Germán Sánchez Arce

(In colaboration with Maria Gonzalez Cabrera & Joan Alegre Cantón)

PS1.Quantitative Macroeconomics

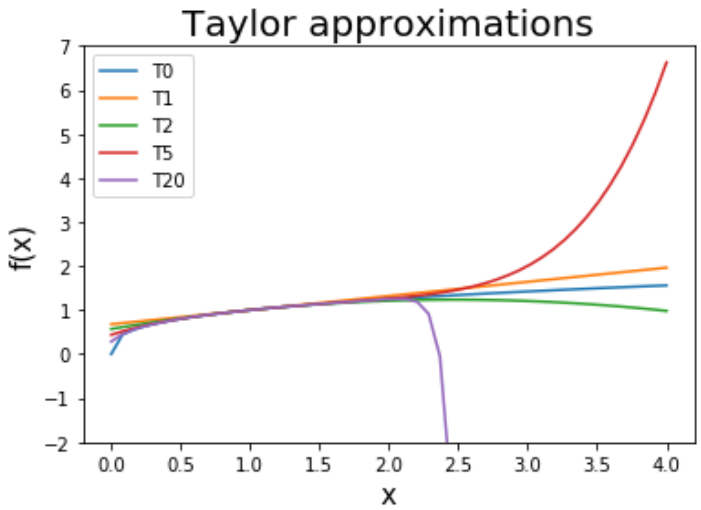
1. Question 1: Function Approximation: Univariate

a. Exercise 1

We need to specify the formula for Taylor series approximation, which is a local approximation method (in this case around a=1). In the following code we can observe the specific case of the first order approximation formula, and the general formula for any order approximation. This will give a us a different function for each order approximation:1,2,5,10,20.

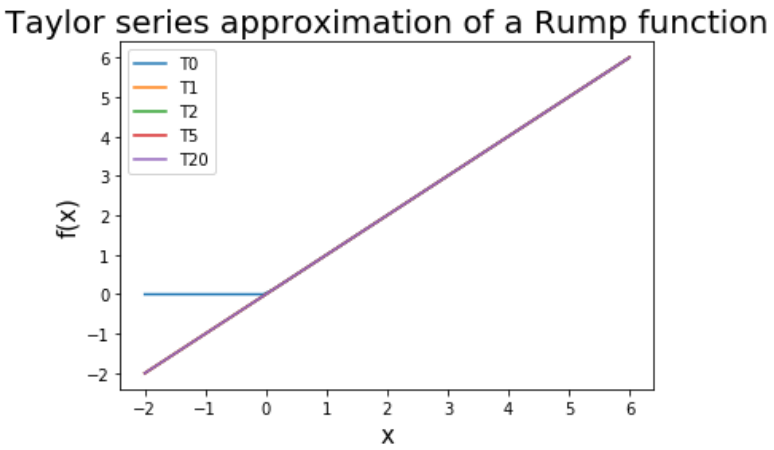
```
a=1
f=x**0.321  #function
taylor1=f.subs(x,a)+f.diff(x).subs(x,a)*(x-a)  #using the formula of the first order
print("the first order approximation solution around x=1 is",taylor1)

def taylor(f,a,n):      #general formula of the Taylor expansion
    k = 0
    p = 0
    while k <= n:
        p = p + (f.diff(x,k).subs(x,a))/(factorial(k))*(x-a)**k
        k += 1
    return p
```

Applying this formula and creating the plot we can obtain this result. We can observe how the real function T0 is approximated around x=1. However, as it moves away that point it is becoming less precise. This is because it is a local approximation method.



b. Exercise 2: $(x + |x|) / 2$

We apply again the same general Taylor approximation definition and get the following result:



As we can observe, since the original function is a straight line for the range $x \in (0,4)$, all the order approximations will have the same shape, since from the *nth* derivative of the function with *n>2* will equal 0. It can be understood observing the general formula (the additional components of the *nth* order approximation will add 0 to the previous order approximation, so they will be the same). Specifically, these functions will be $T_1(x) = T_2(x) = T_5(x) = T_{20}(x) = x$

