Quantitative Macroeconomics - Homework 1

Gabriela Barbosa

September 28, 2018

Contents

Part I

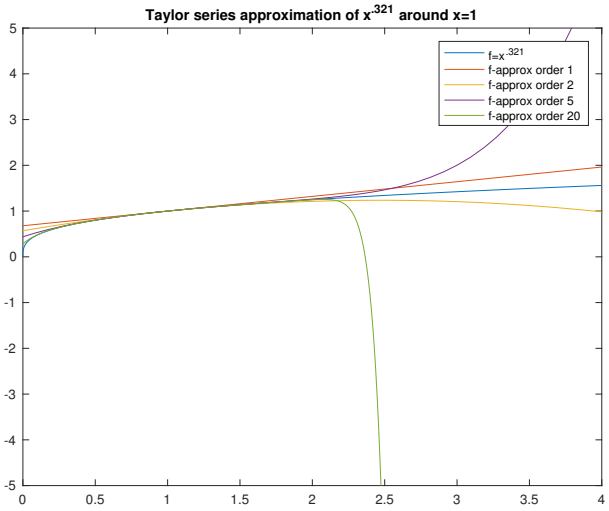
Univariate Function Approximation

1. Taylor series: $f(x) = x^{.321}$

Given an order n, the approximation around a is given by:

$$\tilde{f}(x) = f(a) + \sum_{i=1}^{n} \frac{\partial^n f(a)}{n!}(x-a)^n \tag{1}$$

Below are the graphical results for the approximation of the given function around $\bar{x} = 1$ over $(0, 4)$ [1].



This function has its only singularity at $x = 0$. By Theorem 6.1.2 (Judd), the Taylor series cannot reliable approximate $f(.)$ at any point further away then 1 (in absolute distance) from $a = 1$. That is, at any point $x > 2$ the approximation is not reliable, even with high-order approximations.

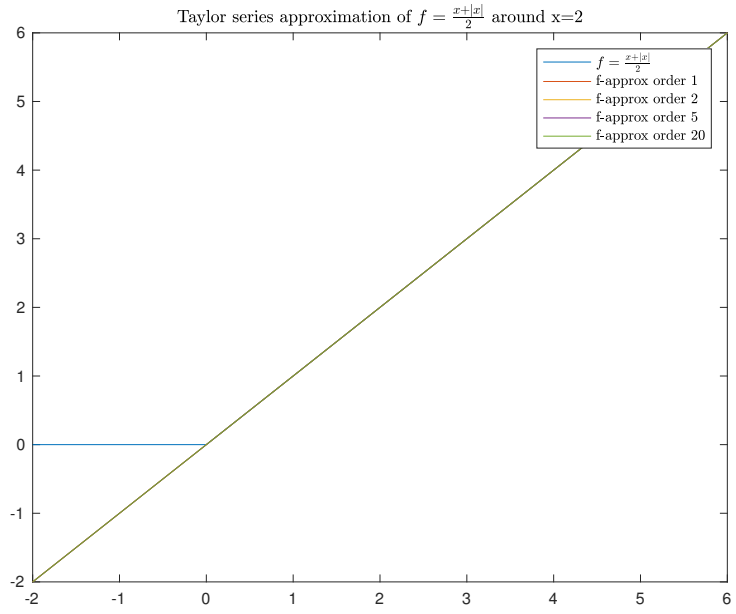
We can see this on the graphical results above. The approximation is good only for values with $|1|$-distance from $a = 1$. The highest-order approximation (20th) even fails completely after $x = 2$.

2. Taylor series: $f(x) = \frac{x+|x|}{2}$

The same algorithim (1) described above is used.

[1] Matlab code: hmw1.m

Below are the graphical results for the approximation of the given function around $\bar{x} = 2$ over $(0, 6)$ [2].



This function it is not defined at any point $x < 0$. Since the approximation is around $a = 2$, and by Theorem 6.1.2 (Judd), the Taylor series cannot reliable approximate $f(.)$ at any point below 0. That is, at any point $x < 2$ the approximation is not reliable, even with high-order approximations.

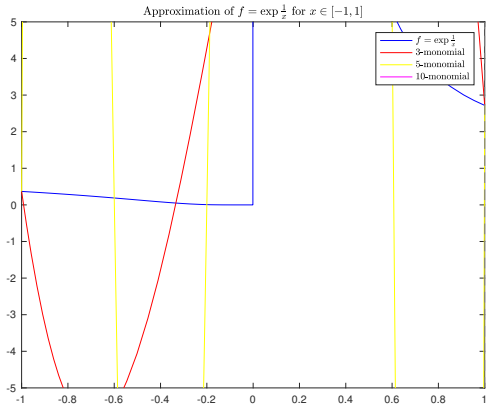
We can see this on the graphical results above. The approximation is good only for non-negative values.

