Homework 5

Math 651
Fall 2019

Due Monday, October 21, 2019

Problem 1 (1+2+1+2 points)**.**

1. *Suppose that for some reason you wish to compute* $\exp(-20)$*. Formulate this task precisely as a numerical analysis problem. That is, specify a space* X *of data, a space* Y *of solutions, and a problem mapping* $f : X \to Y$*. What are the relative and absolute condition numbers of the problem? Would you say that the problem is well-conditioned?*

2. *Define*
$$S(x, n) := \sum_{k=0}^{n} \frac{x^k}{k!}.$$
Recall that $\lim_{n\to\infty} S(x, n) = \exp(x)$*. Compute* $S(-20, n)$ *in in floating point arithmetic for* $n = 1, \ldots, 100$*. Plot the logarithm of the relative error*
$$\log_{10}\left(\frac{|S(-20, n) - \exp(-20)|}{\exp(-20)}\right)$$
against n*.*

3. *You should have observed above that* $S(x, n)$ *does not converge to* $\exp(-20)$ *when calculated in floating point arithmetic. Why not?*

4. *Now plot*
$$\log_{10}\left(\frac{|1/S(20, n) - \exp(-20)|}{\exp(-20)}\right)$$
against n*.*

Computing finite difference approximations to derivatives in floating point arithmetic is notoriously unstable. In the following problem, you will show that the errors of a naive approximation to the derivative cannot decrease faster than $\sqrt{\varepsilon_m}$ as ε_m tends to 0.

Problem 2 (2+2+2+2 points)**.**

1. *Compute*

$$d(h) := \frac{exp(h) - exp(0)}{h}$$

for $h \in \{2^{-k} : k = 0, \ldots, 60\}$. *Now do the following:*

- *Plot* $-\log_{10}(h)$ *vs.* $|d(h) - 1|$.
- *Plot* $-\log_{10}(h)$ *vs.* $\log_{10}(|d(h) - 1|)$. *(Most likely, when you generate this plot, numpy.log*10 *will return an error message. Make sure to understand and fix the error.)*
- *Compute and report*

$$\min_{h \in \{2^{-k} : k = 0, \ldots, 60\}} |d(h) - 1|.$$

2. *Let* $f : \mathbb{R} \to \mathbb{R}$. *Assume for convenience that* f *is bounded. Let* $x, h \in \mathbb{F}$, *and let* ε_m *denote machine precision. Assume that* $\varepsilon_m \leq 1$. *Show that*

$$\left| (fl(f(x+h)) \ominus fl(f(x))) \oslash h - \frac{f(x+h) - f(x)}{h} \right| \leq \frac{C\varepsilon_m \|f\|_\infty}{|h|}$$

for some constant $C > 0$ *that does not depend on* ε_m *or* f. Hint: *Use the fundamental axiom of floating point. It is probably easiest to do a forwards analysis instead of a backwards analysis.*

3. *Now let* $f \in C^2$. *That is, assume that* f *is twice continuously differentiable and that* f, f', *and* f'' *are bounded. Show that*

$$\left| \frac{f(x+h) - f(x)}{h} - f'(x) \right| \leq |h| \|f''\|_\infty.$$

.

4. *Combining the results above, we see that*

$$|(fl(f(x+h)) \ominus fl(f(x))) \oslash h - f'(x)| \leq \frac{C\varepsilon_m \|f\|_\infty}{|h|} + |h| \|f''\|_\infty.$$

Define

$$E(\varepsilon_m, h) := \frac{C\varepsilon_m \|f\|_\infty}{|h|} + |h| \|f''\|_\infty.$$

Now let $h(\varepsilon_m)$ *be the value of* h *that minimizes* $E(\varepsilon_m, h)$ *for fixed* ε_m. *That is, define*

$$h(\varepsilon_m) := \operatorname*{argmin}_h E(\varepsilon_m, h).$$

Show that $E(\varepsilon_m, h(\varepsilon_m)) = O(\sqrt{\varepsilon_m})$ *as* $\varepsilon_m \to 0$, *so even in the best case the error of the finite difference approximation decreases only like* $\sqrt{\varepsilon_m}$ *with machine precision. Do the computations in part 1 support this theoretical result?*

Problem 3 (1+1 points)**.**

1. *Compute a random matrix with known QR factorization using the following python code:*

```
import numpy as np
np.random.seed(123)
R=np.triu(np.random.randn(50,50))
Q,foo=np.linalg.qr(np.random.randn(50,50))
A=np.dot(Q,R)
```

   Now compute the QR factorization of the matrix A by Householder orthogonalization using the code

```
Q2,R2=np.linalg.qr(A).
```

   Of course, the computed $Q2$ and $R2$ are not the same as the exact Q and R, but the size of the difference may surprise you. Compute the relative errors in the ℓ^∞ operator norm using the code

```
infty_err=lambda A,B: np.linalg.norm(A-B,ord=np.inf)/np.linalg.norm(B,ord=np.inf)
err_Q=infty_err(Q2,Q)
err_R=infty_err(R2,R)
err_A=infty_err(np.dot(Q2,R2),A)
```

   (The function np.linalg.norm computes the operator norm using exactly the formula that you proved on the homework.) Report the values of err_Q, err_R, and err_A. How is it possible that err_A is roughly machine precision, but the other two are many orders of magnitude larger?

2. *Use google to look up how to compute the SVD in numpy. (For any reasonable query, the first search result will be on the scipy documentation website, and this is what you want.) Compute the SVD of A, and use the computed SVD to estimate the condition number*

$$\kappa_2(A) = \|A\|_2 \|A^{-1}\|_2.$$

   Report the value of the condition number.