LAB5 – Projeto de Controlador PI Digital

PTC 2619/PTC 3418 – Laboratório de Automação
1º semestre de 2017
Bruno Angélico

Laboratório de Automação e Controle
Departamento de Engenharia de Telecomunicações e Controle
Escola Politécnica da Universidade de São Paulo

Objetivo

- Controle PI digital:
  - Projeto em tempo contínuo (plano-*s*) seguido de discretização;
  - Projeto direto em tempo discreto (plano-*z*);
  - Implementar sistema anti-windup.

Planta

- Controle de Velocidade do Servomecanismo.
- Modelo nominal linear:

$$G(s) = \frac{KK_t}{Ts + 1}$$

- Assuma T = 0,3 s, K = 50, K_t = 0,017.

Discretização do PI Contínuo

- PI contínuo:

$$C(s) = K_p\left(1 + \frac{1}{T_i s}\right)$$

- Discretização da parcela integrativa por *Tustin*:

$$s = \frac{2}{T_s}\frac{(z-1)}{(z+1)}$$

- PI Posicional:

$$u_P(t) = K_P\, e(t) \quad \Rightarrow \quad u_P[n] = K_P\, e[n]$$

$$u_I(t) = \frac{K_P}{T_I}\int_0^t e(\tau)\, d\tau \quad \Rightarrow \quad u_I[n] = u_I[n-1] + \frac{K_P T_s}{2T_I}(e[n] + e[n-1])$$

$$u[n] = u_P[n] + u_I[n]$$

Discretização do PI Contínuo

- PI Incremental:

$$\Delta u[n] = u[n] - u[n-1] = \Delta u_P[n] + \Delta u_I[n]$$

$$u[n] = u[n-1] + \Delta u_P[n] + \Delta u_I[n]$$

As parcelas incrementais proporcional e integral são:

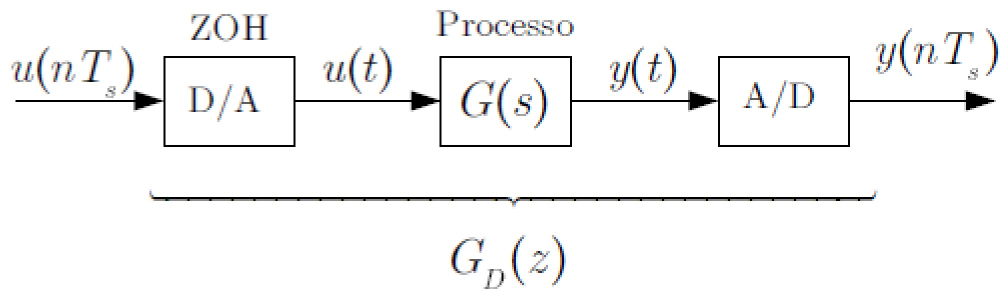
$$\Delta u_P[n] = u_P[n] - u_P[n-1] = K_P(e[n] - e[n-1]),$$

$$\Delta u_I[n] = \frac{K_P T_s}{2T_I}(e[n] + e[n-1])$$

- Desvantagem da discretização direta: efeito do atraso do ZOH não é considerado no controle.

Discretização do PI Contínuo

- Projeto no plano-*z*:



Equivalente discreto:

$$G_D(z) = \left(1 - z^{-1}\right) \mathcal{Z}\left\{\frac{G(s)}{s}\right\}$$

No Matlab: `G_D = c2d(G, T_s, ‘zoh’).`

Ao aplicar no modelo nominal, tem-se

$$G_D(z) = \frac{K K_t(1 - e^{-T_s/T})}{z - e^{-T_s/T}}$$

Discretização do PI Contínuo

A função de transferência do PI discreto é dada por:

$$C_D(z) = K_c \frac{z - a}{z - 1}$$

Ao escolher *a* para cancelar o polo em $z = e^{-T_s/T}$, tem-se em malha aberta:

$$G_{D_{MA}}(z) = \frac{K_c K K_t (1 - e^{-T_s/T})}{z - 1}$$

Em malha fechada:

$$G_{D_{MF}}(z) = \frac{K_c K K_t (1 - e^{-T_s/T})}{z - (1 - K_c K K_t (1 - e^{-T_s/T}))}$$

