Homework 4

Please answer the following questions in complete sentences in a clearly prepared manuscript and submit the solution by the due date on Gradescope (Due morning of Nov 6.)

Remember that this is a graduate class. There may be elements of the problem statements that require you to fill in appropriate assumptions. You are also responsible for determining what evidence to include. An answer alone is rarely sufficient, but neither is an overly verbose description required. Use your judgement to focus your discussion on the most interesting pieces. The answer to "should I include 'something' in my solution?" will almost always be: Yes, if you think it helps support your answer.

Problem 0: Homework checklist

- Please identify anyone, whether or not they are in the class, with whom you discussed your homework. This problem is worth 1 point, but on a multiplicative scale.
- Make sure you have included your source-code and prepared your solution according to the most recent Piazza note on homework submissions.

Problem 1: Flop counts

1. Let $\boldsymbol{H} = \boldsymbol{I} - 2\frac{\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}}$ be a Householder matrix. Starting from the vector $\mathbf{v}$ and matrix $\boldsymbol{A}$. Show how many addition operations it takes to compute $\boldsymbol{H}\boldsymbol{A}$, along with subtraction and multiplication operations. Assume that $\boldsymbol{H}$ is $n \times n$ and $\boldsymbol{A}$ is $n \times c$. You may lose points if your count is not $O(nc)$.
2. Consider computing trace($\boldsymbol{A}^T\boldsymbol{B}$) where $\boldsymbol{A}$ and $\boldsymbol{B}$ are in compressed sparse column format and have the same dimensions. Suppose we have access to a fused multiply add operation. This is an operation that takes three inputs α, γ, β and computes
   $\theta \leftarrow \alpha * \gamma + \beta$ in a single operation. Explain how to use this operation when computing the trace and compute how much such operations you need along with any other floating point operations (additions, multiplications, divisions, etc.)

Problem 2: Inner-products are backwards stable.

1. Find a proof that computing $\mathbf{x}^T\mathbf{y}$ is backwards stable. Explain this proof in enough detail for a classmate to understand it without having read the document. This could take up to a page to give enough detail.
2. Show that computing a matrix-vector product $\mathbf{y} = \boldsymbol{A}\mathbf{x}$ is backwards stable.

Problem 3: Accurate summation

Consider a list of n numbers. For simplicity, assume that all numbers are positive so you don't have to write a lot of absolute values.

1. Show that the following algorithm is backwards stable.

```
function mysum(x::Vector{Float64})
  s = zero(Float64)
  for i=1:length(x)
    s += x[i]
  end
  return s
end
```

Which requires showing that $\text{mysum}(\mathbf{x}) = \sum \hat{x}_i$ where $\|\hat{\mathbf{x}} - \mathbf{x}\|/\|\mathbf{x}\| \leq C_n \varepsilon$ where ε is the unit-roundoff for Float64. (You may want to solve problem 2 first.)

2. Consider adding three positive numbers together a, b, c. Describe how to compute $s = a + b + c$ with the greatest accuracy.

3. Use the results of part 2 to describe a way to permute the input $\mathbf{x}$ to `mysum` to attain the greatest accuracy. Find an input vector $\mathbf{x}$ where this new ordering gives a measurable change in the floating point accuracy as determined by the number of correct digits in the mantissa. (Hint, this means you should know the true sum of your vector so that you can identify it's best floating point representation.)

4. Lookup the Kahan summation algorithm and implement it to sum a vector. Compare the accuracy with what you found in part 3.

Problem 4: Quadratic equations

Read through the stack exchange post on solving quadratic equations. https://math.stackexchange.com/questions/311382/solving-a-quadratic-equation-with-precision-when-using-floating-point-variables

This suggests a number of approaches to compute the roots of a quadratic equation through closed form solutions.

An alternative approach is to use an iterative algorithm to estimate that root of an equation. In this case, we can use a simple bisection approach, which works quite nicely for finding the root.

