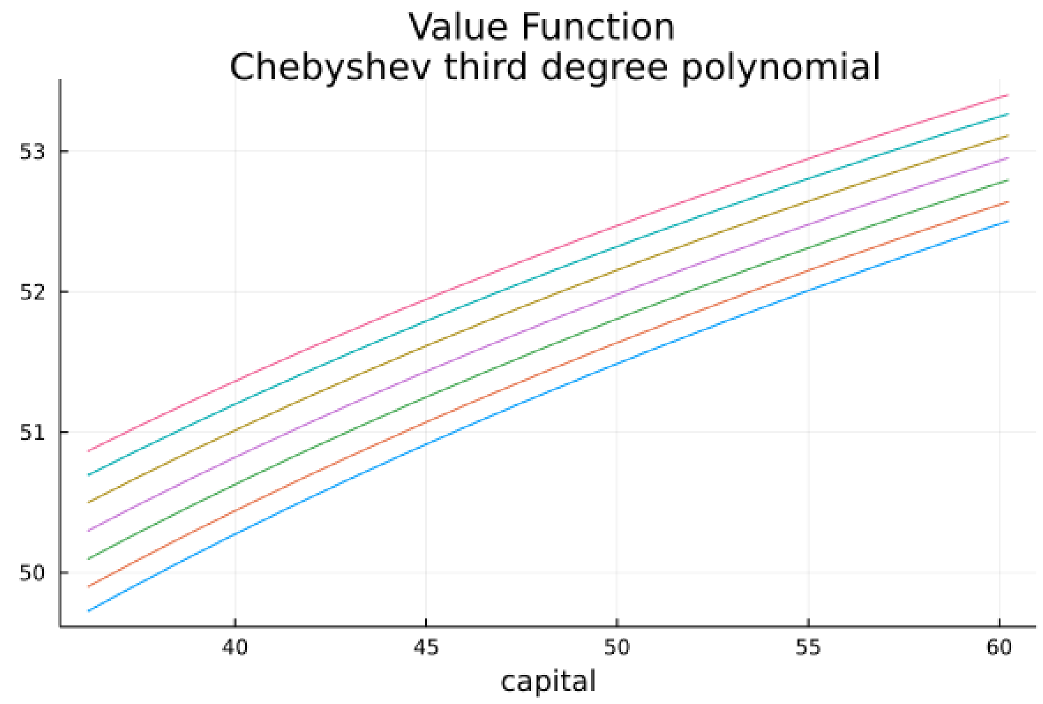
Métodos numéricos - Lista 3

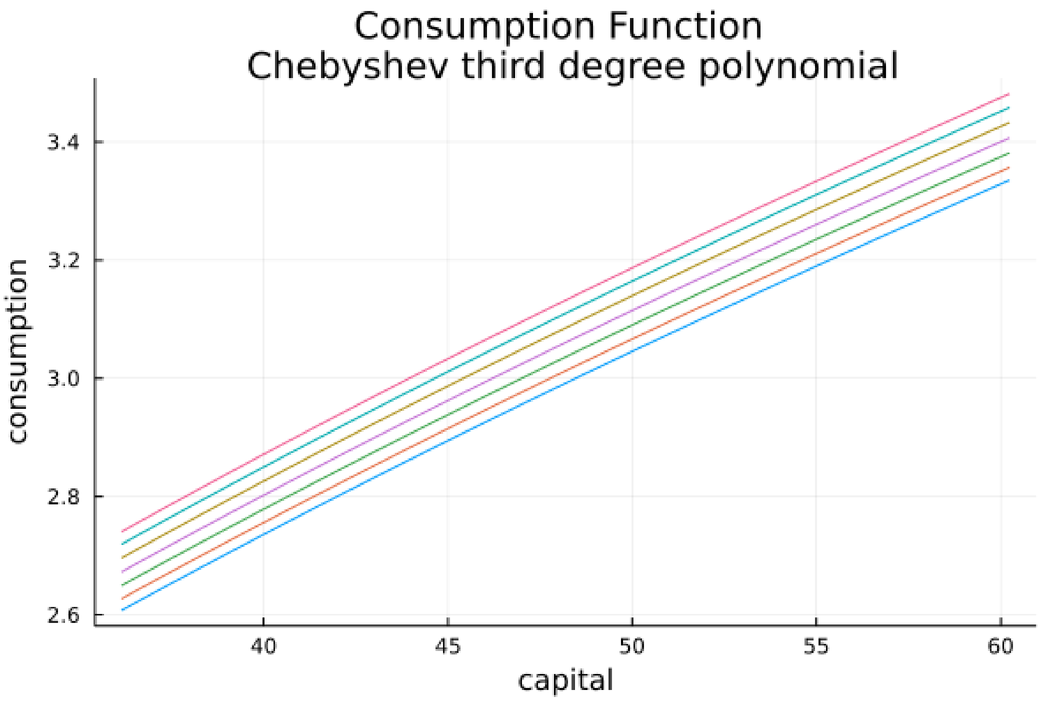
João Vitor Dias Gonçalves

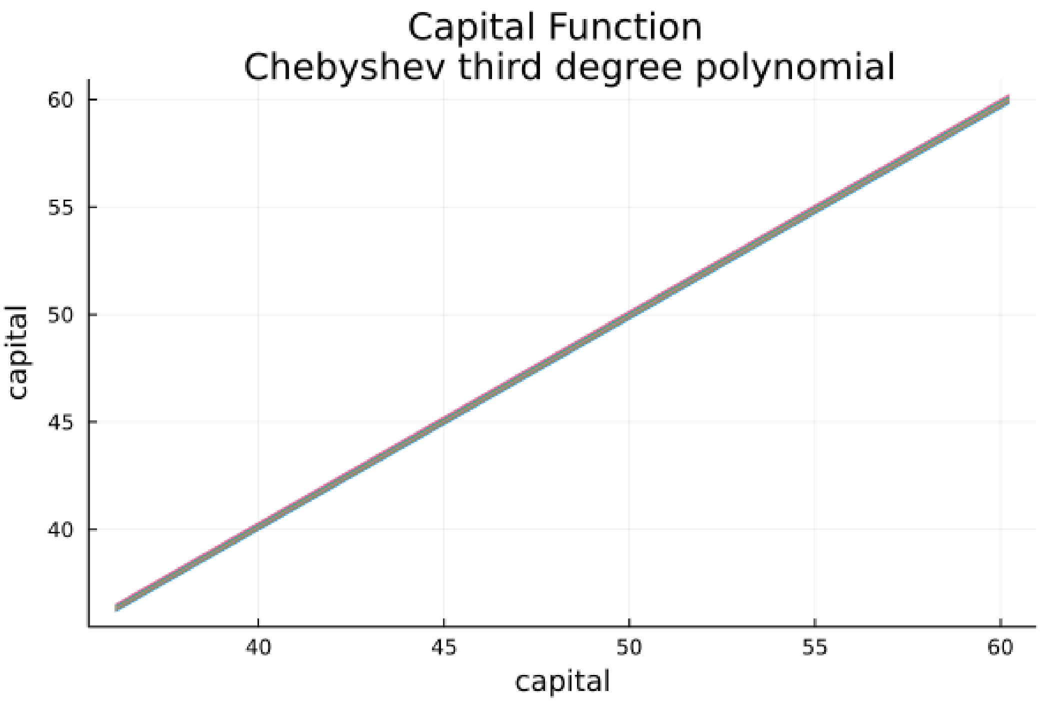
1º trimestre 2023

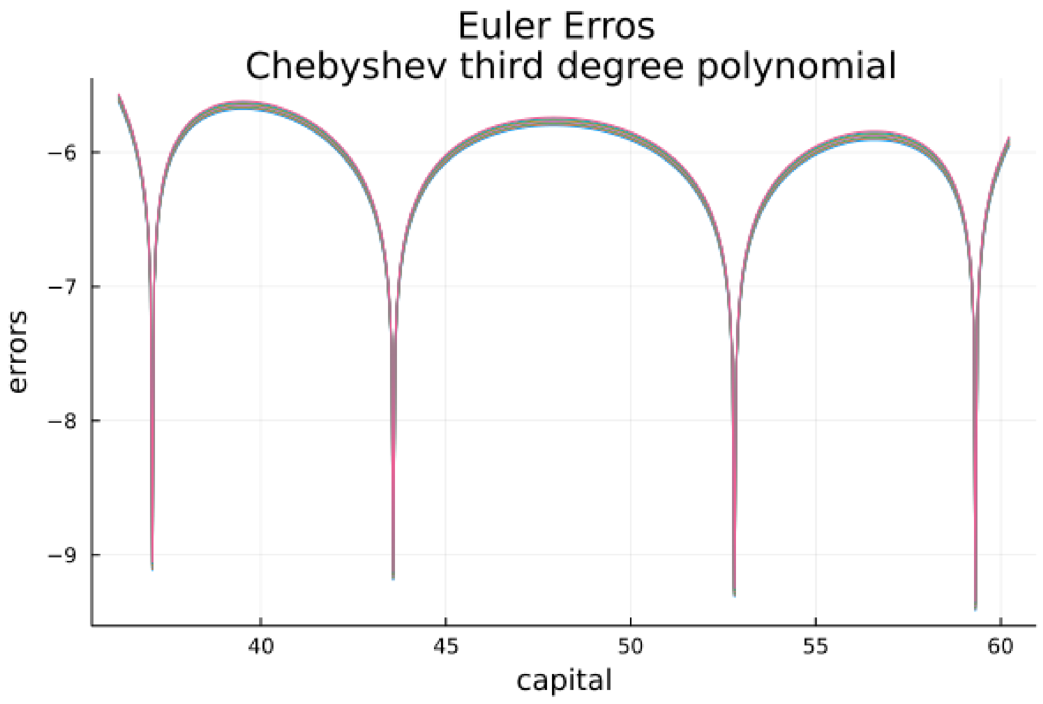
Question 1

The first thing I did in this exercise was creating two auxiliary functions, one to calculate the Chebyshevs polynomials, where the inputs are the capital and the degree of the polynomial, and another function to calculate the roots of the polynomial of degree d. After that I did my consumption function using $d+1$ polynomials of Chebyshev and adding them multiplied by the parameters *theta*. Then I can have a function with theta as input to calculate the residuals. Here $k0$ is a capital between -1 and 1, it will be our collocations points. So inside the function I have to transform this capital to the "normal size" one. In this function for each state I calculate our consumption. Using this consumption I calculate the capital of tomorrow and then the expected value of the consumption of tomorrow. The output of this residual function is a $1 \times NZ$ vectors. So we have a residual for each state based on the initial collocation point we used. In this method the collocation points are the roots of the Chebyshev polynomial of degree d. My function $final\_f$ uses the residual function to create the linear system. I apply the residual function for each collocation point. We have $d+1$ collocations points. So we have $d+1 \times NZ$ thetas to discover. The last function I created is one to solve the model and calculate the consumption, the capital and the value functions. As this method is very sensitive to the initial guess for the thetas I constructed a loop, where I use a initial guess for $d = 1$ and then add a line of zeros in the matrix of the estimated thetas and use it as initial guess for $d = 2$ and keep doing this until I find the d that I want. I used the function *solve_system* from the package *NLsolve.jl* to solve the linear system. I used a 2×7 matrix of ones in the first line and half in the second. It took 0.03 seconds to calculate the thetas for a 500 size capital grid and $d = 3$. The mean Euler Equation Error was -6, what means one dollar of error for each million spent, much more precise than the problem set 2, where we got a dollar for each thousand. Using a sixth degree polynomial took 0.08 seconds and got a -9 EEE. Here I'm posting the graphs for the third degree polynomial only since they are basically the same.


Value Function
Chebyshev third degree polynomial
53
52
51
50
40
45
50
55
60
capital

