Macro III: Problem Set 3

Deadline: Friday, 17/09/2018

Aluno: Bruno Tebaldi de Queiroz Barbosa (C174887)

Professor: Tiago Cavalcanti

Source code disponível em: https://github.com/btebaldi/Macro3/tree/master/PSet_03

Questao 2

Item A

Limpeza de variaveis

```
clearvars
clc
```

Cria um processo de markov com as caracteristicas especificadas.

```
sigma = ((1-0.98^2)*0.621)^0.5;
mkv = MarkovProcess(0.98, sigma ,2,7,0);
mkv.AR.sigma2_y
```

```
ans = 0.6210
```

```
[chain,state] = MarkovSimulation(mkv.TransitionMatrix, 1000, mkv.StateVector, 3);
```

```
probability: 1.000000
Warning: The probabilities don`t sum to 1.
probability: 1.000000
Warning: The probabilities don`t sum to 1.
probability: 1.000000
Warning: The probabilities don`t sum to 1.
probability: 1.000000
Warning: The probabilities don`t sum to 1.
probability: 1.000000
Warning: The probabilities don`t sum to 1.
```

```
% plot(chain);
```

Item b

As *households* resolvem o seguinte problema de maximização:

$$max\left\{ E_0 \left[\sum_{t=0}^{\infty} \beta^t \left(\frac{C_t^{1-\sigma_c}}{1-\sigma_c} + \gamma \frac{(1-l_t)^{1-\sigma_l}}{1-\sigma_l} \right) \right] \right\}$$

sujeito a

$$a_{t+1} + c_t = (1+r)a_t + w_t z_t$$

$$\sigma_c, \sigma_l > 0$$

$$a_{t+1} \geq -\frac{wz}{r}$$

Item C

As firmas representativas resolvem o seguinte problema

$$max\left\{ K_t^{\alpha} N_t^{1-\alpha} - w_t N_t - r_t K_t \right\}$$

Item D

O Equilibrio recursivo stacionario é uma taxa de juros, r, uma taxa de salario, w, uma função política, g(a,z), e uma distribuição estacionária. Tal que:

1. Dado r e w, a função política g(a,z) resolve o problema do consumidor;
2. Dado r e w, a firma representativa maximiza os lucros; $w = F_N(K;N)$ e $r + \delta = F_K(K;N)$
3. Markets clear: $K_0 = \sum_{z,a} \lambda(a,z) a \;\; N = \pi' z$
4. A distribuição estacionária $\lambda(a,z)$ é induzida por $(P;z)$ e $g(a;z)$

$$\lambda(B) = \sum_{X=[a_L, a_U] \times Z \in B} Q(X,B)$$

Item E

Algoritimo:

1. Definir um valor para $r_j \in (-\delta, 1/\beta - 1)$;
2. Determinar o capital-labor ratio $r = F_K(k) - \delta$
3. Determinar w: $w = F_L(k)$;
4. Resolver o problema dos consumidores e determinar $a_{t+1}(a;z)$, $c_t(a;z)$, $l_t(a;z)$;
5. Calcular a distribuição estacionária $\lambda(a;z)$
6. Calcula a oferta de capital agregado e oferta de trabalho;
7. Avaliar o excesso de capital $D(r) = k - \frac{K}{L}$;
8. Se $D(r) > 0$, então $r_{j+1} > r_j$; se $D(r) < 0$, então $r_{j+1} < r_j$.
9. Interate until convergence.

```
eps = 1e-5;
```

```
eco_param.alpha = 0.4;
eco_param.beta = 0.96;
eco_param.delta = 0.08;
eco_param.gamma = 0.75;

eco_param.sigma_c = 2;
eco_param.sigma_l = 2;

% Determina os parametros r e w da economia
eco_param.r_UpperBound = 1/eco_param.beta -1;
eco_param.r_LowerBond = -eco_param.delta;
eco_param.r = (eco_param.r_UpperBound + eco_param.r_LowerBond)/2;
eco_param.r_tilda = eco_param.r + eco_param.delta;

% Determina os Grids
WageShocks.Values = exp(mkv.StateVector);
WageShocks.Grid.Min = min(WageShocks.Values);
WageShocks.Grid.Max = max(WageShocks.Values);
WageShocks.Grid.N = mkv.QtdStates;
WageShocks.PI = mkv.TransitionMatrix;

% Determina as caracteristicas do grid de trabalho
Labor.Grid.N = 20;
Labor.Grid.Min = 0.01;
Labor.Grid.Max = 1;
Labor.Values = linspace(Labor.Grid.Min, Labor.Grid.Max, Labor.Grid.N);

for nContador =1:100
```

