To recap

- A system's TF is a complex function
  – Represented in terms of its magnitude and phase
- Bode plots
  – plot of magnitude and phase
- Bode plots of complex TFs can be expressed in terms of simpler terms
- Stability criteria:
  – The feedback control system is <u>stable</u> if and only if all the *poles of the* ***closed loop transfer function*** G_{CL} have a negative real part. Otherwise the system is unstable.

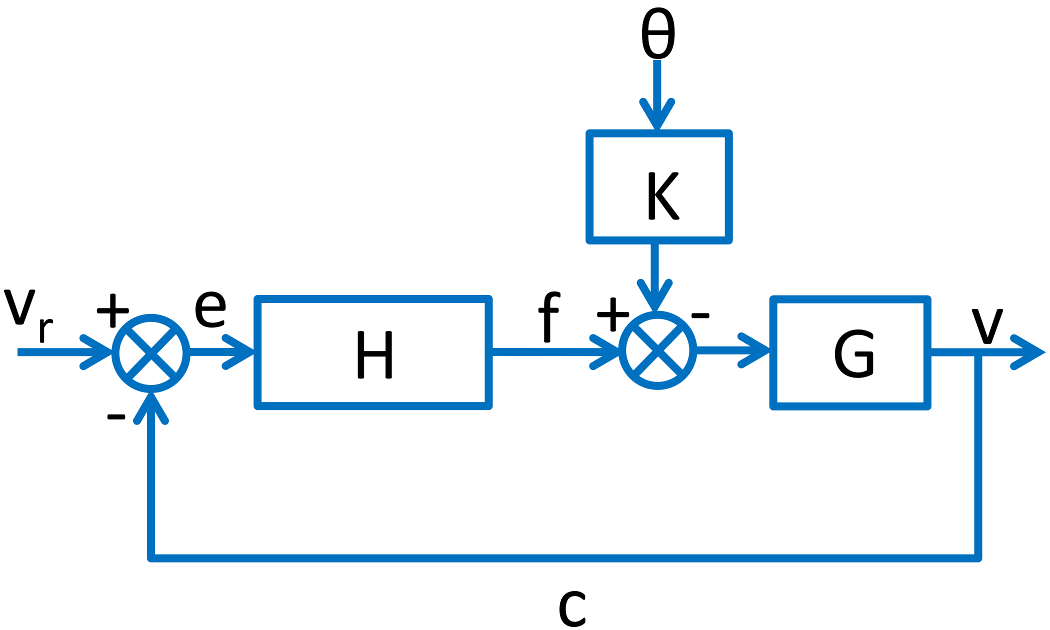
To recap

- Stability in terms of the ***open loop gain***
  - A closed loop system is stable if the unity gain frequency is lower than the -180^0 crossing.
  - Rule of thumb: the system is (almost always) stable if $|G_{OL}| \propto \frac{1}{f}$ at the unity gain frequency
- How close to instability is a system?
  - Gain and phase margin
- Typical compensators
  - Phase-lag
  - Phase-lead
  - “Boost”

SIMULINK

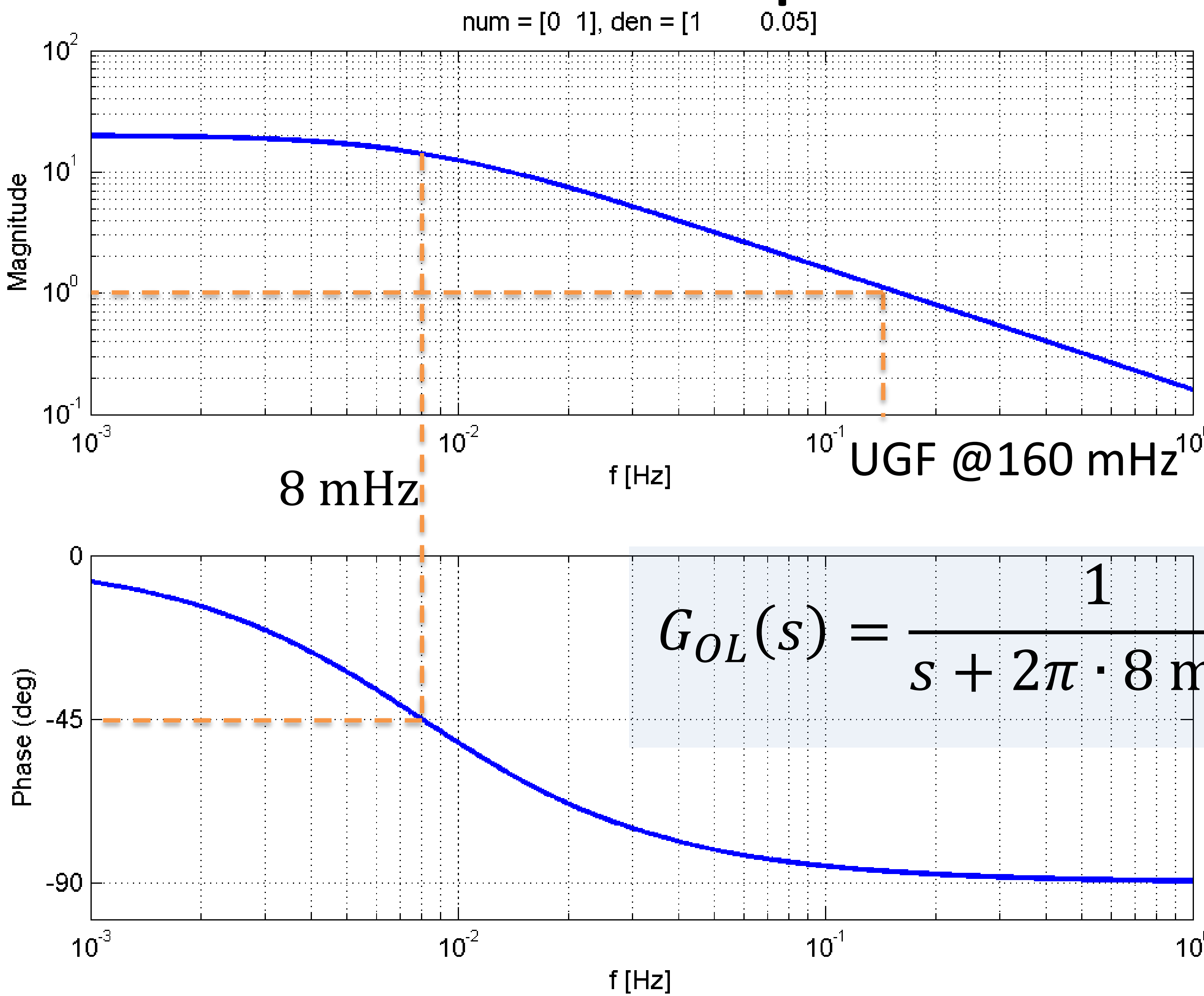
- Simulating systems in the time-domain
- Let's refer back to the cruise control example
- Parameters used previously
  - $H = 1000\, {}^{N}/_{(m/s)}$
  - $G = \frac{1/m}{s+b/m}$
  - $m = 1000\ kg$
  - $b = 50\ kg/s$

$$e = \frac{1}{1+G\cdot H} v_r + \frac{K\cdot G}{1+G\cdot H}\theta$$



$$G_{OL}(s) = \frac{1}{s + 2\pi\cdot 8\ \text{mHz}}$$

OL TF bode plot



$$G_{OL}(s) = \frac{1}{s + 2\pi \cdot 8\ \mathrm{mHz}}$$

cruise_freqdomain.m

Building the model

First we need to model the response of the car v to an input force f.

- This was previously described by transfer function G
- G was derived from

$$m\frac{dv}{dt} = f - bv$$

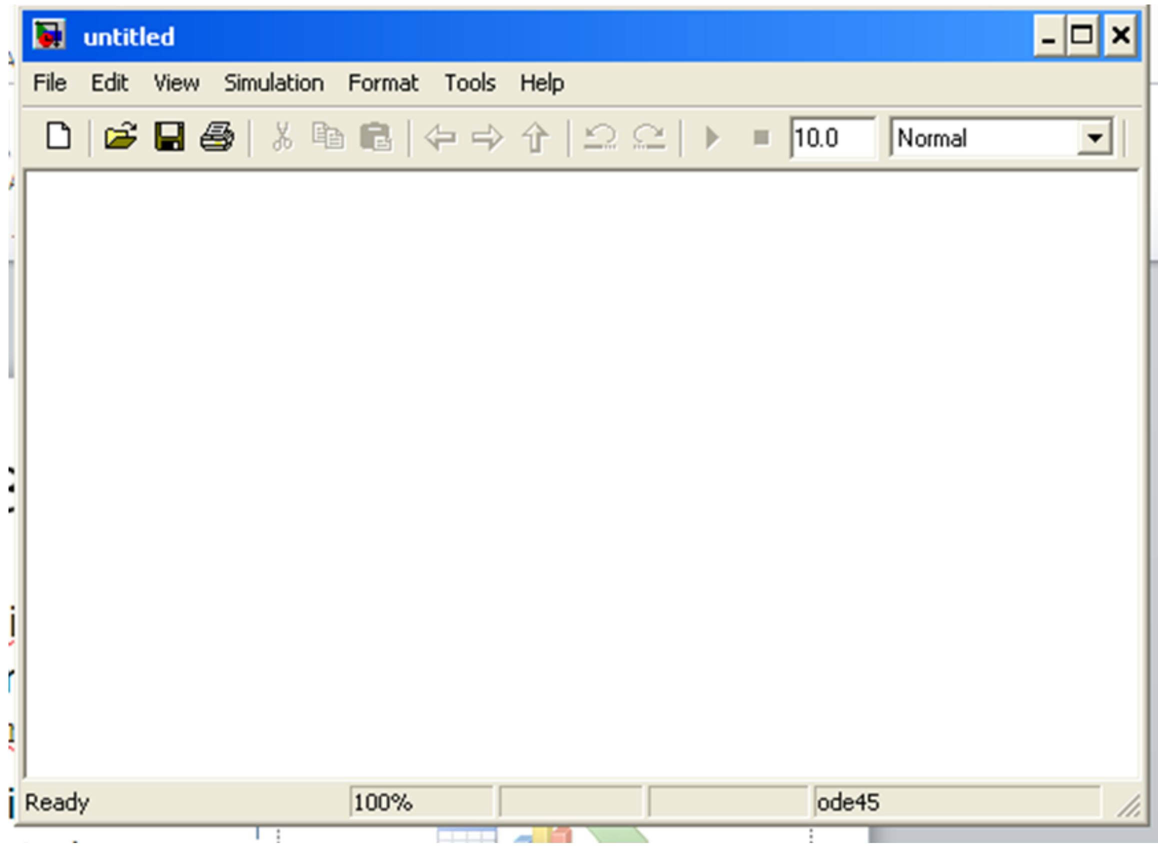
- SIMULINK simulates this differential equation

Building the model

- Type simulink at the MATLAB prompt

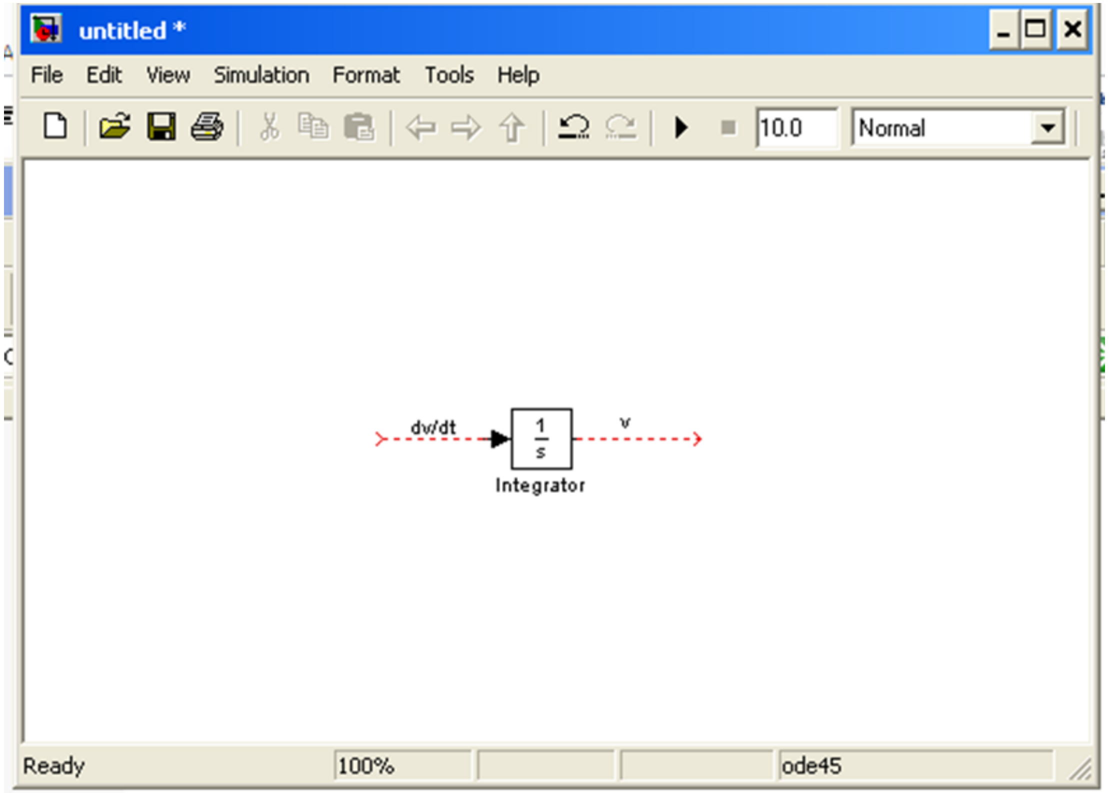
```
>> simulink
```

- The “Simulink Library Browser” window opens.
- File → New → Model
- The model window opens



