COL726

Assignment - 3

Rajat Jaiswal
2017CS50415

Q-1 (a) $U = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1m} \\ 0 & a_{22} & \cdots & & a_{2m} \\ 0 & & a_{33} & & a_{3m} \\ \vdots & & & \ddots & \vdots \\ 0 & & & & a_{mm} \end{bmatrix}$

Can be converted to a Lower triangular matrix L.

$$U = P_1 L P_2$$

$$P_1 = P_2 = \underset{m \times m}{\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 1 \\ & & & & 1 & 0 \\ & & & 1 & 0 & 0 \\ & & & & & \\ 1 & & & & 0 & 0 \end{bmatrix}} = \begin{cases} 1 & \text{when } i+j=m+1 \\ 0 & \text{otherwise} \end{cases}$$

Swap Column n with Column 1, Column n-1 with Column 2 and so on, then Swap row m with row m-1, row 2 with row m-2 & so on.

Given $A = PUQ$ (PLT Matrix)

$$A = PP_1 L P_2 Q = (PP_1) L (P_2 Q) = P' L Q'$$

$P' = PP_1 \quad Q' = P_2 Q$.

Product of 2 permutation matrices is also a permutation matrix. L is lower triangular.

Hence Proved.

$$A = \begin{bmatrix} \leftarrow a_1 \rightarrow \\ \leftarrow a_2 \rightarrow \\ \vdots \\ \leftarrow a_m \rightarrow \end{bmatrix} \qquad P_1 A = \begin{bmatrix} \leftarrow a_m \rightarrow \\ \leftarrow a_{m-1} \rightarrow \\ \vdots \\ \leftarrow a_2 \rightarrow \\ \leftarrow a_1 \rightarrow \end{bmatrix}$$

a_i = row i of A.

$B = \begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ b_1 & b_2 & \cdots & b_m \\ \downarrow & \downarrow & & \downarrow \end{bmatrix}$ b_i = Column i of B

$BP_2 = \begin{bmatrix} \uparrow & \uparrow & & \uparrow & \uparrow \\ b_m & b_{m-1} & \cdots & b_2 & b_1 \\ \downarrow & \downarrow & & \downarrow & \downarrow \end{bmatrix}$

Hence, $P_1 \cup P_2 = L$

(b) This can be done by just counting zeros in every row & every column.

Idea: $A = PLQ$ where $l_{ij} = \begin{cases} \text{non-zero when } i \geq j \\ 0 \text{ otherwise.} \end{cases}$

multiplication by P & Q should preserve no. of zeros in a permuted row & no. of zeros in a permuted column.

→ CountZeros (A):

```
C = 0.
for k = 1 to m:
    C += 1 if A[k] > 0 else 0
return C
```

→ ComputePQ (A).

```
P = I, Q = I, L = A
Zero_in_row = Zero of size m array
Zero_in_column = Zero of size m array
for i = 1 to m:
    Zero_in_row[i] = ComputeZeros(L[i, :])
    Zero_in_Column[i] = ComputeZeros(L[:, i])
End.
```

```
for i = 1 to m:
    n = i
    for j = i+1 to m
        if zeros_in_row[j] > zeros_in_row[n]
            n = j
    if zeros_in_column[j] < zeros_in_column[n]
        n = j
    L[i, :] ↔ L[n, :]   (interchange)
    P[i, :] ↔ P[n, :]
    zeros_in_row[i] ↔ zeros_in_row[n].

for i = 1 to m:
    n = i
    for j = i+1 to m
        if zeros_in_column[j] < zeros_in_column[n]
            n = j
    L[:, i] ↔ L[:, n]
    Q[i, :] ↔ Q[n, :]
    zeros_in_column[i] ↔ zeros_in_column[n]

return P, Q
```

Say $c_1, c_2, c_3, \ldots c_m$ are count of zeros in columns & $r_1, r_2, r_3, \ldots r_m$ are count of zeros in rows then

$Q = [q_1 \mid q_2 \mid \ldots \mid q_m]$ where $q_i = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$ ← at index c_i starting with 0

$P = \begin{bmatrix} \leftarrow p_1 \rightarrow \\ \leftarrow p_2 \rightarrow \\ \vdots \\ \leftarrow p_m \rightarrow \end{bmatrix}$ where $p_i = [0.0 \ldots \underset{\downarrow}{1} \ldots 0]$

at index $(m - r_i - 1)$ starting with 0.

