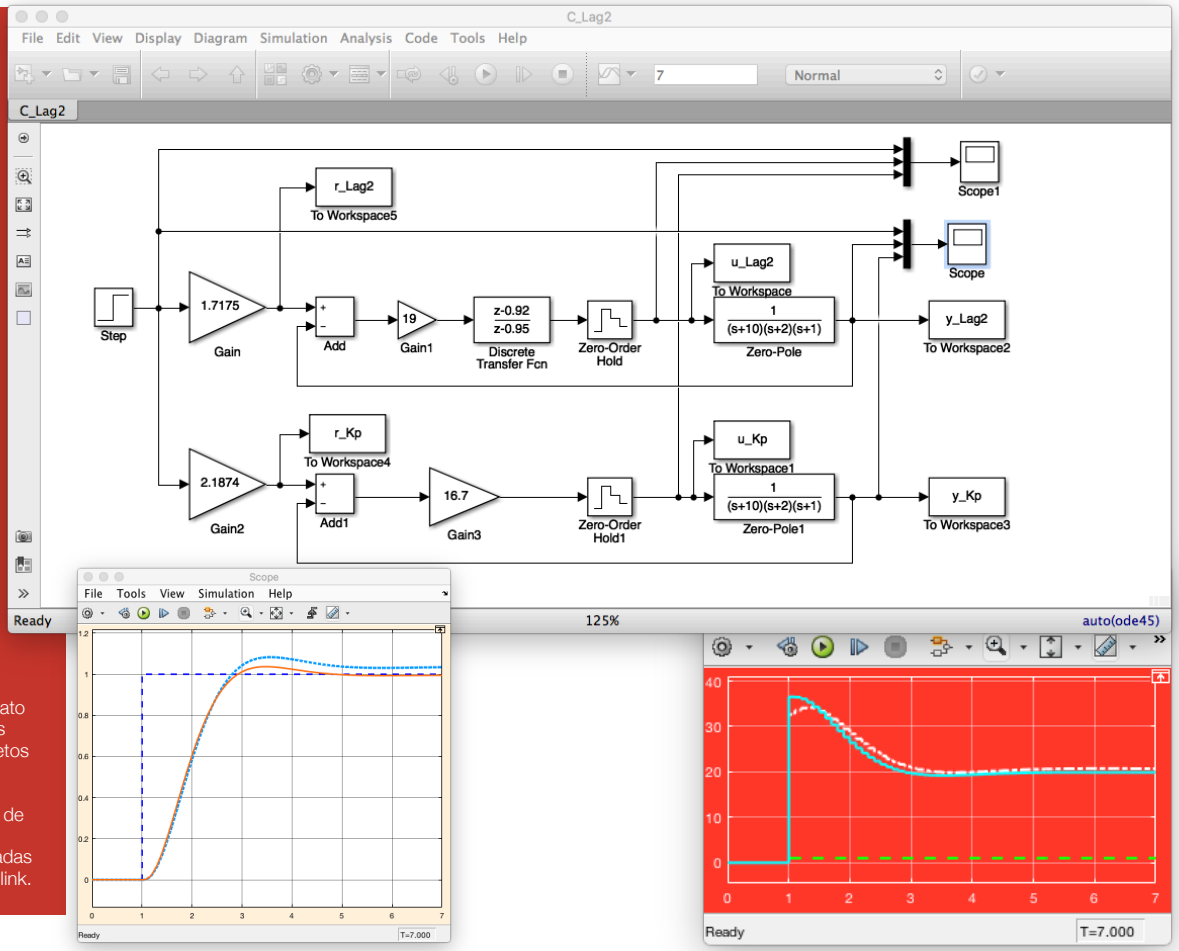
12 de abril de 2017

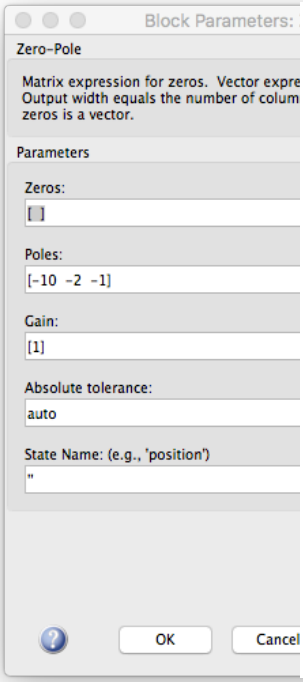


Observações qunato a simular sistemas contínuos e discretos e como recuper gráficos melhor acabados à partir de Scopes ou simulações realizadas no MATLAB/Simulink.

Simulações de Controladores Digitais

Controlador Proporcional x Lag_2

Prof. Fernando Passold



Primeiramente carregamos os dados de algum projeto já realizado anteriormente ou aproveitamos os dados que levantamos do projeto atual.

No caso: carregando dados de projeto já realizado em aula anterior (na janela de comandos do MATLAB).

Apesar de que o único dado que utilizaremos na exposição do modelo usado neste documento, seja o período de amostragem, T:

```
>> T=0.1;
```

Já podemos aproveitar e pedir para o MATLAB carregar o Simulink:

```
>> simulink
```

A planta do mundo contínuo usada para o estudo de caso é dada pela equação:

$$G(s) = \frac{1}{(s+1)(s+2)(s+10)}$$

Para entrar com esta informação no Simulink é mais fácil arrastar um bloco ‘ZeroPole’ da seção ‘Continous’ e informar na caixa de propriedades deste bloco:

Zeros:
[ ]
Poles:
[-10 -2 -1]

Gain:
[1]

(Deixar a caixa de texto referente à ‘Zeros’ como um vetor vazio ou nulo, ‘[ ]’ . Não esquecer de clicar em “Apply” ao final).

A idéia no caso deste documento é simular 2 controladores diferentes e observar as amplitudes de controle desenvolvidas.

Simulações de Controladores Digitais
Controlador Proporcional x Lag_2

Prof. Fernando Passold

