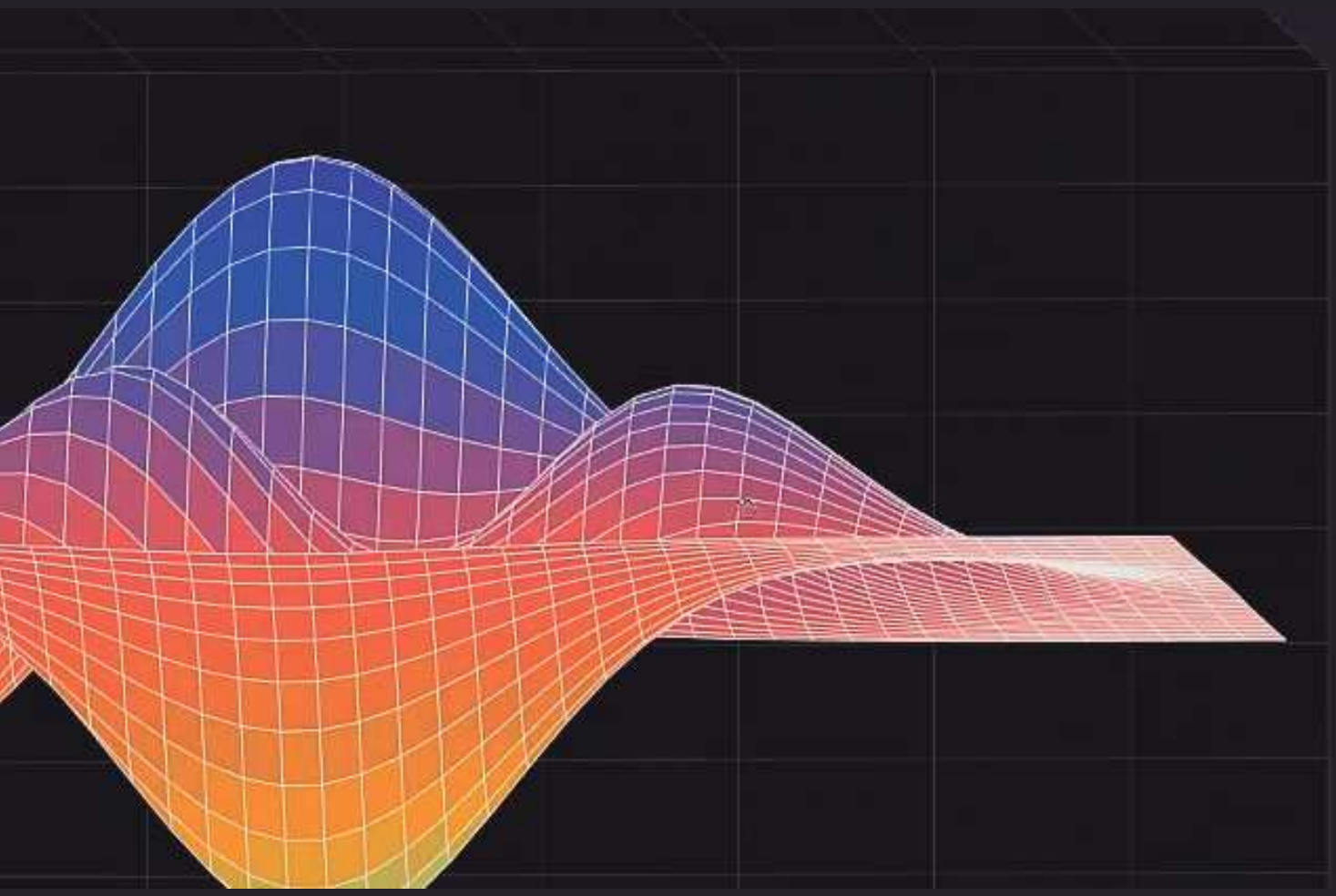
MINICURSO DE MatLab



Módulo 3 e 4
Estruturas condicionais, Laços Condicionais, Gráficos e Importação de Arquivos.

