Assignment One

October 6, 2019

MCSC 6020G
Fall 2019
Submitted by Derick Smith

Question One:

Skew-symmetric Matrix B, $B^T = -B$.

a) Example 4x4.

$$B = \begin{bmatrix} 0 & -1 & 2 & -3 \\ 1 & 0 & -4 & 5 \\ -2 & 4 & 0 & 6 \\ 3 & -5 & 6 & 0 \end{bmatrix}$$

$$-B = \begin{bmatrix} 0 & 1 & -2 & 3 \\ -1 & 0 & 4 & -5 \\ 2 & -4 & 0 & 6 \\ -3 & 5 & -6 & 0 \end{bmatrix}$$

$$(-B)^T = \begin{bmatrix} 0 & -1 & 2 & -3 \\ 1 & 0 & -4 & 5 \\ -2 & 4 & 0 & 6 \\ 3 & -5 & 6 & 0 \end{bmatrix}$$

b) Show Orthogonality of $A = (\mathbb{I} + B)(\mathbb{I} - B)^{-1}$ with B skew-symmetrix.

$A := (I + B)(I - B)^{-1}$

Inverse of A:

$$\begin{aligned} A^{-1} &= \left[(I + B)(I - B)^{-1}\right]^{-1} \\ &= (I - B)(I + B)^{-1} \end{aligned}$$

Transpose of A:

$$\begin{aligned} A^T &= \left[(I + B)(I - B)^{-1}\right]^T \\ &= \left[(I - B)^T\right]^{-1} (I + B)^T \\ &= (I + B)^{-1}(I - B) \end{aligned}$$

Inverse of Transpose of A:

$$\begin{aligned} \left[A^T\right]^{-1} &= \left[(I + B)^{-1}(I - B)\right]^{-1} \\ &= (I - B)^{-1}(I + B) \\ &= A \end{aligned}$$

Transpose of A dot A:

$$\begin{aligned} A^T A &= (I + B)^{-1}(I - B)(I - B)^{-1}(I + B) \\ &= (I + B)^{-1}(I)(I + B) \\ &= (I + B)^{-1}(I + B) \\ &= I \end{aligned}$$

Question Two:

Prove $\frac{1}{n}\|v\|_1 \leq \|v\|_\infty \leq \|v\|_2$**,** $v \in \mathbb{C}^{\mathbf{n}}$**:**

$\forall k$**,** $v_k = a_k + b_k i$**, where,** $\{a_k, b_k\} \in \mathbb{R}$**,**

and $|v_k| = [\overline{v}_k \cdot v_k]^{\frac{1}{2}} = [(a_k - b_k i) \cdot (a_k + b_k i)]^{\frac{1}{2}} = (a_k^2 + b_k^2)^{\frac{1}{2}}$

Prove $\frac{1}{n}\|v\|_1 \leq \|v\|_\infty$**:**

With $\|v\|_\infty = \max_{\forall k} |v_k|$ choose $k = 1$ so $\|v\|_\infty = |v_1| = \left(a_1^2 + b_1^2\right)^{\frac{1}{2}}$,

where, $\forall k$, $\left(a_k^2 + b_k^2\right)^{\frac{1}{2}} \leq \left(a_1^2 + b_1^2\right)^{\frac{1}{2}}$

$$\begin{aligned}
\left(a_1^2 + b_1^2\right)^{\frac{1}{2}} &\leq \left(a_1^2 + b_1^2\right)^{\frac{1}{2}} \\
\left(a_1^2 + b_1^2\right)^{\frac{1}{2}} + \left(a_2^2 + b_2^2\right)^{\frac{1}{2}} &\leq 2 \cdot \left(a_1^2 + b_1^2\right)^{\frac{1}{2}} \\
&\vdots \\
\sum_{k=1}^{n} \left(a_k^2 + b_k^2\right)^{\frac{1}{2}} &\leq n \cdot \left(a_1^2 + b_1^2\right)^{\frac{1}{2}} \\
\frac{1}{n} \sum_{k=1}^{n} \left(a_k^2 + b_k^2\right)^{\frac{1}{2}} &\leq \left(a_1^2 + b_1^2\right)^{\frac{1}{2}}
\end{aligned}$$

$$\frac{1}{n}\|v\|_1 \leq \|v\|_\infty .$$

Prove $\|v\|_\infty \leq \|v\|_2$:

With $\|v\|_\infty = \max_{\forall k} |v_k|$ choose $k = 1$ so $\|v\|_\infty = |v_1| = \left(a_1^2 + b_1^2\right)^{\frac{1}{2}}$,
and $\|v\|_2 = \left[\sum_{k=1}^{n} \left(\overline{v}_k \cdot v_k\right)\right]^{\frac{1}{2}} = \left[\sum_{k=1}^{n} \left(a_k^2 + b_k^2\right)\right]^{\frac{1}{2}}$,

$$
\begin{aligned}
0 &\leq \sum_{k=2}^{n} \left(a_k^2 + b_k^2\right) \\
0 &\leq \left[\sum_{k=2}^{n} \left(a_k^2 + b_k^2\right)\right] + \left(a_1^2 + b_1^2\right) - \left(a_1^2 + b_1^2\right) \\
0 &\leq \left[\sum_{k=1}^{n} \left(a_k^2 + b_k^2\right)\right] - \left(a_1^2 + b_1^2\right) \\
\left(a_1^2 + b_1^2\right) &\leq \sum_{k=1}^{n} \left(a_k^2 + b_k^2\right) \\
\left(a_1^2 + b_1^2\right)^{\frac{1}{2}} &\leq \left[\sum_{k=1}^{n} \left(a_k^2 + b_k^2\right)\right]^{\frac{1}{2}}
\end{aligned}
$$

$$\|v\|_\infty \leq \|v\|_2 .$$

Therefore,

$$\frac{1}{n} \|v\|_1 \leq \|v\|_\infty \leq \|v\|_2 . \qquad \square$$

Example:

$v = \{(1+2i), (3+4i), (5+6i)\}$

For $\frac{1}{n}\|v\|_1 \leq \|v\|_\infty$:

$$\begin{aligned}
\frac{1}{3}\left([(1-2i)(1+2i)]^{\frac{1}{2}} + [(3-4i)(3+4i)]^{\frac{1}{2}} + [(5-6i)(5+6i)]^{\frac{1}{2}}\right) &\leq [(5+6i)(5+6i)]^{\frac{1}{2}} \\
\frac{1}{3}\left[(1+2^2)^{\frac{1}{2}} + (3^3+4^2)^{\frac{1}{2}} + (5^2+6^2)^{\frac{1}{2}}\right] &\leq (5^2+6^2)^{\frac{1}{2}} \\
(1+2^2)^{\frac{1}{2}} + (3^3+4^2)^{\frac{1}{2}} + (5^2+6^2)^{\frac{1}{2}} &\leq 3\cdot(5^2+6^2)^{\frac{1}{2}} \\
(1+2^2)^{\frac{1}{2}} + (3^3+4^2)^{\frac{1}{2}} &\leq 2\cdot(5^2+6^2)^{\frac{1}{2}}.
\end{aligned}$$

For $\|v\|_\infty \leq \|v\|_2$:

$$\begin{aligned}
[(5+6i)(5+6i)]^{\frac{1}{2}} &\leq [(1-2i)(1+2i) + (3-4i)(3+4i) + (5-6i)(5+6i)]^{\frac{1}{2}} \\
(5^2+6^2)^{\frac{1}{2}} &\leq \left[(1+2^2) + (3^3+4^2) + (5^2+6^2)\right]^{\frac{1}{2}} \\
(5^2+6^2) &\leq (1+2^2) + (3^3+4^2) + (5^2+6^2) \\
0 &\leq (1+2^2) + (3^3+4^2).
\end{aligned}$$

Question Three:

(a) How to Compute det(A) from LUP Factorization.