Um dos controladores é do tipo Proporcional e o outro controlador é um de Atraso (C_Lag2), ambos desenvolvidos em aulas anteriores.

$$K_P = 16,7$$

$$C\_Lag_2(z) = 17,0769\frac{(z-0,92)}{(z-0,95)}$$

No caso do controlador por Atraso foi inserido o bloco '**Transfer Function**' da seção '**Discrete**' do Simulink, com os seguintes dados:

Numerator: [**1 -0.92**]
Denominator: [**1 -0.95**]
Sample time:
T

Notar que em todos os blocos relacionados com uma função discreta é imperativo informar o período de amostragem adotado sob pena de obter uma simulação incorreta. No caso desta aula, não esquecer de informar o valor **T** nos blocos: 'Discrete Transfer Fcn' e 'Zero-Order Hold'.

Notar que se preferiu não informar ganhos dos controladores já dentro de seus blocos, mas como um bloco externo de ganho, o que facilita alterações e testes com outros valores.

Para exportar num gráfico o próprio modelo criado no Simulink sugere-se o uso do comando `print(.)` na linha de comandos do MATLAB -- ver caixa de texto na próxima página. Detalhe: ao criar-se arquivos JPEG ou PNG é melhor aumentar sua resolução, caso contrário, os arquivos criados são de apenas 72 dpi (resolução adequada apenas para uso em tela mas notadamente baixa no caso de impressões). Para aumentar esta resolução sugere-se agregar mais uma opção ao comando print: '-r<resolução_em_dpi>'

Depois de inserir todos os dados, podemos iniciar as simulações deste sistema. No caso, o período de simulação adotado foi de 7 segundos.

