UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia

Departamento de Computação

Controle e Servomecanismo

Prática 1 - Revisão

Professor: Prof. Roberto Santos Inoue

Alexandre Strabello, 770076, Engenharia Física

São Carlos, 23 de maio de 2023

1 Execução da Prática

Conforme autorizado pelo professor da disciplina, a presente prática de revisão foi realizada utilizando o MATLAB® ao invés do SCILAB®, decorrente das futuras atividades a serem desenvolvidas em outras disciplinas.

As seções da atividade são dependentes, tal que os elementos criados na primeira parte são utilizados para as demais. Contudo, a representação das mesmas será feita de maneira individualizada, visando tornar a discussão mais organizada.

1.1 Vetores e Matrizes

A construção dos vetores e matrizes foi realizada de acordo com o código apresentado a seguir:

```
%Vetores
x = [1 2 3 4 5];
y = x';
z = randi([0,100],5,1);
%Matrizes
A = randi([0,100],4,4);
B = A';
C = rand(4) + ones(4,4);
%Apresentacao dos resultados
disp('a)')
disp(x)
disp('b)')
disp(A)
disp('c)')
disp(y)
disp('d)')
disp(B)
disp('e)')
disp(z)
disp('f)')
disp(C)
```

Optou-se por criar as matrizes com elementos aleatórios apenas por explorar as funções disponíveis no software. Para a matriz D havia o requisito de que todos os elementos fossem não inteiros superiores a um, de modo que utilizou-se a função $rand()$ para gerar números reais entre 0 e 1, seguido da adição de uma matriz unitária para cumprir o requisito. Os resultados obtidos para essa seção estão apresentados na Figura 1.

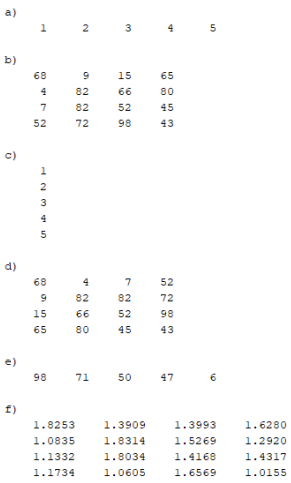

Figura 1: Apresentação dos resultados para criação dos vetores e matrizes

1.2 Formatos de Apresentação

Como as atividades dessa parte consistem apenas na apresentação dos resultados executados na seção 1.1, foi elaborado o código abaixo:

```
for i=1:5
    switch i
        case 1
            %5 digitos
            format short
            disp("Formato: short")
        case 2
            %5 digitos com exponencial
            format shorte
            disp("Formato: shorte")
        case 3
            %15 digitos
            format long
            disp("Formato: long")
        case 4
            %15 digitos com exponencial
            format longe
            disp("Formato: longe")
        otherwise
            %Apresentacao racional
            format rat
            disp("Formato: rat")
    end
    disp("x")
    disp(x)
    disp("A")
    disp(A)
    disp("z")
```

```
    disp(z)
    disp("C")
    disp(C)
    disp("-------------------")
end
```

Decidiu-se utilizar um $switch\,case$ para reduzir a quantidade de linhas do código. Como foi necessário apresentar as mesmas variáveis em cinco formatos distintos, foi construída uma estrutura condicional, onde cada caso apresentava um formato distinto. Os resultados obtidos estão apresentados nas Figuras 2 e 3.