The determinant of A can be found by the product of the determinants of its decomposed matrices, i.e.

$$PA = LU$$

$$A = P^{-1}LU$$

$$det(A) = det(P^{-1}) \cdot det(L) \cdot det(U).$$

(b) LUP det(A) Pseudo-code.

Matrix A = mxA has dimensions n by n. Matrix U = mxU deep copy of mxA. Matrix L = mxL is n by n identity matrix.

- pSwaps = 0 # used for determinant of P
- i = 0 # outer loop counter
- While i < (n-1):
  - Search mxU for row with largest nonzero absolute leading element for ith column
  - If larger than current ith row:
    - Swap those rows
    - pSwaps++
  - l = i + 1 # inner loop counter
  - While l < n:
    - mxL[l, i] = mxU[l, i] / mxU[i, i]
    - mxU[l,] = mxU[l,] - mxL[l, i] * mxU[i,]
    - l++
  - i++
- i = 0 # reset counter
- mxAdet = (-1)^pSwaps # this is equivalent to det(P^-1)
- While i < n:
  - mxDetA = mxDetA * mxU[i,i]
  - i++
- Return mxDetA

(c) CoFactor det(A) Pseudo-code.

Matrix A = mxA has dimensions n by n.

Define function CofactorDet(mx):

- n = length of mx
- If n == 1: return mx[0,0]
- If n==2: return mx[0, 0] * mx[1, 1] - mx[1, 0] * mx[0, 1]
- mxDet = 0 # to be accumulated throughout regressive algorithm
- i = 0 # loop counter
- While i < n:
  - tempMx = mx - ith row - ith column
    # temporary matrix cofactor
  - mxDet = MxDet + (-1)^(i) * mx[i,0] * CofactorDet(tempMx)
    # recursive call
- return mxDet

(d) Test Algorithms for Time and Error.

Notes:

1. See files for scripts.
2. Both methods errors were calculated individually compared to their difference (absolute and relative) to Python's Numpy linear algebra function det().
3. Python standard library function time.clock() was tracked before and after each method call, including Python's det().
4. Each method was computed 50 times at each dimension, from n=2 to n=11. Their mean and standard deviation for time and error were then computed for analysis.
5. Testing beyond n=11 was not feasible without a more powerful computer. The CoFactor recursive method processing time is not polynomial.

(i) LUP Method.

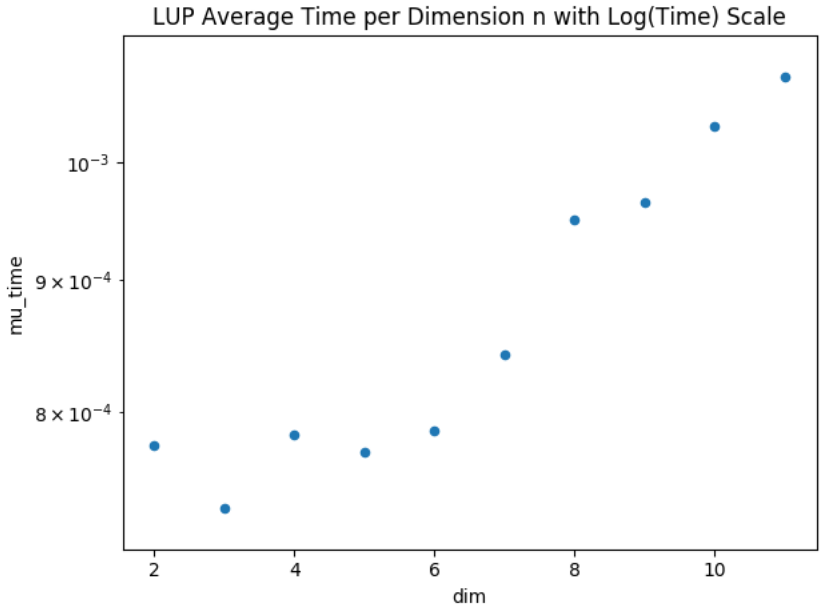
Time:

The time taken by the LUP method was super linearly with the increase in matrix dimension.

