University of Nevada Las Vegas
Department of Mechanical Engineering
ME 320-1001
Dynamics of Machine

Project 1 Description:
Analysis of a 4 Bar Mechanism and Planar 3R Manipulator

Prepared for: Jameson Lee

Prepared by:
Dillard, Alec
Takagi, Hidero
Fallon, Brandon
Arambula, Thomas

Due Date: 4/23/2018

1 Abstract:

The group members were tasked to construct a code in order to analyze four-bar and three bar mechanisms in a convenient manner. To be able to do this, the calculations of the velocity and acceleration must be derived using both analytical and numerical methods. These equations are used for obtaining the position and angle of the links, angular velocity and angular acceleration of the mechanism.

The code written for this project is broken into 6 sections including the following*: Main_4bar, Main_3R, Grashofsanity, Link, Mechanisms, pathPlan*. The *Main_4bar* file acts as the central hub of information for the entirety of the code for the four-bar mechanism. This is where the input lengths are located, as well as the general settings of the simulation. The input lengths will then be sent to *Grashofsanity* to first determine if the mechanism is Grashof or Non-Grashof.

If the mechanism is Non-Grashof an error code will display the message, “This is a non-predictable behavior. This is not an assigned. Four Bar Mechanism.” If the Mechanism is classified as a Change-point, the same error message will be displayed because this type was not assigned. If the mechanism is classified as Grashof, the general settings and the input lengths will be sent to the *Link, Mechanisms, and pathPlan* classes. Once all the information is returned to *Main_4bar*, the simulation and data is displayed.

Link stores the data of the link positions, angular displacement, angular velocity, and angular acceleration. This class is called to process the start and end points of each link in order to

run the simulation. *Links* also uses the stored data to develop the graphs shown after the simulation is finished running. Finally, the positions of each link are converted to the global frame.

Mechanisms receives information based on what type of mechanism/system the input lengths generate. Depending on the type of system (Crank-Rocker, Crank-Crank or Rocker-Rocker) it initializes the positions of known values so the simulation does not start in abnormal locations. Using the "fsolve" function the system of equation determines the unknown angles, positions, velocities and accelerations in tandem with the kinematic formulas derived and the known values (which is also dependent on the type of system). For example, the Crank-Rocker or Crank-Crank four-bar mechanism, theta2 and theta4 are determined since theta1 and theta3 are the known values along with the length inputs.

PathPlan is the "super-class" that calls to both *Mechanisms* and *Link.* This super class is important because it essentially drives the code by communicating the known and unknown values between the two classes. The functions within *Mechanisms* are called within *pathPlan* to solve for the unknown angular displacements, velocities and accelerations. The positions of each bar are then updated to *Link.* Lastly, the endpoints of each of the bars are connected in consecutive succession, from r1 to r2, r3 to r4, and finally r2 to r4.

We wrote this project to plot position, angular velocity, angular acceleration of four-bar mechanisms and three-bar mechanism

2 Introduction:

2.1 Derivation of Position:

$$V_2 + V_3 = V_1 + V_4$$

General Position: $V_2 e^{i(\theta_2+\beta_2)} + V_3 e^{i(\theta_3+\beta_3)} = V_1 e^{i(\theta_1+\beta_1)} + V_4 e^{i(\theta_4+\beta_4)}$

Re: $V_2 cos(\theta_2 + \beta_2) + V_3 cos(\theta_3 + \beta_3) = V_1 cos(\theta_1 + \beta_1) + V_4 cos(\theta_4 + \beta_4)$

Im: $V_2 sin(\theta_2 + \beta_2) + V_3 sin(\theta_3 + \beta_3) = V_1 sin(\theta_1 + \beta_1) + V_4 sin(\theta_4 + \beta_4)$

Crank Rocker Configurations

Known variables: $r_1, r_2, r_3, r_4, \theta_1, \theta_3$

Unknown: θ_2, θ_4

Convert into system of equations

$$r_1 c_1 + r_2 c_2 = r_3 c_3 + r_4 c_4$$
$$r_1 s_1 + r_2 s_2 = r_3 s_3 + r_4 s_4$$

Isolate θ_2 *or* θ_4 for crank rocker/crank-crank configuration

$$r_4 c_4 = r_1 c_1 + r_2 c_2 - r_3 c_3$$
$$r_4 s_4 = r_1 s_1 + r_2 s_2 - r_3 s_3$$

Square Both sides

$$(r_4 c_4)^2 = (r_1 c_1 + r_2 c_2 - r_3 c_3)^2$$
$$(r_4 s_4)^2 = (r_1 s_1 + r_2 s_2 - r_3 s_3)^2$$

Add the functions together

$$(r_4c_4)^2+(r_4s_4)^2 = (r_1c_1 + r_2c_2-r_3c_3)^2 + (r_1s_1 + r_2s_2-r_3s_3)^2$$

Expand and Organize

$$r_4^2(c_4^2 + s_4^2) = r_1^2(c_1^2 + s_1^2) + r_2^2(c_2^2 + s_2^2) + r_3^2(c_3^2 + s_3^2) + 2r_1r_2c_1c_2 - 2r_2r_3c_2c_3 - 2r_1r_3c_1c_3 + 2r_1r_2s_1s_2 - 2r_2r_3s_2s_3 - 2r_1r_3s_1s_3$$

Use identity $(c^2 + s^2) = 1$ and move variables

$$r_4^2 - r_1^2 - r_2^2\text{-}r_3^2 + 2r_1r_3(c_1c_3 + s_1s_3) = 2r_2(r_1c_1 - r_3c_3)c_2 + 2r_2(r_1s_1 - r_3s_3)s_2$$

Use the identity $c_3c_1 + s_3s_1 = \cos(\theta_3-\theta_1)$

$$r_4^2 - r_1^2 - r_2^2-r_3^2 + 2r_1r_3\cos(\theta_3-\theta_1) = 2r_2(r_1c_1 - r_3c_3)c_2 + 2r_2(r_1s_1 - r_3s_3)s_2$$

$A_2 = 2r_2(r_1c_1 - r_3c_3)$

$B_2 = 2r_2(r_1s_1 - r_3s_3)$

$C_2 = A_2c_2+B_2s_2$

Use half tangent identities

$$C_2 = A_2\left(\frac{1 - \tan^2\left(\frac{\theta_2}{2}\right)}{1 + \tan^2\left(\frac{\theta_2}{2}\right)}\right)+B_2\left(\frac{2\tan\left(\frac{\theta_2}{2}\right)}{1 + \tan^2\left(\frac{\theta_2}{2}\right)}\right)$$

Multiply by $1 + \tan^2(\frac{\theta_2}{2})$

$$C_2\left(1 + \tan^2\left(\frac{\theta_2}{2}\right)\right) = A_2\left(1 - \tan^2\left(\frac{\theta_2}{2}\right)\right) + B_2\left(2\tan\left(\frac{\theta_2}{2}\right)\right)$$

Use the Quadratic Formula

$$(A_2 + C_2)\ \tan^2\left(\frac{\theta_2}{2}\right) - 2B_2 \tan\left(\frac{\theta_2}{2}\right) + (C_2 - A_2) = 0$$

$$\boldsymbol{\theta_2 = 2\tan^{-1}\left(\frac{B_2 \pm \sqrt{B_2^2 + A_2^2 - C_2^2}}{C_2 + A_2}\right)}$$

Repeat the same steps for θ_4

$$r_2c_2 = r_3c_3 + r_4c_4 - r_1c_1$$

$$r_2s_2 = r_3s_3 + r_4s_4 - r_1s_1$$

$A_4 = 2r_4(r_3c_3 - r_1c_1)$

$B_4 = 2r_4(r_3s_3 - r_1s_1)$

$C_4 = A_4c_4 + B_4s_4$

$$\boldsymbol{\theta_4 = 2\tan^{-1}\left(\frac{B_4 \pm \sqrt{B_4^2 + A_4^2 - C_4^2}}{C_4 + A_4}\right)}$$

Rocker-Rocker Configurations

Same steps can be performed as Crank-Rocker

$$r_1c_1 = r_3c_3 + r_4c_4 - r_2c_2$$

$$r_1s_1 = r_3s_3 + r_4s_4 - r_2s_2$$

$A_3 = 2r_3(r_4c_4 - r_2c_2)$

$B_3 = 2r_3(r_4s_4 - r_2s_2)$

$C_3 = A_3c_3 + B_3s_3$

$$\boldsymbol{\theta_3 = 2\tan^{-1}\left(\frac{B_3 \pm \sqrt{B_3^2 + A_3^2 - C_3^2}}{C_3 + A_3}\right)}$$

$$r_2c_2 = r_3c_3 + r_4c_4 - r_1c_1$$

$$r_2s_2 = r_3s_3 + r_4s_4 - r_1s_1$$

$A_1 = 2r_1(r_2c_2 - r_4c_4)$

$B_1 = 2r_1(r_2s_2 - r_4s_4)$

$C_1 = A_1c_1 + B_1s_1$

$$\boldsymbol{\theta_1 = 2\tan^{-1}\left(\frac{B_1 \pm \sqrt{B_1^2 + A_1^2 - C_1^2}}{C_1 + A_1}\right)}$$

Using the found angles, any position can be calculated.

