1

The goal of minimization is to find the minima out of a set of data, while maxima is the opposite. We can have discrete or continuous minimization. Continues can be our regular functions, what discrete is solving for example integral problems as the traveling salesperson.

Then we have various ways of finding said minima, here I would divide to techniques which required knowing the derivative (Newton's) and some the doesn't (GSS, parabolic interpolation)

Also, it can be divided into methods of constrained or unconstrained minimisation methods.
For constrained minimization we can for example include a linear programming problem, when we need to minimize a given value with linear constraints. A method for solving it can be the "ellipsis method" which is not part of this lecture but is a valuable method.
Unconstrained methods include many of the methods we learned - Newton's, GSS, gradient descent, conjugate gradient which we are using in this exercise

Some minimization applications can involve both constrained and unconstrained.

2

parabola fitting requires choosing 3 points carefully to ensure the parabola represents the real shape of the given function. parabolic fitting can lead to poor results if we for example choose 3 co-linear points.

For a case that gives good parabolic results but not newton's we can think of $f(x) = |x|$.

The derivative of which is not defined at $x=0$ while parabolic fitting with good points can give accurate results.

3 Fit a parabola $f(\lambda) = a\lambda^2 + b\lambda + c$ given values for $f(0)$, $f'(0)$ and $f(\lambda_{max})$. and find minima λ^*

$$f'(\lambda) = 2a\lambda + b \Rightarrow f'(0) = b$$

$$f(0) = a \cdot 0 + b \cdot 0 + c = c$$

Knowing b and c, we assign λ_{max} to the equation to find a

$$f(\lambda_{max}) = a\lambda_{max}^2 + b\lambda_{max} + c$$

$$a = \frac{f(\lambda_{max}) - f(0) - f'(0)\lambda_{max}}{\lambda_{max}^2}$$

Now the minima is at $f'(\lambda^*) = 0 = 2a\lambda^* + b$

$$\lambda^* = -\frac{b}{2a} = -\frac{f'(0)}{2 \cdot \frac{f(\lambda_{max}) - f(0) - f'(0)\lambda_{max}}{\lambda_{max}^2}} = -\frac{f'(0)\lambda_{max}}{2\left[f(\lambda_{max}) - f(0) - f'(0)\lambda_{max}\right]}$$

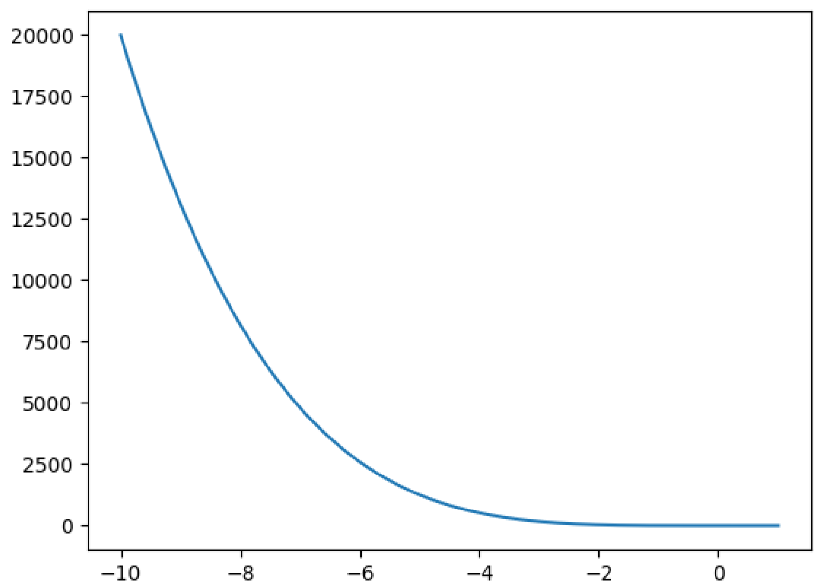
And the full parabola equation is:

$$f(\lambda) = \frac{f(\lambda_{max}) - f(0) - f'(0)\lambda_{max}}{\lambda_{max}^2} \cdot \lambda^2 + f'(0) \cdot \lambda + f(0)$$

