Introduction to Simulink

Simulation of nonlinear and hybrid-time systems

Riccardo Antonello
(riccardo.antonello@unipd.it)

Giulia Michieletto
(giulia.michieletto@unipd.it)

Dipartimento di Tecnica e Gestione dei Sistemi Industriali

Università degli Studi di Padova

4 Marzo 2024

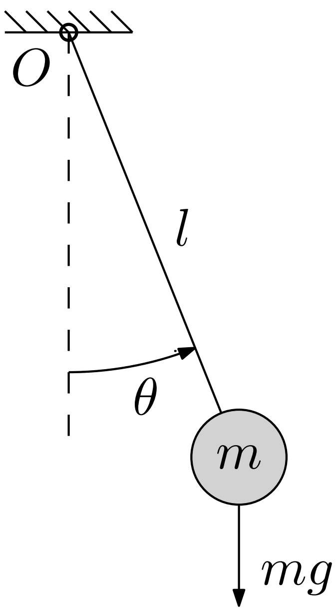
Outline

- Simulation of a *nonlinear system*:
  - Modelling of the nonlinear pendulum.
  - Simulation using Lord Kelvin's scheme.
  - Period of the undamped nonlinear pendulum.
  - Comparison with the linearized model.

- Simulation of a *hybrid-time system*:
  - Continuous-time speed PI control of a DC motor.
  - Discrete-time speed PI control loop of a DC motor.

Simulation of nonlinear systems

Example: simulate the natural response of a simple pendulum, starting from given ICs.



Newton-Euler 2nd law

$$J\,\ddot{\theta} = \sum_n \tau_n$$

Moment of Inertia

$$J = m\,l^2$$

Torque due to weight

$$\tau_1 = -mgl\sin\theta$$

Friction torque

$$\tau_2 = -b\,\dot{\theta}$$

Nonlinear ODE

$$ml^2\,\ddot{\theta} + b\,\dot{\theta} + mgl\,\sin\theta = 0$$

Simulation of nonlinear systems

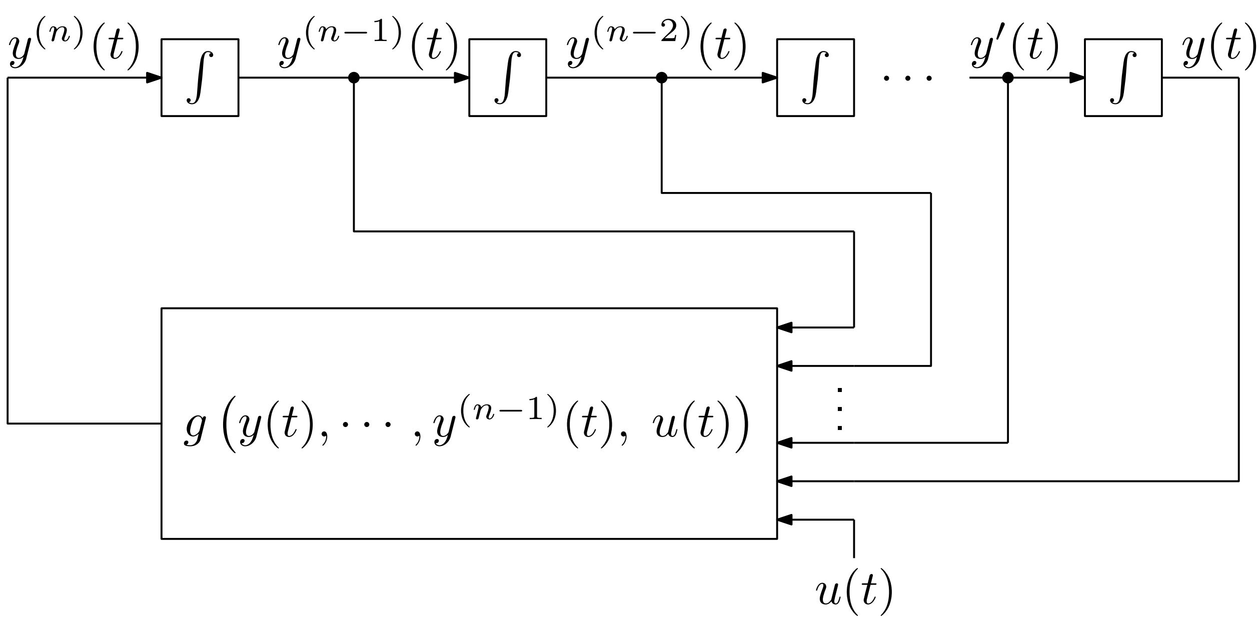
Nonlinear ODE (implicit form)

$$f\left(y(t), \cdots, y^{(n-1)}(t), y^{(n)}(t),\ u(t)\right) = 0$$

Nonlinear ODE (explicit form)

$$y^{(n)}(t) = g\left(y(t), \cdots, y^{(n-1)}(t),\ u(t)\right)$$

Lord Kelvin's scheme (*chain of integrators*)



Simulation of nonlinear systems

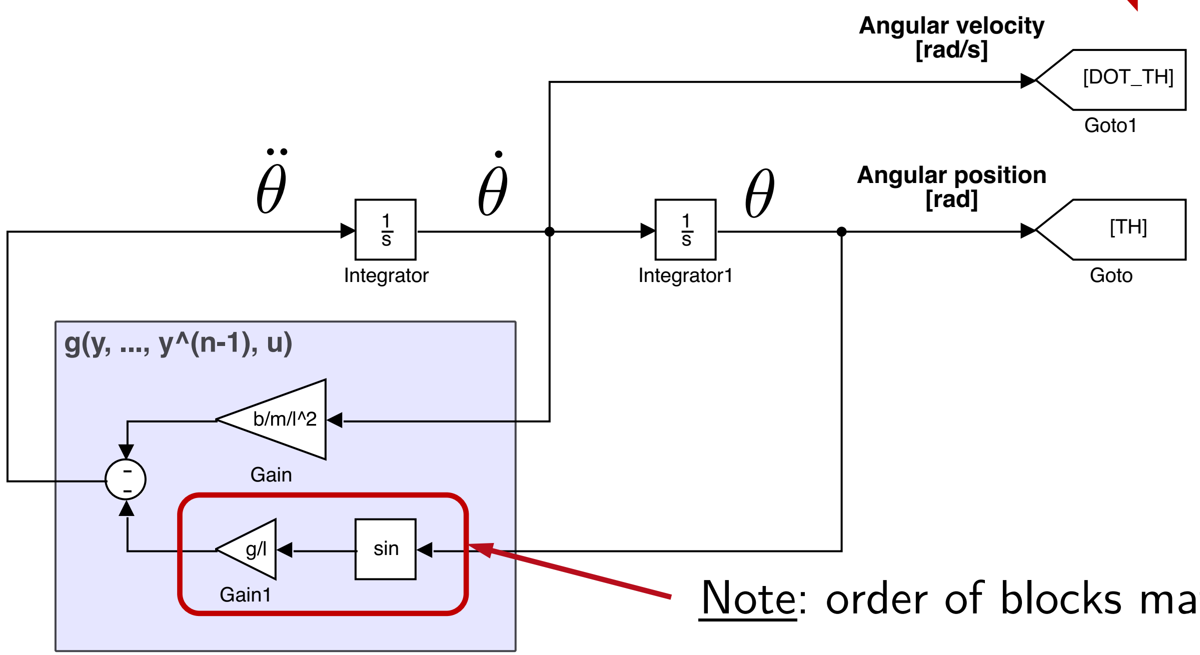
Nonlinear ODE (implicit form)

$$ml^2\,\ddot{\theta} + b\,\dot{\theta} + mgl\,\sin\theta = 0$$

Nonlinear ODE (explicit form)