1.1.1 3R Manipulator:

Known: θ_3, r_1, r_2, r_3

Unknown: θ_1, θ_2

$$L_1 + L_2 + L_3 = 0$$

General Position: $L_1e^{i\theta_1} + L_2e^{i\theta_2} + L_3e^{i\theta_3} = 0$

$$\textbf{Re}: L_1 cos(\theta_1) + L_2 cos(\theta_2) + L_3 cos(\theta_3) = 0$$

$$\textbf{Im}: L_1 sin(\theta_1) + L_2 sin(\theta_2) + L_3 sin(\theta_3) = 0$$

Solving for θ_2

$$r_1 c_1 = -r_2 c_2 - r_3 c_3$$

$$r_1 s_1 = -r_2 s_2 - r_3 s_3$$

Square both Equations

$$(r_1 c_1)^2 = (-r_2 c_2 - r_3 c_3)^2$$

$$(r_1 s_1)^2 = (-r_2 s_2 - r_2 s_2)^2$$

Add the equations together. Use identity $(c^2 + s^2) = 1$

$$r_1^2 = (-r_2 s_2 - r_2 s_2)^2 + (-r_2 c_2 - r_3 c_3)^2$$

Expand the equation

$$r_1^2 = r_2^2(c_2^2 + s_2^2) + 2r_2 r_3(c_2 c_3 + s_2 s_3) + r_3^2(c_3^2 + s_3^2)$$

Use Identity $c_2 c_3 + s_2 s_3 = \cos(\theta_2 - \theta_3)$

$$r_1^2 = r_2^2 + 2r_2 r_3 \cos(\theta_2 - \theta_3) + r_3^2$$

Organize

$$r_1^2 - r_2^2 - r_3^2 = 2r_2 r_3 \cos(\theta_2 - \theta_3)$$

Solve for θ_2

$$\theta_2 = \theta_3 + \cos^{-1}\left(\frac{r_1^2 - r_2^2 - r_3^2}{2r_2 r_3}\right)$$

Repeat same procedure for θ_1

$$\theta_1 = \theta_2 + \cos^{-1}\left(\frac{r_3^2 - r_2^2 - r_1^2}{2r_2 r_1}\right)$$

1.2 Derivation of Velocity:

The derivation of velocity allows users to calculate the velocity of any link on the 4-bar mechanism or utilize the equation to find other values such as angles and acceleration.

1.2.1 Four Bar Mechanism:

General Velocity: $V_2\dot{\beta}_2 e^{i(\theta_2+\beta_2)} + V_3\dot{\beta}_3 e^{i(\theta_3+\beta_3)} = V_4\dot{\beta}_4 e^{i(\theta_4+\beta_4)}$

Real: $V_2\dot{\beta}_2 \cos(\theta_2 + \beta_2) + V_3\dot{\beta}_3 \cos(\theta_3 + \beta_3) = V_4\dot{\beta}_4 \cos(\theta_4 + \beta_4)$

Imaginary: $V_2\dot{\beta}_2 \sin(\theta_2 + \beta_2) + V_3\dot{\beta}_3 \sin(\theta_3 + \beta_3) = V_4\dot{\beta}_4 \sin(\theta_4 + \beta_4)$

1.3 Derivation of Acceleration:

The derivation of acceleration allows users to calculate the acceleration of any link on the 4-bar mechanism or utilize the equation to find other values such as angles and velocities.

1.3.1 Four Bar Mechanism

General:

$$V_2\ddot{\beta}_2 e^{i(\theta_2+\beta_2)} + V_2\dot{\beta}^2{}_2 e^{i(\theta_2+\beta_2)} + V_3\ddot{\beta}_3 e^{i(\theta_3+\beta_3)} + V_3\dot{\beta}^2{}_3 e^{i(\theta_3+\beta_3)} = V_4\ddot{\beta}_4 e^{i(\theta_4+\beta_4)} + V_4\dot{\beta}^2{}_4 e^{i(\theta_4+\beta_4)}$$

Real:

$$V_2\ddot{\beta}_2 \cos(\theta_2 + \beta_2) + V_2\dot{\beta}^2{}_2 \cos(\theta_2 + \beta_2) + V_3\ddot{\beta}_3 \cos(\theta_3 + \beta_3) + V_3\dot{\beta}^2{}_3 \cos(\theta_3 + \beta_3)$$
$$= V_4\ddot{\beta}_4 \cos(\theta_4 + \beta_4) + V_4\dot{\beta}^2{}_4 \cos(\theta_4 + \beta_4)$$

Imaginary:

$$V_2\ddot{\beta}_2 \sin(\theta_2 + \beta_2) + V_2\dot{\beta}^2{}_2 \sin(\theta_2 + \beta_2) + V_3\ddot{\beta}_3 \sin(\theta_3 + \beta_3) + V_3\dot{\beta}^2{}_3 \sin(\theta_3 + \beta_3)$$
$$= V_4\ddot{\beta}_4 \sin(\theta_4 + \beta_4) + V_4\dot{\beta}^2{}_4 \sin(\theta_4 + \beta_4)$$

3 Results:

The following information outlines how the code pieces together multiple

3.1 Four Bar Mechanism

There are four supporting files to the main 4bar Mechanism. They consist of Grashofsanity, Link, Mechanisms, and pathPlan. The code starts with defining the link. The link class will define all the properties of the links which include: instantaneous positions, start and end vectors, orientation and velocity and acceleration data. This is derived from the inputs of the user. Once generated, it is converted to the global frame and called into the main 4 bar code.

Following the link is the grashofsanity. This takes the inputs of the user and finds the type of mechanism by systematically defining the shortest and longest variables for the input. This is done by using functions such as max, min, and indexing to sort the lengths. Once it has defined the S and L variables, the code will apply them them to the general grashofsanity constraints to choose the type of mechanism that the 4bar is. This information is sent back to the main code to be used in defining the remaining components.

Once the link parameters have been set up and the type of mechanism is determined by grashof, the mechanisms class file is called on. This class performs all the necessary calculations

needed to determine the position, velocity, acceleration, and theta's. These values are calculated for every iteration and sent back to the main 4bar.

Next, pathPlan is called to piece the link and mechanism class together. This class file updates the link and mechanism for every iteration, meaning that it will update the mechanism type for the iteration, solve for all the values, and then proceed to update the link class. Once the values have been determined and the link has been updated, pathPlan then pieces together each link into a singular 4bar mechanism by determining the starting and end positions of the links.

Following this, the main 4bar file will begin applying iterations, updating the link file for each one and storing the data. Once this has been completed, the motion plot will be generated from this data. Subsequently, the angular displacements, velocities, and accelerations are plotted from the stored data.

