Lista 4
Macroeconomia I

Yuri Passuelo – yuripassuelo@usp.br

20 de maio de 2025

Resolução dos exercícios da quarta lista da disciplinas de Macroeconomia I.

Problem 1

Considere um consumidor de vida infinita que tem uma renda constante, y, a cada período. Este consumidor encara o seguinte problema.

$$\max_{\{c_t, a_{t+1}\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t u(c_t)$$

$$\text{s.a } c_t + a_{t+1} \leq y + (1 + r_t)a_t \ \forall t, \ a_0 \text{ dado}$$

em que para cada período t, a_t representa a riqueza do consumidor, c_t é o consumo, r_t é a taxa de juros e $\beta \in (0, 1)$. Suponha que o consumidor conhece $\{r_t\}_{t=0}^{\infty}$ e que $u(c) = \frac{c^{1-\sigma}}{1-\sigma}$ com $\sigma > 0$ e $\sigma \neq 1$.

(a) Escreva o problema recursivo do consumidor.

(b) Encontre a equação de Euler associada ao problema do consumidor.

(c) Mostre que se $\beta = (1 + r_{t+1})^{-1}$, então $c_{t+1} = c_t$.

(d) O que ocorre com o consumo se $\beta > (1 + r_{t+1})^{-1}$? e se $\beta < (1 + r_{t+1})^{-1}$?

(e) Escreva os passos de um algoritmo para computar a solução do problema recursivo do consumidor.

Solution

(a)

Para montar o problema recursivo do consumidor vamos primeiro usar a função utilidade em seu formato genérico $u(c_t)$, note também que nesse caso estamos falando de uma situação aonde não temos produção e toda renda advém de uma dotação de renda dada por y que é recebida em cada período mais o estoque de riqueza que rende a uma taxa de juros r, portanto é necessário fazer algumas simples adaptações.

$$v(a) = \max_{0 \leq a' \leq y+(1+r)a} \{u(y + (1+r)a - a') + \beta[v(a')]\}$$

(b)

Para encontrar a equação de Euler associada vamos diferenciar nossa equação em relação a a'

$$[a'] : \frac{\partial v(a)}{\partial a'} = -u'(y + (1+r)a - a') + \beta[\frac{\partial v(a')}{\partial a'}]$$

Usando o teorema de Benveniste Sheickman conseguimos diferenciar a função valor de modo que

$$[a] : \frac{\partial v(a)}{\partial a} = (1+r)u(y + (1+r)a - a') + \beta[\frac{\partial v(a')}{\partial a}]$$

Portanto substituindo a' por a e a' por a''

$$[a'] : \frac{\partial v(a')}{\partial a'} = (1+r)u'(y + (1+r')a' - a'')$$

Substituindo no itém anterior temos:

$$[a'] : \frac{\partial v(a)}{\partial a'} = -u'(y + (1+r)a - a') + \beta(1+r)u'(y + (1+r')a' - a'') = 0$$

Logo:

$$\beta(1+r)u'(y + (1+r')a' - a'') = u'(y + (1+r)a - a')$$

De forma mais simplificada temos:

$$\beta(1+r)u'(c') = u'(c)$$

A interpretação é que a utilidade marginal do consumo em $t+1$ tem que ser igual à utilidade marginal do consumo em t, descontada do fator de impaciência β e remunerada pela taxa de juros r.

(c)

Se tivermos que $\beta = (1+r_t)^{-1}$ então:

$$\beta(1+r)u'(c') = u'(c) \implies \frac{1+r}{1+r}u'(c') = u'(c)$$

Logo

$$u'(c') = u'(c)$$

Como temos que $u(c) = \frac{c^{1-\sigma}}{1-\sigma}$ então $u'(c) = c^{-\sigma}$ portanto:

$$c'^{-\sigma} = c^{-\sigma}$$

Portanto:

$$c' = c$$

Logo consumo é constante e portanto $c_{t+1} = c_t$

(d)

O que ocorre agora se $\beta < (1+r_{t+1})^{-1}$ ou $\beta > (1+r_{t+1})^{-1}$. Vamos portanto voltar a nossa equação de Euler:

$$\beta(1+r)u'(c') = u'(c)$$

Vamos primeiro assumir que $\beta < (1+r_{t+1})^{-1}$, se multiplicarmos a desigualdade por $(1+r_{t+1})$ teremos:

$$(1+r_{t+1})\beta < 1$$

Logo voltando a equação de Euler:

$$u'(c') < u'(c)$$

Substituindo a utilidade marginal temos então que:

$$c' < c$$

Usando a mesma lógica para $\beta > (1+r_{t+1})^{-1}$ temos:

$$c' > c$$

Portanto quando $\beta < (1+r_{t+1})^{-1}$ consumo é decrescente, e quando ocorre o inverso, consumo é crescente no tempo.

(e)

Vamos agora aos passos de um algoritmo que computa a solução para o problema recursivo do consumidor:

1. Construir o *Grid* de ativos A, de modo que temos:

$$A = \{a_1, ..., a_n\}, \text{ aonde } a_i < a_j \forall j > i$$

2. Parametrizar nossa economia, aqui precisamos parametrizar nossa economia, escolhendo parâmetros como: função utilidade, β, σ, y, r.

3. Chutes iniciais criando vetores de chutes iniciais para função valor e politica:

$$\text{v\_ini } = \{0, ..., 0\}$$

4. Tolerância e vetores de iteração:

$$\text{V\_it } = \{0, ..., 0\}$$

$$\text{g\_it } = \{0, ..., 0\}$$

$$\text{tol } = 10^{-7}$$

5. Iteração da função valor:

   Para cada $a \in A$,

   - Criamos o vetor

$$\text{v\_it } = \{0, ..., 0\}$$

   Para cada $a' \in A$, computamos:

$$u(y + (1 + r)a - a') + \beta[v(a')]$$

   Aqui devemos nos atentar que $y + (1 + r)a - a'$ não pode ser negativo, caso o termo ser negativo. Guardamos o valor relativo a computação em v_it, caso $y + (1 + r)a - a'$ seja negativo guardamos um valor muito pequeno, por exemplo -999.

   Para cada valor computado guardamos em g_it(a) a' que maximiza função valor.

   Para cada a guardamos o valor máximo de v_it em V_it(a).

6. Ao fim da iteração checamos para a convergência da função valor de modo que queremos que:

$$||\text{V\_it} - \text{v\_ini}|| < \text{tol}$$

Caso a condição seja atendida, paramos o processo, caso contrário:

$$\text{v\_ini} \;= \text{V\_it}$$

E continuamos a iteração.

