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

QUESTÃO 1 (a)-(e)

LIMPEZA DE VARIAVEIS

```
clearvars
clc
```

ITEM A DEFINICAO DE PARAMETROS

Defininimos os parametros que sao utilizados pelo problema

```
Econom_param.PeriodsPerYear = 6;
Econom_param.Beta_anual = 0.96;
Econom_param.Beta = (Econom_param.Beta_anual)^(1/Econom_param.PeriodsPerYear);
Econom_param.Sigma = 1.5;

eps = 1e-5;
```

DEFINICAO DO GRID DE INCOME

Vamos Construir a matrix de transição baseada nas informações fornecidas

```
Income.Grid.N = 2;
Income.Grid.Max = 1;
Income.Grid.Min = 0.1;
Income.Values = linspace(Income.Grid.Max, Income.Grid.Min, Income.Grid.N);
Income.PI = [0.925 (1-0.925); 0.5 (1-0.5)];
Income.P_LongRun = CalculaLongoPrazo(Income.PI);
```

Logo a renda media de equilibrio no longo prazo sera.

```
Income.Average = Income.Values*Income.P_LongRun;
Income.Early = Income.Average * Econom_param.PeriodsPerYear;
```

A duração média do desemprego é dada por 1/f, onde f neste caso é 0.5. Sendo assim a duracao média do desemprego é dois periodos. Como o periodo é de dois meses, temos que a duração média de desemprego é de 4 meses.

DEFINICAO DO GRID de ASSET

```
Borrow.Limit = -Income.Early;
Asset.Grid.N = 20;
Asset.Grid.Max = 3*Income.Average;
Asset.Grid.Min = Borrow.Limit;
Asset.Values = linspace(Asset.Grid.Min, Asset.Grid.Max, Asset.Grid.N);
```

DEFINICAO DA TAXA DE JUROS

Inicia a taxa de juros

```
Econom_param.r_anual = 0.034;
Econom_param.r_start = ((1+Econom_param.r_anual)^(1/Econom_param.PeriodsPerYear))-1;
Econom_param.r_UpperLimit = 1/Econom_param.Beta - 1;
Econom_param.r_LowerLimit = 0;
Econom_param.r = Econom_param.r_start;

% parametro que indica se a taxa esta fixa ou nao.
Econom_param.r_IsFixed = 0;
```

ACHANDO O EQUILIBRIO.

1. Determinar um r inicial
2. Resolver o Problema do Consumidor e obter a Politica
3. Determinar Lambda
4. Verificar se existe eequilibrio no mercado de assets
5. Se nao ha equilibrio estabelecer novo r, (voltar a ponto 2)

   onde $e > 0 \rightarrow r_j + 1 < r_j \quad e < 0 \rightarrow r_j + 1 > r_j$

```
nIntLimit = 1000;

for i=1:nIntLimit

    % (2) Resolve o Problema do consumidor
    [V0, U_Cube, Policy] = SolveConsumerProblem(Asset, Income, Econom_param);

    % (3) Determina a distribuicao estacionaria
    Lambda = ConstructLambda(Policy, Asset, Income);

    % (4) Determina demanda de assets
    Demanda = Lambda(:)' * Policy.AssetPrime.Values(:);

    % (5) Verifica se há equilibrio
    if abs(Demanda) < eps
        break;
    elseif Demanda > eps
        Econom_param.r_UpperLimit = Econom_param.r;
```

```
    elseif  Demanda < -eps
        Econom_param.r_LowerLimit = Econom_param.r;
    end

    % Caso a taxa esteja fixa não ha mais nada o que fazer
    if Econom_param.r_IsFixed == 1
        break;
    end

    % determina novo r
    r_old = Econom_param.r;
    Econom_param.r = (Econom_param.r_UpperLimit + Econom_param.r_LowerLimit)/2;

    % Caso a precisao da taxa seja muito pequena paramos a execucao.
    if abs(Econom_param.r - r_old) < eps^2
        break;
    end

    fprintf('Inter:%4d\tr_0: %1.6f\tr_1: %1.6f\tDem: %2.15f\n', i, r_old, Econom_param.r, Dema
end
```

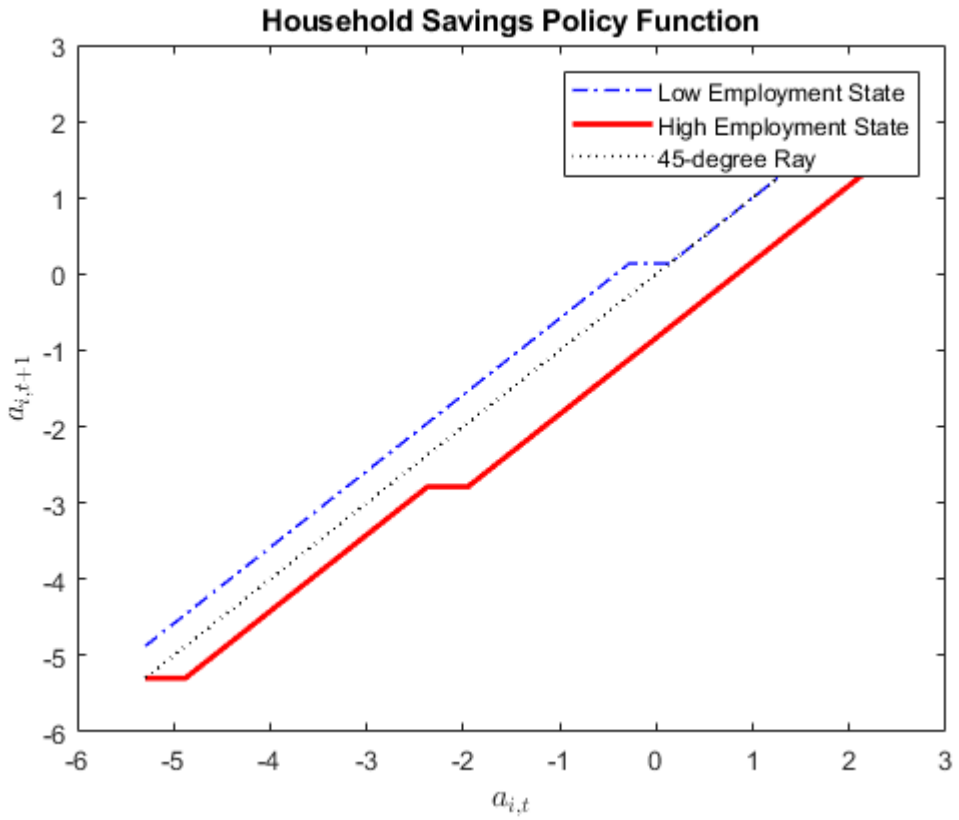
```
Inter:   1 r_0: 0.005588 r_1: 0.006207 Dem: -0.739618742602491
Inter:   2 r_0: 0.006207 r_1: 0.005898 Dem: 0.036345050466770
Inter:   3 r_0: 0.005898 r_1: 0.006053 Dem: -0.339060751860335
Inter:   4 r_0: 0.006053 r_1: 0.006130 Dem: -0.339060751860335
Inter:   5 r_0: 0.006130 r_1: 0.006169 Dem: -0.339060751860335
Inter:   6 r_0: 0.006169 r_1: 0.006149 Dem: 0.036345050466770
Inter:   7 r_0: 0.006149 r_1: 0.006159 Dem: -0.339060751860335
Inter:   8 r_0: 0.006159 r_1: 0.006164 Dem: -0.339060751860335
Inter:   9 r_0: 0.006164 r_1: 0.006161 Dem: 0.036345050466770
Inter:  10 r_0: 0.006161 r_1: 0.006163 Dem: -0.339060751860335
Inter:  11 r_0: 0.006163 r_1: 0.006163 Dem: -0.339060751860335
Inter:  12 r_0: 0.006163 r_1: 0.006163 Dem: 0.036345050466770
Inter:  13 r_0: 0.006163 r_1: 0.006163 Dem: 0.036345050466770
Inter:  14 r_0: 0.006163 r_1: 0.006163 Dem: -0.339060751860335
Inter:  15 r_0: 0.006163 r_1: 0.006163 Dem: -0.339060751860335
Inter:  16 r_0: 0.006163 r_1: 0.006163 Dem: 0.036345050466770
Inter:  17 r_0: 0.006163 r_1: 0.006163 Dem: -0.339060751860335
Inter:  18 r_0: 0.006163 r_1: 0.006163 Dem: 0.036345050466770
Inter:  19 r_0: 0.006163 r_1: 0.006163 Dem: -0.339060751860335
Inter:  20 r_0: 0.006163 r_1: 0.006163 Dem: 0.036345050466770
Inter:  21 r_0: 0.006163 r_1: 0.006163 Dem: -0.339060751860335
Inter:  22 r_0: 0.006163 r_1: 0.006163 Dem: -0.339060751860335
Inter:  23 r_0: 0.006163 r_1: 0.006163 Dem: 0.036345050466770
```

```
% Limpa variaveis nao utilizadas
clear nIntLimit r_old
```

GRAFICO DA FUNCAO POLITICA DE A_{t+1} CONTRA A_t

```
fig1 = figure();
plot(Policy.AssetDomain, Policy.AssetPrime.Values(:,1),'b-.','LineWidth',1); hold on;
plot(Policy.AssetDomain, Policy.AssetPrime.Values(:,2),'r','LineWidth',2); hold on;
plot(Policy.AssetDomain, Policy.AssetDomain,'k:','LineWidth',1); hold off;
```

```
legend({'Low Employment State','High Employment State','45-degree Ray'});
title('Household Savings Policy Function');
xlabel('$a_{i,t}$','FontSize',12,'Interpreter','latex');
ylabel('$a_{i,t+1}$','FontSize',12,'Interpreter','latex');
```



SIMULACAO DE LIFE HISTORY

define uma funcao que indica, dado um valor de asset, qual o valor do grid que mais se aproxima do valor informado. Em outras palavras informa aonde um valor fornecido estaria no grid.

```
Policy.AssetIndexFunction = @(value) round((value-Policy.AssetDomain(1))/(Policy.AssetDomain(2

% Total de simulacoes
Simulation.N = 10000;

% Nivel inicial de ativos
asset_0 = 0;

% Estado inicial: Empregado.
Simulation.S0 = 1;
Simulation.Wage = 1;

% Inicializa os vetores da simulacao.
Simulation.Asset = nan(Simulation.N,1);
Simulation.AssetIndex = nan(Simulation.N,1);
Simulation.Consumption = nan(Simulation.N,1);
Simulation.Investment = nan(Simulation.N,1);
```

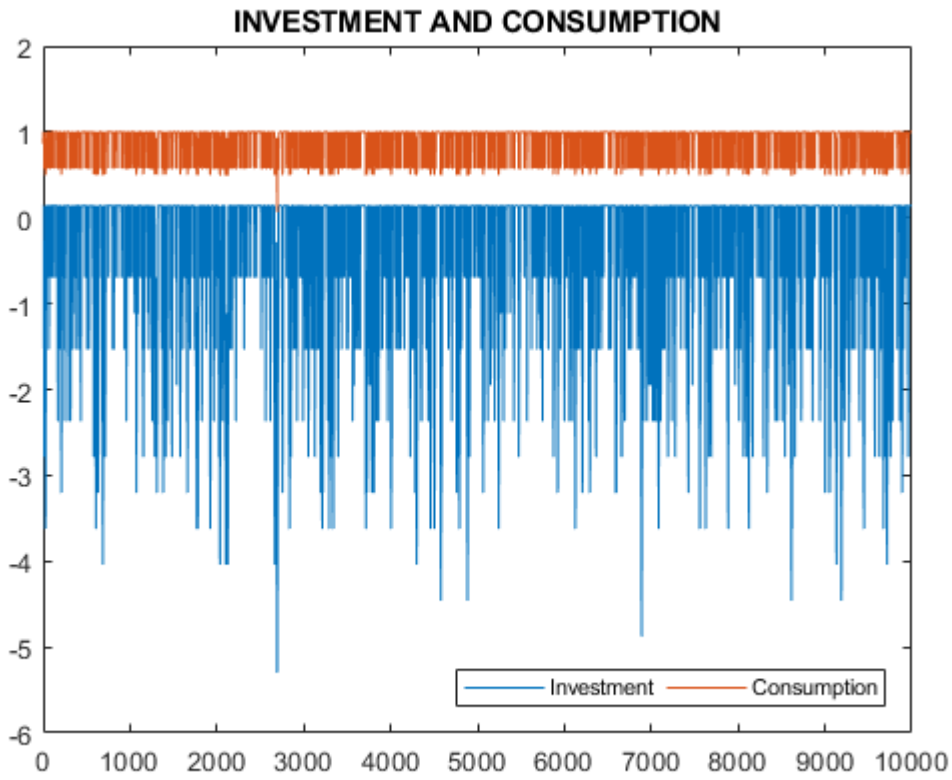
```
Simulation.LaborIncome = nan(Simulation.N,1);
Simulation.AssetIncome = nan(Simulation.N,1);
Simulation.Shock = nan(Simulation.N,1);

% Gera uma cadeia de choques no salario
[stateValue, stateIndex] = MarkovSimulation(Income.PI, Simulation.N, Income.Values, Simulation

for i = 1:Simulation.N
    % pega o estado do choque.
    Simulation.Asset(i) = asset_0;
    Simulation.AssetIndex(i) = Policy.AssetIndexFunction(asset_0);

    Simulation.Shock(i) = stateValue(i);

    % determina o investimento baseado no estado atual (asset\capital e
    % choque)
    Simulation.Investment(i) = Policy.AssetPrime.Values(Simulation.AssetIndex(i), stateIndex(i

    % Determina a renda proveniente do trabalho
    Simulation.LaborIncome(i) = Simulation.Wage*stateValue(i);

    % Determina a renda proveniente do rendimento dos assets
    Simulation.AssetIncome(i) = (1 + Econom_param.r)*Simulation.Asset(i);

    % Determina o consumo pela equacao de equilibrio
    Simulation.Consumption(i) = Simulation.AssetIncome(i) + Simulation.LaborIncome(i) - Simula

    % Atualiza o asset
    asset_0 = Simulation.Investment(i);
end
fprintf('DONE\n');
```

```
DONE
```

GRAFICOS

Grafico de investimento vs consumo

```
x = 1:Simulation.N;
plot(x, Simulation.Investment, x, Simulation.Consumption);
title('INVESTMENT AND CONSUMPTION');
legend({'Investment','Consumption'},'FontSize', 8, 'Location','southeast', 'Orientation','Hori
```



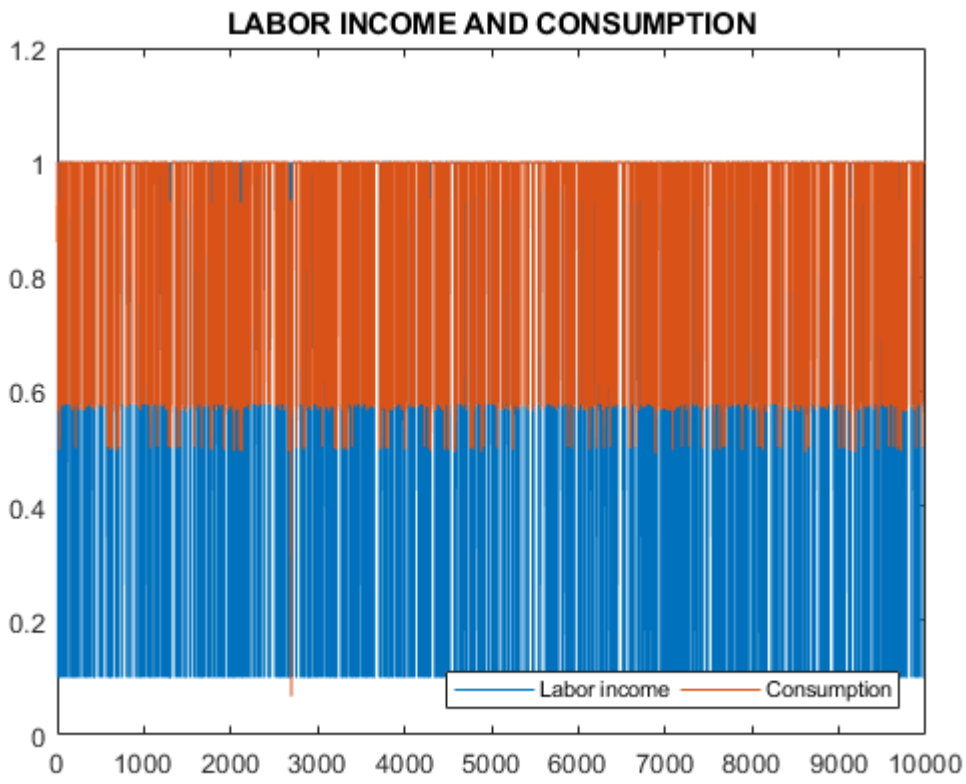
Covariancia de consumo e investimento

```
cov(Simulation.Consumption,Simulation.Investment)
```

```
ans =
    0.0344    0.0954
    0.0954    0.7458
```

Grafico de renda vs consumo

```
figure
plot(x, Simulation.LaborIncome, x, Simulation.Consumption);
title('LABOR INCOME AND CONSUMPTION');
legend({'Labor income','Consumption'},'FontSize', 8, 'Location','southeast', 'Orientation','Ho
```



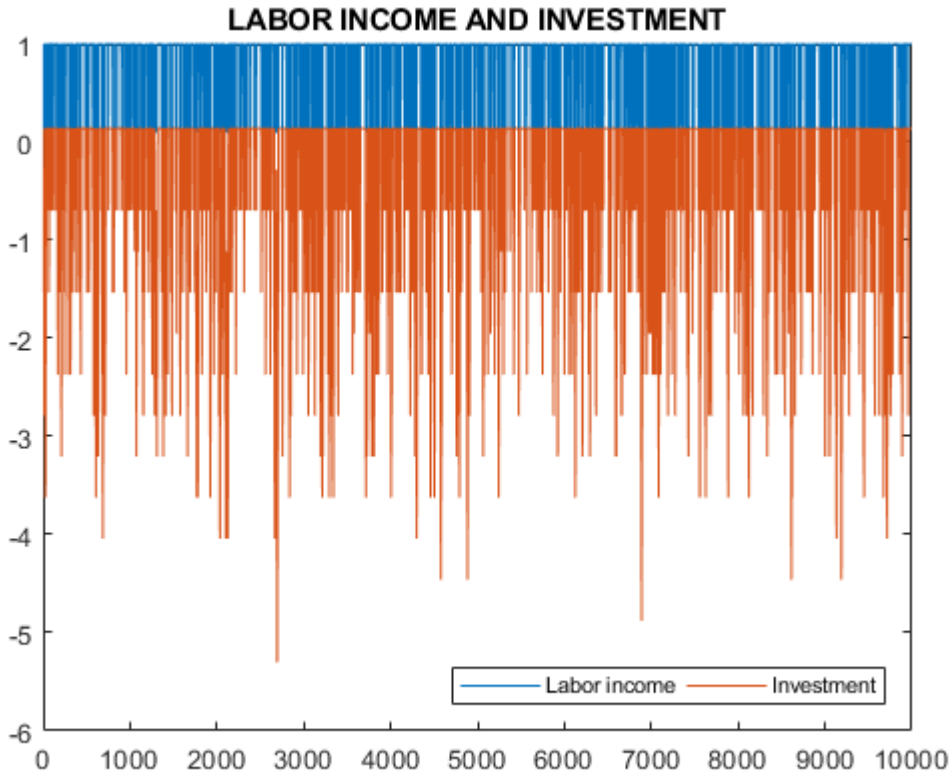
Covariancia de renda e consumo

```
cov(Simulation.LaborIncome, Simulation.Consumption)
```

```
ans =
    0.0891    0.0031
    0.0031    0.0344
```

