University of Minnesota
ECON 8581
Homework 1
Angelo Mendes

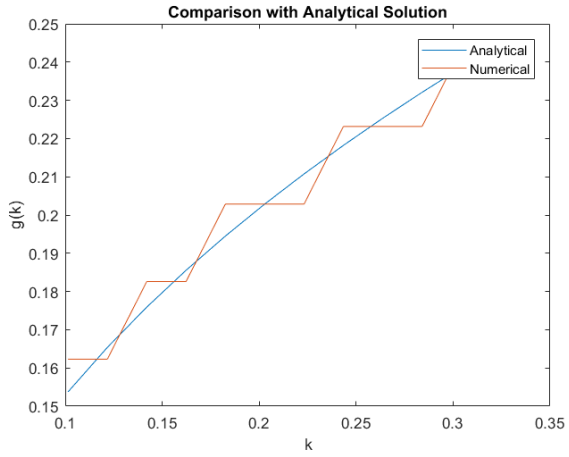
Prof. Fatih Guvenen
November 3, 2022

---

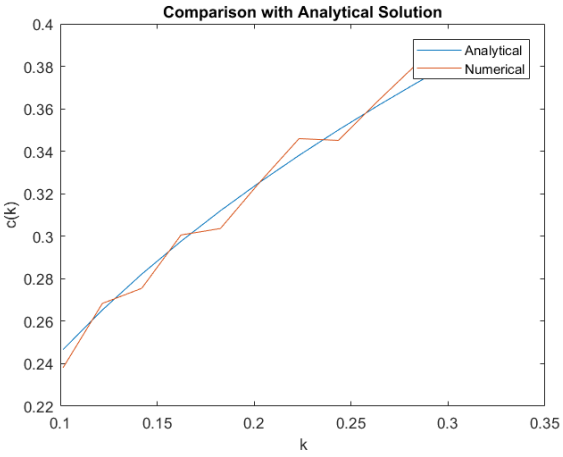
## Value Function Iteration

Along this section we run VFI with different number of gridpoints and parameters. Also, we introduce some algorithms to improve our codes. We start by the figures when $\delta = 1$.

