M.Sc. in High-Performance Computing
5633A - Numerical Methods for High-performance Computing
Programming Assignment 2

Kirk M. Soodhalter (ksoodha@maths.tcd.ie)
School of Mathematics, TCD

Rules

To submit, make a single tar-ball with all your code and a pdf of any written part you want to include. Submit this via Microsoft Teams by 30. October, 2020. Late submissions without prior arrangement or a valid explanation will result in reduced marks.

Question

On Pages 51 and 58 of *Numerical Linear Algebra* by Trefethen and Bau (accompanying this homework sheet), two algorithms for orthogonalizing vectors are given. Algorithm 7.1 on page 51 is the version one learns in a Linear Algebra course, the Gram-Schmidt Algorithm. This algorithm is not stable. Algorithm 8.1 on page 58 is the modified version, which is mathematically equivalent (up to possible sign differences) to Algorithm 7.1 but is *backwards stable*.

1. Implement both algorithms in Matlab as functions

$$[\mathtt{Q},\mathtt{R}] = \mathtt{gs}(\mathtt{A})$$
$$[\mathtt{Q},\mathtt{R}] = \mathtt{mgs}(\mathtt{A}).$$

   The functions should orthonormalize the columns of $A \in \mathbb{R}^{m\times n}$ (assume $m \geq n$ for this assignment), store the orthonormalized columns in the matrix `Q`, and store the orthogonalization coefficients in the upper triangular matrix `R`.

2. Let us construct a matrix whose columns are structured such that they become increasingly close to being linearly dependent. We do this by constructing a matrix

with decaying singular values[1]. Construct a matrix as follows

$$
\begin{aligned}
&[\mathtt{U}, \sim] = \mathtt{qr}(\mathtt{rand}(100));\\
&[\mathtt{V}, \sim] = \mathtt{qr}(\mathtt{rand}(100));\\
&\mathtt{S} = \mathtt{diag}(2\,.\widehat{}\,(-1:-1:-100));\\
&\mathtt{A} = \mathtt{U} * \mathtt{S} * \mathtt{V};.
\end{aligned}
$$

We should expect the norm of each new vector (stored in the diagonal of $\mathbf{R}$) created by the Gram-Schmidt algorithm will decay. In one figure, use the `semilogy()` plotting function to plot the diagonals of $\mathbf{R}$ as outputted by both. In addition, plot horizontal lines at $\epsilon_{machine}$ and $\sqrt{\epsilon_{machine}}$. Describe what this figure shows.

3. Experimenting with random matrices and collecting statistics is not the same as making a mathematical proof, but it can be helpful to illustrate a point. This is an exercise in such an experiment. Using the same $\mathbf{U}$ and $\mathbf{V}$, consider generating $\mathbf{S}$ randomly as follows

$$\mathtt{S} = \mathtt{diag}(2\,.\widehat{}\,(2 * \mathtt{randn}(100, 1)));.$$

For $10,000$ randomly generated matrices $\mathbf{S}$, run both versions of your Gram-Schmidt code on the resulting matrices. For each experiment, store the forward error quantity `norm(A - Q*R)` and the backward error quantity `norm(Q'*Q - eye(100))` in vectors. Plot the forward errors and backward errors in different figures as points using different colors to differentiate the two experiments. Explain what you see in this figure.

[1] We will learn the details about the singular value decomposition in a few lectures.