Exemplo abaixo Faz o passo a passo no Python:

```
# Import libraries
import numpy as np

# 1. Grid de Ativos
A = np.linspace( 0, 5, 100 )

# 2. Parametrizando a economia
# Funcao Utilidade no Formato CRRA
def uti( c, sigma ):
    return (c**(1-sigma))/(1-sigma)
# Parametros Iniciais
y     = 5
beta  = 0.95
sigma = 1.5
r     = 0.05

# 3. Chutes Iniciais
v_ini = [ 0 for i in range(0,len(A))]#np.zeros( len(A) )

# 4. Toler ncia e chute inicial
#np.zeros( len(A) )
g_it  = [ 0 for i in range(0,len(A))]
tol = 1e-5

# 5. Itera  o da fun  o valor
dist = 1000
n_it = 0
# Iteracao
while ( dist > tol )  :
    V_it  = [ 0 for i in range(0,len(A))]
    # Iteracao
    for i in range(0,len(A)):
        v_it  = np.zeros( len(A) )
        for j in range(0,len(A)):
            # Checa se Consumo   Negativo
            cons = y + (1+r)*A[i] - A[j]
```

```
            if cons < 0:
                v_it[j] = -999
            if cons > 0:
                v_it[j] = uti( cons, sigma ) + beta*v_ini[j]
        # Guarda valores m ximos
        V_it[i] = np.max( v_it )
        g_it[i] = A[np.argmax( v_it )]
        #print( V_it, v_ini )

    # Calcula Distancia
    dist = np.sqrt( np.sum( np.square( np.array(V_it) - np.array(v_ini) )
                                        ) )
    print( dist, n_it )
    # Atualiza Chute
    v_ini = V_it
    # Soma Iteracao
    n_it = n_it + 1
    if n_it > 1000:
        break
```

Aqui notamos basicamente que a melhor escolha diante da maximização da utilidade intertemporal é de consumir toda a renda, uma vez que a renda é certa, o comportamento que maximiza utilidade é consumir toda a renda disponível inclusive todos os ativos no período zero, abaixo temos alguns gráficos que exemplificam.

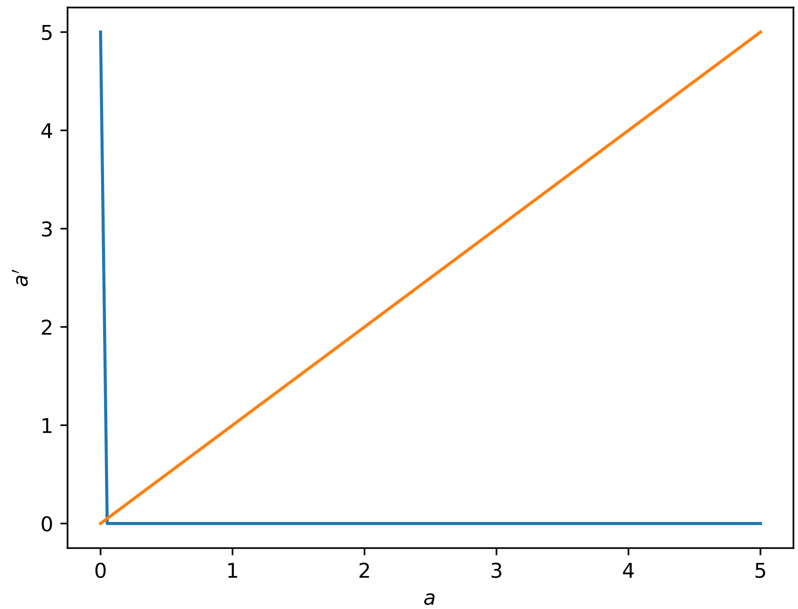

Figura 1: Função Valor para níveis de ativo a'

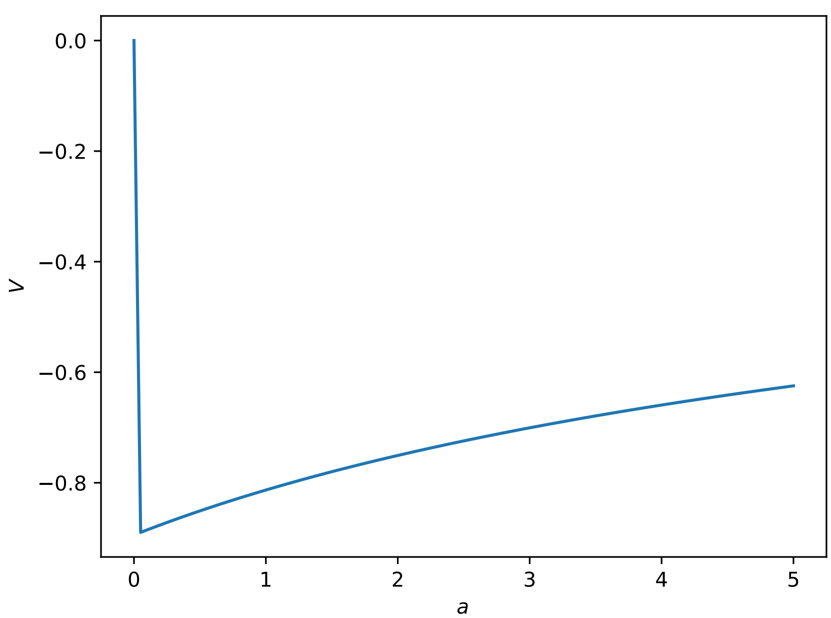

Figura 2: Gráfico da função politica de a'

Problem 2

Considere uma ilha (economia) com uma única arvore de Lucas.

- Os frutos (chamados de dividendos pelos habitantes da ilha) que crescem na arvore são a única fonte de consumo.
- Esses frutos seguem o seguinte processo estocástico:
  - $d_{t+1} = \gamma d_t$ com probabilidade π ou $d_{t+1} = d_t$ (com probabilidade $(1 - \pi)$).
  - se em um dado per ıodo T temos que $d_T = d_{T-1}$ então para todo $t > T$ vale que $d_t = d_T$.

Existe uma massa unitária de pessoas iguais nessa ilha com preferências sobre um fluxo de consumo dadas por:

$$U = \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t u(c_t),$$

em que $u(c) = \frac{c^{1-\sigma}}{1-\sigma}$. Suponha que $\sigma > 0$, $\sigma \neq 1$, $0 < \beta < 1$, $\gamma > 1$ e que $\beta\gamma^{(1-\sigma)} < 1$.

(a) Escreva o problema recursivo de um agente representativo.

(b) Defina um equilíbrio competitivo recursivo para essa economia.

(c) Encontre a condição de primeira ordem do problema do agente representativo e, utilizando a condição de equilíbrio, escreva a equação a apreçamento do ativo.

(d) Escreva os preços de equilíbrio como uma função dos dividendos.

(e) Seja T o primeiro per ıodo tal que $d_{T-1} = d_T$. Vale que $p_{T-1} > p_T$? Encontre condições para que isto seja verdade. Interprete os resultados.

Solution

(a)

Aqui precisamos adaptar o que já conhecemos e adaptar para o caso de...

Relembrando alguns conceitos:

- Arvore denominada s_t, que gera frutos, que podem ser denominados x_t ou d_t;
- Arvore é particionada entre os indivíduos, e tem preço p_t.
- Variáveis de estado e controle são:
  - Controle: c_t, s_{t+1};

- Estado: s_t e d_t ou x_t

- Preço da arvore p é função do valor dos dividendos d

A restrição orçamentária do problema é dada por:

$$p_t s_{t+1} + c_t = (p_t + d_t)s_t$$

Do lado esquerdo da equação temos os gastos e receitas ou as posses de ativos, ou seja, no presente o individuo possui como ativos, a arvore precificada em p_t e os frutos/dividendos d_t gerados, e sua alocação se dá na decisão entre consumir c_t mas ao mesmo tempo se preocupar com o fato de comprar a arvore para o próximo período $(t + 1)$.