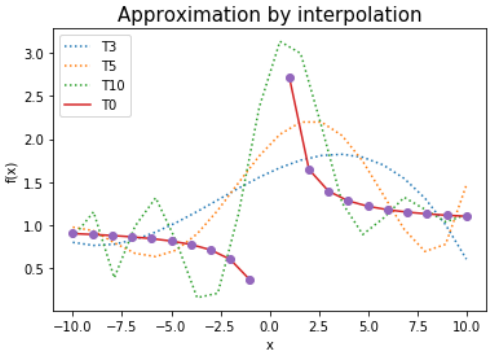
c. Exercise 3

Function: $e^{(1/x)}$

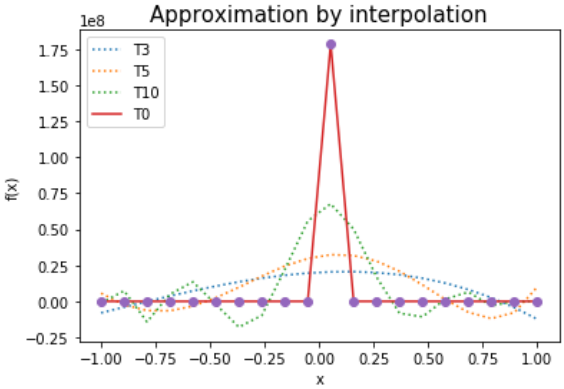
In this case we have a function with a discontinuity in x=0. Although in the exercise it asks you for the range $x \in (-1,1)$, I first show the range $x \in (-10,10)$ in order to observe this more clearly. In the first part we are going to use the following code in order to polynomial approximate with equally spaced nodes. This equally spaced nodes are created with the linear space of x (in this case we have 20 nodes). Additionally, I added another linear space of odd nodes (*n=21*) so as to represent better the original function T0 (if it were an even number it would represent the x=0 so there would be representation errors).

```
x = np.linspace(-10, 10, num=20, endpoint=True) #I did in this range
xf = np.linspace(-10, 10, num = 21, endpoint=True)
y=e**(1/x)
y2=e**(1/xf)

cheb3=np.polyfit(x,y,3)    #Monomial approximation of degree 3
val3=np.polyval(cheb3,x)
cheb5=np.polyfit(x,y,5)    #Monomial approximation of degree 5
val5=np.polyval(cheb5,x)
cheb10=np.polyfit(x,y,10)  ##Monomial approximation of degree 10
val10=np.polyval(cheb10,x)

```

"*Polyfit*" function returns the coefficients of the polynomials and "*Polyval*" evaluate a polynomial at specific values. This needs to be repeated with each single order approximation that we need. As a result, we have the following graph:



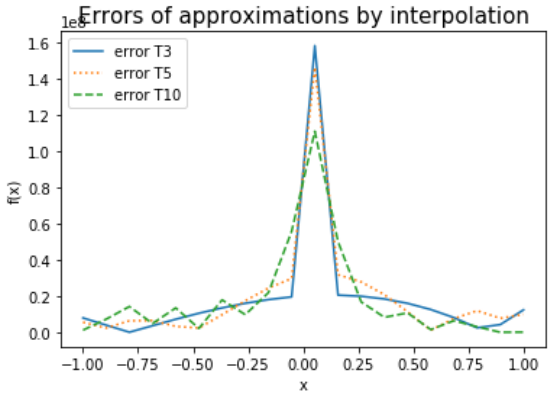
We can observe more clearly in a range of x ϵ (0,10) how the approximations estimate the real function as if there were not a discontinuity in x=0. So, monomial approximation is not a good idea in cases like this.



In the following graph it is represented in the domain x ϵ (0,10). Again, there is an important approximation error in the point x=0.

Just using these codes, that represent the absolute value of the difference between the real function and the approximations we can obtain the error approximations of the 3rd,5th and 10th orders:

```
45
46 error3 = abs(y1-val3) #error of the 3rd order approximation
47 error5 = abs(y1-val5) #error of the 5th order approximation
48 error10 = abs(y1-val10) #error of the 10th order approximation
49
```



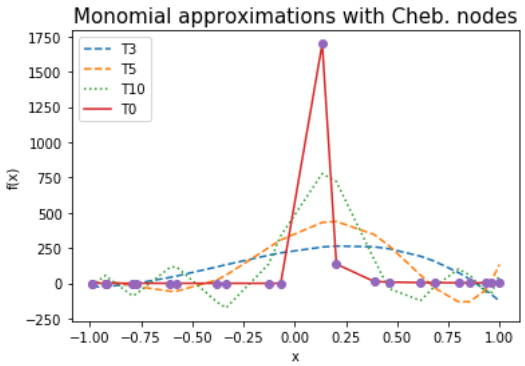
As we can see, in this special case the main error comes from the middle of the graph, where we have the discontinuity. However, in the tails we can observe also the *Runge phenomenon*, with oscillation at the edges.