3.1.1 Decision Tree for Grashof Mechanism

The GrashofSanity.m code is an essential part of this code, where the type of four bar mechanism is determined and if the type of mechanism is within the scope of this project. First the shortest length and the longest length are initialized to their own variable and which arm they represent is saved, for example a four bar mechanism with arms [10 3 4 6] the shortest variable S is determined as 3 representing arm r2. Next, the value returning variables are initialized to their default values. After the variables are initialized the decision tree is started. The three types of grashof four bar mechanism that is simulated in this project are the crank-rocker, Rocker-Rocker, and Crank-Crank mechanisms. These three types of mechanisms are saved in the code as a string in variable type and with value 2, 3, and 4, respectively, for their typeCode variable. If the typeCode variable is equal to 1, that means that the four bar mechanism is a non-grashof four bar

mechanism. And if typeCode variable is equal to 0 then the mechanism is a change point mechanism and that is not in the scope of this project. The type of four bar mechanism will be displayed once the decision tree has been finished so that an error message may be displayed with the type of four bar mechanism and the error on why this mechanism will not run. The variables, typeCode, type, and warning, are returned to the main code.

3.1.2 Adaptability in Code

. The scope of this project is to create a code that simulates a few Grashof four-bar mechanisms, the crank-crank, crank-rocker, and rocker-rocker type mechanisms. Additionally, the code should display an error message when a change-point or non-Grashof four-bar mechanism is inputted. This is determined inside of the GrashofSanity.m function file, where the value-returning variables are 'typeCode,' 'type,' and 'warning.' So that, if the mechanism is a crank- Rocker mechanism the typeCode will return the value 2, and type as 'Crank-Rocker'. The types of four bar mechanisms that is within this scope are the Crank-Rocker, Crank-Crank, and Rocker-Rocker type four bar mechanisms. If the type of four-bar mechanism is not within the scope of the project the error message will be displayed from GrashofSanity, and again in the main_4bar file as well. For example, if the mechanism is non-Grashof the typeCode returns as 1, and warning is "Non-Grashof' and a warning is recorded as 'Non-predictable behavior. This type is not assigned." The typeCode variable is sent through the code so that the right algorithms can be used for calculating the position, velocity, and acceleration. Multiple other changes can be done to the main code where the simulation setting can change. The first one is on line 36, the ground slope is determined on this line, as shown in figure 1 below.

```
36 -  th3 = pi/6;
```

```
36 -  th3 = pi/12;
```

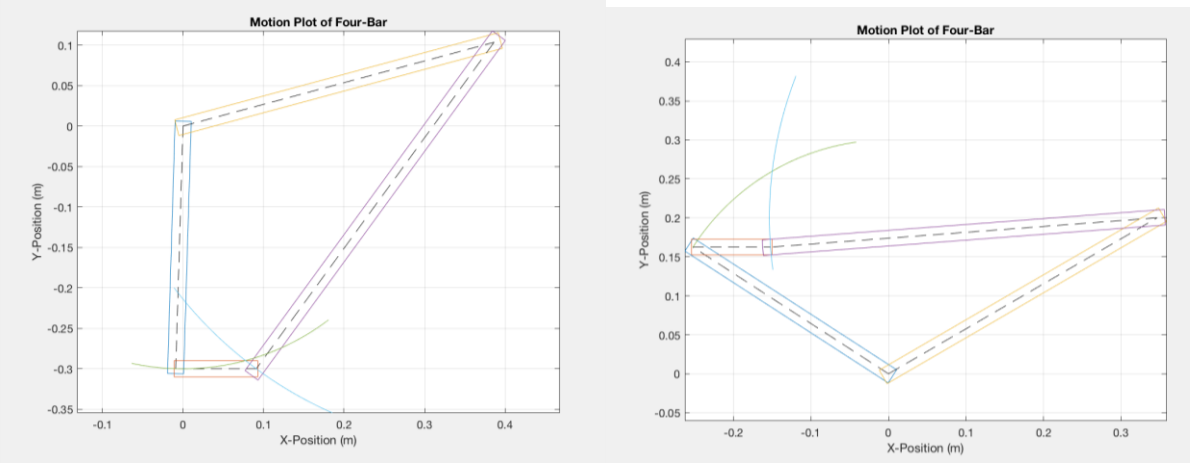


Figure 1: initial θ_3 adaptability

In figure 1 above, the initial angle for the ground arm, θ_3, is changed from pi/12 to pi/6. This can be seen in the figure as the yellow bar is at 15 degrees on the global frame on the plot on the left and the right figure shows the bar at 30 degrees. The motions of the arms for the rest of the mechanism stay the same relative to r3. The change affects the calculations in mechanisms so that the new plot is not only rotated but the arms are flipped to have a better angle due to the method of reaching the unknown angles will make it so that it is switched.

```
48 -  Wid = .02*ones(1,4);
49 -  box_x = [-.04 1.04 1.04 -.04 -.04];
50 -  box_y = [1 1 -1 -1 1];
```

```
48 -  Wid = .04*ones(1,4);
49 -  box_x = [-.04 1.04 1.04 -.04 -.04];
50 -  box_y = [0.2 0.2 -0.2 -0.2 0.2];
```

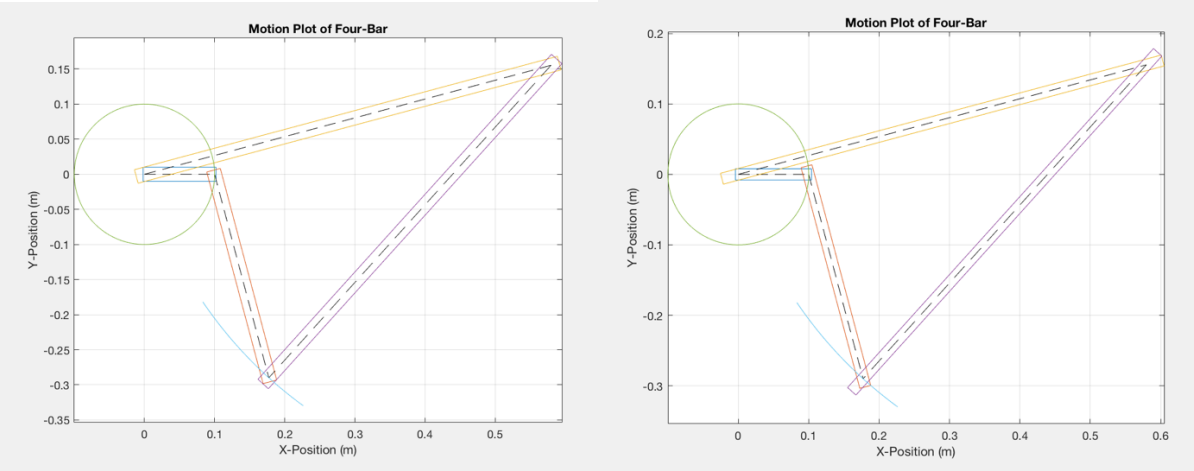


Figure 2: Box size change.

The motion plot for the four-bar mechanism is plotted with rectangles overlaying on top of a line to make it easier to understand the motion. These setting are used to change width and end lengths of the rectangles. In figure 2 above, the left plot is the original plot with unchanged widths and lengths, looking at the plot to the right, it is evident that the arms are skinnier and the ends of the arms go much farther. These changes stay within the main_4bar code and only effects the simulation.

```
54 -    dt = 0.01;
55 -    N = 100;
56 -    t = linspace(0,2*pi,N);
57 -    time = linspace(0,dt*N,N);
```

```
54 -    dt = 0.03;
55 -    N = 20;
56 -    t = linspace(0,2*pi,N);
57 -    time = linspace(0,dt*N,N);
```