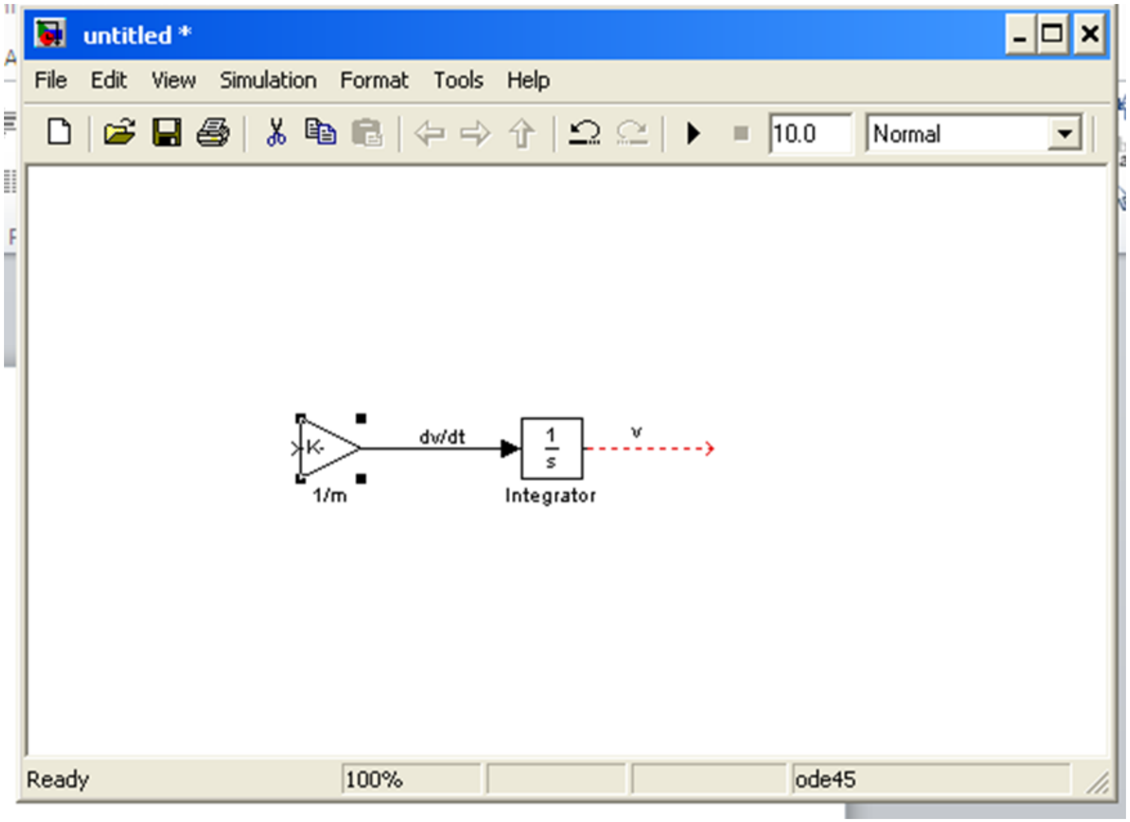
Building the model

- The relationship between v and dv/dt is simply $v = \int \frac{dv}{dt} dt$. Add the *Integrator* block by selecting it and dragging it to the model window. The integrator block is in the simulink library browser under the continuous section.
- Extend the signal in and signal out lines, double click on top of each line and label them as v and dv/dt



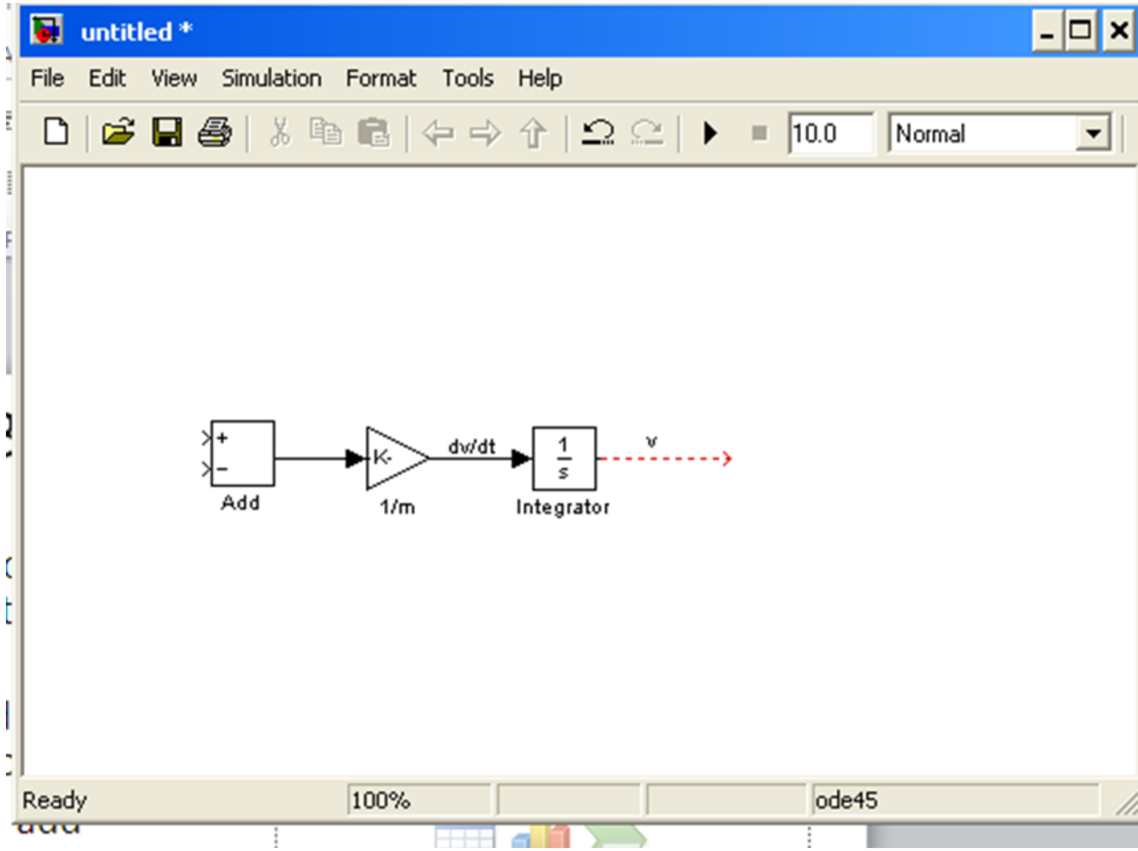
Building the model

- Re-arranging terms $\frac{dv}{dt} = \frac{1}{m}(f - bv)$: the change in velocity is $1/m$ times $(f - bv)$. In the library browser, under the math operations section, select and drag the *Gain* block.
- Connect its output to the integrator's input.
- Double-click on the gain block, set the gain value to $1/m$, and click ok.
- Double-click right under the gain block and label it as $1/m$.



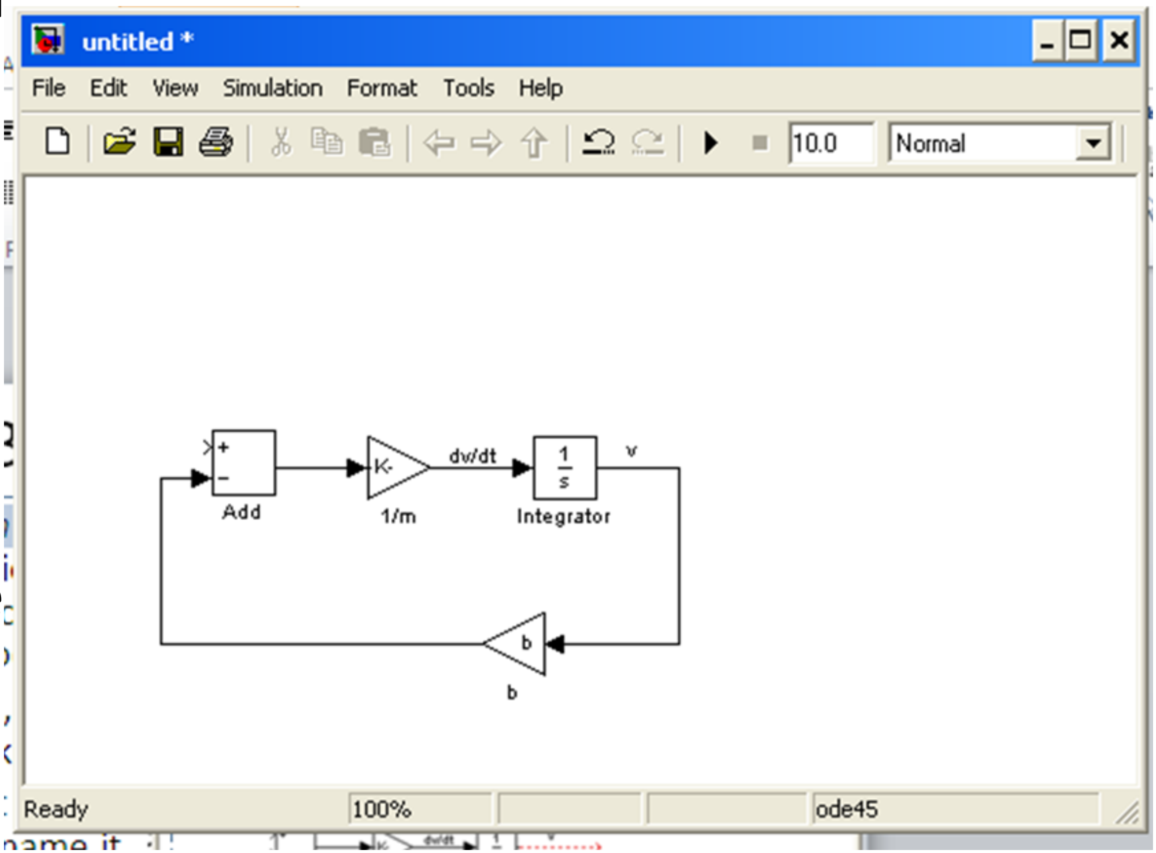
Building the model

- Under the math operations section, select and drag the *Add* block.
- Connect the add block with the gain block
- Double-click the add block and change the sign as shown in figure $(f - bv)$.
- One input to the add block will be the engine force f while the other will be the friction force $- bv$



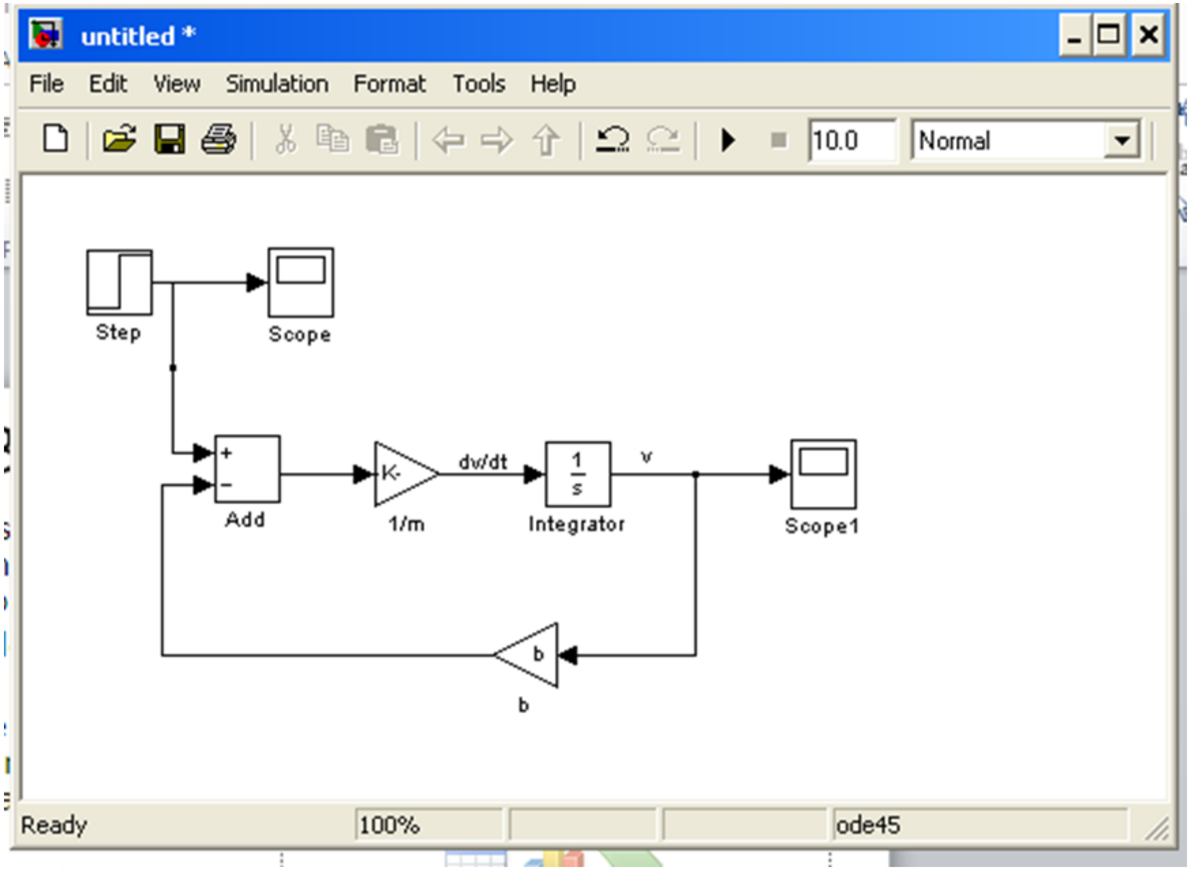
Building the model

- Add another *Gain* block, and change its orientation by selecting it, clicking on Format → Flip block
- Double-click on it, set the gain to b and click ok.
- Double-click right under the block and re-name it as b.
- Connect the output of the integrator to the input gain block b.
- Connect the output of gain block b with the negative input of the add block.