Table 1: LUP Det(mxA) Time

n	μ	σ
2	$7.7608 \cdot 10^{-4}$	$1.367747 \cdot 10^{-4}$
3	$7.3372 \cdot 10^{-4}$	$8.63071352786121 \cdot 10^{-5}$
4	$7.8406 \cdot 10^{-4}$	$1.332449 \cdot 10^{-4}$
5	$7.7188 \cdot 10^{-4}$	$4.85240723765062 \cdot 10^{-5}$
6	$7.8668 \cdot 10^{-4}$	$2.79295112739264 \cdot 10^{-5}$
7	$8.4252 \cdot 10^{-4}$	$3.6811 \cdot 10^{-5}$
8	$9.5012 \cdot 10^{-4}$	$1.048978 \cdot 10^{-4}$
9	$9.6452 \cdot 10^{-4}$	$3.21124524130671 \cdot 10^{-5}$
10	$1.03284 \cdot 10^{-3}$	$1.147178 \cdot 10^{-4}$
11	$1.08 \cdot 10^{-3}$	$1.997 \cdot 10^{-4}$

Plot 1: LUP Det(mA) Log(Time)

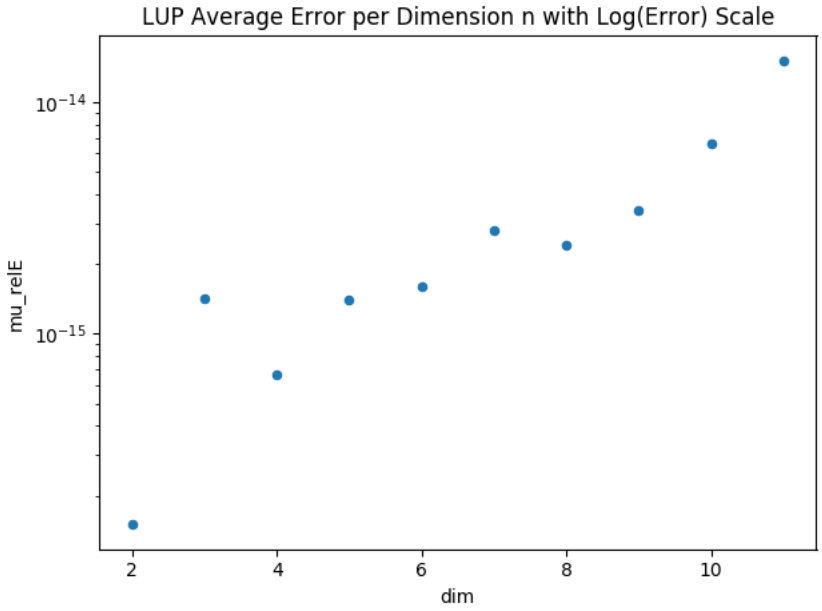


Error:

Table 2: LUP Det(mxA) Error

n	μ	σ
2	$1.50272219720192 \cdot 10^{-16}$	$1.68254760626659 \cdot 10^{-16}$
3	$1.41076335325028 \cdot 10^{-15}$	$3.59600036074122 \cdot 10^{-15}$
4	$6.62001749035301 \cdot 10^{-16}$	$1.13693313632224 \cdot 10^{-15}$
5	$1.40118844089626 \cdot 10^{-15}$	$3.33296631257066 \cdot 10^{-15}$
6	$1.59773540700325 \cdot 10^{-15}$	$2.77370625824385 \cdot 10^{-15}$
7	$2.80480124651704 \cdot 10^{-15}$	$7.6498009965789 \cdot 10^{-15}$
8	$2.41997718548759 \cdot 10^{-15}$	$4.18019888960937 \cdot 10^{-15}$
9	$3.42766999183274 \cdot 10^{-15}$	$4.74788735226157 \cdot 10^{-15}$
10	$6.57323839705876 \cdot 10^{-15}$	$2.43972206315903 \cdot 10^{-14}$
11	$1.50681037874643 \cdot 10^{-14}$	$6.6924959307498 \cdot 10^{-14}$

Plot 2: LUP Det(mA) Log(Error)



(ii) CoFactor Method.