Próxima etapa:
Como melhor exportar os gráficos das simulações obtidos das janelas Scopes

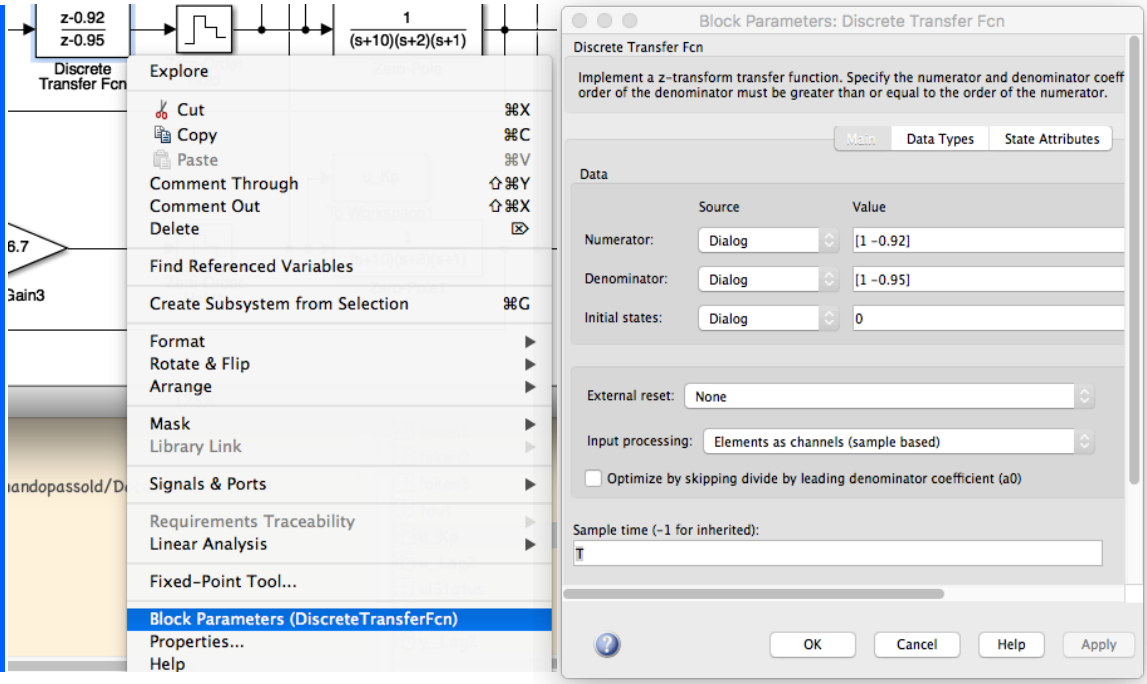
Melhor que tentar capturar telas "print screen" é capturar os dados e usar comandos 'plot' na janela de comandos do MATLAB. Para tanto, devemos acrescentar blocos do tipo 'To Workspace' à partir de pontos do bloco cujos dados nos interessam.

Notar que há 4 tipos de (captura de) dados disponíveis à partir do bloco 'To Workspace':

- Structure With Time
- Scructure
- Array
- Timeseries

O formato mais antigo (e mais fácil de manipular é o 'Array' que cria um simples vetor de dados; mas o formato adotado nas últimas versões do MATLAB é o 'Timeseries' (na realidade um

Detalhes
SEguem detalhes de alguns blocos usados no Simulink.



