CCI-22 2017
Lab6 Eigenvalues

26 de abril de 2018

.

1 Implementação (AUTOTEST):

Implementar as seguintes funções em MATLAB (cada uma em um arquivo .m separado).:

1.1 [2 pt]: `[x,lambda,niter]=invpowereig(A,maxiter)`

Dada uma matriz $\mathbf{A}$, real, quadrada e simétrica, utiliza o método inverso da potência para encontrar $\mathbf{x}$, o maior autovetor de $\mathbf{A}$, e λ, o maior autovalor de $\mathbf{A}$. Também é retornado o número de iterações `niter`.

- O parâmetro `maxiter` define o número máximo de iterações. Se alcançar este número de iterações, pare o algoritmo e retorne a solução corrente, e `niter = maxiter`. Este parâmetro é importante para evitar loop infinito nos testes e permitir testes com entradas que não permitem convergência.
- Atenção: meça a diferença entre a iteração corrente e a anterior para ambos $\mathbf{x}$ e λ, e não pare até que ambos sejam estabilizados.
- Como chute inicial, utilize $\mathbf{x} = \mathbf{1}$ e $\lambda = 1$, onde $\mathbf{1}$ é um vetor de uns de tamanho apropriado. Dica: o comando matlab `ones(l,1)` gera um vetor coluna com l linhas, com todos os elementos setados com o valor 1.
- Há muitas mensagens de erro sobre matrizes malcondicionadas na inversão (o mesmo se tentar escapar da inversão resolvendo um sistema linear). Isto é um sinal que o algoritmo não é robusto, e uma das razões pelas quais ele não é usado na prática.

1.2 [2.5 pt] `[Q,R] = hqr(A)`

Dada uma matriz $\mathbf{A}$, real, quadrada e simétrica, decomponha-a na forma $\mathbf{A}=\mathbf{Q} \cdot \mathbf{R}$ (decomposição QR), onde $\mathbf{Q}$ é ortonormal e $\mathbf{R}$ é triangular superior, através da aplicação de transformações de householder.

- É fornecida uma função para calcular a transformação de householder que zera os termos embaixo da diagonal para uma determinada coluna. Também um demo que zera elementos da primeira coluna.

Não utilize ortogonalização de Gran-Schmidt para implementar a decomposição. É um método com problemas numéricos. Além disso, o resultado mesmo quando correto, será diferente, e os testes não passarão. Um propósito do exercício é aprender a usar a transformação de Householder.

1.3 [2.5 pt] `[D,QQ,niter]=eigenqr(A,maxiter)`

Dada uma matriz $\mathbf{A}$, real, quadrada e simétrica, utilize o método iterativo com decomposição QR para decompor a matriz na forma $\mathbf{A}{=}\mathbf{QQ} \cdot \mathbf{D} \cdot \mathbf{QQ}^{\mathrm{T}}$, onde:

- $\mathbf{D}$ é diagonal
- A diagonal de $\mathbf{D}$ contém os autovalores de $\mathbf{A}$
- $\mathbf{QQ}$ contém, em suas colunas, os autovetores de $\mathbf{A}$.

Ou seja, o resultado corresponde a diagonalizar a matriz $\mathbf{A}$. Também é retornado o número de iterações `niter`.

- O parâmetro `maxiter` define o número máximo de iterações. Se alcançar este número de iterações, pare o algoritmo e retorne a solução corrente, e `niter = maxiter`. Este parâmetro é importante para evitar loop infinito nos testes e permitir testes com entradas que não permitem convergência.
- Atenção: a ordem dos autovalores e autovetores do resultado não precisa ser a mesma ordem dos mesmos autovalores e autovetores retornados pela função do matlab `eig()`. Além disso, o sinal dos autovetores pode ser trocado.
- Atenção: se a matriz de entrada não tem autovalores reais, o método pode convergir para valores que não são os autovalores. Isto é um problema do método em si, não da sua implementação.
- condição de parada: $\left|a_{ij}^{k} - a_{ij}^{k-1}\right| < 10^{-10}$, onde a_{ij}^{k} representa cada elemento de $\mathbf{A}$ na iteração k, e i e j percorrem as linhas e colunas de $\mathbf{A}$. Ou seja, pare quando a iteração não modificar mais nenhum dos elementos de $\mathbf{A}$. Para comparar $\mathbf{A}$ da iteração atual com o $\mathbf{A}$ da iteração anterior, veja a dica na seção 4. Atenção ao fato que a tolerância é 10^{-10}.

1.4 [1 pt] `proots = polyRootCompanion(p)`

Dado um vetor que representa um polinômio em no formato de `polyval`, e dado que este polinômio possui apenas raízes reais, retorne as raizes deste polinômio, em ordem crescente, utilizando a sua função `eigenqr()` para calcular os autovalores da matriz companheira deste polinômio.

- use a função `compan` para montar a matriz companheira.
- use a função `sort` para ordenar um vetor.

2 Análise

2.1 [1pt] `eigenqr` sem simetria?

Note que uma matriz companheira não é simétrica. Apenas matrizes simétricas podem sempre ser diagonalizadas, e `eigenqr` gera uma diagonalização da matriz. E os nossos algoritmos sempre supuseram matrizes reais e simétricas. Mostre com um exemplo numérico:

1. como é $\mathbf{D}$ quando aplicamos `eigenqr` na matriz companheira? Continua diagonal?
2. os autovalores da matriz companheira continuam na diagonal de $\mathbf{D}$?
3. os autovetores da matriz companheira continuam nas colunas de $\mathbf{QQ}$?

Mostre com um exemplo numérico e preencha uma tabela similar à tabela abaixo com as respostas objetivas as perguntas acima:

	sim	não
1		
2		
3		

2.2 [1pt] A companheira está complexada?

Mostre com um exemplo numérico o que acontece se o polinômio tiver raizes complexas. Os nossos algoritmos sempre supuseram matrizes reais e simétricas.

1. `eigenqr` continua encontrando as raizes?
2. a função `eig` do matlab, ainda consegue encontrar as raízes?

Mostre com um exemplo numérico e preencha uma tabela similar à tabela abaixo com as respostas objetivas as perguntas acima:

	sim	não
1		
2		

- Note: Mesmo com raízes complexas, os coeficientes são reais e portanto os elementos da matriz companheira são reais. Mas, nesse caso, a matriz companheira possui autovalores complexos.
- Dica: Use `poly` para encontrar um polinômio dado um vetor com as suas raízes, e defina algumas raizes complexas. Ou gere um polinomio aleatorio e use `roots` para checar que ele possui raizes complexas.

3 Instruções:

- Valem as mesmas regras anteriores sobre o testador automático. Já que o testador está com vocês, qualquer erro, mesmo trivial, que não passe pela correção automática leva a princípio a zero no exercício.
- Os arquivos .m implementados devem ser entregues juntamente com um relatório.

4 Dicas:

- Os seguintes comandos provêm formas convenientes de comparar matrizes e vetores no MATLAB:

  `max(max(abs(A2-A1)))`, com `A1` e `A2` matrizes.

  `max(abs(b2-b1))`, com `b1` e `b2` vetores.

- Caso necessite comparar autovetores encontrados por uma das suas soluções com os encontrados pela função `eig()` ou outras implementações, note que a ordem pode ser diferente, e sinais podem estar trocados. Para facilitar, foi fornecida a função `compEigenVectors` que compara matrizes de autovetores, usando ordenação e uma convenção de sinais para considerar as variações de resposta citadas.