CCI-22 2017
Lab3 Sistemas Lineares Parte 1

15 de março de 2018

A parte 1 vale 8 pontos, porque 2 pontos dela estão na parte 2 que vale 12 pontos. Cada parte vale por uma nota na média.

1 Implementação (AUTOTEST):

Implementar as seguintes funções em MATLAB (cada uma em um arquivo .m separado). Cada uma delas será testada independentemente e automaticamente. O arquivo de teste `testLab3LinSysAluno.m` contem um teste para cada uma delas:

1. [0.5 pt]`x = SolucaoTriangularSuperior(A, b)`: resolve um sistema linear de equações $\mathbf{Ax} = \mathbf{b}$ em que a matriz $\mathbf{A}$ é triangular superior e retorna a solução $\mathbf{x}$. No caso, considera-se que $\mathbf{A} \in \mathbb{R}^{n \times n}, \mathbf{x} \in \mathbb{R}^{n \times 1}$ e $\mathbf{b} \in \mathbb{R}^{n \times 1}$ (note que $\mathbf{x}$ e $\mathbf{b}$ são vetores coluna).

2. [2 pt]`[Ap, bp] = EliminacaoGauss(A, b)`: executa Eliminação de Gauss **com pivoteamento parcial** no sistema linear de equações $\mathbf{Ax} = \mathbf{b}$ e retorna o sistema resultante $\mathbf{A}_p\mathbf{x} = \mathbf{b}_p$, em que $\mathbf{A}_p$ é triangular superior.

3. [0.5 pt]`x = SolucaoTriangularInferior(A, b)`: resolve um sistema linear de equações $\mathbf{Ax} = \mathbf{b}$ em que a matriz $\mathbf{A}$ é triangular inferior e retorna a solução $\mathbf{x}$.

4. [2.5 pt]`[L, U, P] = DecomposicaoLU(A)`: realiza Decomposição LU **com pivoteamento parcial** da matriz $\mathbf{A}$ e retorna as matrizes resultantes, em que $\mathbf{L}$ é triangular inferior, $\mathbf{U}$ é triangular superior e $\mathbf{P}$ é a matriz de permutações devido ao pivoteamento parcial.

5. [0.5 pt]`x = SolucaoLU(L, U, P, b)`: soluciona o sistema linear de equações $\mathbf{LU} = \mathbf{Pb}$ através da solução sucessiva dos sistemas $\mathbf{Ly} = \mathbf{Pb}$ e $\mathbf{Ux} = \mathbf{y}$. $\mathbf{LU} = \mathbf{Pb}$ é o sistema resultante da Decomposição LU com pivoteamento parcial aplicada a um sistema $\mathbf{Ax} = \mathbf{b}$. O retorno da função é a solução $\mathbf{x}$.

2 [2 pt] Análise

Verificar o ganho de desempenho obtido com decomposição LU no caso de solução de vários sistemas com mesma matriz A , onde apenas b varia. Para isso, resolva 100 sistemas

lineares de dimensão 100 utilizando ambos os métodos implementados: Eliminação de Gauss e Decomposição LU. Para cada método, plote graficamente como o tempo total de execução (considerando desde a solução do primeiro sistema) varia com o número de sistemas resolvidos. Compare os resultados obtidos.

- Não incluir os tempos necessários para construir A e b
- Incluir o tempo necessário para realizar a decomposição na solução do primeiro sistema usando Decomposição LU.

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.
- A primeira etapa do processo de correção consistirá em submeter as funções implementadas a vários casos de teste de forma automatizada. Assim, os cabeçalhos das funções devem ser seguidos **rigorosamente**. Arquivos .m com os nomes destas funções e os cabeçalhos já implementados foram fornecidos juntamente com este roteiro. Dê preferência a implementar seu laboratório a partir destes arquivos .m fornecidos para evitar erros.
- **Não** é permitido o uso de funções ou comandos prontos do MATLAB que realizem toda a funcionalidade atribuída a uma certa função: `x = A \ b`, `x = linsolve(A, b)` e `[L, U, P] = lu(A)` etc. Entretanto, o uso destas funções para verificação das implementações realizadas é encorajado. Em caso de dúvida quanto à permissão de uso de alguma função ou comando, recomenda-se consultar o professor.
- Não é necessário se preocupar com verificação dos dados de entrada: assuma que as dimensões de A e b são compatíveis, que a matriz A é não singular e que os sistemas usados são bem condicionados. Para funções que esperam matrizes triangulares, pode assumir que a matriz de entrada é realmente triangular.