We are given that L has no extra zeros & so we can exploit that fact, for L the no. of zero entries decreases in direction of rows from top to bottom & increases in column from left to right.

Counting the zeros happen in $O(m^2)$ time overall.
$O(m)$ for each row & $O(m)$ for each column so $O(m^2 + m^2)$.
Finding matrix P and Q takes $O(m^2)$ each.
Swapping rows or columns are $O(m)$ each.
Finding the index for each row or column takes $O(m)$ time
so overall the algorithm remains $O(m^2)$ time.

Q.2 (a) Let $A = \begin{bmatrix} a_{11} & w^* \\ w & K \end{bmatrix}$ $a_{11} \neq 0$

we wouldn't take $x = \sqrt{a_{11}}$ instead,

$$A = \underbrace{\begin{bmatrix} 1 & 0 \\ w/a_{11} & I \end{bmatrix}}_{R_1^*} \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & K - \frac{ww^*}{a_{11}} \end{bmatrix}}_{A_1} \underbrace{\begin{bmatrix} 1 & w^*/a_{11} \\ 0 & I \end{bmatrix}}_{R_1}$$

If A is a positive semidefinite then $(R_1^*)^{-1} A_1 (R_1)^{-1}$ is also positive semidefinite matrix

$\Rightarrow$ $K - \frac{ww^*}{a_{11}}$ is positive semidefinite

The diagonal entries of $K - \frac{ww^*}{a_{11}}$ is also non-zero

we can continue this and finally we will get

$$A = \underbrace{R_2^* R_2^* \cdots R_m^*}_{R^*} D \underbrace{R_m R_{m-1} \cdots R_2 R_1}_{R}$$

$\Rightarrow A = R^* D R$ where $R = R_m R_{m-1} \cdots R_2 R_1$ & D is a diagonal matrix.

R_i's are upper triangular matrix so R^* is an upper

triangular matrix. Hence, R^* is a lower triangular matrix.

$L = R^*$, $L^* = R$

we get, $A = LDL^*$.

Q.2. (b) a_{11} need not be positive, but a_{11} can be zero. Hence this algorithm doesn't work for all non-singular Hermitian matrices.

A trivial counterexample can be

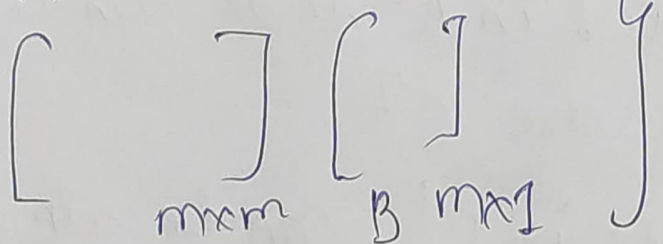
$$A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$ It is non-singular.

The algorithm fails at the very first step as $a_{11} = 0$ and division by 0 is not possible.

Q.3 (a) $A, B \in \mathbb{C}^{m \times m}$ are 2 C-Sparse matrices.

If we can prove that any column of Product matrix $C = AB$ is C^2-Sparse then WLOG, it is C^2-Sparse row-wise as well. & hence the matrix C is C^2-Sparse.

