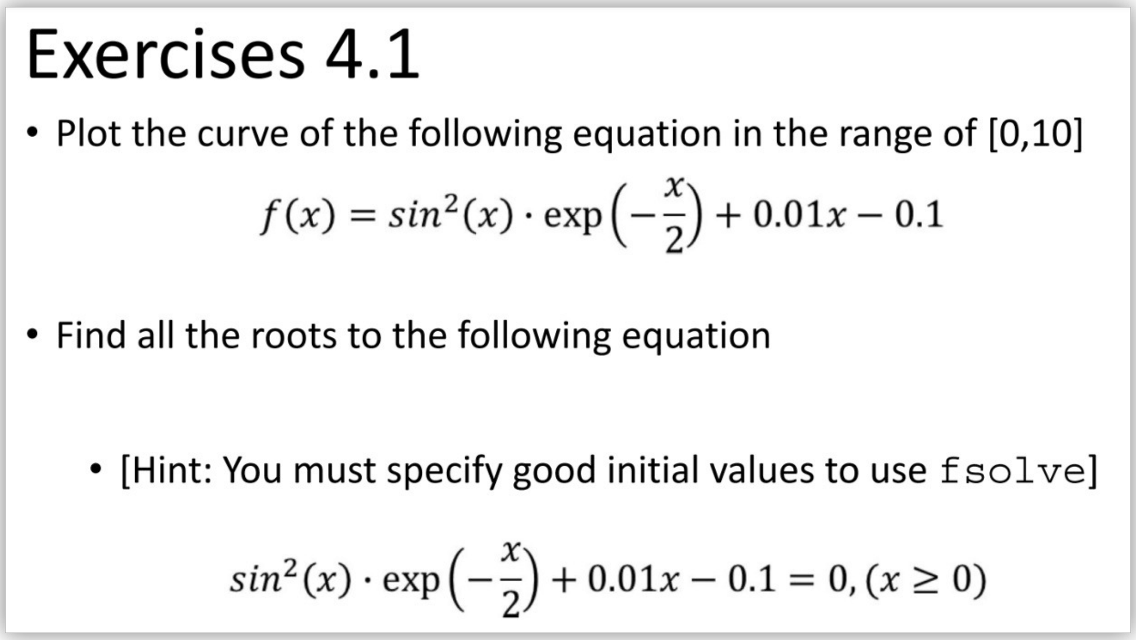
Report 4

Maximilian Fernaldy - C2TB1702



Problem 1

First, we want to plot the curve of the function:

$$f(x) = sin^2(x) \cdot exp\left(-\frac{x}{2}\right) + 0.01x - 0.1$$

To do this, we first define the domain for x:

```
x = 0:0.1:10;
```

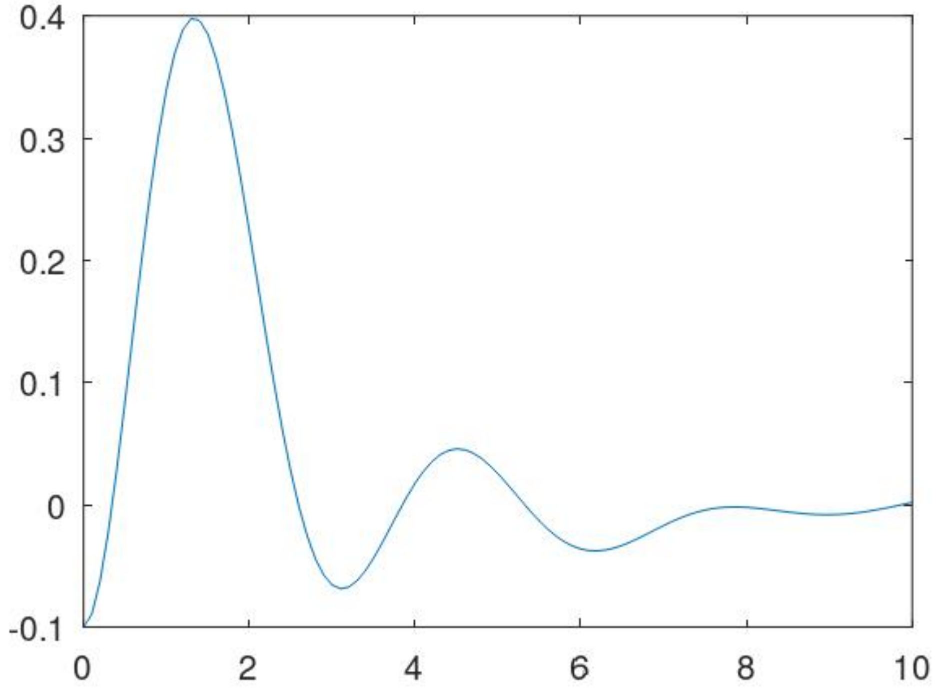
and define y as a function of x, i.e., $y = f(x)$:

```
y = sin(x).^2.*exp(-x/2)+0.01.*x-0.1;
```

