Lista III - Métodos Numéricos

EPGE - 2018

Professor: Cézar Santos

Aluno: Raul Guarini Riva

Problema 1. Segui a mesma calibração da lista anterior e encontrei as funções política do consumo e do capital através da técnica de colocação aliada aos polinômios de Chebyshev. O código principal da lista está no arquivo `ps3.m`. O tempo de execução desta técnica de projeção espectral foi o menor dentre todas as técnicas até agora (desde a Lista 2): ao redor de 0.4 segundos.

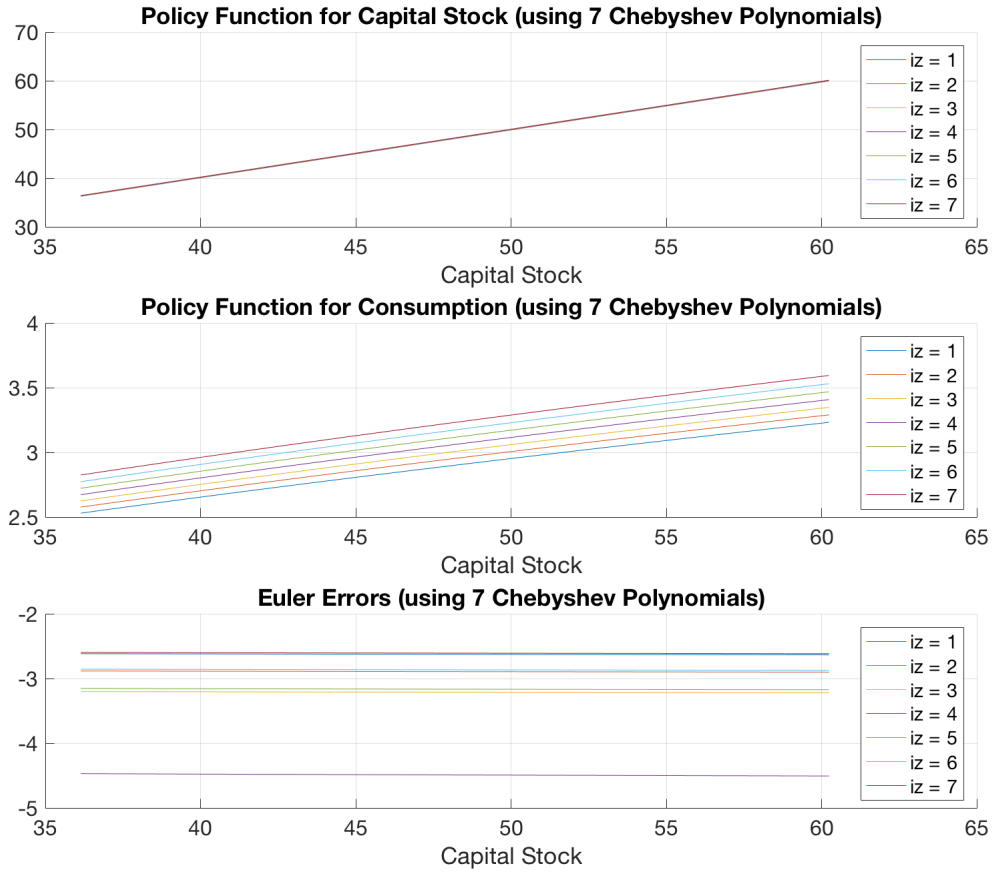
A discretização do processo estocástico da TFP foi feito através da técnica de Tauchen, da mesma maneira das outras listas. Utilizei 7 polinômios de Chebyshev para computar a projeção ($d = 6$, na notação dos slides). O problema é muito bem comportado e, portanto, não foi necessário d muito grande. Testei dimensões maiores mas os valores correspondente de γ_j rapidamente iam para zero, indicado não ser necessário mais graus de liberdade para a projeção.

A função `chebyshev_poly` computa raízes de um polinômio de Chebyshev de ordem k e pode avaliá-lo em qualquer ponto de $[-1, 1]$. A função `C_proj` computa a projeção da função política do consumo dado um vetor $(d + 1)$-dimensional γ e um grid no qual o valor do capital de interesse se encontra. Finalmente a função `risk_function` computa o resíduo[1] da projeção. É esta a função que define o sistema de $(d + 1)$ equações que a função do MATLAB `fsolve` resolve. A próxima figura ilustra as funções política e Euler Errors.

