Web-based Mixed Synthesis of Planar Four-bar Mechanisms

A Thesis presented

by

Shashank Sharma

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Master of Science

in

Mechanical Engineering

Stony Brook University

January 2018

Stony Brook University

The Graduate School

Shashank Sharma

We, the thesis committee for the above candidate for the

Master of Science degree, hereby recommend

acceptance of this thesis

Anurag Purwar - Thesis Advisor
Research Associate Professor, Mechanical Engineering Department

---

Qiaode Jeffrey Ge - Thesis Co-advisor
Professor and Chair, Mechanical Engineering Department

Nilanjan Chakraborty - Committee Chair
Assistant Professor, Mechanical Engineering Department

This thesis is accepted by the Graduate School

Charles Taber
Dean of the Graduate School

Abstract of the Thesis

Web-based Mixed Synthesis of Planar Four-bar Mechanisms

by

Shashank Sharma

Master of Science

in

Mechanical Engineering

Stony Brook University

2018

This thesis presents MotionGen 2.0, a Web-based application for Path, Motion and Mixed synthesis. MotionGen 2.0 has been developed to work across the web and also as a mobile app.

Synthesis of mechanisms has remained the holy grail for machine designers for a long time now. The inherent mathematical complexity of synthesis has prevented widespread design automation in the field. MotionGen2.0 proposes to fix this gap by introducing a user-centric design package capable of motion, path and mixed synthesis of four-bar linkages. Combined with its extensive simulation capability and feature-rich environment, it will empower professionals and students to create, invent and innovate.

Table of Contents

List of Figures

List of Tables

Acknowledgements

Firstly I would like to thank my parents, Mr. Shashi Kant and Mrs. Pammi Dixit for always being there for me through thick and thin. I am extremely grateful for the ethics and values instilled in me by them. Their unwavering trust in my abilities has motivated me to push my limits.

I am grateful to Professor Anurag Purwar, my advisor, for giving me this immense opportunity to work with him. He has always given me the freedom to shape my path and encouraged me to focus on research. His support and guidance have made this work possible.

I would also like to thank my colleagues Anshul Lodha and Shrinath Deshpande for their immense support, valuable advice and making this a fun-filled journey.

I want to express gratefulness to my uncles and aunt, Mr. Chandra Kant, Mrs. Kamini Chandra and Mr. Sunny Singh for making me feel comfortable and at home here.

I would like to express my thanks to my committee members, Professor Qiaode Jeffrey Ge and Professor Nilanjan Chakraborty, for taking their precious time in attending my presentation.

Chapter 1

Introduction

Maker culture is a new trend built upon DIY and hacker movement which encourages the creation of new devices and tinkering with the old. It spans electronics, robotics, 3-D printing, metalworking, and woodworking. The rise of the maker culture is closely associated with the rise of hackerspaces, Fab Labs and other "makerspaces" where like-minded individuals share ideas, skills, and tools. University campuses have especially been a hotbed for makerspaces. With the rise of cheap 3-D printing technologies and low-cost sensors, actuators, and micro-controllers, manufacturing tools have become accessible to many.

Thus, with the recent maker-movement and democratization of manufacturing capability, the creation of new devices is within everyone's grasp. However, there is still one missing piece. There is a total lack of machine design tools for mechanism synthesis which can be used by creators. Unfortunately, design theory for even the simplest of mechanism i.e. a four bar is too complex for the uninitiated. This chasm needs to be filled for real innovation to thrive.

The purpose of this thesis is twofold. First, a discussion of existing and new algorithms which can automate the process of path and mixed synthesis of four-bar mechanisms is carried out in detail. Use of local and global optimization routines to generate prospective the solution is explored. The focus is on getting the user a good approximate solution in short period of time.

Secondly, an actual web-based application called MotionGen2.0 is created to implement the proposed algorithm which can be used by creators. User interaction and productivity have been kept as the centerpiece. The application is scalable by design and accessible to everyone through the web and mobile app. With the availability of MotionGen2.0, professionals and students will now be able to unleash their imagination. Empowering the right people using the right tools is the motivation for this work.