Grafico de renda vs investimento

```
figure
plot(x, Simulation.LaborIncome, x, Simulation.Investment);
title('LABOR INCOME AND INVESTMENT');
legend({'Labor income','Investment'},'FontSize', 8, 'Location','southeast', 'Orientation','Ho
```



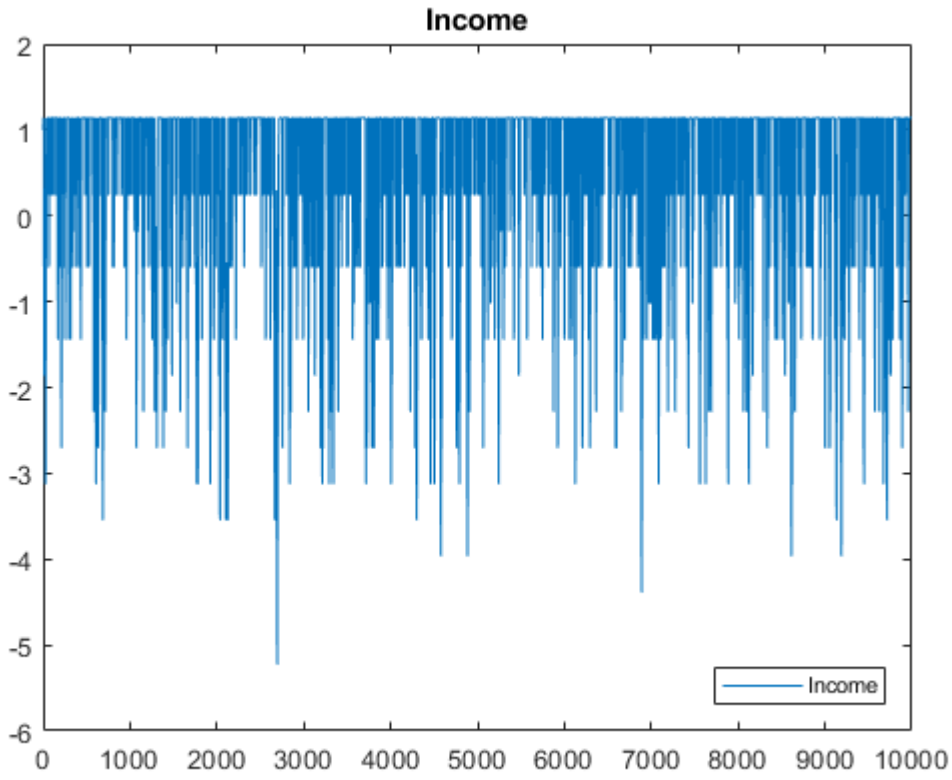
Covariancia de renda e investimento

```
cov(Simulation.LaborIncome, Simulation.Investment)
```

```
ans =
    0.0891    0.1495
    0.1495    0.7458
```

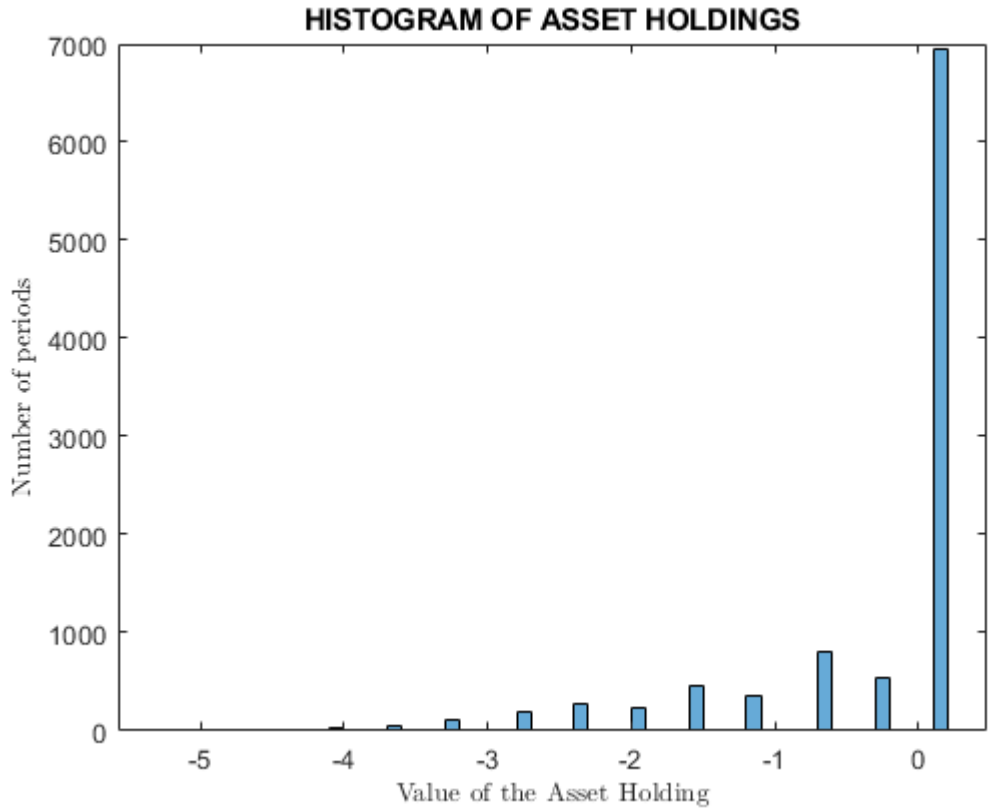
Grafico de riqueza total

```
figure
plot(x, Simulation.LaborIncome + Simulation.AssetIncome);
title('Income');
legend({'Income'},'FontSize', 8, 'Location','southeast', 'Orientation','Horizontal');
```



Histograma de ativos

```
figure
histogram(Simulation.Investment)
title('HISTOGRAM OF ASSET HOLDINGS')
xlabel('Value of the Asset Holding','FontSize',10,'Interpreter','latex');
ylabel('Number of periods','FontSize',10,'Interpreter','latex')
```




```
% limpa variaveis
clear x
```

CALCULO DOS VALORES EXPERADOS

average value of asset holdings

```
mean(Simulation.Asset)
```

```
ans = -0.3163
```

Average decline in consumption in response to entering unemployment

```
Delta.Consumption = Simulation.Consumption(2:end) - Simulation.Consumption(1:end-1);
Delta.Shock =  Simulation.Shock(2:end) - Simulation.Shock(1:end-1);
mean(Delta.Consumption(Delta.Shock < 0))
```

```
ans = 0.0318
```

```
clear Delta
```

Average consumption conditional on (i) employed; (ii) unemployed; (iii) unemployed for the last 12 months.

```
mean(Simulation.Consumption(Simulation.Shock==1))
```

```
ans = 0.8888
```

```
mean(Simulation.Consumption(Simulation.Shock==0.1))
```

```
ans = 0.8573
```

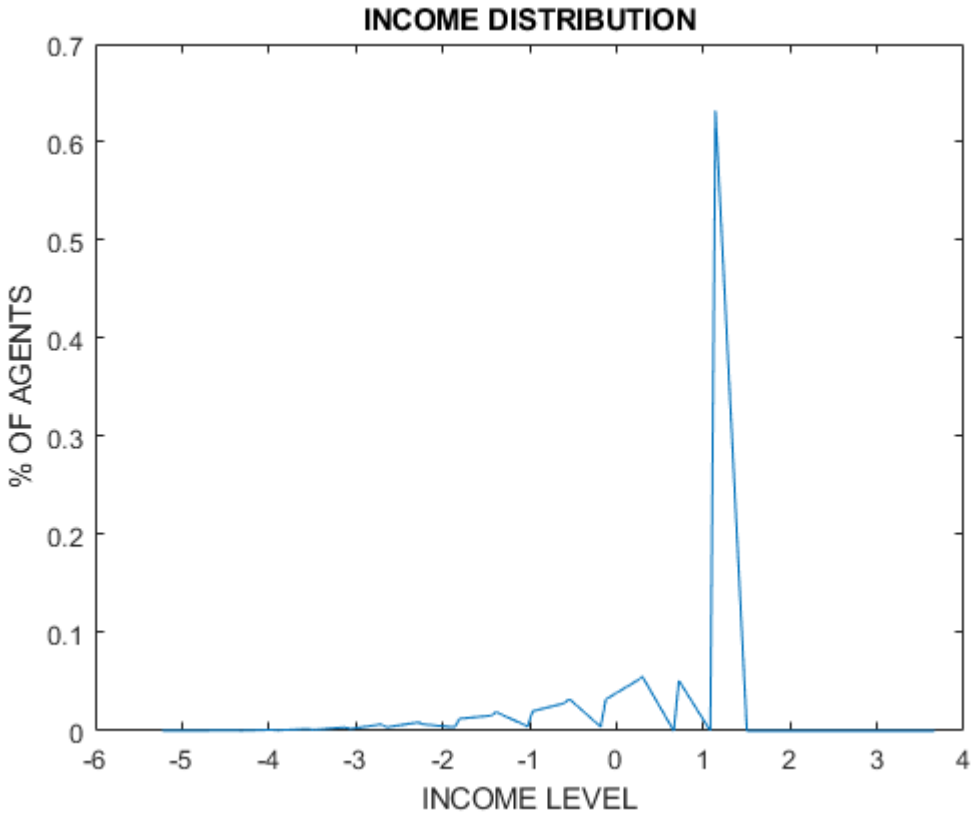
```
A = (Simulation.Shock==0.1)';
B = [1 1 1 1 1 1];
Unemployed6m = strfind(A,B)';

Consumption6Months = 0;
TotalPeriods = size(Unemployed6m,1);
for i=1:TotalPeriods
    Consumption6Months = Consumption6Months + mean(Simulation.Consumption(Unemployed6m(i):Uner
end
Consumption6Months/TotalPeriods
```

```
ans = 0.6358
```

```
clear A B Consumption6Months TotalPeriods Unemployed6m
```

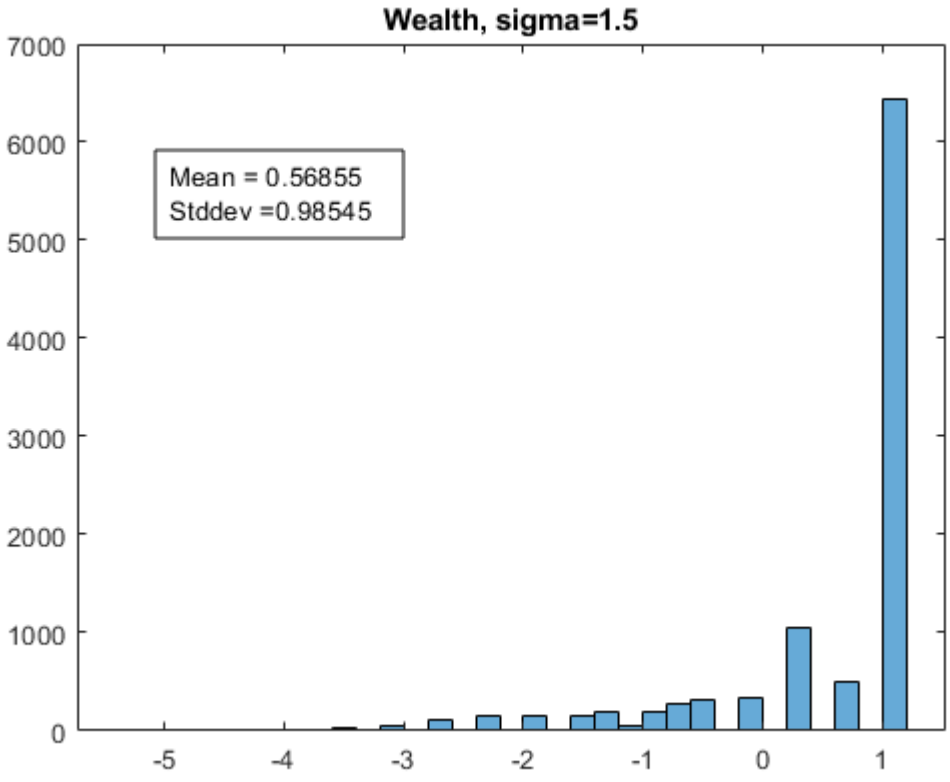
%% INCOME DISTRIBUTION

```
Total_Income =  ((1+Econom_param.r)*Asset.Values)'*ones(1,Income.Grid.N) + ones(1,Asset.Grid.N
[sortedIncome, index] = sort(Total_Income(:));
lambda_aux = Lambda(:);

figure
plot(sortedIncome, lambda_aux(index));
title('INCOME DISTRIBUTION');
xlabel('INCOME LEVEL');
ylabel('% OF AGENTS');
```



STANDARD MEASURES FOR WEALTH DISPERSION

```
mean_w = mean(Simulation.AssetIncome+Simulation.LaborIncome);
stdr_w = std(Simulation.AssetIncome+Simulation.LaborIncome);

figure
histogram(Simulation.AssetIncome+Simulation.LaborIncome);
annotation('textbox',[.2 .3 .4 .5],...
    'String',{['Mean = ' num2str(mean_w)],['Stddev =' num2str(stdr_w)]},...
'FitBoxtoText','on');
title(sprintf('Wealth, sigma=%1.1f',Econom_param.Sigma));
```


Wealth, sigma=1.5
Mean = 0.56855
Stddev =0.98545
7000
6000
5000
4000
3000
2000
1000
0
-5
-4
-3
-2
-1
0
1

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

QUESTÃO 1 (f)

LIMPEZA DE VARIAVEIS

```
clearvars
clc
```

ITEM A DEFINICAO DE PARAMETROS

Defininimos os parametros que sao utilizados pelo problema

```
Econom_param.PeriodsPerYear = 6;
Econom_param.Beta_anual = 0.96;
Econom_param.Beta = (Econom_param.Beta_anual)^(1/Econom_param.PeriodsPerYear);
Econom_param.Sigma = 3;

eps = 1e-5;
```

DEFINICAO DO GRID DE INCOME

Vamos Construir a matrix de transição baseada nas informações fornecidas

```
Income.Grid.N = 2;
Income.Grid.Max = 1;
Income.Grid.Min = 0.1;
Income.Values = linspace(Income.Grid.Max, Income.Grid.Min, Income.Grid.N);
Income.PI = [0.925 (1-0.925); 0.5 (1-0.5)];
Income.P_LongRun = CalculaLongoPrazo(Income.PI);
```

Logo a renda media de equilibrio no longo prazo sera.

```
Income.Average = Income.Values*Income.P_LongRun;
Income.Early = Income.Average * Econom_param.PeriodsPerYear;
```

A duração média do desemprego é dada por 1/f, onde f neste caso é 0.5. Sendo assim a duracao média do desemprego é dois periodos. Como o periodo é de dois meses, temos que a duração média de desemprego é de 4 meses.

DEFINICAO DO GRID de ASSET

```
Borrow.Limit = -Income.Early;
Asset.Grid.N = 20;
Asset.Grid.Max = 3*Income.Average;
Asset.Grid.Min = Borrow.Limit;
Asset.Values = linspace(Asset.Grid.Min, Asset.Grid.Max, Asset.Grid.N);
```

DEFINICAO DA TAXA DE JUROS

Inicia a taxa de juros

```
Econom_param.r_anual = 0.034;
Econom_param.r_start = ((1+Econom_param.r_anual)^(1/Econom_param.PeriodsPerYear))-1;
Econom_param.r_UpperLimit = 1/Econom_param.Beta - 1;
Econom_param.r_LowerLimit = 0;
Econom_param.r = Econom_param.r_start;

% parametro que indica se a taxa esta fixa ou nao.
Econom_param.r_IsFixed = 0;
```

ACHANDO O EQUILIBRIO.

1. Determinar um r inicial
2. Resolver o Problema do Consumidor e obter a Politica
3. Determinar Lambda
4. Verificar se existe eequilibrio no mercado de assets
5. Se nao ha equilibrio estabelecer novo r, (voltar a ponto 2)

   onde $e > 0 \rightarrow r_j + 1 < r_j \quad e < 0 \rightarrow r_j + 1 > r_j$

```
nIntLimit = 1000;

for i=1:nIntLimit

    % (2) Resolve o Problema do consumidor
    [V0, U_Cube, Policy] = SolveConsumerProblem(Asset, Income, Econom_param);

    % (3) Determina a distribuicao estacionaria
    Lambda = ConstructLambda(Policy, Asset, Income);

    % (4) Determina demanda de assets
    Demanda = Lambda(:)' * Policy.AssetPrime.Values(:);

    % (5) Verifica se há equilibrio
    if abs(Demanda) < eps
        break;
    elseif Demanda > eps
        Econom_param.r_UpperLimit = Econom_param.r;
```

```
    elseif  Demanda < -eps
        Econom_param.r_LowerLimit = Econom_param.r;
    end

    % Caso a taxa esteja fixa não ha mais nada o que fazer
    if Econom_param.r_IsFixed == 1
        break;
    end

    % determina novo r
    r_old = Econom_param.r;
    Econom_param.r = (Econom_param.r_UpperLimit + Econom_param.r_LowerLimit)/2;

    % Caso a precisao da taxa seja muito pequena paramos a execucao.
    if abs(Econom_param.r - r_old) < eps^2
        break;
    end

    fprintf('Inter:%4d\tr_0: %1.6f\tr_1: %1.6f\tDem: %2.15f\n', i, r_old, Econom_param.r, Dema
end
```