Discretização do PI Contínuo

Se a constante de tempo desejada em malha for igual a τ, então deve-se resolver a seguinte equação a fim de encontrar o ganho *K* do controlador:

$$e^{-T_s/\tau} = 1 - K_c K K_t (1 - e^{-T_s/T})$$

Note que a equação de diferenças que implementa o controlador possui, naturalmente, a forma incremental:

$$u[n] = u[n-1] + K_c \left(e[n] - a e[n-1] \right)$$

Discretização do PI Contínuo

- PI incremental com anti-windup: uma forma simples consiste em “congelar” a ação de controle quando há saturação:

```
% PID incremental com anti-windup
e[n] = r[n]-y[n];
Du_P = K_P*(e[n]-e[n-1]);
Du_I=(K_P*T_s/(2*T_I))*(e[n] + e[n-1]);
u[n] = u[n-1] + Du_P + Du_I;
% PID incremental com anti-windup (v.2)
if (u[n] ≥ sat)
u[n] = sat;
elseif (u[n] ≤ - sat)
u[n] = -sat;
end
```

Discretização do PI Contínuo

- Atividades Prévias:

Projete um controlador PI assumindo que a constante de tempo em malha fechada seja $\tau = 0{,}8T$. Considere $Ts = 1/20$ s.

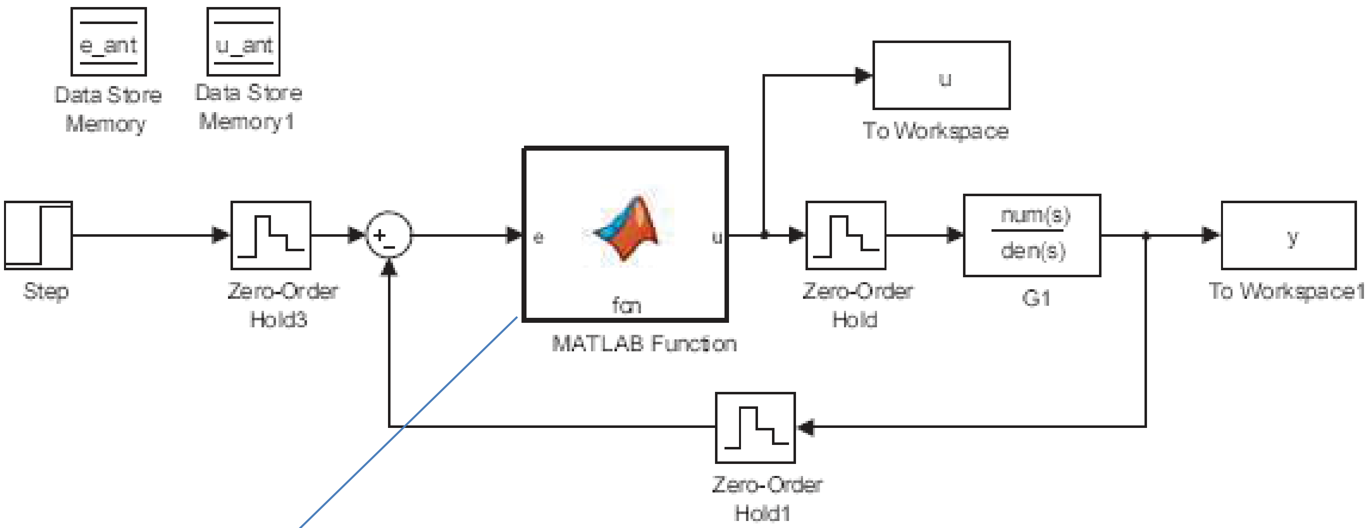
- Atividades Práticas (simulação e planta real):
  a) Implemente PI digital posicional. Discretize o controlador com $Ts = 1/20$ s. Rode o experimento por 10 s. Como entrada, considere o degrau de amplitude 3V aplicado em 1 segundo.
  b) Implemente o controlador PI incremental. Discretize o controlador com Ts = 1/20 s. Rode o experimento por 10 s. Como entrada, considere o degrau de amplitude 3V aplicado em 1 segundo.

