Homework 1
IST 597
Physics-Informed Machine Learning

Question 1

Compute the computational complexity of the Jacobi method, the Gauss-Siedel method, and the Cholesky decomposition for solving a linear system.

Ans:

Jacobi Method

The element-based equation of a Jacobi iteration is:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right), i = 1, 2, 3, ..., n$$

where n is the number of unknowns.

From the above equation, we can count the number of operations required per iteration to update the value of an unknown as:

1. Multiplication: $(n - 1)$
2. Addition: $(n - 1) - 1 = (n - 2)$
3. Subtraction: 1
4. Division: 1

The total number of operations per iteration to update the value of an unknown is the sum of all the above operations, i.e., $(2n - 1)$.

Since there are n unknowns, the total number of operations per iteration to update the values of all the unknowns is $n * (2n - 1)$, which is equal to $(2n^2 - n)$.

Hence, the computational complexity of a Jacobi iteration is n^2 with 2 as the leading term i.e., $\mathcal{O}(2n^2)$.

Gauss-Siedel Method

The element-based equation of a Gauss-Siedel iteration is:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \left(\sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} + \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) \right), i = 1, 2, 3, ..., n$$

where n is the number of unknowns.

From the above equation, we can count the number of operations required per iteration to update the value of an unknown as:

1. Multiplication: $(n - 1)$
2. Addition: $(n - 1) - 1 = (n - 2)$

3. Subtraction: 1

4. Division: 1

The total number of operations per iteration to update the value of an unknown is the sum of all the above operations, i.e., $(2n - 1)$.

Since there are n unknowns, the total number of operations per iteration to update the values of all the unknowns is $n * (2n - 1)$, which is equal to $(2n^2 - n)$.

Hence, the computational complexity of a Gauss-Siedel iteration is n^2 with 2 as the leading term i.e., $\mathcal{O}(2n^2)$.

Cholesky Decomposition

In Cholesky decomposition, A, is a symmetric, positive-definite matrix is decomposed as:

$$A = LL^T$$

where L is a lower triangular matrix.

To compute the elements of L, the formulas are as follows:
For diagonal elements of L:

$$L_{ii} = \sqrt{A_{ii} - \sum_{k=1}^{i-1} L_{ik}^2}$$

For the off-diagonal elements of L (when $i > j$):

$$L_{ij} = \frac{1}{L_{jj}} \left(A_{ij} - \sum_{k=1}^{j-1} L_{ik} L_{jk} \right)$$

Diagonal Elements

Number of diagonal elements: n

Number of operations:

1. Square-root: $1 * n$ (one for each diagonal element)

2. Multiplication: $\frac{n(n-1)}{2}$

3. Addition/ Subtraction: $\frac{n(n-1)}{2}$

4. Division: $i0$

Explanation of Multiplication Count:

Number of multiplication for each row of lower triangular matrix:

i	No. of multiplications
1	0
2	1
3	2
...	...
n	n-1

Table 1: Number of multiplications

Hence, adding all of them, we get,

$\#multiplication = 0 + 1 + 2 + 3 + ... + (n-1) = \frac{n(n-1)}{2}$

Non-Diagonal Elements

Number of diagonal elements: $\frac{n(n-1)}{2}$

Number of operations:

1. Square-root: 0
2. Multiplication: $\frac{(n-1)\cdot n\cdot(n-2)}{6}$
3. Addition/ Subtraction: $\frac{(n-1)\cdot n\cdot(n-2)}{6}$
4. Division: $1 * \frac{n(n-1)}{2}$ (one for each non-diagonal element)

Explanation of Multiplication Count:

Number of elements in each column of a lower triangular matrix:

Column	No. of elements
1	n-1
2	n-2
3	n-3
...	...
n-1	n-(n-1)
n	0

Table 2: Number of elements in each column of a lower triangular matrix

Number of multiplications for elements of each column:

Column	No. of multiplications
1	0
2	1
3	2
...	...
n	n-1

Table 3: Number of multiplications for each element in a column

Hence, adding all the $\#multiplication$ for all the elements, we get, $\#multiplication = (n-1)*0 + (n-2)*1 + (n-3)*2 + ... + (n-(n-1))*(n-1) + 0*(n-1)$

In summation notation,

$$S = \sum_{k=2}^{n-1} (n-k)\cdot(k-1)$$

We can split this into two separate sums:

$$S = \sum_{k=2}^{n-1} [n(k-1) - k(k-1)]$$

Expanding the summation:

$$S = \sum_{k=2}^{n-1} n(k-1) - \sum_{k=2}^{n-1} k(k-1)$$

First Summation

$$\sum_{k=2}^{n-1} n(k-1)$$

Factoring out n:

$$= n \sum_{k=2}^{n-1} (k-1)$$

Calculating $\sum_{k=2}^{n-1}(k-1)$:

$$\sum_{k=2}^{n-1}(k-1) = \sum_{k=1}^{n-2} k = \frac{(n-2)(n-1)}{2}$$

Thus:

$$n\sum_{k=2}^{n-1}(k-1) = n \cdot \frac{(n-2)(n-1)}{2} = \frac{n(n-2)(n-1)}{2}$$