Now let's use Chebyshev's nodes rather than equally spaced nodes. This will help us to reduce the errors at the edges allocating the most part of the nodes at the extremes of the graph.

For this we use this code using the pre-made function "Chebroots":

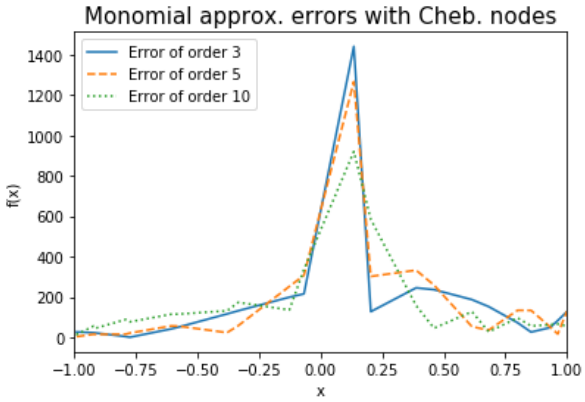
```
64 vector=np.linspace(-1,1,24)
65 ch=np.polynomial.chebyshev.chebroots(vector)
```

Again, we use the same interpolation as before and we will have the next result:



We can observe how the error tend to 0 at the edges (Runge phenomenon has been partially corrected).

Let's compare the errors if we use eq. spaced nodes or Cheb. Nodes:



Although it cannot be observed properly in the graph because the y-axis is in a different scale, we can see how the errors with the Cheb. Nodes tend to 0 at the edges. However, if we use equally spaced nodes (plot on the left) the error seems to be stable along the graph (except around x=0).

Now, let's use Chebyshev's polynomials with Chebyshev's root nodes:

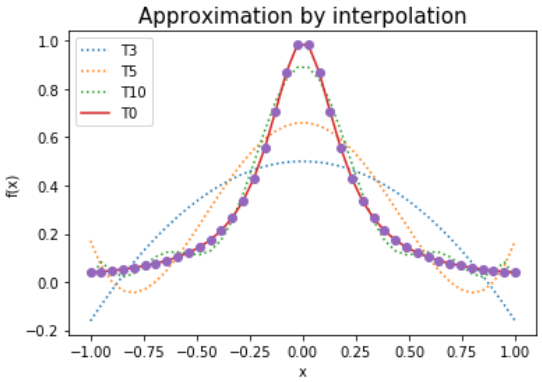
```
114 cheb3=np.polynomial.chebyshev.chebfit(ch,y,3)     #Chebyshev approximation of degree 3
115 val3=np.polynomial.chebyshev.chebval(ch,cheb3)
116 cheb5=np.polynomial.chebyshev.chebfit(ch,y,5)     #Chebyshev approximation of degree 5
117 val5=np.polynomial.chebyshev.chebval(ch,cheb5)
118 cheb10=np.polynomial.chebyshev.chebfit(ch,y,10)   ##Chebyshev approximation of degree 10
119 val10=np.polynomial.chebyshev.chebval(ch,cheb10)
120
```

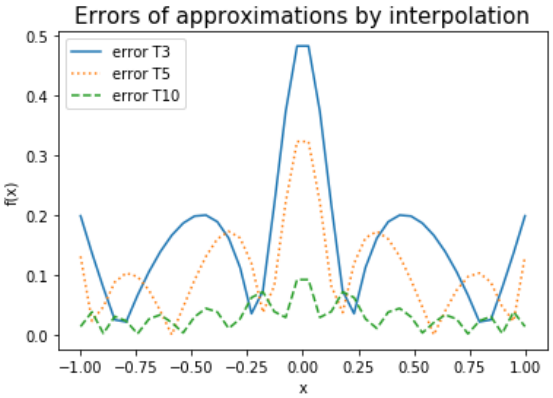
This code follows the same logic as Polyfit and Polyval but with Chebyshev's Polynomials.