Montando o problema de forma resumida temos:

$$v(d, s) = \max_{\substack{c, s' > 0 \\ c \leq (p+d)s - ps'}} \{u(c) + \beta \mathbb{E}[v(d', s')]\}$$

Podemos substituir o consumo por $(p(d) + d)s - p(d)s'$

$$v(d, s) = \max_{\substack{c, s' > 0 \\ c \leq (p+d)s - ps'}} \{u((p(d) + d)s - p(d)s') + \beta \mathbb{E}[v(d', s')]\}$$

Por último como a função utilidade é dada temos:

$$v(d, s) = \max_{\substack{c, s' > 0 \\ c \leq (p+d)s - ps'}} \left\{ \frac{((p(d) + d)s - p(d)s')^{1-\sigma}}{1 - \sigma} + \beta \mathbb{E}[v(d', s')] \right\}$$

(b)

Para a definição de um equilíbrio competitivo recursivo vamos usar Junior (2025) Modelos com Incerteza Parte 2 *pg.2.* Um equilíbrio competitivo recursivo é uma coleção de funções $\{V, g, p\}$ (V a função valor, g a função politica e p a função preço) tal que:

1. Dados r e w, nossa coleção de funções $\{V, g, p\}$resolve o problema do consumidor.

2. Equilíbrio de Mercado, aonde $g(s, d) = 1$ para todo par (s, d) de modo que $s = s' = 1$

$$c + p(d) = p(d) + d$$

Portanto

$$c = d$$

Ou seja, consumo será igual os frutos (dividendos) gerados pela arvore.

(c)

Vamos a condição de primeira ordem relacionada ao problema do agente representativo, como temos duas variáveis de controle que no nosso problema modificado é s'

$$[s'] : \frac{\partial v(d,s)}{\partial s'} = -p(d)((p(d)+d)s - ps')^{-\sigma} + \beta \frac{\partial \mathbb{E}[v(d',s')]}{\partial s'}$$

$$= -p(d)((p(d)+d)s - p(d)s')^{-\sigma} + \beta \mathbb{E}[\frac{\partial v(d',s')}{\partial s'}]$$

Aqui vamos usar o teorema de Benveniste Scheikman para diferenciar nossa função valor em relação à s', primeiro passo é usar a função valor em t e diferenciar em relação a s_t.

$$\frac{\partial v(d,s)}{\partial s} = (p(d)+d)((p(d)+d)s - p(d)s')^{-\sigma}$$

Substituindo s por s' e d por d' temos:

$$\frac{\partial v(d',s')}{\partial s'} = (p(d')+d'')((p(d')+d'')s' - p(d')s'')^{-\sigma}$$

Logo:

$$[s'] : \frac{\partial v(d,s)}{\partial s'} = -p((p+d)s - ps')^{-\sigma} + \beta \mathbb{E}[(p(d')+d'')((p(d')+d'')s' - p(d')s'')^{-\sigma}] = 0$$

(c)

Vamos escrever os preços em função dos dividendos, usando a C.P.O

$$-p(d)((p(d)+d)s - p(d)s')^{-\sigma} + \beta \mathbb{E}[(p(d')+d')((p(d')+d'')s' - p(d')s'')^{-\sigma}] = 0$$

$$p(d)((p(d)+d)s - p(d)s')^{-\sigma} = \beta \mathbb{E}[(p(d')+d')((p(d')+d'')s' - p(d')s'')^{-\sigma}]$$

$$p(d) = E[\beta \frac{(p(d')+d'')((p(d')+d')s' - p(d')s'')^{-\sigma}}{((p(d)+d)s - p(d)s')^{-\sigma}}]$$

Vamos colocar nossa equação de Euler com os índices temporais para facilitar as contas.

$$p_t = \beta E[(p_{t+1}+d_{t+1}) \frac{((p_{t+1}+d_{t+1})s_{t+1} - p_{t+1}s_{t+2})^{-\sigma}}{((p_t+d_t)s_t - p_t s_{t+1})^{-\sigma}}]$$

Temos que em equilíbrio $c_t = d_t$ logo:

$$p_t = \beta \mathbb{E}[(p_{t+1}+d_{t+1})(\frac{d_{t+1}}{d_t})^{-\sigma}]$$

Abrindo a esperança:

$$p_t = \beta\mathbb{E}[d_{t+1}(\frac{d_{t+1}}{d_t})^{-\sigma}] + \beta\mathbb{E}[p_{t+1}(\frac{d_{t+1}}{d_t})^{-\sigma}]$$

Aqui vamos usar o formato recursivo da equação de modo que usemos p_{t+j+1} em p_{t+j}, fazendo isso T vezes, temos um somatório na primeira parte da soma e na segunda acabamos ter uma expressão multiplicada $T-1$ vezes, dessa forma obtemos:

Obs: De exemplo vamos substiruir p_{t+1} em p_t, assim temos que p_{t+1} é:

$$p_{t+1} = \beta\mathbb{E}[d_{t+2}(\frac{d_{t+2}}{d_{t+1}})^{-\sigma}] + \beta\mathbb{E}[p_{t+2}(\frac{d_{t+2}}{d_{t+1}})^{-\sigma}]$$

Substituindo em p_t

$$p_t = \beta\mathbb{E}[d_{t+1}(\frac{d_{t+1}}{d_t})^{-\sigma}] + \beta\mathbb{E}[[\beta\mathbb{E}[d_{t+2}(\frac{d_{t+2}}{d_{t+1}})^{-\sigma}] + \beta\mathbb{E}[p_{t+2}(\frac{d_{t+2}}{d_{t+1}})^{-\sigma}]](\frac{d_{t+1}}{d_t})^{-\sigma}]$$

Expandindo a segunda parte

$$p_t = \beta\mathbb{E}[d_{t+1}(\frac{d_{t+1}}{d_t})^{-\sigma}] + \beta^2\mathbb{E}[d_{t+2}(\frac{d_{t+2}}{d_t})^{-\sigma}] + \beta^2\mathbb{E}[p_{t+2}(\frac{d_{t+2}}{d_t})^{-\sigma}]$$

Se contiuarmos então teremos:

$$p_t = \mathbb{E}\left\{\sum_{j=1}^{T}\beta^j(\frac{d_{t+1}}{d_t})^{-\sigma}d_{t+j}\right\} + \mathbb{E}\left\{\beta^T p_{t+T}(\frac{d_{t+T}}{d_t})^{-\sigma}\right\}$$

Aplicando o limite de $T \to \infty$ temos que o elemento β que se situa entre 0 e 1 tende à zero e portanto ficamos apenas com a expressão:

$$\lim_{T\to\infty} p_t = \mathbb{E}\left\{\sum_{j=1}^{\infty}\beta^j(\frac{d_{t+j}}{d_t})^{-\sigma}d_{t+j}\right\}$$

