Justin Santer

Professor Yuefan Deng

AMS 326

March 27, 2020

Homework 2 Report

Problem 2.1

Description:

Generate two uniformly random (range from -1 to 1) 1024 x 1024 matrices. Multiply them once using the naïve algorithm, and then again using Strassen's algorithm. Estimate the operation count (multiplications + additions).

Algorithm Pseudocode:

```
NAIVE METHOD:

naive(A[n][n], B[n][n], C[n][n]):

   for i in rows:
      for j in columns:
         element = 0:

         for k from 0 to n-1:
            element += A[i][k] * B[k][j]

         C[i][j] = element

   return
```

STRASSEN METHOD:

```
strassen(A[n][n], B[n][n], C[n][n], levels):

   if n = 2 or levels = 0:
      naive(A, B, C)
      return

   next_size = n / 2

   A_11, A_12, A_21, A_22 = get_submatrices(A)
   B_11, B_12, B_21, B_22 = get_submatrices(B)

   strassen(A_11 + A_22, B_11 + B_22, M_1[next_size][next_size], levels-1)
   ...
   strassen(A_12 - A_22, B_21 + B_22, M_7[next_size][next_size], levels-1)

   C_11 = M_1 + M_4 - M_5 + M_7
   C_12 = M_3 + M_5
   C_21 = M_2 + M_4
   C_22 = M_1 - M_2 + M_3 + M_6

   C = combine_submatrices(C_11, C_12, C_21, C_22)

   return
```

Example of Program Results:

```
Generating first random 1024 x 1024 matrix...
Generating second random 1024 x 1024 matrix...

Multiplying the two matrices using Naive Multiplication...
Number of operations when using Naive Multiplication: 2147483648
Time taken using Naive Multiplication: 0.17 minutes

Multiplying the two matrices using Strassen's algorithm for three levels...
Number of operations when using Strassen's algorithm: 1438646272
Time taken using Strassen's algorithm: 0.03 minutes

The matrices are EQUAL

Generating first random 4096 x 4096 matrix...
Generating second random 4096 x 4096 matrix...

Multiplying the two matrices using Naive Multiplication...
Number of operations when using Naive Multiplication: 138877599744
Time taken using Naive Multiplication: 13.20 minutes

Multiplying the two matrices using Strassen's algorithm for three levels...
Number of operations when using Strassen's algorithm: 92073361408
Time taken using Strassen's algorithm: 4.43 minutes

The matrices are EQUAL
```

Conclusion and Comments:

As one can see, the Strassen method is tremendously more efficient than its Naïve counterpart. Its running time is shorter by magnitudes and the number of operations is quite lower. These results were expected. I worked with classmate Gino Giacoio in developing potential implementations to these algorithms.

Problem 2.2

Description:

Using the given data about COVID-19 patients and the rates of spread/recovery, generate data points pertaining to the number of patients over the course of the first 90 days. With these data points, use polynomial interpolation to get a curve for the spreading phase of the virus. Then, fit the recovery phase data into an exponential curve.

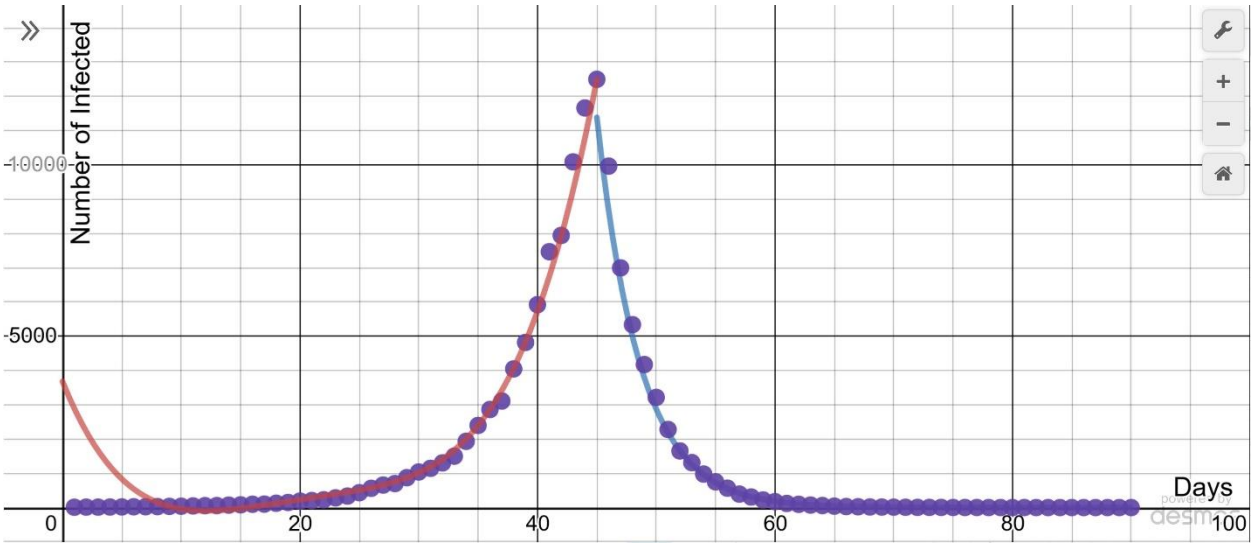
Algorithm Pseudocode:

```
main():

    // GENERATE VALUES USING NORMAL DISTRIBUTION
    data_points = generate_values()

    print(data_points)

    xvalues = get_x_values(data_points)
    yvalues = get_y_values(data_points)

    // POLYNOMIAL INTERPOLATION SECTION
    matrix = construct_system_matrix(xvalues, yvalues)
    coefficients = gaussian_elimination(matrix)
    print_equation(coefficients)

    // EXPONENTIAL LINE FITTING SECTION
    a, b = fit_line(data_points) // Converts exponential function to linear one and
                                    utilizes the Kenney and Keeping formulas to
                                    find alpha and beta
    print_equation(a, b)

    return
```

Example of Program Results:

```
First 45 Days Polynomial Interpolation:
f(x) = 3684.236398 + (-861.310753)x^1 + (69.083574)x^2 + (-2.307596)x^3 + (0.028767)x^4

Second 45 Days Line Fitting:
f(x) = 2731881213.419218exp(-0.275269x)
```

Here's the graph with each data point and the two curves using this example:



Conclusion and Comments:

The points in the example mostly line up well among the calculated curves, the exception being the beginning of the polynomial interpolation curve. I believe this is the case because the first point we use for the interpolation is at Day 9. If we were to be using another point at Day 1, the curve would be more accurate for the beginning days. Once again, I worked with Gino Giacoio in coming up with possible algorithms to use on this problem.