UNICAMP RACING

Conteúdo

1 Estruturas Condicionais

1.1 If, Else, Elseif

O *if* é uma estrutura condicional que avalia a veracidade de uma expressão. Quando a expressão for verdadeira o código será executado. A sua sintaxe é a seguinte:

```
if <condição>
        <codigo>
end
```

A estrutura *else* é utilizada em conjunto com a *if*. O código dessa estrutura somente será executada se a condição do if **não** for satisfeita.

```
if <condição>
        <codigo1>
else
        <codigo2>
end
```

```
if a > 0 && b > 0
        disp ('a e b são maiores do que zero');
else
        disp ('a, b, ou ambos, são menores do que zero');
end
```

Já a *elseif* é uma combinação das duas estruturas, de forma que o código do *else* somente será executado se ele satisfazer uma nova condição.

```
% Escolha valores de a e b adequados para testar esse exemplo
if a > 0 && b > 0
        disp ('Ambos a e b são maiores do que zero');
elseif a>0
        disp ('Somente a é maior do que zero');
else
        disp('Somente b é maior do que zero')
end
```

Estruturas condicionais também conseguem analisar expressões com matrizes.

```
a = [1 2 3];
a > 2
if a > 2 % Verdadeiro se todos os elementos de a forem
   maiores que 2
        disp ('Todos os elementos são maiores que 2');
else
        disp ('Nem todos os elementos são maiores que 2');
```

```
end
```

Como pôde ser notado pelos exemplos acima, a condição estipulada sobre um vetor/-matriz somente retornava verdadeiro quando todos os elementos da mesma atendiam tal condição. A função *any* permite uma análise mais profunda do vetor/matriz.

```
any (a>2)   % Verdadeiro se pelo menos um elemento de a for
   maior que 2
if any(a>2)
        disp('Pelo menos um elemento de a é maior que 2');
end
```

Exercício 1:

> Crie uma função que receba como entrada uma matriz e retorne ao usuário os elementos dessa matriz que são maiores do que 0.5 e mostre na *Command Window* a porcentagem de elementos que satisfazem essa restrição.

1.2 Switch,Case,Otherwise

A estrutura *switch* compara o dado armazenado em uma certa variável com uma lista de possíveis valores (cases). Se a comparação for verdadeira para algum *case* é executada uma ação. Somente um *case* pode ser executado em cada execução do *switch*.

É possível, porém não obrigatório, incluir um *otherwise* na estrutura. A ação estipulada no *otherwise* somente será executada se nenhum *case* for executado.

```
switch <variavel>
case <label_1>
        <codigo_1>
case <label_2>
        <codigo_2>
otherwise
        <codigo_3>
end
```

```
x = 42;
switch x
case 'Hello'
        disp('x = Hello');
case 42
        disp('x = 42');
otherwise
        disp(Desconhecido);
end
```

2 Laços Condicionais (Estruturas de Repetição)

2.1 While

O *while* é uma estrutura de repetição o código escrito dentro dela será repetido enquanto a condição estabelecida é verdadeira.

```
while <condicao>
        <acao>
end
```

```
k = 1;
while k<10
        k = k+1
end
```

Deve-se tomar cuidado para não entrar em um *loop* infinito quando se trabalha com estruturas de repetição, principalmente com *while*. O atalho **CTRL+C** cancela qualquer comando que o MATLAB está executando, sendo útil para ocasiões como essa.

```
while 1          % Loop infinito, o atalho CTRL+C para a execução
        k = k+1
end
```

Exercício 2:

> Abra o gerenciador de tarefas do seu computador e observe o uso da CPU pelo MATLAB. Execute um loop infinito qualquer e veja a diferença de uso.
>
> Dica: o comando CTRL+C para a execução do loop a qualquer momento.

2.2 For

O *for* é uma estrutura de repetição que realiza um *loop* por um número específico de vezes, incrementando ou decrementando uma variável de controle *após* cada iteração.