Assim como antes, a função política do capital é praticamente linear e não varia muito de acordo com a TFP. Entretanto, níveis maiores de TFP levam a níveis maiores de capital no próximo período, dado o mesmo capital inicial. Este tipo de monotonicidade também é verificada na função política do consumo. Nota-se que esta tem maior curvatura do que a função política do capital.

Curiosamente, o aspecto das curvas de Euler Errors é diferente daquele encontrado, por exemplo, na implementação do grid endógeno na Lista 2. As curvas do erro são basicamente funções constantes de k mas apresentam um comportamento interessante em z. Prestando atenção na legenda, vemos que o menor erro de aproximação foi alcançado no valor central do grid de $iz = 4$ e as outras curvas parecem estar dispostas duas a duas de acordo com a distância ao centro. Quanto mais próximo a este, menor o erro de aproximação. Apesar desta observação, não consegui interpretar este resultado.

[1] Não sei bem o motivo, mas de alguma forma achei que o $R(\gamma, K)$ dos slides era "Risk" e não "Residual". Implementei tudo e usei a função várias vezes, só notei no final a confusão. Achei que iria semear bugs se trocasse o nome, então deixei desse jeito. Desculpa!



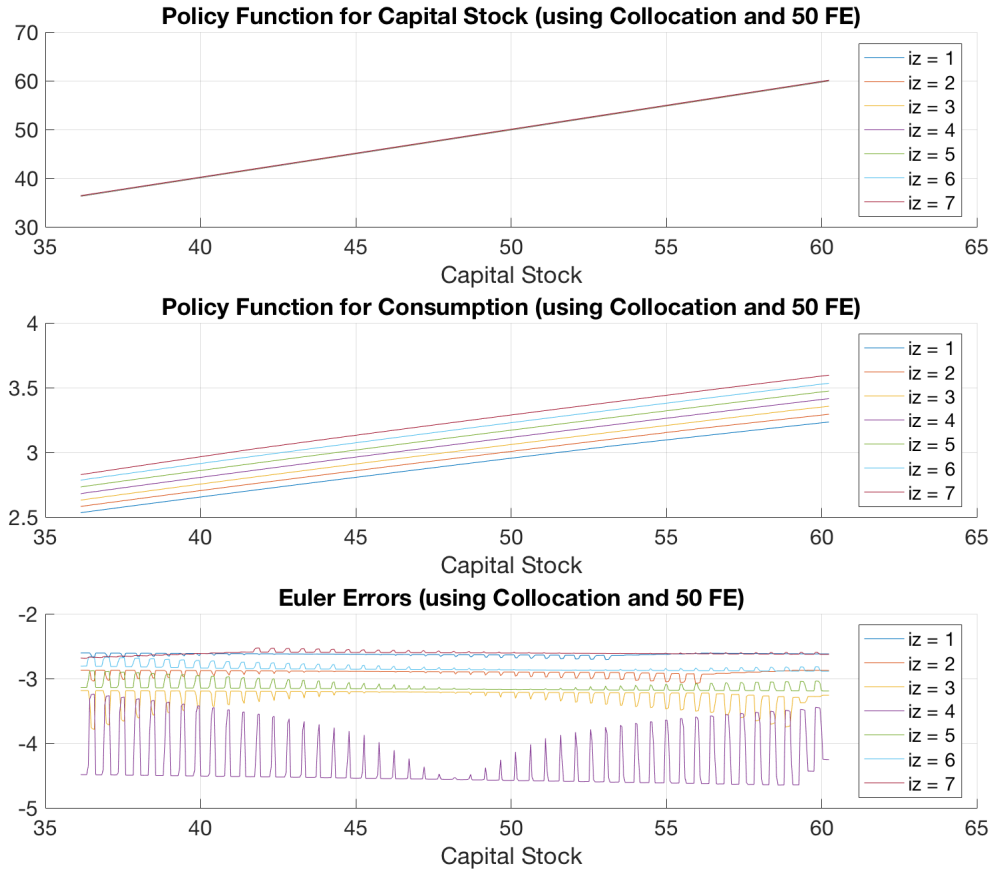
Problema 2. Resolvi primeiro o problema utilizando o método da colocação e elementos finitos. Esta abordagem se mostrou também bastante rápida comparada aos métodos da Lista 2, ainda que tenha tido desempenho ligeiramente pior do que a solução usando Polinômios de Chebyshev: 0.62 segundos.