Definir um valor para $r_j \in (-\delta, 1/\beta - 1)$;

```
    eco_param.r = (eco_param.r_UpperBound + eco_param.r_LowerBond)/2;
```

Determinar o capital-labor ratio $r = F_K(k) - \delta$

```
    k = ((eco_param.r + eco_param.delta)/eco_param.alpha)^(1/(eco_param.alpha -1));
```

Determinar w: $w = F_L(k)$;

```
    w = (1 - eco_param.alpha)*k^(eco_param.alpha);
```

Resolver o problema dos consumidores e determinar $a_{t+1}(a;z)$, $c_t(a;z)$, $l_t(a;z)$;

```
    % Como temos um limite natural do ativo vamos definir os grids.
    Asset.Grid.N = Labor.Grid.N;
    Asset.Grid.Min = - w * WageShocks.Grid.Min/eco_param.r;
    Asset.Grid.Max = 100;
    Asset.Values = linspace(Asset.Grid.Min, Asset.Grid.Max, Asset.Grid.N);

    [V0, U_Cube, Policy] = SolveConsumerProblem(Asset, Labor, WageShocks, w, eco_param);
```

Calcular a distribuição estacionária $\lambda(a;z)$

```
    Lambda = ConstructLambda(Policy, Asset, WageShocks);
```

Calcula a oferta de capital agregado e oferta de trabalho

```
    K = Lambda(:)' * Policy.Asset.Values(:);
    L = Lambda(:)' * Policy.Wages.Values(:);
    Demanda = k - K/L;
```

Se $D(r) > 0$, então $r_{j+1} > r_j$; se $D(r) < 0$, então $r_{j+1} < r_j$.

```
    if abs(Demanda) < eps
        break;
    elseif Demanda < -eps
        eco_param.r_UpperBound = eco_param.r;
    elseif  Demanda > eps
        eco_param.r_LowerBond = eco_param.r;
    end

    fprintf('Inter:%4d\tr: %1.6f\tDem: %2.6f\n', nContador, eco_param.r, Demanda);
```

```
Inter:   1 r: -0.019167 Dem: -11.197505
Inter:   2 r: -0.049583 Dem: 57.406843
Inter:   3 r: -0.034375 Dem: 15.903377
Inter:   4 r: -0.026771 Dem: 2.420929
Inter:   5 r: -0.022969 Dem: -3.823886
Inter:   6 r: -0.024870 Dem: -0.849313
Inter:   7 r: -0.025820 Dem: 0.768291
Inter:   8 r: -0.025345 Dem: -0.171065
Inter:   9 r: -0.025583 Dem: 0.312142
Inter:  10 r: -0.025464 Dem: -0.011769
Inter:  11 r: -0.025523 Dem: 0.232203
Inter:  12 r: -0.025494 Dem: 0.028110
Inter:  13 r: -0.025479 Dem: 0.008168
Inter:  14 r: -0.025471 Dem: -0.001801
Inter:  15 r: -0.025475 Dem: 0.003183
Inter:  16 r: -0.025473 Dem: 0.000691
Inter:  17 r: -0.025472 Dem: -0.000555
Inter:  18 r: -0.025473 Dem: 0.000068
Inter:  19 r: -0.025472 Dem: -0.000243
Inter:  20 r: -0.025473 Dem: -0.000088
```

```
end

fprintf('interest rate: %f\nwage rate: %f\n',eco_param.r, w);
array2table(Policy.AssetPrime.Values)
array2table(Policy.Labor.Values)
array2table(Lambda)
```

```
interest rate: -0.025473
wage rate: 2.265243
```

ans = *20×7 table*

...

	Var1	Var2	Var3	Var4	Var5	Var6
1	100.0000	18.3892	18.3892	18.3892	18.3892	22.6845
2	18.3892	22.6845	22.6845	22.6845	22.6845	26.9798
3	22.6845	26.9798	26.9798	26.9798	26.9798	31.2751
4	26.9798	26.9798	26.9798	26.9798	31.2751	35.5704
5	31.2751	31.2751	35.5704	35.5704	35.5704	39.8657
6	35.5704	35.5704	39.8657	39.8657	39.8657	44.1610
7	39.8657	39.8657	44.1610	44.1610	44.1610	48.4563
8	44.1610	44.1610	48.4563	48.4563	48.4563	48.4563
9	48.4563	48.4563	48.4563	52.7516	52.7516	52.7516
10	52.7516	52.7516	52.7516	57.0469	57.0469	57.0469
11	57.0469	57.0469	57.0469	61.3422	61.3422	61.3422
12	61.3422	61.3422	61.3422	65.6375	65.6375	65.6375
13	65.6375	65.6375	65.6375	69.9329	69.9329	69.9329
14	69.9329	69.9329	69.9329	74.2282	74.2282	74.2282
15	74.2282	74.2282	74.2282	78.5235	78.5235	78.5235
16	78.5235	78.5235	78.5235	78.5235	82.8188	82.8188

ans = *20×7 table*

...