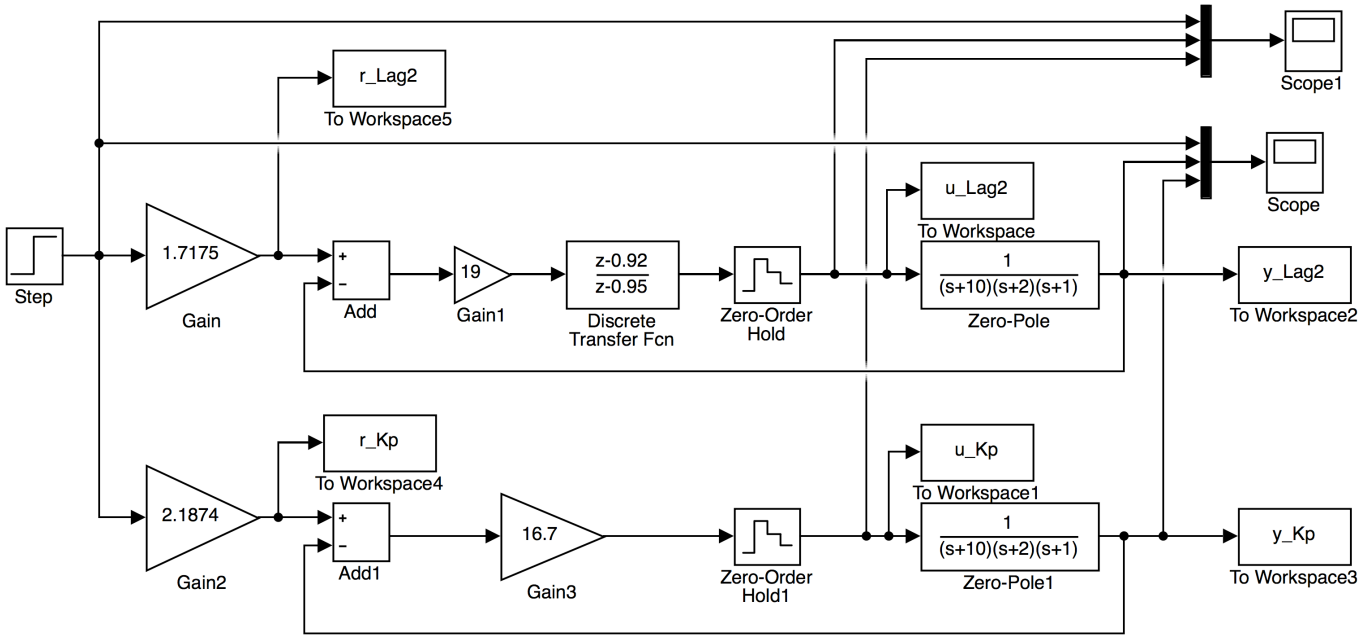
Comandos usados no MATLAB para exportar a figura associado com o diagrama em blocos criado anteriormente:

```
>> print -sC_Lag2 -dpdf 'C_Lag2.pdf'
>> print -sC_Lag2 -dpng -r300 'C_Lag2.png'
>> print -sC_Lag2 -djpeg -r300 'C_Lag2.jpg'
```

A primeira linha acima cria um arquivo PDF (com margens). A segunda linha um arquivo PNG (C_Lag2.png, 201 KB) com 300 dpi de resolução. E a terceira linha cria um arquivo JPG (C_Lag2.jpg, 277 KB).

Comando 'print':

```
>> help print
  print Print or save a figure or model.
    A subset of the available options is presented below. For more details
    see the documentation.

    print, by itself, prints the current figure to your default printer.
    Use the -s option to print the current model instead of the current figure.
      print      % print the current figure to the default printer
      print -s   % print the current model to the default printer

    print(filename, formattype) saves the current figure to a file in the
    specified format. Vector graphics, such as PDF ('-dpdf'), and encapsulated
    PostScript ('-depsc'), as well as images such as JPEG ('-djpeg') and PNG ('-dpng')
    can be created. Use '-d' to specify the formattype option
      print(fig, '-dPDF', 'myfigure.pdf'); % save to the 'myfigure.pdf' file
>>
```



objeto), algo mais complexo de ser manipulado. Cada tipo de formato traz suas vantagens e desvantagens. O formato Array não agrega a variável tempo associado com cada instante amostrado presente no vetor de dados, o que significa que é o usuário que deve criar este vetor.

