PHY407: Lab 04

Milica Ivetic & Madeline Nardin
milica.ivetic@mail.utoronto.ca
maddy.nardin@mail.utoronto.ca

October 7, 2022

Questions 1 & 3 done by Milica Ivetic
Question 2 done by Madeline Nardin

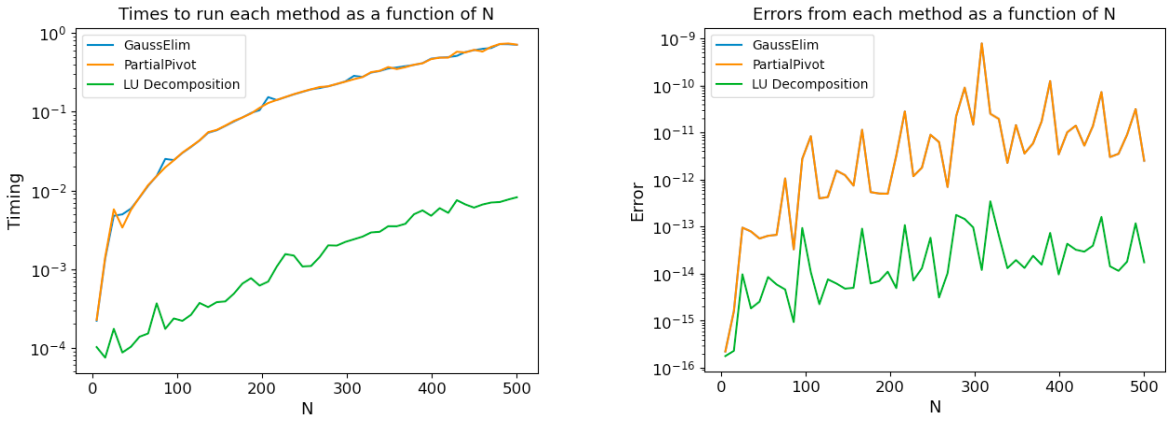
1 Question 1

1(a)

See printed output in Lab04_Q1.py.

1(b)

See code and pseudocode in Lab04_Q1.py and SolveLinear.py.



(a) Plot showing timings as a function of N for the different methods considered.

(b) Plot showing errors as a function of N for the different methods considered.

Figure 1: Plots showing timings and errors as a function of N for Gaussian elimination, partial pivoting, and LU decomposition methods.

Figure 1 shows the plots of the timings and errors as a function of N for Gaussian elimination, partial pivoting, and LU decomposition methods. The timings and errors were

calculated for 50 values of N between 5 and 500. Note that the scale for the y-axes for both plots is logarithmic.

In Figure 1a, we see that the LU decomposition method for any value of N takes less time to compute than both of the other methods. For larger N the difference in timings ranges over a few orders of magnitude. As for Gaussian elimination and partial pivoting, they take roughly the same amount of time to compute. There are points when one takes slightly longer than the other for a certain value of N, however there is no obvious pattern.

In Figure 1b, we see that LU decomposition is the most accurate method out of the three, by several orders of magnitude. Note that the accuracy is about the same between all three for small N. Also, for every value of N, Gaussian elimination and partial pivoting completely overlap. This overlap is expected since the two methods are the same calculation, the only difference is the swapping of rows when necessary for partial pivoting.

2 Question 2

*Note parts 2(a) and 2(b) have nothing to submit

2(c)

See L04_Q2.py for code that calculates values elements of Hamiltonian matrix H according to eq.5 in assignment sheet. We found the solution for the first ten energy levels of the quantum well of a 10×10 matrix H. The printed output is shown in table 1.

State	Energy (eV)
0	5.83684005
1	11.18197256
2	18.66434608
3	29.1464572
4	42.65837215
5	59.18982506
6	78.73542897
7	101.2932856
8	126.86115177
9	155.56730124

Table 1: First ten energy levels of a 10×10 matrix H

2(d)

The same code used in part 2(c) was used to calculate the solution for the first ten energy levels of the quantum well of a 100×100 matrix H.The printed output is shown in table 2.

State	Energy (eV)
0	5.83683965
1	11.18197123
2	18.66434421
3	29.1464484
4	42.65836303
5	59.18977247
6	78.73537714
7	101.29265462
8	126.86031937
9	155.4376678

Table 2: First ten energy levels of a 100×100 matrix H

We note that H matrices of 100×100 and 10×10 are accurate (with respect to each other) to 4 decimal places.

