

# INTRODUÇÃO AO MATLAB



**AULA 02**

**SEATEL 2021**

**LEONARDO BARBOSA DA SILVA**

A grayscale background image showing a person's hands typing on a laptop keyboard. The laptop screen displays CSS code, including rules for 'em.mail' and 'em.phonef'.

**MISSÃO DO CURSO:**

**MOstrar QUE NÃO PRECISA SER  
PROGRAMADOR PARA FAZER CÓDIGOS  
EM MATLAB!**

# AGENDA

- APRESENTAÇÃO DO PROGRAMA
- CONCEITOS BÁSICOS DE PROGRAMAÇÃO
- CRIAÇÃO DE SCRIPTS E FUNÇÕES CUSTOMIZADAS
- OPERAÇÕES COM VETORES E MATRIZES
- GERAÇÃO E FILTRAGEM DE SINAIS
- CRIAÇÃO DE GRÁFICOS
- TRABALHAR COM DADOS .TXT E .CSV
- EXPORTAR PROJETOS COMO RELATÓRIOS
- EXEMPLOS DE PROJETOS EM MATLAB NO ESCOPO DA GRADUAÇÃO

**AULA 01**

**AULA 02**

# GERAÇÃO E FILTRAGEM DE SINAIS

- **COMBINANDO O QUE JÁ APRENDEMOS ONTEM:**

- Podemos usar as funções `ones()` e `zeros()` para criar alguns sinais fundamentais, como o degrau unitário por exemplo

```
>> a = zeros(1,10);  
>> b = ones(1,10);  
>> c = [a b]
```

```
c =
```

```
Columns 1 through 11
```

```
0    0    0    0    0    0    0    0    0    0    1
```

```
Columns 12 through 20
```

```
1    1    1    1    1    1    1    1    1
```

# GERAÇÃO E FILTRAGEM DE SINAIS

- **COMBINANDO O QUE JÁ APRENDEMOS ONTEM:**

- Outro sinal muito simples de ser criado é a função impulso

```
>> a = zeros(1,11);  
>> a(1,6) = 1;  
>> a
```

a =

0 0 0 0 0 1 0 0 0 0 0

# GERAÇÃO E FILTRAGEM DE SINAIS

- **COMBINANDO O QUE JÁ APRENDEMOS ONTEM:**

- As funções matemáticas já vistas, também recebem vetores. Assim, podemos criar diversos sinais com muita facilidade:

```
>> x = linspace(0,2*pi,100);  
>> y = sin(x)
```

```
y =
```

```
Columns 1 through 6
```

```
0    0.0634    0.1266    0.1893    0.2511    0.3120
```

```
Columns 7 through 12
```

```
0.3717    0.4298    0.4862    0.5406    0.5929    0.6428
```

```
Columns 13 through 18
```

# GERAÇÃO E FILTRAGEM DE SINAIS

- **COMBINANDO O QUE JÁ APRENDEMOS ONTEM:**

- As funções matemáticas já vistas, também recebem vetores. Assim, podemos criar diversos sinais com muita facilidade:

```
>> y = exp(x)
```

```
y =
```

```
Columns 1 through 6
```

```
1.0000    1.0655    1.1353    1.2097    1.2890    1.3735
```

```
Columns 7 through 12
```

```
1.4635    1.5593    1.6615    1.7704    1.8864    2.0100
```

```
Columns 13 through 18
```

```
2.1417    2.2820    2.4316    2.5909    2.7606    2.9415
```

# GERAÇÃO E FILTRAGEM DE SINAIS

- **COMBINANDO O QUE JÁ APRENDEMOS ONTEM:**
  - Algumas outras funções que geram sinais característicos:

Forma de onda	Função
Onda quadrada	<code>square(x)</code>
Onda dente de serra	<code>sawtooth(x)</code>
Onda triangular	<code>sawtooth(x, 0.5)</code>



# GERAÇÃO E FILTRAGEM DE SINAIS

- **PARA APLICARMOS UM FILTRO, PRIMEIRO PRECISAMOS TER O QUE FILTRAR...**

- Em teorias de comunicação, um dos distúrbios mais básicos em qualquer sinal é o ruído branco gaussiano
- Podemos simular o efeito dele de duas maneiras:
  - Somando valores randômicos (função “rand( )”) ao sinal original
  - Utilizando a função “awgn( )”, que aplica um ruído branco gaussiano médio dada uma certa relação sinal-ruído (SNR) mínima

**Obs:** O SNR pode ser interpretado como a razão da potência do sinal e do ruído, de maneira que quanto maior esta relação, menos o sinal é afetado pela interferência. Ou seja, maiores SNRs indicam sinais de maior fidelidade

# GERAÇÃO E FILTRAGEM DE SINAIS

- **PARA APLICARMOS UM FILTRO, PRIMEIRO PRECISAMOS TER O QUE FILTRAR...**
  - Comparando um sinal senoidal antes e depois de passar por um canal senoidal “awgn( )” com um SNR de 25 dB:

```
Sinal_ruidoso = awgn( sinal_original, SNR )
```

```
>> x = linspace(0,2*pi,100);  
>> y = sin(x);  
>> y_channel = awgn(y,25);
```

# GERAÇÃO E FILTRAGEM DE SINAIS

- **PARA APLICARMOS UM FILTRO, PRIMEIRO PRECISAMOS TER O QUE FILTRAR...**

- Comparando um sinal senoidal antes e depois de passar por um canal senoidal “awgn( )” com um SNR de 25 dB:

```
>> y(1:10)
```

```
ans =
```

```
Columns 1 through 6
```

```
0    0.0634    0.1266    0.1893    0.2511    0.3120
```

```
Columns 7 through 10
```

```
0.3717    0.4298    0.4862    0.5406
```

```
>> y_channel(1:10)
```

```
ans =
```

```
Columns 1 through 6
```

```
0.0196    0.0230    0.2265    0.1404    0.2608    0.4269
```

```
Columns 7 through 10
```

```
0.4309    0.4288    0.5131    0.6758
```

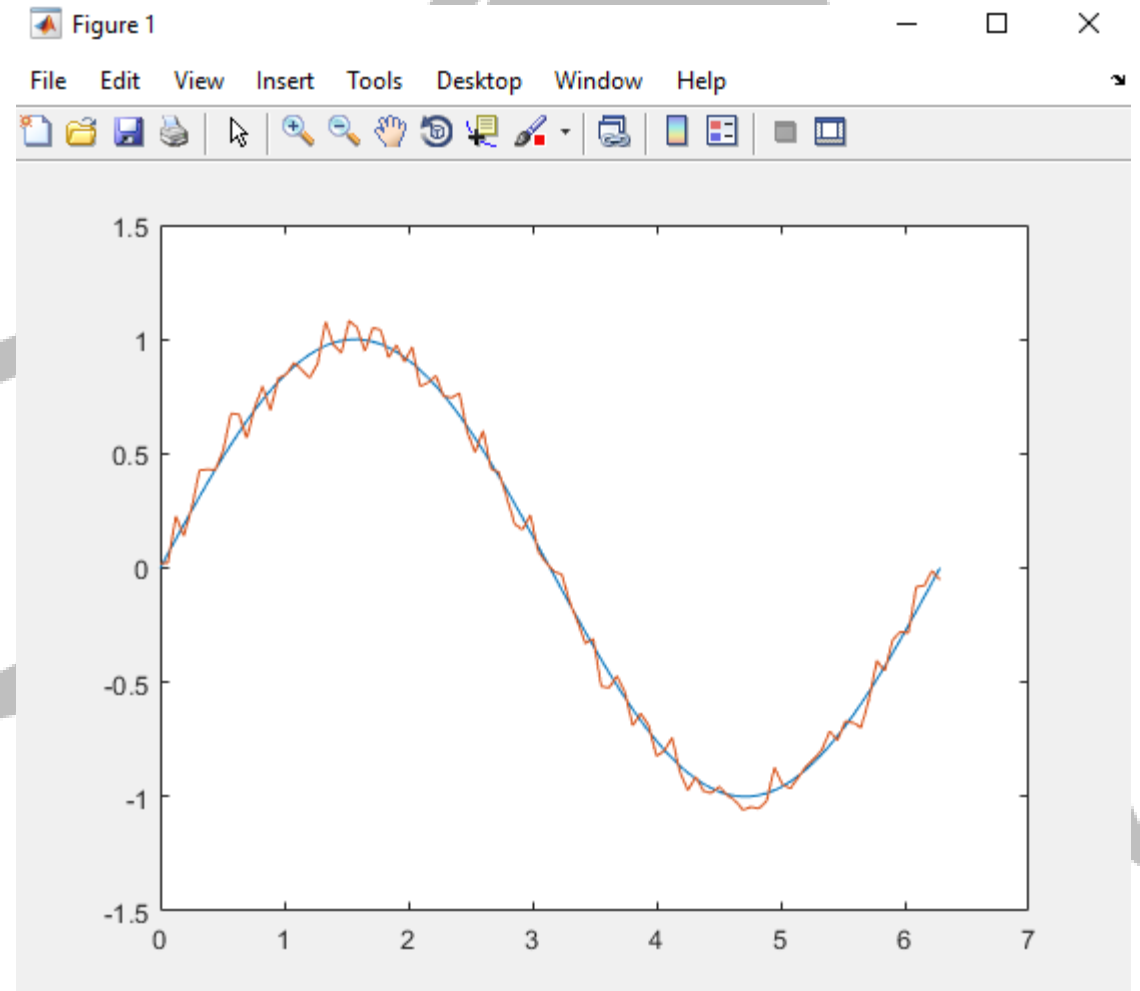
# GERAÇÃO E FILTRAGEM DE SINAIS

- **PARA APLICARMOS UM FILTRO, PRIMEIRO PRECISAMOS TER O QUE FILTRAR...**

- Para visualizar melhor o efeito do ruído, podemos verificar as duas curvas em um gráfico:

```
>> plot(x,y)
>> hold
Current plot held
>> plot(x,y_channel)
```

