**APPM 4600 Lab 13**
Playing with the linear algebra solvers

# 1 Overview

In this lab, you will investigate the performance of different linear algebra solvers. For square systems, you will explore the use of the built in Python solver versus precomputing an LU factorization and using it solve the same linear system with many right hand sides. For over determined systems, you will explore the performance of solving the normal equation versus solving the linear system via the QR factorization solver.

# 2 Before lab

Download the `lab13_code.py` file.

# 3 Lab day

## 3.1 Solving square linear systems

In lecture we talked about how once you build an LU factorization, you can solve a problem with different right hand sides via two triangular solves. In the exercises, for this section, you will investigate for what size of matrix and how many right hand sides, precomputing the LU factorization is faster than using the built in solve command from `SCIPY`.

## 3.2 Exercises

1. The `SCIPY` command `lu` computes the LU factorization of a matrix. The `SCIPY` command `lu_solve` takes the resulting factorization and does the two triangular solves to give you the solution to the linear system.
   For this problem, modify the `lab13_code.py` file so that the linear system

   $$\mathbf{Ax} = \mathbf{b}$$

   can be solved via the LU solution technique in addition to the technique that is already implemented.

2. For $N = 100, 500, 1000, 2000, 4000, 5000$ measure the time it takes to solve the linear system with the two different techniques. For the LU solution technique, break the time into two components: (i) the construction of the LU factorization and (ii) the time for processing a solve.

3. With the results from the last problem, determine for how many right hand sides is using the LU solver faster than using the solver built into the Python. Interpret the results and how it relates to your preconceived ideas.

### 3.3 Solving an over determined linear system

In class, we have talked about two ways of solving an over determined linear system: (i) the normal equation and (ii) via the QR factorization. The subroutine `create_rect` builds an ill-conditioned tall rectangular matrix. The condition number of the matrix returned is 1 divided by the smallest entry in the vector **d**.

### 3.4 Exercises

1. Modify `lab13_code.py` file so that it solves

$$\mathbf{Ax} = \mathbf{b}$$

   via the normal equation and the QR factorization technique.

2. Without changing the smallest entry in the vector **d**, test the two techniques for solving the linear system. How are you going to validate your answer? Which method performs better and why?

3. Change the smallest entry in **d** so that is smaller than $10^{-10}$. How does this effect the behavior of the different solvers?

### 3.5 Deliverables

All codes should be pushed to git and your responses to the questions should be entered into Canvas.