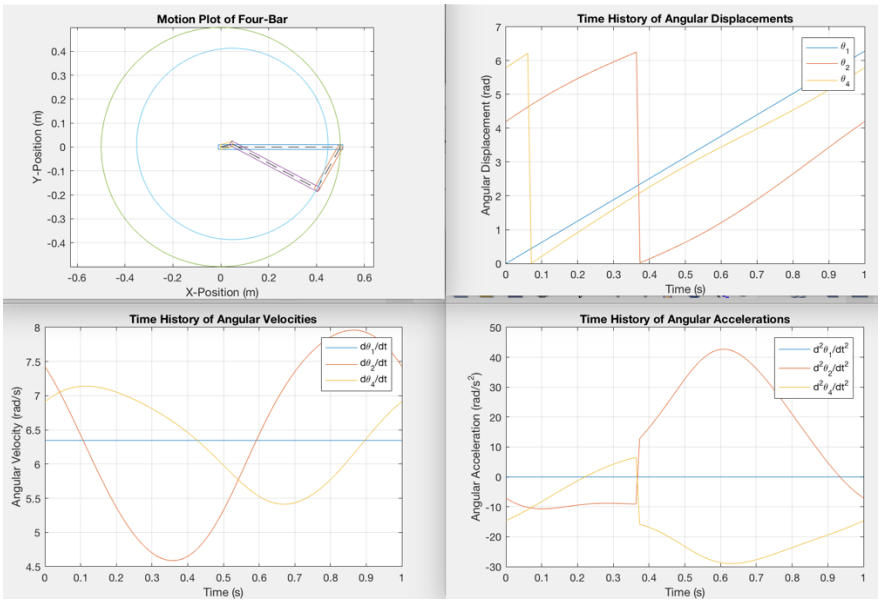


figure 3: Original plots with 100 iterations

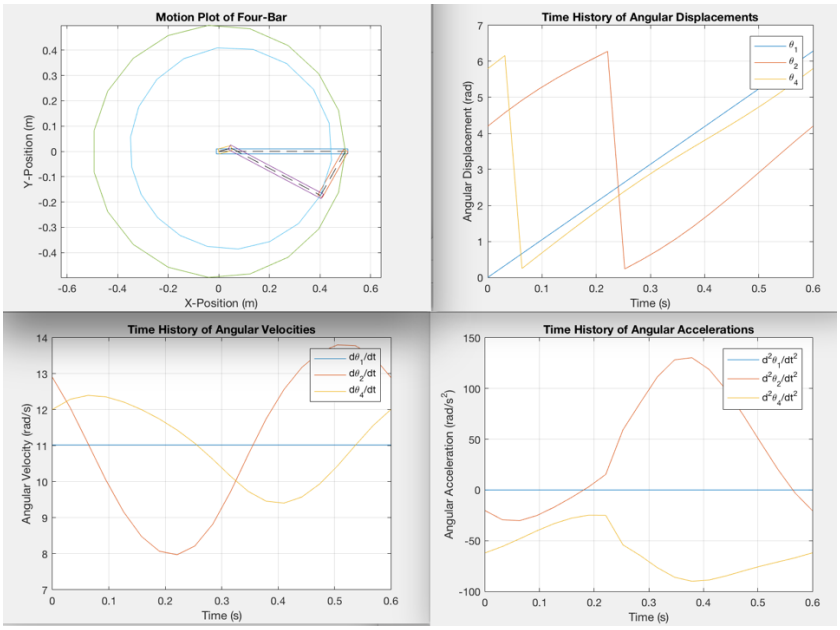


Figure 4: Updated plots with 20 iterations

On line 54, the time increment between each iteration is instantiated so that if a larger value is placed the speed of the simulation is slowed, and vice versa if the value is less. The next line in the code defines the amount of increments that will be calculated in one revolution of the driver arm. The more increments will result in more accurate data so that the smoothness of the lines are clearly seen in plots in figures 3 and 4 showing position, velocity, and acceleration. The graphs in figure 4 are obviously less smooth as the lines that create the curve are clearly seen. Changing the time increment will only change the time so that velocity and acceleration will be skewed accordingly to the time. However, changing the amount of increments will change both the accuracy of the results and the skew the calculations that are time based.

```
59 -    closure = 1;
60 -    %closure = 2;
```

```
59      %closure = 1;
60 -    closure = 2;
```

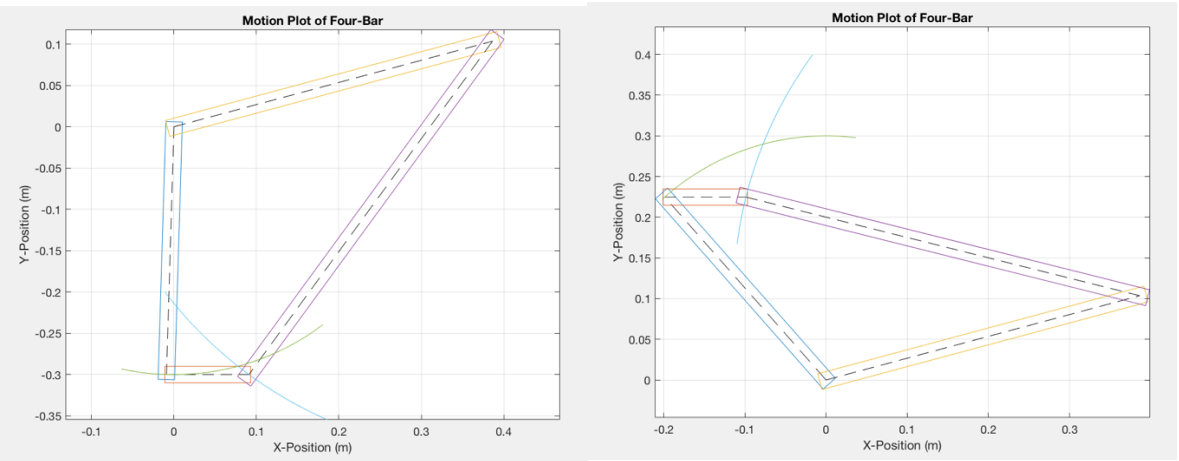


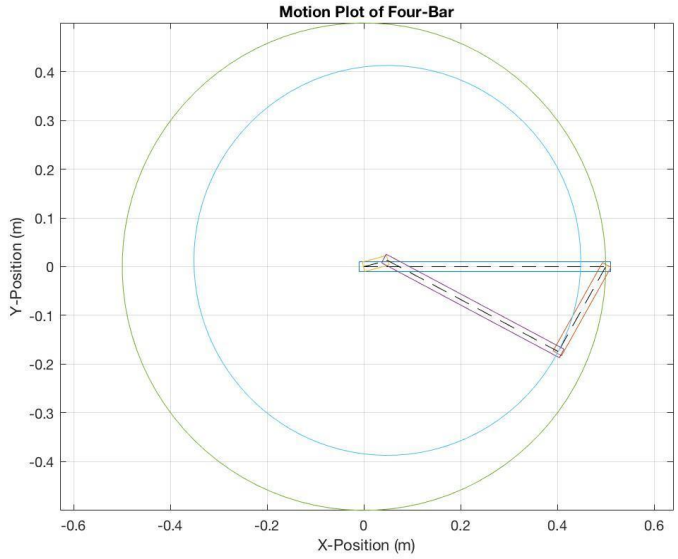
Figure 5: Closure 1 (left) Closure 2 (right)

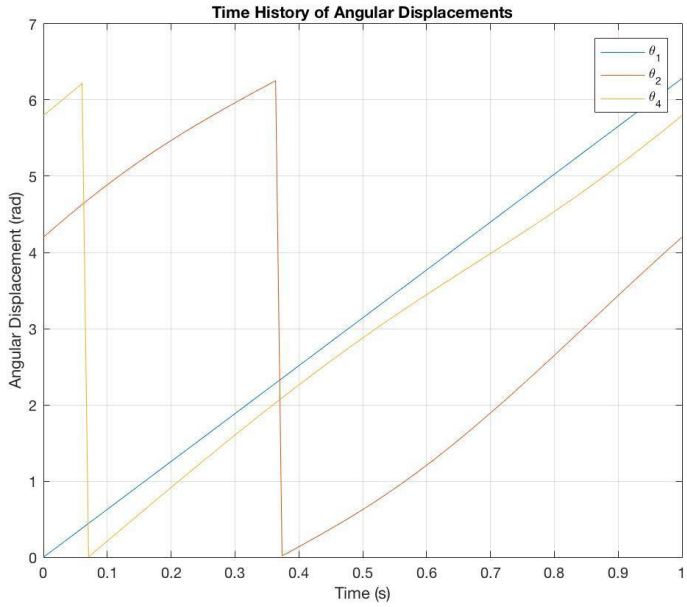
In line 58 and 60, each four-bar mechanism has two different solutions where it can be positive or negative so that if a particular result is needed it can be found by changing the closure variable. The closure directly changes the calculations done in mechanisms as if the closure is 2