	Var1	Var2	Var3	Var4	Var5	Var6
1	0.0100	0.8437	0.6874	0.5311	0.4268	0.8437
2	0.0100	0.8958	0.7395	0.5832	0.4789	0.8437
3	0.0100	0.9479	0.7395	0.5832	0.4789	0.8437
4	0.0100	0.0100	0.0100	0.0100	0.4789	0.8437
5	0.0100	0.0100	0.8437	0.6353	0.5311	0.8437
6	0.0100	0.0100	0.8958	0.6874	0.5311	0.8958
7	0.0100	0.0100	0.8958	0.7395	0.5311	0.8958
8	0.0100	0.0100	0.9479	0.7395	0.5832	0.4268

	Var1	Var2	Var3	Var4	Var5	Var6
9	0.0100	0.0100	0.0100	0.7916	0.5832	0.4789
10	0.0100	0.0100	0.0100	0.7916	0.6353	0.4789
11	0.0100	0.0100	0.0100	0.8437	0.6353	0.4789
12	0.0100	0.0100	0.0100	0.8437	0.6353	0.4789
13	0.0100	0.0100	0.0100	0.8958	0.6874	0.4789
14	0.0100	0.0100	0.0100	0.8958	0.6874	0.5311
15	0.0100	0.0100	0.0100	0.9479	0.6874	0.5311
16	0.0100	0.0100	0.0100	0.0100	0.7395	0.5311

ans = *20×7 table*

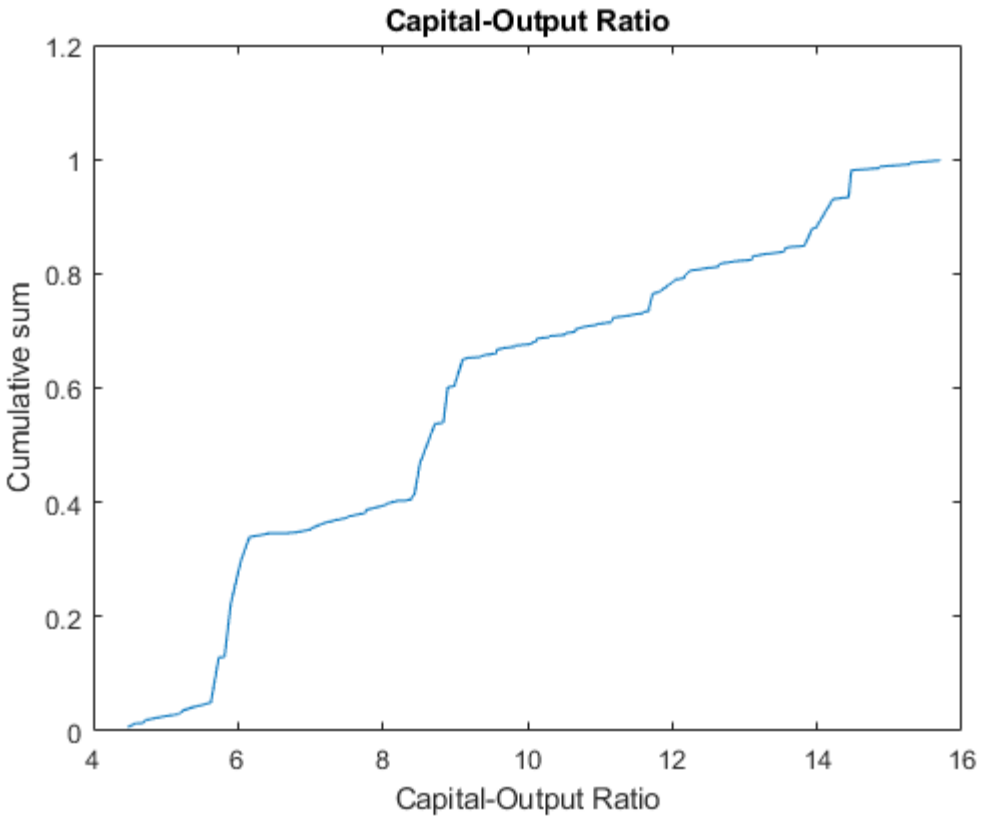
...