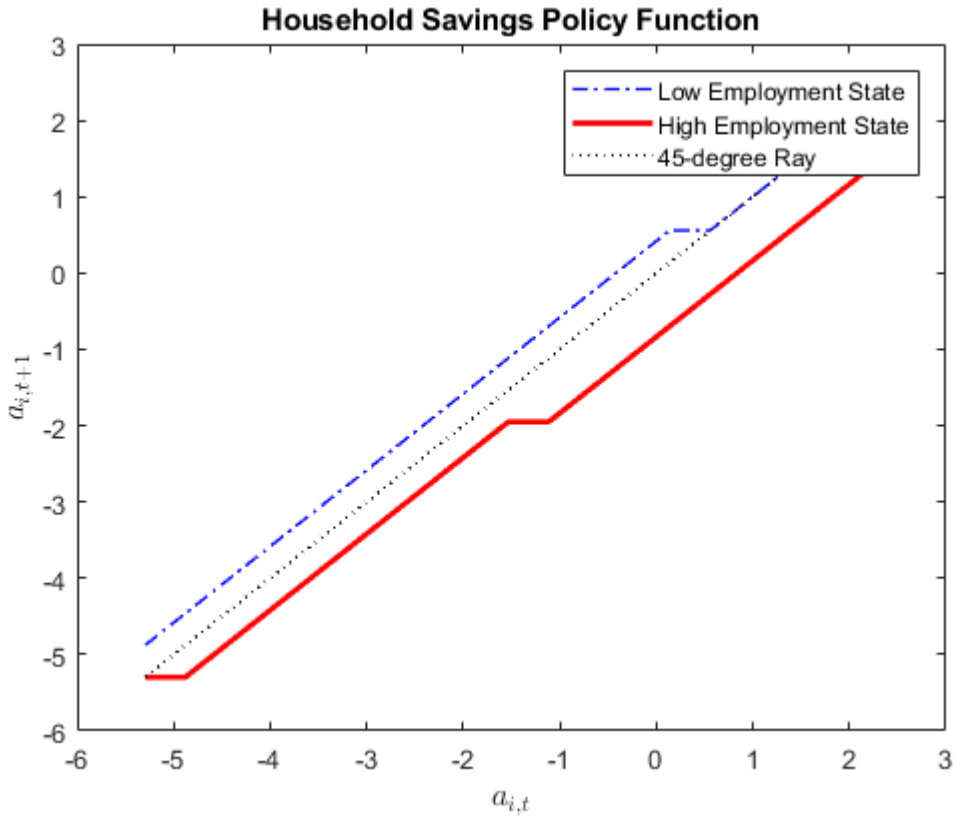
```
Inter:   1 r_0: 0.005588 r_1: 0.002794 Dem: 1.691933671066469
Inter:   2 r_0: 0.002794 r_1: 0.001397 Dem: 0.094030769977306
Inter:   3 r_0: 0.001397 r_1: 0.000699 Dem: 0.094030769977306
Inter:   4 r_0: 0.000699 r_1: 0.001048 Dem: -0.664402180409014
Inter:   5 r_0: 0.001048 r_1: 0.001222 Dem: -0.262711239376189
Inter:   6 r_0: 0.001222 r_1: 0.001310 Dem: -0.262711239376189
Inter:   7 r_0: 0.001310 r_1: 0.001353 Dem: -0.262711239376189
Inter:   8 r_0: 0.001353 r_1: 0.001375 Dem: -0.262711239376189
Inter:   9 r_0: 0.001375 r_1: 0.001364 Dem: 0.094030769977306
Inter:  10 r_0: 0.001364 r_1: 0.001359 Dem: 0.094030769977306
Inter:  11 r_0: 0.001359 r_1: 0.001356 Dem: 0.094030769977306
Inter:  12 r_0: 0.001356 r_1: 0.001355 Dem: 0.094030769977306
Inter:  13 r_0: 0.001355 r_1: 0.001355 Dem: -0.262711239376189
Inter:  14 r_0: 0.001355 r_1: 0.001356 Dem: -0.262711239376189
Inter:  15 r_0: 0.001356 r_1: 0.001356 Dem: 0.094030769977306
Inter:  16 r_0: 0.001356 r_1: 0.001356 Dem: -0.262711239376189
Inter:  17 r_0: 0.001356 r_1: 0.001356 Dem: 0.094030769977306
Inter:  18 r_0: 0.001356 r_1: 0.001356 Dem: 0.094030769977306
Inter:  19 r_0: 0.001356 r_1: 0.001356 Dem: 0.094030769977306
Inter:  20 r_0: 0.001356 r_1: 0.001356 Dem: 0.094030769977306
Inter:  21 r_0: 0.001356 r_1: 0.001356 Dem: -0.262711239376189
Inter:  22 r_0: 0.001356 r_1: 0.001356 Dem: 0.094030769977306
Inter:  23 r_0: 0.001356 r_1: 0.001356 Dem: 0.094030769977306
Inter:  24 r_0: 0.001356 r_1: 0.001356 Dem: -0.262711239376189
Inter:  25 r_0: 0.001356 r_1: 0.001356 Dem: -0.262711239376189
```

```
% Limpa variaveis nao utilizadas
clear nIntLimit r_old
```

GRAFICO DA FUNCAO POLITICA DE A_{t+1} CONTRA A_t

```
fig1 = figure();
plot(Policy.AssetDomain, Policy.AssetPrime.Values(:,1),'b-.','LineWidth',1); hold on;
```

```
plot(Policy.AssetDomain, Policy.AssetPrime.Values(:,2),'r','LineWidth',2); hold on;
plot(Policy.AssetDomain, Policy.AssetDomain,'k:','LineWidth',1); hold off;
legend({'Low Employment State','High Employment State','45-degree Ray'});
title('Household Savings Policy Function');
xlabel('$a_{i,t}$','FontSize',12,'Interpreter','latex');
ylabel('$a_{i,t+1}$','FontSize',12,'Interpreter','latex');
```



SIMULACAO DE LIFE HISTORY

define uma funcao que indica, dado um valor de asset, qual o valor do grid que mais se aproxima do valor informado. Em outras palavras informa aonde um valor fornecido estaria no grid.

```
Policy.AssetIndexFunction = @(value) round((value-Policy.AssetDomain(1))/(Policy.AssetDomain(2

% Total de simulacoes
Simulation.N = 10000;

% Nivel inicial de ativos
asset_0 = 0;

% Estado inicial: Empregado.
Simulation.S0 = 1;
Simulation.Wage = 1;

% Inicializa os vetores da simulacao.
Simulation.Asset = nan(Simulation.N,1);
Simulation.AssetIndex = nan(Simulation.N,1);
```

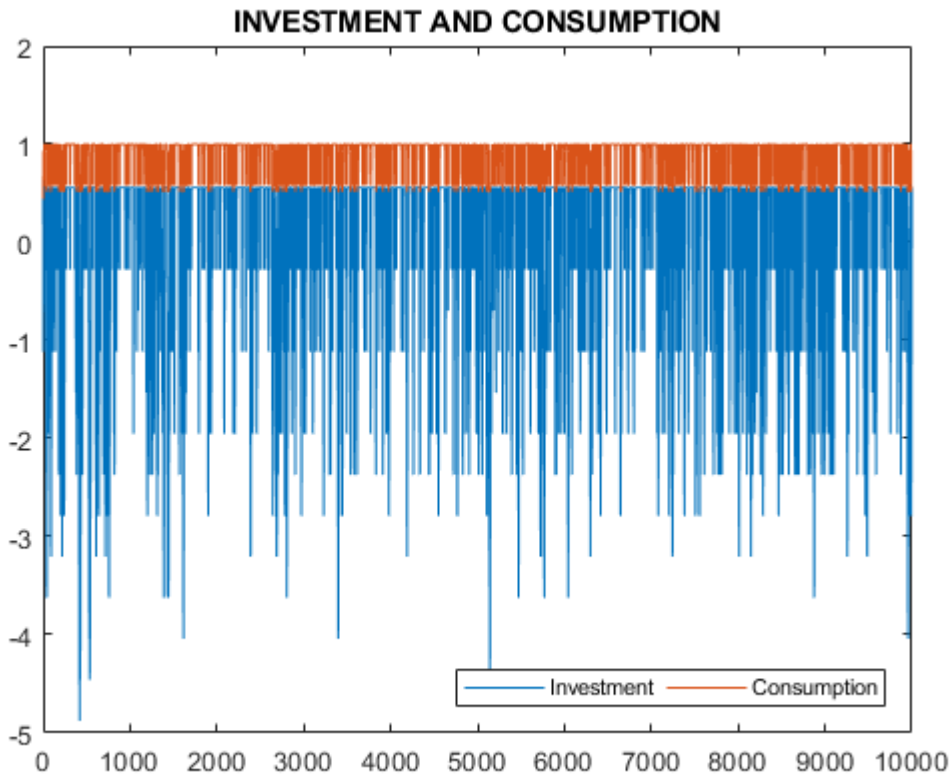
```
Simulation.Consumption = nan(Simulation.N,1);
Simulation.Investment = nan(Simulation.N,1);
Simulation.LaborIncome = nan(Simulation.N,1);
Simulation.AssetIncome = nan(Simulation.N,1);
Simulation.Shock = nan(Simulation.N,1);

% Gera uma cadeia de choques no salario
[stateValue, stateIndex] = MarkovSimulation(Income.PI, Simulation.N, Income.Values, Simulation

for i = 1:Simulation.N
    % pega o estado do choque.
    Simulation.Asset(i) = asset_0;
    Simulation.AssetIndex(i) = Policy.AssetIndexFunction(asset_0);

    Simulation.Shock(i) = stateValue(i);

    % determina o investimento baseado no estado atual (asset\capital e
    % choque)
    Simulation.Investment(i) = Policy.AssetPrime.Values(Simulation.AssetIndex(i), stateIndex(i

    % Determina a renda proveniente do trabalho
    Simulation.LaborIncome(i) = Simulation.Wage*stateValue(i);

    % Determina a renda proveniente do rendimento dos assets
    Simulation.AssetIncome(i) = (1 + Econom_param.r)*Simulation.Asset(i);

    % Determina o consumo pela equacao de equilibrio
    Simulation.Consumption(i) = Simulation.AssetIncome(i) + Simulation.LaborIncome(i) - Simula

    % Atualiza o asset
    asset_0 = Simulation.Investment(i);
end
fprintf('DONE\n');
```

```
DONE
```

GRAFICOS

Grafico de investimento vs consumo

```
x = 1:Simulation.N;
plot(x, Simulation.Investment, x, Simulation.Consumption);
title('INVESTMENT AND CONSUMPTION');
legend({'Investment','Consumption'},'FontSize', 8, 'Location','southeast', 'Orientation','Hori
```



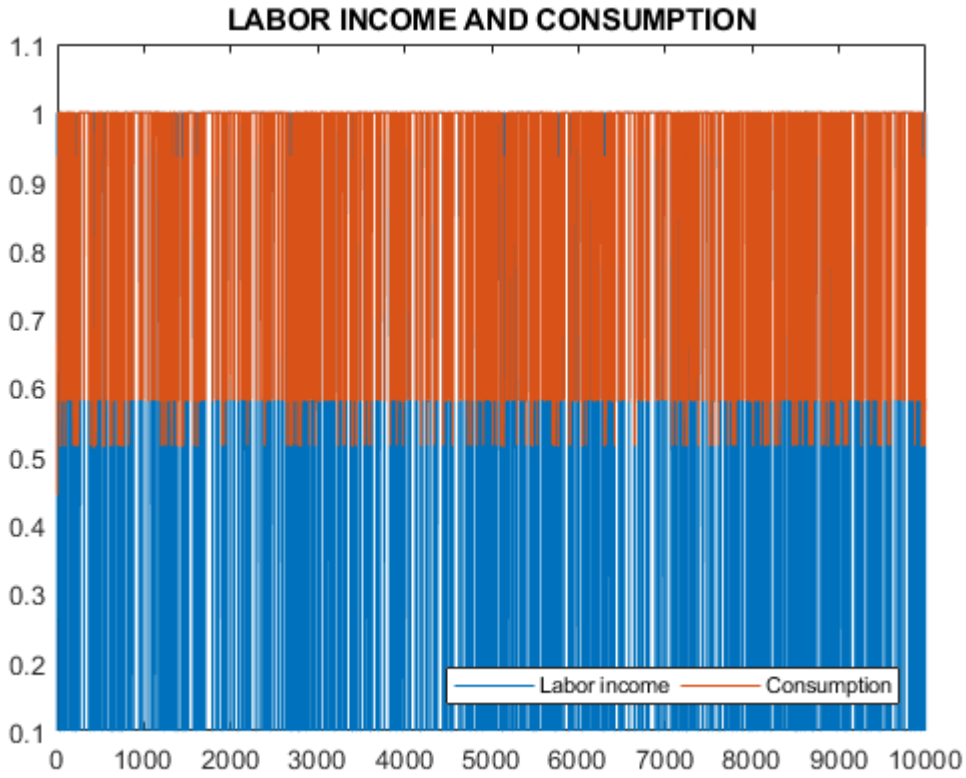
Covariancia de consumo e investimento

```
cov(Simulation.Consumption,Simulation.Investment)
```

```
ans =
    0.0351    0.1023
    0.1023    0.8041
```

Grafico de renda vs consumo

```
figure
plot(x, Simulation.LaborIncome, x, Simulation.Consumption);
title('LABOR INCOME AND CONSUMPTION');
legend({'Labor income','Consumption'},'FontSize', 8, 'Location','southeast', 'Orientation','Ho
```



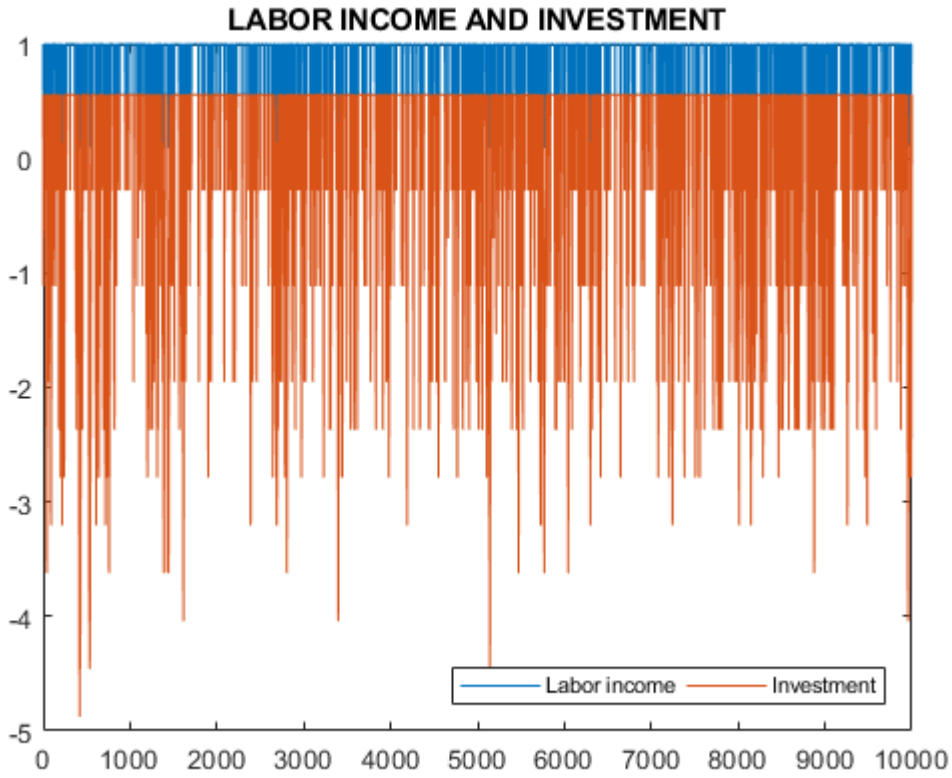
Covariancia de renda e consumo

```
cov(Simulation.LaborIncome, Simulation.Consumption)
```

```
ans =
    0.0951    0.0044
    0.0044    0.0351
```

Grafico de renda vs investimento

```
figure
plot(x, Simulation.LaborIncome, x, Simulation.Investment);
title('LABOR INCOME AND INVESTMENT');
legend({'Labor income','Investment'},'FontSize', 8, 'Location','southeast', 'Orientation','Ho
```



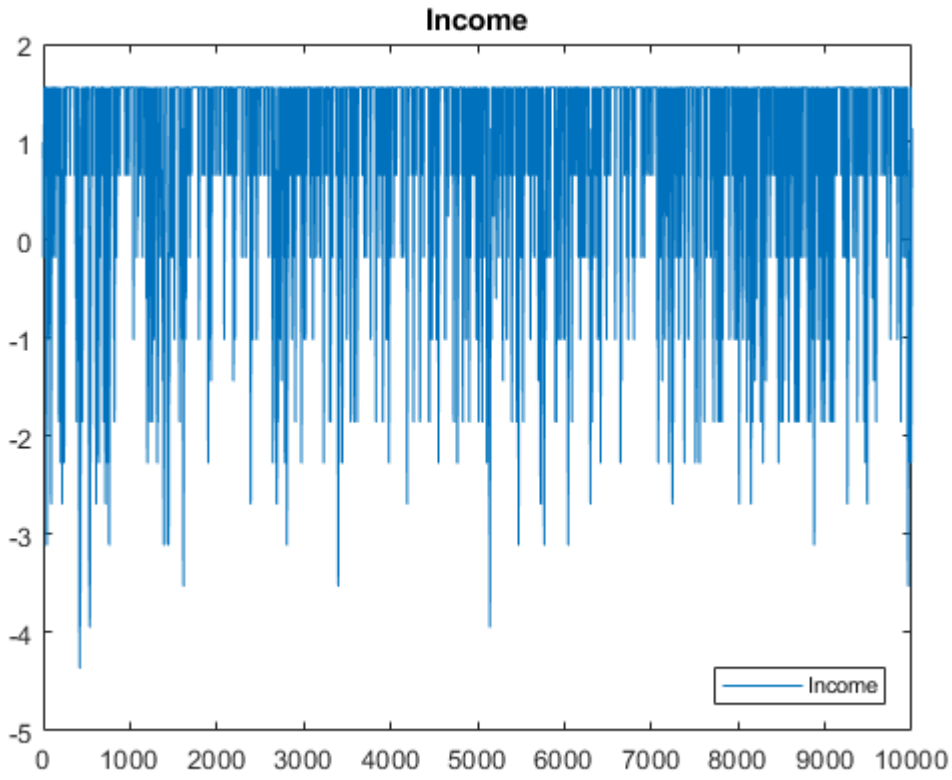
Covariancia de renda e investimento

```
cov(Simulation.LaborIncome, Simulation.Investment)
```

```
ans =
    0.0951    0.1619
    0.1619    0.8041
```

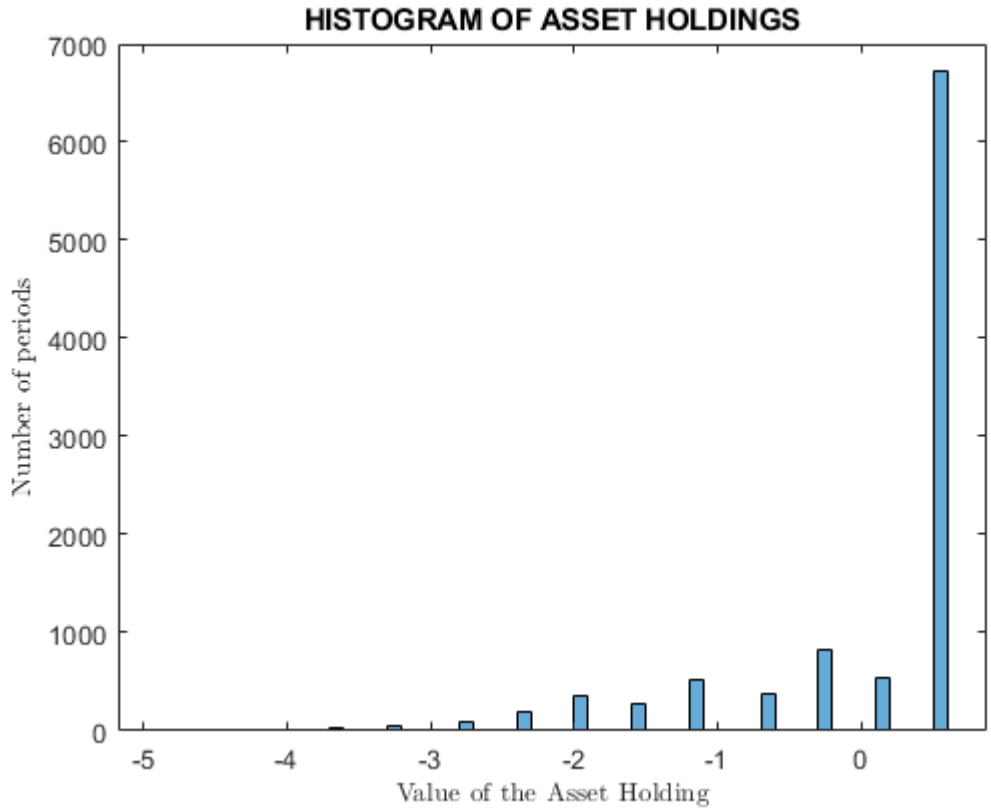
Grafico de riqueza total

```
figure
plot(x, Simulation.LaborIncome + Simulation.AssetIncome);
title('Income');
legend({'Income'},'FontSize', 8, 'Location','southeast', 'Orientation','Horizontal');
```