Building the model

- In the library browser, under the sources section, select and drag the *Step* block.
- Connect the step block to the add block.
- Double-click on the step block and set the final value to 500 (the same value we had used before)
- In the Sinks section, select Scope and drag it to the window. Connect it to the output of the step block.
- Select and drag a second scope and connect it to the integrator's output
- We are ready for the simulation

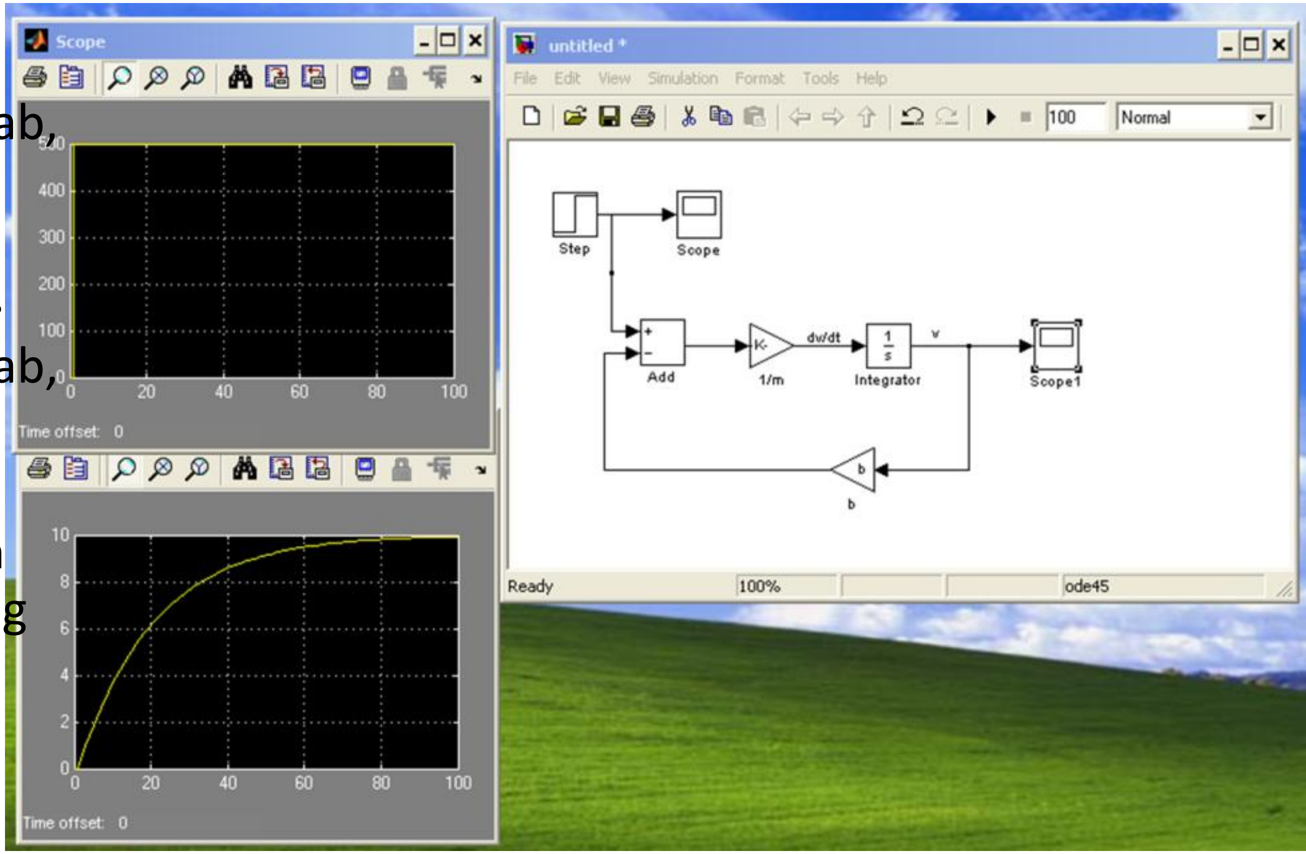


$$\frac{dv}{dt} = \frac{1}{m}(f - bv)$$

Results

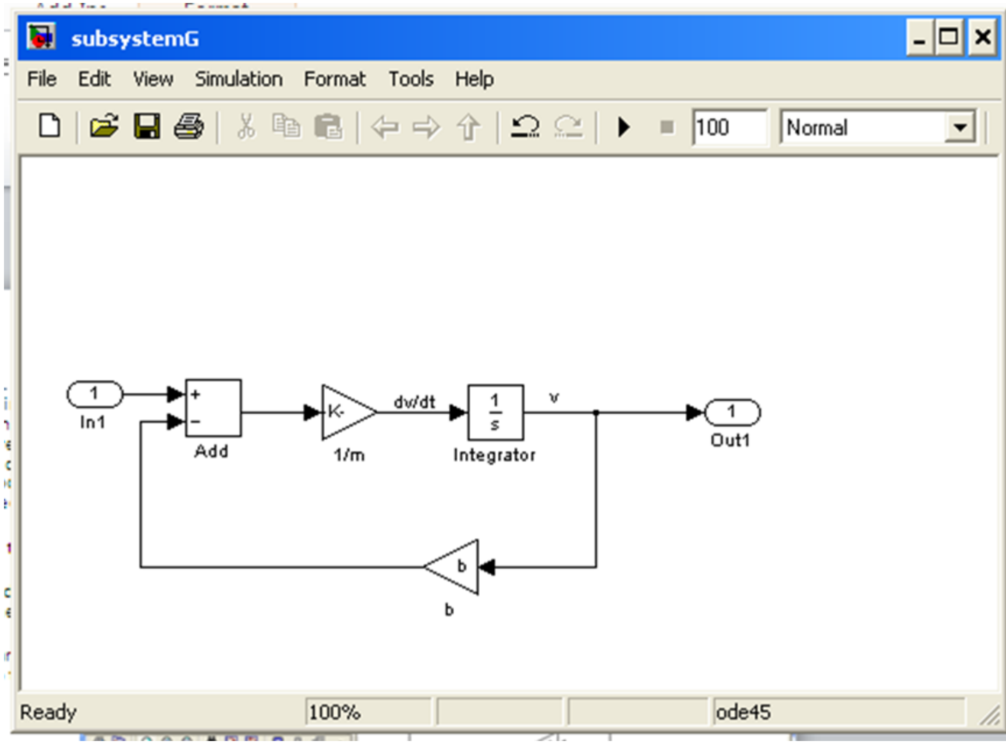
- At the MATLAB prompt set the parameters
 `>> f=500;b=50;m=1000;`
- Under the Simulations tab, click Configuration parameters, set the end time to 100 and click ok.
- Under the Simulations tab, click Start.
- Once the simulation is finished, double-click on each scope, and graphing windows appear (you probably need to autoscale)
- The time evolution is identical to the one we had obtained before.

cruiseG.mdl



Results

- SIMULINK is a time-domain simulation and handles linear and non-linear systems. The frequency-domain analysis presented so far can only handle linear models. To produce bode plots we need to linearize the model.
- Erase the two scopes and the step block.
- In the Sources section, add an *In* block and connect it to the add block.
- In the Sinks section, add an *Out* block and connect it to the output of the integrator.
- Save the model as “SubsystemG.mdl”
- At the MATLAB prompt type the commands on the right which reproduce the G transfer function we were using before



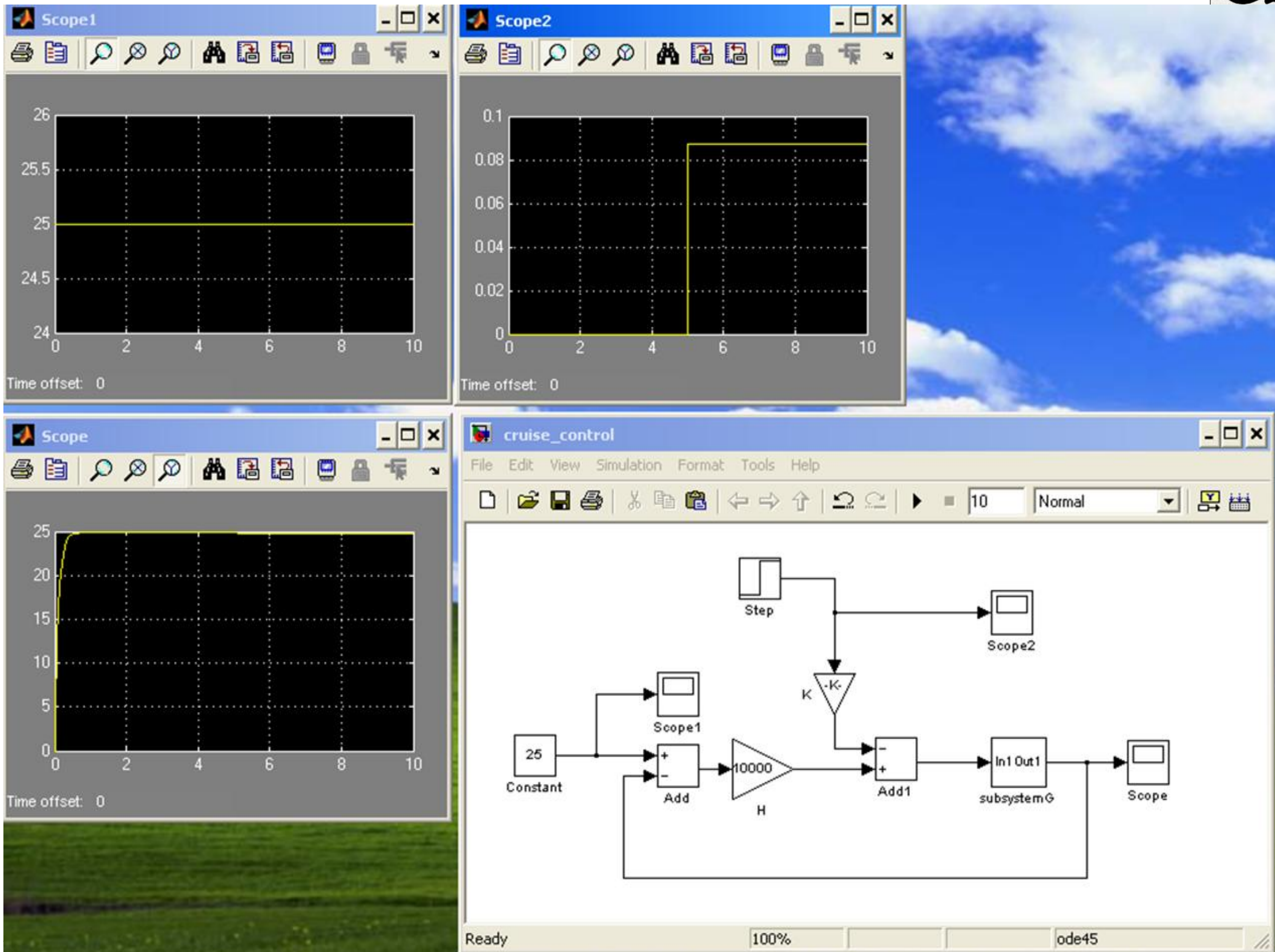

```
>> [A,B,C,D]=linmod('subsystemG');
>> [num,den]=ss2tf(A,B,C,D);
>> H=tf(num,den)
 Transfer function:
 0.001
--------
s + 0.05
```

Note: A, B, C, D is a state space representation of the system

Implementing feedback

- Create a new model calling it "cruise_control.mdl"
- Select and drag from the library the Ports & Subsystems → Subsystem block.
- Double-click on the subsystem block, erase the contents
- Double-click on the subsystemG.mdl model, select all and copy. Paste in the subsystem block.

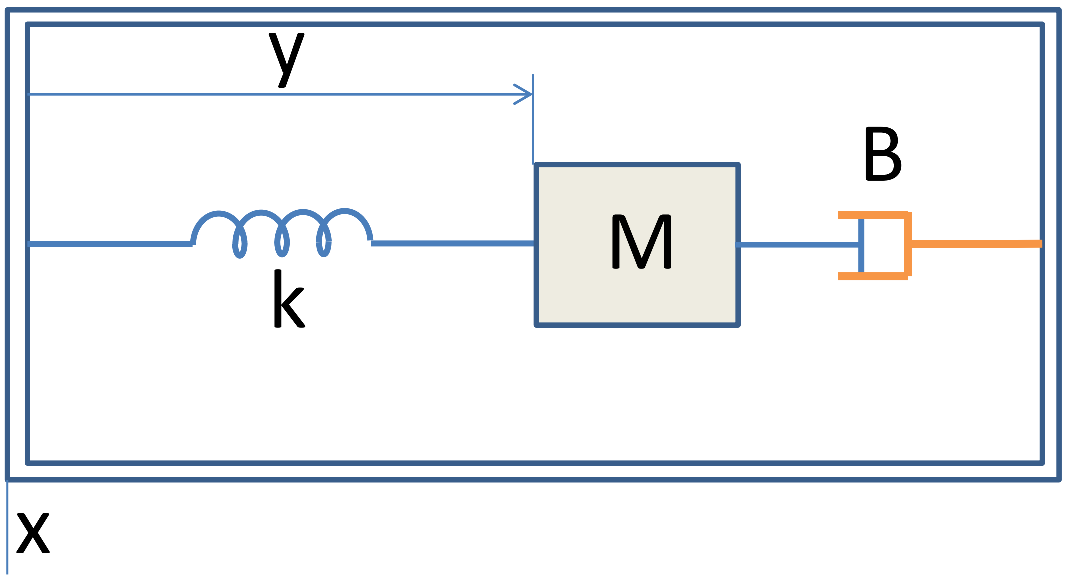
Implementing feedback



cruise_control.mdl

Let's setup a SIMULINK model of the accelerometer shown before

Position *y* is with respect of the case, the case's position is *x*. What is the transfer function between the input acceleration A $(a = d^2x/dt^2)$ and the output *y*?