Se for usado "Array", e for adotado um simples comando do tipo:

```
>> plot (u_Lag2)
```

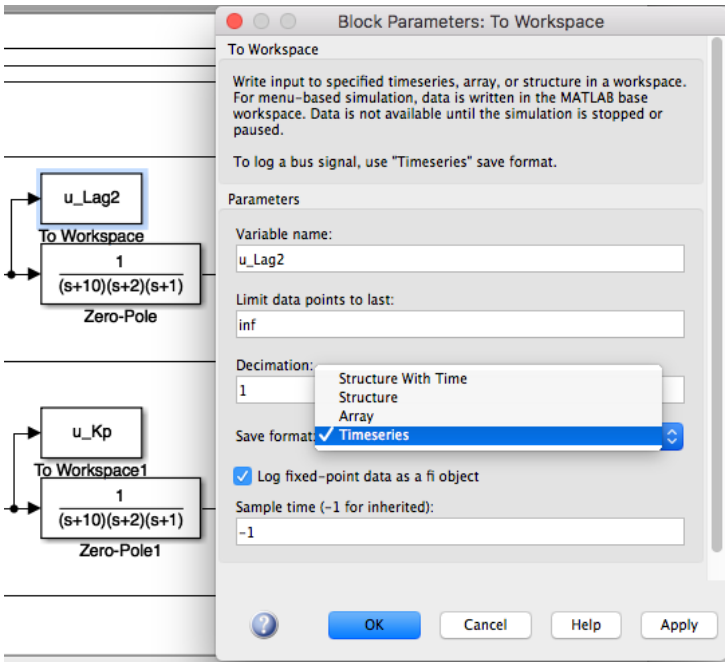
será mostrando um gráfico cujo eixo de abcissas corresponde ao instante da amostragem mas não o instante de tempo (em segundos) da amostra.

Se foi selecionado "Timeseries" o mesmo comando adotado anteriormente gera apropriadamente o eixo de abcissas na escala de tempo em segundos (7 segundos).

Mas a idéia é num mesmo gráfico juntar as respostas de diferentes controladores.

Opção 1) Usando "Timeseries":

Quando a simulação é realizando usando-se "Timeseries" percebe-se que cada cojunto de dados (classe 'timeseries') traz consigo valores (propriedade 'data'), tempo (propriedade



Opções de tipos de dados.

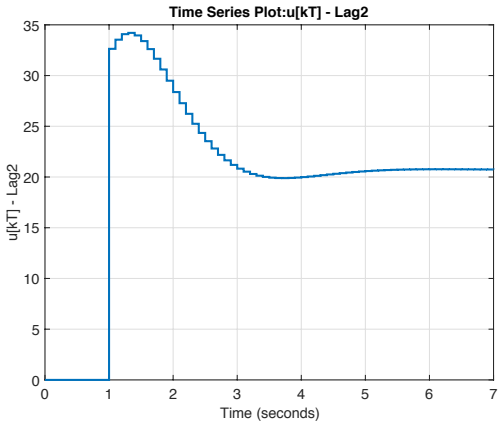
Simulações de Controladores Digitais
Controlador Proporcional x Lag_2