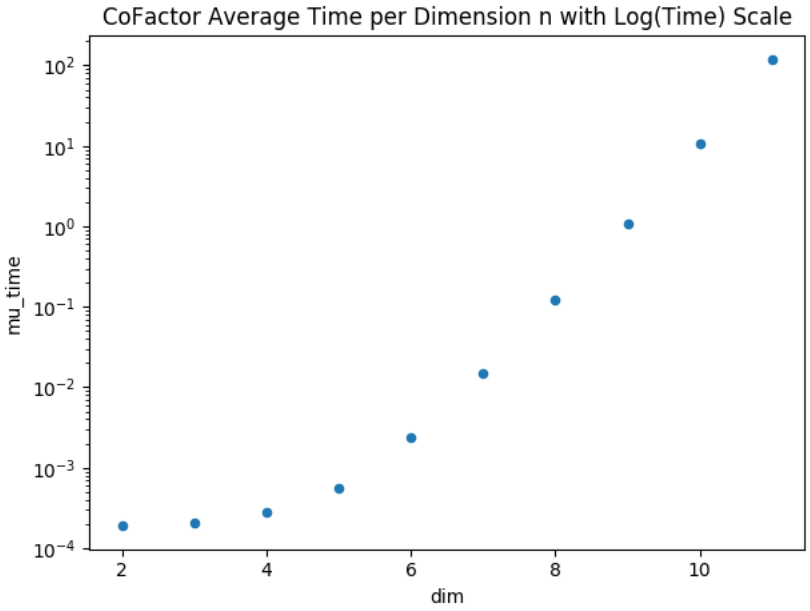
Time:

The time taken by the CoFactor method was super linearly with the increase in matrix dimension. Significantly greater than that of the LUP method.

Table 3: CoFactor Det(mxA) Time

n	μ	σ
2	$1.92599999999983 \cdot 10^{-4}$	$1.35262707351354 \cdot 10^{-5}$
3	$2.09400000000004 \cdot 10^{-4}$	$5.88897274575044 \cdot 10^{-6}$
4	$2.82459999999993 \cdot 10^{-4}$	$3.15412174780973 \cdot 10^{-5}$
5	$5.65819999999988 \cdot 10^{-4}$	$1.04741395828176 \cdot 10^{-5}$
6	$2.39327999999998 \cdot 10^{-3}$	$1.22689044335656 \cdot 10^{-4}$
7	$1.511576 \cdot 10^{-2}$	$2.03081713603174 \cdot 10^{-4}$
8	$1.2063036 \cdot 10^{-1}$	$8.22970832046882 \cdot 10^{-4}$
9	1.09444122	$1.70025335268485 \cdot 10^{-2}$
10	$1.075855772 \cdot 10$	$1.55878988870087 \cdot 10^{-1}$
11	$1.178086217 \cdot 10^{2}$	1.70881036448183

Plot 3: CoFactor Det(mA) Log(Time)