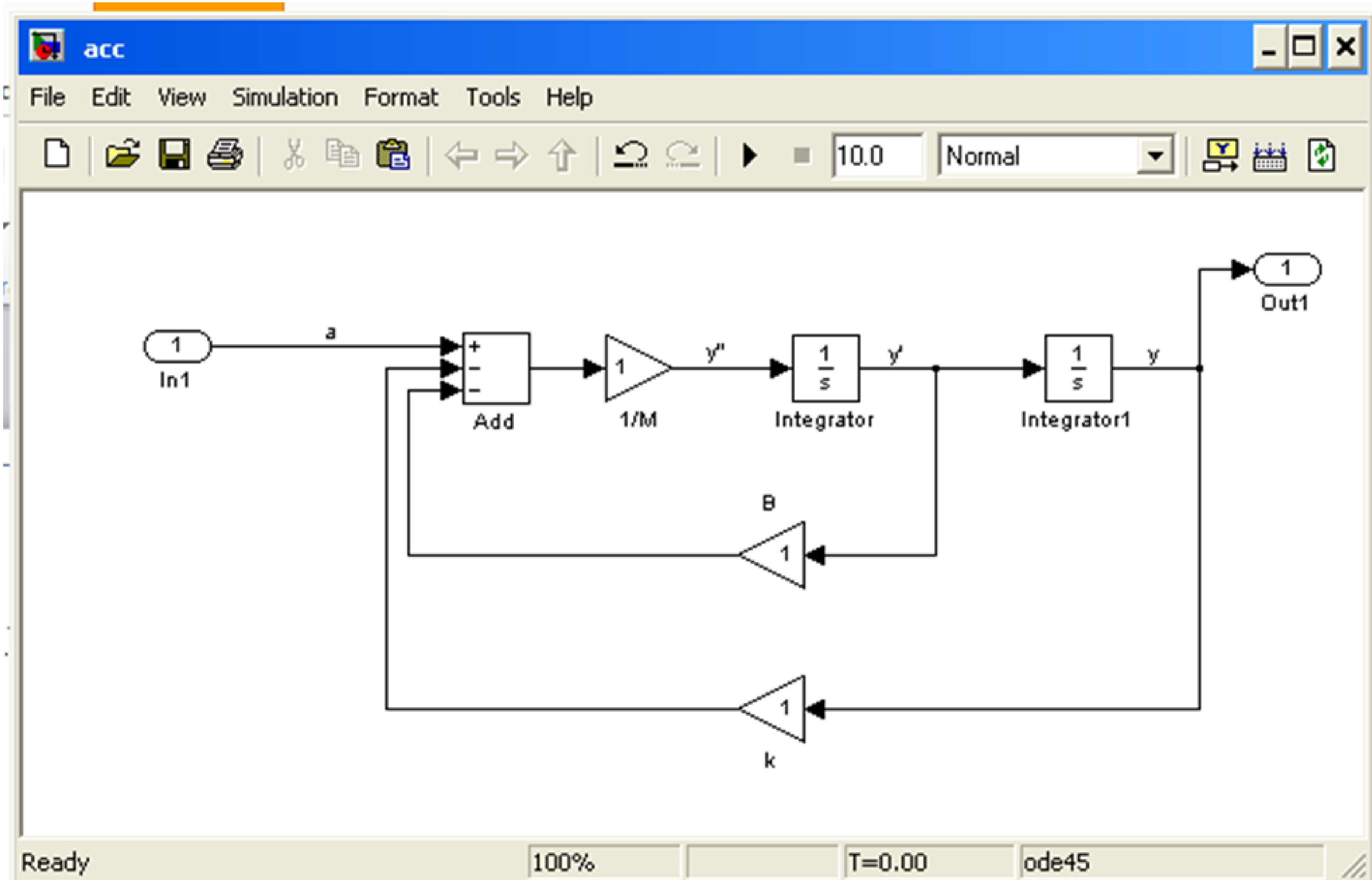
$$-B\frac{dy}{dt} - ky$$
$$= M\frac{d^2}{dt^2}(y - x)$$

Rule of thumb

- For a system represented by an nth order input/output ordinary differential equation it is necessary to integrate the highest derivative n times to obtain the output.
- Rearranging terms

$$\frac{d^2y}{dt^2} = \frac{1}{M}\left(-B\frac{dy}{dt} - ky + Ma\right)$$

The model



acc.mdl

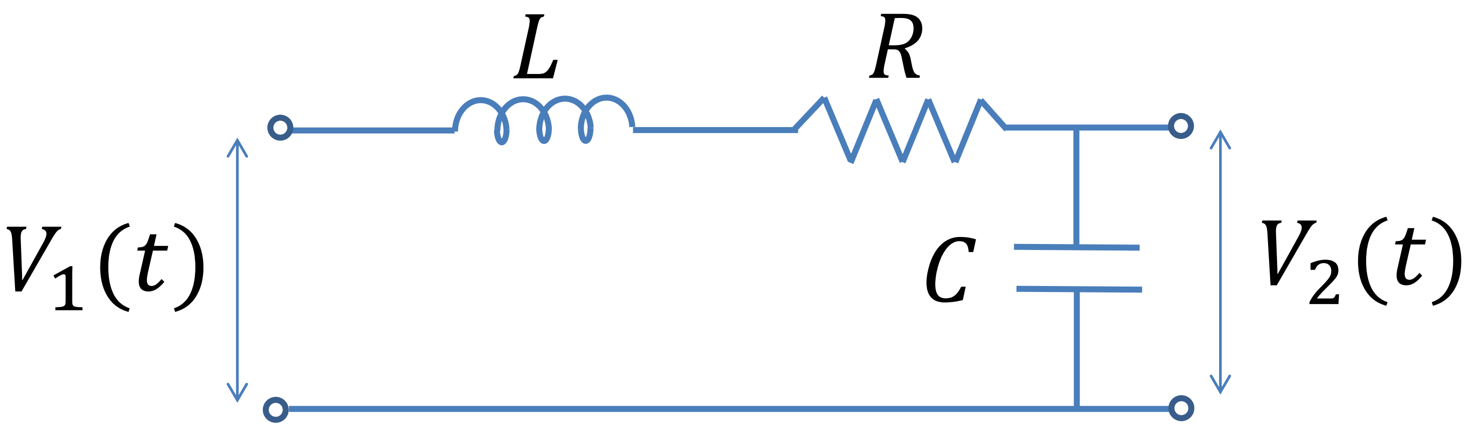
Results

```
>> m=1;k=1;B=1;
>> [A,B,C,D]=linmod('acc');
>> [num,den]=ss2tf(A,B,C,D);
>> H=tf(num,den)

Transfer function:
     1
-----------
s^2 + s + 1
```

Which confirms our previous finding.

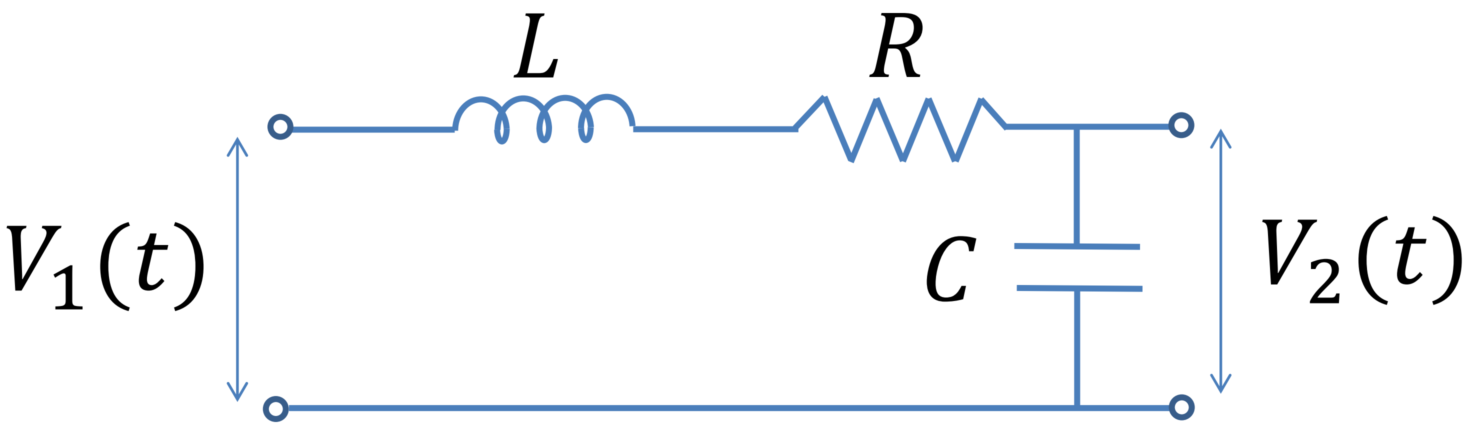
Model the following LRC circuit



$$V_1(t) = L\frac{d}{dt}i(t) + R\,i(t) + V_2(t)$$

$$i(t) = C\,\frac{dV_2(t)}{dt}$$

Model the following LRC circuit



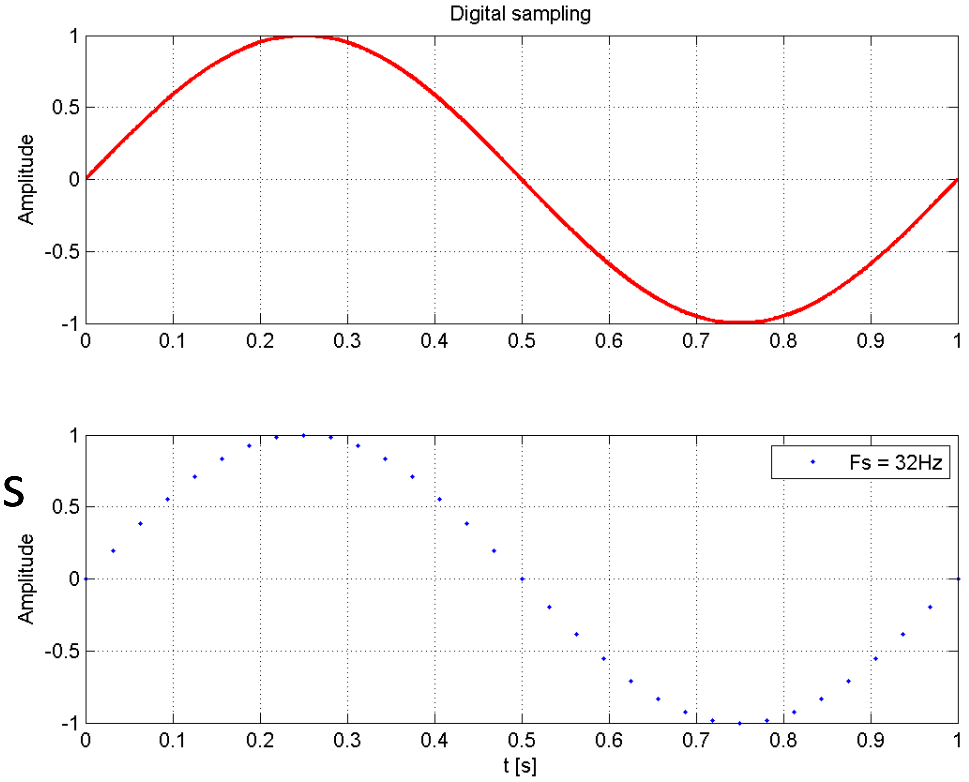
$$V_1(t) = L\frac{d}{dt}i(t) + R\,i(t) \ + V_2(t)$$

$$i(t) = C\,\frac{dV_2(t)}{dt}$$

$$\ddot{V}_2(t) = \frac{1}{LC}\left(V_1 - RC\,\dot{V}_2 - V_2\right)$$

Digital Signal Processing 1

- Moving to the digital world
- Two classes of signals
  - Analog
  - Discrete
- Analog signal
  - Denoted with $x(t)$
  - t represents time in seconds
- Discrete signal
  - Number sequence $x(n)$
  - n is an integer, represents discrete instances in time