Deve-se especificar como será esse incremento/decremento da variável de controle, desde os valores iniciais e finais até o de quanto será o passo.

```
for variavel = valor
        <codigo>
end
```

```
for k = 1:10     % Passo de 1 em 1
        k
end
```

```
4
5  tic
6  for k = 1:.1:10 % Passo de 0.1 em 0.1
7          k
8  end
9  t_exec = toc
```

Exercício 3:

> A função *tic/toc* é muito útil para analisar o tempo de execução de um script. Utilizando a mesma, crie as seguintes estruturas de repetição, analisando o tempo de processamento de cada uma delas.
>
> **a)** Um laço *for* que se repete 10^5 vezes, incrementando uma variável de zero até 10^5 e mostrando na *Command Window* o valor dessa variável a cada repetição.
>
> **b)** Um laço idêntico ao anterior, mas sem mostrar na *Command Window* o valor da variável.

3 Gráficos

3.1 Gráficos 2D

Gráficos 2D são plotados em uma figura (*figure*) utilizando a função *plot*. Embora seu uso é sendo algo bem simples, bastando passar como argumento o dado a ser plotado em x e y, existem inúmeros argumentos extras que podem ser passados de forma a mudar, por exemplo, aspectos da curva como cor e espessura.

A função *plot* com apenas um argumento plota os dados em função do número de pontos.

```
1  f1 = 0:10^-3:2*pi;
2  sig1 = sin(f1);
3  plot(sig1);      % Plota o sinal em função do número de pontos
4  plot(t,sig1);    % plota o sinal em função de f
```

Percebe-se que quando é executado um comando *plot* e já existe uma figura aberta, o mesmo é plotado nesta figura, independentemente se já existe algo na mesma. Se for desejado desenhar esse gráfico na mesma figura (um em cima do outro), deve-se colocar o comando *hold on* após o comando de plotar o **primeiro** gráfico. Se for desejado criar uma nova figura, utiliza-se o comando *figure*.

```
1  f1 = 0:10^-3:2*pi;
2  f2 = 0:10^-3:4*pi;
3  plot(f1,sin(f1))
4  hold on          % Dois plots na mesma figura
5  plot(f2,2*sin(f2))
```

```
f1 = 0:10^-3:2*pi;
f2 = 0:10^-3:4*pi;
plot(f1,sin(f1))
figure()        % Dois plots em figuras diferentes
plot(f2,2*sin(f2))
```

É possível modificar certos parâmetros de forma a melhorar a visualização do gráfico. Alguns são utilizados como argumentos da própria função *plot*, como a cor e tipo de linha, enquanto outros são funções próprias, como *xlabel*, *grid*.

```
grid on;
plot(t,sig1,'linewidth',2);    % Altera grossura da linha
plot(t,sig1,'r')   % Cor da linha será vermelha (r)
plot(t,sig1,'--')  % Tipo da linha será listrado (--)
plot(t,sig1,'r--') % Uniao dos parametros

xlim ([0 8*pi]);          % Estabelece limites do eixo x
ylim ([-1 4]);            % Estabelece limites do eixo y
xlabel ('f');             % Nomeia o eixo x
ylabel ('sig');           % Nomeia o eixo y
title ('Meu primeiro gráfico');  % Título  do gráfico
```

Lembrando que se for criado um gráfico de um valor em função do outro (dois argumentos de entrada, x e y) ambos os dados devem possuir a mesma dimensão.

É possível realizar as modificações também na própria janela do gráfico. Por meio da aba *Insert* é possível inserir legendas, título, nomear eixo, entre outras ferramentas.

