ of $x_0$ in the nullspace is not changed by CG, we will get (in the notation above) $x_n \rightarrow \sum_{i \leq k} \gamma_i q_i + \sum_{i > k} \frac{\beta_i}{\lambda_i} q_i$.

(d) Just set $b = 0$ and pick $x_0$ to be a random vector, and from above it will converge to a vector in the nullspace in $O(\sqrt{\lambda_m/\lambda_{k+1}})$ iterations at worst.

**Problem 4: Rayleigh quotients (20 points)**

Let the smallest-$\lambda$ eigensolution of $B$ be $B\lambda_1 = \lambda_1 q_1$ where $q_1'q_1 = 1$. Let $x = \left( \begin{array}{c} q_1 \\ 0 \end{array} \right)$, in which case the Rayleigh quotient is $r(x) = \lambda_1$ by inspection, and since this is an upper bound for the smallest eigenvalue of $A$, we are done.

**Problem 5: Norms and SVDs (20 points)**

If $B$ were just a real number $b$, this would be a $2 \times 2$ matrix $A = \left( \begin{array}{cc} 1 & b \\ b & 1 \end{array} \right)$, which has eigenvalues $1 \pm b$ for eigenvectors $\left( \begin{array}{c} 1 \\ \pm 1 \end{array} \right)$. We would immediately obtained the desired result since $\|B\| = |b|$ and $\|A\|_2$ is the ratio of the maximum to the minimum eigenvalue. Now, we just want to use a similar strategy for the general case where $B$ is $m \times n$, where from the SVD we can write:

$$A = \left( \begin{array}{c} I \\ B^* \\ I \end{array} \right) = \left( \begin{array}{c} I \\ V^T U^* \Sigma V^* \\ I \end{array} \right).$$
That is, we expect to get ± combinations of eigenvectors of $B$.

For simplicity, let’s start with the case where $B$ is square $m \times m$, in which case $\Sigma = \Sigma^T$ (diagonal) and $U$ and $V$ are all $m \times m$. In this case, consider the vectors corresponding to the columns of $X_\pm = \begin{pmatrix} U \pm V \end{pmatrix}$. In this case,

$$AX_\pm = \begin{pmatrix} I & U \Sigma V^* \\ V \Sigma U^* & I \end{pmatrix} \begin{pmatrix} U \pm V \\ V \Sigma \pm V \end{pmatrix} = X_\pm (I \pm \Sigma),$$

Since the matrix at right is diagonal, this means that the columns of $X_\pm$ are eigenvectors of $A$, with eigenvalues $1 \pm \sigma_i$ where $\sigma_i$ are the singular values of $B$ (possibly including some zeros from the diagonal of $\Sigma$ if $B$ is not full rank). These are, moreover, all of the 2$m$ eigenvalues of $A$. Since $A$ is Hermitian, eigenvalues are the same thing as the singular values, and hence the maximum singular value of $A$ is $1 + \max \sigma_i$ and the minimum is $1 - \max \sigma_i$ (since we are given that $\|B\|_2 < 1$ and hence $\sigma_i < 1$), and hence $\kappa(A) = (1 + \max \sigma_i)/(1 - \max \sigma_i) = (1 + \|B\|_2)/(1 - \|B\|_2)$. Q.E.D.

What about the case where $B$ is not square? Suppose $m > n$, in which case $U$ is bigger than $V$ so it doesn’t make sense to write $X_\pm$ as above. However, there is a simple fix. In the definition of $X_\pm$, just pad $V$ with $m - n$ columns of zeros to make an $n \times m$ matrix $V_0$. Then $V^*V_0$ is the $n \times n$ identity matrix plus $m-n$ columns of zeros. Then we get

$$AX_\pm = \begin{pmatrix} I & U \Sigma V^* \\ V \Sigma U^* & V \Sigma V^* \end{pmatrix} \begin{pmatrix} U \pm V_0 \\ V \Sigma \pm V_0 \end{pmatrix} = X_\pm (I \pm \Sigma_0),$$

where $\Sigma_0$ is $\Sigma$ padded with $m-n$ columns of zeros to make a diagonal $m \times m$ matrix, noting that $V \Sigma^T = V_0 \Sigma_0^T = V_0 \Sigma_0$. The result follows as above. If $m < n$, the analysis is similar except that we pad $U$ with $n-m$ columns of zeros.

**Problem 6: Least-squares problems (20 points)**

We want to minimize $(Ax - b)^*W(Ax - b)$. The best thing to do is to turn this into a regular least-squares problem by breaking $W$ in “halves” and putting half on the left and half on the right. For example, we can compute the Cholesky factorization $W = R^*R$, and then we are minimizing $(RAx - Rb)^*(RAx - Rb)$, which is equivalent to solving the least-squares problem for $RA$ and $Rb$. This we could do, e.g., by computing the QR factorization $RA = Q'R'$, and then solve $R'x = Q^*Rb$ by backsubstitution. None of these steps has any particular accuracy problems.

Of course, there are plenty of other ways to do it. You could also compute $\sqrt{W}$ by diagonalizing $W = Q\Lambda Q^*$ and then using $\sqrt{W} = Q\sqrt{\Lambda}Q^*$. This might be a bit more obvious if you have forgotten about Cholesky. Again solving the least-squares problem with $\sqrt{WA}$ and $\sqrt{Wb}$, this works, but is a bit less efficient because eigenproblems take many more operations than Cholesky factorization.

We could also write down the normal equations $A^*WAx = A^*Wb$, derived from the gradient of $(Ax - b)^*W(Ax - b)$ with respect to $x$. However, solving these directly sacrifices some accuracy because (as usual) it squares the condition number of $A$. 