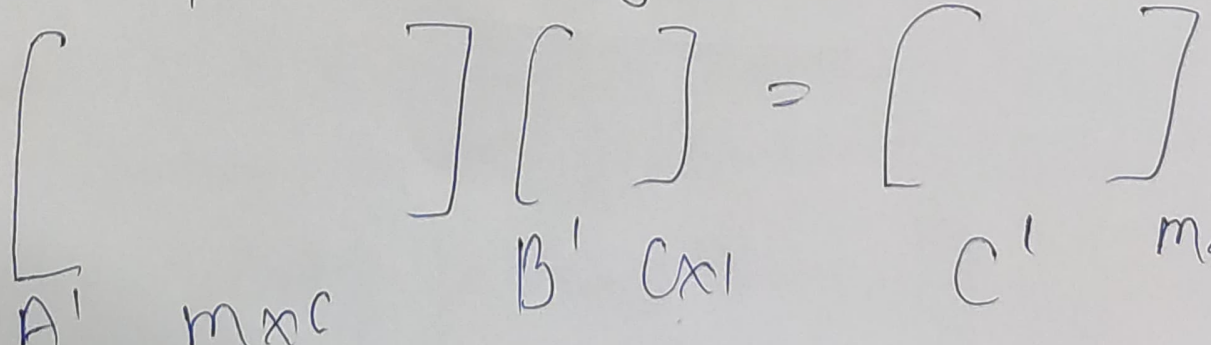
Take any one column of B & multiply with A.



To prove this has at max C^2 elements that are non-zero.

↳ This column of B has at max C non zero elements

So we can take these rows from Colum of B & so corresponding columns from A can be ignored.



All entries of B' are non-zero.
Each column of A' has at most C non-zero elements & there are C columns of A' so A' has atmost C^2 non-zero elements

In matrix C', the number of elements that are non-zero = No. of rows in A' that are non-zero (since B' has all elements as non-zero)

Since A' has atmost C^2 non-zero elements

$\Rightarrow$ max. no. of rows in A' that can be non-zero are C^2

$\Rightarrow$ no. of elements in C' can be atmost C^2

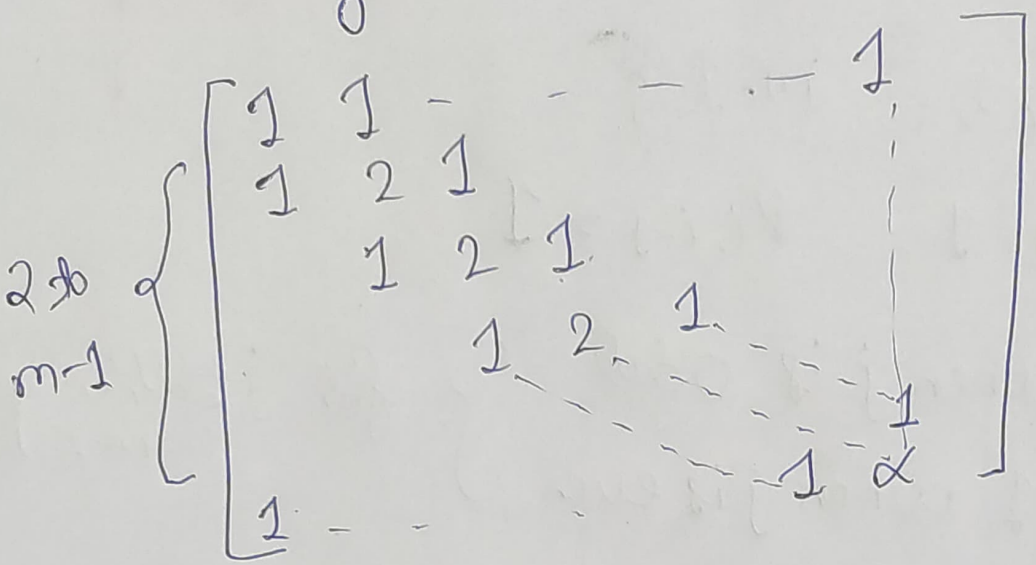
Hence C' has C^2-Sparsity

$\Rightarrow$ $C = AB$ has at most C^2-Sparsity

Hence Proved

Q.3. (b) Such a matrix A is of the form of

Proof by Construction.



for a matrix of size $m \times m$

$a_{11} = 1 \quad a_{12} = 1 \quad a_{1m} = 1$

$a_{m1} = 1 \quad a_{m,m-1} = 1 \quad a_{mm} = \alpha$

$\forall i \in \{2, 3, \ldots m-1\} \quad a_{ii} = 2 \quad a_{i,i+1} = 1$

$a_{i,i-1} = 1$

Choose α in a way to make it H.P.D.

