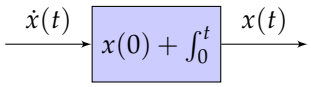
Laboratory handout 2 – Block diagrams and simulation

Given the derivative $\dot{x}(t) := \mathrm{d}x(t)/\mathrm{d}t$ of the signal $x(t)$, we recover $x(t)$ by integration:

$$x(t) = x(0) + \int_0^t \dot{x}(\tau)\,\mathrm{d}\tau. \tag{1}$$

This relationship is represented in the following **block diagram**.



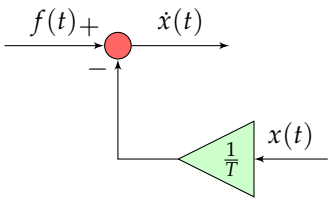
Given the **differential equation**

$$\dot{x}(t) + \frac{1}{T}x(t) = f(t), \tag{2}$$

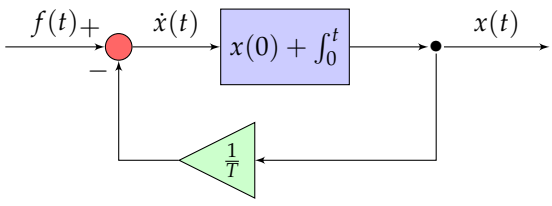
it follows that

$$\dot{x}(t) = f(t) - \frac{1}{T}x(t). \tag{3}$$

The following **partial block diagram** shows the dependence of the derivative $\dot{x}(t)$ on $x(t)$ and the input $f(t)$.



A **complete block diagram** is obtained by combining the partial block diagram with the representation of the relationship between $\dot{x}(t)$ and $x(t)$:



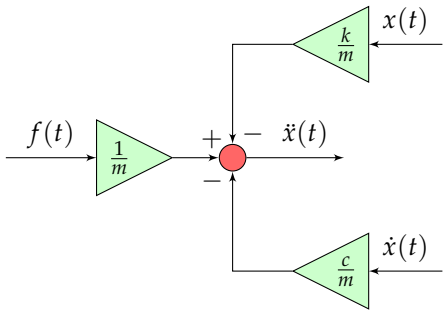
Given the differential equation

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) = f(t), \tag{4}$$

it follows that

$$\ddot{x}(t) = \frac{1}{m}\big(f(t) - c\dot{x}(t) - kx(t)\big), \tag{5}$$

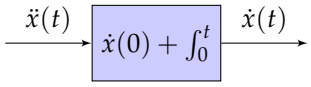
where $\ddot{x}(t) = \mathrm{d}^2x(t)/\mathrm{d}t^2$. The following partial block diagram shows the dependence of the derivative $\ddot{x}(t)$ on $x(t)$, $\dot{x}(t)$, and the input $f(t)$.



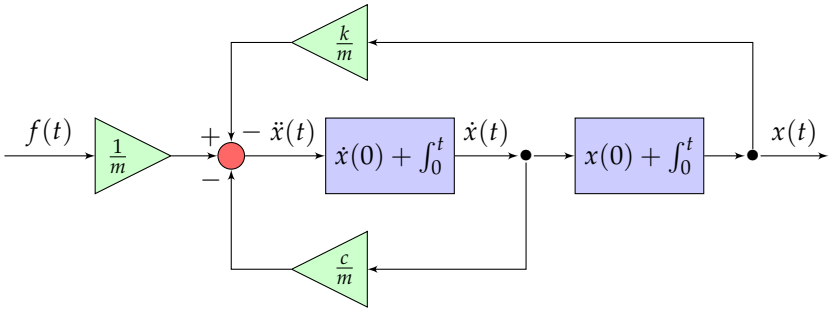
Since

$$\dot{x}(t) = \dot{x}(0) + \int_0^t \ddot{x}(\tau)\,\mathrm{d}\tau \tag{6}$$

or, equivalently,



the following block diagram provides a complete representation of the original differential equation.



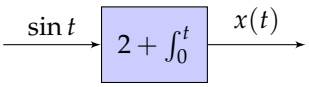
In MATLAB, the SIMULINK environment provides support for

graphical construction of block diagrams and **simulation** of the resulting dynamical system. To start SIMULINK, enter `simulink` on the command line:

```
>> simulink
```

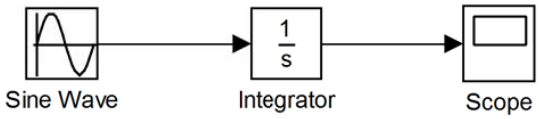
When the Simulink Library Browser window is open, enter <ctrl>+n on your keyboard to open a new **model** window.

To build the single-component block diagram



open the Continuous Library. Select the *Integrator* block and drag it to your model. Next, open the Library Browser. Select the *Sine Wave* block and drag it to your model. Finally, open the Sinks Library. Select the *Scope* block and drag it to your model. Drag each of the components in the model window to arrange them on the model canvas.

Connect the components sequentially by clicking on the appropriate output port and dragging the marker to the appropriate input port. The complete model should look like this:



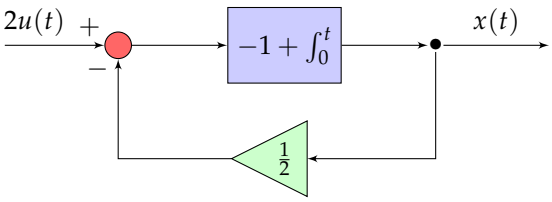
Finally, to set the properties of each component, double-click on each icon and assign the appropriate parameters, for example the initial value 2 for the integrator. Note that double-clicking on the *Scope* opens up a separate window that can be used to view the output of the *Integrator*.

To simulate the output $x(t)$ to the input $\sin t$, click the run button to start the simulation. The *Scope* shows a graph of the function

$$2 + \int_0^t \sin\tau \, \mathrm{d}\tau = 3 - \cos t. \tag{7}$$

Click on the *Autoscale* icon to fit the graph to the *Scope* window.

Let $u(t) = 1$ for $t \geq 0$ and 0 otherwise. To build a SIMULINK model representing the following block diagram



focus on the Simulink Library Browser window, and enter <ctrl>+n on your keyboard to open a new model window and save this to disk. Add an *Integrator* from the Continuous Library, a *Gain* and a *Sum* from the Math Operations Library, a *Step* from the Library Browser, and a *Scope* from the Sinks Library, and arrange these on the model canvas. You can flip the *Gain* component horizontally by selecting its icon on the canvas and entering <ctrl>+i on your keyboard.

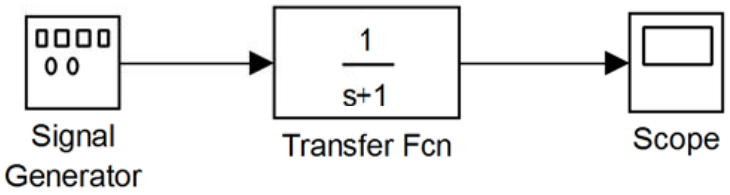
To set the properties of the components, double-click on each component and enter the appropriate settings. For the *Sum*, enter `|+-` in the "List of signs" field. (The `|` symbol is accessed by pressing <shift>+\.)

Finally, connect the components sequentially by clicking on the appropriate output port and dragging the marker to the appropriate input port. You can split off a separate connection to the *Scope* from the connection between the *Integrator* and the *Gain* by right-clicking at some point on the corresponding wire and dragging the marker to the input port on the *Scope*.

To save the output of a simulation to the MATLAB workspace, add a *To Workspace* component from the Sinks Library to your model and connect it appropriately. Double-click on its icon to set the variable name for the output, e.g., "xout". Set the "Save Format" to "Array". Once the simulation is complete, you can use the MATLAB `plot` command to graph the time history, e.g.,

```
>> plot(tout, xout)
```

The complete model should look like this:



Provided that the signal $f(t)$ is bounded by some exponential function for all $t \geq 0$ in the **time domain**, then its **Laplace transform**

$$\mathcal{L}\big[f(\#)\big](s) := \int_0^\infty f(t)e^{-st}\,\mathrm{d}t \tag{8}$$

is defined for all complex numbers s in some right half plane in the **frequency domain**.

When both sides are defined,

$$\mathcal{L}\big[\alpha f(\#)\big](s) := \alpha\mathcal{L}\big[f(\#)\big](s), \tag{9}$$

$$\mathcal{L}\big[f(\#) + g(\#)\big](s) := \mathcal{L}\big[f(\#)\big](s) + \mathcal{L}\big[g(\#)\big](s), \tag{10}$$

and

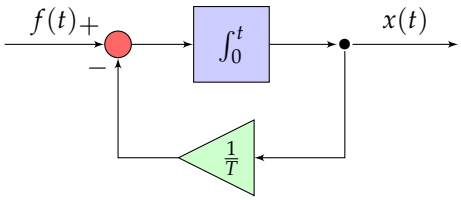
$$\mathcal{L}\left[\int_0^\# f(\tau)\,\mathrm{d}\tau\right](s) := \frac{1}{s}\mathcal{L}\big[f(\#)\big](s). \tag{11}$$

When the relationship between the Laplace transforms $X(s)$ and $Y(s)$ is of the form of a product

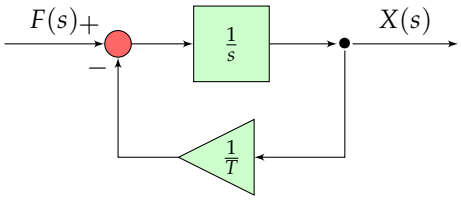
$$Y(s) = H(s) \cdot X(s) \tag{12}$$

for some **transfer function** $H(s)$, given by the Laplace transform of the corresponding **unit impulse response** $h(t)$, then $y(t)$ is given by the convolution $\big(h(\#) * x(\#)\big)(t)$.

Provided that $x(0) = 0$, the time-domain block diagram



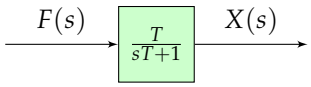
is equivalent to the frequency-domain block diagram



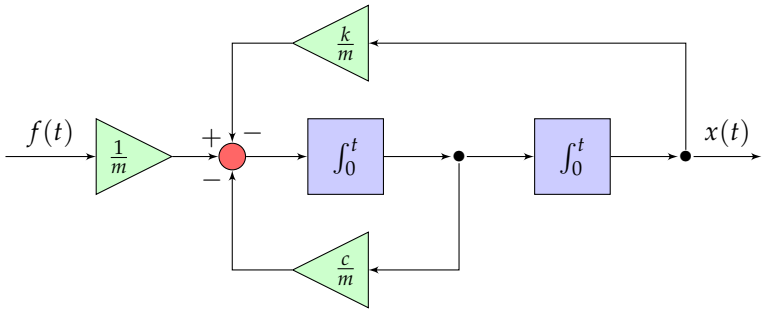
where $F(s) := \mathcal{L}\big[f(\#)\big](s)$ and $X(s) := \mathcal{L}\big[x(\#)\big](s)$. This implies that

$$X(s) = \frac{1}{s}\left(F(s) - \frac{1}{T}X(s)\right) \Rightarrow X(s) = \frac{T}{sT+1}F(s) \tag{13}$$

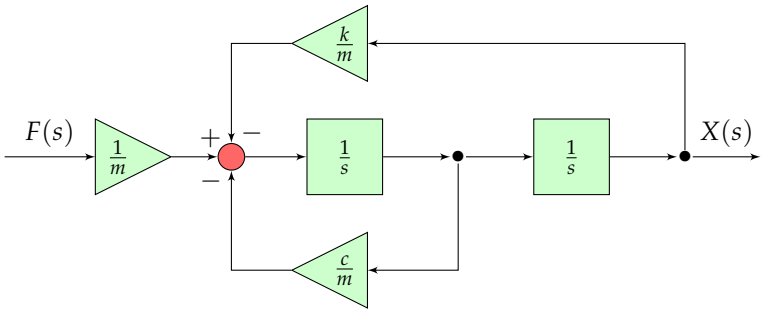
in terms of the transfer function $H(s) = T/(sT+1)$, or, equivalently,



Similarly, provided that $x(0) = 0$ and $\dot{x}(0) = 0$, the time-domain block diagram



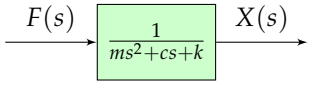
is equivalent to the frequency-domain block diagram



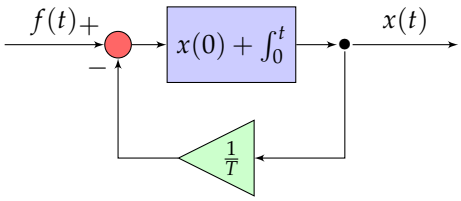
where $F(s) := \mathcal{L}\big[f(\#)\big](s)$ and $X(s) := \mathcal{L}\big[x(\#)\big](s)$. This implies that

$$X(s) = \frac{1}{ms^2}\big(F(s) - csX(s) - kX(s)\big) \Rightarrow X(s) = \frac{1}{ms^2 + cs + k}F(s) \quad (14)$$

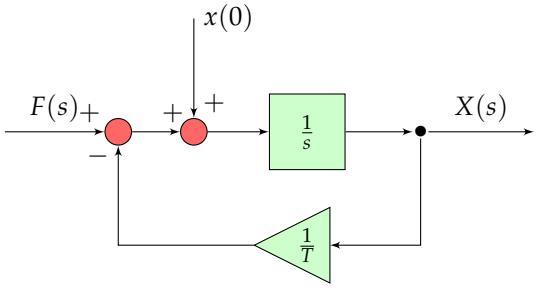
in terms of the transfer function $H(s) = 1/(ms^2 + cs + k)$, or, equivalently,



When $x(0) \neq 0$, the time-domain block diagram



is equivalent to the frequency-domain block diagram



where $F(s) := \mathcal{L}\big[f(\#)\big](s)$ and $X(s) := \mathcal{L}\big[x(\#)\big](s)$. This implies that

$$X(s) = \frac{1}{s}\left(x(0) + F(s) - \frac{1}{T}X(s)\right) \Rightarrow X(s) = \frac{T\big(x(0) + F(s)\big)}{sT + 1} \quad (15)$$

The **free response** when $f(t) = 0$ for all t is then equal to the unit impulse response corresponding to the transfer function

$$\frac{Tx(0)}{sT + 1}. \quad (16)$$

In MATLAB, a transfer function that is the ratio of two polynomials in s may be constructed using the `tf` command. The command

```
>> sys=tf(2,[2 1])
```

assigns a representation of the transfer function

$$H(s) = \frac{2}{2s+1} \tag{17}$$

to the variable `sys`. A plot of the corresponding unit impulse response is obtained using the `impulse` command:

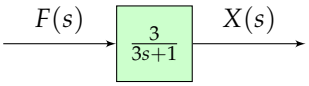
```
>> impulse(sys)
```

Similarly, If $\mathcal{L}\big[x(\#)\big](s) = H(s) \cdot \mathcal{L}\big[f(\#)\big](s)$ and $f(t) = 1$ for all $t \geq 0$, then a plot of the corresponding **unit step response** $x(t)$ is obtained using the `step` command:

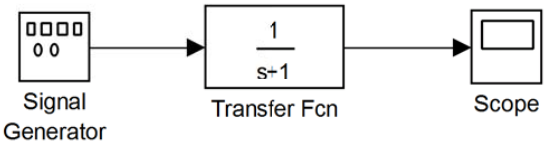
```
>> step(sys)
```

When the relationship between the Laplace transforms of the input and output to a dynamical system can be expressed in terms of a transfer function $H(s)$ that is the ratio of two polynomials in s, then the corresponding block diagram can be constructed in SIMULINK by adding a *Transfer Fcn* component from the Continuous Library to a model.

The SIMULINK model corresponding to the block diagram



may then given by



where the numerator and denominator coefficients entered in the corresponding fields of the *Transfer Fcn* component are 3 and [3 1], respectively. Note that the input and output signals to the *Transfer Fcn* component are functions in the time domain. The frequency-domain transfer function represents a time-domain convolution.

Exercises

1. Draw a time-domain block diagram with input $f(t)$ and output $y(t)$ representing the differential equation

$$\dot{y}(t) = f(t).$$

2. Draw a time-domain block diagram with input $f(t)$ and output $x(t)$ representing the differential equation

$$\ddot{x}(t) - x(t) = f(t).$$

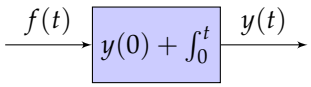
3. Draw a frequency-domain block diagram with input $F(s) := \mathcal{L}\big[f(\#)\big](s)$ and output $Y(s) := \mathcal{L}\big[y(\#)\big](s)$ representing the differential equation

$$\ddot{y}(t) = f(t).$$

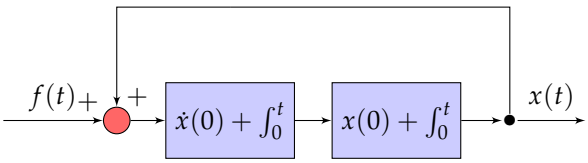
Find an expression for $Y(s)$ in terms of $F(s)$, $y(0)$, and $\dot{y}(0)$.

Solutions

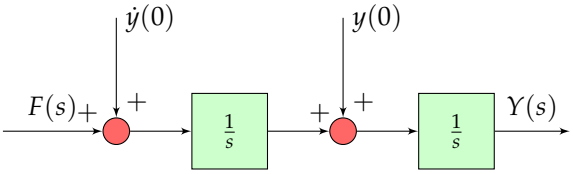
1. Here,



2. Here,



3. Here,



This implies that

$$Y(s) = \frac{1}{s}\left(y(0) + \frac{1}{s}\left(\dot{y}(0) + F(s)\right)\right) = \frac{sy(0) + \dot{y}(0) + F(s)}{s^2}.$$

Prelab Assignments

Complete these assignments before the lab. Show all work for credit.

1. Consider the differential equation

$$\ddot{x}(t) + 2\dot{x}(t) + 40x(t) = f(t).$$

   (a) Draw a time-domain block diagram representation of the original differential equation in terms of integrators, amplifiers, summing junctions, and splitting junctions. Don't forget the initial conditions. Label all connections between components to show the corresponding signals.

   (b) Draw a frequency-domain block diagram representation for the case of 0 initial conditions and use this to express $X(s) := \mathcal{L}\big[x(\#)\big](s)$ in terms of $F(s) := \mathcal{L}\big[f(\#)\big](s)$ and a transfer function $H(s)$.

2. Consider the differential equation

$$\frac{1}{6}\ddot{\theta}(t) + 2\dot{\theta}(t) + 9.8\cos\theta(t) = 0$$

   (a) Draw a time-domain block diagram representation of the original differential equation in terms of integrators, amplifiers, summing junctions, splitting junctions, and a component representing the nonlinear relationship $\theta \mapsto \cos\theta$. Don't forget the initial conditions. Label all connections between components to show the corresponding signals.

Lab instructions[1]

[1] These notes are an edited version of handouts authored by Andrew Alleyne.

In this lab you will learn how to use the SIMULINK environment in MATLAB to model and simulate dynamic systems using block diagrams. You can think of SIMULINK as a tool that allows programming in a graphical manner. Instead of large amounts of code, you can simply add pre-made components into a "model" window and connect the components' inputs and outputs to create a system that can be simulated in SIMULINK. The progress of the simulation can be monitored while the simulation is running, and the final results can be made available in the MATLAB workspace when the simulation is complete.

The steps required in order to simulate a system using SIMULINK are listed below.

1. Write the governing differential equations of the system.
2. Create a partial block diagram representing the relationship between the highest derivative in these equations and the remaining terms.
3. Use relations between the inputs and outputs of the partial block diagram to created a full simulation model using only integrators, amplifiers, summing, and splitting junctions.
4. Convert your block diagram to a SIMULINK representation and assign appropriate parameters to all components.
5. Decide on configuration parameters and run the simulation.
6. Display and analyze the results.

