Trabalho 1

Gustavo Giro Resende - 11340571

Parte 1) Eliminaçao de Gauss com e sem pivotamento

Apresentação de um caso exemplo:

Matriz qualquer:

[[1 1 1]
 [1 0 10]
 [0 10 1]]

Vetor 'b' do sistema: [0 -48 25]

Matriz escalonada pelo metodo de solução sem pivotamento:

[[1 1 1]
 [0 -1 9]
 [0 0 91]]

Solucao encontrada pelo metodo de solucao sem pivotamento:

x = [2. 3. -5.]

Solucao encontrada pelo metodo de solucao sem pivotamento:

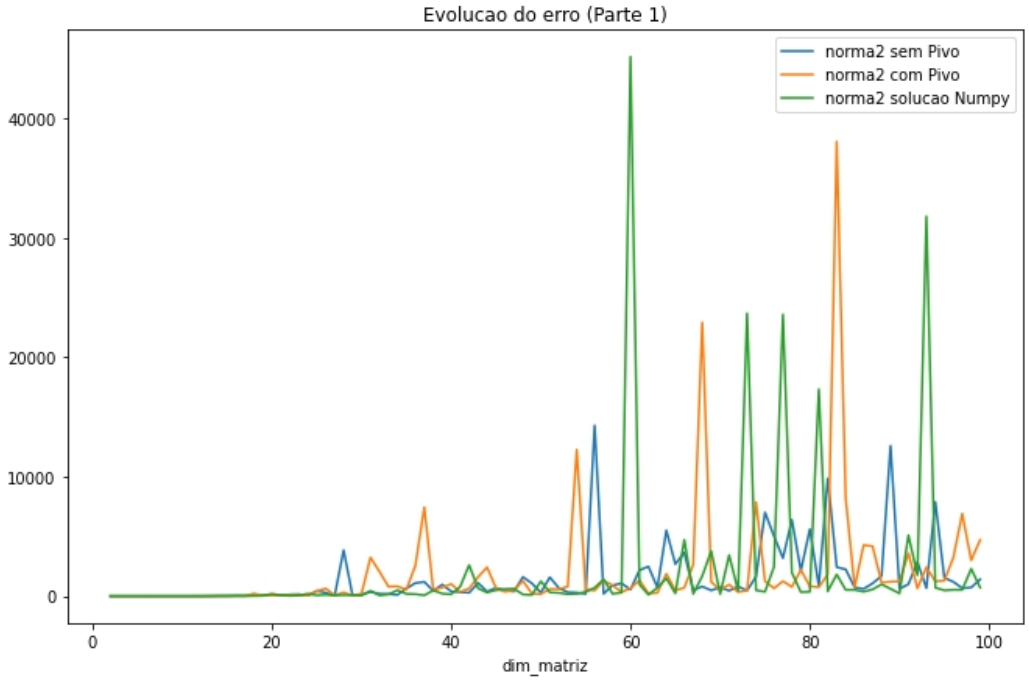
x = [2. 3. -5.]

Estudo e resultados encontrados:

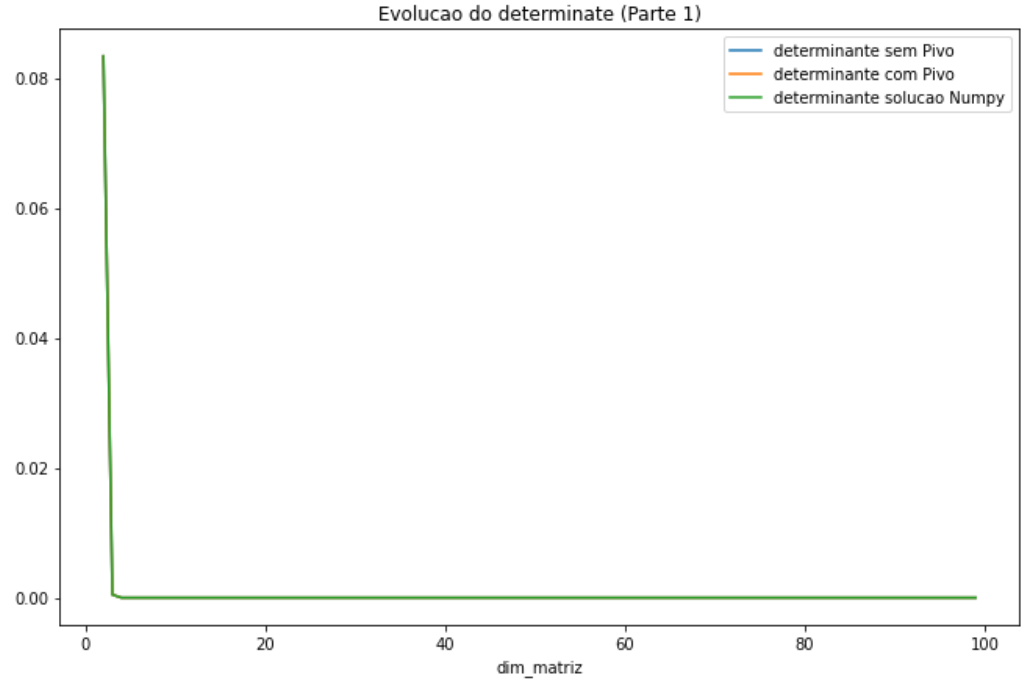
A hipótese inicial é de que o metódo de eliminação de Gauss com pivotamento parcial é mais eficiente para matrizes com numeros do tipo float (reais), com muitas casas decimais, devido ao fato de que viza diminuir o erro numérico. Foram comparados soluções com o metodo de eliminação de Gauss sem pivotamento (simples), com pivotamento parcial, com pivotamento parcial escalado e soluções encontradas com a função interna *linalg.solve* do numpy.