Dividi o intervalo correspondente ao grid do capital (o mesmo das outras questões) em 50 elementos finitos, isto é, 50 subintervalos. Dentro de cada elemento, fitei uma função afim do capital do tipo intercepto$_i$+$a_i k$, quando o valor k cai no i-ésimo elemento. Portanto, computei ao todo 100 parâmetros. Para tal, utilizei o método da colocação em 100 pontos linearmente espaçados ao longo do grid original do capital.

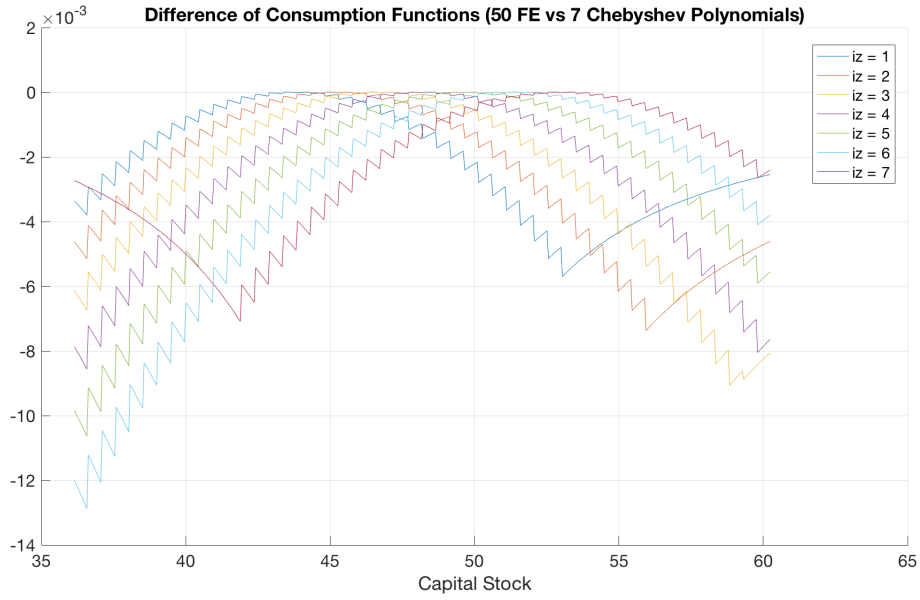
A função `find_element` indica em qual elemento determinado valor de k se encontra utilizando o método "Divide and Conquer" dos slides, visando aumentar a performance nesta parte do algoritmo. A função `C_proj_finel` é análoga à função `C_proj` e computa, dados os parâmetros, um valor de z e um valor de k, o consumo ótimo. Por último, a função `risk_function_finel` computa[2] o vetor de resíduos que a rotina `fsolve` tentará igualar a zero tendo os 100 parâmetros em questão como graus de liberdade.

A próxima figura ilustra a função política do capital induzida pela função política do consumo, a própria função política do consumo e os Euler Errors:

[2] O mesmo disclaimer sobre Risk e Residual...