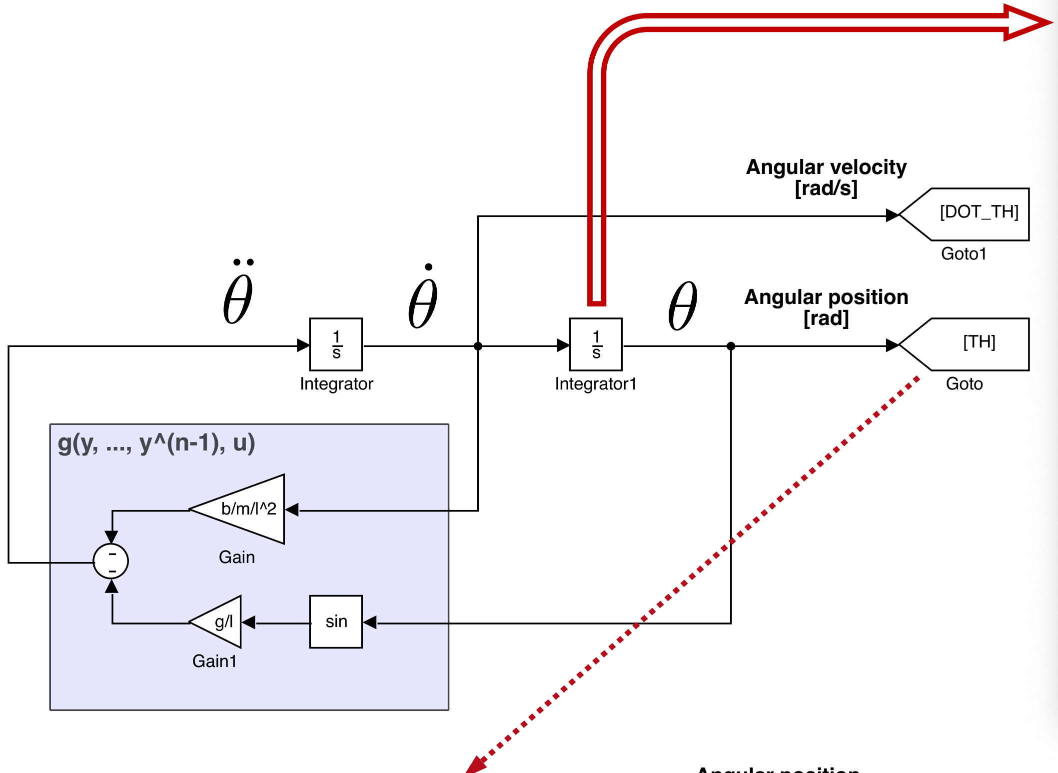
$$\ddot{\theta} = -\frac{b}{ml^2}\dot{\theta} - \frac{g}{l}\sin\theta$$

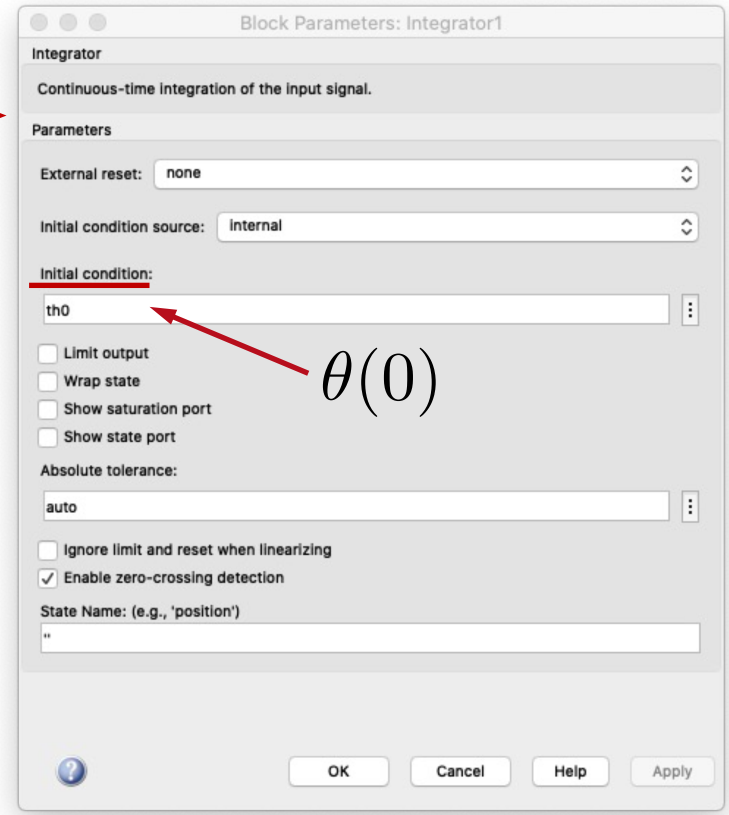
Lord Kelvin's scheme (*chain of integrators*)



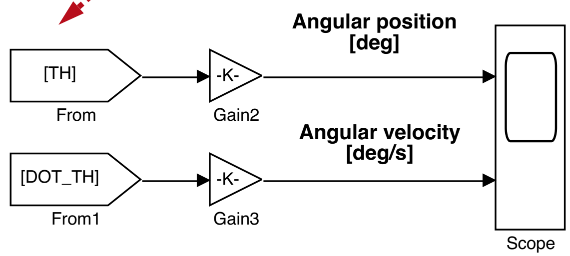
Note: order of blocks matters !

Simulation of nonlinear systems





A **Goto** block propagates a signal of the **From** block with same label.

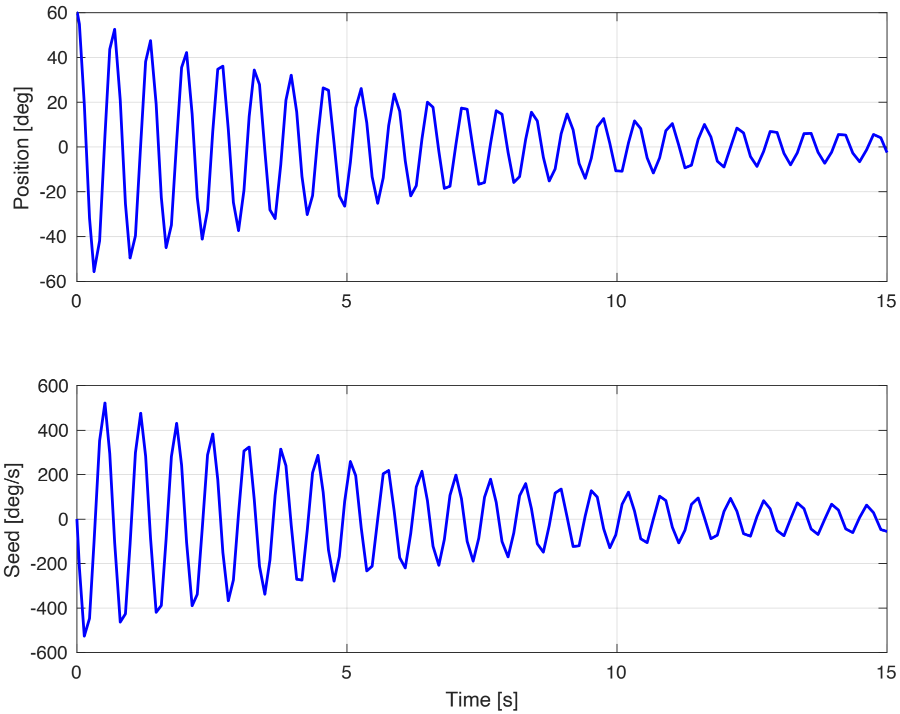


Initial state is set in the initial conditions (ICs) of the integrators.

