# Section 5: Simulating the Motion of Robots (Kinematics)

We previously reviewed robot locomotion and reviewed basic physical principles and mechanisms by which robots achieve motion (see Fig. 1). We continue along this path and begin our study of *kinematics* – a branch of physics that mathematically describes motion in terms of the geometry of the system (i.e., the relationships between positions, orientations, speeds and accelerations). For this reason it sometimes called the "geometry of motion". *Dynamics* is a also a study of motion, but from the perspective of how motion is generated and sustained by *forces*. The dynamics of most robots can be derived by drawing a free body diagram and applying Newton's Laws.

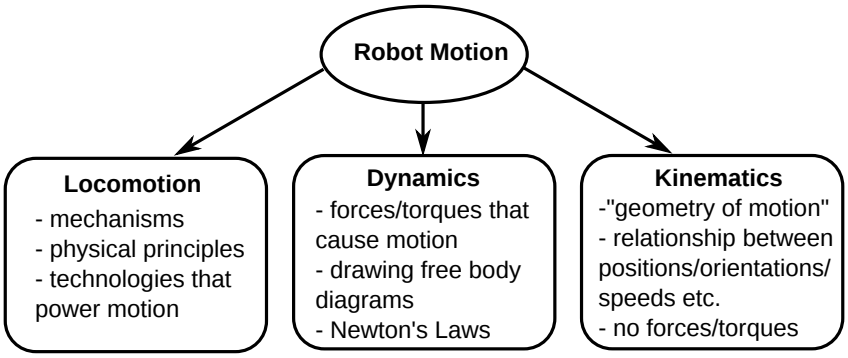



Figure 1: Branches of study regarding the motion of robots

The ultimate goal of kinematics and dynamics is to develop *equations of motion* that describe how a system moves. In our case, we are interested in the equations of motion describing how a robot moves. You may learn to *derive* equations of motion in another physics or engineering course. However, in this course, we will focus on how to *use* equations of motion for the purposes of simulation, stability and control analysis, and motion planning. Equations of motion are usually expressed as a system of ordinary differential equations (ODEs).

## Ordinary Differential Equations (ODEs)

A system of first-order *ordinary differential equations* (ODEs) written as follows:

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(t, \mathbf{x}) \tag{1}$$

and has the following main ingredients:

- a *state vector*

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

  that represents $n$ quantities of interest (e.g., the joint angles of a manipulator arm, or the position and orientation of an autonomous car).

- an independent variable, usually time $t$ (we are mainly interested in how the state $\mathbf{x}(t)$ evolves with time from some initial condition $\mathbf{x}(t_0)$ at an initial time $t_0$.)

- the derivative $d\mathbf{x}/dt$ of the state is a function of both time $t$ and the state $\mathbf{x}$ itself. This is captured by the fact the right hand side of Eq. 1 has both time $t$ and state $\mathbf{x}$ dependencies (written as $\mathbf{f}(t, \mathbf{x})$)

You may be familiar with calculus problems of the form $dx/dt = t^3 + 2t$ where the entire right-hand side is a pure function of the independent variable $t$ (only). The moment we make the right hand side a function of $x$ itself, it becomes a differential equation. For example: $dx/dt = t^3 + 2tx$ is a differential equation, $dx/dt = x$ is also a differential equation, whereas, $dx/dt = t$ is not.

In some cases it is possible to solve a differential equation using the *separation of variables* principle, in which all of the $x$ terms are brought to one side, and all of the $t$ terms are brought to the other, then both sides are integrated, as shown in the following example:

---

**Example: Solving an ODE using the separation of variables principle**
Consider the equation in which $x$ is a function of $t$ (i.e., $x(t)$) but we suppress this notation for clarity:

$$\frac{dx}{dt} = x$$

Put the $x$ terms on the left, $t$ terms on the right:

$$\left(\frac{1}{x}\right) dx = dt$$

Then integrate both sides to obtain

$$\int \left(\frac{1}{x}\right) dx = \int dt$$
$$ln(x) = t + C$$

where $C$ is an integration constant. Taking the exponent of both sides

$$e^{ln(x)} = e^{t+C}$$
$$x = \underbrace{e^C}_{K} e^t$$
$$x = Ke^t$$

we arrive at the result. The new constant $K$ can be determined by considering the desired initial condition, if one is provided.

---

The above example shows how ODEs can be solved "by hand" in some cases. However, in this course we will not be concerned with obtaining analytical solutions to differential equations. Instead, we will simulate them numerically (on our computers) and we will be content with plotting the results (rather than obtaining a mathematical expression for the solution).

In Eq. 1 we wrote the state as a vector quantity $\mathbf{x}$. The fact that the derivative $d\mathbf{x}/dt$ depends on the vector $\mathbf{x}$ means that the ODE for each scalar $x_i$ can be a function of any of the other variables in $\mathbf{x}$ In other words, the equations are *coupled*. Consider for example the following ODE

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(t, \mathbf{x})$$
$$\begin{pmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \end{pmatrix} = \begin{pmatrix} x_1 \\ x_1 - x_2 \end{pmatrix}$$

While $dx_1/dt$ depends only on $x_1$ (as considered in the previous example), the expression for $dx_2/dt$ depends on both $x_1$ and $x_2$. Clearly this coupled system of equations needs to be solved simultaneously because changes in $x_1$ will affect how $x_2$ evolves.

## Initial Value Problems

An *initial value problem* (IVP) is one in which the the ODE is given along with information about how the equation starts. An IVP problem has the following components:

- an ODE in standard form:
$$\frac{d\mathbf{x}}{dx} = \mathbf{f}(t, \mathbf{x})$$
- the time at which the equation starts $t_0$ (often the initial time is simply $t_0 = 0$)
- the state at which the equation starts $\mathbf{x}_0$ (often this is written as $\mathbf{x}(t_0) = \mathbf{x}_0$)

When presented with an IVP, the problem is to determine how the equation evolves over time (i.e., to determine $\mathbf{x}(t)$). That is, we want to know the state $\mathbf{x}$ at future times $t > t_0$.

---

**Example: An example initial value problem**
Consider the following vector ODE:

$$\frac{d\mathbf{x}}{dt} = \begin{pmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \end{pmatrix} = \begin{pmatrix} \sin t - x_1 x_2 \\ x_1 - 2x_2 \end{pmatrix}$$

with initial conditions

$$\mathbf{x}(t_0) = \begin{pmatrix} x_1(t_0) \\ x_2(t_0) \end{pmatrix} = \begin{pmatrix} -1 \\ 2 \end{pmatrix} \qquad \text{and} \qquad t_0 = 0$$

The initial value problem is to determine how $x_1$ and $x_2$ evolve from their initial values given above, as shown in Fig. 2. The various trajectories starting from the same initial condition illustrate the nature of the problem. We wish to determine the correct pair of trajectories that will satisfy the ODE given above (at each time $t$ the derivatives $dx_1/dt$ and $dx_2/dt$ must be equal to the expression on the right hand side of the ODE that is a function of the values of $t$ and the values $x_1(t)$ and $x_2(t)$ at that particular instant).

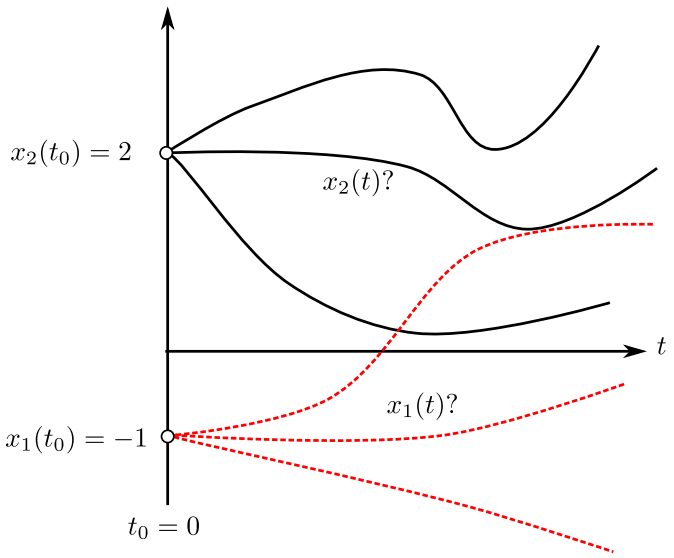



Figure 2: An initial value problem

---

In the context of robotics, we may be interested in how the robot moves as a result of some action over a

short period of time (e.g., if $\mathbf{x}$ represents the joint angles of a robotic leg, we might be interested in how these joint angles evolve during the time when the robot takes a single step). In this case, we're interested in determining $\mathbf{x}(t)$ over a time interval $T$. At other times, we might be particularly interested in what happens "eventually" to the state $\mathbf{x}$ with no clear $T$ in mind. For example, if an aircraft is flying into a wind gust and experiences some turbulence we may want to know how it eventually responds to this disturbance. The state $\mathbf{x}$ (e.g., representing the altitude and pitch angle of the aircraft) may eventually return back to a steady cruise altitude and orientation, or perhaps if the aircraft is poorly designed the turbulence will cause the altitude and orientation to diverge and lead to catastrophic consequences. By studying the equations of motion we can answer such questions.

## Shorthand Notation

In the robotics and control literature, we will often see ODEs written with a shorthand notation that omits the familiar $d/dt$ differentiation symbols you may have seen in an earlier calculus course. Instead, a dot or sometimes a prime symbol is placed directly above or next to the symbol being differentiated (i.e., $\dot{\mathbf{x}}$ or $\mathbf{x}'$) with the understanding that this represents differentiation with respect to time (or another independent variable that is made clear from context). Thus, the following three terms have the same meaning:

$$\dot{\mathbf{x}} = \mathbf{x}' = \frac{d\mathbf{x}}{dt}$$

Our system of first order ODEs (from Eq. 1) is then written as:

$$\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}) \tag{2}$$

## Difference Between State and Configuration

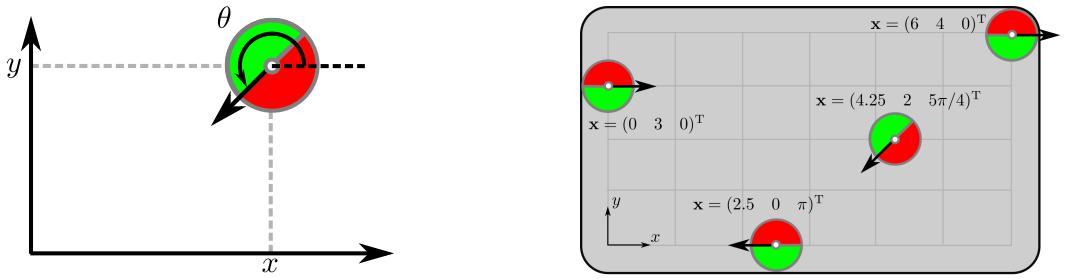
There are some subtle differences between a robot's *state* and it's *configuration*. A configuration is a complete specification of the position of every point of that system. For example, the configuration of a ground robot might be the position of a point on the robot (e.g., its center of mass) and the orientation in which it is facing. A state on the other hand gives a complete specification of not just the position, but also the velocities and other quantities related to the dynamics of the robotic system. The state is usually denoted $\mathbf{x}$ by control theorists, and the configuration is usually denoted $\mathbf{q}$ by roboticists. For this course, we won't distinguish between the two and we will use the two terms interchangeably (denoting both as $\mathbf{x}$).

## Representing the Robot's State/Configuration

Let's step back for a minute and think more about how we define the robot's state/configuration $\mathbf{x}$ in the equations of motion (Eq. 2). The state is the set of variables that represent some aspect of what the robot is doing that we are particularly interested in. For example, if a differential drive ground robot is constrained inside a room and we are interested in controlling its position, then we might define the state/configuration as $\mathbf{x} = (x \quad y \quad \theta)^{\mathrm{T}}$ where $(x, y)$ is the planar position in the room and $\theta$ is the heading (planar orientation angle) of the robot. The choice of configuration variables is not unique. It is determined in part by the robot's geometry, in part by the coordinate system chosen, and in part by the modeler's judgment of what is important for the task at hand. For example, a more sophisticated configuration model of the robot in Fig. 3 might also include the wheel angles. Both the simpler and more complex configurations describe the same robot and are perfectly valid.

The *configuration space* or *C-space* (denoted $\mathcal{C}$) is the space of all possible configurations of the system. It tells us what are the possible values that $\mathbf{x}$ might take on. Suppose the room is 6 units long and 4

units wide (measured from the bottom left corner in Fig. 3) and the differential drive robot can rotate to face any direction. Then the $x$ value can take on a value in the interval $[0, 6]$, the $y$ value can take on a value in the interval $[0, 4]$, and the heading angle of the robot can take on any angle in the interval $[0, 2\pi)$. Putting all of this together, we say that the the configuration space is $\mathcal{C} = [0, 6] \times [0, 4] \times [0, 2\pi)$. To denote that $\mathbf{x}$ is constrained to be inside $\mathcal{C}$ we write $\mathbf{x} \in \mathcal{C}$. The "$\in$" symbol represents "element of" or "is in". Thus we would read the statement "$\mathbf{x} \in \mathcal{C}$" as "$\mathbf{x}$ is in (or is an element of) the configuration space $\mathcal{C}$".




(a) The configuration can be described by two positions and one heading angle: $\mathbf{x} = (x \quad y \quad \theta)^{\mathrm{T}}$

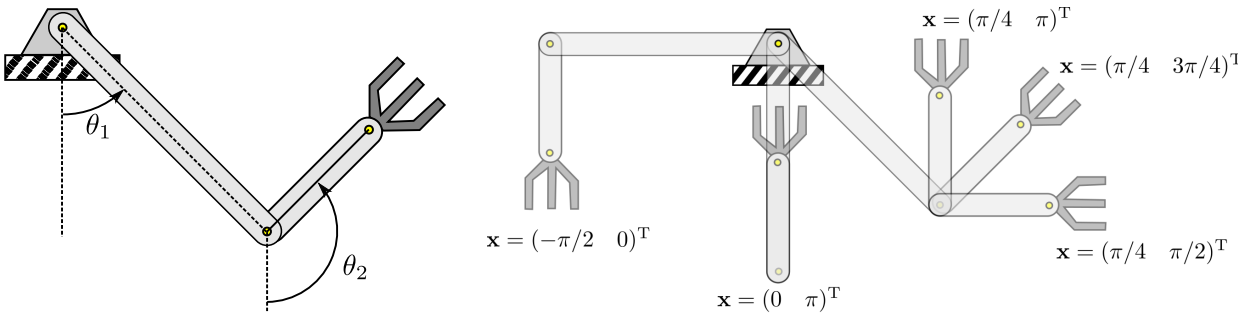
(b) Various example configurations in the configuration space $\mathcal{C} = [0, 6] \times [0, 4] \times [0, 2\pi)$

Figure 3: A planar wheeled mobile robot model constrained to operate in a room

It is important to make the distinction between reference frames and configuration. While reference frames and homogeneous transformations provide a means to express vectors in different coordinate systems, configuration variables describe the robot's degrees of freedom in terms of geometry and position. Although, we may sometimes define the robot's configuration as a certain body-fixed reference frame this is not required.

## Degrees of Freedom

Another example of a configuration space is shown in Fig. 4 – a robotic arm modeled as a double pendulum. The number of *degrees of freedom* is the dimensionality of the configuration space (i.e., the number of independent parameters needed to fully describe the robot's configuration). In the mobile robot example there are three degrees of freedom, and in the double pendulum example there are two.




(a) The configuration can be described by two angles $\mathbf{x} = (\theta_1 \quad \theta_2)^{\mathrm{T}}$

(b) Various example configurations in the configuration space $\mathcal{C} = [0, 2\pi) \times [0, 2\pi)$

Figure 4: A robotic arm modeled as a double pendulum

**Side Note: Set Notation**

In mathematics (and robotics) you will often see variables defined using set theoretic notation. The symbol $\in$ means "element of", $\notin$ means "not element of", the symbol $\subset$ means "subset of". The set of

real, complex and integer number is denoted $\mathbb{R}$, $\mathbb{C}$, and $\mathbb{Z}$, respectively. An interval defined with square brackets $[x, y]$ is "closed" and contains both endpoints $x$ and $y$. An "open" interval with both rounded brackets $(x, y)$ is similar to $[x, y]$ but does not contain the endpoints. For example:

- $x \in X$ (the variable $x$ is an element of the set $X$)
- $x \in \mathbb{R}$ (the variable $x$ is an element of the set of real numbers)
- $X = [0, 1] \subset \mathbb{R}$ (the set $X$ is the interval from 0 to 1 which is a subset of the real numbers)
- $\mathbf{x} \in \mathbb{R}^2$ ($\mathbf{x}$ is a two-dimensional vector of real numbers)
- if $x = 1$ then $x \in [0, 1]$ but $x \notin [0, 1)$
- $X = [1, 2] \times [3, 4]$ (the set $X$ corresponds to all vectors $(x \quad y)^{\mathrm{T}}$ such that $x \in [1, 2]$ and $y \in [3, 4]$.)

---

## Configuration Constraints

Robot configurations can be constrained by the environment or the robot's own design. For example, if a single link robot's configuration (Fig. 5) is represented by the planar $(x, y)$ position of the link's endpoint, then a constraint imposed by the robot's geometry is that $x^2 + y^2 = L$ where $L$ is the length of link. Because this equation is algebraic (i.e., it does not contain any derivatives of the configuration variables) we say this is a *holonomic constraint*. In general, holonomic constraints can be written in the form

$$G(\mathbf{x}) = 0 \tag{3}$$

For the single-link robot with $\mathbf{x} = (x \quad y)^{\mathrm{T}}$, we can write $G(\mathbf{x}) = x^2 + y^2 - L = 0$. When holonomic constraints exist, they sometimes indicate the configuration space has more dimensions than the number of degrees of freedom. Indeed, the single configuration variable $\theta$ can be used to replace the configuration variables $x$ and $y$ in the example of Fig. 5. The single-link robot has only one degree of freedom, and with the choice $\mathbf{x} = \theta$ there is no need to for additional constraints.

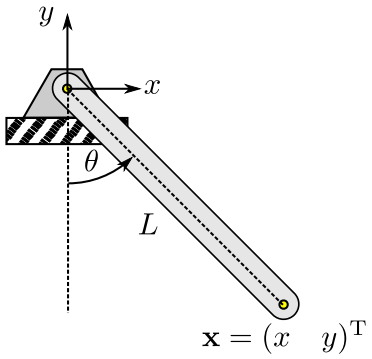



Figure 5: Single-link robot

In some systems, there is a constraint that cannot be expressed geometrically. Returning to the example in Fig. 3, suppose the robot can only drive forward at speed $v$ in the direction it is pointed. The configuration of the robot is $\mathbf{x} = (x \quad y \quad \theta)^{\mathrm{T}}$. To enforce this constraint we need a relationship between the velocities $\dot{x}$ and $\dot{y}$ and the heading angle $\theta$. From vector geometry we know that the $x$ component of the velocity vector is $v \cos\theta$ and the $y$ component is $v \sin\theta$ as shown in Fig. 6.

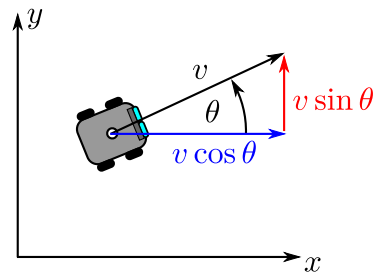



Figure 6: Velocity vector components of a robot heading in the direction $\theta$

Thus we write

$$\dot{x} = v\cos\theta \tag{4}$$

$$\dot{y} = v\sin\theta \tag{5}$$

The constraints (4)-(5) involve derivatives of the configuration variable and are called *nonholonomic* constraints. The general form of a nonholonomic constraint is

$$G(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}, \cdots) = \mathbf{0} \tag{6}$$

For example, for the system Eqs.(4)-(5) we have $\dot{\mathbf{x}} = (\dot{x} \quad \dot{y} \quad \dot{\theta})^{\mathrm{T}}$ and the constraints can be written

$$G(\mathbf{x}, \dot{\mathbf{x}}) = \begin{pmatrix} \dot{x} - v\cos\theta \\ \dot{y} - v\sin\theta \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Some constraints containing derivatives may be *integrable* and can be converted into the form (3) through integration. In that case they are holonomic.

---

**Example: Equations of Motion for a Single-link Robot Arm (Pendulum)**
The equations of motion of a pendulum can be written as

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \underbrace{\begin{pmatrix} x_2 \\ -\left(\frac{g}{L}\right)\sin x_1 - dx_2 \end{pmatrix}}_{\mathbf{f}(t,\mathbf{x})}$$

where $x_1$ is the angle the pendulum makes with the horizontal, $x_2$ is the angular velocity of the pendulum, $g$ is the gravitational constant, $L$ is the length of the pendulum and $d$ is a damping parameter to account for friction.

---

## Simulating Equations of Motion: Euler's Method

Recall that our ultimate goal is to solve the initial value problem and determine how the robot's state evolves with time. We call this solution $\mathbf{x}(t)$ a *trajectory* of the system. A well known numerical approach to integrating ODEs is *Euler's method*. Euler's method begins from the initial state $\mathbf{x}_0$ at the initial time $t_0$. The trajectory history is approximated by a sequence of $N$ discrete states $\mathbf{x}_k = \mathbf{x}(t_k)$ that are separated in time by a timestep $h$ as shown in Fig. 7. The main idea is that we assume the state-rate

$\dot{\mathbf{x}}(t)$ is constant over the timestep $h$. In a sense, we are assuming that successive states $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ are connected by straight line segments, when in fact the ODE tells us the derivative changes continuously during this interval.

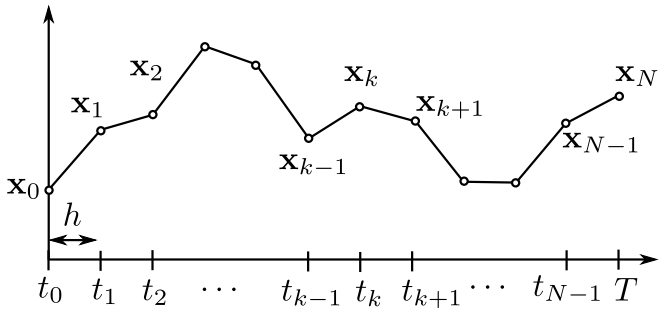



Figure 7: An approximated trajectory

The time intervals at which we are evaluating the trajectory and the corresponding state are given by the recursive relations:

$$t_k = t_{k-1} + h \tag{7}$$

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \mathbf{f}(t_{k-1}, \mathbf{x}_{k-1})h \tag{8}$$

Note that the "slope" of the line connecting $\mathbf{x}_{k-1}$ to $\mathbf{x}_k$ is given by evaluating the state-rate at the previous timestep $\mathbf{f}(t_{k-1}, \mathbf{x}_{k-1})$. This procedure is applied recursively until the final simulation time $T$ is reached. The number of timesteps will be $N = T/h$.

---

**Example: Euler's Method for a Single-link Robot Arm (Pendulum)**
Recall the equations of motion for a pendlum are:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \underbrace{\begin{pmatrix} x_2 \\ -\left(\frac{g}{L}\right)\sin x_1 - dx_2 \end{pmatrix}}_{\mathbf{f}(t,\mathbf{x})}$$

where $x_1$ is the angle the pendulum makes with the horizontal, $x_2$ is the angular velocity of the pendulum, $g = 9.81\text{m/s}^2$ is the gravitational constant, $L = 1.0$ m is the length of the pendulum and $d = 0.5$ is a damping parameter to account for friction. Suppose the pendulum starts at a time $t_0 = 0$ from an angle $x_1(t_0) = 2.0944$ rad (equal to 120 deg.) and is released from rest so the initial angular rate is $x_2(t_0) = 0$ rad/s. The the initial condition is

$$\mathbf{x}(t_0) = \mathbf{x}_0 = \begin{pmatrix} x_1(t_0) \\ x_2(t_0) \end{pmatrix} = \begin{pmatrix} 2.0944 \\ 0.0000 \end{pmatrix} \qquad \text{where} \qquad t_0 = 0$$

Now let's simulate the pendulum's motion for a very short time after it is released. Let's simulate the next 0.04 seconds using a timestep of $h = 0.01$. Thus we will need to use the recursive relations Eqs. 7-8 iteratively $N = T/h = 4$ times to generate the values $\{t_1, t_2, t_3, t_4\}$ and $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$.

Starting from $k = 1$ we use Eqs. 7-8 to obtain:

$$t_1 = t_0 + h \tag{9}$$
$$= 0.0 + 0.01 \tag{10}$$
$$= 0.01 \tag{11}$$
$$\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{f}(t_0, \mathbf{x}_0)h \tag{12}$$
$$= \begin{pmatrix} x_1(t_0) \\ x_2(t_0) \end{pmatrix} + \begin{pmatrix} x_2(t_0) \\ -\left(\frac{g}{L}\right)\sin x_1(t_0) - dx_2(t_0) \end{pmatrix} h \tag{13}$$
$$= \begin{pmatrix} 2.0944 \\ 0.00000 \end{pmatrix} + \begin{pmatrix} 0 \\ -\left(\frac{9.81}{1.0}\right)\sin(2.0944) - 0.5(0) \end{pmatrix} 0.01 \tag{14}$$
$$= \begin{pmatrix} 2.09440 \\ -0.08496 \end{pmatrix} \tag{15}$$

Notice that the angle has not changed but that the angular rate is now non-zero and negative. The pendulum will begin to swing down in the next step. Using Eqs. 7-8 again for $k = 2$

$$t_2 = t_1 + h \tag{16}$$
$$= 0.02 \tag{17}$$
$$\mathbf{x}_2 = \mathbf{x}_1 + \mathbf{f}(t_1, \mathbf{x}_1)h \tag{18}$$
$$= \begin{pmatrix} x_1(t_1) \\ x_2(t_1) \end{pmatrix} + \begin{pmatrix} x_2(t_1) \\ -\left(\frac{g}{L}\right)\sin x_1(t_1) - dx_2(t_1) \end{pmatrix} 0.01 \tag{19}$$
$$= \begin{pmatrix} 2.09440 \\ -0.08496 \end{pmatrix} + \begin{pmatrix} -0.08496 \\ -\left(\frac{9.81}{1.0}\right)\sin(2.09440) - 0.5(-0.08496) \end{pmatrix} 0.01 \tag{20}$$
$$= \begin{pmatrix} 2.09355 \\ -0.16949 \end{pmatrix} \tag{21}$$

Now the pendulum's angle has decreased a bit, and the angular velocity is increasing in the downward direction (the pendulum is speeding up as it swings down). Repeating this process again for $k = 3$

$$t_3 = t_1 + h \tag{22}$$
$$= 0.03 \tag{23}$$
$$\mathbf{x}_3 = \mathbf{x}_2 + \mathbf{f}(t_2, \mathbf{x}_2)h \tag{24}$$
$$= \begin{pmatrix} x_1(t_2) \\ x_2(t_2) \end{pmatrix} + \begin{pmatrix} x_2(t_2) \\ -\left(\frac{g}{L}\right)\sin x_1(t_2) - dx_2(t_2) \end{pmatrix} 0.01 \tag{25}$$
$$= \begin{pmatrix} 2.09355 \\ -0.16949 \end{pmatrix} + \begin{pmatrix} -0.16949 \\ -\left(\frac{9.81}{1.0}\right)\sin(2.09355) - 0.5(-0.16949) \end{pmatrix} 0.01 \tag{26}$$
$$= \begin{pmatrix} 2.09185 \\ -0.25364 \end{pmatrix} \tag{27}$$

The angle decreases and the angular rate increases further (in the negative direction). Lastly, for $k = 4$

$$t_4 = t_2 + h \tag{28}$$
$$= 0.04 \tag{29}$$
$$\mathbf{x}_4 = \mathbf{x}_3 + \mathbf{f}(t_3, \mathbf{x}_3)h \tag{30}$$
$$= \begin{pmatrix} x_1(t_3) \\ x_2(t_3) \end{pmatrix} + \begin{pmatrix} x_2(t_3) \\ -\left(\frac{g}{L}\right)\sin x_1(t_3) - dx_2(t_3) \end{pmatrix} 0.01 \tag{31}$$
$$= \begin{pmatrix} 2.09185 \\ -0.25364 \end{pmatrix} + \begin{pmatrix} -0.25364 \\ -\left(\frac{9.81}{1.0}\right)\sin(2.09185) - 0.5(-0.25364) \end{pmatrix} 0.01 \tag{32}$$
$$= \begin{pmatrix} 2.08931 \\ -0.33745 \end{pmatrix} \tag{33}$$

Since we've only simulated $T = 0.04$ sec., the pendulum has moved very little, but we can see the trend of the pendulum speeding up as it is released from rest. If we implement Euler's Method on a computer, we can easily repeat this procedure to simulate a longer time. Consider the following MATLAB code that simulates the pendulum's motion for $T = 10$ sec.

Listing 1: Euler's Method For Simulating Pendulum Motion

```
% prepare workspace
clear all;close all;clc;

% intial conditions
start_angle = 120*pi/180; %radians
start_angular_rate = 0*pi/180; %radians/sec
x0 = [start_angle start_angular_rate]'; % initial state
t0 = 0.0; %initial time

% simulation parameters
T = 10.0; % total simulation time, sec
h = 0.01; % time-step
N = T/h; % number of time-steps

% constant parameters for pendulum model
g = 9.81; % gravity
L = 1.0; % pendulum length
d = 0.5; % damping factor

% initialize simulation
i = 1; % initial step
% intialize matrix to store the state history
% this will be a 2 x N matrix with each column corresponding to a state
x(:,i) = x0;
% initial vector to store the time history
% this will be a vector with each element corresponding to a state
t(i) = t0;

% iterate through the remaining N steps
```

```
for i=2:1:N
  % evaluate the right hand side of the ODE: (the state-rate)
  xdot(1,1) = x(2,i-1);
  xdot(2,1) = -g/L*sin(x(1,i-1)) - d*x(2,i-1);
  % euler's method, recursive relations:
  x(:,i) = x(:,i-1) + h*xdot;
  t(i) = t(i-1) + h;
end

% plot angle history
figure;
plot(t,x(1,:),'ro-','linewidth',2)
hold on;
plot(t,x(2,:),'bo-','linewidth',2)
set(gca,'FontSize',22)
xlabel('Time (sec)')
legend('Angle (rad.)','Angular Rate(rad/s)');
print(1,'-dpdf','pendulumHistory.pdf')
```

The above code simulates the pendulum's motion and generates the right hand side of Fig. 8. The left hand side of Fig. 8 shows a zoomed in view of the first four iterations of Euler's method that we computed by hand above. Notice that the angle does not change in the first step from $t_0$ to $t_1$, just like we found previously.

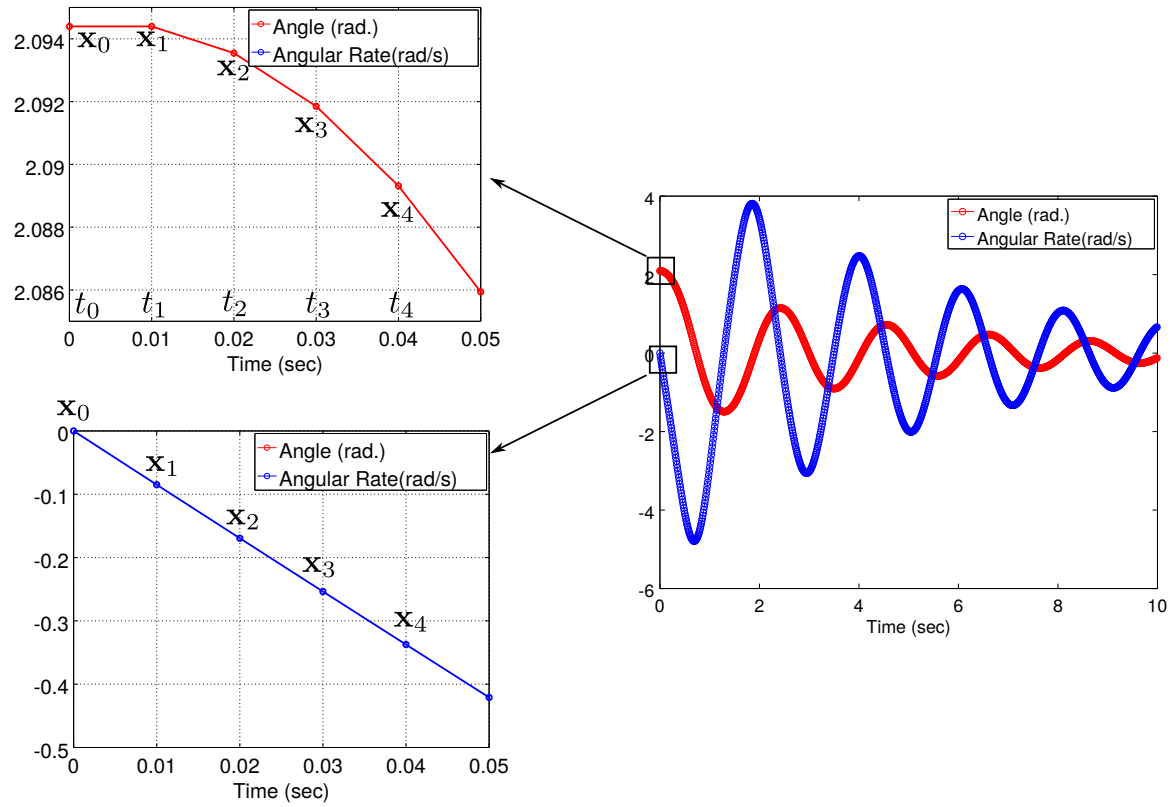



Figure 8: Output from simualting the pendulum's motion using Euler's Method