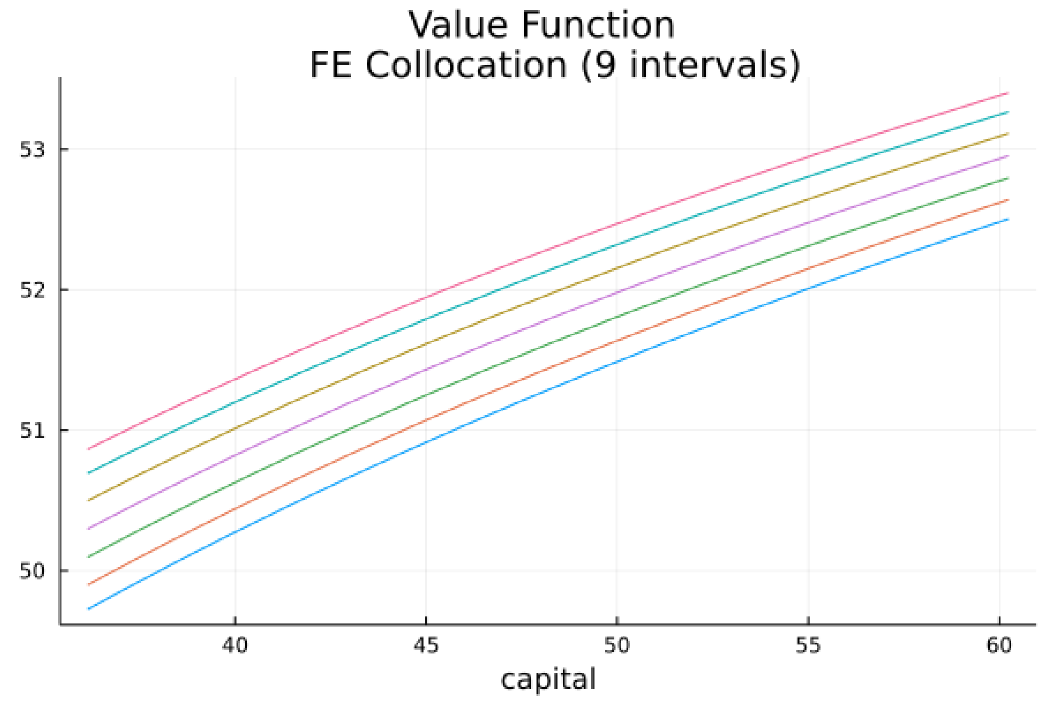

Consumption Function
Chebyshev third degree polynomial
consumption
3.4
3.2
3.0
2.8
2.6
40
45
50
55
60
capital

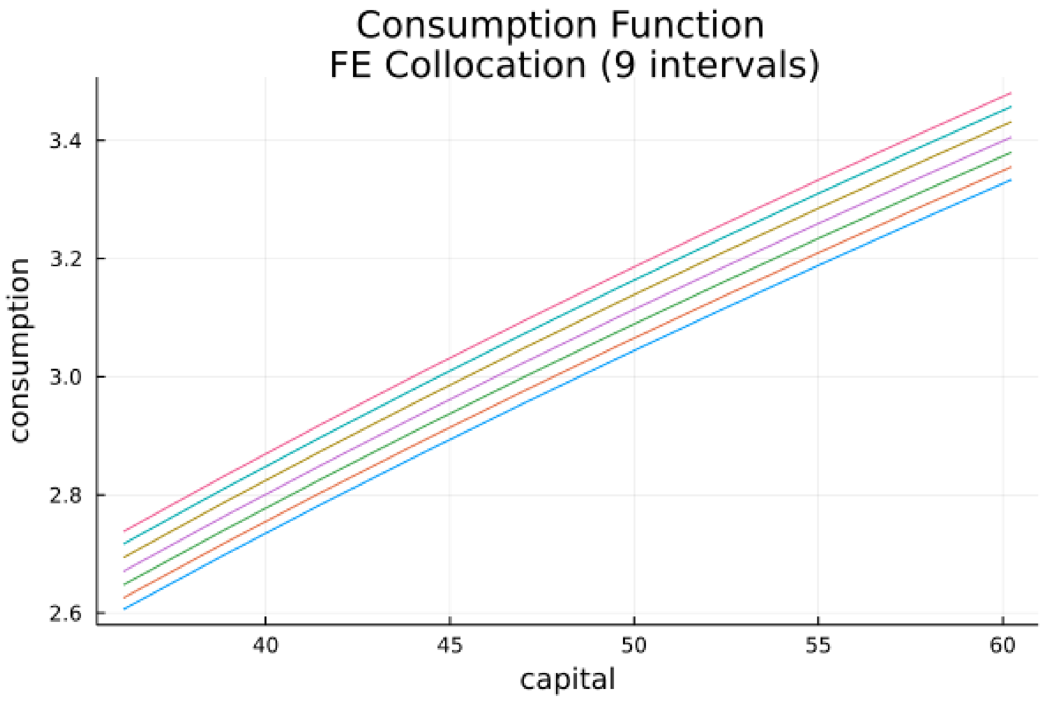

Capital Function
Chebyshev third degree polynomial
capital
capital
60
55
50
45
40
40
45
50
55
60