Prof. Fernando Passold

'time') unidade de tempo (propriedade 'TimeInfo'), nome para o conjunto de dados (propriedade 'Name') e que é mais importante, a quantidade varia mesmo no caso de uma simulação como esta:

```
>> whos
  Name          Size              Bytes  Class       Attributes
  T             1x1                   8  double
  ans           1x60                120  char
  r_Kp          1x1                 1568  timeseries
  r_Lag2        1x1                 1568  timeseries
  tout         77x1                  616  double
  u_Kp          1x1                  920  timeseries
  u_Lag2        1x1                  920  timeseries
  y_Kp          1x1                 1580  timeseries
  y_Lag2        1x1                 1580  timeseries

>> size(u_Lag2.data) % quantos dados dentro de u_Lag2?
ans =
    71     1
>> % Monstrando dados entre 10 < k < 15
>> [u_Lag2.time(10:15) u_Lag2.data(10:15)]
ans =
    0.9000         0
    1.0000   32.6325
    1.1000   33.5357
    1.2000   34.0802
    1.3000   34.2058
    1.4000   33.9542
>> u_Lag2.TimeInfo % unidade empregada para 'time'
  tsdata.timemetadata
  Package: tsdata

  Uniform Time:
    Length       71
    Increment    1.000000e-01 seconds

  Time Range:
    Start        0 seconds
    End          7 seconds

  Common Properties:
          Units: 'seconds'
         Format: ''
      StartDate: ''

  More properties, Methods
>> % Associando um nome para 'u_Lag2':
>> set(u_Lag2, 'Name', 'u[kT] - Lag2')
>> u_Lag2.Name

ans =

u[kT] - Lag2

>> plot(u_Lag2)
```

Figura gerada para 'u_Lag2' (formato 'Timeseries").

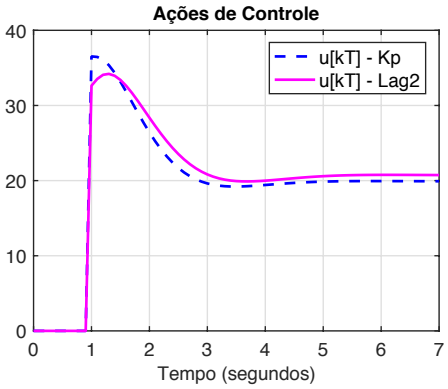


O último comando gera o gráfico como o mostrado na figura acima.

Então para mesclar num mesmo gráfico os dados de 2 objetos do tipo "timeseries" se faz necessário usar o comando plot de forma mais explícita:

```
>> set(u_Kp, 'Name', 'u[kT] - Kp')
>> plot(u_Kp.time, u_Kp.data, 'b--', u_Lag2.time,
u_Lag2.data, 'm-')
>> legend(u_Kp.Name, u_Lag2.Name)
>> xlabel('Tempo (segundos)');
>> title('Ações de Controle');
```

O que faz surgir a figura:



Opção 2) Usando "To Worskpace" no formato "Structure With Time"

Modificando apenas '**u_Lag2**' e '**u_Kp**' verificamos que:

```
>> whos
  Name          Size              Bytes  Class       Attributes
  T             1x1                   8  double
  ans           1x87                174  char
  r_Kp          1x1                 1568  timeseries
  r_Lag2        1x1                 1568  timeseries
  tout         77x1                  616  double
  u_Kp          1x1                 2240  struct
  u_Lag2        1x1                 2238  struct
  y_Kp          1x1                 1580  timeseries
  y_Lag2        1x1                 1580  timeseries
```

Simulações de Controladores Digitais
Controlador Proporcional x Lag_2

Prof. Fernando Passold

```
>>
```

Notar que este tipo de dado é algo mais simples:

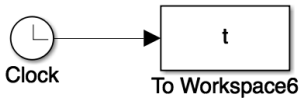
```
>> u_Kp
u_Kp =
  struct with fields:
         time: [71×1 double]
      signals: [1×1 struct]
    blockName: 'C_Lag2/To Workspace1'

>> u_Kp.signals
ans =
  struct with fields:
       values: [71×1 double]
   dimensions: 1
        label: ''

>> [u_Kp.time(10:15) u_Kp.signals.values(10:15)]
ans =
    0.9000         0
    1.0000   36.5296
    1.1000   36.4550
    1.2000   36.0802
    1.3000   35.3610
    1.4000   34.3556
>>
```

Então para criar o gráfico com os 2 sinais superpostos:

```
>> plot(u_Kp.time, u_Kp.signals.values, 'b--',
u_Lag2.time, u_Lag2.signals.values, 'm-')
>> legend('u[kT] - Kp', 'u[kT] - Lag2')
```