Fig. 1.1 represents the areas this thesis focuses on in a pictorial format. Clarity in the exact scope of discussion is essential to understanding. It enforces that Path and Mixed synthesis of four-bar mechanisms are discussed. A brief review of Motion synthesis is also present. Function generation lies outside the scope of this thesis. In addition, synthesis of serial coupled chains has also been dealt with.

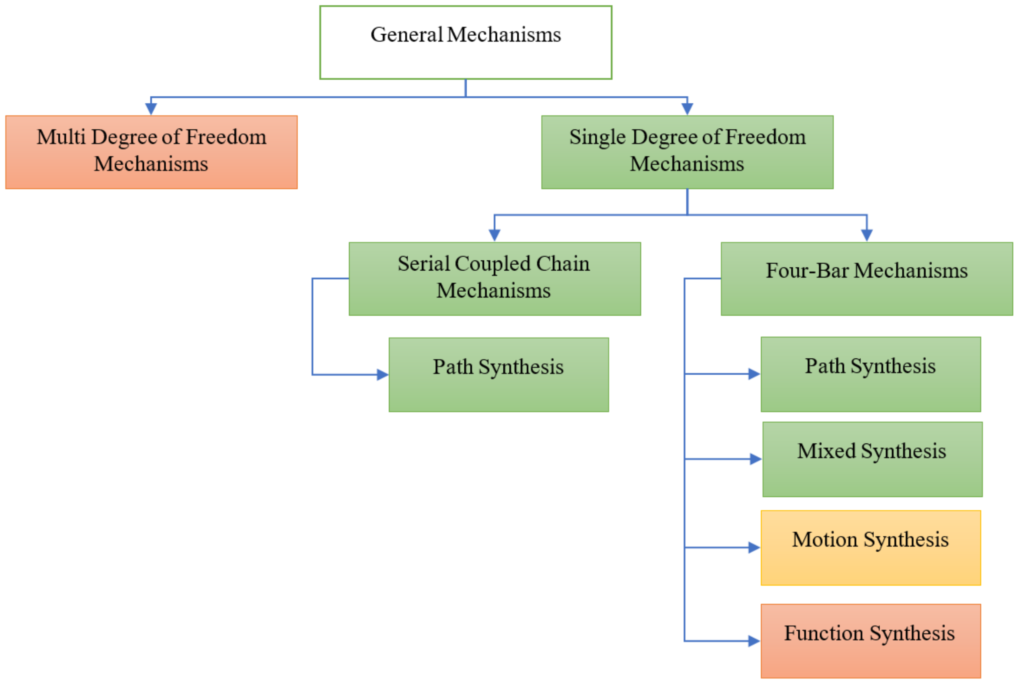


Figure 1.1: Focus areas of this thesis in Mechanism Design domain (Green- areas where new contributions made, Yellow- areas reviewed, Red- areas not covered)

Chapter 2

Path Synthesis

Mechanism synthesis problem can mostly be classified into path, motion and function synthesis problems. Each of these problems has been too complex to be solved for a generalized n-bar one d.o.f. mechanism. Thus, most of the literature deals with synthesis problem for the simplest case in the family of all one d.o.f mechanisms, the 4-bar mechanism. Different types of possible joints (revolute or prismatic) and coupler point motion (closed or open) introduces further complexities into the system. Many analytic and approximate approaches have been proposed in an attempt to solve these problems. In this chapter, the focus is on Path synthesis problem for 4-bar mechanisms.

2.1 Literature review

Path Synthesis problem can be described as calculating a mechanism whose coupler point passes through a set of user-defined path points. A sample path synthesis problem solution is shown in Fig. 2.1 where the cross-hairs represent user-inputted path points. The visible mechanism has been calculated such that its coupler point closely passes through the path points. Input path points are specified by their x and y coordinates. The mechanism is defined using a set of nine parameters i.e. $(l_1, l_2, l_3, l_4, x_0, y_0, \theta_1, r, \alpha)$ where l_1, l_2, l_3, l_4 are the link lengths, x_0, y_0 are the actuating fixed pivot coordinates, θ_1 is the fixed link orientation and r, α are coupler point parameters. These mechanism parameters have been visualized in Fig. 2.2.

