Trabalho Prático 3

João Pedro de Abreu Marciano

Questão 4

Teste 1

Testando a função com a matriz $A = \begin{bmatrix} 2 & 3 \\ 3 & 5 \end{bmatrix}$ e o vetor inicial $x_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

Os autovalores são $\lambda_1 = \dfrac{7+\sqrt{45}}{2} = 6,8541019$ e $\lambda_2 = \dfrac{7-\sqrt{45}}{2} = 0,145898034$

```
-->[lambda,x1,k,n_erro] = Metodo_potencia(A,x0,10^(-10),100)
 n_erro  =

   2.970D-11

 k  =

   7.

 x1  =

   0.618034
   1.

 lambda  =

   6.854102
```

```
-->[lambda,x1,k,n_erro] = Metodo_potencia_2(A,x0,10^(-10),100)
 n_erro  =

   2.149D-11

 k  =

   7.

 x1  =

   0.5257311
   0.8506508

 lambda  =

   6.854102
```

```
-->tic;[lambda,x1,k,n_erro] = Metodo_potencia(A,x0,10^(-10),100);toc
 ans  =

   0.000392


-->tic;[lambda,x1,k,n_erro] = Metodo_potencia_2(A,x0,10^(-10),100);toc
 ans  =

   0.000499
```

Vemos que ambos os algorítmos resolveram com 7 iterações porém o tempo do $Metodo\_potencia\_2$ demorou um pouco a mais nesse caso. Os autovetores resultantes foram um múltiplo um do outro já que as normas são diferentes em ambos os algorítmos.

Teste 2

Testando a função com a matriz $A = \begin{bmatrix} 2 & 1 & 1 \\ 2 & 3 & 4 \\ -1 & -1 & -2 \end{bmatrix}$ e o vetor inicial $x_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$.

Os autovalores são $\lambda_1 = 3$, $\lambda_2 = 1$ e $\lambda_3 = -1$,

```
-->[lambda,x1,k,n_erro] = Metodo_potencia(A,x0,10^(-10),100)
 n_erro  =

   8.970D-11

 k  =

   22.

 x1  =

   0.6666667
   1.
 - 0.3333333

 lambda  =

   3.
```

```
-->[lambda,x1,k,n_erro] = Metodo_potencia_2(A,x0,10^(-10),100)
 n_erro  =

   8.339D-11

 k  =

   22.

 x1  =

   0.5345225
   0.8017837
 - 0.2672612

 lambda  =

   3.
```

```
-->tic;[lambda,x1,k,n_erro] = Metodo_potencia(A,x0,10^(-10),100);toc
 ans  =

   0.000712


-->tic;[lambda,x1,k,n_erro] = Metodo_potencia_2(A,x0,10^(-10),100);toc
 ans  =

   0.000679
```

Vemos que ambos os algoritmos resolveram com 22 iteraçõe. Desta vez, o tempo do $Metodo\_potencia$ demorou um pouco a mais. Por enquanto, a diferença de tempo de execução entre os algorítmos foi praticamente insignificante.

Teste 3

Testando a função com a matriz $A = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 4 \\ 0 & 4 & 9 \end{bmatrix}$ e o vetor inicial $x_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$.

Os autovalores são $\lambda_1 = 11$, $\lambda_2 = 2$ e $\lambda_3 = 1$,

```
-->[lambda,x1,k,n_erro] = Metodo_potencia(A,x0,10^(-10),100)
 n_erro  =

   2.942D-11

 k  =

   15.

 x1  =

   6.537D-12
   0.5
   1.

 lambda  =

   11.
```

```
-->[lambda,x1,k,n_erro] = Metodo_potencia_2(A,x0,10^(-10),100)
 n_erro  =

   2.631D-11

 k  =

   15.

 x1  =

   5.847D-12
   0.4472136
   0.8944272

 lambda  =

   11.
```

```
-->tic;[lambda,x1,k,n_erro] = Metodo_potencia(A,x0,10^(-10),100);toc
 ans  =

   0.000335


-->tic;[lambda,x1,k,n_erro] = Metodo_potencia_2(A,x0,10^(-10),100);toc
 ans  =

   0.000468
```

Vemos que ambos os algoritmos resolveram com 15 iteraçõe. Desta vez, o tempo do $Metodo\_potencia\_2$ demorou um pouco a mais.

Teste 4

Testando a função com a matriz $A = \begin{bmatrix} 5 & 8 & 16 \\ 4 & 1 & 8 \\ -4 & -4 & -11 \end{bmatrix}$ e o vetor inicial $x_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$.

Os autovalores são $\lambda_1 = 1$, $\lambda_2 = \lambda_3 = -3$.

```
-->[lambda,x1,k,n_erro] = Metodo_potencia(A,x0,10^(-10),100)
 n_erro  =

   2.4

 k  =

   101.

 x1  =

   4.2
   1.8
  -3.

 lambda  =

   2.1428571
```

```
-->[lambda,x1,k,n_erro] = Metodo_potencia_2(A,x0,10^(-10),100)
 n_erro  =

   2.

 k  =

   101.

 x1  =

   2.3050495
   0.9878783
  -1.6464639

 lambda  =

  -3.
```

```
-->tic;[lambda,x1,k,n_erro] = Metodo_potencia(A,x0,10^(-10),100);toc
 ans  =

   0.002147


-->tic;[lambda,x1,k,n_erro] = Metodo_potencia_2(A,x0,10^(-10),100);toc
 ans  =

   0.002531
```

Vemos que o algorítmo *Metodo_potencia* não resolveu e atingiu o número máximo de iterações. O algorítmo *Metodo_potencia_2* também atingiu o número máximo de iterações, porém, intrigantemente, o valor de *lambda* é o esperado.

O motivo pelo qual o *Metodo_potencia* não funcionou é o fato de haver dois autovalores que possuem o maior módulo. Logo, a condição de convergência do algorítmo não é satisfeita. Já em *Metodo_potencia_2* o *lambda* é valor esperado, porém o *n_erro*, ou seja, a diferença $x_{k+1} - x_k$ é grande. Isso ocorreu provavelmente porque os vetores x_k e $xk + 1$ estão caindo no mesmo autoespaço porém com sentidos opostos. Para poder compara certo esse tipo de caso ao calcular a norma apartir de agora utilizaremos a norma da diferença dos vetores com valores absolutos.

O que pode estar causando a convergência em lambda nesse segundo algorítmo pode ser o uso de norma 2 ao invés de norma infinito.

Questão 5

Definimos a matriz $A = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 7 & 1 \\ 1 & 1 & 10 \end{bmatrix}$ e o vetor inicial $x_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$.

Os círculos de Gerchgorin são disjuntos. Logo, sabemos que tem um autovalor em cada um deles. E, além disso, esses autovalores são reais já que a matriz é simétrica.

Sejam λ_1, λ_2 e λ_3 os autovalores da matriz. Utilizando os círculos, temos:

$$|\lambda_1 - 10| \leq 2, \quad |\lambda_2 - 7| \leq 2, \quad |\lambda_3 - 2| \leq 2$$

```
-->[lambda,x1,k,n_erro] = Potencia_deslocada_inversa(A,x0,10^(-10),10,100)
 n_erro  =

   9.049D-11

 k  =

   13.

 x1  =

   0.1472762
   0.3113357
   0.9388183

 lambda  =

   10.488499
```

```
-->[lambda,x1,k,n_erro] = Potencia_deslocada_inversa_Ray(A,x0,10^(-10),10,100)
 n_erro  =

   7.657D-16

 k  =

   2.

 x1  =

   0.1472762
   0.3113357
   0.9388183

 lambda  =

   10.488499
```

```
-->tic;[lambda,x1,k,n_erro] = Potencia_deslocada_inversa(A,x0,10^(-10),10,100);toc
 ans  =

   0.003985


-->tic;[lambda,x1,k,n_erro] = Potencia_deslocada_inversa_Ray(A,x0,10^(-10),10,100);toc
 ans  =

   0.000537
```

```
-->[lambda,x1,k,n_erro] = Potencia_deslocada_inversa(A,x0,10^(-10),7,100)
 n_erro  =

   2.892D-11

 k  =

   10.

 x1  =

   0.1265795
   0.9354313
 - 0.3300696

 lambda  =

   6.7824639
```

```
-->[lambda,x1,k,n_erro] = Potencia_deslocada_inversa_Ray(A,x0,10^(-10),7,100)
 n_erro  =

   6.231D-16

 k  =

   2.

 x1  =

  -0.1265795
  -0.9354313
   0.3300696

 lambda  =

   6.7824639
```

```
-->tic;[lambda,x1,k,n_erro] = Potencia_deslocada_inversa(A,x0,10^(-10),7,100);toc
 ans  =

   0.005639


-->tic;[lambda,x1,k,n_erro] = Potencia_deslocada_inversa_Ray(A,x0,10^(-10),7,100);toc
 ans  =

   0.00104
```

```
-->[lambda,x1,k,n_erro] = Potencia_deslocada_inversa(A,x0,10^(-10),2,100)
 n_erro  =

   6.507D-12

 k  =

   10.

 x1  =

   0.9809625
 -0.1674465
 -0.0983581

 lambda  =

   1.7290369
```

```
-->[lambda,x1,k,n_erro] = Potencia_deslocada_inversa_Ray(A,x0,10^(-10),2,100)
 n_erro  =

   1.178D-16

 k  =

   2.

 x1  =

   0.9809625
 -0.1674465
 -0.0983581

 lambda  =

   1.7290369
```

```
-->tic;[lambda,x1,k,n_erro] = Potencia_deslocada_inversa(A,x0,10^(-10),2,100);toc
 ans  =

   0.002632


-->tic;[lambda,x1,k,n_erro] = Potencia_deslocada_inversa_Ray(A,x0,10^(-10),2,100);toc
 ans  =

   0.000662
```

Ambos os algorítmos encontraram os autovalores $\lambda_1 = 10,488499$, $\lambda_2 = 6,7824639$ e $\lambda_3 = 1,7290369$ que, defato, estão corretos. Porém, entre esses algorítmos ouve uma diferença de tempo de execução significativa. O algorítmo com iteração de Rayleigh foi mais rápido e, além disso, atingiu a mesma precisão em menos iterações.

Questão 6

Acho que novamente vale voltar a conclusão tirada com o Teste 4. A partir do teste pude concluir que para que se considere de forma correta a convergência, deve-se calcular não a norma de $x_{k+1} - x_k$, mas sim a norma de $|x_{k+1}| - |x_k|$, onde $|v|$ significa o valor absoluto de cada uma das coordenadas do vetor v. Isso ocorre pois pode ser que os vetores da iteração caiam no mesmo autoespaço porem com sentidos opostos.