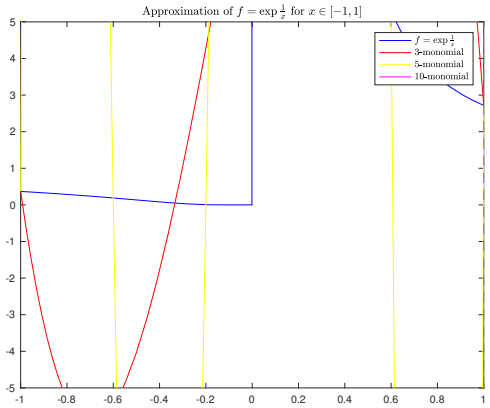
Moreover, since its derivative is not well-defined after the second-order, all order approximation deliver the same result.

[2] Matlab code: hmw2.m

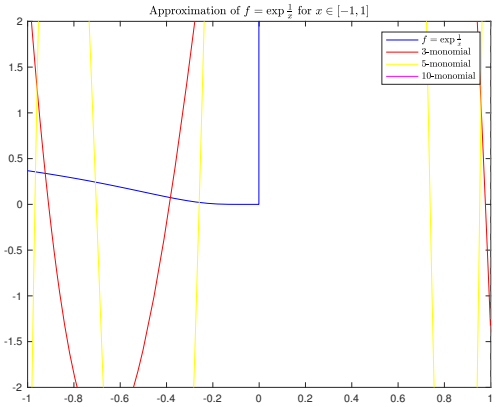
3a. $f(x) = e^{\frac{1}{x}}$

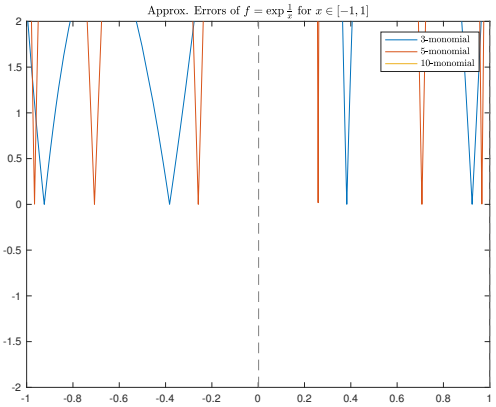
Evenly spaced nodes and monomials approximation



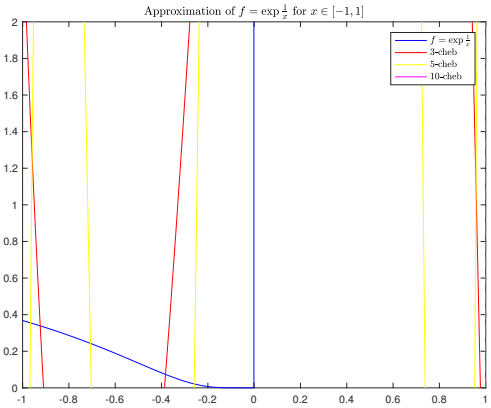


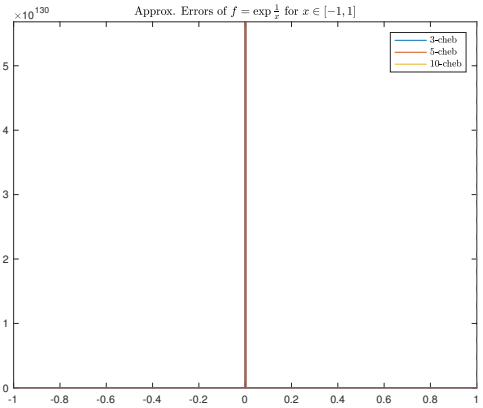
Chebyshev nodes and monomials approximation





Chebyshev nodes and Chebyshev approximation



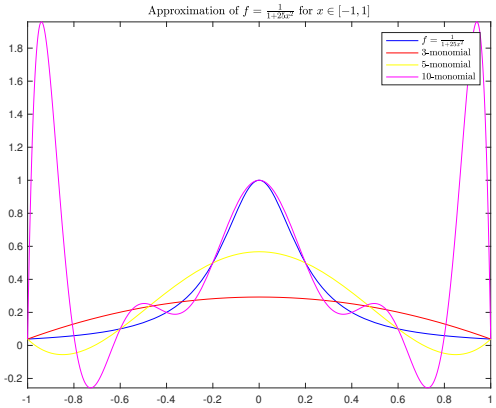


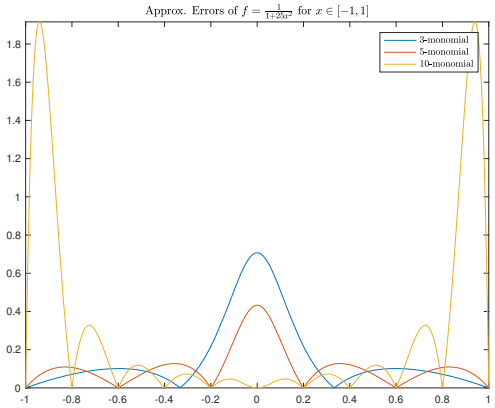
Comments

This function has a discontinuity at $x = 0$, at which it is not well defined, and it delivers only positive values. None of the apporixmations are able to capture this. That's mainly because the approximation methods rely on the function beign smooth.