Simulation of nonlinear systems

Natural response with $\theta(0) = 60°$, $\dot{\theta}(0) = 0$.

```
%  set simulation parameters
set_param('nonlin_pend', ...
    'SolverType', 'Variable-step', ...
    'Solver', 'ode45', ...
    'MaxStep', 'auto', ...
    'MinStep', 'auto', ...
    'AbsTol', 'auto', ...
    'RelTol', '1e-3', ...
    'StopTime', '15');

%  initial conditions
th0 = 60 * deg2rad;
dot_th0 = 0;

%  run simulation
sim('nonlin_pend');
```

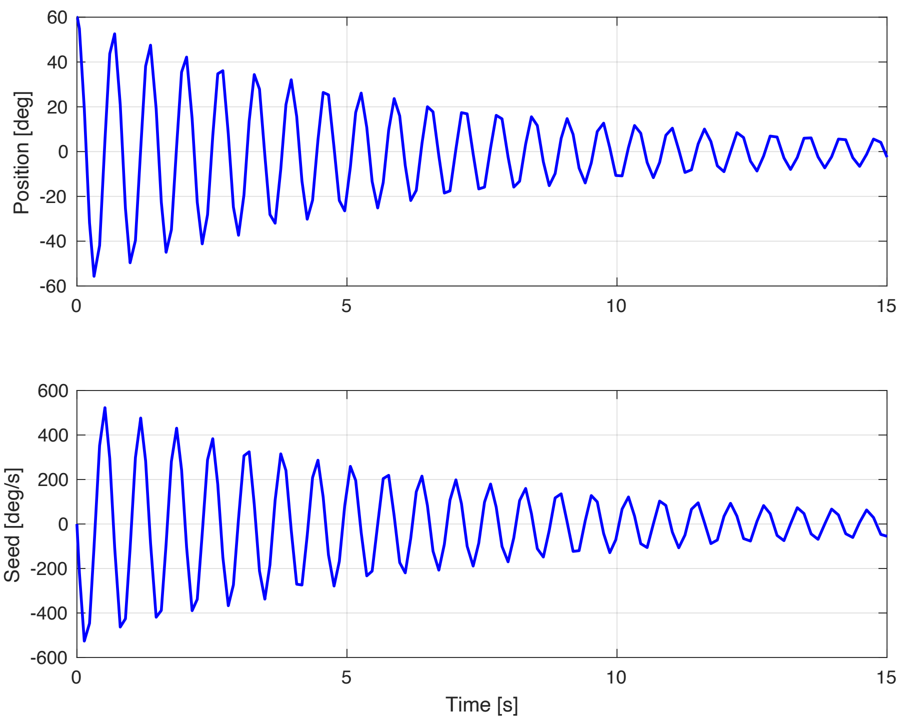


Max step too large ⇒ reduce max step size ...

Simulation of nonlinear systems

Simulation with reduced max step size:

```
%  set simulation parameters
set_param('nonlin_pend', ...
    'SolverType', 'Variable-step', ...
    'Solver', 'ode45', ...
    'MaxStep', 'auto', ...
    'MinStep', 'auto', ...
    'AbsTol', 'auto', ...
    'RelTol', '1e-3', ...
    'StopTime', '15');

%  initial conditions
th0 = 60 * deg2rad;
dot_th0 = 0;

%  run simulation
sim('nonlin_pend');
```

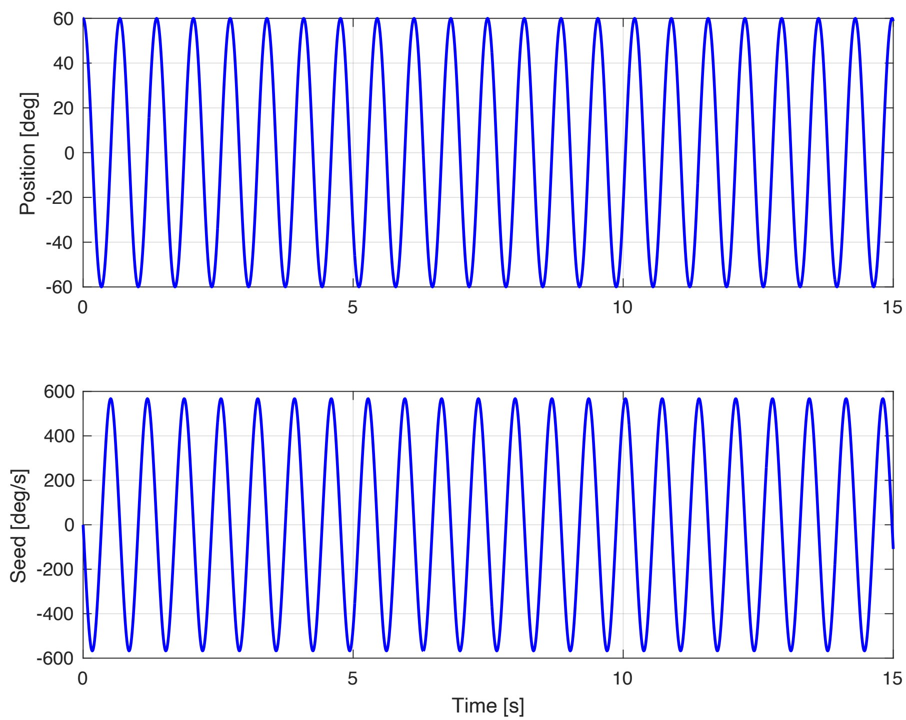


⇒ By increasing the number of integration instants, the response is smoother.

Simulation of nonlinear systems

Natural response with $b = 0$ (*undamped* sys):

```
%  set damping to zero
b = 0;

%  run simulation
sim('nonlin_pend');
```

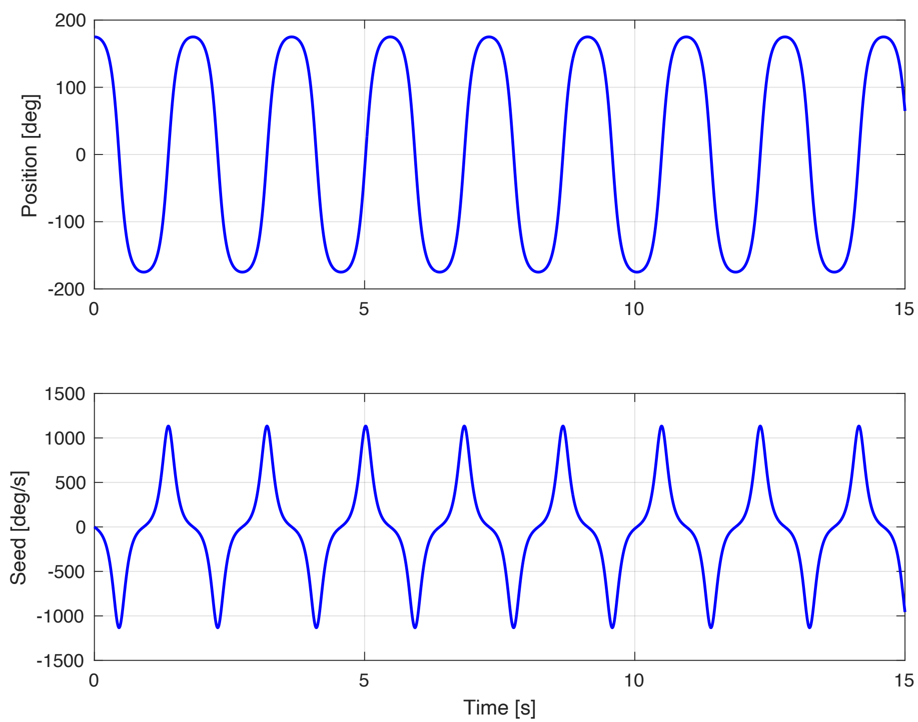


⇒ With no damping (no energy dissipation), the response is periodic.