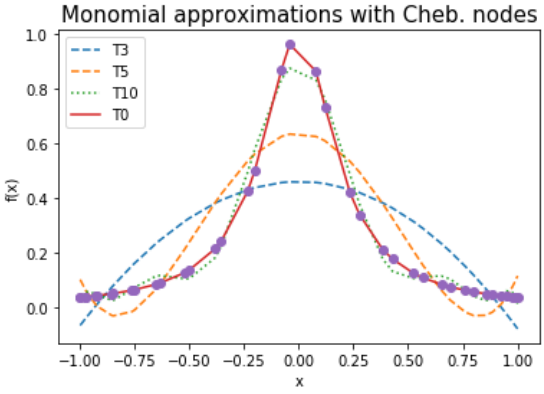
As we can see, both the approximations and errors are very similar to those of the monomial approximation with Chebyshev's roots.

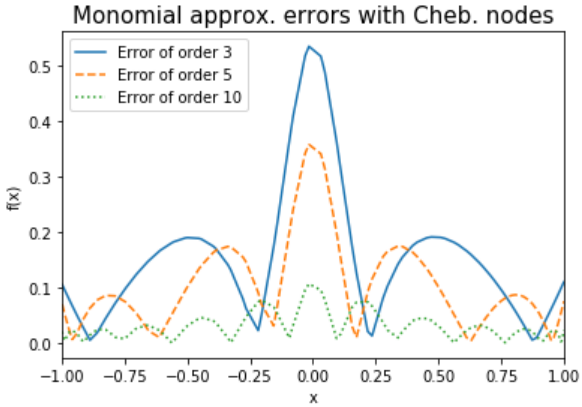
For the other two functions we are going to follow the same procedure, so I just limit to show and comment the graphs.