$n = 11$

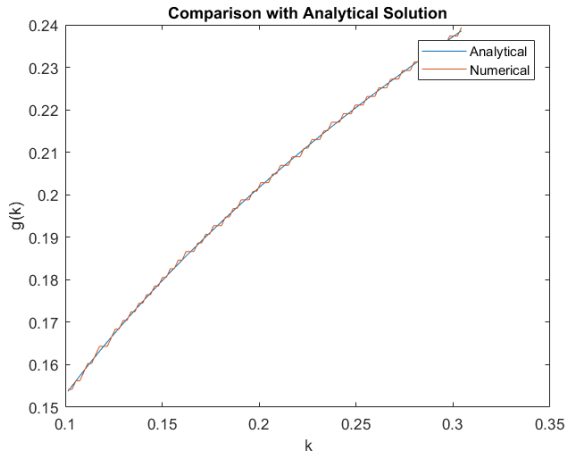



(a) $g(k)$

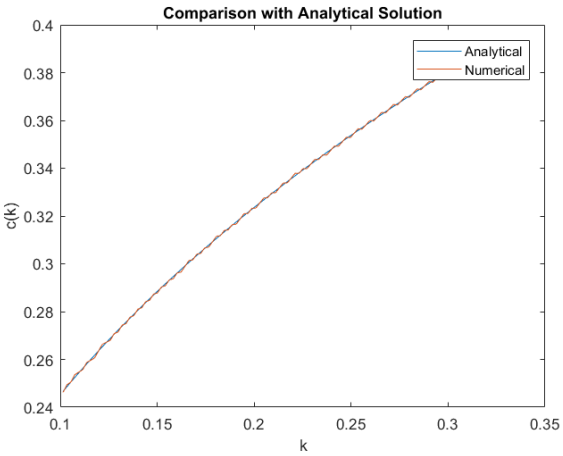



(b) $c(k)$

$n = 101$

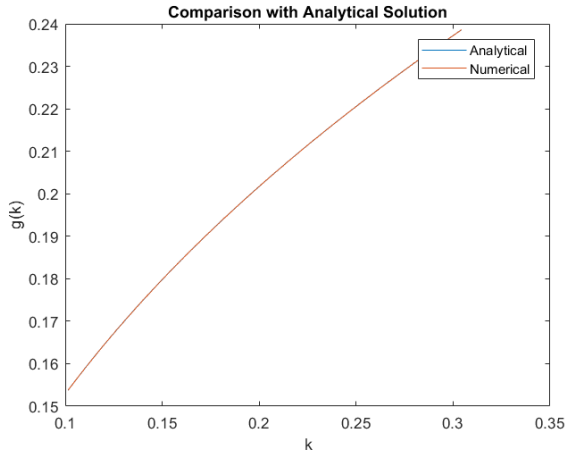



(c) $g(k)$

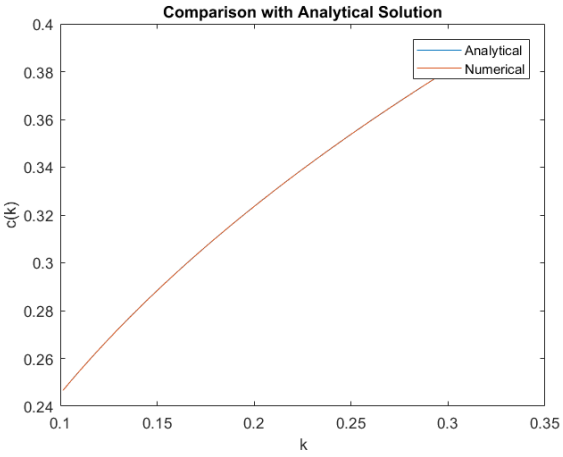



(d) $c(k)$

$n = 1001$



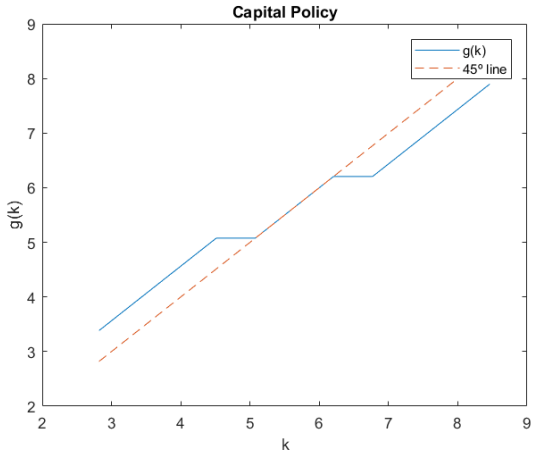

(e) $g(k)$



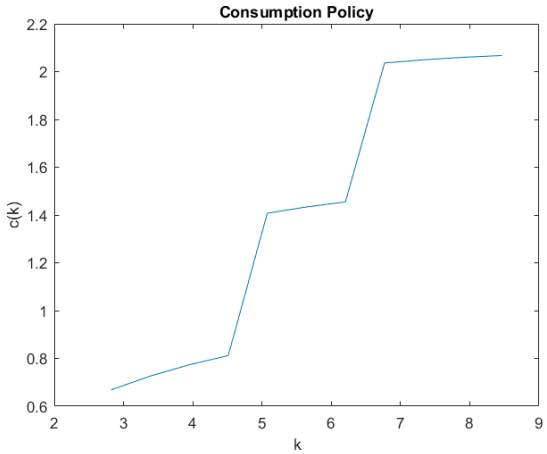

(f) $c(k)$

We now turn to the case $\delta = 0.1$

$n = 11$