2(e)

The function was computed via the equation in the text utilizing the eigenvectors from part 2(d) to represent ψ_n. The probability density was computed given $|\psi(x)|^2 = \psi(x)\psi(x)^*$. To check if $|\psi(x)|^2$ satisfied the condition $\int_0^L |\psi(x)|^2 dx = 1$ given in the question, the integral was computed via Simpsons method with the function written in Lab 02. We found the integral to be equal to $\approx 3.08e-11$ at the ground state which is evidently not equal to 1. Figure 2 displays the normalized probability density $|\psi(x)|^2$ in the ground state and first two excited states of the well.

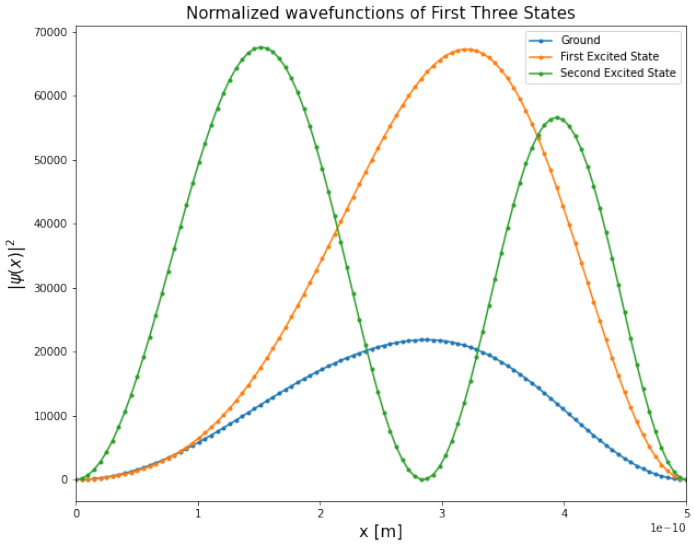


Figure 2: Normalized probability density $|\psi(x)|^2$ in the ground state and first two excited states of the quantum well.

3 Question 3

3(a)

6.10a

See code and pseudocode in Lab04_Q3.py.

6.10b

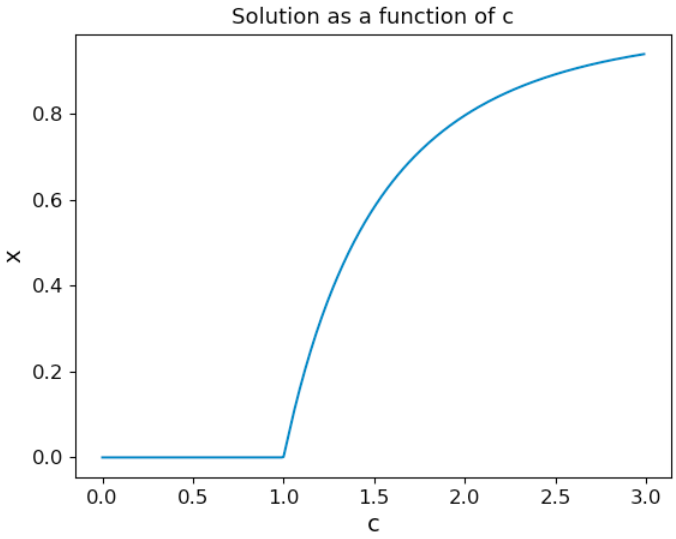


Figure 3: Solutions to equation as a function of c.

Figure 3 shows the solutions to the equation considered in this question as a function of c. The equation being considered was $x = 1 - e^{-cx}$. We considered a range of c from 0 to 3 in steps of 0.01. At c = 1 we see a very obvious transition from $x = 0$ to a regime of non-zero x.

3(b)

6.11b

See printout in Lab04_Q3.py.

6.11c

See printouts, pseudocode and code in Lab04_Q3.py. Using the relaxation method, it takes 16 iterations for the solution to converge to 0.7968118244957711. Using overrelaxation it takes 8 iterations to converge to 0.7968120970781651, so half the amount compared to the first method. This was achieved with the value of $\omega = 0.5$.

6.11d

Yes, there are circumstances under which using a value $\omega < 0$ would help us find a solution faster than we can with the ordinary relaxation method. These circumstances depend on the function we are considering. For example, consider the equation $x = e^{1-x^2}$. Using the ordinary relaxation method, it takes 5200968 iterations to converge. Whereas for For ω = -0.5, it takes 35 iterations to converge. Note that in both cases the initial guess was 0.5. Code for this is in Lab04_Q3.py, but the lines for the output are commented out.

3(c)

6.13b

See printouts, pseudocode, and code in Lab04_Q3.py. Using the binary search method, the displacement constant was found to be 0.002899778142984342.

6.13c

See printouts, pseudocode, and code in Lab04_Q3.py. From the given equations and value for λ, we find that the estimated surface temperature of the Sun is 5776.4504840325535 K. This is very close to the theoretical value of 5778 K. How the estimated value was obtained is explained in the comments of the code.