Cálculo Numérico – SME0104 – ICMC-USP

Lista 1: MATLAB e Ponto Flutuante

Parte 1: Conceitos Básicos de MATLAB

Lembrete: dúvidas sobre as funções do MATLAB use o comando help.

1. Qual a resposta da sequência de comandos:

>‌> u = [1, 2, 3];

>‌> v = [4; 2; 1];

>‌> u’\*v’

Resposta:

Ans = 11

1. Dada sequência de comandos:

>‌> A = [1.1 2.2 3.3; 4.4 5.5 6.6];

>‌> b = [10, 20];B

>‌> B = ones(2);

Resposta:

B =

1 1

1. 1

B= ones(2), gera todos os elementos da matriz igual a 1

Diga se é verdadeiro ou falso:

* + 1. >‌> c = [A,b] não gera erro. Falso, horizontal dimension mismatch(2x3 vc 1x2)
    2. >‌> c = [A’,b] não gera erro. Falso, horizontal dimension mismatch(3x2 vc 1x2)
    3. >‌> size([A;b’]) gera erro. Verdadeiro, pois da um erro de dimensions mismatch (2X3 vs 2x1)
    4. (d) >‌> size([A,b’]) gera erro. Falso, da verdadeiro . Ans = 2 4
    5. >‌> size([A,b’]) fornece ans = 2 4. verdadeiro
    6. >‌> size([A,b’]) não gera erro. verdadeiro
    7. >‌> c = A\*B não gera erro. Falso, gera erro “error: operator \*: nonconformant arguments (op1 is 2x3, op2 is 2x2)”
    8. >‌> c = A\*B - B\*A fornece ans = 0. Falso, da erro e o mesmo do item anterior

(i) >‌> B\*A fornece ans = 5.5 7.7 9.9. Verdadeiro max

* + 1. >‌> max(A) fornece ans = 4.4 5.5 6.6. Verdadeiro
    2. >‌> max(max(A)) fornece ans = 2 2 2. Falso. Ans = 6.6
    3. >‌> c = A(:,1)\*b não gera erro. Verdadeiro. Ans = c = 11 22

44 88

(m) >‌> c = A(3,:)\*b não gera erro.falso = error: A(3,\_): but A has size 2x3

1. Execute os seguintes comandos:
   * 1. Declare a matriz A = [2 10 7 6; 3 12 25 9].
     2. Altere o elemento A(2,1) para 18.
     3. Acrescente uma terceira linha a matriz com os elementos 30 21 19 1.
     4. Defina o elemento A(2,3) como -16.
     5. Defina uma matriz B que contenha as duas primeira linhas da matriz A e as colunas de 2 a 4.
2. Criar um vetor com componentes ímpares entre 31 e 75.
3. Seja x = [2 5 1 6].
   * 1. Some 16 a cada elemento.
     2. Some 3 apenas para as componentes com índice ímpar.
     3. Calcule a raiz quadrada de cada elemento.(d) Calcule o quadrado de cada elemento.
4. Seja x = [3 2 6 8]’ e y = [4 1 3 5]’ (vetores colunas).
   * 1. Some x e y.
     2. Eleve cada elemento de x a uma potência dada pelo correspondente elemento de y.
     3. Divida cada elemento de y pelo correspondente elemento de x.
     4. Multiplique cada elemento de x pelo correspondente elemento de y, chamando o resultado de z.
     5. Some todos os elemento de z e guarde-o em uma variável w. (f) Calcule x’\*y - w e interprete o resultado.
5. Crie um vetor x=1:10 e calcule a soma de todos os elementos.
6. Crie uma matriz de dimensão 4 × 4 e a chame de A. (Sugestão: crie uma matriz com números aleatórios.)
   * 1. apague a segunda linha de A;
     2. apague a terceira linha de A;
7. Construir uma matriz 3 × 5, em que a lei de formação da primeira linha é seno, a segunda é cosseno e a terceira é raiz quadrada. Utilize para cada linha o vetor v = 0:0.25:1.
8. Qual o resultados dos seguintes comandos:
   1. abs(-3.51);
   2. sign(-3.51);
   3. ceil(8.73);
   4. floor(8.73);
   5. round(8.73);
   6. lcm(12,8);
   7. gcd(12,8);
   8. rem(11,6);

