Snake Robot Locomotion

Biomechanics and Robotics Project

Qizong Wu

1. Introduction

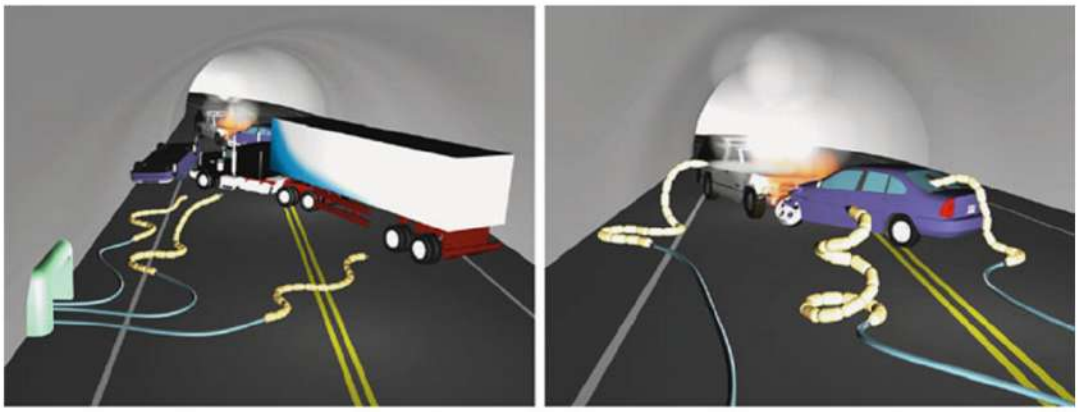
1.1 Background and Motivation

A snake robot is a robotic mechanism designed to move like a biological snake. Inspired by the robustness and stability of biological snake locomotion, snake robots carry the potential of meeting the growing need for robotic mobility in unknown and challenging environments. These mechanisms typically consist of many serially connected joint modules capable of bending in one or more planes. The many degrees of freedom of snake robots make them difficult to control but provide potential locomotion skills in cluttered and irregular environments which surpass the mobility of more conventional wheeled, tracked, and legged robots.

Figure 1.1 The water hydraulic snake robot *Anna Konda*

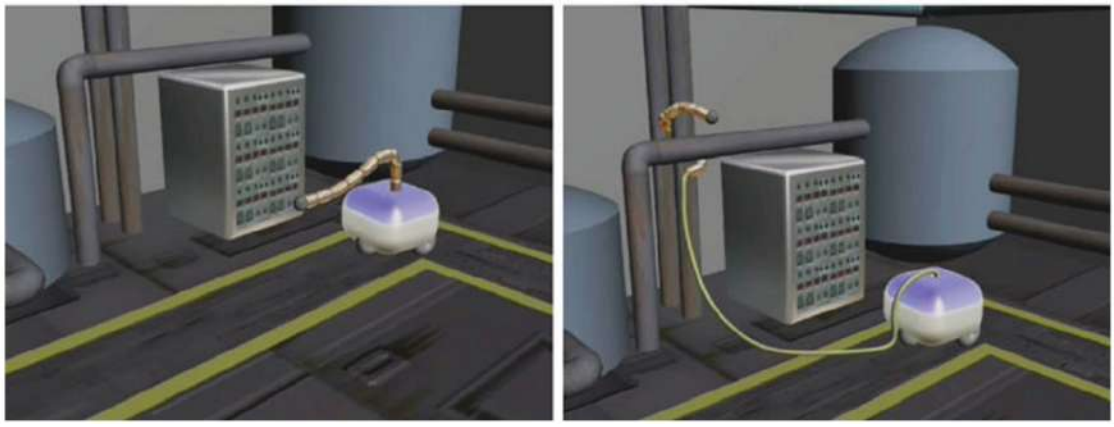
The robot, which was named *Anna Konda*, is shown in Figure 1.1. Anna Konda can move over relatively flat surfaces and can spray water through nozzles in its head. The robot is, however, far from ready for operating in harsh environments.

A very relevant and important application of future snake robots involves search and rescue missions in earthquake areas. In particular, the earthquake in Haiti in January 2010 and the Tsunami which struck Japan in March 2011 are natural disasters in recent time which illustrate the need for technology that can be employed to efficiently locate survivors inside the debris of collapsed buildings. Other potential applications of snake robots include inspection and intervention operations in hazardous environments of industrial plants, manipulator tasks in tight spaces, and space and subsea operations. Figure 1.2 illustrates some of these applications.



(a) Fire fighting inside a road tunnel.

(b) Searcch and rescue after an earthquake.



(c) Inspection and maintenance inside a process plant.

Figure 1.2 Examples of future application of snake robots

1.2 Existing Snake Robots

Several snake robots models have been designed and made before [4,8]. Mainly these multilinked mechanisms get their motion, basically, by electric motors. Currently the systems use motors to move the segments of frame, executing the serpentine motion for the displacement over solid surfaces or water. Some models include little and small wheels to improve the displacement. Others kind of robots use "caterpillars" in every segment, which are

driven by motors. Also, other mechanisms are able to execute motion over water, acquiring hydro-dynamical properties of motion in the segment frames by motors also.

Although electric motors are the mainly manner to drive snake robots, other biologically inspired robots were developed. One example is one swimming snake-like robot developed by Tokyo Institute of Technology (TIT) [1]. The swimming snake-like robot gets its motion by the deflection of IPMC films, this strips joins the three segments of the system. These IPMC films or strips have the property of deforming (bending) when a voltage difference is applied to the strips. This bending deformation in conjunction to the tangential and normal friction forces causes the segments motion and then the propulsion effect for the displacement over the water. It is a small model, about a few centimeters long, and it is light because the frame segments were made of styrene foam.

2. Project Robot Description

Figure 2.1 shows the locomotion of a snake and it is motivated by the ground contact force.

