EE 1130 Freshman Eng. Design for Electrical and Computer Eng. Class 2

Signal Processing Module (DSP).

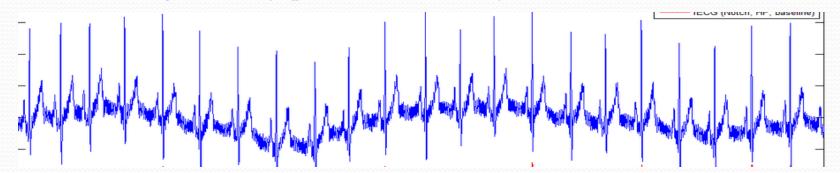
• Matlab and Simulink.

Signal Processing Engineer

- We are detectives of hidden information in signals.
 - Communication signals: obtain the information.
 - Signals from CO2 sensors, heat sensors, etc.
 - Data from Hard Drives.
 - Data from Computers to computers.
- In order to do that we need to master MATHEMATICS!!

Problem Statement

- Imagine you are an Electrical Engineer Signal Processing proficient. A medical doctor running a Hospital building hires you to solve the following problem:
 - Doc: "Our EKG readings show a garbled signal and I can not diagnose my patients correctly".

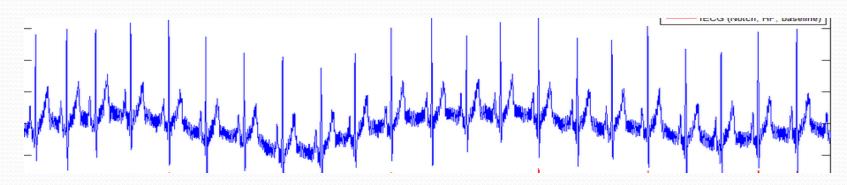


• He is expecting this kind of signal:



Problem Statement

• What is the difference?.



- The above signal has and ondulation (low frequency envelope)
- Besides that, it is noisy (has a lot of ripples or high frequency component)



Matlab

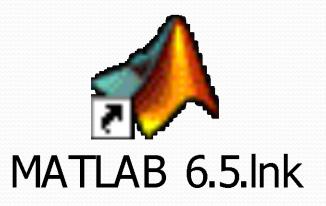
- Matlab is a powerful tool for mathematical/engineering research and development. It is also useful to students to easily computate or solve almost all mathematical and engineering problems.
- This Signal Processing Module will use Matlab as a development and teaching tool.
- Matlab is learned in the course **EE 3220 Software Applications** in Electric Engineering.

Matlab

- We will use the **Simulink** part of Matlab for two main purposes inside this DSP Module:
 - We will emulate/modelate a low frequency signal corrupted with an additive noise (high frequency signal).
 - Design and implement a FILTER that will eliminate a high frequency component (ripple or noise) meanwhile leaving untouched a low frequency sine wave.
- This phenomenon is common in any electrical system, where the 60Hz signal from the power lines corrupt a signal of interest as an Electro EncephaloGram (EEG), Electro CardioGram (EKG), or just a sinewave.



• To run Matlab, just double click on the matlab icon.



Matlab

• The Matlab window application will appear. We can see different sub-windows:

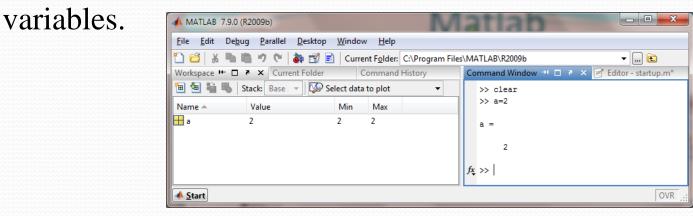
A MATLAB 7.9.0 (R2009b)		- • ×
<u>F</u> ile <u>E</u> dit De <u>b</u> ug <u>P</u> arallel <u>D</u> esktop <u>W</u> ir	ndow <u>H</u> elp	
🗋 🗃 👗 🖿 🛍 🦛 😫 📄 🕻	urrent Folder: C:\Program Files\MATLAB\R2009b	 €
Workspace I← □ ₹ × Current Folder Cc►	Command Window → □ ? × 💽 Editor - startup.m*	
🛅 🗃 📲 🧠 Stack: Base 👻 🕼	<i>fx</i> ; >>	
Name 🔺 Value M		
exc <1x1 MException>		
▲ <u>S</u> tart		OVR .::



