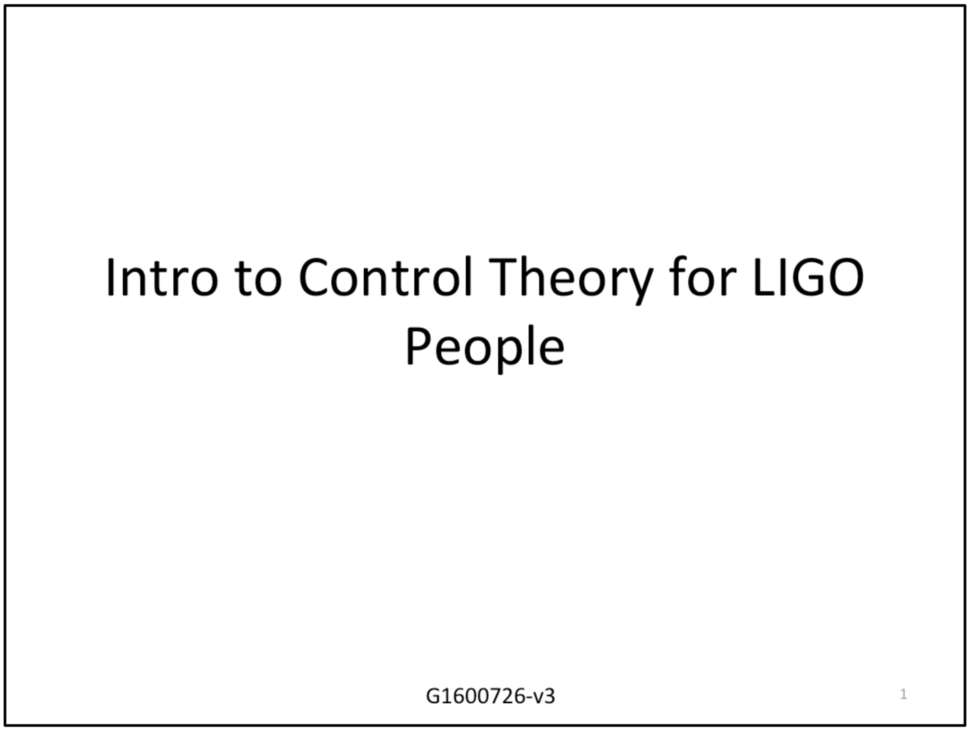
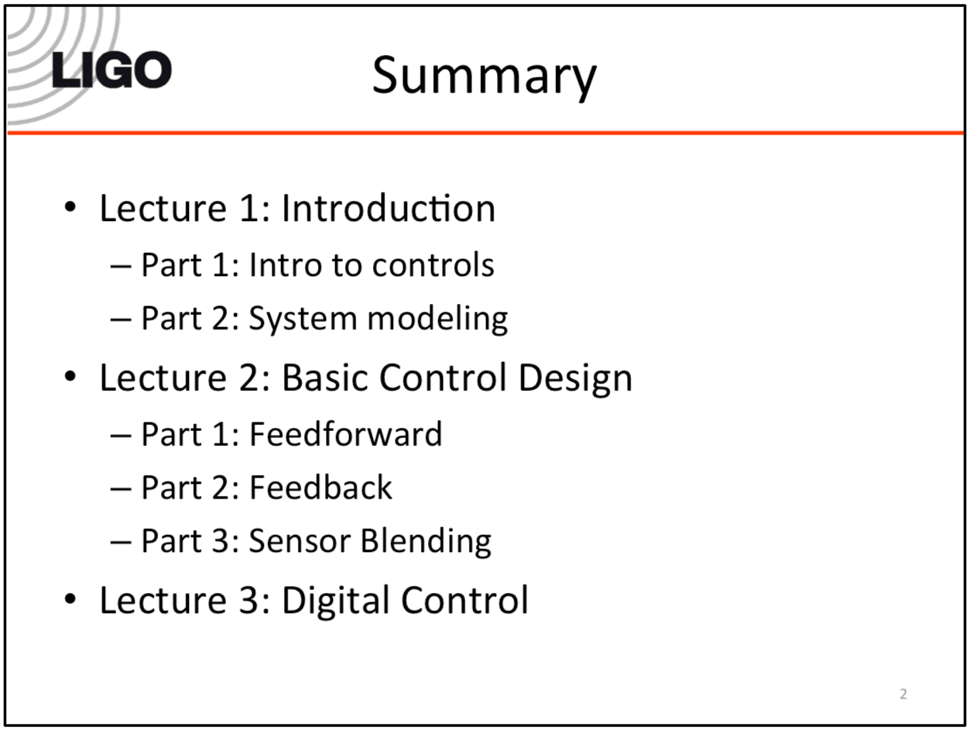


Intro to Control Theory for LIGO People

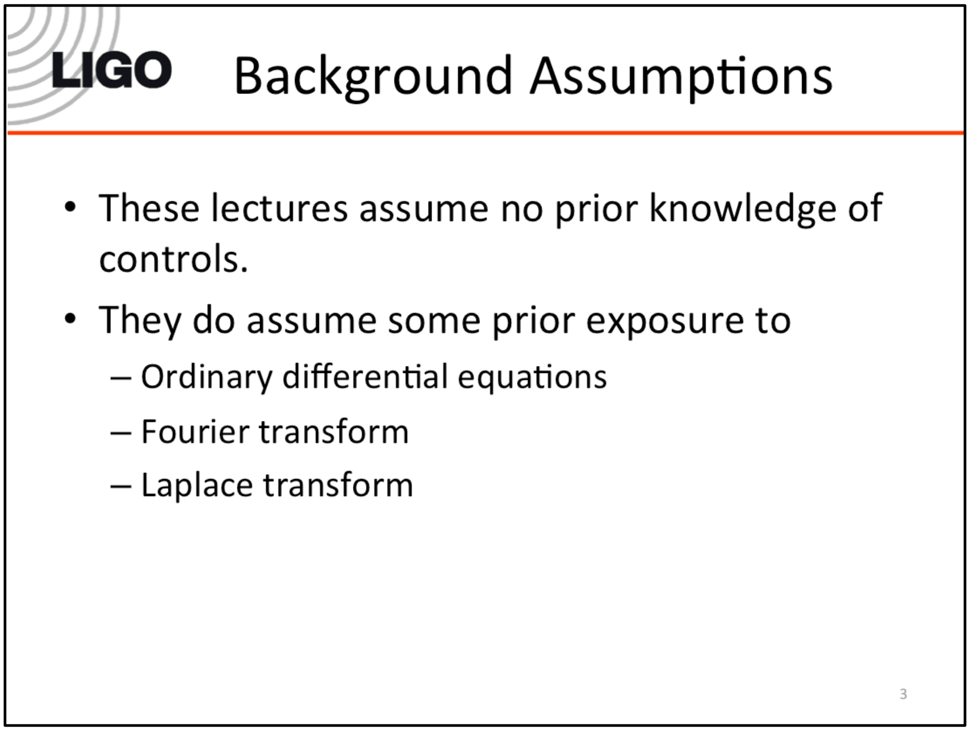
G1600726-v3

1



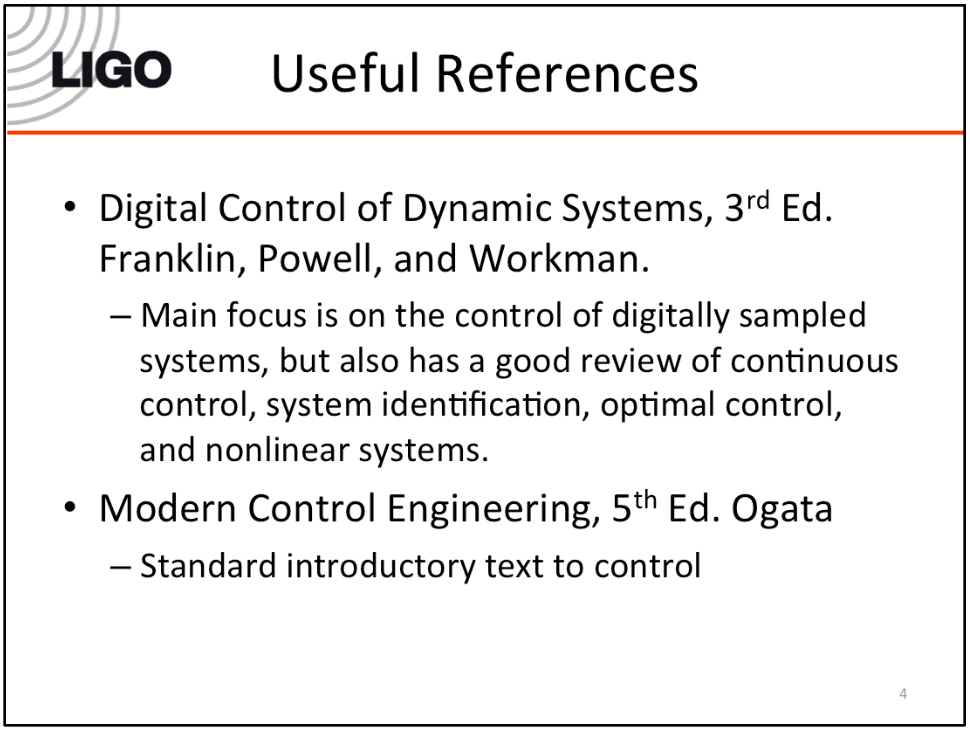
Summary

- Lecture 1: Introduction
  - Part 1: Intro to controls
  - Part 2: System modeling
- Lecture 2: Basic Control Design
  - Part 1: Feedforward
  - Part 2: Feedback
  - Part 3: Sensor Blending
- Lecture 3: Digital Control



Background Assumptions

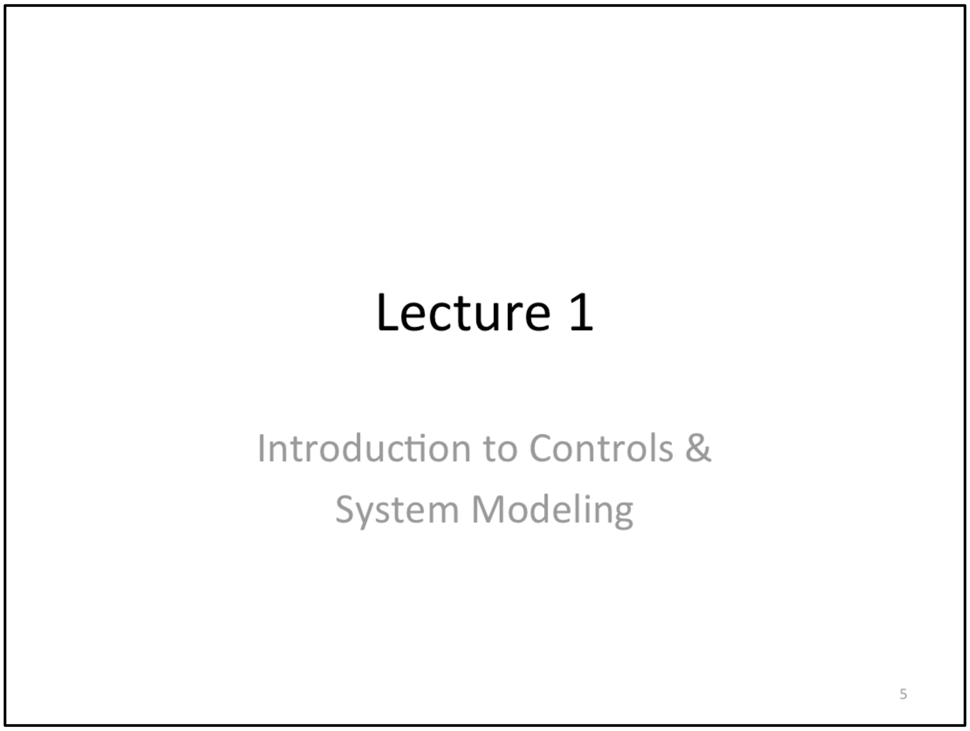
- These lectures assume no prior knowledge of controls.
- They do assume some prior exposure to
  - Ordinary differential equations
  - Fourier transform
  - Laplace transform



Useful References

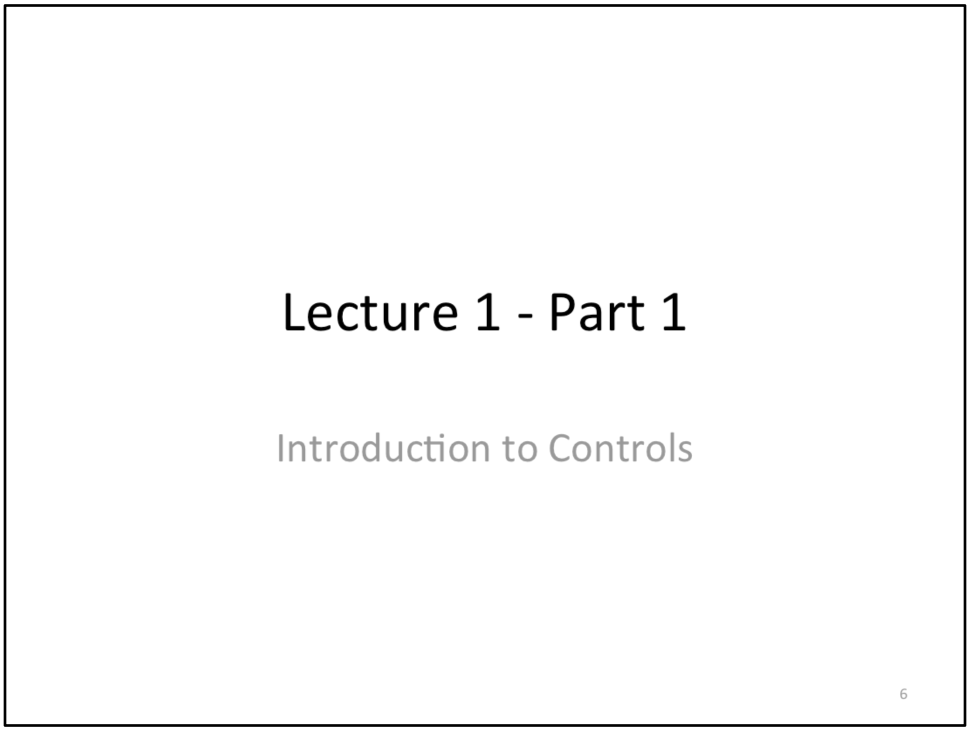
- Digital Control of Dynamic Systems, 3rd Ed. Franklin, Powell, and Workman.
  - Main focus is on the control of digitally sampled systems, but also has a good review of continuous control, system identification, optimal control, and nonlinear systems.
- Modern Control Engineering, 5th Ed. Ogata
  - Standard introductory text to control

Most of what you need will be in the first one. The second does have some complimentary information, for example it talks more about continuous controls, and it talks about the math behind stability vs. instability.



Lecture 1

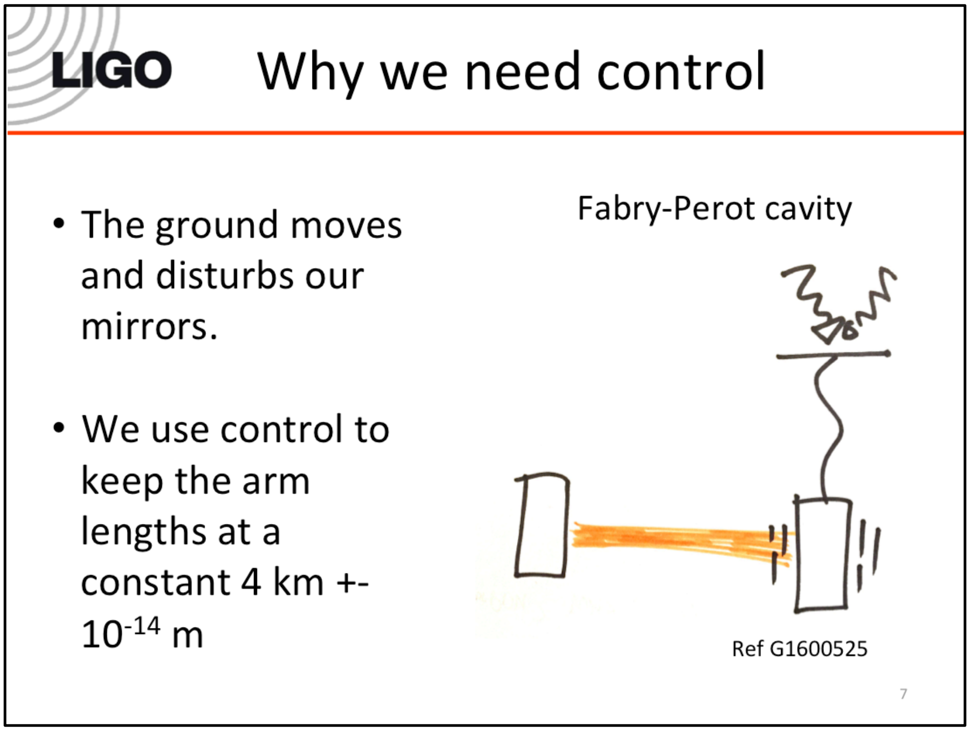
Introduction to Controls &
System Modeling



Lecture 1 - Part 1

Introduction to Controls

6



LIGO

Why we need control

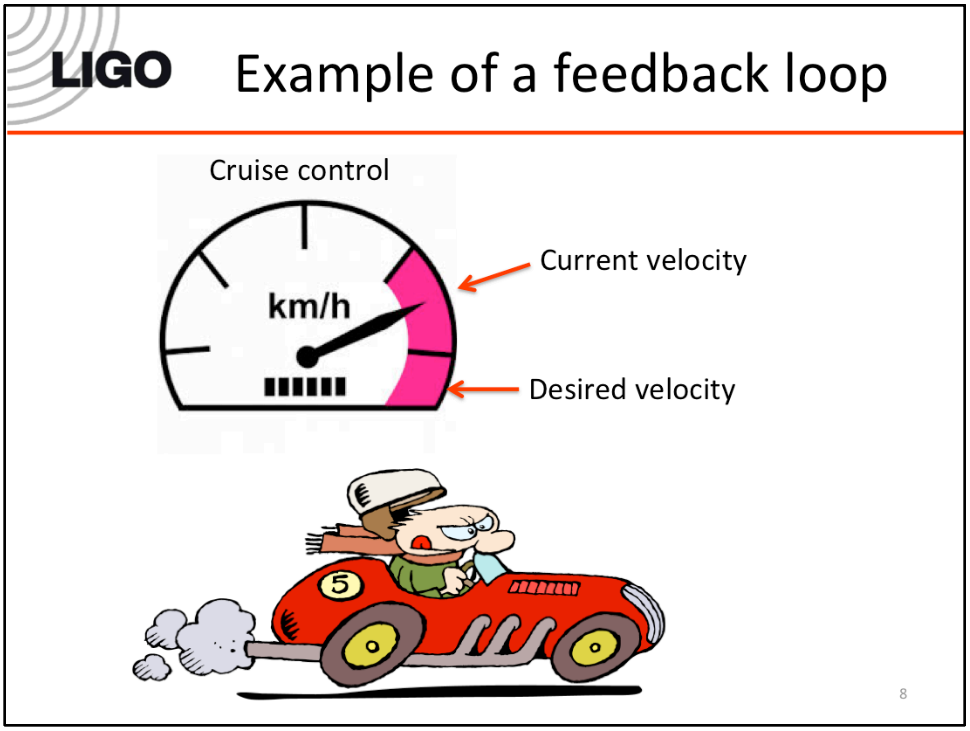
- The ground moves and disturbs our mirrors.
- We use control to keep the arm lengths at a constant 4 km +- 10^{-14} m

Fabry-Perot cavity

Ref G1600525

7

We need control because we need our relative mirror displacements to be within 10^-14 m rms (and some angular tolerance), but the ground moves them more than this by many orders of magnitude, around 10^-6 m.



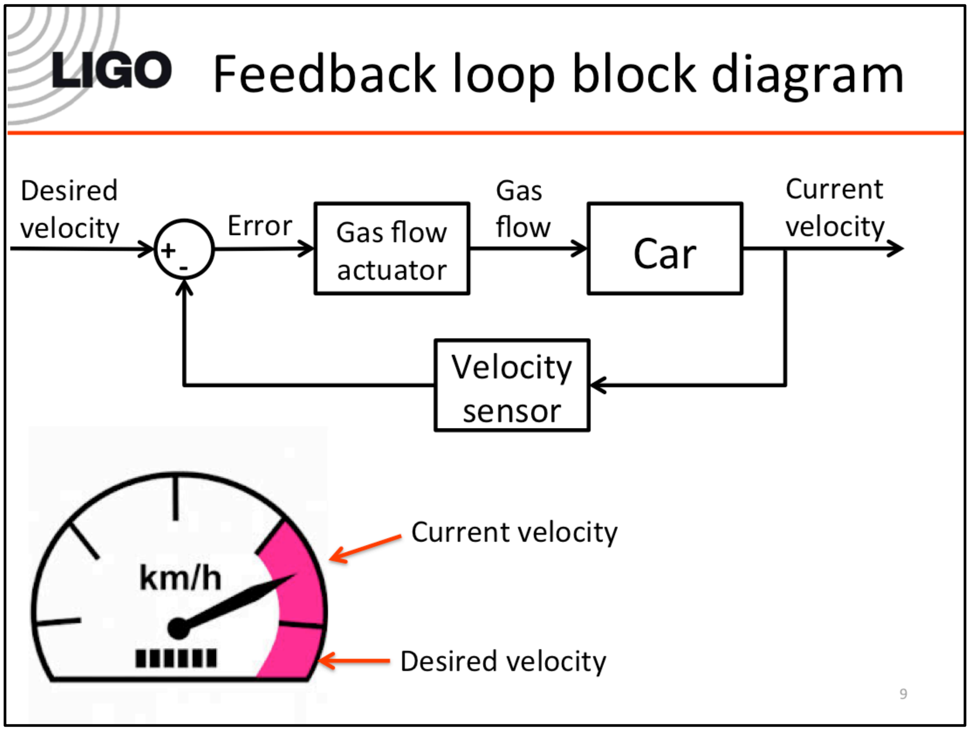
Example of a feedback loop

Cruise control

Current velocity

Desired velocity

Cruise control in your car is a familiar example of a relatively simple feedback loop. You set your car to go at some particular velocity, but you're currently going at some different velocity. The control is designed to adjust your velocity until it matches the desired velocity.



LIGO

Feedback loop block diagram

Desired velocity

Error

Gas flow actuator

Gas flow

Car

Current velocity

Velocity sensor

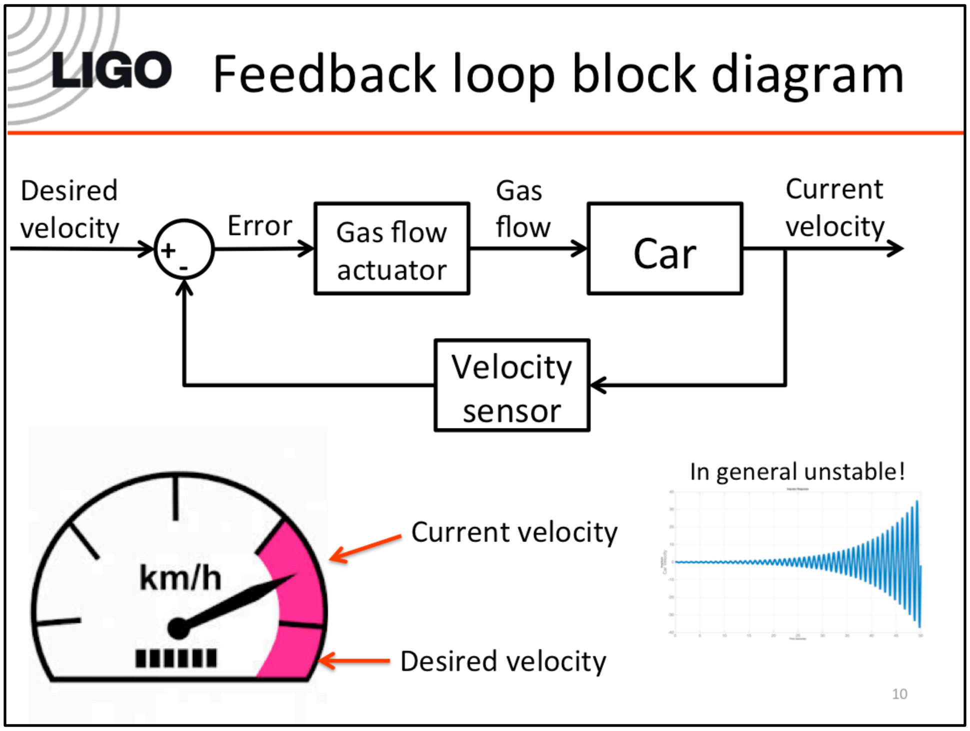
km/h

Current velocity

Desired velocity

9

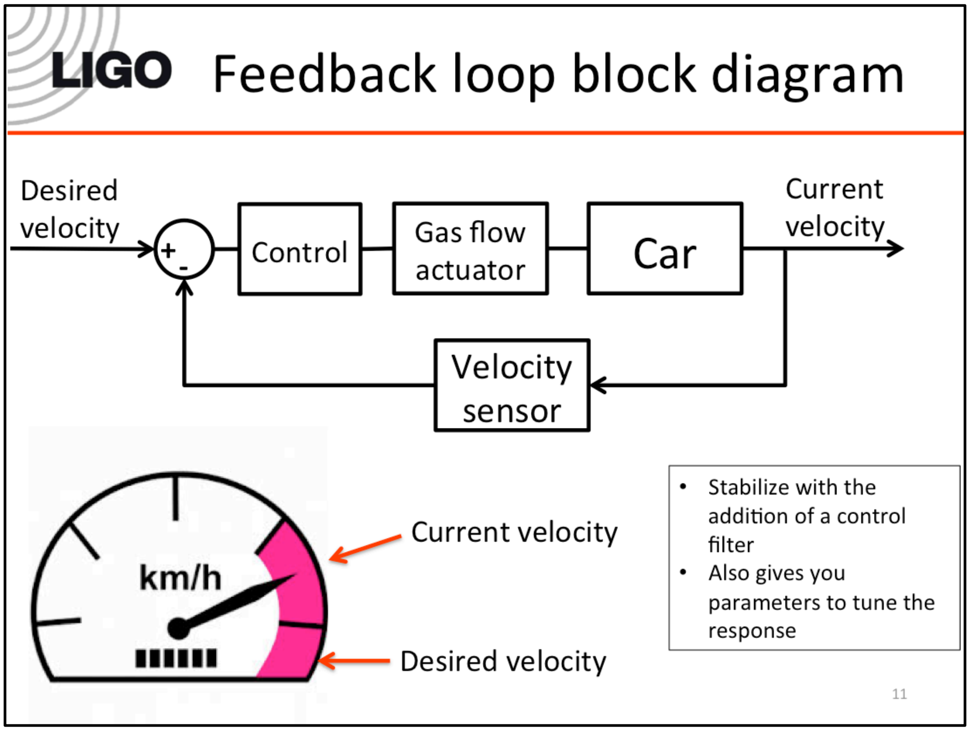
This is done with a feedback loop. There is a velocity sensor that measures how fast you are going. This signal is subtracted from the desired velocity to generate an error signal that you want to make small. There is also an actuator that gives the car gas. Naively, you could just take the velocity sensor and feed it back directly to the gas actuator, as shown here. That way, if you're going too slow, the error signal is positive and the car gets more gas. If you're going to fast, the error is negative and the car gets less gas.



LIGO Feedback loop block diagram

In general unstable!

However, whether or not this loop is stable depends entirely on the dynamic response of the car. It is quite likely that it will in fact be unstable. The plot in the bottom right corner is an example of an unstable response where the velocity oscillates and increases exponentially in amplitude.



LIGO

Feedback loop block diagram

Desired velocity

+

-

Control

Gas flow actuator

Car

Current velocity

Velocity sensor

km/h

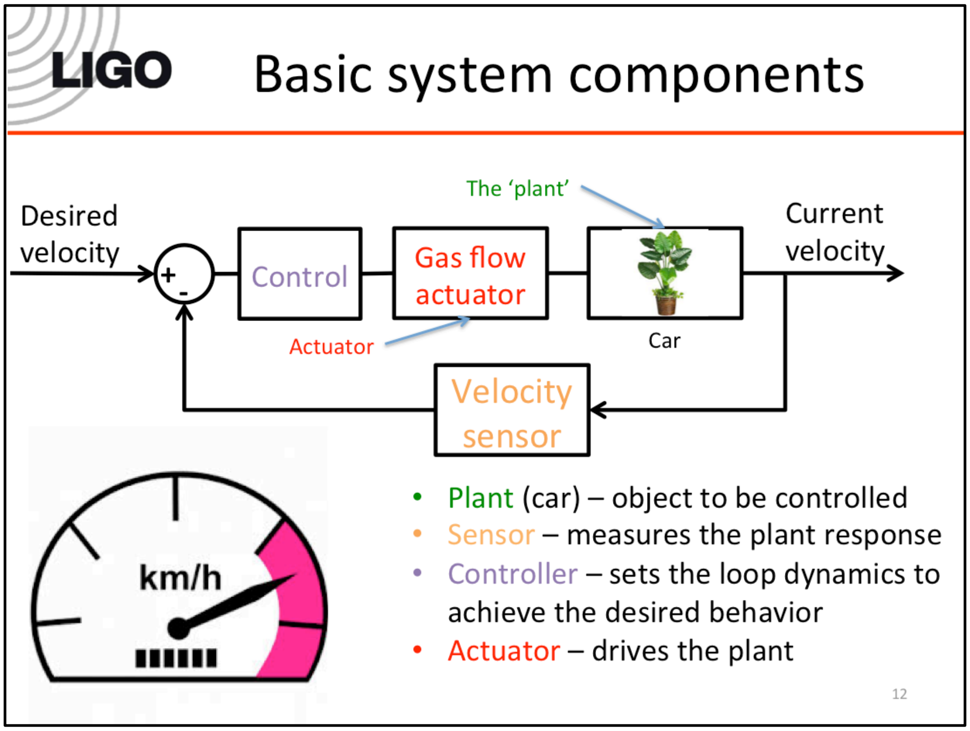
Current velocity

Desired velocity

- Stabilize with the addition of a control filter
- Also gives you parameters to tune the response

11

To avoid these issues we always include a control filter in the loop. The control filter contributes its own dynamics to the loop, which compensate for the dynamics of the car, allowing the loop to be stable. Also, the controller gives you a place to tune the response so the loop isn't just stable, but responds in a desirable way (avoids oscillations, offsets, etc).



Basic system components

Desired velocity

+

-

Control

Gas flow actuator

The ‘plant’

Car

Current velocity

Actuator

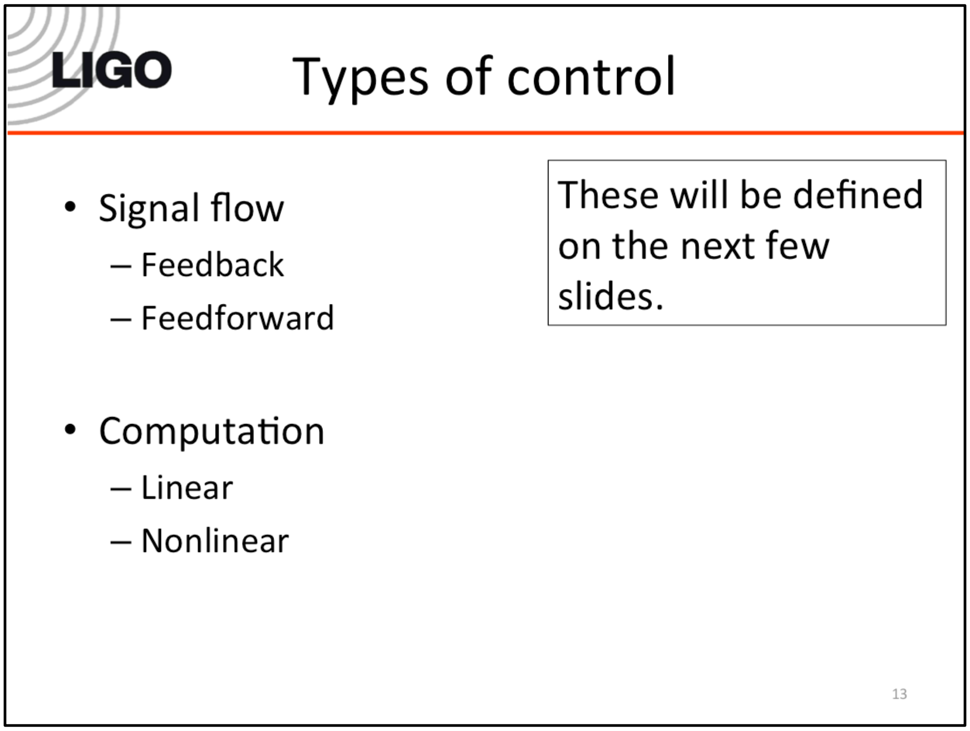
Velocity sensor

km/h

- Plant (car) – object to be controlled
- Sensor – measures the plant response
- Controller – sets the loop dynamics to achieve the desired behavior
- Actuator – drives the plant

12

The basic components of a feedback loop are a controller, an actuator, a sensor, and the plant. The plant is the name for the thing you want to control.



LIGO

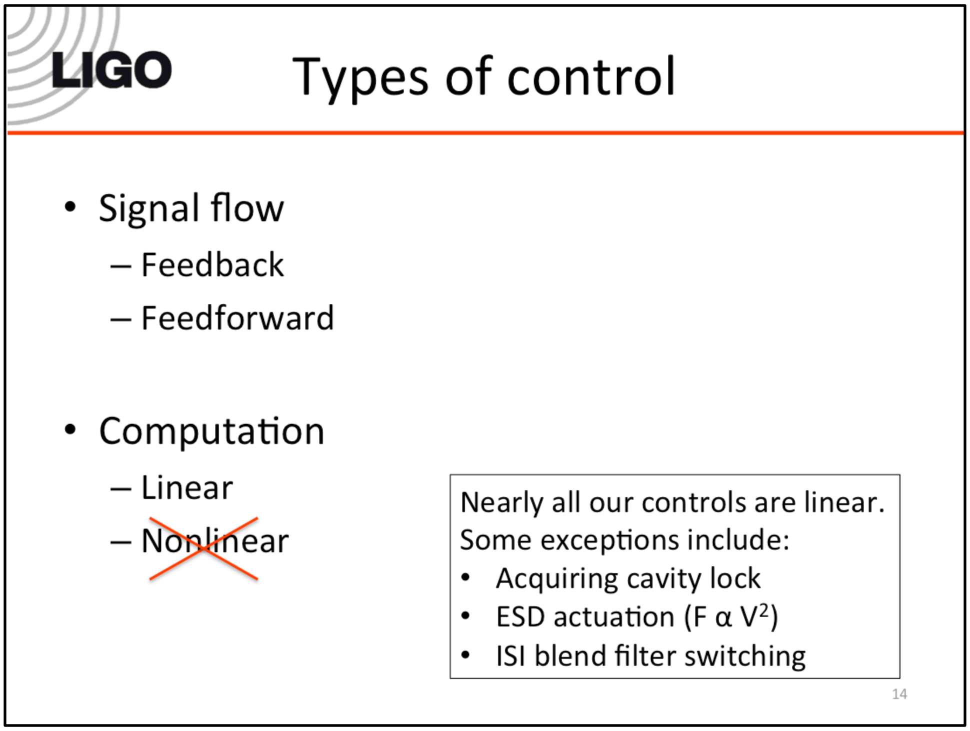
Types of control

- Signal flow
  - Feedback
  - Feedforward
- Computation
  - Linear
  - Nonlinear

These will be defined on the next few slides.

13

Control can take various forms. In can be either feedback, which we just saw, or feedforward. Both of these can then be linear or nonlinear.



Types of control

- Signal flow
  - Feedback
  - Feedforward
- Computation
  - Linear
  - ~~Nonlinear~~

Nearly all our controls are linear.
Some exceptions include:

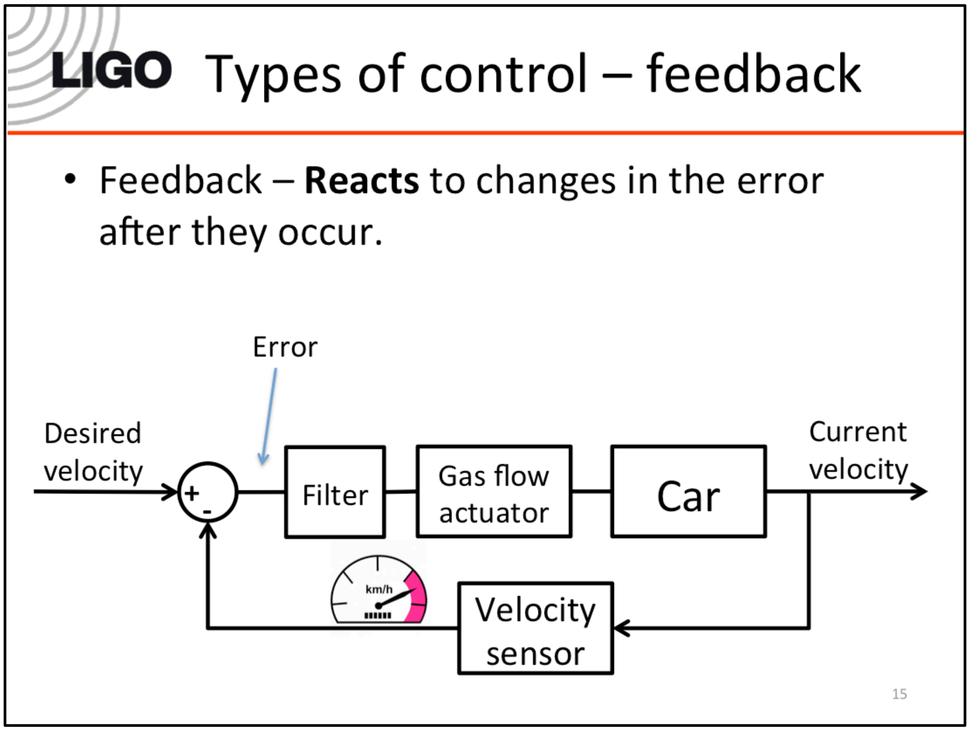
- Acquiring cavity lock
- ESD actuation (F α V^2)
- ISI blend filter switching

We’re not going to go over nonlinear control. With a few exceptions, all our systems in LIGO or linear. We work very hard to make them that way. And even the systems we have that are nonlinear are typically controlled with linear control filters as well.

In the case of cavity locking, the signal only exists when you swing through a fringe. Traditionally, to acquire lock, we simply apply the control signal only when that signal exists. If the cavity velocity isn’t too high then this works pretty well. There is research out there to improve this by estimating what the signal is when we can see it, using a model of the cavity.

In the case of the electrostatic drive (ESD), which is the actuator applied directly to the test mass, the force is proportional to the square of the ESD voltage. To make this linear, we simply apply a square root to the control signal within the controller software, before sending it to the ESD.

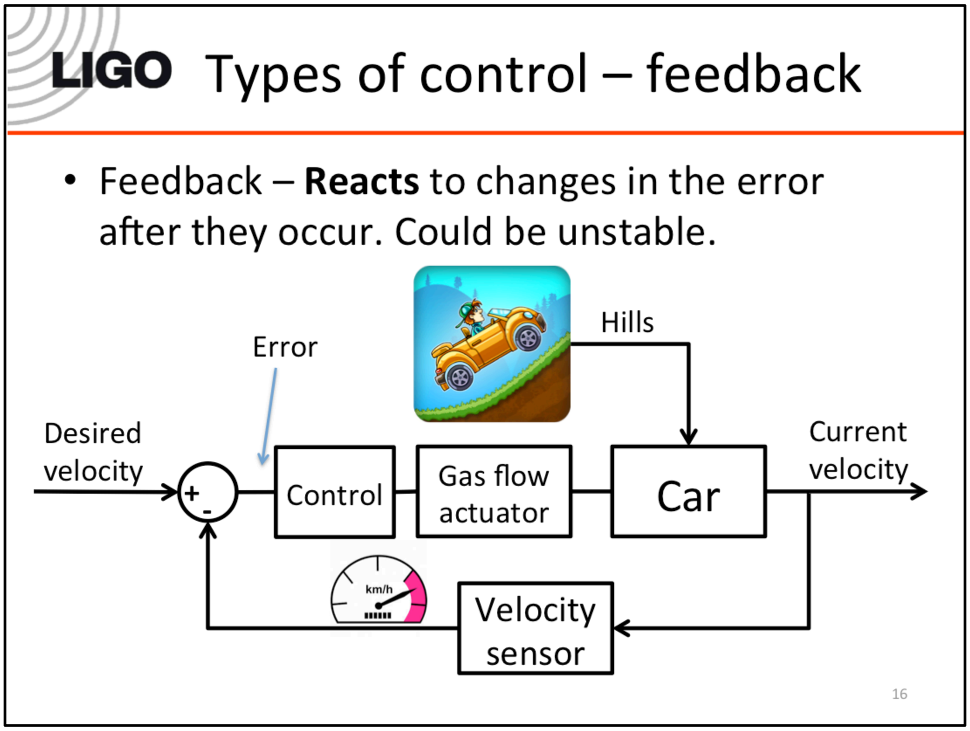
In the case of blend switching, where we switch from one set of blend filters to the other. This is done with a very slow ramp. The slowness of the ramp ensures the loops is always quasi-linear.