Second Summation

$$\sum_{k=2}^{n-1} k(k-1)$$

Rewriting $k(k-1)$:

$$k(k-1) = k^2 - k$$

Thus:

$$\sum_{k=2}^{n-1} k(k-1) = \sum_{k=2}^{n-1}(k^2 - k) = \sum_{k=2}^{n-1} k^2 - \sum_{k=2}^{n-1} k$$

Using known summation formulas:

$$\sum_{k=2}^{n-1} k^2 = \frac{(n-1)n(2n-1)}{6} - 1$$

$$\sum_{k=2}^{n-1} k = \frac{(n-1)n}{2} - 1$$

So:

$$\sum_{k=2}^{n-1} k(k-1) = \frac{(n-1)n(2n-1)}{6} - 1 - \left(\frac{(n-1)n}{2} - 1\right)$$

Rewriting $\frac{(n-1)n}{2}$ with a denominator of 6:

$$\frac{(n-1)n}{2} = \frac{3(n-1)n}{6}$$

Thus:

$$
\begin{aligned}
\sum_{k=2}^{n-1} k(k-1) &= \frac{(n-1)n(2n-1)}{6} - \frac{3(n-1)n}{6} \\
&= \frac{(n-1)n\left((2n-1)-3\right)}{6} \\
&= \frac{(n-1)n(2n-4)}{6} \\
&= \frac{2(n-1)n(n-2)}{6} \\
&= \frac{(n-1)n(n-2)}{3}
\end{aligned}
$$

Combined Result

$$S = \frac{n(n-2)(n-1)}{2} - \frac{(n-1)n(n-2)}{3}$$

To combine these, we use a common denominator of 6:

$$S = \frac{3n(n-2)(n-1)}{6} - \frac{2(n-1)n(n-2)}{6}$$

Factoring out $\frac{(n-1)n(n-2)}{6}$:

$$
\begin{aligned}
S &= \frac{(n-1)n(n-2)\,(3-2)}{6} \\
&= \frac{(n-1)n(n-2)}{6}
\end{aligned}
$$

Thus, the simplified formula for the number of multiplications is:

$$S = \frac{(n-1)n(n-2)}{6}$$

Number of Each Operations

$\#squareroot = n + 0 = n$

$\#division = 0 + \frac{n(n-1)}{2} = \frac{n(n-1)}{2}$

$\#multiplication = \frac{n(n-1)}{2} + \frac{(n-1)n(n-2)}{6} = \frac{(n-1)n(n+1)}{6}$

$\#addition/subtraction = \frac{n(n-1)}{2} + \frac{(n-1)n(n-2)}{6} = \frac{(n-1)n(n+1)}{6}$

Total Number of Operations

$$
\begin{aligned}
\#operations &= \#squareroot + \#division + \#multiplication + \#addition/subtraction \\
&= n + \frac{n(n-1)}{2} + \frac{(n-1)n(n+1)}{6} + \frac{(n-1)n(n+1)}{6} \\
&= \frac{2n^3 + 3n^2 + 3n}{6} \\
&= \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{2}n
\end{aligned}
$$

Hence, the computational complexity of Cholesky decomposition is n^3 with $\frac{1}{3}$ as the leading term i.e., $\mathcal{O}(\frac{1}{3}n^3)$.

Question 2

Use the following reference: https://math.nist.gov/ MatrixMarket/data/Harwell-Boeing/astroph/mcca.html to download an A matrix.

You can use https://docs.scipy.org/doc/scipy/reference/ generated/scipy.io.mmread.html to read this into numpy.

Deploy the Gauss-Siedel and Jacobi solution techniques for solving this system. Measure run-time for multiple true values of x (which will give you the right-hand sides) and provide statistical estimates.

Ans:

The Jacobi method diverged when trying to solve the matrix given in the question as it was not diagonally dominant.