• **Command window**: is the one where the user writes the variables and where the results are displayed

MATLAB 7.9.0 (R2009b)	
<u>F</u> ile <u>E</u> dit De <u>b</u> ug <u>P</u> arallel <u>D</u> esktop <u>W</u> indow <u>H</u> elp	
🛅 🗃 👗 ங 🛍 🤊 🍽 鷸 🗊 🖹 🛛 Current F <u>o</u> lder: C:\Program Files\MATLAB\R2009b	🖻
Workspace I → □ ₹ × Current Folder Corr Command Window → □ ₹ × 🖻 Editor - startup.m*	
1 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	
Name A Value N Value N	
<u>▲ Start</u>	OVR .::

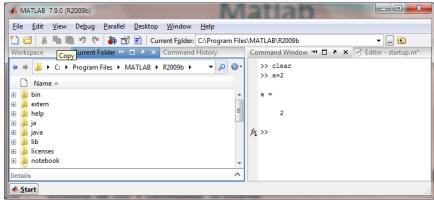
• Workspace window: is the one where the user can see the



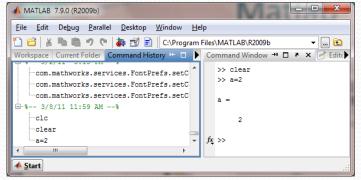


Matlab

• **Current Folder window**: is the one where the user can see the different files stored in the current directory. The current directory is the one that Matlab uses to store and read files.

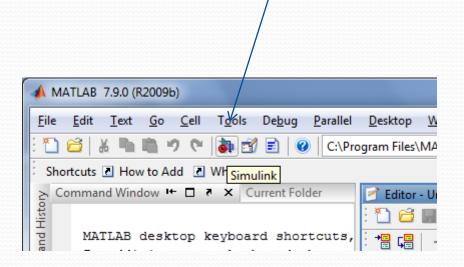


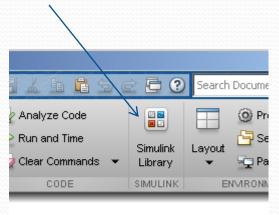
• **Command History window** that shows us all the commands written in the Command Window.



Simulink

- Simulink is part of Matlab. Simulink works by interconecting blocks. Each block is in fact a matlab function with input and output parameters.
- To open simulink click on the Simulink icon as shown:

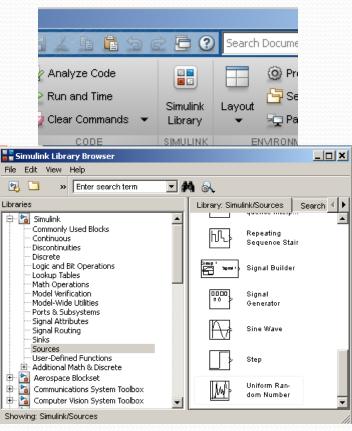




Simulink

 A new window appears, the library Browser: Notice Continuous, Math Operations, Source, and Sink groups that we will be using

