18.335 Take-Home Midterm Exam: Spring 2022 Solutions

Posted 1pm Wednesday April 6, due **1pm Friday April 8.**

Problem 0: Honor code

Copy and sign the following in your solutions:

I have not used any resources to complete this exam other than my own 18.335 notes, the textbook, running my own Julia code, and posted 18.335 course materials.

_____

your signature

Problem 1: (10+10+8+5 points)

Consider the two real vectors $u = (u_1, u_2, \ldots, u_n)^T$ and $v = (v_1, v_2, \ldots, v_n)^T$ and recall that the 1-norm of a vector x is defined as $\|x\|_1 = |x_1| + |x_2| + \cdots + |x_n|$.

(a) Show that computing the dot product $f(u,v) = u_1v_1 + u_2v_2 + \cdots + u_nv_n$ in floating point arithmetic with left to right summation is backward stable, and in fact that $\hat{f}(u,v) = f(u+\delta u, v) = f(u, v+\delta v)$ for some δu and δv satisfying $\|\delta u\|_1 \le \|u\|_1 \mathcal{O}(\varepsilon_{\text{mach}})$ and $\|\delta v\|_1 \le \|v\|_1 \mathcal{O}(\varepsilon_{\text{mach}})$.

Solution: By the fundamental theorem of floating point arithmetic, we know that for $1 \le i \le n$,

$$u_i \otimes v_i = u_i v_i (1+\delta_i), \qquad \text{where} \qquad |\delta_i| \le \varepsilon_{\text{mach}}.$$

Setting $s_i = u_i v_i (1+\delta_i)$, the floating point dot product is then given by

$$\hat{f}(u,v) = s_1 \oplus s_2 \oplus \cdots \oplus s_n.$$

Now apply the backward stability result for the summation of n floating point numbers that is written down in the lecture notes:

$$\hat{f}(u,v) = \sum_{i=1}^{n} s_i (1+\eta_i), \qquad \text{where} \qquad |\eta_i| \le n\varepsilon_{\text{mach}}.$$

Noting that $s_i(1-\eta_i) = u_i v_i (1+\delta_i)(1+\eta_i) = u_i v_i (1+\delta_i+\eta_i+\delta_i\eta_i)$, we conclude that

$$\hat{f}(u,v) = \sum_{i=1}^{n} [u_i + u_i(\delta_i+\eta_i+\delta_i\eta_i)]v_i = \sum_{i=1}^{n} u_i[v_i + v_i(\delta_i+\eta_i+\delta_i\eta_i)],$$

Now, denote $\delta u_i = u_i(\delta_i+\eta_i+\delta_i\eta_i)$ and $\delta v_i = v_i(\delta_i+\eta_i+\delta_i\eta_i)$ for $1 \le i \le n$, and apply the bounds for δ_i and η_i to conclude that

$$\|\delta u\|_1 = \sum_{i=1}^{n} |\delta u_i| \le \varepsilon_{\text{mach}}(n+1+n\varepsilon_{\text{mach}}) \sum_{i=1}^{n} |u_i| = \|u\|_1 \mathcal{O}(\varepsilon_{\text{mach}}).$$

This together with an analogous bound for $\|\delta v\|_1$ establishes both backward stability statements.

(b) Calculate the relative condition number of $f(u,v)$ with respect to its first input u in the 1-norm.

Solution: The relative 1-norm condition number of $f(u,v)$ with respect to the first input u is determined by the Jacobian of the linear map $g(u) = v^T u$. From equation (12.6) in the textbook, we have that

$$\kappa_f(u) = \|v^T\|_1 \frac{\|u\|_1}{|v^T u|}.$$

Note that the norm of the Jacobian here is the *matrix norm of the row vector* v_1^T, i.e.,

$$\|v^T\|_1 = \sup_{x\in\mathbb{R}^n} \frac{|v^T x|}{\|x\|_1} = \max_{1\le i\le n} |v_i|.$$

The second equality follows from equation (3.9) in the textbook (alternatively, see the example in the lecture notes about backward stability and condition numbers for summation). Substituting $\|v^T\|_1 = \max_{1\le i\le n}|v_i| = \|v\|_\infty$ into the expression for $\kappa_f(u)$, we conclude that

$$\kappa_f(u) = \frac{\|u\|_1\|v\|_\infty}{|v^T u|}.$$

(c) Using your results in parts (a) and (b), derive a bound on the forward error in $f(u,v)$. You should write the first order term explicitly (don't use "big-oh" notation).