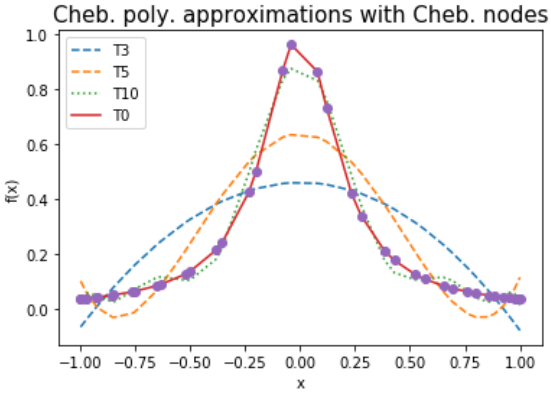
Function: $1 / (1 + 25 * x^2)$

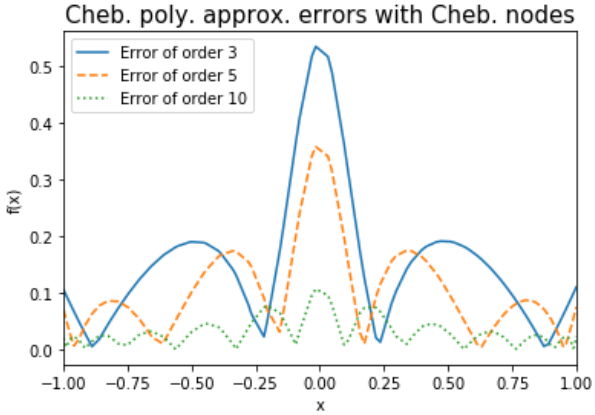








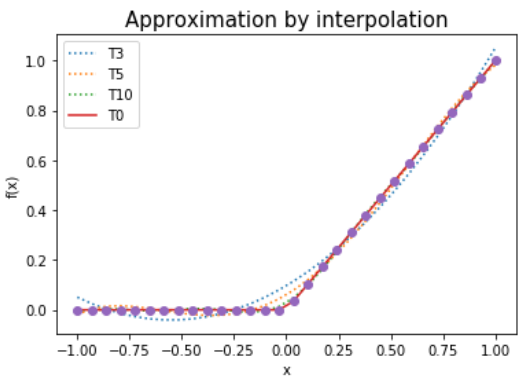


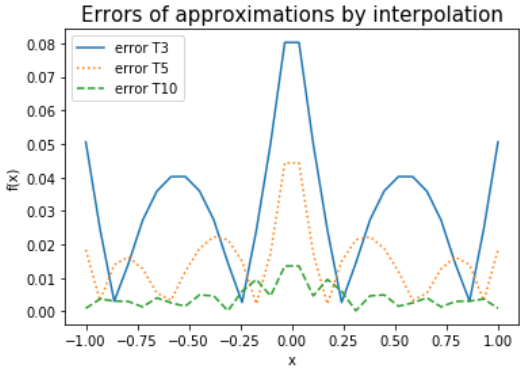


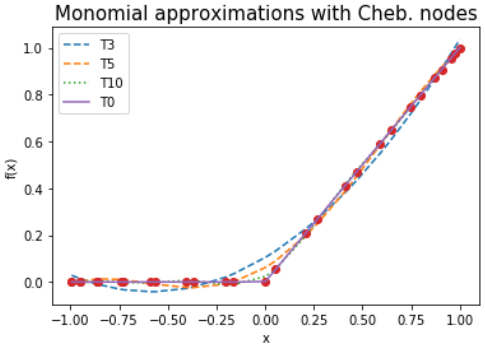
In this function we can observe clearly the effect of the Runge phenomenon. Using the approximation by interpolation with equally distant roots we can observe how the error is bigger at the edges (around 0.2 for the case of T3) than when we use Cheb. nodes (around 0.2 in the case of T3). Actually, we can observe that if we use Cheb. nodes the polynomials exhibit equioscillant nodes. On the other hand, we can also figure out that if we increase the order of the monomials or polynomials, the error tends to be lower.

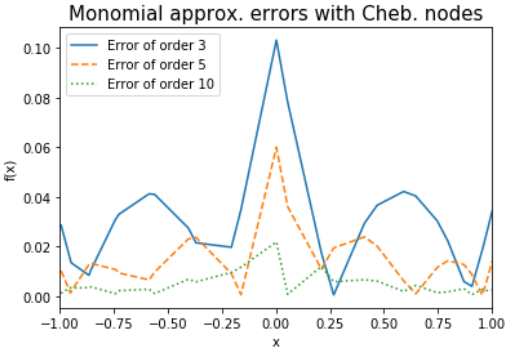
On the other hand, we can also check that in the cases of approximations with Cheb. nodes both the approximations and errors coincide. In both cases the nodes are allocated at the extremes, so the middle of the graph has more peaks, since it is defined with less nodes. However, the Cheb. polynomials should perform smoother approximations, although they present problems related with kinks of the original function.

