Lab 4, PHY407

Habiba Zaghloul, Ryan Cunningham

07 October 2022

Question 1a

The vector we get using Gaussian elimination is [2. -1. -2. 1.] and what we get when we do partial pivoting is [2. -1. -2. 1.]

Question 1b

Pseudocode

```
# Set N to 300 and create empty arrays to be used in the for loop
# Create a for loop that creates a random matrix and vector with dimensions NxN and N, respectively
# For each of the Gaussian, partial pivot and LU decomposition methods, record the time it takes to
# Find the vector x using these 3 methods
# Stop the time (each of them done separately), take the difference end-start and append it to a lis
# Find the true vector v by taking the dot product of the matrix and corresponding vector x
# Check the error by taking the mean of the absolute value of the difference of true vector v and ve
# Plot the errors of all 3 methods on one plot
# Plot the times of all 3 methods on a different plot
# Use plt.loglog in order to see all plots on the same scale to be able to easily compare
```

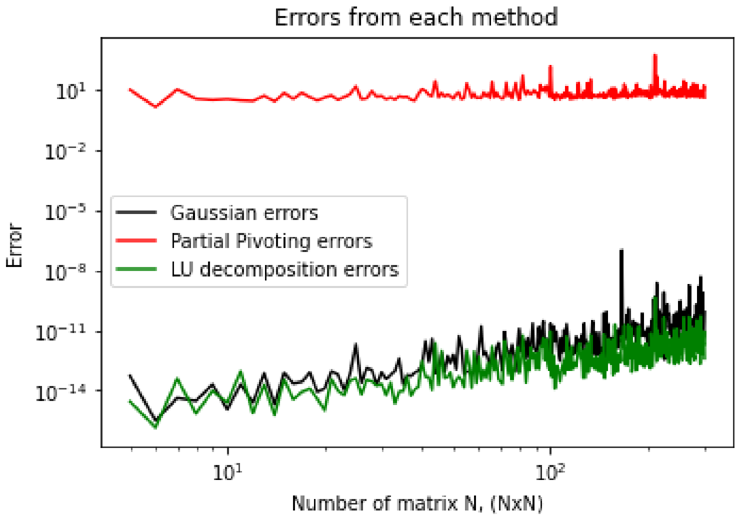

Figure 1: These errors show how accurate each method was at calculating the true values of the vector v in Ax=v

Figure 1 tells us that using LU decomposition is the safest as the errors are much smaller than the other two methods. Gaussian elimination is almost as accurate as LU decomposition. The partial pivoting method had the largest errors telling us that this method is not near as accurate as LU and Gaussian elimination.

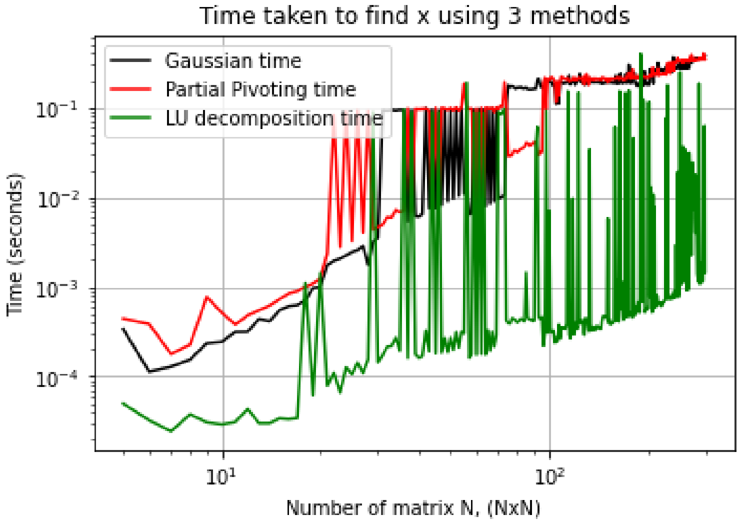

Figure 2: A plot showing how much time it took to find the vector x in Ax=v using different sizes of matrices and different methods

By looking at figure 2, we can see a general increasing trend where as the dimension of the matrix increases, so does the time. This is true for partial pivoting and Gaussian elimination. However, the LU decomposition times seem to be somewhat constant as N increases, compared to the other two methods. This plot satisfies the conclusion we made from figure 1 where LU decomposition was the best/most accurate method since its errors were the smallest and it takes the least amount of time to compute. The times for Gaussian elimination and partial pivoting are close, but looking at the accuracy in figure 1, we can safely say using Gaussian elimination is better as well as faster at finding our missing vector than partial pivoting.

Question 2c

Answers

The Eigenvalues for mmax = nmax = 10 are:

```
5.82044878 eV
11.13581093 eV
18.56054361 eV
28.96118885 eV
42.36858598 eV
58.7724212 eV
78.1672683 ev
100.55121491 eV
125.92200407 eV
154.40886831 eV
```

Question 2d

Answers

The First Ten Eigenvalues for mmax = nmax = 100 are:

```
5.82044878 eV
11.13581093 eV
18.56054361 eV
28.96118885 eV
42.36858598 eV
```

```
58.7724212 eV
78.1672683 ev
100.55121491 eV
125.92200407 eV
154.40886831 eV
```

It can be seen that that the first ten eigenvalues for the 100 x 100 matrix are the same as those for the 10 x 10 matrix. This makes sense as the value for the energy levels should not change for the wave function as the size of the matrix changes.

