PHY407 Lab04

(Yonatan Eyob Q2, Q3c), (Kivanc Aykac Q1, Q3a-b)
Student Numbers: 1004253309, 1004326222

October 9^{th}, 2020

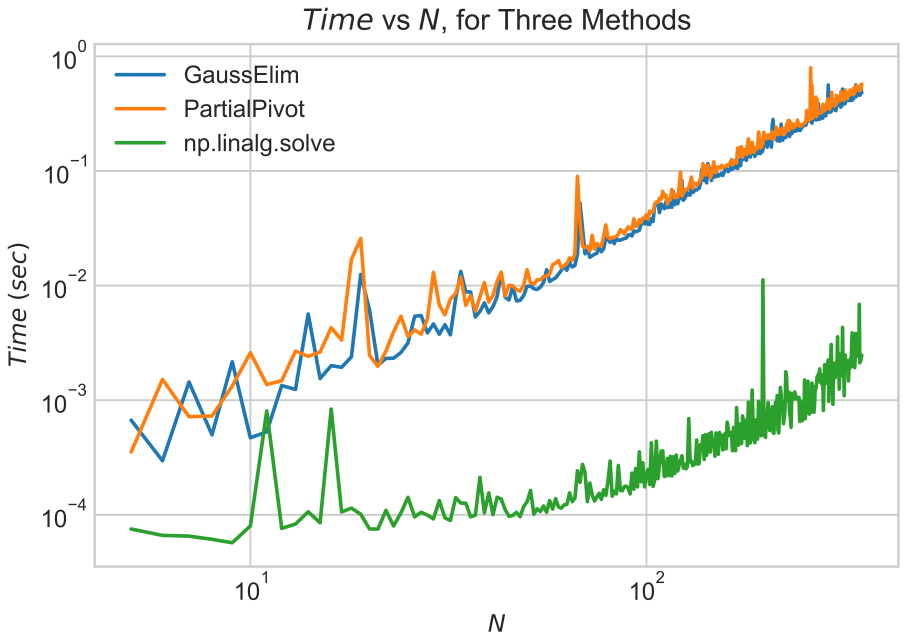
1 Solving linear systems

1.a

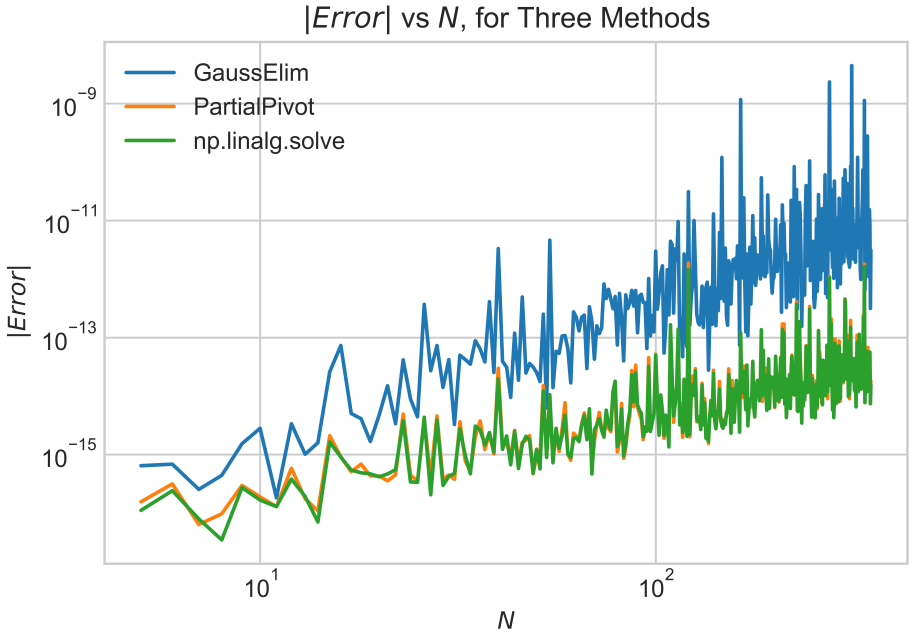
Nothing to submit.