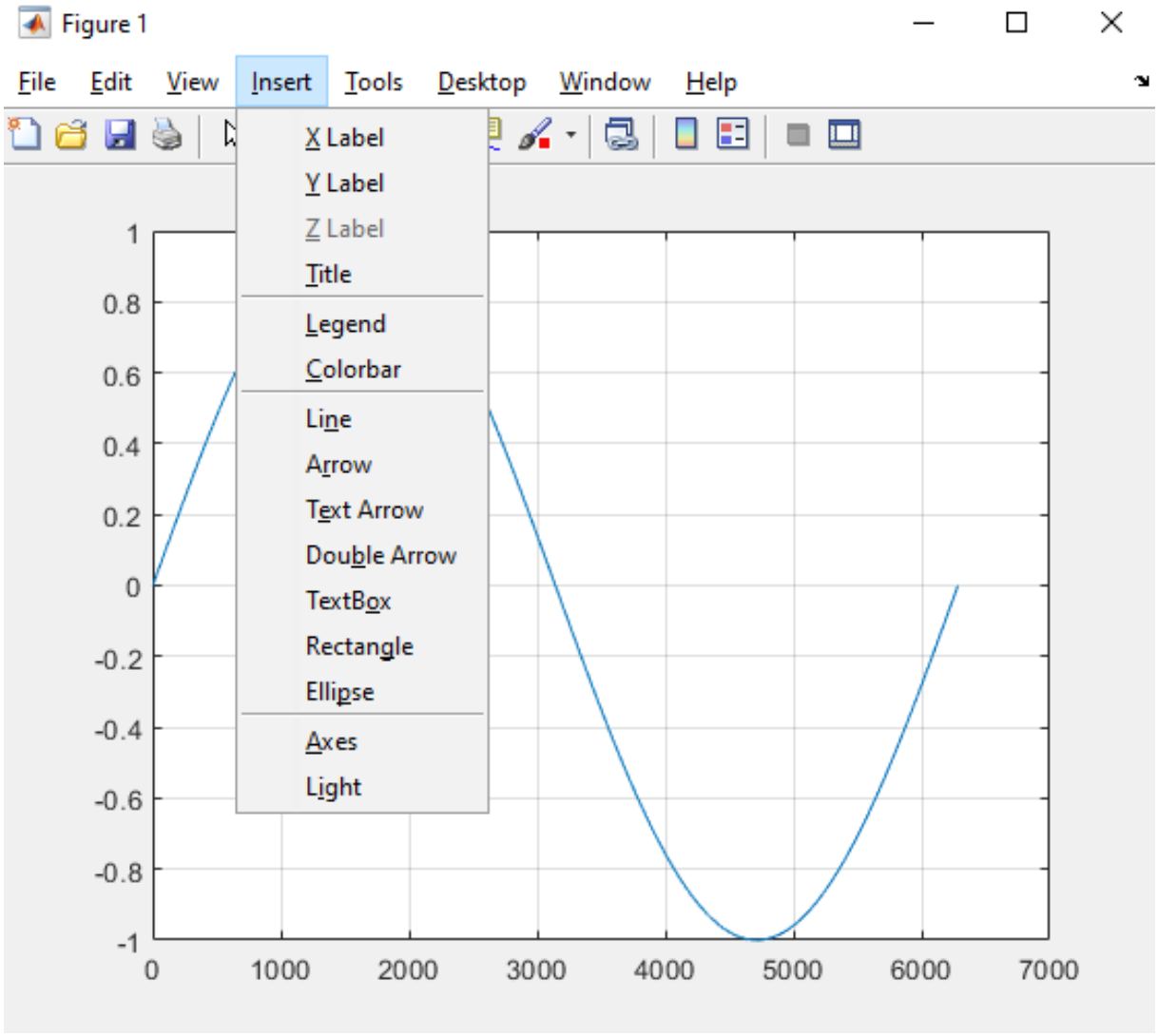


Figura 3.1: Aba Insert

Exercício 4:

> Plote a função $f(x) = 10x + 20$ no intervalo [0 50] com 100 pontos utilizando o comando *plot*.