Parte 2: Scripts e Funções no MATLAB

1. Dada a sequência de comandos abaixo, diga qual a saída:

n = 3; i = n; j = 1;

A = (n-1)\*ones(n) - eye(n); while true

if i == 0

break;

end

if i == j

A(i,j) = A(i,j) - i; else

A(i,j) = i + j;

end

i = i - 1; j = n - i;

end A

1. Considere as seguintes funções:

function M = f(A, a, b, k) M = A; % M recebe uma cópia de A if k

M(a,:) = A(b,:);

M(b,:) = A(a,:); else

M(:,a) = A(:,b);

M(:,b) = A(:,a);

end end% fim da função function v = g(B)

[m n] = size(B); v = zeros(1,n); for i=1:n v(i) = sum(B(:, i));

end

end % fim da função function M = h(A) M = A; v = g(A); n = length(v); for i=1:n-1 for j=1:n-i if v(j) > v(j+1) t = v(j); v(j) = v(j+1); v(j+1) = t;

M = f(M,j,j+1,0); end

end

end

end % fim da função

Diga se as afirmações abaixo são verdadeiras ou falsas:

1. A função f, se k = 0, devolve uma matriz M igual a A só que com as colunas a e b, se existirem, trocadas.
2. A função g retorna o valor da soma de todos os elementos da matriz B.
3. >‌> T = [1 2 3; 4 5 6; 7 8 9];

>‌> f(T,2,1,1)

>‌> ans =

4 5 6

1 2 3

7 8 9

1. >‌> T = [9 3 7; 1 0 2; 6 4 8];

>‌> g(A)

>‌> ans =

19 3 18

1. >‌> A = [0 2 5; 4 1 2; 7 3 1];

>‌> B= f(A,2,3,1);

>‌> g(B)

>‌> ans =

11 8 6

1. >‌> G = [2 3 7; 4 6 2; 1 0 8];

>‌> G = f(G,1,3,0);

>‌> f(G,3,2,1)

>‌> ans =

* 1. 3 2
  2. 0 1

2 6 4

1. >‌> A=[9 3 7; 1 0 2; 6 4 8];

>‌> h(A’)

>‌> ans =

* 1. 6 9
  2. 8 7

0 4 3

1. >‌> A=[184;510;726];

>‌> B = h(A);

>‌> g(B);

>‌> ans =

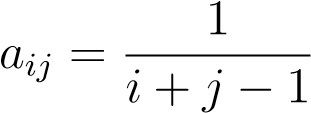
10 11 13

1. A função h permuta as colunas da matriz de tal maneira que fiquem ordenadas com soma (da coluna) crescente.
2. Dada a *função* que calcula a soma de dois vetores de mesma dimensão, identifique e corrija os erros.

function z = Soma z = ones(n); for i=1:n z(i) = x(i) + y;