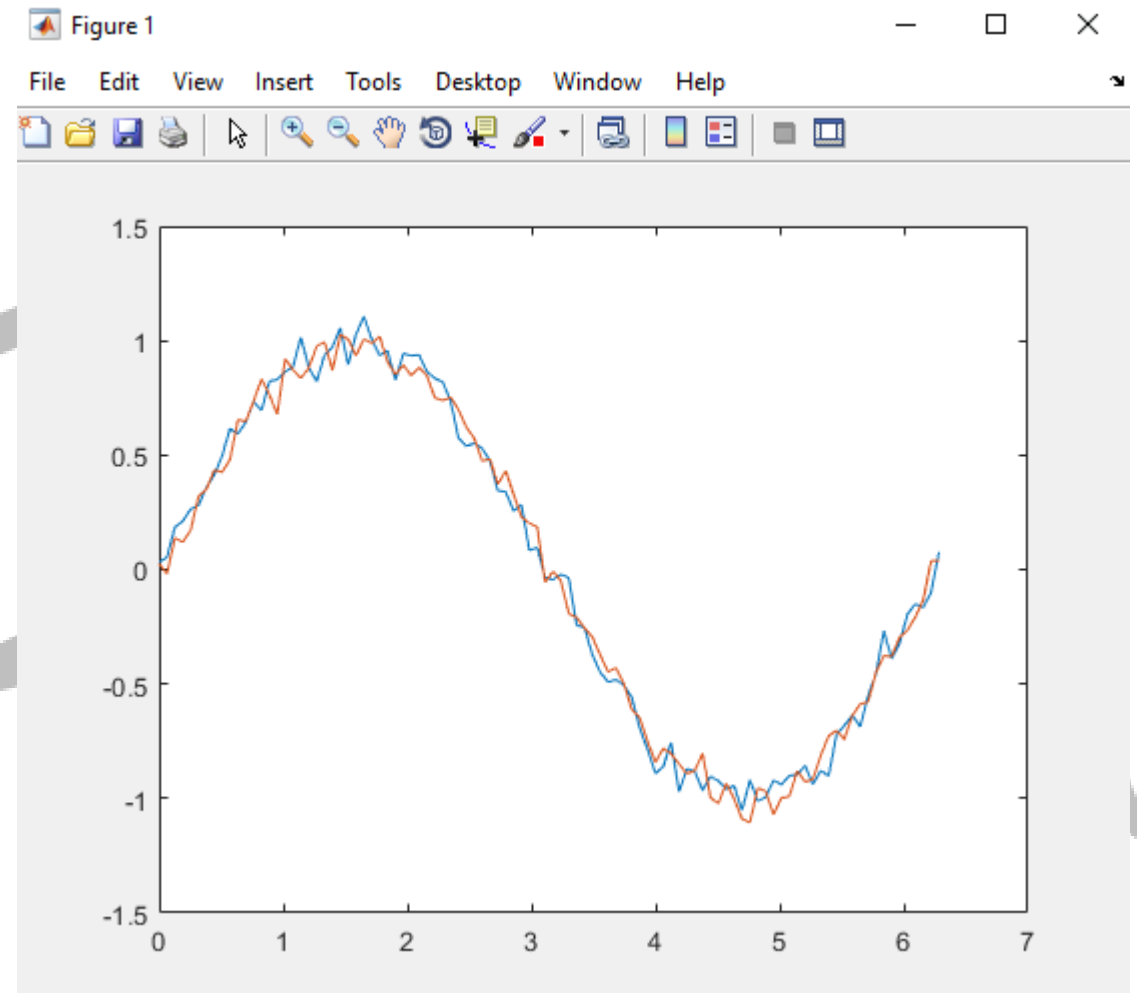
- Em azul, temos o seno original e em vermelho o sinal com ruído
- Se mudarmos o SNR para valores menores, o sinal se torna ainda mais ruidoso!



# GERAÇÃO E FILTRAGEM DE SINAIS

- **PARA APLICARMOS UM FILTRO, PRIMEIRO PRECISAMOS TER O QUE FILTRAR...**
  - Podemos verificar que este ruído é aleatório plotando em um mesmo gráfico duas iterações de `y_channel` com os mesmos parâmetros

```
>> y_channel = awgn(y,25);  
>> plot(x,y_channel)  
>> hold  
Current plot held  
>> y_channel = awgn(y,25);  
>> plot(x,y_channel)  
..
```



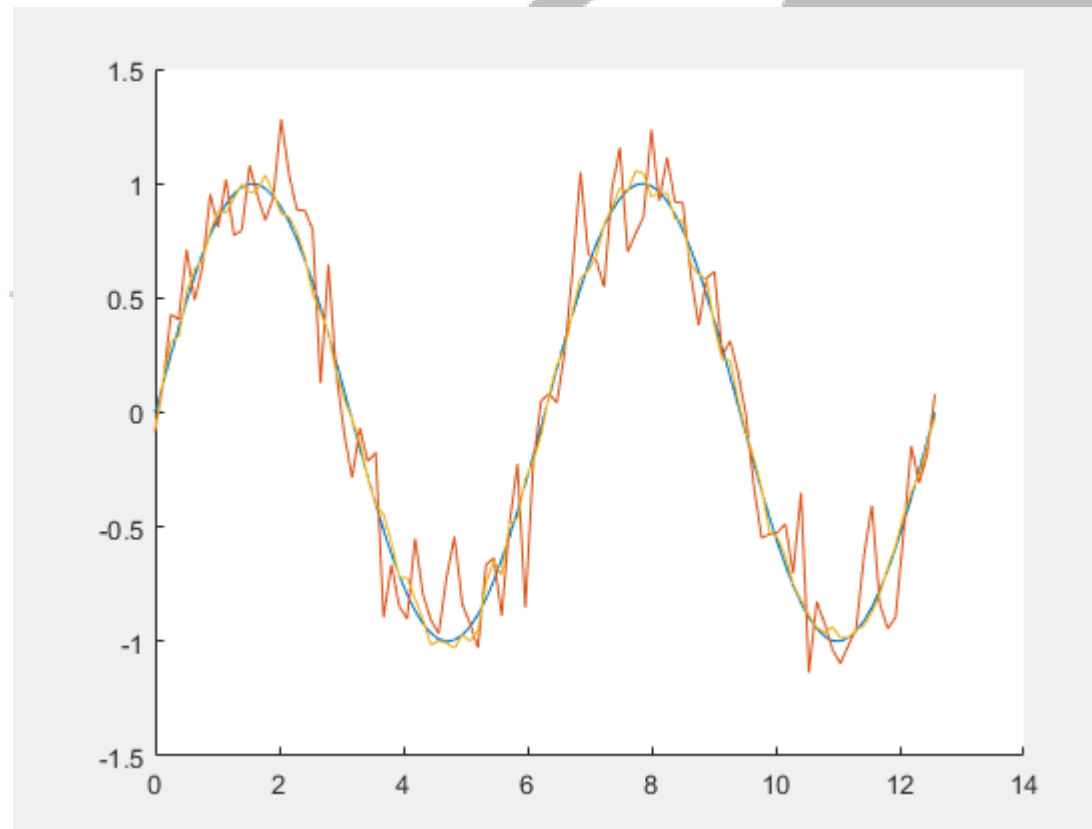
# GERAÇÃO E FILTRAGEM DE SINAIS

- **SABENDO DISSO, PODEMOS IMPLEMENTAR UMA PRIMEIRA OPÇÃO DE FILTRO**
  - Dado que o ruído  $awgn$  tem média zero e distribuição normal, podemos atenuar seu efeito ao realizar médias de diversas realizações do sinal
  - Ou seja, podemos coletar  $n$  períodos do sinal ruidoso e fazer a média
  - É uma técnica muito eficiente, porém nem sempre temos a nossa disposição mais de uma medição do sinal

# GERAÇÃO E FILTRAGEM DE SINAIS

- SABENDO DISSO, PODEMOS IMPLEMENTAR UMA PRIMEIRA OPÇÃO DE FILTRO

```
filter_realizations.m  X +
1 -   clc
2 -   clear all
3 -   close all
4 -
5 -   x = linspace(0,2*pi*2,100);
6 -   y = sin(x);
7 -   N = 2;
8 -
9 -   y_channel = awgn(y,15);
10 -
11 -   y_list = zeros(N,100);
12 -
13 -   for n = 1:N
14 -       y_channel_loop = awgn(y,25);
15 -       y_list(n,:) = y_channel_loop;
16 -   end
17 -
18 -   y_filtered = sum(y_list,1)/N;
19 -
20 -   hold on
21 -   plot(x,y)
22 -   plot(x,y_channel)
23 -   plot(x,y_filtered)
```



■ y  
■ y\_channel  
■ y\_filtered

# GERAÇÃO E FILTRAGEM DE SINAIS

- **EXEMPLO DA FUNÇÃO FILTER**

- Outra opção de filtragem é através da função “filter()”, que tem como argumentos os a e b de uma dada função transferência
- Um dos filtros mais simples que podemos implementar desta maneira é o “Média móvel”
- Nele, cada valor do sinal é resultado da média de L elementos do seu entorno, geralmente de medições diretamente anteriores
- Nesta técnica, atrasamos levemente o sinal em troca da suavização dele