Simulation of nonlinear systems

The oscillation is not *harmonic* (i.e. not a pure sinusoidal motion). This is more noticeable for large swings (e.g. $\theta(0) = 175°$, $\dot{\theta}(0) = 0$).

```
%  initial conditions
th0 = 175 * deg2rad;
dot_th0 = 0;

%  run simulation
sim('nonlin_pend');
```



Simulation of nonlinear systems

For an *undamped* nonlinear pendulum with $\theta(0) = \theta_0$ and $\dot{\theta}(0) = 0$, the period of the oscillation is equal to [1,2]:

$$T = 4\sqrt{\frac{l}{g}}\,K\left(\sin\frac{\theta_0}{2}\right)$$

where:

$$K(m) = \int_0^{\pi/2} \frac{d\theta}{\sqrt{1 - m^2 \sin^2\theta}}$$

Elliptical integral of the 1st kind

(1) https://www.math24.net/nonlinear-pendulum/
(2) http://www.sciencedirect.com/science/article/pii/S089812211200017X

Simulation of nonlinear systems

```
%  period of the nonlinear pendulum
k = sin(th0/2);
K = ellipke(k^2);
T_nlin = 4*sqrt(l/g)*K
```

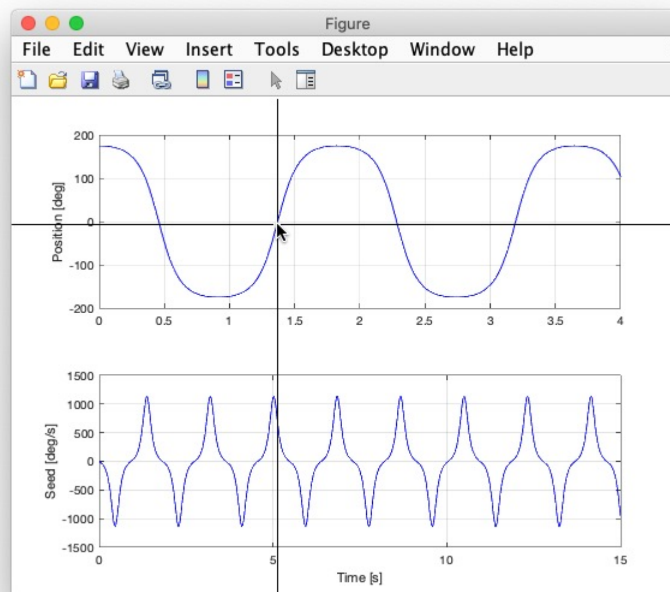
```
T_nlin = 1.8255
```

<u>Note</u>: the **`ellipke`** routine evaluates the elliptical integral of the 1st kind in the form:

$$K(m') = \int_0^{\pi/2} \frac{d\theta}{\sqrt{1 - m' \sin^2 \theta}}$$

Check the result by measuring the period on the plot with the **`ginput`** command ...

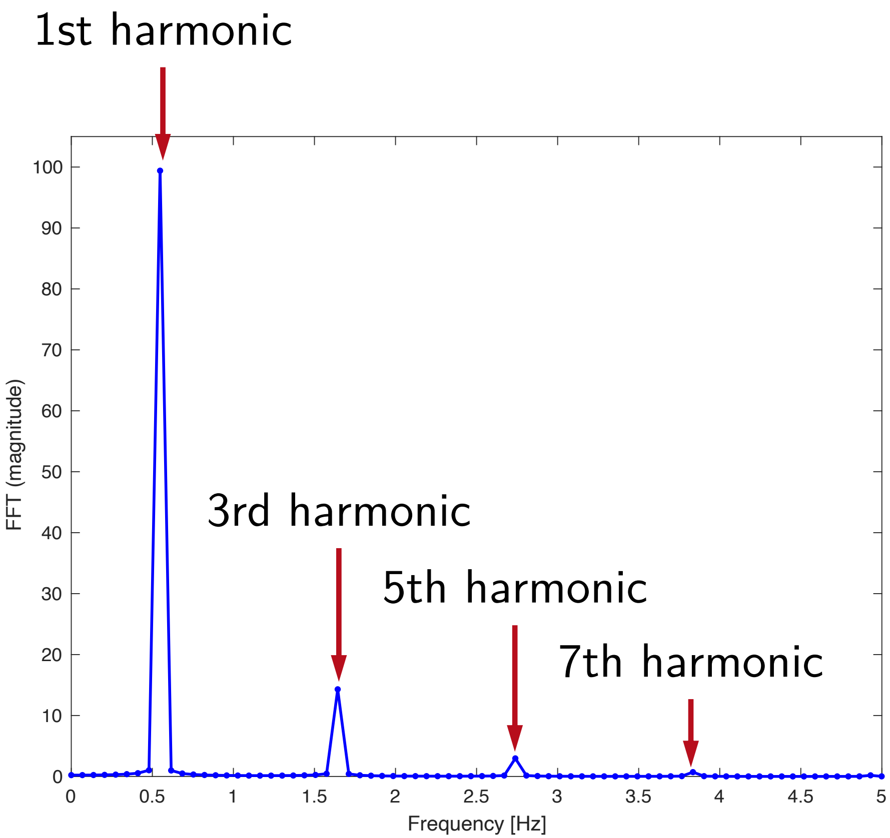
```
Command Window
New to MATLAB? See resources for Getting Start
>> ginput

ans =

    1.3779   -4.1667
    3.1935   -4.1667

>> diff(ans(:,1))

ans =

    1.8157

fx >>
```



Simulation of nonlinear systems

Spectrum (FFT) of the nonlinear oscillation:

```
%  select final time to consider
%  an integer number of periods
t1f = T_nlin * floor(t(end)/T_nlin);

%  resample at fixed rate
Ts = T_nlin/100;
t1 = 0:Ts:t1f;
th1 = interp1(t, th, t1);

%  compute fft
Fs = 1/Ts;
N1 = length(t1);
f1 = linspace(0, Fs, N1);

th1_fft = (1/N1)*fft(th1);
```



Simulation of nonlinear systems

For small oscillations, the dynamics can be *linearized* with the approximation $\sin\theta \approx \theta$:

Nonlinear ODE

$$ml^2\,\ddot{\theta} + b\,\dot{\theta} + mgl\,\sin\theta = 0$$

$$\sin\theta \approx \theta$$

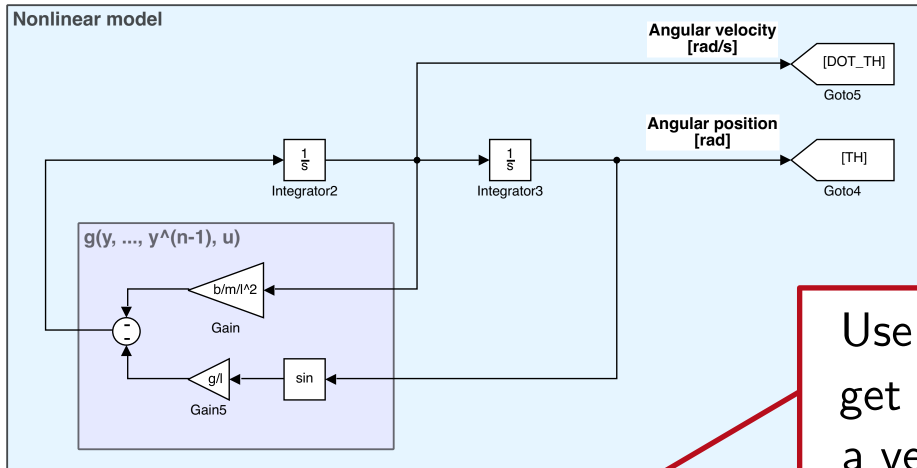
Linearized ODE

$$ml^2\,\ddot{\theta} + b\,\dot{\theta} + mgl\,\theta = 0$$

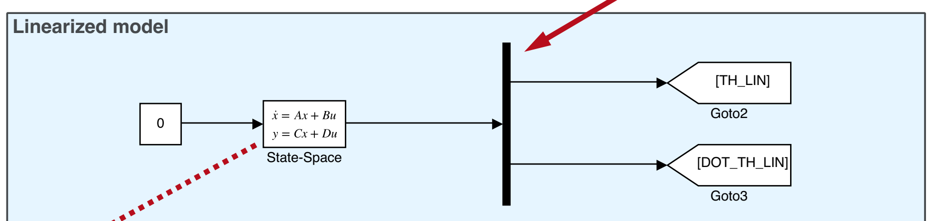
Linear state-space model

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\dfrac{g}{l} & -\dfrac{b}{ml^2} \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}$$