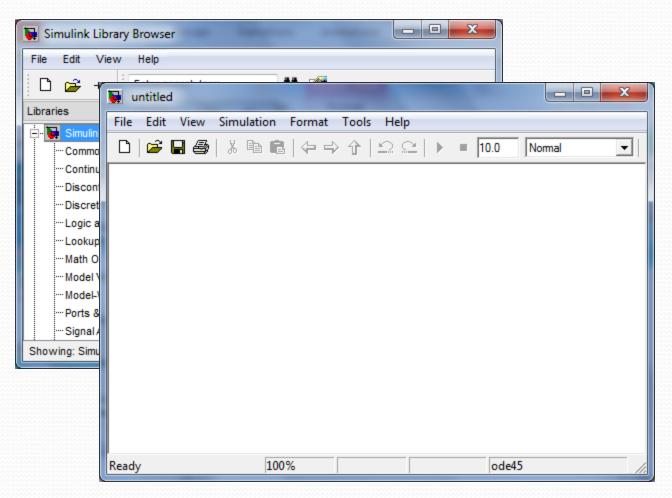
MATLAB 7.9.0 (R2009b)		
<u>File Edit T</u> ext <u>G</u> o <u>C</u> ell	T <u>o</u> ols De <u>b</u> ug <u>P</u> arallel <u>D</u> esktop <u>W</u>	
- 🗠 🛍 📠 👗 🛅 🚰	🚵 🗊 📄 🕜 🛛 C:\Program Files\MA	
Simulink Library Browser		x
File Edit View Help		
🗋 🚔 🛥 🛛 Enter search term	n 🗸 🚧 📺	
Libraries	Library: Simulink Search Results: (none)	•
Simulink Commonly Used Blocks Continuous Discontinuities Discrete Logic and Bit Operations Lookup Tables	Commonly Used Blocks	•
	Continuous	
	Discontinuities	
Math Operations Model Verification Model-Wide Utilities	Discrete	
Ports & Subsystems Signal Attributes	Logic and Bit II ⇐ Operations	Ŧ
Showing: Simulink		н





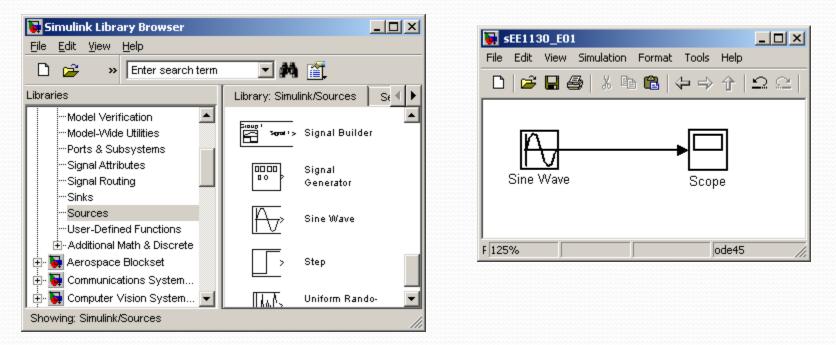
Simulink

• Click on the new document to open:



• Lets build an easy example of a sinewave and scope it:

 $x(t) = \sin(2\pi f t)$ with f = 1



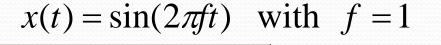
• We inserted a sinewave from the sources, and a scope from the sink library groups respectively.

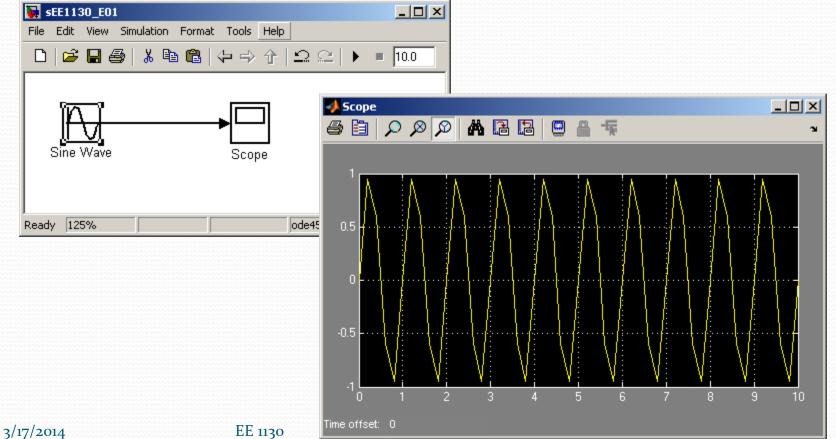
• By double clicking on the Sine Wave box we open its properties box and set the angular frequecy to 2*pi*1:

 $x(t) = \sin(2\pi f t)$ with f = 1

🙀 Source Block Parameters: Sine Wave	×	😽 sEE1130_E01 📃 🗖	×
Blas:		File Edit View Simulation Format Tools Help	
0		□ ☞ 🖩 🚳 % 🖻 🛍 (수 → 수 으 으	:
Frequency (rad/sec):			
2*pi*1			
Phase (rad):			
0		Sine Wave Scope	
Sample time:			
0		F 125% ode45	_
•			_//_
<u>O</u> K <u>C</u> ancel <u>H</u> elp <u>Apply</u>	/		

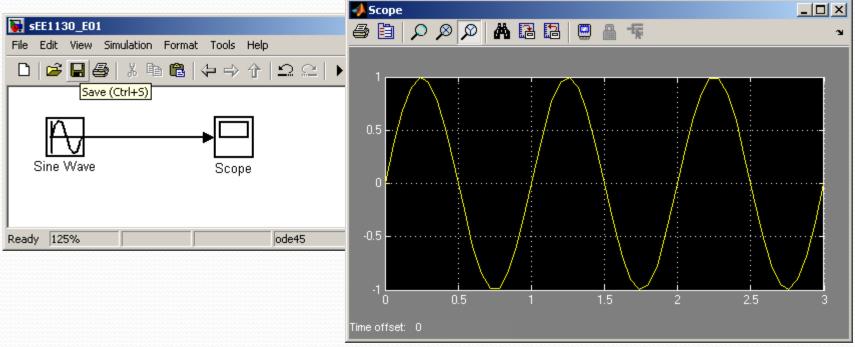
- To open the scope we double click on the Scope box.
- We hit the play icon to run the simulation.





• To obtain a better view of the sinewave, we reduce the running time to 3 seconds as shown in next figure:

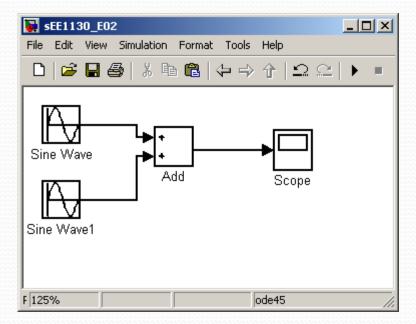
 $x(t) = \sin(2\pi f t)$ with f = 1



• Click on the binoculars to zoom the signal so it fills the Scope.

• Lets perform a summation of two sinewaves. One of 1Hz of frequency and 1 volt of amplitude and another of 60Hz frequency and 0.2 volts of amplitude:

$$x(t) = \sin(2\pi 1t) + 0.2\sin(2\pi 60t)$$



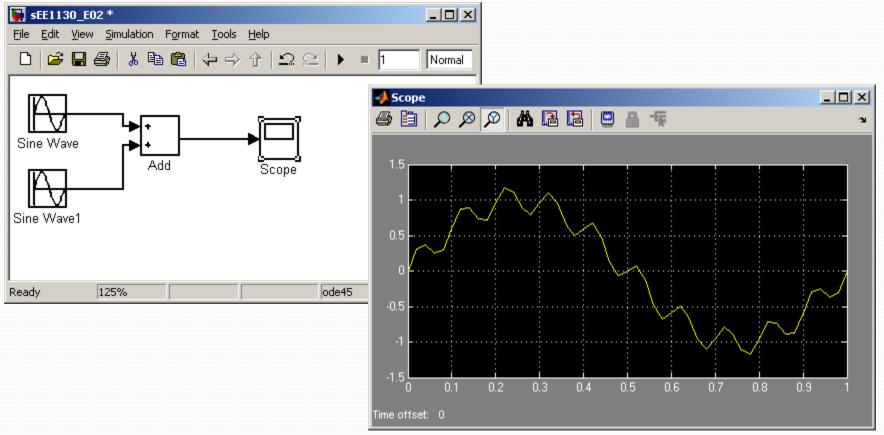
• We need to click on each sinewave box to set up the amplitude and frequencies: $x(t) = \sin(2\pi 1t) + 0.2\sin(2\pi 60t)$