Thus in a path synthesis problem, n sets of (x, y) path point coordinates are used to calculate nine design parameters $(l_1, l_2, l_3, l_4, x_0, y_0, \theta_1, r, \alpha)$ representing a four bar mechanism. The input path points and synthesized mechanism parameters for Fig. 2.1 are shown in Table 2.1 and Table 2.2.

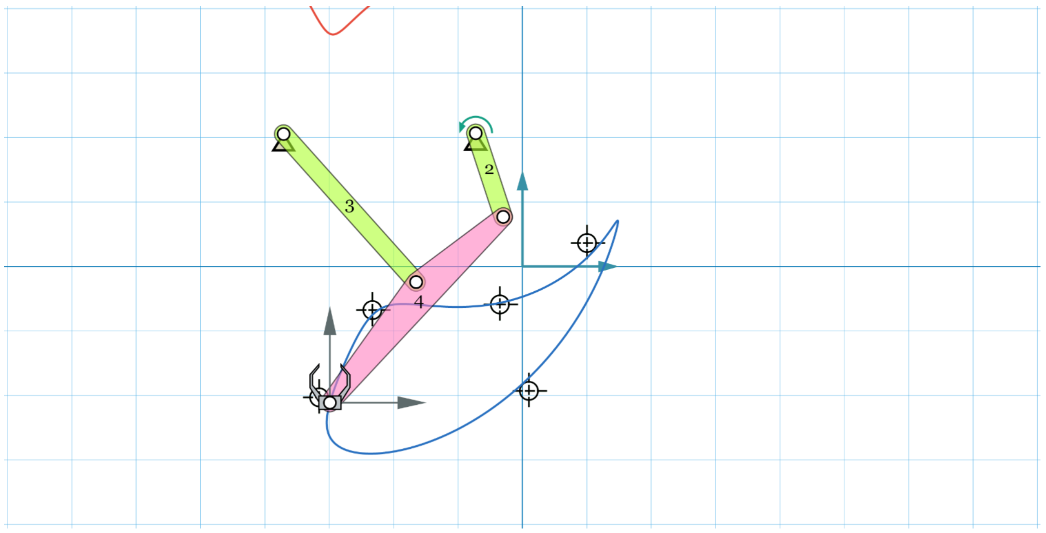


Figure 2.1: An example of Path Synthesis problem

Table 2.1: Input Path Points for Path Synthesis problem shown in Fig. 2.1

X	Y
−3.158	−2.030
−2.331	−0.677
−0.351	−0.589
1.003	0.363
0.100	−1.930

Table 2.2: Synthesized mechanism parameters for Path Synthesis problem shown in Fig. 2.1

l_1	l_2	l_3	l_4	x_0	y_0	θ_1	r	α
2.984	1.367	1.689	3.084	−0.726	2.067	−3.137	3.944	0.178

Path synthesis can be solved analytically for 4-bar mechanisms and gives exact solutions. Analytical methods include algebraic methods [1–3], complex number methods [4] or displacement matrix methods [5]. However, it