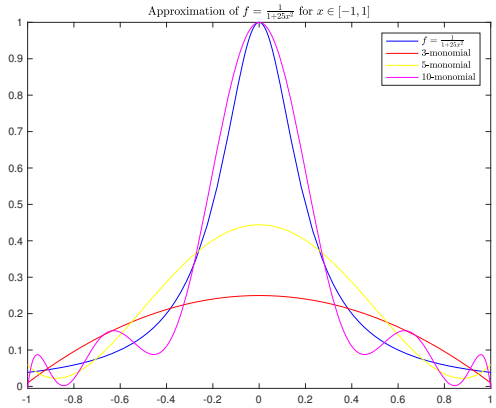
3b. $f(x) = \frac{1}{1+25x^2}$

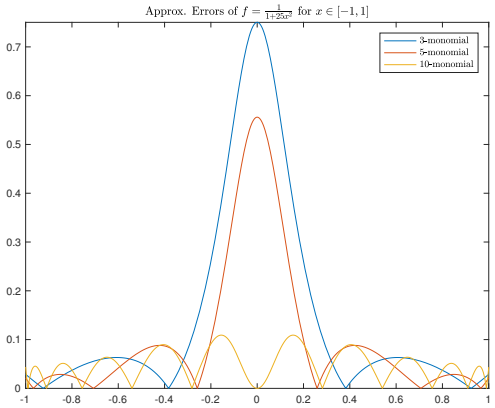
Evenly spaced nodes and monomials approximation



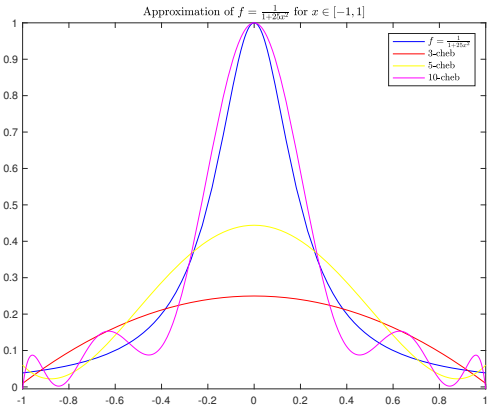


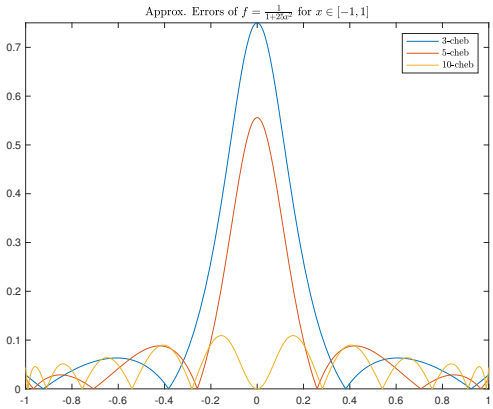
Chebyshev nodes and monomials approximation





Chebyshev nodes and Chebyshev approximation





Comments

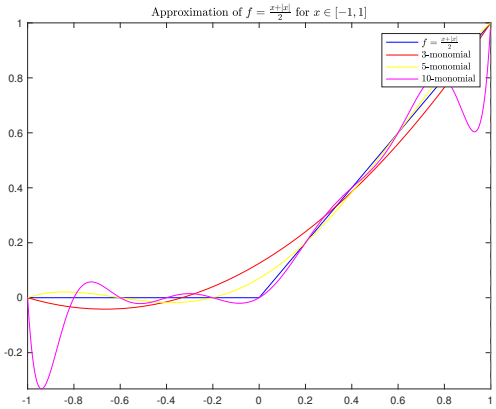
The first approximation method is more accurate for higher-order monomials. Moreover, it perfoms poorly for the end tails of the function. As a result, the errors are oscilant and vary on the x-domain. The furthest away from the midpoint node $x = 0$, the less reliable is the approximation and greater the error.

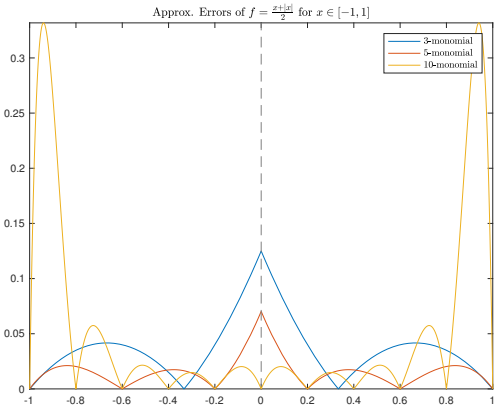
When introducing Chebyshev interpolation nodes along with the polynomial, the approximation improves its accuracy, since further nodes get more 'weight' on the algorithm. However, approximation does a better job only for the highest order polynomial.