As a rule of thumb choose $\alpha \geq 2^m$

Clearly A is 3-Sparse as each row & column have 3 elements only.

after cholesky factorization $A = RR^T$ where R is lower triangular.

R Comes out to be of the form of

$R_{11} = 1$

for $i \in \{2, 3, \ldots m-1\}$

$R_{ii} = 1 \quad R_{i,i-1} = 1$

$R_{mj} = 1$ when j is odd $\}$ for $j \in \{1, 2, \ldots m-2\}$
& $R_{mj} = -1$ when j is even $\}$

$R_{m\,m-1} = 0$ when m is odel

$R_{m\,m-1} = 2$ when m is even.

R_{mm} can be calculated using $\alpha = R_m \cdot R_m^T$ we know all elements of R_m from 2 to $m-1$

Precisely $R_{mm} = \sqrt{\alpha - (m-2)}$ when m is odd

$R_{mm} = \sqrt{\alpha - (m+2)}$ when m is even.

we can clearly see that ~~2 rows entries~~ $m-1$ entries of R_m are ~~greater than~~ non-zero, hence R is not ~~2-sparse~~ at all Sparse matrix.

e.g.

m=3

$$A=\begin{bmatrix}1&1&1\\1&2&1\\1&1&8\end{bmatrix}=\underbrace{\begin{bmatrix}1&0&0\\1&1&0\\1&0&2.6458\end{bmatrix}}_{R}\underbrace{\begin{bmatrix}1&1&1\\0&1&0\\0&0&2.6458\end{bmatrix}}_{R^T}$$

m=4

$$A=\begin{bmatrix}1&1&0&1\\1&2&1&0\\0&1&2&1\\1&0&1&6\end{bmatrix}=\underbrace{\begin{bmatrix}1&0&0&0\\1&1&0&0\\0&1&1&0\\1&-1&2&\sqrt{10}\end{bmatrix}}_{R}\underbrace{\begin{bmatrix}1&1&0&1\\0&1&1&-1\\0&0&1&2\\0&0&0&\sqrt{10}\end{bmatrix}}_{R^T}$$

m=5

$$A=\begin{bmatrix}1&1&0&0&1\\1&2&1&0&0\\0&1&2&1&0\\0&0&1&2&1\\1&0&0&1&32\end{bmatrix}=\underbrace{\begin{bmatrix}1&0&0&0&0\\1&1&0&0&0\\0&1&1&0&0\\0&0&1&1&0\\1&-1&1&0&\sqrt{29}\end{bmatrix}}_{R}\underbrace{\begin{bmatrix}1&1&0&0&1\\0&1&1&0&-1\\0&0&1&1&1\\0&0&0&1&0\\0&0&0&0&\sqrt{29}\end{bmatrix}}_{R^T}$$

m=6

$$A=\begin{bmatrix}1&1&0&0&0&1\\1&2&1&0&0&0\\0&1&2&1&0&0\\0&0&1&2&1&0\\0&0&0&1&2&1\\1&0&0&0&1&64\end{bmatrix}=\underbrace{\begin{bmatrix}1&0&0&0&0&0\\1&1&0&0&0&0\\0&1&1&0&0&0\\0&0&1&1&0&0\\0&0&0&1&1&0\\1&-1&1&-1&2&\sqrt{56}\end{bmatrix}}_{R}\underbrace{\begin{bmatrix}1&1&0&0&0&1\\0&1&1&0&0&-1\\0&0&1&1&0&1\\0&0&0&1&1&-1\\0&0&0&0&1&2\\0&0&0&0&0&\sqrt{56}\end{bmatrix}}_{R^T.}$$

Q4. (a) We will use the QR factorization of $A_{k_{n-1}}$ to update or compute the QR factorization of A_{k_n}.