SEE1130_E02	
File Edit View Simulation	Format Tools Help
D 😅 🖬 🚭 X 🖻	
Source Block Parameters: Sine Wave	Source Block Parameters: Sine Wave1
Parameters	Amplitude:
Sine type: Time based	0.2
Time (t): Use simulation time	Bias:
Amplitude:	
1	Frequency (rad/sec):
Bias:	2*pi*60
0	Phase (rad):
Frequency (rad/sec):	
2*pi*1	Sample time:
<u>O</u> K <u>C</u> ancel <u>H</u> elp <u>Apply</u>	<u>QK</u> <u>Cancel</u> <u>H</u> elp <u>Apply</u>

3/17/2014

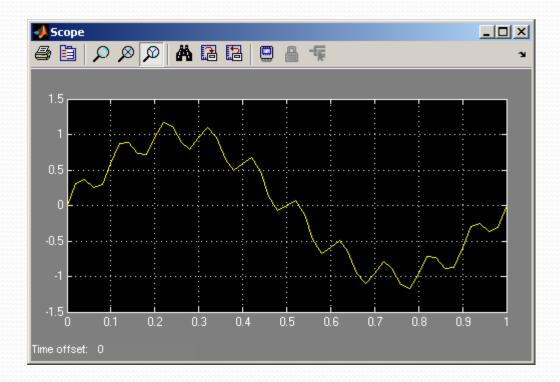
• Set the running time to 1 second, double click on the Scope box to open it up and hit play:

 $x(t) = \sin(2\pi 1t) + 0.2\sin(2\pi 60t)$



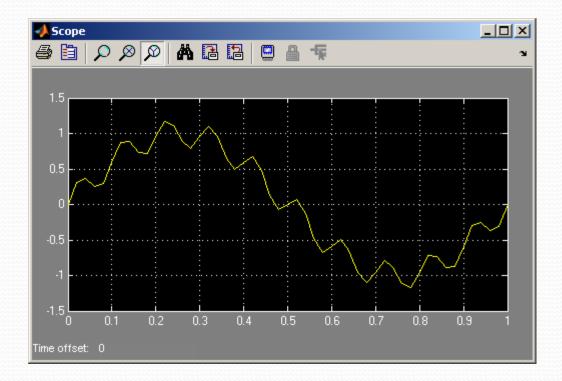
EE 1130

• We see the 60Hz wave riding on the 1Hz wave. This is called 60Hz noise or ripple.



 $x(t) = \sin(2\pi 1t) + 0.2\sin(2\pi 60t)$

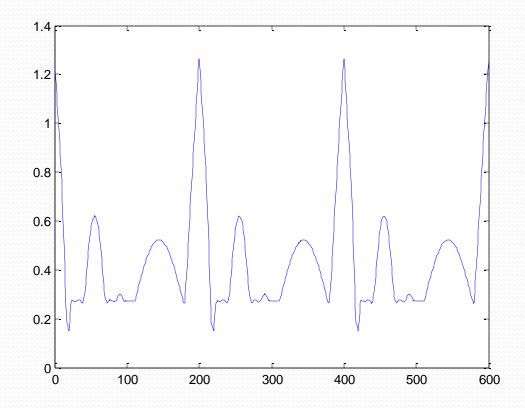
• The objective of next class is to get rid of the ripple and keep the 1Hz sinewave clean of noise!!! This is Signal Processing!!



 $x(t) = \sin(2\pi 1t) + 0.2\sin(2\pi 60t)$

Emulation of an EKG signal

• There is a code that I copied emulaing a EKG signal!!

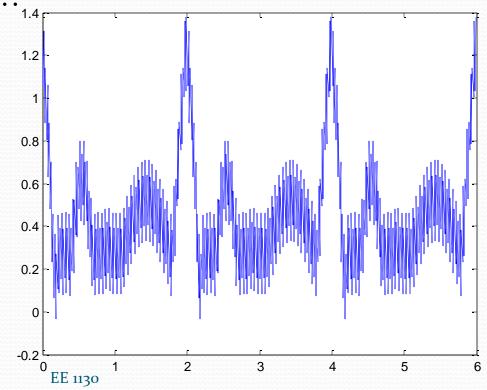


Emulation of an EKG signal

• The location is!!



• I modified the code to create a noisy signal. This is the power of Matlab!!



End of Class