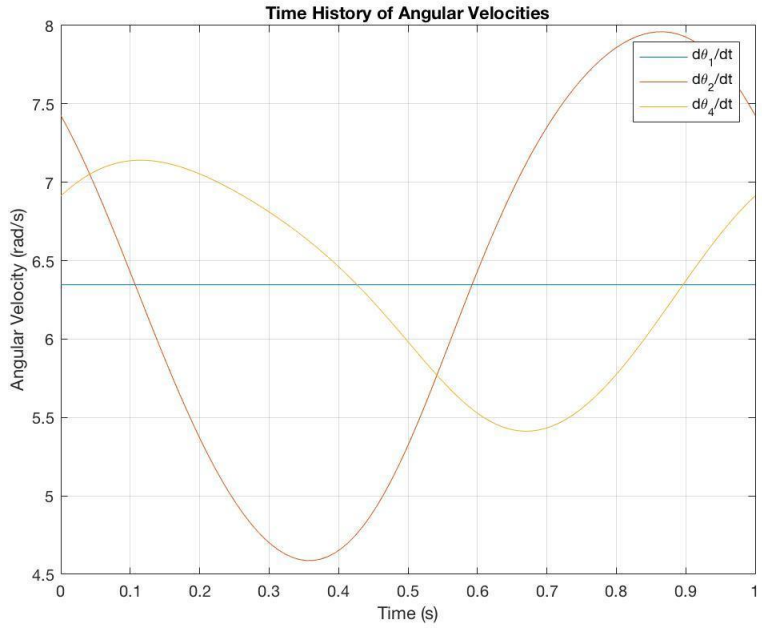
then the initial angles are shifted by adding pi to the initial guess of where to find the particular variable in the system of equations.

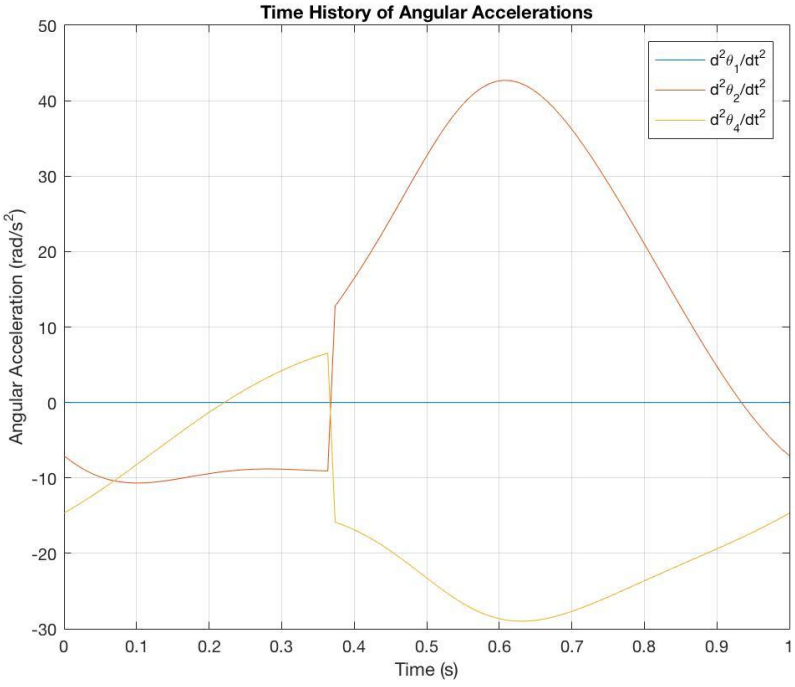
3.1.3 Figures for all Plottable Configurations

Crank-Crank

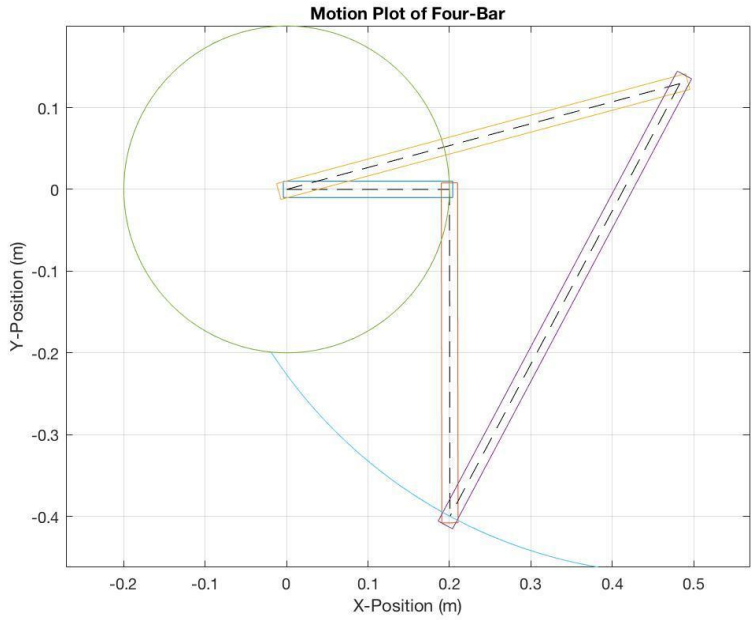


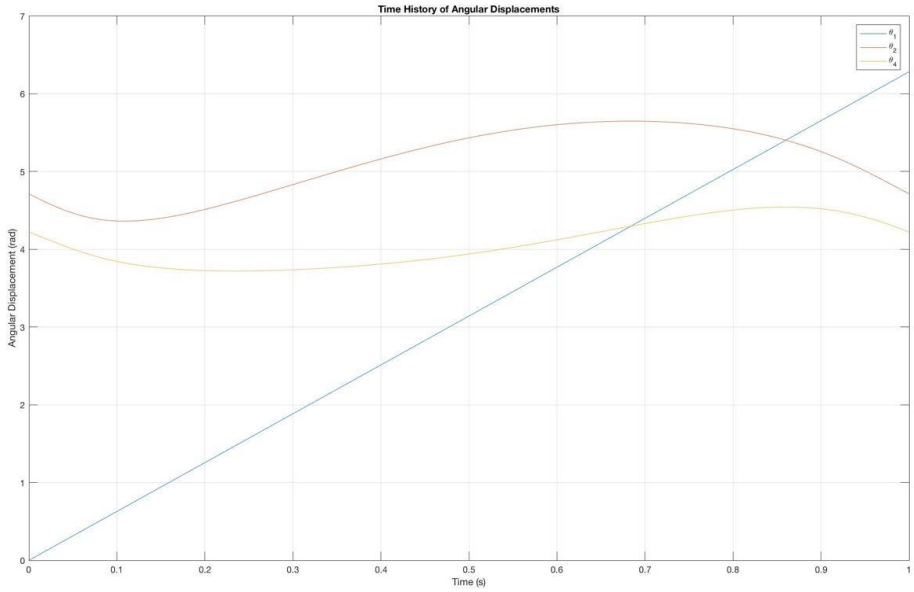


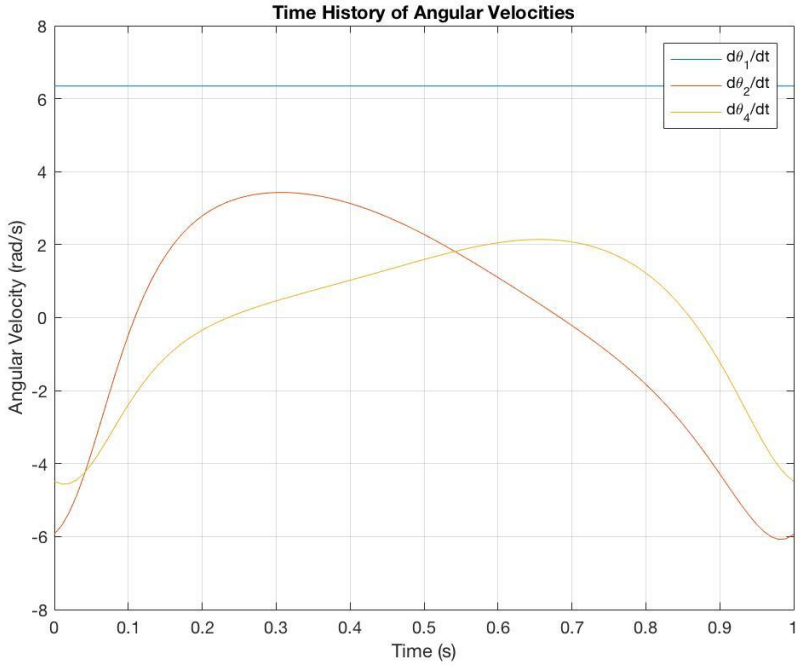

Time History of Angular Velocities
Angular Velocity (rad/s)
Time (s)
dθ1/dt
dθ2/dt
dθ4/dt
8
7.5
7
6.5
6
5.5
5
4.5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