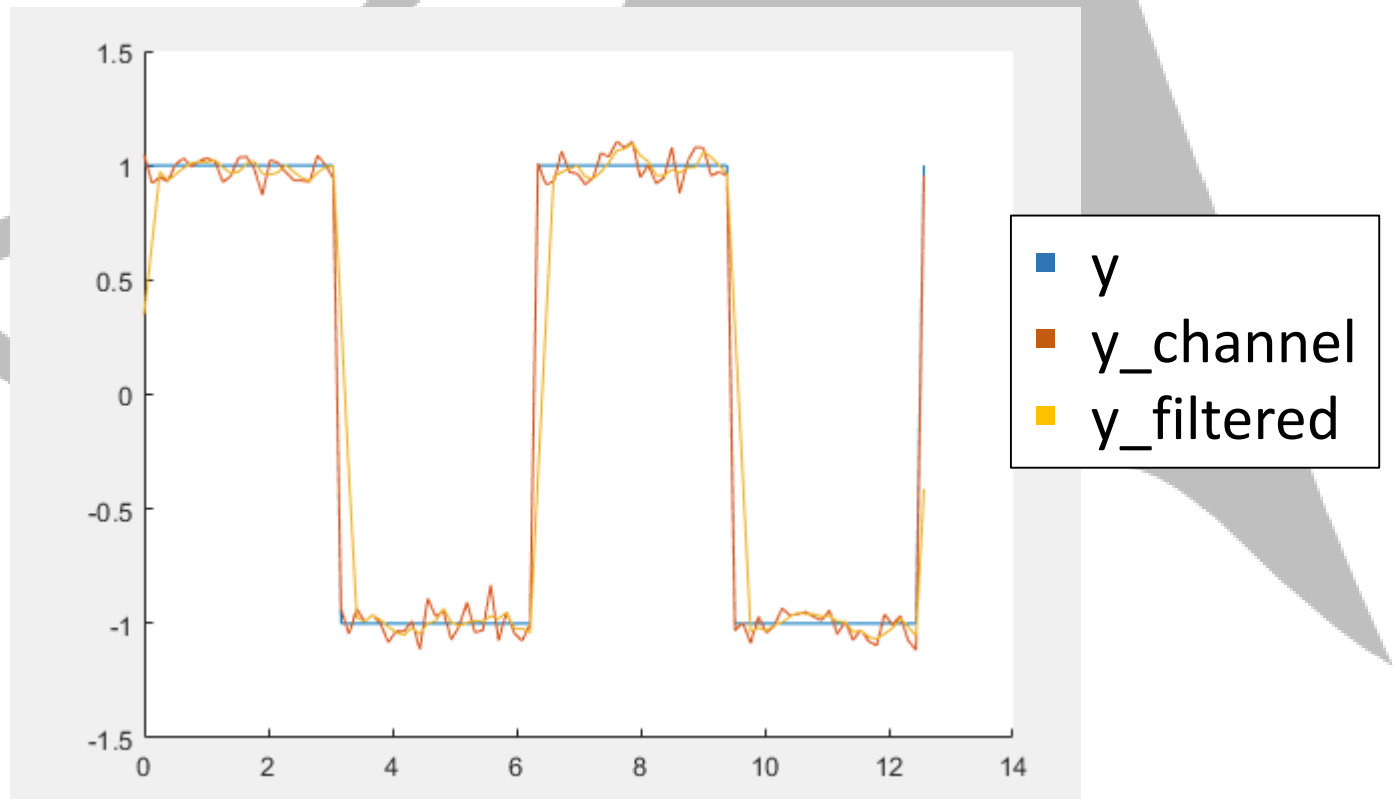
$$f(x) = \frac{1}{L} (y(n) + y(n - 1) + \dots + y(n - (L - 1)))$$



# GERAÇÃO E FILTRAGEM DE SINAIS

- EXEMPLO DA FUNÇÃO FILTER

```
1 - clc
2 - clear all
3 - close all
4
5 - x = linspace(0,2*pi*2,100);
6 - y = square(x);
7 - y_channel = awgn(y,25);
8
9 - L = 3;
10 - B = ones(1,L)/L;
11 - y_filtered = filter(B,1,y_channel);
12
13 - hold on
14 - plot(x,y)
15 - plot(x,y_channel)
16 - plot(x,y_filtered)
```

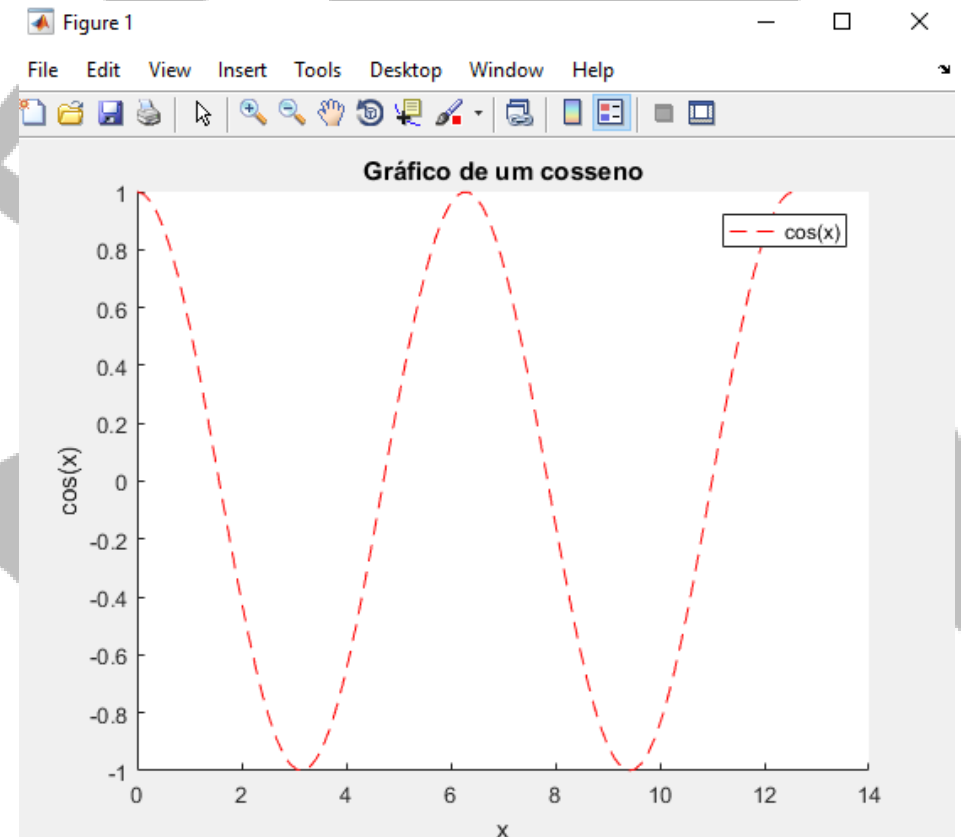
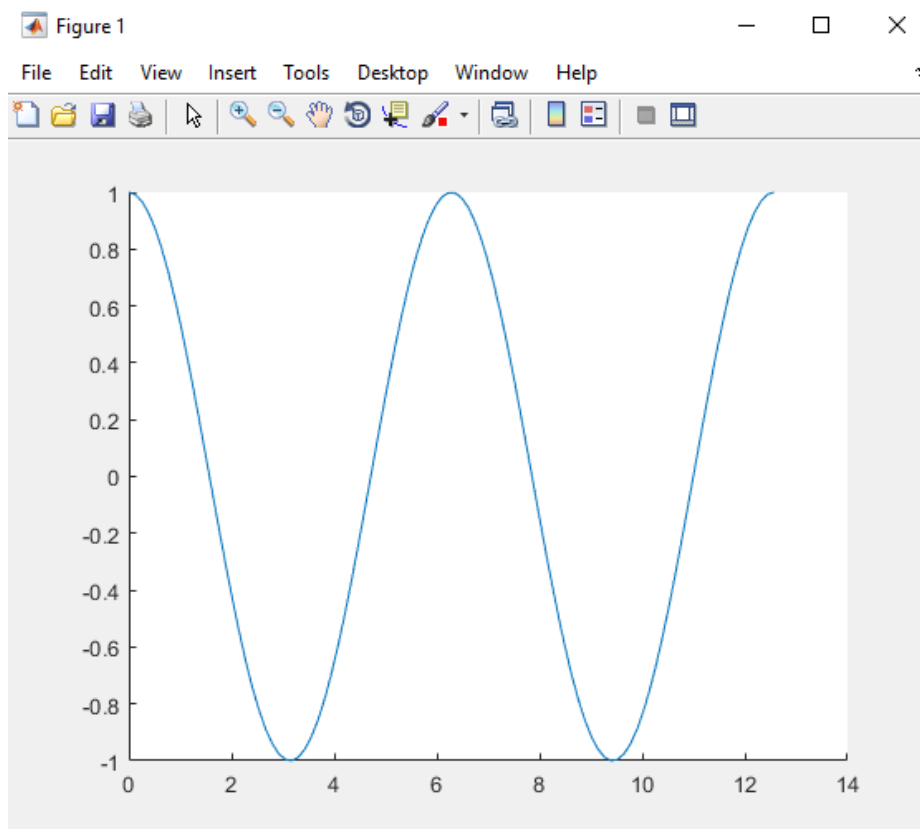


# GERAÇÃO E FILTRAGEM DE SINAIS

- **O QUE MAIS PODEMOS ESTUDAR EM RELAÇÃO A FILTROS?**
  - Filtros em relação a banda de passagem (passa-alta, passa-baixa, passa-faixa e rejeita-faixa)
  - Modelos de filtros (Butterworth, Elípticos, etc)
  - Filtros IIR e FIR em geral
  - Filtros adaptativos (LMS, RLS, AP, etc)

# CRIAÇÃO DE GRÁFICOS

- JÁ CONHECEMOS O PLOT, MAS O QUE MAIS PODEMOS FAZER?