Logo usando a formulação recursiva apresentada acima temos:

$$p_t = \mathbb{E}\left\{\sum_{j=1}^{\infty}\beta^j(\frac{d_{t+j}}{d_t})^{-\sigma}d_{t+j}\right\}$$

Como descrito no enunciado temos duas situações possíveis, a primeira em que o valor do dividendo cresce ou seja $d_{t+1} = \gamma d_t$ com $\gamma > 1$, isso ocorre co probabilidade pi, e temos outra em que o valor do dividendo permanece constante ou seja $d_t = d_{t+1}$ com probabilidade $1-\pi$, caso isso ocorra o valor do dividendo permanece constante para sempre. Vamos começar pela segunda situação aonde primeiramente vamos voltar a equação de precificação:

$$p_t = \mathbb{E}\left\{\sum_{j=1}^{\infty}\beta^j(\frac{d_{t+j}}{d_t})^{-\sigma}d_{t+j}\right\}$$

Como $d_t = d_{t+1}$ então:

$$p_t = \sum_{j=1}^{\infty} \beta^j d_{t+j} = \frac{\beta}{1-\beta} d_t$$

Vamos agora a primeira situação aonde $d_{t+1} = \gamma d_t$:

$$p_t = \sum_{j=1}^{\infty} \beta^j (\frac{\gamma d_{t+j}}{d_t})^{-\sigma} d_{t+j}$$

Aqui note que estamos na situação que os dividendo evoluem de forma $\gamma d_t = d_{t+1}$, então:

$$d_{t+j} = \gamma d_{t+j-1} = \gamma^2 d_{t+j-2} = \gamma^3 d_{t+j-3} = ... = \gamma^j d_t$$

Substituindo na nossa função de preço teremos:

$$p_t = \sum_{j=1}^{\infty} \beta^j (\frac{\gamma^j d_t}{d_t})^{-\sigma} \gamma^j d_t$$

$$p_t = \sum_{j=1}^{\infty} \beta^j \gamma^{j(1-\sigma)} d_t = \frac{\beta\gamma^{1-\sigma}}{1-\beta\gamma^{1-\sigma}} d_t$$

Como a precificação parte de uma esperança em dois cenários o que temos que fazer é pondera-los:

$$p_t = \pi \frac{\beta}{1-\beta} d_t + (1-\pi) \frac{\beta\gamma^{1-\sigma}}{1-\beta\gamma^{1-\sigma}} d_t$$

$$p_t = d_t \Big(\pi \frac{\beta}{1-\beta} + (1-\pi) \frac{\beta\gamma^{1-\sigma}}{1-\beta\gamma^{1-\sigma}} \Big)$$

(e)

Quais condições para que $p_{T-1} > p_T$, sendo T o momento em que d_T passa a ser igual a d_{T-1}? Para isso primeiro vamos organizar alguns fatos.

- No momento $T-1$ temos que o valor do dividendo está crescendo e portanto sua precificação se da de maneira estocástica ou seja:

$$p_t = d_t \Big(\pi \frac{\beta}{1-\beta} + (1-\pi) \frac{\beta\gamma^{1-\sigma}}{1-\beta\gamma^{1-\sigma}} \Big)$$

- A partir do momento em que $d_{T-1} = d_T$ temos que o preço passa a ser constante e não tem variação de maneira estocástica como anteriormente, uma vez que nos é dito no enunciado que a partir desse momento se valor será constante.

$$p_t = \frac{\beta}{1-\beta} d_t$$

Portanto temos que:

$$p_{T-1} > p_T$$

$$d_{T-1}\Big(\pi \frac{\beta}{1-\beta} + (1-\pi)\frac{\beta\gamma^{1-\sigma}}{1-\beta\gamma^{1-\sigma}}\Big) > \frac{\beta}{1-\beta} d_T$$

Anulando d_{T+1} e d_T já que são iguais

$$\pi \frac{\beta}{1-\beta} + (1-\pi)\frac{\beta\gamma^{1-\sigma}}{1-\beta\gamma^{1-\sigma}} > \frac{\beta}{1-\beta}$$

Passando $\pi\frac{\beta}{1-\beta}$ para o lado direito

$$(1-\pi)\frac{\beta\gamma^{1-\sigma}}{1-\beta\gamma^{1-\sigma}} > (1-\pi)\frac{\beta}{1-\beta}$$

Cancelando $1-\pi$

$$\frac{\beta\gamma^{1-\sigma}}{1-\beta\gamma^{1-\sigma}} > \frac{\beta}{1-\beta}$$

Cancelando β

$$\frac{\gamma^{1-\sigma}}{1-\beta\gamma^{1-\sigma}} > \frac{1}{1-\beta}$$

$$\gamma^{1-\sigma}(1-\beta) > 1 - \beta\gamma^{1-\sigma}$$

$$\gamma^{1-\sigma} - \beta\gamma^{1-\sigma} > 1 - \beta\gamma^{1-\sigma}$$

$$\gamma^{1-\sigma} > 1$$

$$\gamma > 1$$

Logo a condição necessária é que $\gamma > 1$. Lógica é bem simples, dado que até T dividendo tinha o processo em que seu valor crescia em proporção γ a cada período, e depois vê seu valor ficar constante, então para que o preço desse dividendo até $T-1$ seja maior que em T é necessário que o dividendo estivesse crescendo, uma vez que se $\gamma = 1$ então o valor do dividendo continuaria andando de lado e portanto $p_T = p_{T-1}$, e caso $\gamma < 1$ o preço seria menor.

Problem 3

Seja a matriz estocástica

$$M = \begin{pmatrix} 0.20 & 0.30 & 0.40 & 0.10 \\ 0.10 & 0.10 & 0.70 & 0.10 \\ 0.95 & 0.024 & 0.025 & 0.001 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{pmatrix}$$

(a) Defina a distância entre duas matrizes como a soma do quadrado das diferenças entre cada entrada das matrizes. No Python, defina uma distribuição inicial

$$P_0 = \begin{pmatrix} 0.25 & 0.25 & 0.25 & 0.25 \end{pmatrix}$$

Ache o k, tal que a distancia entre P_{k+1} e P_k seja menor do que 10^{-7}

(b) Reporte a distribuição invariante de M. Há mais do que uma?

Solution

```
import numpy as np

# Exercic o 3

# Matriz do Processo de Markov
M = np.matrix( [[0.20, 0.30, 0.40, 0.10],
                [0.10, 0.10, 0.70, 0.10],
                [0.95, 0.024, 0.025, 0.001],
                [0.25, 0.25, 0.25, 0.25]])

# Distribui  o Inicial
p0 = np.array([ 0.25, 0.25, 0.25, 0.25 ])

# Toler ncia
tol = 10e-7

# Iteracao
p  = p0
q  = p0*M
k = 1

dist = np.sum( np.square(p-q) )

while dist > tol:
    # Atualiza vetores
    p = q
    q = q * M
```

```python
    # Calcula distancia
    dist = np.sum( np.square(p-q))
    k = k + 1

print(k)
```

Pelo procedimento adotado acima chegaremos em um valor de $k = 7$ e a distribuição invariante de:

$$\begin{pmatrix} 0.42690625 & 0.17294405 & 0.31978859 & 0.08036111 \end{pmatrix}$$

A distribuição invariante é única.