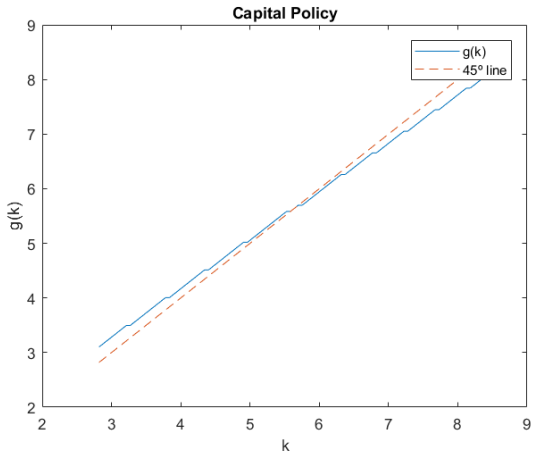



(g) $g(k)$

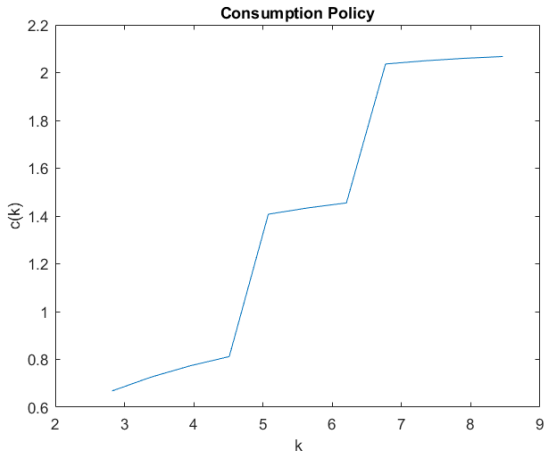



(h) $c(k)$

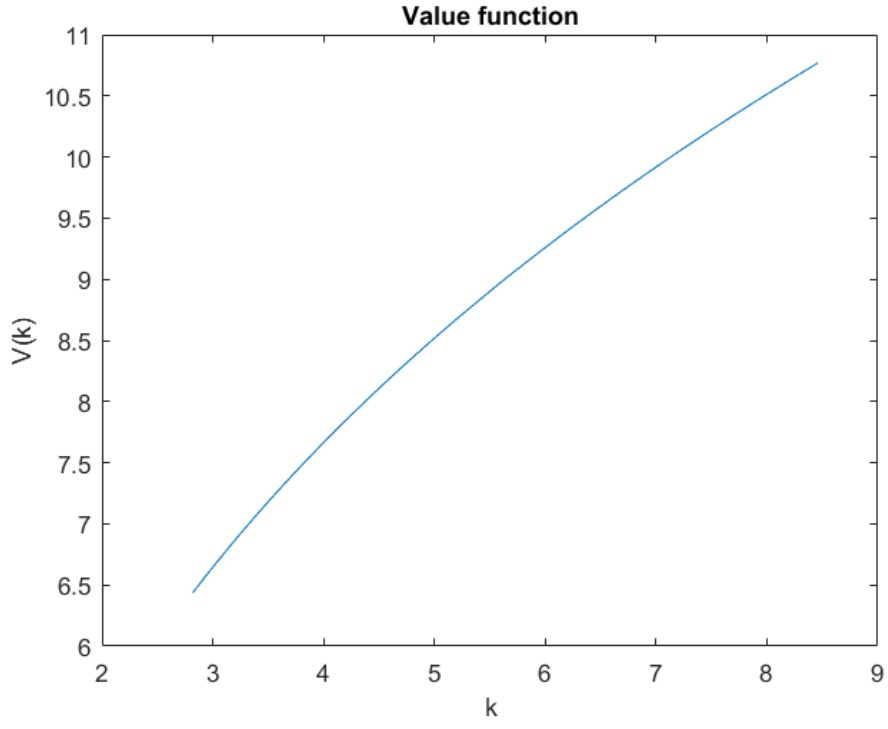
$n = 101$



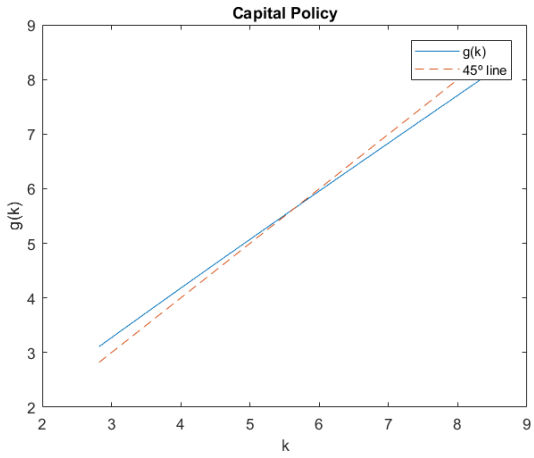

(i) $g(k)$



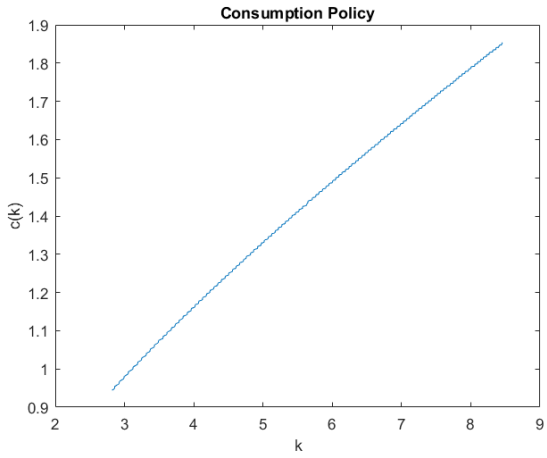

(j) $c(k)$



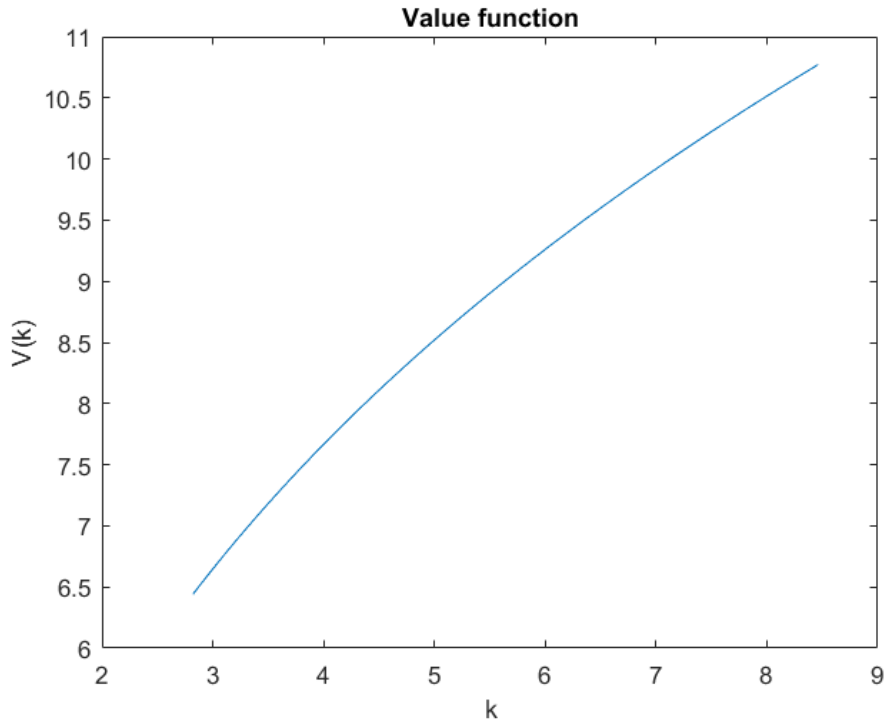

$n = 1001$




(k) $g(k)$




(l) $c(k)$




In exercise (b) we are asked to introduce Howard's algorithm to improve the codes.

| | $m=5$ | | $m=10$ | | $m=25$ | | $m=100$ | | $m=500$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| $n\backslash\delta$ | 0.1 | 1 | 0.1 | 1 | 0.1 | 1 | 0.1 | 1 | 0.1 | 1 |
| 11 | 0.009 | 0.12 | 0.006 | 0.009 | 0.003 | 0.004 | 0.003 | 0.004 | 0.008 | 0.01 |
| 101 | 0.25 | 0.26 | 0.13 | 0.15 | 0.07 | 0.07 | 0.06 | 0.038 | 0.08 | 0.06 |
| 1001 | 17.58 | 18.44 | 10.35 | 10.46 | 4.76 | 4.62 | 4.48 | 2.86 | 4.69 | 3 |