With the arrays for x and y defined, we can now plot the curve:

```
plot(x,y)
```

Running the program, we find this graph, completing the first problem:



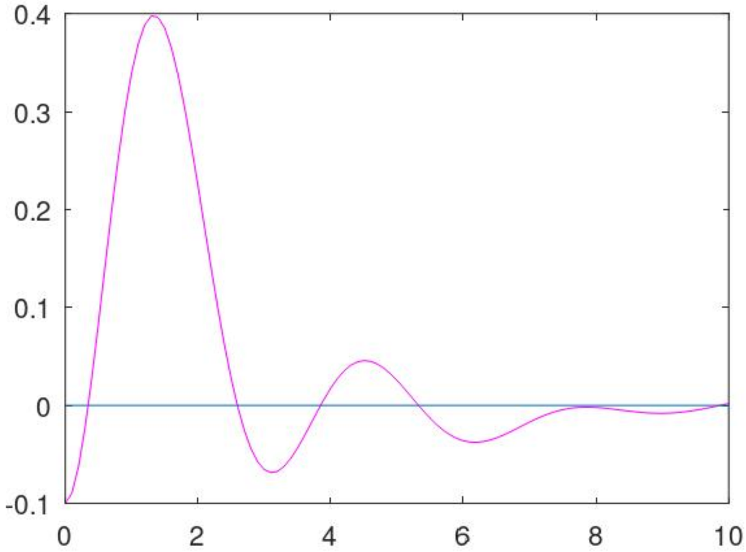
Problem 2

Next, we want to find the roots to the equation:

$$sin^2(x) \cdot exp\left(-\frac{x}{2}\right) + 0.01x - 0.1 = 0, (x \geq 0)$$

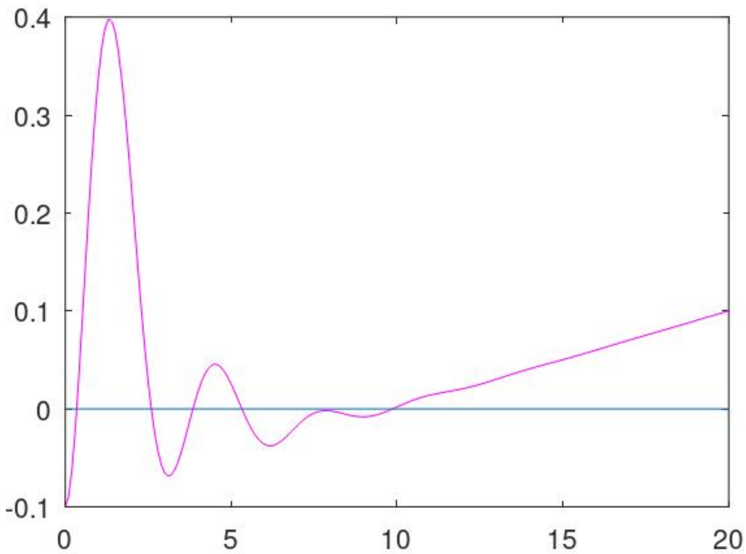
It's obvious that this equation is the function in the first problem, but with $f(x) = 0$. The roots of the equation are the x-coordinates of where the curve intersects the x-axis, or also called the $y = 0$ line. We can intuitively see that the curve crosses the $y = 0$ line several times. To make these intercepts clearer, we draw a line for $y = 0$. To do this, define a one-row array called `intercept` and filling it with zeros everywhere on `x`. We then change the plot line to also include `(x,intercept)`. We make the curve and line different colors to better distinguish them.

```
intercept = zeros(1,length(x));
plot(x, y, "m", x , intercept)
```



