18.335 Problem Set 3 Solutions

Problem 2: QR and orthogonal bases (10+(5+5+5) pts)

(a) If $A = QR$, then $B = RQ = Q^*AQ = Q^{-1}AQ$ is a similarity transformation, and hence has the same eigenvalues as shown in the book. Numerically (and as explained in class and in lecture 28), doing this repeatedly for a Hermitian A (the unshifted QR algorithm) converges to a diagonal matrix Λ of the eigenvalues in descending order. To get the eigenvectors, we observe that if the Q matrices from each step are Q_1, Q_2, and so on, then we are computing $\cdots Q_2^* Q_1^* A Q_1 Q_2 \cdots = \Lambda$, or $A = Q\Lambda Q^*$ where $Q = Q_1 Q_2 \cdots$. By comparison to the formula for diagonalizing A, the columns of Q are the eigenvectors.

(b) Trefethen, problem 10.4:

(i) e.g. consider $\theta = \pi/2$ ($c = 0$, $s = 1$): $Je_1 = -e_2$ and $Je_2 = e_1$, while $Fe_1 = e_2$ and $Fe_2 = e_1$. J rotates clockwise in the plane by θ. F is easier to interpret if we write it as J multiplied on the right by $[-1,0;0,1]$: i.e., F corresponds to a mirror reflection through the y (e_2) axis followed by clockwise rotation by θ. More subtly, F corresponds to reflection through a mirror plane corresponding to the y axis rotated clockwise by $\theta/2$. That is, let $c_2 = \cos(\theta/2)$ and $s_2 = \cos(\theta/2)$, in which case (recalling the identities $c_2^2 - s_2^2 = c$, $2s_2c_2 = s$):

$$\begin{pmatrix} c_2 & s_2 \\ -s_2 & c_2 \end{pmatrix}\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} c_2 & -s_2 \\ s_2 & c_2 \end{pmatrix} = \begin{pmatrix} -c_2 & s_2 \\ s_2 & c_2 \end{pmatrix}\begin{pmatrix} c_2 & -s_2 \\ s_2 & c_2 \end{pmatrix} = \begin{pmatrix} -c & s \\ s & c \end{pmatrix} = F,$$

which shows that F is reflection through the y axis rotated by $\theta/2$.

(ii) The key thing is to focus on how we perform elimination under a single column of A, which we then repeat for each column. For Householder, this is done by a single Householder rotation. Here, since we are using 2×2 rotations, we have to eliminate under a column one number at a time: given 2-component vector $x = \begin{pmatrix} a \\ b \end{pmatrix}$ into $Jx = \begin{pmatrix} \|x\|_2 \\ 0 \end{pmatrix}$, where J is clockwise rotation by $\theta = \tan^{-1}(b/a)$ [or, on a computer, atan2(b,a)]. Then we just do this working "bottom-up" from the column: rotate the bottom two rows to introduce one zero, then the next two rows to introduce a second zero, etc.

(iii) The flops to compute the J matrix itself are asymptotically irrelevant, because once J is computed it is applied to many columns (all columns from the current one to the right). To multiply J by a single 2-component vector requires 4 multiplications and 2 additions, or 6 flops. That is, 6 flops per row per column of the matrix. In contrast, Householder requires each column x to be rotated via $x = x - 2v(v^*x)$. If x has m components, v^*x requires m multiplications and $m-1$ additions, multiplication by $2v$ requires m more multiplications, and then subtraction from x requires m more additions, for $4m-1$ flops overall. That is, asymptotically 4 flops per row per column. The 6 flops of Givens is 50% more than the 4 of Householder.

The reason that Givens is still considered interesting and useful is that (as we shall see in the next problem set), it can be used to exploit *sparsity*: because it rotates only two elements at a time in each column, from the bottom up, if a column ends in zeros then the zero portion of the column can be skipped.

Problem 2: Matrix addition and the CPU (10+5+10)

See the pset 3 solution IJulia notebook.

Problem 3: Schur fine (10 + 15 points)

(a) First, let us show that T is normal: substituting $A = QTQ^*$ into $AA^* = A^*A$ yields $QTQ^*QT^*Q^* = QT^*Q^*QTQ^*$ and hence (cancelling the Qs) $TT^* = T^*T$.