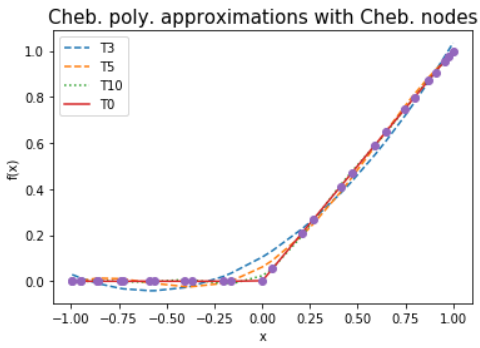
Function: $(x + |x|) \,/\, 2$

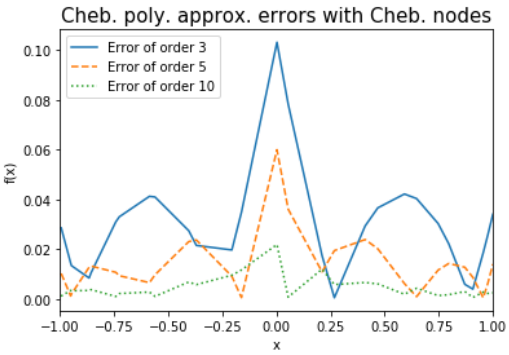






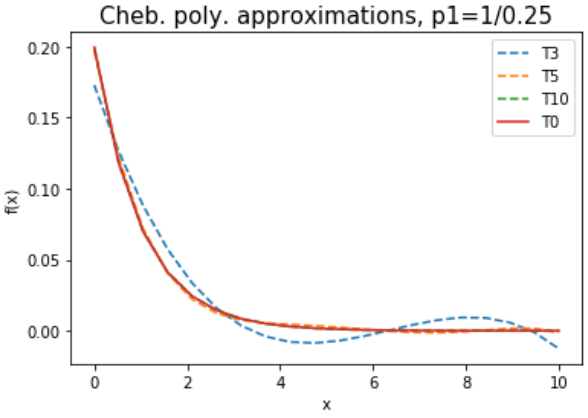


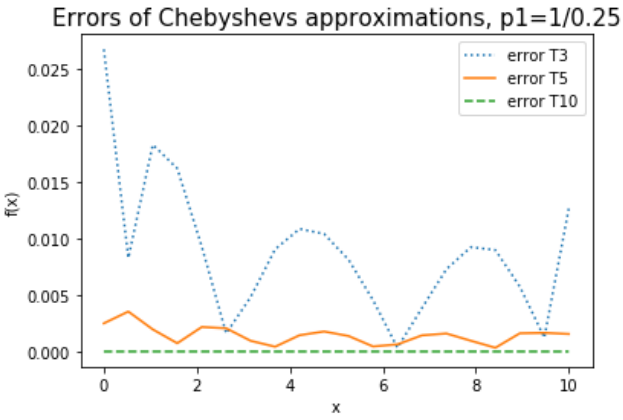




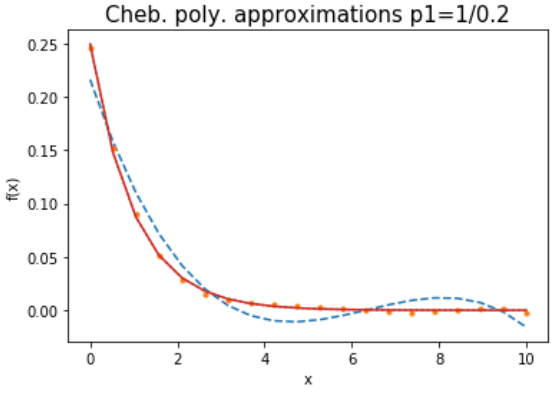
Again, we can check that the errors in the approximations with Cheb. nodes at the extremes are higher than the monomial approximation with equally distant nodes. However, in the middle of the graph the error is lower in this last case, since there are more nodes placed on that part of the graph.

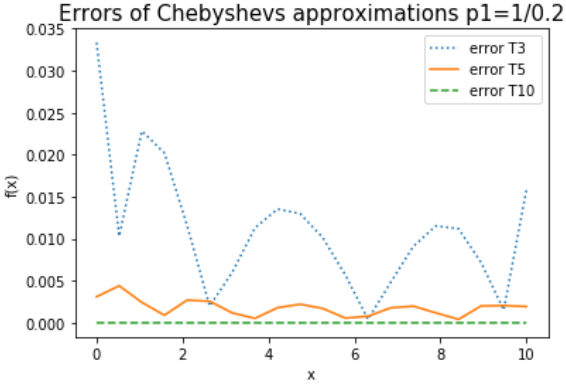
d. Exercise 4: $p(x) = e^{-\alpha x} / (p1 + p2e^{-\alpha x})$



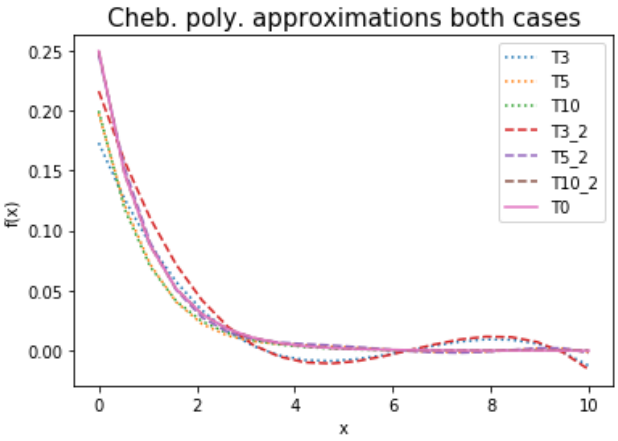


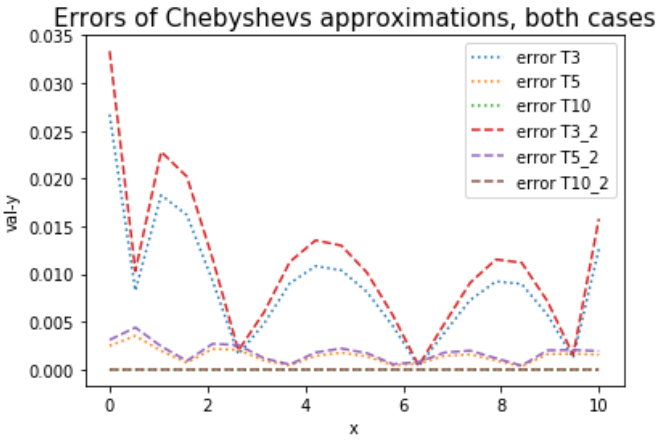
In the first case we have orders 3rd,5th and 10th order approximations with Chebyshev's polynomials. Looking at the errors, we can see how it approximates as long as the polynomial's order increases. In that sense, 3rd order approximation presents a clear error, while with the 5th order the error is minimum. Finally, the 10th order has almost 0 error so the function representation coincides with the original function T0.