However, we don't know if the curve keeps crossing over the line or not without calculating. To visually confirm if there are no more roots in the domain $x \geq 0$, we can extend the array where we defined x to be longer. We can then see more of the curve.

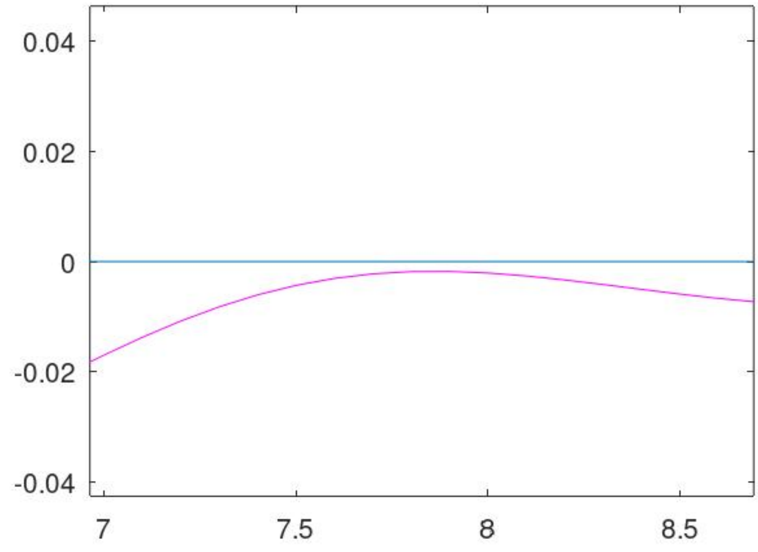
```
x = 0:0.1:20; % extending x array to 20
y = sin (x).^2.*exp(-x/2)+0.01.*x-0.1;
intercept = zeros(1,length(x)); % Draw a line for y = 0 to find intercep
plot(x, y, "m", x , intercept);
```



As we can see from this extended plot, the curve diverges from the x-axis, which means it won't cross the line again after the point around $x = 10$.

At first glance, there seems to be 6 points where the curve intersects the x axis. However, upon closer inspection, there really only are 5.

Zooming in on the point close to $x = 8$:

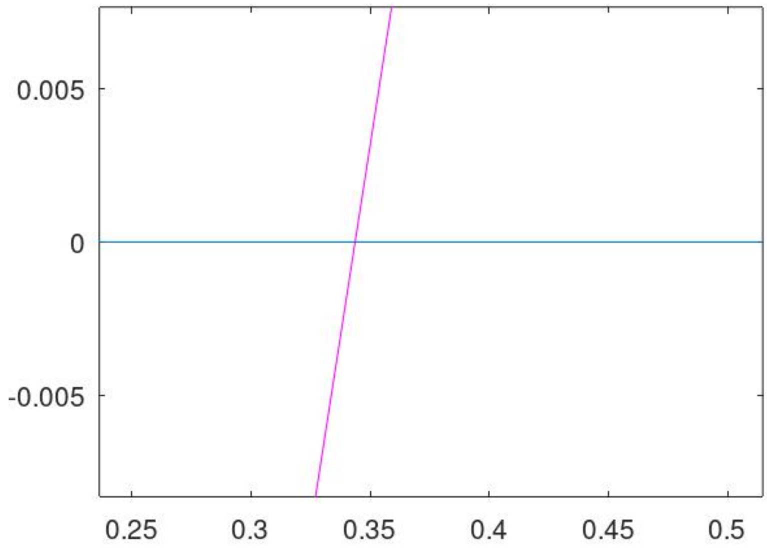


While it is close, the curve does not quite touch the line.