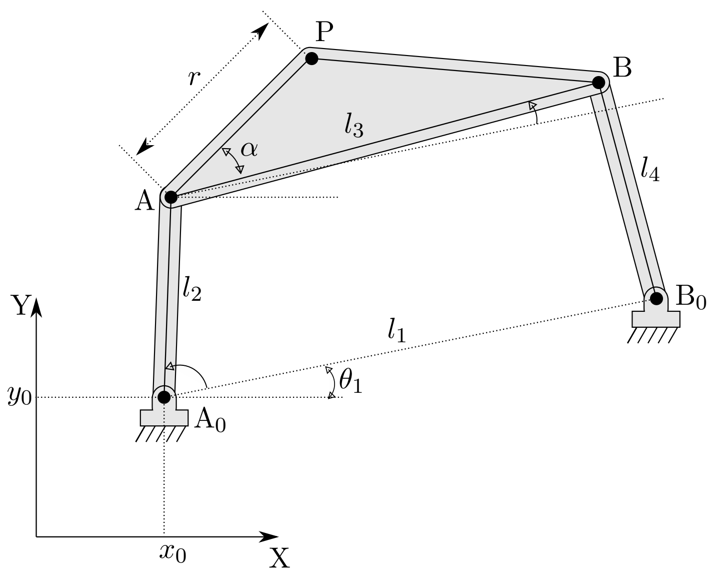


Figure 2.2: Visualization of parameters describing a four-bar mechanism

can handle only up to 9 path points above which the system of equations becomes over-constrained. Also, these solutions suffer from branch and circuit defects which make the practical use of results questionable. Even the exact results up to 9 path points are extremely complex and can rarely be calculated. To counter these drawbacks, path synthesis problem can be solved using an approximation. These methods can handle any number of path points. However, compromise on accuracy is made in favor of approximate solutions.

Most of the approximation techniques restructure the synthesis problem into an optimization problem [6–16]. These techniques are characterized by three major entities- domain space variables, objective function and optimization algorithm. Domain space variables depend on our reformulation of problem. Objective function is the measure of how good a prospective mechanism is in fulfilling our target path requirements using domain space variables. Optimization algorithm can be broadly classified as global/local or deterministic/stochastic. There is no best algorithm and use of each depends on problem. A fast, accurate and easy to code/formulate optimization is desirable.

In practical problems, the designer is interested in the general shape of coupler path rather than a small number of precision points on coupler path.

Thus, path synthesis can be formulated as a problem in the area of Shape Analysis [13,16–19] by interpolating a curve using path points. By converting point fitting problem to curve matching problem, possible output mechanisms have been constrained to practical 'smooth' choices and usually rejects huge displacements in between precision points. To compare two curves, a metric need to be established which tends to zero for similar curves. Various metrics like distance measures, Fourier descriptors [20], wavelets [21], cumulative angular deviant [13], etc. have been used over the years to compare two curves. Frechet distance, Dynamic time warping or basic sample euclidean distance are the predominant distance measures which have been used for curve matching using point sampling [22].

Standardization of curve data using Procrustes analysis can be done for better shape matching [23] which involves optimal translation, scaling and rotation of curves for matching, determined using a variety of mathematical formulations. Standardization also helps isolate variables in synthesis problems which are dependent on location, orientation and scale, hence reducing domain space variables in our optimization function. All these operations are valid for synthesis problems because the domain space variables are invariant under them.

The coming sections focus on solving the path synthesis problem using two methods, namely, Fourier based path approximation and Frenet frame based path approximation. Original contribution in Fourier based path approximation involves point sampling re-parameterization to generate better task curves. Frenet frame based path approximation involves formulating the path synthesis problem into motion synthesis problem as its easier to solve using the technique of kinematic mapping and singular value decomposition. Technical details on both the methods have been elaborated in the following sections.

2.2 Fourier based Path Approximation

2.2.1 Path Synthesis Algorithm Review

Fourier based path synthesis fits the four-bar Coupler path to a computed Task path. Task path is a curve interpolating or approximating input path points, described using Fourier coefficients. Coupler path is a curve which is traced by coupler point for some four-bar mechanism. Task path and four-bar Coupler path for the Fourier based path synthesis have been visualized in the Fig. 2.3. Their Fourier descriptors, which will be discussed soon, are given in Table 2.3.