LIGO Types of control – feedback

- Feedback – **Reacts** to changes in the error after they occur.

The key feature of feedback is that it reacts to changes in the error signal. Thus, the control would do nothing if the error signal was zero. So, by definition feedback can never drive the error signal to exactly zero (it can at single frequencies with infinite loop gain, but not it general). It can in theory make the error arbitrarily small as the loop gain gets arbitrarily large, but not zero.



LIGO Types of control – feedback

- Feedback – **Reacts** to changes in the error after they occur. Could be unstable.

Hills

Error

Desired velocity

+
-

Control

Gas flow actuator

Car

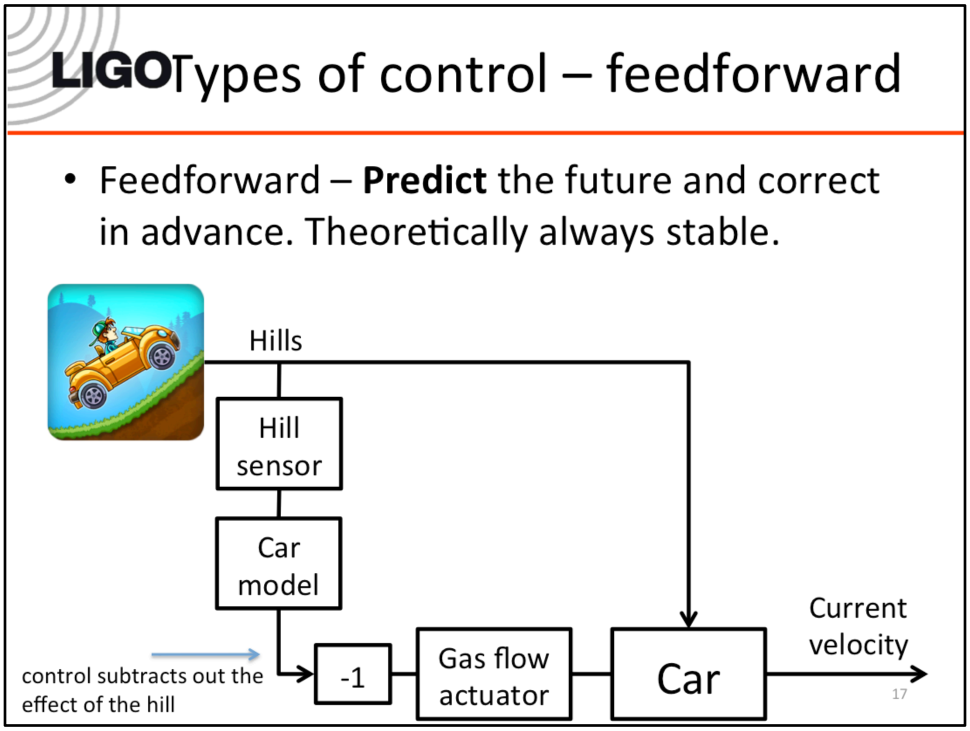
Current velocity

Velocity sensor

16

An example of something that will tend to increase the error size is when your car approaches a hill. The hill is a disturbance that tends to slow the car. The controller will only start to respond when it sees the error signal increase. It reacts after the error begins to change.

Another defining feature of feedback is that it can go unstable if the controller isn’t designed appropriately for the plant (or if something in the plant changes or breaks).



LIGO Types of control – feedforward

- Feedforward – **Predict** the future and correct in advance. Theoretically always stable.

Hills

Hill sensor

Car model

-1

Gas flow actuator

Car

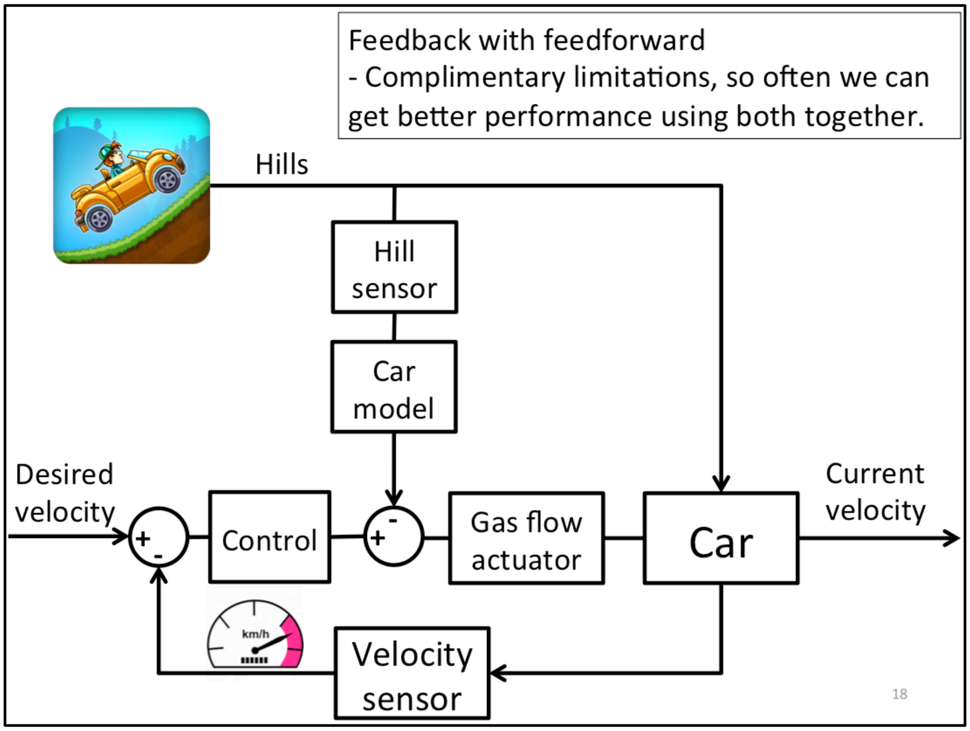
Current velocity

control subtracts out the effect of the hill

Feedforward is in many respects the opposite to feedback. It tries to predict how the car (or plant) is going to respond, and corrects for it in advance. So, if the car had a sensor that saw the hill coming, and then a model of how the car would react, a feedforward controller could adjust the gas flow before the hill even had the chance to slow the car down. Mathematically, it subtracts the hill out of the loop, so it's like it isn't even there. To do this well, you need a good model of the plant, and a sensor that has good coherence to the plant's response.

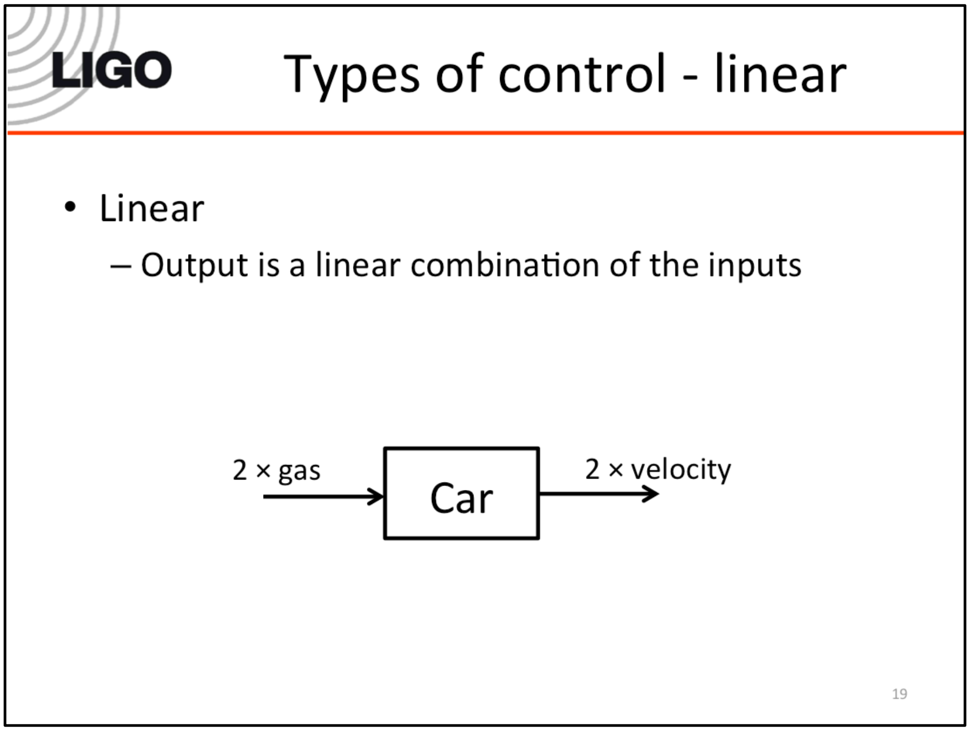
An example of this in LIGO, is the feedforward used by the ISIs. An inertial sensor is placed on the ground, which measures the ground's displacement. The feedforward controller is a filter that models how the ISI will respond to this ground motion. This controller then sends a correction signal to the ISI actuators to cancel out any forces the ground imposes on the ISI, before the ISI has the chance to respond. The reason this works, without needing a sensor that can see the future, is because it takes the ISI some time to respond to the ground. So if you can apply the correction forces quickly enough, you can realize a useful feedforward controller. Clearly, any phase delays will degrade the performance.

An important feature of feedforward is that it is, in theory, always stable. Feedback is required for instability to happen, and there is no feedback here. In practice, instability can happen due to parasitic feedback paths. For example, when the



Feedback with feedforward
- Complimentary limitations, so often we can get better performance using both together.

Feedback and feedforward have complimentary limitations. So in general you get the best performance by using both simultaneously. The idea being, you reduce the size of the disturbance as much as possible with feedforward, then suppress it further with feedback. So, if for example, you can reduce the disturbance by a factor of 10 with feedforward, and 100 with feedback, then with both you'll get 1000.

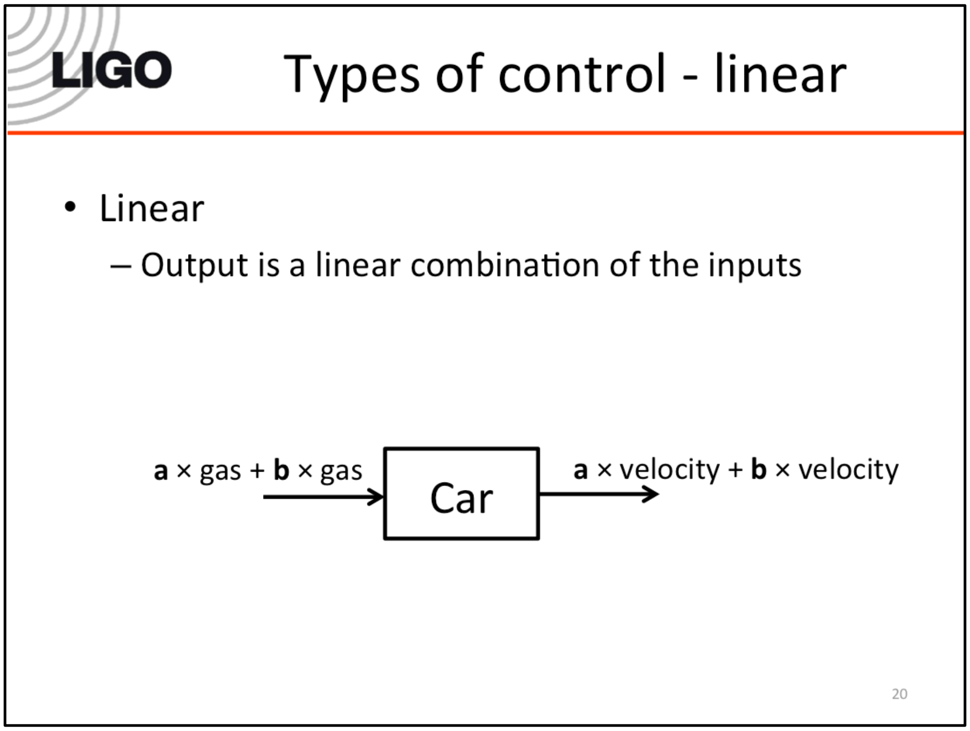


Types of control - linear

- Linear
  - Output is a linear combination of the inputs

2 × gas → Car → 2 × velocity

A linear system has the property that its outputs are a linear combination of its inputs. So if you double the input, you double the output.



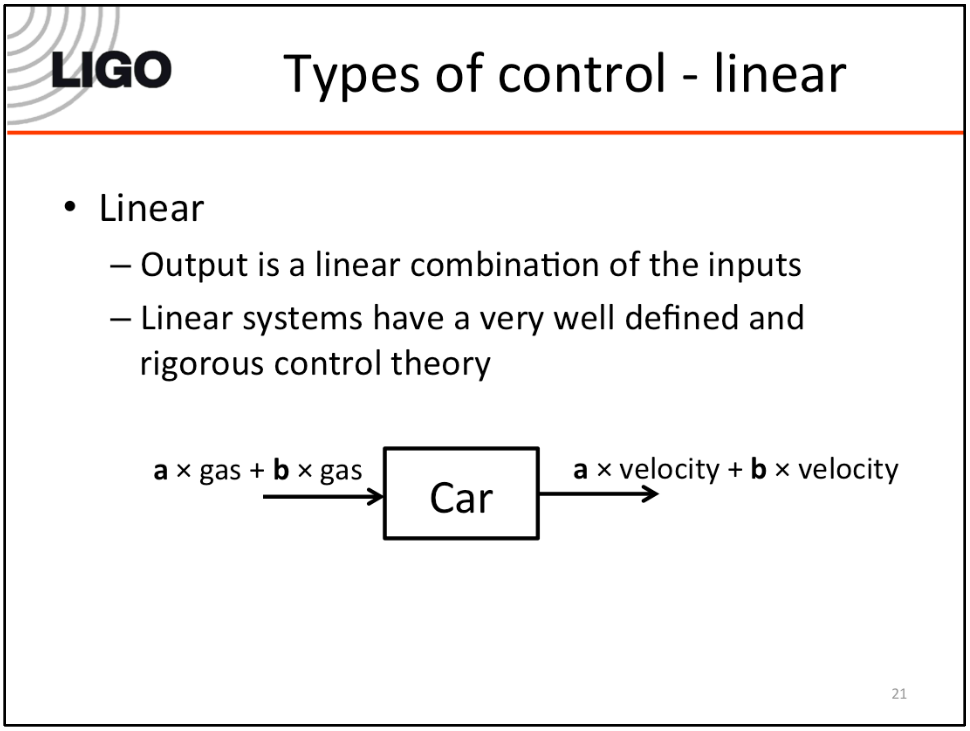
LIGO

Types of control - linear

- Linear
  - Output is a linear combination of the inputs

a × gas + **b** × gas
Car
a × velocity + **b** × velocity

Or more generally…



LIGO

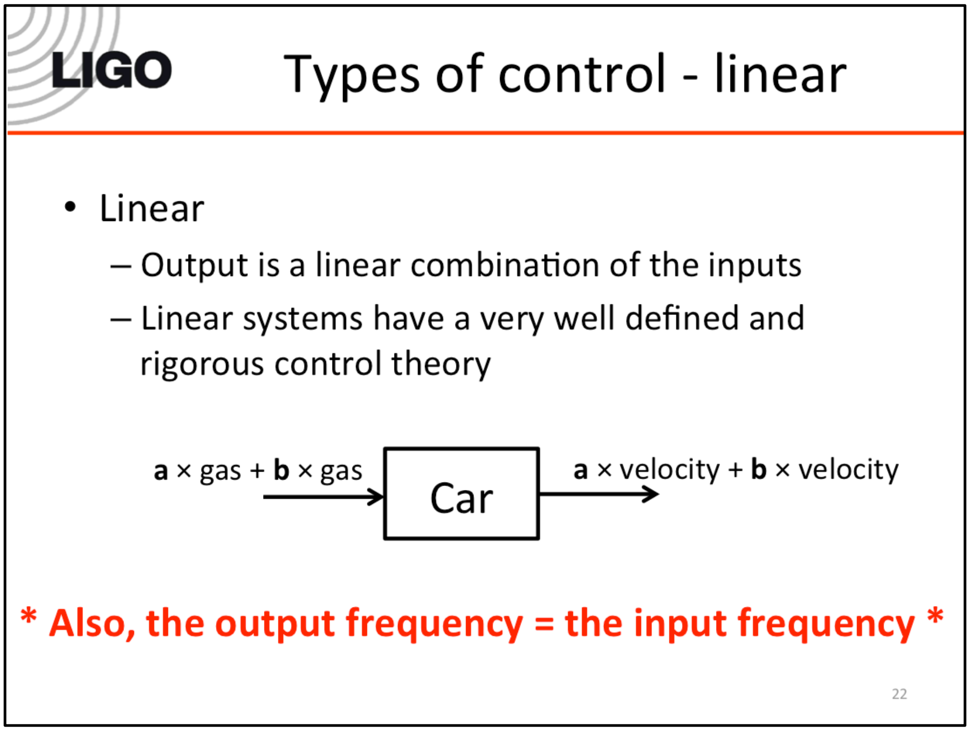
Types of control - linear

- Linear
  - Output is a linear combination of the inputs
  - Linear systems have a very well defined and rigorous control theory

a × gas + **b** × gas → Car → **a** × velocity + **b** × velocity

21

Linear systems are useful because there is a very well defines and rigorous theory to control them.



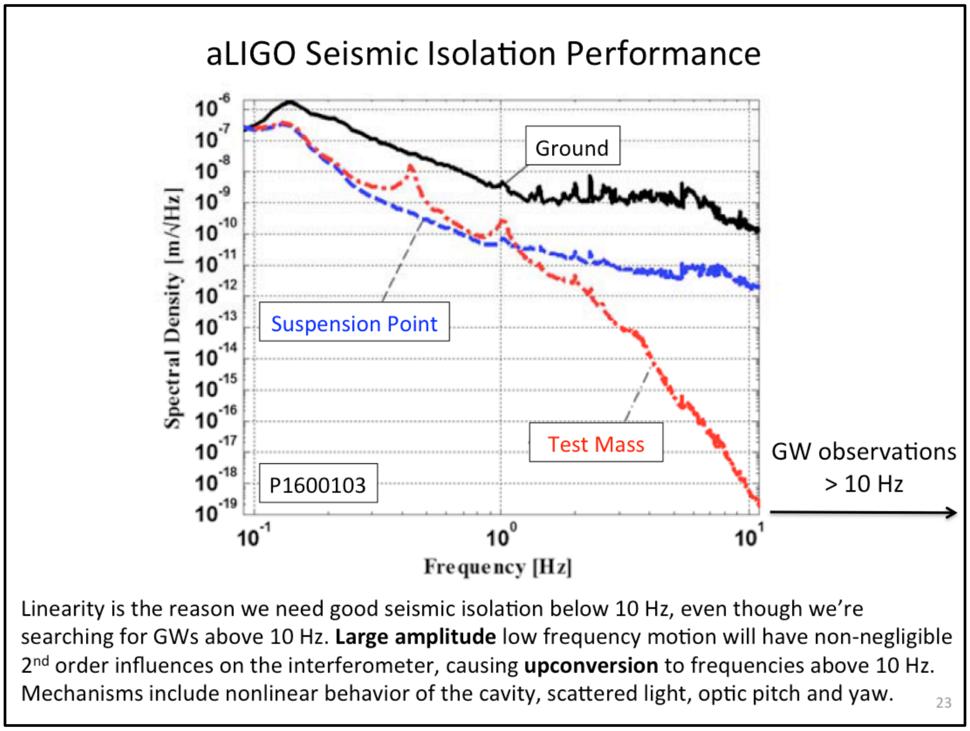
Types of control - linear

- Linear
  - Output is a linear combination of the inputs
  - Linear systems have a very well defined and rigorous control theory

a × gas + **b** × gas → Car → **a** × velocity + **b** × velocity

*** Also, the output frequency = the input frequency ***

Another important feature is that the output frequency is always the same as the input frequency. This is an extremely important property. We often see this fail when our systems develop large amplitude oscillations, because the interferometer behaves non-linearly at large amplitudes.