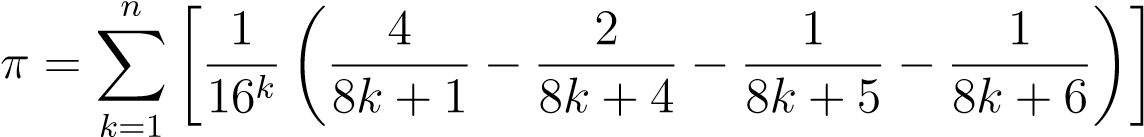
end end

1. Indique qual será a saída para as chamadas f4(1,-5,6) e m4([1:3:100]) das funções:
   1. function r1=f4(a,b,c) r1 = [−b+sqrt(b2−4∗a∗c)/(2∗a) −b−sqrt(b2−4∗a∗c)/(2∗a)]; end
   2. function y = m4(x) y = sum(x)/length(x); end
2. Faça uma *função* que receba uma matriz *A* de ordem *n* e retorne a submatriz formada pelas primeira e penúltina linhas e segunda e última colunas.
3. Faça um *script* em MATLAB que leia uma matriz do R*n* e em seguida verifique se a primeira e a última linha são ortogonais. A saída deve ser uma mensagem ORTOGONAIS em caso afirmativo e NAO ORTOGONAIS, caso contrário.
4. Implemente em MATLAB um programa que faça a ordenação dos elementos de um vetor na ordem crescente. E em seguida:
   * + Compare o resultado obtido com a função sort(v).
     + Usando os comandos tic e toc para calcular o tempo de execução, monte um gráfico (no. de elementos×tempo), usando a função sort(v) e seu programa. • Justifique se seu programa é mais eficiente do que o comando sort(v). Use vetores com 10*,*50*,*100*,*150*,*200 elementos.
5. Faça um *script* que leia as medidas dos três lados de um triângulo e o classifique quanto aos lados (equilátero, isósceles ou escaleno) e quanto aos ângulos (acutângulo, rectângulo ou obtusângulo). Verifique previamente se é possível formar um triângulo com as medidas dadas.
6. Dado um vetor *t*, de comprimento *n*, faça *funções* que calculem as expressões:
   * 1. *ln*(2 + *t* + *t*2);
     2. *et*(1+cos(3*t*));
     3. cos2(*t*) + sen2 (*t*);
     4. tan−1(1);
     5. sec2(*t*) + cot(*t*) − 1.
7. Faça um *script* que determine o maior entre *N* números. A condição de parada é a entrada de um valor 0, ou seja, o algoritmo deve ficar calculando o maior até que a entrada seja igual a 0 (ZERO).
8. Faça um *função* em MATLAB que leia um número inteiro positivo e retorne 0 (falso) caso esse número não seja *perfeito* e 1 (verdadeiro) caso contrário. Um número inteiro é dito perfeito quando ele é igual à soma dos seus divisores próprios, exceto ele mesmo (isto é, o número 6 possui divisores 1, 2, 3 e 6, portanto 1 + 2 + 3 = 6; 6 é um número perfeito – o número 8 possui divisores 1, 2, 4 e 8, portanto 1 + 2 + 4 = 7; 8 não é um número perfeito).
9. Faça uma *função* que receba um número inteiro natural e que devolva um vetor com os divisores desse número. Se o número não for inteiro, a função deve considerar só a parte inteira desse número. Se o número for inferior a um, a função deve devolver um vetor vazio.
10. Escreva uma *função* que retorne 1 se um número for primo e 0 em caso contrário.
11. Implemente em MATLAB uma *função* que receba um inteiro *n* e que retorne uma *matriz de Hilbert* de ordem *n* × *n*. Definição: Uma matriz de Hilbert é uma matriz cujos coeficientes são definidos por:



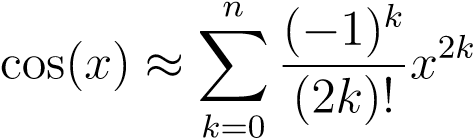
Dica: Compare o resultado obtido com a função hilb(n).

1. Implemente em MATLAB a fórmula de *Bailey-Borwein-Plouffe (BBP)* para computar o número *π*. A fórmula é dada por:



Sua função deve receber *n* e retornar o valor aproximado de *π* (use format long).

1. Uma aproximação de cos(*x*) pode ser feita utilizando a seguinte série:



Tomando *n* = 10, implemente em MATLAB uma *função* que receba de entrada um número *x* ∈R em radianos e calcule o valor aproximado de cos(*x*) utilizando a série acima. Compare o resultado obtido com a função cos(x) através de um gráfico.

Parte 3: Noções Básicas de Ponto Flutuante

Lembrete (informação que vai estar disponível na prova)

Os números de F(*β,t,m,M*) são todos da forma:

*x* = ±(0*.d*1*d*2 *...dt*)*β* × *βe*

com 1 ≤ *d*1 ≤ (*β* −1), 0 ≤ *di* ≤ (*β* −1) para *i* = 2*,...,t* e *m* ≤ *e* ≤ *M*. O sub-índice *β* indica a base na qual o número está representado.

1. Dados os números decimais: *x* = 7*.*125 e *y* = 35*.*27. Escreva-os na base 2.
2. Dados os números binários: *x* = 11*.*0111 e *y* = 111*.*001. Escreva-os na base 10.
3. Considere os números na base 4: *x* = 33, *y* = 0*.*31 e *z* = 21*.*013. Escreva-os na base 5.
4. Considere o sistema F(2*,*5*,*−3*,*2).
   1. Quantos números podemos representar neste sistema?
   2. Qual o menor e maior número em valor absoluto na base 10 que podemos representar neste sistema?
   3. Qual é a precisão de máquina nesse sistema?
5. Faça uma *função* em MATLAB que faça conversão de um número real positivo decimal para binário.
6. Faça uma *função* em MATLAB que faça conversão de um número real positivo binário para decimal.