

Curso de Controle Clássico

Prof: Dennis

Aula Prática #02C – Simulação de sistema malha fechada com toolbox de controle do python.

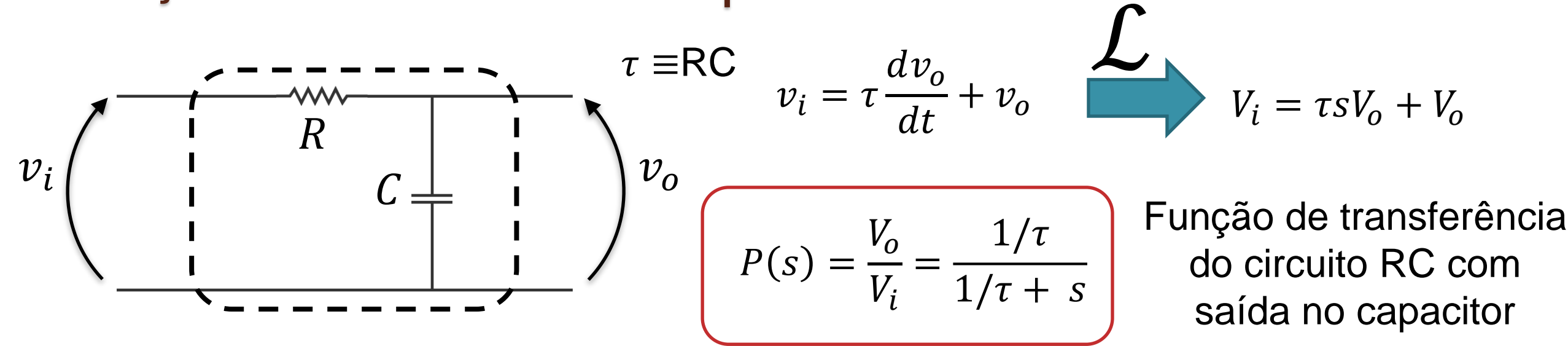
O que faremos nesta aula

- Nesta aula voltaremos a usar o python e a toolbox “control” ,gratuitos, para encontrar a resposta em malha fechada de um sistema;
- Existem diversos compiladores python, pagos e gratuitos, inclusive online, que lhe permitirão realizar essas simulações mesmo de um smartfone ou tablet.
- Por simplicidade usaremos uma ferramenta web, o “google colab”;
- Uma vantagem de usar o python, é que se trata de uma linguagem com muito mais bibliotecas e possibilidades do que o Scilab.

O que faremos nesta aula

- O exemplo escolhido para as simulações é um circuito RC, nas aulas anteriores vimos com usar o python para encontrar a resposta em malha aberta;
- Também já vimos como obter a resposta ao degrau em malha fechada:
 - Graficamente, montando o diagrama do sistema no Xcos do Scilab, como faríamos no simulink do Matlab;
 - Diretamente usando um código na linguagem do Scilab;
- Veremos nesta aula como usar um código em python para fazer a mesma simulação.
- Veremos como instalar a toolbox de controle e como fazer a simulação;
- Novamente obteremos o mesmo resultado, que os dois métodos usando o Scilab, ficando a escolha do estudante qual dos três deseja usar;

Função de transferência da planta:



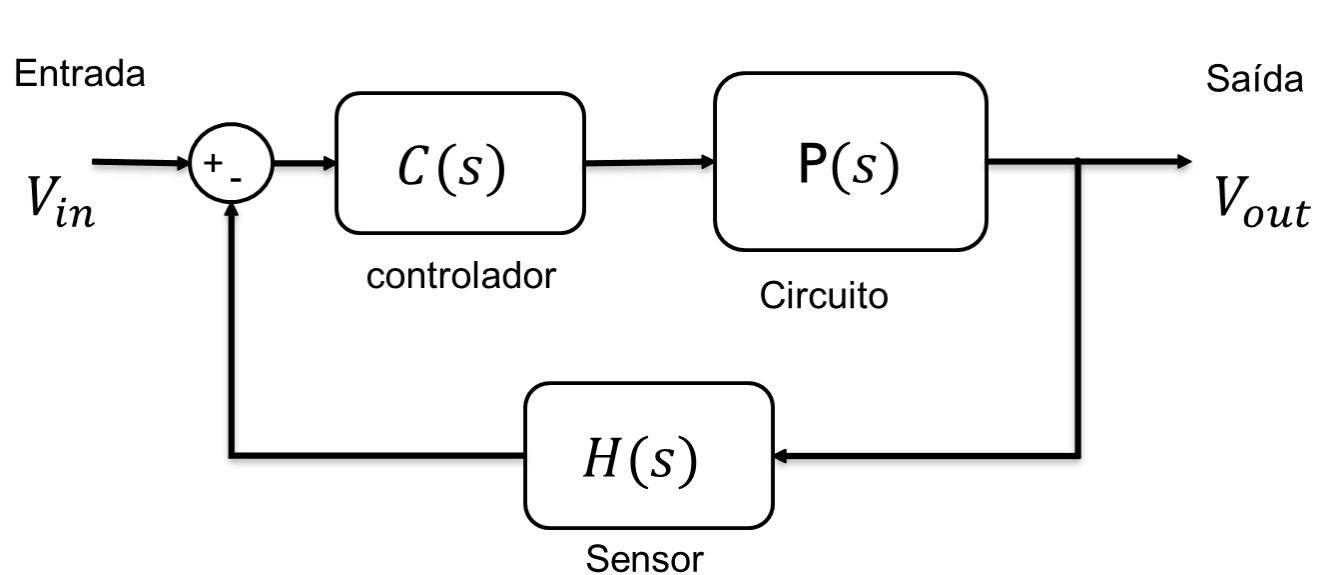
Capacitor: $C = 10\mu F$
Resistor: $R = 20k\Omega$

$$\tau = 2 \times 10^{-1} \text{ seg}$$

$$P(s) = \frac{1/(2 \times 10^{-1})}{1/(2 \times 10^{-1}) + s} = \frac{5}{5 + s}$$

Malha fechada:

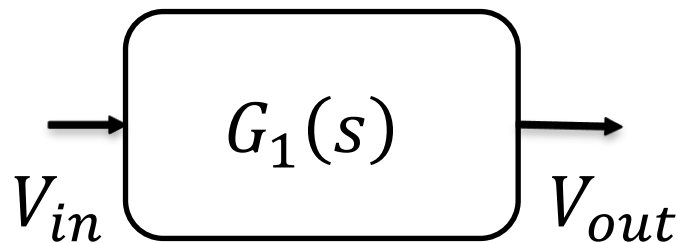
- Para fechar a malha, vamos precisar de um controlador, um sensor e um bloco para fazer a subtração:



Circuito: $\Rightarrow P(s) = \frac{5}{5 + s}$

Sensor: $\Rightarrow H(s) = 1$

Controlador: $\Rightarrow C(s) = K = 3$



$$G_1(s) = \frac{C(s)P(s)}{1 + C(s)P(s)H(s)} = \frac{3 \cdot \frac{5}{5 + s}}{1 + 3 \cdot \frac{5}{5 + s} \cdot 1} = \frac{15}{20 + s}$$