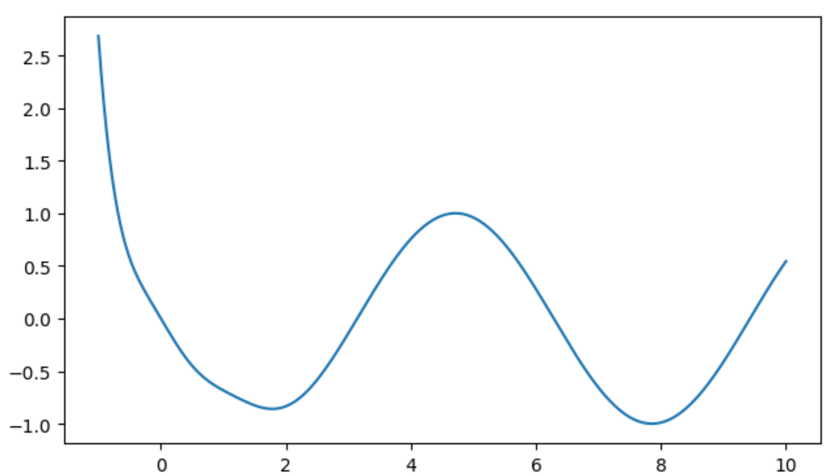
4

golden.py runs using python3 with numpy, scipy and matplotlib as prerequisites.

Plotting the function we see that the minima is somewhere around x=0.
The function shows very high values when going x < 10 therefore I plot only this part.



But then actually the plot lies a bit, and if we would plot the interval [-1, 10] we would see the function fluctuates because of the sin function:



Now we know that the function has many minimas at x > 0 and our golden ratio minima function will return one depending on the bracket. Then if we choose brackets according to the plot we get:

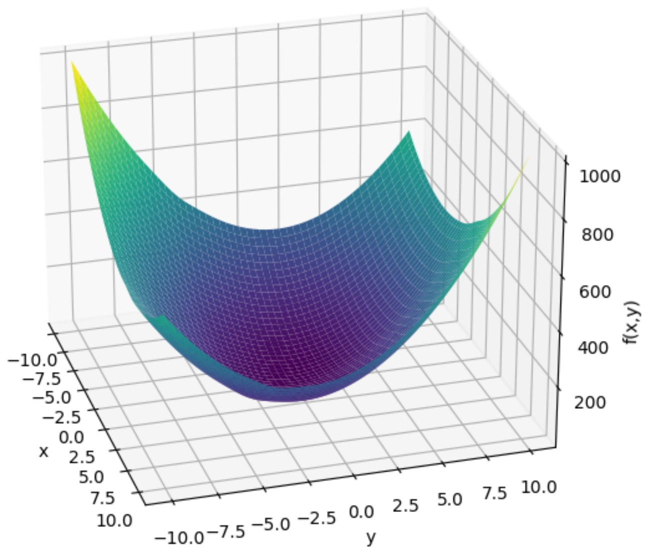
```
bracket=[4, 10], minima=7.853981644976766
bracket=[0, 4], minima=1.7821338003740406
```

And those minima indeed seem to be correct according to the plot.

5

minimize.py runs using python3 with numpy and scipy, matplotlib as prerequisites

First I have plotted (also for much bigger and smaller values to make sure) the function to get:



Seems that the answer is somewhere around the (0.05, 0) points, hence I will use (1, 1) as the initial guess. The result of the code is:

```
Nelder-Mean: [-0.07102868  0.01423573]
Powell's: [-0.07101993  0.01420334]
CG: [-0.07101994  0.01420334]
BFGS: [-0.07101854  0.01420302]
```

Now including runtime for each calling with the *minimize_with_perf* function, we get the following results for different initial guesses with the 2nd value being the runtime in seconds (`0.001955` for example at Nelder-Mean is it's runtime)

(20, -10)

```
Nelder-Mean: (array([-0.07101859,  0.01424471]),
0.001955747604370117)
Powell's: (array([-0.07101995,  0.01420334]), 0.0011289119720458984)
CG: (array([-0.07102021,  0.01420312]), 0.002456188201904297)
BFGS: (array([-0.07102097,  0.01420387]), 0.0019388198852539062)
```

(2, -4)

```
Nelder-Mean: (array([-0.07100177,  0.01416979]),
0.002518892288208008)
Powell's: (array([-0.07101976,  0.01420331]), 0.0015380382537841797)
CG: (array([-0.07101995,  0.01420333]), 0.004824161529541016)
BFGS: (array([-0.07101993,  0.01420333]), 0.0025141239166259766)
```

Seems that in general they all get very similar minima points, with slight changes.
The fastest is Powell's for both initial guesses: 0.0011 and 0.0015. CG is the slowest while at the (2, -4) initial guess CG gives twice slower than the rest calculation time (0.0048).