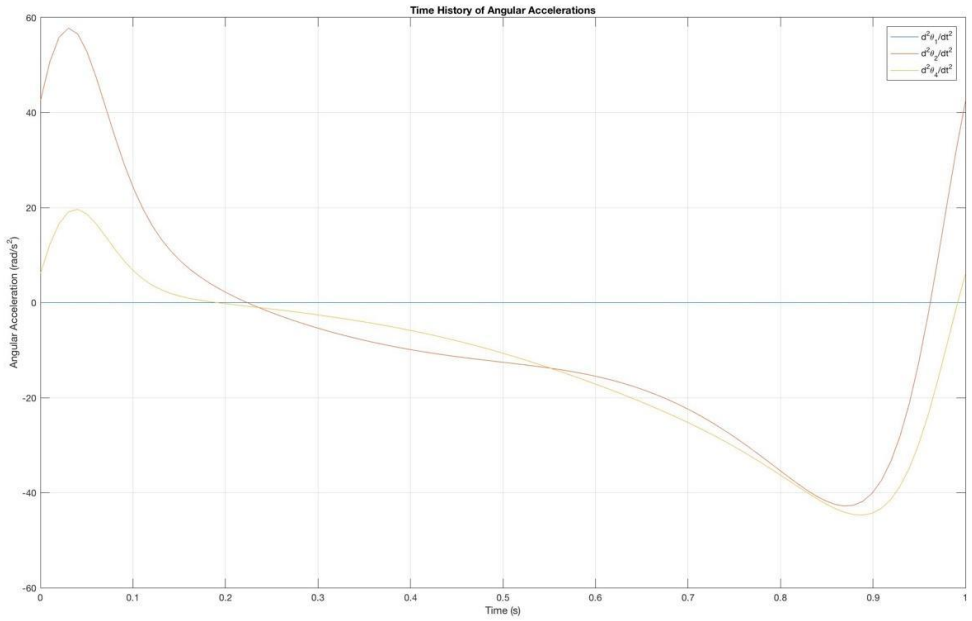

Time History of Angular Accelerations
Angular Acceleration (rad/s2)
Time (s)
d2θ1/dt2
d2θ2/dt2
d2θ4/dt2
50
40
30
20
10
0
-10
-20
-30
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Crank-Rocker

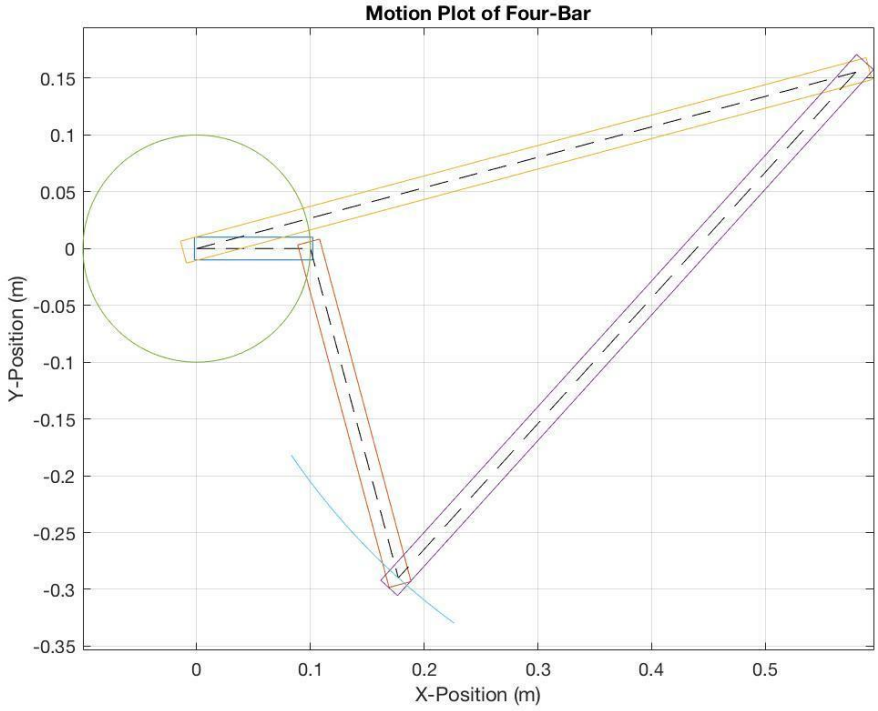


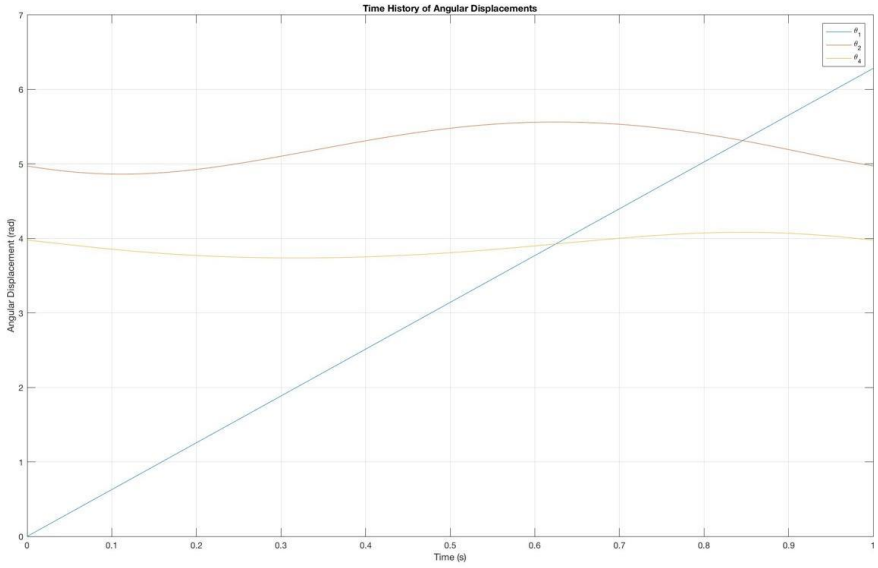


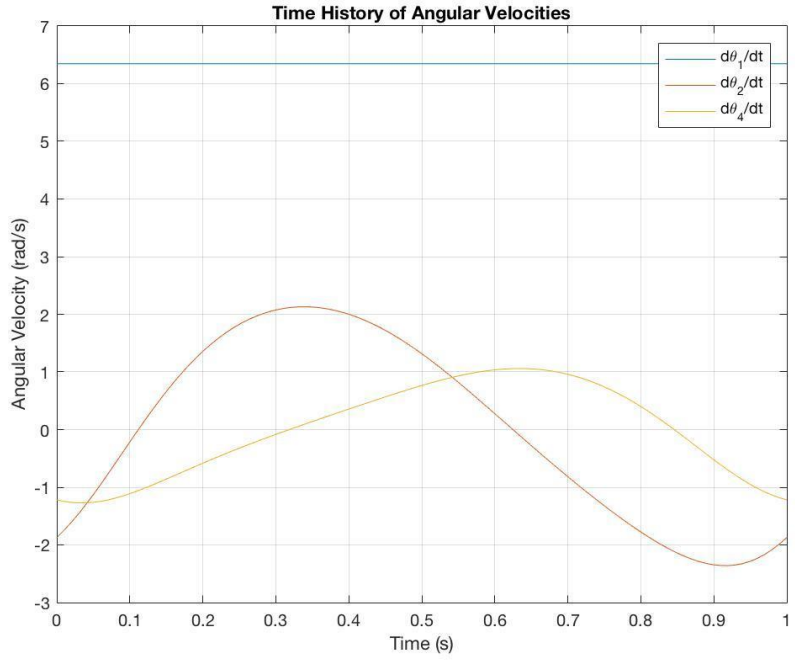

Time History of Angular Velocities
dθ1/dt
dθ2/dt
dθ4/dt
Angular Velocity (rad/s)
Time (s)
8
6
4
2
0
-2
-4
-6
-8
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