# CRIAÇÃO DE GRÁFICOS

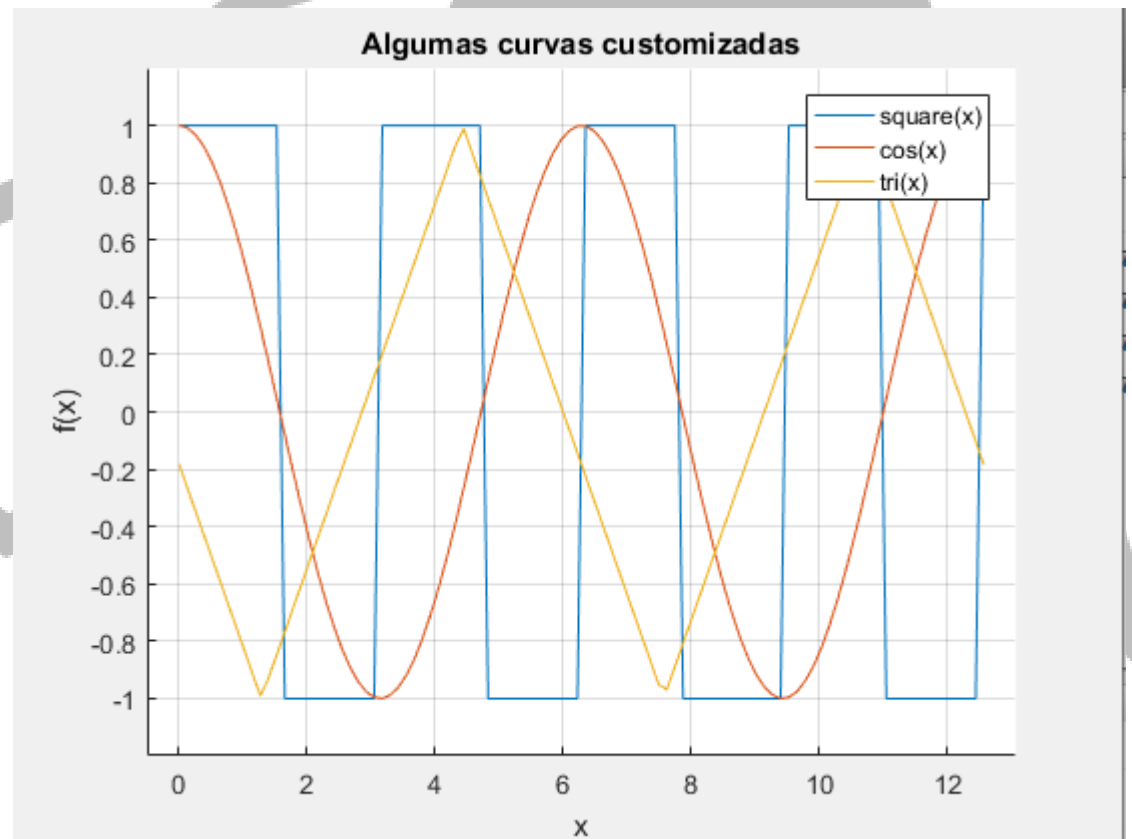
- **ALGUMAS FUNÇÕES PARA USARMOS JUNTO COM O PLOT**

Função	Uso
close	Fecha as janelas de plots abertos
hold	Mantém todos os plots na mesma figura
figure	Cria um objeto de figura
grid	Habilita linhas de grade
title( 'nome')	Adiciona um título para o plot
xlabel( 'nome')	Adiciona uma legenda no eixo x
ylabel( 'nome')	Adiciona uma legenda no eixo y
xlim([a b])	Limita o plot horizontal entre a e b
ylim([a b])	Limita o plot vertical entre a e b
legend( 'curva1','curva2',...)	Permite dar um nome para cada sinal do plot

# CRIAÇÃO DE GRÁFICOS

- ALGUMAS FUNÇÕES PARA USARMOS JUNTO COM O PLOT

```
x = linspace(0,2*pi*2,100);  
w = square(x*2);  
y = cos(x);  
z = sawtooth(x+5,0.5);  
  
hold on  
plot(x,w)  
plot(x,y)  
plot(x,z)  
grid on  
title('Algumas curvas customizadas')  
xlabel('x')  
ylabel('f(x)')  
legend('square(x)', 'cos(x)', 'tri(x)')  
xlim([-0.5 4*pi+0.5])  
ylim([min(y)-0.2 max(y)+0.2])
```



# CRIAÇÃO DE GRÁFICOS

- **PODEMOS AINDA CUSTOMIZAR O TIPO DE TRAÇO E A COR DE CADA CURVA**

- Estes atributos podem ser declarados no comando de plot

Tipos de traços
--
—
-.
.
*
o
+
x

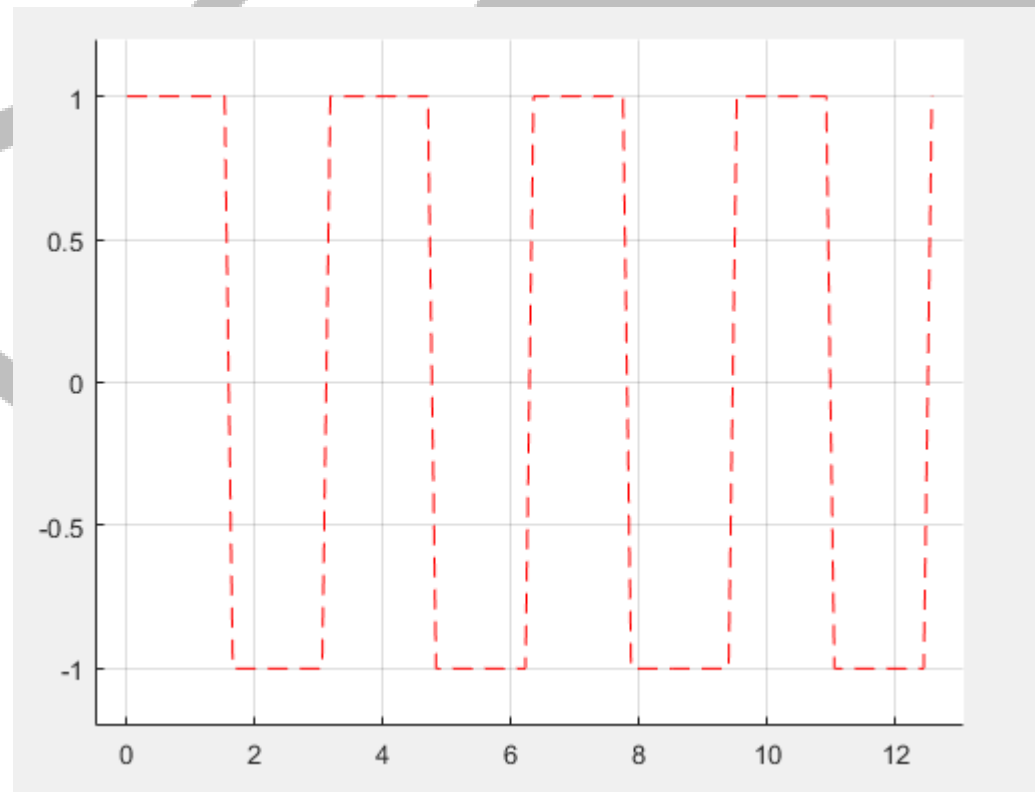
Cor	Letra
Ciano	c
Magenta	m
Amarelo	y
Preto	k
Vermelho	r
Verde	g
Azul	b
Branco	w

# CRIAÇÃO DE GRÁFICOS

- **PODEMOS AINDA CUSTOMIZAR O TIPO DE TRAÇO E A COR DE CADA CURVA**

- Estes atributos podem ser declarados no comando de plot

```
x = linspace(0,2*pi*2,100);  
w = square(x*2);  
plot(x,w, '--r')
```