For the last scenario, when using both Chebyshev interpolation nodes and polynomail, we can see that there is no much change in the approximation results. That's because the monomials and Chebyshev polynomials are similar.

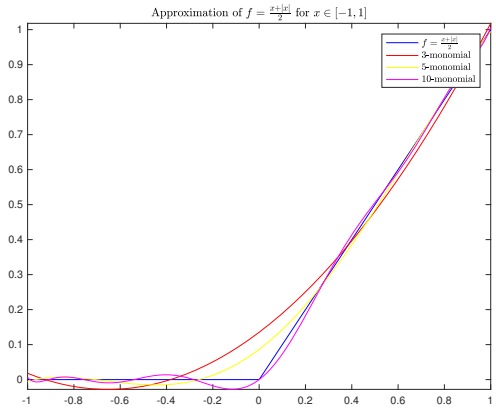
3c. $f(x) = \frac{x+|x|}{2}$

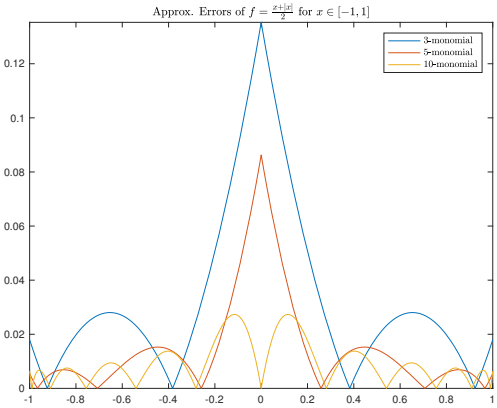
Evenly spaced nodes and monomials approximation



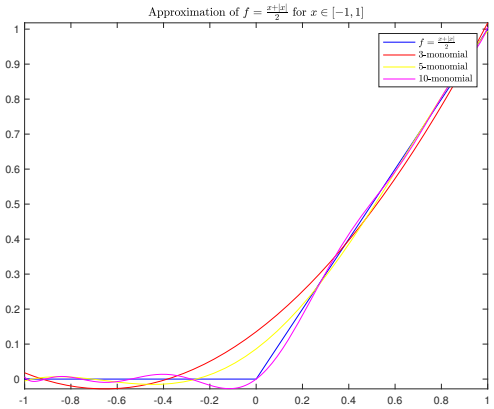


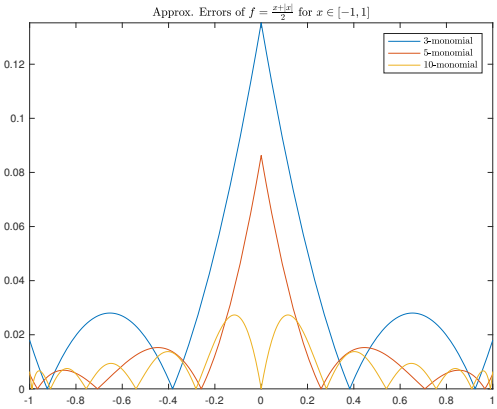
Chebyshev nodes and monomials approximation





Chebyshev nodes and Chebyshev approximation





Comments

The function has a discontinuity at $x = 0$: for all non-positive x-values the function is always non-negative; and for any $x < 0$ the function is 0.

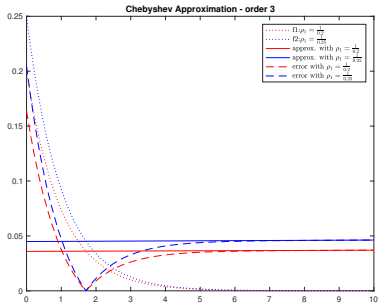
The first approximation method is more accurate for smaller-order monomials. Moreover, it perfoms poorly the further away from $x = 0$. As a result, the errors are oscilant and vary on the x-domain. Also, the approximation is not able to capture the discontinuity at 0 or the constant value for $x < 0$.

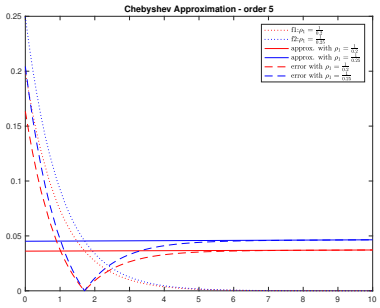
When introducing Chebyshev interpolation nodes along with the polynomial, the approximation improves its accuracy, since further nodes get more 'weight' on the algorithm. Approximation does a better job only for the highest order polynomial.

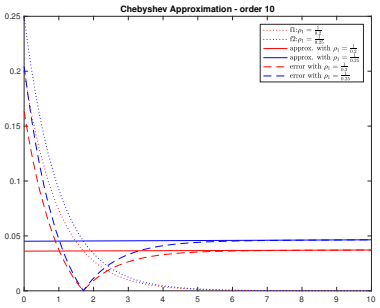
For the last scenario, when using both Chebyshev interpolation nodes and polynomail, we can see that there is no much change in the approximation results.

