18.335 Midterm Solutions, Fall 2012

Problem 1: (25 points)

Note that your solutions in this problem don't require you to know how sin, ln, and Γ are calculated on a computer, because the answers rely on properties of the *functions* (and of floating-point arithmetic in general, of course), not of the algorithms to compute the functions. (Contrary to what many students assumed, Taylor series are *not* the only way to compute special functions like this, nor are they usually the best way except in limiting cases, nor should you generally use the *same* Taylor series for all x.)

(a) The condition number of $f(x) = \sin(x)$ is $\kappa(x) = \left|\frac{f'(x)}{f(x)/x}\right| = \left|x\frac{\cos x}{\sin x}\right|$. As $x \to 0$, $\kappa(x) \to 1$ (since $\frac{\sin x}{x} \to 1$), so it is well conditioned near $x = 0$ and we should expect an accurate answer is possible even if there is a small (relative) rounding error in x. In particular, the Taylor expansion of $\sin x$ near $x = 0$ clearly becomes more and more accurate as $x \to 0$, in which limit $\sin x \approx x$ and the function $f(x) = x$ can obviously be computed accurately (with the forward error approaching the relative error in x). On the other hand, $x \to 2\pi$, $\frac{\sin x}{x} \to 0$ and hence $\kappa(x) \to \infty$: the problem is ill-conditioned near 2π and a small forward error *may* not be possible, depending upon how we define the problem.

In particular, if x contains a small relative error, e.g. because it was rounded from a non-representable real number, then we should not expect a small forward error near 2π: the large condition number means that a small error in x produces a large error in $\sin x$. For example, for $x = 2\pi + \delta$ with $\delta \sim \varepsilon_{\text{machine}}$, a roundoff error to $\tilde{x} = 2\pi + \delta + \varepsilon_{\text{machine}}$ will roughly double the magnitude of $\sin(x)$, giving a relative error of order 1.

If the input x is exactly computed in floating point, on the other hand then it is possible to compute an accurate answer. Suppose that we computed $\sin(x)$ near $x = 2\pi$ by first computing $y = x - 2\pi$ and then computing $\sin y$. If we naively computed y by $y = x \ominus \text{fl}(2\pi)$, we could easily get a large cancellation error in computing y since 2π is not exactly representable. However, if we instead computed $\text{fl}(x - 2\pi) = (x - 2\pi)[1 + O(\varepsilon_{\text{machine}})]$, e.g. by performing the subtraction in a higher precision, then we could obtain a small forward error in $\sin y = \sin x$.

(b) For $|x| < \varepsilon_{\text{machine}}$, $1 + x$ will be rounded to 1 and hence `log(1+x)` would give 0 (a relative error of 1 for $x \neq 0$!). Therefore, we need a specialzed `log1p(x)` function if we wish to compute $\ln(1 + x)$ accurately for small $|x|$.

Equivalently, the function $\ln(y)$ has a condition number $\left|\frac{1/y}{\ln(y)/y}\right|$ that diverges as $y \to 1$, making it extraordinarily sensitive to rounding errors in computing the argument $y = 1 + x$, while the function $f(x) = \ln(1 + x)$ has condition number $\left|\frac{1/(1+x)}{\ln(1+x)/x}\right| \to 1$ as $x \to 0$.

A possible implementation might use the Taylor expansion $\ln(1 + x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + O(x^5)$ for small x, and compute $\ln(1 + x)$ directly for larger x. e.g.

$$\text{log1p}(x) = \begin{cases} x\left(1 - x\left(\frac{1}{2} - x\left(\frac{1}{3} - \frac{x}{4}\right)\right)\right) & |x| < 10^{-3} \\ \log(1 + x) & \text{otherwise} \end{cases},$$

(where for extra niceness I evaluated $x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4}$ by Horner's method). This four-term Taylor series should be accurate to machine precision for $|x| < 10^{-3}$.