Simulation of nonlinear systems

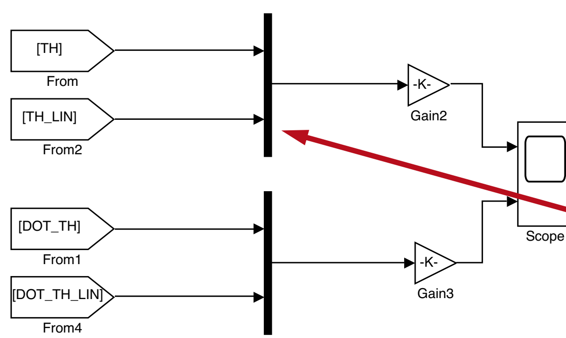


Use a **Demux** block to get the components of a vector-valued signal.



State-Space block parameters

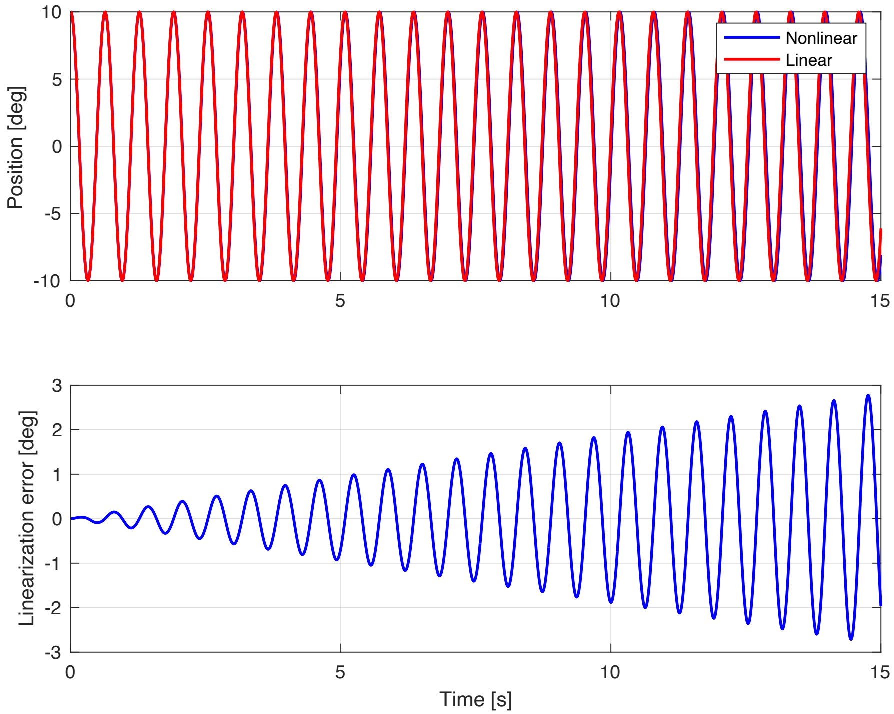
```
% initial conditions
th0 = 10 * deg2rad;
dot_th0 = 0;

% linear model parameters
A = [0, 1; -g/l, -b/m/l^2];
B = zeros(2,1);
C = eye(2);
D = zeros(2,1);
x0 = [th0; dot_th0];
```



Use a **Mux** block to form a vector-valued signal from its components.

Simulation of nonlinear systems

```
%  set simulation parameters
set_param('nonlin_vs_lin_pend', ...
    'SolverType', 'Variable-step', ...
    'Solver', 'ode45', ...
    'MaxStep', '0.01', ...
    'MinStep', 'auto', ...
    'AbsTol', 'auto', ...
    'RelTol', '1e-3', ...
    'StopTime', '15');

%  run simulation
sim('nonlin_vs_lin_pend');
```



The two oscillations are not identical; in particular, their periods are slightly different …

Simulation of nonlinear systems

The undamped linearized pendulum is an *harmonic* oscillator:

Linearized ODE (with $b = 0$)

$$ml^2\,\ddot{\theta} + b\,\dot{\theta} + mgl\,\theta = 0 \quad \Rightarrow \quad \ddot{\theta} + \omega_0^2\,\theta = 0\,, \quad \omega_0 = \sqrt{\frac{g}{l}}$$

The period of the oscillation is:

$$T_l = \frac{2\pi}{\omega_0} = 2\pi\sqrt{\frac{l}{g}}$$

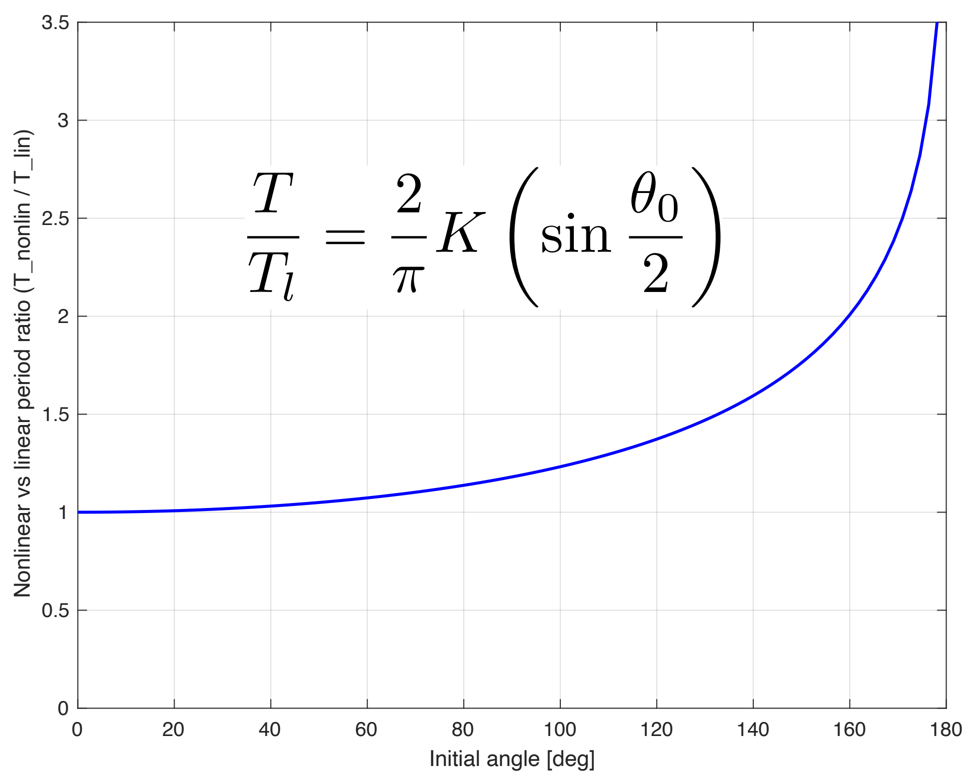
Simulation of nonlinear systems

Nonlinear undamped pendulum

$$T = 4\sqrt{\frac{l}{g}}\,K\left(\sin\frac{\theta_0}{2}\right)$$

Linear undamped pendulum

$$T_l = 2\pi\sqrt{\frac{l}{g}}$$



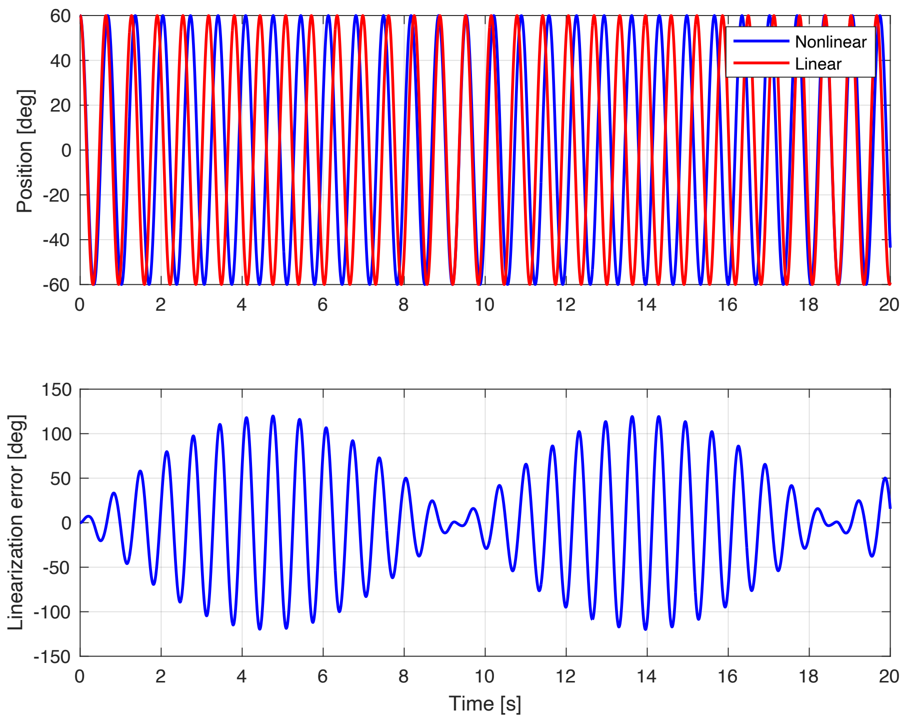
Simulation of nonlinear systems

The linearization error is:

High Order Harmonics

$$e(t) = \theta_0 \left(\cos \omega t - \cos \omega_l t\right) + \text{H.O.H.}$$

$$= -2\theta_0 \sin\left(\frac{\omega - \omega_l}{2}\, t\right) \sin\left(\frac{\omega + \omega_l}{2}\, t\right) + \text{H.O.H.}$$



$\theta(0) = 60°$
$\dot{\theta}(0) = 0$

Simulation of a sampled-data system

Example: consider the following simplified model of a DC motor:

$$P(s) = \frac{Y(s)}{U_a(s)} = \frac{k}{T\,s+1}, \qquad k = 8.3, \quad T = 0.028$$

with the armature voltage u_a [V] as input, and the shaft speed y [rad/s] as output.

Want to simulate the response of a speed PI control loop, assuming to directly measure the shaft speed (e.g. by using a *tachometer*).

Simulation of a sampled-data system

Start with the *continuous-time* PI controller:

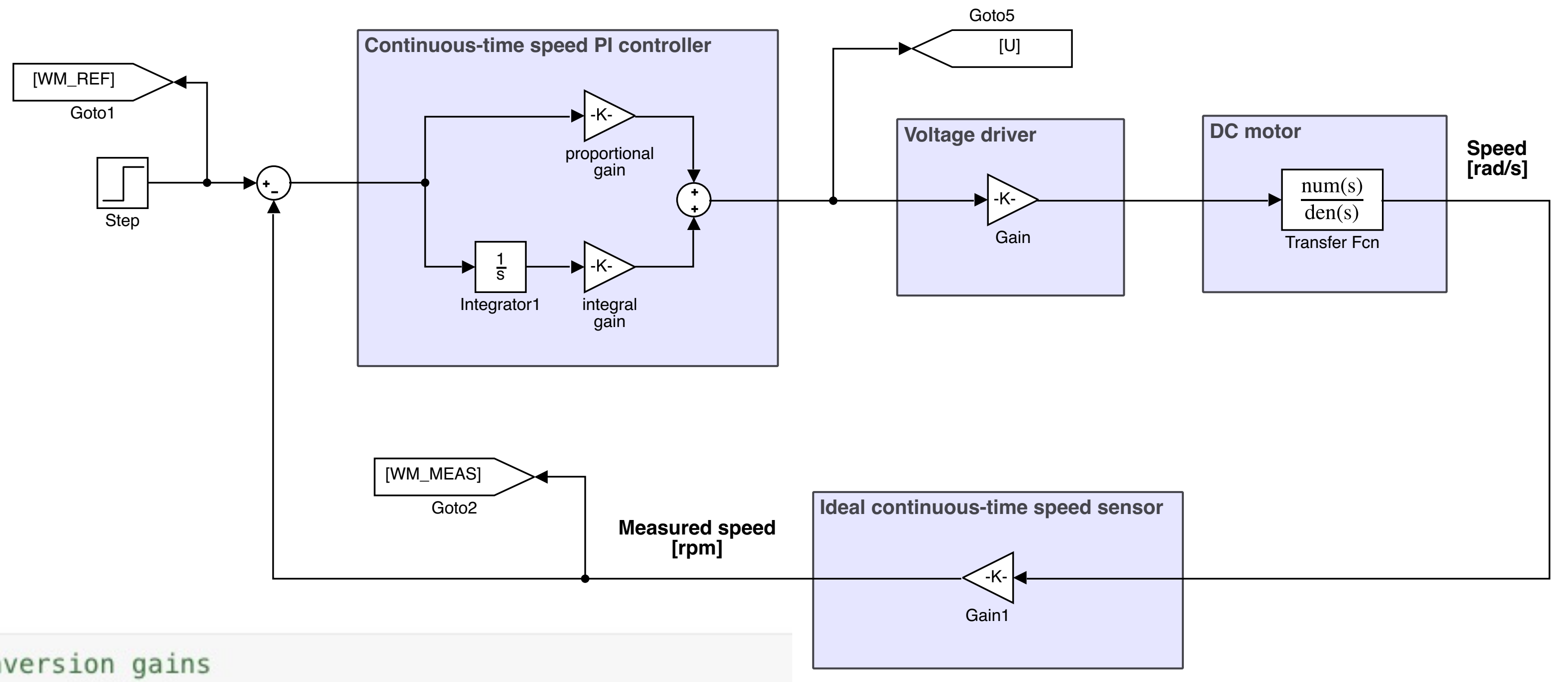
$$C(s) = \frac{U(s)}{E(s)} = K_P + \frac{K_I}{s}, \qquad \begin{aligned} K_P &= 8.4 \times 10^{-3} \\ K_I &= 1.05 \end{aligned}$$

whose input is the speed error e [rpm], and output is the voltage command u [V] provided to the motor voltage driver.