Problem 4

Suponha que o planejador deseja maximizar

$$\mathbb{E}\left\{\sum_{t=0}^{\infty}\beta^t u(c_t,l_t)\right\},\ 0<\beta<1$$

sujeito a restrição de recursos

$$c_t + k_{t+1} = z_t f(k_t,l_t) + (1-\delta)k_t,\ 0<\delta<1$$

com condições iniciais $k_0 > 0$ e $z_0 > 0$. A produtividade z_t evolui de acordo com um processo de Markov com probabilidade de transição $F(z'|z) = Prob[zt+1z'|zt = z]$ e média incondicional $\bar{z} > 0$. Neste problema o planejador escolhe quanto trabalho l_t ofertar. Assuma que $u(c_t,l_t)$ é estritamente crescente e estritamente côncava em c_t, e e estritamente decrescente e estritamente convexa em l_t. A função de produção $f(k_t,l_t)$ e estritamente crescente e estritamente côncava em ambos argumentos e tem retornos constantes de escala.

(a) Seja $v(k,z)$ a função valor do planejador. Escreve e explique a equação de Bellman que determina $v(k,z)$.

(b) Derive as condições de otimalidade do problema do planejador.

(c) Suponha que

$$u(c,l) = \log c - \frac{l^{1+\phi}}{1+\phi},\ \phi > 0$$

e,

$$F(k,l) = k^\alpha l^{1-\alpha},\ 0<\alpha<1$$

Encontre os valores de *steady state* não estocástico de consumo, capital e trabalho em termos dos parâmetros do modelo. Suponha que existe um aumento permanente no nível de produtividade $\bar{z}$. Explique como isto muda os valores de estado estacionário do consumo, capital e trabalho. Dê uma intuição econômica para seus resultados.

(d) Suponha que os poss ıveis valores de z_t são $\mathcal{Z} = \{0.8, 1, 1.2\}$ e que a matriz de transição é dada por

$$\pi = \begin{pmatrix} 0.2 & 0.5 & 0.3 \\ 0.1 & 0.6 & 0.3 \\ 0.25 & 0.25 & 0.5 \end{pmatrix}$$

Sejam $\alpha = 0.3$, $\beta = 1/1.05$, $\delta = 0.05$, $\phi = 1$. Escreva um código em Python para calcular a função valor e funções políticas. Use um *grid* para o capital entre 0 e 100[1].

[1] 1Usar o zero para capital n ao e uma boa ideia.

Solution

(a)

O problema acima se trata de uma versão mais complexa dos modelos neoclássicos até então vistos, note que temos duas "complexidades"a mais no modelo

- A primeira se trata da inclusão do trabalho, aqui temos a inclusão de uma variável a mais de controle, agora o individuo escolhe o quanto de k' e l, ou seja, quando de capital e trabalho serão alocados.
- A segunda se trata da inclusão de um componente estocástico, ou seja, nossa função de produção $f(k_t, l_t)$ sofre choques dados por z_t que sofrem um processo markoviano.

Vamos então a construção do problema recursivo:

$$v(k, l) = \max_{k' \leq zF(k,l)+(1-\delta)k} \{u(zF(k, l) + (1 - \delta)k - k', 1 - l) + \beta\mathbb{E}_z[v(k', l')]\}$$

Aonde temos que:

- Variáveis de Estado: k e z.
- Variáveis de Controle: c, k' e l.

(b)

Vamos agora as condições de otimalidade do problema, vamos relembra-las:

1. Condição de Primeira ordem relacionada ao capital k'
2. Condição de Primeira ordem relacionada ao trabalho l
3.

Vamos as C.P.Os:

$$[k'] : \frac{\partial v(k, l)}{\partial k'} = -u_k(zF(k, l) + (1 - \delta)k - k', 1 - l) + \beta\frac{\partial\mathbb{E}_z[v(k', l')]}{\partial k'}$$

Abrindo a esperança:

$$\frac{\partial\mathbb{E}_z[v(k', l')]}{\partial k'} = \mathbb{E}_z[\frac{\partial v(k', l')}{\partial k'}]$$

Para a diferenciação da função valor, vamos usar o teorema de Benveniste Scheikman:

$$\frac{\partial v(k, l)}{\partial k} = u_k(zF(k, 1 - l) + (1 - \delta)k - k', l)[zF_k(k, l) + (1 - \delta)]$$

Substituindo k por k', z por z' e l por l':

$$\frac{\partial v(k,l)}{\partial k} = u_k(z'F(k',1-l') + (1-\delta)k' - k'', l)[zF_k(k',l') + (1-\delta)]$$

Vamos substituir na C.P.O de k'.

$$[k'] : \frac{\partial v(k,l)}{\partial k'} = -u_k(zF(k,l)+(1-\delta)k-k', 1-l)+\beta\mathbb{E}_z[u_k(z'F(k',l')+(1-\delta)k'-k'', 1-l')[z'F_k(k',l')+(1-\delta)]]$$

Para simplificar os termos, vamos substituir $zF(k,l) + (1-\delta)k - k'$ por c e $z'F(k',l') + (1-\delta)k' - k''$ por c', logo:

$$[k'] : \frac{\partial v(k,l)}{\partial k'} = -u_c(c, 1-l) + \beta\mathbb{E}_z[u_c(c', 1-l)[z'F_k(k',l') + (1-\delta)]] = 0$$

$$\beta\mathbb{E}_z[u_c(c', 1-l')[z'F_k(k',l') + (1-\delta)]] = u_c(c, 1-l)$$

Vamos agora a C.P.O em relação ao trabalho:

$$[l] : \frac{\partial v(k,l)}{\partial l} = u_c(c, 1-l)[zF_l(k,l)] - u_l(c, 1-l)$$

Por ultimo temos:

(c)

Agora temos funções tanto de produção quanto utilidade para nosso problema, ou seja, temos:

$$u(c,l) = \log c - \frac{l^{1+\phi}}{1+\phi}$$

$$F(k,l) = k^{\alpha}l^{1-\alpha}$$

Vamos substituir as funções na nossa forma recursiva:

$$v(k,l) = \max_{k' \leq zk^{\alpha}l^{1-\alpha}+(1-\delta)k} \left\{ \log\,(zk^{\alpha}l^{1-\alpha} + (1-\delta)k - k') - \frac{l^{1-\phi}}{1-\phi} + \beta\mathbb{E}_z[v(k',l')] \right\}$$

Vamos às C.P.O's:

$$[k'] : -\frac{1}{zk^{\alpha}l^{1-\alpha} + (1-\delta)k - k'} + \beta\mathbb{E}_z[\frac{\partial v(k',l')}{\partial k'}]$$

Usando o teorema de Benveniste Sheickman para diferenciar a função valor temos:

$$\frac{\partial v(k,l)}{\partial k} = \frac{\alpha z(\frac{l}{k})^{1-\alpha} + (1-\delta)}{zk^{\alpha}l^{1-\alpha} + (1-\delta)k - k'}$$