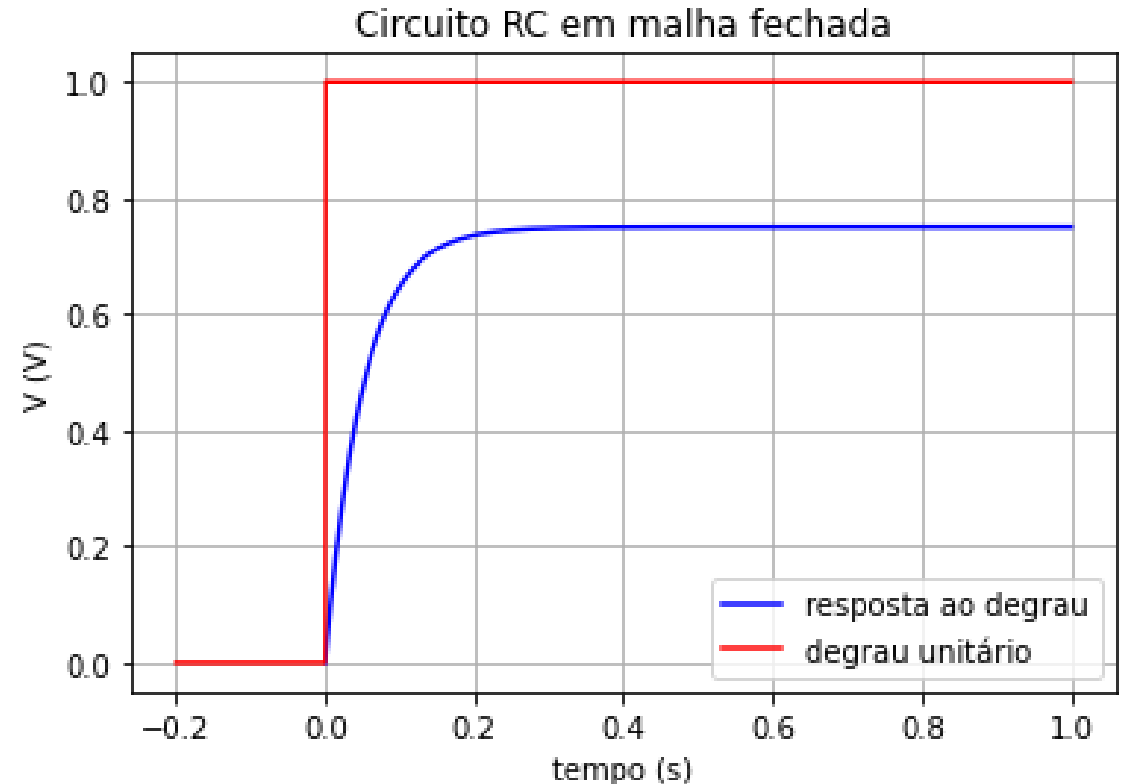
Biblioteca “control” do python

- É uma toolbox desenvolvida para utilizar as ferramentas numéricas do python, como o “numpy” e o “scipy” e de gráficos como o “matplotlib” aplicados a sistemas de controle;
- Seu site é: <https://python-control.readthedocs.io/en/0.9.0/>
- Deve ser instalada antes de ser usada:
- Sugestão use o comando: **pip install control**

Código do python

```
import control as ctl
import matplotlib.pyplot as plt
import numpy as np
# cria a função de transferência em malha aberta
R = 20.0e3; C=10.0e-6; tau=R*C;
Tsim=1.;
# cria a função de transferência em malha aberta
numerador = [1/tau]; denominador = [1., 1/tau]
P_s = ctl.tf(numerador, denominador)
print(' FT malha aberta= ',P_s)
# cria a função de transferência do controlador
C_s=ctl.tf([3.],[1.])
print(' FT controlador= ',C_s)
# cria a função de transferência do sensor
H_s=ctl.tf([1.],[1.])
#Funcao de transf MF
#G1_s=(C_s*P_s)/(1+C_s*P_s*H_s)
G_s=ctl.series(C_s, P_s);
G1_s=ctl.feedback(G_s, H_s, sign=-1);
print(' FT malha fechada= ',G1_s)
#calcula a resposta ao degrau
T_mf, yout_mf = ctl.step_response(G1_s, Tsim)
```

```
#calcula um degrau unitário
T2=np.linspace(-0.2,Tsim,1000)
degrau=np.ones_like(T2)
degrau[T2<0]=0;
#plota os resultados
plt.plot(T_mf,yout_mf,'b-')
plt.plot(T2,degrau,'r-')
plt.ylabel('V (V)'); plt.xlabel('tempo (s)')
plt.legend(['resposta ao degrau','degrau unitário'])
plt.title('Circuito RC em malha fechada'); plt.grid()
```