Discretização do PI Contínuo

c) Implemente o controlador PI projetado no domínio-*z*, com *Ts* = 1/20 s. Rode o experimento por 10 s. Como entrada, considere o degrau de amplitude 3V aplicado em 1 segundo.

d) Considere uma das formas incrementais previamente projetadas. Rode o experimento por 30 segundos. Como entrada, considere o degrau de amplitude 3,5V aplicado em 1 s. Em *t* = 5 s, abaixe o freio e mantenha abaixado até *t* = 10 s. Verifiquei efeito do windup. Aplique a técnica anti-windup, execute o experimento novamente e compare como o resultado sem anti-windup.

Simulação




```
function u  = fcn(e)
%#eml
global u_ant; global e_ant;
a = 0.8; K = 10;
u = u_ant + K*e - K*a*e_ant;
u_ant = u; e_ant = e;
```

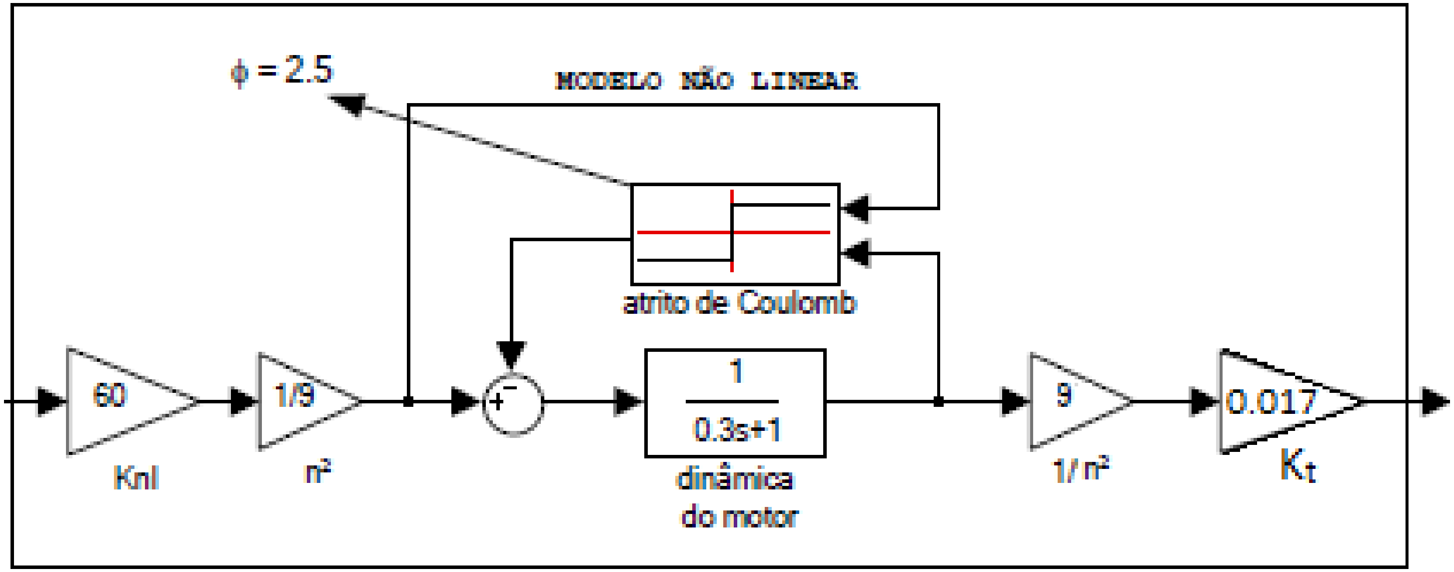
Exemplo:

$$u[n] = u[n-1] + Ke[n] - Kae[n-1]$$

$$K = 10; a = 0,8$$

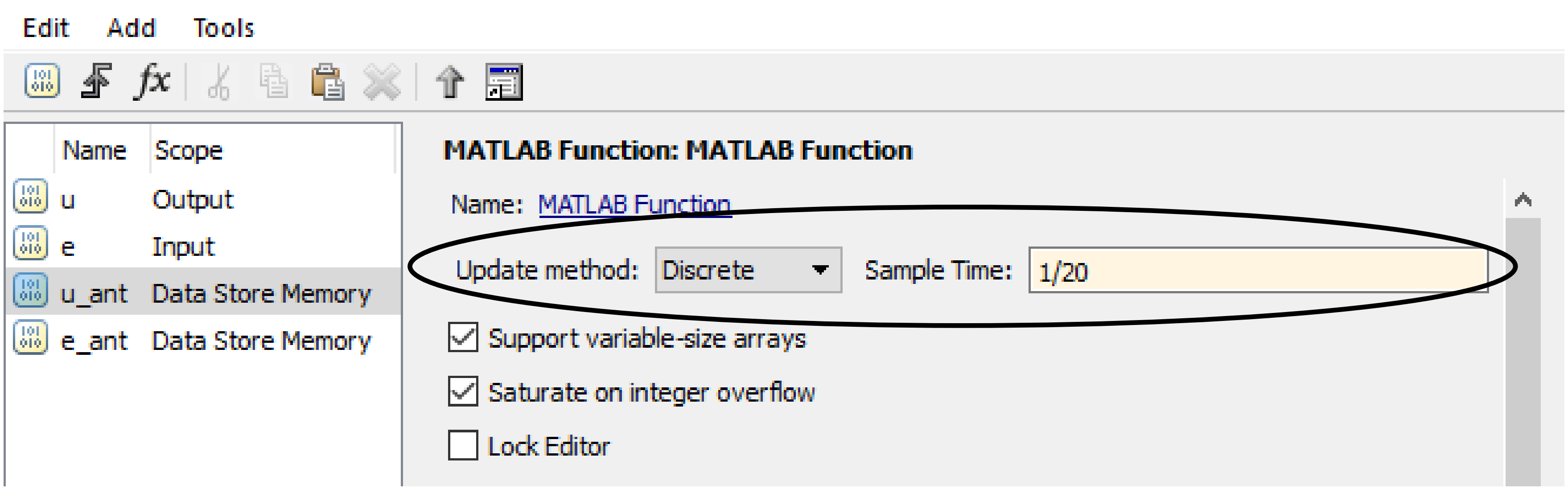
Simulação

Na simulação, trocar G(s) pelo modelo não linear



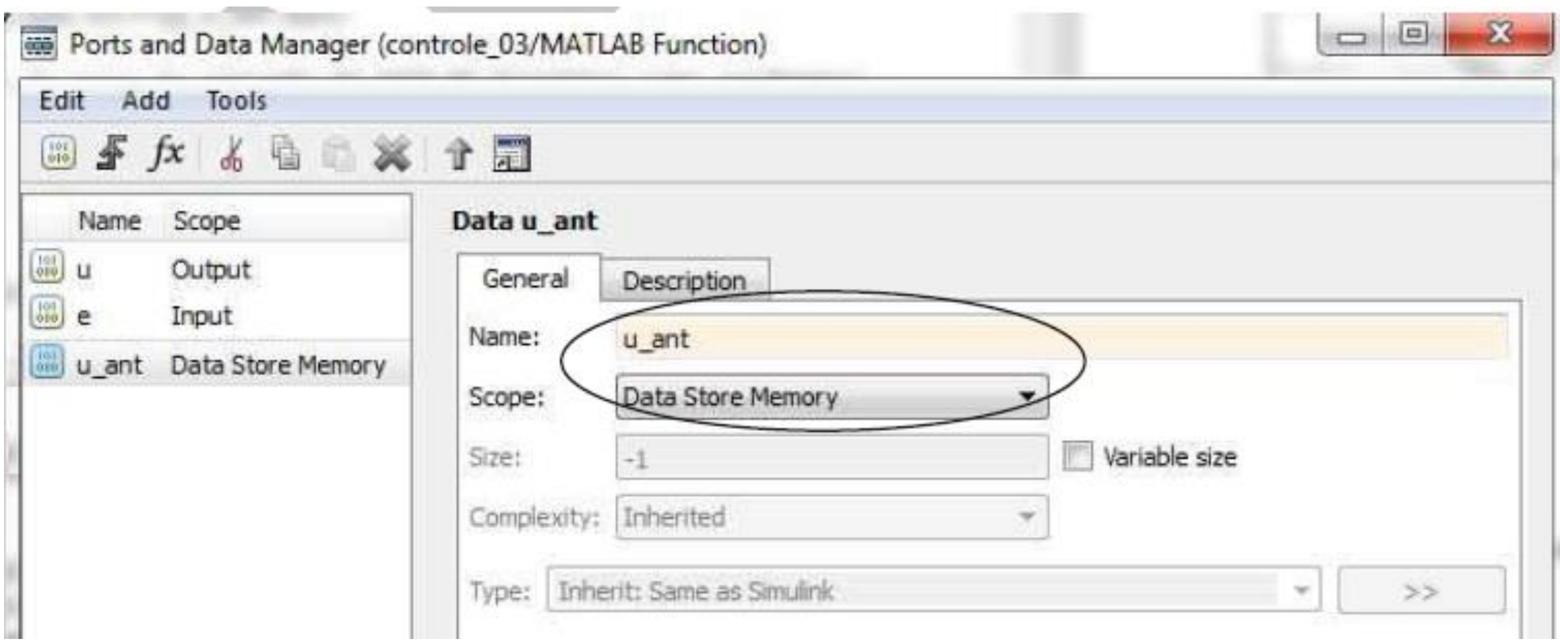
Simulação

Nas propriedades da *Matlab Function*, configurar o período de amostragem:



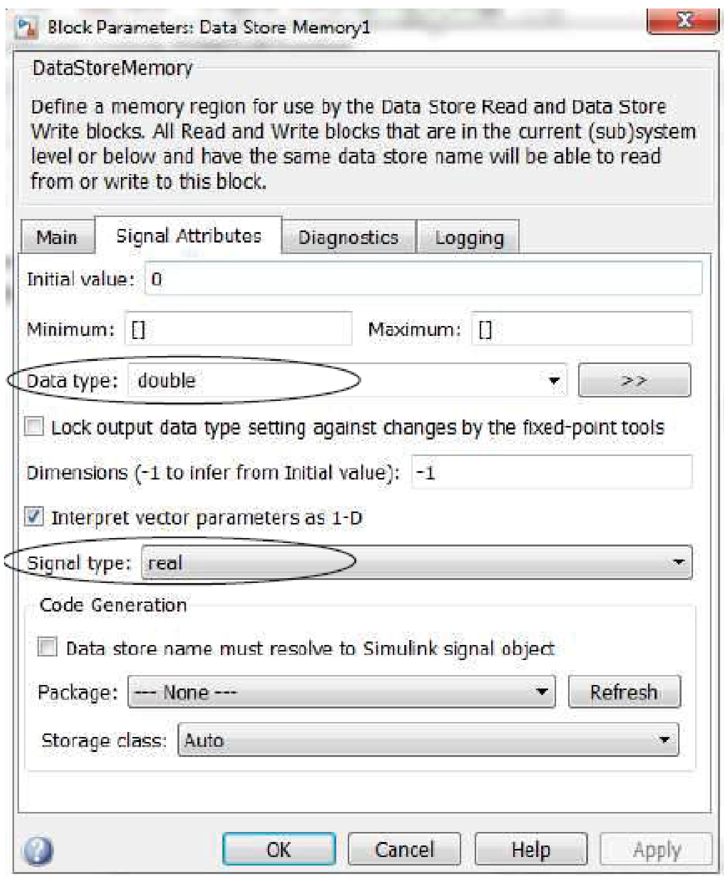
Simulação

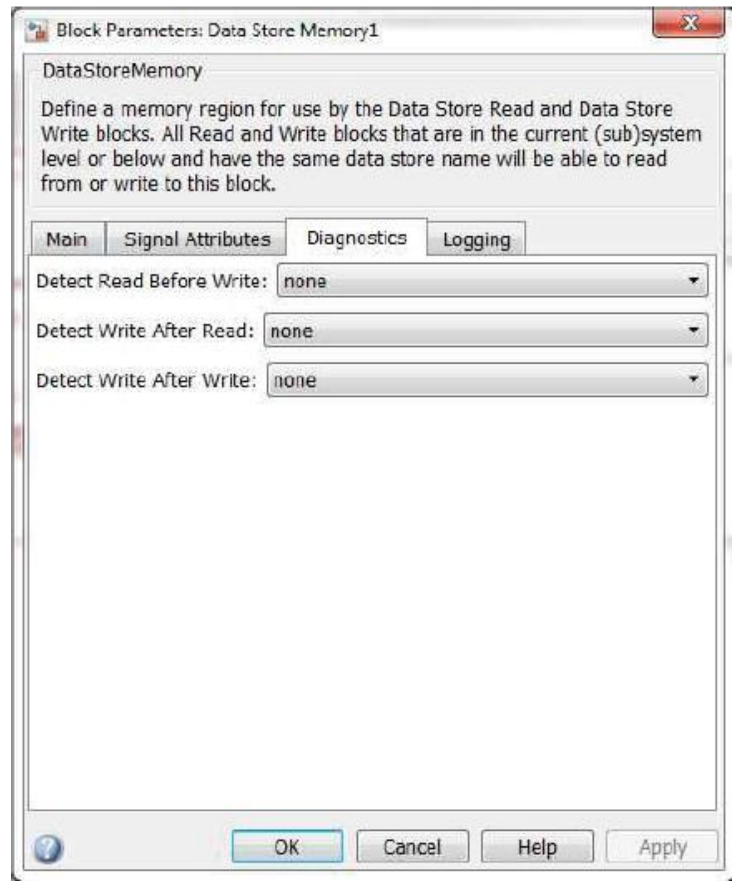
Nas propriedades da *Matlab Function*, inserir as variáveis que serão gravadas na memória da seguinte forma:



Simulação

A configuração dos blocos *Data Store Memory* é feita da seguinte forma:



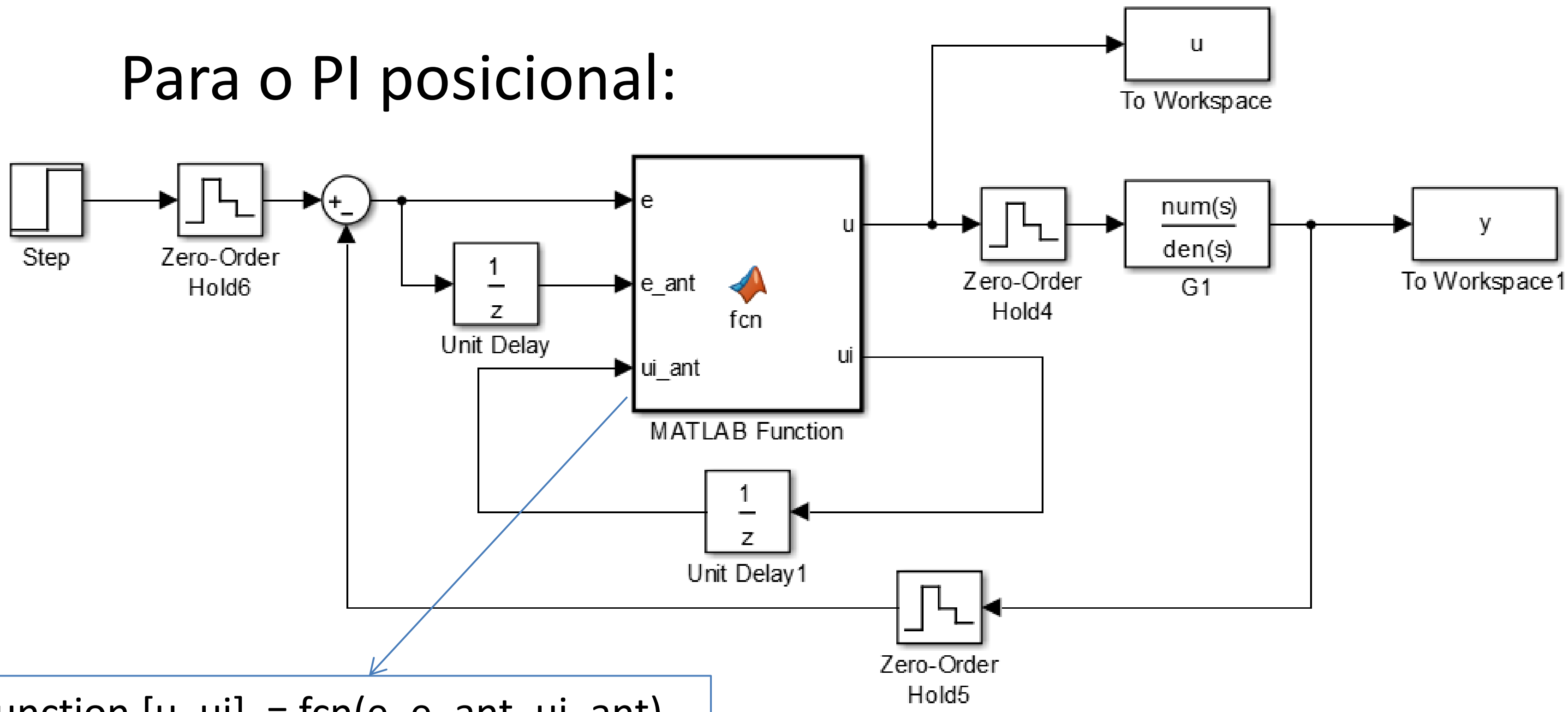


Simulação

Infelizmente, as versões 2008 e 2009 do Matlab não permitem criar variáveis globais na *Matlab Function*. Assim, as seguintes modificações precisam ser feitas:

Simulação

Para o PI posicional:

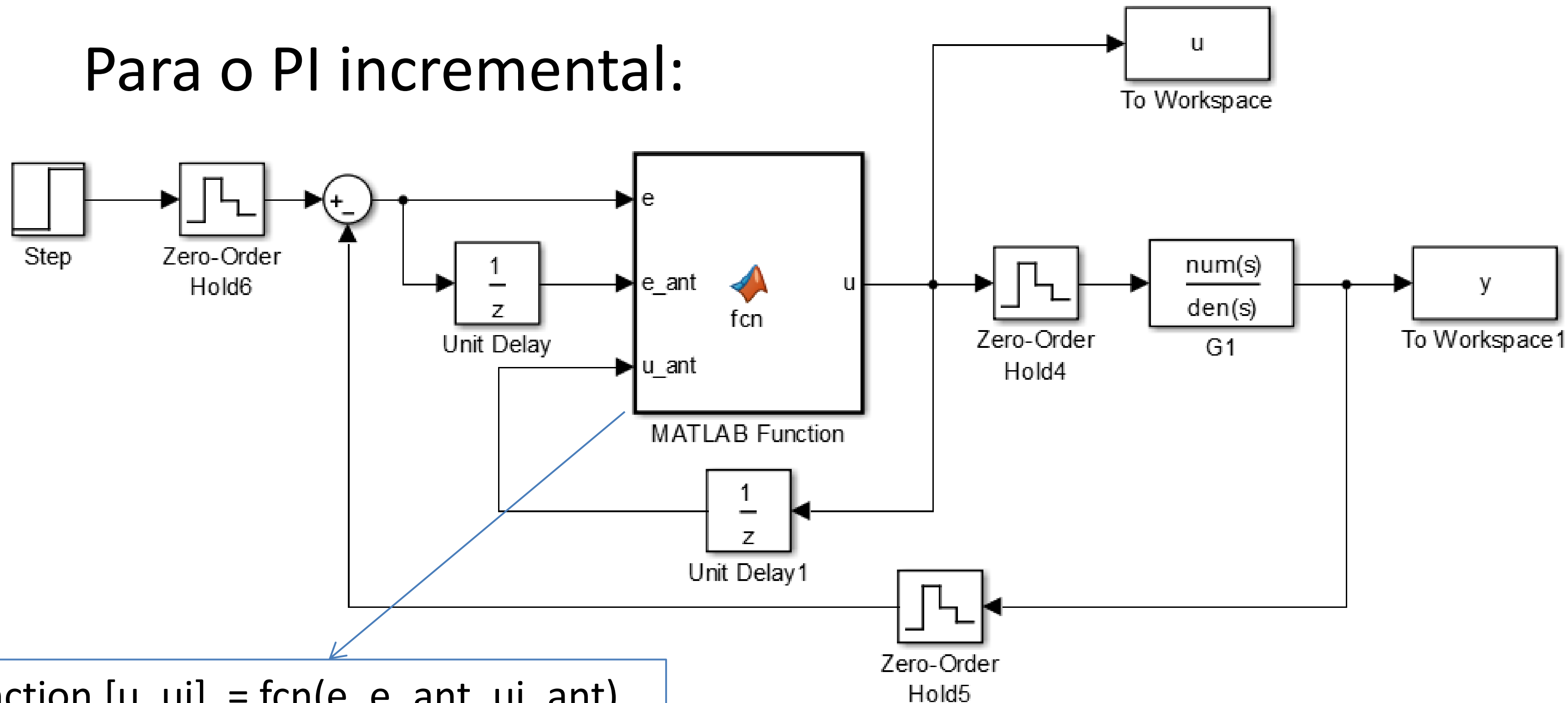



```
function [u, ui]  = fcn(e, e_ant, ui_ant)
%#eml
up = Kp*e;
ui = ui_ant + (Kp*Ts/(2*Ti))*(e+e_ant);
u = up+ui;
```