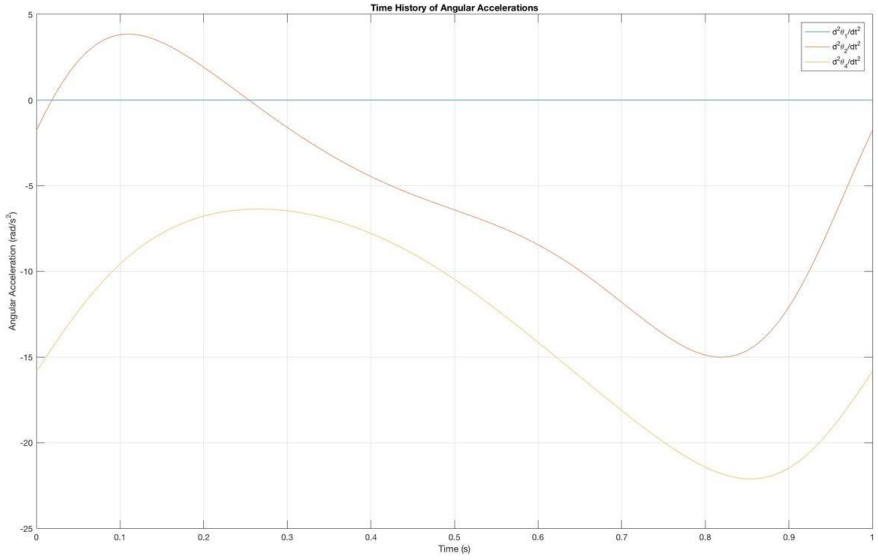

Time History of Angular Accelerations
d²θ1/dt²
d²θ2/dt²
d²θ4/dt²
Angular Acceleration (rad/s²)
Time (s)
60
40
20
0
-20
-40
-60
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Crank Rocker

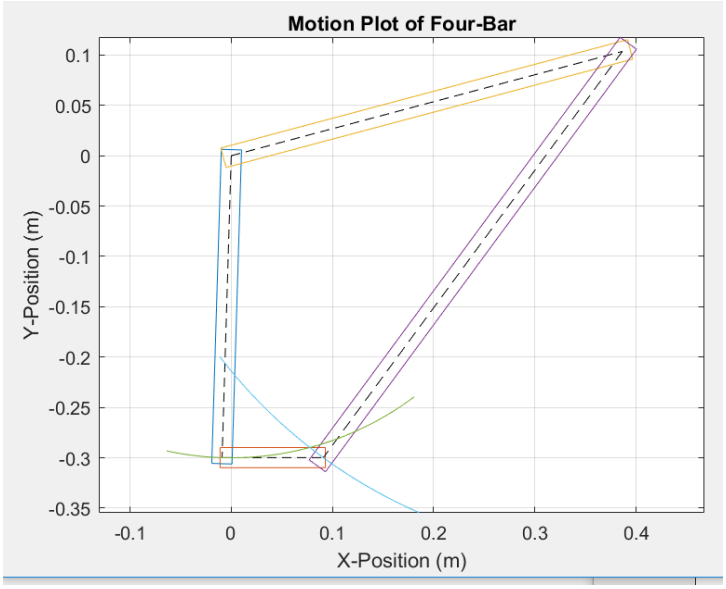


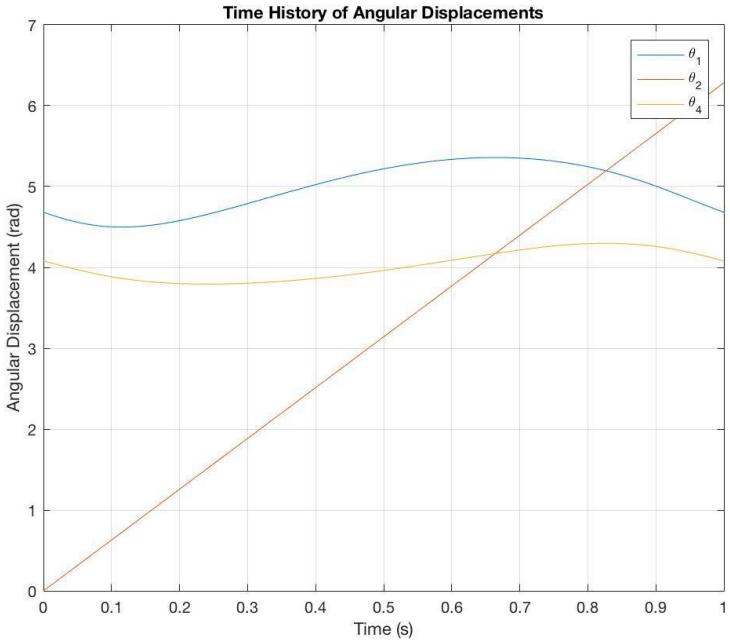


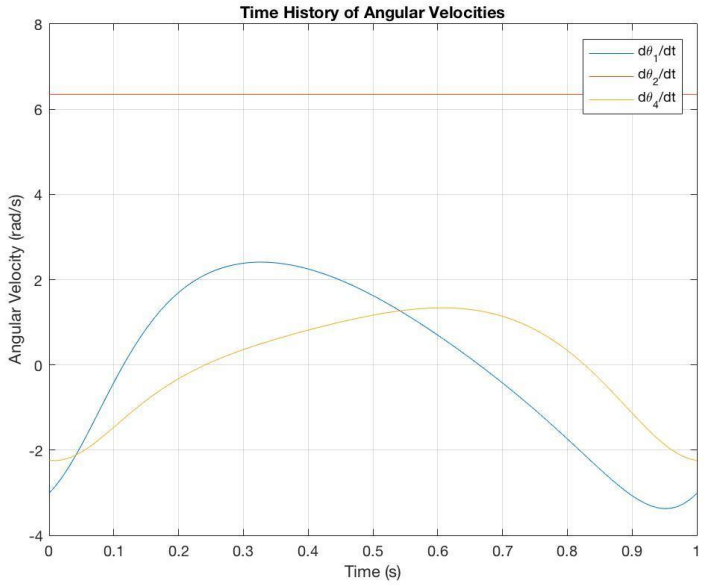

Time History of Angular Velocities
Angular Velocity (rad/s)
Time (s)
dθ1/dt
dθ2/dt
dθ4/dt

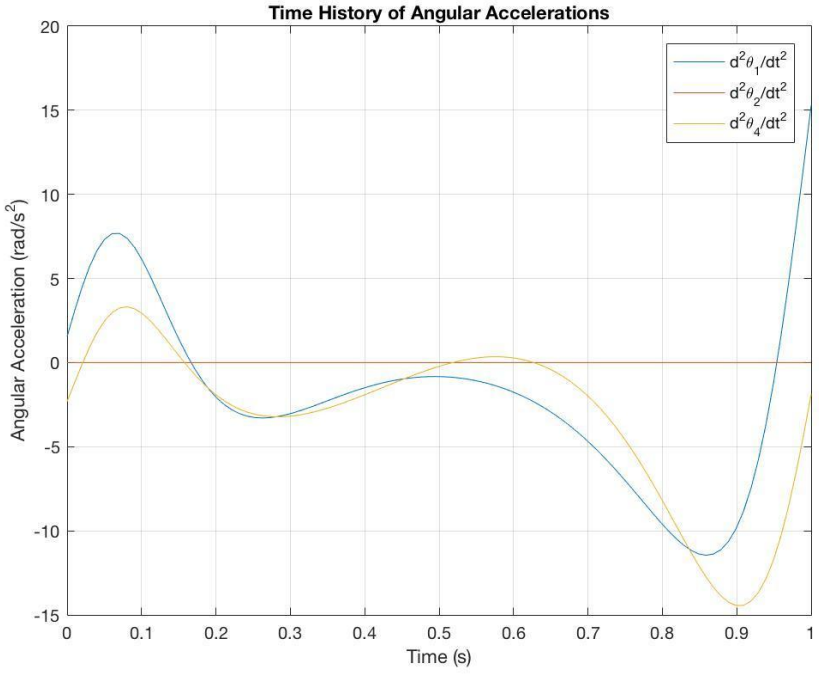

Time History of Angular Accelerations
Angular Acceleration (rad/s²)
Time (s)