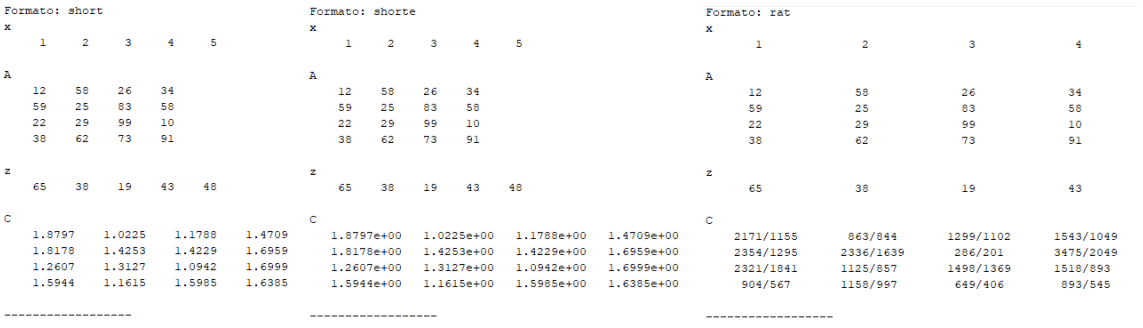

Figura 2: Apresentação das variáveis no formato $short$, $shorte$ e rat.

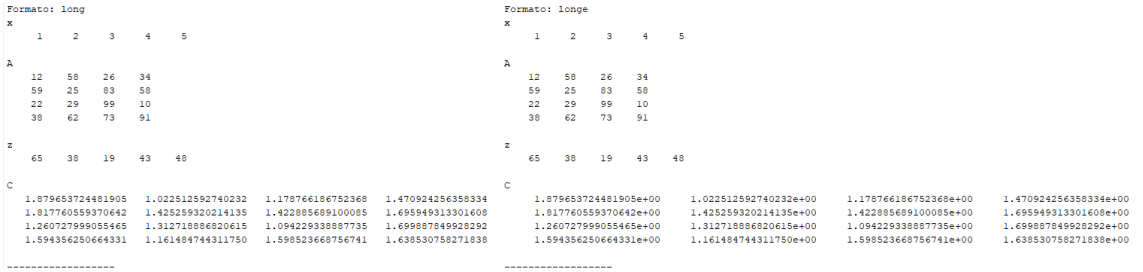

Figura 3: Apresentação das variáveis no formato $long$ e $longe$.

1.3 Números Complexos

Para construção das variáveis com números complexos adicionou-se apenas a componente imaginária i na declaração. O código utilizado é dado a seguir:

```
k = [3+5i, 2-10j];
D = [3 + 5i, 2-10i; 7-13i, 1.7-4i];
%Vetor m dulo de k
mod_k = abs(k);
%Vetor fase de k, representado em radianos
ang_k = angle(k);
%Matriz do m dulo de D
mod_D = abs(D);
%Matriz da fase de D, representado em radianos
```

```
ang_D = angle(D);
%Matriz transposta
E = D.';
%Matriz conjugada transposta
F = D';
%Apresentacao dos resultados
disp('a)')
disp(k)
disp('b)')
disp(D)
disp('c)')
disp(mod_k)
disp('d)')
disp(ang_k)
disp('e)')
disp(mod_D)
disp('f)')
disp(ang_D)
disp('g)')
disp(E)
disp('h)')
disp(F)
```

Não houve regulamento quanto ao formato desejado para os ângulos dos números complexos, de modo que optou-se por deixar no padrão presente no MATLAB®, correspondendo a radianos. Na Figura 4 são apresentadas as variáveis complexas produzidas de acordo com as questões solicitadas na prática.

```
a)
   3.0000 + 5.0000i   2.0000 -10.0000i

b)
   3.0000 + 5.0000i   2.0000 -10.0000i
   7.0000 -13.0000i   1.7000 - 4.0000i

c)
    5.8310   10.1980

d)
    1.0304   -1.3734

e)
    5.8310   10.1980
   14.7648    4.3463

f)
    1.0304   -1.3734
   -1.0769   -1.1689

g)
   3.0000 + 5.0000i   7.0000 -13.0000i
   2.0000 -10.0000i   1.7000 - 4.0000i

h)
   3.0000 - 5.0000i   7.0000 +13.0000i
   2.0000 +10.0000i   1.7000 + 4.0000i
```

Figura 4: Representação de números complexos produzidos no MATLAB®.