If we modify the parameter p1=1/0.2, the shape of the functions is maintained but now the error is going to increase every single point of the domain. In the following graphs we can observe this comparing the approximations and errors of both cases.





Question 2:

1. $$f(k,h) = ((1-\alpha)k^{\frac{\sigma-1}{\sigma}} + \alpha h^{\frac{\sigma-1}{\sigma}})^{\sigma/(\sigma-1)}$$

First, we need to calculate the marginal productivity of labor (MPL) and the marginal productivity of capital (MPK):

- MPH = $\frac{df(k,h)}{dh} = \frac{\sigma}{\sigma-1}[(1-\alpha)k^{\sigma-1/\sigma} + \alpha h^{\sigma-1/\sigma}]^{\sigma/(\sigma-1)} \frac{(1-\alpha)(\sigma-1)}{\sigma} k^{-1/\sigma}$

- MPK = $\frac{df(k,h)}{dk} = \frac{\sigma}{\sigma-1}[(1-\alpha)k^{\sigma-\frac{1}{\sigma}} + \alpha h^{\sigma-\frac{1}{\sigma}}]^{\sigma/(\sigma-1)} \frac{\alpha(\sigma-1)}{\sigma} h^{-\frac{1}{\sigma}}$

Dividing both marginal productivities we get:

$$\frac{\frac{df(k,h)}{dk}}{\frac{df(k,h)}{dh}} = \frac{(1-\alpha)h^{1/\sigma}}{\sigma\, k^{1/\sigma}} = \frac{(1-\alpha)}{\sigma}\left(\frac{h}{k}\right)^{1/\sigma} = \frac{MPK}{MPH}$$

Now, taking logarithms:

$$\log\left(\frac{MPK}{MPH}\right) = \log\left(\frac{1-\alpha}{\sigma}\right) + \frac{1}{\sigma}\log\left(\frac{h}{k}\right) \qquad (1)$$

Since we are looking for the marginal rate of substitution of capital and labor, we need to take derivative in (1) with respect to $\log(\frac{k}{h})$ (or w.r.t. $\log(\frac{h}{k})$ and take the inverse). The result we obtain is:

$$\varepsilon_{kh} = \sigma$$

2. Obtain the labor share of an economy with CES production function.

We know that labor share is given by:

$$s = \frac{MPH\ h}{f(k,\ h)} \qquad (2)$$

Therefore:

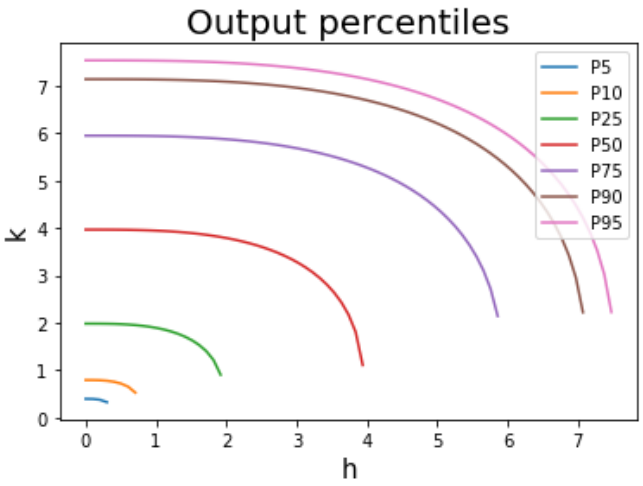
$$MPH\ h = \alpha h[(1-\alpha)k^{\sigma-1/\sigma} + \alpha h^{\sigma-1/\sigma}]^{\sigma/(\sigma-1)}\ k^{-1/\sigma}$$

Call A to $(1-\alpha)k^{\sigma-1/\sigma} + \alpha h^{\sigma-1/\sigma}$. Following (2), we end up to the following result:

$$s = \frac{1}{A}\alpha h^{\sigma-1/\sigma}$$

3. I don't know how to do the code for this part

4. <u>Percentiles</u>:



In this graph we can see the different isoquants associated with the percentiles of output. For those percentiles that are higher we need a higher level of inputs. Then, there is a positive relationship between inputs and output.