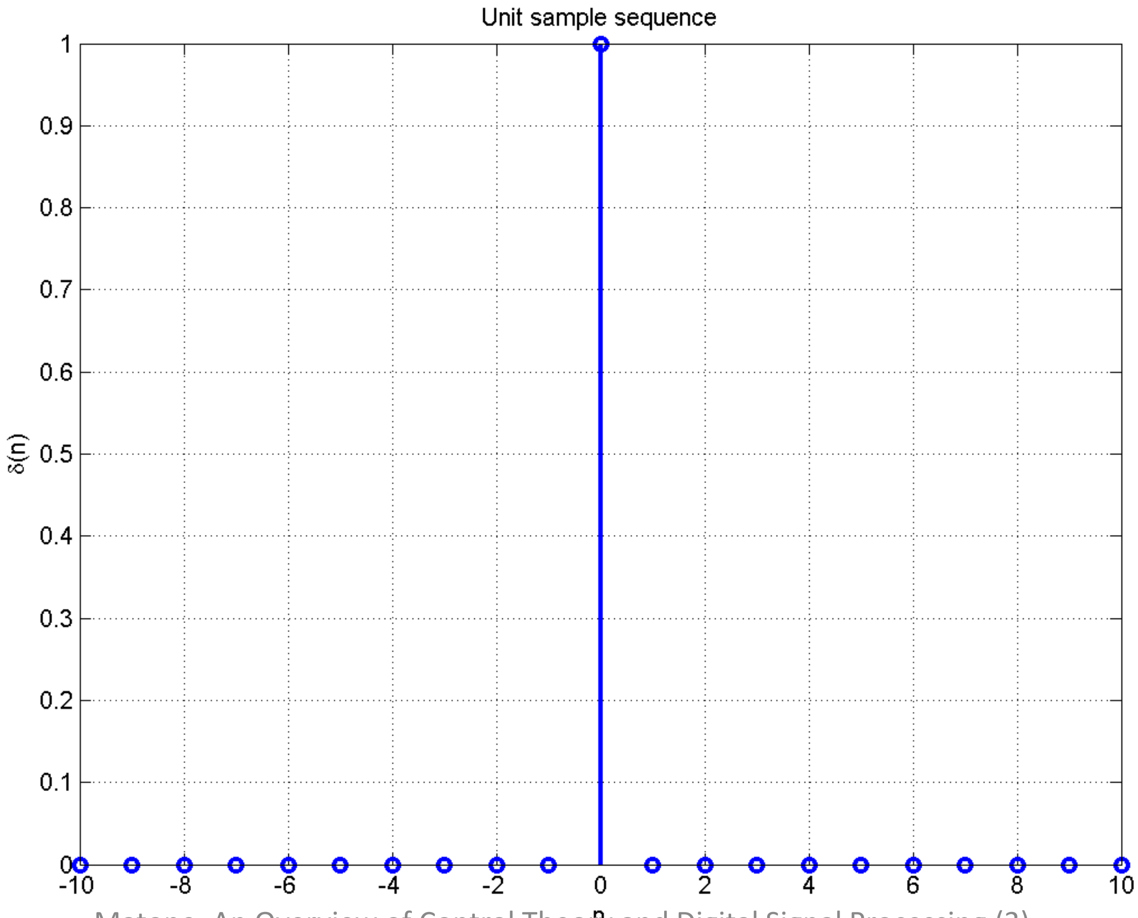


digsig.m

Types of sequences

Unit sample sequence

$$\delta(n-n_0)=\begin{cases}1, & n=n_0\\ 0, & n\neq n_0\end{cases}$$

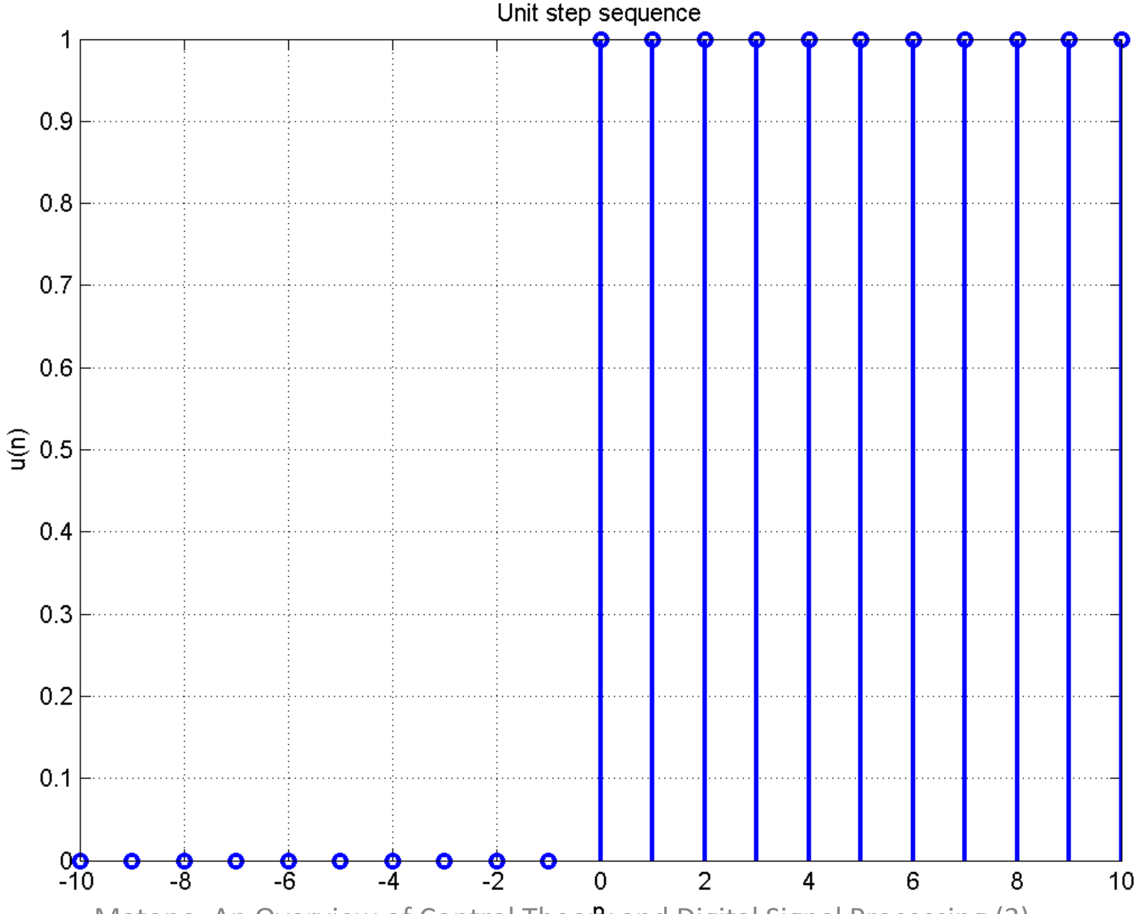


sequences.m

Types of sequences

Unit step sequence

$$u(n-n_0)=\begin{cases}1, & n \geq n_0\\ 0, & n < n_0\end{cases}$$



sequences.m

Discrete systems

- Linear time-invariant (LTI) system $\mathcal{L}$

$$y(n) = \mathcal{L}[x(n)]$$

  - Satisfies the principle of superposition

$$\mathcal{L}[a_1 x_1(n) + a_2 x_2(n)] =$$
$$= a_1 \mathcal{L}[x_1(n)] + a_2 \mathcal{L}[x_2(n)]$$

  - The input-output pair, $x(n)$ and $y(n)$, is invariant to a shift k

$$y(n-k) = \mathcal{L}[x(n-k)]$$

Discrete systems

Any sequence $x(n)$ can be written in terms of scaled and delayed unit sample sequences

$$x(n) = \sum_{k=-\infty}^{\infty} x(k)\,\delta(n-k)$$

The response of an (LTI) system can then be re-written as

$$y(n) = \mathcal{L}[x(n)] = \mathcal{L}\left[\sum_{k=-\infty}^{\infty} x(k)\,\delta(n-k)\right]$$

$$= \sum_{k=-\infty}^{\infty} x(k)\,\mathcal{L}[\delta(n-k)]$$

Discrete systems

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)\, \mathcal{L}[\delta(n-k)]$$

$$= \sum_{k=-\infty}^{\infty} x(k)\ h(n-k)$$

- $h(n)$ is the *unit sample* or *impulse* response of LTI system
- Convolution

$$\boxed{y(n) = x(n) \star h(n) = \sum_{k=-\infty}^{\infty} x(k)\ h(n-k)}$$

Let's compute the convolution between two sequences

k	x(k)	y(k)	-k	y(-k)	x(k)y(-k)
-4	-	-	4	-	-
-3	-	-	3	-	-
-2	-	-	2	1	-
-1	-	-	1	2	-
0	1	3	0	3	3
1	2	2	-1	-	-
2	3	1	-2	-	-
3	-	-	-3	-	-
4	-	-	-4	-	-
5	-	-	-5	-	-
					3

Let's define $x(n) = \{1,2,3\}$ and $y(n) = \{3,2,1\}$ and let's determine $z(n) = x(n) \star y(n)$

For $n = 0$

$$z(0) = \sum_{k=-\infty}^{+\infty} x(k)y(-k)$$
$$= 3$$

Let's compute the convolution between two sequences

k	x(k)	y(k)	1-k	y(1-k)	x(k)y(1-k)
-4	-	-	5	-	-
-3	-	-	4	-	-
-2	-	-	3	-	-
-1	-	-	2	1	-
0	1	3	1	2	2
1	2	2	0	3	6
2	3	1	-1	-	-
3	-	-	-2	-	-
4	-	-	-3	-	-
5	-	-	-4	-	-
					8

Let's define $x(n) = \{1,2,3\}$ and $y(n) = \{3,2,1\}$ and let's determine $z(n) = x(n) \star y(n)$

For $n = 1$

$$z(1) = \sum_{k=-\infty}^{+\infty} x(k)y(1-k) = 8$$

Let's compute the convolution between two sequences

n	$z(n) = x(n) \star y(n)$
0	3
1	8
2	14
3	8
4	3

Let's define $x(n) = \{1,2,3\}$ and $y(n) = \{3,2,1\}$ and let's determine $z(n) = x(n) \star y(n)$

Using MATLAB to confirm results

```
>> conv([3 2 1],[1 2 3])
ans =
     3     8    14     8     3
```

Stability

An LTI system $\mathcal{L}$ is stable if and only if its impulse response is absolutely summable

$$\sum_{-\infty}^{+\infty} |h(n)| < \infty$$

Correlation of sequences

The correlation between two sequences $x(n)$ and $y(n)$ is defined as

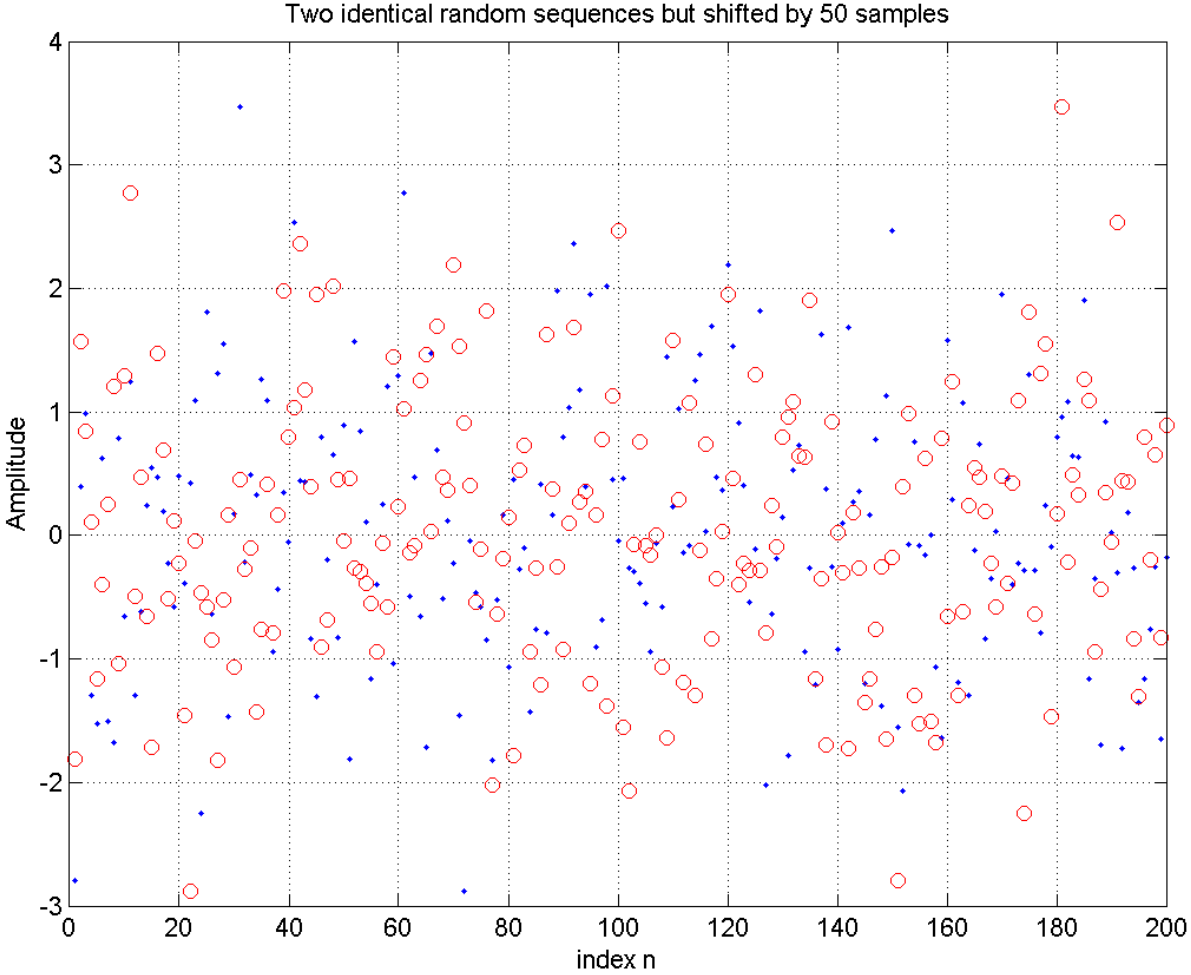
$$r_{x,y}(l) = \sum_{n=-\infty}^{\infty} x(n)y(n-l) = x(l) \star y(-l)$$

where

- r is the correlation (degree to which the two signals are similar) and
- l is the lag.

Example

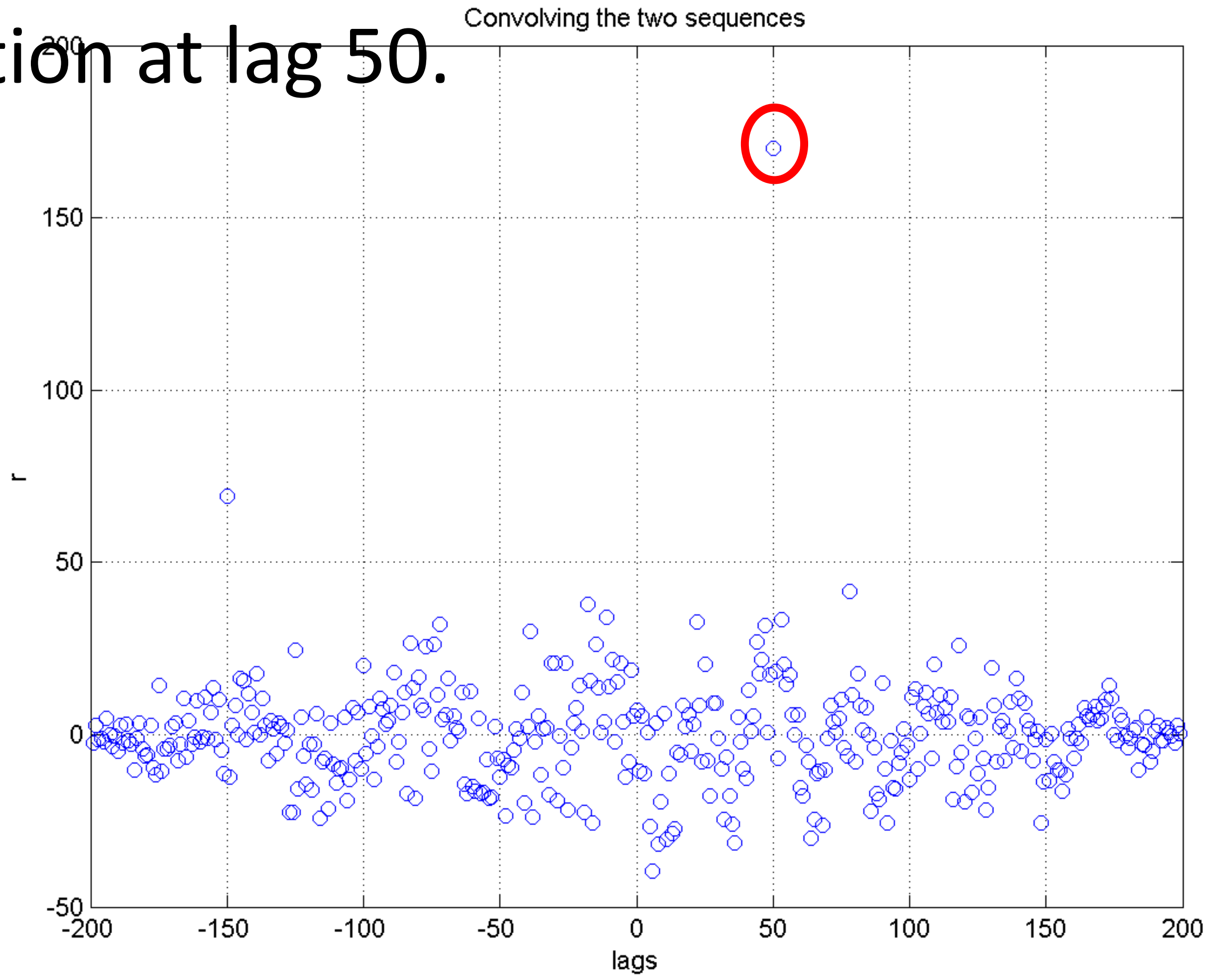
Generating two identical random sequences, one of them shifted by 50 samples.