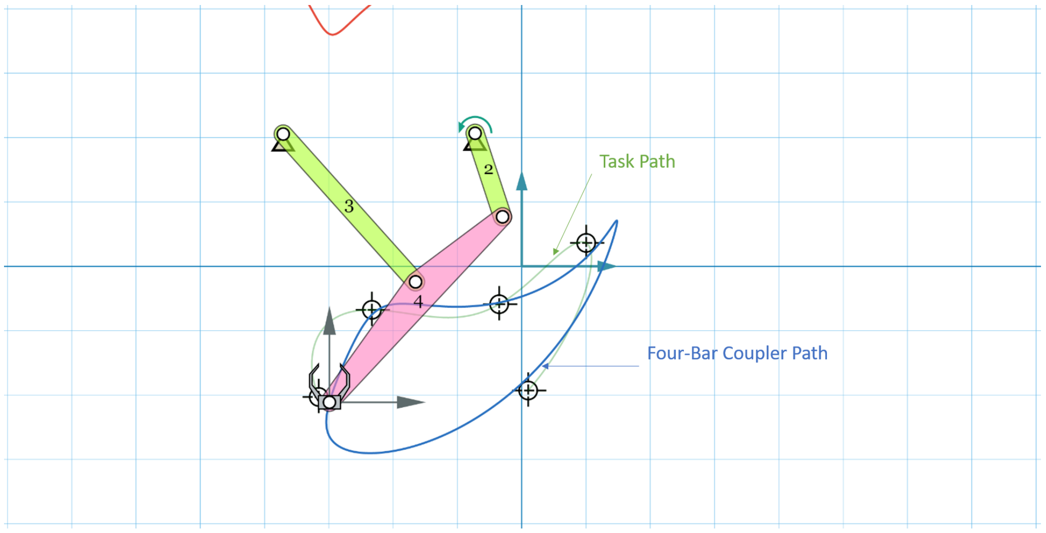


Figure 2.3: Visualization of Task path and Coupler coupler for Fourier based path synthesis algorithm

Table 2.3: Task path and Coupler path Descriptor Data shown in Fig. 2.3

	Task path		Coupler path	
Descriptor	Complex Value	Magnitude	Complex Value	Magnitude
p=−2	$-0.4779 + 0.1294i$	0.4951	$-0.4214 + 0.222i$	0.4763
p=−1	$-1.213 - 1.033i$	1.5935	$-1.164 - 1.062i$	1.5757
p=0	$-0.9959 - 1.089i$	1.4758	$-0.9959 - 1.089i$	1.4758
p=1	$-0.6851 - 0.09537i$	0.6917	$-0.6851 - 0.09537i$	0.6917
p=2	$0.2141 + 0.05825i$	0.2219	$0.2752 - 0.08812i$	0.2889

The Fourier based path synthesis algorithm is a two step process. First, an optimum Task Path using the user-inputted path points is calculated. The subsequent motivation is to find a mechanism whose Coupler path closely approximates the Task path. Detailed methodology has been discussed in sections below. Work done by [24–26] on path synthesis of a four bar mechanism using this approach is summarized in this section.

2.2.1.1 Calculation of Task Path

The initial objective is to convert inputted points into a task path. Path points inputted by the user can vary practically from one to infinity. For one to three points, the solution is a trivial single link whose rotation form a circle on which up to three points fall. Thus, no mechanism needs to be generated in these cases. This algorithm handles greater than 4 path points and processes them to give a task path curve.

Any closed curve can be described mathematically using periodic functions. Fourier basis prove to be exceptionally suited to the task since they have the ability to represent all the possible periodic functions. Let a path be defined by $z(t) = x(t) + iy(t)$ where x and y are the coordinates of a point on the curve and i is the imaginary unit, then

$$z(t) = \sum_{m=-\infty}^{\infty} a_m e^{2\pi m i t} \qquad \forall \quad t \in [0, 1) \tag{2.1}$$

where m are the frequencies, a_m are the Fourier coefficients and t is the time parameter periodic over $[0, 1)$. The $\|m\|^{th}$ frequency is called as Harmonics. For example, first harmonic involves the curves described by frequencies $m = 1, -1$. The above Fourier curve can also be described for discrete data using uniform parametrization as follows

$$z\left(\frac{2\pi k}{N}\right) = \sum_{m=-\infty}^{\infty} a_m e^{mi\left(\frac{2\pi k}{N}\right)} \qquad \forall \quad k \in [0, N-1] \tag{2.2}$$

where k is the sample point index and N is the total number of sample points. Thus, a series of points can easily be used to recover the curve they represent using Discrete Fourier Transform (DFT). The representation of curve is dependent on Fourier coefficients which can be calculated by

$$a_m = \frac{1}{N} \sum_{k=0}^{N-1} z_k e^{-mi\left(\frac{2\pi k}{N}\right)} \tag{2.3}$$