# CRIAÇÃO DE GRÁFICOS

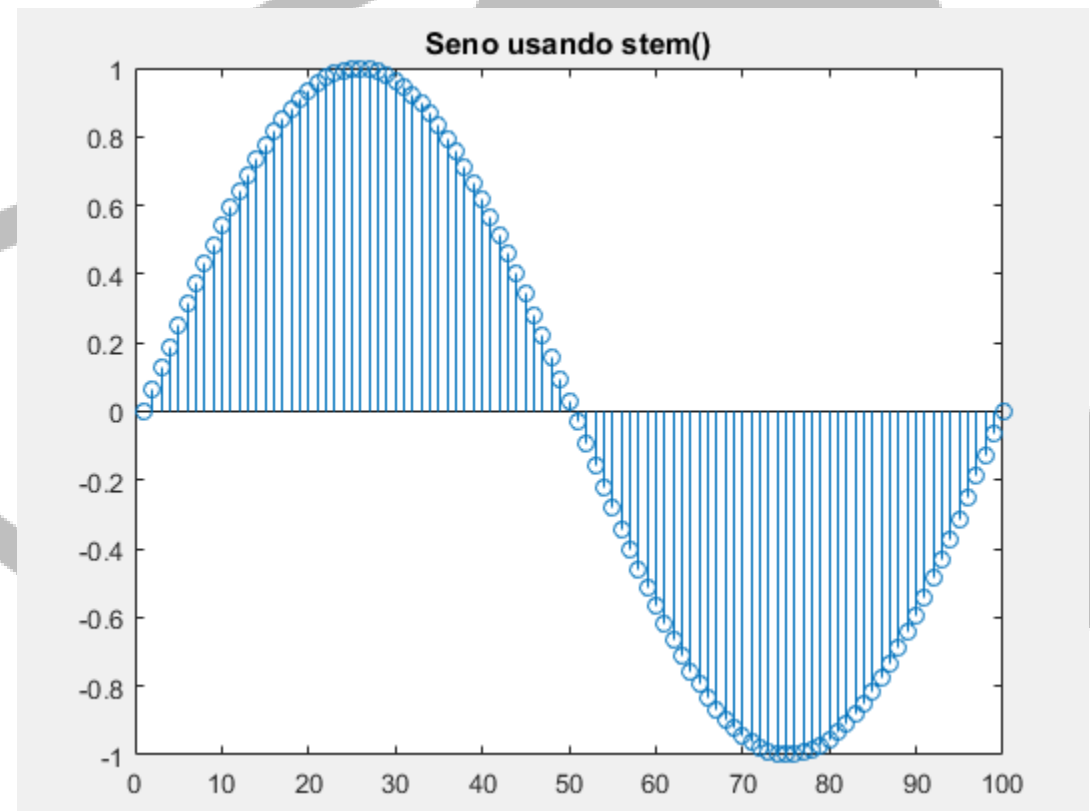
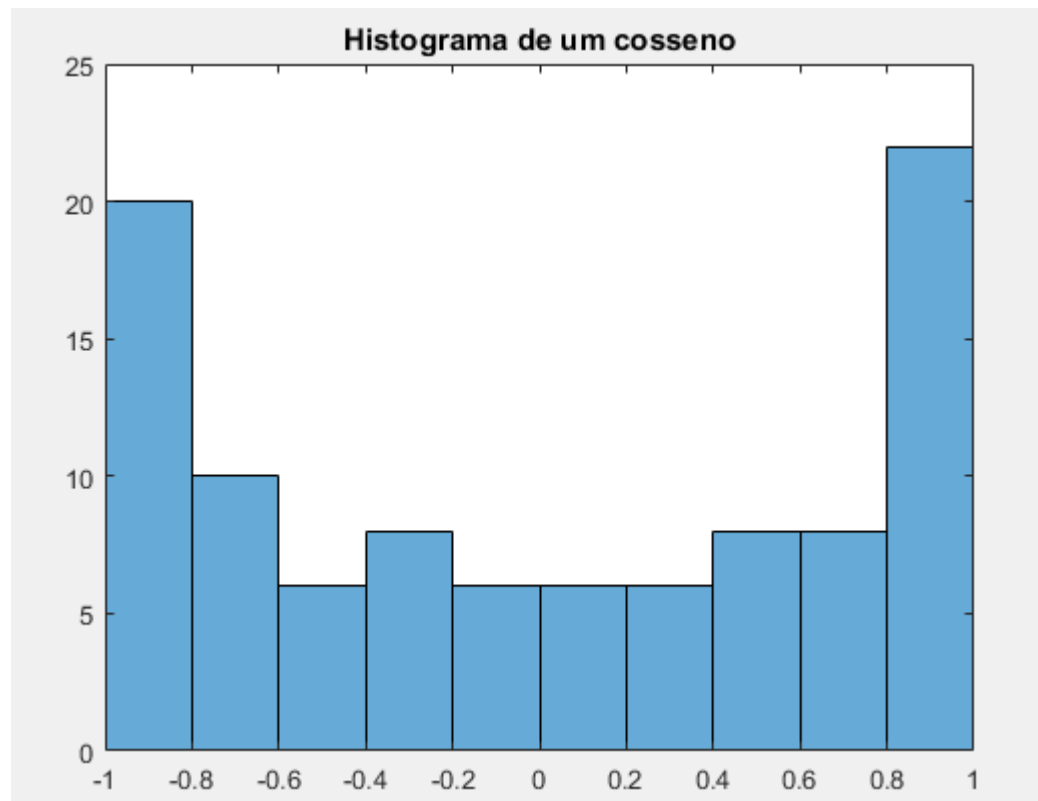
- **ALGUNS TIPOS DE GRÁFICOS 2D**

Função	Uso
plot()	Gráfico em escala linear
loglog()	Gráfico em escala log
semilogx()	Gráfico em escala log no eixo x e linear no eixo y
semilogy()	Gráfico em escala log no eixo y e linear no eixo x
polar()	Gráfico em coordenadas polares
stem()	Gráfico em escala linear de variáveis discretas
hist()	Histograma
scatter()	Gráfico de dispersão



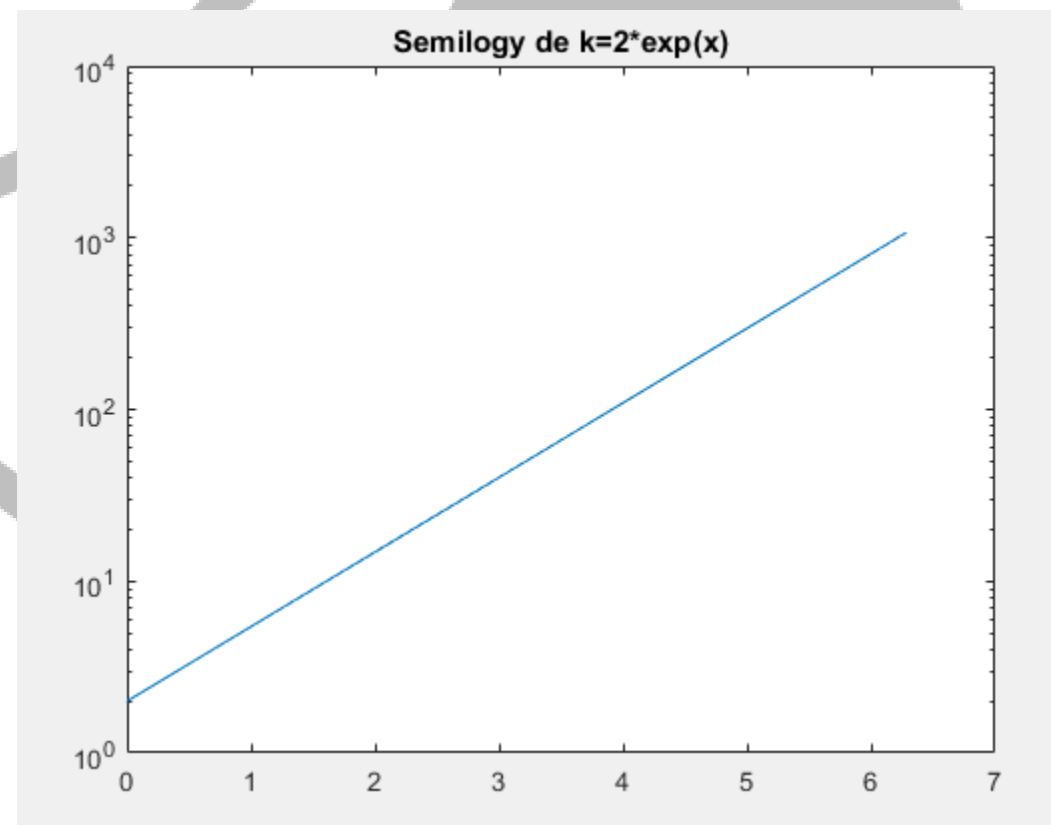
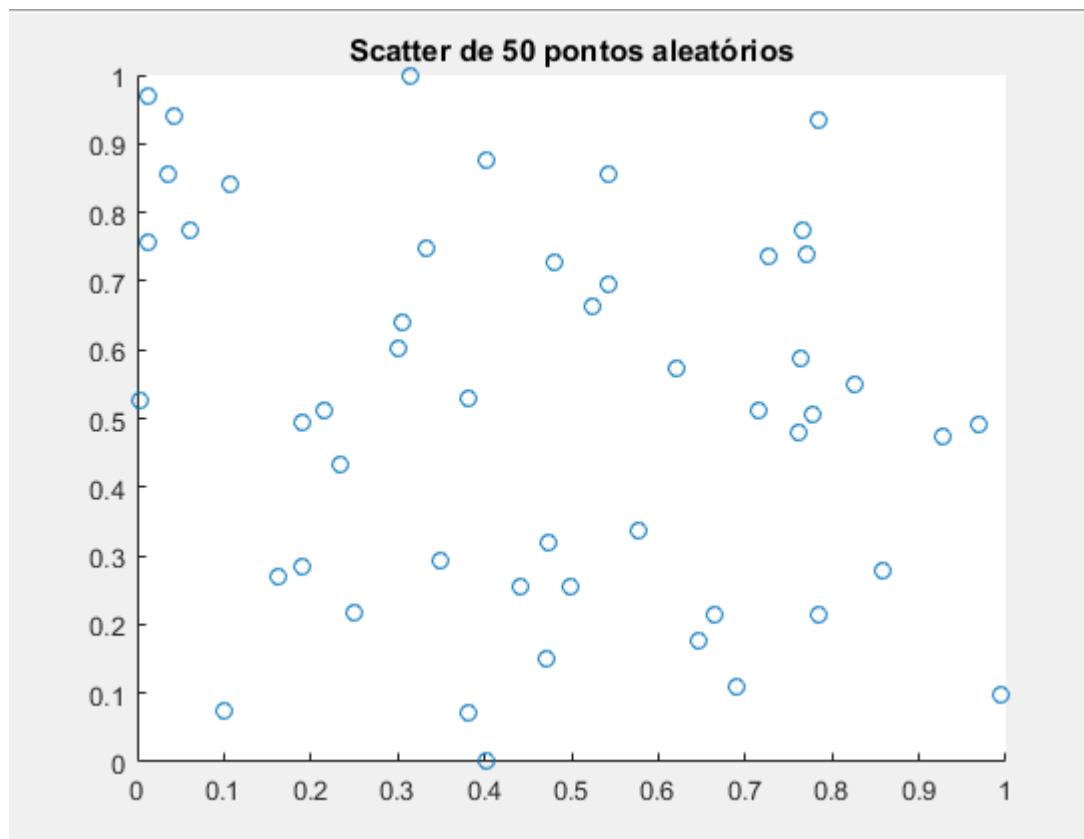
# CRIAÇÃO DE GRÁFICOS