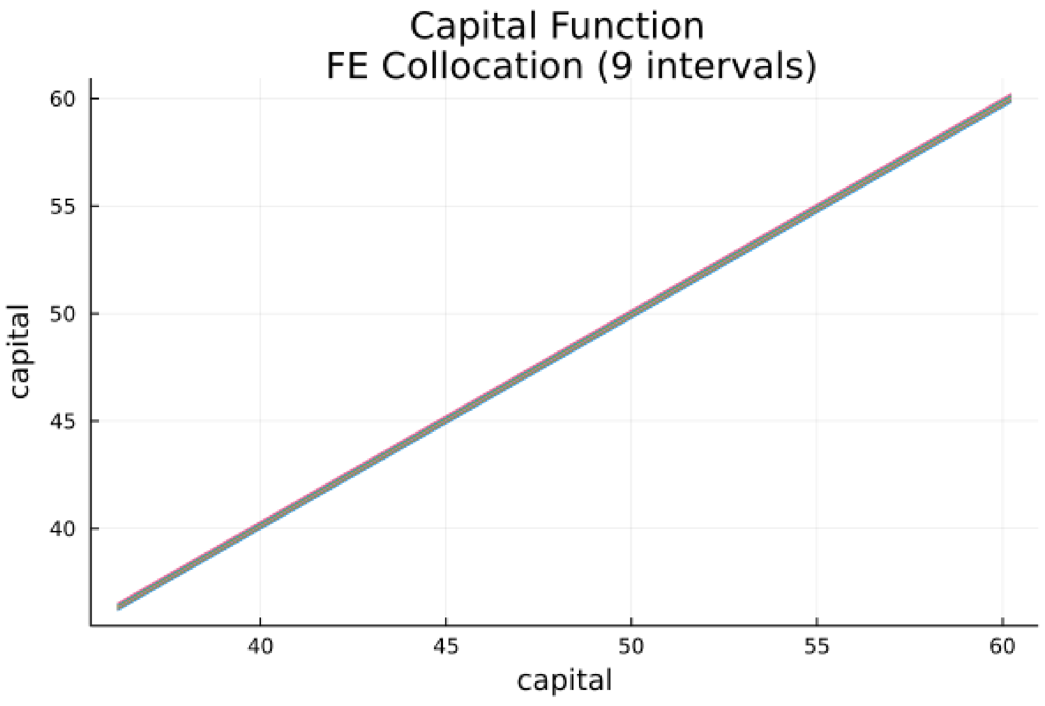

Euler Erros
Chebyshev third degree polynomial
errors
capital
−6
−7
−8
−9
40
45
50
55
60