4 Dicas:

- A correção automática não passa se as suas funções não esperam matrizes de dimensões apropriadas. Não é necessário checar o formato da entrada, pode supor o formato correto, mas o ponto é: *precisa funcionar com o formato correto.* Por exemplo, o vetor b em um sistema linear $Ax = b$ é um vetor coluna, não vetor linha. Se o seu programa funciona com entrada vetor linha e não com vetor coluna, os testes falharão.
- A correção automática também não passa se a saída não possuir as dimensões corretas. Se a saída for um vetor coluna, um vetor linha com os mesmos elementos não passa. Há uma mensagem de erro específica para este caso.

- Para criar gráficos com alta qualidade em formato PNG para inclusão em arquivos do Microsoft Word, utilize o comando: `print -dpng -r300 grafico.png`.

- Se utilizar LaTeX, dê preferência para incluir gráficos em formato vetorizado. No Linux, utilizando `pdflatex`, você pode gerar um gráfico em formato EPS usando "`print -depsc2 grafico.eps`" e depois convertê-lo para PDF usando o comando de terminal "`epstopdf grafico.eps`". O arquivo PDF é aceito pelo `pdflatex`.

- Para quem não conhece latex, sugiro o lyx (www.lyx.org). Ainda se pode incluir código latex diretamente, mas a maioria das necessidades já está coberta no nível mais alto. Provavelmente não será mais necessário usar o latex puro (ótimo mas muito 1980!) Em lyx, também gráficos vetorizados são preferidos (afinal, o latex está na camada inferior) mas arquivos .eps são diretamente reconhecidos e o lyx lida com a geração do pdf final.

- Utilize `A = rand(N)` e `b = rand(N,1)` para gerar sistemas lineares de equações rapidamente, em que N é a dimensão do sistema. Como o sistema é gerado aleatoriamente, pode-se obter um sistema mal condicionado. Assim, verifique o número de condição de A com `cond(A,Inf)` antes de usar o sistema.

- As seguintes funções e comandos do MATLAB podem ser úteis para verificar suas implementações:

  `x = A \ b`: resolve o sistema linear de equações $Ax = b$ e retorna a solução x .

  `[L, U, P] = lu(A)`: realiza Decomposição LU seguindo o mesmo formato pedido para a função `DecomposicaoLU`.

- Submeta suas funções a vários casos de teste e compare com os resultados obtidos usando funções e comandos prontos do MATLAB. Devido a imprecisões numéricas, os resultados podem diferir um pouco, porém espera-se que as diferenças sejam bem pequenas.

- 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.

  - Cuidado: errar a ordem do `abs` e do `max` é um erro comum! `max(abs())` é diferente de `abs(max())` !

- Diagonal de matrix: um loop lendo A(i,i) funciona, mas existe um comando que já retorna a diagonal de uma matriz: `diag(A)`

- A ordem de implementação das funções foi definida para que a implementação de uma função possa aproveitar as funções já implementadas. Aproveite isso para poupar trabalho!

- Todas estas funções serão utilizadas na segunda parte do laboratório.

4.1 Somar linhas e/ou colunas: use `sum()`

Na discussão abaixo, A é uma matriz; v é um vetor.

`sum(v)` retorna a soma dos elementos do vetor

`sum(A)` ou `sum(A')`, uma delas retorna um vetor com a soma das colunas, a outra um vetor com a soma das linhas. Experimentem pra ver qual eh qual.

Muito mais fácil que fazer um loop! Associando a isso o mecanismo de indexação dos elementos de uma matriz, fica mais fácil ainda, e.g. para somar parte da linha i de uma matriz A de tamanho n, apenas a parte depois da diagonal: `sum(A(i,i+1:n))`

E talvez nem sequer precise de um caso especial para a ultima linha, pois no caso `i=n`

```
A(n,n+1:n) %note o indice invalido para as colunas.  n+1 não é coluna válida
```

Retorna uma matriz vazia (Empty matrix), que podemos representar por `[]` no matlab. Como `sum( [] )` de uma empty matriz eh zero, facilita implementar os criterios de linhas e sassenfeld.

Além de ser mais fácil de implementar, usar a indexação de linhas/colunas com o operador inicio:end é mais eficiente também. Lembre-se: sempre evite loops explícitos em matlab.