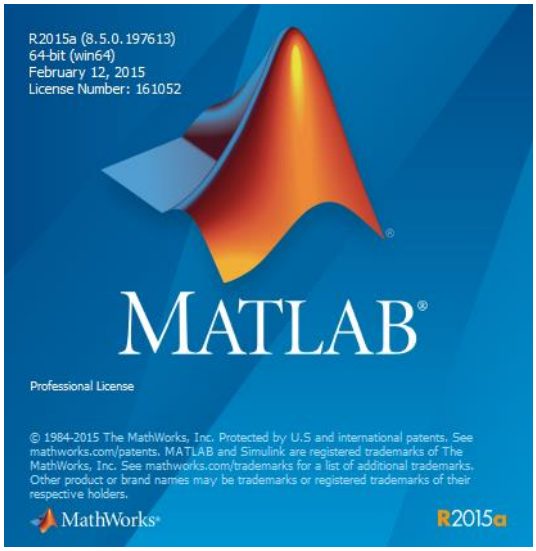
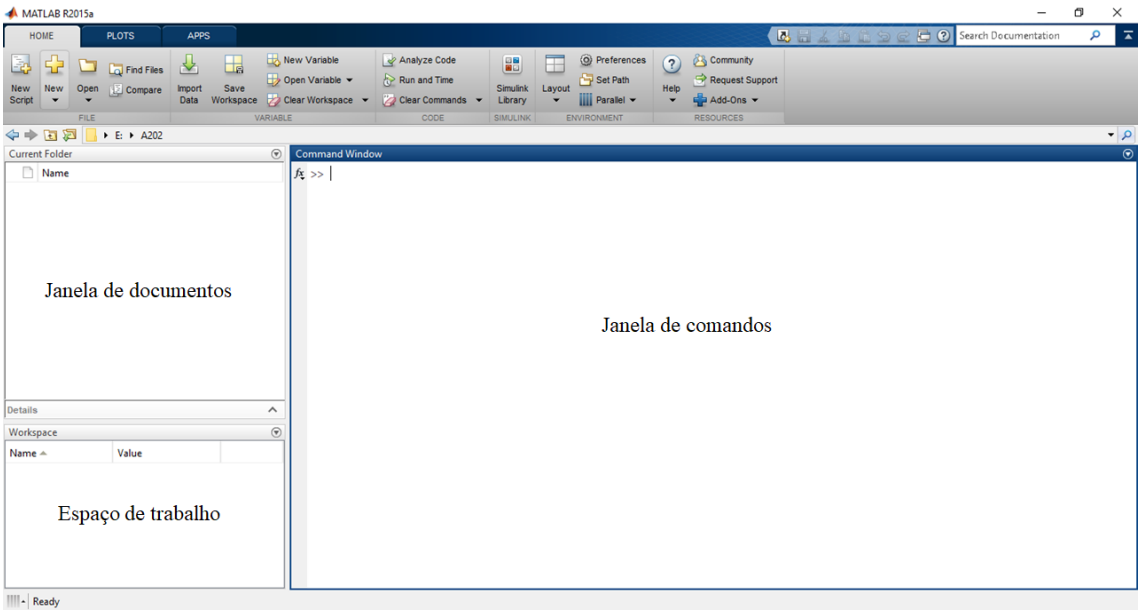
Inatel Instituto Nacional de Telecomunicações	**1º Relatório**	**Turma:** C213 __	**Data:**
	Introdução ao MATLAB		
Nome:			

Introdução ao MATLAB

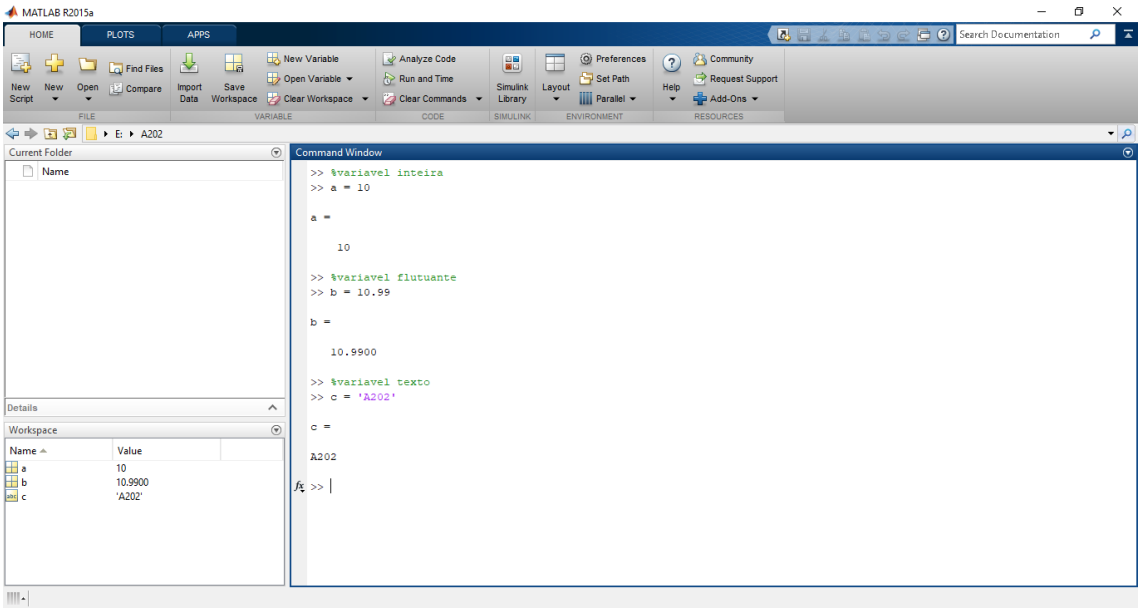
O MATLAB é um ambiente de computação técnico científica para desenvolvimento de sistemas sofisticados e eficientes, utilizado para facilitar os cálculos e simulações. Está presente nos departamentos de engenharia e desenvolvimento de grandes empresas e instituições do país, tais como a Companhia Vale do Rio Doce, Embraer, Renault do Brasil, entre outras.



O ambiente de desenvolvimento é apresentado na figura abaixo.



Na janela de documentos é possível navegar pelos arquivos como scripts e simulações já elaboradas e é o ambiente que serão digitados os comandos que serão executados e processados. Já no espaço de trabalho são apresentadas as variáveis criadas na janela de comandos. A figura a seguir apresenta a criação de vários tipos dessas variáveis:



Alguns dos operadores que podem ser utilizados estão apresentados na tabela abaixo:

Função matemática	**Símbolo**	**Exemplo**
Adição	+	soma = 3 + 22
Subtração	-	sub = 54.4 – 16.5
Multiplicação	*	mult = 3.14 * 6
Divisão	/ ou \	div = 19.54/7 ou div = 19.54\7
Potenciação	^	pot = 2^8
Exponencial natural (e^t)	exp( )	exp(10)
Raiz quadrada	sqrt( )	sqrt(25)

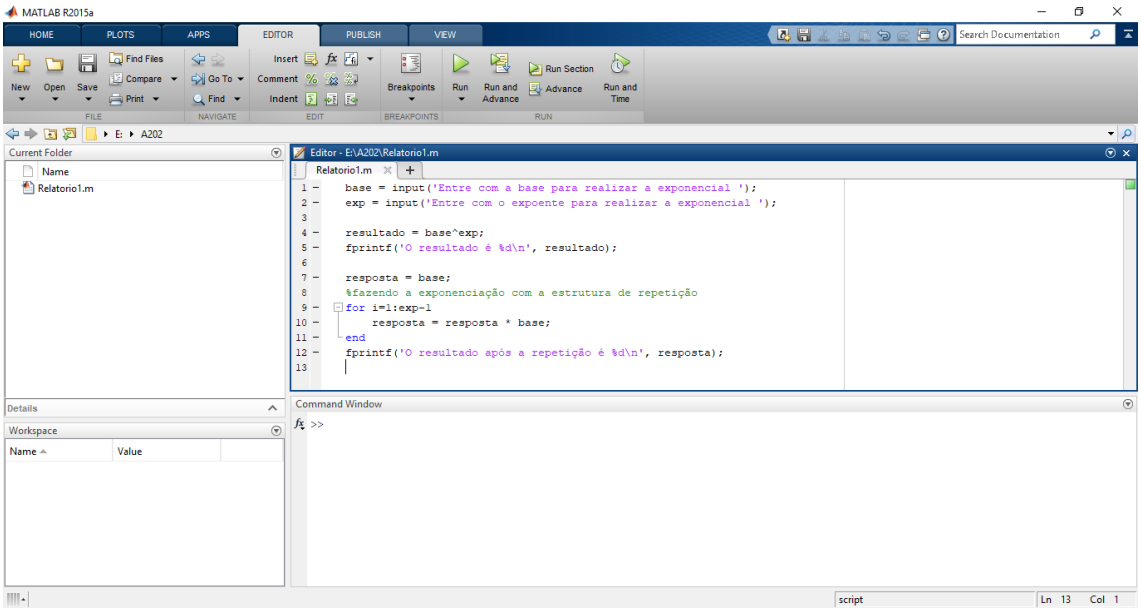
Função trigonométrica	**Descrição**
cos	Cosseno com ângulo em radianos
cosd	Cosseno com ângulo em graus
sin	Seno com ângulo em radianos
sind	Seno com ângulo em radianos
tan	Tangente com ângulo em radianos
tand	Tangente com ângulo em graus

Função Exponencial	**Descrição**
log	Logaritmo natural - ln
log10	Logaritmo na base 10
log2	Logaritmo na base 2

Como fazer um código no MATLAB?

Todo código tem objetivo tornar processos repetitivos mais simples, no MATLAB são realizados *script* que são arquivos com a extensão **.m**. Para adicionar um script basta clicar no ícone presente no canto superior esquerdo, como o apresentado na figura a seguir, ou apertar as teclas **Ctrl + N**, ou digitar ***edit*** na *Command Window*.

Outra aba especifica para escrever os códigos será aberta e nela é possível pedir ao usuário digitar informações, utilizar estruturas de decisão, repetição entre outras. Para realizar a leitura de uma variável é utilizado o comando **input**, e a saída utiliza-se o **fprintf**, como mostrado no exemplo a seguir.



Para utilizar o comando **fprintf** é necessário saber com qual tipo de variável se está utilizando, na tabela abaixo encontra-se as conversões necessárias para elaborar a mensagem de saída. Para mostrar um texto é necessário utilizar aspas simples.

Exemplo 1: `fprintf('A variável tem o valor de %f',a)`

Tipo	Saída	Exemplo
`%d` ou `%i`	Número decimal	-10
`%f`	Número decimal flutuante	25.86
`%e`	Número em notação científica	8.7e+2
`%s`	Cadeia de caracteres	Ola
`%c`	Caractere	A

A estrutura condicional é feita pelos comandos:

```
if condição
   comando
elseif condição
   comando
else
   comando
end
```

Para realizar as condições deve-se utilizar os seguintes operadores:

Operador	Descrição
==	Igualdade
>	Maior
<	Menor
>=	Maior igual
<=	Menor igual
~=	Diferente
&&	E
||	OU

A estrutura de repetição é feita pelo comando:

```
for nome_da_variavel = Valor_Inicial:Valor_Incremento:Valor_Final
   comandos;
end
```

Exemplo1: `for i = 0:0.1:10` → irá de 0 a 10 com passo de 0.1 em 0.1, resultarão em 101 passos dentro da repetição.

Exemplo 2: `for i = 1:0.1:10` → irá de 1 a 10 com passo de 0.1 em 0.1, resultarão em 91 passos dentro da repetição.

Exemplo 3: `for i = 0:2:10` → irá de 0 a 10 com passo de 2 em 2, resultarão em 6 passos dentro da repetição.

Exemplo 4: `for i = 0:10` → irá de 0 a 10 com passo de 1 em 1, resultarão em 11 passos dentro da repetição.

Lembrete: para encerrar as estruturas anteriores deve-se colocar o comando ***end***.

Exercícios

1. Faça um *script* onde o usuário entre com 3 notas e exiba na saída a soma delas.

2. Complemente o *script* anterior exibindo a média aritmética das notas.

3. Para doar sangue é necessário ter no mínimo 18 anos e no máximo 67 anos. Faça um *script* onde o usuário entre com seu nome e sua idade e exiba se ele pode ser um doador ou não.

4. Escreva um *script* para calcular N! (fatorial de N), sendo que o valor inteiro de N é fornecido pelo usuário.

Vetores e matrizes

1) Vetores

1.1) O que é um vetor

Um vetor é uma estrutura de dados unidimensional composta formada por uma sequência de variáveis, sendo todas do mesmo tipo de dados e que possuem um mesmo identificador (nome). Todas as variáveis que formam o vetor são alocadas sequencialmente na memória e em cada posição (índice) encontra-se um valor.

1.2) Como criar um vetor

Os comandos para se criar um vetor podem ser digitados na própria janela de comandos ou então em um *script* criado no editor. Para um vetor linha os elementos são separados por um espaço. Já para um vetor coluna os elementos são separados por ponto e vírgula, sendo que o ponto e vírgula insere uma nova linha no vetor.

Exemplo 1: `A = [1 2 3 4]`, cria um vetor linha com os elementos.

Exemplo 2: `B = [1; 2; 3; 4; 5]`, cria um vetor coluna com os elementos.

```
>> A = [1 2 3 4]
A =

   1   2   3   4
```

```
>> B = [1; 2; 3; 4; 5]
B =

   1
   2
   3
   4
   5
```

Para aplicações futuras em *scripts* é interessante o conhecimento sobre comandos que facilitam na criação de vetores que tenham um tamanho determinado ou que os valores que formam o vetor tenham um incremento conhecido.

Exemplo 3: Suponha a necessidade de um vetor que tenha valores entre 1 e 1.5 com um passo de 0.1 entre cada um dos valores do vetor. A variável pode ser criada com o comando: `C = [1 1.1 1.2 1.3 1.4 1.5].`

Nessa situação, todos os valores do vetor são digitados manualmente. Essa tarefa pode ser trabalhosa e levará mais tempo quanto maior for o tamanho do vetor. Logo, para isso, o vetor pode ser preenchido utilizando a mesma sintaxe da estrutura *for*:

```
Nome_do_vetor = valor_inicial:incremento:valor_final
```

Com isso, o mesmo vetor poderia ser criado com o comando `C = 1:0.1:1.5`, que cria um vetor com elementos entre 1 e 1.5 com incremento de 0.1.

```
>> C = 1:0.1:1.5
C =

    1.0000    1.1000    1.2000    1.3000    1.4000    1.5000
```

Nesses casos não é necessário digitar manualmente todos os valores que formam o vetor, deixando que o próprio *software* faça esse trabalho. Outra forma de se fazer essa tarefa é com a função *linspace*. Esta função é responsável por gerar valores igualmente espaçados entre dois números:

```
>> %linspace(a,b,n) -> gera n numeros uniformemente
>> %distribuidos entre a e b, inclusive
>>
>> D = linspace(1, 1.5, 6)
D =

   1.0000   1.1000   1.2000   1.3000   1.4000   1.5000
```

1.3) Acessando posições do vetor

Um vetor no *Octave* ou no *MatLab* é indexado em 1, ou seja, o primeiro elemento do vetor ocupa a posição 1. Dessa forma os elementos no vetor são dispostos da posição 1 até a posição equivalente ao seu tamanho.

Para acessar o valor em uma posição do vetor utiliza-se o comando: `A(k)`, sendo *k* a posição corresponde no vetor A a qual se deseja saber o valor.

Exemplo 4: Tomando-se por referência o vetor C criado no exemplo 3, pode-se acessar o valor que está na sua posição 3:

```
>> C = 1:0.1:1.5
C =

    1.0000    1.1000    1.2000    1.3000    1.4000    1.5000

>> C(3)
ans =  1.2000
```

Nessa situação, dois erros podem ser cometidos: um deles é tentar acessar a posição 0 do vetor, sendo que essa posição não existe, pois o vetor é indexado em 1. Outro erro é tentar acessar uma posição maior do que o tamanho do vetor. Os dois erros são mostrados a seguir:

```
>> C = 1:0.1:1.5
C =

    1.0000    1.1000    1.2000    1.3000    1.4000    1.5000

>> C(0)
error: C(0): subscripts must be either integers 1 to (2^63)-1 or logicals
>>
>> C(7)
error: C(7): out of bound 6
>>
```

Na primeira situação os dois *softwares* são claros ao dizer que as posições do vetor devem ser acessadas entre 1 e $2^{63} - 1$, que é o tamanho máximo que um vetor pode ter para que seja alocado na memória. No segundo caso, o retorno é que a posição que está tentando ser acessada está fora do limite de 6, que é o tamanho do vetor C.

1.4) Encontrando o maior elemento de um vetor

Para encontrar o maior valor dentro do vetor, duas informações são importantes: primeiro deve-se conhecer a magnitude dos valores que compõe o vetor (principalmente o seu valor mínimo), e, em segundo, o tamanho do vetor deve ser conhecido. O tamanho do vetor pode ser retornado por meio da função *size*.

A função vai retornar quantos elementos tem o vetor na forma de linhas e colunas. Com essas informações também é possível saber se o vetor é um vetor linha ou se é um vetor coluna. A função *size* tem dois retornos, logo deve ser chamada com duas variáveis. Com os comandos:

```
>> A = [1 2 3 4];
>> [linha, coluna] = size(A)
linha =  1
coluna =  4
>>
>> B = [1; 2; 3; 4; 5];
>> [linha, coluna] = size(B)
linha =  5
coluna =  1
```

O vetor é linha se ele tiver uma linha, e é coluna se ele tiver uma única coluna. Para cada tipo de vetor, o seu tamanho é dado pela variável diferente de 1, ou seja, o tamanho de um vetor linha é dado pelo número de colunas e o tamanho de um vetor coluna é dado pelo número de linhas. No exemplo acima, o vetor A tem tamanho 4 e o vetor B, tamanho 5.

Para encontrar o maior valor do vetor, uma variável deve ser criada com um valor inicial. Para a maioria das aplicações o valor inicial dessa variável será 0, pois todos os valores dos vetores serão positivos. Repare que o valor inicial da variável deve ser menor que todos os valores do vetor, por isso a magnitude dos valores deve ser conhecida.

Uma estrutura de repetição percorre todo o vetor, da sua posição inicial até a última e, em cada passo, é verificado se o valor daquela posição é maior do que o valor que está salvo na variável. O trecho de código a seguir, escrito no *Octave*, vai encontrar o maior valor do vetor E, um vetor de 7 elementos entre 0 e 1 criado com valores de forma aleatória:

```
>> E = rand(1,7)
E =

   0.038698   0.361253   0.019093   0.910295   0.504607   0.995939   0.802346

>> [linha, coluna] = size(E)
linha =  1
coluna =  7
>>
>> maior_valor = 0;
>> for k = 1:coluna
if E(k) > maior_valor
maior_valor = E(k);
endif
endfor
>>
>> maior_valor
maior_valor =  0.99594
```

Lembrando que, caso o código fosse escrito no *MatLab*, o fechamento das estruturas *if* e *for* seria apenas com o comando *end*, e não com *endif* e *endfor*.

1.5) Leitura de valores em um vetor

A leitura de informações para um vetor pode se dar em duas situações: uma em que o tamanho do vetor (quantidade de informações) é conhecido e em outras em que essa quantidade não é conhecida. Na segunda opção, a leitura de dados é feita enquanto uma determinada condição *(flag)* for respeitada:

```
%Entrada de dados em um vetor
clear
clc
display('');

%Conhecendo o tamanho do vetor
tam = input('Tamanho do vetor: ');
display('');
for k = 1:tam
  vetor1(k) = input('Digite um valor: ')
endfor

display('');
vetor1

%Não conhecendo o tamanho
k = 1;
num = input('Digite um valor: ');
while num ~= 0
  vetor2(k) = num;
  k = k + 1;
  num = input('Digite um valor: ');
endwhile

display('');
vetor2

```

```
Tamanho do vetor: 4

Digite um valor: 1
Digite um valor: 2
Digite um valor: 3
Digite um valor: 4


vetor1 =

   1   2   3   4

Digite um valor: 1
Digite um valor: 2
Digite um valor: 3
Digite um valor: 4
Digite um valor: 0

vetor2 =

   1   2   3   4
```

Lembrando que, caso o código fosse escrito no *MatLab*, o fechamento das estruturas *for* e *while* seria apenas com o comando *end*, e não com *endfor* e *endwhile*.

2) Matrizes

2.1) O que é uma matriz

Uma matriz é uma estrutura de dados multidimensional que armazena uma sequência de dados, todos do mesmo tipo, em posições consecutivas de memória. O conceito de matrizes é uma extensão do conceito de vetores, sendo que as matrizes são vetores em que cada posição há um outro vetor.

2.2) Como criar uma matriz

Os comandos para se criar uma matriz podem ser digitados na própria janela de comandos ou então em um *script* criado dentro do editor. Nos vetores, foi visto que um espaço separa valores que compõe uma mesma linha e que um ponto e vírgula insere uma nova linha. Com a junção desses dois conceitos há uma matriz. Valores na mesma linha são separados por um espaço sendo que o ponto e vírgula introduz uma nova linha na matriz. O exemplo a seguir mostra a criação de uma matriz F de dimensão 3x3:

```
>> F = [1 2 3; 4 5 6; 7 8 9]
F =

   1   2   3
   4   5   6
   7   8   9
```

2.3) Acessando posições de uma matriz

Para acessar um elemento de uma matriz, o índice da posição pode ser passado de duas formas: a primeira forma é acessar um elemento pela sua posição em relação a linha e a coluna. Numa matriz, assim como em um vetor, as linhas e as colunas também são indexadas em 1.

Outra forma de se acessar um elemento é pela sua posição absoluta. Em uma matriz o primeiro elemento é o da posição 1, e, assim por diante, os elementos são numerados de acordo com a sua posição de cima para baixo e da esquerda para a direita. Assim, pode-se consultar os valores guardados nas posições da matriz:

```
>> F = [1 2 3; 4 5 6; 7 8 9]
F =

   1   2   3
   4   5   6
   7   8   9

>> F(2,1)
ans =  4
>>
>> F(7)
ans =  3
```

2.4) Operações com matrizes

Para as matrizes, alguns comandos já vêm instalados e prontos para serem usados dentro dos códigos. Sendo os vetores um caso particular das matrizes, todos os comandos a seguir também são válidos nos vetores:

Função	Comando	Exemplo

Cria uma matriz quadrada randômica de ordem n	`A = magic(n)`	`>> A = magic(2)`
		`A =`
		`4 3`
		`1 2`

Cria uma matriz identidade de ordem n	`B = eye(n)`	`>> B = eye(2)`
		`B =`
		`Diagonal Matrix`
		`1 0`
		`0 1`

Calcula a inversa de uma matriz	`C = inv(A)`	`>> C = inv(A)`
		`C =`
		`0.40 -0.60`
		`-0.20 0.80`

Determina matriz transposta	`D = A'`	`>> D = A'`
		`D =`
		`4 1`
		`3 2`

Matriz de ordem n x m com todos os elementos iguais a 1	`E = ones(n,m)`	`>> E = ones(2,3)`
		`E =`
		`1 1 1`
		`1 1 1`

Matriz de ordem n x m com todos os elementos iguais a 0	`F = zeros(n,m)`	`>> F = zeros(2,3)`
		`E =`
		`0 0 0`
		`0 0 0`

Retorna a diagonal principal da matriz	`G = diag(C)`	`>> G = diag(C)`
		`G =`
		`0.40`
		`0.80`

Calcula o determinante da matriz	`H = det(A)`	`>> H = det(A)` `H = 5`

Também é possível realizar algumas operações com matrizes, desde que seja respeitada a relação entre as ordens das matrizes para cada operação a ser realizada:

Operação	Comando	Exemplo

Soma ou subtração de uma constante	AA = A + k	`>> AA = A + 2`
		`AA =`
		`6 5`
		`3 4`

Soma ou subtração entre duas matrizes (1)	BB = A + C	`>> BB = A + C`
		`BB =`
		`4.4 2.4`
		`0.8 2.8`

Multiplicação entre duas matrizes (2)	CC = A * D	`CC = A*D`
		`CC =`
		`25 10`
		`10 5`

Observações:

(1) Para realizar a soma ou subtração entre duas matrizes é necessário que elas sejam da mesma ordem, caso contrário não será possível realizar a operação e um erro será apresentado pelo *software* na janela de comandos;

(2) Para realizar a multiplicação entre duas matrizes é necessário que o número de colunas da primeira matriz seja igual ao número de linhas da segunda:

$$(A = {}_{(a_{ij})_{m\,x\,n}}) * (B = {}_{(b_{ij})_{n\,x\,p}}) = (C = {}_{(c_{ij})_{m\,x\,p}})$$

Também é possível pegar uma matriz e dividi-la em vetores, sendo a divisão feita pelas linhas ou pelas colunas:

```
>> F = [1 2 3; 4 5 6; 7 8 9]
F =

   1   2   3
   4   5   6
   7   8   9

>> Linha1 = F(1,:)
Linha1 =

   1   2   3

>> Coluna1 = F(:,1)
Coluna1 =

   1
   4
   7
```

O mesmo procedimento pode ser feito para quaisquer linhas ou colunas da matriz.

Assim como nos vetores, o comando *size* retorna o tamanho da matriz, salvando as informações em duas variáveis, as linhas e as colunas. Se é necessário saber quantos elementos há na matriz sem ser conhecida as suas dimensões, também pode ser utilizado o comando *numel:*

```
>> F = [1 2 3; 4 5 6; 7 8 9]
F =

   1   2   3
   4   5   6
   7   8   9

>> [linha, coluna] = size(F)
linha =  3
coluna =  3
>>
>> elementos = numel(F)
elementos =  9
>>
```

2.5) Leitura de valores em uma matriz

A leitura de informações para uma matriz é feita com duas repetições, sendo uma para o número de linhas e outra para o número de colunas:

```
%Entrada de dados em uma matriz
clear
clc
display('');

linha = input('Número de linhas da matriz: ');
coluna = input('Número de colunas da matriz: ');
display('');

for i = 1:linha
  for j = 1:coluna
    mat(i,j) = input('Digite um número: ');
  endfor

  display('');
endfor

mat
```

```
Número de linhas da matriz: 2
Número de colunas da matriz: 3

Digite um número: 1
Digite um número: 2
Digite um número: 3

Digite um número: 11
Digite um número: 22
Digite um número: 33

mat =

    1    2    3
   11   22   33
```

Os códigos nesse relatório são baseados no *Octave*. Para o *MatLab* o comando que fecha as estruturas é apenas o *end* ao invés de *endif* e *endfor*.

Exercícios

(1) Faça um *script* em que o usuário possa entrar com oito valores e armazene essas informações em um vetor. Ao final da leitura, exiba o vetor de trás para a frente, ou seja, com as informações em ordem inversa a de leitura.

(2) Preencha duas matrizes A e B, ambas com 3 linhas e 4 colunas com valores aleatórios. Faça um *script* que mostre a matriz C, tal que C = A + B.

(3) Faça um *script* que percorra uma matriz com 4 linhas e 5 colunas e que troque todos os elementos maiores que 10 por 0. Os demais elementos devem ser mantidos.

Referência

D. Hanselman e B. Littlefield, Matlab 6 - Curso Completo, São Paulo: Pearson Education, 2003.