Opção 3) Usando 'Array'

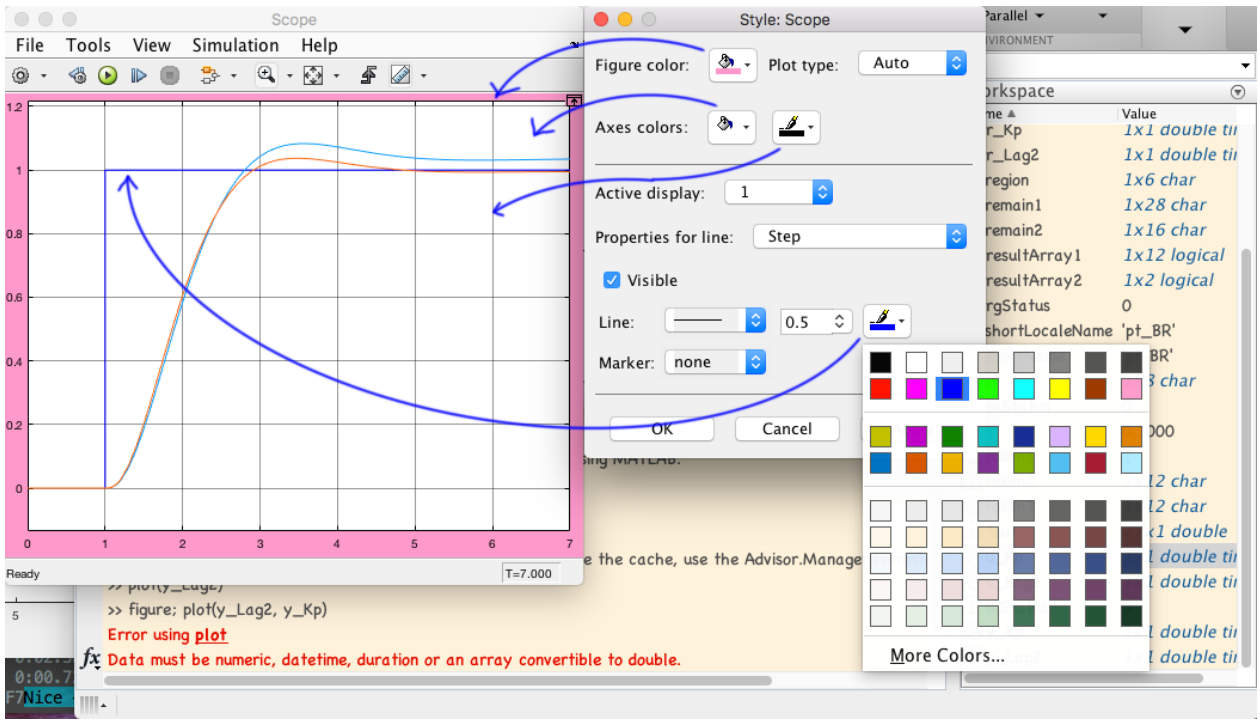
Esta é a opção relacionada com as primeiras versões do MATLAB, porém têm a tendência à ser descontinuada. Note que este tipo de dado cria simplesmente um vetor sem associação com o instante de tempo em que cada dado foi criado (amostrado). Para isto é necessário acrescentar um bloco extra, chamado "Clock" (presente na secão "Sources") ao diagrama de blocos à fim de capturar esta informação extra (o instante de tempo de cada amostra de dado):



E claro, ainda alterar o tipo de dados de todos os blocos "To Workspace" para array. Mas após uma nova simulação percebe-se que são gerados vetores de diferentes tamanhos.

```
>> clear all % recomeçar tudo de novo
>> T=0.1;
>> whos
  Name          Size              Bytes  Class
Attributes

  T             1x1                 8  double
  r_Kp         77x1               616  double
  r_Lag2       77x1               616  double
  t            77x1               616  double
  tout         77x1               616  double
  u_Kp         71x1               568  double
  u_Lag2       71x1               568  double
```

Style dentro de Scopes Modificando as cores adotadas na janelas Scope.