1.b

The goal of this part is to compare error and time measures of three different methods of finding the solutions of a randomly generated simultaneous linear equations. The methods are Gaussian elimination, partial pivoting, LU decomposition. For the Gaussian elimination method the code that was received from Prof. Grisouard is going to be used. For partial pivoting, an algorithm that is very similar to the one that is used for Gaussian elimination method will bu used, but it will be slightly adjusted inside the loop so that the pivoting is done for every row after each elimination. And for the last method, LU decomposition, the algorithm is not going to be typed by hand but instead `numpy.linalg.solve` function will be used, which does the same thing as the textbook recommends.
To achieve this goal, a for range of `N` values were created from 5 to 350 so that the time and error calculations can be plotted. And for each N value a new matrix, `A`, and vector, `v`, was generated. For each `N` value, only a one simultaneous linear equations were generated so that the methods can be tried on the *same* system and can be compared later on with graphs.
The plots show the results below.



Time vs N in logarithmic scale.



$|Error|$ vs N in logarithmic scale.

As can be seen on the first plot, partial pivoting and Gaussian elimination methods follow more or less the same time line. Partial pivoting method is very slightly above Gaussian elimination. This was expected since partial pivoting is basically an addition of 9 lines that include a for loop, but this doesn't have a significant effect since the loop is not a nested one and is pretty basic. The more interesting part is with the time plot of `np.linalg.solve` as its values are significantly lower than of the other two methods. This makes `np.linalg.solve` preferable for the calculations of large systems.
The second plot, depicts the magnitudes of the errors in the methods. When calculating this error value, $v_{sol} = dot(A, x)$ was used as the reference value and $err = mean(abs(v - v_{sol}))$ was the formula for $|Error|$. As can be seen, Gaussian elimination starts of slightly above the other two methods and its $|Error|$ magnitude increases more than the others as N increases and the other two methods' errors increase in similar amounts. The difference between Gaussian elimination and the other two methods is around two order of magnitude by the last value of N, $\sim 10^{-11}$ and $\sim 10^{-13}$.

1.c

In this part goal is to use the skills (partial pivoting) gained from 1.b on a Kirchhoff's Law problem. First part deals with finding the solutions for x_1, x_2, x_3 in the system (1).

$$\begin{aligned} \left(\frac{1}{R_1} + \frac{1}{R_4} + i\omega C_1\right) x_1 - i\omega C_1 x_2 &= \frac{x_+}{R_1} \\ -i\omega C_1 x_1 + \left(\frac{1}{R_2} + \frac{1}{R_5} + i\omega C_1 + i\omega C_2\right) x_2 - \mathrm{i}\omega C_2 x_3 &= \frac{x_+}{R_2} \\ -i\omega C_2 x_2 + \left(\frac{1}{R_3} + \frac{1}{R_6} + i\omega C_2\right) x_3 &= \frac{x_+}{R_3} \end{aligned} \tag{1}$$

With the values (equation 2),

$$\begin{aligned} R_1 = R_3 = R_5 &= 1\mathrm{k}\Omega \\ R_2 = R_4 = R_6 &= 2\mathrm{k}\Omega \\ C_1 = 1\mu\mathrm{F}, \quad C_2 &= 0.5\mu\mathrm{F} \\ x_+ = 3\mathrm{V}, \quad \omega &= 1000\mathrm{rads}^{-1} \end{aligned} \tag{2}$$

With the voltage definition as in 3,

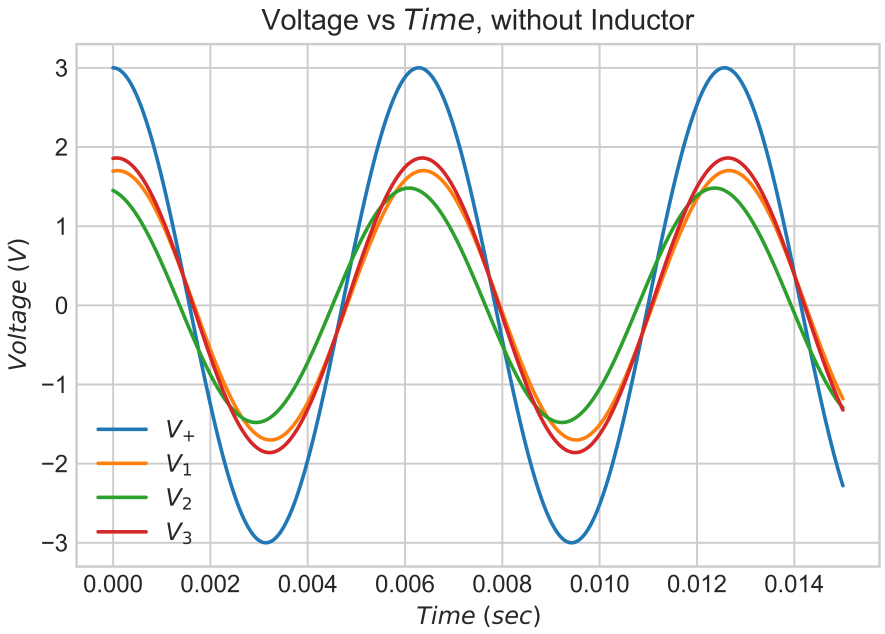
$$\begin{aligned} V_\sigma &= x_\sigma e^{i\omega t} \\ \sigma &= \{+, 1, 2, 3\} \end{aligned} \tag{3}$$

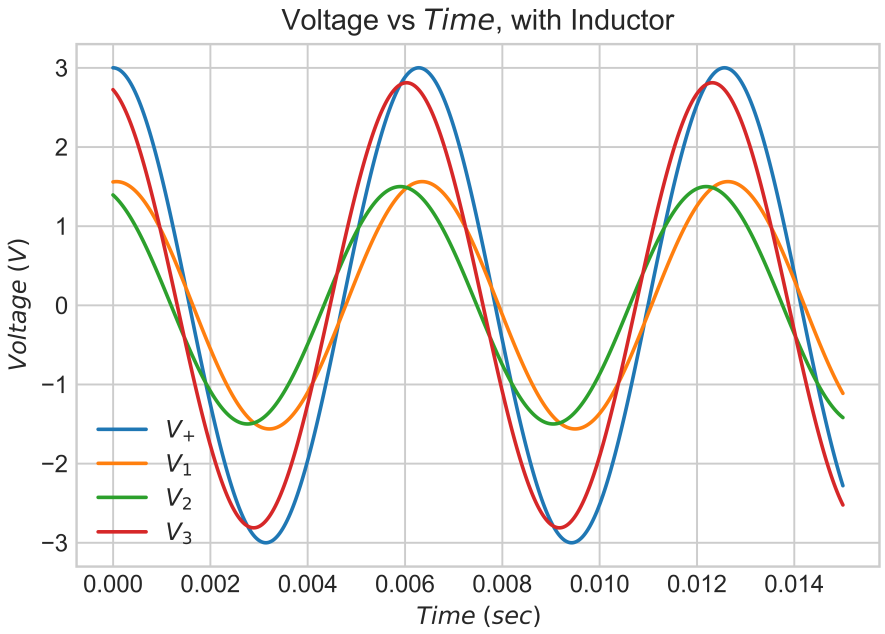
And in the second part of the problem, the R_6 term is going to be replaced by $i * \omega * L$ where L term is going to be taken as: $L = R_6/\omega = 2H$. Then systems without and with the inductor are going to be compared.
The printed outputs for $t = 0$ are as follows:

```
Without Inductor
|V_1|: 1.701439 V,  Phase_1 -5.469095 degrees
|V_2|: 1.480605 V,  Phase_2 11.583419 degrees
|V_3|: 1.860769 V,  Phase_3 -4.164673 degrees

With Inductor
|V_1|: 1.562118 V,  Phase_1 -4.025909 degrees
|V_2|: 1.499429 V,  Phase_2 21.639283 degrees
|V_3|: 2.811276 V,  Phase_3 14.352480 degrees
```

As can be seen, $|V_1|$ affected from the inductor placement much more than $|V_2|$ as R_1 resistance is smaller than R_2 resistance. Greatest change is in $|V_3|$ as that was the place where inductor was placed. At the same time the replacement of R_6 with iR_6 created a phase change too causing voltage to differ.
Below are the graphs for this part.





As can be seen from the graphs replacing R_6 with an inductor has a huge effect on voltage (real part). Although magnitude did not change the spread of the real and imaginary parts of the voltage changed as the phase changed. And this is visible from the unchanging V_+ plot and the changing $V_{1,2,3}$ plots.