Quando se deseja plotar funções ou expressões numéricas é mais fácil utilizar a função *fplot*.

```
1 fplot(@(x) 10*x+100, [0 50], 'MeshDensity', 100)
2 flot( @(x) sin(x), [0 16*pi])
```

3.2 Gráficos 3D

Gráficos 3D também podem ser plotados. Para gráficos simples utiliza-se a função *plot3* e para superfícies *surf*.

```
x = 0:0.01:10;
y = sin(x);
z = cos(x);
plot3(x,y,z,'linewidth',3);


N = [1000 2000 3000 4000 5000 6000 7000];
RK = [10 20 30 40 50 60 70 80 90 100];
KF = (1E-6*( (N-2500).^2)' * (RK-50) )';
surf(N,RK,KF);
xlabel('N');
ylabel('RK');
zlabel('KF');
colorbar_handle = colorbar;
carac_colorbar = get(colorbar_handle)
set(colorbar_handle,'location','south')
set(colorbar_handle,'location','southoutside')
```

3.3 Subplot

Subplot é uma função que permite plotar mais de um gráfico na mesma figura mas sem sobrepô-los (como aconteceria se fosse utilizado o comando *hold on*). Ao utilizar essa função, sua *figure* será dividida e cada gráfico será plotado em um lugar diferente.

```
subplot(2,3,1) % 2 linhas de gráficos, 3 colunas, sendo o
   primeiro dele selecionado
plot( sin( linspace(0,6*pi,10) ) )
subplot(2,3,2) % Segundo gráfico selecionado
plot( sin( linspace(0,6*pi,20) ) )
subplot(2,3,3) % Terceiro gráfico selecionado
plot( sin( linspace(0,6*pi,30) ) )
subplot(2,3,4)
plot( sin( linspace(0,6*pi,40) ) )
subplot(2,3,5)
plot( sin( linspace(0,6*pi,50) ) )
subplot(2,3,6)
plot( sin( linspace(0,6*pi,60) ) )
```

3.4 Handles

Handles são um tipo de dado que permite realizar a chamada de uma função de modo indireto. É possível utilizar um *handle* como um argumento de uma função. No momento em que é criada, torna-se necessário que a função associada esteja definida no *path* do MATLAB e/ou no *Current Folder*.

Quando são criadas, *handles* guardam toda a informação necessária para a função ser executada. Por exemplo, pode-se executar uma sub-função fora do seu arquivo (.m file

onde está definida) utilizando um *function handle*, desde que o *handle* tenha sido criado no arquivo da subfunção (.m file onde está definida). A função *figure* cria uma nova figura que, a princípio, não possui nenhuma informação. Ao atribuí-la a uma variável, criamos um *handle* dela, sendo assim possível alterar parâmetros na figura posteriormente.

```
1 fig_handle_1 = figure(1);
2 carac_fig_1 = get(fig_handle_1)
3 set(fig_handle_1,'WindowStyle','docked');
4 set(fig_handle_1,'visible','off');
5 set(fig_handle_1,'visible','on');
```

4 Importação de Arquivos

A importação de dados presentes em arquivos externos é algo bem fácil de se fazer, podendo ser feito tanto por linhas de código quanto utilizando a própria interface do MATLAB.

As figuras a seguir mostram como importar utilizando a interface do MATLAB, que é a forma mais simples.

Primeiramente deve-se clicar em *Import Data* e selecionar o arquivo a ser importado.

