# MACM 316 – Computing Assignment 2

- Read the *Guidelines for Assignments* first.
- Write a one-page PDF report summarizing your finding. This report must also include all your figures.
- Submit the one-page PDF report using Crowdmark and add your Matlab code to it (as extra pages). You must use the link sent by Crowdmark to your SFU email address. Follow the instructions on the Crowdmark to upload your file. If the report is hand-written then use the CamScanner app with your cellphone to scan the report and every page of the code. You will lose marks for poor quality pictures.
- You must acknowledge any collaborations/assistance from fellow students, discussion forums, TAs, instructors, etc.

## QR Factorization

The LU decomposition that we saw in lectures is just one of number different factorizations used in numerical analysis. Another popular choice is the *QR factorization*. In this factorization, an $n \times n$ matrix $\boldsymbol{A}$ is decomposed as

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R},$$

where $\boldsymbol{R}$ is an $n \times n$ upper triangular matrix and $\boldsymbol{Q}$ is an $n \times n$ orthogonal matrix; that is, $\boldsymbol{Q}$ satisfies $\boldsymbol{Q}^{-1} = \boldsymbol{Q}^\top$. In Matlab, you can obtain the QR factorization as follows:

```
[Q R] = qr(A); % Compute the QR factorization of A
```

One use of QR factorization is solving linear systems of equations. The system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is equivalent to the upper triangular linear system $\boldsymbol{R}\boldsymbol{x} = \boldsymbol{Q}^\top \boldsymbol{b}$, which can be solved easily by back substitution.

Your objective in this assignment is to compare the **robustness** and **efficiency** of solving a linear system using QR factorization versus Gaussian elimination with partial pivoting. You can assume the latter is implemented via the backslash command as follows:

```
xhat = A\b; % Solve the linear system using GE with partial pivoting
```

Specifically, you will consider solving the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, where $\boldsymbol{x} = (1, 1, \ldots, 1)^\top$ is a vector of ones and $\boldsymbol{A}$ is an $N \times N$ random matrix with unit normal entries (use the `randn` command in Matlab). Write a code that solves this system using both the above approaches, and for each computes the error

$$e_N = \max_{i=1,\ldots,N} |\hat{x}_i - x_i|,$$

and the computational time $t_N$. Since $\boldsymbol{A}$ is random, you should average these quantities over a suitable number of trials. You may wish to look at the demo *TicToc.m* for further information.

Run your code for a range of $N$ and produce figures showing $e_N$ versus $N$ and $t_N$ versus $N$. Use a suitable number of trials, range of $N$ and axes for your figures, and briefly justify your choice for each. What can you say about the comparative efficiency and robustness of both methods? Make sure to relate your discussion to the relevant concepts seen in the lecture notes.

Next, repeat the above experiments, but with $\boldsymbol{A} = \boldsymbol{B}^\top \boldsymbol{B}$, where $\boldsymbol{B}$ is an $N \times N$ random matrix with unit normal entries. Compare your results with the previous case and discuss, again using concepts from the lectures where appropriate.

Document your results and conclusions in your one-page report.