Implementados os algoritmos que geram a matriz de Hilbert e realizam a eliminação de Gauss, com e sem pivotamento, sobre essa matriz, podemos observar nos dados abaixo que conforme a ordem da matriz vai aumentando, sua proximidade com uma matriz singular (dada pelo calculo do determinante) é bem alta. Com um determinante próximo de zero, a solução do sistema Ax = b encontrada por uma inversão da matriz A deixa de ser viável, o que explicaria o aumento do erro numerico (considerando a matriz A quadrada de ordem n x n). Podemos observar essa relação pelos graficos que se seguem (os dados encontram-se na tabela ‘*resultados_matrizHilbert.xlsx*’):

1) Dimensão da matriz X Norma do erro



2) Dimensão da matriz X Determinante



Comparando as medias dos valores encontrados, pordemos observar que o metodo de eliminação com pivotamento parcial escalado obteve o menor valor, o que já era esperado mas, em contraste, o metodo sem pivotamento obterve uma norma^2 menor (em media), que o metodo com pivotamento parcial. A maior norma^2 media observada foram nas soluções econtradas pelo próprio *linalg.solve* do Numpy. Os dados seguem na tabela:

	norma2 sem Pivo	norma2 com Pivo	norma2 com Pivo Escalado	norma2 solucao Numpy
count	7.900000e+01	7.900000e+01	7.900000e+01	7.900000e+01
mean	1.138150e+03	1.230417e+03	1.086361e+03	1.659172e+03
std	2.153057e+03	3.080271e+03	2.137511e+03	6.220002e+03
min	8.005932e-16	8.005932e-16	8.005932e-16	8.005932e-16
25%	5.871685e+01	5.227842e+01	5.871685e+01	5.096532e+01
50%	3.914121e+02	4.973457e+02	3.919891e+02	1.886830e+02
75%	1.064548e+03	9.316820e+02	1.046194e+03	5.438470e+02
max	1.428824e+04	2.291615e+04	1.485411e+04	4.515595e+04

Os determinantes encontrados por ambos os métodos tendem rapidamente para 0, para ordens a partir 10x10 da matriz. Já o erro para ordens maiores de matriz, a partir de 40x40, é menor nos metódos sem pivotamento e com pivotamento parcial escalado.

O que poderia explicar essa diferença é o tipo de matriz que esta sendo trabalhada nesse caso. A matriz de Hilbert, por definição, possui valores cada vez menores conforme aumenta ‘i’ e ‘j’, mantendo sempre os elementos da diagonal principal maiores que os demais na mesma linha e coluna, o que pode ter favorecido o metodo mais simples de eliminação.

Parte 2) Decomposição de Cholesky

Apresentação de um caso exemplo:

Matriz A do sistema:

```
[[4 2 2]
 [2 6 2]
 [2 2 5]]
```

Vetor 'b' do sistema: [8 10 9]

Decomposicao de A em L e L_transposta:

(L)

```
[[2.         0.                   0.        ]
 [1.         2.23606798           0.        ]
 [1.         0.4472136  1.94935887]]
```

(L_transposta)

[[2. 1. 1.]
[0. 2.23606798 0.4472136]
[0. 0. 1.94935887]]

Solucao encontrada pelo metodo de Cholesky:

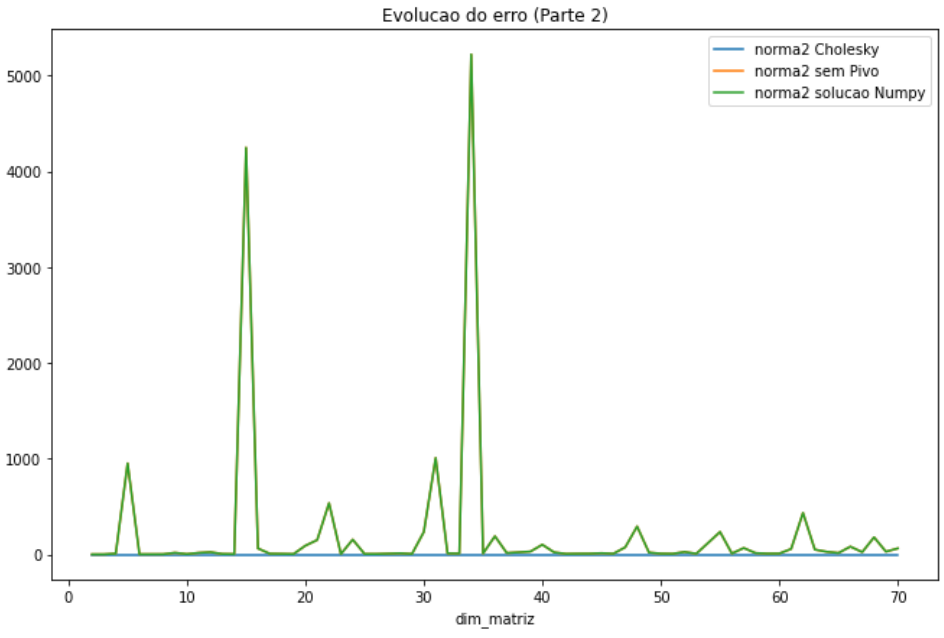
x = [1. 1. 1.]

Estudo e resultados encontrados:

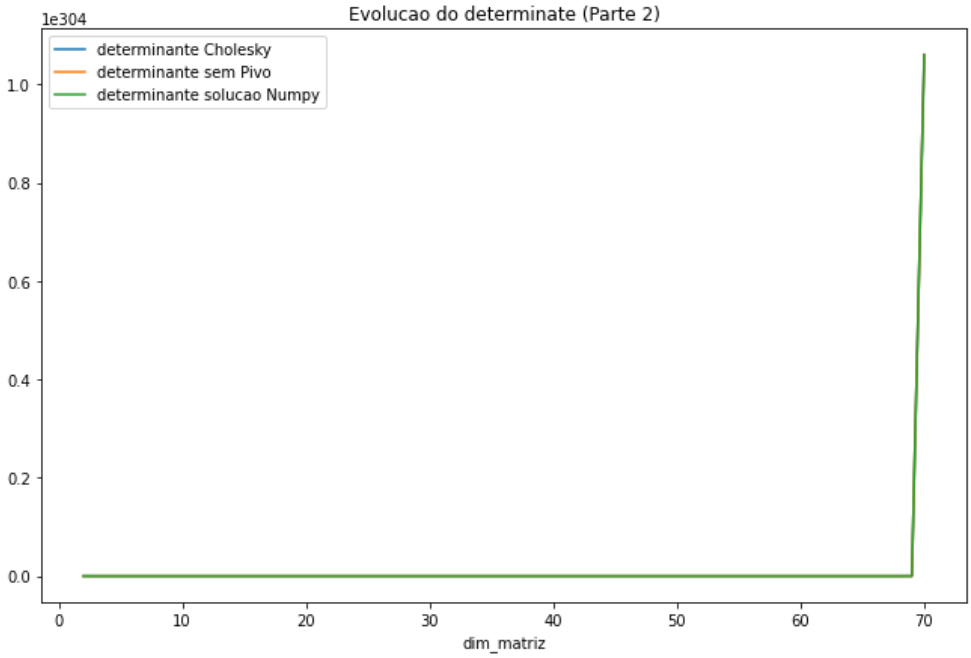
A hipótese estudada é que, segundo o teorema 6.26 (*Numerical Analysis*, Burden e Faires, pagina: 417), matrizes simétricas positivas definidas possuem cálculos estáveis em relação ao crescimento do arredondamento dos erros. Foram comparadas soluções encontradas pelo método de fatorização de Cholesky em matrizes L e L_transposta, pelo método de eliminação de Gauss simples e pelo *linalg.solve* do Numpy. As matrizes usadas nessa segunda parte foram geradas com o comando *np.ramdom.int()* do numpy, de forma que os numeros trabalhados fossem inteiros.

Realizando a fatoração das matrizes simetricas, é possível observar que a norma^2 do erro mantem-se estável para todas as soluções encontradas, variando apenas em alguns picos nos metodos de eliminação de Gauss e do próprio Numpy. Conforme as dimensões das matrizes foram ficando maiores, foi possível observar que os determinantes encontrados também o foram, sendo que não houve divergências entre um metodo ou outro neste caso. Essas relações podem ser observadas pelos graficos que se seguem (os dados encontram-se na tabela '*resultados_matrizHilbert.xlsx*'):

1) Dimensão da matriz X Norma do erro



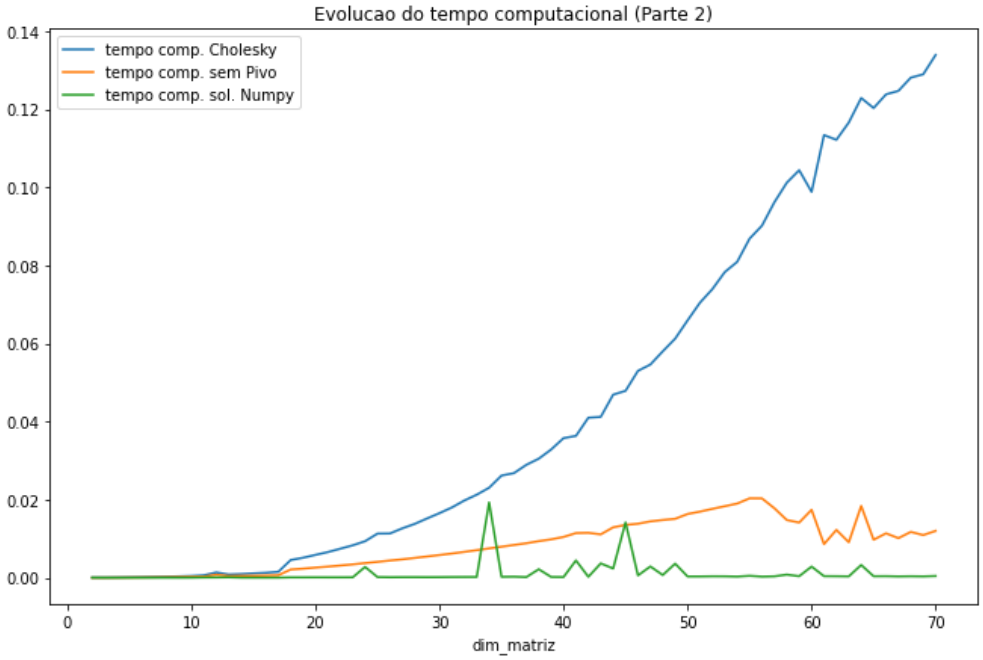
2) Dimensão da matriz X Determinante



Comparando as medias dos valores encontrados, pordemos observar que o metodo de Cholesky obteve um valor bem menor que os demais, o que corrobora com o teorema acima, enquanto os metodos de eliminação de Gauss simples e o metodo do Numpy obtiveram a mesma media. Os dados podem ser observados na tabela:

	norma2 Cholesky	norma2 sem Pivo	norma2 solucao Numpy
count	6.900000e+01	69.000000	69.000000
mean	1.270746e-10	219.527510	219.527510
std	3.891997e-10	811.125798	811.125797
min	0.000000e+00	1.606934	1.606934
25%	2.309711e-12	8.246811	8.246811
50%	1.638443e-11	15.737768	15.737768
75%	1.035641e-10	75.395627	75.395627
max	2.952116e-09	5220.042517	5220.042515

Por fim, por mais eficiente que seja o metodo de fatoração de Cholesky, ele é o que possui o maior tempo computacional, quando comparado com os demais. Conforme aumenta-se a dimensão da matriz, o tempo cresce bem mais rapido que os demais, enquanto o metodo do Numpy é o mais performático, mantendo-se estavel para dimensões acima de 60x60.



Diante do exposto, conclui-se que o metodo de Cholesky, para as matrizes utilizadas nesse estudo, é o mais eficiente diante dos demais. Embora seu tempo computacional seja maior, pode ser necessário para casos em que se exigem resultados mais estáveis e onde as matrizes que estão sendo trabalhadas sejam como as descritas acima.

Verificação dos dados

Para verificação das informações expostas acima, é possível utilizar o programa T1.py no mesmo diretório deste documento, chamando o programa como se segue:

Ex:

a) python T1.py 1 0 → Esta chamando o programa T1 para a primeira parte, sem ser teste

b) python T1.py 2 0 → Esta chamando o programa T1 para a segunda parte, sem ser teste

c) python T1.py 1 1 → Esta chamando o programa T1 para a primeira parte, com uma matriz de teste

d) python T1.py 2 1 → Esta chamando o programa T1 para a segunda parte, com uma matriz de teste

Os exemplos “a” e “b” geram, cada um, um arquivo *resultados_matrizHilbert.xlsx* e *resultados_Cholesky.xlsx*, respectivamente. Podem ser acessados tambem os arquivos .csv no mesmo diretório, de acordo com a preferência.