For nonuniform time sampling, a slightly different formulation for Fourier coefficients is calculated using trapezium rule.

$$a_m = \frac{1}{2}[z_{i+1} + z_i] e^{-2\pi m i t} \tag{2.4}$$

The task curve is defined using such a Fourier basis i.e. is a collection of sine and cosine functions of varying frequencies. The total number of these functions are fixed for a good enough task curve estimation within acceptable computation time. Consequently, maximum number of harmonics used in literature has been 7 harmonics or 15 Fourier coefficients. A limited number of these functions are sufficient to describe a simple path approximating the path points. With a large number of harmonics, we run into the problem of over-fitting the sample data. Also, considering a high number of harmonics when the input points are low leads to many free harmonics which gives the curve unnecessary finer detail not specified by the user.

There is infinite possibilities of drawing a curve between the inputted path points. The optimum task curve is calculated to exactly fit or best approximate the inputted path points. The optimizing metric used is the usual P2 norm of point-to-point euclidean distance difference.

However, there lies a problem in the Fourier curve description. The coupler curves to four bar mechanisms can be open or closed paths depending on the permissible rotation of the driven link, while the Fourier curves only work with closed paths. To counter this inherent limitation of Fourier curves, another similar construct called trigonometric polynomial curve is used which describes the Task path as

$$z(t) = \sum_{m=-p}^{p} \alpha_m e^{\omega_o m i t} \qquad \forall \quad t \in [0, T], T < 1 \tag{2.5}$$

where T is the interval over which the curve is defined.

Since DFT is not applicable, the approximation of such a curve uses least square fitting approach. As a result, the coefficients received are not Fourier coefficients unless the fitting is applied to a closed path. For the sake of convenience, they have been termed as Fourier descriptors. The least square problem can be formulated as

$$\Delta = \sum_{i=1}^{n} \left\| z(t_i) - \sum_{k=-p}^{p} \alpha_k e^{ik\omega_o t_i} \right\|^2 \tag{2.6}$$

Analytically solving the least square problem gives a linear system of equation. This can be solved to find the descriptors which best fits the input path points. This system of equation can be defined as follows

$$\Omega \mathbb{X} = \mathbb{Y} \tag{2.7}$$

where

$$\mathbb{X} = [\ldots, \underset{m\rightarrow}{\alpha_m}, \ldots]^T \tag{2.8}$$

$$\Omega = \begin{array}{c} \left[\begin{array}{ccc} & \cdots & \\ \vdots & \sum_{i=0}^{n} e^{i(k-m)\theta_i} & \vdots \\ & \cdots & \end{array}\right] m \downarrow \\ k\rightarrow \end{array} \tag{2.9}$$

$$\mathbb{Y} = [\ldots, \underset{m\rightarrow}{\sum_{i=0}^{n} z(t_i)e^{-im\theta_i}}, \ldots]^T \tag{2.10}$$

Here, k and m vary from -p to p. LU decomposition can be used to solve the above system.

Further details for deriving the above relations are available in [24].

The domain for the open task path representing inputted path points remains unknown and can range anywhere from $(0, 2\pi]$. Finding this domain (T) is a one-dimensional optimization problem for minimum error measure defined in Eqn. 2.6. Once the parameter distribution and total domain of open curve are calculated, the target curve is fully defined.