- **ALGUNS TIPOS DE GRÁFICOS 2D**



# CRIAÇÃO DE GRÁFICOS

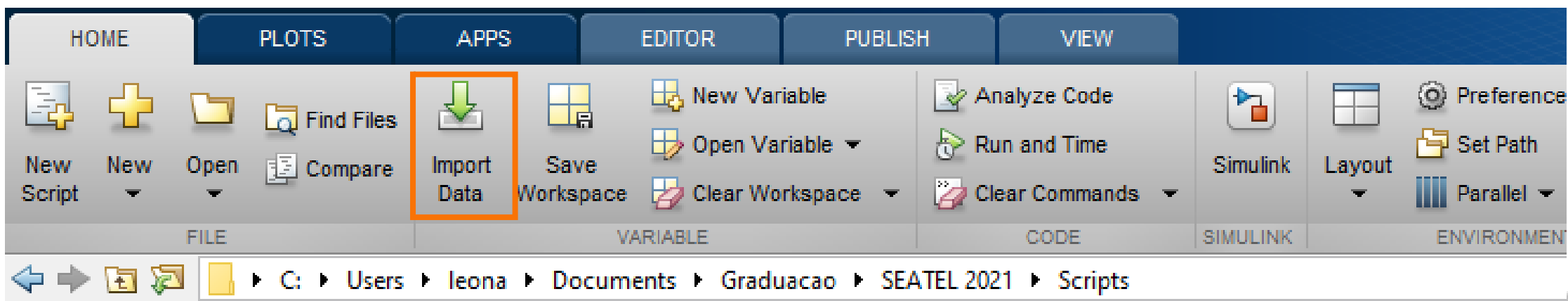
- **ALGUNS TIPOS DE GRÁFICOS 2D**



# TRABALHAR COM DADOS .TXT E .CSV

- USAR A FUNÇÃO IMPORT DATA NA ABA HOME

 MATLAB R2016b



# TRABALHAR COM DADOS .TXT E .CSV

- **USAR A FUNÇÃO IMPORT DATA NA ABA HOME**

- A partir dela, podemos abrir arquivos .txt ou .csv obtidos de outros softwares ou medições
- Podemos então selecionar de que forma queremos importar os dados
- Alguns cuidados:
  - A. É importante selecionar o delimitador correto dos dados do arquivo base
  - B. Também é interessante selecionarmos o nome que cada vetor terá
  - C. Recomendo usar a função “Column Vectors”, na qual cada coluna do arquivo base é adicionada à Workspace como um vetor e escolhermos que as linhas com dados que não podem ser importados sejam deletadas

# TRABALHAR COM DADOS .TXT E .CSV

- **USAR A FUNÇÃO IMPORT DATA NA ABA HOME**
  - Usando por exemplo o arquivo “reservatorio.csv”

Import - C:\Users\leona\Documents\Graduacao\SEATEL 2021\Scripts\reservatorio.csv

**A**

IMPORT VIEW

Delimited ☒ Column delimiters: Semicolon **C**

Fixed Width ☐ More Options

Range: A2:B46

Variable Names Row: 1

Column vectors  
Numeric Matrix  
Cell Array

Exclude rows with unimportable cells

Import Selection

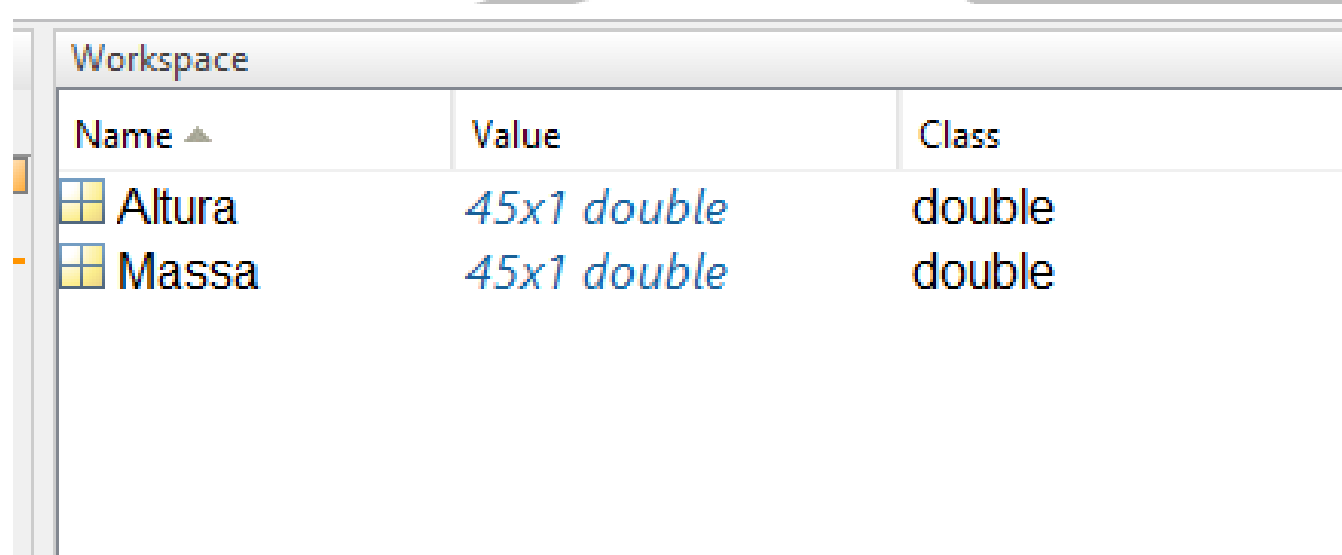
**B**

	A	B
	Altura	Massa
	NUMBER	NUMBER
1	Altura (mm)	Massa (g)
2	0	67
3	5	85
4	10	100
5	15	123
6	20	140
7	25	163
8	30	179
9	35	197
10	40	214
11	45	235

# TRABALHAR COM DADOS .TXT E .CSV

- **USAR A FUNÇÃO IMPORT DATA NA ABA HOME**

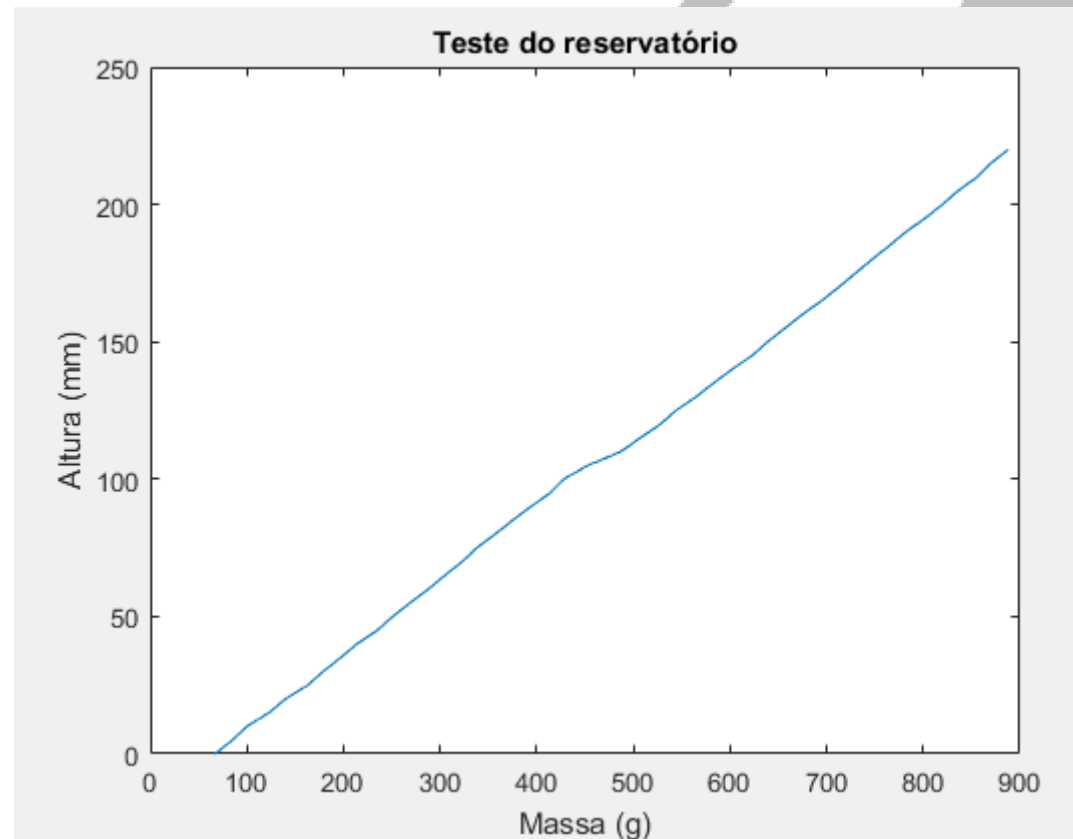
- Feito isso, os vetores serão importados para a Workspace
- Estes dados são de medições realizadas por um extensômetro usado para determinar a altura de uma coluna d'água de um reservatório, a partir do aumento da sua massa

A screenshot of the MATLAB Workspace window. The title bar says 'Workspace'. It contains a table with three columns: 'Name', 'Value', and 'Class'. There are two rows of data. The first row has 'Altura' in the Name column, '45x1 double' in the Value column, and 'double' in the Class column. The second row has 'Massa' in the Name column, '45x1 double' in the Value column, and 'double' in the Class column. To the left of the table, there is a vertical toolbar with icons for workspace operations.