Solution: By part (a), we know that $\hat{f}(u,v) = f(u+\delta u, v) = g(u+\delta u)$ where $\|\delta u\|_1 \le \|u\|_1\varepsilon_{\text{mach}}(n+1) + \mathcal{O}(\varepsilon_{\text{mach}}^2)$. Therefore, we have that

$$\frac{|\hat{f}(u,v) - f(u,v)|}{|f(u,v)|} = \frac{|g(u+\delta u) - g(u)|}{|g(u)|} \le \kappa_f(u)\frac{\|\delta u\|_1}{\|u\|_1}.$$

Plugging in the results for $\kappa_f(u)$ and the bound on $\|\delta u\|_1$ and simplifying, we obtain the forward error bound

$$\frac{|\hat{f}(u,v) - f(u,v)|}{|f(u,v)|} \le \frac{\|u\|_1\|v\|_\infty}{|v^T u|}\frac{\|\delta u\|_1}{\|u\|_1} = \frac{\|u\|_1\|v\|_\infty}{|v^T u|}(n+1)\varepsilon_{\text{mach}} + \mathcal{O}(\varepsilon_{\text{mach}}^2).$$

(d) Explain why computing the outer product $F(u,v) = uv^T$ is not backward stable.

Solution: A backward stable algorithm would have to compute a rank-one matrix of the form

$$\hat{F}(u,v) = F(u+\delta u, v+\delta v) = (u+\delta u)(v+\delta v)^T$$

with δu and δv sufficiently small. However, the fundamental theorem of floating point arithmetic implies that actual computed matrix $\hat{F} = \hat{F}(u,v)$ has entries

$$\hat{F}_{ij} = u_i v_j(1+\delta_{ij}), \qquad \text{where} \qquad |\delta_{ij}| \le \varepsilon_{\text{mach}} \qquad \text{for} \qquad 1\le i,j\le n.$$

In other words, we have $\hat{F} = uv^T + \Delta$,with $\|\Delta\| = \mathcal{O}(\varepsilon_{\text{mach}})$. But since each entry of Δ may very independently, in general, $\hat{F}$will **not** be a rank one matrix.

Problem 2: (9 + 9 + 9 + 6 points)

Consider a tall skinny matrix X with m rows and $n \ll m$ linearly independent columns. Let $X = QR$ be the reduced QR factorization of X and let $A = LL^T$ be the Cholesky factorization of the $n\times n$ symmetric positive definite matrix $A = X^T X$.

(a) Prove that the triangular factors in the two decompositions are equal, i.e., that $R = L^T$.

Solution: Since X has linearly independent columns, X has a unique reduced QR factorization, $X = QR$, with positive entries on the diagonal of R. Moreover, $A = X^T X$ is symmetric positive definite and has a unique Cholesky decomposition, $A = LL^T$. We calculate directly:

$$A = (QR)^T(QR) = R^T Q^T QR = R^T R.$$

Since R is upper triangular and has positive entries on the diagonal, this is a Cholesky decomposition of A. Since the Cholesky decomposition is unique, it is *the* Cholesky decomposition of A. Therefore, $R = L^T$.

(b) Using (a), describe an algorithm that computes both Q and R from the Cholesky factorization of A.

Solution: From part (a), we have $R = L^T$ and therefore $Q = XR^{-1} = XL^{-T}$. We suggest the following algorithm. Given X,

- Step 1) Compute the matrix product $A = X^T X$.
- Step 2) Compute the Cholesky factorization $A = LL^T$.
- Step 3) Solve the linear system $LQ^T = X^T$ (notice, this is a square linear system with m right-hand sides) for Q^T by forward substitution.

The outputs are $R = L^T$ and Q.

(c) How many flops does each step from your algorithm in (b) require? Include the cost of forming A and computing the Cholesky decomposition. You should write the leading order terms explicitly, i.e. $\sim 2n^2$, rather than $\mathcal{O}(n^2)$.

Solution: In Step 1), each entry of A requires computing a dot product between two vectors of length m, which costs $\sim 2m$ flops. There are n^2 entries in A, but because of symmetry we only need to compute $n(n+1)/2 \sim n^2/2$ entries in the lower triangular part. Therefore the cost of Step 1) is $\sim mn^2$ flops. In Step 2), the Cholesky factorization of A costs $\sim \frac{1}{3}n^3$ flops (see Lecture 23 in the textbook). And in Step 3), we solve m square linear systems of dimension n by forward substitution. Forward substitution has the same cost as back substitution, namely $\sim n^2$ flops (see Lecture 17 in the textbook), so the total cost of Step 3) is mn^2 flops. The total cost to leading order in n and m is therefore

$$\#\text{ flops} \sim 2mn^2 + \frac{1}{3}n^3.$$

Notice that when $m \gg n$, this is comparable to the flop count for both the Gram-Schmidt and Householder QR factorizations.

(d) Explain why the orthogonal factor $\hat{Q}$ computed by the algorithm in (b) in floating point may not be close to an orthogonal matrix when X is ill-conditioned.