Comandos importantes:

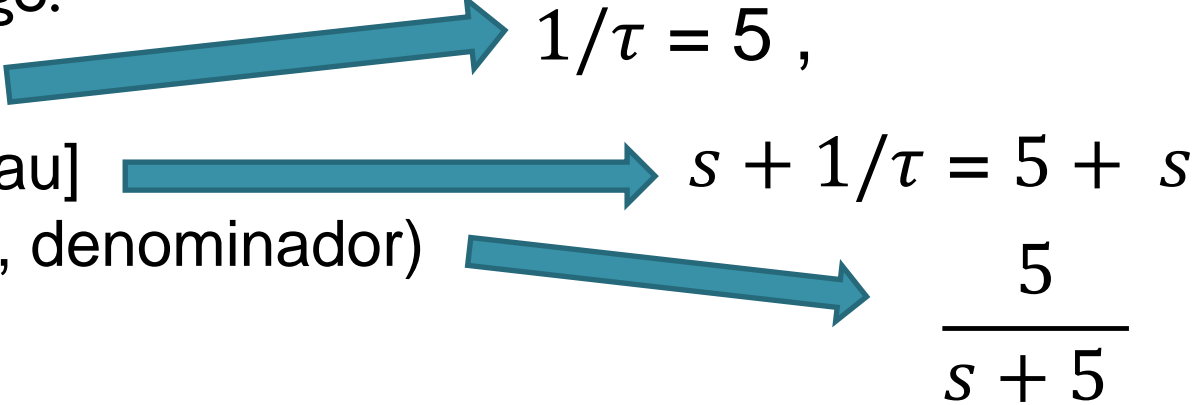
ctl.tf (ou control.tf)

- O comando “**control.tf**” defini um sistema linear em python (já explicado em aula anterior)
- Exemplo do nosso código:

numerador = [1/tau];

denominador = [1., 1/tau]

P_s = ctl.tf(numerador, denominador)



$$1/\tau = 5,$$

$$s + 1/\tau = 5 + s$$


$$\frac{5}{s + 5}$$

ctl.step_response (ou control.step_response)

- O comando “**ctl.step_response**” calcula a resposta ao degrau a partir de uma função de transferência; (já explicado em aula anterior)

Tsim=10;  número que define o tempo final da simulação;

T_mf, yout_mf = ctl.step_response(G1_s, Tsim)

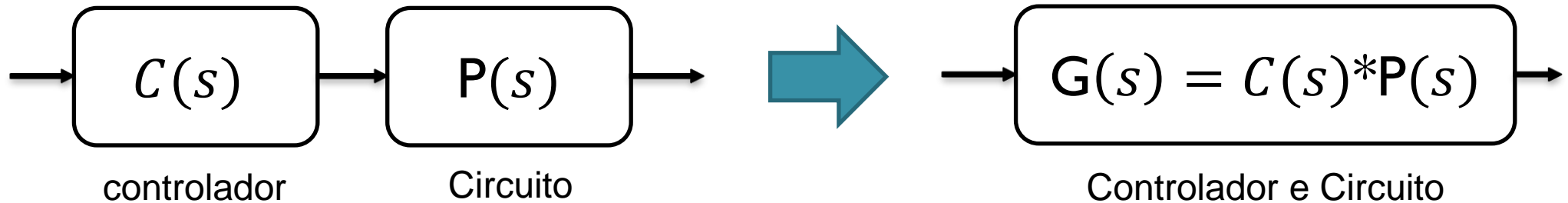


yout_mf: Resposta ao degrau da função de transferência G1_s de 0 até 10s;
T_mf: vetor de tempos correspondentes aos valores da resposta yout;

Comandos de diagrama de blocos: `ctl.series` (ou `control.series`)