conv_example2.m

Example

Taking the correlation between them using MATLAB's `xcorr` command we find highest correlation at lag 50.



conv_example2.m

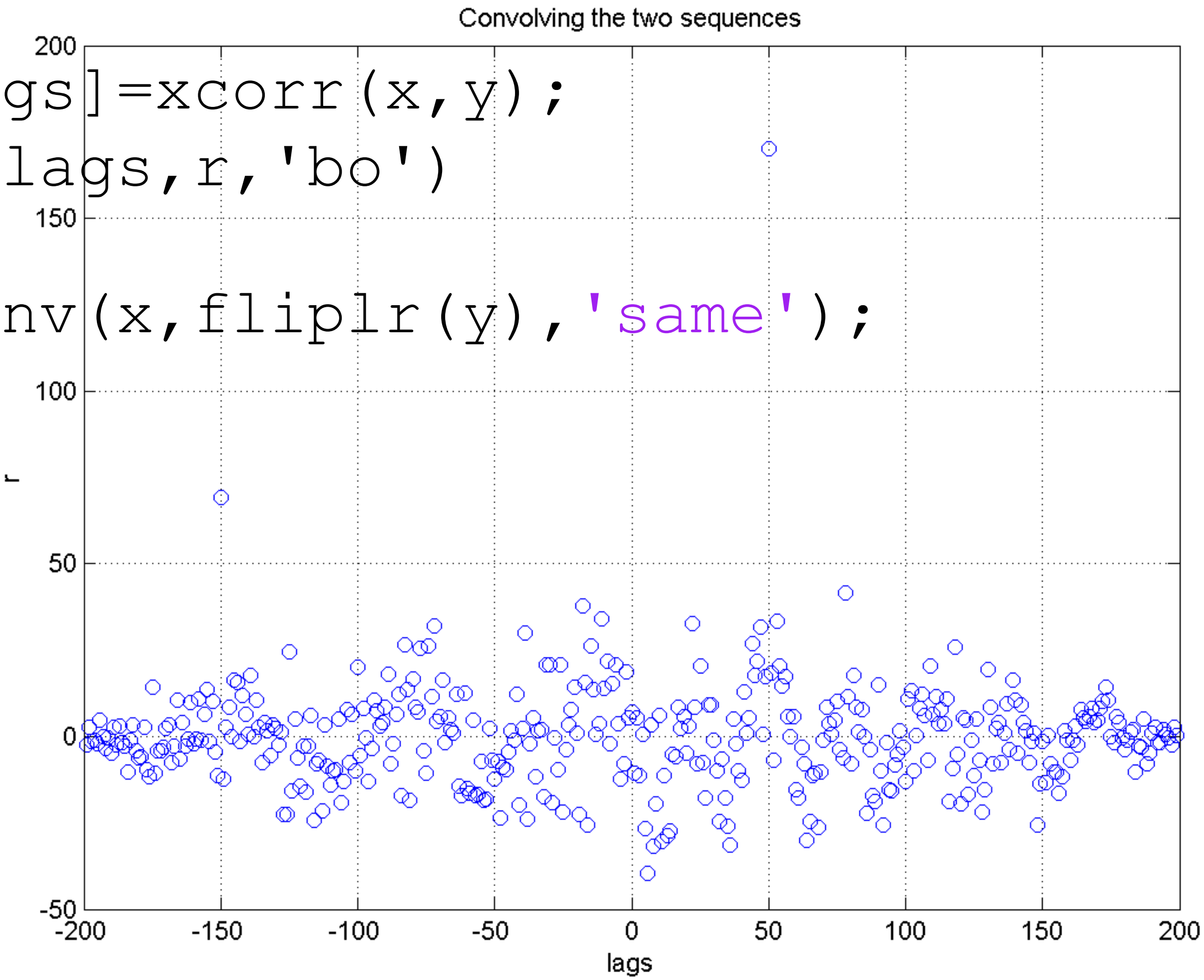
Example

Using MATLAB's `xcorr` or `conv` command.

```
>> [r,lags]=xcorr(x,y);
>> plot(lags,r,'bo')
```

or

```
>> r2=conv(x,fliplr(y),'same');
```



conv_example2.m

Exercise

In a certain concert hall, echoes of the original audio signal $x(n)$ are generated due to reflections at the walls and ceiling. The audio signal experienced by the listener $y(n)$ is a combination of $x(n)$ and its echoes. Let

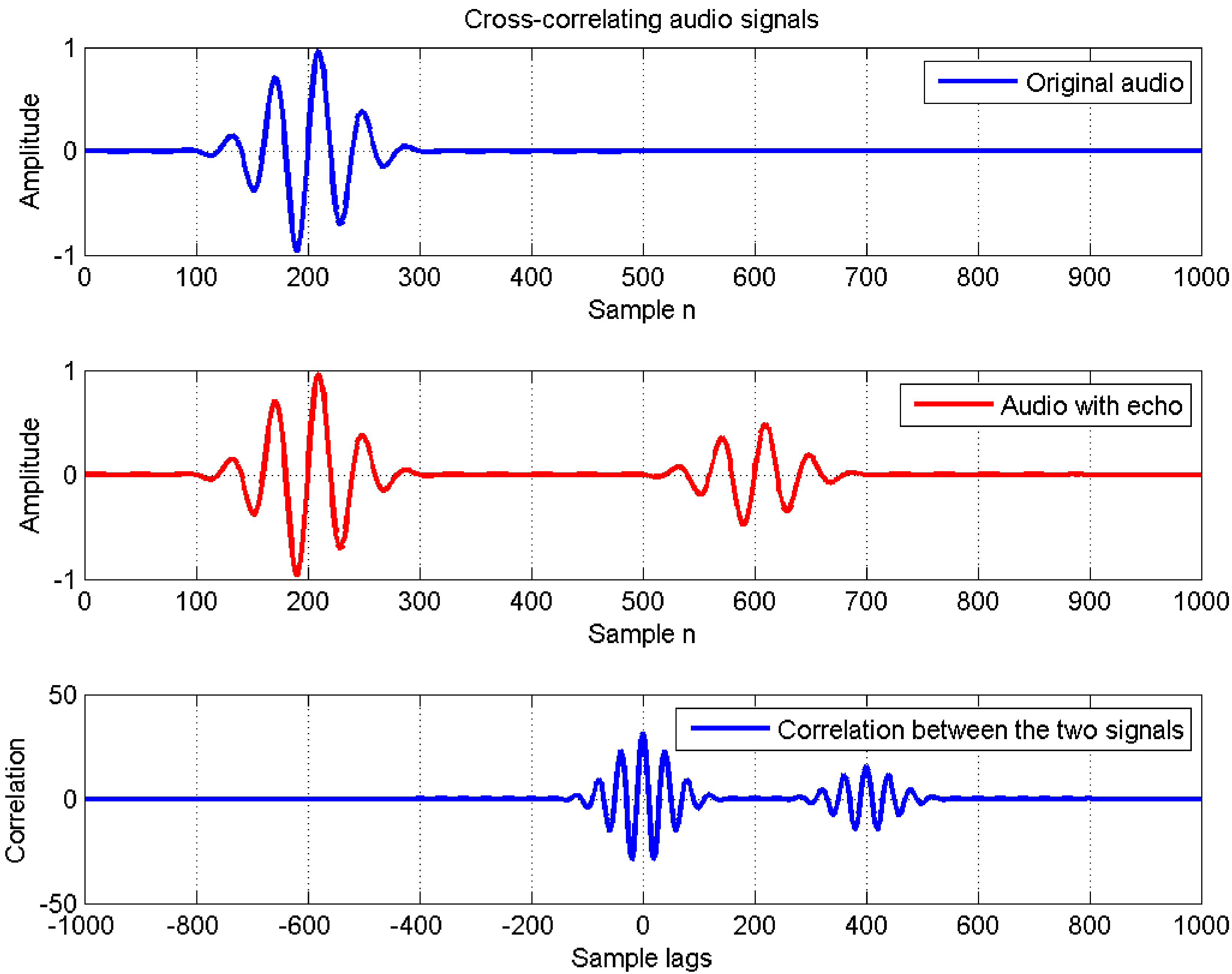
$$y(n) = x(n) + \propto x(n-k)$$

where k is the amount of delay in samples and $\propto$ is its relative strength.

Estimate the delay k assuming the original signal is a Sine-Gaussian

$$x(n) = \sin(0.05\,\pi\,n)\,e^{-\frac{(n-n_0)^2}{\tau^2}}$$

with $n_0 = 200$, $\tau = 50$ and $\propto = 50\%$.


Cross-correlating audio signals
Original audio
Amplitude
Sample n
Audio with echo
Amplitude
Sample n
Correlation between the two signals
Correlation
Sample lags

ex28.m

Signal Energy

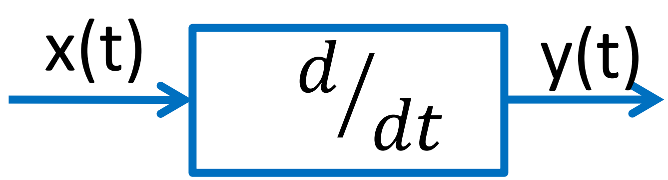
The energy of a sequence $x(n)$ is given by

$$\mathcal{E}_x = \sum_{-\infty}^{\infty} |x(n)|^2$$

Differential to *difference* equations

In the analog world:

$$y(t) = \frac{dx(t)}{dt}$$



Differential to *difference* equations

In the limit $F_s \to \infty$

- In the digital world:

$$y(t) = \frac{dx(t)}{dt} = \lim_{\Delta t \to 0} \frac{x(t) - x(t - \Delta t)}{\Delta t}$$

$$\cong \frac{x(t) - x(t - \Delta t)}{\Delta t}$$

$$\boxed{y(n) = x(n) - x(n-1) \qquad (\Delta t = 1)}$$

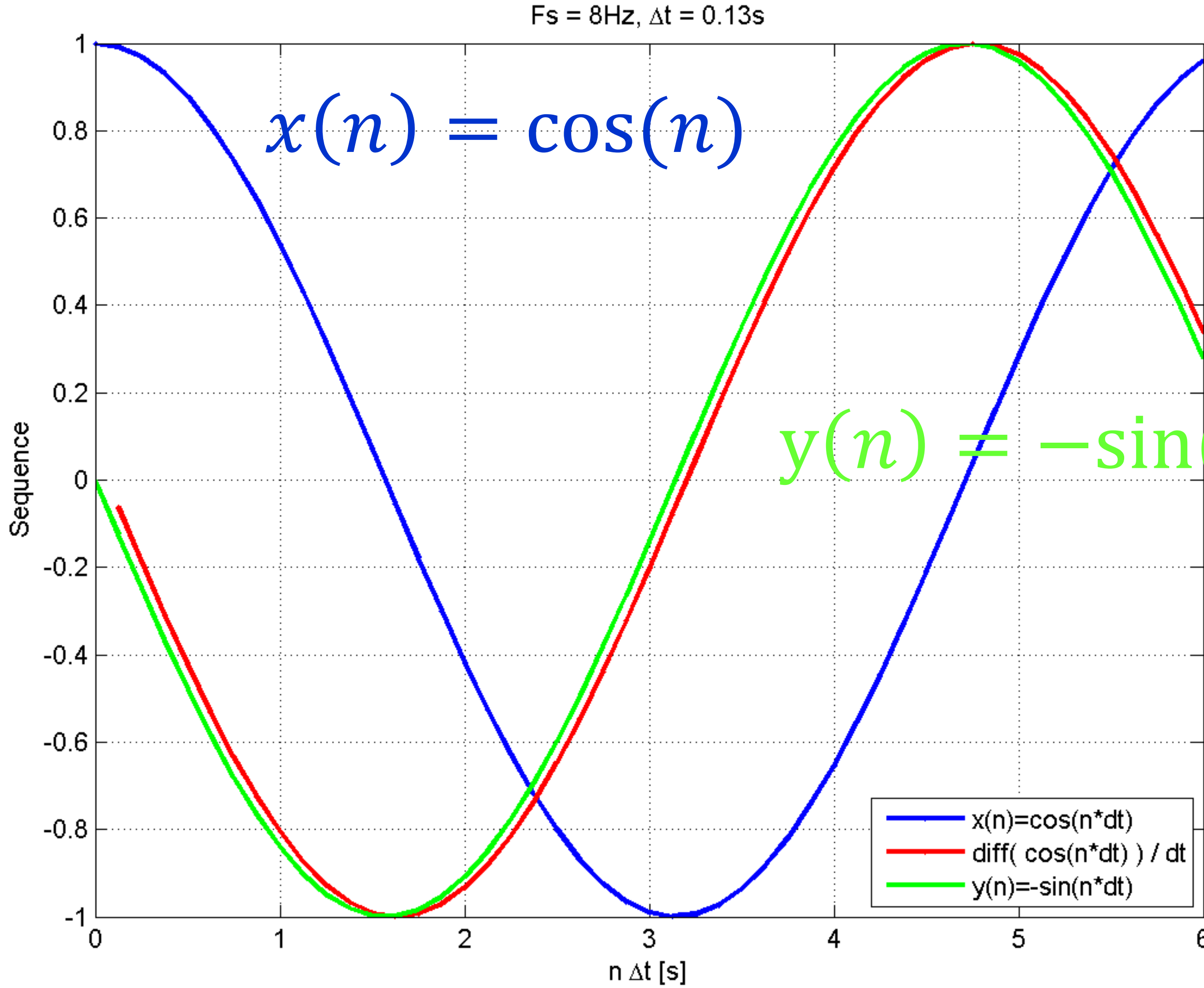
The original *differential* equation is approximated by a *difference* equation