Name ▲	Value	Class
Altura	45x1 double	double
Massa	45x1 double	double

# TRABALHAR COM DADOS .TXT E .CSV

- **USAR A FUNÇÃO IMPORT DATA NA ABA HOME**
  - Plotando estes vetores: `plot(Massa,Altura)`



# TRABALHAR COM DADOS .TXT E .CSV

- **USAR A FUNÇÃO IMPORT DATA NA ABA HOME**

- Uma função de regressão linear, como a “polyfit()” pode ser usada para determinar a equação da reta que se aproxima do comportamento medido
- Pelo plot, notamos que é um polinômio de grau 1

```
>> p=polyfit(Massa,Altura,1)
```

```
p =
```

```
0.2633 -16.4754
```

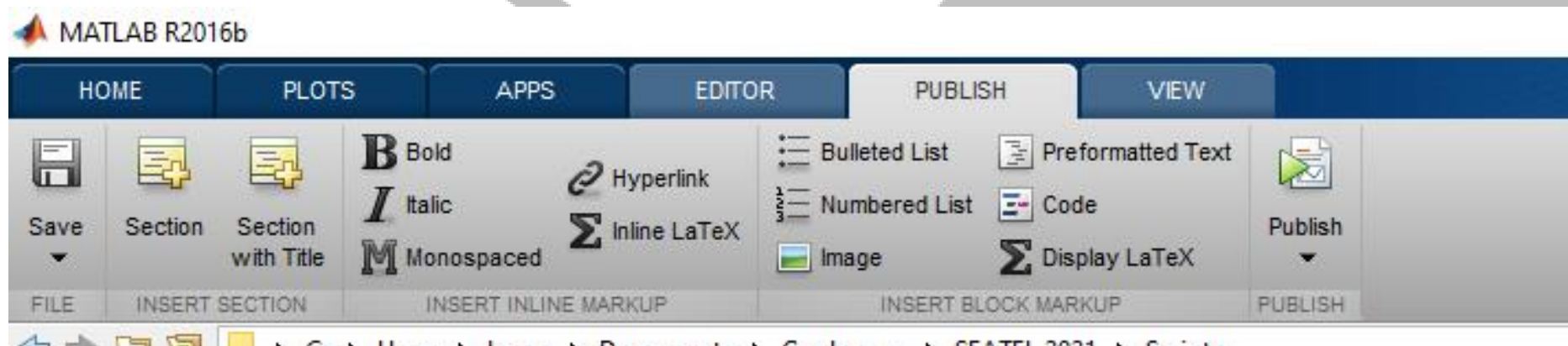
- A partir deste resultado, temos que a equação que caracteriza o sistema é:  
 $y = 0.2633 \cdot x - 16.4754$



# EXPORTAR PROJETOS COMO RELATÓRIOS

- **A FERRAMENTA PUBLISH:**

- Usando as funções desta aba, podemos inserir seções, títulos ou até fórmulas em formato LaTeX junto com o script
- Estes elementos são inseridos como comentários e usam sintaxe de markups



# EXPORTAR PROJETOS COMO RELATÓRIOS

- **A FERRAMENTA PUBLISH:**

- Estes recursos, são úteis pois permitem que o script seja exportado como um arquivo html ou pdf em formato de relatório
- Assim, podemos produzir todo um trabalho computacional em um único arquivo
- Para demonstrar isso, o script da média móvel será escrito novamente, mas agora com a intenção de gerar um pequeno relatório

# EXPORTAR PROJETOS COMO RELATÓRIOS

- A FERRAMENTA PUBLISH:

```
%% Sript para publish
% Programa desenvolvido para demonstrar funcionalidades
% MATLAB para exportar relatórios

% Funções de inicialização padrão
clc
clear all
close all
```

Em script

## Sript para publish

Programa desenvolvido para demonstrar funcionalidades MATLAB para exportar relatórios

```
% Funções de inicialização padrão
clc
clear all
close all
```

Em relatório

# EXPORTAR PROJETOS COMO RELATÓRIOS

- A FERRAMENTA PUBLISH:

```
% Utilizaremos como exemplo a função de média móvel  
% Começamos com a criação de um vetor x que vai  
% de 0 até 2*pi com 100 pontos. Em seguida, uma onda quadrada  
% é criada e passa por um canal awgn com SNR de 25 dB
```

```
x = linspace(0,2*pi*2,100);  
y = square(x);  
y_channel = awgn(y,25);
```

Em script

```
Utilizaremos como exemplo a função de média móvel  
Começamos com a criação de um vetor x que vai  
de 0 até 2*pi com 100 pontos. Em seguida, uma onda quadrada  
é criada e passa por um canal awgn com SNR de 25 dB
```

```
x = linspace(0,2*pi*2,100);  
y = square(x);  
y_channel = awgn(y,25);
```

Em relatório

# EXPORTAR PROJETOS COMO RELATÓRIOS

- A FERRAMENTA PUBLISH:

```
%%  
%  
% Antes de aplicarmos a média móvel, devemos lembrar que:  
%%  
% $f(x)=\frac{1}{L}(y(n)+y(n-1)+...+y(n-(L-1)))$  
%  
% No script, podemos fazer isso da seguinte forma:
```

Em script

Antes de aplicarmos a média móvel, devemos lembrar que:

$$f(x) = \frac{1}{L}(y(n) + y(n-1) + \dots + y(n-(L-1)))$$

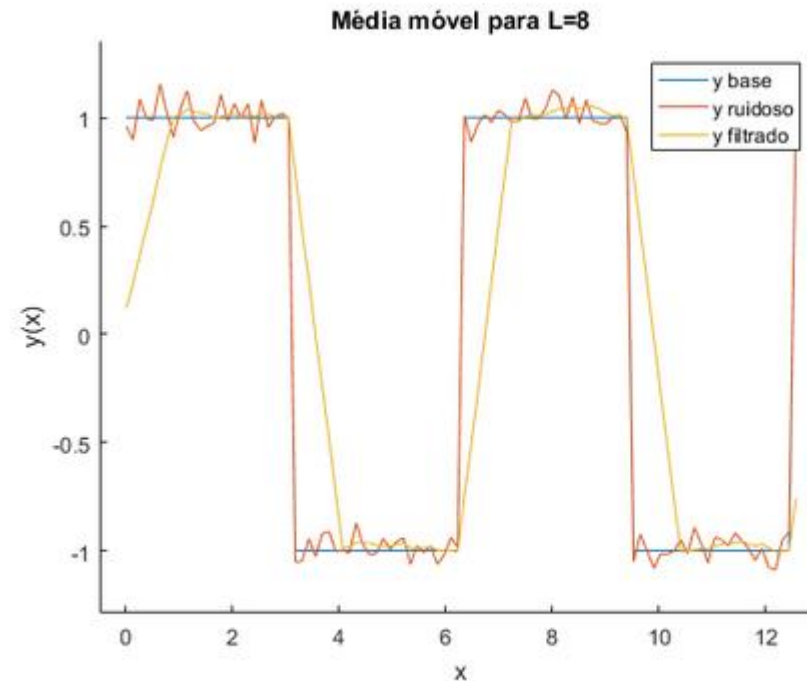
No script, podemos fazer isso da seguinte forma:

Em relatório

# EXPORTAR PROJETOS COMO RELATÓRIOS

- **A FERRAMENTA PUBLISH:**

- Neste formato, os plots ficam logo em seguida dos blocos de código em que a figura foi criada

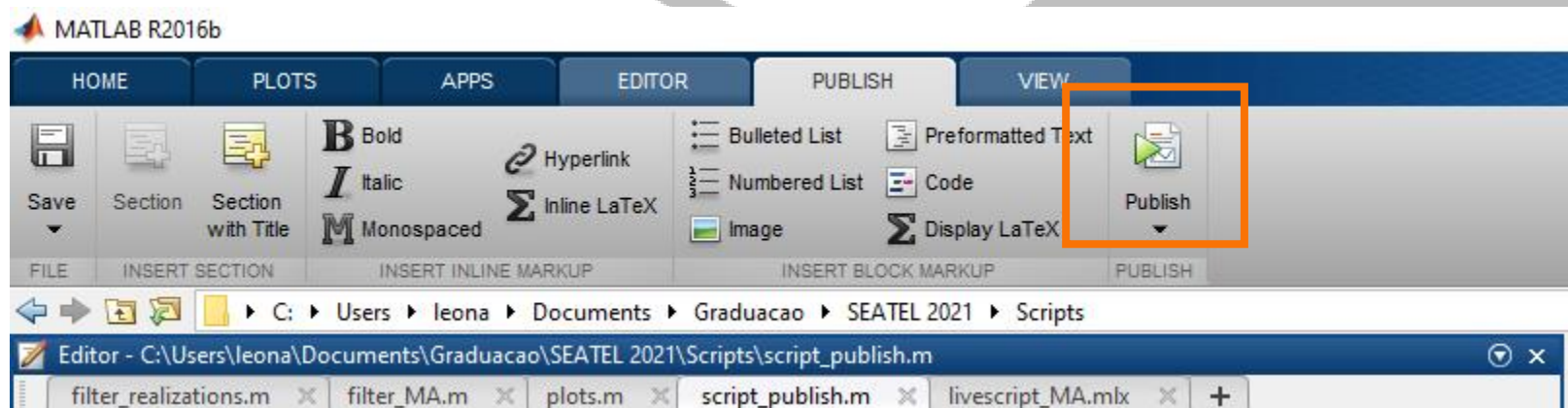


Conclusão: Conforme esperado, conforme maior a janela mais suave a curva, porém maior o atraso também.