Histograma de ativos

```
figure
histogram(Simulation.Investment)
title('HISTOGRAM OF ASSET HOLDINGS')
xlabel('Value of the Asset Holding','FontSize',10,'Interpreter','latex');
ylabel('Number of periods','FontSize',10,'Interpreter','latex')
```




```
% limpa variaveis
clear x
```

CALCULO DOS VALORES EXPERADOS

average value of asset holdings

```
mean(Simulation.Asset)
```

```
ans = 0.0567
```

Average decline in consumption in response to entering unemployment

```
Delta.Consumption = Simulation.Consumption(2:end) - Simulation.Consumption(1:end-1);
Delta.Shock =  Simulation.Shock(2:end) - Simulation.Shock(1:end-1);
mean(Delta.Consumption(Delta.Shock < 0))
```

```
ans = 0.0313
```

```
clear Delta
```

Average consumption conditional on (i) employed; (ii) unemployed; (iii) unemployed for the last 12 months.

```
mean(Simulation.Consumption(Simulation.Shock==1))
```

```
ans = 0.8835
```

```
mean(Simulation.Consumption(Simulation.Shock==0.1))
```

```
ans = 0.8415
```

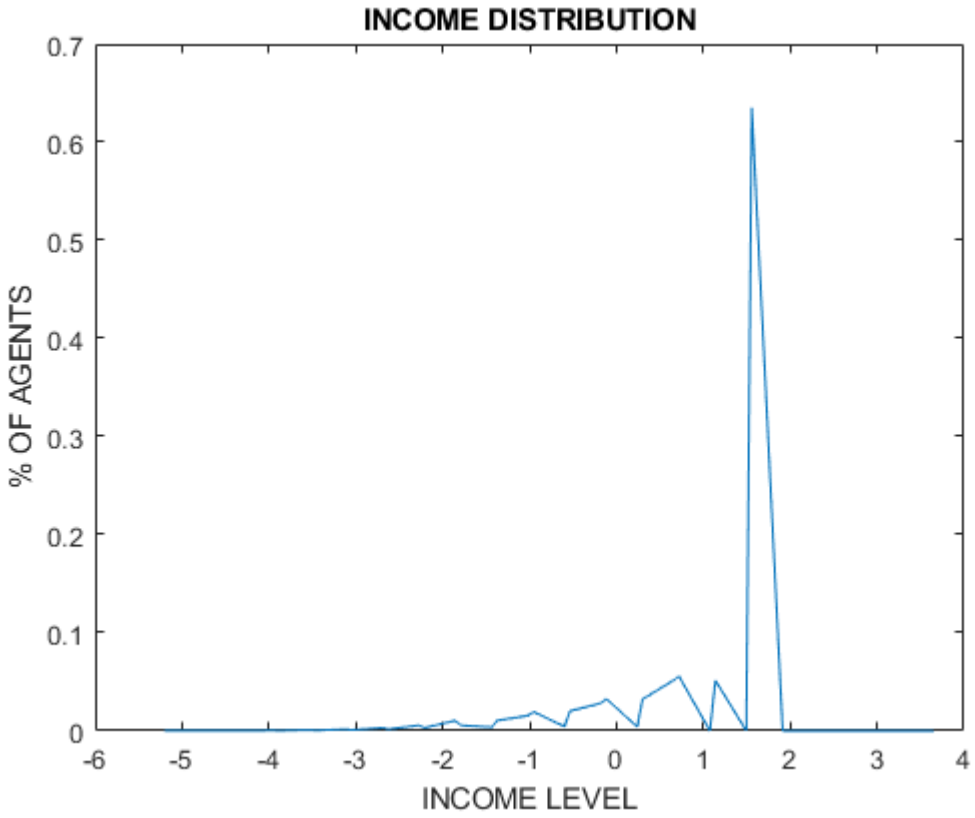
```
A = (Simulation.Shock==0.1)';
B = [1 1 1 1 1 1];
Unemployed6m = strfind(A,B)';

Consumption6Months = 0;
TotalPeriods = size(Unemployed6m,1);
for i=1:TotalPeriods
    Consumption6Months = Consumption6Months + mean(Simulation.Consumption(Unemployed6m(i):Uner
end
Consumption6Months/TotalPeriods
```

```
ans = 0.6497
```

```
clear A B Consumption6Months TotalPeriods Unemployed6m
```

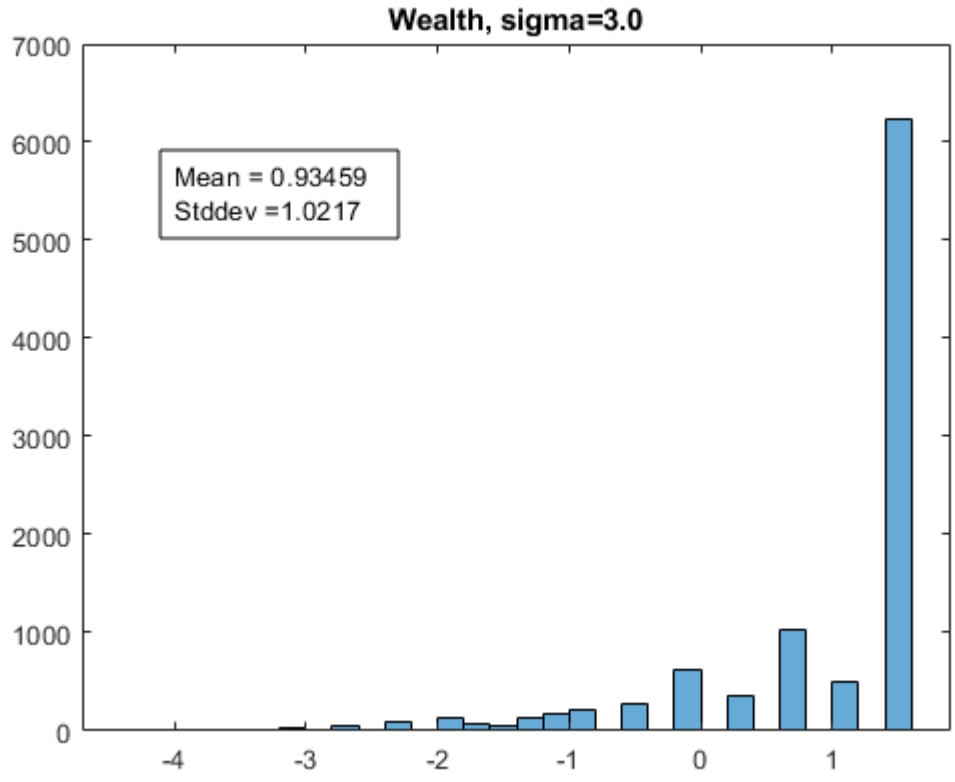
%% INCOME DISTRIBUTION

```
Total_Income =  ((1+Econom_param.r)*Asset.Values)'*ones(1,Income.Grid.N) + ones(1,Asset.Grid.N
[sortedIncome, index] = sort(Total_Income(:));
lambda_aux = Lambda(:);

figure
plot(sortedIncome, lambda_aux(index));
title('INCOME DISTRIBUTION');
xlabel('INCOME LEVEL');
ylabel('% OF AGENTS');
```



STANDARD MEASURES FOR WEALTH DISPERSION

```
mean_w = mean(Simulation.AssetIncome+Simulation.LaborIncome);
stdr_w = std(Simulation.AssetIncome+Simulation.LaborIncome);

figure
histogram(Simulation.AssetIncome+Simulation.LaborIncome);
annotation('textbox',[.2 .3 .4 .5],...
    'String',{['Mean = ' num2str(mean_w)],['Stddev =' num2str(stdr_w)]},...
'FitBoxtoText','on');
title(sprintf('Wealth, sigma=%1.1f',Econom_param.Sigma));
```


Wealth, sigma=3.0
Mean = 0.93459
Stddev =1.0217
7000
6000
5000
4000
3000
2000
1000
0
-4
-3
-2
-1
0
1

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*

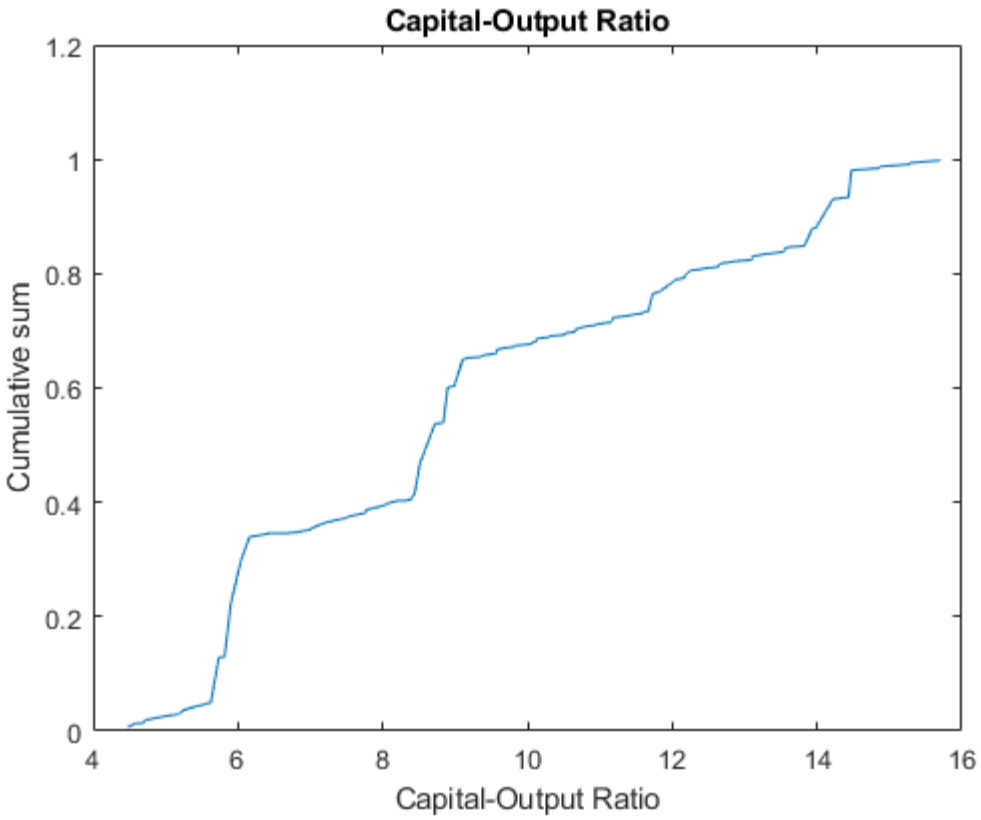
...

	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

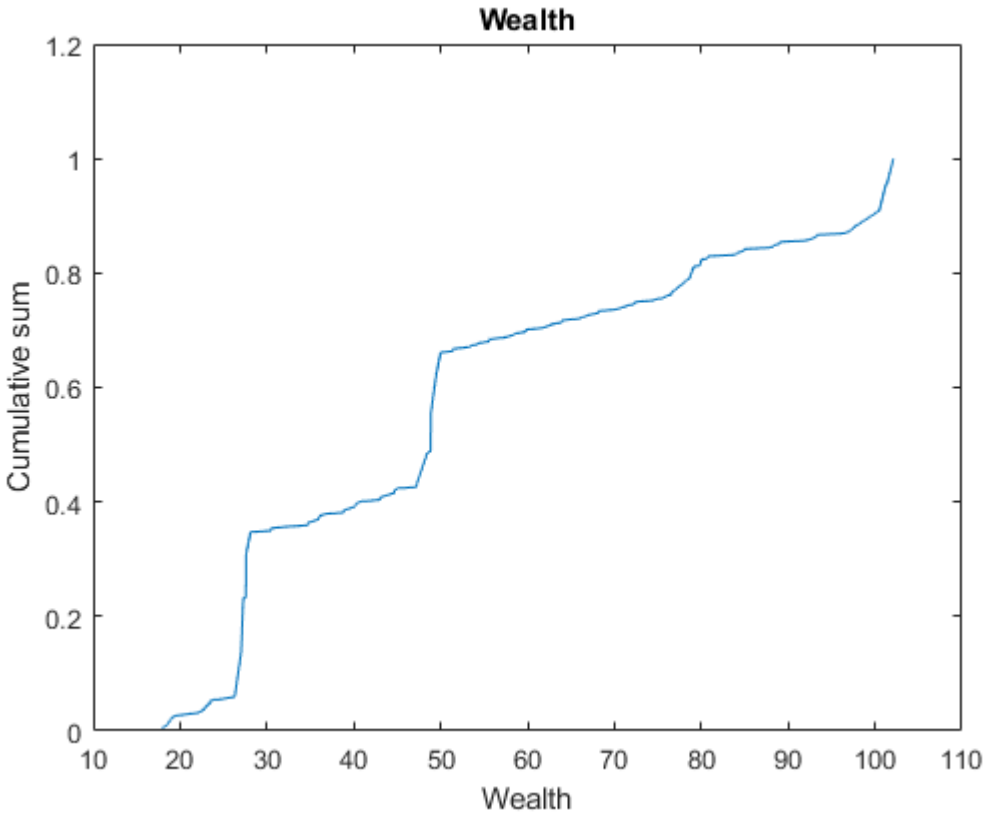
Determine Statistics

```
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

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 F

A arrecadação do governo pode ser escrita como:

$$G = \tau_y K_t^{\alpha} N_t^{1-\alpha} + \tau_n w_t N_t$$

Em termos per capta temos:

$$g = \tau_y k_t^{\alpha} + \tau_n w_t$$

Se a arrecadação é constante, pelo teorema da função implicita temos:

$$\frac{d\tau_y}{d\tau_n} = \frac{-w}{k^{\alpha}}$$

substituindo a equacoes de w temos:

$$\frac{d\tau_y}{d\tau_n} = -\frac{(1-\tau_y)(1-\alpha)}{(1+\tau_n)}$$

$$\frac{d\tau_y}{d\tau_n} = -0.48$$

Utilizando o conceito de diferencial temos:

$$\Delta\tau_y = -0.48\Delta\tau_n = (-0.48) * (-0.05) = 0.024$$

Logo é esperado que τ_y seja próximo de 0.024.

```
% Lipeza 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.
```

```
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;

eco_param.tau_n = 0.25;
eco_param.tau_y = 0;

% 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);

G0 = CalculaRendaGov(eco_param.tau_y ,Labor, WageShocks, eco_param, 1);
```

```
tau_y:0.000 tau_n:0.250
Inter:   1 r: -0.019167 Dem: -8.253269
Inter:   2 r: -0.049583 Dem: 61.290115
Inter:   3 r: -0.034375 Dem: 20.366287
Inter:   4 r: -0.026771 Dem: 6.357693
Inter:   5 r: -0.022969 Dem: 0.695067
```