Now we implement the Mac-Queen Porteus bounds every iteration:

| | $m = 5$ | | $m = 10$ | | $m = 25$ | | $m = 100$ | | $m = 500$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| $n \backslash \delta$ | 0.1 | 1 | 0.1 | 1 | 0.1 | 1 | 0.1 | 1 | 0.1 | 1 |
| 11 | 0.08 | 0.003 | 0.007 | 0.001 | 0.005 | 0.002 | 0.006 | 0.003 | 0.006 | 0.01 |
| 101 | 0.19 | 0.03 | 0.11 | 0.03 | 0.07 | 0.029 | 0.05 | 0.04 | 0.077 | 0.05 |
| 1001 | 10.18 | 2.76 | 6.58 | 2.6 | 4.1 | 2.93 | 4.37 | 2.82 | 4.54 | 2.97 |

After that, I also implemented other algorithms to improve the code. First, using the fact that the policy function is monotone:

1. Compute $k'(k_1)$ by checking all $k_1, k_2, \ldots, k_N$, set $i = 1$;
2. Compute $k_{i+1}$ by checking all $k'(k_i), \ldots, k_N$;
3. If $i + 1 = N$, stop. Otherwise, increment $i$ and go to 2.

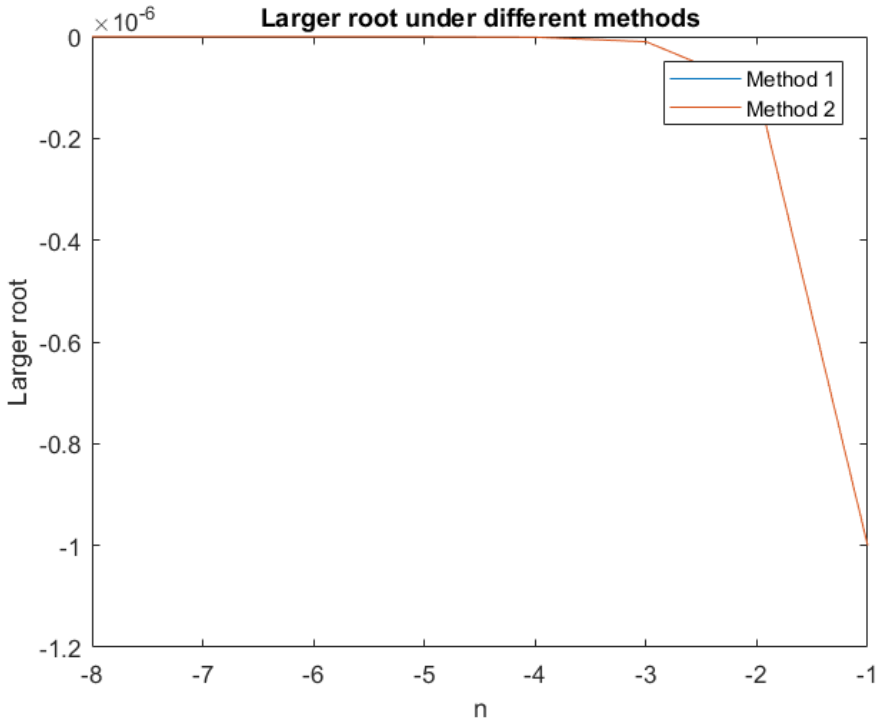
We can also exploit value function concavity. Let $H(k') = u(c(k, k')) + \beta V(k')$:

1. Compute $H(k' = k_1)$ be a known value. Let $i = 1$;
2. If $i = N$ or $H(k_{i+1}) < H(k_i)$, then stop. The optimum is $k' = k_i$. Otherwise, increment $i$ and go to 2.

By using both techniques, I was able to run the specification with $n = 1001$ and $\delta = 0.1$ in 2.3 seconds, with $m = 500$ and applying Mac-Queen Porteus bounds. Despite that, running the code in a (much) better computer, allowed me to reduce this to 0.98 seconds. This code could still be improved by parallelizing the loops or vectorizing the operations.

### Rounding Error

To be honest, I did not understand this exercise well. Using both methods the root is the same since actually we are just relabeling the one variable. It could be the case that storing a value in the computer and then using it would give a different result then computing the roots all at the same time. If this is the case, probably my computer is storing many decimal cases already.