1.4 Operações com Matrizes

Nessa parte foram realizadas operações matemáticas utilizando as matrizes construídas na seção 1.1. O código implementado abaixo apresenta cada operação com os nomes op_a, op_b, $op\_c \ldots$, representando os itens $a)$, $b)$, $c) \ldots$ solicitados.

```
format short

op_a = A+C;
disp('a)')
disp(op_a)

op_b = A-C;
disp('b)')
disp(op_b)

op_c = A*C;
disp('c)')
disp(op_c)

op_d = A.*C;
disp('d)')
disp(op_d)

op_e = A^-1;
disp('e)')
disp(op_e)

op_f = A\C;
disp('f)')
disp(op_f)

op_g = A/C;
disp('g)')
disp(op_g)

op_h = A./C;
disp('h)')
disp(op_h)

op_i = exp(x);
disp('i)')
disp(op_i)

op_j = exp(A);
disp('j)')
disp(op_j)
```

```
op_k = x.^4;
disp('k)')
disp(op_k)

op_l = A^2;
disp('l)')
disp(op_l)
```

Como a estratégia utilizada para as matrizes era dada por uma geração aleatória, não é possível comparar todos os resultados diretamente com aqueles apresentados na Figura 1, já que a cada execução do programa os vetores e matrizes são alterados. De toda forma, os resultados das operações realizadas estão presentes na Figura 5.

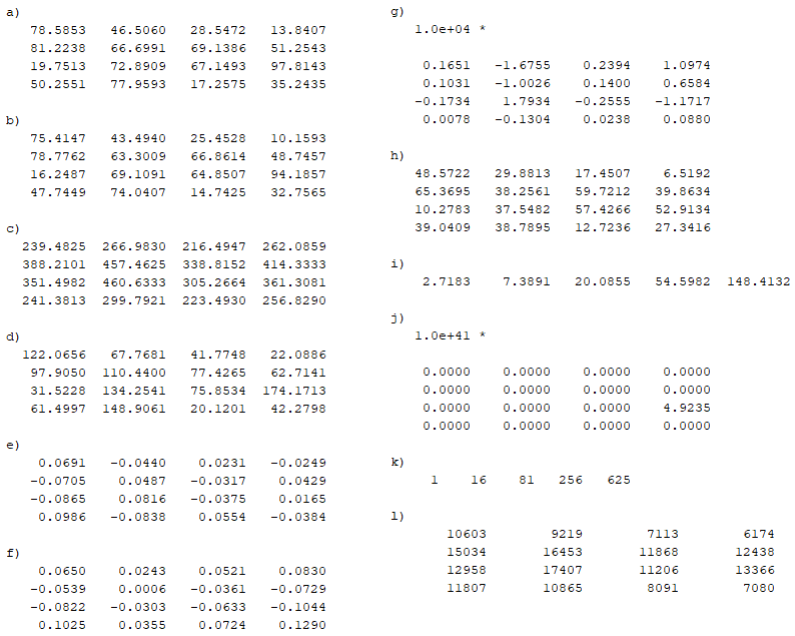

Figura 5: Resultado das operações matemáticas realizadas com os vetores e matrizes.

1.5 Vetores com espaçamento controlado

Nessa parte foram construídos diferentes vetores com diferentes passos. O código desenvolvido é dado a seguir:

```
t1 = linspace(0,10,11);
t2 = linspace(0,10,10/0.1+1);
t3 = linspace(-10,10,5);
t4 = linspace(50,0,50/10+1);
```

Haviam dois diferentes vetores solicitados, um informava o espaçamento, o outro, o tamanho e a condição de que os elementos deveriam ser linearmente espaçados. Em

ambos foi utilizada a função $linspace()$, porém com estratégias distintas. O terceiro argumento informado na função representa a quantidade de elementos do vetor a ser criado, cujos elementos serão espaçados linearmente entre os limites informados, assim, se é desejado criar um vetor com espaçamento fixo previamente informado, basta dividir o intervalo a ser utilizado pelo passo, depois incrementar o valor 1 para adicionar o limite inferior.

Em virtude do tamanho do vetor $t2$, não serão apresentados todos as variáveis produzidas, contudo, na Figura 6 estão apresentados os vetores $t1$ e $t3$, para mostrar que a função $linspace()$ foi suficiente para produzir as variáveis solicitadas.

```
t1
     0     1     2     3     4     5     6     7     8     9    10

t3
   -10    -5     0     5    10
```

Figura 6: Vetores criados com espaçamento fixo.