The motor driver is modelled as a static gain:

$$\frac{U_a(s)}{U(s)} = k_{\mathrm{drv}} = 0.6$$

Simulation of a sampled-data system

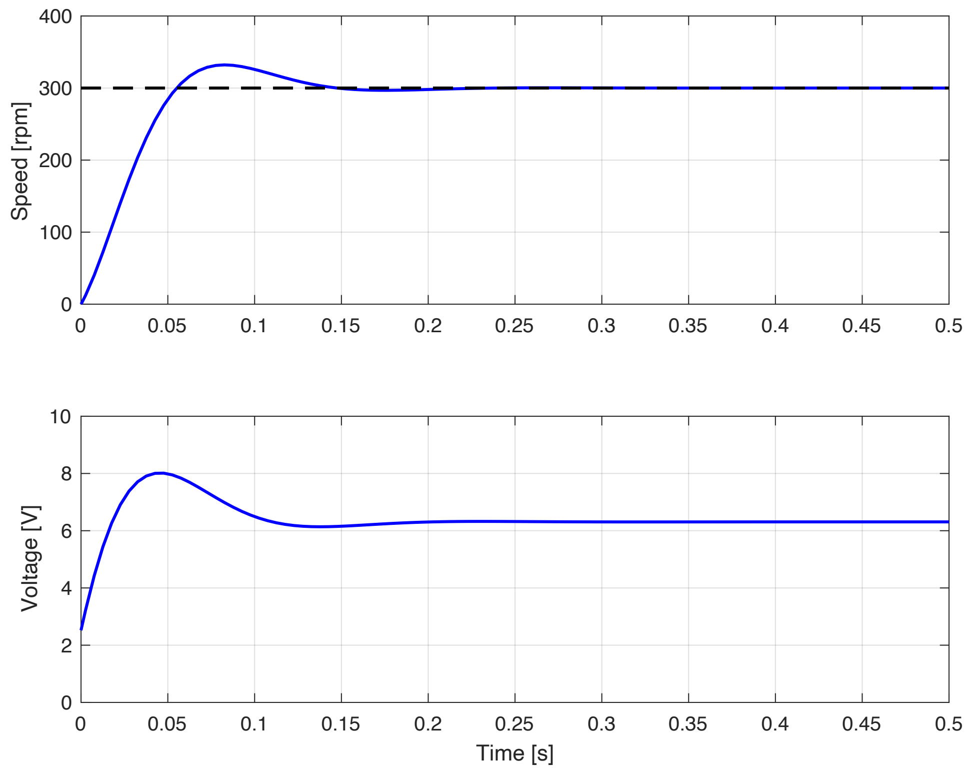



```
%   conversion gains
rads2rpm = 60/2/pi;
rpm2rads = 2*pi/60;

%   DC motor data
mot.k = 8.3;                %   static gain
mot.T = 0.028;              %   dominant time constant

%   voltage driver data
drv.kdc = 0.6;              %   attenuation gain

%   speed PI controller data
wctrl.kp = 8.4e-3;          %   proportional gain
wctrl.ki = 1.05;            %   integral gain
```

Simulation of a sampled-data system

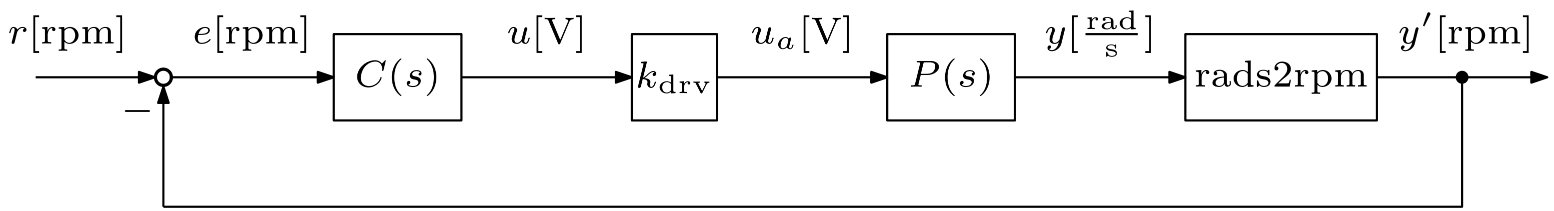
Simulation results for a step speed reference of $300\,\mathrm{rpm}$, applied at $t = 0\,s$:

```
%   set simulation params
set_param('speed_ctrl_1', ...
    'SolverType', 'Variable-step', ...
    'Solver', 'ode45', ...
    'MaxStep', '0.005', ...
    'StopTime', '0.5');

%   run simulation
sim('speed_ctrl_1');
```



Simulation of a sampled-data system

<u>Alternative</u>: use CST (the control system is a continuous-time LTI model !).




```
% plant tf
sysP = tf(mot.k, [mot.T, 1])
```

```
sysP =

      8.3
  -----------
  0.028 s + 1

Continuous-time transfer function.
```

```
% controller tf
sysC = tf([wctrl.kp, wctrl.ki], [1, 0])
```

```
sysC =

  0.0084 s + 1.05
  ---------------
         s

Continuous-time transfer function.
```

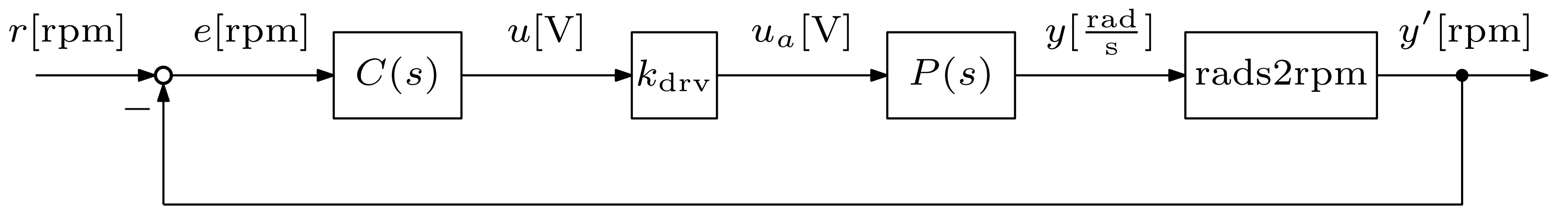
```
% closed-loop tf (from ref to out)
sysT = feedback( sysC * drv.kdc * sysP * rads2rpm, 1 )
```

```
sysT =

        0.3995 s + 49.93
  ---------------------------
  0.028 s^2 + 1.399 s + 49.93

Continuous-time transfer function.
```

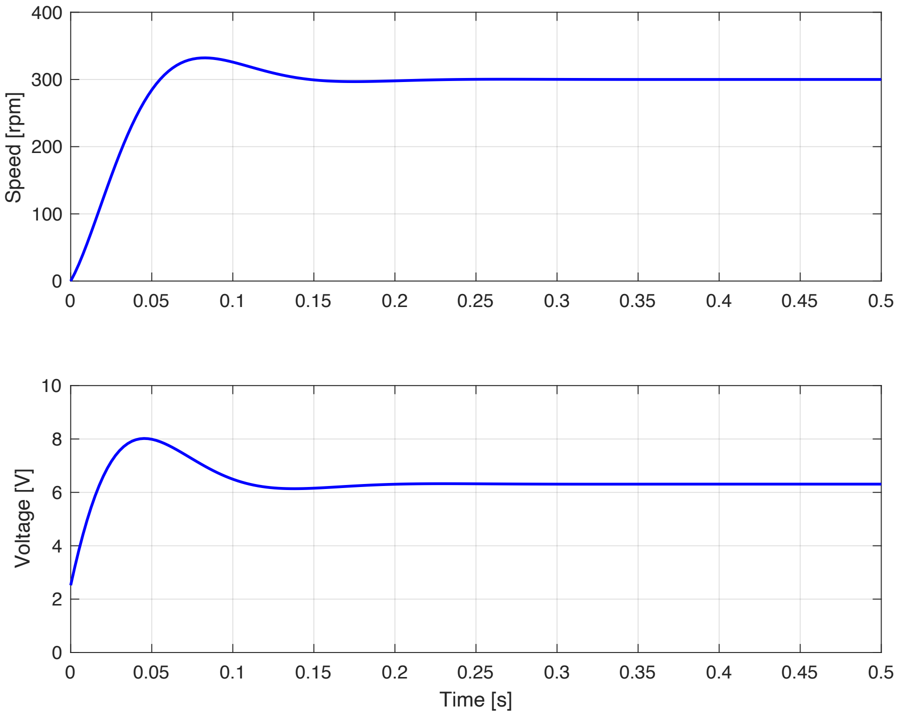
```
% closed-loop tf (from ref to controller out)
sysCS = feedback( sysC, drv.kdc * sysP * rads2rpm )
```

```
sysCS =

  0.0002352 s^2 + 0.0378 s + 1.05
  -------------------------------
    0.028 s^2 + 1.399 s + 49.93

Continuous-time transfer function.
```