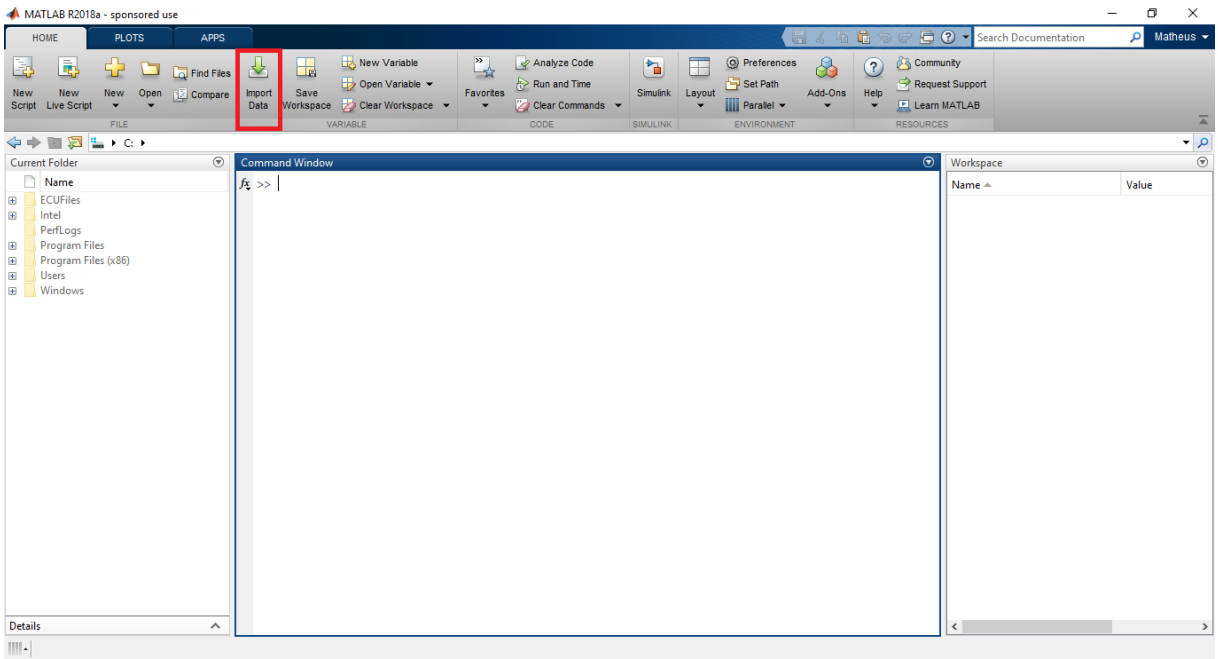


Figura 4.1: Import Data

Em seguida selecione as colunas e linhas desejadas, assim como o formato do dado de saída.

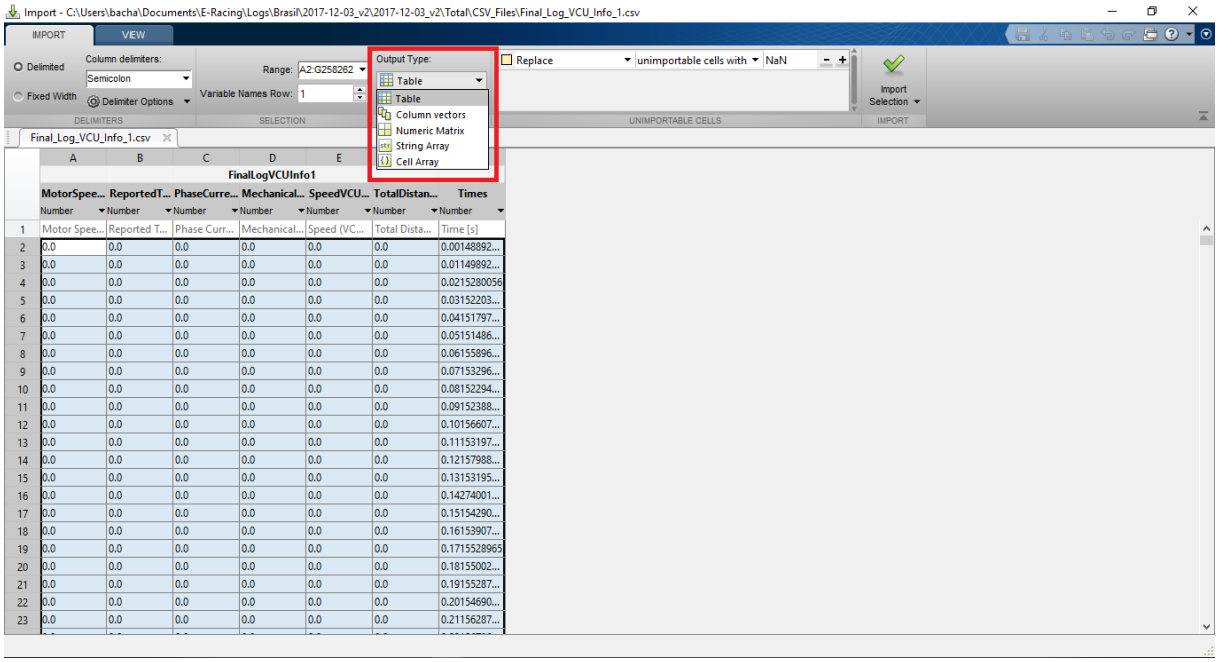


Figura 4.2: Formato de saída

Clique no *check* verde para importar os dados ou, alternativamente, pode-se criar um script ou uma função para repetir esta importação ou realizá-la automaticamente em um algoritmo. Para isso, clique em *Import Selection* e escolha *Generate Script* ou *Generate Function*, conforme for o desejado.