1.6 Manipulação de matrizes

Para a parte de manipulação de matrizes, foi desenvolvido o código abaixo:

```
A2 = A(3:4,1:2);
disp('A2')
disp(A2)
A3 = zeros(length(A)+1,length(A));
A3(1:4,1:4) = A;
A3(5,:) = x(1:4);
disp('A3')
disp(A3)
```

A matriz $A2$ foi criada simplesmente selecionando um intervalo na matriz A, enquanto a matriz A3 foi definida inicialmente como uma matriz nula, com o mesmo tamanho da matriz A adicionado de uma linha. Em seguida, foram colocados os elementos solicitados. O resultado obtido é ilustrado na Figura 7.

```
A2
    18    71
    49    76

A3
    77    45    27    12
    80    65    68    50
    18    71    66    96
    49    76    16    34
     1     2     3     4
```

Figura 7: Criação de matrizes com elementos previamente existentes.

1.7 Funções e Gráficos

Decidiu-se reunir as seções envolvendo a criação de funções e a apresentação de resultados em gráficos. A função foi implementada em um arquivo distinto do principal, porém presente no mesmo repositório. O código da mesma é dado a seguir:

```
function [p] = funcTeste(t)
    p = t.^2 + sin(t)+50;
end
```

No programa principal foi chamada a função previamente produzida e realizada a produção do gráfico correspondente. O código implementado é dado a seguir, enquanto o gráfico está ilustrado na Figura 8.

```
p1=funcTeste(t1);
p2=funcTeste(t2);
plot(t1,p1,'r*',t2,p2,'g-')
legend('p1 vs t1','p2 vs t2','Fontsize',20,'Location','northwest')
title('Gr fico Atividade 1')
xlabel('x','FontSize',20)
ylabel('funcTeste(x)','FontSize',20)
ax = gca;
ax.FontSize = 15;
grid on
```

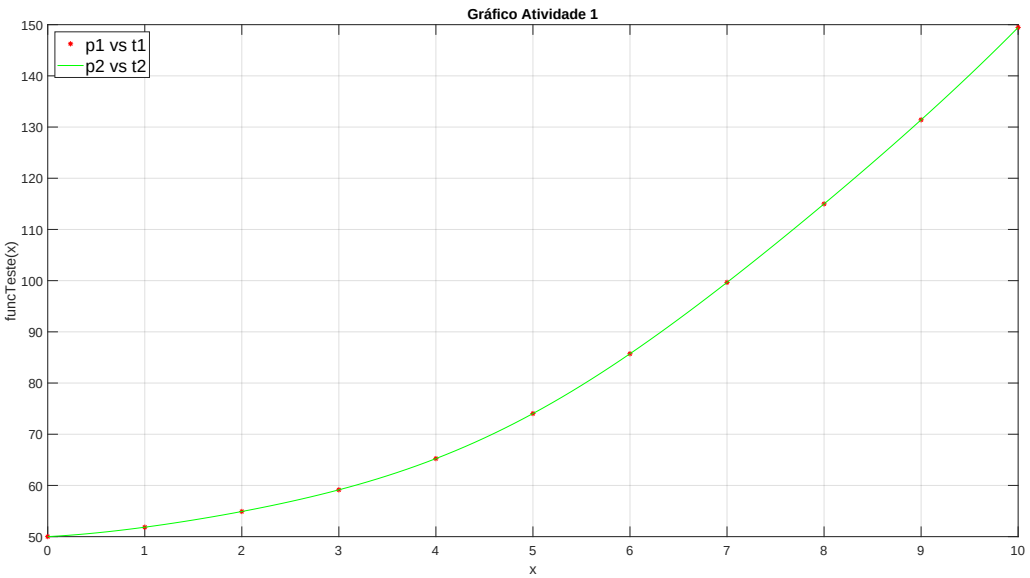

Figura 8: Apresentação da função Teste nos vetores $t1$ e $t2$.