- “`ctl.series`” associa duas funções de transferência, ou blocos, em série:
- Exemplo do nosso código:

```
G_s=ctl.series(C_s, P_s);
```

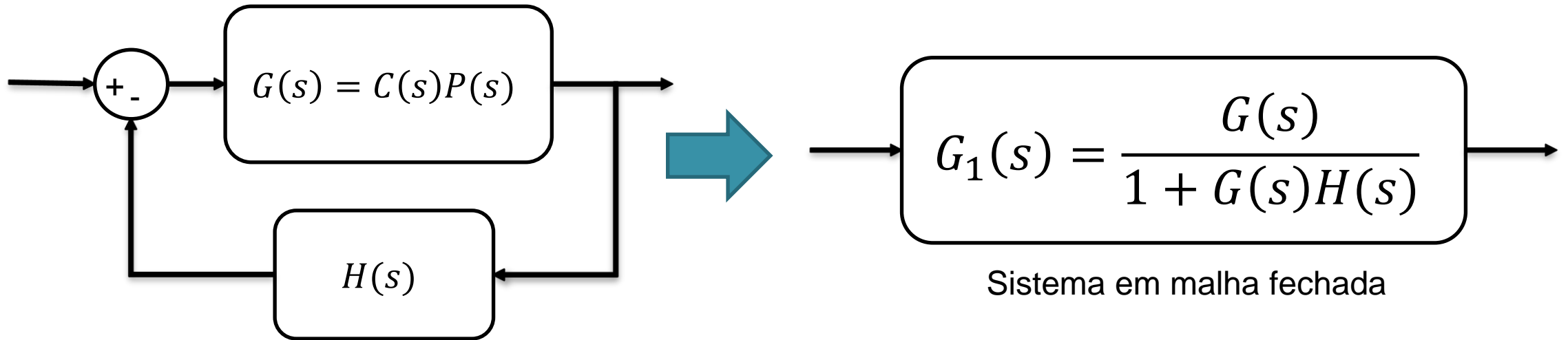


Obs: O python permite um resultado equivalente a essa função pode ser obtido simplesmente multiplicando as funções de transferência:

Comandos de diagrama de blocos: `ctl.feedback`(ou `control.feedback`)

- “`ctl.feedback`” associa duas funções de transferência, em uma malha de realimentação, seja positiva ou negativa;
- Exemplo do nosso código:

`G1_s=ctl.feedback(G_s, H_s, sign=-1);` \Rightarrow `sign = -1` \Rightarrow Realimentação negativa



Obs: O python permite um resultado equivalente fazendo a equação diretamente:

```
G_s=ctl.series(C_s, P_s);  
G1_s=ctl.feedback(G_s, H_s, sign=-1);
```



$G1_s = (C_s * P_s) / (1 + C_s * P_s * H_s)$

Entendendo o Código em python

```
import control as ctl  
import matplotlib.pyplot as plt  
import numpy as np
```

} → *importa as bibliotecas (toolbox) necessárias*

```
# cria a função de transferência em malha aberta → comentário
```

```
R = 20.0e3; C=10.0e-6; tau=R*C; → cria as variáveis R, C e tau e dá valor a elas
```

```
Tsim=1.;
```

```
# cria a função de transferência em malha aberta
```

```
numerador = [1/tau]; denominador = [1., 1/tau] → cria os vetores que serão os polinômios: "1/tau" e "1/tau + s"
```

```
P_s = ctl.tf(numerador, denominador) → usa os vetores para criar P_s
```

```
print(' FT malha aberta= ', P_s) → Mostra a função de transferência em MA
```

```
# cria a função de transferência do controlador
```

```
C_s=ctl.tf([3.],[1.]) → cria a função de transferencia do controlador C_s = 3/1
```

```
print(' FT controlador= ', C_s)
```

```
# cria a função de transferência do sensor
```

```
H_s=ctl.tf([1.],[1.]) → cria a função de transferencia do sensor H_s = 1/1
```

```
#Funcao de transf MF
```

```
#G1_s=(C_s*P_s)/(1+C_s*P_s*H_s)
```

```
G_s=ctl.series(C_s, P_s);
```

```
G1_s=ctl.feedback(G_s, H_s, sign=-1);
```

```
print(' FT malha fechada= ', G1_s) → Mostra a função de transferência em MF
```

```
#calcula a resposta ao degrau
```

```
T_mf, yout_mf = ctl.step_response(G1_s, Tsim) → calcula a resposta ao degrau (step) para Tsim e G1_s
```

