University of California, Berkeley

ME/EECS/BioE C106B

Robotic Manipulation & Interaction

Massimiliano de Sa

These notes were written for the UC Berkeley course ME/EECS/BioE C106B in the Spring 2023 semester. I hope you find them useful!
They are primarily based off of *A Mathematical Introduction to Robot Manipulation* by Murray, Li, and Sastry, *Nonlinear Systems: Analysis, Stability, and Control* by Sastry, and *Feedback Systems: An Introduction for Scientists and Engineers* by Astrom and Murray, and have been designed to closely follow the 106B course lectures in both ordering and content.
If you have any questions, please reach out to me at mz.desa at berkeley.

Contents

Chapter 1

Dynamical Systems

Welcome to 106B! In this course, we'll discuss concepts ranging from dynamics to feedback control, path planning, vision, and beyond! We'll build upon the foundations of dynamics and control we built in 106A to form a comprehensive yet deep overview of several key fields in robotics.

1.1 Modeling Dynamical Systems

In this section, we'll begin our discussion of dynamical systems, systems who evolve with the passage of time. We'll consider different methods of describing these systems, and see how differential equations provide us with the mathematical tools we need for their analysis. Let's get started!

1.1.1 Classifying Dynamical Systems

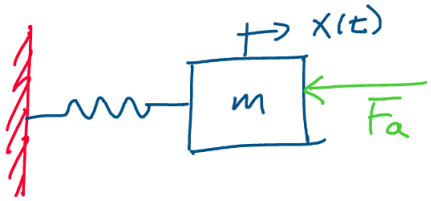
Nonlinear Systems

How can we describe the evolution of a system in time? As a robot arm moves, for example, we know that torques and forces acting on the arm govern its path through space. How can we mathematically describe the relationship between these forces and the eventual trajectory of the arm?

Using the language of *differential equations*, we may describe how systems such as robot arms, drones, autonomous vehicles, and more move with the passage of time!

As we progress through the course, we'll find it exceedingly useful to have *general* descriptions for dynamical systems. This will enable us to explore results that hold not just for one system but for *arbitrary* systems. To achieve this generality in our results, it's important that we come up with certain conventions of how to write out the differential equations for these systems.

Let's write out the equations of motion for a few systems and see if there are any common elements that stand out. Let's begin with a simple system: a mass that oscillates on a spring in the presence of an applied force, F_a:

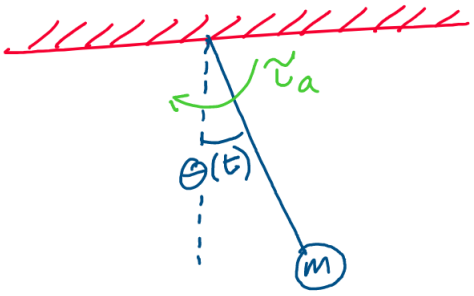


Above: A mass oscillates on a spring with an applied force, F_a.

From our knowledge of dynamics, we know that the equations of motion of this system are given by the differential equation:

$$m\ddot{x} = -kx - F_a \tag{1.1}$$

Where x is the position of the mass, $\ddot{x}$ is the acceleration of the mass, k is the spring constant, and F_a is the applied force.
Note that we use dot notation for convenience when expressing the time derivative of a variable. Recall that the number of dots above a variable is the number of time derivatives that have been taken. For instance, $\ddot{x}$ refers to the second time derivative of x.
Let's think about another physical system, a mass that swings on a pendulum around a fixed point with an applied torque τ_a. As we write its equations of motion, see if there any any common themes you can pick out between the first two systems.

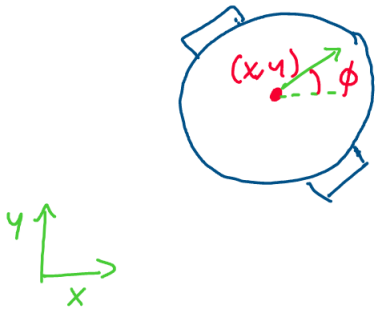


Above: A pendulum of mass m swings around a fixed point

If this pendulum has a length l, mass m, and angular position θ, its dynamics may be described by the differential equation:

$$ml^2\ddot{\theta} = -mg\sin\theta - \tau_a \tag{1.2}$$

Finally, let's consider a third system, a turtlebot! Instead of describing the equations of motion of the turtlebot using methods of dynamics, we can make use of *kinematic constraints* to arrive at the governing differential equations. We'll perform an in-depth exploration of how this is done in the coming sections.



Above: A turtlebot traveling in an (x, y) coordinate frame

We may show that the kinematics of this turtlebot are described by the following equations of motion:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} v\cos\phi \\ v\sin\phi \\ \omega \end{bmatrix} \tag{1.3}$$

Where (x, y, ϕ) are the coordinates and orientation of the turtlebot with respect to a world coordinate frame and v and ω are the speed and angular steering rate of the turtlebot, which may be controlled by a user.

What similarities do these three dynamical systems share? Firstly, something that we notice is that all of the systems have a set of variables that describe the *position* or *velocity* of the system at any given point! For the mass-spring system, this was the x position of the mass, for the pendulum the angle θ, and for the turtlebot the coordinates (x, y) and orientation ϕ.

Secondly, we observe that each system has some sort of input! For the mass, there was an applied force, F_a, for the pendulum an applied torque, τ_a, and for the turtlebot a velocity v and angular steering rate ω.

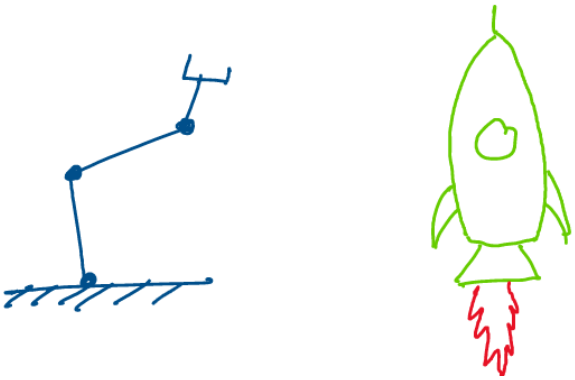
With these thoughts in mind, we may introduce the standard form for a non-linear system, known as the **state space representation**:

$$\dot{x} = f(x, u), x \in \mathbb{R}^n, u \in \mathbb{R}^m \qquad \textit{State Equation} \tag{1.4}$$

$$y = h(x, u),\ y \in \mathbb{R}^p \qquad \textit{Output Equation} \tag{1.5}$$

The first equation, $\dot{x} = f(x, u)$ is known as the **state equation**. This equation *completely* describes how the system will move as time passes, and encodes all of the information about the physical laws and processes governing the system. If the state equation $f(x, u)$ is not an explicit function of time, the system is said to be **time invariant**. If the state equation *is* an explicit function of time, the state equation is written by convention as $f(x, u, t)$ and the system is said to be **time variant**.

Most of the robotic systems we'll study are time invariant, as the governing dynamics of systems such as robot arms typically do not change as time passes. A system such as a rocket, however, whose mass changes with time as fuel is burned, would be considered time variant.



Above: Robotic systems are typically time invariant, as the physical properties of a robot arm do not change in time. A rocket is time variant, as it loses mass by burning fuel as time passes.

The vector $x \in \mathbb{R}^n$ is known as the **state vector** of the system. It contains smallest possible set of variables necessary to describe the configuration of the system at any given point. For the turtlebot, for example, the state vector would be:

$$x = \begin{bmatrix} x \\ y \\ \phi \end{bmatrix} \in \mathbb{R}^3 \tag{1.6}$$

Notice that we *cannot* control the states of the system directly!
The **input vector** $u \in \mathbb{R}^m$, on the other hand, is a vector of all variables that we have *complete* control over! We can change the elements of the input vector at our will to influence the behavior of the system. For the turtlebot, for example, we can completely control the velocity and steering rate of the vehicle to drive it to a particular location!
The input vector of the turtlebot would be:

$$u = \begin{bmatrix} v \\ \omega \end{bmatrix} \in \mathbb{R}^2 \tag{1.7}$$

What role does the second equation, $y = h(x, u)$, play in the state space representation? $y = h(x, u)$ is known as the **output equation**. The output equation has *no effect* on the actual dynamics of the system! We can pick the function $h(x, u)$ to be *anything* we want. The **output vector** of the system, $y \in \mathbb{R}^p$, typically contains our variables of interest.
For example, in the case of the turtlebot, we may be interested in controlling the x and y position of the turtlebot as it moves through the environment, but not as interested in the orientation, ϕ. In this case, we could define the output of the system to be:

$$y = \begin{bmatrix} x \\ y \end{bmatrix} \in \mathbb{R}^2 \tag{1.8}$$

The output of a system is typically a function of the state vector but can also be a function of the input vector in some cases, hence the inclusion of u in $y = h(x, u)$.
In addition to being variables of interest, the outputs of a system are also commonly chosen to be the states of the system that are observable from different sensors. This choice of output is particularly common when designing filters to determine the state of a system.
Let's think critically for a moment about the state space description of a system. What challenges do we notice with this representation?

$$\dot{x} = f(x, u), \ x \in \mathbb{R}^n, u \in \mathbb{R}^m \tag{1.9}$$
$$y = h(x, u), \ y \in \mathbb{R}^p \tag{1.10}$$

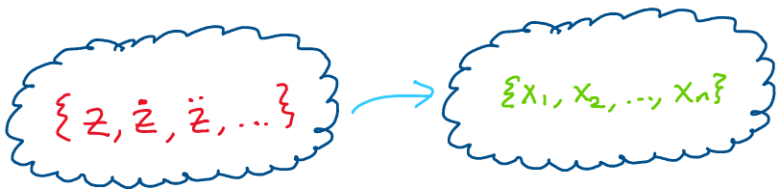
A potential problem that we notice with this formulation is that this is a *first order* system of differential equations. When we take a closer look at the dynamics equations for the mass-spring and pendulum systems, however, what do we observe?

$$m\ddot{x} = -kx - F_a \tag{1.11}$$
$$ml^2\ddot{\theta} = -mgl \sin\theta - \tau_a \tag{1.12}$$

Although the state space representation of a system, $\dot{x} = f(x, u), \ y = h(x, u)$ is a *first order* system of differential equations, both of these equations are second order, due to the appearance of a second derivative with respect to time! Does this mean that our general state space representation *fails* to describe all systems, or is there a way we can *transform* higher order systems to first order systems with equivalent behavior?
Let's see if we can convert these higher order nonlinear systems into a system of first order differential equations. Suppose that we have an n^{th} order, nonlinear differential equation, described by the following equation:

$$z^{(n)} = \frac{d^n z}{dt^n} = h(z, u) \tag{1.13}$$

Where $z \in \mathbb{R}$ is a scalar, $u \in \mathbb{R}^m$ is an input vector, and $h(z, u)$ is a function describing the dynamics of the system. Note that $z^{(n)}$ is shorthand for the n^{th} time derivative of z.
Let's see if we can find a *magic* change of variables from z and its derivatives to a new set, $\{x_1, x_2, ..., x_n\}$, that *transform* this system from an n^{th} order differential equation to a *system* of n first order differential equations!



Consider the following choice of variables:

$$x_1 = z \tag{1.14}$$

$$x_2 = \frac{dz}{dt} \tag{1.15}$$

$$x_3 = \frac{d^2z}{dt^2} \tag{1.16}$$

$$\vdots \tag{1.17}$$

$$x_n = \frac{d^{n-1}z}{dt^{n-1}} \tag{1.18}$$

We can put these x_i together into a vector, which we call x:

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n \tag{1.19}$$

Does this choice of variables allow us to *rewrite* our original system as $\dot{x} = f(x, u)$? Let's take the first derivative of each x_i and see what happens! For $1 \leq i < n$, we find:

$$\dot{x}_1 = \frac{dz}{dt} = x_2 \tag{1.20}$$

$$\dot{x}_2 = \frac{d^2z}{dt^2} = x_3 \tag{1.21}$$

$$\vdots \tag{1.22}$$

$$\dot{x}_{n-1} = \frac{d^{n-1}z}{dt^{n-1}} = x_n \tag{1.23}$$

When we take the first time derivative of each x_i, for $1 \leq i < n$, we get x_{i+1}! What happens when we take the time derivative of x_n?

$$\dot{x}_n = \frac{d}{dt}(z^{(n-1)}) = z^{(n)} = h(z, u) \tag{1.24}$$

Thus, when we take first time derivative of x_n, we get our original differential equation, $h(z, u) = h(x_1, u)$! Thus, by choosing these variables, we have preserved the dynamics of the system and converted an n^{th} order differential equation into a system of n first order differential equations.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} x_2 \\ x_3 \\ \vdots \\ h(x_1, u) \end{bmatrix} \tag{1.25}$$

We have now successfully converted our original n^{th} order differential equation into a system of n first order differential equations in state space form:

$$\dot{x} = f(x, u),\ x \in \mathbb{R}^n, u \in \mathbb{R}^m \tag{1.26}$$

The choice of variables $\{x_1, x_2, ..., x_n\} = \{z, \dot{z}, ..., z^{(n-1)}\}$ that we used to accomplish this transformation are known as **phase variables**.
Let's gain some practice converting higher order systems into state space using phase variables. Recall the dynamics of the mass-spring system we discussed above:

$$m\ddot{x} = -kx - F_a \tag{1.27}$$

We notice that this system is *second order*. Thus, we need two phase variables, since we will convert this second order equation to a system of *two* first order equations. Using the phase variable convention, we choose:

$$x_1 = x,\ x_2 = \dot{x} \tag{1.28}$$

Since F_a is an applied force that we can control, we define an input vector $u = F_a$. Now, we rewrite the system in terms of these variables:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{-k}{m}x_1 - \frac{1}{m}u \end{bmatrix} \tag{1.29}$$

Thus, we have successfully rewritten our second order system in the form:

$$\dot{x} = f(x, u),\ x = [x_1, x_2] = [x, \dot{x}],\ u = F_a \tag{1.30}$$

Note that because x is a variable that is commonly used in the description of physical systems, the variable q is often used in the place of x as the state vector of the system to avoid confusion.
As a general rule of thumb for checking your choice of phase variables, the highest derivative value in your choice of phase variables should be *one lower* than the order of your original differential equation. For instance, in the example above, our original differential equation was second order, and the highest derivative phase variable, $\dot{x}$, was first order. This comes from the fact that in state space, we always take the first time derivative of our entire state vector.

Control Affine Systems

Now that we have a general form for describing nonlinear systems, we can ask the question: are there certain *subclasses* of nonlinear systems that are easier to work with?
As we'll soon discover in our study of feedback control, designing a general feedback controller to control the state vector of an arbitrary nonlinear system:

$$\dot{x} = f(x, u) \tag{1.31}$$

Is quite complex! Because of the ambiguous nature of the function f - that f can be almost *any* function in the general nonlinear case - it's difficult to design controllers for the most general class of nonlinear systems!
Certain subsets of the class of nonlinear systems, however, have the advantage of being particularly common in practice and much simpler to design feedback controllers for! One such class is the set of **control affine** nonlinear systems. These are nonlinear systems whose state equation may be written in the form:

$$\dot{x} = f(x) + g(x)u,\ x \in \mathbb{R}^n, u \in \mathbb{R}^m \tag{1.32}$$

Now, instead of having the input and state dynamics combined into a single function, $\dot{x} = f(x, u)$, the input dynamics may be *separated* from those purely related to the state vector! Let's break down the different elements of a control-affine system.
The function $f(x)$ is known as the **drift dynamics** of the system. These are the dynamics that have *nothing* to do with the inputs to the system, and are purely a function of the state vector. $f(x) : \mathbb{R}^n \to \mathbb{R}^n$ is a vector-valued function:

$$f(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_n(x) \end{bmatrix} \in \mathbb{R}^n,\ f_i(x) : \mathbb{R}^n \to \mathbb{R}, 1 \leq i \leq n \tag{1.33}$$

Where each f_i within the vector $f(x)$ is a scalar-valued function. If the input u to the system were zero, the state vector would *drift* according to the value of the function $f(x)$.
In the input term, $g(x) : \mathbb{R}^n \to \mathbb{R}^{n \times m}$ is a matrix-valued function of x that determines the effect of the input on the evolution of the state vector. Each element of $g(x)$ is a scalar function:

$$g(x) = \begin{bmatrix} g_{11}(x) & \dots & g_{1m}(x) \\ \vdots & \ddots & \vdots \\ g_{n1}(x) & \dots & g_{nm}(x) \end{bmatrix} \in \mathbb{R}^{n \times m} \tag{1.34}$$

Because of they separate the control input from the rest of the system dynamics, control affine systems prove to be much easier to work with when designing feedback controllers. Although not every nonlinear system is control affine, a sizeable number of complex robotic systems are, making these systems of key importance to us.
What does the term affine actually mean? In general, a function $h(x) : \mathbb{R}^n \to \mathbb{R}^n$ is said to be *affine in* x if it can be expressed:

$$h(x) = Ax + b \tag{1.35}$$

Where A is a matrix and b is a vector. Since a control affine system is of the form:

$$\dot{x} = f(x) + g(x)u \tag{1.36}$$

It is considered to be *affine* in the control input, u. This property gives the class of control affine systems its name!

Linear Systems

Control affine systems provide a significant reduction in challenges with respect to full nonlinear systems, $\dot{x} = f(x, u)$. Are there any subclasses of dynamical systems that can further simplify our analyses?
The class of **linear systems** is perhaps the most simple yet one of the most useful class of dynamical systems. Linear systems are conventionally written in the form:

$$\dot{x} = Ax + Bu,\ x \in \mathbb{R}^n, u \in \mathbb{R}^m \tag{1.37}$$

$$y = Cx + Du,\ y \in \mathbb{R}^p \tag{1.38}$$

If A, B, C, D are constant matrices that don't change as time passes, the system is said to be **linear time invariant** (LTI). If these matrices *do* change in time, they are written as $A(t), B(t), C(t), D(t)$, and the system is said to be **linear time variant** (LTV).
Inspecting the structure of the equations, we have the familiar setup of a state equation and output equation expressing the dynamics of the system. However, in this case, both the state vector, x, and input vector, u, are simply scaled by matrices instead of being manipulated by various nonlinear functions.
Since such a representation allows us to use the elegant language of linear algebra to perform system analysis, the class of linear differential equations proves to be exceptionally convenient in robotics and control.
As we'll see shortly, we can make remarkably deep conclusions about many systems by inspecting linear algebraic objects such as eigenvalues and eigenvectors.
Mathematically, what does it mean for a system to be linear?
If, u_1, u_2 are arbitrary inputs, α, β are scalar constants, and $y(t, x_0, u)$ is the output of a system at time t starting at initial condition x_0 with an input u, the following three properties must be satisfied for a system to be considered **input-output linear**:

1. $y(t, \alpha x_1 + \beta x_2, 0) = \alpha y(t, x_1, 0) + \beta y(t, x_2, 0)$
2. $y(t, \alpha x_1, \beta u) = \alpha y(t, x_1) + \beta y(t, 0, u)$
3. $y(t, 0, \alpha u_1 + \beta u_2) = \alpha y(t, 0, u_1) + \beta y(t, 0, u_2)$

These three conditions express the linearity of the output with respect to the initial conditions and inputs to the system. For the linear system, $\dot{x} = Ax + Bu, y = Cx + Du$, we can verify these facts using the properties of matrix algebra.

1.1.2 Solving Differential Equations

Now that we have established conventions for representing three classes of differential equations: nonlinear, control affine, and linear, what conclusions can

we come to regarding their solutions? Is it always possible to solve a differential equation? If a solution is guaranteed, is it unique?
In this section, we'll explore some properties of *initial value problems*, which seek to find a solution $x(t)$ that solves a differential equation:

$$\dot{x} = f(x, u, t), \; x(t_0) = x_0 \tag{1.39}$$

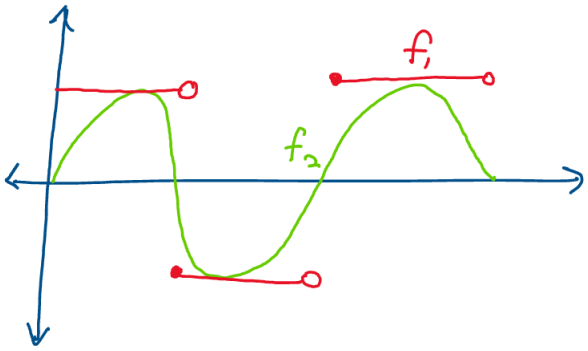
Subject to the initial condition $x(t_0) = x_0$.
To simplify our analysis, we'll primarily discuss the questions of existence and uniqueness for differential equations with no input ($u = 0$). Note that these results can be extended to systems with inputs without too much further effort.

Existence and Uniqueness of Solutions

Let's begin by thinking about the question of existence. How do we know a solution to a differential equation will exist? Although this might seem like a strange question at first, let's take a moment to think about why it might be relevant. Consider the time variant nonlinear system:

$$\dot{x} = f(x, t) \tag{1.40}$$

If $f(x, t)$ can be *any* nonlinear function of x, t, the possibilities for f are *endless*! f could be a sine function, a square wave function, a step function - the possibilities are virtually endless!



Above: We can choose anything for f in the general case.

For all of these choices of f, is it realistic to expect there to be a function $x(t)$ for which $\dot{x} = f(x, t)$ and $x(0) = x_0$? As it turns out, not quite! To understand the conditions required of f, we'll undertake a short discussion of continuity.
In mathematics, there are several formal definitions of what it means for a function to be continuous, with some definitions having stricter demands than others. Let's begin by discussing a strong type of continuity known as **Lipschitz continuity**.

Definition 1 ***Lipschitz Continuity***
A function $f(x) : \mathbb{R}^n \to \mathbb{R}^n$ is said to be globally Lipschitz continuous if there exists a finite scalar constant $L \in \mathbb{R}$ such that for all $x, y \in \mathbb{R}^n$:

$$||f(x) - f(y)|| \leq L||x - y|| \tag{1.41}$$

Let's think for a moment about what this definition of continuity states. Rearranging the definition of Lipschitz continuity, we require that for all $x, y \in \mathbb{R}^n$, there exists a constant L such that:

$$||f(x) - f(y)|| \leq L||x - y|| \tag{1.42}$$

$$\frac{||f(x) - f(y)||}{||x - y||} \leq L \tag{1.43}$$

For a simple interpretation of this inequality, we can think of the single variable case, where $n = 1$. This inequality then becomes:

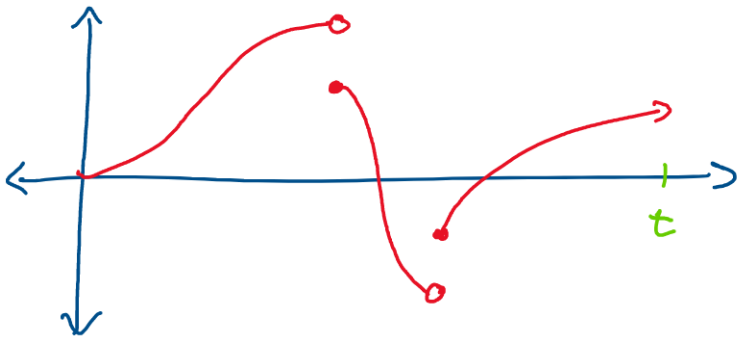
$$\frac{|f(x) - f(y)|}{|x - y|} \leq L \tag{1.44}$$

If we take the limit of both sides of this expression as $y \to x$, we observe that the Lipschitz condition bounds the derivative of the function by the Lipschitz constant!

$$\lim_{y \to x} \frac{|f(x) - f(y)|}{|x - y|} \leq \lim_{y \to x} L \tag{1.45}$$

$$\frac{df}{dx} \leq L \tag{1.46}$$

This means that a function such as a square wave, which has sudden jumps in its value across its domain, is *not* Lipschitz continuous.
Before we relate this material back to the study of differential equations, let's discuss a much less restrictive form of continuity, **piecewise continuity**. Although we won't state a formal mathematical definition, you can think of a piecewise continuous function as a bounded function with at most a finite number of breaks. For example, consider the function:



Above: A piecewise continuous function

Since this function has a finite number of breaks on the interval $[0, t]$ and remains bounded (doesn't "blow up" to infinity), it is said to be piecewise continuous on the domain $[0, t]$.
With these definitions in mind, we are now ready to understand the conditions for existence and uniqueness of solutions to differential equations.

Theorem 1 ***Global Existence and Uniqueness Theorem***
An initial value problem $\dot{x} = f(x,t)$, $x(t_0) = x_0$ has a unique solution if $f(x,t)$ is piecewise continuous with respect to t and for all $T \in [t_0, \infty)$, there exist finite constants $L_1, L_2 \in \mathbb{R}$ such that for all $t \in [0, T]$:

$$||f(x,t) - f(y,t)|| \leq L_1 ||x - y||, \text{ for all } x, y \in \mathbb{R}^n \tag{1.47}$$

$$||f(x_0,t)|| \leq L_2 \tag{1.48}$$

Let's break down the different parts of this theorem. First, we need f to be piecewise continuous with respect to t - as we vary t, there should only be a finite number of jumps in the value of f. Secondly, we require f to be globally Lipschitz continuous across all possible time intervals from 0 to ∞. Thirdly, we require the value of $f(x_0, t)$ to be bounded for all time. If these conditions are satisfied, the initial value problem is guaranteed to have a unique solution!
Let's try applying this theorem to show that a linear time invariant system will have a unique solution. Consider the linear system with constant A with all finite entries and a finite initial condition:

$$\dot{x} = Ax, \; x \in \mathbb{R}^n, \; x(t_0) = x_0 \tag{1.49}$$

Let's start out by identifying the function f! In this case, the function $f : \mathbb{R}^n \to \mathbb{R}^n$ is as follows:

$$f(x) = Ax \tag{1.50}$$

Since $f(x) = Ax$ is a constant function with respect to time, this means that it must be piecewise continuous with respect to time. This checks off the first required condition!
Next, let's show that f is globally Lipschitz continuous. Looking at the definition of Lipschitz continuity, we start by analyzing $||Ax - Ay||$, where $x, y \in \mathbb{R}^n$ are arbitrary vectors. We know:

$$||Ax - Ay|| = ||A(x - y)|| \tag{1.51}$$

Where do we go from here? We'd like to find some way of bringing A out of the norm, $||\cdot||$, and forming an inequality. If we can do this, we'll have demonstrated that $||Ax - Ay|| \leq L||x - y||$ for some finite L!
How can we extract A? One way of doing this is to use something known as a *matrix norm*. Just like we can assign a norm to a vector to measure its size, we can do the same thing for a matrix! There are many ways of doing this. The matrix 2-norm is defined as follows:

$$||A||_2 = \sup_{x \neq 0 \in \mathbb{R}^n} \frac{||Ax||}{||x||} = \sigma_{max} \tag{1.52}$$

Where $\sup_{x \neq 0 \in \mathbb{R}^n}$ refers to the *supremum* of $||Ax||/||x||$, the smallest possible upper bound of $||Ax||/||x||$ as we change the value of x. Miraculously, it can be shown that this norm is equal to the maximum singular value of of A, σ_{max}!
By definition of the supremum, σ_{max} is an upper bound for $||Ax||/||x||$. This means that for all $x \in \mathbb{R}^n$, we can form the inequality:

$$\frac{||Ax||}{||x||} \leq \sigma_{max} \tag{1.53}$$

$$||Ax|| \leq \sigma_{max}||x|| \tag{1.54}$$

Let's apply this back to our discussion of Lipschitz continuity! Recall:

$$||Ax - Ay|| = ||A(x - y)|| \tag{1.55}$$

Applying our result from the 2-norm of A, we reach the inequality:

$$||Ax - Ay|| \leq \sigma_{max}||x - y|| \tag{1.56}$$

Thus, as long as the entries of A are all finite, $f(x) = Ax$ is a Lipschitz continuous function! This checks off the second requirement for global existence and uniqueness.
Finally, we must check the third result: that there exists an L_2 such that $||f(x_0, t)|| \leq L_2$ for all t. Since $f(x) = Ax$ is not a function of t, this simplifies the requirement to:

$$||f(x_0)|| = ||Ax_0|| \leq L_2 \tag{1.57}$$

For some L_2. Since $||Ax_0||$ is a constant, as long as A and x_0 have all finite entries, this expression is bounded. Thus, the third condition is checked off.

Since $\dot{x} = Ax, x(t_0) = x_0$ satisfy all of the required conditions, we conclude that linear time invariant systems *must* have a unique solution $x(t)$ for all finite initial conditions x_0.
Note that we can also extend this result with minimal further conditions to the case where the system has a nonzero input, u, and the matrix A is time-varying.

Solving Linear Differential Equations

Now that we've shown that a unique solution to a linear differential equation must exist for all well-behaved initial conditions, let's try finding what the solution actually is!
We'll start by dealing with the case where the system has no input. Consider the linear, time invariant system:

$$\dot{x} = Ax,\ x \in \mathbb{R}^n,\ x(t_0) = x_0 \tag{1.58}$$

How would we go about finding a solution to this differential equation? Let's begin by reviewing the *scalar* case, where $n = 1$. The scalar case of this differential equation would be:

$$\dot{x} = \frac{dx}{dt} = ax,\ x(t_0) = x_0 \tag{1.59}$$

To solve this equation, we can use a technique known as *separation of variables*, where we bring all of the quantities involving x to one side of the equation and all quantities involving t to the other side. Let's begin this process by expanding the derivative $\frac{dx}{dt}$:

$$\frac{dx}{dt} = ax \tag{1.60}$$

$$\frac{dx}{x} = adt \tag{1.61}$$

Now, we may integrate both sides. On the left hand side, we will integrate from x_0 to x, and on the right hand side, from t_0 to t. Note that when integrating, we use "dummy variables" χ and τ in the place of x and t for the sake of proper mathematical convention.

$$\int_{x_0}^{x} \frac{1}{\chi} d\chi = \int_{t_0}^{t} a d\tau \tag{1.62}$$

$$[ln|\chi|]_{x_0}^{x} = [a\tau]_{t_0}^{t} \tag{1.63}$$

$$ln\frac{|x|}{|x_0|} = a(t - t_0) \tag{1.64}$$

Where we applied a logarithm rule to turn $ln(a) - ln(b)$ into $ln(a/b)$. Assuming that $x(t)$ and x_0 have the same sign, we drop the absolute values and solve for x by exponentiating both sides.

$$\frac{x}{x_0} = e^{a(t-t_0)} \tag{1.65}$$

$$x(t) = e^{a(t-t_0)} x_0 \tag{1.66}$$

Thus, we have found the solution to our differential equation! Let's turn back to the case where $x \in \mathbb{R}^n$, and use the scalar case to guide us to a vector solution. We'd now like to solve the initial value problem:

$$\dot{x} = Ax,\ x \in \mathbb{R}^n,\ x(t_0) = x_0 \tag{1.67}$$

Since we found that the solution to the scalar case was $e^{a(t-t_0)}x_0$, we hypothesize that the solution to the vector case will be:

$$x(t) = e^{A(t-t_0)}x_0 \tag{1.68}$$

Where $e^{A(t-t_0)}$ is the **matrix exponential** of $A(t-t_0)$, defined according to the Taylor series of e^x.

$$e^{At} = I + At + \frac{(At)^2}{2!} + \frac{(At)^3}{3!} + ... = \sum_{k=1}^{\infty} \frac{(At)^k}{k!} \in \mathbb{R}^{n\times n} \tag{1.69}$$

We can prove that for all finite A, this series will actually converge to an $n \times n$ matrix! Using this definition, let's show that our hypothesized solution satisfies the differential equation $\dot{x} = Ax$ and the initial condition $x(t_0) = x_0$.
Let's first check that this solution satisfies the differential equation. We can start by taking the time derivative of $x(t)$:

$$\dot{x}(t) = \frac{d}{dt}\Big(e^{A(t-t_0)}\Big)x_0 \tag{1.70}$$

We may use the series definition of the matrix exponential to find the derivative of $e^{A(t-t_0)}$.

$$\frac{d}{dt}\Big(e^{A(t-t_0)}\Big) = \frac{d}{dt}\Big[I + A(t-t_0) + \frac{(A(t-t_0))^2}{2!} + \frac{(A(t-t_0))^3}{3!} + ...\Big] \tag{1.71}$$

$$= 0 + A + A^2(t-t_0) + \frac{A^3(t-t_0)^2}{2!} + ... \tag{1.72}$$

$$= A\Big[I + A(t-t_0) + \frac{(A(t-t_0))^2}{2!} + ...\Big] \tag{1.73}$$

$$= Ae^{A(t-t_0)} \tag{1.74}$$

Let's substitute this into our expression for the time derivative of x and see what we get!

$$\dot{x} = \frac{d}{dt}\Big(e^{A(t-t_0)}\Big)x_0 \tag{1.75}$$

$$\dot{x} = Ae^{A(t-t_0)}x_0 \tag{1.76}$$

Now, we notice that $e^{A(t-t_0)}x_0$ equal to our proposed solution, $x(t)$! Thus:

$$\dot{x} = Ax \tag{1.77}$$

This solution therefore satisfies our differential equation. Using the definition of the matrix exponential, we can also check that this solution satisfies the initial condition. Thus, the matrix exponential provides us with the unique solution to a linear, time invariant differential equation with no input for an arbitrary initial condition.
Is there a similarly general solution to the case where we *do* apply an input to the system? Although we won't prove it here, we can show that the solution to the initial value problem:

$$\dot{x} = Ax + Bu,\ x \in \mathbb{R}^n, u \in \mathbb{R}^m,\ x(t_0) = x_0 \tag{1.78}$$

Is given by the following expression:

$$x(t) = e^{A(t-t_0)}x_0 + \int_{t_0}^{t} e^{A(t-\tau)}Bu(\tau)d\tau \tag{1.79}$$

Notice that if $u(t) = 0$ for all time, this solution simplifies to the zero-input case discussed above!

Evaluating the Matrix Exponential

So far, we've shown that the solution to a variety of linear systems relies on the use of the matrix exponential. How can we actually compute what the value of the matrix exponential is?
Before we discuss techniques for computing the exponential, it's important to remember - the matrix exponential is *not* simply the exponential of each entry in the matrix! That is, if we have a matrix:

$$A = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \in \mathbb{R}^{n \times n} \tag{1.80}$$

Its matrix exponential is *not* generally computed by taking the exponential of each a_{ij}.

$$e^{At} \neq \begin{bmatrix} e^{a_{11}t} & \dots & e^{a_{1n}t} \\ \vdots & \ddots & \vdots \\ e^{a_{n1}t} & \dots & e^{a_{nn}t} \end{bmatrix} \text{ (For general } A \in \mathbb{R}^{n \times n}) \tag{1.81}$$

How, then, can we compute its value? The matrix exponential is simplest to compute for a diagonal matrix. Suppose we have a diagonal matrix A, defined:

$$A = \begin{bmatrix} \lambda_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_n \end{bmatrix} \in \mathbb{R}^{n \times n} \tag{1.82}$$

From linear algebra, we know that we can compute the p^{th} power of a diagonal matrix by raising each diagonal entry to the p^{th} power:

$$A^p = \begin{bmatrix} \lambda_1^p & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_n^p \end{bmatrix} \in \mathbb{R}^{n \times n} \tag{1.83}$$

Let's use this fact to compute the matrix exponential of a diagonal A. Recall:

$$e^{At} = I + At + \frac{(At)^2}{2!} + \frac{(At)^3}{3!} + ... \tag{1.84}$$

$$= \begin{bmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{bmatrix} + \begin{bmatrix} \lambda_1 t & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_n t \end{bmatrix} + \begin{bmatrix} \frac{\lambda_1^2 t^2}{2!} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \frac{\lambda_n^2 t^2}{2!} \end{bmatrix} + ... \tag{1.85}$$

$$= \begin{bmatrix} 1 + \lambda_1 t + \frac{\lambda_1^2 t^2}{2!} + & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 + \lambda_n t + \frac{\lambda_n^2 t^2}{2!} + ... \end{bmatrix} \tag{1.86}$$

Now, we recognize each diagonal entry as the scalar Taylor series of $e^{\lambda_i t}$. Thus, for diagonal A, we conclude:

$$e^{At} = \begin{bmatrix} e^{\lambda_1 t} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & e^{\lambda_n t} \end{bmatrix} \in \mathbb{R}^{n \times n} \tag{1.87}$$

Using this simple diagonal structure, we may devise a method to compute the matrix exponential of *any* diagonalizable matrix!
Suppose we have a diagonalizable matrix $A \in \mathbb{R}^{n \times n}$. Since A is diagonalizable, we know there exists a transformation matrix $T \in \mathbb{R}^{n \times n}$ such that:

$$D = TAT^{-1} \tag{1.88}$$

$$A = T^{-1}DT \tag{1.89}$$

Where D is a diagonal matrix with all of the eigenvalues of A along its diagonal!

$$D = \begin{bmatrix} \lambda_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_n \end{bmatrix} \in \mathbb{R}^{n \times n} \tag{1.90}$$

Let's see if we can compute the matrix exponential of A using what we now know about diagonal matrices.
Using the transformation defined above, let's take the matrix exponential of

$At = T^{-1}DTt$ and see if we can extract e^{Dt}, which we know how to compute.

$$e^{At} = e^{T^{-1}DTt} \tag{1.91}$$

$$e^{At} = I + T^{-1}DTt + \frac{(T^{-1}DTt)^2}{2!} + \frac{(T^{-1}DTt)^3}{3!} + ... \tag{1.92}$$

$$e^{At} = I + T^{-1}DTt + \frac{T^{-1}DTT^{-1}DTt^2}{2!} + \frac{(T^{-1}DTt)^2T^{-1}DTt}{3!} + ... \tag{1.93}$$

$$e^{At} = I + T^{-1}(Dt)T + \frac{T^{-1}(Dt)^2T}{2!} + \frac{T^{-1}(Dt)^3T}{3!} + ... \tag{1.94}$$

$$e^{At} = T^{-1}\Big[I + Dt + \frac{(Dt)^2}{2!} + \frac{(Dt)^3}{3!} + ...\Big]T \tag{1.95}$$

$$e^{At} = T^{-1}e^{Dt}T \tag{1.96}$$

Therefore, to calculate e^{At} for any diagonalizable A, all we need to do is calculate the matrix exponential of the associated diagonal matrix, e^{Dt}, and transform it *back* to e^{At} using $T^{-1}e^{Dt}T$. This enables us to compute the closed form of the matrix exponential for a wide variety of matrices.
Note that if A is *not* a diagonalizable matrix, it may be transformed into another form called the **Jordan canonical form** (JCF) for easy computation of the matrix exponential. Interested readers are encouraged to consult *Linear Algebra* by Friedberg, Insel, and Spence for an in-depth treatment of the Jordan canonical form.

1.1.3 Equilibrium Points

We now have a significant body of language which we may use to understand and interpret dynamical systems. Let's begin the process of analyzing these systems, and examine their key points in closer detail.
The first concept we'll study is that of an **equilibrium point**.

Definition 2 ***Equilibrium Point***
An equilibrium point of a system $\dot{x} = f(x, u)$ is a pair (x_e, u_e) such that:

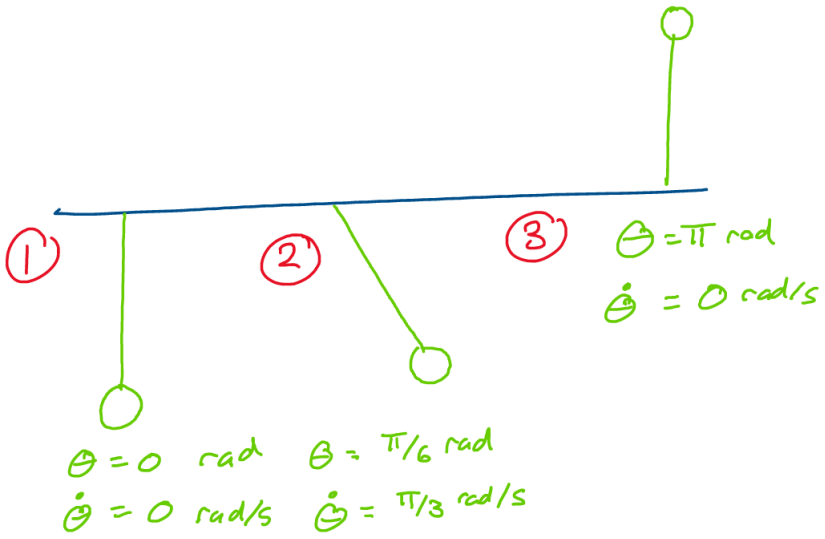
$$0 = f(x_e, u_e) \tag{1.97}$$

Let's break this definition down and interpret the name "equilibrium point" physically. If we are at an equilibrium point (x_e, u_e), this means that the derivative of the state vector of the system is zero:

$$\dot{x} = 0 = f(x_e, u_e) \tag{1.98}$$

This means that at this point, the evolution of the system is entirely *frozen* - none of the state variables are changing with respect to time!

What might an equilibrium point look like in a physical system? Let's consider the example of a simple pendulum to gain some intuition. We'll then proceed to verify our results mathematically.



Above: Three different positions and velocities of a simple pendulum

Let's consider the first position-velocity pair, where the pendulum sits in a vertically downward position with zero angular velocity. In this configuration, will the state vector be changing with respect to time?

Intuitively, we know that if we apply no forces to a pendulum in its downwards position, it will stay there - neither the position nor velocity of the pendulum should change! Thus, we hypothesize that this first configuration is an equilibrium point of the system.

What about the second configuration? This configuration captures the pendulum mid-swing - will this be an equilibrium point? Since the pendulum is actively swinging in this configuration, we know that the system will not remain in that configuration as time goes on. The pendulum will continue to swing up and down. This second configuration is therefore not an equilibrium point.

What about the third configuration? In this configuration, the pendulum is perfectly balanced at 180 degrees from the downwards position. Assuming that there are no small disturbances, the pendulum should remain perfectly balanced in this configuration. Thus, we hypothesize that the third configuration is also an equilibrium point.

Notice how although the first and third configurations are both equilibrium points, the behavior of the pendulum at and around those points is quite different! If we tried to balance the pendulum by hand in its lower position, we would find it to be quite easy.

If we were to try and balance the pendulum in its upper configuration, however, it would be significantly more challenging, as it would have a tendency to fall

right down! We'll soon learn how to formalize these qualitative differences as we shift our discussion to the topic of stability.
Let's mathematically verify our hypotheses about which configurations are equilibrium points and which are not.The dynamics of a simple frictionless pendulum with length l, mass m, and no external forces are expressed:

$$ml^2\ddot{\theta} = -mgl\sin\theta \tag{1.99}$$

Let's convert this equation into phase variable form and solve for its equilibrium points. Choosing $\theta, \dot{\theta}$ as state variables, we may rewrite this system as:

$$\dot{q} = \begin{bmatrix}\dot{\theta}\\ \ddot{\theta}\end{bmatrix} = \begin{bmatrix}\dot{\theta}\\ -\frac{g}{l}\sin\theta\end{bmatrix} = f(q) \tag{1.100}$$

To solve for the equilibrium points, we set $f(q) = 0$.

$$\begin{bmatrix}0\\0\end{bmatrix} = \begin{bmatrix}\dot{\theta}\\ -\frac{g}{l}\sin\theta\end{bmatrix} \tag{1.101}$$

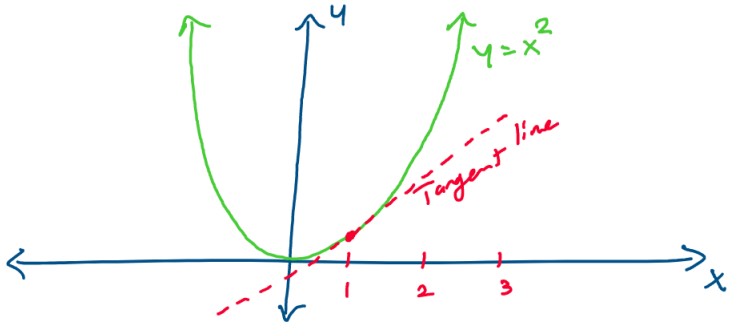
$$0 = \dot{\theta} \tag{1.102}$$

$$0 = \sin\theta \tag{1.103}$$

Thus, all points of the form $(n\pi, 0) = (\theta, \dot{\theta})$, where $n \in \mathbb{Z}$ is an integer, must be equilibrium points of the system!
Does this match up with our earlier hypotheses? Since even values of n correspond to the downwards position of the pendulum and odd values of n correspond to the upwards position of the pendulum, we conclude that our earlier analyses were correct - both the upwards and downwards configurations are equilibrium points.

1.1.4 Linearization

Thus far in our treatment of dynamical systems, we've considered linear and nonlinear systems largely as separate groups of systems. As we've seen thus far, linear systems have simple and interpretable solutions compared to nonlinear systems, and allow us to apply the often-elegant tools of linear algebra in our analyses.
Is there some way we can meaningfully *relate* a nonlinear system to a linear system? If we can achieve this, we can apply the tools of linear analysis to study nonlinear systems. To answer this question, let's take a moment to turn our attention away from the study of dynamical systems and towards the study of calculus.
One of the most fundamental concepts in calculus is that of the derivative. We know that using the derivative of a function, we can find the line tangent to a function at any point where its derivative is defined. For instance, consider the function $y = x^2$, which has been sketched below.



Above: The function $y = x^2$. How do we find the equation of its tangent line?

Suppose we wanted to find the equation for the line tangent to the curve at $x = 1$. Firstly, we could find the slope of the tangent line by evaluating the derivative of the function at $x = 1$.

$$\frac{dy}{dx}\Big|_{x=1} = 2 \tag{1.104}$$

Note that the notation $dy/dx|_{x=1}$ means evaluate the value of dy/dx at $x = 1$. After we have the slope, to find the equation of the tangent line, all we need are the coordinates of a point on the line. In this case, we can pick $(1, 1)$, the point of intersection of the tangent line with the function. The equation of the tangent line is then:

$$y = 2(x - 1) + 1 \tag{1.105}$$

We can generalize this equation to find the tangent line at *any* point where a function is differentiable. The line tangent to a function $f(x)$ at a point x_t is given by:

$$y = \left(\frac{df}{dx}\Big|_{x=x_t}\right)(x - x_t) + f(x_t) \tag{1.106}$$

Let's take a moment to think critically about the applications of this equation. When we found the tangent line of the *nonlinear* function $y = x^2$ at the point $x = 1$, we found a *linear* function that, in a small region around $x = 1$, looked similar to the original, nonlinear function. This is something we can observe if we zoom in close to the point $x = 1$:

Above: In a small region around $x = 1$, the tangent line and the nonlinear function appear to be similar.

With this similarity in mind, could we use the tangent line as a local linear approximation of the nonlinear function?
Let's perform an informal mathematical study of this question, and see how the difference between the nonlinear function and tangent line change as we move further away from $x = 1$. Let's defined the error between the nonlinear function and tangent line to be:

$$e(x) = y_{nonlinear} - y_{tangent} \tag{1.107}$$
$$e(x) = x^2 - 2(x-1) - 1 \tag{1.108}$$
$$e(x) = x^2 - 2x + 1 \tag{1.109}$$
$$e(x) = (x-1)^2 \tag{1.110}$$

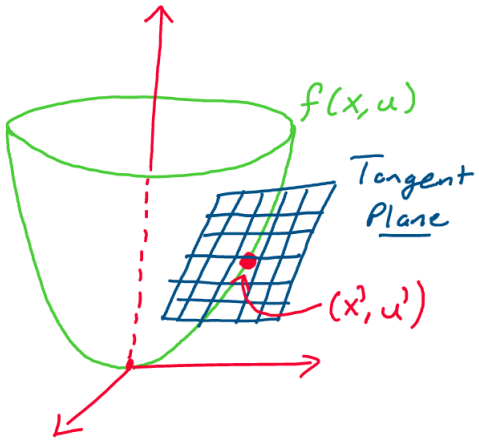
Thus, as long as we don't move too far away from the tangent point, $x = 1$, the error between the actual function and approximation will remain small! If we move away from the tangent point, our error will grow at an unbounded, quadratic rate. Thus, we can use a tangent line for a good *local* approximation of the nonlinear function!
Let's discuss how we can apply this concept to locally approximate nonlinear systems as linear ones! We'd like to approximate a nonlinear system of the form:

$$\dot{x} = f(x,u),\ x \in \mathbb{R}^n, u \in \mathbb{R}^m \tag{1.111}$$

As a linear system:

$$\dot{z} = Az + Bv,\ z \in \mathbb{R}^n, v \in \mathbb{R}^m \tag{1.112}$$

For *some* choice of A, B, z, v. How can we accomplish this? Since our functions are now multivariable, instead of finding a tangent line to approximate a curve, we now look for a tangent *plane* that approximates a surface!



Above: We wish to find the equation of the plane tangent to a surface.

Let's begin by finding the equation of the plane tangent to $f(x,u)$ at the point (x', u'). To find the equation of the tangent plane, we follow the same procedure

as for finding a tangent line! The tangent plane is defined:

$$\frac{\partial f}{\partial x}\Big|_{(x,u)=(x',u')}(x-x') + \frac{\partial f}{\partial u}\Big|_{(x,u)=(x',u')}(u-u') + f(x',u') \tag{1.113}$$

Where $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial u}$, known as the **Jacobians** of f with respect to x and u, are defined:

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{n\times n} \tag{1.114}$$

$$\frac{\partial f}{\partial u} = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial u_1} & \cdots & \frac{\partial f_n}{\partial u_m} \end{bmatrix} \in \mathbb{R}^{n\times m} \tag{1.115}$$

Notice that the definition of the tangent plane follows the same format as that of a tangent line! We first take the derivative of the nonlinear function with respect to x, evaluate it at the tangent point, and multiply it by the difference between x and the point of approximation. Since this is a multivariable function, we repeat this procedure for u. Following this, we add on the value of the function at the point (x', u').

Now that we have the equation of the tangent plane at our point of approximation, we notice a few problems! This equation looks *nothing* like our desired format for a linear system:

$$\dot{z} = Az + Bv \tag{1.116}$$

The first step we'll take towards transforming our tangent plane equation to the correct, linear system form is defining a change of variables. We want a change of variables such that we have one matrix multiplied by a vector z and another matrix multiplied by a vector v.

Looking at our formula for the tangent plane, we notice that we have one Jacobian matrix multiplied by $x - x'$ and another Jacobian matrix multiplied by $u - u'$. Based on this observation, we define z and v as follows:

$$z = x - x' \tag{1.117}$$

$$v = u - u' \tag{1.118}$$

Rewriting our tangent plane equation in terms of z and v, we have:

$$\frac{\partial f}{\partial x}\Big|_{(x,u)=(x',u')} z + \frac{\partial f}{\partial u}\Big|_{(x,u)=(x',u')} v + f(x',u') \tag{1.119}$$

This is much closer to the format we're looking for! We have one remaining term we'd like to get rid of: $f(x', u')$. If we can somehow ensure $f(x', u') = 0$, we'll have found our linear approximation of the nonlinear system!

We know that $f(x', u') = 0$ if (x', u') is an equilibrium point of the system $\dot{x} = f(x, u)$. Thus, to ensure our linear approximation is valid, we require that the point at which we take our linear approximation is an equilibrium point, $(x', u') = (x_e, u_e)$, of the nonlinear system. Applying the assumption that the point of approximation is an equilibrium point, we finally arrive at the following equation for the tangent plane:

$$\frac{\partial f}{\partial x}\Big|_{(x,u)=(x',u')} z + \frac{\partial f}{\partial u}\Big|_{(x,u)=(x',u')} v \tag{1.120}$$

Therefore, if we define A, B, z, v as follows:

$$A = \frac{\partial f}{\partial x}\Big|_{(x,u)=(x',u')} \tag{1.121}$$

$$B = \frac{\partial f}{\partial u}\Big|_{(x,u)=(x',u')} \tag{1.122}$$

$$z = x - x' = x - x_e \tag{1.123}$$

$$v = u - u' = u - u_e \tag{1.124}$$

We may approximate the nonlinear system $\dot{x} = f(x, u)$ by its tangent plane at its equilibrium points as:

$$\dot{z} = Az + Bv \tag{1.125}$$

This linear system is known as the **Jacobian linearization** of the nonlinear system, due to its use of the Jacobians of $f(x, u)$ in defining A and B.
Let's summarize the process of taking a Jacobian linearization of a nonlinear system with a step by step procedure.

Definition 3 ***Jacobian Linearization***
To find the Jacobian linearization of a nonlinear system, $\dot{x} = f(x, u)$, follow the procedure:

1. *Ensure that the point at which you are taking your linearization is an equilibrium point (x_e, u_e), where $f(x_e, u_e) = 0$.*
2. *Find the Jacobians of f with respect to x and u, and evaluate them at the point $x = x_e, u = u_e$. Define these to be the A and B matrices of your approximated system.*

$$A = \frac{\partial f}{\partial x}\Big|_{(x,u)=(x_e,u_e)} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{n \times n} \tag{1.126}$$

$$B = \frac{\partial f}{\partial u}\Big|_{(x,u)=(x_e,u_e)} = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial u_1} & \cdots & \frac{\partial f_n}{\partial u_m} \end{bmatrix} \in \mathbb{R}^{n \times m} \tag{1.127}$$

3. Define a change of variables:

$$z = x - x_e \tag{1.128}$$
$$u = u - u_e \tag{1.129}$$

4. Write the approximate linear system in terms of A, B, z, v:

$$\dot{z} = Az + Bv \tag{1.130}$$

Generally, this approximation is only valid within a small region surrounding the equilibrium point (x_e, u_e).

As we move forward into our study of stability, we'll find the Jacobian linearization to be a powerful tool that enables us to make nontrivial conclusions about the behavior of the nonlinear system using linear algebra.

1.1.5 Discrete Time Systems

Thus far, our discussion of dynamical systems has focused strictly on the study of continuous time systems. Continuous time systems are systems whose states evolve smoothly as time passes. These are the types of systems we deal with in the real world.

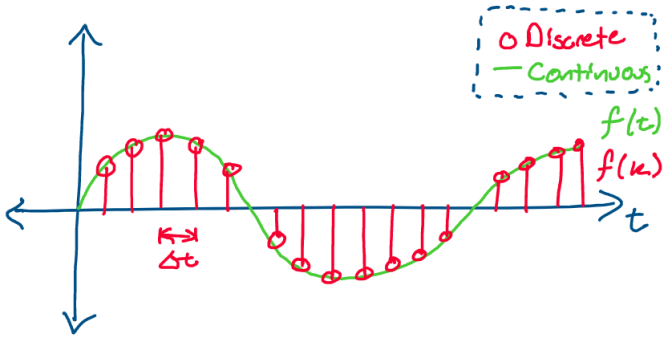
As an apple falls from a tree or a drone flies across a field, its state will change continuously as every second, microsecond, or any other increment of time passes by. These continuous time systems are *all* governed by some form of differential equation. Are there any other types of systems out there?

Discrete time systems are another important class of dynamical system. Instead of caring about the state of the system at *every* instant in time, a discrete time system jumps from one state to the next at *fixed time intervals*.

Our digital computers are one example of a discrete time system - they perform computations and interact with the physical world in small discrete intervals.

The fact that the *physical* systems we try to control are continuous, but the computers we might use to control them are discrete brings up an important point in the study of dynamical systems. Since our computers operate in discrete time, and perceive the physical world through sampling from sensors at set time intervals, it's important to have an understanding of how we can model discrete time systems.

What characterizes a discrete time signal? Consider the following graph, where a discrete time sine wave, plotted in red, and a continuous time sine wave, plotted in green, have been overlaid. Note that a discrete time signal is typically drawn with a circle at the value of the signal and a line extending to the time axis.



Above: A digital signal (in red) and a continuous signal (in green).

The first, and perhaps most important quality of a discrete time signal is that it's only defined at *discrete time intervals* - we only care about the value of a discrete signal at certain instants in time. The **sampling time**, Δt, is the time between these intervals.

What consequences does this characteristic have for our description of discrete time systems? If we only care about the value of a signal at increments of Δt, we can track the total time that has passed by taking integer multiples of Δt:

$$t = k\Delta t,\ k = 0, 1, 2, ... \in \mathbb{Z}_0^+ \tag{1.131}$$

Where $\mathbb{Z}_0^+$ is the set of positive integers including zero. Thus, if the sampling time Δt is fixed, all we need to describe t in a discrete time system is a single integer, k, the number of Δt intervals that have passed.

Because all we need to describe time is k, discrete time signals are typically written as functions of the form:

$$x(k),\ k = 0, 1, 2, ... \in \mathbb{Z}_0^+ \tag{1.132}$$

Let's consider some examples of discrete time signals described in this manner. A discrete time sine wave with amplitude a, frequency ω, and sampling period Δt could be written:

$$x(k) = a\sin(\omega k\Delta t) \tag{1.133}$$

Using functions such as these, we can construct discrete time systems that are every bit as rich and interesting as continuous time systems.

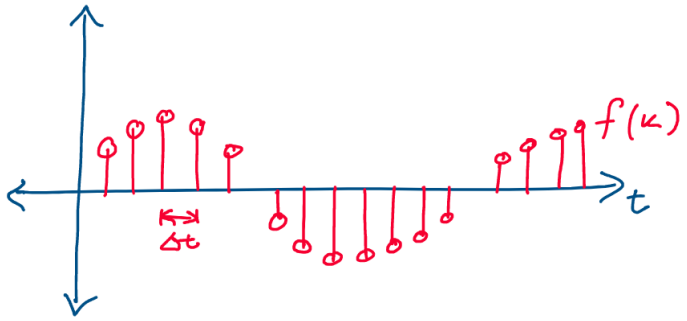
Now that we've come up with a method of describing discrete time signals, let's discuss how we can formulate discrete time dynamical systems. Recall that for continuous time systems, we use models of the form:

$$\dot{x} = f(x, u) \tag{1.134}$$

Does this description still make sense for a discrete time system? For this model to be applicable to a discrete time system, the time derivative of the state vector must be well-defined! This means that the limit:

$$\frac{dx}{dt} = \lim_{\delta t \to 0} \frac{x(t + \delta t) - x(t)}{\delta t} \tag{1.135}$$

Must exist. Does this limit make sense for a discrete time signal $x(k)$? Consider the following discrete time signal:



Above: does it make sense to take the derivative of this discrete signal?

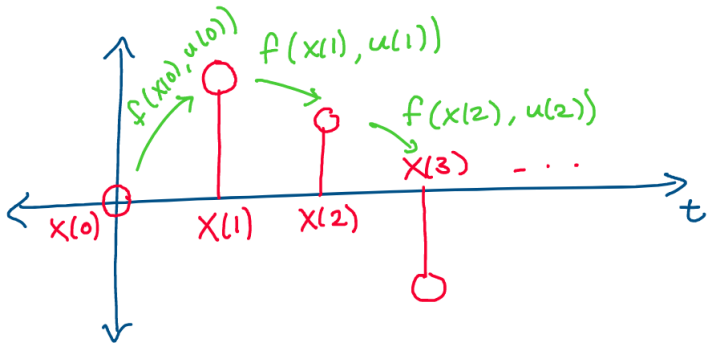
Since this signal is only defined at individual instants, it is not differentiable with respect to time at *any* t! Thus, we have no way of taking the limit as $\delta t \to 0$. This means that we *cannot* use the description $\dot{x} = f(x, u)$ to describe a discrete time dynamical system.

How, then, can we describe the evolution of such a system? Instead of using the derivative of the state vector to describe the evolution of a system, discrete time systems use a function f that maps from the state and input vectors at time step k to the state vector at time step $k + 1$.

Mathematically, the conventional representation of a discrete time system with a state x and an input u is written:

$$x(k+1) = f(x(k), u(k)),\ x \in \mathbb{R}^n,\ u \in \mathbb{R}^m,\ k \in \mathbb{Z}_0^+ \tag{1.136}$$

As you can see in the equation above, instead of taking in an input and state and returning a state derivative, the function f takes in an input and a state at step k and returns the state at step $k + 1$.



Above: $f(x(k), u(k))$ maps to the next state vector, $x(k + 1)$.

Just like with continuous time systems, there are several important subclasses of discrete time systems, all with the same definitions as their continuous time

counterparts. There are control affine discrete time systems, which have the form:

$$x(k+1) = f(x(k)) + g(x(k))u(k) \tag{1.137}$$

And linear discrete time systems, which have the form:

$$x(k+1) = Ax(k) + Bu(k) \tag{1.138}$$

Where $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times m}$ are matrices. Note that in the most general case, the mapping from state and input to next state may also be an explicit function of the time step, k.

Discretization

Since we often use digital computers to design our controllers, in many control design problems, it proves useful to approximate a continuous time system by a discrete time system! This is particularly common in model predictive control, which formulates a control and path planning problem as a large optimization problem. We'll discuss this strategy in detail in the coming sections!
The process of approximating a continuous time system by a discrete time system is known as **discretization**. How can we perform discretization in a meaningful way that *preserves* some behavior of our original system?
Let's begin our analysis of this problem with the general nonlinear system:

$$\dot{x} = f(x, u) \tag{1.139}$$

Where we assume f is a smooth mapping. Let's see if we can extract a discrete time model from this continuous system. To begin, recall the limit definition of a derivative:

$$\dot{x} = \lim_{\Delta t \to 0} \frac{x(t + \Delta t) - x(t)}{\Delta t} \tag{1.140}$$

Instead of taking the limit of this expression as $\Delta t \to 0$, let's try *approximating* the value of the derivative by fixing Δt to be some small constant. Then, for small Δt:

$$\dot{x} \approx \frac{x(t + \Delta t) - x(t)}{\Delta t} \tag{1.141}$$

Since $\dot{x} = f(x, u)$, this tells us:

$$\frac{x(t + \Delta t) - x(t)}{\Delta t} \approx f(x, u) \tag{1.142}$$

$$x(t + \Delta t) - x(t) \approx f(x, u)\Delta t \tag{1.143}$$

$$x(t + \Delta t) \approx f(x, u)\Delta t + x(t) \tag{1.144}$$

This format, where we have $x(t + \Delta t)$ on the left hand side, and $x(t)$ on the right hand side, looks similar to the format of a discrete time system! Let's

define the sampling time of our discrete time system to be Δt, the small value that we chose when approximating the derivative of the state vector.
Using this Δt, we can express the total time passed at each sampling interval as $t = k\Delta t$, where k is an integer. Let's plug this into the expression above and see what we get!

$$x(k\Delta t + \Delta t) \approx f(x(k\Delta t), u(k\Delta t))\Delta t + x(k\Delta t) \tag{1.145}$$
$$x((k+1)\Delta t) \approx f(x(k\Delta t), u(k\Delta t))\Delta t + x(k\Delta t) \tag{1.146}$$

Thus, we can now *entirely* identify the dynamics of the system with a single integer k! To convert into the correct notation, we define a discrete time estimate of the state vector, $\hat{x}(k) = x(k\Delta t)$ and an input vector $\hat{u}(k) = u(k\Delta t)$. Notice that we *only* use the values of the state and input vector *exactly* at $t = k\Delta t$. We lose any information about how $x(t), u(t)$ vary during each sampling period. This enables us to rewrite our approximation as:

$$\hat{x}(k+1) = f(\hat{x}(k), \hat{u}(k))\Delta t + \hat{x}(k) \tag{1.147}$$

We have successfully approximated our continuous time system with a discrete time system! This type of approximation is known as **Euler discretization**, and can yield fairly good results over short time periods for small Δt.
If we were to apply this approximation for long periods of time, however, note that the error between $x(t)$ and $\hat{x}(k)$ might accumulate at an unbounded rate. To slow the accumulation of error, other methods of discretization that use higher derivatives of the state vector, such as **Runge-Kutta discretization**, may be used.