2 Asymmetric quantum well

in this section we will observe the 10 eigenvalues of a 10x10 Hamiltonian matrix and compare them to the first 10 eigenvalues of a 100x100 Hamiltonian matrix. Then we will analyze the probability density graphs of the ground state and the first two excited states.

2.a

[Nothing to submit]

2.b

[Nothing to submit]

2.c

first we will list the eigenvalues found in the 10x10 Hamiltonian matrix:

10x10 Hamiltonian matrix	
Energy Level	Energy Eigenvalues
1	5.836342710505357eV
2	11.180995533911492eV
3	18.662671474681883eV
4	29.143804203246102eV
5	42.65445877230795eV
6	59.18437004965767eV
7	78.7281514610879eV
8	101.28390485767937eV
9	126.84938719810758eV
10	155.55286348370188eV

The Energy eigenvalues increase as the energy levels increase as expected.

2.d

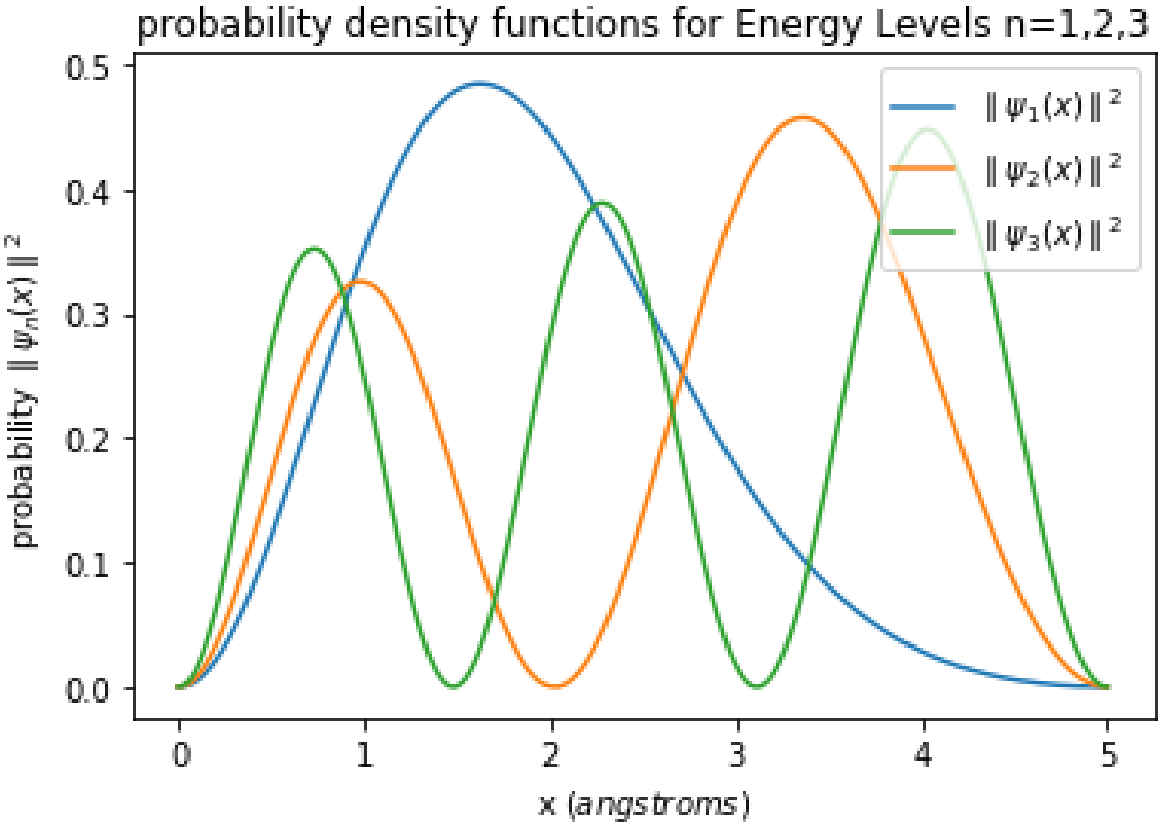
Now we will list the first 10 eigenvalues found in the 100x100 Hamiltonian matrix:

100x100 Hamiltonian matrix	
Energy Level	Energy Eigenvalues
1	5.836342309816193eV
2	11.180994210619549eV
3	18.662669603635223eV
4	29.14379540344378eV
5	42.65444965203365eV
6	59.18431746576108eV
7	78.72809963306105eV
8	101.28327391546033eV
9	126.84855484969455eV
10	155.42323889865548eV

The first ten eigenvalues in the 100x100 Hamiltonian matrix are very similar to the ten eigenvalues from the 10x10 Hamiltonian matrix but not exactly the same (they should be the same). This concludes that the accuracy of the eigenvalues found from the 10x10 and 100x100 matrices aren't perfectly accurate but they are a decent estimation.