The (1,1) entry of T^*T is the squared L_2 norm ($\|\cdot\|_2^2$) of the first column of T, i.e. $|t_{1,1}|^2$ since T is upper triangular, and the (1,1) entry of TT^* is the squared L_2 norm of the first row of T, i.e. $\sum_i |t_{1,i}|^2$. For these to be equal, we must obviously have $t_{1,i} = 0$ for $i > 1$, i.e. that the first row is diagonal.

We proceed by induction. Suppose that the first $j-1$ rows of T are diagonal, and we want to prove this of row j. The (j,j) entry of T^*T is the squared norm of the j-th column, i.e. $\sum_{i \le j} |t_{i,j}|^2$, but this is just $|t_{j,j}|^2$ since $t_{i,j} = 0$ for $i < j$ by induction. The (j,j) entry of TT^* is the squared norm of the j-th row, i.e. $\sum_{i \ge j} |t_{j,i}|^2$. For this to equal $|t_{j,j}|^2$, we must have $t_{j,i} = 0$ for $i > j$, and hence the j-th row is diagonal. Q.E.D.

(b) The eigenvalues are the roots of $\det(T - \lambda I) = \prod_i (t_{i,i} - \lambda) = 0$—since T is upper-triangular, the roots are obviously therefore $\lambda = t_{i,i}$ for $i = 1, \ldots, m$. To get the eigenvector for a given $\lambda = t_{i,i}$, it suffices to compute the eigenvector x of T, since the corresponding eigenvector of A is Qx.

x satisfies

$$0 = (T - t_{i,i}I)x = \begin{pmatrix} T_1 & u & B \\ & 0 & v^* \\ & & T_2 \end{pmatrix} \begin{pmatrix} x_1 \\ \alpha \\ x_2 \end{pmatrix},$$

where we have broken up $T - t_{i,i}I$ into the first $i-1$ rows $(T_1\, u\, B)$, the i-th row (which has a zero on the diagonal), and the last $m-i$ rows T_2; similarly, we have broken up x into the first $i-1$ rows x_1, the i-th row α, and the last $m-i$ rows x_2. Here, $T_1 \in \mathbb{C}^{(i-1)\times(i-1)}$ and $T_2 \in \mathbb{C}^{(m-i)\times(m-i)}$ are upper-triangular, and are non-singular because by assumption there are no repeated eigenvalues and hence no other $t_{j,j}$ equals $t_{i,i}$. $u \in \mathbb{C}^{i-1}$, $v \in \mathbb{C}^{m-i}$, and $B \in \mathbb{C}^{(i-1)\times(m-i)}$ come from the upper triangle of T and can be anything. Taking the last $m-i$ rows of the above equation, we have $T_2 x_2 = 0$, and hence $x_2 = 0$ since T_2 is invertible. Furthermore, we can scale x arbitrarily, so we set $\alpha = 1$. The first $i-1$ rows then give us the equation $T_1 x_1 + u = 0$, which leads to an upper-triangular system $T_1 x_1 = -u$ that we can solve for x_1.

Now, let us count the number of operations. For the i-th eigenvalue $t_{i,i}$, to solve for x_1 requires $\sim (i-1)^2 \sim i^2$ flops to do backsubstitution on an $(i-1)\times(i-1)$ system $T_1 x_1 = -u$. Then to compute the eigenvector Qx of A (exploiting the $m-i$ zeros in x) requires $\sim 2mi$ flops. Adding these up for $i = 1 \ldots m$, we obtain $\sum_{i=1}^m i^2 \sim m^3/3$, and $2m\sum_{i=0}^{m-1} i \sim m^3$, and hence the overall cost is $\sim \frac{4}{3}m^3$ flops ($K = 4/3$).

Problem 4: Caches and backsubstitution (5+15 points)

(a) For each column of X we get a cache miss to read each entry r_{ij} of the matrix R, an there are roughly $m^2/2$ of these. For large enough m, where $m^2 > Z$, these are no-longer in-cache when the next column is prcoessed, and hence incur new misses on each of the n columns. Hence there are at least $m^2 n/2$, or $\Theta(m^2 n)$ misses. There are also $\Theta(m^2 n)$ misses in reading X, but this only change the constant factor. The $\Theta(mn)$ misses in reading B (each entry is used exactly once) are asymptotically negligible. [Since $\Theta(m^2 n)$ does not depend on Z there is no asymptotic benefit from the cache: we have a *pessimal* cache-oblivious algorithm that achieves the worst case independent of Z.]