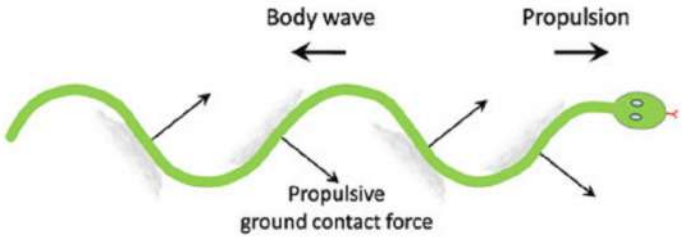


Figure 2.1 Snake locomotion

To develop a snake robot, composed of eight uniform segments or modules that must be capable of movement in two dimensions on the ground. Each joint could be actuated by a motor. Also we must properly join the complete arrangement of modules to obtain the desired deformation, a particular shape, and complete motion or displacement.

The goal of this project is to simulate a modular snake-like robot with the ability of moving itself in forward direction imitating the locomotion of a snake, and specially the serpentine locomotion. This robot must imitate the snake's movement as accurate as possible.

To achieve the goal design conditions that the system must satisfy need to be determined. First, we analyzed the mathematical relations of serpentine movements and displacement [3] to acquire the knowledge of how the system will work. Then we must design a prototype of a single segment of the robot. It includes the frame of the segment only, and the drawings that showed the dimensions. After that we simulate the model in Matlab to see the motion of the robot and plot the angular velocity of each joint given desired forward speed.

3. Mathematical Model for Serpentine Motion

3.1 Snake locomotion

We can mention the contribution of J. Gray [6] in *The Mechanism of Locomotion in Snakes* and S. Hirose [4] in *Biologically Inspired Robots* in the fields of zoology and mechanical engineering which helped us to understand the locomotion of snakes. There are 3 mayor gaits used by snakes to move: (1) lateral undulation; (2) concertina; (3) modified concertina.

Lateral undulation, also called serpentine locomotion, is used by the animal in grass or in irregular surfaces; forward movement can be observed in this type of gait and is caused by a series of sinusoidal curves along the body of the snake where the body and tail follows the path taken by the head. The concertina movement is used when the snake is confined within a channel. When snakes are in open field and moving in a smooth surface a variation of the concertina movement is used. In this variation two parts of the body are at rest relative to the ground and the curves are restricted to the central part. Knowing how snakes move will help us in the design of the robot itself because it can be flexible or modular.

We decided to implement the undulatory locomotion or serpentine locomotion. S. Hirose studied the locomotion of snakes and derived the equations for this type of locomotion. In addition Hirose studied the key property in snakes in achieving serpentine locomotion, which is the difference in the friction coefficients for the tangential and normal directions with respect to the snake's body. The normal friction will prevent the snake from slipping sideways, thus the normal friction coefficient must be large and the tangential friction coefficient must be small to enable the snake robot to move in forward direction. Hirose's "Serpenoid Curve" is described by the following equation

$$x(s) = \int_0^s \cos(\tau)\, d\sigma$$

$$y(s) = \int_0^s \sin(\tau)\, d\sigma \qquad (1)$$

$$\tau = a\cos(b\sigma) + c\sigma$$

where $(x(s), y(s))$ are the coordinates of the point along the curve at arc length s from the origin, and where a, b and c are positive scalars; different types of serpenoid curves can be defined by varying them. σ represents the position on the curve and the speed of motion can be defined by the speed of changes in σ. a specifies undulation, b periods and c the angular speed.

Equation (1) is for a continuous body like the snake's body. We need an approximation to these equations because our preliminary snake robot consists of 8 rigid segments connected by 7 joints. According to what M. Saito, M. Fukaya and T. Iwasaki concluded in their *Modeling, Analysis and Synthesis of Serpentine Locomotion with Multilink Robotic Snake* [2], we have an approximation to equation (1):

$$x_i = \sum_{k=1}^{i} \frac{1}{n} \cos\left(a \cdot \cos\left(\frac{kb}{n}\right) + \frac{kc}{n}\right)$$

$$y_i = \sum_{k=1}^{i} \frac{1}{n} \sin\left(a \cdot \cos\left(\frac{kb}{n}\right) + \frac{kc}{n}\right) \qquad (2)$$

Equation (2) approximates Hirose's equation to n-1 joints connected by n segments. Figure 3.1 shows different serpenoid curves obtained by varying these parameters with 20 segments.

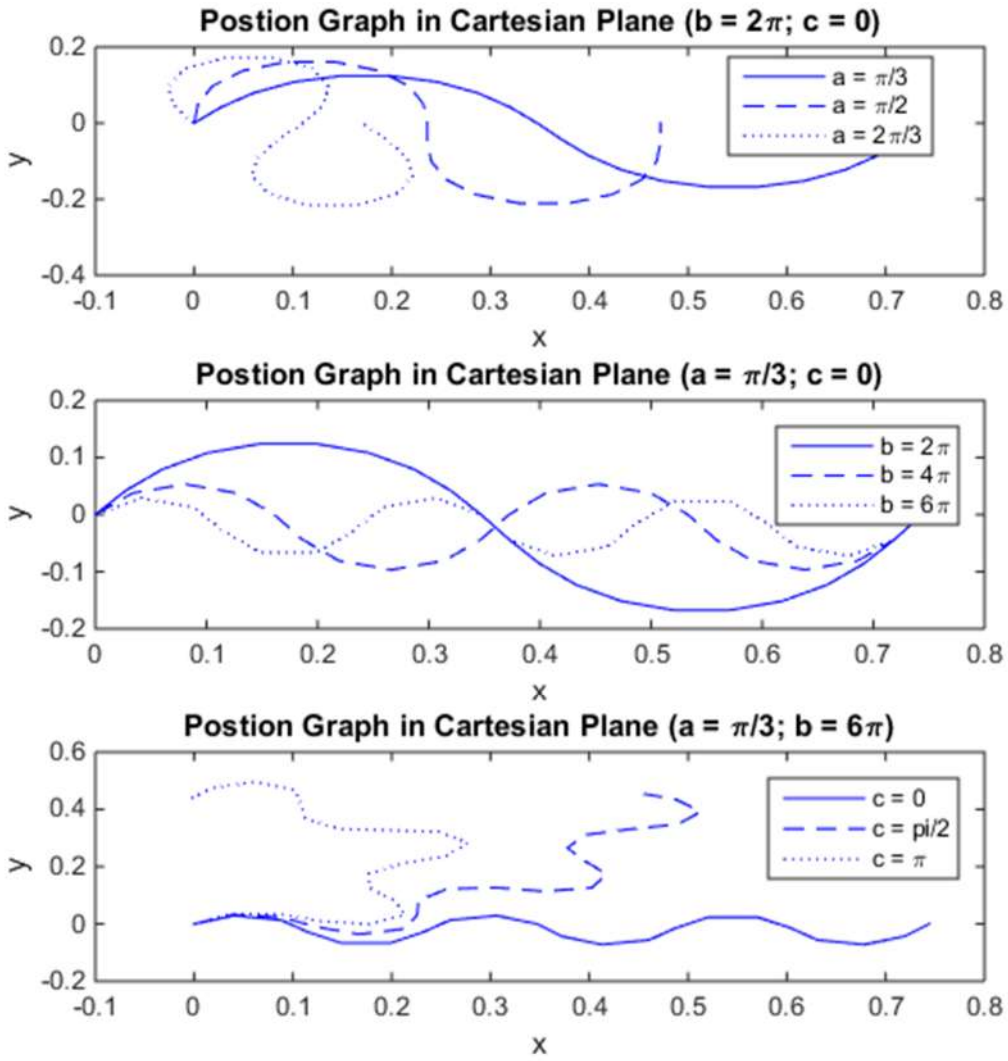


Figure 3.1 Different types of serpenoid curves

The actual model with 8 segments and 7 joints is as follows.

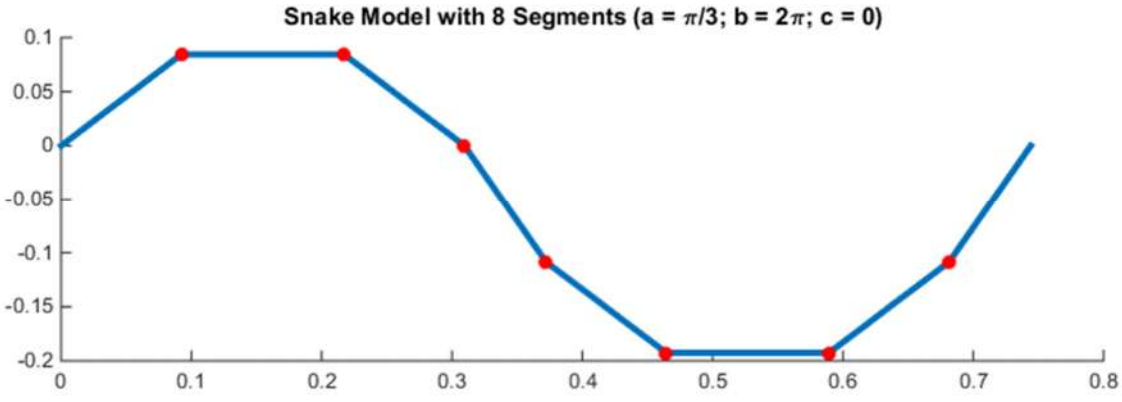


Figure 3.2 Actual model of 8-link snake-robot

To make the robot move in the serpentine way we need to change the joint angles in a certain manner. From [5] we have that the joint angles for every i joint changes with time accordingly to the following expression:

$$\theta_i(t) = A \cdot \sin(2\pi \cdot f \cdot t + 2\pi \cdot \Delta\phi \cdot (i-1)) \quad (3)$$

As we can see, the previous equation does not contain any of the parameters defined in equation (1), namely, a, b, and c. From [3] we have a formula similar to equation (3), which describes how the relative angle between joints changes with time. The gait pattern lateral undulation is achieved by moving the joints of a planar snake robot according to

$$\phi_i(t) = \alpha \cdot \sin(\omega t + (i-1)\beta), \quad (4)$$

where i ∈ {1,...,n − 1}, α and ω are the amplitude and angular frequency, respectively, of the sinusoidal joint motion, β determines the phase shift between the joints.

3.2 Mathematical Drag Force (Anisotropic Viscous Friction) Relationships

We assume that the viscous ground friction forces act on the CM of the links only. Some parameters are presented in Figure 3.3.

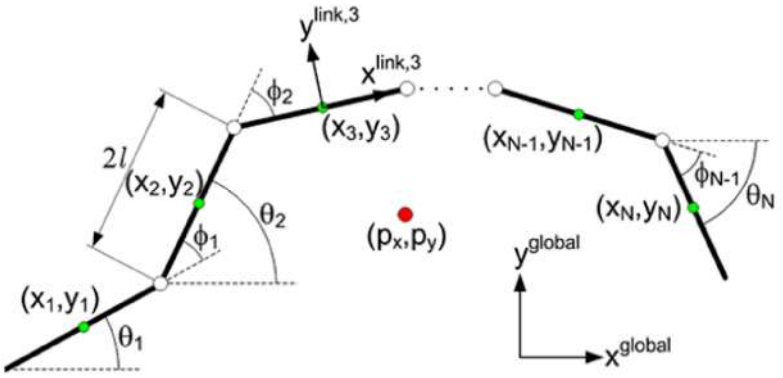


Figure 3.3 Snake robot kinematics parameters

We present the viscous friction model for the different cases of isotropic versus anisotropic viscous. Under anisotropic friction conditions, a link has two viscous friction coefficients, c_t and c_n, describing the friction force in the tangential (along link x axis) and normal (along link y axis) directions of the link, respectively. We define the viscous friction force on link I in the local link frame

$$\boldsymbol{f}_{R,i}^{link,i} = -\begin{bmatrix} c_t & 0 \\ 0 & c_n \end{bmatrix} \boldsymbol{v}_i^{link,i},$$

where $\boldsymbol{v}_i^{link,i} \in \mathbb{R}^2$ is the is the link velocity expressed in the local link frame. Using rotational matrix we can express the global frame viscous friction force on link I as

$$\boldsymbol{f}_{R,i} = \boldsymbol{f}_{R,i}^{global} = \boldsymbol{R}_{link,i}^{global} \boldsymbol{f}_{R,i}^{link,i}$$

$$= -\boldsymbol{R}_{link,i}^{global} \begin{bmatrix} c_t & 0 \\ 0 & c_n \end{bmatrix} \boldsymbol{v}_i^{link,i}$$

$$= -\boldsymbol{R}_{link,i}^{global} \begin{bmatrix} c_t & 0 \\ 0 & c_n \end{bmatrix} \left(\boldsymbol{R}_{link,i}^{global}\right)^T \begin{bmatrix} \dot{x}_l \\ \dot{y}_l \end{bmatrix}$$

$$\boldsymbol{f}_{R,i} = \begin{bmatrix} c_t \cos^2\theta_i + c_n \sin^2\theta_i & (c_t - c_n) sin\theta_i \cos\theta_i \\ (c_t - c_n) sin\theta_i \cos\theta_i & c_t \sin^2\theta_i + c_n cos^2\theta_i \end{bmatrix} \begin{bmatrix} \dot{x}_l \\ \dot{y}_l \end{bmatrix}$$

3.3 Joint Angles and body velocity by non-side-slippery condition

To make the tangential friction coefficient c_t small enough and normal friction coefficient c_n large enough, we assume that there is no side slipping when the robot moves. For this assumption we model the robot based on where each link has passive wheels that prevent the links from side slipping but have no effect in forward movement, which is similar to the configuration in Figure 3.4.

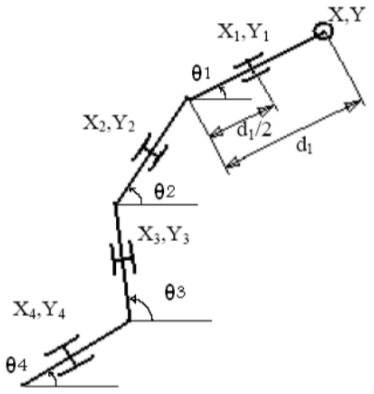


Figure 3.4 A snake robot modeled with passive wheels

For a robot with same link length d, holonomic constraints for joints are presented by

$$x_i = x - \sum_{j=1}^{i-1} d\cos\theta_j - \frac{d}{2} cos\theta_i$$

$$y_i = y - \sum_{j=1}^{i-1} d\sin\theta_j - \frac{d}{2} sin\theta_i$$

$$\dot{x}_l = \dot{x} + \sum_{j=1}^{i-1} d\sin\theta_j \cdot \dot{\theta}_j + \frac{d}{2} sin\theta_i \cdot \dot{\theta}_l$$

$$\dot{y}_l = \dot{y} - \sum_{j=1}^{i-1} d\cos\theta_j \cdot \dot{\theta}_j - \frac{d}{2} cos\theta_i \cdot \dot{\theta}_l \quad (5)$$

where $i = 1,2,\dots,n$. For each link, nonholonomic constraint of no side slipping is described by

$$\dot{x}_l sin\theta_i - \dot{y}_l cos\theta_i = 0 \quad (6)$$

Substituting (6) in (5) gives

$$\frac{d}{2}\dot{\theta}_l + \sum_{j=1}^{i-1} d\cos(\theta_i - \theta_j) \cdot \dot{\theta}_j = cos\theta_i \cdot \dot{y} - sin\theta_i \cdot \dot{x}$$

Defining $S_i = sin\theta_i,\ C_i = cos\theta_i, C_{ij} = \cos(\theta_i - \theta_j)$, the entire constraint equations are

$$\begin{bmatrix} \frac{d}{2} & 0 & 0 & & \\ dC_{21} & \frac{d}{2} & 0 & \cdots & 0 \\ dC_{31} & dC_{32} & \frac{d}{2} & & \\ & \vdots & & \ddots & \vdots \\ dC_{n1} & dC_{n2} & dC_{n3} & \cdots & \frac{d}{2} \end{bmatrix}_{n\times n} \cdot \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \vdots \\ \dot{\theta}_n \end{bmatrix} = \begin{bmatrix} -S_1 & C_1 \\ -S_2 & C_2 \\ \vdots & \vdots \\ -S_n & C_n \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \vdots \\ \dot{\theta}_n \end{bmatrix} = F_\theta \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}$$

Hence, for joint angles we have

$$\Rightarrow \begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \\ \vdots \\ \dot{\phi}_n \end{bmatrix} = \begin{bmatrix} 1 & -1 & & & \\ & 1 & -1 & \cdots & 0 \\ & & 1 & & \\ & \vdots & & \ddots & \vdots \\ & 0 & & \cdots & 1 & -1 \end{bmatrix}_{(n-1)\times n} \cdot \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \vdots \\ \dot{\theta}_n \end{bmatrix} = F_\phi \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}$$

where

$$F_\phi = \begin{bmatrix} 1 & -1 & & & & \\ & 1 & -1 & \cdots & & 0 \\ & & 1 & & & \\ & \vdots & & \ddots & & \vdots \\ & 0 & & \cdots & 1 & -1 \end{bmatrix} \begin{bmatrix} \frac{d}{2} & 0 & 0 & & \\ dC_{21} & \frac{d}{2} & 0 & \cdots & 0 \\ dC_{31} & dC_{32} & \frac{d}{2} & & \\ & \vdots & & \ddots & \vdots \\ dC_{n1} & dC_{n2} & dC_{n3} & \cdots & \frac{d}{2} \end{bmatrix} \begin{bmatrix} -S_1 & C_1 \\ -S_2 & C_2 \\ \vdots & \vdots \\ -S_n & C_n \end{bmatrix} \quad (7)$$

Equation (7) gives joints angular velocities according to the head's velocity, based on the whole body configuration so that there is no side slipping. This way, one can restrict the head to a serpenoid curve. The head velocity is then defined and the desired motion is obtained by (7) [7].

4. Simulation in Matlab of Serpentine Locomotion

4.1 Joint Angles in 20 seconds

We coded a program in Matlab that helped us determine the change in the i-th joint angle with respect to time. The Matlab script is attached in appendix. For 8 segments (n = 8), the changes in joint angles in 20 seconds are shown in Figure 4.1.

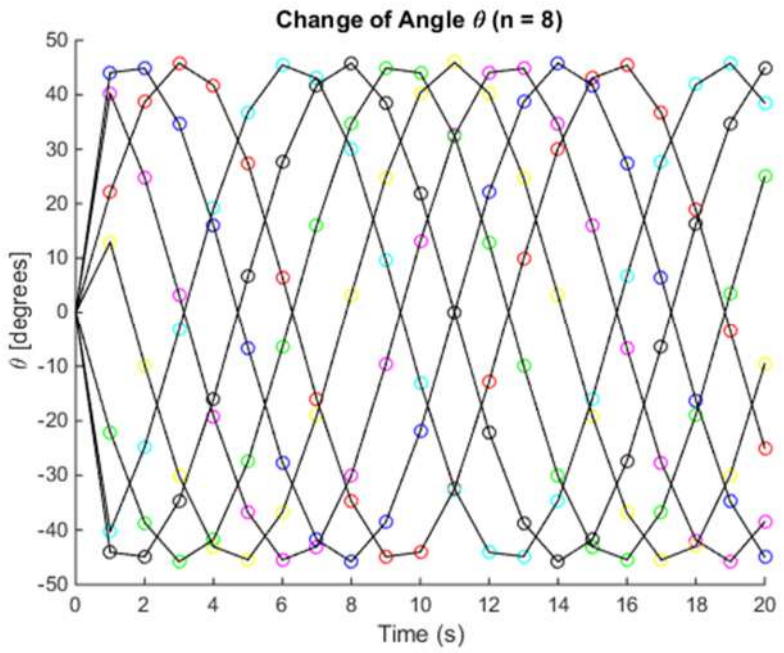


Figure 4.1 Change of angles in 20 seconds

4.2 Robot Shape in 20 seconds

In appendix there is also a script that shows us how the snake changes its shape every second for 20 seconds. The change in shape of an 8-link snake-like robot can be seen in Figure 4.2. In this figure the reader can observe how the sinusoidal wave travels through the body of the snake starting in the head and propagating along its body and ending in the snake's tail. This move is done by changing the joint angles accordingly to equation (4).

During the forward motion, the traveling speed of the sinusoidal wave along the snake's body can be changed by changing the angular frequency ω in equation (4) and the tail is fixed at coordinates (0, 0) to make the simulation easier to calculate and observe. Form the simulation we can observe that the amplitude of the sinusoidal wave diminishes as it travels to the tail.

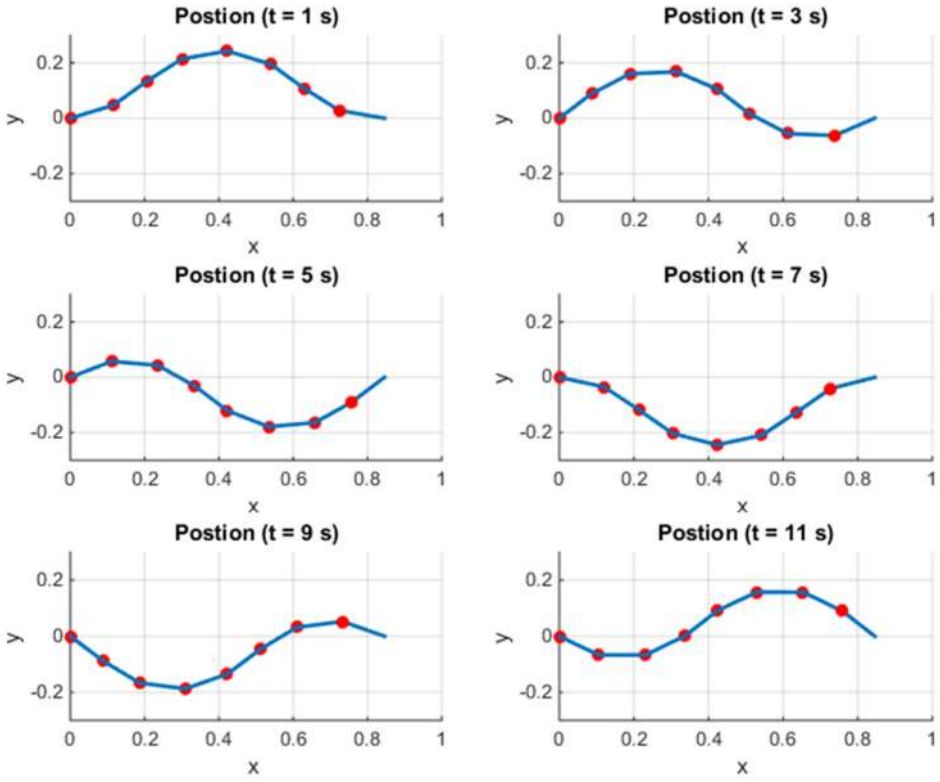


Figure 4.2 Change in shape

4.3 Angular Velocity given Desired Forward Speed

From equation (7) we calculated the angular velocity of each joint given a forward speed of Vx = 2 m/s, Vy = 0. The angular velocities are shown as follows.

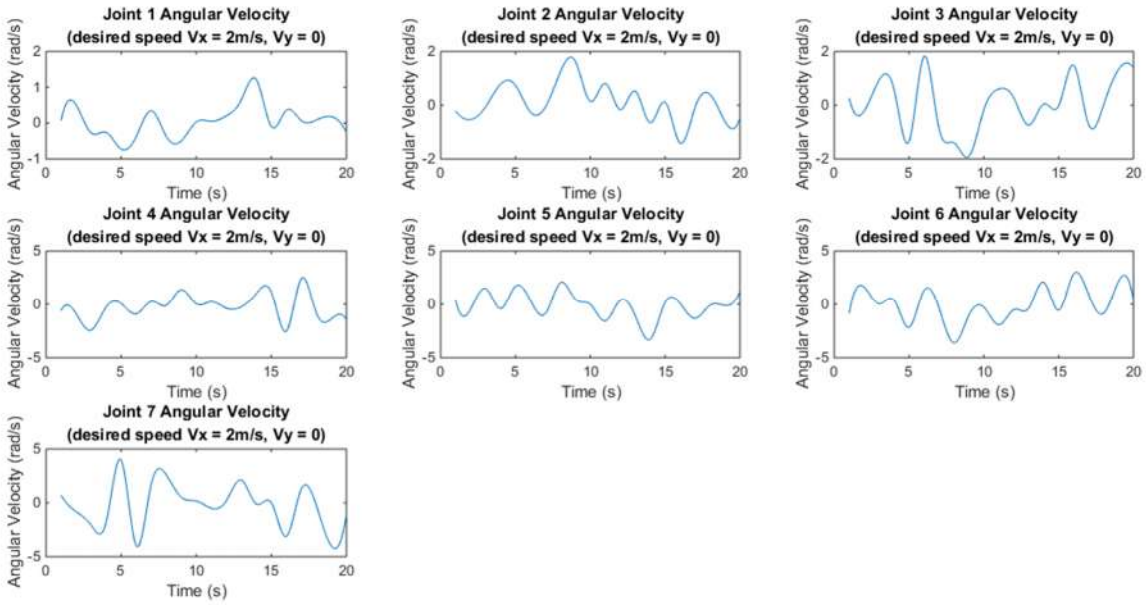


Figure 4.3 Joint angular velocities when Vx = 2 m/s

5. Summary and Future Work

In this project, the background information of snake robot was first introduced. Then in section 2 the project goal was presented. In section 3, the mathematics of a serpentine movement including the locomotion analysis, the anisotropic viscous friction analysis, and the relationship between joint angular velocity and forward speed was shown. In section 4, we assumed the robot uses passive wheels (no side slipping) and simulated the model in Matlab to see the shape of the robot and the change of angles given an 8-liink snake robot with certain dimension.

Wheeled robots are the most efficient, because an ideal rolling (but not slipping) wheel loses no energy. This is in contrast to legged robots which suffer an impact with the ground at heel strike and lose energy as a result. In this project we only analyzed and simulated the robot locomotion in x-y plane. Other locomotion such as climbing and control of the snake-like robot remain to realize in the future.

6. Reference

[1] Masaki Yamakita, Norihiro Kamamichi, Takahiro Kozuki, Kinji Asaka, Zhi-Wei Luo, *A Snake-Like Swimming Robot Using IPMC Actuators and verification of Doping Effect*, Tokyo Institute of Technology, Research Institute of Cell Eng., Bio-Mimetic Controlresearch Center

[2] Richard C. Dort & Robert H. Bishop, "*Modern Control Systems*", 9th edition; Prentice Hall

[3] M. Saito, M. Fukaya and T. Iwasaki, "*Modeling, Analysis and Synthesis of Serpentine Locomotion with a Multilink Robotic Snake"*

[4] Shingeo Hirose, "*Biologically Inspired Robots"*, Oxford University Press, 1987.

[5] Alessandro Crespi, Andre Badertsher, Andre Guignard, Auke Jan Ijspeert, "*AmphiBot I: an amphibious snake-like robot"*, School of Computer and Communication Sciences, Swiss Federal Institute of Technology

[6] J. Gray, "*The Mechanism of Locomotion in Snakes"*, Department of Zoology, University of Cambridge, 1946.

[7] Mohammad Dehghani, M. J. Mahjood, "*A Modified Serpenoid Equation for Snake Robots"*, Center for Mechatronics and Automation School of Mechanical Engineering, Univ. of Tehran, International Conference on Robotics and Biomimetics, Bangkok, Thailand, February 21 - 26, 2009

[8] Kristin Y. Pettersen, Jan Tommy Gravdahl, *"Snake Robots: Modelling, Mechatronics, and Control"*, Yoseph Bar-Cohen, 2013

7. Appendix

Matlab Script:

```
%Model the Serpenoid curve that
%---- Use Serpentine locomotion
%---- Depend on a, b and c parameters
%---- by Qizong Wu
clc
clear all
% Snake Locomotion Parameters
n =20;          % number of segments
L = 32;          % lenght of the snake (not used for now)
a1 = pi/3;       % degree of undulation
a2 = pi/2;
a3 = pi*2/3;
a1DEG = a1*(180/pi);
a2DEG = a2*(180/pi);
a3DEG = a3*(180/pi);
b1 = 2*pi;        % periods of unit lenghts
b2 = 4*pi;
b3 = 6*pi;
c1 = 0;           % changes the direction of propagation
c2 = pi/2;
c3 = pi;
omega=.5         % in rad/s
frequency= omega/(2*pi) %in Hz
lag = (2*pi)/n  % in rad
% The X-Y position of the center of each segment can
% be aproximated by the following code:
x1(1) = 0;       %For the origin or tail position
y1(1) = 0;       %For the origin or tail position
x2(1) = 0;
y2(1) = 0;
x3(1) = 0;
y3(1) = 0;
%position is plotted in the graph
%to calculate the remaining x-y positions following the formulas
figure(1)
% a varies
subplot(3,1,1)
for k=1:n
    x1(k+1) = x1(k)+(1/n)*cos(a1*cos(k*b1/n)); %x position for i-th segment
    y1(k+1) = y1(k)+(1/n)*sin(a1*cos(k*b1/n)); %y position for i-th segment
    x2(k+1) = x2(k)+(1/n)*cos(a2*cos(k*b1/n)); %x position for i-th segment
    y2(k+1) = y2(k)+(1/n)*sin(a2*cos(k*b1/n)); %y position for i-th segment
    x3(k+1) = x3(k)+(1/n)*cos(a3*cos(k*b1/n)); %x position for i-th segment
    y3(k+1) = y3(k)+(1/n)*sin(a3*cos(k*b1/n)); %y position for i-th segment
end
plot(x1,y1,'b-',x2,y2,'b--',x3,y3,'b:'); %plot the x-y coordinates
xlabel('x')
ylabel('y')
title('Postion Graph in Cartesian Plane (b = 2\pi; c = 0)')
legend('a = \pi/3','a = \pi/2','a = 2\pi/3')
% b varies
subplot(3,1,2)
for k=1:n
```

```
    x1(k+1) = x1(k)+(1/n)*cos(a1*cos(k*b1/n)); %x position for i-th segment
    y1(k+1) = y1(k)+(1/n)*sin(a1*cos(k*b1/n)); %y position for i-th segment
    x2(k+1) = x2(k)+(1/n)*cos(a1*cos(k*b2/n)); %x position for i-th segment
    y2(k+1) = y2(k)+(1/n)*sin(a1*cos(k*b2/n)); %y position for i-th segment
    x3(k+1) = x3(k)+(1/n)*cos(a1*cos(k*b3/n)); %x position for i-th segment
    y3(k+1) = y3(k)+(1/n)*sin(a1*cos(k*b3/n)); %y position for i-th segment
end
plot(x1,y1,'b-',x2,y2,'b--',x3,y3,'b:'); %plot the x-y coordinates
xlabel('x')
ylabel('y')
title('Postion Graph in Cartesian Plane (a = \pi/3; c = 0)')
legend('b = 2\pi','b = 4\pi','b = 6\pi')
% c varies
subplot(3,1,3)
for k=1:n
    x1(k+1) = x1(k)+(1/n)*cos(a1*cos(k*b3/n)+k*c1/n); %x position for i-th segment
    y1(k+1) = y1(k)+(1/n)*sin(a1*cos(k*b3/n)+k*c1/n); %y position for i-th segment
    x2(k+1) = x2(k)+(1/n)*cos(a1*cos(k*b3/n)+k*c2/n); %x position for i-th segment
    y2(k+1) = y2(k)+(1/n)*sin(a1*cos(k*b3/n)+k*c2/n); %y position for i-th segment
    x3(k+1) = x3(k)+(1/n)*cos(a1*cos(k*b3/n)+k*c3/n); %x position for i-th segment
    y3(k+1) = y3(k)+(1/n)*sin(a1*cos(k*b3/n)+k*c3/n); %y position for i-th segment
end
plot(x1,y1,'b-',x2,y2,'b--',x3,y3,'b:'); %plot the x-y coordinates
xlabel('x')
ylabel('y')
title('Postion Graph in Cartesian Plane (a = \pi/3; b = 6\pi)')
legend('c = 0','c = pi/2','c = \pi')

%% 8 Segments Snake Model
clc
clear
hold on
n =8                %number of segments
L = 32;             %lenght of the snake (not used for now)
a = pi/3            %degree of undulation
aDEG = a*(180/pi)
b = 2*pi            %periods of unit lenghts
c = 0;              %changes the direction of propagation
omega=.5            %in rad/s
frequency= omega/(2*pi) %in Hz
lag = (2*pi)/n      %in rad
% The X-Y position of the center of each segment can
% be aproximated by the following code:
x(1) = 0;           %For the origin or tail position
y(1) = 0;           %For the origin or tail position
%position is plotted in the graph
%to calculate the remaining x-y positions following the formulas
%in the paper "Snake Analysis"
figure(2)
for i=1:1:n+1
    for k=1:1:i
        x(k+1) = x(k)+(1/n)*cos(a*cos(k*b/n)); %x position for i-th segment
        y(k+1) = y(k)+(1/n)*sin(a*cos(k*b/n)); %y position for i-th segment
        if i<=n
        line([x(k),x(k+1)],[y(k),y(k+1)],'LineWidth',3)
        end
    end
```

```
hold on
end
plot(x(2:8),y(2:8),'ro','MarkerFaceColor','r'); %plot the x-y coordinates
title('Snake Model with 8 Segments (a = \pi/3; b = 2\pi; c = 0)')
%pause(.05); %pause to see it like an animation
hold off
%% Calculate the change in theta with time
% Theta angles for each joint changing with time for 20 seconds
figure(3)
for t=1:1:20
for i=1:1:n
theta(t,i) = 2*a*(180/pi)*abs(sin(b/(2*n)))*sin(omega*t+(b/n)*(i-1));
end
%Plot the theta angle with time
hold on
plot(t,theta(t,1),'-or')
plot(t,theta(t,2),'-ob')
plot(t,theta(t,3),'-om')
plot(t,theta(t,4),'-oy')
plot(t,theta(t,5),'-og')
plot(t,theta(t,6),'-ok')
plot(t,theta(t,7),'-oc')
xlabel('Time (s)')
ylabel('\theta [degrees]')
title('Change of Angle \theta (n = 8)')
end
for i=1:1:n-1
line([0,1],[0,theta(1,i)],'color','k')
for t=1:1:19
line([t,t+1],[theta(t,i), theta(t+1,i)],'color','k')
end
end
hold off

%% 6 plots of Snake motion
x(1) = 0; %For the origin or tail position
y(1) = 0; %For the origin or tail position
%position is plotted in the graph
figure(4)
%to calculate the remaining x-y positions following the formulas