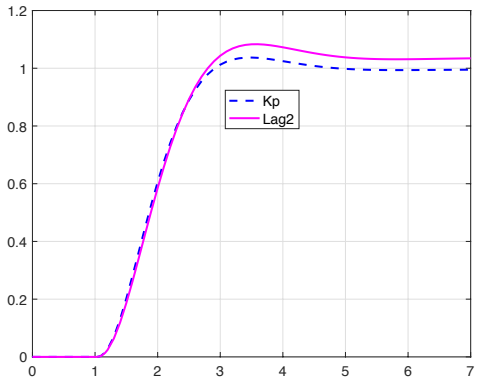
Simulações de Controladores Digitais
Controlador Proporcional x Lag_2

Prof. Fernando Passold

```
y_Kp        77x1          616  double
y_Lag2      77x1          616  double

>>
```

Notar que há vetores com 77 dados e outros com 71 dados (os relacionados com o sinal puramente digital); isto limita os gráficos que podem ser obtidos, já que dependemos da variável '**t**' que no caso resultou num vetor de 77 x 1. Então é possível se comparar o desempenho de saída dos 2 controladores:

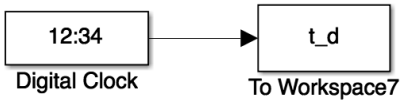
```
>> figure; plot(t, y_Kp, 'b--', t,y_Lag2, 'm-')
```



A falta de observação quanto ao tamanho dos diferentes vetores pode levar à erros.

```
>> plot(t, u_Kp,'b--',  t, u_Lag2,'m-')
Error using plot
Vectors must be the same length.

>>
```

Mas existe uma solução para este caso, a introdução do bloco "Digital Clock":



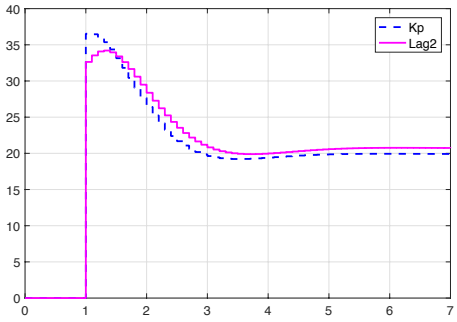
Porém note que nas propriedades do bloco "Digital Clock" deve ser informado o período de amostragem adotado: Sampletime: **T** (no nosso caso)! E então temos vetores com tamanhos compatíveis entre si:

```
>> whos
  Name          Size              Bytes  Class
Attributes

T           1x1                 8  double
r_Kp        77x1              616  double
r_Lag2      77x1              616  double
t           77x1              616  double
t_d         71x1              568  double
tout        77x1              616  double
u_Kp        71x1              568  double
u_Lag2      71x1              568  double
y_Kp        77x1              616  double
y_Lag2      77x1              616  double

>>
```

E podemos então comparar as diferentes amplitudes de controle desenvolvidas por estes 2 diferentes controladores:

```
>> [x1,y1]=stairs(t_d, u_Kp);
>> [x2,y2]=stairs(t_d, u_Lag2);
>> figure; plot(x1,y1,'b--', x2,y2,'m-');
>> legend('Kp', 'Lag2')
>> grid
```

O que gera a figura:



Fim.

stairs(.):
>> **help stairs**

stairs Stairstep plot.

stairs(Y) draws a stairstep graph of the elements of vector Y.

stairs(X,Y) draws a stairstep graph of the elements in vector Y at the locations specified in X.

[XX,YY] = stairs(X,Y) does not draw a graph, but returns vectors X and Y such that PLOT(XX,YY) is the stairstep graph.

Stairstep plots are useful for drawing time history plots of zero-order-hold digital sampled-data systems.