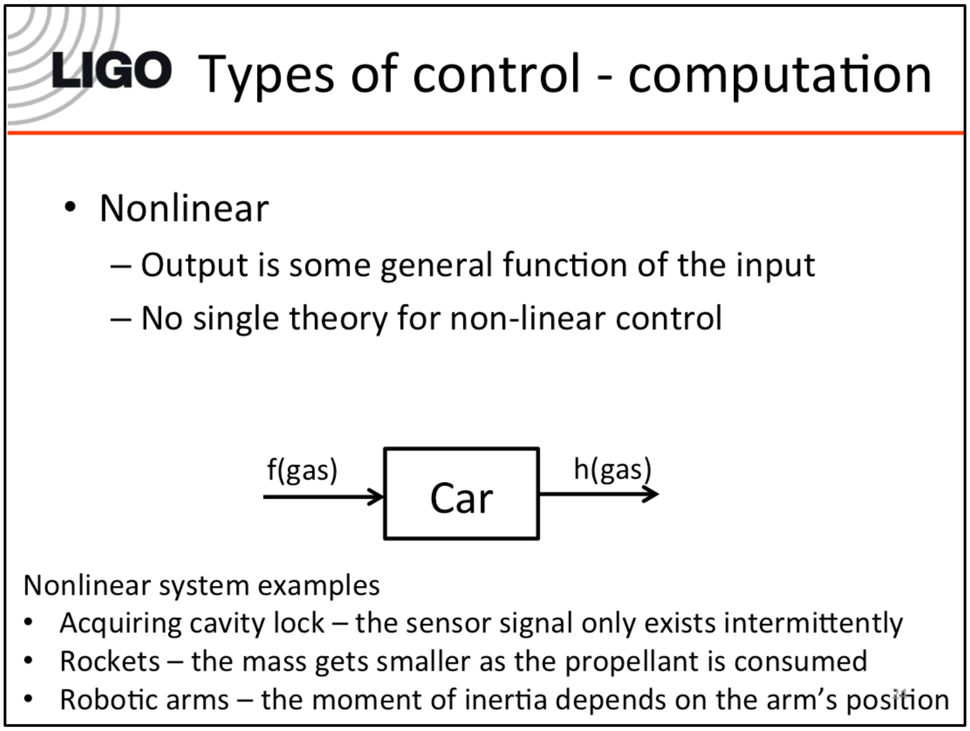


aLIGO Seismic Isolation Performance

Linearity is the reason we need good seismic isolation below 10 Hz, even though we're searching for GWs above 10 Hz. **Large amplitude** low frequency motion will have non-negligible 2nd order influences on the interferometer, causing **upconversion** to frequencies above 10 Hz. Mechanisms include nonlinear behavior of the cavity, scattered light, optic pitch and yaw.

23

To stress this issue of large amplitude oscillations, this is why we work hard to control the seismic noise below 10 Hz, even though we're observing gravitational waves only above 10 Hz. If we didn't do this, we wouldn't even be able to lock the interferometer. If we didn't do it well, we might lock the interferometer, but the low frequency noise would **upconvert** above 10 Hz.



LIGO Types of control - computation

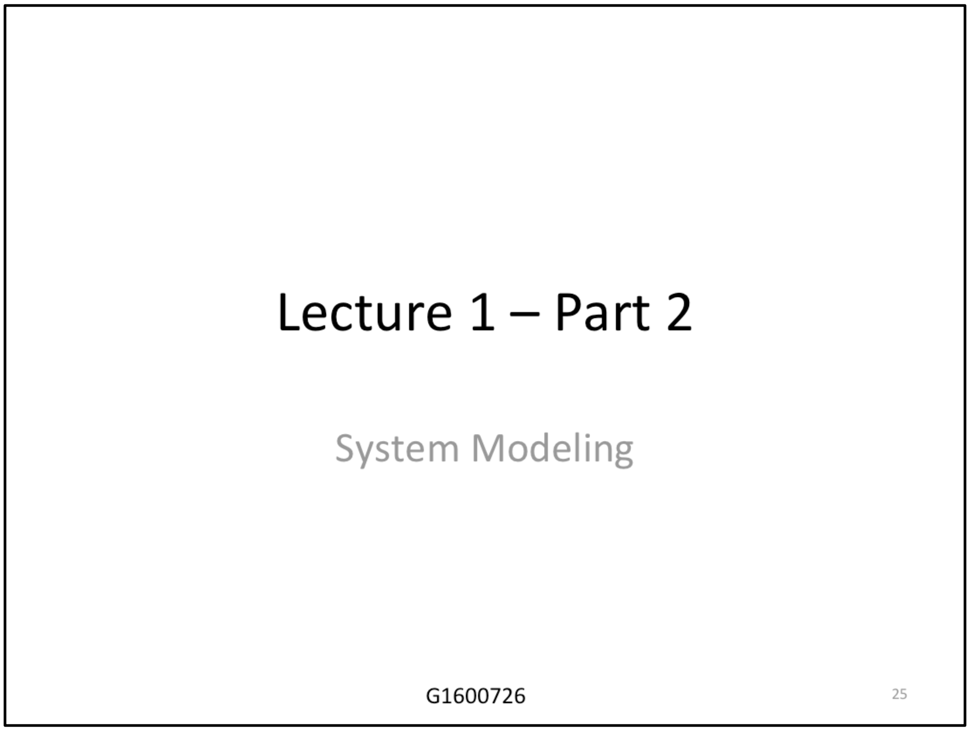
- Nonlinear
  - Output is some general function of the input
  - No single theory for non-linear control

f(gas) → Car → h(gas)

Nonlinear system examples

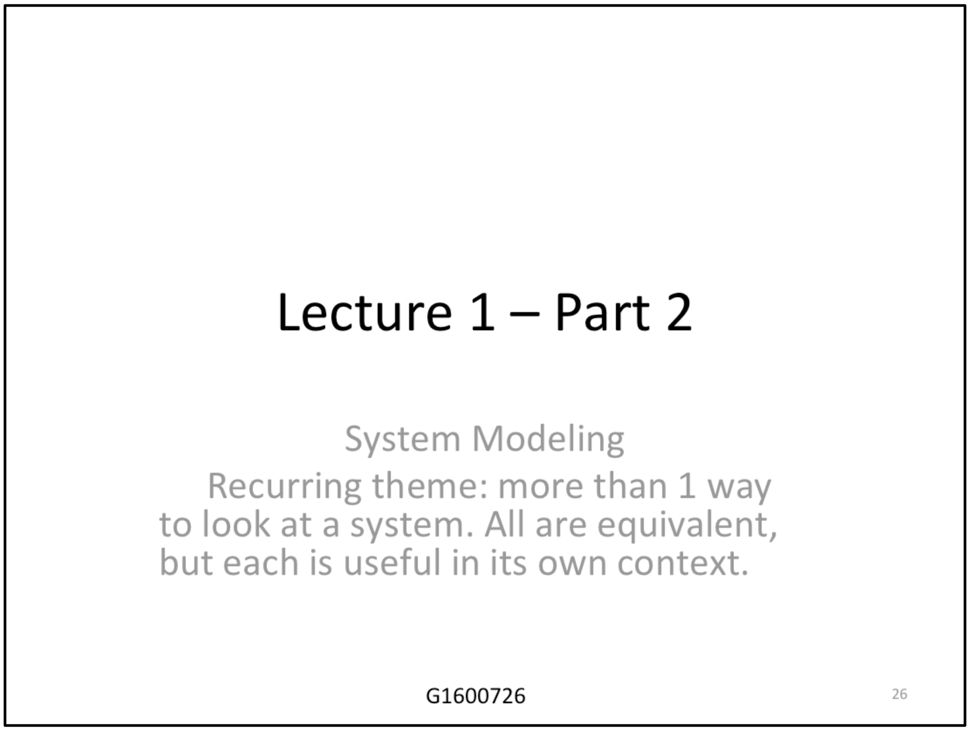
- Acquiring cavity lock – the sensor signal only exists intermittently
- Rockets – the mass gets smaller as the propellant is consumed
- Robotic arms – the moment of inertia depends on the arm's position

Nonlinear systems can be nonlinear in infinite ways. Their outputs can be any function of the input. Therefore there is no single theory to control them. We work hard to make sure our systems are linear to avoid the issues associated with nonlinear systems (upconversion, poorly defined control laws, etc).



Lecture 1 – Part 2

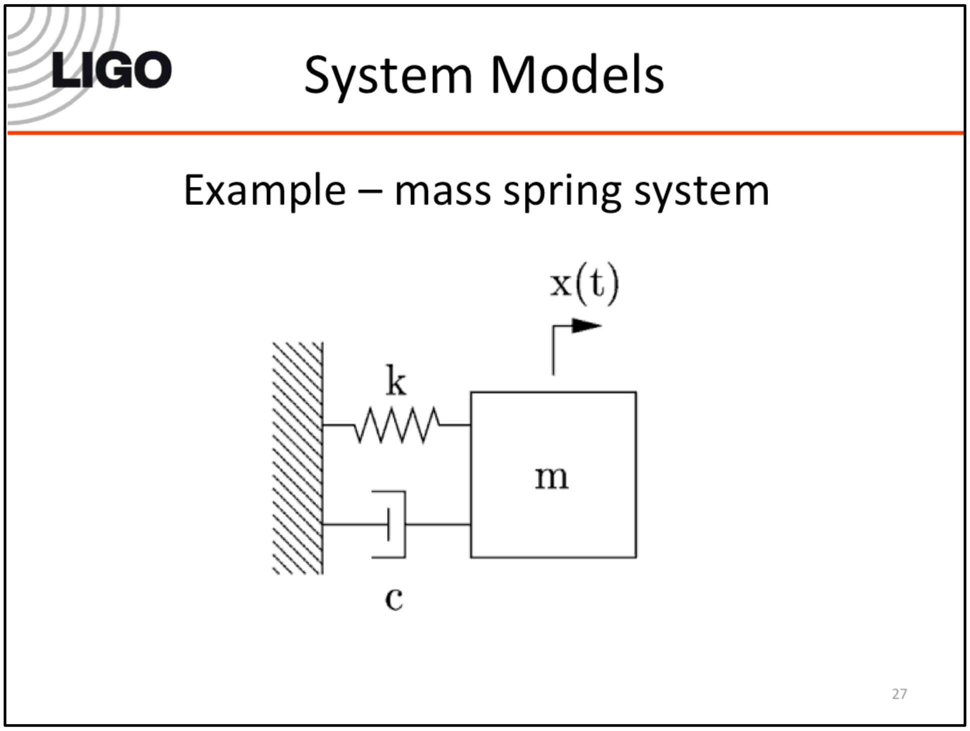
System Modeling



Lecture 1 – Part 2

System Modeling

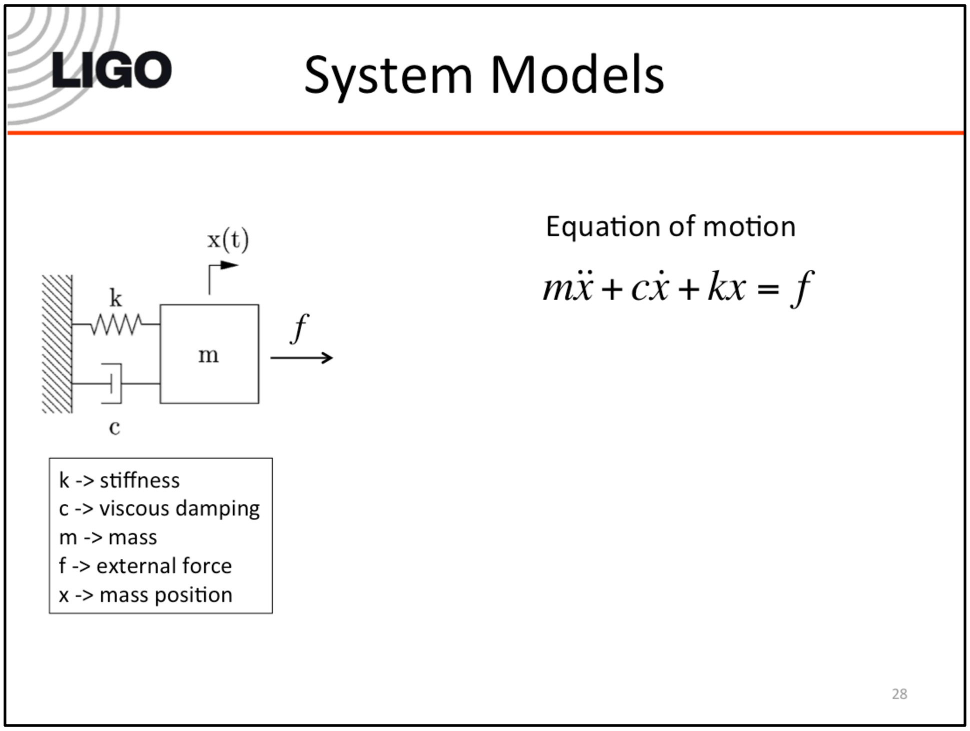
Recurring theme: more than 1 way to look at a system. All are equivalent, but each is useful in its own context.



System Models

Example – mass spring system

We’ll follow the example of a simple mass spring system (which turns out to be a half decent model for the HAM-ISI).



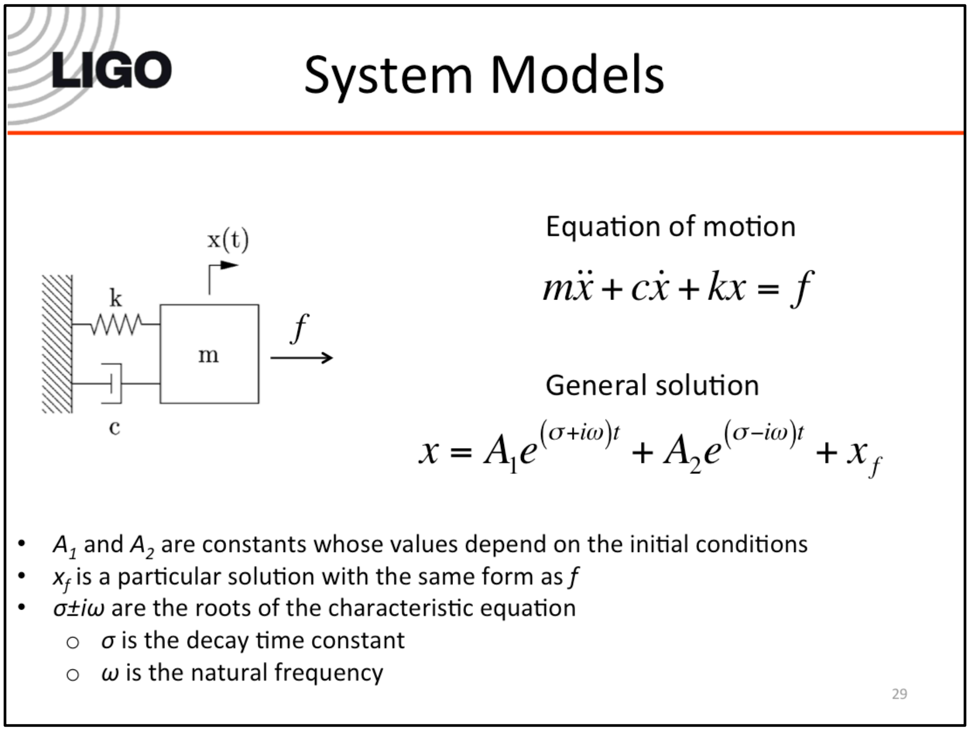
System Models

k -> stiffness
c -> viscous damping
m -> mass
f -> external force
x -> mass position

Equation of motion

$$m\ddot{x} + c\dot{x} + kx = f$$

Here is it’s equation of motion.



System Models

Equation of motion

$$m\ddot{x} + c\dot{x} + kx = f$$

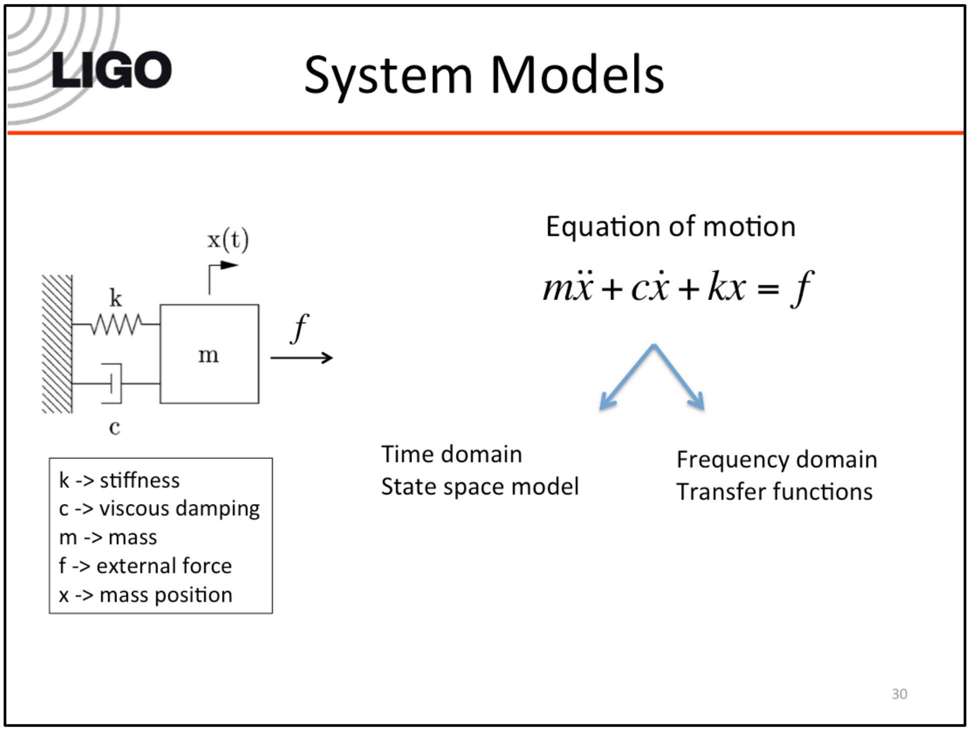
General solution

$$x = A_1 e^{(\sigma + i\omega)t} + A_2 e^{(\sigma - i\omega)t} + x_f$$

- A_1 and A_2 are constants whose values depend on the initial conditions
- x_f is a particular solution with the same form as f
- $\sigma \pm i\omega$ are the roots of the characteristic equation
  - σ is the decay time constant
  - ω is the natural frequency

29

It has this general solution. The exponents of this solution are very important, as we'll see in the remainder of this lecture.



System Models

k -> stiffness
c -> viscous damping
m -> mass
f -> external force
x -> mass position

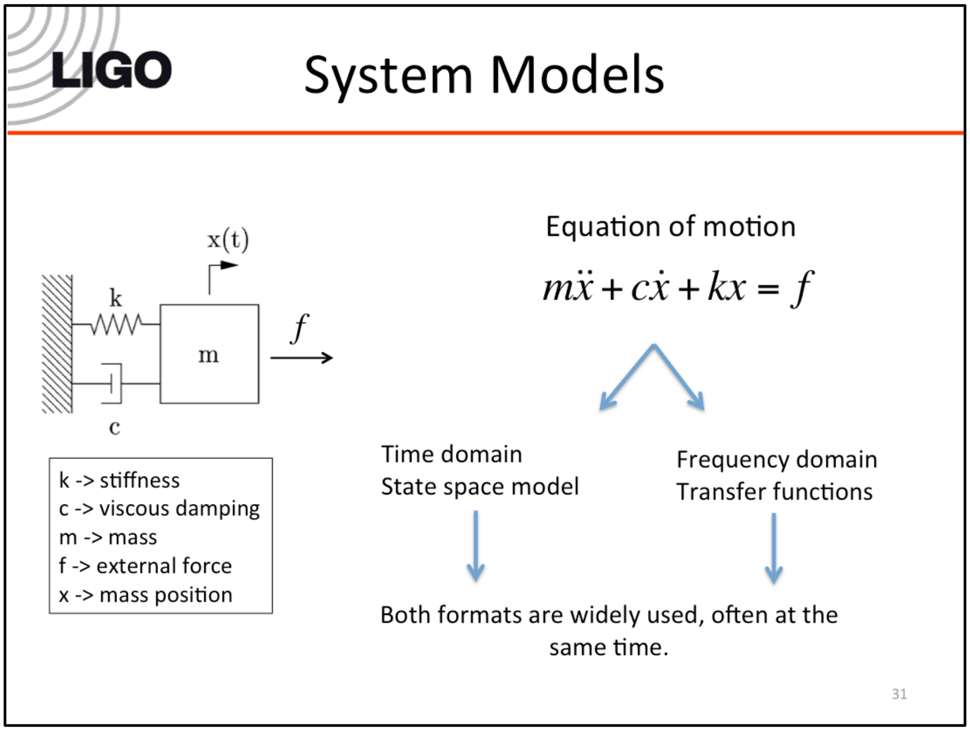
Equation of motion

$$m\ddot{x} + c\dot{x} + kx = f$$

Time domain
State space model

Frequency domain
Transfer functions

This system can be modeled in the time domain or the frequency domain.



LIGO

System Models

k -> stiffness
c -> viscous damping
m -> mass
f -> external force
x -> mass position

Equation of motion

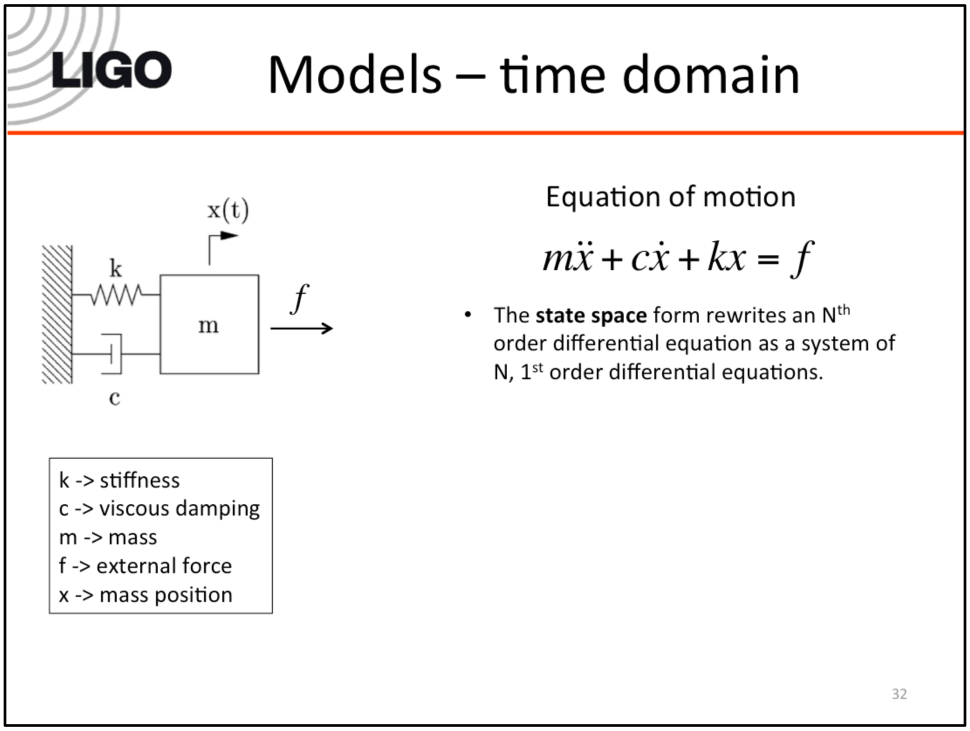
$$m\ddot{x} + c\dot{x} + kx = f$$

Time domain
State space model

Frequency domain
Transfer functions

Both formats are widely used, often at the same time.

The time domain is done with the State Space format. The frequency domain uses transfer functions.



Models – time domain

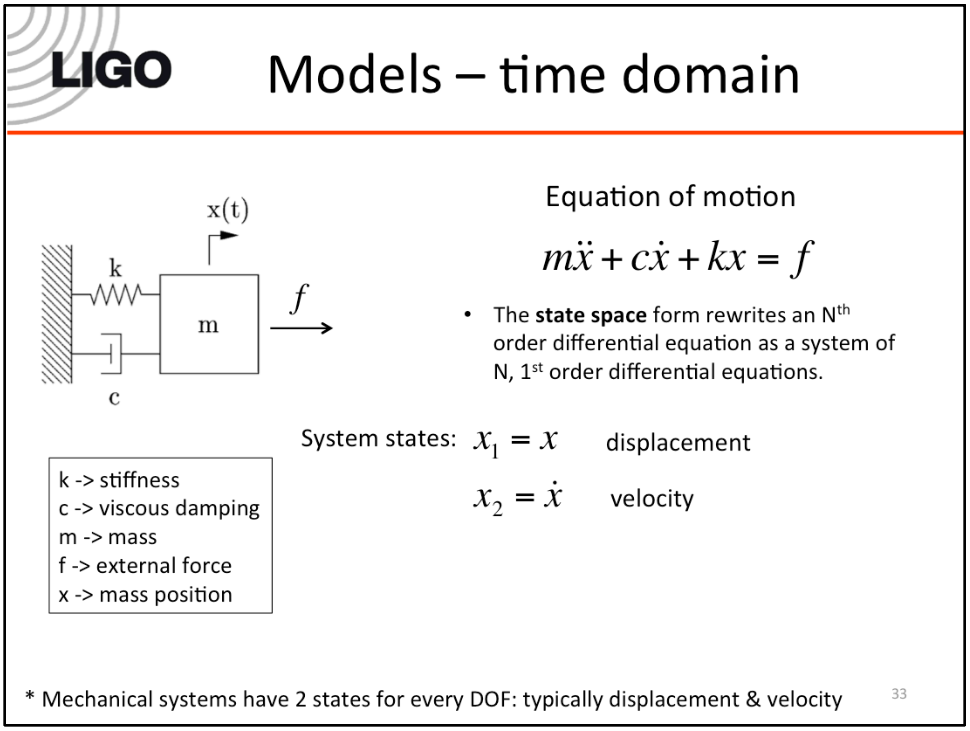
Equation of motion

$$m\ddot{x} + c\dot{x} + kx = f$$

- The **state space** form rewrites an Nth order differential equation as a system of N, 1st order differential equations.

k -> stiffness
c -> viscous damping
m -> mass
f -> external force
x -> mass position

State space takes an Nth order differential equation and converts it to a system of N first order differential equations.


LIGO
x(t)
k
m
f
c

Models – time domain

Equation of motion

$$m\ddot{x} + c\dot{x} + kx = f$$

- The **state space** form rewrites an Nth order differential equation as a system of N, 1st order differential equations.

k -> stiffness
c -> viscous damping
m -> mass
f -> external force
x -> mass position

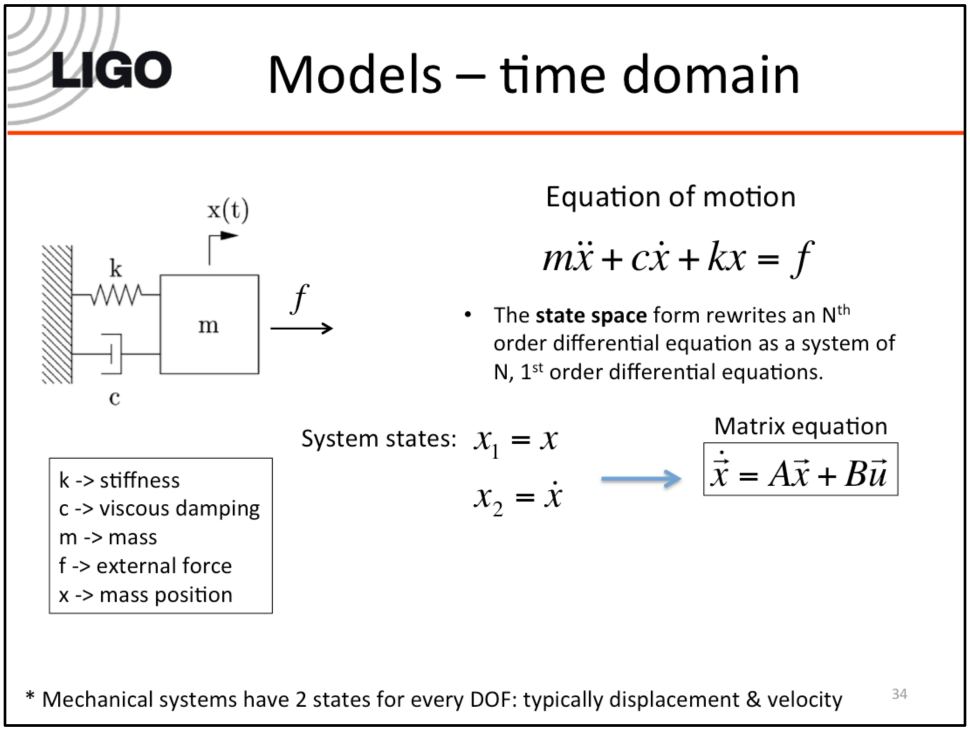
System states: $x_1 = x$ displacement

$x_2 = \dot{x}$ velocity

* Mechanical systems have 2 states for every DOF: typically displacement & velocity

33

This is done with a change of variables, defining the system ‘states’. There is no one correct choice for the state variables. A common choice for mechanical systems is to choose displacement as the first state, and velocity as the second.



Models – time domain

Equation of motion

$$m\ddot{x} + c\dot{x} + kx = f$$

- The **state space** form rewrites an Nth order differential equation as a system of N, 1st order differential equations.

System states: $x_1 = x$

$x_2 = \dot{x}$ $\longrightarrow$

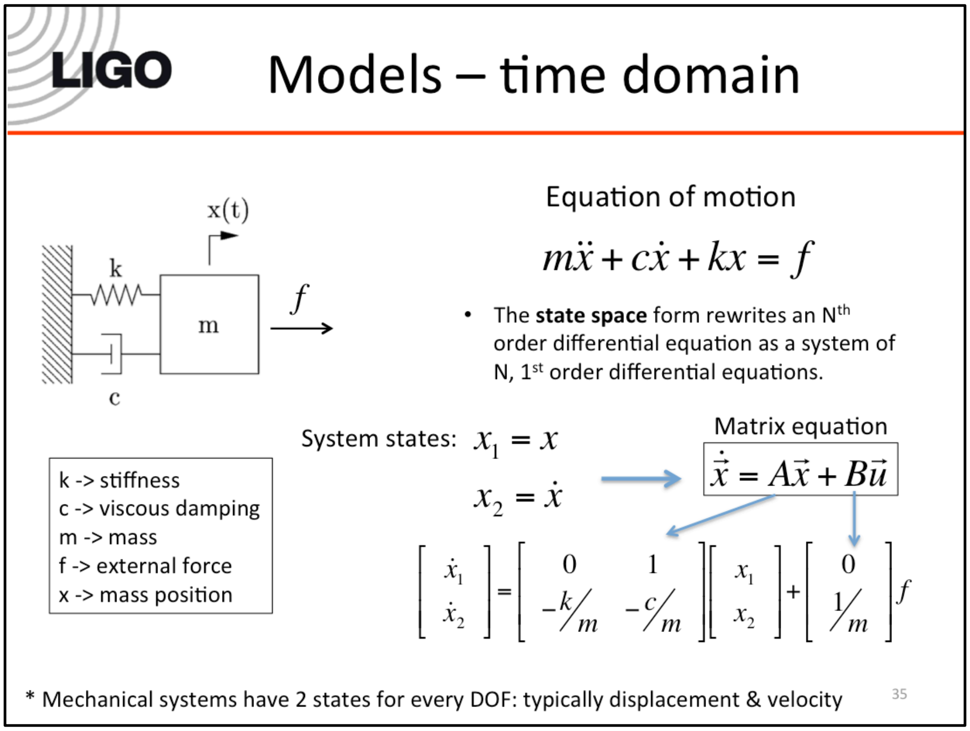
Matrix equation

$$\boxed{\dot{\vec{x}} = A\vec{x} + B\vec{u}}$$

k -> stiffness
c -> viscous damping
m -> mass
f -> external force
x -> mass position

* Mechanical systems have 2 states for every DOF: typically displacement & velocity

The new resulting first order differential equations are grouped together into a single matrix equation.



Models – time domain

Equation of motion

$$m\ddot{x} + c\dot{x} + kx = f$$

- The **state space** form rewrites an Nth order differential equation as a system of N, 1st order differential equations.

System states: $x_1 = x$

$x_2 = \dot{x}$

Matrix equation

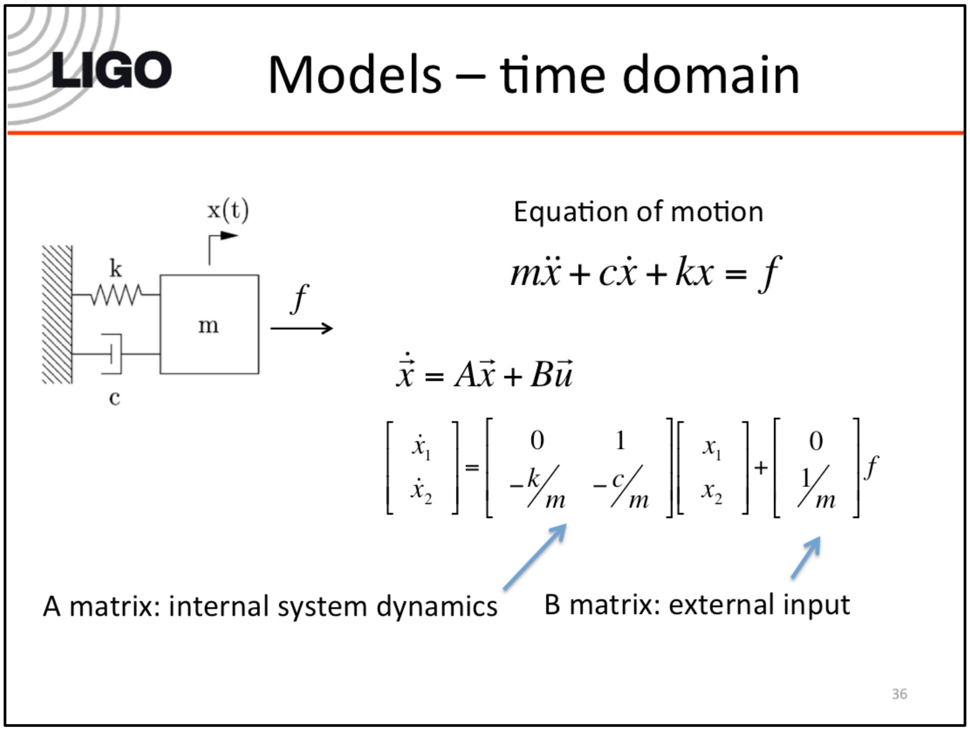
$$\boxed{\dot{\vec{x}} = A\vec{x} + B\vec{u}}$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k/m & -c/m \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} f$$

k -> stiffness
c -> viscous damping
m -> mass
f -> external force
x -> mass position

* Mechanical systems have 2 states for every DOF: typically displacement & velocity

This is what the matrices look like for this simple example.



Models – time domain

Equation of motion

$$m\ddot{x} + c\dot{x} + kx = f$$

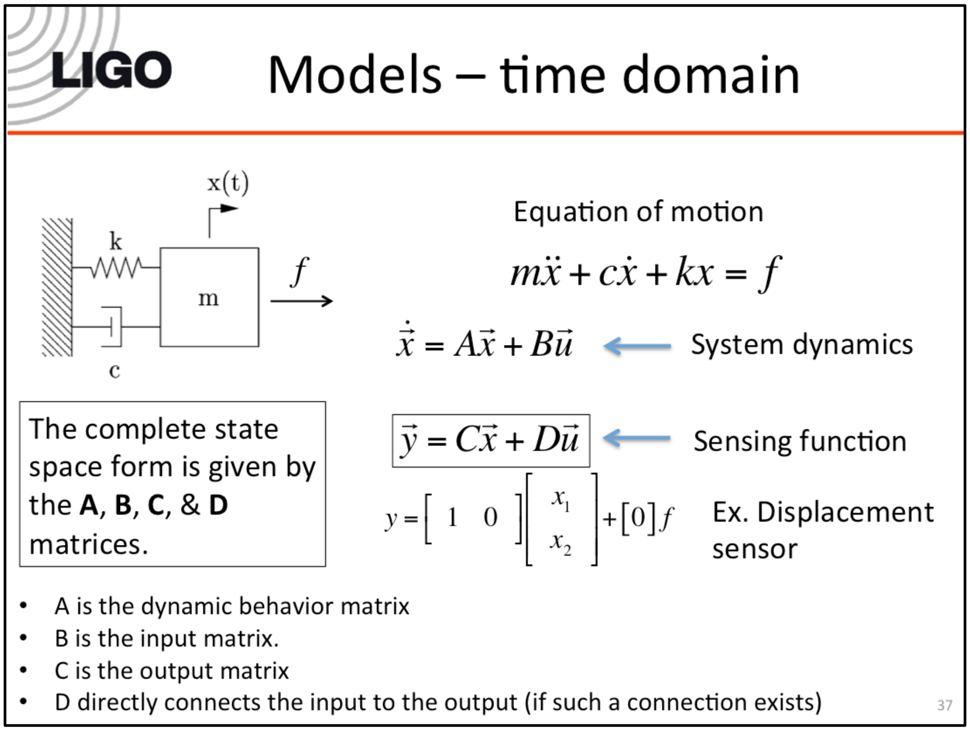
$$\dot{\vec{x}} = A\vec{x} + B\vec{u}$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k/m & -c/m \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} f$$

A matrix: internal system dynamics

B matrix: external input

The A matrix represents the dynamic behavior of the mass spring system. The B matrix describes the influence of the external input.



Models – time domain

Equation of motion

$$m\ddot{x} + c\dot{x} + kx = f$$

$$\dot{\vec{x}} = A\vec{x} + B\vec{u}$$ ← System dynamics

$$\boxed{\vec{y} = C\vec{x} + D\vec{u}}$$ ← Sensing function

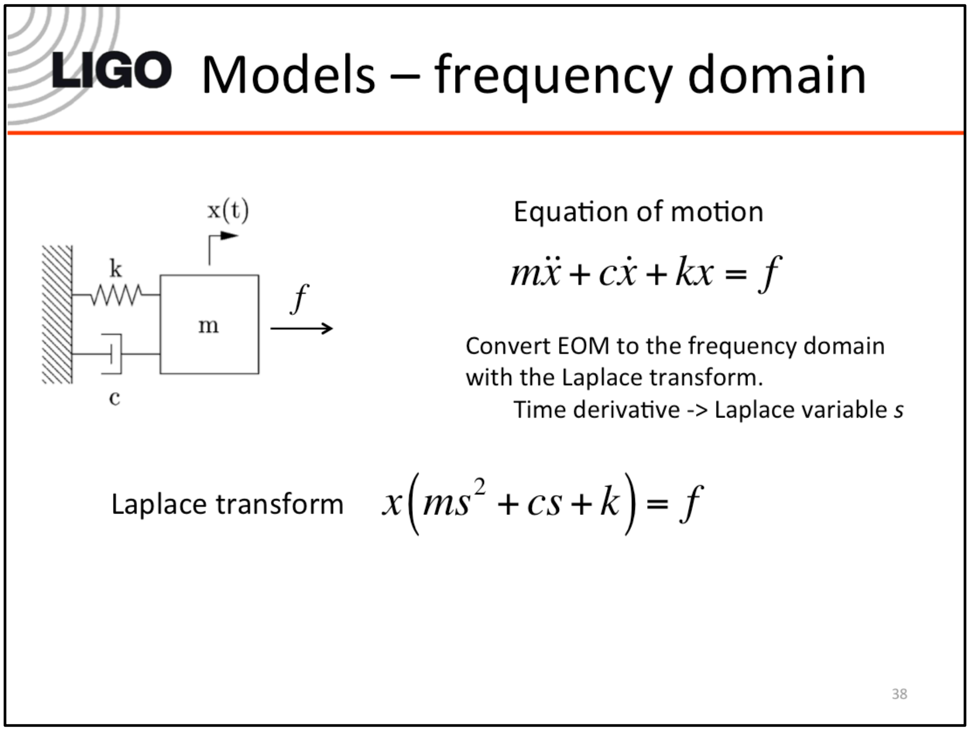
$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} f$$ Ex. Displacement sensor

The complete state space form is given by the **A**, **B**, **C**, & **D** matrices.

- A is the dynamic behavior matrix
- B is the input matrix.
- C is the output matrix
- D directly connects the input to the output (if such a connection exists)

37

There are also 2 output matrices, C and D, which can be thought of as a sensing function. If we have a displacement sensor on the mass, then the C and D are as follows. In most, but not all, cases the D matrix is zero. It is used in the case where your sensor also directly sees the input. An example of this would be if the input is ground displacement, and our displacement sensor measures the relative gap between the mass and the ground. In that case C = [1 0], as shown here, but D = -1, not 0.
All 4 matrices are required to complete the state space model.



LIGO

Models – frequency domain

Equation of motion

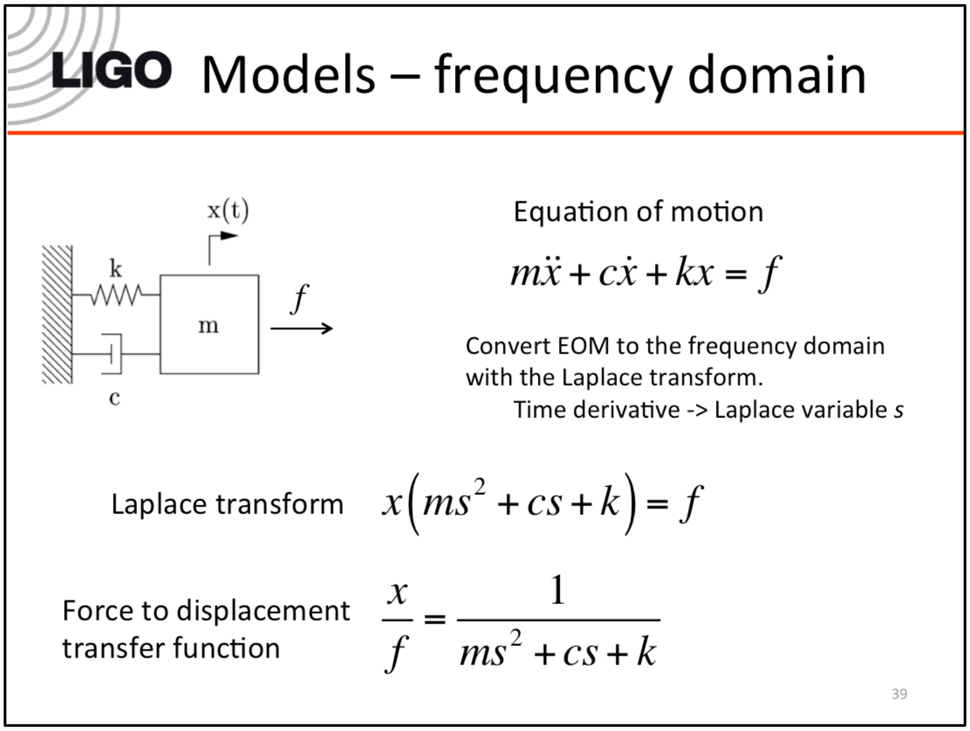
$$m\ddot{x} + c\dot{x} + kx = f$$

Convert EOM to the frequency domain with the Laplace transform.

Time derivative -> Laplace variable *s*

Laplace transform $x\left(ms^2 + cs + k\right) = f$

Frequency domain transfer functions are produced using the Laplace transform.



LIGO Models – frequency domain

Equation of motion

$$m\ddot{x} + c\dot{x} + kx = f$$

Convert EOM to the frequency domain with the Laplace transform.

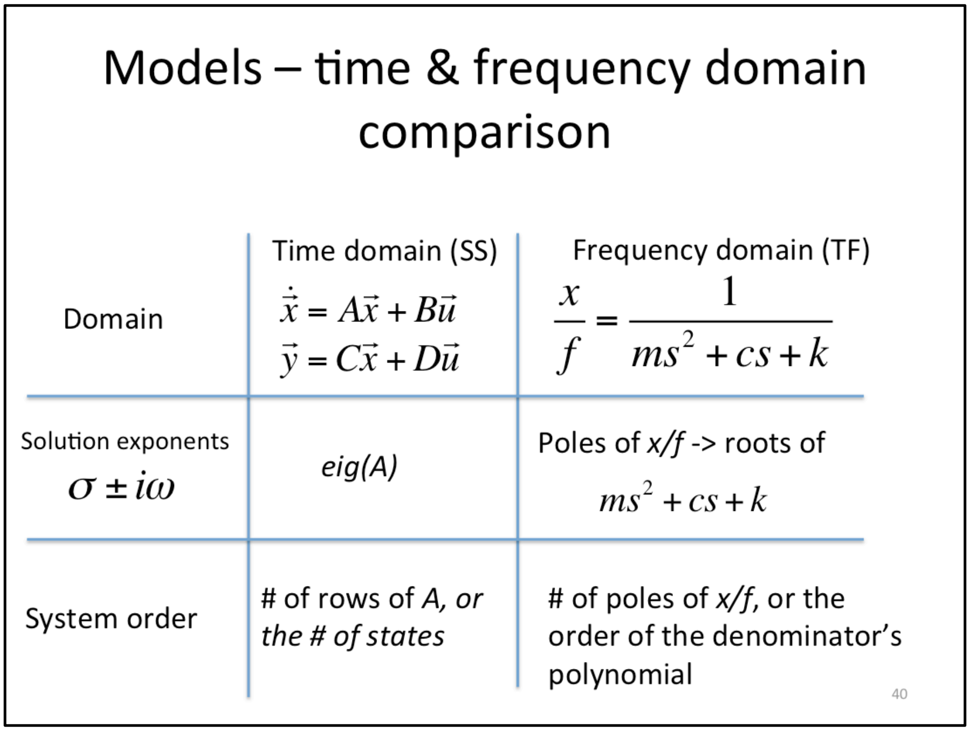
Time derivative -> Laplace variable *s*

Laplace transform $x\left(ms^2 + cs + k\right) = f$

Force to displacement transfer function $\dfrac{x}{f} = \dfrac{1}{ms^2 + cs + k}$

39

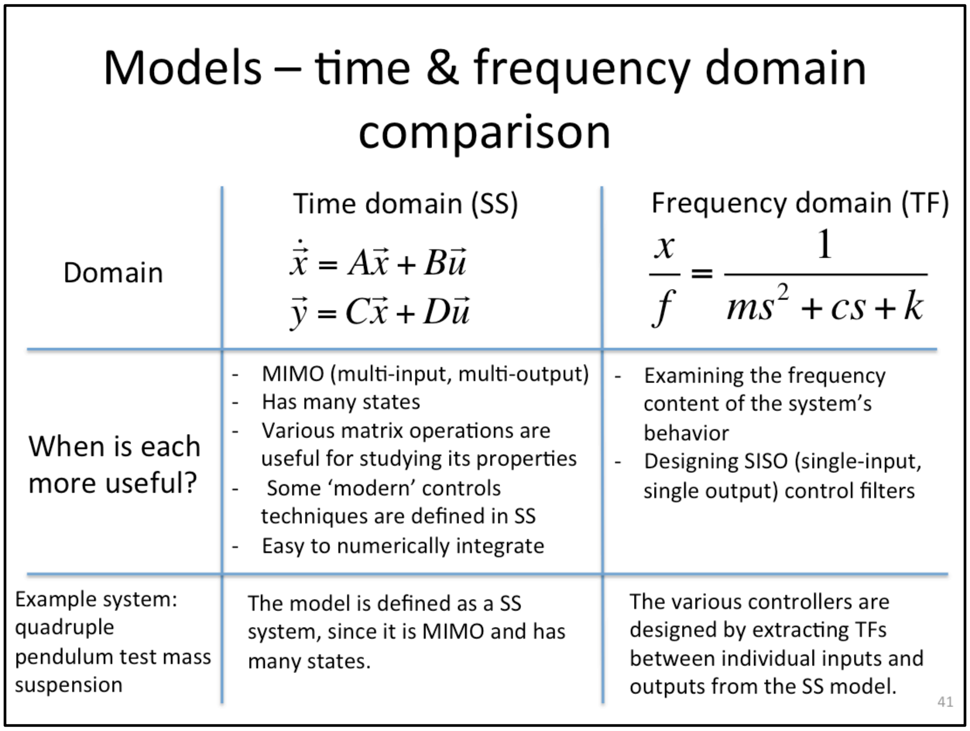
You can then algebraically solve for the relation between the input f, and the output x.



Models – time & frequency domain comparison

	Time domain (SS)	Frequency domain (TF)
Domain	$\dot{\vec{x}} = A\vec{x} + B\vec{u}$ $\vec{y} = C\vec{x} + D\vec{u}$	$\frac{x}{f} = \frac{1}{ms^2 + cs + k}$
Solution exponents $\sigma \pm i\omega$	*eig(A)*	Poles of *x/f* -> roots of $ms^2 + cs + k$
System order	# of rows of *A, or the # of states*	# of poles of *x/f*, or the order of the denominator's polynomial

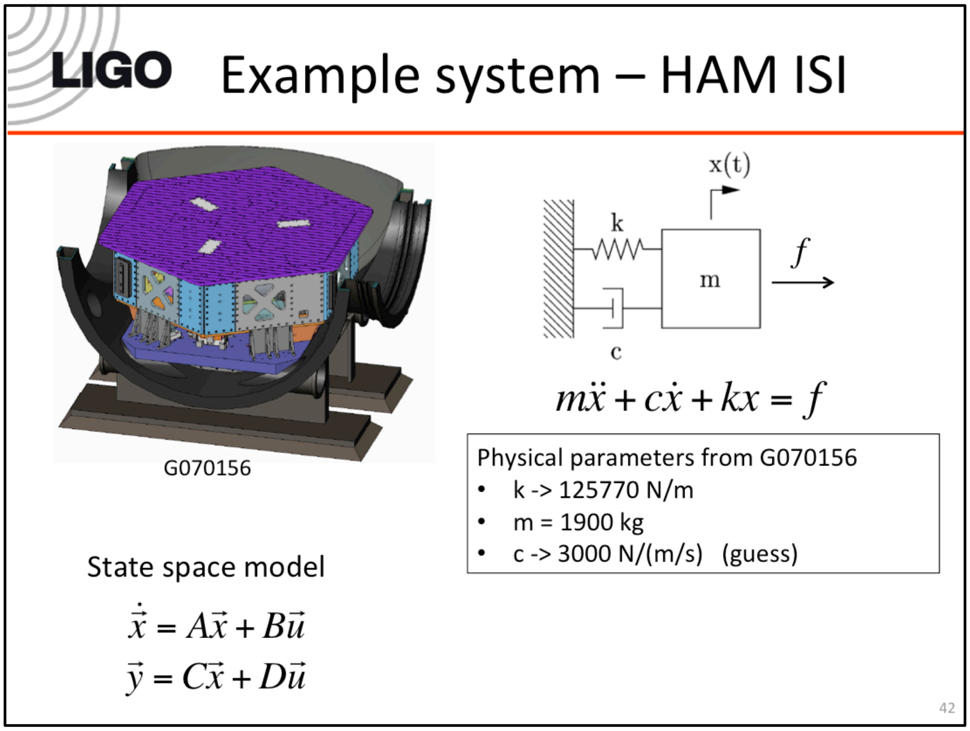
This table compares the properties of state space matrices to the frequency domain transfer functions. Note, the exponents to the equation of motion solution are given by the eigenvalues of the A matrix and the poles of the transfer function (or roots of its denominator). The order of the system is also the number of rows or columns of the matrix A (it is square), and the number of poles of the transfer function.



Models – time & frequency domain comparison

	Time domain (SS)	Frequency domain (TF)
Domain	$\dot{\vec{x}} = A\vec{x} + B\vec{u}$ $\vec{y} = C\vec{x} + D\vec{u}$	$\dfrac{x}{f} = \dfrac{1}{ms^2 + cs + k}$
When is each more useful?	- MIMO (multi-input, multi-output) - Has many states - Various matrix operations are useful for studying its properties - Some 'modern' controls techniques are defined in SS - Easy to numerically integrate	- Examining the frequency content of the system's behavior - Designing SISO (single-input, single output) control filters
Example system: quadruple pendulum test mass suspension	The model is defined as a SS system, since it is MIMO and has many states.	The various controllers are designed by extracting TFs between individual inputs and outputs from the SS model.

Each is useful in various contexts, as discussed here. Often, we define a model in state space form, and then examine its properties by looking at the transfer functions. This is done seamlessly in Matlab.



Example system – HAM ISI

G070156

$$m\ddot{x} + c\dot{x} + kx = f$$

Physical parameters from G070156

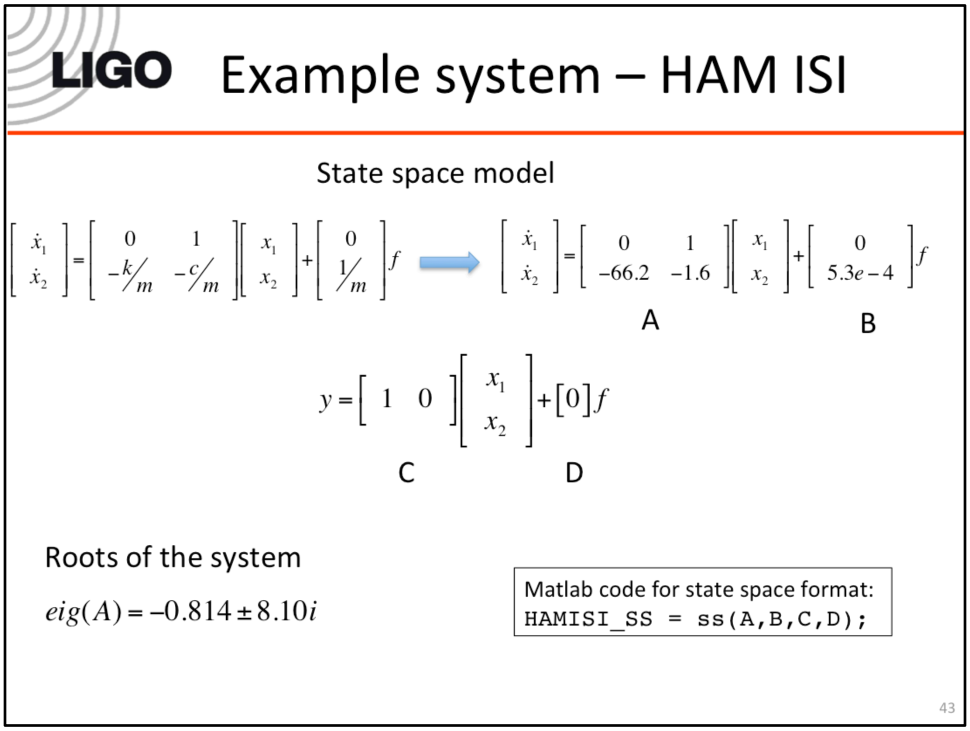
- k -> 125770 N/m
- m = 1900 kg
- c -> 3000 N/(m/s) (guess)

State space model

$$\dot{\vec{x}} = A\vec{x} + B\vec{u}$$

$$\vec{y} = C\vec{x} + D\vec{u}$$

Let’s look at the example of the HAM-ISI. It’s physical parameters are given in G070156.



Example system – HAM ISI

State space model

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k/m & -c/m \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} f \Rightarrow \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \underset{A}{\begin{bmatrix} 0 & 1 \\ -66.2 & -1.6 \end{bmatrix}} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \underset{B}{\begin{bmatrix} 0 \\ 5.3e-4 \end{bmatrix}} f$$

$$y = \underset{C}{\begin{bmatrix} 1 & 0 \end{bmatrix}} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \underset{D}{\begin{bmatrix} 0 \end{bmatrix}} f$$

Roots of the system

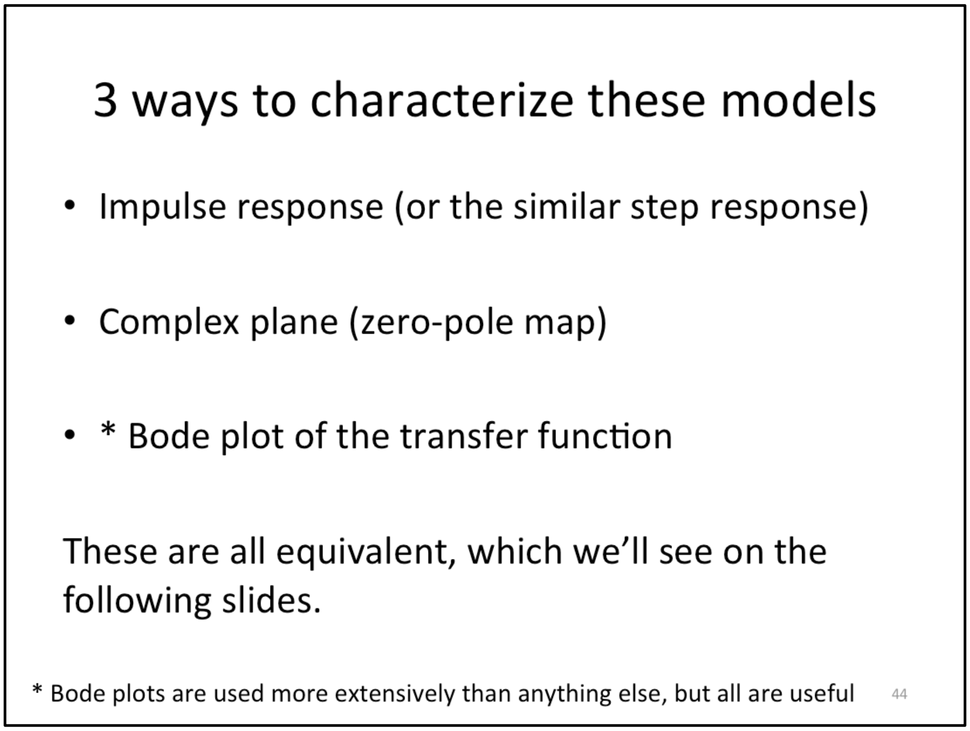
$$eig(A) = -0.814 \pm 8.10i$$

Matlab code for state space format:

```
HAMISI_SS = ss(A,B,C,D);
```

43

Plugging in the parameters, we get these state space matrices. The matlab code eig function on the A matrix gives you the system poles. The ss function generates the state space system.



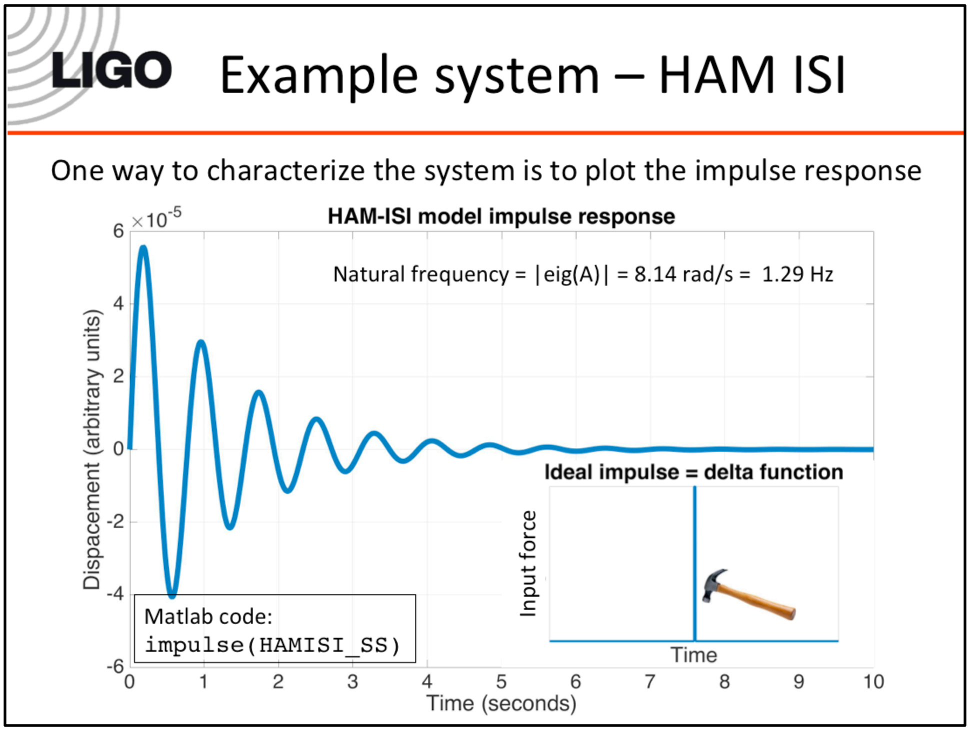
3 ways to characterize these models

- Impulse response (or the similar step response)
- Complex plane (zero-pole map)
- * Bode plot of the transfer function

These are all equivalent, which we'll see on the following slides.

* Bode plots are used more extensively than anything else, but all are useful 44

There are 3 common ways to characterize these models, the impulse response, the complex plane, and bode plots. All are equivalent, but the bode plots are used the most. We'll take a look at all 3 and see how they relate to each other.

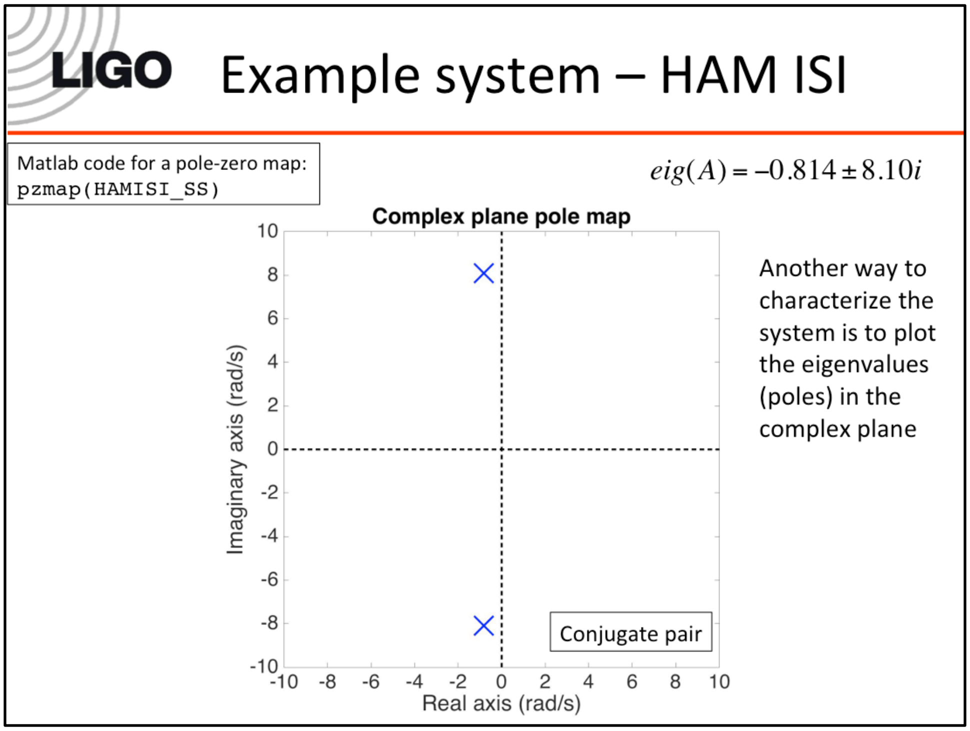


Example system – HAM ISI

One way to characterize the system is to plot the impulse response

Matlab code:
```
impulse(HAMISI_SS)
```

The impulse response shows how the systems responds when you apply a delta function to the input, that is an input that is infinite in amplitude, but also infinitely short in time, so that the area of the curve equals 1. Matlab will generate these plots with the impulse function. The system oscillates at its natural frequency. This frequency is the absolute value of the system poles.



LIGO

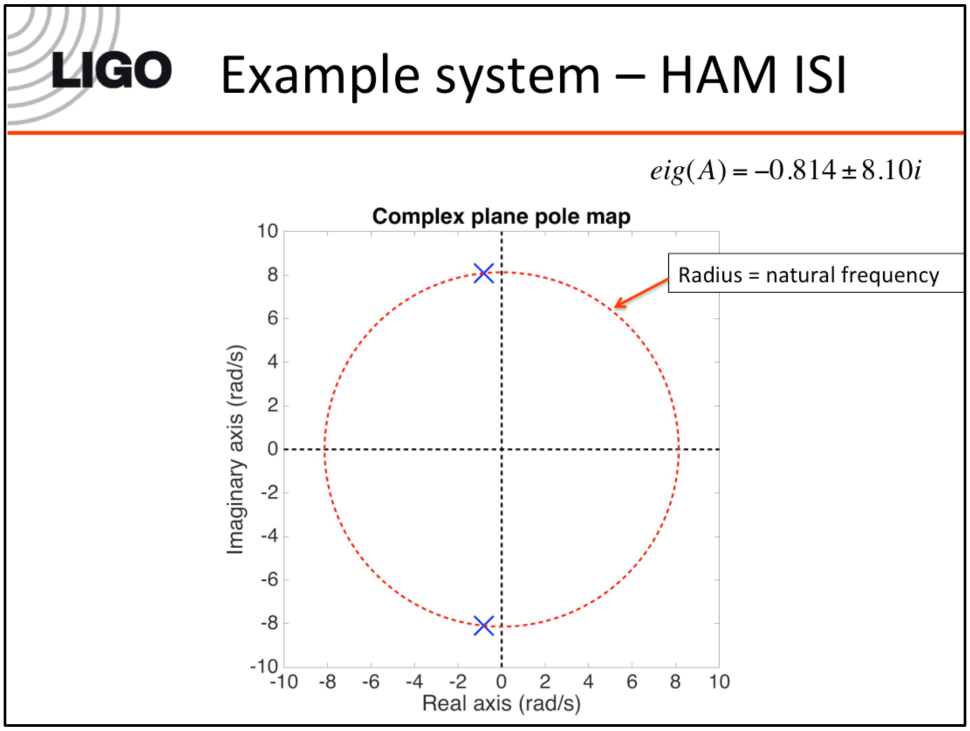
Example system – HAM ISI

Matlab code for a pole-zero map:
`pzmap(HAMISI_SS)`

$$eig(A) = -0.814 \pm 8.10i$$

Another way to characterize the system is to plot the eigenvalues (poles) in the complex plane

The complex plane plots the system poles and zeros with respect to real and imaginary axes. The matlab function pzmap generates these plots. Note that complex poles and zeros are always conjugate pairs (for linear systems). Real valued poles and zeros do not need to come in pairs. In this example we just have poles, denoted by the X's. Zeros would be denoted by O's.

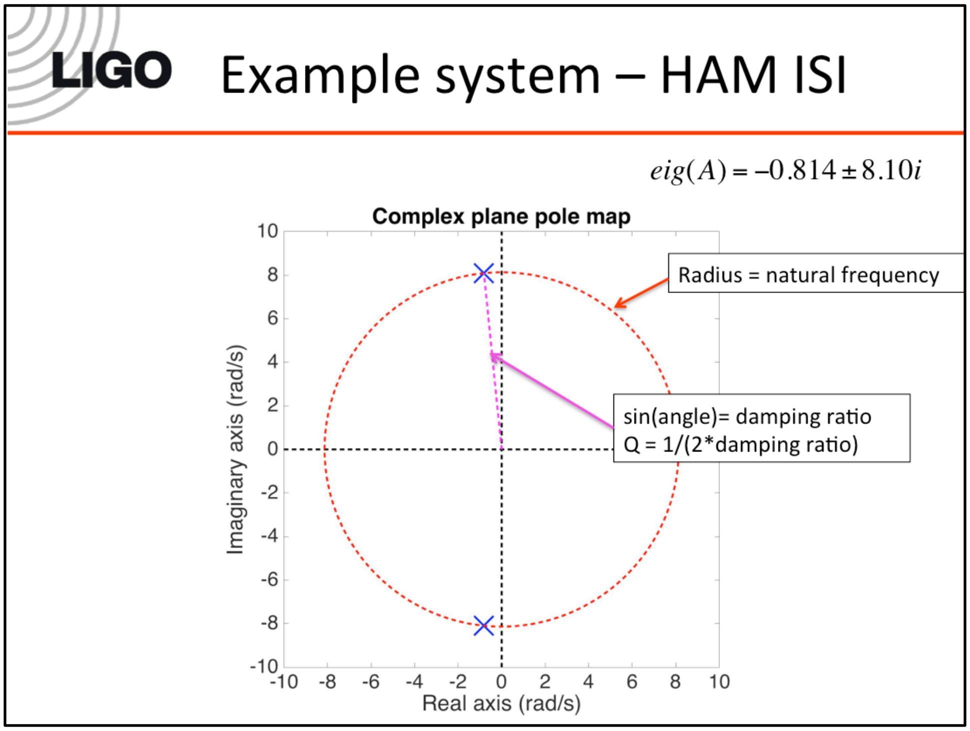


Example system – HAM ISI

$eig(A) = -0.814 \pm 8.10i$

Radius = natural frequency

The radius of the circle these poles and zeros lie on tells you their frequency.



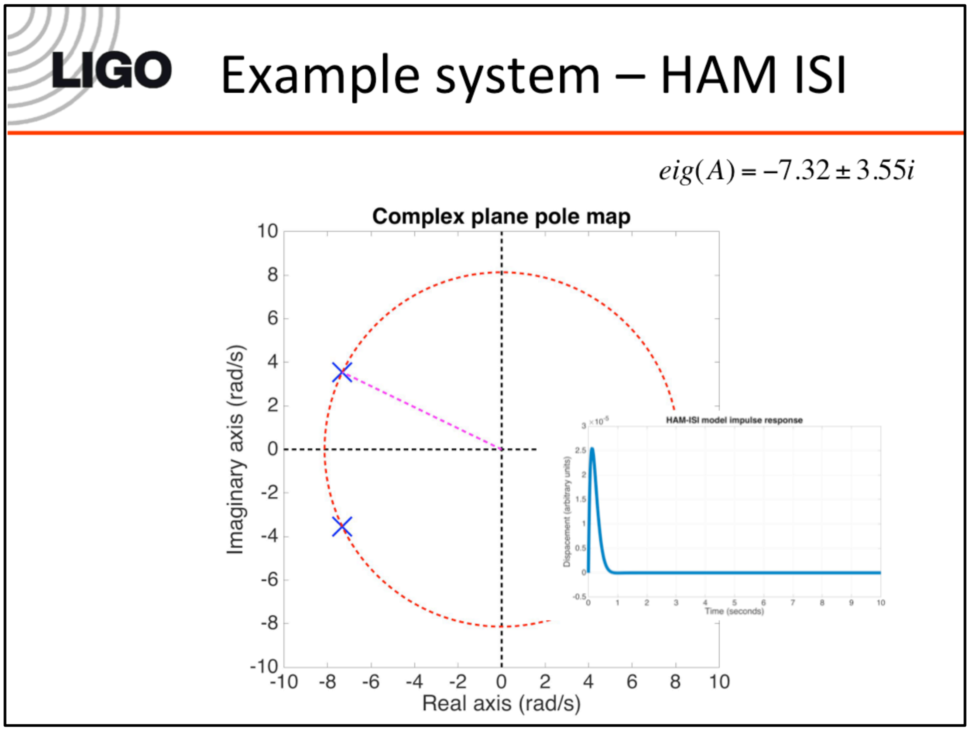
Example system – HAM ISI

$$eig(A) = -0.814 \pm 8.10i$$

Radius = natural frequency

sin(angle)= damping ratio
Q = 1/(2*damping ratio)

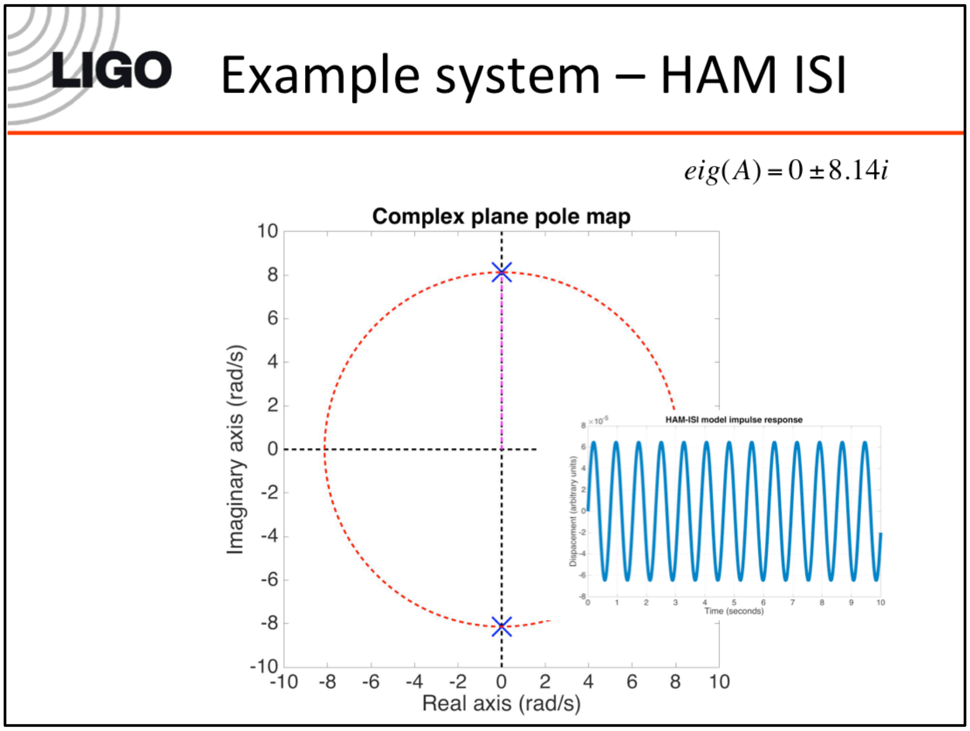
The angle they make with the imaginary axis is the damping. Specifically, the damping ratio is the sine of the angle. The Quality factor is 1 / (2 * sin(angle)).



Example system – HAM ISI

$$eig(A) = -7.32 \pm 3.55i$$

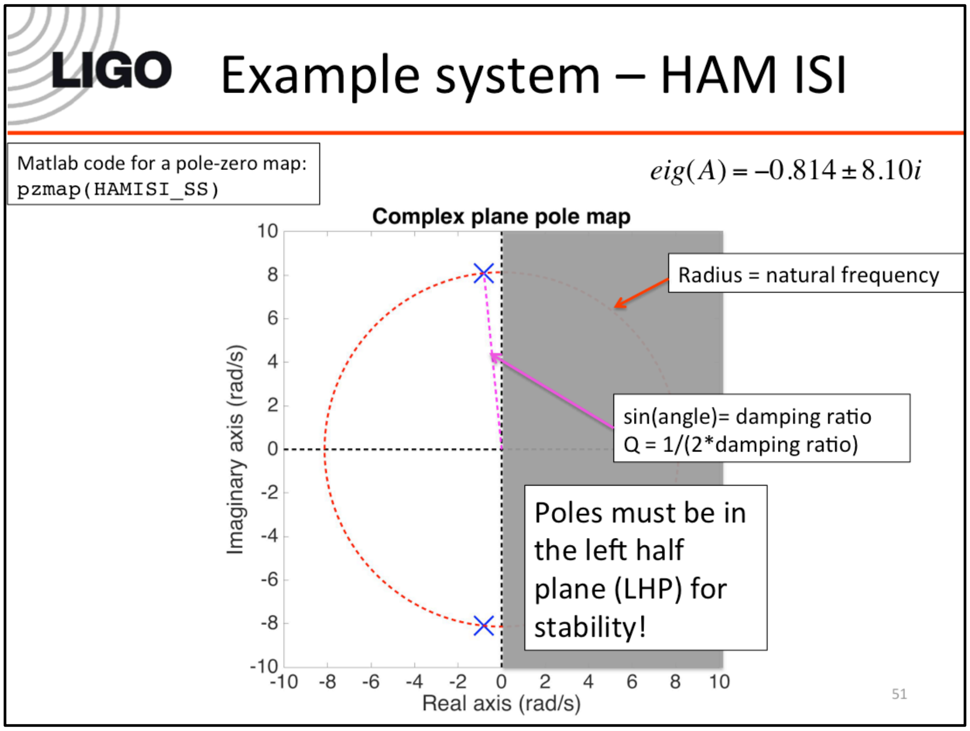
This is an example of the system with will damped poles.



Example system – HAM ISI

$eig(A) = 0 \pm 8.14i$

This is an example with zero damping.



Example system – HAM ISI

Matlab code for a pole-zero map:
`pzmap(HAMISI_SS)`

$$eig(A) = -0.814 \pm 8.10i$$

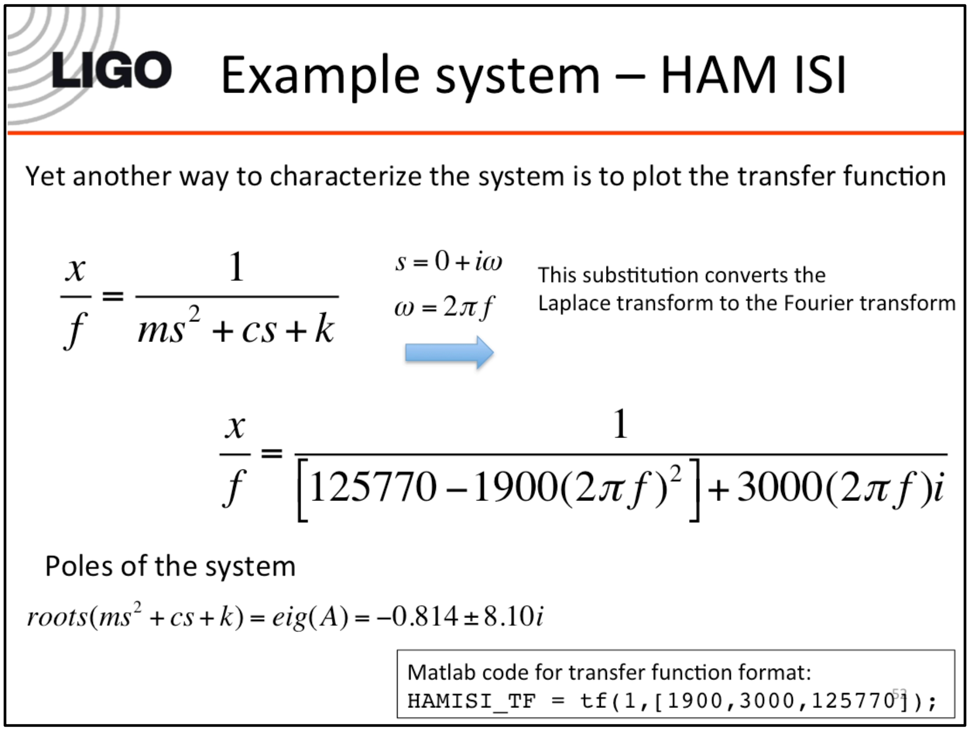
Complex plane pole map

Radius = natural frequency

sin(angle)= damping ratio
Q = 1/(2*damping ratio)

Poles must be in the left half plane (LHP) for stability!

For a system to be stable, all its poles must lie in the Left-Half-Plane (LHP).



Example system – HAM ISI

Yet another way to characterize the system is to plot the transfer function

$$\frac{x}{f}=\frac{1}{ms^2+cs+k} \qquad \begin{matrix} s=0+i\omega \\ \omega=2\pi f \end{matrix} \quad \Rightarrow$$

This substitution converts the Laplace transform to the Fourier transform

$$\frac{x}{f}=\frac{1}{\left[125770-1900(2\pi f)^2\right]+3000(2\pi f)i}$$

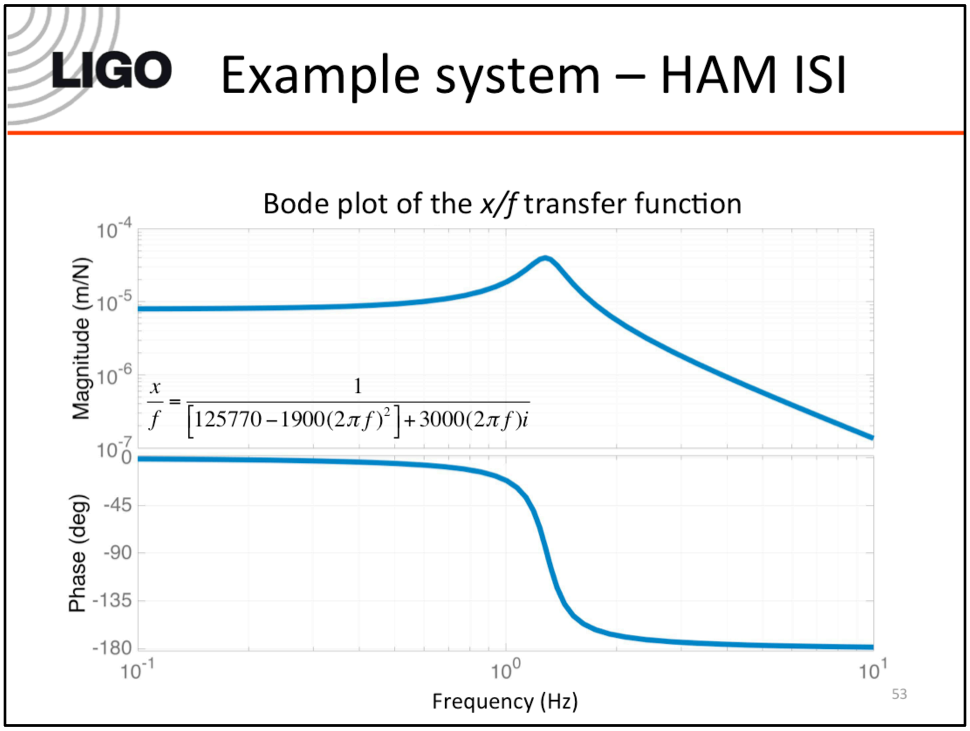
Poles of the system

$$roots(ms^2+cs+k)=eig(A)=-0.814\pm 8.10i$$

Matlab code for transfer function format:

```
HAMISI_TF = tf(1,[1900,3000,125770]);
```

The most typical way to look at a system is to plot the Bode plot of the transfer function. This is done by setting the Laplace variable s = 0 + i*omega, where omega is 2*pi*frequency. In general, s has both real and imaginary parts. For the bode plot however, we only need the imaginary part. Note, discarding the real part of s converts the Laplace transform to the Fourier transform. So, the bode plot is showing us the systems Fourier transform. There is a little bit more discussion on this in the backup slides for Lecture 1.



Example system – HAM ISI

Bode plot of the *x/f* transfer function

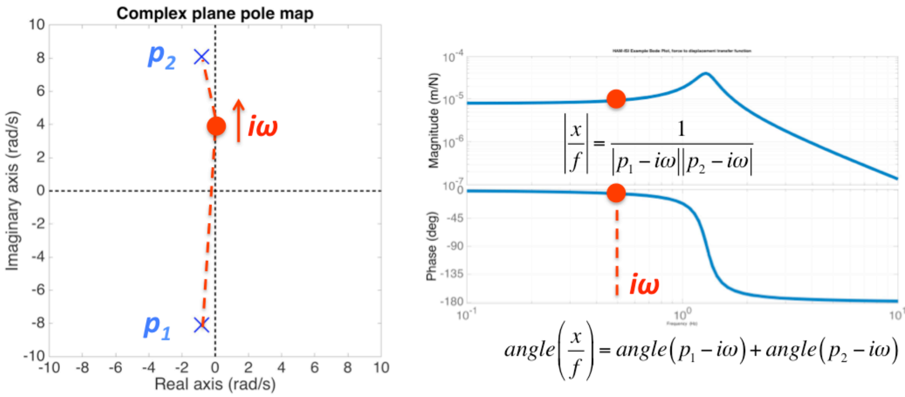
$$\frac{x}{f}=\frac{1}{\left[125770-1900(2\pi f)^{2}\right]+3000(2\pi f)i}$$

Here is the Bode plot for our example. The plot shows the magnitude of the transfer function on top, and the phase below.

