PHY407F: Explanatory Notes For Lab 4

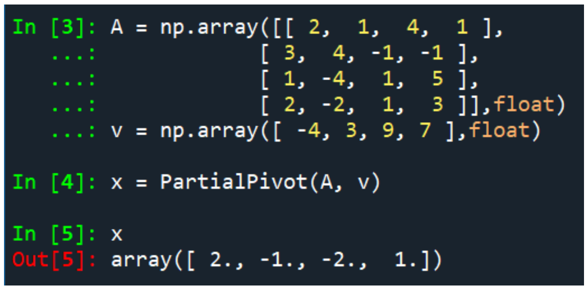
Idil Yaktubay and Souren Salehi

7 October 2022

1 Question 1 - by Idil Yaktubay

(a) We have modified the module `SolveLinear.py` to implement a function, called `PartialPivot()`, that incorporates partial pivoting to the Gaussian elimination method to solve linear systems. To test this function, we have verified that the numerical solution `x = [w, x, y, z]` to equation 1 -taken from equation 6.2 in Newman's *Computational Physics*- is indeed given by $w = 2$, $x = -1$, $y = -2$, $z = 1$. Figure 1 depicts the solution output from our code.

$$\mathbf{Ax} = \begin{pmatrix} 2 & 1 & 4 & 1 \\ 3 & 4 & -1 & -1 \\ 1 & -4 & 1 & 5 \\ 2 & -2 & 1 & 3 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -4 \\ 3 \\ 9 \\ 7 \end{pmatrix} = \mathbf{v} \tag{1}$$



```
In [3]: A = np.array([[ 2,  1,  4,  1 ],
   ...:               [ 3,  4, -1, -1 ],
   ...:               [ 1, -4,  1,  5 ],
   ...:               [ 2, -2,  1,  3 ]],float)
   ...: v = np.array([ -4, 3, 9, 7 ],float)

In [4]: x = PartialPivot(A, v)

In [5]: x
Out[5]: array([ 2., -1., -2.,  1.])
```

Figure 1: Output of the code that calculates the solution to equation 1 using the `PartialPivot()` function. This verifies that `PartialPivot()` works as intended.

(b) To test the accuracy and timing of the Gaussian elimination, partial pivoting, and LU decomposition approaches, we have written a program that creates random N-dimensional linear systems for $N = 5, 6, 7, ..., 300$ and finds the corresponding solutions `x` using the three different methods for each linear system. These linear systems are random in the sense that the entries of matrix $\mathbf{A}$ and vector $\mathbf{v}$ are randomly chosen for each value of N. Further, we have calculated the error associated with each solution `x` as the average of the absolute value of the difference of the dot product of `A` and `x` and the vector `v`, or `err = mean(abs(v-dot(A, x)))`. Lastly, we have recorded the time it takes for the computer to solve each linear system for the three different methods.

Figure 2 depicts log-log plots of error versus N for each of the three methods. Evidently, although the three methods are mathematically equivalent, the overall error associated with the Gaussian elimination method is significantly higher than the errors associated with the partial pivoting and LU decomposition methods. We also see that the accuracies of the partial pivoting and the LU decomposition methods are so close that the two plots are barely distinguishable on Figure 2. To better distinguish them, we have noted that the average errors of the partial pivoting and the LU decomposition methods are $\approx 1.6 \times 10^{-12}$ and $\approx 2.1 \times 10^{-12}$, respectively (for the random set of matrices we generated). Further, as shown on Figure 3, the

LU decomposition method is the most efficient one, whereas the Gaussian elimination and partial pivoting methods are much slower, as expected.