$$y(t) = \frac{dx(t)}{dt}$$

```
>> diff(x)
```

$$y(n) = x(n) - x(n-1)$$

In the limit $F_s \to \infty$



$$x(n) = \cos(n)$$

$$y(n) = -\sin(n)$$

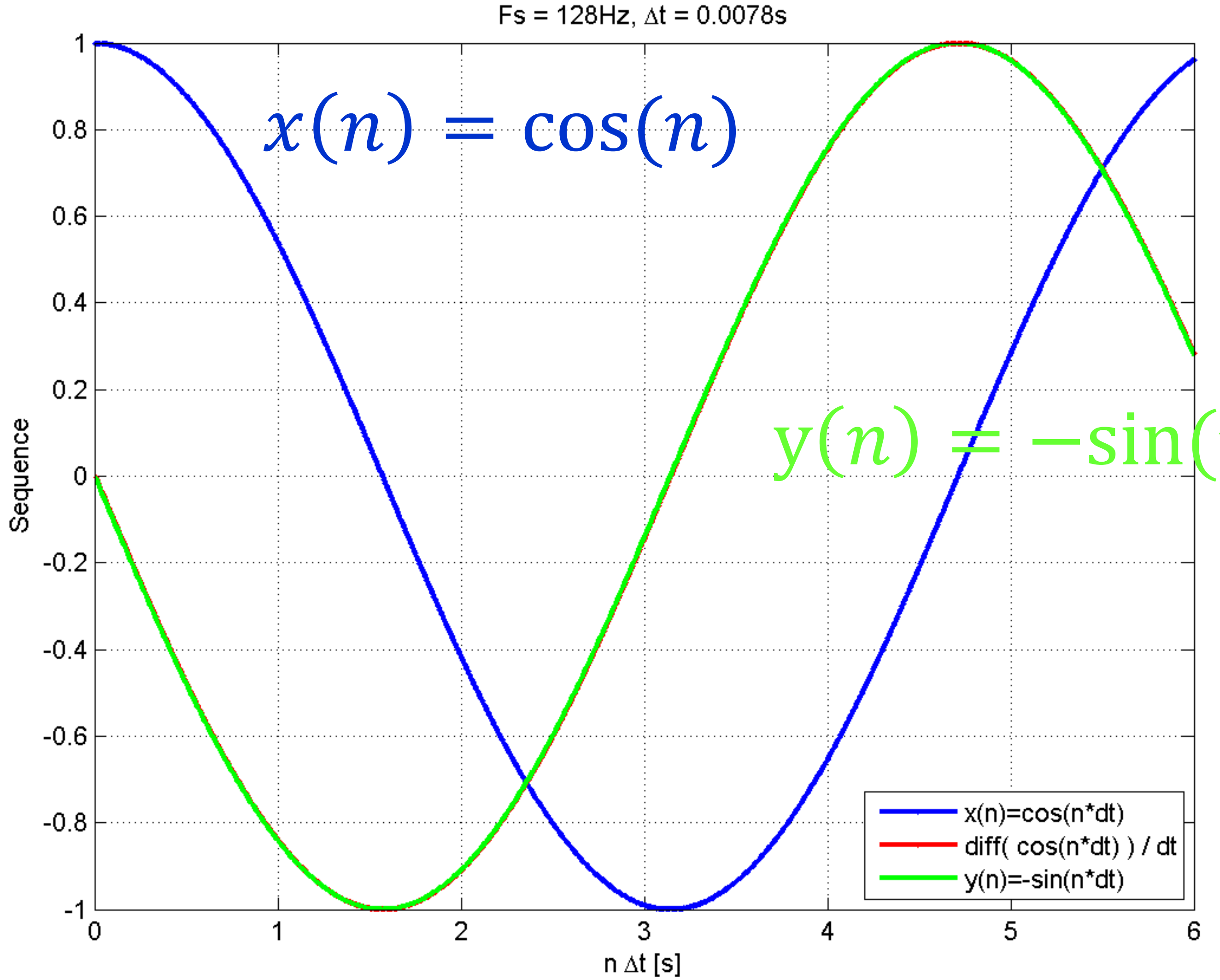
difference.m

```
>> diff(x)
```

$$y(t) = \frac{dx(t)}{dt}$$

$$y(n) = x(n) - x(n-1)$$

In the limit $F_s \to \infty$



$$x(n) = \cos(n)$$

$$y(n) = -\sin(n)$$

difference.m

Difference equations

- In general, a difference equation is of the form

$$\sum_{k=0}^{N} a_k\, y(n-k) = \sum_{m=0}^{M} b_m x(n-m)$$

  – where $x(n)$ is the input sequence
  – $y(n)$ is the output sequence and
  – a_k and b_m are the coefficients of $y(n)$ and $x(n)$ respectively.
- The MATLAB `filter` command solves the difference equations numerically – *filtering* input sequence $x(n)$

```
>> y = filter(b, a, x)
```

Impulse response $h(n)$

To generate the impulse response of an LTI system described by the difference equation

$$\sum_{k=0}^{N} a_k\, y(n-k) = \sum_{m=0}^{M} b_m x(n-m)$$

use the `filter` command:

```
>> h = filter(b, a, delta)
```

By plotting h you can visualize if the system is stable

Example

An LTI system is described by the following difference equation

$$y(n) - y(n-1) + 0.9\, y(n-2) = x(n)$$

Plot the impulse response h for $0 \leq n \leq 100$ and determine if system is stable.

Example

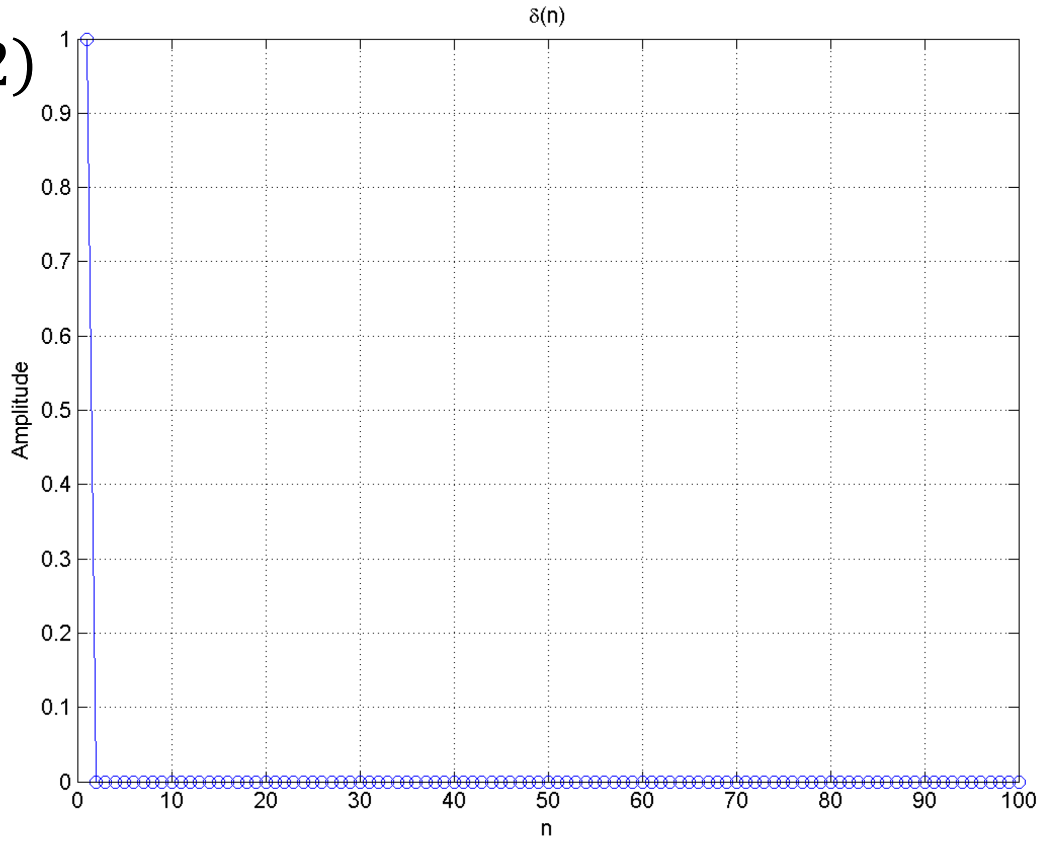
An LTI system is described by the following difference equation

$$y(n) - y(n-1) + 0.9\,y(n-2) = x(n)$$

Plot the impulse response h for $0 \leq n \leq 100$ and determine if system is stable.

Generating the delta function with MATLAB

```
>> x = zeros(1,100);
>> x(1) = 1;
>> plot(x,'bo-')
```



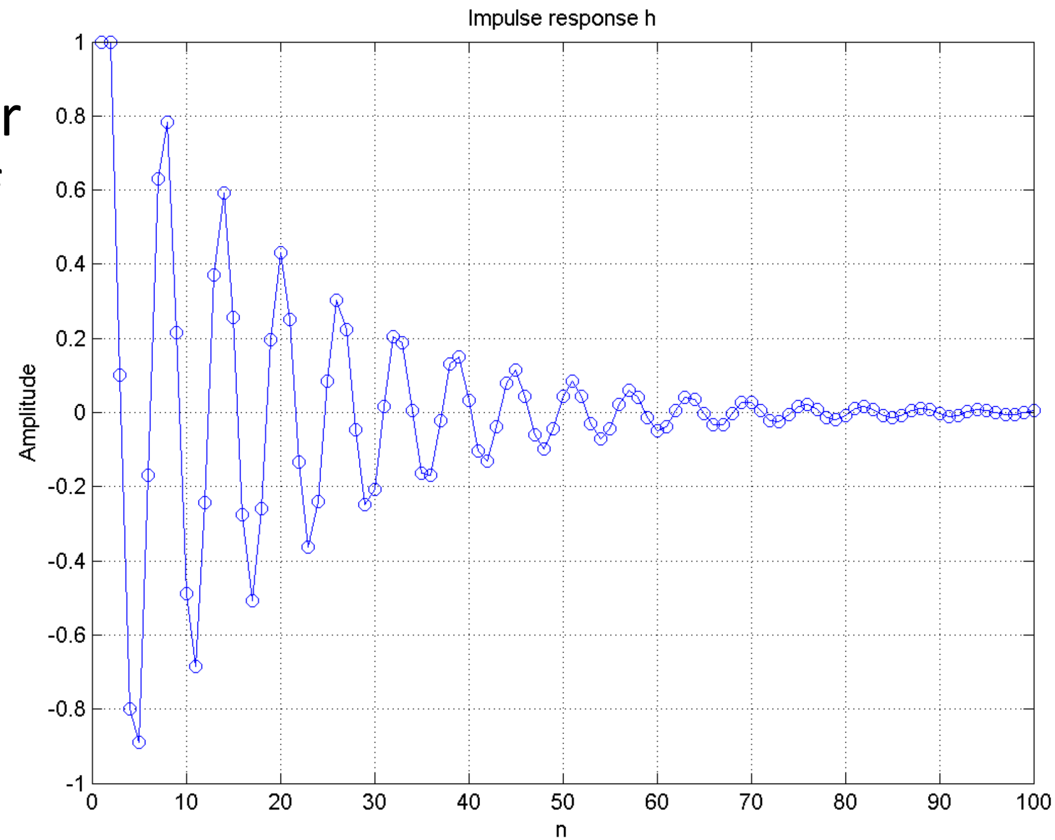
ex29.m

Example

An LTI system is described by the following difference equation

$$y(n) - y(n-1) + 0.9\,y(n-2) = x(n)$$

Plot the impulse response h for $0 \leq n \leq 100$ and determine if system is stable.