$Q'R' = A_{k_{n-1}}$ $\quad \therefore A_{k_n} = [Q'R' \mid A^n b]$

$q_1, q_2, q_3, \ldots q_{n-1}$ be n orthonormal columns of Q', we can compute q_n as

$$q_n = A^n b - \sum_{i=1}^{n-1} r_{in} q_i \qquad r_{ij} = q_i^* (A^j b)$$

$$r_{nn}$$

Each r_{in} will take $O(m)$ time & there are a total of n such r_{in} so q_n takes $O(mn)$ time. So the QR updation can happen in $O(mn)$ time

$Ax = b$

$x = k_n y$

$A_{k_n} y = b$

Let $A_{k_n} = Q'' R''$

$Q'' R'' x = Q'' \cdot Q''^* b$

$\Rightarrow R'' x = Q''^* b$

Algo for $n = 1, 2, 3, \ldots$

(1) add $A^n b$ to matrix

(2) update QR factorization. (Q'' R'')

(3) Compute $Q''^* b$.

(4) Solve triangular system using backsubstitution for $R'' x = Q''^* b$ for x.

(2) takes $O(mn)$ times, (3) takes $O(mn)$ time (4) takes $O(n^2)$ time. So overall the algorithm takes $O(n(mn + n^2))$

$= O(n^2(m+n))$ time.

4.(b)

Given $A = V \Lambda V^*$ $\Rightarrow$ $A^2 = V \Lambda^2 V^*$

$\Rightarrow$ $A^n = V \Lambda^n V^*$

$$A K_n = [Ab \mid A^2 b \mid \ldots \mid A^n b]$$

Assume, $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \ldots \geq \lambda_m$

$A K_n e_n = A^n b = V \Lambda^n V^* b$ $\quad V^* b = [1\,1\,1 \ldots 1]^T$

$\|A K_n e_n\|_2 = \|\Lambda^n\|_2 = \lambda_1^n$

$\|A K_n e_1\|_2 = \|Ab\|_2 = \|\Lambda\|_2 = \lambda_1$

(a) $\lambda_1 < 1$ $\quad \|A^k e_n\|_2 = \lambda_1^n$

$\|A^k e_1\|_2 = \lambda_1$

$$K(A^k) = \frac{\sup_{\|x\|=1} \|A^k x\|_2}{\inf_{\|x\|=1} \|A^k x\|_2} > \frac{\|A^k e_1\|_2}{\|A^k e_n\|_2}$$

$$\Rightarrow K(A^k) = \frac{\lambda_1}{\lambda_1^n}$$

$$= (\lambda')^{n-1}$$

$\lambda_1 < 1$

$\lambda' = \frac{1}{\lambda_1} > 1$

which grows exponentially

(c) $\lambda_1 > 1$

$$K(A_{k_n}) = \frac{\sup_{\|x\|=1} \|A_{k_n} x\|_2}{\inf_{\|x\|=1} \|A_{k_n} x\|_2} > \frac{\|A_{k_n} e_n\|_2}{\|A_{k_n} e_1\|_2}$$

$$K(A_{k_n}) = (\lambda_1)^{n-1}, \quad \lambda_1 > 1$$

which again grows exponentially.

So the conditions of the least squares problem grows exponentially with n.