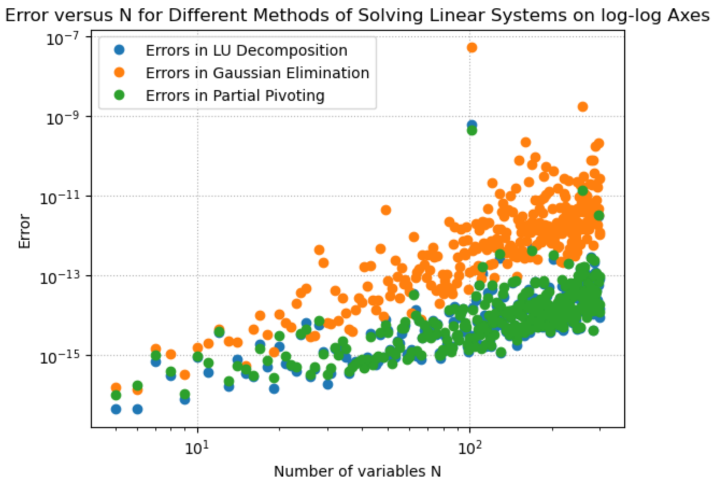


Figure 2: A log-log plot of error in the Gaussian elimination, partial pivoting, and LU decomposition methods versus the number of variables in the linear system. We generated a random linear system for each value of N, and the same linear system was solved using the three methods.

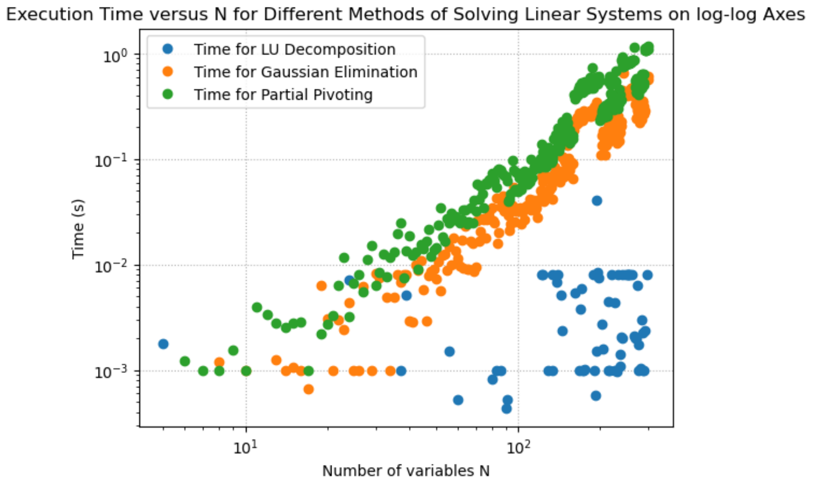


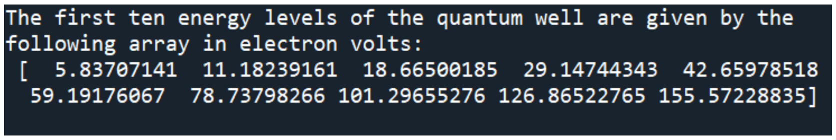
Figure 3: A log-log plot of execution time for the Gaussian elimination, partial pivoting, and LU decomposition methods versus the number of variables in the linear system. Again, the linear systems were generated randomly for each N.

2 Question 2 - by Idil Yaktubay

(a), (b), (c) In this question, we have translated Schrodinger's equation for an electron inside an asymmetric quantum well to an eigenvalue problem given by equation 2, where **H** is the Hamiltonian matrix, E is the energy of the system, and ψ is the wavefunction corresponding to the energy E. The potential inside the well is given to us by $V(x) = \frac{ax}{L}$, where $L = 5\mathring{A}$ and $a = 10eV$. To find the first ten energy levels of the system, we have written a Python program that calculates the 10x10 portion of the infinite matrix **H** using equation 3 and finds the eigenvalues of this matrix. The first ten energy levels of the system are given by Figure 4.