} → *Constroi a FT em Malha fechada a partir das demais FT*
(deixei comentada a outra forma de construir essa FT em MF)

Entendendo o Código em python

```
#calcula um degrau unitário
T2=np.linspace(-0.2,Tsim,1000)
degrau=np.ones_like(T2)
degrau[T2<0]=0;
#plota os resultados
plt.plot(T_mf,yout_mf,'b-')
plt.plot(T2,degrau,'r-')
plt.ylabel('V (V)'); plt.xlabel('tempo (s)')
plt.legend(['resposta ao degrau','degrau unitário'])
plt.title('Circuito RC em malha fechada'); plt.grid()
```

→ *calcula um degrau unitário, de - 0.2 até Tsim, para comparar com a saída*

→ *Plota a resposta ao degrau*

→ *Plota o degrau (pode parar aqui, se quiser)*

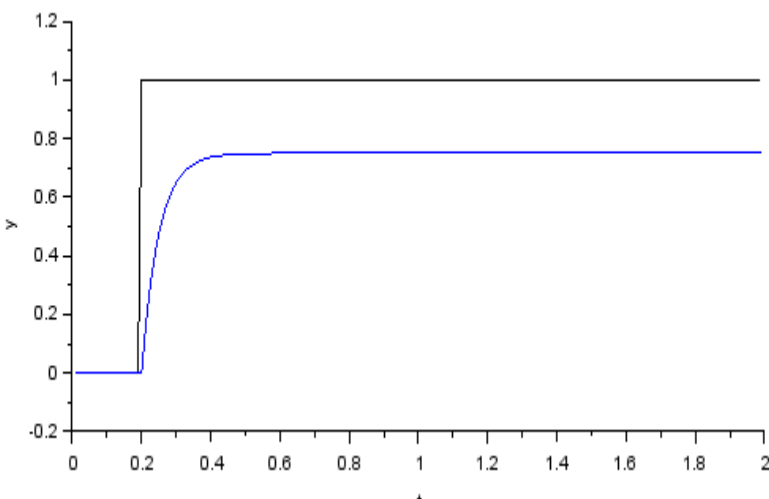
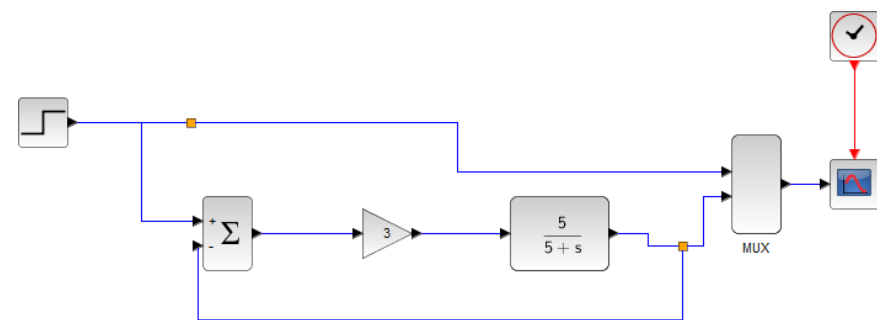
→ *coloca títulos no gráfico e nos eixos, uma legenda e um grid para deixar o gráfico mais apresentável*