Your task for this problem is to implement a bisection algorithm to return all the solutions of $ax^2 + bx + c = 0$ when $c \neq 0$.

```
""" Return all the solutions to ax^2 + bx + c. It is acceptable to return
NaN instead of a root as well. """
function roots(a::Float32,b::Float32,c::Float32)
end
```

The input to this method is Float32 so you can compare to higher-accuracy solutions with Float64 and to elucidate some of the issues that arise with slightly lower-precision.

Compare the accuracy of this procedure to the methods suggested on the stack exchange page and explain your results. Note that you may need to look for extremal inputs. In this case, Float32 is handy because there are only 4 billion inputs for each input value a, b, c. This is still too many to test all combinations. But there are only two choices for the roots, which greatly reduces the space.

Problem 5: The advantages of Float64

Consider the Chutes and Ladders problem from HW2 where we worked out the expected length of a game based on an infinite summation. Repeat this analysis with Float16 arithmetic and also with BigFloat analysis. (This problem may require julia to use both Float16 and BigFloat.) Make sure that all intermediate computations use these types. To declare a vector of Float16 or BigFloats,

use `Vector{Float16}` or `Matrix{BigFloat}` also helpful are `zero(Float16)`, `one(Float16)`. If there are questions about using these types, please post to Piazza. Which answer is more accurate?

Problem 6: Condition numbers

Consider the following computations. Discuss if they are well-conditioned or ill-conditioned. If the answer depends on the types of input, please provide some rough guidance. (e.g. for subtraction, it's ill-conditioned if the numbers are close by)

1. The entropy of a probability distribution is $H(\mathbf{p}) = -\sum_{i=1}^{n} p_i \log p_i$ where $0 < p_i < 1$. Compute the condition number of the entropy function.
2. A matrix vector product $\mathbf{y} = \boldsymbol{A}^T\mathbf{x}$.
3. Evaluating a neural network layer $\mathbf{y} = f(\boldsymbol{W}^T\mathbf{x})$ where the elements are $y_i = f(w_i^T x)$ and f the soft-plus function $\log(1 + e^x)$.

Problem 7: Experience with the SVD

Produce the analytic SVDs of the following matrices. (That is, no numerical approximations, but feel free to let Julia give you a good guess!). It's important to think about these questions because I may give similar questions on a midterm or final and you'll be expected to remember these. It's also handy to think about how to construct these, even though we haven't seen any algorithms yet. You should be able to work them all out directly from the definition.

1. $\begin{bmatrix} 0 & -3 \\ 0 & 0 \end{bmatrix}$
2. $\begin{bmatrix} -5 & 0 \\ 2 & 0 \end{bmatrix}$
3. $\begin{bmatrix} 1 & -2 \\ 2 & -4 \\ 0 & 0 \end{bmatrix}$
4. $\begin{bmatrix} 2 & 0 \\ 0 & 5 \end{bmatrix}$

Problem 8: Backwards stability

1. Let $f(x) = \sqrt{x}$. Suppose you have an algorithm where $\texttt{myf}(x) = \sqrt{x + \mu}$ where μ is the machine precision. Is $\texttt{myf}(x)$ backwards stable?
2. Suppose that you have a fancy implementation of $\sqrt{x}$ and you compute $\texttt{mysqrt}(0.1\mu) = -1 \cdot 10^{-16}\mu$. Is this a backwards stable implementation?

Problem 9: Condition numbers

Show that

$$\frac{1}{\kappa(\boldsymbol{A})}$$

measures the relative distance from $\boldsymbol{A}$ to the space of singular matrices. That is

$$\frac{1}{\kappa(\boldsymbol{A})} = \text{ smallest } \frac{\|\boldsymbol{D}\|}{\|\boldsymbol{A}\|} \text{ such that } \boldsymbol{A} + \boldsymbol{D} \text{ is singular.}$$

Everything here involves the matrix 2-norm.

Optional Problems

The following problem is optional, but highly recommended.

Problem X - Fun with the SVD

Need to think of a fun problem here.