Atividades cap 1 a cap 5 - apostila análise numérica

Nome: Graziele de Cássia Rodrigues
Matrícula: 21.1.8120

1.1)

Operação	Resposta
12.6987+ 91.3376	104.04
63.2359 X 9.7540	616.80
27.8454.68	9.8745e+78
e^0.15	1.1618
log2(970)	9.9218
a = 957+ 485.37	a = 1442.4
b = (957+ 485.37) X (35.8 + 7.8)	b = 6.2887e+04
c = [(957+ 485.37) X (35.8 + 7.8)] -41.57	c = 3.3399e-200
d = {[(957+ 485.37) X (35.8 + 7.8)] -41.57}x-29.2207	d = -9.7593e-199
cos(90º)	0
sen(45º)	0.7071
√15.57	3.9459

1.2)

Termos		Operação	Resultado
A = [10.7773 -15.5478]	B = [-25.7740 -24.3132]	- A+B - A-B - Multiplica B por A - Divide B por A	-14.997 -39.861 -277.77 378.02 -0.4181 0.6395 -0.6280
A = [32.8 22.3 -32.3 -20.6 45.3 -19.4 46.8 40.2 18.2]	B = [-45.0 11.8 31.3 46.5 -26.5 1.0 6.1 -29..5 5.4]	- Divida elemento A(2,1) por A(1,1) - Divida A(3,1) por A(1,1) - Multiplique a	-0.6280 -18.400 54.095 23.689 68.002 -11.881 -20.174

		divisao de A(2,1) e A(1,1) por todos elementos da linha 1 de A - A*B^T	854.32 -2177.75 -1566.77
		Crie vetor com os valores [100.00 100.1 100.2 …. 1000]	[100:0.1:1000]

1.3)

A)

```
carbono = [0.5 1 0.1; 3 5 0.3; 4 7 0.8];
trator = [20; 30; 5];
resultado = carbono' * trator;
disp(resultado);

#Será necessario comprar 120 unidades de SAE 1020, 205 unidades de SAE 8620 e 15 unidades de SAE 5115
```

B)

```
valores = [5; 20; 15];
valorGasto = resultado'*valores

#Será gasto por mês R$4925,00
```

C)

```
GastoTrator = carbono*valores

#Para cada tipo de trator será gasto respectivamente R$24,00,R$119,5,R$172
```

1.4)

a.

```
y = [1:10];
plot(y);
```

b.

```
x = [-31.4724 -40.5792 37.3013 -41.3376]
y = x.^2 + 4;
plot(x,y);
```

c.

```
x = linspace(-10,10,100);
y = cos(x);
plot(x,y);
```

d.

```
x = [-45.7507 -46.4889 34.2387 -47.0593];
y = exp(x-40);
plot(x,y);
```

1.5)

a)

```
format short
valor = (sqrt(5) + 3)/0.3541;
fprintf("%.4f\n", valor);
erro_arredonamento = (14.7870 - valor)/14.7870
```

b)

```
format bank
valor = (e^3+log(5))/(sin(3)+tan(0.5));
fprintf("%.3f\n", valor);
erro_arredonamento = (31.560 - valor)/31.560
```

c)

```
format bank
valor = log(5)/log(3);
fprintf("%.2f\n", valor);
erro_arredonamento = (1.46 - valor)/1.46
```

d)

```
format short
valor = (3.16)^(1/3);
fprintf("%.1f\n", valor);
erro_arredonamento = (1.5 - valor)/1.5;
```

1.6)

```
function resultado = senoMaclaurin(x, n)
   resultado = 0; % Inicializa o resultado como zero

   for k = 0:n
      termo = ((-1)^k) * (x^(2*k + 1)) / factorial(2*k + 1);
      resultado = resultado + termo;
   end
end
```

a) senoMaclaurin(2,6)
b) sin(2)
c) erro = 2.4694e-08

1.7)

x	$y=\sqrt[3]{x}$	$f(x)=\frac{2}{3}+\frac{x}{3}+\ldots$	$y-f(x)$
0,7	0,887904	0,887926	-0,000022
0,8	0,928317	0,928319	-0,000002
0,9	0,965489	0,965489	0,000000
1,0	1,00000	1,000000	0,000000
1,1	1,03228	1,03228	0,000000

x	$y=\sqrt[3]{x}$	$f(x)=\frac{2}{3}+\frac{x}{3}+\ldots$	$y-f(x)$
10,3	2,17576	1835,77	1833,59
10,8	2,21041	2400,00	-2397,78
11,3	2,24401	3095,42	-3093,17