Q.5. Consider the roots of the polynomials at the midpoint of the intervals i.e

$$\frac{a_1+b_1}{2}, \frac{a_2+b_2}{2}, \ldots, \frac{a_n+b_n}{2}$$

We know in a CG's nth iteration.

$$\frac{\|e_n\|_A}{\|e_0\|_A} \leq \inf_{p \in P_n} \max_{\lambda \in \Lambda(A)} |p(\lambda)|$$

So the polynomial,

$$p(\lambda) = \frac{1}{p(0)}\left(\lambda - \left(\frac{a_1+b_1}{2}\right)\right)\left(\lambda - \left(\frac{a_2+b_2}{2}\right)\right) \cdots \left(\lambda - \left(\frac{a_n+b_n}{2}\right)\right)$$

$$\Rightarrow p(0) = 1.$$

we took $\lambda_1 \in [a_1, b_1]$, $\lambda_2 \in [a_2, b_2]$ ...

$\lambda_n \in [a_n, b_n]$.

...

So $\lambda_1 < \lambda_2 < \lambda_3 \ldots < \lambda_n$.

So, $$\frac{\|e_n\|_A}{\|e_0\|_A} \leq \max\left(p(\lambda_1), p(\lambda_2), \ldots p(\lambda_n)\right)$$

If for every i, $[a_i, b_i]$ interval is small enough, then $\forall i$ $p(\lambda_i)$ will contribute a small factor

with $\left(\lambda_i - \left(\frac{a_i + b_i}{2}\right)\right)$ & with others terms not so big, the entire terms turns out to be quite small.

for $2n$ iteration, $p'(\lambda) = (p(\lambda))^2$.

given $p(0) = I \Rightarrow p'(0) = I$.

$$\frac{\|e_{2n}\|_A}{\|e_0\|_A} \leq \max\left(p'(\lambda_1), p'(\lambda_2), \ldots p'(\lambda_n)\right).$$

when $a_i = b_i$, $\forall i$

$$p(\lambda) = (\lambda - a_1)(\lambda - a_2) \ldots (\lambda - a_n)$$

$\because$ @ $p(\lambda_1) = p(\lambda_2) = p(\lambda_3) = \ldots p(\lambda_n) = 0$

hence, $\frac{\|e_n\|_A}{\|e_0\|_A} \leq 0$. So the error becomes 0 after n iterations.

This also comes as a result of Theorem 38.4 of Trefethan & Bau that if A has n distinct eigenvalues, then CG iteration converges after n iterations (at most n). Here $a_1 < a_2 < \ldots a_n$ so the error will become 0 after n iterations. ∎

Q.6. (a) Submitted q6_2017CS50415.py

(b) We will solve $PAQ = LU$

P is a permutation matrix applied on the rows &
Q is a permutation matrix applied on columns on right

$PAQ = LU$ $\quad Ax = b$ $\quad A = P^{-1} L U Q^{-1}$

$LUQ^{-1}x = Pb$ $\quad Lz = Pb$ } get z

$Uy = z$ } get y $\quad \boxed{x = Qy}$

Algorithm :- $U = A \quad L = I \quad P = I \quad Q = I$

for $k = 1$ to $m-1$:

$i, j =$ Pivot (U, k)

$U_{k,k:m} \leftrightarrow U_{i,k:m}$ (interchange rows)

$U_{:,k} \leftrightarrow U_{:,j}$ (interchange columns)

$L_{k,1:k-1} \leftrightarrow L_{i,1:k-1}$

$P_{k,:} \leftrightarrow P_{i,:}$

~~$Q_{k,:} \leftrightarrow Q_{i,:}$~~ $Q_{:,k} \leftrightarrow Q_{:,j}$

~~for n in range = k+1 to m~~

~~if $U_{i,k} / U_{k,k}$~~

for $n = k+1$ to m:

$l_{n,k} = u_{nk} / u_{kk}$

$u_{n,k:m} = u_{n,k:m} - l_{nk} * u_{k,k:m}$

return P, Q, L, U

(c) Submitted q6_2017CS50415.py

The plots are on the next page.