Generating the response

```
>> b=1;
>> a=[1 -1 0.9];
>> h=filter(b,a,x);
>> plot(h,'bo-')
```

System is stable

ex29.m

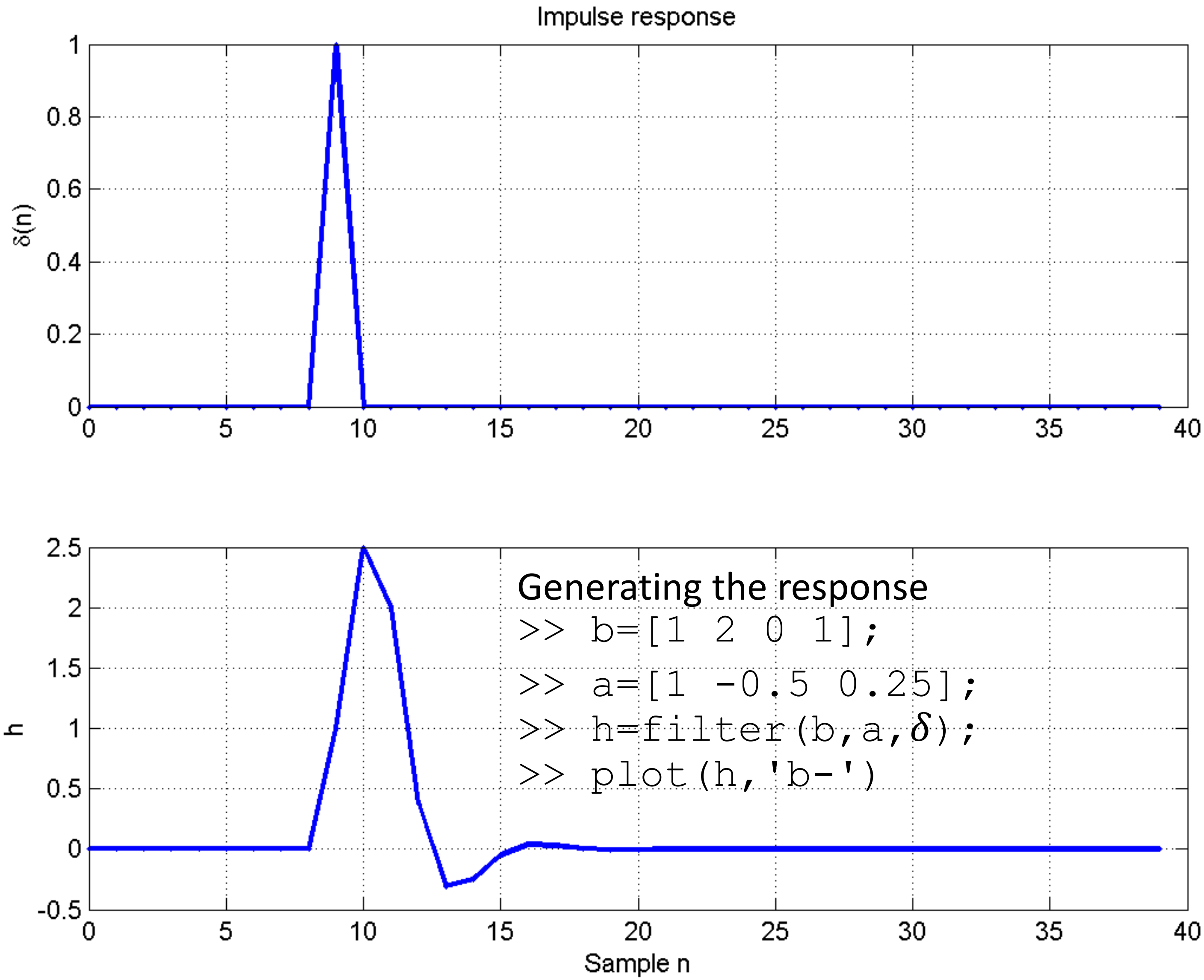
Exercise

An LTI system is described by the difference equation

$$y(n) - 0.5\, y(n-1) + 0.25\, y(n-2) \\ = x(n) + 2\, x(n-1) + x(n-3)$$

Plot its impulse response h and determine the stability of the system.

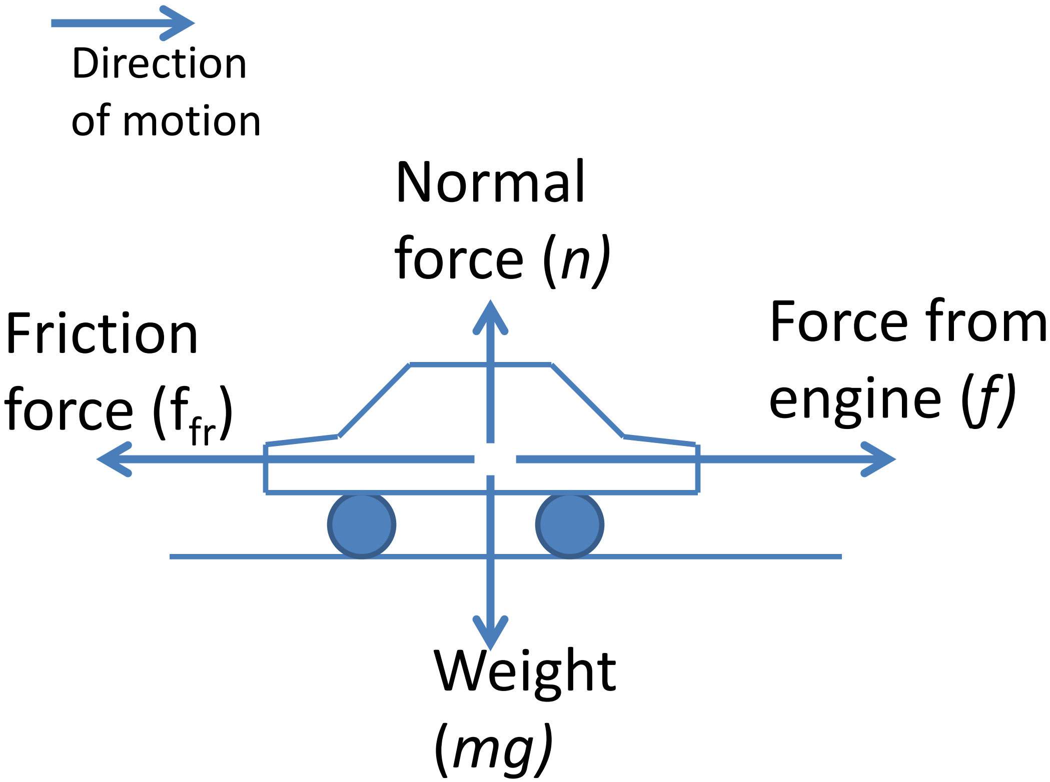
ex216.m



Generating the response

```
>> b=[1 2 0 1];
>> a=[1 -0.5 0.25];
>> h=filter(b,a,δ);
>> plot(h,'b-')
```

Cruise control case



- Physical system described by the following equation of motion

$$\sum_{road} f = f - f_{fr} = ma$$

- Simplifying and assuming friction force is proportional to speed

$$\boxed{m\frac{dv}{dt} = f - bv}$$

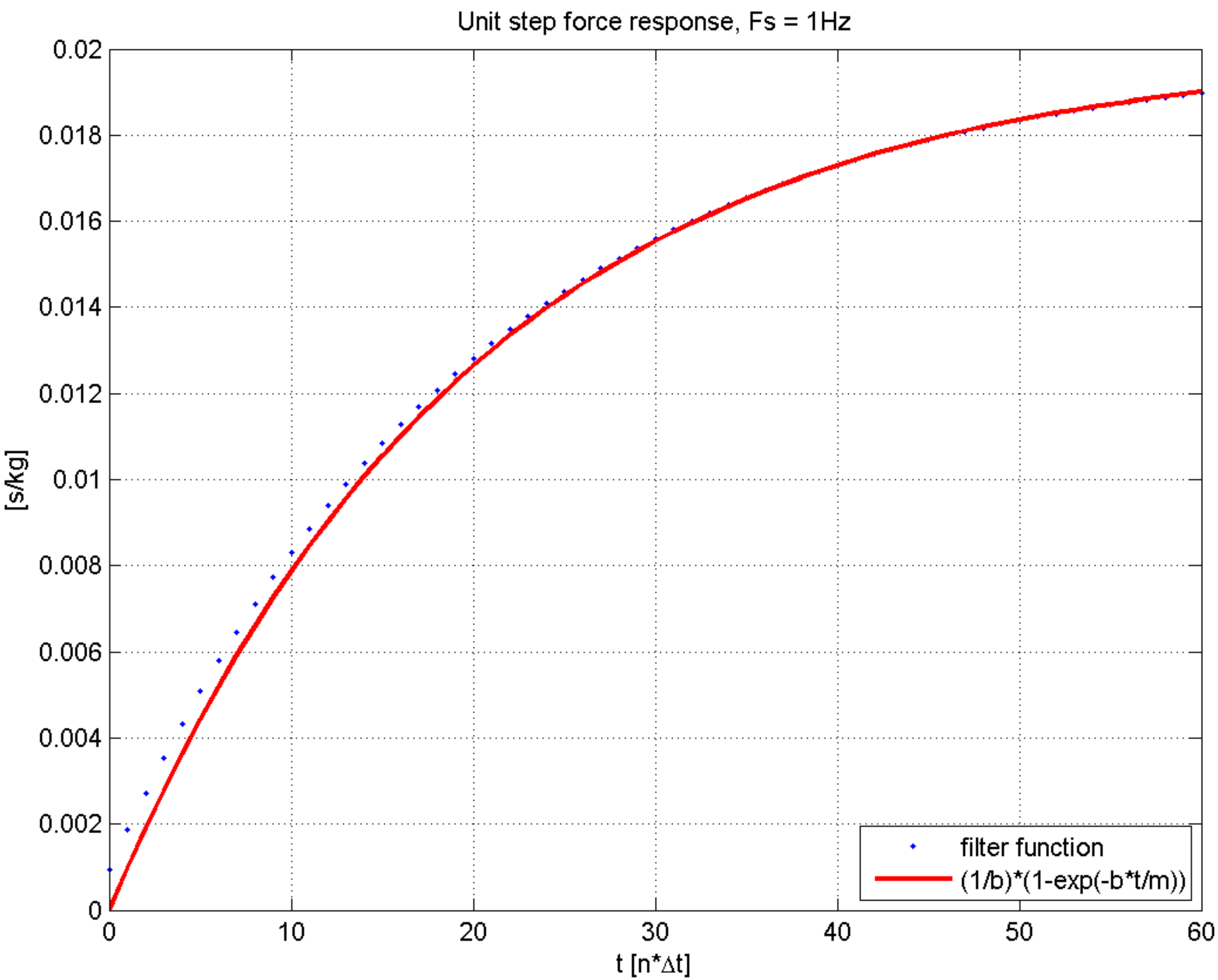
In the digital world

$$m\frac{dy(t)}{dt} = x(t) - by(t)$$

$$1050\ y(n) - 1000\ y(n-1) = x(n) \qquad (\Delta t = 1)$$

Numerical solution

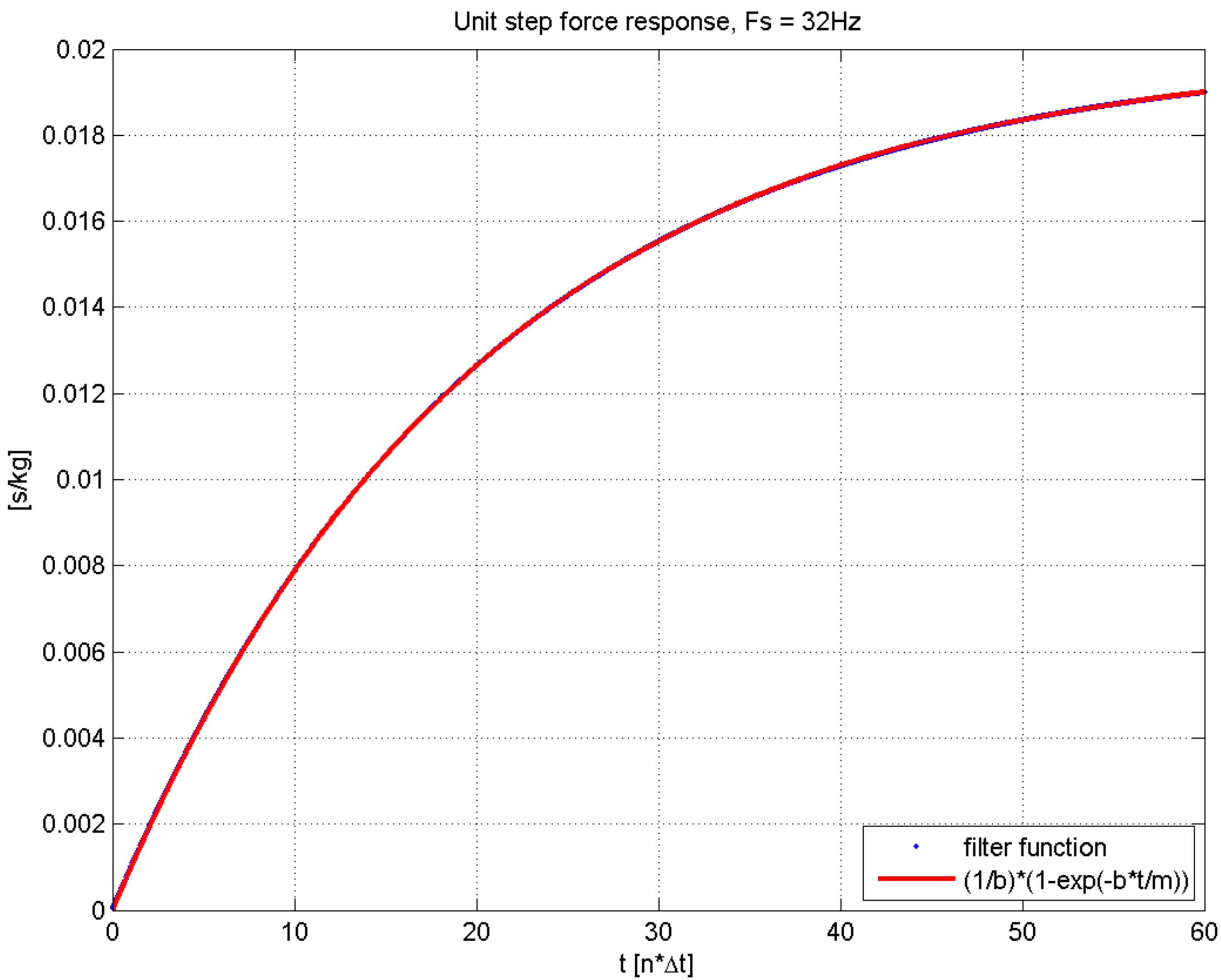
In the limit $F_s \to \infty$



difference2.m

Numerical solution

In the limit $F_s \to \infty$



difference2.m

Summary

- SIMULINK – time domain simulation, can handle non-linear systems.
  - Tutorial
  - A suggestion for re-writing the differential equation is given in order to facilitates designing the model
- Analog to digital
  - Signals to sequences
  - Impulse response of a system h

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)\, \mathcal{L}[\delta(n-k)] = \sum_{k=-\infty}^{\infty} x(k)\;\, h(n-k)$$

  - Convolution and correlation

$$z(n) = x(n) \star y(n)$$
$$r_{x,y}(l) = x(l) \star y(-l)$$

  - Condition of stability

$$\sum_{-\infty}^{+\infty} |h(n)| < \infty$$

  - Differential to difference equations
  - General form of a difference equation
  - MATLAB's `filter` to numerically solve difference equations
  - Impulse response can be determined by using the filter command and setting the input to a delta sequence.