```
Inter:   6 r: -0.021068 Dem: -3.167669
Inter:   7 r: -0.022018 Dem: -0.961379
Inter:   8 r: -0.022493 Dem: -0.000415
Inter:   9 r: -0.022731 Dem: 0.372352
Inter:  10 r: -0.022612 Dem: 0.159592
Inter:  11 r: -0.022553 Dem: 0.079587
Inter:  12 r: -0.022523 Dem: 0.039594
Inter:  13 r: -0.022508 Dem: 0.019590
Inter:  14 r: -0.022501 Dem: 0.009588
Inter:  15 r: -0.022497 Dem: 0.004586
Inter:  16 r: -0.022495 Dem: 0.002086
Inter:  17 r: -0.022494 Dem: 0.000835
Inter:  18 r: -0.022494 Dem: 0.000210
Inter:  19 r: -0.022494 Dem: -0.000102
Inter:  20 r: -0.022494 Dem: 0.000054
Inter:  21 r: -0.022494 Dem: -0.000024
Inter:  22 r: -0.022494 Dem: 0.000015
interest rate: -0.022494
wage rate: 1.749060
```

ans = *20×7 table*

...

	Var1	Var2	Var3	Var4	Var5	Var6
1	100.0000	16.0791	16.0791	16.0791	16.0791	16.0791
2	16.0791	20.4960	20.4960	20.4960	20.4960	20.4960
3	20.4960	24.9129	24.9129	24.9129	24.9129	24.9129
4	24.9129	24.9129	24.9129	24.9129	24.9129	29.3298
5	29.3298	29.3298	33.7467	33.7467	33.7467	33.7467
6	33.7467	33.7467	38.1636	38.1636	38.1636	38.1636
7	38.1636	38.1636	38.1636	38.1636	38.1636	38.1636
8	42.5805	42.5805	42.5805	46.9974	42.5805	42.5805
9	46.9974	46.9974	46.9974	51.4142	51.4142	51.4142
10	51.4142	51.4142	51.4142	55.8311	55.8311	55.8311
11	55.8311	55.8311	55.8311	60.2480	60.2480	60.2480
12	60.2480	60.2480	60.2480	64.6649	64.6649	64.6649
13	64.6649	64.6649	64.6649	64.6649	69.0818	64.6649
14	69.0818	69.0818	69.0818	69.0818	73.4987	73.4987
15	73.4987	73.4987	73.4987	73.4987	77.9156	77.9156
16	77.9156	77.9156	77.9156	77.9156	82.3325	82.3325

ans = *20×7 table*

...

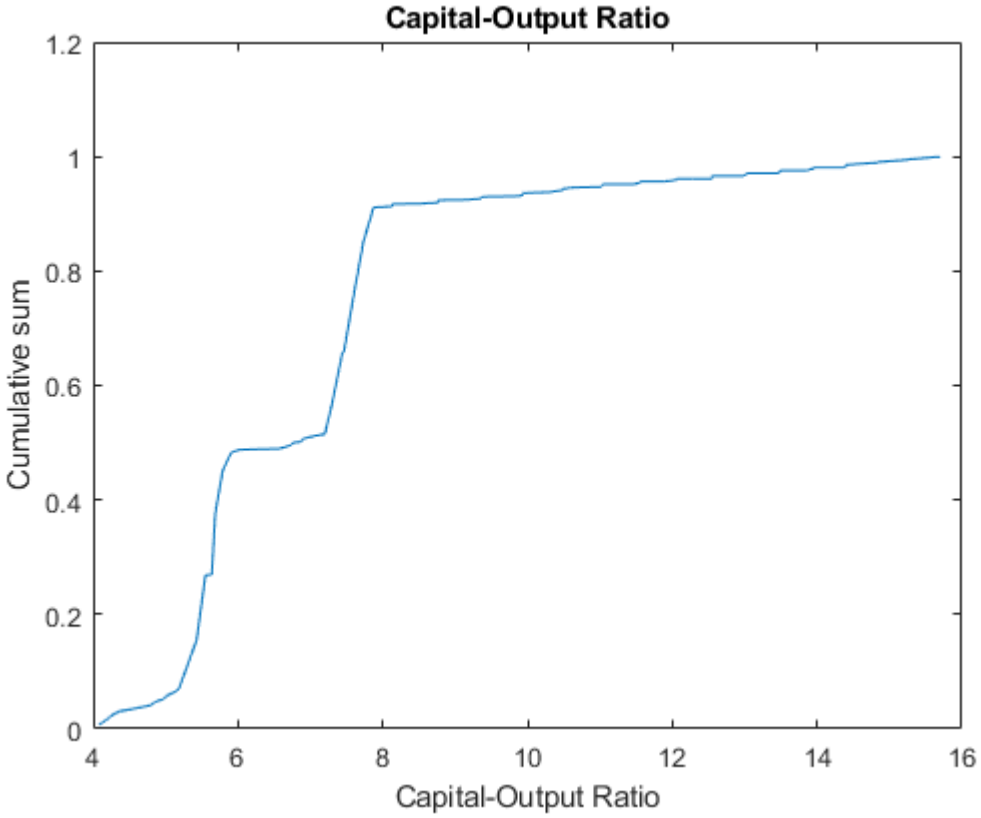
	Var1	Var2	Var3	Var4	Var5	Var6
1	0.0100	0.8437	0.6874	0.5832	0.4789	0.3747
2	0.0100	0.8958	0.7395	0.5832	0.4789	0.4268
3	0.0100	0.9479	0.7916	0.6353	0.5311	0.4268
4	0.0100	0.0100	0.0100	0.0100	0.0100	0.4268

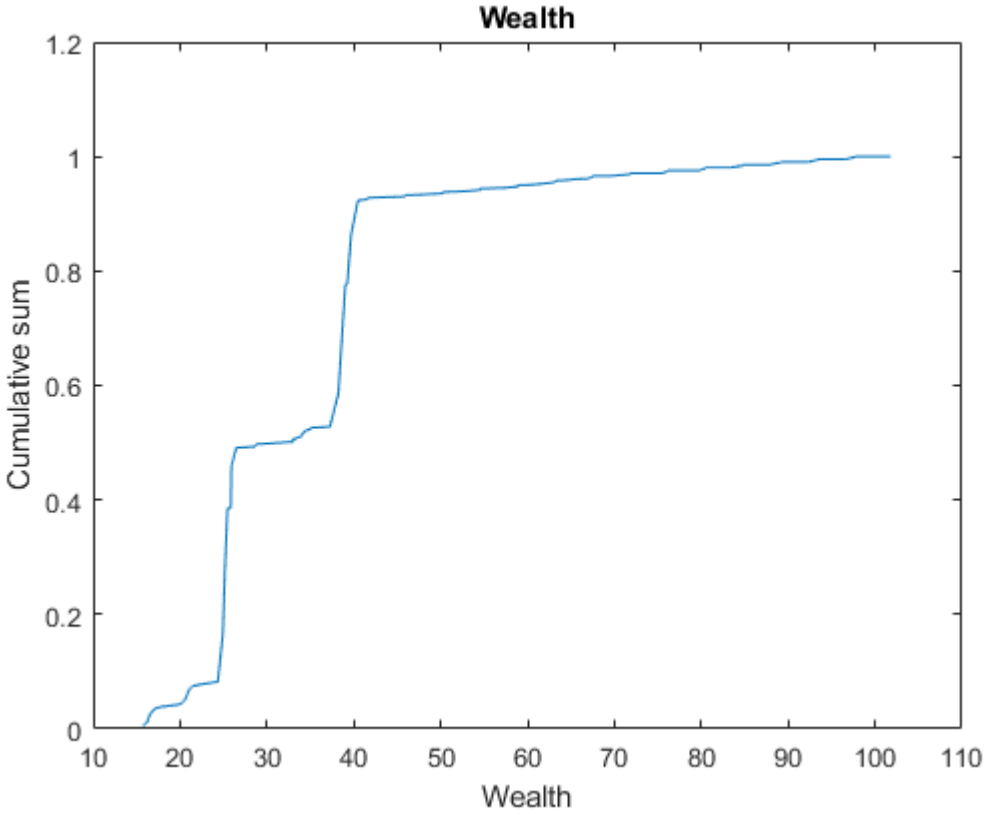
	Var1	Var2	Var3	Var4	Var5	Var6
5	0.0100	0.0100	0.8958	0.6874	0.5311	0.4268
6	0.0100	0.0100	0.8958	0.7395	0.5832	0.4789
7	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100
8	0.0100	0.0100	0.0100	0.7916	0.0100	0.0100
9	0.0100	0.0100	0.0100	0.8437	0.6353	0.4789
10	0.0100	0.0100	0.0100	0.8437	0.6353	0.5311
11	0.0100	0.0100	0.0100	0.8958	0.6874	0.5311
12	0.0100	0.0100	0.0100	0.8958	0.6874	0.5311
13	0.0100	0.0100	0.0100	0.0100	0.7395	0.0100
14	0.0100	0.0100	0.0100	0.0100	0.7395	0.5832
15	0.0100	0.0100	0.0100	0.0100	0.7395	0.5832
16	0.0100	0.0100	0.0100	0.0100	0.7916	0.5832

ans = *20×7 table*

...

	Lambda1	Lambda2	Lambda3	Lambda4	Lambda5	Lambda6
1	0.0048	0.0074	0.0090	0.0085	0.0058	0.0024
2	0.0049	0.0075	0.0093	0.0088	0.0061	0.0026
3	0.0050	0.0836	0.1124	0.1073	0.0742	0.0313
4	0.0021	0.0037	0.0002	0.0000	0.0001	0.0009
5	0.0021	0.0040	0.0056	0.0060	0.0049	0.0029
6	0.0021	0.0040	0.0532	0.0894	0.1009	0.0866
7	0.0022	0.0018	0.0008	0.0000	0.0000	0.0000
8	0.0023	0.0018	0.0008	0.0004	0.0000	0.0000
9	0.0024	0.0017	0.0007	0.0006	0.0004	0.0002
10	0.0025	0.0017	0.0006	0.0005	0.0004	0.0002
11	0.0026	0.0017	0.0006	0.0008	0.0009	0.0008
12	0.0028	0.0016	0.0005	0.0013	0.0011	0.0008
13	0.0029	0.0015	0.0003	0.0000	0.0000	0.0000
14	0.0031	0.0014	0.0003	0.0000	0.0000	0.0000
15	0.0032	0.0013	0.0002	0.0000	0.0000	0.0000
16	0.0034	0.0012	0.0002	0.0000	0.0000	0.0000






```
eco_param.tau_n = 0.20
```

```
eco_param = struct with fields:
            alpha: 0.4000
             beta: 0.9600
            delta: 0.0800
            gamma: 0.7500
          sigma_c: 2
          sigma_l: 2
            tau_n: 0.2000
            tau_y: 0
     r_UpperBound: 0.0417
      r_LowerBound: -0.0800
                r: -0.0192
         r_tilda: 0.0608
```

```
f = @(tau_y)( G0 - CalculaRendaGov(tau_y, Labor, WageShocks, eco_param, 0));

sol = fsolve(f,0.024);
```

```
tau_y:0.024 tau_n:0.200
tau_y:0.024 tau_n:0.200
tau_y:0.023 tau_n:0.200
tau_y:0.023 tau_n:0.200
tau_y:0.021 tau_n:0.200
tau_y:0.021 tau_n:0.200
tau_y:0.021 tau_n:0.200
tau_y:0.021 tau_n:0.200
tau_y:0.021 tau_n:0.200
tau_y:0.021 tau_n:0.200
tau_y:0.021 tau_n:0.200
tau_y:0.021 tau_n:0.200
tau_y:0.021 tau_n:0.200
tau_y:0.021 tau_n:0.200
tau_y:0.021 tau_n:0.200
tau_y:0.021 tau_n:0.200
tau_y:0.021 tau_n:0.200
tau_y:0.021 tau_n:0.200
tau_y:0.021 tau_n:0.200

Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the default value of the function tolerance, and
the problem appears regular as measured by the gradient.

<stopping criteria details>
```

```
CalculaRendaGov(sol, Labor, WageShocks, eco_param, 1);
```

```
tau_y:0.021 tau_n:0.200
Inter:   1 r: -0.019167 Dem: -9.573394
Inter:   2 r: -0.049583 Dem: 58.723891
Inter:   3 r: -0.034375 Dem: 17.357920
Inter:   4 r: -0.026771 Dem: 5.390330
Inter:   5 r: -0.022969 Dem: -0.160819
Inter:   6 r: -0.024870 Dem: 2.792012
Inter:   7 r: -0.023919 Dem: 1.075791
Inter:   8 r: -0.023444 Dem: 0.473222
Inter:   9 r: -0.023206 Dem: 0.150688
Inter:  10 r: -0.023088 Dem: -0.004259
Inter:  11 r: -0.023147 Dem: 0.078635
Inter:  12 r: -0.023117 Dem: 0.034880
```

```
Inter:  13 r: -0.023102 Dem: 0.015311
Inter:  14 r: -0.023095 Dem: 0.005526
Inter:  15 r: -0.023091 Dem: 0.000633
Inter:  16 r: -0.023089 Dem: -0.001813
Inter:  17 r: -0.023090 Dem: -0.000590
Inter:  18 r: -0.023091 Dem: 0.000022
Inter:  19 r: -0.023091 Dem: -0.000284
Inter:  20 r: -0.023091 Dem: -0.000131
Inter:  21 r: -0.023091 Dem: -0.000055
Inter:  22 r: -0.023091 Dem: -0.000016
interest rate: -0.023091
wage rate: 1.769679
```

ans = *20×7 table*

...

	Var1	Var2	Var3	Var4	Var5	Var6
1	100.0000	15.8481	15.8481	15.8481	15.8481	15.8481
2	15.8481	20.2771	20.2771	20.2771	20.2771	20.2771
3	20.2771	24.7062	24.7062	24.7062	24.7062	24.7062
4	24.7062	24.7062	29.1352	24.7062	24.7062	29.1352
5	29.1352	29.1352	33.5643	33.5643	33.5643	33.5643
6	33.5643	33.5643	37.9933	37.9933	37.9933	37.9933
7	37.9933	37.9933	37.9933	37.9933	37.9933	37.9933
8	42.4224	42.4224	42.4224	46.8514	46.8514	42.4224
9	46.8514	46.8514	46.8514	51.2805	51.2805	51.2805
10	51.2805	51.2805	51.2805	55.7095	55.7095	55.7095
11	55.7095	55.7095	55.7095	60.1386	60.1386	60.1386
12	60.1386	60.1386	60.1386	64.5676	64.5676	64.5676
13	64.5676	64.5676	64.5676	64.5676	68.9967	68.9967
14	68.9967	68.9967	68.9967	68.9967	73.4257	73.4257
15	73.4257	73.4257	73.4257	73.4257	77.8548	77.8548
16	77.8548	77.8548	77.8548	77.8548	82.2838	82.2838

ans = *20×7 table*

...

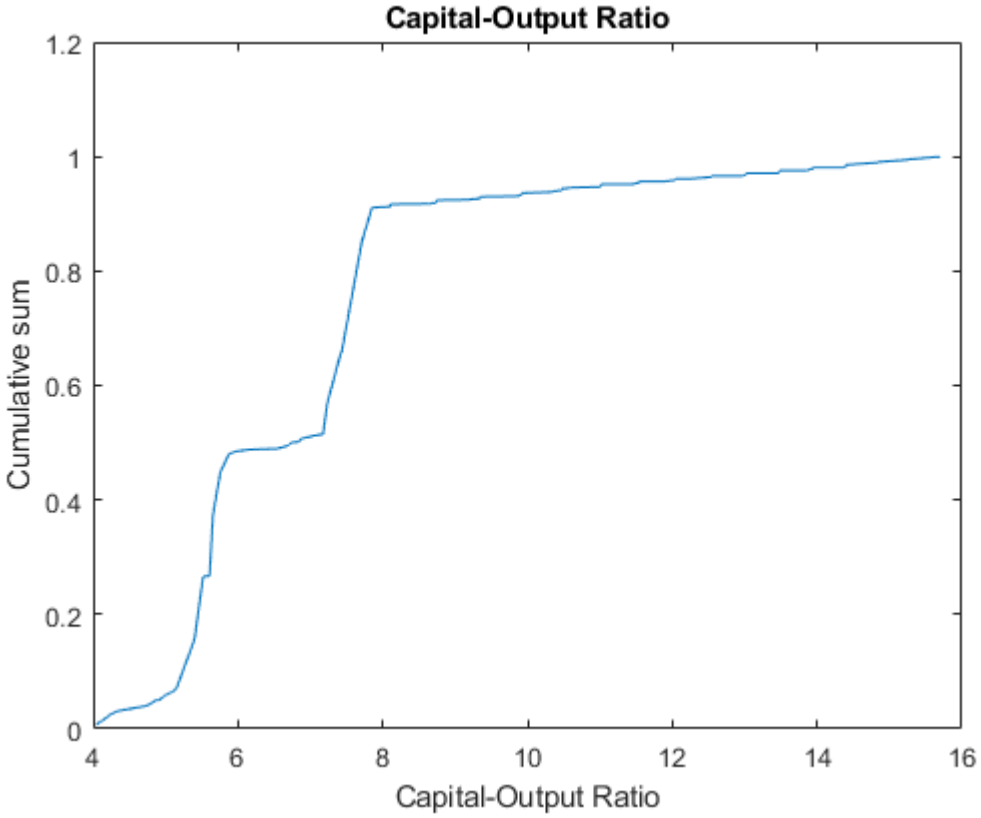
	Var1	Var2	Var3	Var4	Var5	Var6
1	0.0100	0.8437	0.6874	0.5832	0.4789	0.3747
2	0.0100	0.8958	0.7395	0.5832	0.4789	0.4268
3	0.0100	0.9479	0.7916	0.6353	0.5311	0.4268
4	0.0100	0.0100	0.8437	0.0100	0.0100	0.4268
5	0.0100	0.0100	0.8958	0.6874	0.5311	0.4268
6	0.0100	0.0100	0.9479	0.7395	0.5832	0.4789
7	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100

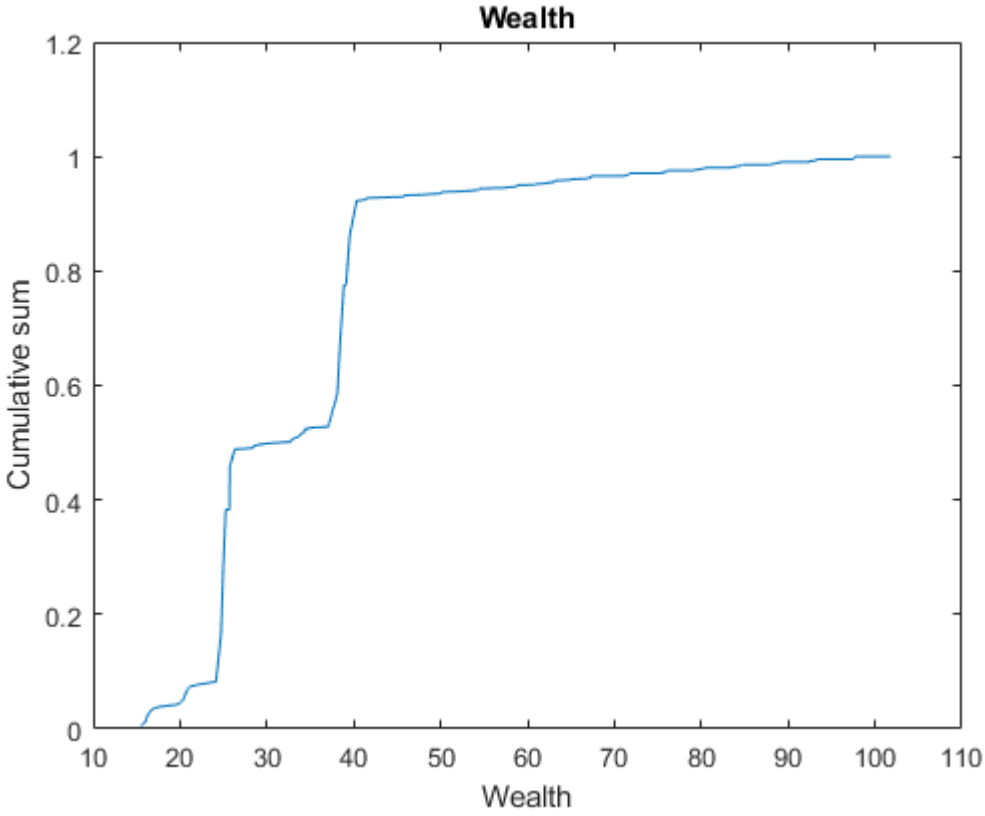
	Var1	Var2	Var3	Var4	Var5	Var6
8	0.0100	0.0100	0.0100	0.7916	0.6353	0.0100
9	0.0100	0.0100	0.0100	0.8437	0.6353	0.4789
10	0.0100	0.0100	0.0100	0.8437	0.6353	0.5311
11	0.0100	0.0100	0.0100	0.8958	0.6874	0.5311
12	0.0100	0.0100	0.0100	0.8958	0.6874	0.5311
13	0.0100	0.0100	0.0100	0.0100	0.7395	0.5311
14	0.0100	0.0100	0.0100	0.0100	0.7395	0.5832
15	0.0100	0.0100	0.0100	0.0100	0.7395	0.5832
16	0.0100	0.0100	0.0100	0.0100	0.7916	0.5832

ans = *20×7 table*

...

	Lambda1	Lambda2	Lambda3	Lambda4	Lambda5	Lambda6
1	0.0048	0.0074	0.0090	0.0085	0.0058	0.0024
2	0.0049	0.0075	0.0093	0.0088	0.0061	0.0026
3	0.0050	0.0835	0.1101	0.1071	0.0742	0.0313
4	0.0021	0.0038	0.0026	0.0001	0.0001	0.0009
5	0.0021	0.0040	0.0056	0.0060	0.0049	0.0029
6	0.0021	0.0040	0.0532	0.0892	0.1007	0.0866
7	0.0022	0.0018	0.0008	0.0000	0.0000	0.0000
8	0.0023	0.0018	0.0008	0.0006	0.0003	0.0000
9	0.0024	0.0017	0.0007	0.0006	0.0004	0.0002
10	0.0025	0.0017	0.0006	0.0005	0.0004	0.0002
11	0.0026	0.0017	0.0006	0.0008	0.0009	0.0008
12	0.0028	0.0016	0.0005	0.0013	0.0011	0.0008
13	0.0029	0.0015	0.0003	0.0000	0.0000	0.0000
14	0.0031	0.0014	0.0003	0.0000	0.0000	0.0000
15	0.0032	0.0013	0.0002	0.0000	0.0000	0.0000
16	0.0034	0.0012	0.0002	0.0000	0.0000	0.0000


Capital-Output Ratio
Cumulative sum
Capital-Output Ratio
0
0.2
0.4
0.6
0.8
1
1.2
4
6
8
10
12
14
16


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

SCRIPTS REFERENTE A QUESTAO 1

```
function out = CalculaLongoPrazo(PI)

N = size(PI,1);
prob_0 = (1/N)*ones(N,1);
error = 1;
while error > 1e-8
    % Calcula a probabilidade do proximo estado
    prob_1 = PI'*prob_0;

    % determina a variacao entre as duas probabilidades
    error = max(abs(prob_1 - prob_0));
    prob_0 = prob_1;
end

out = prob_0;

end % end of function
```

```
function out = ConstructLambda(Policy, Asset, Income)
% Calcula matrix de distribuicao lambda a partir de uma politica e dos
% estados.