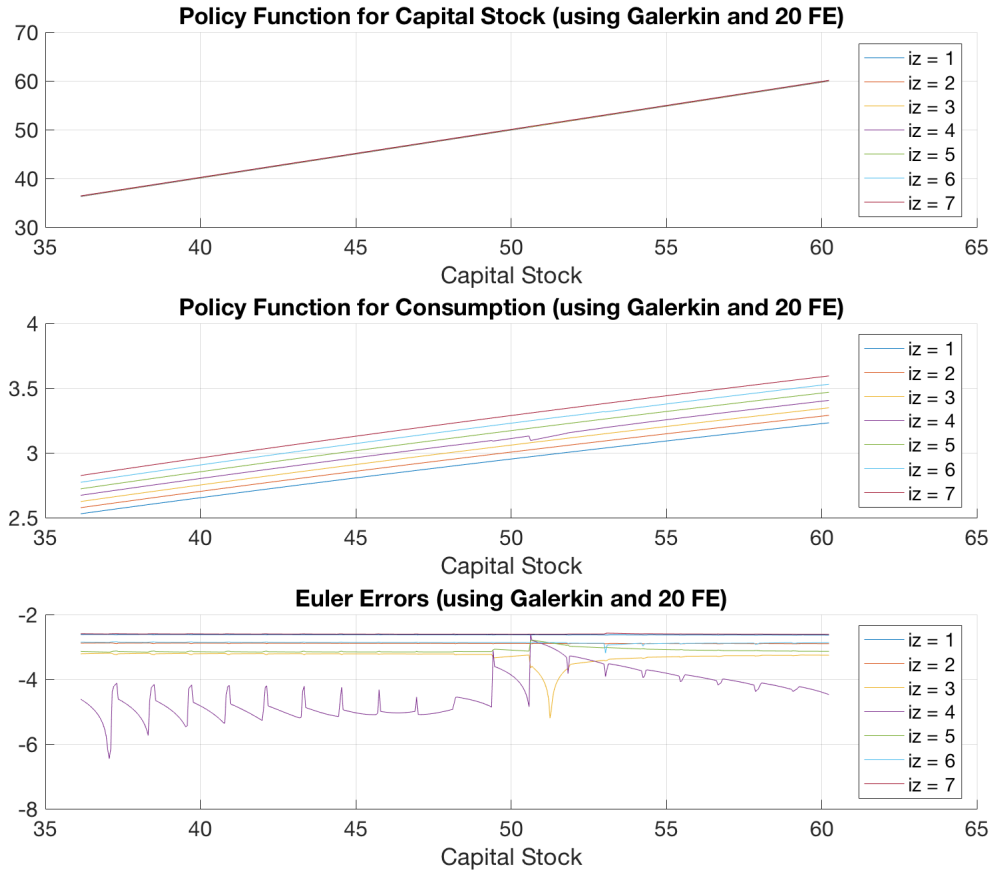
Um outro gráfico interessante é o da diferença ponto a ponto em (k, z) das duas soluções para a função política do consumo encontradas até agora, utilizando polinômios de Chebyshev e Elementos Finitos com Colocação:



A boa notícia é que a diferença (dos níveis) é bem pequena, da ordem de 10^{-2} no pior dos casos, quando

o consumo varia aproximadamente entre 2.5 e 4 no estado estacionário. Curiosamente, esta diferença parece estar limitada superiormente por zero, isto é, parece ser o caso em que esta implementação de elementos finitos gera uma função política do consumo que está sempre acima daquela encontrada pelo método anterior. Contudo, é exatamente este o comportamento documentado nos slides ao comparar os polinômios de Chebyshev e o método de elementos finitos.

Na segunda implementação utilizando elementos finitos, apliquei a técnica de Galerkin dos slides. A figura a seguir mostra que a solução encontrada é similar às outras, a menos de uma quina no caso de $iz = 4$. A diferença é que esta implementação mostrou-se MUITO mais demorada do que as outras: quase 10 *minutos* se passaram até que a rotina `fsolve` conseguisse encontrar a solução.



Com poucos elementos finitos, a solução para a função política do capital continha muitas quinas. Tal qual o caso anterior, dentro de cada elemento a função política é uma função afim do capital. Portanto, foram computados 40 parâmetros, em vista dos 20 elementos finitos.

Acredito que o motivo desta lentidão seja a necessidade de computar integrais numéricas. A função `risk_function_galerkin` utiliza as outras funções já desenvolvidas para a implementação com colocação e a rotina interna do MATLAB `integral` para computar 40 integrais numéricas em cada rodada de `fsolve`.

De acordo com a documentação, este é o método default mais recomendado para a integração de

funções unidimensionais no MATLAB, usando um tipo de quadratura dita "globalmente adaptativa". No entanto, o overhead criado por cada rodada de `fsolve` parece acumular-se rapidamente. No caso de usarmos, por exemplo, 5 elementos finitos a solução é encontrada em pouco menos de 30 segundos. Todavia, com várias quinas indesejáveis. Já com 10 elementos, o tempo de execução foi pouco mais de um minuto com menos quinas, mas ainda não tão bem comportada como a solução acima.

Em conclusão, o método de Galerkin, ao requerer integração numérica, não se mostrou uma boa alternativa aos outros métodos já estudados para resolver este problema. A função política é suave o suficiente para que métodos de projeção global como o método dos Polinômios de Chebyshev sejam capazes de computá-la. O método de elementos finitos com colocação também se mostrou uma abordagem viável nesta questão.