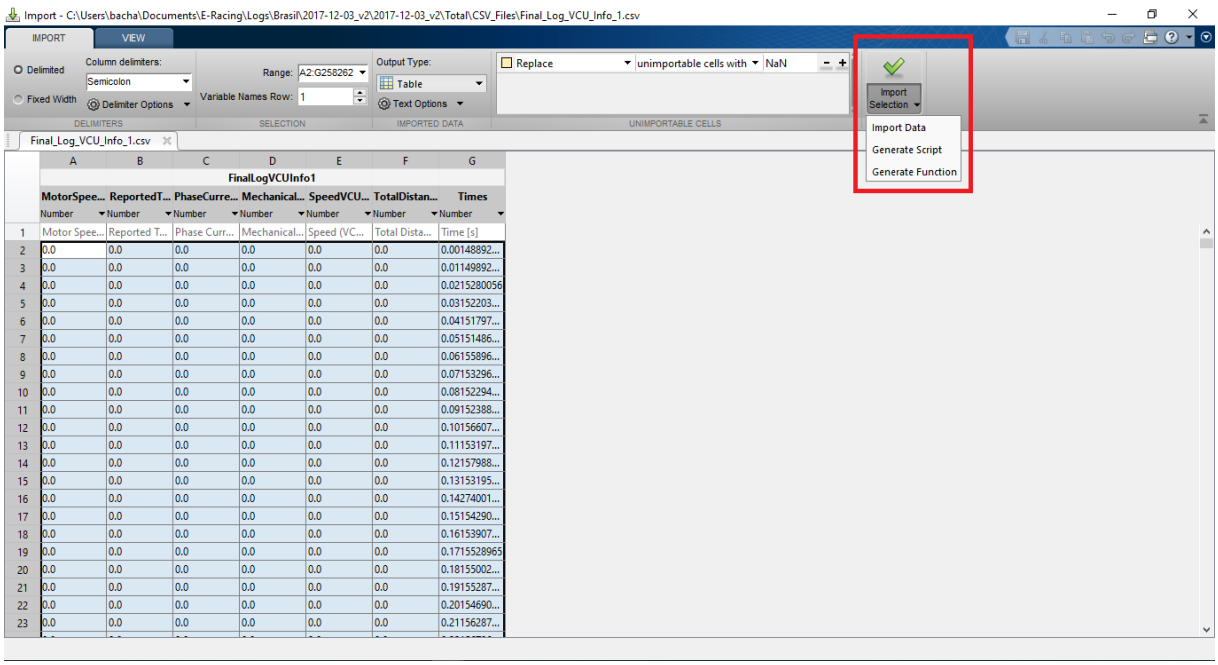


Figura 4.3: Import Selection

Exercício 5:

O arquivo *Final_Log_Sevcon_Info_1.csv* é parte de um log real obtido na competição Formula SAE Electric em Lincoln, no ano de 2017. Utilize o mesmo para realizar os seguintes itens

a) Importe os dados *ReportedTorqueNm* (torque reportado) e *Times* (tempo associado a cada ponto) como vetores coluna.

b) Calcule o torque médio.

c) Plote o torque em função do tempo sendo que os pontos que se encontrarem acima da média devem estar em vermelho. Não se esqueça de nomear os eixos e indicar, por meio de uma legenda, oque são os pontos vermelhos.

d) Calcule a porcentagem de pontos que se encontram acima da média e mostre essa porcentagem na *Command Window*.