[The problem is *not* points near (slightly bigger than) $x = -1$. If you want to compute $\ln(1 + x)$ for such x, there is no way around the fact that you need to know $1 + x$ accurately to know how close the argument of the log is to zero, and cancellation errors will force you to lose a lot of significant digits in finding $1 + x$ if x is not exactly representable. A specialized log1p function won't help. Note also that if $\text{fl}(x) > -1$, we will obtain $1 \oplus x > 0$ in exactly rounded floating-point arithmetic, so rounding won't change the domain of the function.]

(c) The problem in this case is not roundoff errors, but overflow. Remember that floating-point uses a fixed number of digits for its exponent, so it cannot represent arbitrarily large numbers. (In double precision, the maximum magnitude is $\approx 10^{308}$). The factorial function, and hence the $\Gamma(x)$ function, grows faster than exponentially with x, so for $x \gtrsim 172$ it will overflow and simply give ∞. By defining a separate `gammaln(x)` function, Matlab allows you to study the magnitude of the Γ function for much larger x (up to $x \approx 10^{305}$).

Note that, if it weren't for overflow, there wouldn't necessarily be any severe accuracy problem with computing $\ln\Gamma(x)$ by computing $\Gamma(x)$ first. $\ln(x)$ is well-conditioned for large x. The condition number of $\Gamma(x)$ does grow with x, but only relatively slowly ($\approx x\ln x$), so it overflows long before it becomes badly conditioned.

Problem 2: (5+10+10 points)

(a) A simple example would be $\|(x,y)\|_+ = \|x\| + \|y\|$. Another would be $\|(x,y)\|_{\max} = \max(\|x\|, \|y\|)$. More examples are $\|(x,y)\|_p = \sqrt[p]{\|x\|^p + \|y\|^p}$ for any $p \geq 1$. All of these clearly satisfy the positivity, scaling, and triangle properties of norms, inheriting those properties from the norms on x and y (combined with the same properties of the L_p norm).

(b) Second $\Longrightarrow$ First, but not the other way around. That is, the Second definition is a stronger requirement on $\tilde{f}$. [Note that, from class, equivalence of norms means that we only need to prove this for one choice of $\|(x,y)\|$ and it follows for all other choices of norm.]

Suppose that $\tilde{f}$ is backwards stable in the Second sense. Then, using e.g. $\|(x,y)\|_{\max}$ from above, we have $\|\tilde{x} - x\| = \|x\| O(\varepsilon_{\text{machine}}) \leq \|(x,y)\|_{\max} O(\varepsilon_{\text{machine}})$ and $\|\tilde{y} - y\| = \|y\| O(\varepsilon_{\text{machine}}) \leq \|(x,y)\|_{\max} O(\varepsilon_{\text{machine}})$. Hence $\|(\tilde{x},\tilde{y}) - (x,y)\|_{\max} = \max(\|\tilde{x}-x\|, \|\tilde{y}-y\|) = \|(x,y)\|_{\max} O(\varepsilon_{\text{machine}})$, and First follows.

The converse is not true, essentially because we can have $\|x\|$ arbitrarily small compared to $\|(x,y)\|$ by choosing $\|x\| \ll \|y\|$ (or vice versa). From $\|\tilde{x} - x\| \leq \|(\tilde{x},\tilde{y}) - (x,y)\|_{\max} = \|(x,y)\| O(\varepsilon_{\text{machine}})$, we obtain $\|\tilde{x} - x\| = \frac{\|(x,y)\|}{\|x\|} \|x\| O(\varepsilon_{\text{machine}})$. However, it does not follow that $\|\tilde{x} - x\| = \|x\| O(\varepsilon_{\text{machine}})$, because the prefactor $\frac{\|(x,y)\|}{\|x\|}$ can be arbitrarily large, and we required the constants in $O(\varepsilon_{\text{machine}})$ to be independent of x (uniform convergence).