%in the paper "Snake Analysis"
for t=1:2:11
%figure
subplot(3,2,(t+1)/2)
    for i=1:1:n
        for k=1:1:i
            x(k+1) = x(k)+(1/n)*cos(theta(t,k)*(pi/180)); %x position for i-th
segment
            y(k+1) = y(k)+(1/n)*sin(theta(t,k)*(pi/180)); %y position for i-th
segment
                if i<=n
                    line([x(k),x(k+1)],[y(k),y(k+1)],'LineWidth',2)
                end
        end
        hold on
        plot(x(k),y(k),'ro','MarkerFaceColor','r'); %plot the x-y coordinates
        xlabel('x ')
```

```
        ylabel('y ')
        title(['Postion (t = ',num2str(t),' s)'])
        axis([0 1 -.3 .3])
        grid on
    end
    mov(t) = getframe;
end
%% Snake motion Movie
x(1) = 0; %For the origin or tail position
y(1) = 0; %For the origin or tail position
%plot(x(1),y(1),'bo') %plot the tail position in the graph
hold on %hold this position in the graph when another point
%position is plotted in the graph

%to calculate the remaining x-y positions following the formulas
%in the paper "Snake Analysis"
for t=1:20
figure

    for i=1:1:n
        for k=1:1:i
            x(k+1) = x(k)+(1/n)*cos(theta(t,k)*(pi/180)); %x position for i-th
segment
            y(k+1) = y(k)+(1/n)*sin(theta(t,k)*(pi/180)); %y position for i-th
segment
                if i<=n
                    line([x(k),x(k+1)],[y(k),y(k+1)],'LineWidth',2)
                end
        end
        hold on
        plot(x(k),y(k),'ro','MarkerFaceColor','r'); %plot the x-y coordinates
        xlabel('x ')
        ylabel('y ')
        title('Shape of a 8-link Snake Robot')
        axis([0 1 -.3 .3])
        grid on
    end
    mov(t) = getframe;
end
% Moving animation
figure(33)
pause(2);
movie(mov,1,1)
% %to calculate the delay in seconds of the control signals
% % 'Time delay between square-wave control signals.....................'
delay = 2*pi*(b/(n*2*pi))/omega
%% Angular Velocity Calculation
D = [1 -1 0 0 0 0 0 0;
     0 1 -1 0 0 0 0 0;
     0 0 1 -1 0 0 0 0;
     0 0 0 1 -1 0 0 0;
     0 0 0 0 1 -1 0 0;
     0 0 0 0 0 1 -1 0;
     0 0 0 0 0 0 1 -1];
L = 0.32;
tt = 20;
```

```
for i = 1:tt
    L_m(:,:,i) = [L/2 0 0 0 0 0 0 0;
              L*cos(theta(i,2)-theta(i,1)) L/2 0 0 0 0 0 0;
              L*cos(theta(i,3)-theta(i,1)) L*cos(theta(i,3)-theta(i,2)) L/2 0 0 0 0
0;
              L*cos(theta(i,4)-theta(i,1)) L*cos(theta(i,4)-theta(i,2))
L*cos(theta(i,4)-theta(i,3)) L/2 0 0 0 0;
              L*cos(theta(i,5)-theta(i,1)) L*cos(theta(i,5)-theta(i,2))
L*cos(theta(i,5)-theta(i,3)) L*cos(theta(i,5)-theta(i,4)) L/2 0 0 0;
              L*cos(theta(i,6)-theta(i,1)) L*cos(theta(i,6)-theta(i,2))
L*cos(theta(i,6)-theta(i,3)) L*cos(theta(i,6)-theta(i,4)) L*cos(theta(i,6)-
theta(i,5)) L/2 0 0;
              L*cos(theta(i,7)-theta(i,1)) L*cos(theta(i,7)-theta(i,2))
L*cos(theta(i,7)-theta(i,3)) L*cos(theta(i,7)-theta(i,4)) L*cos(theta(i,7)-
theta(i,5)) L*cos(theta(i,7)-theta(i,6)) L/2 0;
              L*cos(theta(i,8)-theta(i,1)) L*cos(theta(i,8)-theta(i,2))
L*cos(theta(i,8)-theta(i,3)) L*cos(theta(i,8)-theta(i,4)) L*cos(theta(i,8)-
theta(i,5)) L*cos(theta(i,8)-theta(i,6)) L*cos(theta(i,8)-theta(i,7)) L/2]
    SC(:,:,i) = [-sin(theta(i,1)) cos(theta(i,1));
          -sin(theta(i,2)) cos(theta(i,2));
          -sin(theta(i,3)) cos(theta(i,3));
          -sin(theta(i,4)) cos(theta(i,4));
          -sin(theta(i,5)) cos(theta(i,5));
          -sin(theta(i,6)) cos(theta(i,6));
          -sin(theta(i,7)) cos(theta(i,7));
          -sin(theta(i,8)) cos(theta(i,8))];
    F(:,:,i) = D * L_m(:,:,i) * SC(:,:,i);
    phid(:,i) = F(:,:,i) * [2;0];
end
figure(6)
subplot(3,3,1)
xt = 1:20;
yt = phid(1,:);
xt1 = 1:0.1:20;
yt1 = spline(xt,yt,xt1);
plot(xt1,yt1)
xlabel('Time (s)')
ylabel('Angular Velocity (rad/s)')
title({'Joint 1 Angular Velocity';'(desired speed Vx = 2m/s, Vy = 0)'})

subplot(3,3,2)
xt = 1:20;
yt = phid(2,:);
xt1 = 1:0.1:20;
yt1 = spline(xt,yt,xt1);
plot(xt1,yt1)
xlabel('Time (s)')
ylabel('Angular Velocity (rad/s)')
title({'Joint 2 Angular Velocity';'(desired speed Vx = 2m/s, Vy = 0)'})

subplot(3,3,3)
xt = 1:20;
yt = phid(3,:);
xt1 = 1:0.1:20;
yt1 = spline(xt,yt,xt1);
plot(xt1,yt1)
```

```
xlabel('Time (s)')
ylabel('Angular Velocity (rad/s)')
title({'Joint 3 Angular Velocity';'(desired speed Vx = 2m/s, Vy = 0)'})

subplot(3,3,4)
xt = 1:20;
yt = phid(4,:);
xt1 = 1:0.1:20;
yt1 = spline(xt,yt,xt1);
plot(xt1,yt1)
xlabel('Time (s)')
ylabel('Angular Velocity (rad/s)')
title({'Joint 4 Angular Velocity';'(desired speed Vx = 2m/s, Vy = 0)'})

subplot(3,3,5)
xt = 1:20;
yt = phid(5,:);
xt1 = 1:0.1:20;
yt1 = spline(xt,yt,xt1);
plot(xt1,yt1)
xlabel('Time (s)')
ylabel('Angular Velocity (rad/s)')
title({'Joint 5 Angular Velocity';'(desired speed Vx = 2m/s, Vy = 0)'})

subplot(3,3,6)
xt = 1:20;
yt = phid(6,:);
xt1 = 1:0.1:20;
yt1 = spline(xt,yt,xt1);
plot(xt1,yt1)
xlabel('Time (s)')
ylabel('Angular Velocity (rad/s)')
title({'Joint 6 Angular Velocity';'(desired speed Vx = 2m/s, Vy = 0)'})

subplot(3,3,7)
xt = 1:20;
yt = phid(7,:);
xt1 = 1:0.1:20;
yt1 = spline(xt,yt,xt1);
plot(xt1,yt1)
xlabel('Time (s)')
ylabel('Angular Velocity (rad/s)')
title({'Joint 7 Angular Velocity';'(desired speed Vx = 2m/s, Vy = 0)'})
```