4. Probability Function Approximation

Below are the graphical results for a function approximation of $p(x) = \frac{\exp(-\alpha x)}{\rho_1 + \rho_2 \exp(-\alpha x)}$ over $x \in (0, 10]$ by Chebyshev nodes and Chebyshev polynomials of orders 3, 5 and 10.







Part II

Multivariate Function Approximation

CES function: $f(k,h) = \left((1-\alpha)k^{\frac{\sigma-1}{\sigma}} + \alpha h^{\frac{\sigma-1}{\sigma}}\right)^{\frac{\sigma}{\sigma-1}}$

a.σ is the Elasticity of Substitution

Consider a bivariate function $f(x_1, x_2)$. Let the elasticity of substitution (ES) between x_1, x_2 be $\epsilon(x_1, x_2)$. Then, ES is defined by:

$$\epsilon = -\frac{d\log(\frac{x_2}{x_1})}{d\log(\frac{f_{x_1}}{f_{x_2}})} \equiv \frac{d(\frac{x_2}{x_1})}{\frac{x_2}{x_1}} \frac{\frac{f_{x_1}}{f_{x_2}}}{d(\frac{f_{x_1}}{f_{x_2}})}$$

Applying this concept to the CES function, we have:

$$\frac{f_k}{f_h} = -\left(\frac{h}{k}\right)^{\frac{1}{\sigma}} \frac{(\alpha-1)}{\alpha} \equiv n$$
$$d = \frac{k}{h}$$
$$\frac{d(n)}{d(d)} = \frac{1}{\sigma}\frac{h}{k}^{1/\sigma-1} \frac{(\alpha-1)}{\alpha}$$
$$\frac{f_k/f_h}{h/k} = -\left(\frac{h}{k}\right)^{1/\sigma-1} \frac{(\alpha-1)}{\alpha}$$
$$\Rightarrow \boxed{\epsilon = \sigma}$$

b.Labor share

Given a function of capital and labor $Y = f(k,h)$, labor share is given by:

$$h_{share} = \frac{w}{p}\frac{h}{Y}$$

With competitive markets, $\frac{w}{p} = f_h$. Then[3]:

$$h_{share} = \frac{hf_h}{f(k,h)}$$

For the CES function we have:

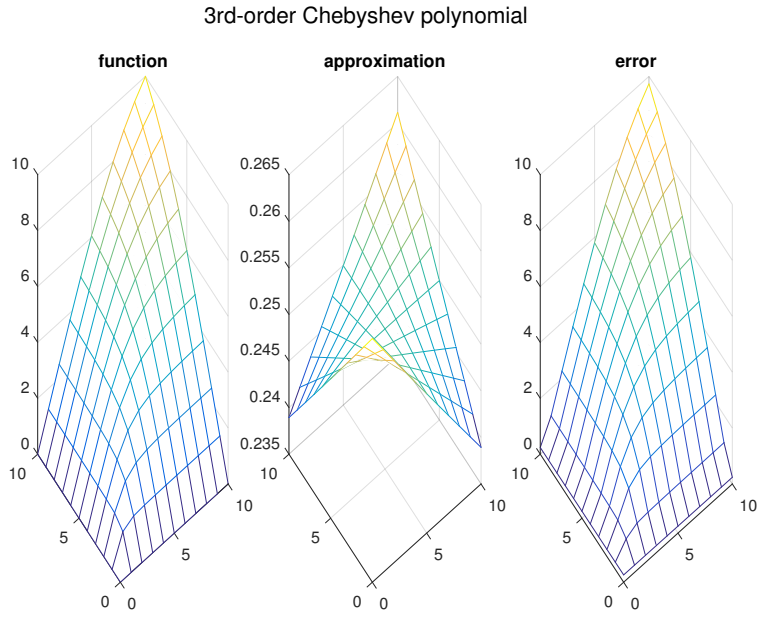
$$\begin{aligned} f_h &= \alpha h^{-1/\sigma} f(k,h)^{1/(\sigma-1)} \\ h_{share} &= \frac{\alpha h^{-1/\sigma} f(k,h)^{1/(\sigma-1)} h}{f(k,h)} \\ \Rightarrow h_{share} &= \frac{\alpha h k^{1/\sigma}}{h^{1/\sigma}k + \alpha h k^{1/\sigma} - \alpha h^{1/\sigma} k} \end{aligned}$$

c. Approximation of $f(k,h)$ with 2-dimensional Chebyshev regression algorithm

$\sigma = 0.25$[4]