Question 2e

Pseudocode

```
# Code a function psi() that acts as the wave function equation
# Code a function psi_sq() that squares this wave function, to be used later
# Code a function simpson_method() that uses the Simpson Method of Integration
# Obtain the first three eigenvalues for the wave function using the H_matrix() function from previo
# Use the eigenvalues to obtain the first three eigenvectors
#  Calculate the wave function for each of the three eigenvectors
# Normalize the wave functions by finding the area under the curve (using the Simpson's Method)
# Plot a graph of psi**2 against x
```

Plot

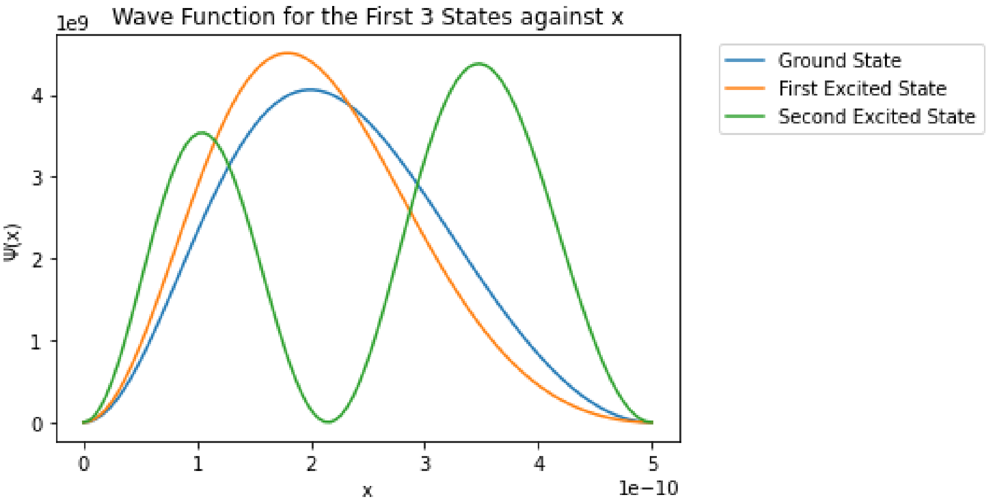

Figure 3: Plot for Question 2 Part e)

Question 3a

Pseudocode

```
# Code the function f(x) to return 1 - e^(-cx)
# Code a function relax_method() that uses the relax method for different values of c
# Use the relax_method() on 300 different values of c ranging from 0 to 3
# Plot the results of f(x) against c
```

Plot

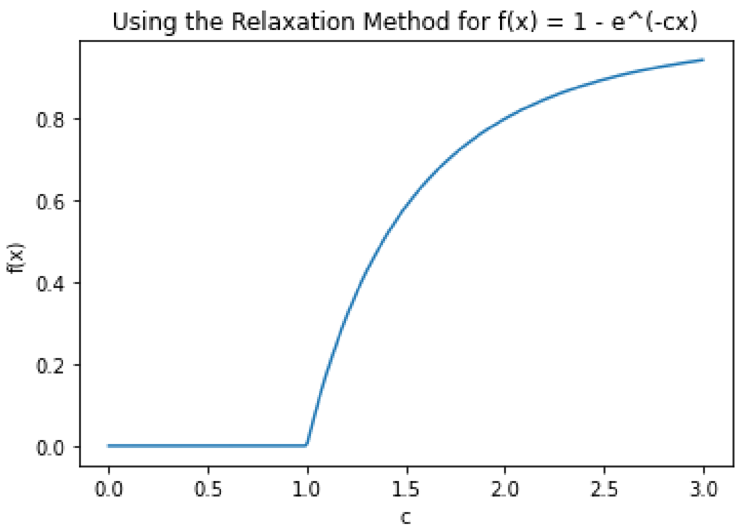

Figure 4:

Question 3b

Pseudocode

```
# Edit the relax_method() function to count how many iterations it completes when it is called
# Create a new function overrelax_method() that has two arguments, c and omega
# Compare how many iterations are needed for each of the functions to return a value for f(x) with t
# Try different omega values to see if the number of iterations can be reduced still
```

Answers

Result from using the relaxation method without over relaxation 0.7968126311118457 with an number of iterations of 14. Using over relaxation with an omega of 0.5 gives the result 0.7968122028149421 with a number of iterations of 7. This suggests that the over relaxation method does in fact reduce the number of iterations by around half. The lowest number of iterations I was able to achieve was 5, with an omega value of 0.65, giving a result of 0.7968121158403483.

Question 3c

Pseudocode

```
# Code a function wien() that acts as the Wien’s Displacement Constant Equation
# Code a function binary() that performs a binary search method on given function, to a specified pr
# Obtain an x value for the Wien displacement constant to an accuracy of 1e-8
# Calculate an estimation for the T_sun by using the given constants and the obtained x value
```

Answers

The x value was found to be 4.965113581866026. Using this to calculate the estimation for the Temperature of the Surface of the Sun, T was found to be equal to 5772.45 K. The true value for the temperature of the surface of the sun is around 5778 K, so our estimation was very close.