% Inicia uma matri de zeros
lambda = zeros(Income.Grid.N*Asset.Grid.N);

for nShockState = 1:Income.Grid.N

    % inicializo a matrix de Transicao gerada pela politica
    M = zeros(Asset.Grid.N);

    % constroi a matrix de Transicao Politica
    for nLinha = 1:Asset.Grid.N
       M(nLinha, Policy.AssetPrime.Index(nLinha, nShockState)) =  1;
    end

    % seleciona as linhas da matrix lambda que serao alteradas
    lines =(1:Asset.Grid.N)+ (Asset.Grid.N*(nShockState -1));

    % altera a matrix lambda
    lambda(lines,:) = kron(Income.PI(nShockState,:),M);
end

    out = CalculaLongoPrazo(lambda);
    out = reshape(out, Asset.Grid.N, Income.Grid.N);
end % end of function
```

```
function [V0, U_Cube, Policy] = SolveConsumerProblem(Asset, Income, Econom_param)

U_Cube = UtilityCube(Asset.Values, Income.Values, Econom_param);

% define vetor inicial de chutes (lin:asset col:income)
V0 = zeros(Asset.Grid.N,2);

% define a variavel de cubo do V0 que sera utilizado na interacao
V0_cube = nan(Asset.Grid.N, Asset.Grid.N, 2);

% declara o vetor de politica
pol_a_idx = nan(Asset.Grid.N, 2);

Test_param.tolerance = 0.0001;
Test_param.error = 2;
nIter = 0;
while Test_param.error > Test_param.tolerance

    % Calcula o valor experado da da funcao valor
    ETV = V0 * Income.PI';

    % Transforma o valor esperado em um Cubo para calculo
    V0_cube = repmat(reshape(ETV,[Asset.Grid.N, 1, 2]), 1, Asset.Grid.N, 1);

    % Calcula o maximo pelo operador TV
    [V1, pol_a_idx] = TV_op(U_Cube.Values, Econom_param, V0_cube);

   % Incrementa o contador de interacoes
    nIter = nIter + 1;

    % Atualiza o erro de interacao
    Test_param.error = norm(V1(:) - V0(:));

    % Atualiza o valor de V0
    V0 = V1;
end

% DEFINICAO DAS POLITICAS
Policy.AssetDomain = U_Cube.a_domain;
Policy.AssetPrime.Values = U_Cube.a_domain(pol_a_idx);
Policy.AssetPrime.Index = pol_a_idx;
Policy.Consumption.Values = ones(1,Asset.Grid.N)'*Income.Values + Asset.Values'*ones(1,↙
Income.Grid.N).*(1 + Econom_param.r) - Policy.AssetPrime.Values;


end % end of fucntion
```

```
function [TV_1, pol_k] = TV_op(U_cube, Econom_param, V0_cube)
    % Calcula a Utilidade em um "cubo" de grid de variaveis.
    % (1)-axis: k1
    % (2)-axis: h
    % (3)-axis: k

    % quantidade de elementos na dimensao de income
    T = size(V0_cube, 3);

    % Inicializacao de variaveis
    TV_1 = nan(size(U_cube,1), T);
    pol_k = nan(size(U_cube,1),T);

    for i=1:T
        % Finds the new TV1
        ChoiceMatrix = U_cube(:,:,i) + Econom_param.Beta .* V0_cube(:,:,i);
        [maxValue, index] = max(ChoiceMatrix);

        TV_1(:,i) = maxValue';

        pol_k(:,i) = index';
    end
end % end of function
```

```
function U = UtilityCube(assetGridValues, incomeStatesValues, Econom_param)

% Constroi o Grid de a, a1 e c
U.a_domain = assetGridValues;
U.z_domain = incomeStatesValues;
[U.k, U.k1, U.z] = meshgrid(U.a_domain, U.a_domain, U.z_domain);

U.c = U.z + U.k.*(1 + Econom_param.r) - U.k1;

U.c(U.c<0) = 0;

U.Values = (U.c.^(1-Econom_param.Sigma)-1) ./ (1 - Econom_param.Sigma);

end % end of fucntion
```

```
%% 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_02
% https://github.com/btebaldi/Macro3/tree/master/PSet_03>
%
%
%% QUESTÃO 1 (a)-(e)

%% LIMPEZA DE VARIAVEIS
clearvars
clc
%% ITEM A DEFINICAO DE PARAMETROS
%
% Defininimos os parametros que sao utilizados pelo problema
Econom_param.PeriodsPerYear = 6;
Econom_param.Beta_anual = 0.96;
Econom_param.Beta = (Econom_param.Beta_anual)^(1/Econom_param.PeriodsPerYear);
Econom_param.Sigma = 1.5;

eps = 1e-5;

%% DEFINICAO DO GRID DE INCOME
%
% Vamos Construir a matrix de transição baseada nas informações fornecidas
Income.Grid.N = 2;
Income.Grid.Max = 1;
Income.Grid.Min = 0.1;
Income.Values = linspace(Income.Grid.Max, Income.Grid.Min, Income.Grid.N);
Income.PI = [0.925 (1-0.925); 0.5 (1-0.5)];
Income.P_LongRun = CalculaLongoPrazo(Income.PI);
%%
% Logo a renda media de equilibrio no longo prazo sera.
Income.Average = Income.Values*Income.P_LongRun;
Income.Early = Income.Average * Econom_param.PeriodsPerYear;
%%
% A duração média do desemprego é dada por 1/f, onde f neste caso é 0.5. Sendo assim a↙
duracao média do desemprego é dois periodos. Como o periodo é de dois meses, temos que a↙
duração média de desemprego é de 4 meses.

%% DEFINICAO DO GRID de ASSET

Borrow.Limit = -Income.Early;
Asset.Grid.N = 20;
Asset.Grid.Max = 3*Income.Average;
Asset.Grid.Min = Borrow.Limit;
```

```
Asset.Values = linspace(Asset.Grid.Min, Asset.Grid.Max, Asset.Grid.N);


%% DEFINICAO DA TAXA DE JUROS
%
% Inicia a taxa de juros
Econom_param.r_anual = 0.034;
Econom_param.r_start = ((1+Econom_param.r_anual)^(1/Econom_param.PeriodsPerYear))-1;
Econom_param.r_UpperLimit = 1/Econom_param.Beta - 1;
Econom_param.r_LowerLimit = 0;
Econom_param.r = Econom_param.r_start;

% parametro que indica se a taxa esta fixa ou nao.
Econom_param.r_IsFixed = 0;

%% ACHANDO O EQUILIBRIO.
% # Determinar um r inicial
% # Resolver o Problema do Consumidor e obter a Politica
% # Determinar Lambda
% # Verificar se existe eequilibrio no mercado de assets
% # Se nao ha equilibrio estabelecer novo r, (voltar a ponto 2) onde
%   $e>0 \rightarrow r_j+1 < r_j$
%   $e<0 \rightarrow r_j+1 > r_j$


nIntLimit = 1000;

for i=1:nIntLimit

    % (2) Resolve o Problema do consumidor
    [V0, U_Cube, Policy] = SolveConsumerProblem(Asset, Income, Econom_param);

    % (3) Determina a distribuicao estacionaria
    Lambda = ConstructLambda(Policy, Asset, Income);

    % (4) Determina demanda de assets
    Demanda = Lambda(:)' * Policy.AssetPrime.Values(:);

    % (5) Verifica se há equilibrio
    if abs(Demanda) < eps
        break;
    elseif Demanda > eps
        Econom_param.r_UpperLimit = Econom_param.r;
    elseif  Demanda < -eps
        Econom_param.r_LowerLimit = Econom_param.r;
    end

    % Caso a taxa esteja fixa não ha mais nada o que fazer
    if Econom_param.r_IsFixed == 1
        break;
    end
```

```
    % determina novo r
    r_old = Econom_param.r;
    Econom_param.r = (Econom_param.r_UpperLimit + Econom_param.r_LowerLimit)/2;

    % Caso a precisao da taxa seja muito pequena paramos a execucao.
    if abs(Econom_param.r - r_old) < eps^2
        break;
    end

    fprintf('Inter:%4d\tr_0: %1.6f\tr_1: %1.6f\tDem: %2.15f\n', i, r_old, Econom_param.r,↙
Demanda);
end

% Limpa variaveis nao utilizadas
clear nIntLimit r_old
%% GRAFICO DA FUNCAO POLITICA DE $A_{t+1}$ CONTRA $A_{t}$

fig1 = figure();
plot(Policy.AssetDomain, Policy.AssetPrime.Values(:,1),'b-.','LineWidth',1); hold on;
plot(Policy.AssetDomain, Policy.AssetPrime.Values(:,2),'r','LineWidth',2); hold on;
plot(Policy.AssetDomain, Policy.AssetDomain,'k:','LineWidth',1); hold off;
legend({'Low Employment State','High Employment State','45-degree Ray'});
title('Household Savings Policy Function');
xlabel('$a_{i,t}$','FontSize',12,'Interpreter','latex');
ylabel('$a_{i,t+1}$','FontSize',12,'Interpreter','latex');


%% SIMULACAO DE LIFE HISTORY
%
% define uma funcao que indica, dado um valor de asset, qual o valor do grid que mais se↙
aproxima do valor informado. Em outras palavras informa aonde um valor fornecido estaria↙
no grid.
Policy.AssetIndexFunction = @(value) round((value-Policy.AssetDomain(1))/(Policy.↙
AssetDomain(2)-Policy.AssetDomain(1))) + 1;

% Total de simulacoes
Simulation.N = 10000;

% Nivel inicial de ativos
asset_0 = 0;

% Estado inicial: Empregado.
Simulation.S0 = 1;
Simulation.Wage = 1;

% Inicializa os vetores da simulacao.
Simulation.Asset = nan(Simulation.N,1);
Simulation.AssetIndex = nan(Simulation.N,1);
Simulation.Consumption = nan(Simulation.N,1);
Simulation.Investment = nan(Simulation.N,1);
```

```
Simulation.LaborIncome = nan(Simulation.N,1);
Simulation.AssetIncome = nan(Simulation.N,1);
Simulation.Shock = nan(Simulation.N,1);

% Gera uma cadeia de choques no salario
[stateValue, stateIndex] = MarkovSimulation(Income.PI, Simulation.N, Income.Values,↙
Simulation.S0);

for i = 1:Simulation.N
    % pega o estado do choque.
    Simulation.Asset(i) = asset_0;
    Simulation.AssetIndex(i) = Policy.AssetIndexFunction(asset_0);

    Simulation.Shock(i) = stateValue(i);

    % determina o investimento baseado no estado atual (asset\capital e
    % choque)
    Simulation.Investment(i) = Policy.AssetPrime.Values(Simulation.AssetIndex(i),↙
stateIndex(i));

    % Determina a renda proveniente do trabalho
    Simulation.LaborIncome(i) = Simulation.Wage*stateValue(i);

    % Determina a renda proveniente do rendimento dos assets
    Simulation.AssetIncome(i) = (1 + Econom_param.r)*Simulation.Asset(i);

    % Determina o consumo pela equacao de equilibrio
    Simulation.Consumption(i) = Simulation.AssetIncome(i) + Simulation.LaborIncome(i) -↙
Simulation.Investment(i);

    % Atualiza o asset
    asset_0 = Simulation.Investment(i);
end
fprintf('DONE\n');

%% GRAFICOS
% Grafico de investimento vs consumo
x = 1:Simulation.N;
plot(x, Simulation.Investment, x, Simulation.Consumption);
title('INVESTMENT AND CONSUMPTION');
legend({'Investment','Consumption'},'FontSize', 8, 'Location','southeast',↙
'Orientation','Horizontal');
%%
% Covariancia de consumo e investimento
cov(Simulation.Consumption,Simulation.Investment)
%%
% Grafico de renda vs consumo
figure
plot(x, Simulation.LaborIncome, x, Simulation.Consumption);
title('LABOR INCOME AND CONSUMPTION');
legend({'Labor income','Consumption'},'FontSize', 8, 'Location','southeast',↙
```

```
'Orientation','Horizontal');
%%
% Covariancia de renda e consumo
cov(Simulation.LaborIncome, Simulation.Consumption)
%%
% Grafico de renda vs investimento
figure
plot(x, Simulation.LaborIncome, x, Simulation.Investment);
title('LABOR INCOME AND INVESTMENT');
legend({'Labor income','Investment'},'FontSize', 8, 'Location','southeast',↙
'Orientation','Horizontal');
%%
% Covariancia de renda e investimento
cov(Simulation.LaborIncome, Simulation.Investment)
%%
% Grafico de riqueza total
figure
plot(x, Simulation.LaborIncome + Simulation.AssetIncome);
title('Income');
legend({'Income'},'FontSize', 8, 'Location','southeast', 'Orientation','Horizontal');
%%
% Histograma de ativos
figure
histogram(Simulation.Investment)
title('HISTOGRAM OF ASSET HOLDINGS')
xlabel('Value of the Asset Holding','FontSize',10,'Interpreter','latex');
ylabel('Number of periods','FontSize',10,'Interpreter','latex')

% limpa variaveis
clear x
%% CALCULO DOS VALORES EXPERADOS
% average value of asset holdings

mean(Simulation.Asset)
%%
% Average decline in consumption in response to entering unemployment
Delta.Consumption = Simulation.Consumption(2:end) - Simulation.Consumption(1:end-1);
Delta.Shock =  Simulation.Shock(2:end) - Simulation.Shock(1:end-1);
mean(Delta.Consumption(Delta.Shock < 0))

clear Delta
%%
% Average consumption conditional on (i) employed; (ii) unemployed; (iii)
% unemployed for the last 12 months.
mean(Simulation.Consumption(Simulation.Shock==1))
mean(Simulation.Consumption(Simulation.Shock==0.1))

A = (Simulation.Shock==0.1)';
B = [1 1 1 1 1 1];
Unemployed6m = strfind(A,B)';
```

```
Consumption6Months = 0;
TotalPeriods = size(Unemployed6m,1);
for i=1:TotalPeriods
    Consumption6Months = Consumption6Months + mean(Simulation.Consumption(Unemployed6m(i):Unemployed6m(i)+6));
end
Consumption6Months/TotalPeriods

clear A B Consumption6Months TotalPeriods Unemployed6m
%% %% INCOME DISTRIBUTION

Total_Income =  ((1+Econom_param.r)*Asset.Values)'*ones(1,Income.Grid.N) + ones(1,Asset.Grid.N)'*Income.Values;
[sortedIncome, index] = sort(Total_Income(:));
lambda_aux = Lambda(:);

figure
plot(sortedIncome, lambda_aux(index));
title('INCOME DISTRIBUTION');
xlabel('INCOME LEVEL');
ylabel('% OF AGENTS');


%%  STANDARD MEASURES FOR WEALTH DISPERSION

mean_w = mean(Simulation.AssetIncome+Simulation.LaborIncome);
stdr_w = std(Simulation.AssetIncome+Simulation.LaborIncome);

figure
histogram(Simulation.AssetIncome+Simulation.LaborIncome);
annotation('textbox',[.2 .3 .4 .5],...
    'String',{['Mean = ' num2str(mean_w)],['Stddev =' num2str(stdr_w)]},...
'FitBoxtoText','on');
title(sprintf('Wealth, sigma=%1.1f',Econom_param.Sigma));
```

```
%% 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_02
% https://github.com/btebaldi/Macro3/tree/master/PSet_03>
%
%
%% QUESTÃO 1 (f)
%% LIMPEZA DE VARIAVEIS
clearvars
clc
%% ITEM A DEFINICAO DE PARAMETROS
%
% Defininimos os parametros que sao utilizados pelo problema
Econom_param.PeriodsPerYear = 6;
Econom_param.Beta_anual = 0.96;
Econom_param.Beta = (Econom_param.Beta_anual)^(1/Econom_param.PeriodsPerYear);
Econom_param.Sigma = 3;

eps = 1e-5;

%% DEFINICAO DO GRID DE INCOME
%
% Vamos Construir a matrix de transição baseada nas informações fornecidas
Income.Grid.N = 2;
Income.Grid.Max = 1;
Income.Grid.Min = 0.1;
Income.Values = linspace(Income.Grid.Max, Income.Grid.Min, Income.Grid.N);
Income.PI = [0.925 (1-0.925); 0.5 (1-0.5)];
Income.P_LongRun = CalculaLongoPrazo(Income.PI);
%%
% Logo a renda media de equilibrio no longo prazo sera.
Income.Average = Income.Values*Income.P_LongRun;
Income.Early = Income.Average * Econom_param.PeriodsPerYear;
%%
% A duração média do desemprego é dada por 1/f, onde f neste caso é 0.5. Sendo assim a↙
duracao média do desemprego é dois periodos. Como o periodo é de dois meses, temos que a↙
duração média de desemprego é de 4 meses.