Agora vamos ver como executar esse código 100% online com o google colab

Comparando Xcos com Scilab e Python

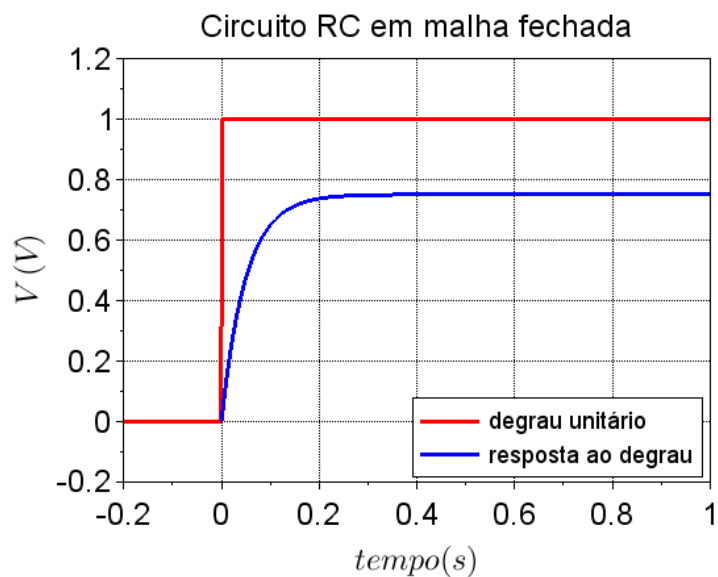
- Note que o resultado é mesmo. São três caminhos bem distintos. Então, qual você vai usar?

Parte A da aula: Xcos



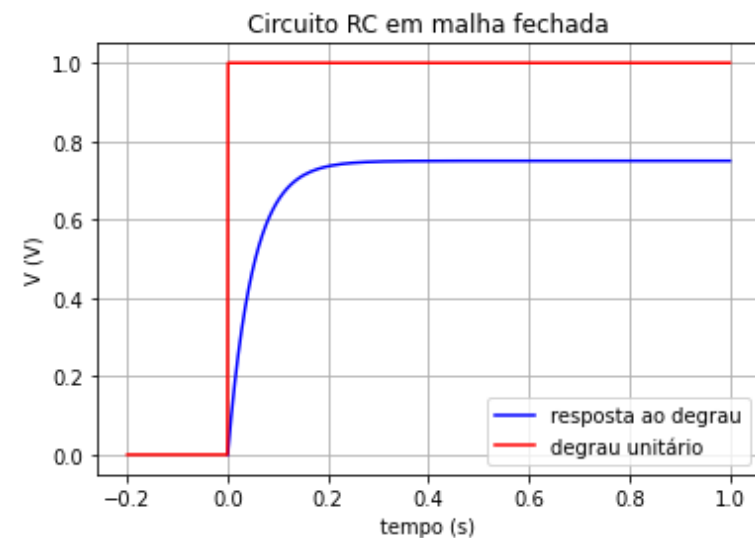
Parte B da aula: Toolbox do scilab

```
P_s = syslin('c',numerador,denominador)
// cria a função de transferência do controlador
C_s=poly([3],'s','c');
// cria a função de transferência do sensor
H_s=poly([1],'s','c');
//calcula a função de transferência em malha fechada
G1_s=(C_s*P_s)/(1+C_s*P_s*H_s)
...
```



Parte C da aula: Toolbox do python

```
H_s=ctl.tf([1.],[1.])
#Funcao de transf MF
#G1_s=(C_s*P_s)/(1+C_s*P_s*H_s)
G_s=ctl.series(C_s, P_s);
G1_s=ctl.feedback(G_s, H_s, sign=-1);
```