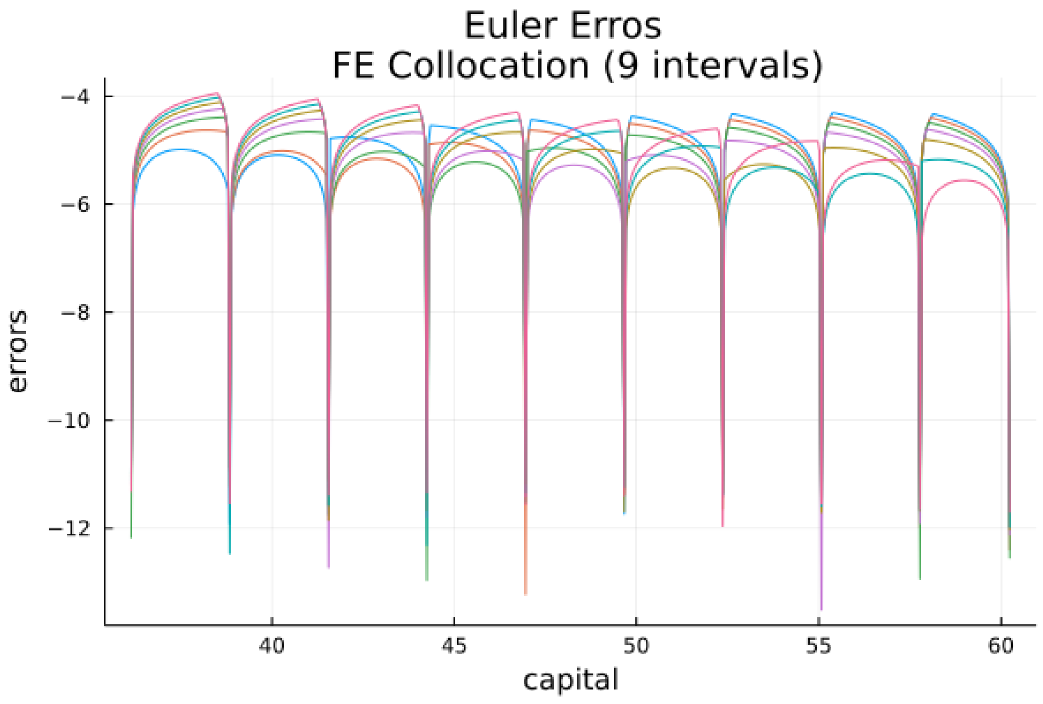
Question 2

For this exercise I tried to use the same nomenclature as before, since the problems are related. The differences are the polynomials and the collocation points only. Firstly I did a function to create the intervals. The output of the function is a $1 \times NKI$ vector, where NKI is the number of intervals plus one. The values of this vector are the bounds of the intervals. After that I created my ϕ_i functions as described in the lecture. The consumption function for this method is the sum of this ϕ_i functions multiplied by the thetas. The residual function here is similar to the last one, the difference is just the consumption function. Again I have a final function to create the linear system. Now we have $NKI \times NZ$ thetas to estimate. I tested two different initial guesses, one a linear spaced grid from one to ten and other a linear spaced grid from three to five. For both I also tested with 4 and 9 intervals. The four results were the same. Here I'm posting the 9 intervals and one to ten grid. The optimization code took 0.25 seconds and the EEE was equal to -5. I run with a 5000 capital grid to compare with the last problem set. This method is much slower than the concave and monotonic brute force method, but it is more accurate.