2.1)

a)

```
#Y=2x2+1, [-8 20], 100 pontos

x = linspace(-8,20,100);
y = 2*x.^2 + 1;
plot(x,y)
title('y = 2*x^2 + 1')
```

b)

```
#Y=cos2(x)+sen2(x), [-12 14], 50 pontos

x = linspace(-12,14,50);
y = (cos(x)).^2 + (sin(x)).^2;
plot(x,y)
title('y = cos(x)^2 + sin(x)^2')
```

c)

```
#Y=e(x^3)+5 , [-20 -2], 10 pontos

x = linspace(-20,2,10);
y = exp(x.^3 + 5);
plot(x,y)
title('y = exp(x^3 + 5)')
```

d)

```
#Y=log3(x^5+2), [10 11], 200 pontos

x = linspace(10,11,200);
y = log(x.^5 + 2)/log(3);
plot(x,y)
title('y = log3(x^5 + 2)')
```

e)

```
#Y=cos(3*x^3+π )+x^3, [1 13], 50 pontos

x = linspace(1,13,50);
y = cos(3*x.^3 + pi) + x.^3;
plot(x,y)
title('y = cos(3*x^3 + pi) + x^3')
```

2.2)

```
n1 = input("Digite o primeiro valor: ");
n2 = input("Digite o segundo valor: ");

if n1 > n2
   disp("O maior número é o primeiro valor\n");
elseif n2 > n1
   disp("O maior número é o segundo valor\n");
else
   disp("Os números são iguais.\n");
endif
```

2.3)

```
function fibonacci_sequence = Fibonacci(n)
   if n <= 0
      disp('O número de termos deve ser um inteiro positivo.');
   endif

   fibonacci_sequence = zeros(1, n);

   if n >= 1
      fibonacci_sequence(1) = 1;
   endif
   if n >= 2
      fibonacci_sequence(2) = 1;
   endif

   for i = 3:n
      fibonacci_sequence(i) = fibonacci_sequence(i - 1) + fibonacci_sequence(i - 2);
   endfor
endfunction


num = input('Digite o número de termos da sequência de Fibonacci: ');
fibonacci_sequence = Fibonacci(num);
disp('Sequência de Fibonacci:');
disp(fibonacci_sequence);
```

2.4)

```
function numeros = ordem()
  n1 = input("Digite o primeiro numero: ");
  n2 = input("Digite o segundo numero: ");
  n3 = input("Digite o terceiro numero: ");
  if n1 == n2 || n1 == n3 || n2 == n3
    disp('Os numeros devem ser diferentes');
    return;
  endif
  disp('Numeros ordenados em ordem crescente');
  # Essa funcao organiza os numeros. Pesquisado no google.
  numeros = sort([n1,n2,n3]);

  n4 = input("Digite o quarto numero numero: ");
  if n4 == n1 || n4 == n2 || n4 == n3
    disp('Os numeros devem ser diferentes');
    return;
  endif
  numeros = [numeros, n4];
  disp('Numeros ordenados em ordem descrescente');
  numeros = sort(numeros, 'descend');
endfunction
```

3.1)

```
function x = sub_sucessiva(L, c)
  n = size(L, 1);
  x(1) = c(1) / L(1, 1);
  for i = 2 : n
    soma = 0;
    for j = 1 : i-1
      soma = soma + (L(i, j) * x(j));
     endfor
     x(i) = (c(i) - soma) / L(i, i);
   endfor
endfunction
```

```
function x = sub_retroativa(U, d)
  n = size(U, 1);
  x = zeros(n,1);
  for i = n-1:-1:1
    soma = 0;
    for j = i+1:n
      soma = soma + U(i, j) * x(j);
    endfor
     x(i) = (d(i) - soma) / U(i, i);
  endfor
endfunction
```

a)

```
A = [2 -3; 0 5];
b = [-1; 4];
x = sub_retroativa(A,b);
```

b)

```
A = [4 -5 2; 0 3 -1; 0 0 2];
b = [1;5;-2];
x =sub_retroativa(A,b);
```

c)

```
A = [4 1 -4 1; 0 -2 8 -3; 0 0 9 -4; 0 0 0 -10];
b = [0.5;7;3;30];
x =sub_sucessiva(A, b);
```