%% DEFINICAO DO GRID de ASSET

Borrow.Limit = -Income.Early;
Asset.Grid.N = 20;
Asset.Grid.Max = 3*Income.Average;
Asset.Grid.Min = Borrow.Limit;
Asset.Values = linspace(Asset.Grid.Min, Asset.Grid.Max, Asset.Grid.N);
```

```
%% DEFINICAO DA TAXA DE JUROS
%
% Inicia a taxa de juros
Econom_param.r_anual = 0.034;
Econom_param.r_start = ((1+Econom_param.r_anual)^(1/Econom_param.PeriodsPerYear))-1;
Econom_param.r_UpperLimit = 1/Econom_param.Beta - 1;
Econom_param.r_LowerLimit = 0;
Econom_param.r = Econom_param.r_start;

% parametro que indica se a taxa esta fixa ou nao.
Econom_param.r_IsFixed = 0;

%% ACHANDO O EQUILIBRIO.
% # Determinar um r inicial
% # Resolver o Problema do Consumidor e obter a Politica
% # Determinar Lambda
% # Verificar se existe eequilibrio no mercado de assets
% # Se nao ha equilibrio estabelecer novo r, (voltar a ponto 2) onde
%   $e>0 \rightarrow r_j+1 < r_j$
%   $e<0 \rightarrow r_j+1 > r_j$


nIntLimit = 1000;

for i=1:nIntLimit

    % (2) Resolve o Problema do consumidor
    [V0, U_Cube, Policy] = SolveConsumerProblem(Asset, Income, Econom_param);

    % (3) Determina a distribuicao estacionaria
    Lambda = ConstructLambda(Policy, Asset, Income);

    % (4) Determina demanda de assets
    Demanda = Lambda(:)' * Policy.AssetPrime.Values(:);

    % (5) Verifica se há equilibrio
    if abs(Demanda) < eps
        break;
    elseif Demanda > eps
        Econom_param.r_UpperLimit = Econom_param.r;
    elseif  Demanda < -eps
        Econom_param.r_LowerLimit = Econom_param.r;
    end

    % Caso a taxa esteja fixa não ha mais nada o que fazer
    if Econom_param.r_IsFixed == 1
        break;
    end
```

```
    % determina novo r
    r_old = Econom_param.r;
    Econom_param.r = (Econom_param.r_UpperLimit + Econom_param.r_LowerLimit)/2;

    % Caso a precisao da taxa seja muito pequena paramos a execucao.
    if abs(Econom_param.r - r_old) < eps^2
        break;
    end

    fprintf('Inter:%4d\tr_0: %1.6f\tr_1: %1.6f\tDem: %2.15f\n', i, r_old, Econom_param.r,↙
Demanda);
end

% Limpa variaveis nao utilizadas
clear nIntLimit r_old
%% GRAFICO DA FUNCAO POLITICA DE $A_{t+1}$ CONTRA $A_{t}$

fig1 = figure();
plot(Policy.AssetDomain, Policy.AssetPrime.Values(:,1),'b-.','LineWidth',1); hold on;
plot(Policy.AssetDomain, Policy.AssetPrime.Values(:,2),'r','LineWidth',2); hold on;
plot(Policy.AssetDomain, Policy.AssetDomain,'k:','LineWidth',1); hold off;
legend({'Low Employment State','High Employment State','45-degree Ray'});
title('Household Savings Policy Function');
xlabel('$a_{i,t}$','FontSize',12,'Interpreter','latex');
ylabel('$a_{i,t+1}$','FontSize',12,'Interpreter','latex');


%% SIMULACAO DE LIFE HISTORY
%
% define uma funcao que indica, dado um valor de asset, qual o valor do grid que mais se↙
aproxima do valor informado. Em outras palavras informa aonde um valor fornecido estaria↙
no grid.
Policy.AssetIndexFunction = @(value) round((value-Policy.AssetDomain(1))/(Policy.↙
AssetDomain(2)-Policy.AssetDomain(1))) + 1;

% Total de simulacoes
Simulation.N = 10000;

% Nivel inicial de ativos
asset_0 = 0;

% Estado inicial: Empregado.
Simulation.S0 = 1;
Simulation.Wage = 1;

% Inicializa os vetores da simulacao.
Simulation.Asset = nan(Simulation.N,1);
Simulation.AssetIndex = nan(Simulation.N,1);
Simulation.Consumption = nan(Simulation.N,1);
Simulation.Investment = nan(Simulation.N,1);
Simulation.LaborIncome = nan(Simulation.N,1);
```

```
Simulation.AssetIncome = nan(Simulation.N,1);
Simulation.Shock = nan(Simulation.N,1);

% Gera uma cadeia de choques no salario
[stateValue, stateIndex] = MarkovSimulation(Income.PI, Simulation.N, Income.Values,↙
Simulation.S0);

for i = 1:Simulation.N
    % pega o estado do choque.
    Simulation.Asset(i) = asset_0;
    Simulation.AssetIndex(i) = Policy.AssetIndexFunction(asset_0);

    Simulation.Shock(i) = stateValue(i);

    % determina o investimento baseado no estado atual (asset\capital e
    % choque)
    Simulation.Investment(i) = Policy.AssetPrime.Values(Simulation.AssetIndex(i),↙
stateIndex(i));

    % Determina a renda proveniente do trabalho
    Simulation.LaborIncome(i) = Simulation.Wage*stateValue(i);

    % Determina a renda proveniente do rendimento dos assets
    Simulation.AssetIncome(i) = (1 + Econom_param.r)*Simulation.Asset(i);

    % Determina o consumo pela equacao de equilibrio
    Simulation.Consumption(i) = Simulation.AssetIncome(i) + Simulation.LaborIncome(i) -↙
Simulation.Investment(i);

    % Atualiza o asset
    asset_0 = Simulation.Investment(i);
end
fprintf('DONE\n');

%% GRAFICOS
% Grafico de investimento vs consumo
x = 1:Simulation.N;
plot(x, Simulation.Investment, x, Simulation.Consumption);
title('INVESTMENT AND CONSUMPTION');
legend({'Investment','Consumption'},'FontSize', 8, 'Location','southeast',↙
'Orientation','Horizontal');
%%
% Covariancia de consumo e investimento
cov(Simulation.Consumption,Simulation.Investment)
%%
% Grafico de renda vs consumo
figure
plot(x, Simulation.LaborIncome, x, Simulation.Consumption);
title('LABOR INCOME AND CONSUMPTION');
legend({'Labor income','Consumption'},'FontSize', 8, 'Location','southeast',↙
'Orientation','Horizontal');
```

```
%%
% Covariancia de renda e consumo
cov(Simulation.LaborIncome, Simulation.Consumption)
%%
% Grafico de renda vs investimento
figure
plot(x, Simulation.LaborIncome, x, Simulation.Investment);
title('LABOR INCOME AND INVESTMENT');
legend({'Labor income','Investment'},'FontSize', 8, 'Location','southeast',↙
'Orientation','Horizontal');
%%
% Covariancia de renda e investimento
cov(Simulation.LaborIncome, Simulation.Investment)
%%
% Grafico de riqueza total
figure
plot(x, Simulation.LaborIncome + Simulation.AssetIncome);
title('Income');
legend({'Income'},'FontSize', 8, 'Location','southeast', 'Orientation','Horizontal');
%%
% Histograma de ativos
figure
histogram(Simulation.Investment)
title('HISTOGRAM OF ASSET HOLDINGS')
xlabel('Value of the Asset Holding','FontSize',10,'Interpreter','latex');
ylabel('Number of periods','FontSize',10,'Interpreter','latex')

% limpa variaveis
clear x
%% CALCULO DOS VALORES EXPERADOS
% average value of asset holdings

mean(Simulation.Asset)
%%
% Average decline in consumption in response to entering unemployment
Delta.Consumption = Simulation.Consumption(2:end) - Simulation.Consumption(1:end-1);
Delta.Shock =  Simulation.Shock(2:end) - Simulation.Shock(1:end-1);
mean(Delta.Consumption(Delta.Shock < 0))

clear Delta
%%
% Average consumption conditional on (i) employed; (ii) unemployed; (iii)
% unemployed for the last 12 months.
mean(Simulation.Consumption(Simulation.Shock==1))
mean(Simulation.Consumption(Simulation.Shock==0.1))

A = (Simulation.Shock==0.1)';
B = [1 1 1 1 1 1];
Unemployed6m = strfind(A,B)';

Consumption6Months = 0;
```

```
TotalPeriods = size(Unemployed6m,1);
for i=1:TotalPeriods
    Consumption6Months = Consumption6Months + mean(Simulation.Consumption(Unemployed6m↙
(i):Unemployed6m(i)+6));
end
Consumption6Months/TotalPeriods

clear A B Consumption6Months TotalPeriods Unemployed6m
%% %% INCOME DISTRIBUTION

Total_Income =  ((1+Econom_param.r)*Asset.Values)'*ones(1,Income.Grid.N) + ones(1,Asset.↙
Grid.N)'*Income.Values;
[sortedIncome, index] = sort(Total_Income(:));
lambda_aux = Lambda(:);

figure
plot(sortedIncome, lambda_aux(index));
title('INCOME DISTRIBUTION');
xlabel('INCOME LEVEL');
ylabel('% OF AGENTS');


%%  STANDARD MEASURES FOR WEALTH DISPERSION

mean_w = mean(Simulation.AssetIncome+Simulation.LaborIncome);
stdr_w = std(Simulation.AssetIncome+Simulation.LaborIncome);

figure
histogram(Simulation.AssetIncome+Simulation.LaborIncome);
annotation('textbox',[.2 .3 .4 .5],...
    'String',{['Mean = ' num2str(mean_w)],['Stddev =' num2str(stdr_w)]},...
'FitBoxtoText','on');
title(sprintf('Wealth, sigma=%1.1f',Econom_param.Sigma));
```

```
function out = AnaliseMarkovToAr(mkvStruct, eps)
% Autor: Bruno Tebaldi Q Barbosa
%
% Adaptação do codigo de Tiago Cavalcanti
% Calculate the invariant distribution of Markov chain by simulating the
% chain to reach a long-run level
%
% Imputs:
% PI: Matriz de transição do processo de markov
% z: Vetor de estados
%
%
%
% eps:
if nargin <2
   eps = 1e-8;
end


% Assume ua probabilidade igual para se iniciar em todos os estados.
prob = (1/mkvStruct.QtdStates)*ones(mkvStruct.QtdStates,1); % initial distribution of↙
states
test = 1;

% Calcula o estado final de equilibrio
while test > eps
    probst1 = mkvStruct.TransitionMatrix'*prob;
    test=max(abs(probst1-prob));
    prob = probst1;
end

% Calculate Properties of Invariant Distribution
meanm = mkvStruct.StateVector*prob;               % mean of invariant distribution
varm = ((mkvStruct.StateVector-meanm).^2)*prob;   % variance of invariant distribution

midaut1 = (mkvStruct.StateVector-meanm)'*(mkvStruct.StateVector-meanm); % cross product↙
of deviation from the
                                % mean of y_t and y_t-1

probmat = prob*ones(1,mkvStruct.QtdStates);     % each column is invariant distribution

midaut2 = mkvStruct.TransitionMatrix.*probmat.*midaut1; % product of the first two terms↙
is
                                % the joint distribution of (Y_t-1,Y_t)

autcov1 = sum(sum(midaut2));      % first-order auto-covariance

alambda = autcov1/varm;           % persistence of discrete process
asigmay = sqrt(varm);            % s.d. of discrete process
```

```
% Calculate the Asymptotic second moments of Markov chain

fprintf('_____________________________________________________\n')
fprintf('                    original process     Markov chain\n')
fprintf('_____________________________________________________\n')
fprintf('Persistence         %16.6f %16.6f\n', mkvStruct.AR.rho, alambda);
fprintf('Standard deviation  %16.6f %16.6f\n', mkvStruct.AR.sigma2_y^0.5, asigmay);
fprintf('_____________________________________________________\n')
end
```

```
function mkv = MarkovProcess(rho,sigma,r,N,mu)
% Autor: Bruno Tebaldi Q Barbosa
%
% Modela um processo AR(1) por um modelo de cadeia de markov.
%  y(t)= (1-rho)*mu + rho*y(t-1) + e(t)
%
% Imputs:
% rho: Fator de correlação do AR(1)
% sigma: Variancia dos erros no AR(1)
% r: Quantidade de desvios padores que os estados mais distantes do
%    processo de Markov deve ter
% N: Quantidade total de estados no processo de Markov.
% mu: Média de longo prazo do processo AR(1)


% Validade\Fill in unset optional values.
switch nargin
    case {0,1,2,3}
        error('função necessita pelo menos de 4 argumentos');
    case 4
        mu=0;
end

% Valida rho
if abs(rho) >= 1
    error('processo ar não estacionario. (rho = %f)', rho);
end

% Discretizacao do espaco de estado
dpy = sqrt(sigma^2/(1-rho^2));    % desvio padrao de y_t (não condicional)
z_N = mu + r*dpy;                 % limite superior
z_1 = mu - r*dpy;                 % limite inferior

d = (z_N-z_1)/(N-1);              % tamanho do intervalo
z = z_1:d:z_N;                    % grid de estados

% Inicia a matriz de tranzicao
PI = nan(N);

% Calculate the transition matrix - see Tauchen
for lin=1:N
    for col=2:N-1
        PI(lin,col)= normcdf(z(col)+d/2-rho*z(lin) -mu*(1-rho), 0, sigma)...
            - normcdf(z(col)-d/2-rho*z(lin) -mu*(1-rho), 0, sigma);
    end

    % Casos de j={1 ,N}
    PI(lin,1) = normcdf(z(1)+d/2-rho*z(lin)-mu*(1-rho),0,sigma);
    PI(lin,N) = 1 - normcdf(z(N)-d/2-rho*z(lin)-mu*(1-rho),0,sigma);
end
```

```
% Verifica se as probabilidade somam um por linha
if sum(PI,2) ~= ones(N,1)
    % find rows not adding up to one
    rowNotSumTo1 = find(sum(PI,2) < 1);  % find rows not adding up to one
    fprintf('Error in transition matrix\n');
    fprintf('row %d does not sum to one\n', rowNotSumTo1);
end

% Cria structure de resposta
mkv.AR.form = 'y(t)= (1-rho)*mu + rho*y(t-1) + e(t)';
mkv.AR.mu = mu;
mkv.AR.rho = rho;
mkv.AR.sigma2 = sigma;
mkv.AR.sigma2_y = dpy^2;
mkv.TransitionMatrix = PI;
mkv.StateVector = z;
mkv.QtdStates = N;
mkv.d = d;
mkv.StateBorder = z(1:end-1) + d/2;

end % end of function MarkovProcess
```

```
function [chain,state] = MarkovSimulation(PI, N, S, S0)
    % INPUTS:
    %   PI  : Transition matrix
    %   N   : length of simulation
    %   S   : State vector
    %   S0  : initial state (index)

    % ------------------------------------------------------------------------------
    % 1. validacoes de inputs

    % Verifica a quantidade de inputs minimo e maximo
    narginchk(2,4)

    [rPI, cPI] = size(PI);
    [rN, cN] = size(N);

    % Validade\Fill in unset optional values.
    switch nargin
        case 2
            S=1:rPI;
            S0=1;
        case 3
            S0=1;
    end

    [rS, cS] = size(S);

    % (a) Checking the total of shocks
    % Checks if N is a scalar or not
    if rN == 1 & cN == 1
        N = round(abs(N));
    else
        error('The first input must be a scalar');
    end

    % (b) checking the Transition matrix
    % Checks if it is a square matrix or not
    if (rPI == 1 | cPI == 1) | (rPI ~= cPI)
        error('The second input must be a square matrix.');
    end

    % changes the sign if some probabilities are negative
    PI = abs(PI);

    % checks if the probabilities sum to one, if not, it normalizes
    for i = 1:rPI
        if sum(PI(i,:)) ~= 1
            fprintf('probability: %f', sum(PI(i,:)));
            warning('The probabilities don`t sum to 1.');
            PI(i,:) = PI(i,:)/sum(PI(i,:));
        end
```

```
    end

    % (c) checking the State Vector
    if rS > 1
        error('State vector must be 1xN.')
    elseif ~(cS==cPI)
        error('Number of state does not match size of Transition matrix')
    end

    % -------------------------------------------------------------------------
    % 2. Creating the shock realizations

    % Cria matrix de somas acumuladas
    cum_PI = [zeros(rPI,1) cumsum(PI')'];

    % Cria o vetor de simulacao
    simulation = rand(N,1);
    state = nan(N,1);
    state(1) = S0;

    for i=2:N
        state(i)=find(((simulation(i) <= cum_PI(state(i-1), 2:cPI+1))&(simulation(i)↙
>cum_PI(state(i-1),1:cPI))));
    end

    chain=S(state);

end %end of function
```

SCRIPTS REFERENTE A QUESTAO 2

```
function out = CalculaLongoPrazo(PI)

N = size(PI,1);
prob_0 = (1/N)*ones(N,1);
error = 1;
while error > 1e-8
    % Calcula a probabilidade do proximo estado
    prob_1 = PI'*prob_0;

    % determina a variacao entre as duas probabilidades
    error = max(abs(prob_1 - prob_0));
    prob_0 = prob_1;
end

out = prob_0;

end % end of function
```

```
function out=CalculaRendaGov(tau_y, Labor, WageShocks, eco_param, report)

    % Fill in unset optional values.
    switch nargin
        case {4}
            report = 0;
    end

    eps = 1e-5;

    eco_param.tau_y = tau_y;

    fprintf('tau_y:%1.3f\ttau_n:%1.3f\n',eco_param.tau_y, eco_param.tau_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)/((1-eco_param.tau_y)*eco_param.alpha))^(1/↙
(eco_param.alpha -1));

        %% Determinar w:  $w = F_L(k)$;
        w = ((1-eco_param.tau_y)/(1+eco_param.tau_n) )*(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);

        [~, ~, 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

        if report == 1
            fprintf('Inter:%4d\tr: %1.6f\tDem: %2.6f\n', nContador, eco_param.r,↙
Demanda);
        end
    end
    Y = K^eco_param.alpha*L^(1-eco_param.alpha);
    out = eco_param.tau_y * Y + eco_param.tau_n*L*w;

    if report == 1

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

        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');
    end

end % end of function
```

```
function out = ConstructLambda(Policy, Asset, WageShocks)
% Calcula matrix de distribuicao lambda a partir de uma politica e dos
% estados.

% Inicia uma matri de zeros
lambda = zeros(Asset.Grid.N*WageShocks.Grid.N);

for nShockState = 1:WageShocks.Grid.N

    % inicializo a matrix de Transicao gerada pela politica
    M = zeros(Asset.Grid.N);

    % constroi a matrix de Transicao Politica
    for nLinha = 1:Asset.Grid.N
       M(nLinha, Policy.AssetPrime.Index(nLinha, nShockState)) =  1;
    end

    % seleciona as linhas da matrix lambda que serao alteradas
    lines =(1:Asset.Grid.N)+ (Asset.Grid.N*(nShockState -1));

    % altera a matrix lambda
    lambda(lines,:) = kron(WageShocks.PI(nShockState,:),M);
end

    out = CalculaLongoPrazo(lambda);
    out = reshape(out, Asset.Grid.N, WageShocks.Grid.N);
end % end of function
```

```
function [V0, U_Cube, Policy] = SolveConsumerProblem(Asset, Labor, WageShocks, wage,↙
eco_param, displayInfo, displayIter)

    % Fill in unset optional values.
    switch nargin
        case {5}
            displayInfo = 0;
            displayIter = 0;
        case 6
            displayIter = 0;
    end


    if displayInfo == 1
        fprintf('Beging solving Consumer Problem.\n');
    end

    U_Cube_1 = UtilityCube(Asset.Values, Labor.Values, wage*WageShocks.Values(1),↙
eco_param);
    U_Cube_2 = UtilityCube(Asset.Values, Labor.Values, wage*WageShocks.Values(2),↙
eco_param);
    U_Cube_3 = UtilityCube(Asset.Values, Labor.Values, wage*WageShocks.Values(3),↙
eco_param);
    U_Cube_4 = UtilityCube(Asset.Values, Labor.Values, wage*WageShocks.Values(4),↙
eco_param);
    U_Cube_5 = UtilityCube(Asset.Values, Labor.Values, wage*WageShocks.Values(5),↙
eco_param);
    U_Cube_6 = UtilityCube(Asset.Values, Labor.Values, wage*WageShocks.Values(6),↙
eco_param);
    U_Cube_7 = UtilityCube(Asset.Values, Labor.Values, wage*WageShocks.Values(7),↙
eco_param);

    % define vetor inicial de chutes (lin:asset col:income)
    V_star_1 = zeros(Asset.Grid.N, 1);
    V_star_2 = zeros(Asset.Grid.N, 1);
    V_star_3 = zeros(Asset.Grid.N, 1);
    V_star_4 = zeros(Asset.Grid.N, 1);
    V_star_5 = zeros(Asset.Grid.N, 1);
    V_star_6 = zeros(Asset.Grid.N, 1);
    V_star_7 = zeros(Asset.Grid.N, 1);

    % Initializa a condicao de parada das interacoes (um para cada estado)
    check=ones(1, 7);

    % inicializa os vetores de politica (K x z)
    policyIndex_L = nan(Asset.Grid.N, 7);
    policyIndex_A = nan(Asset.Grid.N, 7);

    Test_param.tolerance = 0.0001;
    Test_param.error = 1;
    nIter = 0;
```

```
    if displayIter==1
        fprintf('_____________________________________\n');
        fprintf('Interação      | eps (%f)\n', Test_param.tolerance);
        fprintf('-------------------------------------\n');
    end
    while Test_param.error > Test_param.tolerance
        % Incrementa o contador de interacoes
        nIter = nIter + 1;

        % CALCULA AS MATRIZES DE MEDIA DO TVi
        TV1_average = WageShocks.PI(1,1)*V_star_1 + ...
            WageShocks.PI(1,2)*V_star_2 + ...
            WageShocks.PI(1,3)*V_star_3 + ...
            WageShocks.PI(1,4)*V_star_4 + ...
            WageShocks.PI(1,5)*V_star_5 + ...
            WageShocks.PI(1,6)*V_star_6 + ...
            WageShocks.PI(1,7)*V_star_7;