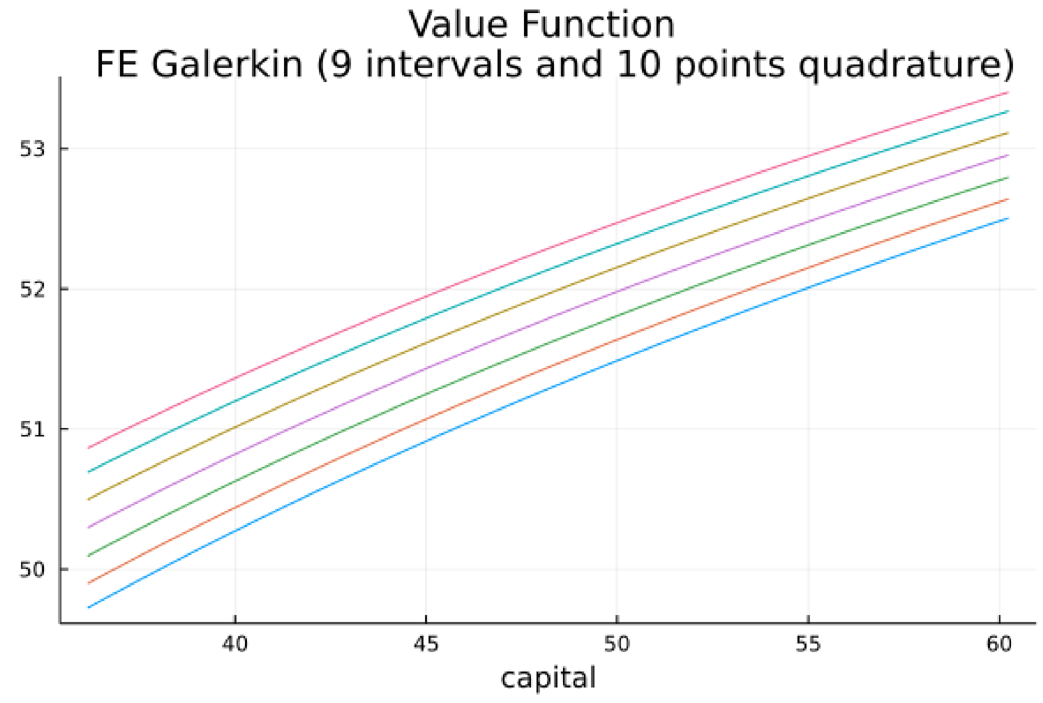


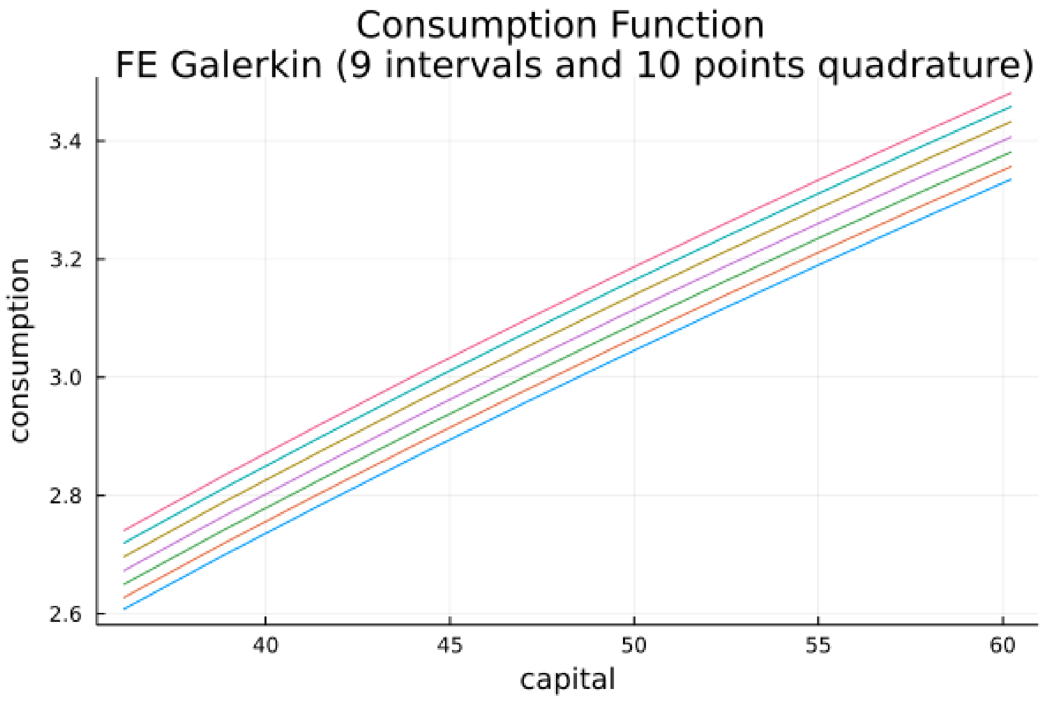

Consumption Function
FE Collocation (9 intervals)
consumption
capital
2.6
2.8