More explicitly, let us construct a counterexample (*not required*). Consider $f(x,A) = bx^* + A$ for $x \in \mathbb{C}^n$, $A \in \mathbb{C}^{n\times n}$, and some fixed $b \in \mathbb{C}^n$. It is straightforward to show that this is backwards-stable in the First sense, by letting $\tilde{x} = x$ and $\tilde{A} = \tilde{f}(x,A) - bx^*$. i.e. $\tilde{A}_{ij} = (b_i \otimes x_j \oplus A_{ij}) - b_i x_j = [b_i x_j(1+\varepsilon_1) + A_{ij}](1+\varepsilon_2) - b_i x_j = A_{ij} + b_i x_j[\varepsilon_1 + O(\varepsilon_{\text{machine}}^2)] + A_{ij}\varepsilon_2$, where $|\varepsilon_{1,2}| \leq \varepsilon_{\text{machine}}$. Hence $|\tilde{A}_{ij} - A_{ij}| \leq (\|x\|_\infty + \|A\|_\infty) O(\varepsilon_{\text{machine}})$ (where $\|A\|_\infty = \max_{i,j} |A_{ij}|$) and we have $\|\tilde{A} - A\|_\infty = \|(x,A)\|_+ O(\varepsilon_{\text{machine}})$. Hence it is backwards stable in the First sense. On the other hand, it is not backwards stable in the Second sense. Consider inputs $A = 0$, in which case $f(\tilde{x},A)$ is rank 1 for any $\tilde{x}$, but $\tilde{f}(x,A)$ will not be rank 1 due to roundoff errors [similar to pset 2 problem 4(b)(ii)], and hence we *must* have $\tilde{A} \neq A$ in order to have $f(\tilde{x},\tilde{A}) = \tilde{f}(x,A)$. But then $\|\tilde{A} - \cancelto{0}{A}\| = \|\tilde{A}\| > \cancelto{0}{\|A\|} O(\varepsilon_{\text{machine}})$, and therefore it cannot satisfy the Second definition.

(c) Choose $\|(x,y)\| = \|x\|_1 + \|y\|_1$, in which case both the norm and the algorithm $\tilde{f}(x,y)$ are exactly equivalent to the summation studied and proved backwards stable in class, applied to a column vector $\begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^{m+n}$. So, it is stable in the First sense.

In fact, it is stable in the Second sense as well! Since it is stable in the First sense, construct $(\tilde{x},\tilde{y})$ with $f(\tilde{x},\tilde{y}) = \tilde{f}(x,y)$ and $\|(\tilde{x},\tilde{y})-(x,y)\|_\infty = \|(x,y)\|_\infty O(\varepsilon_{\text{machine}})$ for $\|(x,y)\|_\infty = \max(\|x\|_\infty, \|y\|_\infty)$. Suppose $\|x\|_\infty \geq \|y\|_\infty$, then it follows that $\|\tilde{x}-x\|_\infty \leq \|(\tilde{x},\tilde{y})-(x,y)\|_\infty = \|x\|_\infty O(\varepsilon_{\text{machine}})$, and we only need to prove the corresponding property for $\tilde{y}-y$, but unfortunately this is not true if $\|y\|_\infty \ll \|x\|_\infty$. Instead, let us construct a new pair $(\tilde{x}',\tilde{y}')$ with $f(\tilde{x}',\tilde{y}') = f(\tilde{x},\tilde{y}) = \tilde{f}(x,y)$ by setting $\tilde{y}' = y$, and $\tilde{x}'_i = \tilde{x}_i + \frac{\sum(\tilde{y}_k - y_k)}{m}$ for $i = 1,\ldots,m$—that is, we have pushed all of the $\tilde{y}-y$ differences into $\tilde{x}'$, while keeping the sum the same. Then $\|\tilde{y}'-y\| = 0 = \|y\|O(\varepsilon_{\text{machine}})$ and $\|\tilde{x}'-x\|_\infty \leq \|\tilde{x}-x\|_\infty + \left|\frac{\sum(\tilde{y}_k - y_k)}{m}\right| \leq \|\tilde{x}-x\|_\infty + \|\tilde{y}-y\|_\infty \leq 2\|(\tilde{x},\tilde{y})-(x,y)\|_\infty = \|x\|_\infty O(\varepsilon_{\text{machine}})$. Similarly if $\|x\|_\infty \leq \|y\|_\infty$, except that we push all the $\tilde{x}-x$ differences into $\tilde{y}'$. Hence it is backwards stable in the Second sense.