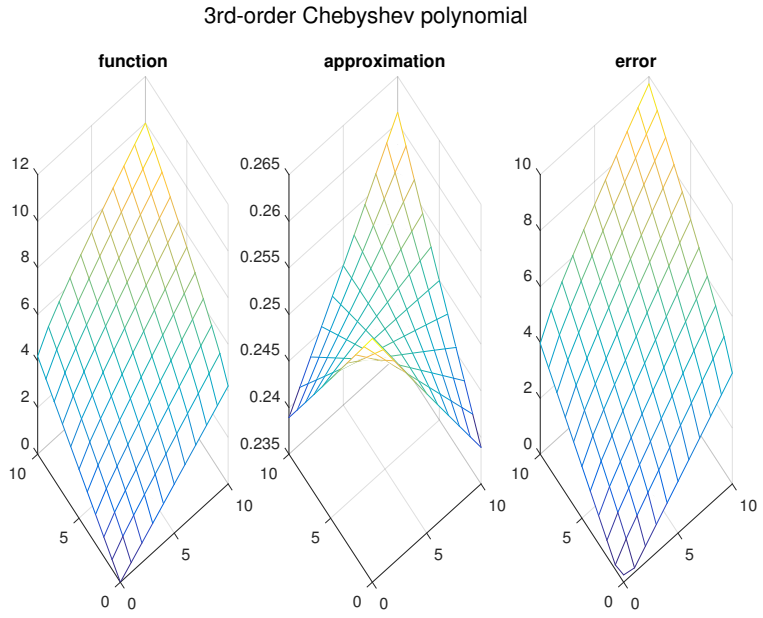
[3] where $f_h = \frac{\partial f(k,h)}{\partial h}$

[4] For sake of simplicity, I only put the plot for the 3rd-order in this documents. For higher-orders, there was not too much graphical difference. You can run the plot for the 6th,9th,12th orders in the code.



$$\boldsymbol{\sigma} = \mathbf{5}^{5}$$

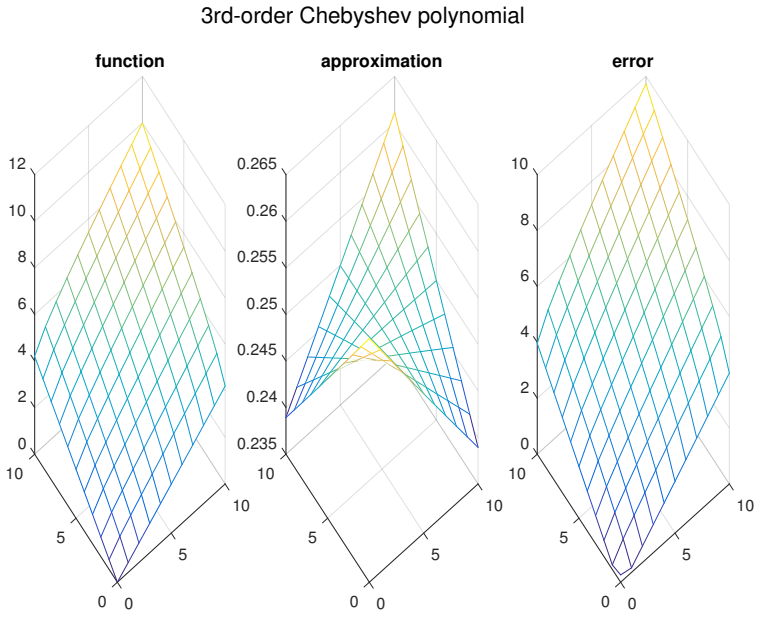
[5] For sake of simplicity, I only put the plot for the 3rd-order in this documents. For higher-orders, there was not too much graphical difference. You can run the plot for the 6th,9th,12th orders in the code.



$$\boldsymbol{\sigma = 1}$$ [6][7]

[6]for this case, I used $\sigma = 0.99$ to get the results, in order to see the behavior of the function in the limit - which is Cobb Douglas. Clearly, when σ is exactly one the function blows up to infinity

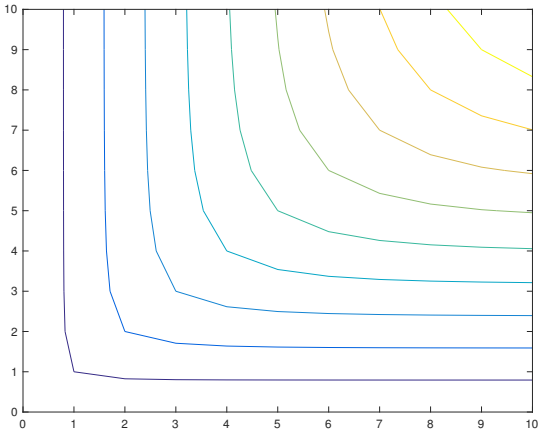
[7]For sake of simplicity, I only put the plot for the 3rd-order in this documents. For higher-orders, there was not too much graphical difference. You can run the plot for the 6th,9th,12th orders in the code.