(b) We can solve this problem in a cache-oblivious or cache-aware fashion. I find the cache-oblivious algorithm to be more beautiful, so let's do that. We'll divide R, X, and B into $\frac{m}{2} \times \frac{m}{2}$ blocks for

sufficiently large m:[1]

$$\begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} \begin{pmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{pmatrix} = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix},$$

where R_{11} and R_{22} are upper-triangular. Now we solve this, analogous to backsubstitution, from bottom to top. First, for $k = 1, 2$, we solve

$$R_{22}X_{2k} = B_{2k}$$

recursively for X_{2k}. Then, for $k = 1, 2$ we solve

$$R_{11}X_{1k} = B_{1k} - R_{12}X_{2k}$$

recursively for X_{1k}. We use a cache-optimal algorithm (from class) for the dense matrix multiplies $R_{12}X_{2k}$, which requires $f(m) \in \Theta(m^3/\sqrt{Z})$ misses for each $\frac{m}{2} \times \frac{m}{2}$ multiply. The number $Q(m)$ of cache misses then satisfies the recurrence:

$$Q(m) = 4Q(m/2) + 2f(m) + 4m^2,$$

where the $4Q(m/2)$ is for the four recursive backsubsitutions and the $4m^2$ is for the two matrix subtractions $B_{1k} - R_{12}X_{2k}$. This recurrence terminates when the problem fits in cache, i.e. when $2m^2 + m^2/2 \leq Z$, at which point only $\sim 3m^2/2$ misses are required. (Since we are only interested in the asymptotic Θ results, these little factors of 3 and 4 don't matter much, and I'll be dropping them soon.) Noting that $f(m/2) \approx f(m)/8$, we can solve this recurrence as in class by just plugging it in a few times and seeing the pattern:

$$\begin{aligned} Q(m) &\approx 4[4Q(m/4) + 2f(m)/8 + 4m^2/4] + 2f(m) + 4m^2 \\ &= 4^2 Q(m/4) + 2f(m)\left[1 + \frac{1}{2}\right] + 4m^2\,[1+1] \\ &\approx 4^3 Q(m/8) + 2f(m)\left[1 + \frac{1}{2} + \frac{1}{2^2}\right] + 4m^2\,[1+1+1] \\ &\approx \cdots \\ &\approx 4^k \Theta[(m/2^k)^2] + 2f(m)\left[1 + \frac{1}{2} + \cdots + \frac{1}{2^{k-1}}\right] + 4m^2\,[k] \\ &\approx \Theta(m^2) + \Theta(m^3/\sqrt{Z}) + \Theta(m^2)k \end{aligned}$$

where $\left[1 + \frac{1}{2} + \cdots + \frac{1}{2^{k-1}}\right] \leq 2$ and k is the number of times we have to divide the problem to fit in cache, i.e. $3(m/2^k)^2/2 \approx Z$ so k is $\Theta[\log(m^2/Z)]$. Hence, for large m where the m^3term dominates over the m^2 and $m^2 \log m$ terms, we obtain

$$Q(m;Z) = \Theta(m^3/\sqrt{Z})$$

and hence we can, indeed, achieve the same asymptotic cache complexity as for matrix multiplication.

We could also get the same cache complexity in a cache-aware fashion by blocking the problem into m/b blocks of size $b \times b$, where b is some $\Theta(\sqrt{Z})$ size chosen so that pairwise operations on the individual blocks fit in cache. Again, one would work on rows of blocks from bottom to top, and the algorithm would look much like the ordinary backsubstitution algorithm except that the numbers b_{ij} etcetera are replaced by blocks. The number of misses is $\Theta(b^2) = \Theta(Z)$ per block, and there are $\Theta(\frac{m}{b} \times \frac{m}{b} \times \frac{n}{b})$ block operations, hence $\Theta(\frac{m^2 n}{b^3} \times b^2) = \Theta(m^2 n/\sqrt{Z})$ misses. This is a perfectly acceptable answer, too.

[1] If m is not even, then we round as needed: R_{11} is $\lceil \frac{m}{2} \rceil \times \lceil \frac{m}{2} \rceil$, R_{12} is $\lceil \frac{m}{2} \rceil \times \lfloor \frac{m}{2} \rfloor$, R_{21} is $\lfloor \frac{m}{2} \rfloor \times \lceil \frac{m}{2} \rceil$, and $R_{22}$is $\lfloor \frac{m}{2} \rfloor \times \lfloor \frac{m}{2} \rfloor$; similarly for X and B. These bookkeeping details don't change anything, though, so it is fine to just assume that m is a power of 2 for simplicity.