Getting Setup

For this project I **highly recommend** using a language that has some built in matrix and vector operations. You will want to easily be able to compute dot products, matrix multiply and vector 2-norm and build matrices with elements coming from a random normal distribution. (Do not reinvent the wheel here!) In addition, you will have to be able to produce plots with a logarithmic scale on the y-axis. It will also be helpful if the language already has a built in QR factorization. For these reasons I suggest using Matlab, Octave or Julia.

- For Matlab you can install a version of Matlab from the campus Sharepoint. See the link on D2L for instructions. I will not be able to troubleshoot any issues you might have with installation but ITS should be able to. Note: if you want to use Matlab while you are off campus you will need to login to the campus VPN first. (https://www.sdsmt.edu/its/help/Connect-to-SDSMT-Network-Through-VPN/)
- Octave is free and very much like Matlab. It is available to download at https://www.gnu.org/software/octave/index.
- Julia can be installed as the engine for a Jupyter notebook. I vaguely know how to do this but you'd probably need to bring your laptop to my office for me to help you get it going. It is also very compatible with Matlab.

The project

- Following Algorithms 7.1 and 8.1 from Trefethen and Bau, write two separate functions to perform Classical Gram Schmidt and Modified Gram Schmidt. Each function will accept an $m \times n$ $(m \geq n)$ matrix A and return an $m \times n$ matrix Q with orthogonal columns that span the same space as the columns of A and an $n \times n$ upper triangular matrix R such that $A = QR$.
- Verify that your functions are working by doing the following steps. These will work fine if you copy and paste them into Matlab or Octave (and maybe Julia) using, of course, whatever names you used for your functions.

```
A = randn(4,3)
[QC,RC] = mycgs(A)
[QM,RM] = mymgs(A)
norm(A - QC*RC)
norm(A - QM*RM)
norm(eye(3,3) - QC'*QC)
norm(eye(3,3) - QM'*QM)
```

The first 2 norms are a measure of whether A is factored correctly. The second 2 are a measure of whether the Q matrices have orthogonal columns. None of them will be exactly zero but all should be close to machine epsilon, that is, approx 2×10^{-16}.

- Build the actual matrix for the experiment as follows:

  ```
  [U,X] = qr(randn(80));
  [V,X] = qr(randn(80));
  S = diag(2.^(-1:-1:-80));
  A = U*S*V;
  ```

  The first two lines here are just to get U and V which are random orthogonal matrices. S is a diagonal matrix with decreasing powers of 2 as diagonal elements $(\frac{1}{2}, \frac{1}{4}, \ldots)$. Building A as the product of these 3 allows us to know what the diagonal elements of our computed R matrices should be. They should be approximately the same as the diagonal elements of S.

- Run your Classical Gram Schmidt and Modified Gram Schmidt algorithms on the experiment matrix A.

- Produce a plot of the diagonal elements of S, RC and RM where the horizontal axis is just the index of the element and the vertical axis is the magnitude of the elements. Use a logarithmic scale for the vertical axis. The Matlab/Octave command to do one of these plots is:

  ```
  semilogy(diag(RC),'rx')
  ```

  In order to add more plots to the same axes enter `hold on` at the command line before proceeding to do more semilogy plots. Use different colors and/or symbols to distinguish the elements of different matrices. (The command above plots the elements with red x's.)

  Explain what you see in the plots. Specifically, at approximately what magnitude do the diagonal elements compute via CGS stop being accurate? At what magnitude to the MGS diagonal elements lose accuracy?

- Also compute the 4 norms that you used above to test your code but this time for your factors of this A. 2 of these should be closed to machine epsilon but 2 will not be.

  Explain what these results tell you.

- Finally, use a built in procedure for a QR factorization to factor A one more time with something like:

```
[QH,RH] = qr(A);
```

Compute the norms to check for accurate factorization and orthogonality for these factors.

How do these norms compare to the CGS and MGS ones? Any ideas why?

To turn in

Upload to the dropbox on D2L a **single PDF file** that contains your plots and your answers/explanations. Include your code for CGS and MGS. I will not be grading or running the code. The dropbox will close at 11:59pm March 11.