$$\mathbf{H}\psi = E\psi \tag{2}$$

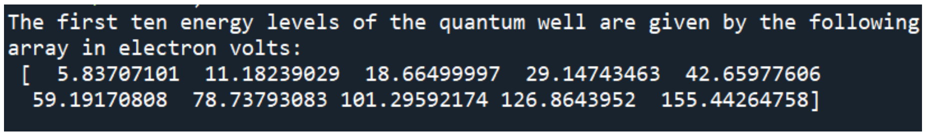
$$H_{mn} = \begin{cases} 0 & \text{if } m \neq n \text{ and both even/odd} \\ -\frac{8amn}{\pi^2(m^2-n^2)^2} & \text{if } m \neq n \text{ one even, one odd, and} \\ \frac{1}{2}a + \frac{\pi^2\hbar^2 m^2}{2ML^2} & \text{if } m = n. \end{cases} \tag{3}$$



```
The first ten energy levels of the quantum well are given by the
following array in electron volts:
 [  5.83707141  11.18239161  18.66500185  29.14744343  42.65978518
  59.19176067  78.73798266 101.29655276 126.86522765 155.57228835]
```

Figure 4: The first ten energy levels of an electron inside a one-dimensional asymmetric quantum well found by calculating the eigenvalues to the 10x10 Hamiltonian matrix.

(d) To compare the previously calculated values of the first ten energy levels of the electron to the results of another set of calculations, we have modified our program to use a 100x100 matrix instead. Figure 5 depicts the first ten energy levels of the electron calculated with this method. As we can see from Figures 4 and 5, the difference between the values obtained with the 10x10 matrix and the values obtained with the 100x100 matrix gets larger as the energy levels increase. Here, the higher energies we have calculated using the 10x10 matrix are less accurate than those calculated using the 100x100 matrix because these energy levels approach the highest possible energies for the 10x10 matrix. In other words, a larger basis size is needed for a higher accuracy in larger eigenvalues.



```
The first ten energy levels of the quantum well are given by the following
array in electron volts:
 [  5.83707101  11.18239029  18.66499997  29.14743463  42.65977606
  59.19170808  78.73793083 101.29592174 126.8643952  155.44264758]
```

Figure 5: The first ten energy levels of an electron inside a one-dimensional asymmetric quantum well found by calculating the eigenvalues to the 100x100 Hamiltonian matrix.

(e) Lastly, we have modified our program to calculate the wavefunctions for the ground state and the first two excited states of the electron inside the well. The wavefunction of the electron at a specific energy level is given by equation 4, where the values of ψ_n are the elements of the eigenvector corresponding to this energy level. To construct the wavefunction for each energy level, we have calculated the corresponding eigenvectors. Then, we have normalized each of the three wavefunctions by finding the normalization factors $\sqrt{A}$ given by $A = \int_0^L |\psi(x)|^2 dx$. We have computed the integrals for each normalization factor using Simpson's Rule. Figure 6 depicts the probability densities, $|\psi(x)|^2$, of the three energy levels.

$$\psi(x) = \sum_{n=1}^{\infty} \psi_n sin\frac{\pi n x}{L} \tag{4}$$

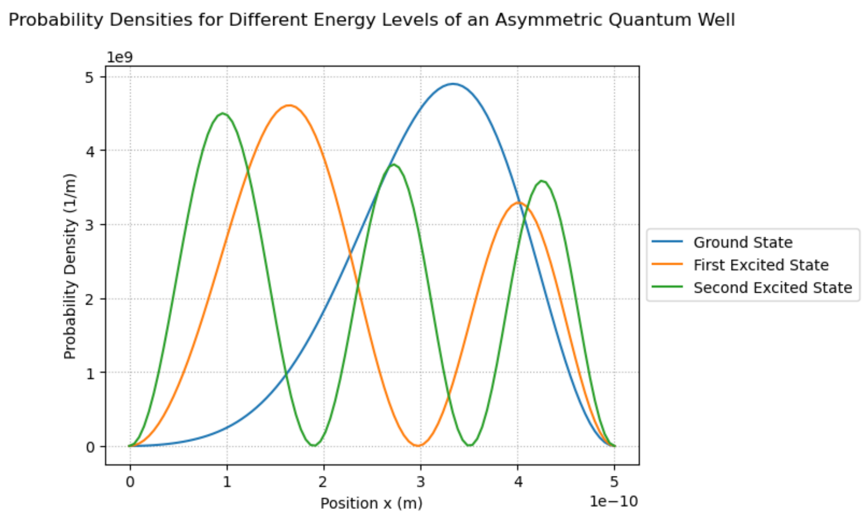


Figure 6: Probability densities for the ground and the first and second excited states of an electron inside an asymmetric quantum well. The wavefunctions corresponding to these probably densities have been normalized.

