MACM 316 - Computing Report #1

David Pham / dhpham@sfu.ca / 301318482

(a) Obtain values $\varepsilon_{res}(N)$ for several values of N.

I chose values

$$N = \{\ n \times 2^4 \mid n \in \mathbb{N}, 1 \leq n \leq 64\ \} \text{ and } N_{ex} = 1,000$$

since I started the assignment pretty late. I needed to both compute $\varepsilon_{res}(N)$ and finish this assignment within an hour or two.

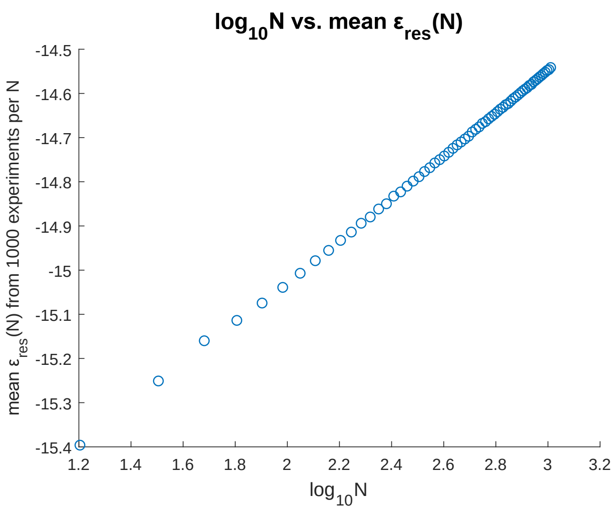
I decided on multiples of $2^4 = 16$ since they gave me a decent amount of data points to work with, while also growing at a fast enough rate that would work within my time constraints.

Accuracy, Robustness, and Efficiency

Of course I'd like to have more accuracy by setting a higher bound for n, but Gaussian Elimination grows at $O(N^3)$. Computing a square matrix of size $(64 \times 2^4)^3 = 1024$ takes 1.07×10^9 operations, so if I double the higher bound of n, then the number of operations required to solve a 2048x2048 matrix is increased by a factor of $2^3 = 8$, almost a factor of ten. Since $N_{ex} = 1,000$, this factor is multiplied by $1,000$. Solving matrices of size 2048x2048 already takes my computer a few minutes, so this increased cost is unaffordable.

I try to get around the lack of gargantuan-sized matrices in my dataset by increasing the number of smaller-sized matrices, and by taking $N_{ex} = 1,000$. I've found that this large value of N_{ex} results in an approximation of the normal distribution by the samples of res_err^{k} for each $k = 1, 2, ..., N_{ex}$.

(b) Include a plot of the points $(log_{10}N, \varepsilon_{res}(N))$.



(c) Use your plot from (b) to argue for your value of an estimated value N^* where $\varepsilon_{res}(N^*) \approx 0$.

The plot from (b) suggests a linear relationship between $log_{10}N$ and $\varepsilon_{res}(N)$. If I take the points $P_0 = (x_0, y_0)$ to be the first point in my dataset and $P_1 = (x_1, y_1)$ to be the last point, then

$$P_0 = (log_{10}(16),\ \varepsilon_{res}(16)) \text{ and } P_1 = (log_{10}(1024),\ \varepsilon_{res}(1024)).$$

The line L running through P_0 and P_1 has the equation $y = L(x) = mx + b$, and an approximation of the slope m can be determined by

$$m \approx \frac{y_1 - y_0}{x_1 - x_0} \approx 0.4773.$$

Using m, we can solve for the y-intercept b by substituting in the values of $P_0(x_0, y_0)$, such that

$$\begin{aligned}
&y = mx + b \\
&\quad \Rightarrow y_0 = mx_0 + b \\
&\quad \Leftrightarrow \varepsilon_{res}(1024) = (0.4773)log_{10}(1024) + b \\
&\quad \Leftrightarrow b \approx -15.9661.
\end{aligned}$$

We can now solve for the x-intercept $(x, 0)$, which can be interpreted as the value of $log_{10}N^*$ that produces a mean residual error $\varepsilon_{res}(N^*)$ equal to the solution, such that no digits of accuracy remain.

$$\begin{aligned}
&y = mx + b \\
&\Rightarrow 0 = (0.4733)x - 15.9661 \\
&\Rightarrow x \approx 33.7336
\end{aligned}$$

If we round up our result for x, we get $x \approx 34$. Since this is the value of $log_{10}(N^*)$, this implies that $N^* = 10^{34}$. We would need a matrix of size $10^{34} \times 10^{34}$ before $\varepsilon_{res}(N^*) \approx 0$, eliminating all accuracy. Clearly, since the number of operations in Gaussian Elimination is bounded by $O(N^3)$, this means that $(10^{34})^3 \approx 10^{102}$ is the number of operations for solving a matrix of size N^*. Even if a computer existed that could perform one quintillion operations per second, that is, 10^{18} operations/sec, the amount of time required to finish one matrix solve is

$$\frac{10^{102} \text{ operations}}{10^{18} \text{ operations/sec}} \cdot \frac{1 \text{ min}}{60 \text{ secs}} \cdot \frac{1 \text{ hr}}{60 \text{ mins}} \cdot \frac{1 \text{ day}}{24 \text{ hrs}} \cdot \frac{1 \text{ year}}{365 \text{ days}} \approx 3 \times 10^{76} \text{ years}.$$

```
%
%  w02w_GEerr_edit.m -- GE truncation error (dhp -- 21 jan 2019)
%

C = 16;         %  C = growth of matrix size per loop;
nmax = 64;      %  nmax = max value of n for n*C
Nex = 10*100;  %  Nex = # of experiments

%  data vector of mean residual error for NxN sized matrices
mean_res_err = zeros(nmax,2);

for n = 1:nmax
    %  N = size of matrix A
    N = n*C

    %  solution of all ones
    x0 = ones(N,1);

    %  data vector of errors
    res_err = zeros(Nex,1);

    for kk = 1:Nex
        %  make random matrix & b-vector
        A = eye(N,N) + randn(N,N)/sqrt(N);
        b = A*x0;

        %  GE via backslash
        x1 = A \ b;

        %  rms residual error
        res_err(kk) = rms(A*x1-b);
    end

    %  mean_res_error for matrices sized NxN
    mean_res_err(n,1) = N;
    mean_res_err(n,2) = mean(res_err);
end


%  plot for mean residual error of NxN sized matrices
figure(1);  clf
subplot(1,1,1)
scatter(log10(mean_res_err(:,1)),log10(mean_res_err(:,2)))

xlabel('log_{10}N','fontsize',12)
ylabel(['mean ?_{res}(N) from ' num2str(Nex) ' experiments per N'])
title('log_{10}N vs. mean ?_{res}(N)','fontsize',14)


N =

    16
```