2.e

Now we will observe the probability density function of the ground state ($\psi_1(x)$) and the first 2 excited states ($\psi_2(x)$, $\psi_3(x)$):



The number of peaks in the probability density functions seem to be equal to the energy level it is representing ($\| \psi_1(x) \|^2$ hasone peak, $\| \psi_2(x) \|^2$ has two peaks, $\| \psi_3(x) \|^2$ has three peaks). Unlike the infinite square well, each probability function is slanted. This is because the potential on the bottom of the well isn't constant. The probability density function tends to be lower (or dampens more) where the potential is high and tends to be high (or dampens less) when the potential is low which explains the apparent slanted sinusoidal pattern. For these probability functions, $\int_0^L \| \psi_n(x) \|^2 \, dx = 1$ is true so that proves that they are normalized.

3 Solving non-linear equations

3.a

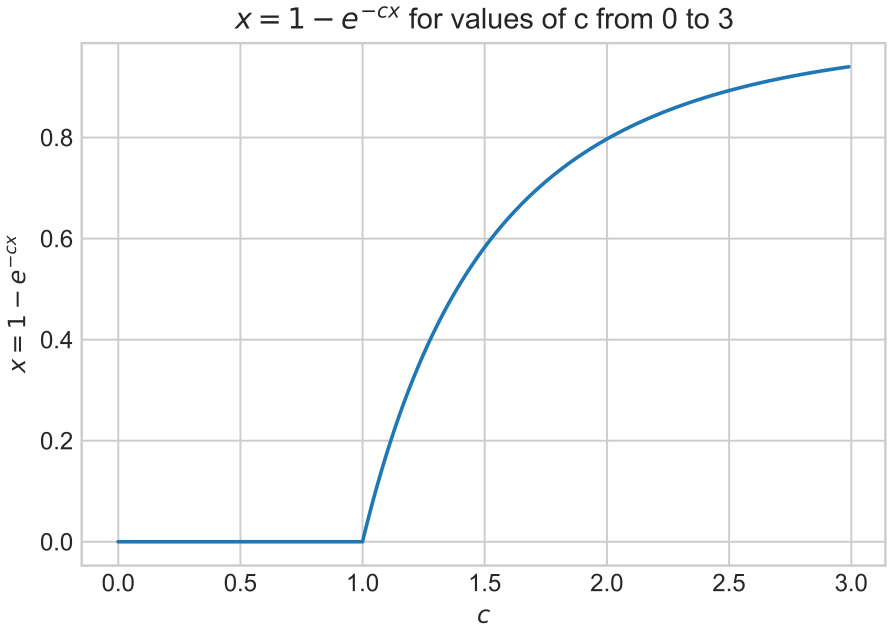
In this part goal is to apply the relaxation method to an accuracy of 10^{-6} using $x = 1 - e^{-cx}$.
First, c is taken to be 2. The printed output of the result is as follows:

```
For c=2, solution is 0.79681182, error is -0.00000075
```

As can be seen, Error is calculated as it is stated for the relaxation method on the textbook:

$$\epsilon' \simeq \frac{x - x'}{1 - 1/f'(x)}$$

Next, the relaxation method will be operated for the above function for different values of c, from 0 to 3 with 0.01 `dstep`.



As can be seen, the nonzero regime of x is starting strictly at $c = 1.0$.

3.b

In this part, the relaxation method skills gained from 3.a will be changed a little so that the function can output also the value for the number of iterations it took algorithm to calculate the function to an accuracy of 10^{-6} for a given c value. Then the method will be changed a little so that it can operate "over-relaxation method".