We can see that the error continuously grows, finally reaching a retractable value (inf) from the figure 1

```
Warning: Matrix A is not diagonally dominant. May not converge.
Iteration:  0 Error:  145.50881861049712
Iteration:  1 Error:  221.45536258830202
Iteration:  2 Error:  41667.27756386591
Iteration:  3 Error:  71121.19895829348
Iteration:  4 Error:  13662260.368374465
Iteration:  5 Error:  24935850.437137652
Iteration:  6 Error:  4515166437.274913
Iteration:  7 Error:  8777225210.277962
Iteration:  8 Error:  1492681415648.1257
Iteration:  9 Error:  3086957099924.0464
Iteration:  10 Error:  493442023015745.7
Iteration:  11 Error:  1084076675550322.4
Iteration:  12 Error:  1.6310691579660902e+17
Iteration:  13 Error:  3.80031768651433e+17
Iteration:  14 Error:  5.391075073986205e+19
Iteration:  15 Error:  1.329673101738989e+20
Iteration:  16 Error:  1.7817430026710126e+22
Iteration:  17 Error:  4.643101519623211e+22
Iteration:  18 Error:  5.888183750118338e+24
Iteration:  19 Error:  1.6181124303079298e+25
Iteration:  20 Error:  1.945737212908681e+27
Iteration:  21 Error:  5.62810226009969e+27
Iteration:  22 Error:  6.429148784313358e+29
Iteration:  23 Error:  1.9538624739731247e+30
...
Iteration:  118 Error:  4.797304406509892e+150
Iteration:  119 Error:  5.602607089677272e+151
Iteration:  120 Error:  1.5783186170316101e+153
Iteration:  121 Error:  inf
```

Figure 1: Q2 Jacobi Divergence

Hence, a new diagonally dominant matrix A was generated to perform the required analysis further.

Statistic Analysis

Matrix Size: 100x100
Number of Samples: 100 (for statistical analysis)

Jacobi:
Mean run time: 1.9904895091056825
Standard deviation of run time: 0.13502804205569616
Median run time: 1.9777642488479614

```
Sample:  21
Sample:  22
Sample:  23
Sample:  24
...
Sample:  99
Mean run time for Jacobi:  1.9904895091056825
Standard deviation of run time for Jacobi:  0.13502804205569616
Median run time for Jacobi:  1.9777642488479614
```

Figure 2: Jacobi Method

Gauss-Siedel:
Mean run time: 0.06706570625305176
Standard deviation of run time: 0.009281696761748117
Median run time for Gauss-Siedel: 0.06645238399505615

```
Sample:  21
Sample:  22
Sample:  23
Sample:  24
...
Sample:  99
Mean run time for Gauss-Siedel:  0.06706570625305176
Standard deviation of run time for Gauss-Siedel:  0.009281696761748117
Median run time for Gauss-Siedel:  0.06645238399505615
```

Figure 3: Gauss-Siedel Method

Code: hw1.ipynb

Question 3

Use the following reference: https://math.nist.gov/ MatrixMarket/data/Harwell-Boeing/acoust/young1c.html to download another A matrix.

Deploy your constructed solvers and perform another analysis. Implement a Cholesky decomposition to solve this linear system and compare the runtimes with our iterative solvers (again statistically).

Ans:

Statistic Analysis

Matrix Size: 817x817
Number of Samples: 100 (for statistical analysis)

Jacobi:
Mean run time: 0.7540401315689087
Standard deviation of run time: 0.013828593532389468
Median run time: 0.751419186592102

```
Sample:  21
Sample:  22
Sample:  23
Sample:  24
...
Sample:  99
Mean run time for Jacobi:  0.7540401315689087
Standard deviation of run time for Jacobi:  0.013828593532389468
Median run time for Jacobi:  0.751419186592102
```

Figure 4: Jacobi Method

Gauss-Siedel:
Mean run time: 0.7407477712631225
Standard deviation of run time: 0.012842114190358838
Median run time for Gauss-Siedel: 0.7368589639663696

```
Sample:  21
Sample:  22
Sample:  23
Sample:  24
...
Sample:  99
Mean run time for Gauss-Siedel:  0.06706570625305176
Standard deviation of run time for Gauss-Siedel:  0.009281696761748117
Median run time for Gauss-Siedel:  0.06645238399505615
```

Figure 5: Gauss-Siedel Method

Cholesky decomposition:
Mean run time: 36.14571184158325
Standard deviation of run time: 0.32697275019986216
Median run time for Gauss-Siedel: 36.05303204059601

```
Sample:  21
Sample:  22
Sample:  23
Sample:  24
...
Sample:  99
Mean run time for Cholesky:  36.14571184158325
Standard deviation of run time for Cholesky:  0.32697275019986216
Median run time for Cholesky:  36.05303204059601
```

Figure 6: Cholesky Decomposition

Comparision
The above data shows that the Cholesky decomposition method took significantly more run time to solve the linear systems in comparison to the approximate methods like Jacobi and Gauss-Siedel.

For a big matrix of size 817x817, this was expected because the computational complexity of Cholesky decomposition is $\mathcal{O}(n^3)$ in comparison to the computational complexity of $\mathcal{O}(n^2)$ for Jacobi and Gauss-Siedel methods.

Code: hw1.ipynb