# EXPORTAR PROJETOS COMO RELATÓRIOS

- **A FERRAMENTA PUBLISH:**

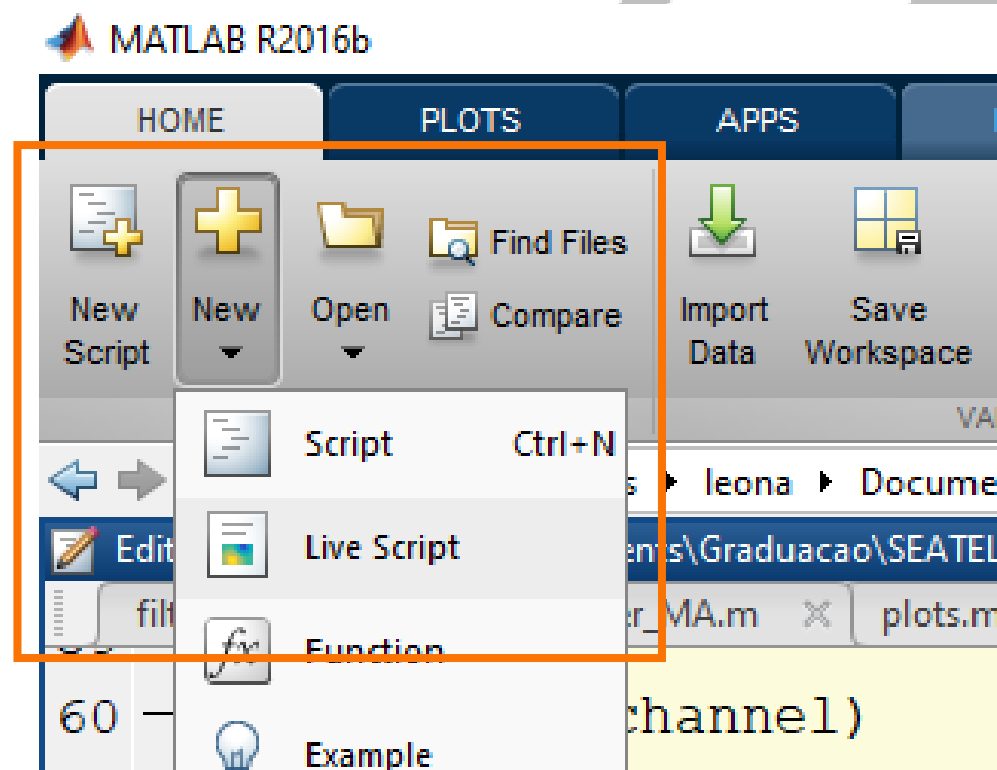
- Com o script finalizado podemos exportá-lo para html clicando no botão “Publish”
- O relatório será salvo em html no diretório do script e uma janela será aberta para visualizar o arquivo gerado
- Podemos então imprimí-lo como pdf



# EXPORTAR PROJETOS COMO RELATÓRIOS

- **A FERRAMENTA LIVE SCRIPT:**

- Outra alternativa é o uso do Live Script, que é um arquivo de extensão “.mlx”

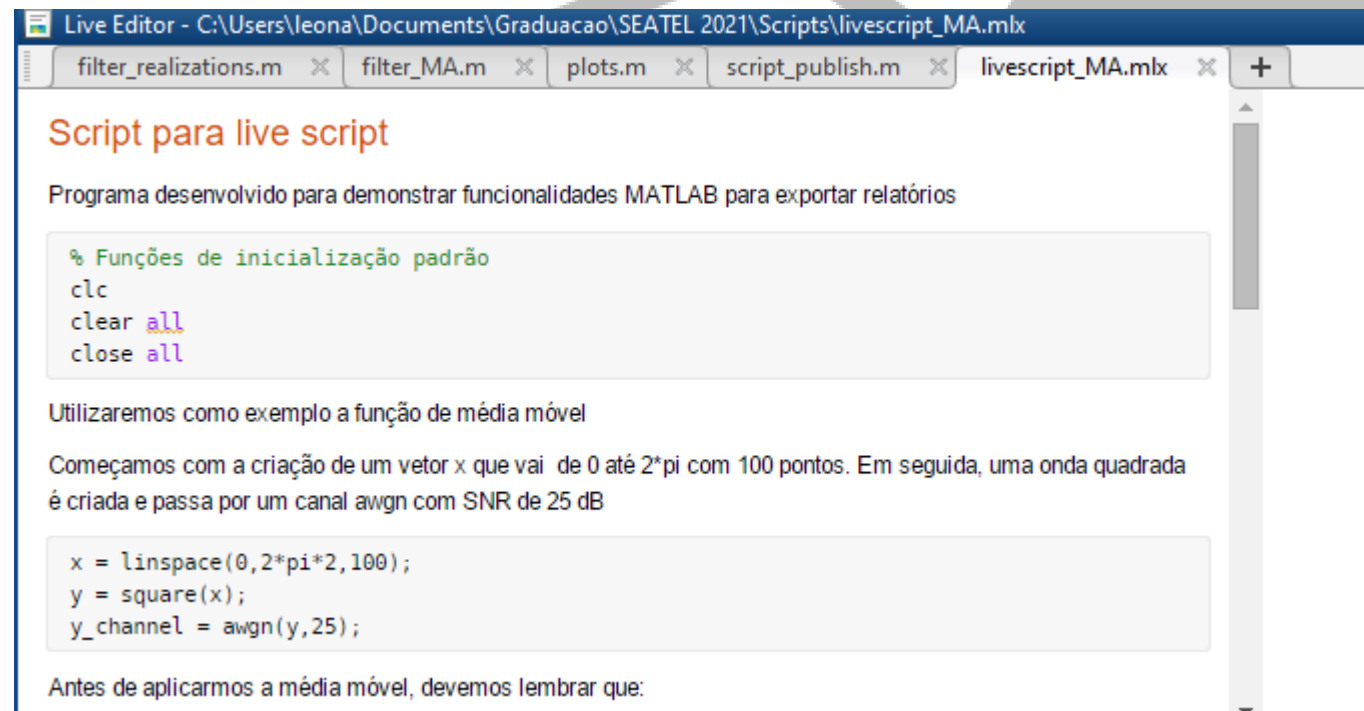




# EXPORTAR PROJETOS COMO RELATÓRIOS

- **A FERRAMENTA LIVE SCRIPT:**

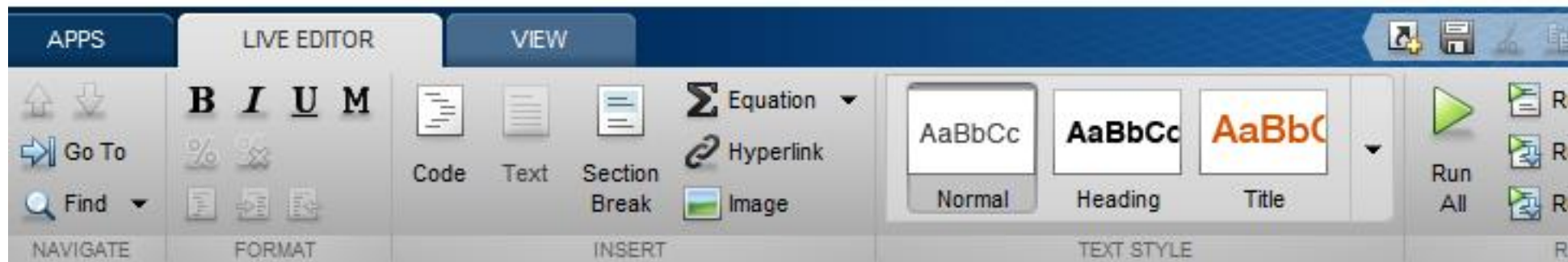
- O script criado como live script é editado na janela “Live Editor”, que é compartilhada pelo Editor
- A interface desta ferramenta é mais intuitiva e similar ao relatório que será produzido



# EXPORTAR PROJETOS COMO RELATÓRIOS

- **A FERRAMENTA LIVE SCRIPT:**

- A aba superior “Live Editor” contém tanto funções de edição do relatório, quanto de execução do script



# EXPORTAR PROJETOS COMO RELATÓRIOS

- **A FERRAMENTA LIVE SCRIPT:**

- Vantagens em relação ao Publish:
  - ✓ Interface mais intuitiva
  - ✓ Scripts criados são mais limpos, pois não precisam do uso de diversos %
  - ✓ Equações podem ser criadas em LaTeX ou com um construtor de equações similar ao do pacote Office
  - ✓ Edição de texto similar ao Word
  - ✓ Arquivo PDF pode ser exportado diretamente no menu “Save”



**OBRIGADO!**  
**DÚVIDAS?**