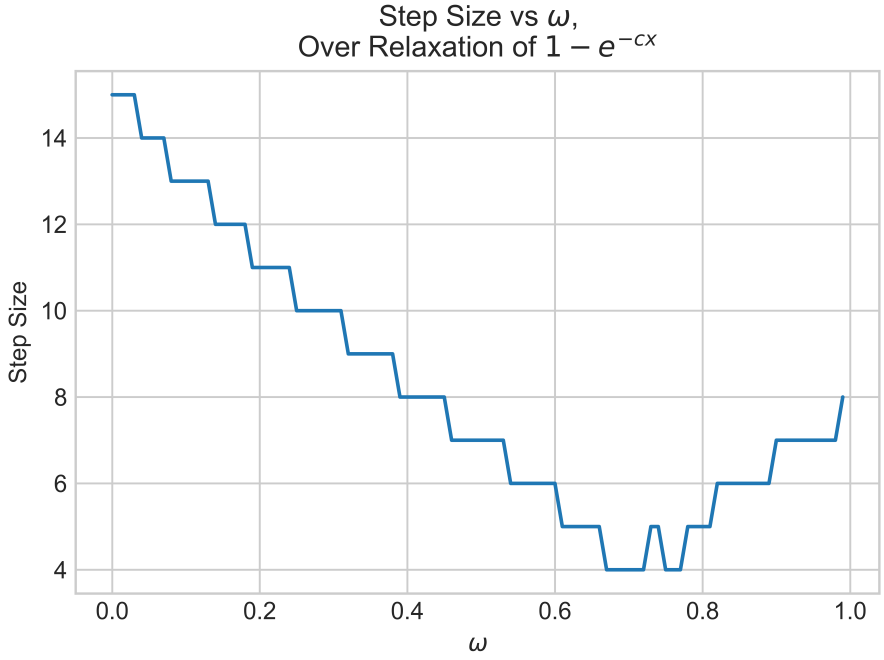
Before, when doing the relaxation method, the way to find the next x value was by putting the previous x value into the RHS of the equation as the equation is an autonomous one. Now this will be changed as follows:

$$x' = (1+\omega)f(x) - \omega x$$

The ω value will be varied and the most efficient (the one that converges the fastest). So, if $\omega = 0$ the over-relaxation does the same thing as relaxation.
First, for comparison reasons with 3.a, setting $c = 2$ and applying the relaxation method:

```
For c=2, solution is 0.79681182,
error is -0.00000075,iteration count is 16
```

To compare over-relaxation method with this, the method should be operated for a range of ω values:



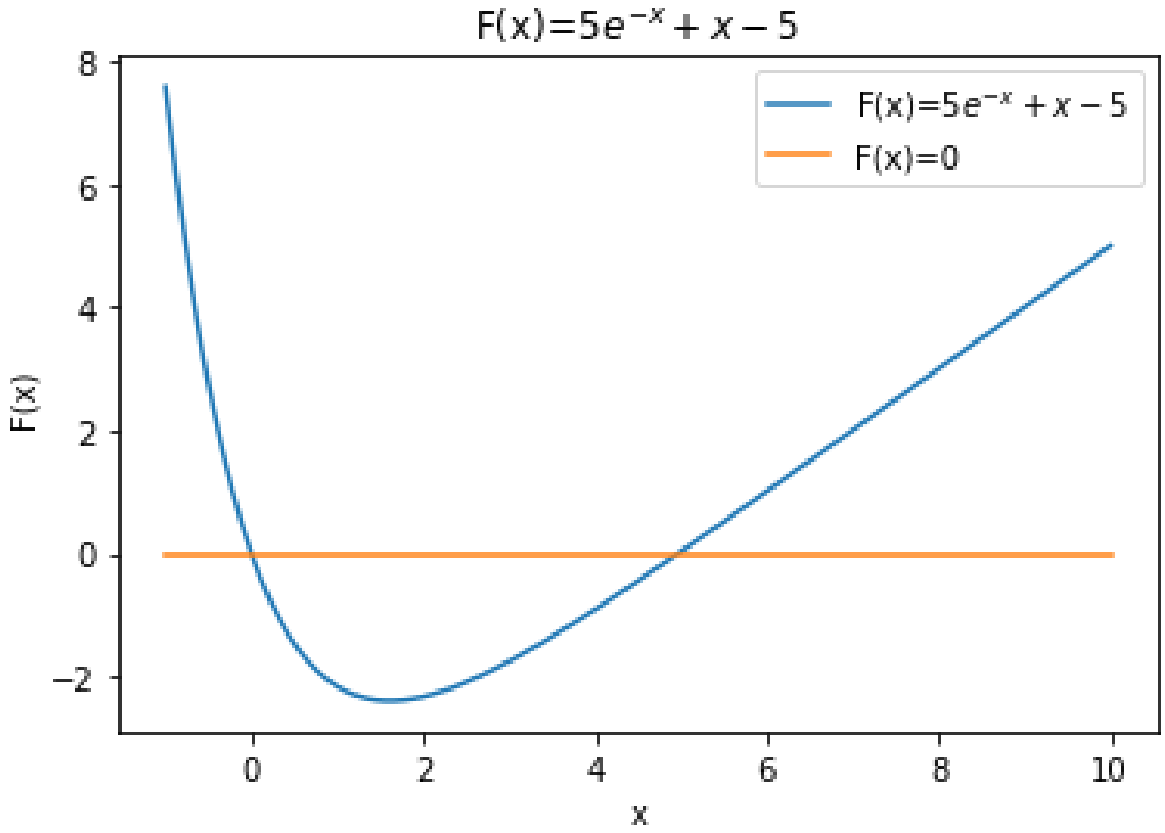
From this plot we see that for some values of ω, it took only 4 iterations to reach to the solution with an accuracy of at least 10^{-6}! Whereas for the relaxation method it took 16 iterations to reach to the same result.
Just to make sure that the value output at one of these efficient ω values give the same result as the relaxation method, let's take a sample. Take $\omega = 0.75$ where iteration count is at 4 steps:

```
For omega 0.75 and number of iterations 4
x value is 0.79681116 with error 0.00000097
```

The ouput of the two methods are in disagreement at the seventh digit after the decimal point but this is nothing of concern since we have a tolerance accuracy of 10^{-6}.
Also, one would expect to find a solution faster with $\omega < 0$ for a function with a negative gradient. In the above examples, the derivative of the function was ce^{-cx} where c was taken to be 2, so the derivative was already giving positive values for any x so testing with $\omega > 0$ values was not a topic of concern.

3.c

the function we wish to find the (non-obvious) root of is F(x)=$5e^{-x} + x - 5$. The roots occur when the two lines below intersect:



We will use three methods to find the root of this function; The Bisection method, Newton's Method, and the relaxation method.
looking at the graph above, it's clear that the root occurs between x=4 and x=6. below are lists of the root found and number of iterations needed by each of the methods depending on the initial conditions. To avoid any errors in the relaxation or Newton method, we will pick initial conditions greater than -ln(1/5).

This will also prevent Newton's method from converging to the obvious root x=0

Bisection method		
initial range	iterations	root (x)
(4, 6)	52	4.5
(4, 10)	54	4.75
(4, 21)	55	4.53125
(3, 10)	54	4.75
(2, 10)	55	4.0
(-1.5994379124341003, 10)	55	4.925245913310082

Relaxation method		
initial guess (x_0)	iterations	root (x)
1.8094379124341002	12	4.965114231744276
2	12	4.965114231744276
4	11	4.965114231744276
50	11	4.965114231744276

Newton's method		
initial guess (x_0)	iterations	root (x)
1.8094379124341002	6	4.965114231744276
2	5	4.965114231744276
4	4	4.965114231744276
50	4	4.965114231744276

the correct root is x≈4.965. The bisection method wasn't very accurate and required the most iterations for most initial conditions tested but the Newton and relaxation methods were very accurate. In all cases where x_0 was above -ln(1/5), Newton's method is the most accurate and requires the least iterations, followed by relaxation method then the bisection method. Newton's method is inefficient when F'(x) is very small (in this case if x_0 is close to -ln(1/5)) and useless when F'(x)=0 (x_0=-ln(1/5)).

Using the root found by the Newton Method (x ≈ 4.965114) we will solve for the Wien Displacement constant:
b=hc/k_bx ≈ 0.002898 meters*Kelvin
With this constant we can determine the estimated surface temperature of the sun:
T=b/502nm ≈ 5772.456274 kelvin