**Golden Mean**

We start by proving the equivalence: $\phi^{n+1} = \phi^n - 1 - \phi^n$

$$\left(\frac{\sqrt{5}-1}{2}\right)^{n+1} = \left(\frac{\sqrt{5}-1}{2}\right)^{n} - 1 - \left(\frac{\sqrt{5}-1}{2}\right)^{n}$$

$$\left(\frac{\sqrt{5}-1}{2}\right)\left(\frac{\sqrt{5}-1}{2}\right)^{n} = \left(\frac{\sqrt{5}-1}{2}\right)^{n-1} - \left(\frac{\sqrt{5}-1}{2}\right)^{n}$$

$$\left(\frac{\sqrt{5}-1}{2}\right) = \left(\frac{\sqrt{5}-1}{2}\right)^{-1} - 1$$

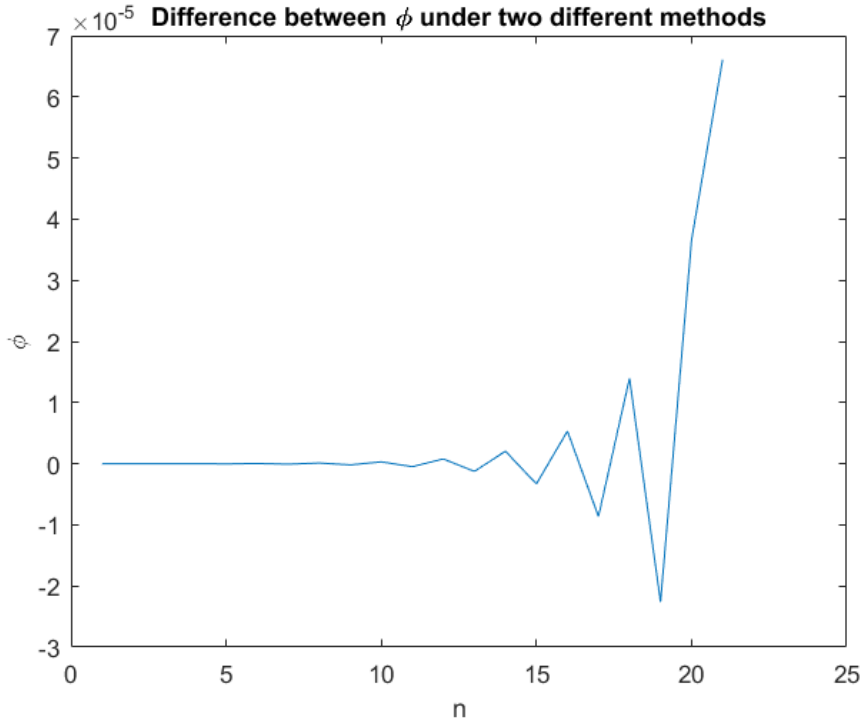
$$\left(\frac{\sqrt{5}-1}{2}\right) = \left(\frac{2}{\sqrt{5}-1}\right)^{-1} - 1$$

$$0 = 0$$

Following Numerical recipies in Fortran 77:

"There is another, different, kind of error that is a characteristic of the program or algorithm used, independent of the hardware on which the program is executed."[...]"The discrepancy between the true answer and the answer obtained in a practical calculation is called the truncation error. Truncation error would persist even on a hypothetical, "perfect" computer that had an infinitely accurate representation and no roundoff error."[...]"Sometimes, however, an otherwise attractive method can be unstable. This means that any roundoff error that becomes "mixed into" the calculation at an early stage is successively magnified until it comes to swamp the true answer. An unstable method would be useful on a hypothetical, perfect computer; but in this imperfect world it is necessary for us to require that algorithms be stable — or if unstable that we use them with great caution."

The example computed below shows a clear example of an unstable vs a stable algorithm. We construct 2 sequences following the two methods proposed in the exercise. We clearly see that when using the unstable algorithm the sequence diverges from its limit point.


Difference between $\phi$ under two different methods
$\times 10^{-5}$
7
6
5
4
3
2
1
0
-1
-2
-3
$\phi$
0
5
10
15
20
25
n