	Lambda1	Lambda2	Lambda3	Lambda4	Lambda5	Lambda6
1	0.0048	0.0065	0.0070	0.0056	0.0028	0.0001
2	0.0049	0.0067	0.0073	0.0059	0.0030	0.0002
3	0.0050	0.0784	0.0943	0.0769	0.0383	0.0018
4	0.0022	0.0040	0.0003	0.0001	0.0011	0.0016
5	0.0023	0.0042	0.0053	0.0051	0.0036	0.0016
6	0.0023	0.0043	0.0054	0.0051	0.0036	0.0016
7	0.0023	0.0043	0.0054	0.0052	0.0036	0.0016
8	0.0023	0.0043	0.0555	0.0689	0.0623	0.0416
9	0.0024	0.0020	0.0021	0.0030	0.0032	0.0026
10	0.0025	0.0019	0.0020	0.0029	0.0031	0.0025
11	0.0026	0.0019	0.0020	0.0028	0.0030	0.0025
12	0.0027	0.0018	0.0019	0.0027	0.0029	0.0024
13	0.0029	0.0017	0.0018	0.0027	0.0029	0.0024
14	0.0030	0.0015	0.0018	0.0026	0.0028	0.0023
15	0.0032	0.0013	0.0018	0.0264	0.0201	0.0096
16	0.0034	0.0012	0.0004	0.0015	0.0022	0.0022

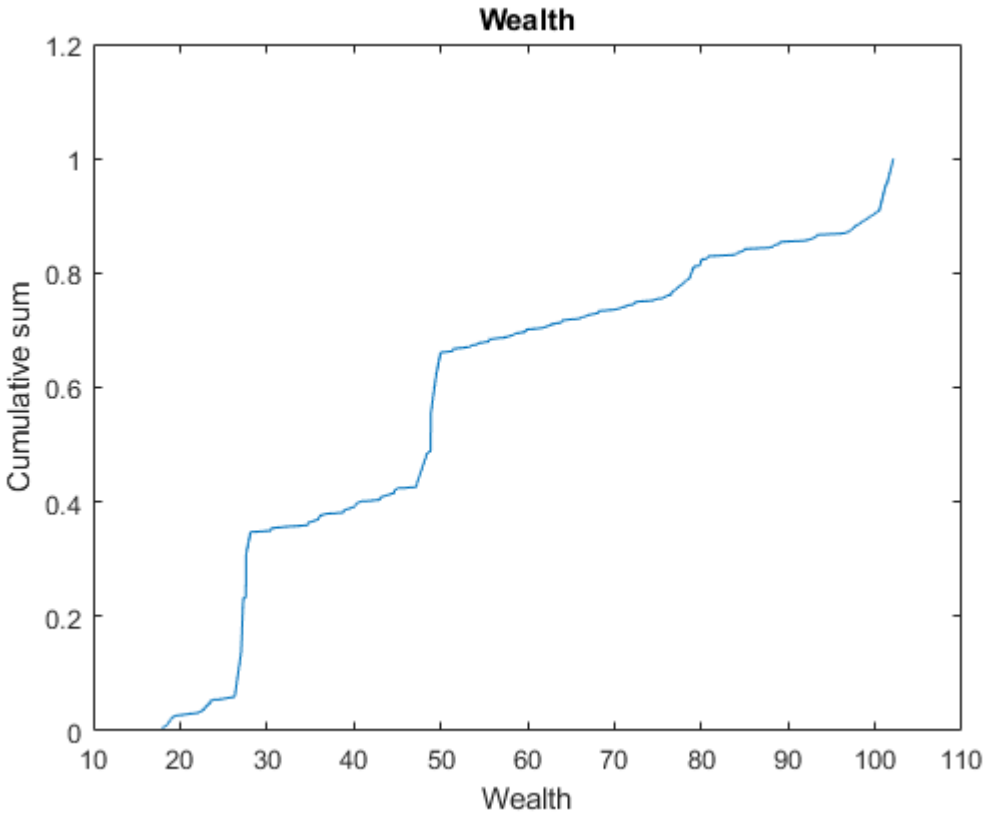
Determine Statistics

```
output = Policy.Asset.Values .^ eco_param.alpha + Policy.Labor.Values .^ (1-eco_param.alpha);

capital_output = Policy.Asset.Values ./ output;

[sortedK, indexK] = sort(capital_output(:));
lambda_aux = Lambda(:);
```

```
figure
plot(sortedK, cumsum(lambda_aux(indexK)));
title('Capital-Output Ratio');
xlabel('Capital-Output Ratio');
ylabel('Cumulative sum');
```




```
[sortedK, indexK] = sort(Policy.Wealth.Values(:));
lambda_aux = Lambda(:);
figure
plot(sortedK, cumsum(lambda_aux(indexK)));
title('Wealth');
xlabel('Wealth');
ylabel('Cumulative sum');
```


Wealth
Cumulative sum
Wealth
0
0.2
0.4
0.6
0.8
1
1.2
10
20
30
40
50
60
70
80
90
100
110