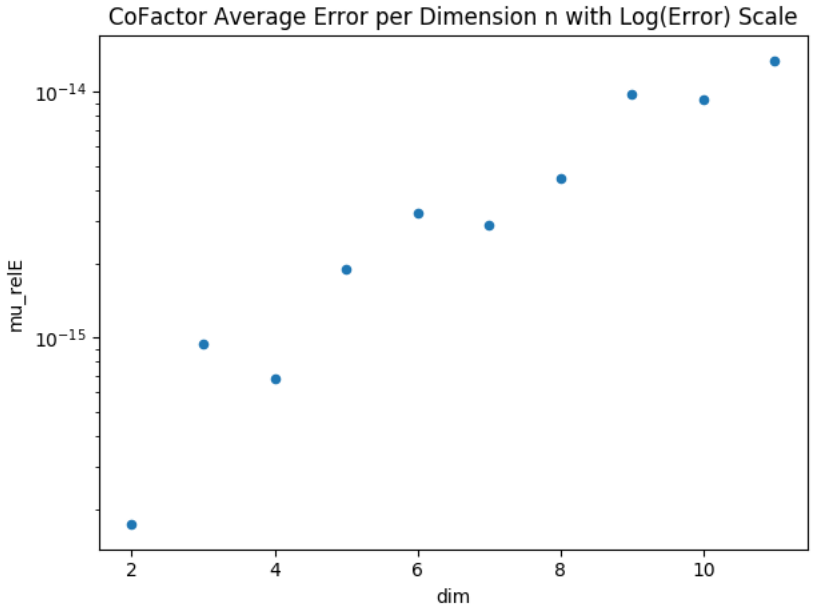


Error:

Table 4: CoFactor Det(mxA) Error

n	μ	σ
2	$1.74415463563628 \cdot 10^{-16}$	$1.86552669289415 \cdot 10^{-16}$
3	$9.48439476322332 \cdot 10^{-16}$	$2.40923167145667 \cdot 10^{-15}$
4	$6.81226180835024 \cdot 10^{-16}$	$1.39005207862167 \cdot 10^{-15}$
5	$1.90477954432557 \cdot 10^{-15}$	$4.11598371212702 \cdot 10^{-15}$
6	$3.22487255952488 \cdot 10^{-15}$	$1.08645960672825 \cdot 10^{-14}$
7	$2.87690455913454 \cdot 10^{-15}$	$8.44074884558458 \cdot 10^{-15}$
8	$4.42657293356288 \cdot 10^{-15}$	$1.17148268098084 \cdot 10^{-14}$
9	$9.81864615141008 \cdot 10^{-15}$	$2.05052323312052 \cdot 10^{-14}$
10	$9.31015858251381 \cdot 10^{-15}$	$3.46051259145651 \cdot 10^{-14}$
11	$1.33660410213406 \cdot 10^{-14}$	$2.81812201167862 \cdot 10^{-14}$

Plot 4: CoFactor Det(mA) Log(Error)



(iii) Numpy det() Method.

Note: Error is not provided as this method was used as the measuring stick which the others were compared with.

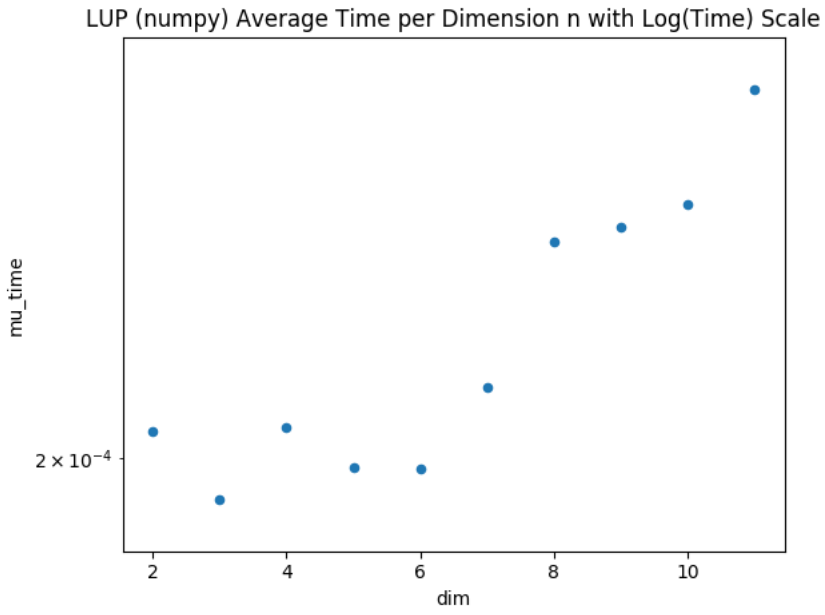
Time:

The time taken for Python's Numpy library was significantly less than both of the other methods.