Solution: The essence of the problem is that Step 3) applies the triangular transformation $R^{-1} = L^{-T}$ to orthogonalize the columns of X. This procedure can be unstable when X is ill-conditioned, as seen in the loss-of-orthogonality suffered by both classical and modified Gram-Schmidt methods. Furthermore, any errors in the computed triangular factor $\hat{R} = \hat{L}^T$ or in the subsequent triangular solve $\hat{L}Q^T = X^T$ can lead to the potential loss of orthogonality in the columns of the computed $\hat{Q}$. We can make a quantitative argument as follows.

- According to the textbook (Lecture 23), the computed Cholesky factor may be innacurate when A is ill-conditioned, with $\|\hat{L} - L\| = \|L\|\mathcal{O}(\kappa(A)\varepsilon_{\text{mach}})$. The singular values of $A = X^T X$ are just the singular values of X squared, so $\kappa(A) = \kappa(X)^2$. Therefore, we can write the triangular factor computed in Step 2) as $\hat{L} = L + E$,with

$$\|E\| = \|\hat{L} - L\| = \|L\|\mathcal{O}(\kappa(X)^2\varepsilon_{\text{mach}}).$$

- In Step 3) we solve $(L+E)\hat{Q}^T = X^T$. If we write $\hat{Q} = Q + \Delta Q$, then we can make the first-order approximation

$$X^T = (L+E)(Q+\Delta Q)^T \approx LQ^T + EQ^T + L\Delta Q^T$$

Solving for ΔQ^T, taking norms, and using the fact that $Q^T = L^{-1}X^T$ we obtain a first-order approximation for the error in the computed orthogonal factor:

$$\|\Delta Q^T\| \approx \| - Q^T + L^{-1}(X^T + EQ^T)\| = \|L^{-1}EQ^T\| \leq \|L^{-1}\|\|L\|\mathcal{O}(\kappa(X)^2\varepsilon_{\text{mach}}).$$

Since $\|L^{-1}\|\|L\| = \kappa(L) = \kappa(X)$, the last factor is proportional to $\mathcal{O}(\kappa(X)^3\varepsilon_{\text{mach}})$. Then, to first-order in $\varepsilon_{\text{mach}}$, we have that

$$\|\hat{Q}^T\hat{Q} - I\| \approx \|\Delta Q^T Q + Q^T \Delta Q\| = \mathcal{O}(\kappa(X)^3\varepsilon_{\text{mach}}).$$

This argument produces only a first-order *estimate* of the *worst case* error; it does not produce a rigorous upper or lower bound on the loss of orthogonality. For example, we have neglected the errors from solving the linear system in floating point (which amounts to dropping a term proportional to $\kappa(X)\varepsilon_{\text{mach}}$), as well as the errors commited while computing A in Step 1). So what do we see in practice? Check out the attached iJulia notebook to see some numerical experiments and related commentary.

Remark: Although Cholesky-based QR appears unstable, it has excellent communication costs. Remarkably, the instability explored in (d) can be overcome by, essentially, iterating the algorithm three times. The result is a state-of-the-art communication-minimizing algorithm! For more, see the paper on shifted Cholesky QR at https://epubs.siam.org/doi/10.1137/18M1218212.

Problem 3: (10 + 18 + 6 points)

In analogy to the QR algorithm for the standard eigenvalue problem $Av = \lambda v$, the QZ algorithm is used to compute generalized eigenvalues of the generalized eigenvalue problem $Av = \lambda Bv$, where A and B are $n \times n$ matrices. Like QR, it proceeds in two stages: use orthogonal transformations (e.g., Householder transformations) applied from the left and right to (i) reduce A and B to upper Hessenberg form and upper triangular form, and (ii) perform QZ iterations that drive the subdiagonal entries of the upper Hessenberg matrix A to zero. In this problem, you will work through stage (i) of the QZ algorithm.

(a) Apply a sequence of Householder transformations on the left of A and B that triangularize B.

Solution: See the attached iJulia notebook.

(b) Apply a sequence of Householder transformations on the left and right of A and B that make A upper Hessenberg and keep B upper triangular. (Hint: use transformations from the left to introduce zeros in A and from the right to keep B upper triangular.)

Solution: See the attached iJulia notebook.

(c) Is stage (i) backward stable? Give a brief explanation (one or two sentences).

Solution: Yes, stage (i) is backward stable. This is because only orthogonal transformations are applied to the inputs A and B to obtain the, respectively, upper Hessenberg and triangular outputs. In homework 2, we saw that applying a product of orthogonal transformations to a vector is backward stable, and it is straightforward to apply this result to the columns and rows of A and B to show that stage (i) is backward stable.

Your answer should include diagrams that show the pattern of zeros and nonzeros after each Householder transformation is applied, as well as Julia code to execute stage (i). You may use Julia code from the notes folder on the course page to compute and apply Householder reflectors.