Lecture 3: Linear Systems Analysis

Linear systems can be solved using analytic and computational tools that are not available for nonlinear systems. The analytic tool most commonly used for linear systems is Laplace Transforms, which you should have already learned. We will review the analytic method to transform a system, because many computational tools make use of the transformed version of the linear system, which we call the transfer function. Computational tools for linear systems, such as Matlab's linear systems toolbox and SIMULINK, are especially helpful when you want to determine the response to complicated time-variant inputs, which can be clumsy with ODE solvers. In order to verify our simulations, we will also review some of the simpler linear systems analysis methods, such as calculating stability, time constants, and steady state solutions from the Transformed equation. On Wednesday we will learn to a related approach using linear algebra to solve the same systems.

Review of Laplace Transforms:

If you haven't learned Laplace transforms or don't remember them at all, you should read Khoo, chapter 2, which describes both the method and its justification. Here is the brief overview:

Linear Systems Definitions Reminder.

This method relies on the linear properties of the system so should only be applied to linear ODEs. A **linear system** has a linear combination of dependent variables (recall that you can think of a derivative of a variable as a variable) on each side of the equation. That is, each term has only parameters multiplied by a single variable.

Superposition. Recall that the response of a linear system can be expressed as the sum of the **Complementary Solution**, which is the response to initial **C**onditions, and the **Particular Solution**, which is the response to any input, or **P**ushing function, that occurs over time. If a system has multiple dependent variables, the Initial condition is a vector of initial conditions for each variable, and the pushing function is a vector of functions of the independent variable (usually time).

Solution Technique

You may have learned my "ETSIE" process of model solving using Laplace Transforms: **E**quations, **T**ransform, **S**implify, **I**nverse Transform, and **E**rror Check. If you use numeric tools to solve the model, however, you will replace the Inverse Transform step with a numeric method, so you may want to remember the "ETSANE" (ETernally SANE, rather than insane) process instead: **E**quations, **T**ransform, **S**implify, **A**nalyze**, N**umeric Solution, and **E**rror Check.

E. Equations. You may start with the simplified system equation from the "DIESE" process of last week, or you may prefer to identify an unsimplified or partially simplified set of equations, and perform the simplification after you transform the system.

T. **T**ransform the system. To transform a system, we define a new variable (often the capital) that is the Laplace transform of each dependent variable. For example, we may define $X(s) = L\big(x(t)\big) = \int_0^\infty x(t)e^{-st}dt$. Transforming the system using this definition is convenient since it converts each differential equation into an algebraic equation. For this class, we will not solve these Laplace integrals, but will simply use look-up tables since all the ones we need have already been solved. While you may encounter some that you need to look up yourselves, we provide the most useful transforms here:

- $L\big(x(t)\big) = X(s)$
- $L\left(\frac{dx}{dt}\right) = sX(s) - x(0)$
- $L\left(\frac{d^2x}{dt^2}\right) = s^2X(s) - sx(0) - \frac{dx}{dt}(0)$
- $L\left(\int_0^t x d\tau\right) = \frac{1}{s}X(s)$
- $L\big(\delta(t)\big) = 1$ (impulse function; infinitely high at t =0)
- $L\big(\phi(t)\big) = \frac{1}{s}$ (step function; steps from 0 to 1 at t = 0).
- $L\big(t\phi(t)\big) = 1/s^2$ (linear increase starting at t =0).
- $L(e^{-at}) = \frac{1}{s+a}$
- $L\big(ax(t) + by(t)\big) = aX(s) + bY(s)$

S. **S**implify the system. Your goal at this point is to express your system with a function H(s) that is a ratio of two polynomials in s, which we refer to as the numerator and denominator. The denominator of the transfer function is called the *characteristic equation*. If you simplified the system to a single ODE prior to transforming, then the single transformed equation can be manipulated to obtain the transfer function H(s) that relates the dependent variables (the output) to the inputs and initial conditions. Specifically, the solution should be the superposition of the particular solution $P(s) = H(s)X(s)$ that is the response to the pushing function (input) $X(s)$ and the complementary solution $C(s) = H(s)\,C_0$ that is the response to the vector of initial conditions, C_0. Thus, the solution should have the form $Y(s) = H(s)X(s) + H(s)\,C_0$. If you hadn't simplified the system, you algebraically combine the equations to remove unneeded variables and then proceed as above. If you have multiple dependent variables or inputs, H(s) will be an array of polynomial fractions.

A. **A**nalyze. We will review tools to analyze linear systems behavior, including stability analysis, steady state gain, and time constants. These are very useful for understanding the general behavior of a system for arbitrary values of parameters. For example, we can predict whether or not it will reach a steady state, and if so, how long it will take to do so, and what state it will reach. This is very powerful for design, since we can determine what parameters are needed to provide a specified behavior.

N. **N**umeric Solutions. We will also learn a set of computational tools to plot the solution given the transfer function identified above, specific parameter values, initial conditions, and input (pushing function). Numeric solutions have two advantages over analytic solutions: they predict and allow you to quickly visualize the full time-dependent response, and can provide solutions much more quickly for complicated systems where analysis is very difficult. However, they also have a major disadvantage: they only show the solution for specific parameters values we assign, including the duration of the simulation. We thus need to know what parameters and time frame are of interest to obtain any information.

E. **E**rror check. Analysis is subject to human error during the manipulations. However, numeric solutions are also subject to human errors such as typographical mistakes entering equations and misunderstanding and thus misuse of software, as well as numeric errors. You can have the highest level of certainty in your conclusions when analytic solutions, numeric solutions, and (when possible) intuition all agree. For details, see the section below on Verification.

Review of Linear Systems Analysis:

Stability. The denominator of the transfer function of the system, H(s), is called the *characteristic equation*. The *roots* of this equation are the values of s for with the equation has the value 0. The real components of the roots give the stability of the system:

1. A stable system requires the real components of all roots to be negative. With no input, the system decays to a set point, and the response to a bounded input is a bounded output.
2. A marginally stable system occurs when the real components of at least one root is zero, and the real components of the rest are negative. With no input, the system has a bounded output, but the response to a bounded input can blow up.
3. An unstable system occurs when the real component of any root is positive. Even with no input, the unstable system will blow up in response to a transient perturbation.

The imaginary components of the roots determine if the system is oscillatory. When the imaginary components are nonzero, the system oscillates while decaying (stable), remaining bounding (marginally stable), or blowing up (unstable).

We can use the roots command in MATLAB to obtain the roots of the characteristic equation, but only for specified parameter values. To use this, you define a polynomial in s as a vector of the coefficients of the polynomial. See the numeric methods section below for more information.

Time constants. The magnitude of the roots of the characteristic equation given information about the speed at which the system responds to a perturbation. The magnitude of the real roots indicate the rate at which the system approaches or leaves the steady state. Specifically, the time constant for each response is the inverse of the root, and is the time it takes for the system to achieve e-fold (2.7-fold) decay or amplification. When a system has multiple time constants, then different stages dominate at each time constants. Finally, the magnitude of the imaginary component indicates the frequency of the oscillations.

Steady state. A system in steady state is not changing over time. This is sometimes interpreted to mean that the derivatives are all zero, but sometimes is interpreted to mean that the system exhibits a consistent constant or oscillatory behavior. Note that even a system with zero derivatives may involve movement, such as electrical current, volumetric flow, mechanical movement or transport of chemicals between compartments, as long as these movements do not change the state of the system. We often refer to the steady state solution of output Y as Yss, which can be calculated from Y(0), or more precisely, from $lim_{s\to 0}sY(s)$, according to the final value theorem. The steady state gain (SSG) of a system is the ratio of output to input at steady state, when the input is constant. That is, if we have a constant input Xss, and a constant output Yss, Then $SSG = \frac{Yss}{Xss}$. The steady state gain can be calculated from H(0), or more precisely $lim_{s\to 0}H(s)$. The units of H(0) should match that required to convert input units to output units. For systems with multiple inputs and/or outputs, the steady state gain is defined by the element of the transfer function matrix that relates a particular output and input. The set point of a system is the steady state output when a particular input is zero.

Numeric Solution methods for Linear Systems:

In this class we will learn numeric methods for calculating the time-dependent response. There are many software packages that can be used to calculate outputs given transfer function representations of a system and inputs, such as the linear systems toolbox in MATLAB. Some software uses a graphical user interface (GUI) to allow the user to build the system using block diagrams. Examples are SPICE (Simulation Program with Integrated Circuit Emphasis), which is commonly used by electrical engineers, and SIMULINK, which is developed by Mathworks, the same company that develops MATLAB. We will address both command line and graphical approaches to building numerical models.

MATLAB Linear Systems Toolbox. You can express a polynomial in s as a vector of coefficients, in order from highest-order in s to lowest, where the last term is the constant term (the coefficient for s^0). Remember to include a zero for any missing terms. Thus, $as^2 + bs + c$ is represented as $[a\ b\ c]$ and $5s$ is represented as $[5\ 0]$, not as $[5]$. If `num` and `den` are the vector representations of the polynomial functions in s that make up the numerator and denominator of a transfer function, you can define a transfer function object with the command

```
H = tf(num,den)
```

You can then plot the time-dependent numeric solution to the unit step and unit impulse functions with the commands

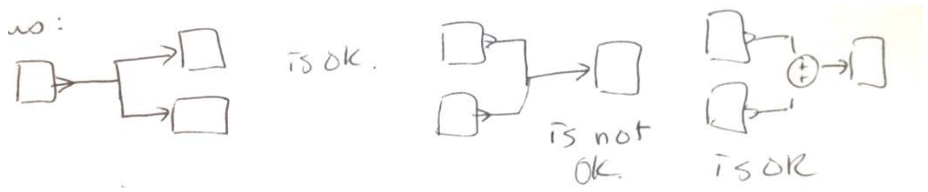
```
figure(1); impulse(H);
```

```
figure(2); step(H);
```

Finally, you can obtain the roots of the characteristic equation, which tell you about stability and time constants, with the command

```
alpha = roots(den)
```

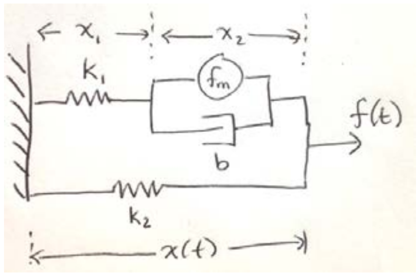
SIMULINK. SIMULINK makes it easy to plot responses to complicated inputs or systems by allowing you to build a system spatially, using block diagrams, instead of writing code. The large number of pre-packaged sources and operations makes this especially powerful. In block diagrams, lines are variables (information) and blocks are operations. We can split lines to use the information for two different operations, but if two lines come together, we need to define the operation of the two incoming variables. Thus:



You can build block diagrams at various levels of complexity. If you prefer to do more algebraic manipulation up front, you can calculate the transfer function for the whole system or part of the system and then represent that part as a single block. If you prefer to do more building of blocks in SIMULINK, you can represent each operation as a block, and build the system from scratch without prior manipulation. However, these unsimplified models can be numerically unstable, in which case they will produce an error that it fails to converge. If this happens, try simplifying the model algebraically, inserting a discrete-time memory loop (see Khoo chapter 2.9), or following the suggestions in the error message you receive.

Verification (Error Checking)

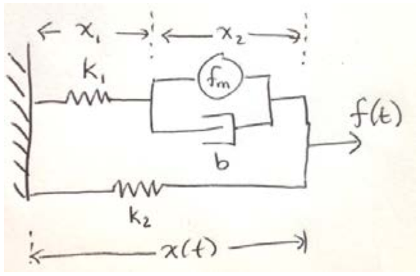
Using numeric solutions for verification. If you draw conclusions from your analytic calculations, then chose specific parameters to demonstrate that the conclusions hold for those parameters.



Use intuition for verification. As you do more modeling throughout the quarter, you should improve your intuition about how a model should behave mathematically, based on either the diagram or the conceptual description of the model. That is, you can use your common sense to predict general behaviors. For example, in a compartmental chemical model, if there is no input into a system, then the amounts of the chemicals in different compartments should decay to zero for compartments, subsystems or systems if there is transport out but not back in. Otherwise these should reach a steady state, but should not increase indefinitely nor become negative. In the mechanical, electrical and fluidic models, depending on whether effort or flow variables are held constant, steady state may require either zero effort or flow across some type of element, which can simplify the model to the point where you can solve the steady state behavior. Finally, you may be able to apply common sense to predict the signs or order of magnitude of solutions.

Using analytic calculations for verification. For complicated systems or inputs, you may not be able to efficiently obtain the full time-dependent analytic solution, but you should be able to obtain a partial analytic solution. For example, if you can analytically calculate steady state values and time constants for the response of a system to a step function, then confirm that the full numeric time dependent solution to the step function reaches that steady state value by several time constants. After this verification, you can trust the response to more complicated inputs. If needed, you can simplify the system by setting some pushing functions, initial conditions, or even parameters to zero. Make sure you test each component of your system at least once by comparing numeric and analytic solutions at least once when it was nonzero.

Example: Muscle



Last week we derived a systems equation for the muscle model shown on the right, where $x(t)$ is the change from equlilibrium length, f_m is the force of muscle contraction, and f is the force applied by the muscle tissue.

We now perform an ETSANE linear analysis on this model, except that we haven't yet learned the tools for NE.

Equation. We already derived the equation

$$\frac{k_1}{b}f + \frac{df}{dt} = \frac{k_1}{b}f_m + \frac{k_1 k_2}{b}x + (k_1 + k_2)\frac{dx}{dt}$$

Transform the equation, letting $F(s) = L(f(t)), X(s) = L(x(t)), and\ F_m(s) = L(f_m(t))$:

$$\frac{k_1}{b}F + sF = \frac{k_1}{b}F_m + \frac{k_1 k_2}{b}X + (k_1 + k_2)sX$$

Simplify to get the output on the left in terms the transfer function times the input function:

$$F = \frac{\frac{k_1}{b}}{s + \frac{k_1}{b}}F_m + \frac{(k_1 + k_2)s + \frac{k_1 k_2}{b}}{s + \frac{k_1}{b}}X$$

This can be rewritten in matrix form to relate the vector of outputs to the vector of inputs:

$$F(s) = \begin{bmatrix} \dfrac{\frac{k_1}{b}}{s + \frac{k_1}{b}} & \dfrac{(k_1 + k_2)s + \frac{k_1 k_2}{b}}{s + \frac{k_1}{b}} \end{bmatrix} \begin{bmatrix} F_m \\ X \end{bmatrix}$$

Thus, the transfer function, H(s), is a vector, $\begin{bmatrix} \frac{\frac{k_1}{b}}{s+\frac{k_1}{b}} & \frac{(k_1+k_2)s+\frac{k_1 k_2}{b}}{s+\frac{k_1}{b}} \end{bmatrix}$.

Analyze. Since H(s) is a vector, we have two characteristic equations, one for each input. However, in this case, they happen to be identical: $s + \frac{k_1}{b} = 0$

Stability. The only root of the characteristic equation is $s = -\frac{k_1}{b}$. This is negative, so any perturbation in either input will decay to zero, and the system is stable.

Time constant. The characteristic time for this decay is $\tau = b/k_1$, again for perturbations in both inputs. Why doesn't the parameter k_2 affect the time constant? Because we are prescribing x, so the force over the sarcolemma (k2) responds instantly.

Steady state. H(0) = [1 k_2], and $F(0) = F_m(0) + k_2 X(0)$. Thus, the steady state gain for the first input, f_m, is 1, while the gain for the second input, x, is k_2. If both muscle contraction and position are held constant at f_m^0 and x^0, respectively, then the force generated will be $f_{ss} = f_m^0 + k_2 x^0$. (Note that these are not the initial values $f_m(0), x(0)$, but rather specific constant values).

In summary, after a characteristic time b/k_1, the tissue exerts a force that is the sum of the muscle contractile force and the force due to passive stretch of the tissue.

Numeric Solutions

For numeric solutions, we need to consider a specific set of parameters, so we will let:

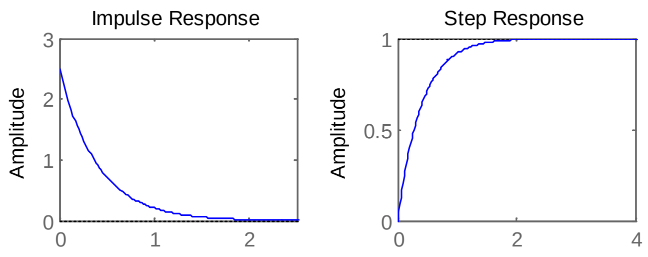
```
>> k1 = 5; k2 = 0.1; b = 2;
```

We define the system response by the polynomials in the form of the vectors of coefficients:

```
>> num1 = [k1/b]; num2 = [k1+k2, k1*k2/b]; den = [1, k1/b];
>> H1 = tf(num1,den); H2 = tf(num2,den);
```

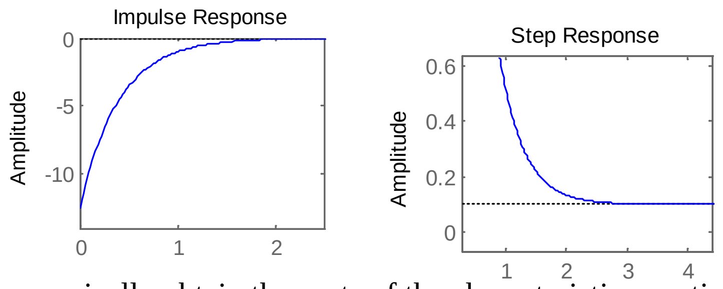
We can plot the response to impulse and step functions in fm below:

```
>> figure(1); subplot(1,2,1); impulse(H1);
>> figure(1); subplot(1,2,2); step(H1);
```



Or we can plot the response to impulse and step in x below:

```
>> figure(2); subplot(1,2,1); impulse(H2);
>> figure(2); subplot(1,2,2); step(H2);
```



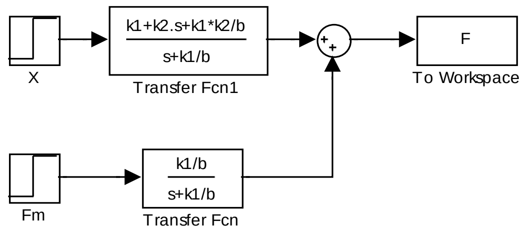
To numerically obtain the roots of the characteristic equation, we write

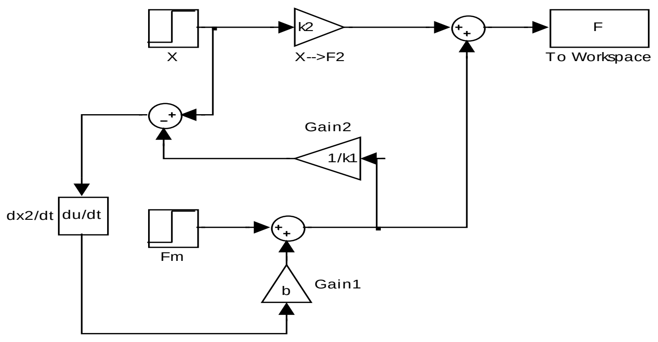
```
>> alpha = roots(den)
```

which returns:

```
alpha = -2.5
```

In simulink, we can build the model with that links each source to its transfer function, then adds the outputs together, as shown to the left below. Note that we could have also built the model based on the individual components, as shown to the right below, but the resulting model will not be as stable.





Error Check

To error check, we will compare analytic, numeric, and when possible intuition, for the stability, and time constants, and steady state gain. We obtained a single negative value for the only root, which is $\alpha - \frac{k_1}{b} = -\frac{5}{2} = -2.5$, which agrees with our intuition that the system should be stable. When we look at the full numeric solution, we see the system returning to zero for impulse responses, as needed for a stable system. The solution is about half way there by 0.4 seconds, and is essentially at steady state by 2 seconds, consistent with a time constant of $\frac{1}{\alpha} = 0.4$ seconds. Moreover, we reached steady state at 1 for a unitary step in fm and at k2 =0.1 for a unitary step in x, consistent with the analysis performed previously that predicted that $f_{ss} = f_m^0 + k_2 x^0$.

Finally, we use our intuition by applying common sense to the diagram, repeated to the right. If we reach equilibrium, there is no force across the dashpot, so the force on the upper branch is simply f_m^0 while the force on the lower branch is $k_2 x^0$. Thus, the total force f is $f_m^0 + k_2 x^0$. This also makes sense that the total force is that due to contraction plus that due to stretching the sarcolemma.

Same system, different question.

If we instead asked about the length of the muscle in response to an external load f, and the contractile force, fm, we would be treating X(s) as the output and Fm(s) and F(s) as inputs. We still start with the equation

$$\frac{k_1}{b}F + sF = \frac{k_1}{b}F_m + \frac{k_1 k_2}{b}X + (k_1 + k_2)sX$$

But now we rearrange to solve for X(s):

$$\frac{k_1 k_2}{b}X + (k_1 + k_2)sX = \frac{k_1}{b}F + sF - \frac{k_1}{b}F_m$$

$$X(s) = \frac{-\frac{k_1}{b}}{(k_1 + k_2)s + \frac{k_1 k_2}{b}}F_m + \frac{s + \frac{k_1}{b}}{(k_1 + k_2)s + \frac{k_1 k_2}{b}}F$$

Now the characteristic equation is $(k_1 + k_2)s + \frac{k_1 k_2}{b} = 0$, which has one root at $s = -\frac{k_1 k_2}{b(k_1+k_2)}$. This is negative since all parameter values are positive, so the system is stable. The time constant this time is $\frac{b(k_1+k_2)}{k_1 k_2}$, which depends on the damping and the stiffer of the two spring constants. The steady state gain is H(0) = $[-1/k_2 \quad 1/k_2]$, so the muscle gets shorter by f_m/k_2 and longer by f/k_2.

Summary

- To build a linear systems model, you build can build the ODE equations as described in week 1, or can identify the separate **equations** but not yet combine them to form a single simplified ODE set of equations.
- **Transform** the equations using the Laplace transforms listed below or additional ones found in look-up tables.

$$L\big(x(t)\big) = X(s)$$

$$L\left(\frac{dx}{dt}\right) = sX(s) - x(0)$$

$$L\left(\frac{d^2x}{dt^2}\right) = s^2X(s) - sx(0) - \frac{dx}{dt}(0)$$

$$L\left(\int_0^t x d\tau\right) = \frac{1}{s}X(s)$$

$$L\big(\delta(t)\big) = 1$$

$$L\big(\phi(t)\big) = \frac{1}{s}$$

$$L\big(t\phi(t)\big) = 1/s^2$$

$$L(e^{-at}) = \frac{1}{s+a}$$

- **Simplify** the equations to get the transform of the output on the left and the transform of the input function(s) on the right, $Y(s) = H(s)X(s) + H(s)C_0$, where X(s) is the vector of input functions, C_0 is the vector of initial conditions, Y(s) is a vector of output functions, and H(s) is the transfer function that converts input to output. H(s) is an array for the general case of multiple inputs and outputs, and each element in the array has a numerator and denominator, both of which are polynomial functions in s.
- **Analyze** the system:
  - The characteristic equation is the denominator of the system transfer function set to zero. When H(s) is an array, there are multiple characteristic equations that relate a particular input function to a particular output function.
  - The roots of this characteristic equation indicate the stability, with all negative real components needed for a stable system, any zero real component making it marginally stable, and any positive real component making it unstable.
  - The inverse of the real components roots of the characteristic equation are the time constants of the system response (the time for e-fold decay). To reach steady state, you will thus run the system for 3 to 10 times the largest time constant. The inverse of the imaginary components give the frequency of oscillations.
  - The steady state gain can be determined by evaluating the transfer function at s = 0, that is H(0). The steady state solution for a particular input, y_{ss}, is determined by evaluating $lim_{s\to 0} sY(s)$.
- **Numeric Solution**. The full time-dependent system response can be solved in simulink or similar software based on block diagrams, using the transfer function. While you can also build these models based on the individual blocks, the numeric solution is more reliably stable when calculating the transfer function analytically and using that.
- **Error check**. You can verify your solutions by comparing the time dependent response to the roots and the steady state gain, as well as our intuition based on the model description and assumptions.