        TV2_average = WageShocks.PI(2,1)*V_star_1 + ...
            WageShocks.PI(2,2)*V_star_2 + ...
            WageShocks.PI(2,3)*V_star_3 + ...
            WageShocks.PI(2,4)*V_star_4 + ...
            WageShocks.PI(2,5)*V_star_5 + ...
            WageShocks.PI(2,6)*V_star_6 + ...
            WageShocks.PI(2,7)*V_star_7;

        TV3_average = WageShocks.PI(3,1)*V_star_1 + ...
            WageShocks.PI(3,2)*V_star_2 + ...
            WageShocks.PI(3,3)*V_star_3 + ...
            WageShocks.PI(3,4)*V_star_4 + ...
            WageShocks.PI(3,5)*V_star_5 + ...
            WageShocks.PI(3,6)*V_star_6 + ...
            WageShocks.PI(3,7)*V_star_7;

        TV4_average = WageShocks.PI(4,1)*V_star_1 + ...
            WageShocks.PI(4,2)*V_star_2 + ...
            WageShocks.PI(4,3)*V_star_3 + ...
            WageShocks.PI(4,4)*V_star_4 + ...
            WageShocks.PI(4,5)*V_star_5 + ...
            WageShocks.PI(4,6)*V_star_6 + ...
            WageShocks.PI(4,7)*V_star_7;

        TV5_average = WageShocks.PI(5,1)*V_star_1 + ...
            WageShocks.PI(5,2)*V_star_2 + ...
            WageShocks.PI(5,3)*V_star_3 + ...
            WageShocks.PI(5,4)*V_star_4 + ...
            WageShocks.PI(5,5)*V_star_5 + ...
            WageShocks.PI(5,6)*V_star_6 + ...
            WageShocks.PI(5,7)*V_star_7;

        TV6_average = WageShocks.PI(6,1)*V_star_1 + ...
```

```
            WageShocks.PI(6,2)*V_star_2 + ...
            WageShocks.PI(6,3)*V_star_3 + ...
            WageShocks.PI(6,4)*V_star_4 + ...
            WageShocks.PI(6,5)*V_star_5 + ...
            WageShocks.PI(6,6)*V_star_6 + ...
            WageShocks.PI(6,7)*V_star_7;

        TV7_average = WageShocks.PI(7,1)*V_star_1 + ...
            WageShocks.PI(7,2)*V_star_2 + ...
            WageShocks.PI(7,3)*V_star_3 + ...
            WageShocks.PI(7,4)*V_star_4 + ...
            WageShocks.PI(7,5)*V_star_5 + ...
            WageShocks.PI(7,6)*V_star_6 + ...
            WageShocks.PI(7,7)*V_star_7;

        % Dimensionaliza as matrizes
        % Atencao, inverto TV1_average pois ele eh funcao de k
        V_cube_1 = repmat(TV1_average', Labor.Grid.N, 1, Asset.Grid.N);
        V_cube_2 = repmat(TV2_average', Labor.Grid.N, 1, Asset.Grid.N);
        V_cube_3 = repmat(TV3_average', Labor.Grid.N, 1, Asset.Grid.N);
        V_cube_4 = repmat(TV4_average', Labor.Grid.N, 1, Asset.Grid.N);
        V_cube_5 = repmat(TV5_average', Labor.Grid.N, 1, Asset.Grid.N);
        V_cube_6 = repmat(TV6_average', Labor.Grid.N, 1, Asset.Grid.N);
        V_cube_7 = repmat(TV7_average', Labor.Grid.N, 1, Asset.Grid.N);


        % Finds the new TV1
        [TV_1, policyIndex_L(:,1), policyIndex_A(:,1)] = TV_op(U_Cube_1.Values,↙
eco_param, V_cube_1);
        [TV_2, policyIndex_L(:,2), policyIndex_A(:,2)] = TV_op(U_Cube_2.Values,↙
eco_param, V_cube_2);
        [TV_3, policyIndex_L(:,3), policyIndex_A(:,3)] = TV_op(U_Cube_3.Values,↙
eco_param, V_cube_3);
        [TV_4, policyIndex_L(:,4), policyIndex_A(:,4)] = TV_op(U_Cube_4.Values,↙
eco_param, V_cube_4);
        [TV_5, policyIndex_L(:,5), policyIndex_A(:,5)] = TV_op(U_Cube_5.Values,↙
eco_param, V_cube_5);
        [TV_6, policyIndex_L(:,6), policyIndex_A(:,6)] = TV_op(U_Cube_6.Values,↙
eco_param, V_cube_6);
        [TV_7, policyIndex_L(:,7), policyIndex_A(:,7)] = TV_op(U_Cube_7.Values,↙
eco_param, V_cube_7);

        % Sets the new numerical value for the stopping rule
        check(1) = norm(TV_1 - V_star_1)/norm(V_star_1);
        check(2) = norm(TV_2 - V_star_2)/norm(V_star_2);
        check(3) = norm(TV_3 - V_star_3)/norm(V_star_3);
        check(4) = norm(TV_4 - V_star_4)/norm(V_star_4);
        check(5) = norm(TV_5 - V_star_5)/norm(V_star_5);
        check(6) = norm(TV_6 - V_star_6)/norm(V_star_6);
        check(7) = norm(TV_7 - V_star_7)/norm(V_star_7);
```

```
        Test_param.error = max(check);

        % Sets V to be the last TV we found
        V_star_1 = TV_1;
        V_star_2 = TV_2;
        V_star_3 = TV_3;
        V_star_4 = TV_4;
        V_star_5 = TV_5;
        V_star_6 = TV_6;
        V_star_7 = TV_7;

        if (displayIter==1) & (mod(nIter, 25) == 0)
            fprintf('  %13d| %12.10f \n', nIter, max(check));
        end
    end
    if displayIter==1
        fprintf('_____________________________________\n');
        fprintf('Total   %7d| %12.10f\n', nIter, max(check));
        fprintf('-------------------------------------\n');
    end

    V0=[V_star_1 V_star_2 V_star_3 V_star_4 V_star_5 V_star_6 V_star_7];

    U_Cube=[U_Cube_1 U_Cube_2 U_Cube_3 U_Cube_4 U_Cube_5 U_Cube_6 U_Cube_7];

    % DEFINICAO DAS POLITICAS
    Policy.AssetDomain = Asset.Values;
    Policy.LaborDomain = Labor.Values;

    Policy.AssetPrime.Index = policyIndex_A;
    Policy.Labor.Index = policyIndex_L;

    Policy.AssetPrime.Values = Policy.AssetDomain(Policy.AssetPrime.Index );
    Policy.Labor.Values = Policy.LaborDomain(Policy.Labor.Index);

    Policy.Wealth.Values = Asset.Values'*ones(1,7).*(1 + eco_param.r) ...
        + Policy.Labor.Values * diag(wage*WageShocks.Values);

    Policy.Consumption.Values = Asset.Values'*ones(1,7).*(1 + eco_param.r) ...
        + Policy.Labor.Values * diag(wage*WageShocks.Values) ...
        - Policy.AssetPrime.Values;

    Policy.Wages.Values = Policy.Labor.Values * diag(wage*WageShocks.Values);

    Policy.Asset.Values = Asset.Values'*ones(1,7);

    if displayInfo == 1
        fprintf('\nEnd of Consumer Problem.\n');
    end
end % end of fucntion
```

```
function [TV_1, pol_L, pol_A] = TV_op(U, eco_param, V0)
    % Calcula a Utilidade em um "cubo" de grid de variaveis.
    % (1)-axis: k1
    % (2)-axis: h
    % (3)-axis: k

    % quantidade de elementos na dimensao k
    T = size(U, 2);

    % Inicializacao de variaveis
    TV_1 = nan(T,1);
    pol_L = nan(size(U,1),1);
    pol_A = nan(size(U,2),1);

    for i=1:T
        % Finds the new TV1
        ChoiceMatrix = U(:,:,i) + eco_param.beta .* V0(:,:,i);
        [maxValue, index] = max(ChoiceMatrix(:));

        TV_1(i) = maxValue;

        [lin,col] = ind2sub(size(ChoiceMatrix), index);
        pol_L(i) = lin;
        pol_A(i) = col;
    end
end % end of function
```

```
function U = UtilityCube(assetGridValues, laborGridValues, wage, eco_param)

% Constroi o Grid de a, a1 e l
U.a_domain = assetGridValues;
U.l_domain = laborGridValues;
[U.a1, U.l, U.a] = meshgrid(U.a_domain, U.l_domain, U.a_domain);

% calcula o cubo de consumo
U.c = (1 + eco_param.r) .* U.a + wage*U.l - U.a1;

% Nao existe consumo negativo
U.c(U.c<=0) = 0.00001;

% Calcula cubo de utilidades
U.Values = (U.c .^(1-eco_param.sigma_c) ./ (1 - eco_param.sigma_c)) + ...
    eco_param.gamma*((1 - U.l) .^(1-eco_param.sigma_l) ./(1-eco_param.sigma_l));

% U.Values = log(U.c) + log(U.l);
end % end of fucntion
```

```
%% 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_02
% 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
%%
[chain,state] = MarkovSimulation(mkv.TransitionMatrix, 1000, mkv.StateVector, 3);

% 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
%
% $a_{t+1} + c_t = (1+r)a_t + w_{t}z_{t}$
%
% $\sigma_c, \sigma_l > 0$
%
% $a_{t+1} \ge -\frac{wz}{r}$

%% Item C
% As firmas representativas resolvem o seguinte problema
%
% $max \left\{ K_t^\alpha N_t^{1-\alpha} -w_tN_t -r_tK_t\right\}$

%% 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:
%
```

```
% # Dado r e w, a função política g(a,z) resolve o problema do consumidor;
% # Dado r e w, a firma representativa maximiza os lucros;
%  $w = F_N(K;N)$ e $r + \delta = F_K(K;N)$
% # Markets clear:
% $K_0 = \sum_{z,a}\lambda(a,z)a$
% $N = \pi ' z$
% # 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:
%
% # Definir um valor para $r_j \in (-\delta, 1/\beta -1)$;
% # Determinar o capital-labor ratio $r = F_K(k) -\delta$
% # Determinar w:  $w = F_L(k)$;
% # Resolver o problema dos consumidores e determinar $a_{t+1}(a;z)$, $c_t(a;z)$, $l_t(a;z)$;
% # Calcular a distribuição estacionária $\lambda(a;z)$
% # Calcula a oferta de capital agregado e oferta de trabalho;
% # Avaliar o excesso de capital $D(r)=k - \frac{K}{L}$;
% # Se $D(r) > 0$, então $r_{j+1} > r_j$; se $D(r) < 0$, então $r_{j+1} < r_j$.
% # 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);
end

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

%% 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');
```

```
%% 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_02
% https://github.com/btebaldi/Macro3/tree/master/PSet_03>
%
%
%% Questao 2
%% Item F
% A arrecadação do governo pode ser escrita como:
%
% $G = \tau_y K_t^{\alpha}N_t^{1-\alpha} + \tau_n w_t N_t$
%
% Em termos per capta temos:
%
% $g = \tau_y k_t^{\alpha} + \tau_n w_t$
%
% Se a arrecadação é constante, pelo teorema da função implicita temos:
%
% $\frac{d\tau_y}{d\tau_n} = \frac{-w}{k^\alpha}$
%
% substituindo a equacoes de w temos:
%
% $\frac{d\tau_y}{d\tau_n} = -\frac{(1-\tau_y)(1-\alpha)}{(1+\tau_n)}$
%
% $\frac{d\tau_y}{d\tau_n} = -0.48$
%
% Utilizando o conceito de diferencial temos:
%
% $\Delta\tau_y = -0.48 \Delta\tau_n = (-0.48)*(-0.05) = 0.024$
%
% Logo é esperado que $\tau_y$ seja próximo de 0.024.

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

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;

eco_param.tau_n = 0.25;
eco_param.tau_y = 0;

% 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);

G0 = CalculaRendaGov(eco_param.tau_y ,Labor, WageShocks, eco_param, 1);

eco_param.tau_n = 0.20
f = @(tau_y)( G0 - CalculaRendaGov(tau_y, Labor, WageShocks, eco_param, 0));

sol = fsolve(f,0.024);

CalculaRendaGov(sol, Labor, WageShocks, eco_param, 1);
```

```
function out = AnaliseMarkovToAr(mkvStruct, eps)
% Autor: Bruno Tebaldi Q Barbosa
%
% Adaptação do codigo de Tiago Cavalcanti
% Calculate the invariant distribution of Markov chain by simulating the
% chain to reach a long-run level
%
% Imputs:
% PI: Matriz de transição do processo de markov
% z: Vetor de estados
%
%
%
% eps:
if nargin <2
   eps = 1e-8;
end


% Assume ua probabilidade igual para se iniciar em todos os estados.
prob = (1/mkvStruct.QtdStates)*ones(mkvStruct.QtdStates,1); % initial distribution of↙
states
test = 1;

% Calcula o estado final de equilibrio
while test > eps
    probst1 = mkvStruct.TransitionMatrix'*prob;
    test=max(abs(probst1-prob));
    prob = probst1;
end

% Calculate Properties of Invariant Distribution
meanm = mkvStruct.StateVector*prob;               % mean of invariant distribution
varm = ((mkvStruct.StateVector-meanm).^2)*prob;   % variance of invariant distribution

midaut1 = (mkvStruct.StateVector-meanm)'*(mkvStruct.StateVector-meanm); % cross product↙
of deviation from the
                                % mean of y_t and y_t-1

probmat = prob*ones(1,mkvStruct.QtdStates);     % each column is invariant distribution

midaut2 = mkvStruct.TransitionMatrix.*probmat.*midaut1; % product of the first two terms↙
is
                                % the joint distribution of (Y_t-1,Y_t)

autcov1 = sum(sum(midaut2));      % first-order auto-covariance

alambda = autcov1/varm;           % persistence of discrete process
asigmay = sqrt(varm);            % s.d. of discrete process
```

```
% Calculate the Asymptotic second moments of Markov chain

fprintf('_____________________________________________________\n')
fprintf('                    original process     Markov chain\n')
fprintf('_____________________________________________________\n')
fprintf('Persistence         %16.6f %16.6f\n', mkvStruct.AR.rho, alambda);
fprintf('Standard deviation  %16.6f %16.6f\n', mkvStruct.AR.sigma2_y^0.5, asigmay);
fprintf('_____________________________________________________\n')
end
```

```
function mkv = MarkovProcess(rho,sigma,r,N,mu)
% Autor: Bruno Tebaldi Q Barbosa
%
% Modela um processo AR(1) por um modelo de cadeia de markov.
%  y(t)= (1-rho)*mu + rho*y(t-1) + e(t)
%
% Imputs:
% rho: Fator de correlação do AR(1)
% sigma: Variancia dos erros no AR(1)
% r: Quantidade de desvios padores que os estados mais distantes do
%    processo de Markov deve ter
% N: Quantidade total de estados no processo de Markov.
% mu: Média de longo prazo do processo AR(1)


% Validade\Fill in unset optional values.
switch nargin
    case {0,1,2,3}
        error('função necessita pelo menos de 4 argumentos');
    case 4
        mu=0;
end

% Valida rho
if abs(rho) >= 1
    error('processo ar não estacionario. (rho = %f)', rho);
end

% Discretizacao do espaco de estado
dpy = sqrt(sigma^2/(1-rho^2));    % desvio padrao de y_t (não condicional)
z_N = mu + r*dpy;                 % limite superior
z_1 = mu - r*dpy;                 % limite inferior

d = (z_N-z_1)/(N-1);              % tamanho do intervalo
z = z_1:d:z_N;                    % grid de estados

% Inicia a matriz de tranzicao
PI = nan(N);

% Calculate the transition matrix - see Tauchen
for lin=1:N
    for col=2:N-1
        PI(lin,col)= normcdf(z(col)+d/2-rho*z(lin) -mu*(1-rho), 0, sigma)...
            - normcdf(z(col)-d/2-rho*z(lin) -mu*(1-rho), 0, sigma);
    end

    % Casos de j={1 ,N}
    PI(lin,1) = normcdf(z(1)+d/2-rho*z(lin)-mu*(1-rho),0,sigma);
    PI(lin,N) = 1 - normcdf(z(N)-d/2-rho*z(lin)-mu*(1-rho),0,sigma);
end
```

```
% Verifica se as probabilidade somam um por linha
if sum(PI,2) ~= ones(N,1)
    % find rows not adding up to one
    rowNotSumTo1 = find(sum(PI,2) < 1);  % find rows not adding up to one
    fprintf('Error in transition matrix\n');
    fprintf('row %d does not sum to one\n', rowNotSumTo1);
end

% Cria structure de resposta
mkv.AR.form = 'y(t)= (1-rho)*mu + rho*y(t-1) + e(t)';
mkv.AR.mu = mu;
mkv.AR.rho = rho;
mkv.AR.sigma2 = sigma;
mkv.AR.sigma2_y = dpy^2;
mkv.TransitionMatrix = PI;
mkv.StateVector = z;
mkv.QtdStates = N;
mkv.d = d;
mkv.StateBorder = z(1:end-1) + d/2;

end % end of function MarkovProcess
```

```
function [chain,state] = MarkovSimulation(PI, N, S, S0)
    % INPUTS:
    %   PI  : Transition matrix
    %   N   : length of simulation
    %   S   : State vector
    %   S0  : initial state (index)

    % ------------------------------------------------------------------------------
    % 1. validacoes de inputs

    % Verifica a quantidade de inputs minimo e maximo
    narginchk(2,4)

    [rPI, cPI] = size(PI);
    [rN, cN] = size(N);

    % Validade\Fill in unset optional values.
    switch nargin
        case 2
            S=1:rPI;
            S0=1;
        case 3
            S0=1;
    end

    [rS, cS] = size(S);

    % (a) Checking the total of shocks
    % Checks if N is a scalar or not
    if rN == 1 & cN == 1
        N = round(abs(N));
    else
        error('The first input must be a scalar');
    end

    % (b) checking the Transition matrix
    % Checks if it is a square matrix or not
    if (rPI == 1 | cPI == 1) | (rPI ~= cPI)
        error('The second input must be a square matrix.');
    end

    % changes the sign if some probabilities are negative
    PI = abs(PI);

    % checks if the probabilities sum to one, if not, it normalizes
    for i = 1:rPI
        if sum(PI(i,:)) ~= 1
            fprintf('probability: %f', sum(PI(i,:)));
            warning('The probabilities don`t sum to 1.');
            PI(i,:) = PI(i,:)/sum(PI(i,:));
        end
```

```
    end

    % (c) checking the State Vector
    if rS > 1
        error('State vector must be 1xN.')
    elseif ~(cS==cPI)
        error('Number of state does not match size of Transition matrix')
    end

    % -------------------------------------------------------------------------
    % 2. Creating the shock realizations

    % Cria matrix de somas acumuladas
    cum_PI = [zeros(rPI,1) cumsum(PI')'];

    % Cria o vetor de simulacao
    simulation = rand(N,1);
    state = nan(N,1);
    state(1) = S0;

    for i=2:N
        state(i)=find(((simulation(i) <= cum_PI(state(i-1), 2:cPI+1))&(simulation(i)↙
>cum_PI(state(i-1),1:cPI))));
    end

    chain=S(state);

end %end of function
```