OBS: apagar as declarações das variáveis globais

Simulação

Para o PI incremental:

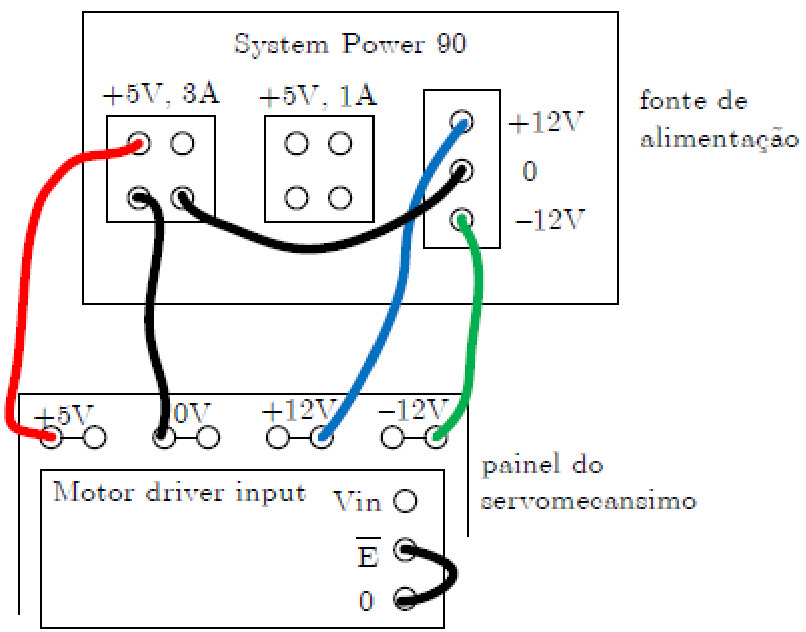



```
function [u, ui]  = fcn(e, e_ant, ui_ant)
%#eml
D_up = Kp*(e-e_ant);
D_ui = (Kp*Ts/(2*Ti))*(e+e_ant);
u = u_ant + D_up + D_ui;
```

OBS: apagar as declarações das variáveis globais

Atividades

- Prática:
  - Template: `prog4.m`
  - Ligações:



Atividades

- Prática:
  - Ligações: