2020

5-Bar Linkage Kinematic Solver and Simulator

Brandon J. Gamble
University of Vermont

5-Bar Linkage Kinematic Solver and Simulator

Brandon Gamble
Department of Mechanical Engineering
College of Engineering and Mathematical Sciences
Honors College
The University of Vermont

April 10, 2020

Abstract

The 5-Bar Linkage (5BL) is a planar robot design that is mechanically far simpler than standard x-y gantry style planar robots. While simpler and cheaper to construct, the kinematics of the 5BL are non-linear and more complex than the gantry style. Making open-source path control software available for 5BL style robots, would greatly increase accessibility to planar robots such as laser cutters, 3D printers, and CNC mills. Singularities, working modes, and error sensitivity are discussed and analyzed. Finally, a simple transform to approximate the forward and inverse kinematics is proposed.

Nomenclature

Latin

A_i = shifting factor for proposed transform
a_i = length of driving link fixed to motor i
B = scaling factor for proposed transform
b_i = length of floating link between end-effector and distal end of driving link i
c_i = distance from motor i to end-effector
M_i = location of motor i
n = number of steps per revolution of stepper motor
R_i = location of distal end of driving link fixed to motor i
P = location of end-effector
S = absolute step position of stepper motor
v = placeholder for readability of forward kinematic equations
w = distance between driving motors
(X_Q, Y_Q) = Euleclidian coordinates of arbitrary point Q

Greek

$\alpha_i = \angle M_j M_i P$ (intermediate angle)
$\beta_i = \angle P M_i R_i$ (intermediate angle)
γ = angle of rotation (xy to $\theta_1\theta_2$)
$\theta_i = \angle M_j M_i R_i$ (motor angle - combination of α and β)
$\mu_i = \angle M_i R_i P$ (transmisssion angle)

Subscripts

$i \in (1, 2)$, $i \neq j$
$j \in (1, 2)$, $j \neq i$

Contents

1 Background

1.1 Maker Movement

The Maker Movement finds its foundations at the intersection between DIY culture and hacker culture. Makers are fueled by creativity and curiosity, inventing and tinkering, with an emphasis on rapid prototyping. Woodwork, metalwork, electronics, robotics, laser cutting, CNC milling, and, most familiar of all, 3D printing, are characteristic of the Maker's skill-set and laboratory.

In the past decade, the ideology of the Maker Movement has found increasing support in communities and schools. Communities are opening Makerspaces - collaborative work spaces where Makers can have access to tools and technologies that are not feasible for the hobbyist to have at home such as full wood shops, metal shops, and machine shops, electronics laboratories, laser cutters, 3D printers, etc. Schools are infusing Maker ideas into the classroom by integrating technology, design thinking, and prototyping into their curricula, focusing on experiential education[5,8] .

While Makerspaces have dramatically increased access to a broad spectrum of technology, not all communities have Makerspaces, and even in those communities that do, there are still many makers who only want access to a select few tools, namely 3D printers, laser cutters, and CNC milling machines, which are all planar robots with exceptional application to rapid prototyping.

1.2 Standard Planar Robot Design: Gantry

Current planar robots are nearly all designed around a 2 degree-of-freedom (DOF) x-y gantry system utilizing belts & pulleys (Fig. 1), rack & pinion gears, or lead screw drives (Fig. 2) to translate the end-effector in two orthogonal linear dimensions. These designs are

not only expensive to purchase and repair, but also both difficult and tedious to assemble, calibrate, and maintain.

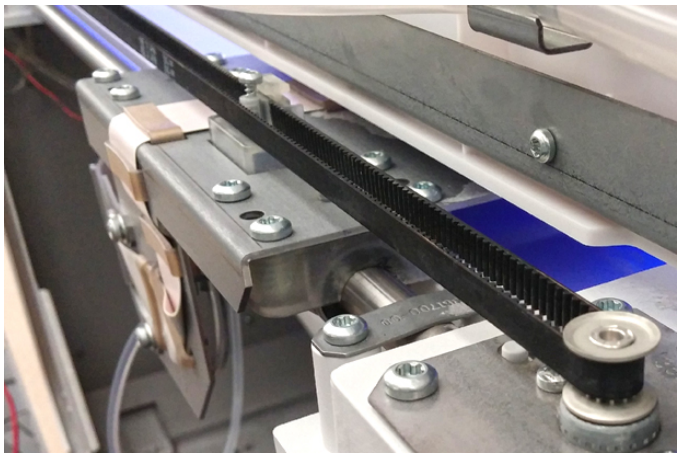

Figure 1: Belt and pulley configuration of x-y gantry[9]

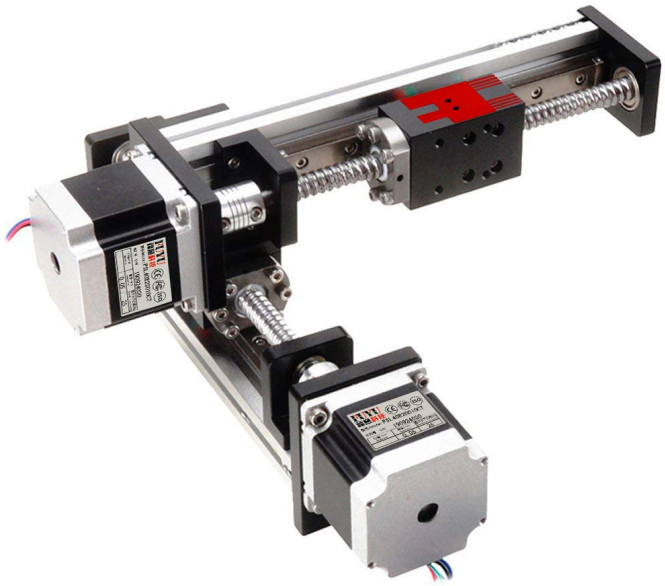

Figure 2: Lead screw configuration of x-y gantry[3]

While it is intuitive to control a planar robot by pairing each motor with a unique linearly independent vector, i.e., one motor controlling motion in x-direction and one motor controlling motion in the y-direction, the designs that implement this configuration, are, as noted, prone to being expensive and mechanically complex.

1.3 Alternative Planar Robot Design: 5-Bar Linkage (5BL)

An alternative design for planar robots is the 5BL (Fig. 3). This design is far simpler than aforementioned planar robot designs. It has no belts, pulleys, nor gears. Rather, it is comprised of just four bars connected in a chain via pin connections and fixed to one of two motors at each end. This design is extremely simple both to manufacture and assemble.

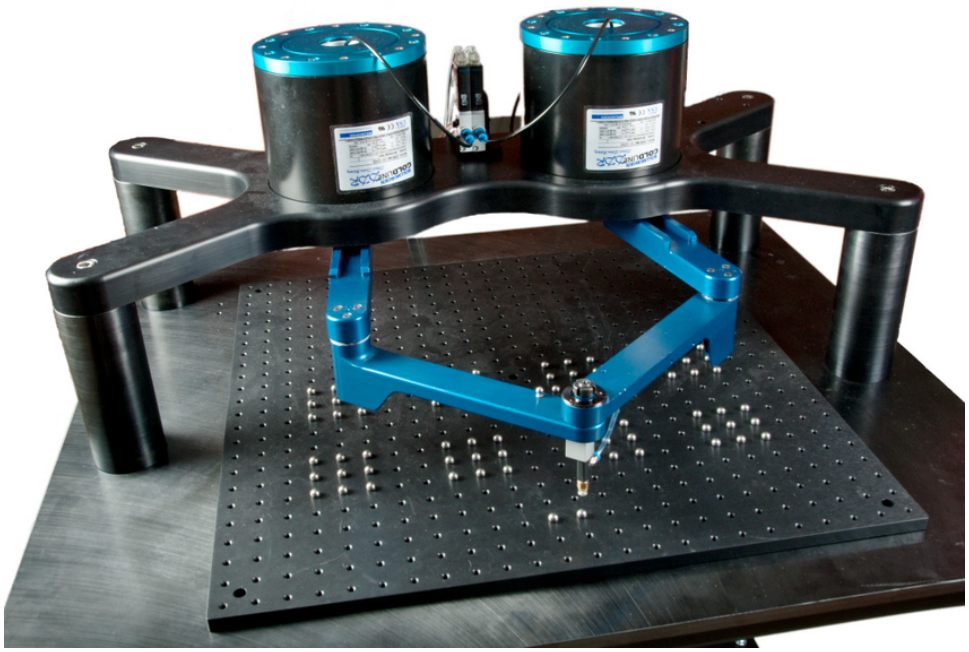

Figure 3: DexTAR, a 5-bar linkage planar robot implementation[7]

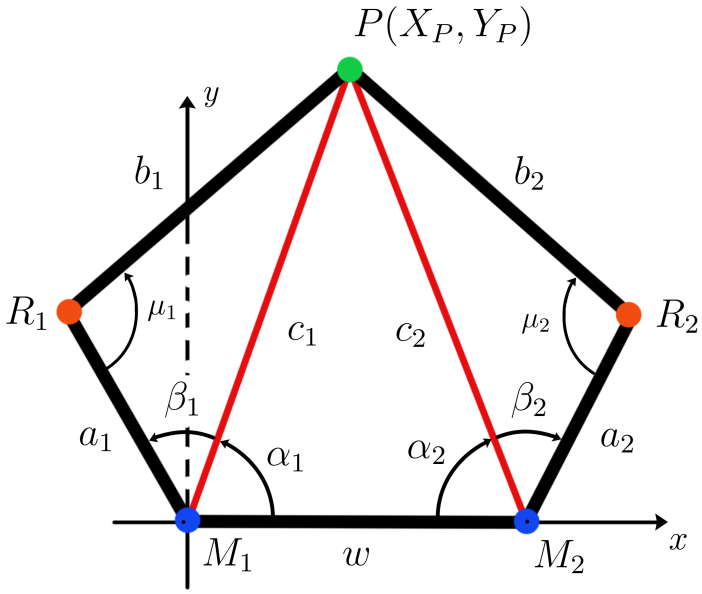

Figure 4: Notation used for 5BL

Figure 4 lays out the notation that is used to describe the 5BL throughout this work.

One major drawback of the 5BL design is the introduction of Working Modes and Singularities which need not be considered in gantry style designs.

1.3.1 Working Modes

In the inverse direction: for any end-effector position, there are four possible combinations of motor angles that will produce that position. These four solutions correspond to four Working Modes (Fig. 5). If the 5BL is thought of as two arms, each originating from a motor and joining at the end-effector, each arm can be reflected across its associated motor-end-effector chord (c_i), giving a convex (+) and concave (−) solution. Two arms with two

configurations each gives the four Working Modes. Table 1 gives the rules for combining the intermediate angles (α and β) to produce motor angles (θ) according to a specific Working Mode.

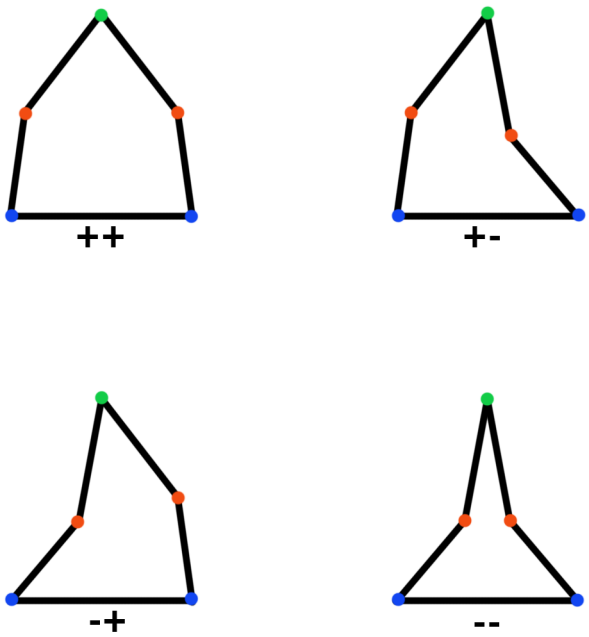

Figure 5: The four Working Modes of 5BL.

Working Mode	θ_1	θ_2
$++$	$\alpha_1 + \beta_1$	$\alpha_2 + \beta_2$
$+-$	$\alpha_1 + \beta_1$	$\alpha_2 - \beta_2$
$-+$	$\alpha_1 - \beta_1$	$\alpha_2 + \beta_2$
$--$	$\alpha_1 - \beta_1$	$\alpha_2 - \beta_2$

Table 1: Rules for combining α_i and β_i to get motor angles θ_i according to Working Mode.

In the forward direction: for any pair of motor angles, there are two possible end-effector locations (Fig. 6). The two solutions are achieved by reflecting the floating links across the line between the distal ends of the driving links ($\overline{R_1 R_2}$), which, similar to the Working Modes, gives a convex (+) solution and a concave (−) solution state.

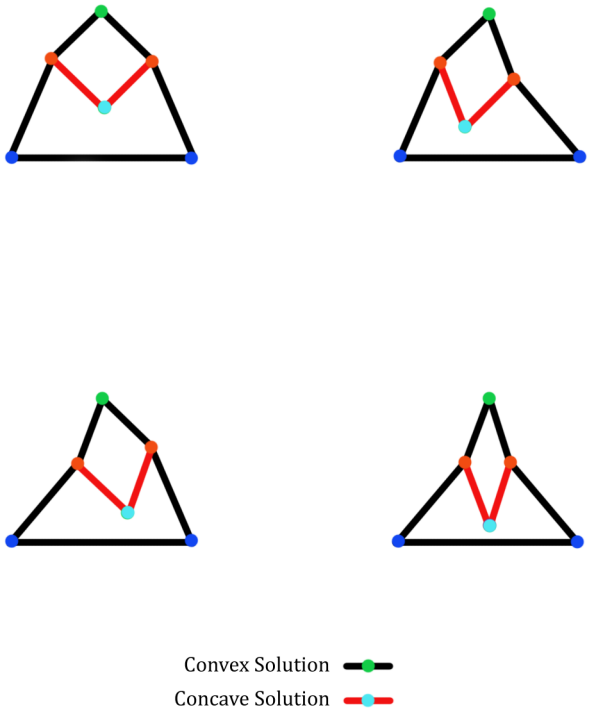

Figure 6: Two solutions for each of 4 Working Modes for a given end-effector location: convex (black), concave (red)

Along with Working Modes, the 5BL requires consideration of Singularities.

1.3.2 Singularities

Singularities occur when any two links (except for the grounding link between the motors) become parallel (Fig. 7). The locations of Singularities in the Workspace depend upon the relative lengths of the links of the 5BL. Mode 2 Singularities, since they involve a driving link, can be navigated out of without issue by activating the associated motor. Mode 1 Singularities, on the other hand, are problematic because they involve two floating links, which introduces an indeterminate degree of freedom that cannot be directly controlled by either of the motors.

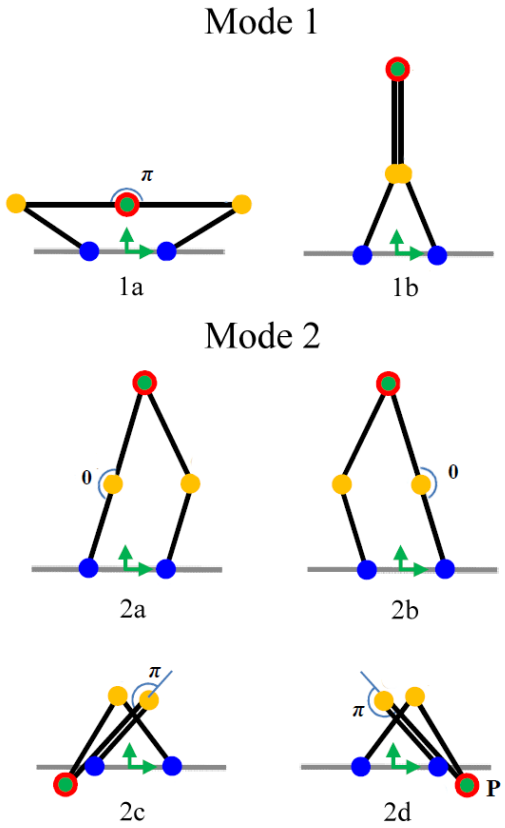

Figure 7: Singularity Modes[6]

Typically, the positive and negative solutions dictate mutually exclusive paths (Fig. 8 Left) and transitioning from one to the other is only possible by commanding the end-effector through a new bridging-path between the positive and negative solution paths. Mode 1a Singularities arise when the positive and negative solutions intersect (Fig. 8 Middle and Right).

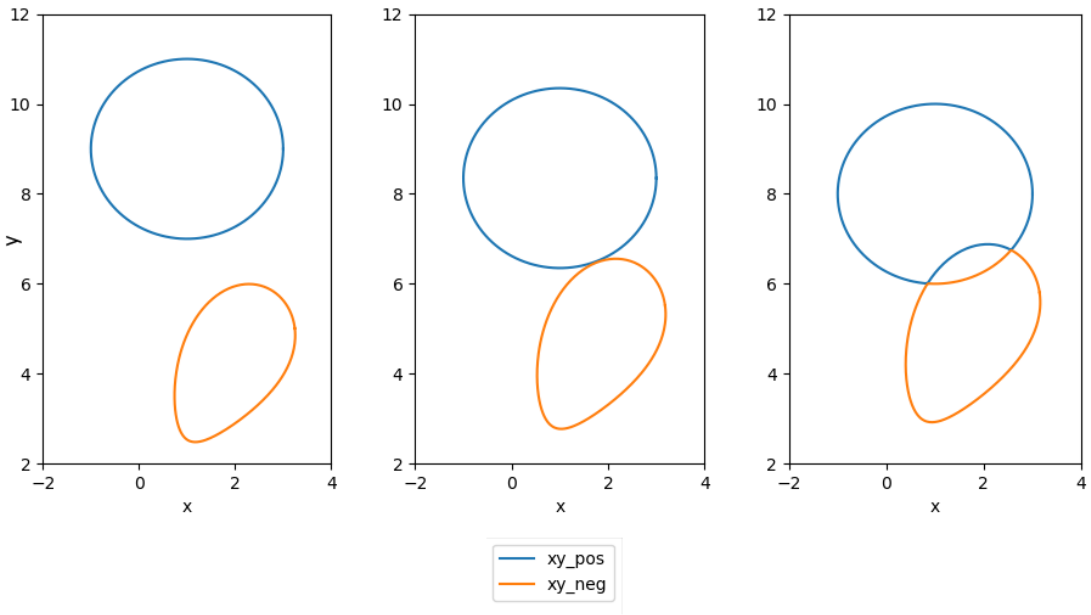

Figure 8: Intersection of positive and negative solutions

At the intersection of a positive and negative solution path, the 5BL can be seen to be in a Mode 1a Singularity state (Fig. 9). When exiting the Mode 1a Singularity, the end-effector can adopt either the positive solution or the negative solution. There is no control over which solution is adopted without introducing a mechanism at one of the R_i joints to force the end-effector toward one solution or the other.

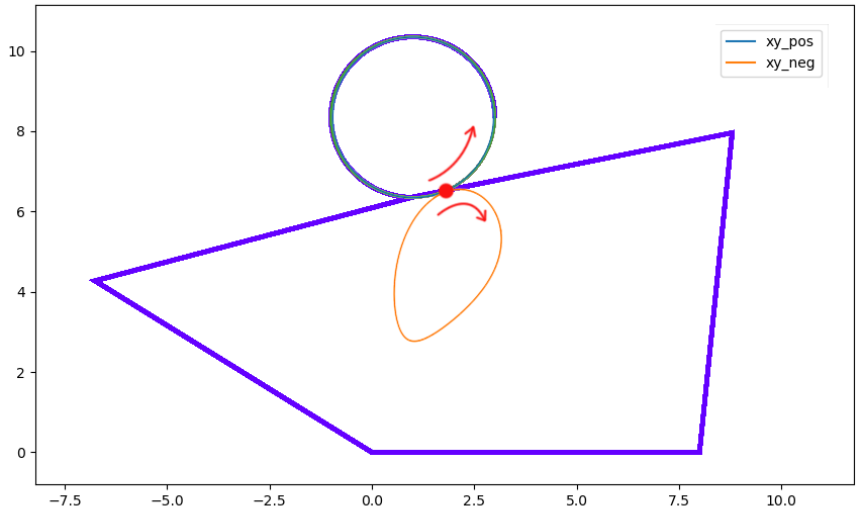

Figure 9: Singularity at intersection of positive and negative solutions

When in a Mode 1a Singularity the 5BL degenerates into an equivalent 4BL momentarily as the b_i links effectively reduce to a single 4BL coupler link. Using 4BL analysis, all the Mode 1a Singularities of the 5BL can be found by computing the coupler curve given by the end-effector location midway along the degenerate coupler linkage. This Singularity path produces a hole in the Workspace (Fig. 10). Keeping the end-effector outside of this hole avoids Mode 1a Singularities in the ++ Working Mode.

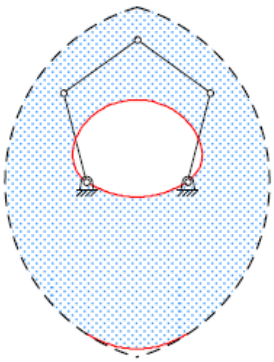

Figure 10: Mode 1a Singularity Trace produces a hole in the Workspace[2]

Mode 1b Singularities have two classes. Class I is when b_1 and b_2 are the same length. This poses a unique problem: while in a Mode 1b Class I Singularity, R_1 and R_2 are coincident, which allows the floating links and end-effector to rotate freely about the Singularity Locus (Fig. 11). The Mode 1b Class I Singularty trace, if it exists, is a circle of radius b_i centered at the location where $R1$ and $R2$ are coincident. Mode 1b Class I Singularities can easily be avoided by making b_1 and b_2 different lengths. By doing so, R_1 and R_2 cannot be coincident anywhere in the workspace. This brings us to Mode 1b Class II Singularities. Mode 1b Class II Singularities, like Mode 2 Singularities, are not problematic because the system can drive itself out of these Singularity cases.

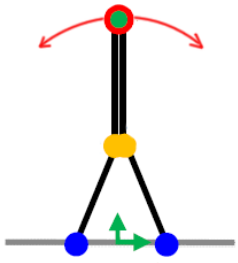

Figure 11: Free motion in Mode 1b Singularity

Mode 1 singularities can sometimes be avoided by using a different Working Mode, as illustrated in Figure 12: in the ++ Working Mode, the system exhibits a Mode 1a Singularity (left) and 1b Singularity (right), but operating in the +− Working Mode, the Mode 1 Singularities are avoided. For a given path, if all Mode 1 singularities can be avoided by using a different Working Mode without introducing any new singularities in the alternative Working Mode, then the path can be traced successfully. if any Mode 1 singularities remain or are introduced by alternative Working Modes, then either the system parameters or the path must be altered.

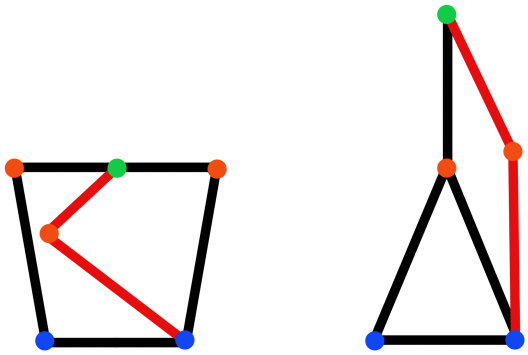

Figure 12: **Black:** Working Mode ++ in Mode 1 singularity. **Red:** Working Mode +- avoids singularity.

Campos et al. further discusses the Workspaces and Singularities associated with each Working Mode[2] for a system with links of equal length.

1.4 Gantry vs 5BL

Gantry	5BL
simple kinematics	complex kinematics
complex & costly assembly	simpler & cheaper assembly
not rigid (belts, cables, etc)	rigid (linkages and pins)
small loading capacity	large loading capacity
slower motion	faster motion
large inertia of end-effector	lower inertia of end-effector[4]
singularities only at boundaries of Workspace	Singularities within Workspace

Table 2: Brief comparison between typical x-y gantry and 5BL

The 5BL is simply comprised of two motors fixed in place, four rigid links, three pin joints (link to link), and two coupled joints (link to motor). Note that no gears, set screws, pulleys, timing belts - which require high tolerance and therefore more costly - are needed in a 5BL assembly. The design is also very versatile: the pin joints could be fashioned from various bearings or hinges, the links could be cut out of nearly any material (wood, sheet metal, acrylic, etc), the motors could be servos or steppers. The flexibility in building materials makes the 5BL far easier to build at home than xy gantry designs, which require purchasing special components such as pulleys or set screws.

As previously discussed, the introduction of Working Modes and Workspace Singularities is a major drawback of the 5BL design. Another drawback is the complexity in the kinematics that govern it and more importantly the lack of software to drive such a robot. If open source software were made available, as well as design guidelines for 5BL robots, then accessibility of planar robot applications (namely 3D printing and laser cutting) would be increased dramatically by making it easier to build and program a planar robot at home.

2 Governing Equations

2.1 Forward Kinematics

The forward kinematics provide a solution for the coordinates of the end-effector when all the characteristic link lengths and angles of driven links are known.

The idea of how to find the coordinates of the end-effector is quite simple. The coordinates of $R1$ and $R2$ are found with a direct trigonometric calculation. Then, using the equation of a circle (effectively the distance formula) we look for the intersection of two circles with centers at $R1$ and $R2$ and radii b_1 and b_2, respectively (Fig. 13).

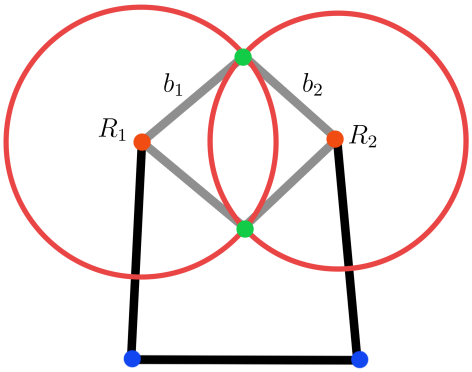

Figure 13: Intersections of two circles representing + and − solutions.

$$x_{R1} = a_1 cos(\theta_1) \tag{1}$$

$$y_{R1} = a_1 sin(\theta_1) \tag{2}$$

$$x_{R2} = a_2 cos(\theta_2) + w \tag{3}$$

$$y_{R2} = a_2 sin(\theta_2) \tag{4}$$

$$b_1^2 = (x_P - x_{R1})^2 + (y_P - y_{R1})^2 \tag{5}$$

$$b_2^2 = (x_P - x_{R1})^2 + (y_P - y_{R2})^2 \tag{6}$$

While simple in conception the algebra is rather arduous. For readability, placeholders v_1 through v_6 will introduced later on.

First, Equations 5 and 6 are rearranged as quadratic equations with respect to x_P. Employing the quadratic formula and rearranging:

$$(x_P - x_{R1})^2 = -y_P^2 - y_{R1}^2 + 2y_P y_{R1} + b_1^2 \tag{7}$$

$$(x_P - x_{R2})^2 = -y_P^2 - y_{R2}^2 + 2y_P y_{R2} + b_2^2 \tag{8}$$

Subtracting Equation 8 from Equation 7 and rearranging for x_P:

$$\begin{aligned} x_P &= \left(\frac{y_{R1} - y_{R2}}{x_{R2} - x_{R1}}\right) y_P + \left(\frac{b_1^2 - b_2^2 - a_1^2 + x_{R2}^2 + y_{R2}^2}{2(x_{R2} - x_{R1})}\right) \\ &= v_1 y_P + v_2 \end{aligned} \tag{9}$$

Where:

$$v_1 = \frac{y_{R1} - y_{R2}}{x_{R2} - x_{R1}} \tag{10}$$

$$v_2 = \frac{b_1^2 - b_2^2 - a_1^2 + x_{R2}^2 + y_{R2}^2}{2(x_{R2} - x_{R1})} \tag{11}$$

Now Equation 9 can be substituted into Equation 5, then solved as a quadratic equation with respect to y_P:

$$y_P = \frac{-v_4 \pm \sqrt{v_4^2 - 4v_3v_5}}{2v_3} \tag{12}$$

Where:

$$v_3 = 1 + v_1^2 \tag{13}$$

$$v_4 = 2(v_1v_2 - v_1x_{R1} - y_{R1}) \tag{14}$$

$$v_5 = a_1^2b_1^2 - 2v_2x_{R1} + v_2^2 \tag{15}$$

As discussed earlier, there are two solutions for a given set of characteristic lengths and motor angles, corresponding to two possible intersections of the circles (Fig. 6).

2.1.1 Application of Forward Kinematic Equations

When given the characteristic link lengths of a 5BL and a set of motor angles, the end-effector path can be found with the following process:

For all times steps:

1. Use Equations 1, 2, 3, and 4 to obtain all $R1$ and $R2$.
2. Solve Equations 10, 11, 13, 14, and 15 to obtain all placeholders.
3. Solve the quadratic Equation 12 using the **positive** radical term. Then solve Equation 9 using the y_P values just obtained. The combination of these x_P and y_P solutions gives us the "positive" forward solution set.

4. Repeat Step 3 using the **negative** radical term in Equation 12. The combination of these new x_P and y_P solutions gives us the "negative" forward solution set.

5. At this point, we have two xy sets. The positive solution is associated with the convex solution; the negative solution is associated with the concave solution. Given that no Mode 1 Singularities are present in the solution, the solutions will be mutually exclusive and the desired solution can be selected.

2.2 Inverse Kinematics

2.2.1 Finding Motor Angles

The inverse kinematics provide a solution for the angles of the driven links when the characteristic link lengths and end-effector coordinates are known.

The inverse kinematics are far simpler than the forward kinematics. They are solved rather simply using the distance formula and the law of cosines.

Using the distance formula, c_i can be expressed as follows:

$$c_i = \sqrt{(x_p - x_{Mi})^2 + (y_p - y_{Mi})^2} \tag{16}$$

Using the law of cosines, α and β can be expressed as follows:

$$\alpha_i = arccos\left(\frac{c_j^2 - c_i^2 - w^2}{-2c_i w}\right) \tag{17}$$

$$\beta_i = arccos\left(\frac{b_i^2 - a_i^2 - c_i^2}{-2a_i c_i}\right) \tag{18}$$

Letting M_1 be at the origin and M_2 be located at $(w, 0)$, as prescribed in Figure 4:

$$c_1^2 = (x_p - x_{M1})^2 + (y_p - y_{M1})^2 = x_p^2 + y_p^2 \tag{19}$$

$$\begin{aligned} c_2^2 &= (x_p - x_{M2})^2 + (y_p - y_{M2})^2 \\ &= (x_p - w)^2 + y_p^2 \\ &= x_p^2 - 2x_p w + w^2 + y_p^2 \\ &= c_1^2 - 2x_p w + w^2 \end{aligned} \tag{20}$$

Substituting Equation 20 into Equation 17 for $i = 1$ allows us to simplify our expression for α_1:

$$\begin{aligned} \alpha_1 &= arccos\left(\frac{c_2^2 - c_1^2 - w^2}{-2c_1 w}\right) \\ &= arccos\left(\frac{c_1^2 - 2x_p w + w^2 - c_1^2 - w^2}{-2c_1 w}\right) \\ &= arccos\left(\frac{x_p}{c_1}\right) \end{aligned} \tag{21}$$

Our expression for α_2 is similarly simplified:

$$\begin{aligned} \alpha_2 &= arccos\left(\frac{c_1^2 - c_2^2 - w^2}{-2c_2 w}\right) \\ &= arccos\left(\frac{c_1^2 - (c_1^2 - 2x_p w + w^2) - w^2}{-2c_2 w}\right) \\ &= arccos\left(\frac{-x_p + w}{c_2}\right) \end{aligned} \tag{22}$$

For β_i, no simplifications arise from algebraic manipulation, so it is kept in the form found directly from the law of cosines (Equation 18).

2.2.2 Finding Transmission Angles

The transmission angle, μ_i, is the angle between the driving link, a_i, and the floating link which it drives, b_i, which can be written as $\angle M_i R_i P$ (Fig. 4). The 5BL has two transmission angles, one for each driving arm. Transmission angle is crucial in the transmission of force through the linkages to move the end-effector. The most effective transmission angle is 90°, and it is recommended not to vary more than ±50° from 90°.[1]

The transmission angles can be found similarly to β_i using the law of cosines:

$$\mu_i = arccos\left(\frac{c_i^2 - a_i^2 - b_i^2}{-2a_i b_i}\right) \tag{23}$$

Transmission angles will be further discussed and analyzed in Section 4.2.

2.2.3 Application of Inverse Kinematic Equations

When given the characteristic link lengths of a 5BL and a set of xy points corresponding to an end-effector path, the motor angles that would produce that path can be found with the following process:

For all time steps:

1. Solve Equation 19 and 20 for c_1 and c_2.
2. Solve Equations 21 and 22 for α_1 and α_2.
3. Solve Equation 18 for β_1 and β_2.
4. To obtain the set of motor angles, θ_1 and θ_2, select a Working Mode and combine α_i and β_i according to Table 1.

3 Software Implementation

3.1 Functionality Overview

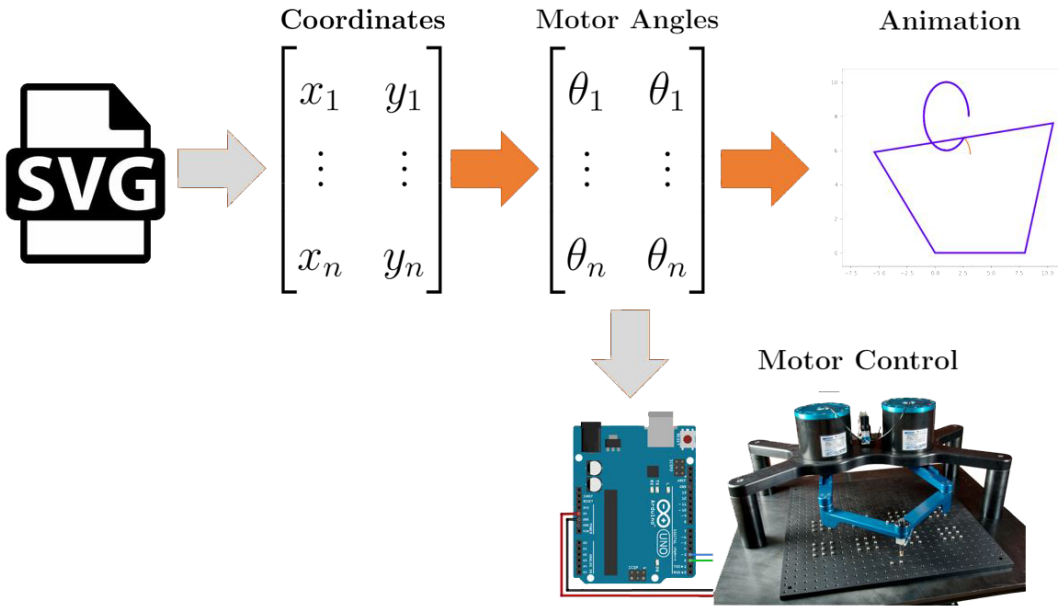

Figure 14: The big picture of 5BL software functionality (orange is implemented, gray is not implemented in the most current code release at the time of writing (Rev 7))

The core functionality of this software (visualized in Figure 14) is taking a a desired path and applying the inverse kinematics of the 5BL to produce a set of motor angle pairs which, when fed to a physical 5BL, would result in the end-effector tracing the desired path. In practice, such paths often manifest as scalable vector graphic (SVG) files[1]. The current implementation outsources the discretization of the path from SVG to xy points to an existing web service, and does not include functionality to output code directly to a micro-controller to drive a 5BL. Additional functionality that has been implemented but not shown in Figure 14 includes solvers for forward kinematics, transmission angles, and error

[1]Chris Coyier gives a great introduction to SVG files.

sensitivity. The transformation from SVG path to motor angles and finally motor control is broken into the following processes:

1. Convert SVG to xy CSV
2. Convert xy CSV to xy Array
3. Convert xy Array to Motor Angle Array
4. Final conversion for motor control

A Python library was written to fulfill this functionality. Python was chosen as the language to implement this software because it is free, open-source, widely-used, and has many robust libraries ready to be imported for use (such as Matplotlib for making plots and NumPy for linear algebra). Furthermore, Python is a very high-level coding language, meaning that it is highly abstracted from the actual computing going on under the hood and tends to resemble natural spoken language more than a low level language such as Assembly; as such, it is very accessible to novice coders. Since the motivation for this software is increasing accessibility, it is logical to use a beginner-friendly language.

The first step in moving from SVG path to motor angles is discretizing the SVG path into a list of xy points. Methods for manipulating SVG paths to xy points are described below:

1. **Python:** Many Python libraries exist for manipulating SVG files. One library that showed promise in functionality was SVG.path. After some experimentation, however, it was found that this library has numerous problems in reading SVG files accurately. The library works by parsing an SVG file and translating it into a new syntax. Once the SVG is translated, it is very easy to parse the new path for a list of xy points, however, as noted, the translation does not always work very well. As such, native Python manipulation of SVG files was not implemented.

2. **JavaScript:** JavaScript is a scripting language originally designed for handling dynamic web-page content, such as image animation. As such, JavaScript was designed to be compatible with SVG and has builtin methods (functions) for finding xy coordinates along SVG paths. Using these methods, it would be very simple to extract xy points from an SVG file using JavaScript.

   It is possible to embed JavaScript within Python, but, due to time constraints and the availability of simpler solutions for SVG translation, this was not pursued for preliminary proof-of-concept work. Potential methods of integrating JavaScript with Python will be discussed as Future Work in Section 7.1.

3. **Coördinator:** Coördinator is an open-source web application that converts SVG paths to xy points. It is very accurate and is able to handle multi-path SVG files. All SVG paths studied in this work were converted to xy CSV files using Coördinator.

   One issue that arises is that for multi-path SVG files there must be a way to differentiate the paths once translated to xy points in order to interrupt the end-effector process between paths. For example, if the 5BL was outfitted with a laser-cutter, the machine must know to turn the laser off when traveling beteween paths of a job. Coördinator does not allow for this functionality, but works as a proof-of-concept for this project.

Once a CSV file of xy points is produced, it must be read into Python and converted to an array structure compatible with linear algebra operations.

Using the builtin CSV Python library or the Pandas library, an xy CSV can be converted to a NumPy array structure that supports linear algebra. Once the path has been discretized into xy points and put into a structure that supports linear algebra methods, it is time to solve the inverse kinematics to produce a set of motor angles for the path.

Equations 16, 21, 22, and 18 are solved for each xy point to produce a set of corresponding system angles: α_1, α_2, β_1, β_2. As summarized in Figure 5, these can be combined in one

of four ways according to the desired Working Mode to finally give the set of motor angle pairs.

For applications where precise positioning is required, either servo motors or stepper motors are used. The selection of motor type is dependent upon the application of the 5BL and availability of components.

Servo motors are typically controlled by sending a specified shaft angle through an internal controller to the motor. The internal controller transforms the shaft angle to a corresponding pulse-width to send to the motor. If servo motors are used to drive the 5BL system, then the motor angles can be sent directly to the servo units.

Stepper motors, on the other hand, are controlled by sending a number of steps and a direction through an external controller to the motor. The step value fed to the controller can be relative to the current shaft rotation or absolute to a fixed shaft rotation. If stepper motors are used to drive the 5BL system, then a simple linear transformation can be applied to convert the motor angles to steps:

$$S = \frac{\theta n}{2\pi} \tag{24}$$

Where S is the absolute step and n is the number of steps per revolution of the motor shaft (given in the datasheet for the specific motor used).

Included in the software is a conversion using Equation 24 that can output both absolute and relative steps for a path.

3.2 Performance Optimization

This software inherently deals with large arrays in order to process complex and high resolution paths. Populating large arrays, if done poorly, can take a lot of computing time. A very simple and effective strategy in speeding up the process of populating large arrays

is vectorizing. Vectorized code operates on multiple components of a vector simultaneously (Fig. 15). Non-vectorized code operates on each component of a vector individually in a serial manner, typically achieved with for loops (Fig. 16).

```
######################
# VECTORIZED DIVISION #
######################
# initialize two vectors of random numbers
# size 10x1
a = np.random.rand(10,1)
b = np.random.rand(10,1)

c = a/b
```

Figure 15: Vectorized division

```
##########################
# NON-VECTORIZED DIVISION #
##########################
# initialize two vectors of random numbers
# size 10x1
a = np.random.rand(10,1)
b = np.random.rand(10,1)
# initialze empty vector to store results
c = np.zeros(10,1)

for k in range(len(a)):
    c[k] = a[k] / b[k]
```

Figure 16: Non-vectorized division

By carrying out operations simultaneously rather than serially, vectorized code is able to reduce computing time significantly. The majority of the software is vectorized for this reason; however, some array operations cannot be vectorized, such as computing the forward kinematic solution: all intermediate computations are vectorized (such as solving the positive and negative solutions), but choosing between the positive and negative solutions at each time step to produce the final forward solution is dependent on an element-by-element comparison of the positive and negative solutions. As such, a variety of methods for serial array population were analyzed for speed:

1. Append python list, convert at end with numpy.asarray()
2. Append python list, convert at end with numpy.reshape()
3. Write into preallocated NumPy array
4. Append Numpy array

The four methods were tested by populating various sized arrays with random numbers. Methods 1, 2, and 3 were shown to be comparable, while Method 4 was up to 100 times slower than the other methods. For comparison, times for each respective array size were normalized against Method 3, which was shown to be the fastest population method. Table 3 shows the data collected for arrays up to size 100,000.

Array Size	**Method 1**	**Method 2**	**Method 4**
100	1.63	1.63	53.8
500	1.19	1.29	46.15
1,000	1.13	1.08	24.19
2,000	1.05	1.07	31.60
10,000	1.13	1.14	46.04
100,000	1.10	1.02	98.33

Table 3: Array Population Time, Normalized Against Method 3

Figure 17 excludes Method 4 since it was drastically slower than Methods 1, 2, and 3. It shows that for small arrays, Methods 1 and 2 take up to 60% longer than Method 3, but for larger arrays the difference drops to around 15%. Method 3, writing into a preallocated Numpy Array, was chosen to be implemented in any non-vectorized array populations in the software.

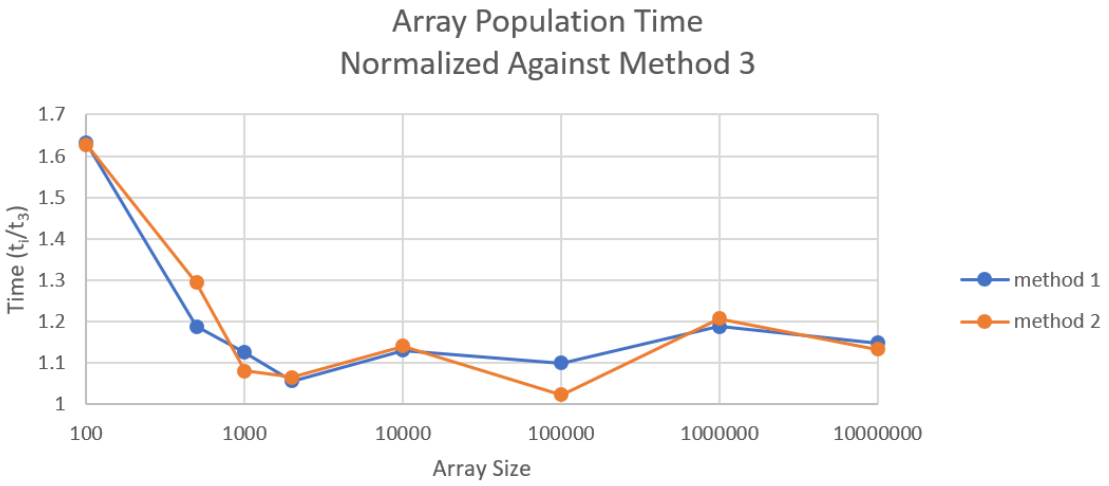

Figure 17: Array Population Time, Normalized Against Method 3

4 System Analysis

For the following analyses:

- System configuration is denoted $[a_1, b_1, a_2, b_2, w]$, in holding with the syntax defined in the accompanying software.
- Error analysis is for a perturbation of $\pm 5^\circ$ of the motors.
- Link length modification is 30% lengthening.
- In each figure, the system is traced in blue for reference.

4.1 Error Sensitivity

Once fabricated, an important source of error in operating a 5BL is motor angle error. No motor can be driven to the exact angle the user desires. It is important, therefore, to understand how the system will respond to this source of error.

The 5BL, being a nonlinear system, will not have uniform error sensitivity throughout its Workspace. The error sensitivity will also depend upon the relative linkage lengths defined by the system configuration.

At each analyzed point, two traces are drawn, forming a skewed 'x' shape: the backward leaning trace is associated with M_1 being fixed while M_2 is perturbed $\pm 5^\circ$, the forward leaning trace is associated with M_1 being perturbed $\pm 5^\circ$ while M_2 is fixed.

A short trace means that the system is less sensitive to error in the perturbed motor since the end-effector is minorly translated by the perturbation. A long trace means that the system is more sensitive to error in the perturbed motor since the end-effector is sigificantly translated by the perturbation.

In Figure 18, a reference configuration is given (subplot A) along with five modified configurations: one for the modification of each link length.

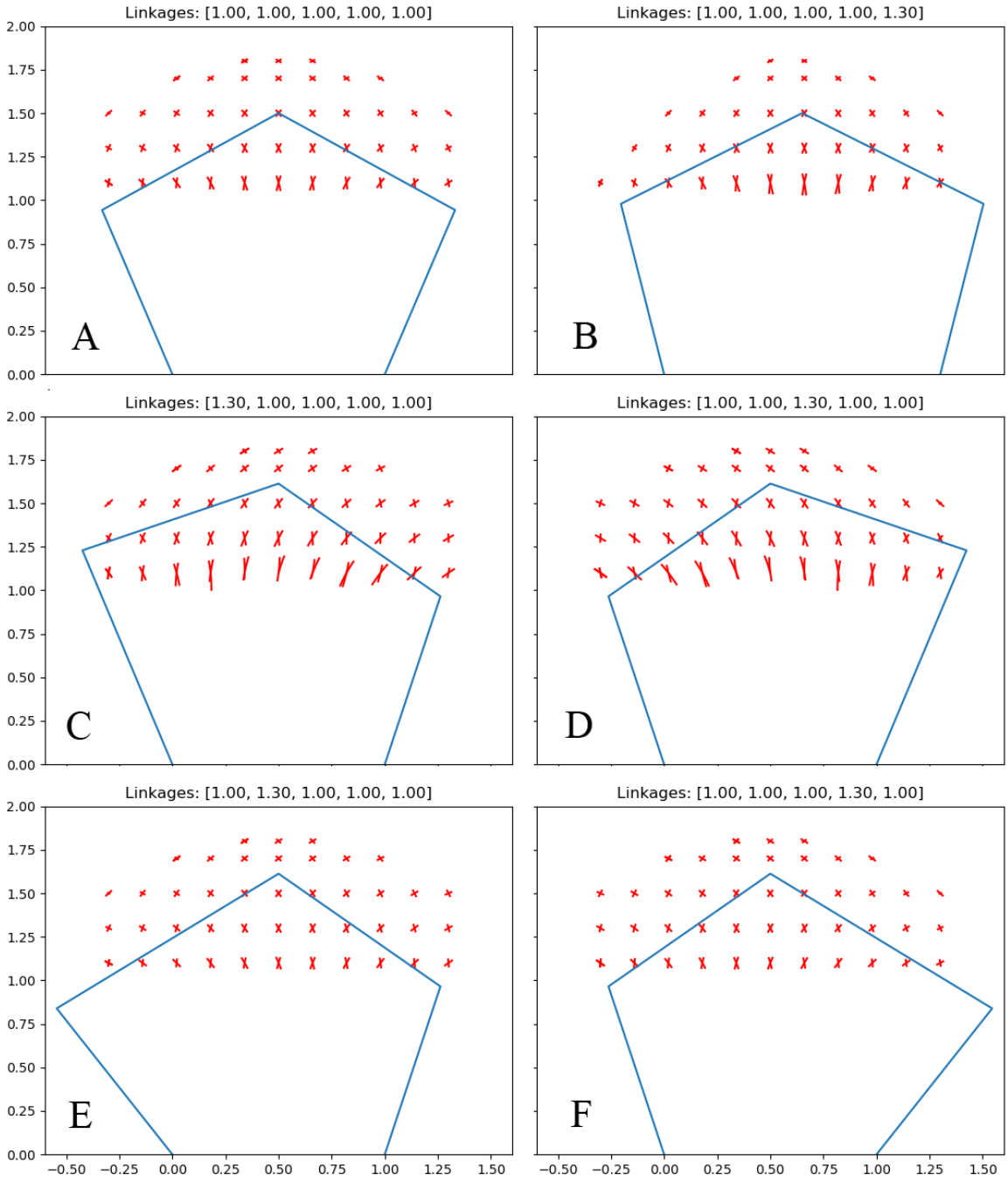

Figure 18: Error Sensitivity Due to $\pm 5^{\circ}$ Motor Perturbation for various System Configurations

As seen in Figure 18, increasing the length of links in direct connection with the motors (subplots B, C, and D) increased error sensitivity in the area local to the centroid of the machine (near $(0.5, 1.0)$). This is evidenced by the stretching of traces in this region. Increasing the lengths of the floating links, however, decreased the error sensitivity of the system, as evidenced by the slight contraction of traces in the region local to the centroid. What happens if both floating links are stretched?

In Figure 19, a reference configuration is given along with three modified configurations: in each, a pair of links is modified.

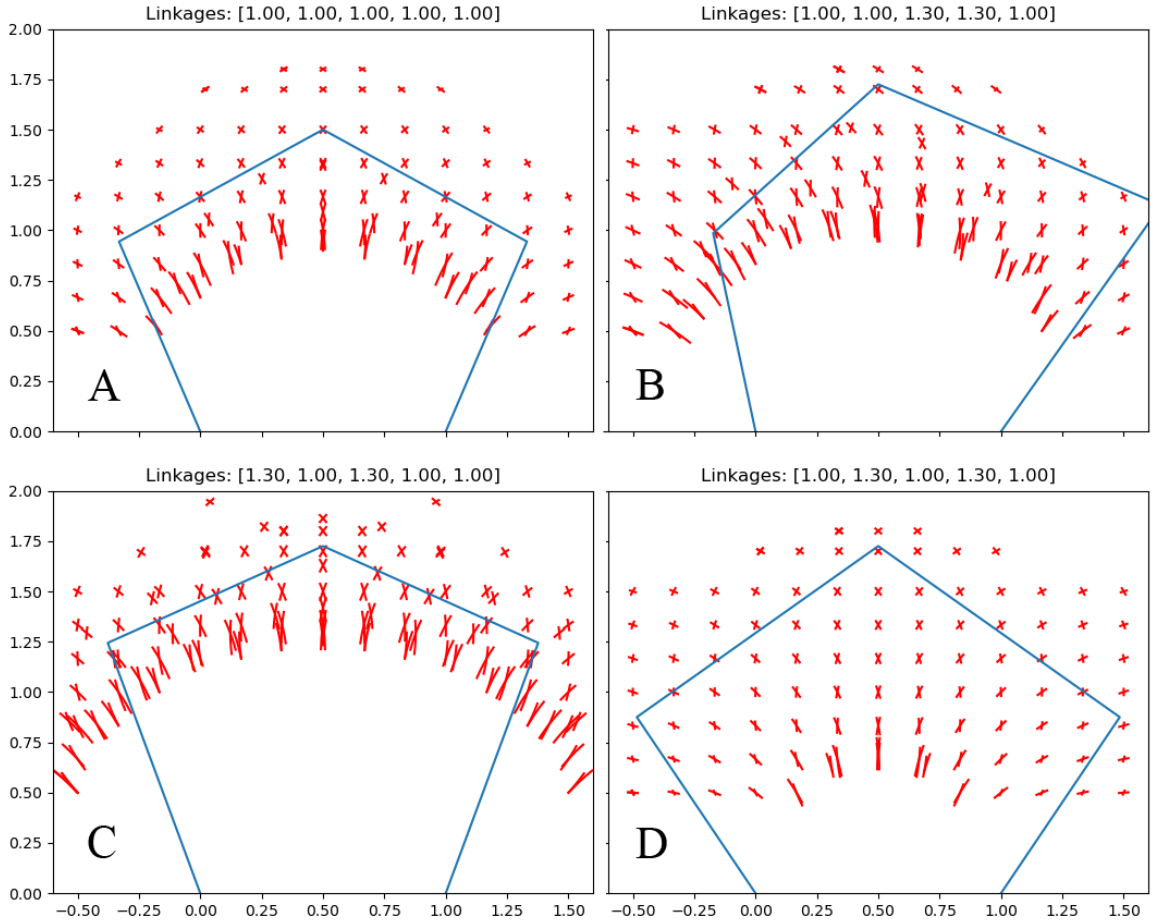

Figure 19: Error Sensitivity Due to $\pm 5^\circ$ Motor Perturbation for various System Configurations

Figure 19 shows that when both of the floating links are lengthened simultaneously (subplot D), the error sensitivity of the system is decreased throughout much of the Workspace. The error response is also made more uniform throughout the workspace. Furthermore, the Workspace is made larger. This is evidenced by the convex boundary at the bottom of sample cloud being pushed down. This boundary represents the Mode 1a Singularity trace (Fig. 10). As previously noted, this trace represents a hole in the workspace that must be avoided. Note the elongation of error traces near this boundary: the system is particularly sensitive to error near the Singularity trace. By stretching the floating links, the driving links are able to open to wider angles and the end-effector is able to move closer to the x-axis before reaching Mode 1a Singularity (Fig. 20).

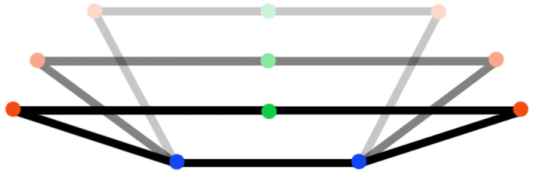

Figure 20: Extending floating links pushes Mode 1a Singularity trace toward x-axis

Designing floating links to be longer than the remaining three links results in a system more tolerant to motor angle error, as well as having a more uniform error response and larger workspace.

4.2 Transmission Angle

As previously noted, having a transmission angle near 90° makes force transmission through the joints more efficient. Various system configurations were tested to see the effect of different link lengths on transmission angles throughout the workspace. Figure 21 shows six system configurations. The transmission angle for all subplots is μ_1. Unsurpris-

ingly, transmission angle increases radially from the associated motor: as the end-effector approaches M_i, the transmission angle μ_i must approach 0° as b_i folds toward a_i; as the end-effector departs from M_i, the transmission angle μ_i must approach 180° as b_i unfolds away from a_i. The transmission angle color maps for all tested system configurations are nearly identical. Modifying the link lengths had a very insignificant effect on the transmission angles throughout the workspace.

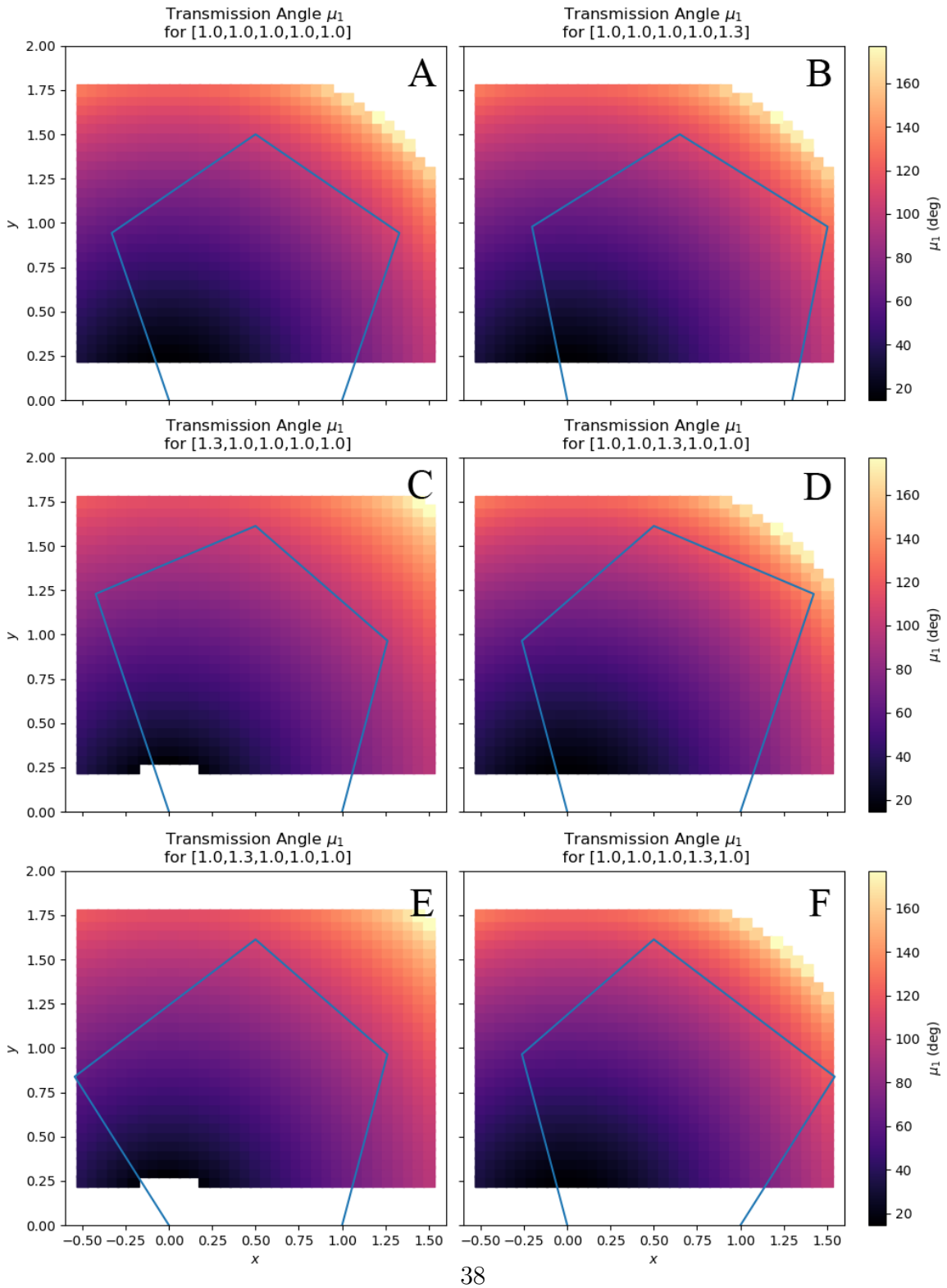

Figure 21: Transmission Angle for Various System Configurations

4.3 Error Sensitivity Relationship

At first glance, superimposing the error sensitivity and average transmission angle plots was thought to reveal an inverse relationship between the two (as transmission angle increases, error sensitivity tends to decrease (Fig. 22)). However, as the system configuration is modified, the error sensitivity changes significantly while the transmission angle distribution stays nearly constant. If the two were closely related, they should be affected to a similar degree by configuration modification.

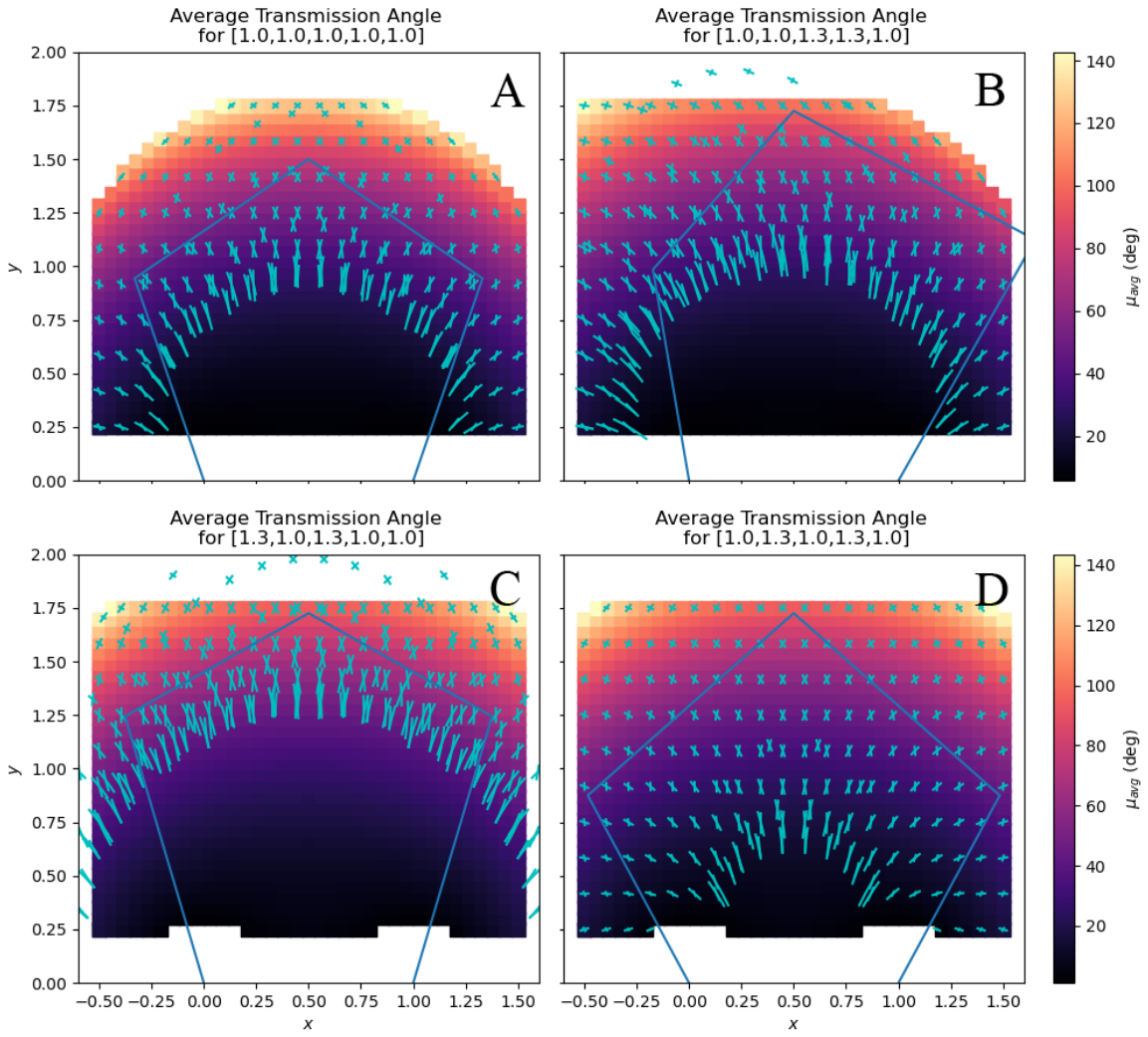

Figure 22: Superposition of Error Sensitivity and Average Transmission Angle for Various System Configurations

Upon further consideration, it seems that the changing error sensitivity is more closely related to the relative link lengths, more specifically how the relative link lengths affect the Mode 1a Singularity trace. If the error sensitivity plot is thought of as the superposition of two slope fields (one for each motor error), these slope fields can be see to be near $\pm 45°$ far from the Singularity trace and as the Singularity trace is approached the slope field

lines approach perpendicularity to the Singularity trace. Further analysis of the relationship between system configuration and Singularity regions is required in order to better analyze error sensitivity.

5 Sample Results: N64 Controller

5.1 Kinematics Solver and Animation

One sample image tested was a Nintendo 64 Controller (Fig 23).

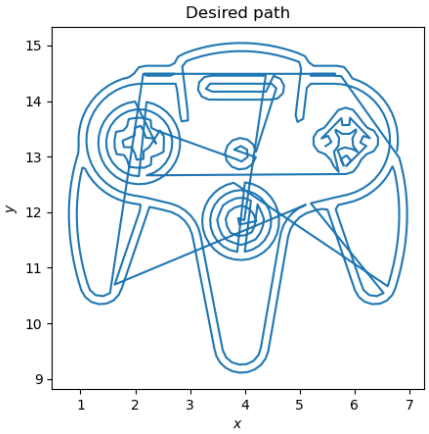

Figure 23: Desired paths for controller image

The path was successfully processed by the 5BL software, including an inverse kinematic solution to generate motor angles, a forward kinematic solution from the motor angles for verification, and animating the system tracing the path.

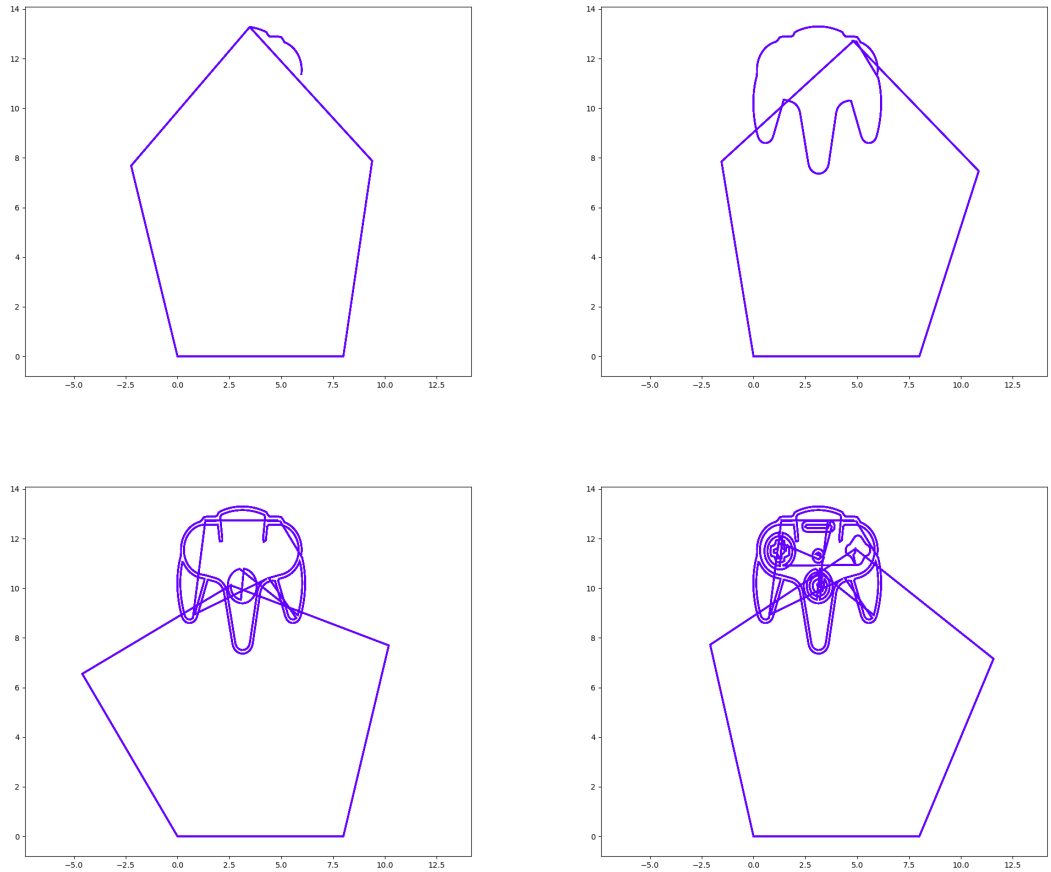

Figure 24: Still frames from animation of path

An interesting discovery made is the relationship between the motor angles and the desired path. Figure 25 shows θ_2 plotted against θ_1. Interestingly, the resulting path resembles the original path but rotated approximately 135° and distorted. This relationship raises a question: could a transform be found that relates the original XY path to the $\theta_1\theta_2$ path (as a function of system configuration and path)?

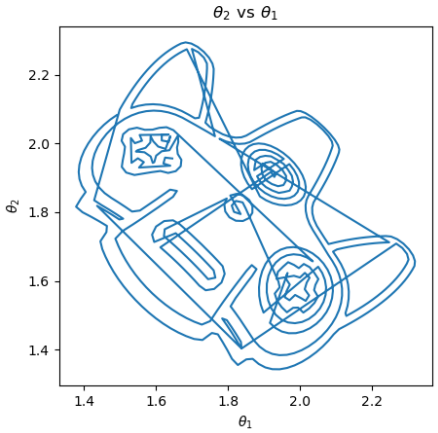

Figure 25: θ_2 plotted against θ_1

5.2 Potential Kinematics Approximation Method

Figures 26 and 27 show how closely related the x and y time signals are related to the θ_1 and θ_2 time signals, respectively. To produce these graphs, the xy points were first rotated 135°, as informed by comparing Figure 25 to Figure 23. Then, the new x_{rot} and y_{rot} signals were scaled and shifted to match the θ_1 and θ_2 signals. The appropriate scaling factor was found to be 7.5, and the shifting factors were 25.5 and 20 for x_{rot} and y_{rot}, respectively.

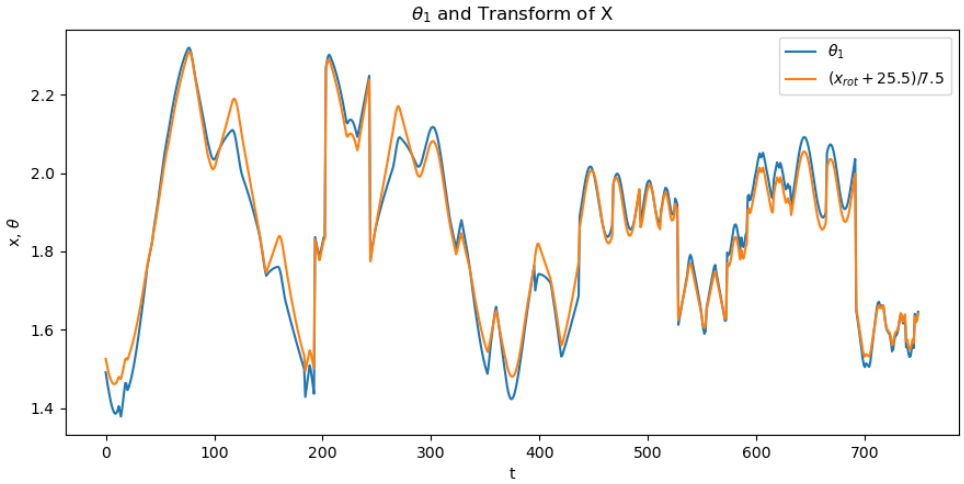

Figure 26: θ_1 and x plotted against time

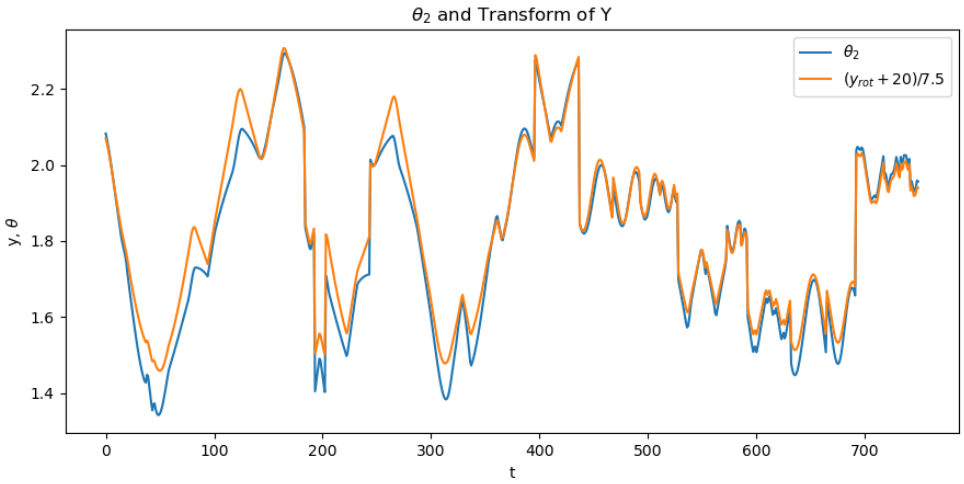

Figure 27: θ_2 and y plotted against time

The transform seems to take the following form:

$$\begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix} \approx \begin{bmatrix}x'\\ y'\end{bmatrix} = \left(\begin{bmatrix} cos\gamma & -sin\gamma \\ sin\gamma & cos\gamma \end{bmatrix} \cdot \begin{bmatrix}x\\ y\end{bmatrix} + \begin{bmatrix}A_1\\ A_2\end{bmatrix} \right) \cdot B \tag{25}$$

Where:

- γ is the rotation factor from xy to $\theta_1\theta_2$ (135°)
- A_1 and A_2 are the shifting factors (25.5 and 20)
- B is the scaling factor (7.5)

The performance of this initial transform, with A and B parameters specified above, was tested for fidelity to the desired path (Fig. 28). Clearly the transform requires some tuning, but these preliminary results are promising.

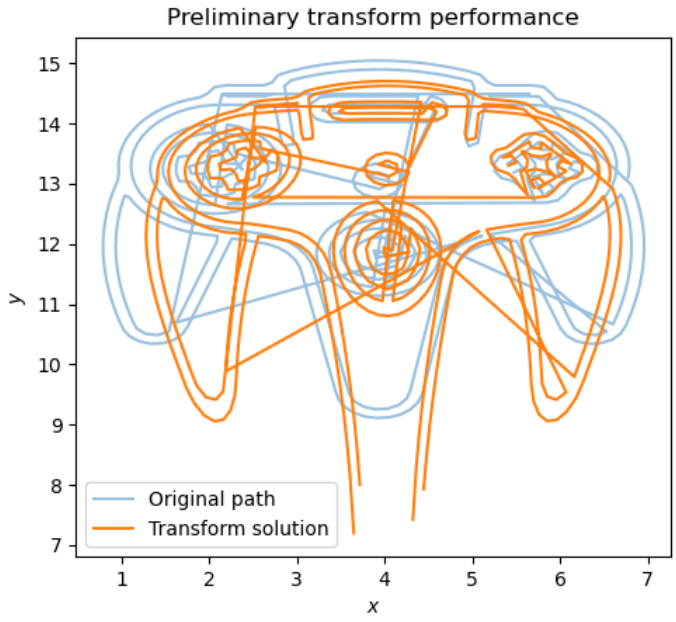

Figure 28: Preliminary transform performance approximating motor angles

Recall from Section 2.2.3 that the inverse kinematic solution required solving a series of equations (Equations 21, 22, 18, 19, and 20). The proposed transform given by Equation 25 is far simpler and faster to solve. Assuming this transform were able to be tuned to give a sufficiently accurate approximation, it would provide an alternative inverse solution for finding the motor angles that place the end-effector at a desired point.

The transform could be applied to scenarios where minimizing computation time is imperative or computing power is very limited. One such application could be real-time control of a 5BL. This could be controlling the 5BL with a joystick or gesture-control. The transform could also be used to approximate the forward kinematic solution:

$$\begin{bmatrix} x \\ y \end{bmatrix} \approx \begin{bmatrix} cos\gamma & sin\gamma \\ -sin\gamma & cos\gamma \end{bmatrix} \left(\frac{1}{B} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} - \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} \right) \tag{26}$$

Recall how complicated the forward kinematics are (Sec. 2.1), and compare these equations to Equation 26. The computational savings made by using this transform rather than the pure forward kinematic solution would be extremely significant. This transform is so simple that it could even be calculated by hand, whereas solving the forward kinematics by hand would be excruciating and error-prone.

6 Applications

Planar robots are the foundation of many robots used in various manufacturing processes. By using various systems as the end-effector of a planar robot a wide range of functionality is possible. As previously discussed, applications that would be of great use to a Maker in the process of rapid-prototyping include:

1. 3d printer
2. laser cutter
3. cnc mill
4. solder machine
5. plotter

7 Future Work

7.1 JavaScript Implementation for SVG Processing

Implementing an SVG conversion function in the software would greatly strengthen the functionality of the software, as this is the only functionality of the software that is currently outsourced. A simple JavaScript program could be written to translate SVG files into xy points. The JavaScript could then be implemented in a number of ways, such as:

- A simple website similar to Coördinator
- Node.js
- Js2Py
- Selenium
- PyExecJS
- v8
- Naked Shell

7.2 Singularities, Working Modes, and Workspace

It would be very helpful to the user if the software included a function to compute the singularities associated with a given system configuration, especially being able to visualize the Mode 1a Singularity trace. Development of mathematics to describe the relationship between the system configuration and Mode 1a Singularity trace would also be beneficial. These mathematics would be an extension of 4BL equations.

Generating a visual representation of the viable Workspace for each working mode of a given system configuration would also be beneficial for path planning. Furthermore, being able to transition between working modes at mid-job could improve functionality and flexibility.

7.3 Linkage Synthesis

Further analysis of how linkage lengths affect singularity location, transmission angle, and error sensitivity throughout the workspace are important analyses that are only discussed at surface level in this work. More in-depth analyses of these topics could inform design parameters on ideal relative linkage lengths.

7.4 Kinematics Approximation

Section 5.2 lays out preliminary ideas on a simple way to approximate the forward and inverse kinematic equations of a 5BL. Further exploration of the proposed transform could produce a very powerful approximation tool for real-time analysis of 5BL kinematics.

7.5 Interactive GUI

The software would greatly benefit from an interactive GUI for translating and scaling the working path, animating the simulation, and importing/exporting data.

8 Acknowledgements

This thesis started out as a passion project in the spring semester of my freshman year. I spent many late nights writing code to compute the forward kinematics of what I would one day learn to be the Five Bar Linkage. As I spent more and more time on this project, I decided I wanted to make it my thesis, but had no idea how to turn an art project into and engineering thesis. Without the help of Professors Zachary Ballard, Josh Bongard, Chris Danforth, Dryver Huston, and John Novotny, I would have a hard-drive full of guilloché spirograph plots and nothing else. I never would have arrived at the Five Bar Linkage, let alone a thesis on the subject.

From the beginning to the end, thank you -

Professor Ballard, for your advice on solving the forward kinematics in the early days.

Professor Novotny, for teaching me about linkages and giving me resources on linkage fundamentals and analysis.

Professor Danforth, for rekindling my passion for this project and helping me find the direction from art project to engineering thesis.

Professsor Bongard, for a pivotal discussion about robotics and sending me to Professor Huston (and also for making Twitch Plays Robotics).

Professor Huston, for being my primary advisor. Your invaluable insights, wisdom, and provoking questions had a tremendous impact on this work.

I am very appreciative of all of your contributions, small and large - in the absence of any of them I would not have made it down this long, windy road.

References

[1] S. S. Balli and S. Chand, "Transmission angle in mechanisms (Triangle in mech)," Mechanism and Machine Theory, vol. 37, no. 2, pp. 175–195, Feb. 2002, doi: 10.1016/S0094-114X(01)00067-2.

[2] L. Campos, F. Bourbonnais, I. Bonev, and P. Bigras, "Development of a Five-Bar Parallel Robot With Large Workspace," presented at the Proceedings of the ASME Design Engineering Technical Conference, Aug. 2010, vol. 2, doi: 10.1115/DETC2010-28962.

[3] "FUYU FSL40 Linear Guide Stage Motion Slide Actuator XY Translation Stage XYZ Table 300mm Stroke X, 300mm Stroke Y, 300mm Stroke Z Ball Screw Module DIY Router [XYZ Stage-T Type]: Amazon.com: Industrial & Scientific." https://www.amazon.com/FUYU-Linear-Motion-Actuator-Translation/dp/B07DVQDSZH (accessed Mar. 30, 2020).

[4] Hoang, Tuong, Trung Vuong, and Bang Pham. 2015. "Study and Development of Parallel Robots Based On 5-Bar Linkage." In.

[5] "How the Maker Movement Is Moving Into Classrooms," Edutopia. https://www.edutopia.org/blog/maker-movement-moving-into-classrooms-vicki-davis (accessed Mar. 31, 2020).

[6] Jung-Hyun Choi, Chan Lee, Ma-Eum Kim, and Jehwon Lee, "Singularity analysis of a 5-bar planar parallel mechanism based on a geometric measures," in IEEE ISR 2013, Oct. 2013, pp. 1–3, doi: 10.1109/ISR.2013.6695723.

[7] "Singularities in parallel robots," Robotics, Jul. 05, 2013. https://walterfarah.wordpress.com/2013/07/05/singularities-in-parallel-robots/ (accessed Mar. 30, 2020).

[8] "The maker movement: A learning revolution — ISTE." https://www.iste.org/explore/In-the-classroom/The-maker-movement%3A-A-learning-revolution (accessed Mar. 31, 2020).

[9] "3D Printers and Industrial Manufacturing - SDP/SI." https://www.sdp-si.com/Markets/Industrial-Manufacturing.php (accessed Mar. 30, 2020).

Appendices

A Project Software

Software can be found in my GitHub Repository.

B Constant Motor Speed: Spirograph

By continuously rotating the motors at fixed speeds and spinning the workspace under the end-effector, the system becomes a complex spirograph plotter capable of producing intricate and organic looking curves.

B.1 Sample Images

Animations of a series of plots as the parameter space is explored produces are extemely satisfying to watch: example 1, example 2.

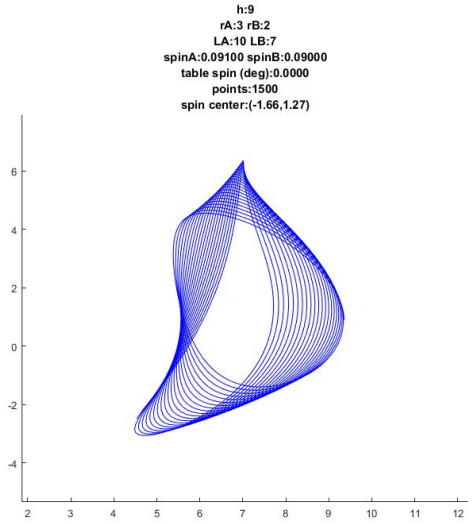

Figure 29

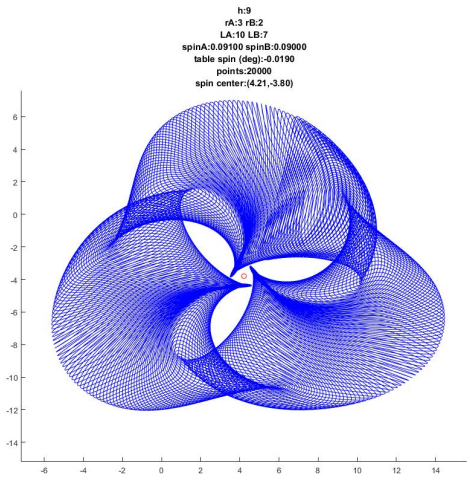

Figure 30

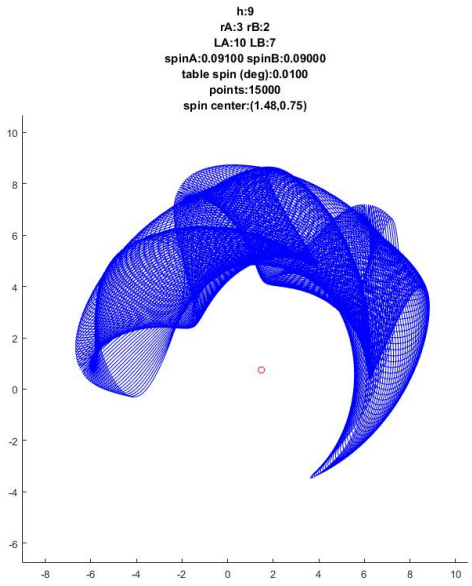

Figure 31

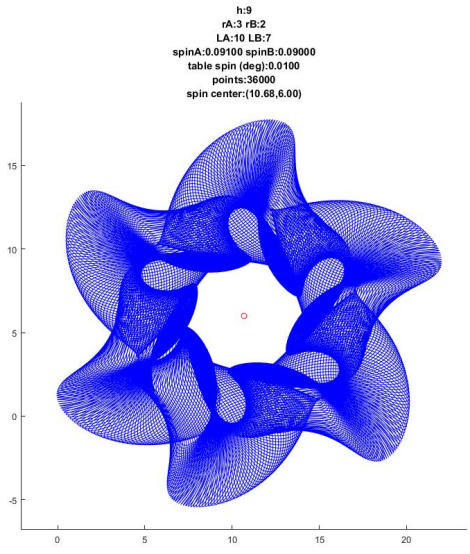

Figure 32

B.2 Periods of Rotation

These spirograph images are produced by spinning the 5BL motors at slightly different constant speeds (ω_1 and ω_2). If the motors both start at $\theta = 0$ at $t = 0$, when will they simultaneously be at $\theta = 0$ again? This interval of time, I is given by the period of the relative speed of the motors:

$$I = \frac{2\pi}{|\omega_2 - \omega_1|} \tag{27}$$

Suppose then, we want to introduce the spinning Workspace. It would be desirable to know how fast to spin the Workspace (ω_3) to produce a specified number of lobes, L, in the drawing. For example, Figure 30 has three lobes and Figure 32 has six lobes.

$$\begin{aligned} \omega_3 &= \frac{2\pi}{LI} = \frac{2\pi}{\frac{2\pi}{|\omega_2-\omega_1|}L} \\ &= \frac{|\omega_2 - \omega_1|}{L} \end{aligned} \tag{28}$$