3.0
3.2
3.4
40
45
50
55
60

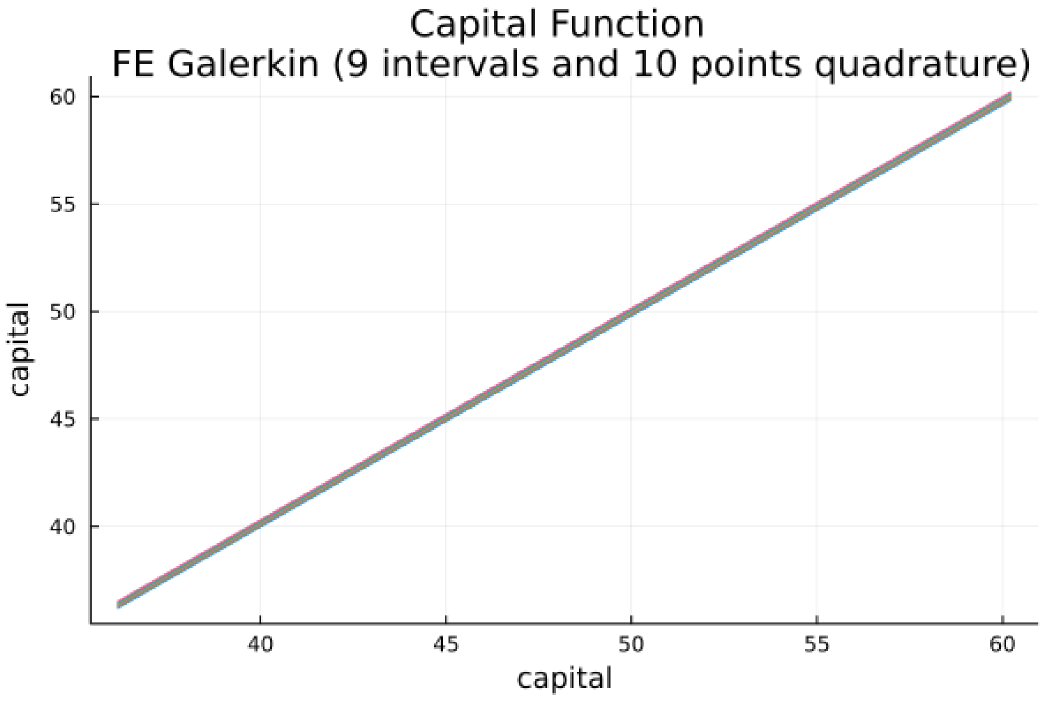

Capital Function
FE Collocation (9 intervals)
capital
capital
40
45
50
55
60
40
45
50
55
60