3.3)

A)

```
A = [3 5; -2 4];
# i)
m21 = -A(2,1)/A(1,1)
# ii)
L1 = m21*A(1,:)
# iii)
A(2,:) = L1 + A(2,:)
```

Ela é triangular superior

B)

```
A = [2 1 -3; 4 -2 5; 1 2 -7];
L = eye(3)
L(2,1)=A(2,1)/A(1,1)
L(3,1)=A(3,1)/A(1,1)
A(2,:)=-L(2,1)*A(1,:)+A(2,:)
A(3,:)=-L(3,1)*A(1,:)+A(3,:)
L(3,2)=A(3,2)/A(2,2)
A(3,:)=-L(3,2)*A(2,:)+A(3,:)
```

Elas são matrizes para decomposição LU

3.4)

a)

```
A = [3 1 4; 8 1 2; 2 5 6];
b = [0 0 0]
[A, b, Det, Info] = eliminacao_gauss (3, A, b)
```

b)

```
B = [-10 2 4 6; 5 -8 4 1; 3 9 12 5; -4 0 5 2];
b = [0 0 0 0]
[B, b, Det, Info] = eliminacao_gauss (4, B, b)
```

código gauss

```
function [A, b, Det, Info] = eliminacao_gauss (n, A, b)
  Det = 1;
  Info = 0;
  for j = 1 : n-1
    p = j;
    Amax = abs(A(j,j));
    for k = j+1 : n
      if abs(A(k,j)) > Amax
        Amax = abs(A(k,j));
        p = k;
      endif
    endfor
    if p ~= j
      for k = 1 : n
        t = A(j,k);
        A(j,k) = A(p,k);
        A(p,k) = t;
      endfor
      t = b(j);
      b(j) = b(p);
      b(p) = t;
      Det = -Det;
    endif
    Det = Det * A(j,j);
    # eliminacao de gauss
    if abs(A(j,j)) ~= 0
      r = 1/A(j,j);
      for i = j+1 : n
        Mult = A(i,j) * r;
        A(i,j) = 0;
        for k = j+1 : n
          A(i,k) = A(i,k) - Mult*A(j,k);
        endfor
        b(i) = b(i) - Mult*b(j);
      endfor
    else
      if Info == 0
```

```
      Info = j;
    endif
  endif
 endfor
 Det = Det*A(n,n);
 if Info == 0 && abs(A(n,n)) == 0
  Info = nn = 4;
 endif
endfunction
```

3.6)

método LU

```
function [A, Det, Pivot] = LU (A)
 n = size(A,1);
 for i = 1 : n
  Pivot(i) = i;
 endfor
 Det = 1;
 for j = 1 : n-1
  p = j;
  Amax = abs(A(j,j));
  for k = j+1 : n
   if abs(A(k,j)) > Amax
    Amax = abs(A(k,j));
    p = k;
   endif
  endfor
  if p ~= j
   for k = 1 : n
    t = A(j,k);
    A(j,k) = A(p,k);
    A(p,k) = t;
   endfor
   m = Pivot(j);
   Pivot(j) = Pivot(p);
   Pivot(p) = m;
   Det = -Det;
  endif
  Det = Det * A(j,j);
  if abs(A(j,j)) ~= 0
   r = 1/A(j,j);
   for i = j+1 : n
    Mult = A(i,j) * r;
    A(i,j) = Mult;
    for k = j+1 : n
     A(i,k) = A(i,k) - Mult * A(j,k);
    endfor
   endfor
  endif
 endfor
 Det = Det * A(n,n);
```

```
endfunction
```