Substituindo k por k', z por z' e l por l' temos:

$$\frac{\partial v(k',l')}{\partial k'} = \frac{\alpha z'(\frac{l'}{k'})^{1-\alpha} + (1-\delta)}{z'k'^{\alpha}l^{1-\alpha} + (1-\delta)k' - k''}$$

Substituindo a esperança da C.P.O em relação à k' então temos:

$$[k'] : -\frac{1}{zk^{\alpha}l^{1-\alpha} + (1-\delta)k - k'} + \beta\mathbb{E}_z[\frac{\alpha z'(\frac{l'}{k'})^{1-\alpha} + (1-\delta)}{z'k'^{\alpha}l^{1-\alpha} + (1-\delta)k' - k''}] = 0$$

Logo nossa CPO será:

$$\beta\mathbb{E}_z[\frac{\alpha z'(\frac{l'}{k'})^{1-\alpha} + (1-\delta)}{z'k'^{\alpha}l^{1-\alpha} + (1-\delta)k' - k''}] = \frac{1}{zk^{\alpha}l^{1-\alpha} + (1-\delta)k - k'}$$

Vamos a C.P.O em relação ao trabalho:

$$[l] : \frac{\partial v(k,l)}{\partial l} = \frac{(1-\alpha)z(\frac{k}{l})^{\alpha}}{zk^{\alpha}l^{1-\alpha} + (1-\delta)k - k'} - l^{-\phi} = 0$$

Logo:

$$l^{\phi} = \frac{(1-\alpha)z(\frac{k}{l})^{\alpha}}{zk^{\alpha}l^{1-\alpha} + (1-\delta)k - k'}$$

Calculadas as condições de primeira ordem, vamos ao problema de *steady state*. Para essa etapa precisamos assumir que k, l estão em valores de *steady state* e que z tem um valor constante $\bar{z}$

Usando a C.P.O do trabalho:

$$l^{\phi} = \frac{(1-\alpha)z(\frac{k}{l})^{\alpha}}{zk^{\alpha}l^{1-\alpha} + (1-\delta)k - k'}$$

a parte de baixo da fração do lado direito pode ser colocada simplesmente como c^*, logo:

$$l^{*\phi} = (1-\alpha)\bar{z}(\frac{k^*}{l^*})^{\alpha}\frac{1}{c^*}$$

podemos substituir $\bar{z}(\frac{k}{l})^{\alpha}$ por $\frac{y}{l}$ logo

$$l^{*\phi} = (1-\alpha)\frac{y^*}{l^*}\frac{1}{c^*}$$

$$l^{*1+\phi} = (1-\alpha)\frac{y^*}{c^*}$$

$$l^* = ((1-\alpha)\frac{y^*}{c^*})^{\frac{1}{1+\phi}}$$

Precisamos portanto encontrar y^* e c^* para encontrar a trajetória ótima de l^*, vamos usar agora o resultado da C.P.O em relação à k', adotaremos a mesma estratégia e substituiremos os denominadores das frações por c^*:

$$\beta\frac{\alpha\bar{z}(\frac{l^*}{k^*})^{1-\alpha} + (1-\delta)}{c^*} = \frac{1}{c^*}$$

$$\beta(\alpha\bar{z}(\frac{l^*}{k^*})^{1-\alpha} + (1-\delta)) = 1$$

$$\alpha\bar{z}(\frac{l^*}{k^*})^{1-\alpha} + (1-\delta) = \frac{1}{\beta}$$

$$\alpha\bar{z}(\frac{l^*}{k^*})^{1-\alpha} = \frac{1}{\beta} - (1-\delta)$$

$$\alpha\bar{z}(\frac{l^*}{k^*})^{1-\alpha} = \frac{1-\beta(1-\delta)}{\beta}$$

$$(\frac{l^*}{k^*})^{1-\alpha} = \frac{1-\beta(1-\delta)}{\beta\alpha\bar{z}}$$

$$\frac{l^*}{k^*} = (\frac{1-\beta(1-\delta)}{\beta\alpha\bar{z}})^{\frac{1}{1-\alpha}}$$

$$\frac{k^*}{l^*} = (\frac{\beta\alpha\bar{z}}{1-\beta(1-\delta)})^{\frac{1}{1-\alpha}}$$

Achamos a razão Capital trabalho, partindo da definição de y

$$y^* = \bar{z}k^{*\alpha}l^{*1-\alpha}$$

$$y^* = \bar{z}(\frac{k^*}{l^*})^{\alpha}l^*$$

$$\frac{y^*}{l^*} = \bar{z}(\frac{k^*}{l^*})^{\alpha}$$

Substituindo $\frac{k^*}{l^*}$ temos:

$$\frac{y^*}{l^*} = \bar{z}(\frac{\beta\alpha\bar{z}}{1-\beta(1-\delta)})^{\frac{\alpha}{1-\alpha}}$$

De $\frac{k^*}{l^*}$ e $\frac{y^*}{l^*}$ podemos calcular $\frac{k^*}{y^*}$ dividindo as duas razões:

$$\frac{k^*}{y^*} = \frac{\frac{k^*}{l^*}}{\frac{y^*}{l^*}} = \frac{\left(\frac{\beta\alpha\bar{z}}{1-\beta(1-\delta)}\right)^{\frac{1}{1-\alpha}}}{\bar{z}\left(\frac{\beta\alpha\bar{z}}{1-\beta(1-\delta)}\right)^{\frac{\alpha}{1-\alpha}}} = \frac{\frac{\beta\alpha\bar{z}}{1-\beta(1-\delta)}}{\bar{z}} = \frac{\alpha\beta}{1-\beta(1-\delta)}$$

Voltando as restrições:

$$y^* = c^* + i^*$$

Temos que $i^* = k^* - (1-\delta)k^*$ logo $i^* = \delta k^*$, portanto:

$$y^* = c^* + \delta k^*$$

$$1 = \frac{c^*}{y^*} + \delta\frac{k^*}{y^*}$$

$$\frac{c^*}{y^*} = 1 - \delta\frac{k^*}{y^*} = 1 - \frac{\alpha\beta\delta}{1-\beta(1-\delta)}$$

$$\frac{c^*}{y^*} = \frac{1-\beta(1-\delta)-\alpha\beta\delta}{1-\beta(1-\delta)} = \frac{1-\beta+\beta\delta-\alpha\beta\delta}{1-\beta(1-\delta)}$$

$$\frac{c^*}{y^*} = \frac{1-\beta(1-\delta-\alpha)}{1-\beta(1-\delta)}$$

O que retiramos das razões calculadas? A razão calculada $\frac{k^*}{l^*}$ mostra que quando $\bar{z}$ cresce a razão também cresce, da mesma forma ocorre com $\frac{y^*}{l^*}$, a razão $\frac{k^*}{y^*}$ é invariante quanto à $\bar{z}$ assim como $\frac{c^*}{y^*}$ ou seja temos que a determinação de l é invariante em relação à $\bar{z}$

(d)

Vamos a questão computacional:

- Note que aqui nosso problema é mais complexo uma vez que envolve introdução do fator trabalho e além disso do fator estocástico.

```
import numpy as np
import time
import matplotlib.pyplot as plt

## Exercicio 4
# - Inclusao de choques estocasticos
# - Inclusao da escolha de trabalho
start = time.time()
# Grid de Capital
k_grid = np.linspace(0.01, 10, 51 )

# Grid de Trabalho
```

```
n_grid = np.linspace(0, 1, 11 )

# Vetor de choques
Z = np.array([0.8, 1.0, 1.2])

# Matriz de transi   o
pi = np.array([[ 0.2,  0.5,  0.3 ],
               [ 0.1,  0.6,  0.3 ],
               [ 0.25, 0.25, 0.5 ]])

# Parametros
alpha = 0.3
beta  = 1/1.05
delta = 0.05
phi   = 1

# Funcao Utilidade
def uti(c,l,phi):
    return np.log(c) - (l**(1+phi))/(1+phi)

# Funcao de Producao
def prod( z,k,l, alpha):
    return z*(k**alpha)*(l**(1-alpha))

# Vetores de chute inicial e de iteracao
v_ini = np.zeros( len(k_grid) )
z_ini = np.array([ v_ini for i in range(0,len(Z)) ])
# Vetores de itera   o
Gk_it = np.zeros( len( k_grid ))
Gn_it = np.zeros( len( k_grid ))
gn_it = np.zeros( len( k_grid ))
# Tolerancia
tol  = 1e-5
dist = 1000
it = 0
# Dist
hist_dist = []
V_hist    = []
k_hist    = []
n_hist    = []
# Loop
while dist > tol:
    # Para Loop se tivermos mais de 1000 Iteracoes
    if it >= 1000:
        break
    # Vetor de Iteracao Z
    Tz = np.array( [v_ini for i in range(0,len( Z ))])
    Gk = np.array( [v_ini for i in range(0,len( Z ))])
    Gn = np.array( [v_ini for i in range(0,len( Z ))])
    # Iteracao sobre vetor Z
    for z in range(0,len(Z)):
```

```
        # Vetor de Iteracao k
        Tv = np.zeros( len( k_grid ))
        # Iteracao sobre k
        for i in range(0,len(k_grid)):
            v_it = np.zeros( len( k_grid ))
            for j in range(0,len( k_grid )):
                # Iteracao sobre sobre vetor n_grid
                n_it = np.zeros( len( n_grid ))
                for n in range(0,len( n_grid )):
                    # Calcula Consumo
                    cons = prod(Z[z],k_grid[i],n_grid[n],alpha)+(1-delta)*
                                                     k_grid[i]-k_grid[
                                                     j]
                    if cons >  0:
                        Ev      = np.dot(pi[z],z_ini)[j] # np.sum((np.
                                                          transpose(
                                                          z_ini)*pi[z])
                                                          [j])
                        n_it[n] = uti(cons,n_grid[n],phi) + beta*Ev
                    if cons <= 0:
                        n_it[n] = -np.inf
                v_it[j]  = np.max( n_it )
                gn_it[j] = n_grid[ np.argmax( n_it ) ]
            Tv[i]   = np.max( v_it )
            Gk_it[i] = k_grid[ np.argmax( v_it ) ]
            Gn_it[i] = gn_it[ np.argmax( v_it ) ]
        # Guarda valores para cada z distinto
        Gn[z] = Gn_it
        Gk[z] = Gk_it
        Tz[z] = Tv
    # Calcula distancia
    dist = np.max( abs( Tz - z_ini ))
    # Historico
    #V_hist.append(Tz)
    #k_hist.append(Gk)
    #n_hist.append(Gn)
    # dist = np.sqrt( np.sum( np.square( Tz - z_ini ) ) )
    # hist_dist.append( dist )
    print( "iteracao: ", it," ,Distancia: ",dist )
    # Atualiza vetores
    z_ini = Tz
    # Atualiza Iteracao
    it = it + 1

end = time.time()
print( end - start )

# Plot Funcao Valor
for item in range(0,len(Tz)):
    plt.plot( k_grid, Tz[item], label = r"$z=$"+str(Z[item]) )
plt.legend()
```

```
plt.xlabel(r"$k$")
plt.ylabel(r"$V$")
plt.show()

# Plot Funcao Politica Capital
for item in range(0,len(Gk)):
    plt.plot( k_grid, Gk[item], label = r"$z=$"+str(Z[item]) )
plt.plot( k_grid, k_grid, linestyle = "dashed", color = "grey", label = "
                                                90 graus" )
plt.legend()
plt.xlabel(r"$k$")
plt.ylabel(r"$k'$")
plt.show()

# Plot Funcao Politica trabalho
for item in range(0,len(Gn)):
    plt.plot( k_grid, Gn[item], label = r"$z=$"+str(Z[item]) )
plt.legend()
plt.xlabel(r"$k$")
plt.ylabel(r"$l$")
plt.show()

# Capital de estado estacionario
for i_z, zs in enumerate( Z ):
    i_k = 0
    it  = 0
    while k_grid[i_k] != Gk[i_z][i_k] and it < 100:
        i_k = np.where( k_grid == Gk[i_z][i_k] )[0][0]

        it = it + 1
    print( "z = ",zs,", k_ss = ",k_grid[i_k])
```

Vamos observar os resultados Gráficos:

É possível identificar que para quanto maior o nível de choque de produtividade z temos que há um valor maior associado, ou seja a utilidade aumenta, uma vez que a produtividade para mesmo níveis de capital é maior o que permite maior consumo e poupança.

Pela função politica do capital vemos que a curva se torna menor horizontal indicando que há um nível de aplicação maior do capital quanto maior o choque de produtividade z, vemos que para a função política do trabalho temos um nível maior de utilização do trabalho, ou seja a utilidade do consumo gerada pelo choque de produtividade compensa a desutilidade do trabalho.

Por último na tabela abaixo temos os valores de capital de *steady state* para os diferentes níveis de choque de produtividade, com respectivo consumo e também respectiva utilização da força de trabalho.