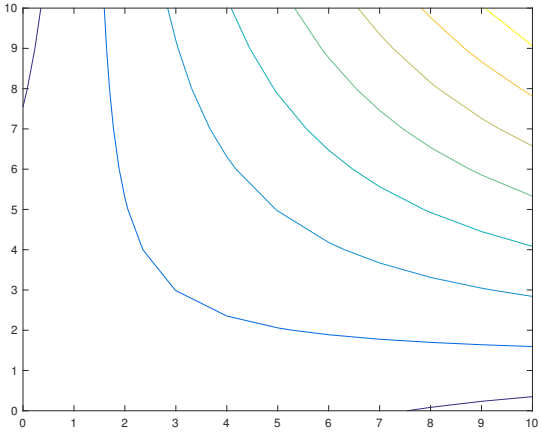
d. Percentiles Isoquants

$$\sigma = 0.25$$

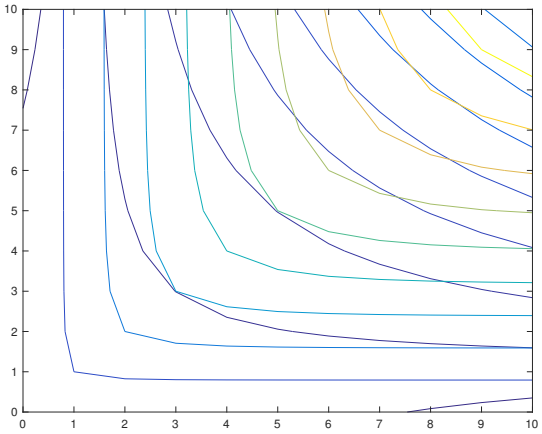
For the exact function:



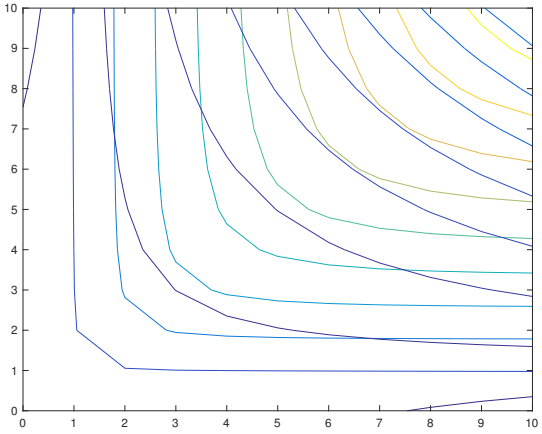
For the approximation:



Graphical comparison of isoquants above:

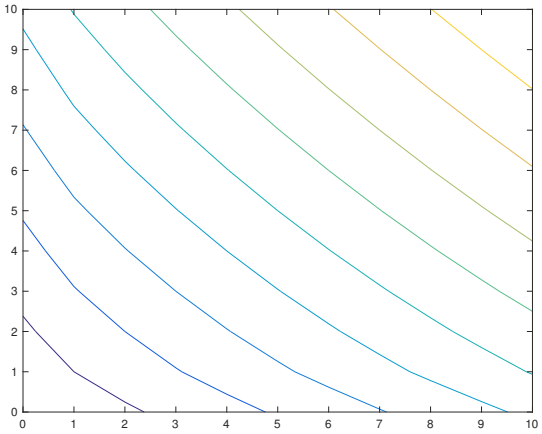


Errors and function appproximation:

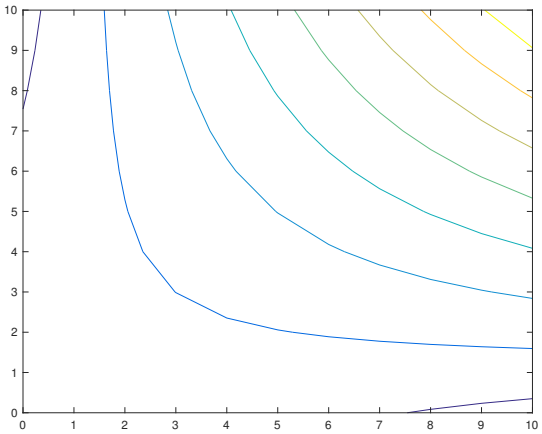


$\boldsymbol{\sigma = 5}$

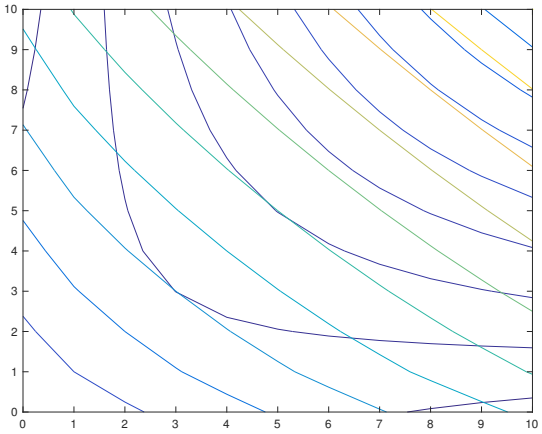
For the exact function:



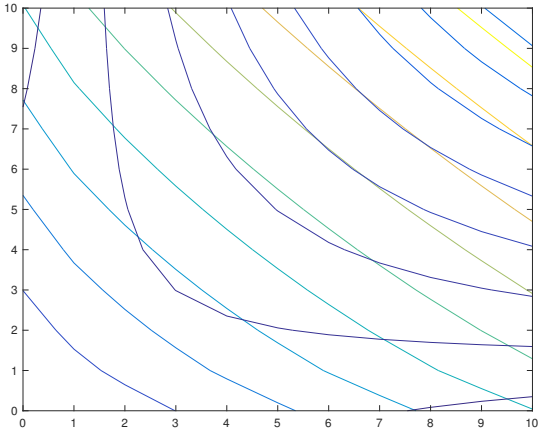
For the approximation:



Graphical comparison of isoquants above:

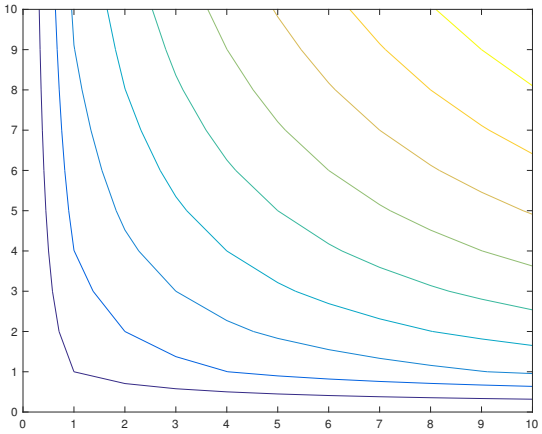


Errors and function appproximation:

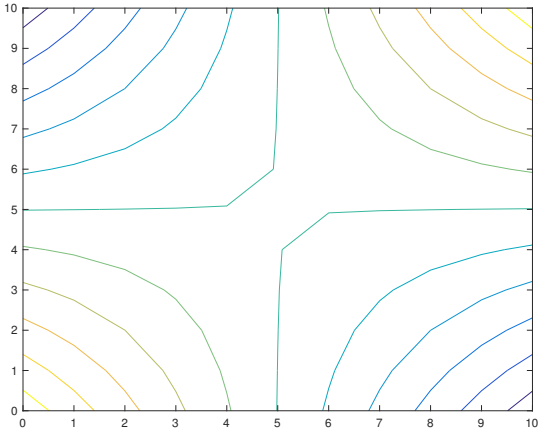


$\boldsymbol{\sigma = 1}$

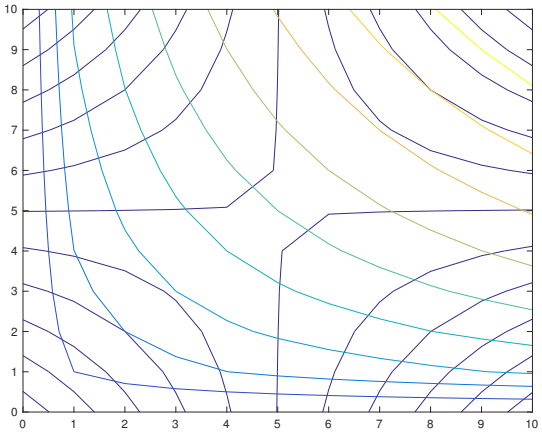
For the exact function:



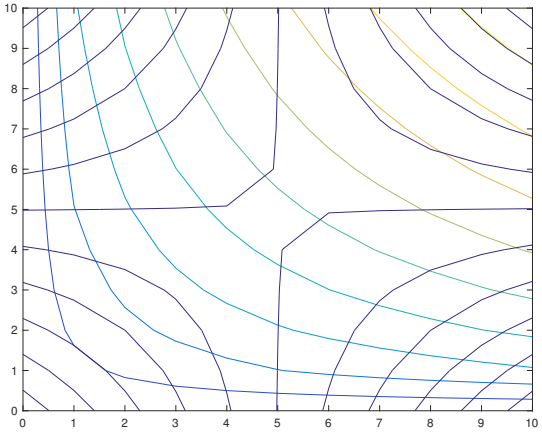
For the approximation:



Graphical comparison of isoquants above:



Errors and function appproximation:



Part III

Code

I've used Matlab to do these exercises. I've submit each question's code separatly for better understanding and analysis of each part; and to run the code more efficient[8]

The code then is dived as follows:

- hmw1.m : question I.1
- hmw2.m : question I.2
- hmw3a.m : question I.3, first topic
- hmw3b.m : question I.3, second topic
- hmw3c.m : question I.3, third topic
- hmw4.m : question I.4
- hmw51.m : question II (complete) for $\sigma = 0.25$
- hmw52.m : question II (complete) for $\sigma = 5$
- hmw53.m : question II (complete) for $\sigma = 1$

I've also created functions that I invoke in my code. These are:

- interpolation n.m : creates evenly-spaced nodes
- interpolation cheb.m : creates chebyshev nodes
- cheb poly.m: approximates a function using chebyshev nodes and polynomial
- taylorSeries.m: approximates a function by with a Taylor series

[8] If I compiled all in one code it would take too long to run it