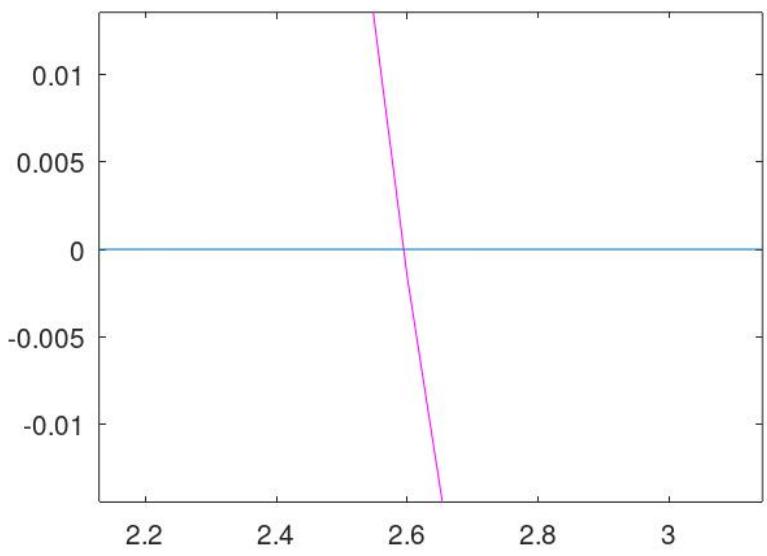
Finding initial guesses to put in `x0`

To get the actual values of the roots, we need to use `fsolve`, as the equation is nonlinear. However, `fsolve` requires us to make initial guesses about where the roots are, and these initial guesses have to be somewhat close to the actual point. Luckily, we have the plot as a tool.

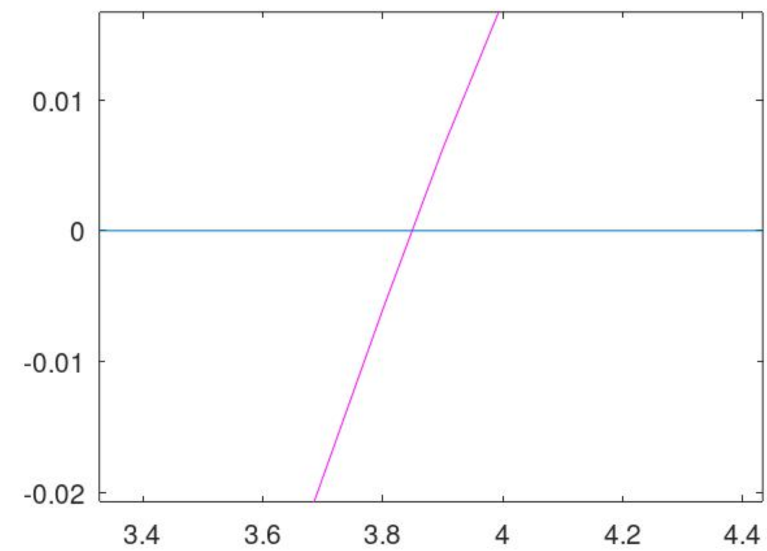
Zooming in on the intersects:



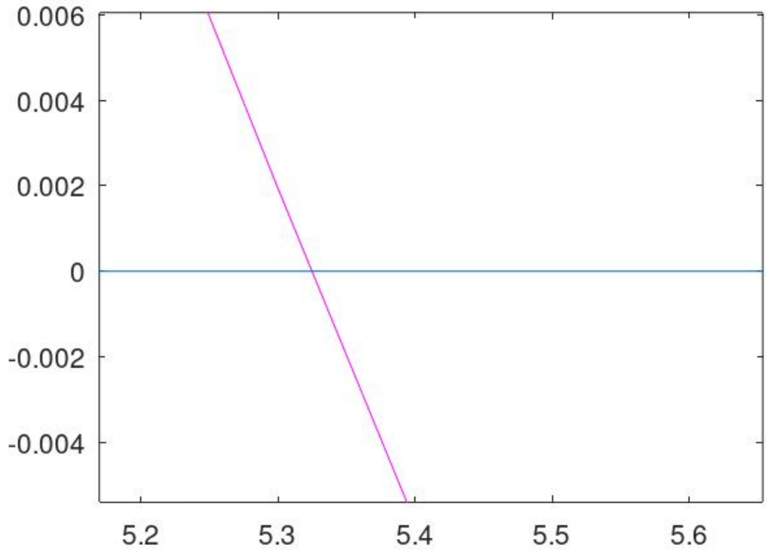
The first intersect is around $x = 0.3$.



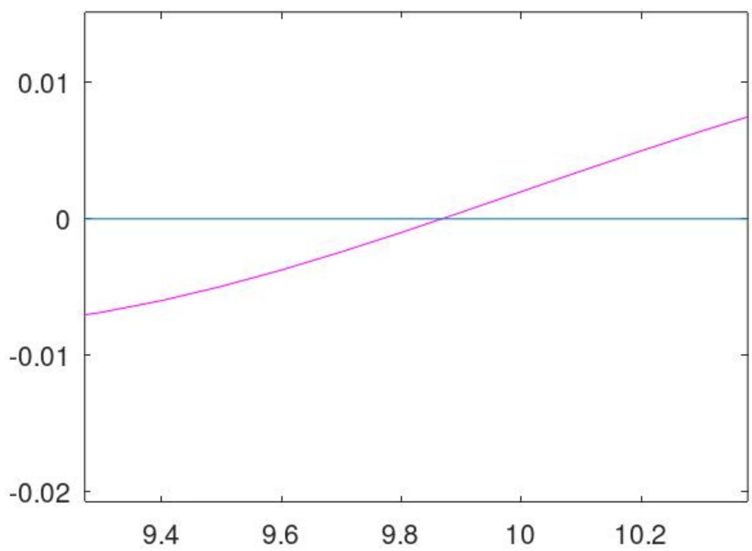
The second intersect is around $x = 2.6$.



The third intersect is around $x = 3.8$.



The fourth intersect is around $x = 5.3$.



The fifth and final intersect is around $x = 9.8$.

With these values, we construct an array `x0`:

```
x0 = [0.3; 2.6; 3.8; 5.3]
```

Now we declare a function `y = f(x)` for the left side of the equation.

```
function y = f(x)
  y = sin (x).^2.*exp(-x/2)+0.01.*x-0.1;
end
```

In order to find the roots of $f(x)$, we can use `fsolve` in this manner:

```
result = fsolve(@f, x0)
```

Which in turn outputs the following as answers.

```
>> CAPS_04_C2TB1702_roots

ans =

    0.3456
    2.5941
    3.8480
    5.3244
    9.8677
```

Checking validity of answers

To check the validity of our result, we can plug in the x-values we get and see if we get $f(x) = 0$.

```
x = result;
y = sin (x).^2.*exp(-x/2)+0.01.*x-0.1
```

Running this after getting the result from our `fsolve` script outputs:

```
>> CAPS_04_C2TB1702_check

y =

    1.6067e-06
    2.8132e-10
   -1.4310e-08
    4.1089e-09
    1.3504e-09
```

These values are very close to zero, validating our `fsolve` script.