Linear Algebra and Control Theory

Myles Pasetsky

What we know

Systems of linear differential equations can be represented with matrix equations

$$x_1' = 6x_1 - 9x_2$$

$$x_2' = -4x_1 + x_2$$

$$\begin{bmatrix} x_1' \\ x_2' \end{bmatrix} = \begin{bmatrix} 6 & -9 \\ -4 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

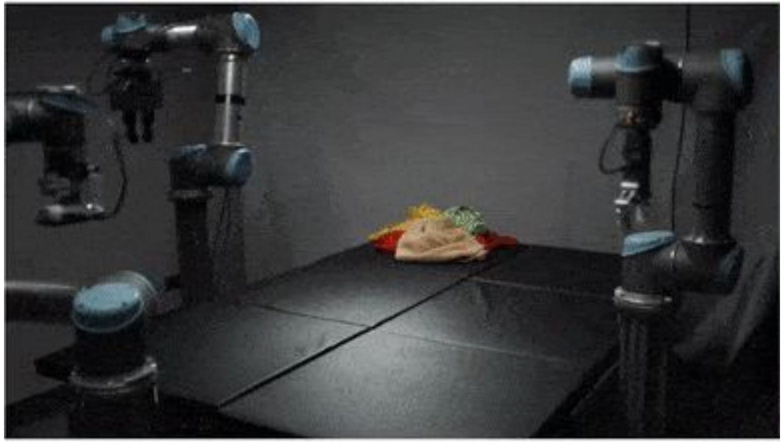
What we know

We know the general form solution of this linear system of differential equations

$$\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = c_1 e^{\lambda_1 t} \boldsymbol{v}_1 + c_2 e^{\lambda_2 t} \boldsymbol{v}_2$$

What is control theory?

Goal of control theory: find an input signal that can guide a system to a desired state



Control theory is the study of the behavior dynamical systems under input.

Linear Algebra in Control Theory

$$x' = Ax + Bu$$

x is a vector containing all the states of the system

A is a matrix describing how the state changes based on its current state

u is a vector containing the inputs to the system

B is a matrix describing how the state changes based on its inputs

x’ is the change in state

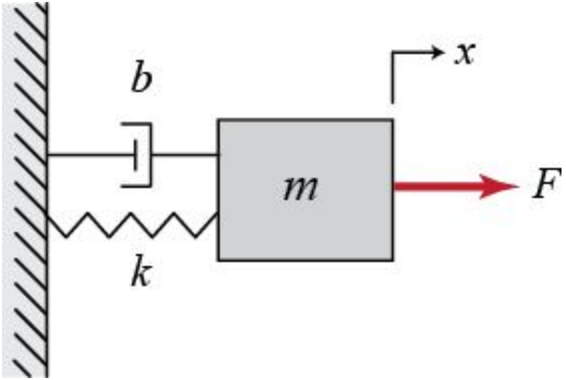
states are properties that allow you to have all the information to fully describe the future behavior of the system. Usually just means position and velocity

Linear Algebra in Control Theory

Linear Algebra in Control Theory allows for

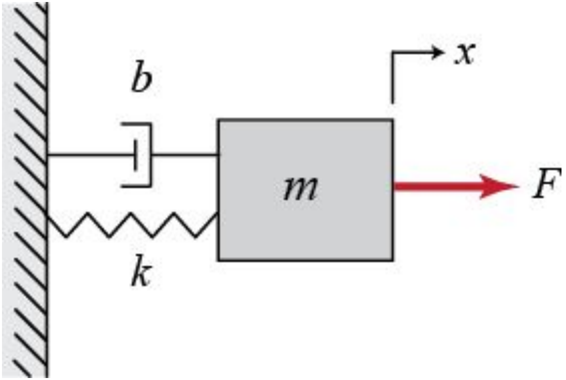
- Compact mathematical notation
- Convenient analysis methods
- Powerful solutions that work for systems of multiple states

Example: Spring-Mass-Damper System



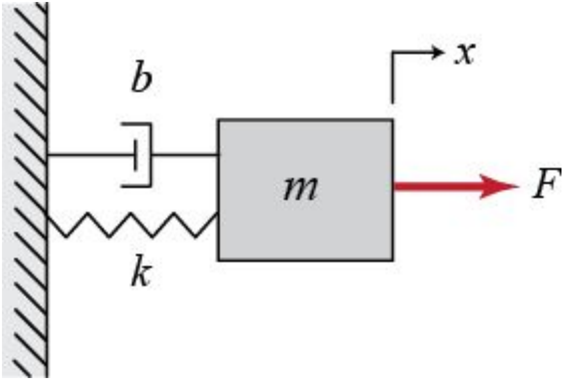
Represent this system with the equation x’ = Ax + Bu

Example: Spring-Mass-Damper System



$$mx'' = -kx - bx' + F$$

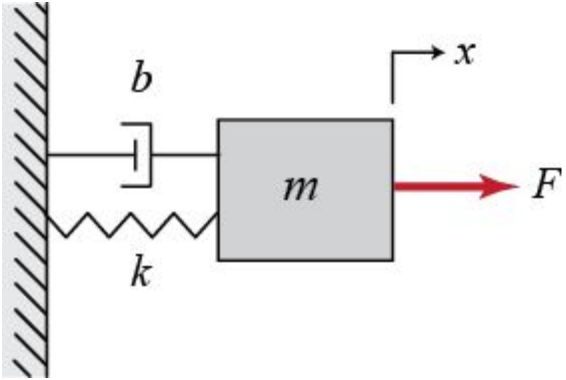
Example: Spring-Mass-Damper System



$$mx'' = -kx - bx' + F$$

$$x'' = -\frac{k}{m}x - \frac{b}{m}x' + \frac{1}{m}F$$

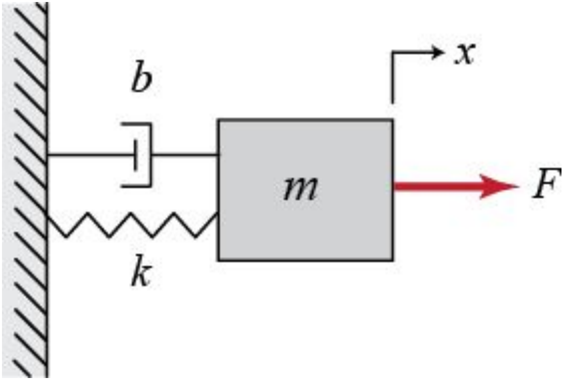
Example: Spring-Mass-Damper System



$$x' = v$$

$$v' = -\frac{k}{m}x - \frac{b}{m}v + \frac{1}{m}F$$

Example: Spring-Mass-Damper System



$$\mathbf{x}' = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$$

$$\begin{bmatrix} x' \\ v' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\dfrac{k}{m} & -\dfrac{b}{m} \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{1}{m} \end{bmatrix} \begin{bmatrix} F \end{bmatrix}$$

Stability

A stable system will tend toward an equilibrium point (one where the state is not changing) over time

One simple way to think about this is if the system was hit off from equilibrium, what is its unforced long term behavior?

Stability

What are the “unforced” dynamics?

Stability

What are the “unforced” dynamics?

$$x' = Ax + Bu \qquad u = 0$$

$$x' = Ax$$

Stability

We know what x is! *(So we know what the system does after a long time)*

$$x = c_1 e^{\lambda_1 t} v_1 + c_2 e^{\lambda_2 t} v_2 + \ldots + c_n e^{\lambda_n t} v_n$$

Stability

$$x = c_1 e^{\lambda_1 t} v_1 + c_2 e^{\lambda_2 t} v_2 + \ldots + c_n e^{\lambda_n t} v_n$$

All eigenvalues of A are real:

if all the eigenvalues are less than 0 the system is **stable** because the state will always return to x=0 (which is when x'=0) because it is a decaying exponential function

If any eigenvalue is greater than 0, x is **unstable** because its state approaches infinity (growth function)

Terms with Eigenvalues equal to 0 do not change with time (considered marginally stable)

Stability

$$x = c_1 e^{\lambda_1 t} v_1 + c_2 e^{\lambda_2 t} v_2 + \ldots + c_n e^{\lambda_n t} v_n$$

Complex eigenvalues:

$$\lambda = a + ib$$

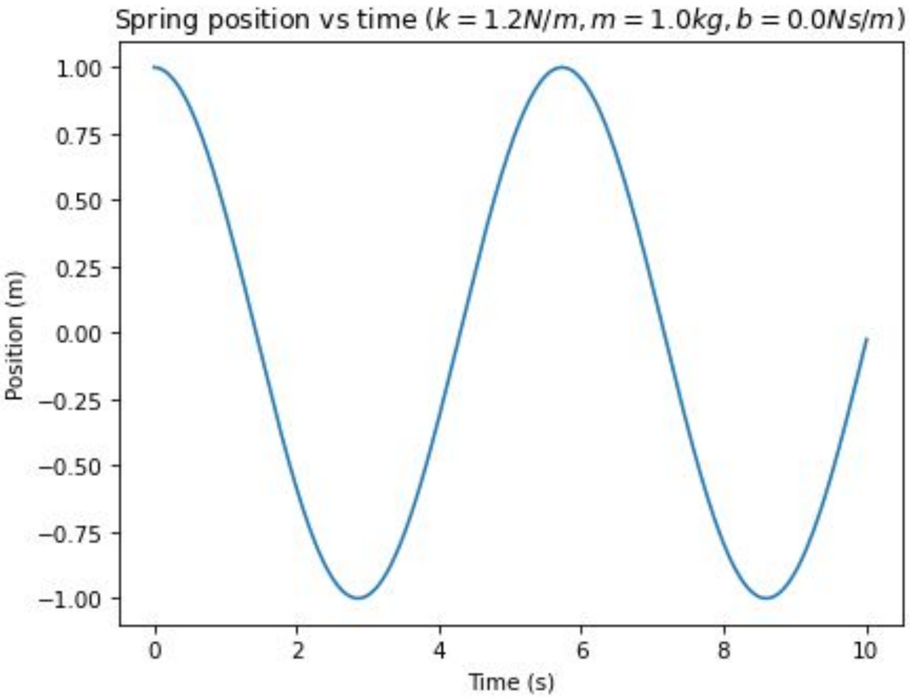
$$e^{\lambda t} = e^{(a + ib)t} =$$

$$e^{at} e^{bt \cdot i} = e^{at}(\cos(bt) + i\sin(bt))$$

There are now oscillations in the system, but we can see long term behavior still depends on an exponential function.

Stability only depends on the sign of the real part of the eigenvalue.

Example: Stability of a spring-mass-damper system

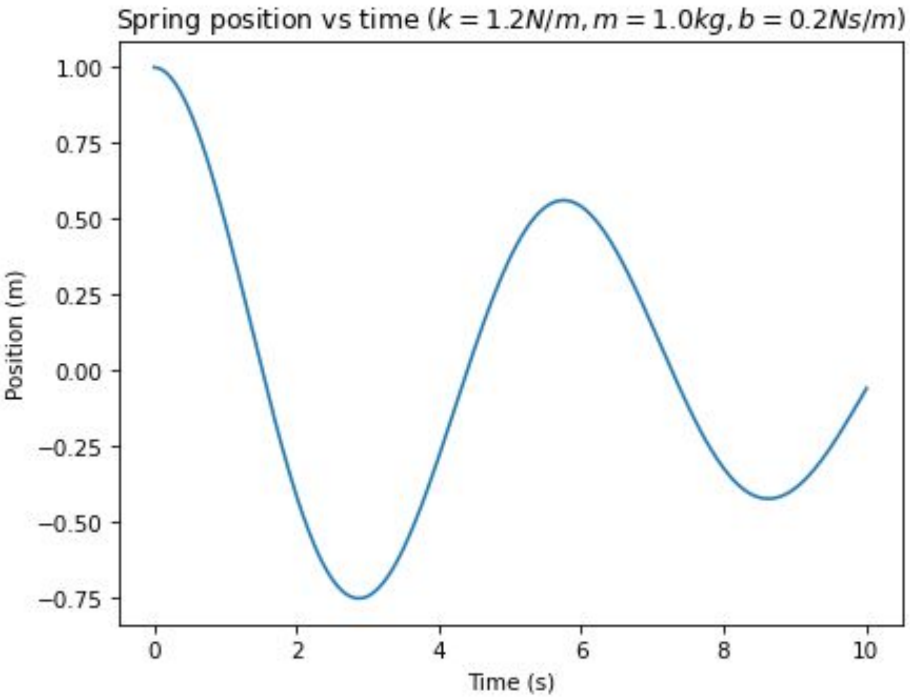


$$\begin{bmatrix} 0 & 1 \\ -1.2 & 0 \end{bmatrix}$$

$$\lambda_1 = -i\sqrt{\frac{6}{5}}$$

$$\lambda_2 = i\sqrt{\frac{6}{5}}$$

Example: Stability of a spring-mass-damper system



$$\begin{bmatrix} 0 & 1 \\ -1.2 & -0.2 \end{bmatrix}$$

$$\lambda_1 = \frac{1}{10}(-1 + i\sqrt{119})$$

$$\lambda_2 = \frac{1}{10}(-1 - i\sqrt{119})$$

Feedback

Goal: Calculating an input to drive a system to its desired state

Feedback is a common way to do this, by feeding the error between a goal state and the current state back to the system as input.

Feedback

We can write a ‘control law’ that will tell us what to feed into our system

$$u = K(r - x)$$

Feedback

$$x' = Ax + Bu$$

$$u = K(r - x)$$

$$x' = Ax + BK(r - x)$$

Now, A-BK will dictate the stability of our system.

$$x' = Ax + BKr - BKx$$

$$x' = (A - BK)x + BKr$$

We can choose the K matrix, which will change the eigenvalues!

This is like changing our A matrix, which before was not possible because A (& B) is inherit to the system.

Feedback

Remember: goal is to find inputs (forces) that we can apply to the spring to get it to reach a reference state

We want to calculate a K matrix such that the eigenvalues of A-BK are in an area of the real-complex plane that will result in stable behavior (e.g. we choose eigenvalues in the left hand side of the plane, ideally on the x-axis)

We can then go back and use this K matrix in the equation $u=K(r - x)$

Feedback

K can be guessed randomly so that A - BK has preferable eigenvalues.

Instead, K can be calculated through with an algorithm called *pole placement*.

- Pick 2 eigenvalues that are stable and find characteristic polynomial
    - Needs to have negative real component
- Find characteristic polynomial of A - BK (where K is [k_1, k_ 2])
- Compare coefficients of polynomial to solve for values of K

Note: pole refers to eigenvalue

Ex: Pole placement of a spring-mass-damper system

$$A - BK =$$

$$\begin{bmatrix} 0 & 1 \\ -1.2 & 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} k_1 & k_2 \end{bmatrix} =$$

$$\begin{bmatrix} 0 & 1 \\ -1.2 - k_1 & -k_2 \end{bmatrix}$$

Ex: Pole placement of a spring-mass-damper system

$$A - BK =$$

$$\begin{bmatrix} 0 & 1 \\ -1.2 & 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} k_1 & k_2 \end{bmatrix} =$$

$$\begin{bmatrix} 0 & 1 \\ -1.2 - k_1 & -k_2 \end{bmatrix}$$

$$A_{CL} = \begin{bmatrix} 0 & 1 \\ -1.2 - k_1 & -k_2 \end{bmatrix}$$

$$det\left(A_{CL} - \lambda I \right) = 0$$

$$\lambda^2 + k_2 \lambda + k_1 + 1.2 = 0$$

Ex: Pole placement of a spring-mass-damper system

$$A_{CL} = \begin{bmatrix} 0 & 1 \\ -1.2 - k_1 & -k_2 \end{bmatrix}$$

$$det\left(A_{CL} - \lambda I\right) = 0$$

$$\lambda^2 + k_2\lambda + k_1 + 1.2 = 0$$

$$\lambda = -2, \ -1$$

$$(\lambda + 2)(\lambda + 1) = 0$$

$$\lambda^2 + 3\lambda + 2 = 0$$

Ex: Pole placement of a spring-mass-damper system

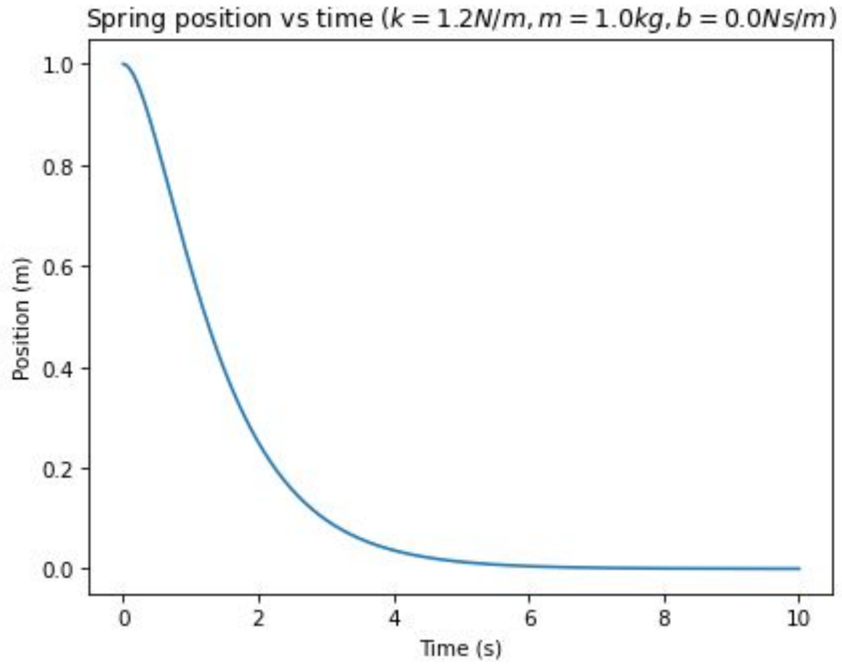
$$k_2 = 3$$

$$k_1 + 1.2 = 2$$

$$k_1 = 0.8$$

$$K = \begin{bmatrix} 0.8 & 3 \end{bmatrix}$$

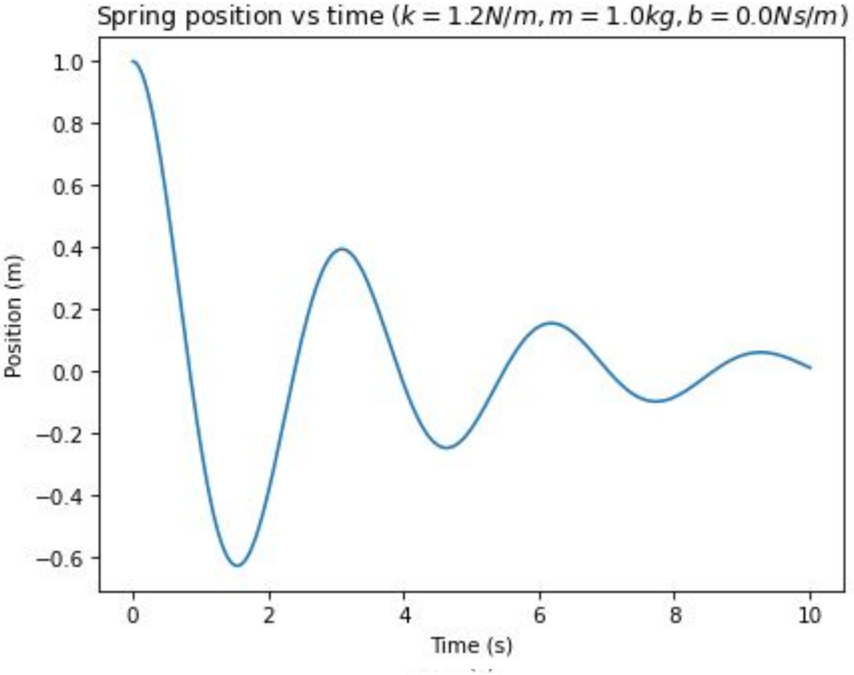
Ex: Pole placement of a spring-mass-damper system



Our perfectly oscillating spring now approaches 0m (the reference position I specified) when the control law u = K(r - x) is applied.

We have control over our spring!

Ex: Pole placement of a spring-mass-damper system



This is the response if I guess K = [3, 0.6].

The eigenvalues change to have negative real components, *but* clearly much worse than picking good eigenvalues first.

Next steps

- Picking eigenvalues requires balancing response time and effort (force) applied. Algorithm to pick eigenvalues for us given the desired balance?
- Converting equation to discrete system (real systems rely on actuators and sensors that only work at certain frequencies)
- Understand the uses of the other equation (complement to $x' = Ax + Bu$) that deals with outputs, using sensor data, and removing noise
- Change of basis / diagonalizing A (position and velocity are arbitrary)