LIGO

Pole map vs Bode plot

$$\frac{x}{f}=\frac{1}{\left[125770-1900(2\pi f)^{2}\right]+3000(2\pi f)i}$$



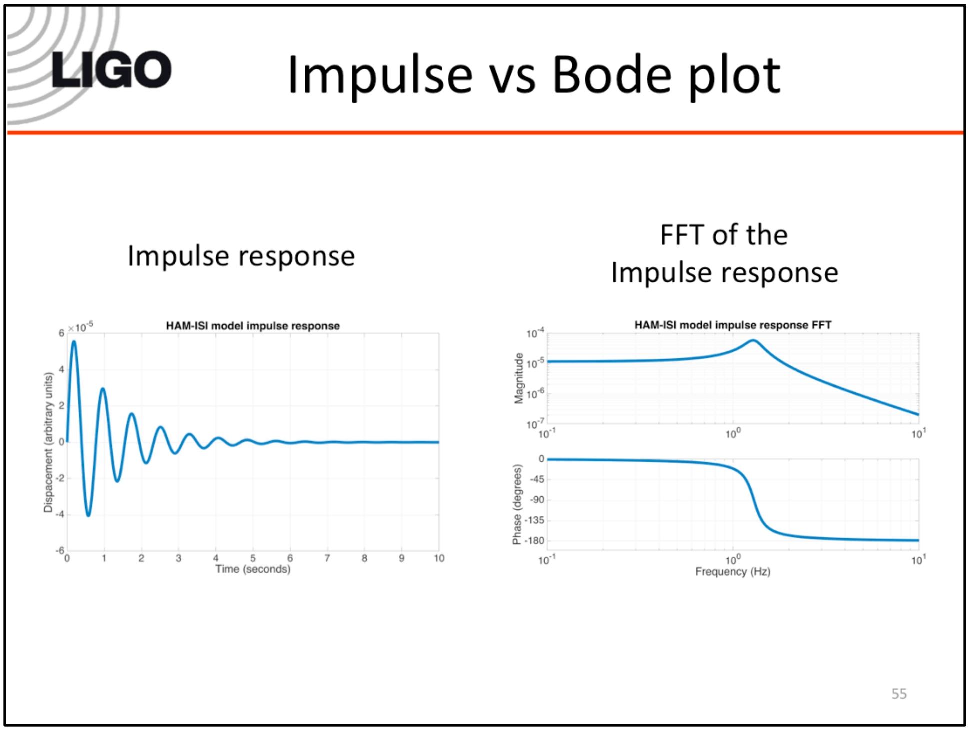
$$\left|\frac{x}{f}\right|=\frac{1}{\left|p_1-i\omega\right|\left|p_2-i\omega\right|}$$

$$angle\left(\frac{x}{f}\right)=angle\left(p_1-i\omega\right)+angle\left(p_2-i\omega\right)$$

Frequency (Hz)

54

The bode plot is equivalent to the pole-zero map in the complex plane. This can be seen by moving up the positive imaginary axis with s = i*omega. The magnitude of the bode plot results from the distances to the poles and zeros for a given frequency. The magnitude is inversely proportional to duistance for the poles, and proportional for zeros. You can see then how lightly damped poles produce large resonance features, because they are close to the imagniary axis, so the distance gets very small as you pass their frequency. The phase then comes from summing the angels to each pole and zero (with the proper signs).



LIGO

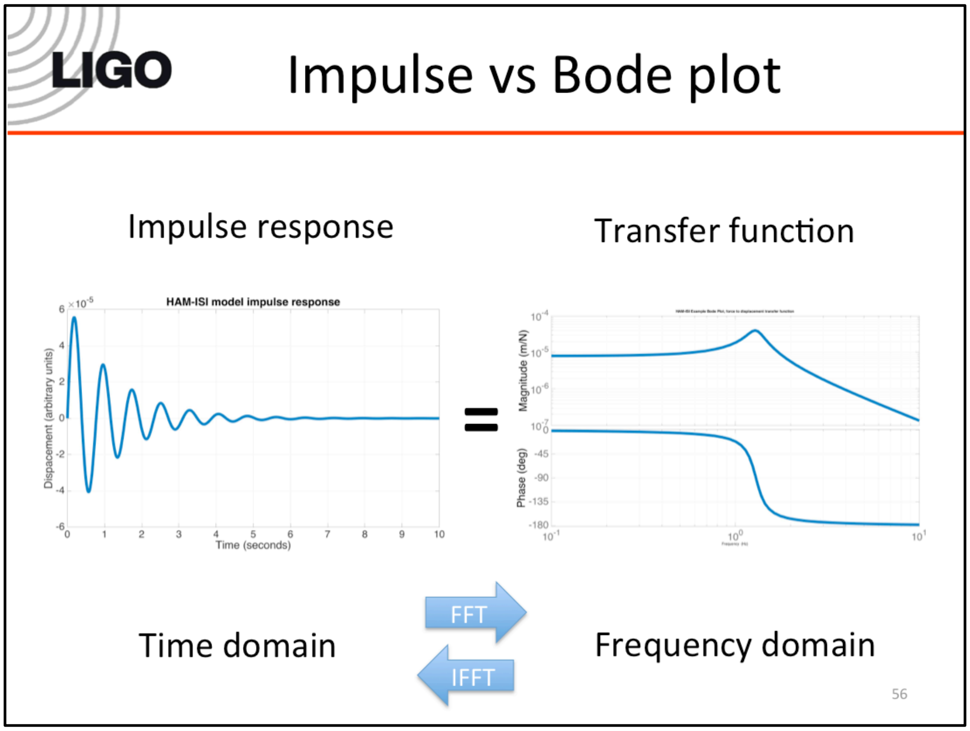
Impulse vs Bode plot

Impulse response

FFT of the Impulse response

55

The bode plot, it turns out, is exactly the Fourier transform of the impulse. Here I've simply taken the transform of the impulse you saw earlier, which reproduces the bode plot.



Impulse vs Bode plot

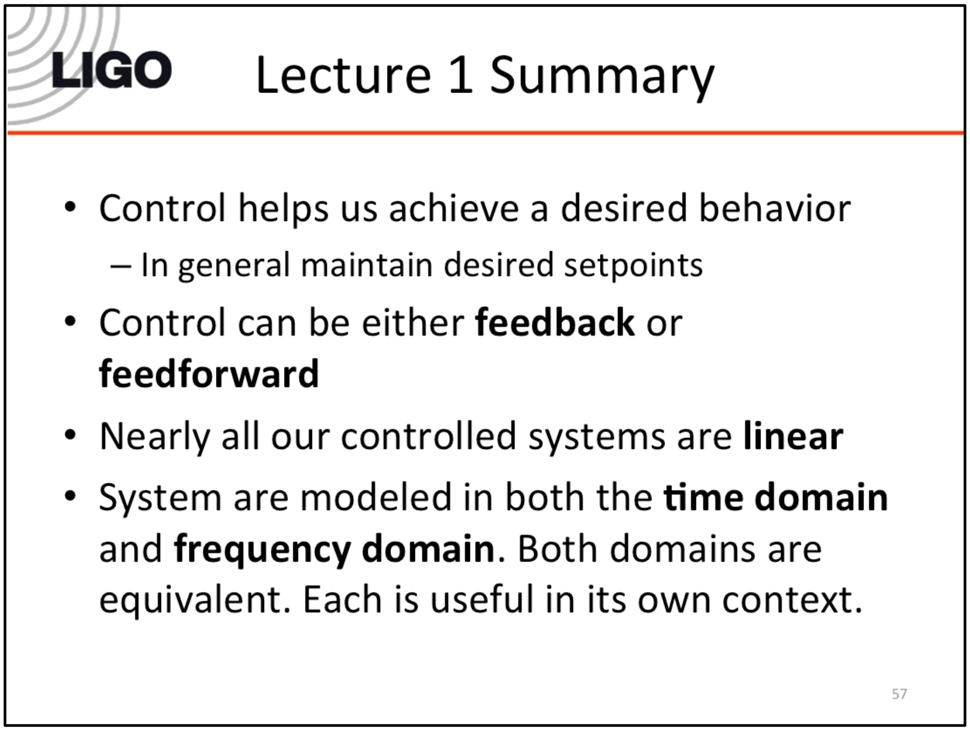
Impulse response

Transfer function

Time domain

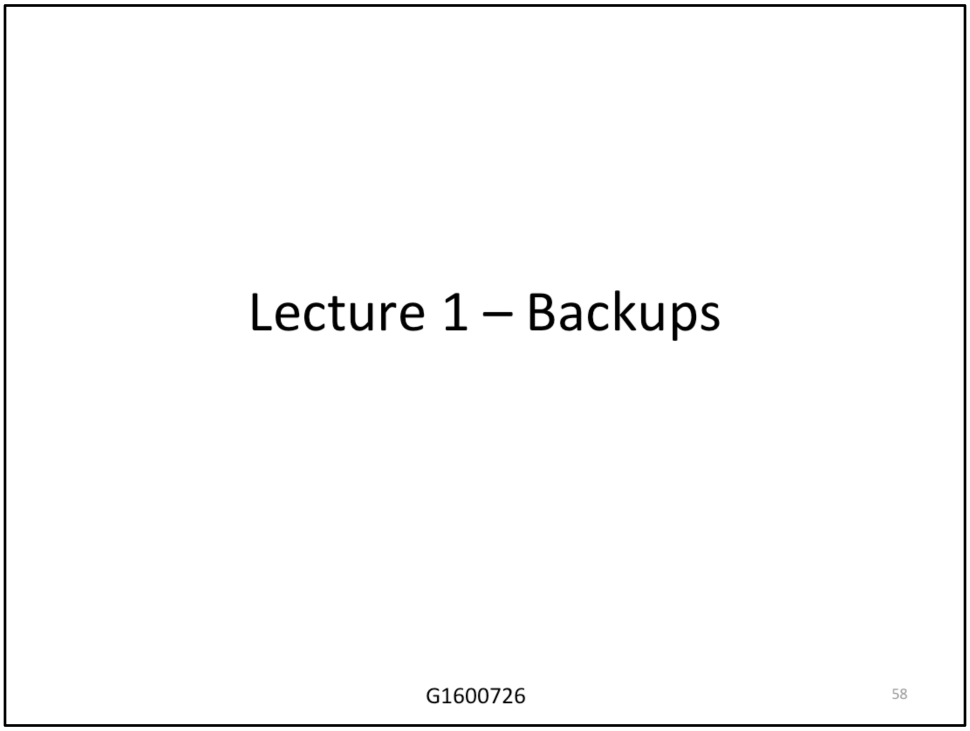
Frequency domain

In fact, you can measure a transfer function by measuring its impulse response. This technique is often used to characterize the suspension cages.

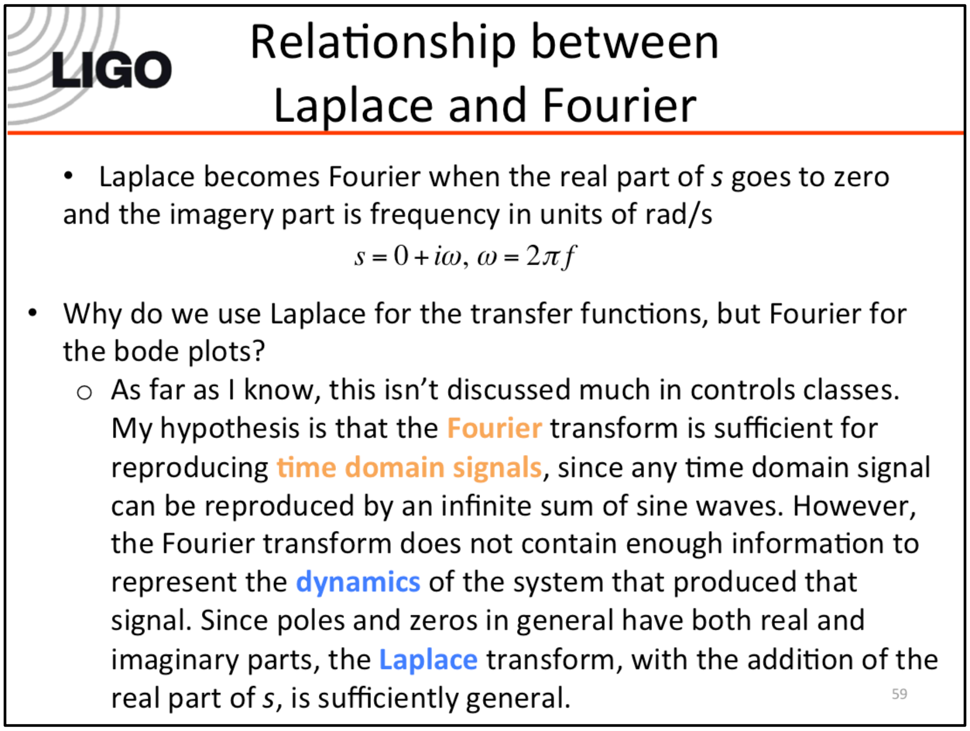


Lecture 1 Summary

- Control helps us achieve a desired behavior
  - In general maintain desired setpoints
- Control can be either **feedback** or **feedforward**
- Nearly all our controlled systems are **linear**
- System are modeled in both the **time domain** and **frequency domain**. Both domains are equivalent. Each is useful in its own context.



Lecture 1 – Backups

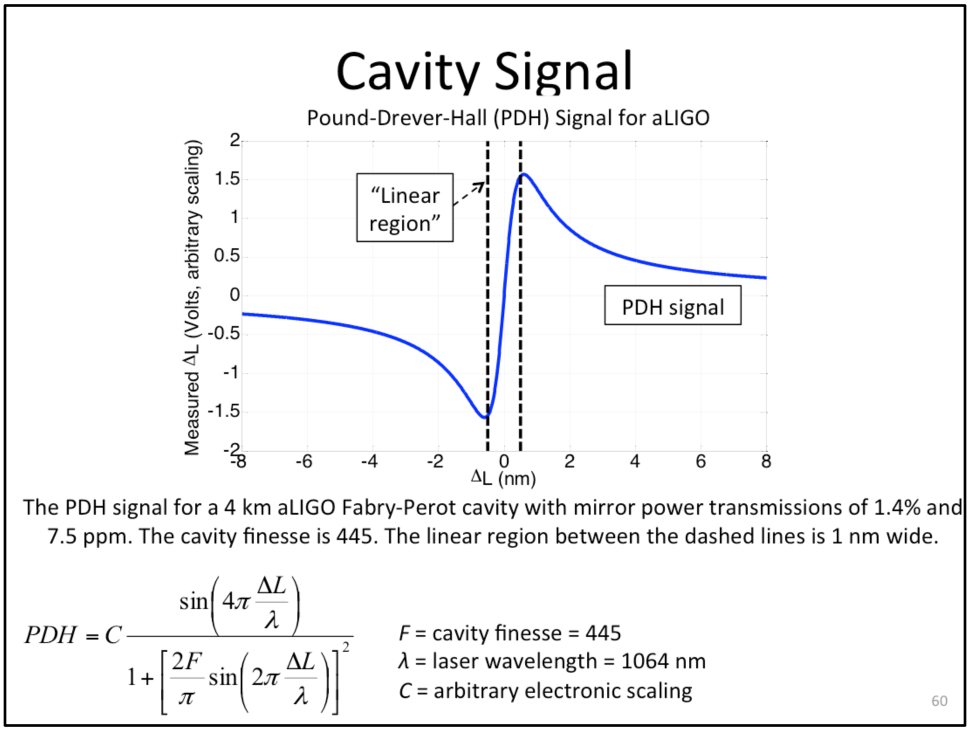


Relationship between Laplace and Fourier

- Laplace becomes Fourier when the real part of *s* goes to zero and the imagery part is frequency in units of rad/s

$$s = 0 + i\omega,\ \omega = 2\pi f$$

- Why do we use Laplace for the transfer functions, but Fourier for the bode plots?
  - As far as I know, this isn't discussed much in controls classes. My hypothesis is that the **Fourier** transform is sufficient for reproducing **time domain signals**, since any time domain signal can be reproduced by an infinite sum of sine waves. However, the Fourier transform does not contain enough information to represent the **dynamics** of the system that produced that signal. Since poles and zeros in general have both real and imaginary parts, the **Laplace** transform, with the addition of the real part of *s*, is sufficiently general.



Cavity Signal

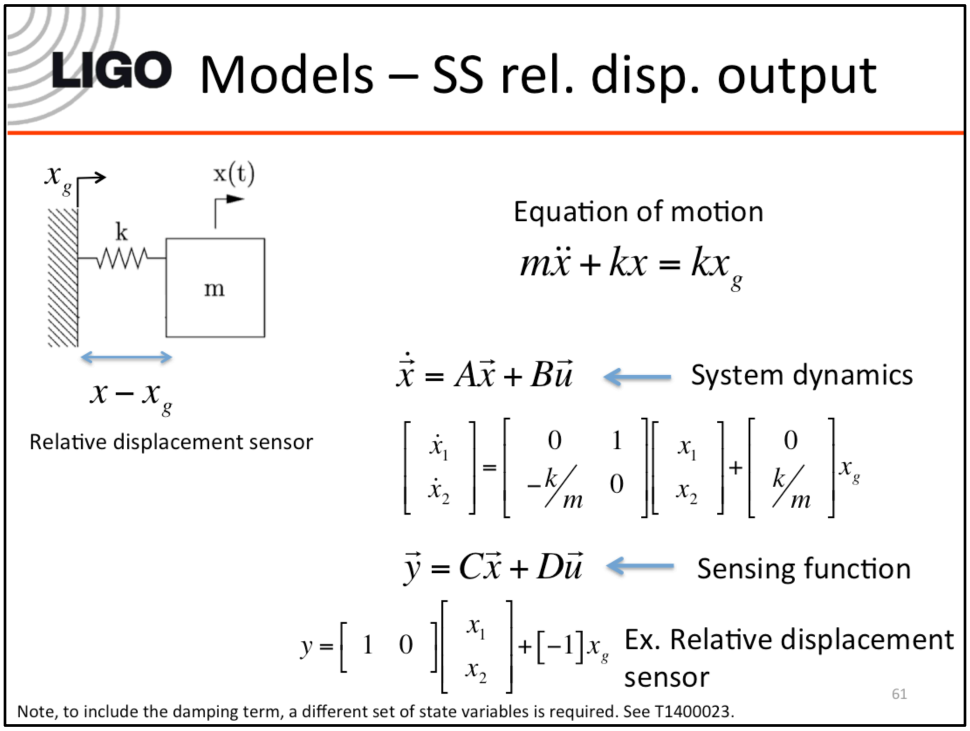
Pound-Drever-Hall (PDH) Signal for aLIGO
Measured ΔL (Volts, arbitrary scaling)
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-8
-6
-4
-2
0
2
4
6
8
ΔL (nm)
“Linear region”
PDH signal

The PDH signal for a 4 km aLIGO Fabry-Perot cavity with mirror power transmissions of 1.4% and 7.5 ppm. The cavity finesse is 445. The linear region between the dashed lines is 1 nm wide.

$$PDH = C\frac{\sin\left(4\pi\frac{\Delta L}{\lambda}\right)}{1+\left[\frac{2F}{\pi}\sin\left(2\pi\frac{\Delta L}{\lambda}\right)\right]^2}$$

F = cavity finesse = 445
λ = laser wavelength = 1064 nm
C = arbitrary electronic scaling

The PDH signal is only linear at the limit where the error signal, Delta-L, goes to zero. Deviations from this from the linear regime can cause upconversion in the interferometer. If the deviations are large enough, the cavity loses lock, because the slope (gain of the loop) changes dramatically in magnitude and sign.



LIGO Models – SS rel. disp. output

Relative displacement sensor

Equation of motion

$$m\ddot{x} + kx = kx_g$$

$$\dot{\vec{x}} = A\vec{x} + B\vec{u}$$ ← System dynamics

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k/m & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ k/m \end{bmatrix} x_g$$

$$\vec{y} = C\vec{x} + D\vec{u}$$ ← Sensing function

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} -1 \end{bmatrix} x_g$$ Ex. Relative displacement sensor

61

Note, to include the damping term, a different set of state variables is required. See T1400023.

This shows the D matrix of a state space system when the output is given by a relative displacement sensor between the mass and the ground. Note, if the damping term is included between the ground and the mass velocity is no longer a useful state variable. See T1400023 for the proper choice of state variables in this case.