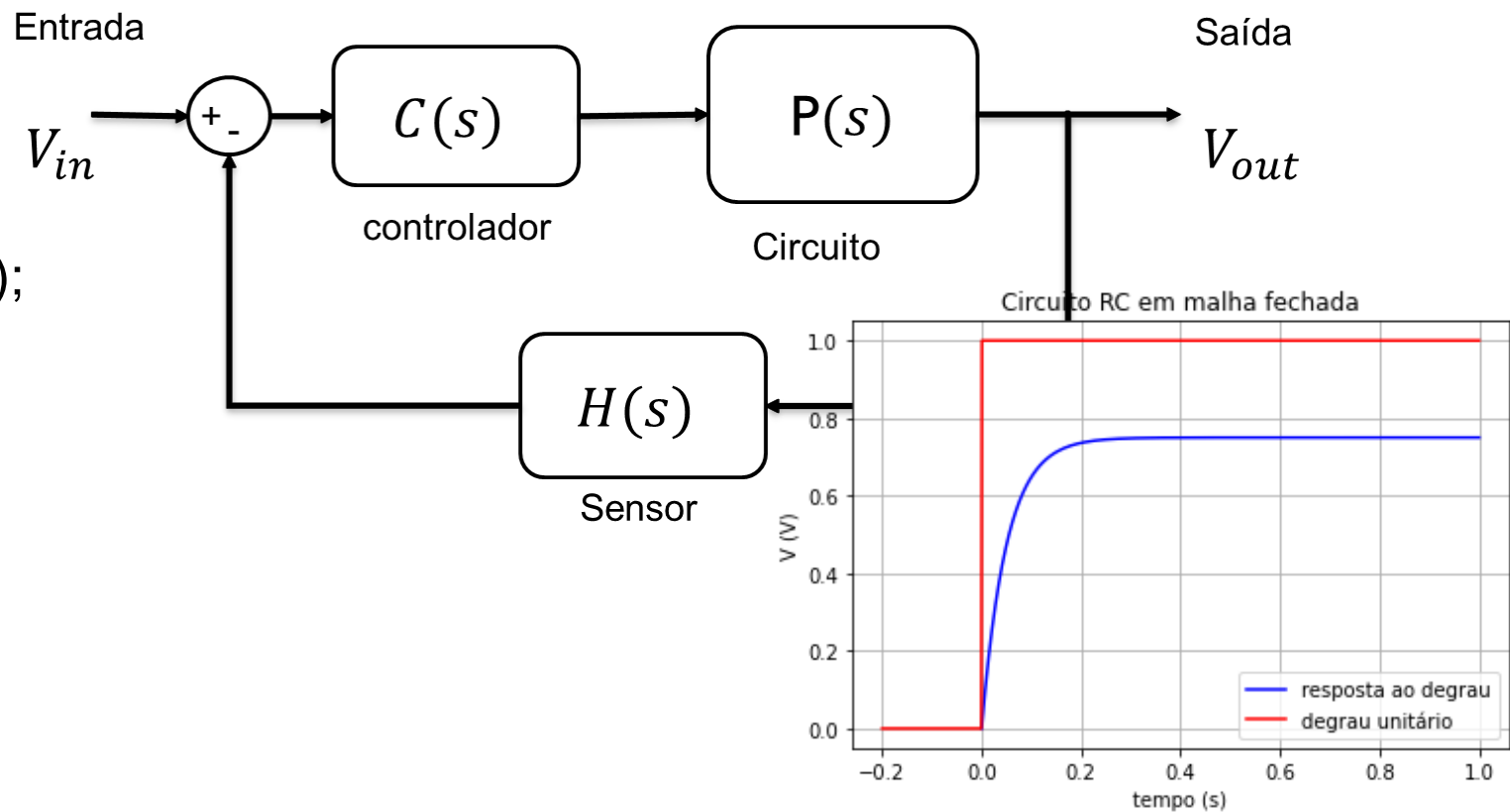


```
H_s=ctl.tf([1.],[1.])  
#Funcao de transf MF  
#G1_s=(C_s*P_s)/(1+C_s*P_s*H_s)  
G_s=ctl.series(C_s, P_s);  
G1_s=ctl.feedback(G_s, H_s, sign=-1);
```

Python



Simulação de controle 100% online e free!



Controle na prática #02C: Simulação de sistema malha fechada com toolbox de controle do Python.