método Cholesky

```
function [A, Det, Info] = cholesky (A)
    n = size(A, 1);
    Info  = 0;
    Det = 1;
    for j = 1 : n
      Soma = 0;
      for k = 1 : j - 1
        Soma = Soma + A(j,k)*A(j,k);
      endfor
      t = A(j,j) - Soma;
      if t > 0
        A(j,j) = sqrt(t);
        r = 1/A(j,j);
        Det = Det * t;
      else
        Info = j;
        disp('A matriz nao e definida positiva');
        return;
      endif
      for i = j+1 : n
        Soma = 0;
        for k=1 : j-1
          Soma = Soma + A(i,k) * A(j,k);
        endfor
        A(i,j) = (A(i,j) - Soma) * r;
      endfor
    endfor
endfunction
```

método LDL

```
function [A, Det] = LDL (A)
    n = size(A,1);
    Det = 1;
    for j = 1 : n
       Soma = 0;
       for k = 1 : j-1
         Soma = Soma + A(j,k)^2 * A(k,k);
       endfor
       A(j,j) = A(j,j) - Soma;
       r = 1/A(j,j);
       Det = Det*A(j,j);
       for i = j+1 : n
         Soma  = 0;
         for k = 1 : j-1
```

```
        Soma = Soma + A(i,k)*A(k,k)*A(j,k);
      endfor
      A(i,j) = (A(i,j) - Soma)*r;
    endfor
  endfor
endfunction
```

3.7)

```
function [A, Info] = inversao(A)
  tam = size(A);
  Info = 1;
  if tam(1) ~= tam(2)
    disp('A matriz deve ser quadrada');
    A = [];
    Info = -1;
    return;
  endif
  n = tam(1);
  identidade = eye(n);
  tmpA = zeros(size(A));
  [A,pivot,PdU,Info] = LU(A);
  if Info ~= 0
    disp('O sistema nao tem solucao');
    A = [];
    return;
  endif
  for i = 1 : n
    b = identidade(:,i);
    y = sub_sucessiva(A, b);
    tmpA(i, :) = sub_retroativa(A, y);
  endfor
  A = tmpA;
endfunction
```

4.1)
Jacobi

```
function [x, Iter, Info] = jacobi (n, A, b, Toler, IterMax)
  for i = 1 : n
    x(i) = b(i)/A(i,i);
  endfor
  Iter = 0;
  while 1
    Iter = Iter + 1;
    for j = 1 : n
      Soma = 0;
      for j = 1 : n
        if i ~= j
```

```
        Soma = Soma + A(i,j)*x(j);
      endif
    endfor
    v(i) = (b(i) - Soma)/A(i,i);
  endfor
  NormaNum = 0;
  NormaDen = 0;
  for i = 1 : n
    t = abs(v(i) - x(i));
    if t > NormaNum
      NormaNum = t;
    endif
    if abs(v(i)) > NormaDen
      NormaDen = abs(v(i));
    endif
    x(i) = v(i);
  endfor
  NormaRel = NormaNum/NormaDen;
  disp(Iter);
  disp('x=');
  disp(x);
  disp(NormaRel);
  if NormaRel <= Toler || Iter >= IterMax
    break;
  endif
 endwhile
 if NormaRel <= Toler
  Info = 0;
 else
  Info = 1;
 endif
endfunction
```

Gauss_Seidel

```
function [x, Iter, Info] = gauss(n, A, b, Toler, IterMax)
 for i = 1 : n
  x(i) = b(i)/A(i,i);
 endfor
 Iter = 0;
 while 1
  Iter = Iter + 1;
  NormaNum = 0;
  NormaDen = 0;
  for i = 1 : n
    Soma = 0;
    for j = 1 : n
      if i ~= j
        Soma = Soma + A(i,j)*x(j);
      endif
    endfor
```

```
      v(i) = x(i);
      x(i) = (b(i) - Soma)/A(i,i);
      t = abs(v(i) - x(i));
      if t > NormaNum
        NormaNum = t;
      endif
      if abs(x(i)) > NormaDen
          NormaDen = abs(x(i));
      endif
    endfor
    NormaRel = NormaNum/NormaDen;
    disp(Iter);
    disp('x');
    disp(x);
    disp(NormaRel);
    if NormaRel <= Toler || Iter >= IterMax
      break;
    endif
  endwhile
  if NormaRel <= Toler
    Info = 0;
  else
    Info = 1;
  endif
endfunction
```