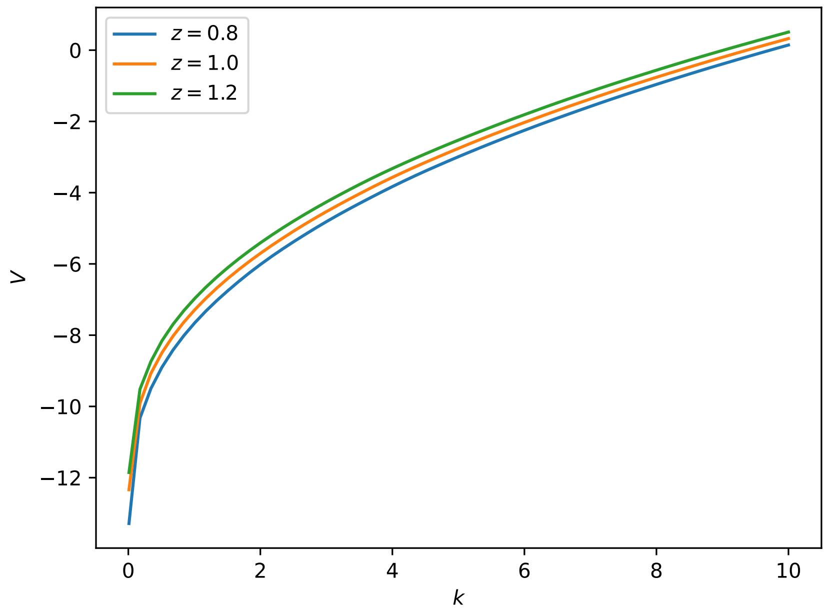

Figura 3: Função Valor para diferentes z

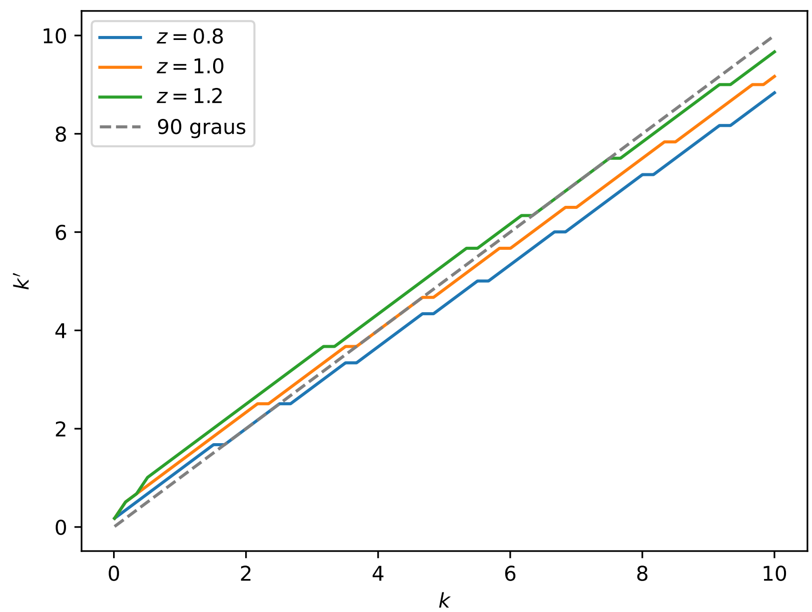

Figura 4: Função Politica de k' para diferentes z

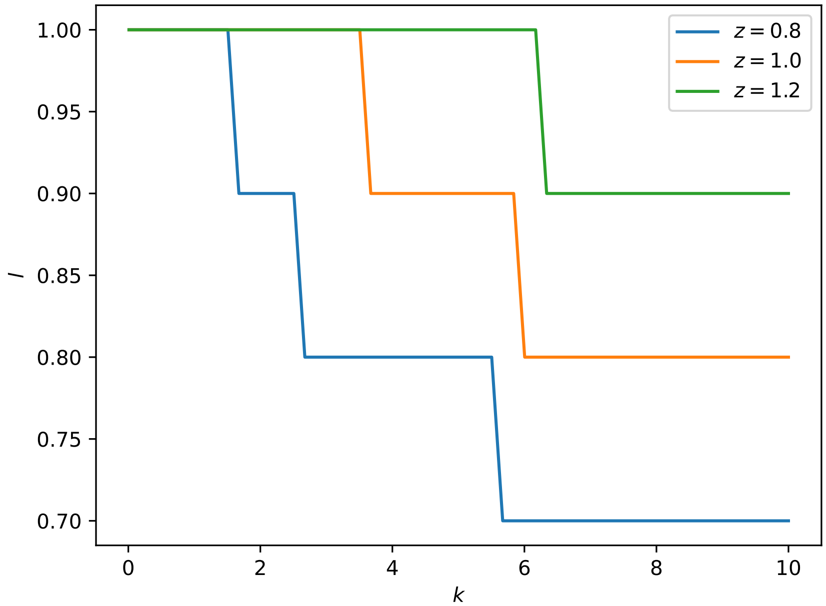

Figura 5: Função Politica de l para diferentes z

z	k^*	c^*	l^*
0.8	1.675	0.784	0.9
1	3.673	1.189	0.9
1.2	6.337	1.623	0.9

Problem 5

Escreva a função valor para cada uma das seguintes descrições de economias com tempo discreto em que os indivíduos tem fator de desconto $\beta \in (0, 1)$:

(a) Considere uma economia onde os indivíduos podem escolher entre trabalhar e não trabalhar. Se trabalham, recebem um salário de W e pagam um imposto de T. Se não trabalharem, recebem subsídio de desemprego de B. No entanto, o governo só pode pagar benefícios a um número limitado de pessoas, portanto, há um limite para o número de pessoas que podem receber benefícios. Suponha que o indivíduo tem um crença de que recebe o benefício com probabilidade $p \in (0, 1)$. Escreva a função de valor de um indivíduo que deve decidir se quer trabalhar ou não, dado o salário, impostos, benefícios e limite de benefícios.

(b) Em um mundo onde existem apenas dois tipos de bens, comida e abrigo, os indivíduos devem decidir como alocar seus recursos limitados entre os dois. Eles podem produzir comida ou abrigo, mas a produção de um tipo de bem requer uma certa quantidade do outro tipo de bem como insumo. Cada indivíduo tem uma função de utilidade, $U(f, s)$, onde f é a quantidade de comida consumida e s é a quantidade de abrigo consumida, e uma função de produção para cada bem, $f = F(f, s)$ e $s = S(f, s)$ respectivamente, onde F e S são diferenciáveis e crescentes em cada argumento, e satisfazem as condições de Inada.

O preço da comida e p_f e o preço do abrigo e p_s. Um indivíduo tem uma restrição orçamentária dada por $p_f f + p_s s = w$, onde w é a renda do indivíduo. Suponha que w siga um processo de Markov com N estados. Escreva a função de valor de um indivíduo que deve decidir quanto de cada bem produzir e consumir, dados os preços dos alimentos e abrigo, a quantidade de recursos disponíveis e sua função de utilidade e funções de produção.

Solution

(a)

Aqui temos basicamente um exercício que lida com coisas mais simples em relação a capacidade de construir problemas recursivos. Para esse exemplo vamos identificar a variável de controle que aqui é basicamente a decisão ou não de trabalhar, a partir disso podemos montar nosso problema recursivo de duas formas distintas.

A primeira vamos usar o fato do individuo decidir se aloca seu tempo entre trabalho ou não por meio de uma variável indicadora I_w.

$$\max_{I_w \in \{0,1\}} \{I_w(1-T)W + (1-I_w)pB\}$$

Ou de forma alternativa:

$$\max \ \{(1-T)W, pB\}$$

(b)

Aqui o procedimento é semelhante, porém temos que ter um pouco mais de cuidado ao construir nosso problema e usar todas as funções disponíveis, dessa forma temos:

$$v(f,s) = \max_{\substack{p_f f + p_s s \leq w \\ f>0 \\ s>0}} \{U(F(f,s), S(f,s)) + \beta E[v(f', s')]\}$$

Ou seja a função valor tem como variaveis de controle a quantidade de comida e abrigo que serão consumidas, seja para consumo ou para produção, o consumo precisa ser positiva para ambas as commodities, e sujeito a restrição orçamentária.

Referências

Junior, F. A. B. (2025). *Notas de Aula*. FEARP/USP. (Ver p. 9).