Table 5: Numpy det(mxA) Time

n	μ	σ
2	$2.02980000000008 \cdot 10^{-4}$	$1.04335804017675 \cdot 10^{-5}$
3	$1.95579999999986 \cdot 10^{-4}$	$8.66507934181306 \cdot 10^{-6}$
4	$2.03359999999986 \cdot 10^{-4}$	$1.76269793214748 \cdot 10^{-5}$
5	$1.99019999999988 \cdot 10^{-4}$	$8.25709391978144 \cdot 10^{-6}$
6	$1.98859999999992 \cdot 10^{-4}$	$7.09086736585715 \cdot 10^{-6}$
7	$2.07879999999978 \cdot 10^{-4}$	$1.07733745873911 \cdot 10^{-5}$
8	$2.24959999999932 \cdot 10^{-4}$	$7.15251004886503 \cdot 10^{-6}$
9	$2.26840000000017 \cdot 10^{-4}$	$5.44558536817928 \cdot 10^{-6}$
10	$2.29580000000738 \cdot 10^{-4}$	$1.46056016684742 \cdot 10^{-5}$
11	$2.44439999983115 \cdot 10^{-4}$	$6.44574775485059 \cdot 10^{-5}$

Plot 5: Numpy det(mA) Log(Time)



(e) Time Complexities.

(i) LUP Method Time Complexity.

As discussed in lecture the time complexity of $O(n^3)$. Flops of the Python script for this method are annotated in the comments.

Time Complexity Calculation:

$$
\begin{aligned}
T(n) &= n + \sum_{i=1}^{n-1} \left(n - 1 + 6n + \sum_{k=1}^{i} (n+1) \right) \\
&= n + \sum_{i=1}^{n-1} \left(7n - 1 + (n+1) \cdot \sum_{k=1}^{i} 1 \right) \\
&= n + \sum_{i=1}^{n-1} (7n - 1 + (n+1) \cdot i) \\
&= n + (n-1)(7n-1) + (n+1) \cdot \sum_{i=1}^{n-1} i \\
&= n + 7n^2 - 8n + 1 + (n+1) \left(\frac{(n-1)(n-2)}{2} \right) \\
&= 7n^2 - 7n + 1 + (n+1) \left(\frac{n^2 - 3n + 2}{2} \right) \\
&= 7n^2 - 7n + 1 + \left(\frac{n^3 - 3n^2 - n + 2}{2} \right) \\
&= \frac{1}{2} \left(n^3 + 11n^2 - 15n + 4 \right) \\
&= O(n^3)
\end{aligned}
$$

(ii) CoFactor Method Time Complexity.

Flops of the Python script for this method are annotated in the comments.

Time Complexity Calculation:

$$\begin{aligned}
T(n) &= \sum_{i=1}^{n} \left(\sum_{i=1}^{n-1} \left(\cdots \left(\sum_{i=1}^{2} \left(\sum_{i=1}^{1} (4) \right) \right) \cdots \right) \right) \\
&= 4 \cdot \sum_{i=1}^{n} \left(\sum_{i=1}^{n-1} \left(\cdots \left(\sum_{i=1}^{2} \left(\sum_{i=1}^{1} (1) \right) \right) \cdots \right) \right) \\
&= 4 \cdot \prod_{i}^{n} (1) \\
&= 4n! \\
&= O(n!)
\end{aligned}$$

This does verify that the datasets and graphs make sense per method.

(iii) Accuracy.

From Table 2, Plot 2, Table 4, and Plot 4, there does not appear to be a substantial difference between the two methods levels of accuracy. At each dimension each methods average relative errors are within the others mean plus or minus three standard deviations. Without more rigorous analysis, it does appear likely that it would fail to reject the null hypothesis test that $\mu_{lup} = \mu_{co}$. That being said, neither seems to be significantly more accurate than the other.