Lista II - Métodos Numéricos

EPGE - 2018

Professor: Cézar Santos

Aluno: Raul Guarini Riva

Problema 1. O problema no planejador consiste em escolher recursivamente quanto consumir no presente e quanto poupar na forma de capital para o próximo período, tendo por base o estado da economia formado pelo estoque de capital atual e a produtividade atual.

A função utilidade das famílias não leva em consideração o lazer. Desta forma, ofertarão trabalho inelasticamente. Normalizando a dotação de trabalho para uma unidade, a equação funcional com a qual o planejador se defronta é a seguinte:

$$V(K,z) = \max_{c,K'}\{u(c) + \beta\mathbb{E}[V(K',z')|z]\}$$
$$\text{s.t.} \quad c + K' \leq zK^{\alpha} + (1-\delta)K$$

A hipótese de u estritamente crescente implica que a restrição de recursos será satisfeita com igualdade em todo instante do tempo. Daí, reescreve-se:

$$V(K,z) = \max_{K'}\{u(zK^{\alpha} + (1-\delta)K - K') + \beta\mathbb{E}[V(K',z')|z]\}$$

Problema 2. Supondo não haver incerteza, normaliza-se o valor da produtividade para a média incondicional do processo:

$$\log(z) = 0 \implies z = 1$$

A equação de Euler é dada por

$$\beta\mathbb{E}\left[\left(\frac{c'}{c}\right)^{-\mu}(z'\alpha K'^{\alpha-1} + 1 - \delta)\big|z\right] = 1$$

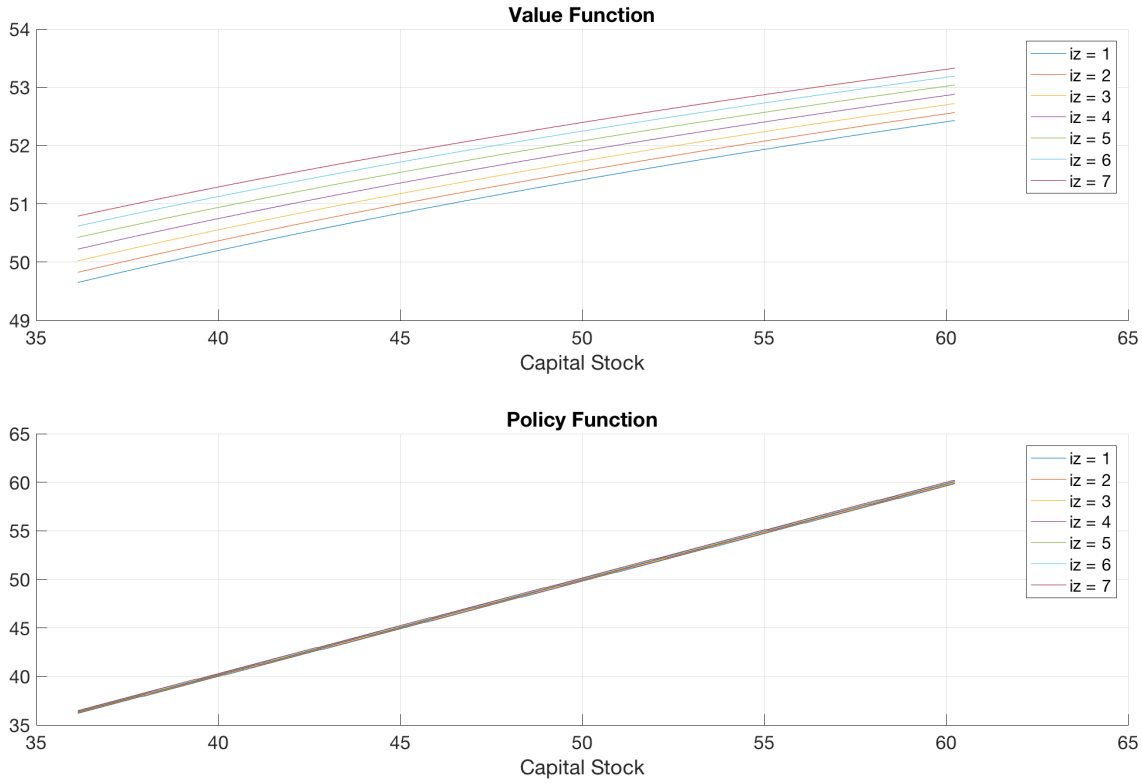
Sem incerteza e já resolvendo para o estado estacionário:

$$\beta\alpha K_{ss}^{\alpha-1} = 1 - \beta(1-\delta)$$
$$\boxed{K_{ss} = \left(\frac{\alpha\beta}{1-\beta(1-\delta)}\right)^{\frac{1}{1-\alpha}}}$$

Com a calibração sugerida na lista, temos $K_{ss} = 48.1905$.

Problema 3. Para esta primeira implementação, explorei a monotonicidade da função política e a concavidade do funcional sendo otimizado. O código está no formato de uma função chamada `VFinder_Iterated.m`, devidamente documentada através do próprio `help` do MATLAB.

A primeira iteração foi feita através de força bruta e as próximas seguiram explorando os aspectos teóricos do problema. Abaixo, a função valor e a função política (como esperado, crescente!!!):



Obtive o resultado, também esperado, de que a função valor, para um valor de k fixo, é crescente na TFP. A legenda iz $= 4$, por exemplo, indica que determinada curva foi computada com o valor da TFP correspondente ao quarto ponto do grid do choque $\log(z)$.

Para comparar o desempenho desta abordagem (que não utiliza nenhum tipo de vetorização), implementei o método de "força bruta vetorizada". Isto é, operar maximizações ao longo das dimensões de arrays do MATLAB evitando criar "for loops". Esta abordagem está no script `brute_force.m`.

Houve um ganho de desempenho considerável. O método de força bruta vetorizada computou a função valor e a função política em 8.85 segundos. O método que utiliza a função `VFinder_Iterated.m` resolveu o mesmo problema em 6.39 segundos. Isto representa um ganho de quase 28%, o que considero expressivo.

Nota-se, entretanto, que o método de força bruta é mais geral que seu concorrente, em vista do fato que funcionaria igualmente bem para um problema em que não tivéssemos um resultado teórico garantindo a monotonicidade da função política ou a concavidade do funcional em questão. Uma pior performance é o preço que se paga pela maior versatilidade.