All the code runs with python(3) and has numpy, matplotlib and scipy as dependencies

1

Running the code at broyden.py we print the result of the Broyden method (with a bit of restructured operations to make it work in my case) and using scipy's broyden1 function.

Both return (x_init = [2, 2])

```
[ 1.43568543 -0.13509984]
[ 1.43568525 -0.13510898]
```

2

a I got f(x) = (x+0.5)/(0.5-x).
To get it, I first found a map from [-inf, inf] to [-½, ½] and then inverted it and substituted it into e^x.

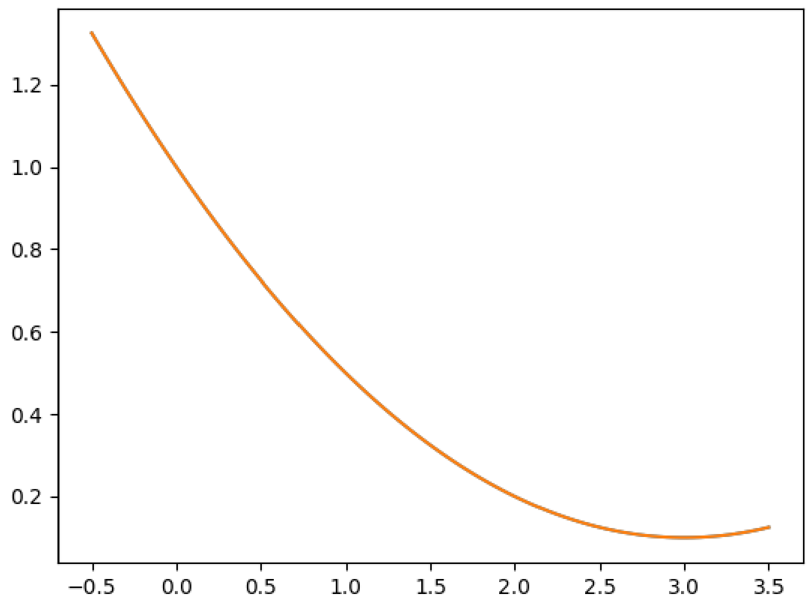
b - For cosx as far as I understand, as it is periodic. I can multiply x by a big number to make it oscillate very fast

c - Using the sigmoid function, I found the inverse which is *x = ln(k/(1-k))* and put arctan(x) instead of the k.

Well I am actually really not sure if I did anything correct here, the function is at exp.py -> myexp(x) and can run using python(3)

3

For the both we got the same polynomial (the calculations are at the next page) whose plot is



$\boxed{3}$ $f(x) = \frac{1}{1+x^2}$ $x_1 = 0$ $f(0) = 1$

$x_2 = 1$ $f(1) = 1/2$

$x_3 = 3$ $f(3) = 1/10$

a) - Lagrange

$$L_0 = \frac{(x-1)(x-3)}{(0-1)(0-3)} = (x^2-4x+3)/3$$

$$L_1 = \frac{x(x-3)}{(1-0)(1-3)} = \frac{x^2-3x}{-2} \qquad L_2 = \frac{x^2-x}{(3-0)(3-1)} = \frac{x^2-x}{6}$$

$$p_2(x) = \frac{x^2-4x+3}{3} - \frac{x^2-3x}{2} \cdot \frac{1}{2} + \frac{1}{10} \frac{x^2-x}{6}$$

$$= 20x^2 - 80x + 60 - 15x^2 + 45x + x^2 - x = 6x^2 - 36x + 60$$

$$= \frac{1}{10}x^2 - \frac{6}{10}x + 1$$

- Newtot

$$p_2(x) = a_0 + a_1(x - x_0) + a_2(x-0)(x-1)$$

$$a_1 = f(1) - \frac{f(0)}{1-0} = -\frac{1}{2} \qquad a_2 = \left(-\frac{1}{5} + \frac{1}{2}\right)/3 = \left(-\frac{2}{10} + \frac{5}{10}\right)/3 = \frac{1}{10}$$

$$p_2(x) = 1 - \frac{1}{2}x + \frac{1}{10}(x^2 - x) = \frac{1}{10}x^2 - \frac{6}{10}x + 1$$

$\boxed{4}$ $p_N(x) = \sum^{N} c_j e^{jx}$

$p_N(x_i) = y_i$ with $y_0, y_1 \dots y_N$ given data.

we would show that there is a unique choice of coef $c_0 \dots c_N$ that satisfy it.

we can substitute $y = e^x$ and then

$\sum^{N} c_j (e^{jx}) = \sum^{N} c_j (e^x)^j = \sum^{N} c_j y^j$ which is just

a polynomial of degree N.

as we have $N+1$ data points y_i ~~[illegible]~~
and their counterpart x_i

we can say that by interpolation theorem, there exist

a unique polynomial of degree n, for $n+1$ data points.

And we indeed have $N+1$ points, for a N polynomial
hence the coefficients C_i are unique.