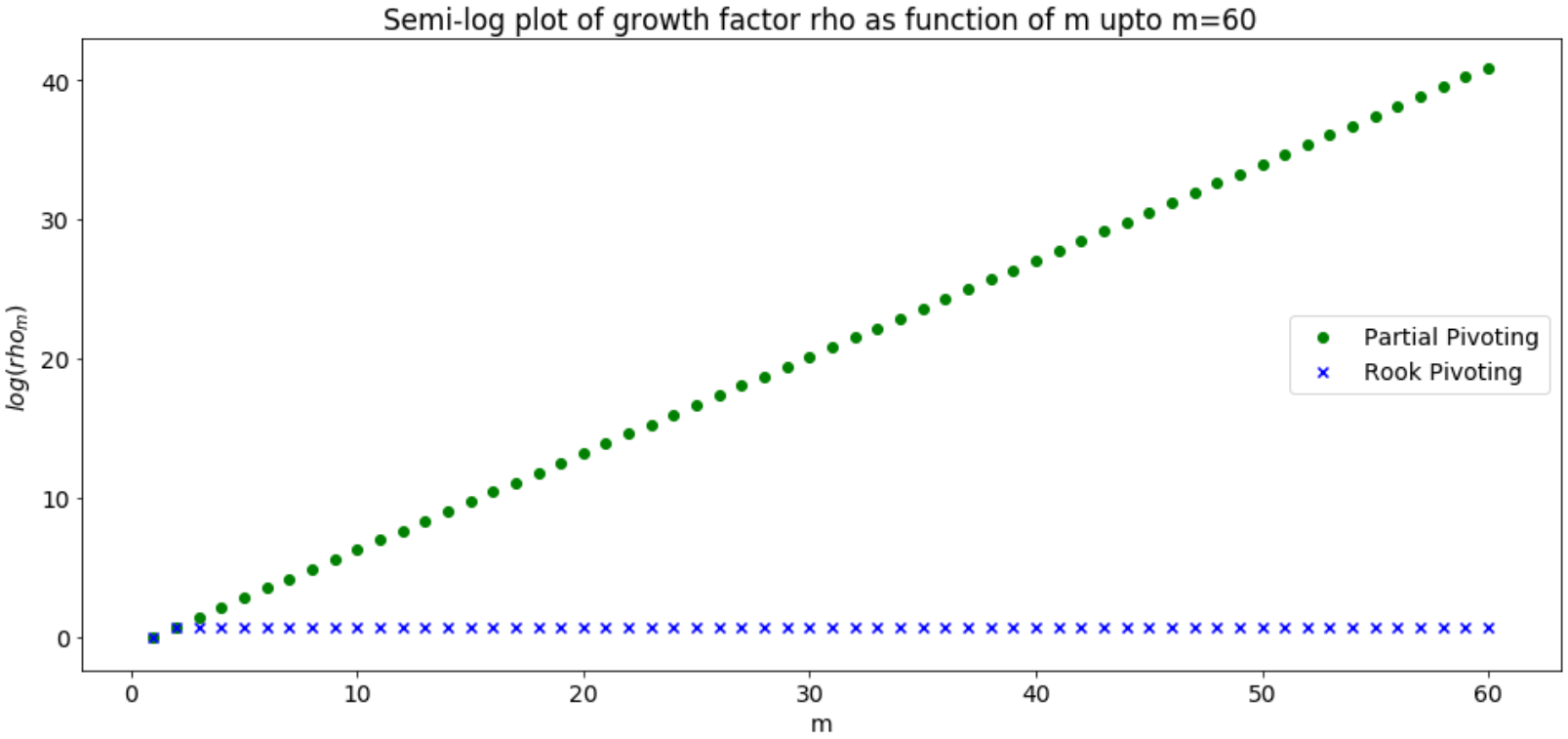


Figure 1: Semi log plot of growth factor with change in m

As evident in the plots above, the growth factor ρ in partial pivoting increases exponentially with increase in m, whereas in case of rook pivoting it remains constant and is of O(1). Hence partial pivoting is not backward stable, as growth factor ρ indicates backward stability. The same is evident in the plots of relative backward error. The relative backward error for partial pivoting increases exponentially with increase in m. Upto certain values it remains in the $O(\varepsilon_m)$ and after that it shoots up for partial pivoting. For rook pivoting it remains of the $O(\varepsilon_m)$