a)

```
A = [7 1 5; 1 10 2; 5 20 11];
b = [7;3;5];
disp('Jacobi');
[x, Iter, Info] = jacobi (3, A, b, 10^(-4), 100)
disp('Gauss-seidel');
[x, Iter, CondErro] = gauss(3, A, b, 10^(-4), 100)
```

b)

```
B = [4 1 5 3; 4 5 2 1; 7 6 10 5; 3 9 1 5];
b = [5;7;6;3];
disp('Jacobi');
[x, Iter, Info] = jacobi (4, B, b, 10^(-4), 100)
disp('Gauss-seidel');
[x, Iter, CondErro] = gauss_seidel (4, B, b, 10^(-4), 100)
```

c)

```
C = [5 3 1; 1.5 10 1; 0.7 2.3 2];
b = [10;2;5];
disp('Jacobi');
[x, Iter, Info] = jacobi (3, C, b, 10^(-4), 100)
disp('Gauss-seidel');
```

```
[x, Iter, CondErro] = gauss_seidel (3, C, b, 10^(-4), 100)
```

d)

```
D = [21 12 5; 4 45 13; 9 14 32];
b = [17;11;21];
disp('Jacobi');
[x, Iter, Info] = jacobi (3, D, b, 10^(-4), 100)
disp('Gauss-seidel');
[x, Iter, CondErro] = gauss_seidel (3, D, b, 10^(-4), 100)
```

e)

```
E = [0.3 0.07 0.1; 0.014 0.05 0.0023; 0.09 0.05 0.18];
d =  [0.15; -1.4; -0.31];
disp('Jacobi');
[x, Iter, Info] = jacobi (3, E, b, 10^(-4), 100)
disp('Gauss-seidel');
[x, Iter, CondErro] = gauss_seidel (3, E, b, 10^(-4), 100)
```

5.1)

```
function Info = malcondicionado(A)
  normA = norm(A, 1);
  normInvA = norm(inversa(A), 1);
  numeroCond = normA * normInvA;

  limite = 1 / eps;
  if numeroCond > limite
     Info = 1;
  else
     Info = 0;
  endif
endfunction
```

a)

```
A = [3.25 1.625 1.083 0.8125; 1.625 1.083 0.8125 0.65; 1.083 0.8125 0.65 0.5417; 0.8125
0.65 0.5417 0.4643];
b = [3.520; 4.496; 4.008; 3.49];

if malcondicionado(A) ~= 1
  [x, Iter, Info] = gauss_seidel (size(A, 1), A, b, 0.0001, 1000);
  aux = x.+0.01;
  Dif = abs(x - aux)/aux
else
  disp('O sistema e malcondicionado');
endif
```

b)

```
B = [-3 -24 5 -17; -24 5 -2 4; 5 -2 3 -8; -17 4 -8 1];
b = [-152; 31; 64; 11];

if malcondicionado(B) ~= 1
  x = sol_decomp_LU (B, b);
  aux = x.+0.01;
  Dif = abs(x - aux)/aux
else
  disp('O sistema e malcondicionado');
endif
```

c)

```
C = [3 1.5 1 0.75; 1.5 1 0.75 0.6; 1 0.75 0.6 0.5; 0.75 0.6 0.5 0.43];
b = [32; 16.2; 10.85; 8.16];

if malcondicionado(C) ~= 1
  [x, Iter, Info] = gauss_seidel (size(C, 1), C, b, 0.0001, 100);
  aux = x.+0.01;
  Dif = abs(x - aux)/aux
else
  disp('O sistema e malcondicionado');
endif
```

d)

```
D = [0.01 0.4 0.125 1; 0.4 0.25 2 0.32; 0.125 2 0.5 1.02; 1 0.32 1.02 0.045];
b = [2.0685; -0.085; 2.4425; 0.1424];

if malcondicionado(D) ~= 1
  x = sol_decomp_LU (D, b);
  aux = x.+0.01;
  Dif = abs(x - aux)/aux
else
  disp('O sistema e malcondicionado');
endif
```

e)

```
E = [3 4; -9 -12];
b = [1; 4];

if malcondicionado(E) ~= 1
  [x, Iter, Info] = gauss_seidel (size(E, 1), E, b, 0.0001, 100);
  aux = x.+0.01;
  Dif = abs(x - aux)/aux
else
  disp('O sistema e malcondicionado');
endif
```