To build a SIMULINK representation, you must first add individual components to your model from the SIMULINK component libraries. You can do this by opening a library, selecting the icon for the desired component, and either dragging the component into the model window, or entering <ctrl>+i on your keyboard.

The components needed for this lab are contained in the Continuous Library (*Integrator*), the Math Library (*Gain* and *Sum*), Source

Library (*Step*), and the Sinks Library (*Scope* and *To Workspace*). Each library can be opened by double-clicking on its icon in the Simulink Library Browser. Browse through each of these libraries to become acquainted with the available components. The function of each components should be obvious from its title. However, if you are unsure about a component's function or its use, double-click on the component to open its dialog box and then select Help to get a detailed explanation of its use.

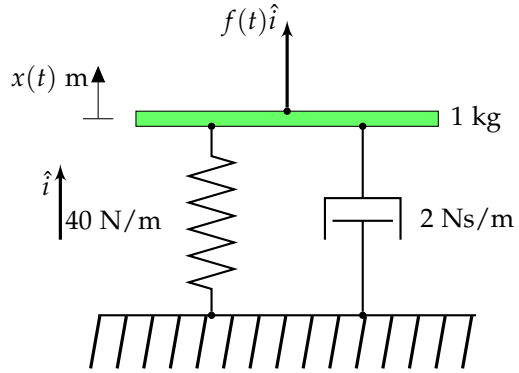
Input and output ports on individual components are represented by angle brackets ">" pointing into or away from the component, respectively. To connect two components, click on the output port, drag the marker to the corresponding input port, and release the mouse button. A line with an arrowhead should now appear showing the connection. To split a connection, right-click on the desired location of the splitting junction, drag the new connection to the appropriate input port, and release the mouse button.

A connection or component can be deleted in SIMULINK by highlighting it and hitting the <delete> key on your keyboard. To change the position of a component, you can simply drag it from one location to another and its connections will remain intact. Finally, if you want to change the size of a component for readability, select the component and drag any corner until it is the desired size.

Configuration parameters are set by entering <ctrl>+E on your keyboard. You will find settings that determine the simulation duration, govern how SIMULINK approximates the solution to the corresponding differential equation, and control data logging during simulation. As an example, to ensure that data is output every 0.01 s, independently of the steps taken by the solver, select the Data Import/Export pane. Scroll down and expand the "Additional parameters" area. Select "Produce specified output only" in the "Output options" drop-down menu and enter 0:0.01:10 in the "Output times" text area.

A mechanical suspension

Consider the motion of the mechanical suspension shown below, able to translate along the direction described by the vector $\hat{\imath}$.



The displacement of the 1 kg mass is governed by the differential equation

$$\ddot{x}(t) + 2\dot{x}(t) + 40x(t) = f(t). \tag{18}$$

If $X(s) := \mathcal{L}\big[x(\#)\big](s)$ and $F(s) := \mathcal{L}\big[f(\#)\big](s)$, then

$$X(s) = \frac{(s+2)x(0) + \dot{x}(0) + F(s)}{s^2 + 2s + 40}. \tag{19}$$

1. Refer to the time-domain block diagram found in the prelab assignment. Construct a SIMULINK representation of this dynamic system by opening a new SIMULINK model window, adding the corresponding components, and connecting these appropriately. Use a *Constant* component from the Library Browser to represent a constant $f(t)$. Note that the orientation of the *Gain* components may be reversed by selecting the corresponding icons and entering <ctrl>+i on the keyboard. Double-click on each *Gain* component to enter the corresponding numerical constant.

2. Add a *Scope* component and a *To Workspace* component to monitor $x(t)$ during simulation and store the corresponding time history to disk, respectively. Double-click on the *To Workspace* component and select "Array" in the "Save Format" drop-down menu. Enter "position" in the "Variable name" field, and 2000 in the "Limit data points to last" field.

3. Free response:

   (a) Enter initial conditions for the two integrators corresponding to an initial position of 0.1 m and an initial velocity of 0 m/s. Enter 0 for the value of the constant input. Enter <ctrl>+s on your keyboard to save your SIMULINK model to the `c\matlab\me340` directory.

   (b) Double-click on the *Scope* and click the run button to start the simulation. Click on the "Autoscale" icon to fit the graph to the window. In the MATLAB command window, enter

   ```
   >> time_ic=tout;
   >> pos_ic=position;
   >> plot(time_ic, pos_ic)
   ```

   to save the sequence of time steps and the position time history to the MATLAB variables `time_ic` and `pos_ic`, respectively, and to graph $x(t)$.

   (c) In the MATLAB command window, use the `tf` command to construct a transfer function equal to the right hand side of (19) when $F(s) = 0$, $x(0) = 0.1$, and $x'(0) = 0$, and store the result in the MATLAB variable `sys`. Enter

   ```
   >> [pos_imp,time_imp] = impulse(sys, 10);
   ```

   to generate and graph the corresponding free response, and to store the resulting sequence of time steps and positions in `time_imp` and `pos_imp`, respectively.

4. Step response:

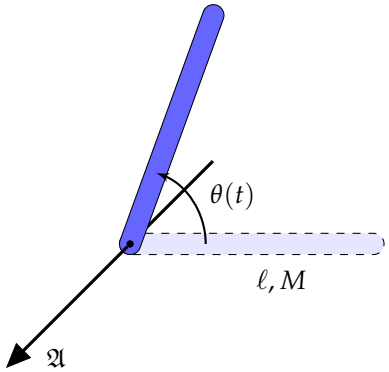
   (a) Substitute a *Step* component from the Library Browser for the *Constant* component representing the input. Double-click on the *Step* component, enter 0 in the "Step time" and "Initial value" fields, and enter 20 in the "Final value" field.

   (b) Enter 0 for the initial conditions of the two integrators and click the run button to start the simulation.

5. Read Report Assignment 1. and produce all the plots before you go on to the next section.

A nonlinear pendulum

Consider the motion of the slender rod of length $\ell = 25$ cm and mass $M = 8$ kg shown below, able to rotate about a horizontal axis $\mathfrak{A}$ that is perpendicular to the rod, under the influence of a vertical gravitational field of acceleration $g = 9.8$ m/s^2 and a viscous damping torque proportional to the angular velocity $\dot{\theta}(t)$ with proportionality constant $b = 0.2$ Nms.



The orientation $\theta(t)$ of the rod is governed by the differential equation

$$\frac{M\ell^2\ddot{\theta}(t)}{3} + b\dot{\theta}(t) + \frac{Mg\ell\cos\theta(t)}{2} = 0 \qquad (20)$$

This equation is **nonlinear** because of the $\cos\theta(t)$ term. Such a nonlinearity may be represented in SIMULINK using a *Trigonometric Function* component from the Math Library.

1. Refer to the time-domain block diagram found in the prelab assignment. Open the SIMULINK representation of the mechanical suspension with constant input equal to 0, and select the "Save As..." menu item from the "File" menu to save a copy to disk with a new name. Add a *Trigonometric Function* component from the Math Library and double-click on this component to select the appropriate function. Modify the connections between the components to represent the nonlinear pendulum. Double-click on each *Gain* component to enter the corresponding numerical constant.

2. Enter 0 for the initial conditions of the two integrators, enter <ctrl>+s on your keyboard to save your model, and click the run button to start the simulation.

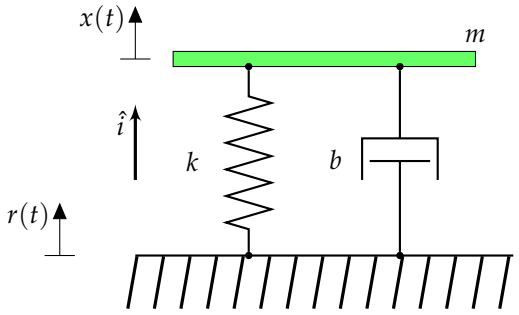
3. For $\theta(t) \approx -\pi/2$, we obtain the **linear approximation** $\cos\theta(t) \approx \pi/2 + \theta(t)$. The governing equation becomes

$$\frac{M\ell^2\ddot{\theta}(t)}{3} + b\dot{\theta}(t) + \frac{Mg\ell}{2}\theta(t) = -\frac{Mg\pi\ell}{4}. \tag{21}$$

   Make the appropriate modifications to your SIMULINK model and select the "Save As..."' menu item from the "File" menu to save to disk with a new name. Simulate this linear approximation with different initial conditions and compare to the nonlinear simulation with same initial conditions. At what initial conditions does the linear approximation become poor? Note: you may want to store the results of your simulations to different MATLAB variables or use the `hold` command with the `plot` command.

A quarter-car model

Consider the quarter-car vehicle-suspension model shown below, where the mass $m = 250$ kg represents 1/4 of the mass of a car body, and the spring with stiffness $k = 2000$ N/m and damper with damping coefficient $b = 3000$ Ns/m represent a suspension spring and shock absorber, respectively. The position $x(t)$ of the car body equals 0 when the road input $r(t)$ equals 0 and the spring supports the weight of the body.



The vertical displacement of the body is governed by the differential equation

$$m\ddot{x}(t) + b\dot{x}(t) + kx(t) = b\dot{r}(t) + kr(t) \tag{22}$$

With $x(0) = \dot{x}(0) = r(0) = 0$, the relationship between the ground input $r(t)$ and the displacement $x(t)$ is described by the transfer function

$$H(s) = \frac{bs + k}{ms^2 + bs + k} \tag{23}$$

1. Construct a SIMULINK model using a *Transfer Fcn* component from the Continuous Library and double-click on this component to enter the polynomial coefficients for the numerator and denominator of $H(s)$. Let the input signal be a step of size 0.3 m and simulate the corresponding dynamic system.

2. Use two *To Workspace* components to store the input and output time histories to the MATLAB workspace and plot the difference $z(t) = x(t) - r(t)$.

Report Assignments

Complete these assignments during the lab. *Show all work for credit.*

1. In the analysis of the mechanical suspension:

   (a) Use the MATLAB `subplot` command to graph the free response using the SIMULINK output with i) the variable-step `ode45` integrator with relative tolerance 10^{-3}; ii) the fixed-step integrator `ode3` with automatic step-size; and iii) the fixed-step integrator `ode3` with step size 0.1; as well as using the output of the `impulse` command. You can modify the simulation tolerances by putting focus on the SIMULINK model, entering <ctrl>+e on your keyboard, and adjusting the "Solver options" entries. Give the plot the title "Plot 1: Simulation and Analytical Response of a Mass-Spring-Damper System", and label the x and y axes. Explain the observed differences.

   (b) Use the MATLAB `hold` command to plot several step responses in the same graph. Use the SIMULINK output with different values in the "Final value" field of the *Step* component.

2. In the analysis of the nonlinear pendulum:

   (a) Use the MATLAB `hold` command to plot the solution of the nonlinear and linearized models for initial conditions where the linear approximation is *invalid*. Give the plot the title "Plot 2: Linear and Nonlinear Simulation in Nonlinear Range", and include a legend, as well as labels for the x and y axes.

   (b) Propose a range of initial conditions for which the linear approximation is reasonable. Justify your answer.

   (c) A nonlinear model is typically more accurate than the linear model, but also more costly to implement. What engineering considerations might determine whether a linear approximation is appropriate?

3. In the analysis of the quarter-car model:

(a) Plot the suspension travel $z(t)$ when the input is a 0.3 m step. Give the plot the title "Plot 3: Suspension Travel of the Quarter-Body Model to a 30 cm Curb", and label the x and y axes.

(b) Plot the body position $x(t)$ when the input is a 0.3 m step. Give the plot the title "Plot 4: Quarter-Body Position Response to a 30 cm Curb", and label the x and y axes.

(c) Suppose that you were designing the suspension so that the passengers of the vehicle would be protected from the effects of the car hitting the curb. What dynamic information would you want to obtain from the quarter-body simulation so that you could tell if a person could be hurt by the collision with the curb? For example, which signal in your block diagram would provide the dynamical information that is most closely related to an injury? Explain your answer.