Rocker-Rocker






Time History of Angular Velocities
Angular Velocity (rad/s)
Time (s)
dθ1/dt
dθ2/dt
dθ4/dt
8
6
4
2
0
-2
-4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Time History of Angular Accelerations
Angular Acceleration (rad/s2)
Time (s)
d2θ1/dt2
d2θ2/dt2
d2θ4/dt2
20
15
10
5
0
-5
-10
-15
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

3.2 3R Manipulator

The class files all work in the same way as the 4bar, but the main 3R runs the code in a different manner. Because there are only 3 links, no grashofsanity is applied here. The main 3R file takes 3 input lengths and allows user to define the box sizes for the vectors. Next, input values for the simulation are laid out and can be easily modified. The link class is then called in the same manner as the 4bar to create all 3 links. Now the code starts to vary. Because grashof can not be applied, the main 3R file calls on the crank-slider configuration data from mechanisms since the 3R is a crank-slider. Following this, pathPlan is applied in the same manner as the main 4bar, updating the link position, velocity, and acceleration data based on the calculations performed in the mechanisms file. Lastly, the motion graph is plotted by pulling the stored link data.

3.2.1 Adaptability in Code

Many of the same aspects of the code can be changed in the 3R as well as the four-bar mechanism, such as the simulation setting where the overlapping rectangle size can be adjusted, the amount of iterations and the time between them are adjustable, and how the closure value can be used to flip the motion of the plot. A unique change that can be done on the 3R mechanism is changing the resulting motion plot, from lines 28 to 32.

```
28 -  r = 0.5;
29 -  x0 = 0.5;
30 -  y0 = 0.25;
31 -  XR = r*cos(t).*sin(2*t) + x0;
32 -  YR = r*sin(t).*cos(4*t) + y0;
```

```
28 -  r = 0.5;
29 -  x0 = 0.5;
30 -  y0 = 0.25;
31 -  XR = r*cos(t).*sin(4*t) + x0;
32 -  YR = r*sin(t).*cos(4*t) + y0;
```

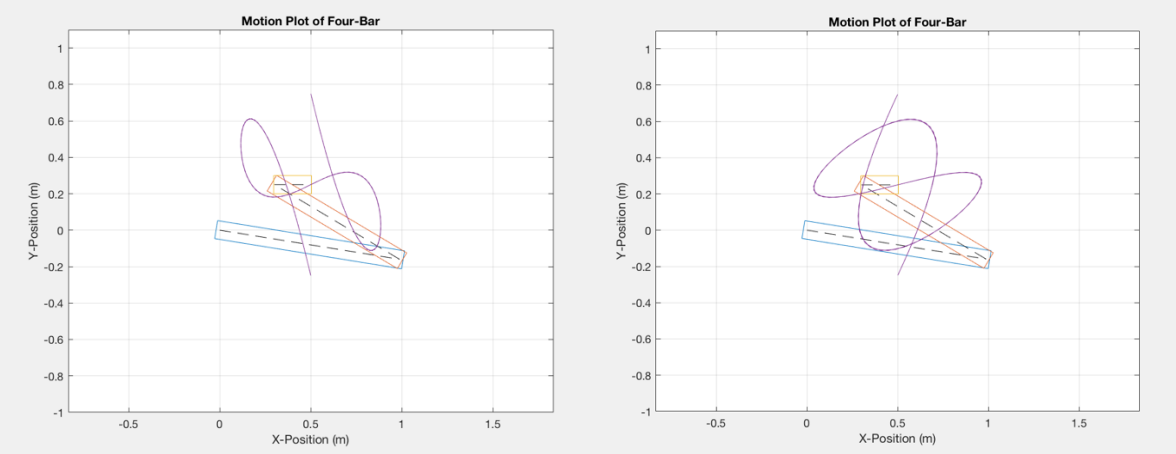
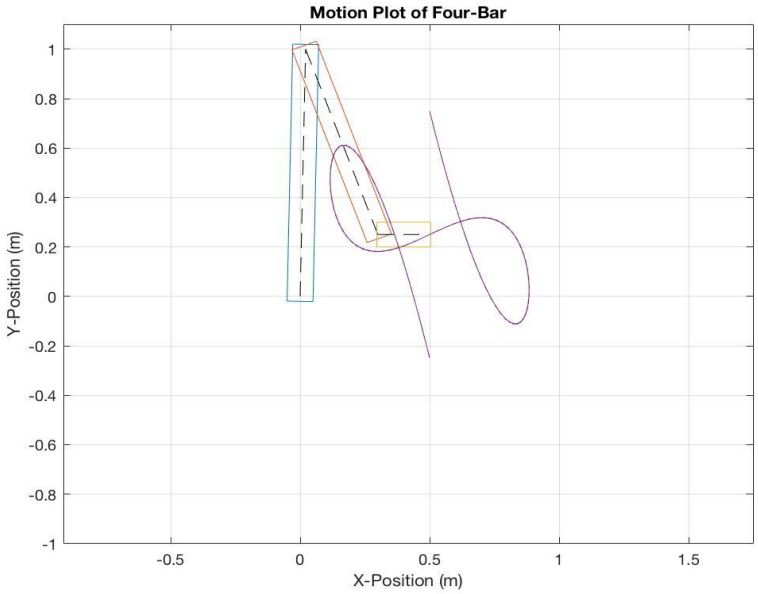


Figure 6: Changing the resulting motion plot

As seen above in figure 6, the resulting motion plot can be manipulated so that the 3R mechanism will follow the path. This can be accomplished by changing the two equations on lines 31 and 32 so that most plots can be followed. The lengths of the arms must be greater than the displacement from the origin for the plot parameters to be met. If this parameter is not followed, the joints will detach when the displacement is too large.

3.2.2 Figures for all Plottable Configurations



4 Conclusion

The goal of this project was to create both a 4bar and 3R simulation to apply knowledge learned in class to an applicable scenario. The main files worked by calling on classes which solved for specific parameters. These classes consisted of Grashofsanity, link, mechanism, and pathPlan. The grashof function file basically determined the type of 4bar mechanism that was input into a matrix. The function determines the shortest and longest links, then runs the values through a flowchart to find which scenario they fit in. These scenarios consist of crank-crank, crank-rocker, and rocker-rocker for the applicable ones to this project. The mechanism class uses given input lengths to mathematically solve for the position, velocity, acceleration, and thetas. The link class stores the data of the positions, angular velocity, angular acceleration, and angular displacement. Furthermore, this class tracks the movement of the link through stored data and is called on in the main file to plot it. PathPlan is a super-class which calls on both the link and mechanism file, allowing them to work in conjunction to both generate and store data to the various iterations. The main 4bar compiles everything, runs the iterations, and plots all the data. The main 3R files works in a similar manner, except there are only 3 links and grashof is not needed. Instead, the mechanisms class solves for just one specified class, being the crank-slider because a 3Bar mechanism is a crank-slider. The rest works in the same manner as the 4Bar. Ultimately, the goal of the project was completed. Both the 4Bar and 3R files ran as they are supposed to and output the correct motion, position, velocity, and acceleration plots. It can be seen in all the plots that one angle stays constant for velocity and acceleration and the position increases with a constant slope, which is expected behavior for the main driving link. Furthermore, the motion plot shows all the links intact and not separating at any point in time, a good indicator that the system was coded correctly. Additionally, each link rotate within the tolerable region, not overextending or staying

stagnant throughout the movement. Concluding, all links for each scenario move as expected and the project goal was met.