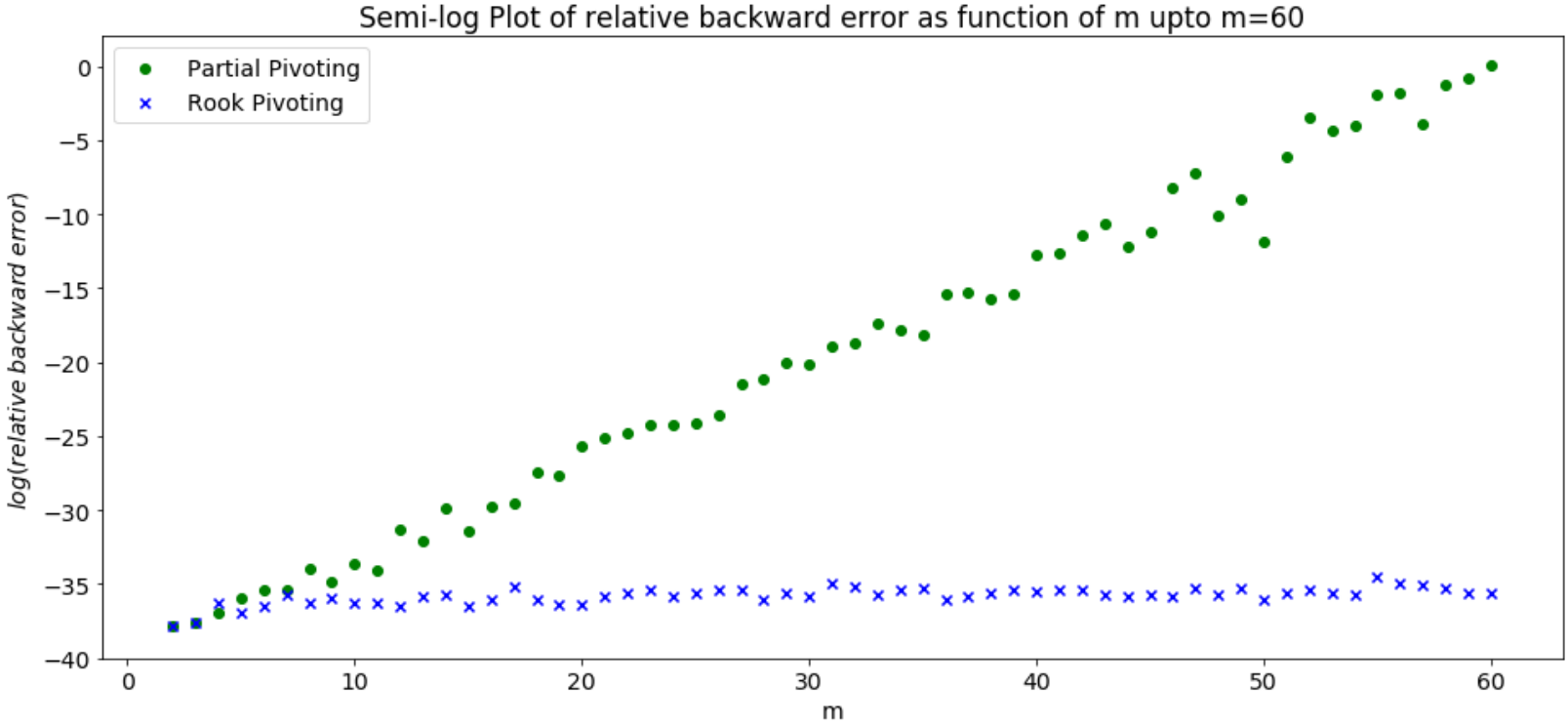


Figure 2: Semi log plot of relative backward error with change in m