You could also use the analysis from pset 2 (or similar) to explicitly construct $\tilde{x}$ and $\tilde{y}$ and thereby prove stability in the Second (hence First) sense.

Problem 3: (25 points)

First, let us follow the hint and show that $q_k = Q^{(n)}e_k$ is in the span $\langle x_1, x_2, \ldots, x_k\rangle$ as $n \to \infty$. We will proceed by induction on k. Let

$$v_k = A^n e_k = \sum_{i=1}^{m} c_i \lambda_i^n x_i,$$

where we have expanded $e_k = \sum c_i x_i$ in the basis of the eigenvectors; we can generically assume that $c_i \neq 0$ for all i, so that v_k is dominated as $n \to \infty$ by the terms with the biggest $|\lambda|$.

- For $k = 1$, $q_1 = v_1/\|v_1\|_2$ (via Gram-Schmidt), and since $v_1 \approx c_1\lambda_1^n x_1$ as $n \to \infty$ we have $q_1 \to x_1$.
- Suppose $q_i \in \langle x_1,\ldots,x_i\rangle$ for $i < k$, and prove for k. For large n,

  $$v_k \approx \sum_{i \leq k} c_i \lambda_i^n x_i \in \langle x_1,\ldots,x_k\rangle,$$

  where we have discarded the $i > k$ terms as negligible. We obtain q_k from v_k by Gram-Schmidt:

  $$q_k = \frac{v_k - \sum_{i<k} q_i q_i^* v_k}{\|\cdots\|_2}.$$

  However, since all of the terms in the numerator are $\in \langle x_1,\ldots,x_k\rangle$, the result follows. (Note that the orthonormality of the q's means that q_k must contain a nonnegligible x_k component, as otherwise it would be in the span of the q_i for $i < k$.)

  It is instructive (but not strictly necessary!) to look at this more carefully. Since the q_i for $i < k$, being independent, necessarily form a *basis* for the $(k-1)$ subspace $\langle x_1,\ldots,x_{k-1}\rangle$, it follows that $(I - \sum_{i<k} q_i q_i^*)x_j = 0$ for $j < k$ (since we are projecting orthogonal to the whole $\langle x_1,\ldots,x_k\rangle$ subspace). Hence,

  $$q_k \approx \frac{c_k\lambda_k^n [x_k - \sum_{i<k} q_i q_i^* x_k]}{\|\cdots\|_2} \in \langle x_1,\ldots,x_k\rangle.$$

  So, like in class, q_k still picks up contributions only from the λ_k^n term in v_k, as all of the larger $|\lambda|$ terms are cancelled by the projection. (At least, in exact arithmetic, but fortunately the QR iteration gives us the same result without the ill-conditioning.) Unlike the Hermitian case in class, however, $q_i^* x_k \neq 0$ in general, so q_k generally has nonzero x_i components for $i < k$.

Now that we have proven this fact, the result is easy. Since $q_k \in \langle x_1,\ldots,x_k\rangle$, it immediately follows that $Aq_k \in \langle x_1,\ldots,x_k\rangle = \langle q_1,\ldots q_k\rangle$, and thus $T_{ij} = q_i^* A q_k = 0$ for $i > k$. Hence $T = Q^*AQ$ is upper triangular, and we have a Schur factorization of A.