Simulation of a sampled-data system




```
%  step response
opt = stepDataOptions('InputOffset', 0, 'StepAmplitude', 300);
t = linspace(0, 0.5, 200);
y = step(sysT, t, opt);
u = step(sysCS, t, opt);

%  plot results
figure;
subplot(2, 1, 1);
plot(t, y, 'b-', 'LineWidth', 1.5);
grid on;
xlim([0, t(end)]);
ylabel('Speed [rpm]');

subplot(2, 1, 2);
plot(t, u, 'b-', 'LineWidth', 1.5);
ylabel('Voltage [V]')
xlabel('Time [s]');
ylim([0, 10]);
xlim([0, t(end)]);
grid on;
```



Simulation of a sampled-data system

Consider now the *discrete-time* PI controller:

$$C(z) = \frac{U(z)}{E(z)} = K_P + \frac{K_I T_s}{z-1}, \qquad K_P = 8.4 \times 10^{-3}, \quad K_I = 1.05, \quad T_s = 0.01\,\mathrm{s}$$

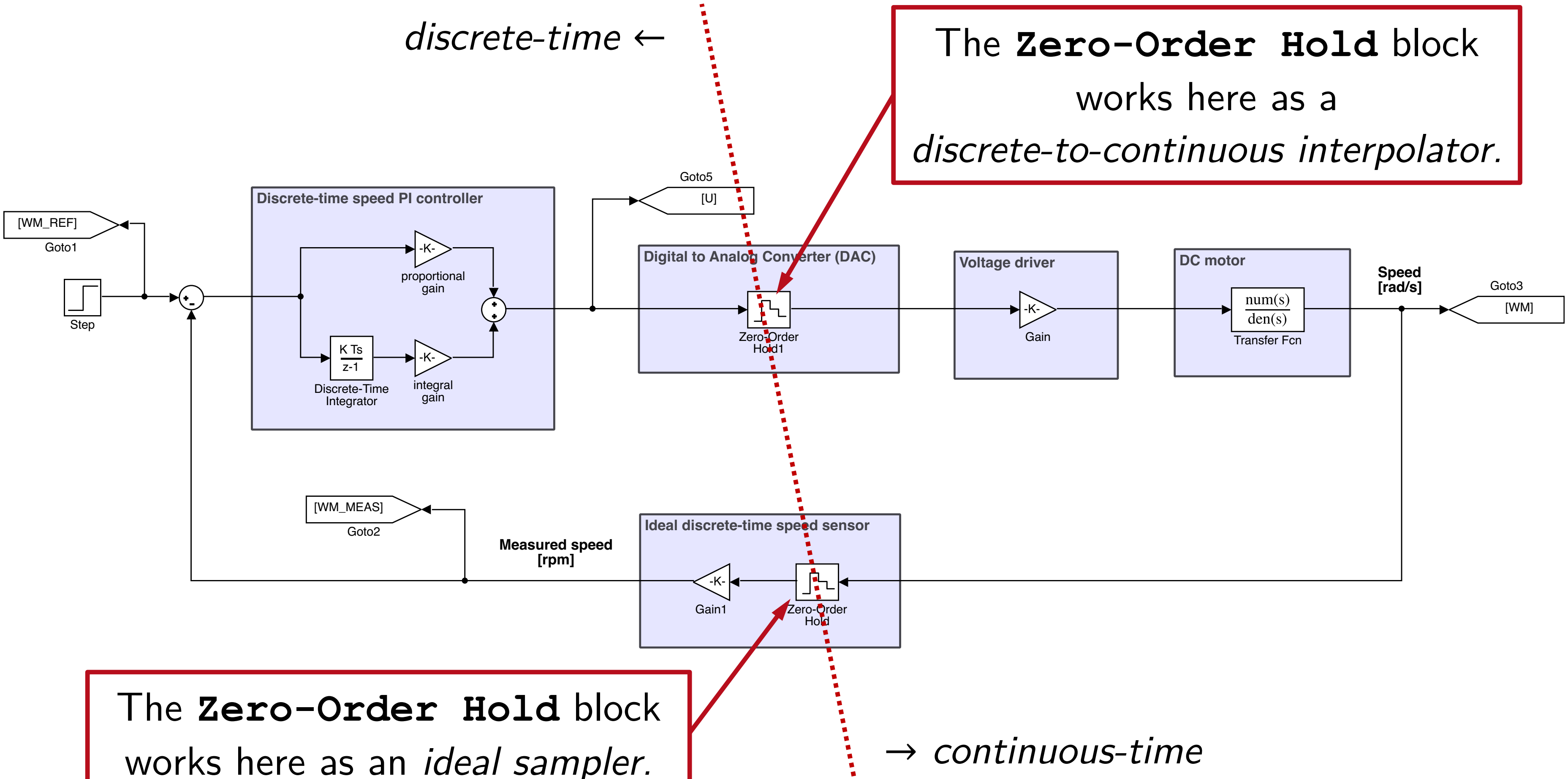
obtained by discretizing the continuous-time controller with the *Forward Euler method*.

Simulation of a sampled-data system

<u>Sampled-data system</u>

discrete-time ←

The **`Zero-Order Hold`** block works here as a *discrete-to-continuous interpolator.*



The **`Zero-Order Hold`** block works here as an *ideal sampler.*

→ *continuous-time*

Simulation of a sampled-data system

```
%   set simulation params
set_param('speed_ctrl_2', ...
    'SolverType', 'Variable-step', ...
    'Solver', 'ode45', ...
    'MaxStep', '0.001', ...
    'StopTime', '0.5');

%   run simulation
sim('speed_ctrl_2');
```

```
%   extract simulation results (discrete-time data)
t = simres2.time;                           %   time vector [s]
wm_meas = simres2.signals(1).values(:,1);   %   measured motor speed [rad/s]
wm_ref = simres2.signals(1).values(:,2);    %   motor speed reference [rad/s]
u = simres2.signals(2).values(:,1);         %   driver command [V]

%   extract simulation results (continuous-time data)
tc = simres2c.time;
wm = simres2c.signals.values(:,1);            %   actual motor speed [rad/s]

%   plot control response
figure;
subplot(2, 1, 1);
stairs(t, wm_meas, 'b-', 'LineWidth', 1.5);
hold on;
plot(tc, wm, 'r-', 'LineWidth', 1.5);
stairs(t, wm_ref, 'k--', 'LineWidth', 1.5);
grid on;
xlim([0, 0.5]);
ylabel('Speed [rpm]');
legend('Actual', 'Measured');

subplot(2, 1, 2);
stairs(t, u, 'b-', 'LineWidth', 1.5);
ylabel('Voltage [V]')
xlabel('Time [s]');
ylim([0, 10]);
xlim([0, 0.5]);
grid on;
```

Use **stairs** for drawing a *stair-step* plot.