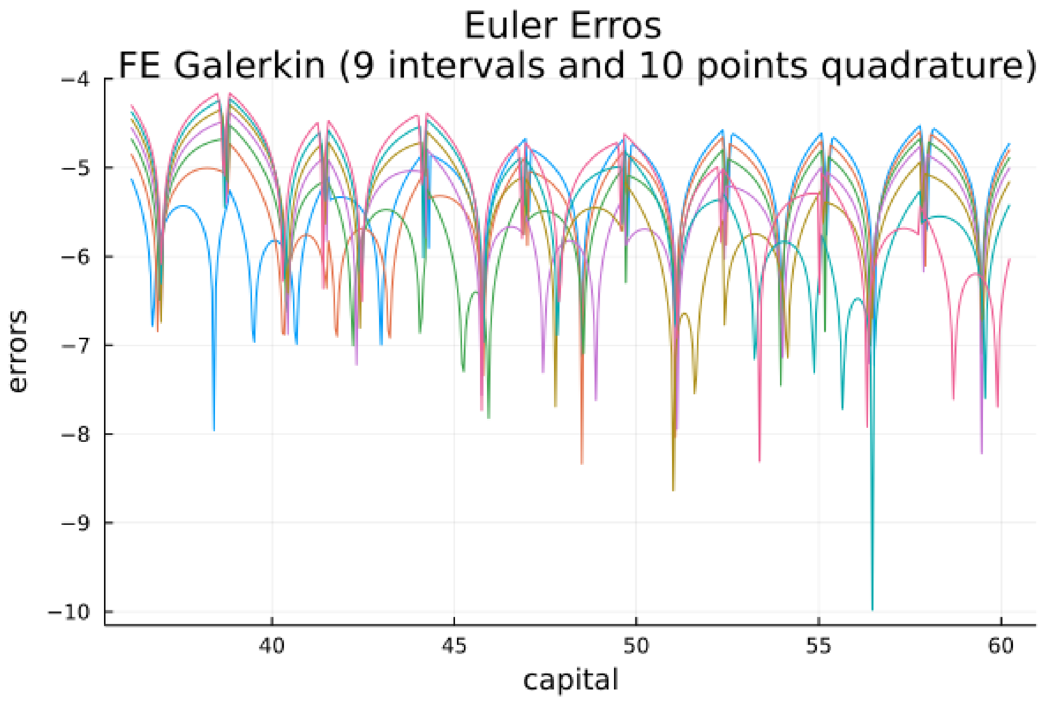

Euler Erros
FE Collocation (9 intervals)
errors
capital
−4
−6
−8
−10
−12
40
45
50
55
60

In the second part the only difference is that I had to create the $E(theta)$ vector. I did that in my E_i function. Here I opted for the Gauss Legendre quadrature. In Judd (1998) is said this is a fast and simple way to calculate them. Since the capital is not in the interval $[-1, 1]$, so I had to transform it, as in the first exercise. In this function I used the residual function from the last exercise. I used the function $gausslegendre$ from the package $FastGaussQuadrature.jl$ to calculate the nodes and weights for the approximation of the integral. Using the weights I did a inner product with the matrix of residuals, this is a $n \times NZ$ matrix, where n is the number of nodes in the quadrature. For each line I apply the residual function with one node as initial capital. This is similar to the collocation method we did before. The inner product results in a $1 \times NZ$ vector. The last function of the exercise again creates the linear system. I used the grid with 10 point from the last exercise. I tested with five and ten quadrature points, both result were the same. The EEE was -5.35 and it took 3.7 seconds to run. Here are displayed just the one with 10 quadrature points.




Consumption Function
FE Galerkin (9 intervals and 10 points quadrature)
consumption
capital
3.4
3.2
3.0
2.8
2.6
40
45
50
55
60


Capital Function
FE Galerkin (9 intervals and 10 points quadrature)
capital
capital
60
55
50
45
40
40
45
50
55
60


Euler Erros
FE Galerkin (9 intervals and 10 points quadrature)
errors
capital
−4
−5
−6
−7
−8
−9
−10
40
45
50
55
60