3 Question 3 - by Souren Salehi

(a), (b) For this question, we have found the solution to the function $x = 1 - e^{-cx}$ for $c = 2$ using the relaxation (question 6.10) and the over-relaxation (question 6.11) methods. Further, we have shown, on Figure 7, how the solution found with the relaxation method varies with a range of c values from 0 to 3. The relaxation method took 14 iterations, whereas the over-relaxation method only took 4 iterations with $\omega = 0.5$. This means that in our case, the over-relaxation method was much more efficient than the relaxation method. Figure 8 shows the results of the relavant calculations. For 6.11(d), if we have a case where the over-relaxation method causes the solution to diverge, we might be able to use a relaxation factor $\omega < 0$ to slow the iterations and cause the answer to converge faster.

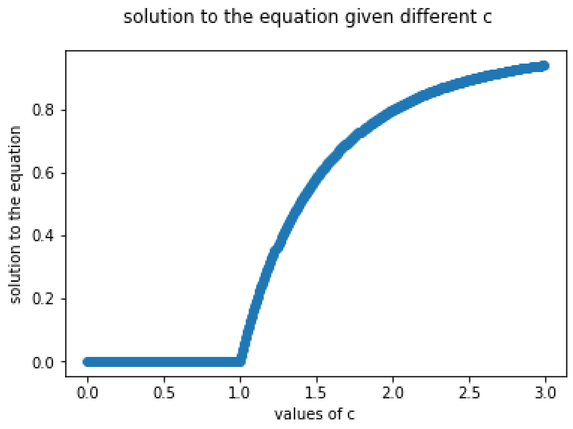
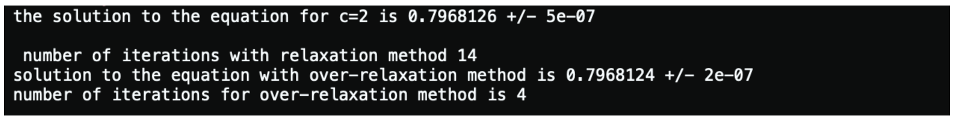


Figure 7: A plot of relaxation solutions to $x = 1 - e^{-cx}$ versus c. As we can see, the solutions start increasing after the threshold value o $c = 1$.

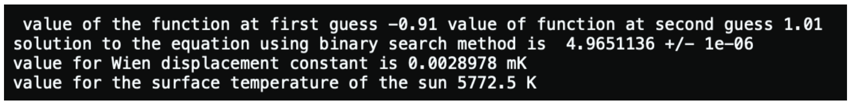


```
the solution to the equation for c=2 is 0.7968126 +/- 5e-07

 number of iterations with relaxation method 14
solution to the equation with over-relaxation method is 0.7968124 +/- 2e-07
number of iterations for over-relaxation method is 4
```

Figure 8: Python code output for calculations from questions 6.10 and 6.11 in the textbook.

(c) For this part of the question, we have found the solution to the equation $5e^{-x} + x - 5 = 0$ using the binary search method. With this solution, we have calculated the Wien displacement constant, as well as the surface temperature of the sun. Figure 7 depicts the code output that shows these values.



```
 value of the function at first guess -0.91 value of function at second guess 1.01
solution to the equation using binary search method is  4.9651136 +/- 1e-06
value for Wien displacement constant is 0.0028978 mK
value for the surface temperature of the sun 5772.5 K
```

Figure 9: Python code output for calculations from question 6.13.