2.2.1.2 Calculation of Four-bar Coupler Path

In this section, equation of coupler point for a four-bar mechanism is approximated using its loop closure equation. The mechanism is represented by the design parameters x_0, y_0, l_1, l_2, l_3, l_4, r, θ_1, α, ϕ_0 as displayed in Fig. 2.2. Finding the relation between coupler angle (λ) and input crank angle (ϕ) leads to the expression

$$e^{i\lambda} = \frac{-B(\phi) \pm \sqrt{\Delta_1(\phi)\Delta_2(\phi)}}{2A(\phi)} \tag{2.11}$$

where

$$A(\phi) = l_3(l_2e^{-i\phi}) - l_1 \tag{2.12}$$

$$B(\phi) = l_1^2 + l_2^2 + l_3^2 - l_4^2 - 2l_1l_2\cos(\phi) \tag{2.13}$$

$$\Delta_1(\phi) = l_1^2 + l_2^2 - (l_3 + l_4)^2 - 2l_1l_2\cos(\phi) \tag{2.14}$$

$$\Delta_2(\phi) = l_1^2 + l_2^2 - (l_3 - l_4)^2 - 2l_1l_2\cos(\phi) \tag{2.15}$$

The $\pm$ sign in the equation represents the two different configurations of a four bar mechanism. It must be noted that this function is periodic as it consists of only trigonometric functions.

However, the Eqn. 2.11 only gives meaningful results when the following feasibility condition is followed

$$\Delta_1(\phi)\Delta_2(\phi) \leq 0 \tag{2.16}$$

Thus, this Eqn. 2.16 constraints the interval over which angle ϕ exist. The maximum it can range is $[0,2\pi]$ in the case it represents a closed curve.

Eqn. 2.11, being a periodic function when ϕ=$[0,2\pi]$, can be rewritten using Fourier basis as

$$e^{i\lambda} = \sum_{k=-\infty}^{\infty} C_k e^{ik\phi} \tag{2.17}$$

Only a small group of C_k associated with low harmonics is considered to approximate $e^{i\lambda}$, just like in approximation of Task path. When ϕ varies only in some part of $[0,2\pi]$, the least square curve fitting method is used to calculate coefficients. The coefficients C_k are calculated numerically by sampling points on the function according to the previous parametrization, evaluating the function on each point based on crank angle and length ratios and then least square curve fitting the results to get an approximate solution. Finding an analytic closed form solution for each coefficient C_k is not possible.

The analytic equation of coupler point P can be given as

$$P = A_0 + l_2 e^{i\theta_1} e^{i\phi} + r e^{i\alpha} e^{i\theta_1} e^{i\lambda} \tag{2.18}$$

$$A_0 = x_0 + iy_0 \tag{2.19}$$

$$\phi = \phi_0 + \omega t \tag{2.20}$$

Using the above equations, we obtain

$$P = \sum_{k=-\infty}^{\infty} P_k e^{ik\omega t} \tag{2.21}$$

where

$$P_0 = r e^{i\alpha} e^{i\theta_1} C_0 + A_0 \tag{2.22}$$

$$P_1 = r e^{i\alpha} e^{i\theta_1} C_1 e^{i\phi_0} + l_2 e^{i(\theta_1+\phi_0)} \tag{2.23}$$

$$P_k = re^{i\alpha}e^{i\theta_1}C_k e^{ik\phi_0}|_{k\neq 0,1} \tag{2.24}$$

Thus, an analytic expression has been calculated to describe the coupler curve using the mechanism parameters x_0, y_0, l_1, l_2, l_3, l_4, r, θ_1, α, ϕ_0. The objective is to fit this equation to Task curve equation.

2.2.1.3 Fitting Coupler path to Task path

To summarize the above sections, Fourier descriptors T_k have been calculated using least square curve fitting and the task curve T is defined as

$$T \approx \sum_{k=-p}^{p} T_k e^{ik\Phi} \tag{2.25}$$

In the equation, Φ lies in range [0,Φ_{max}] where Φ_{max} can lie anywhere between (0,2π].

Also, the coupler path P is approximately represented using Fourier descriptors P_k as

$$P \approx \sum_{k=-p}^{p} P_k e^{ik\omega t} \tag{2.26}$$

where ωt also lies in range [0,Φ_{max}]. Since $\phi = \phi_0 + \omega t$, range of ϕ i.e. input angle for mechanism is [ϕ_0, ϕ_0+Φ_{max}]. It should be noted that the exact value of input angle to reach each coupler point depends on the parametrization of samples i.e. Φ_i. Each Fourier descriptors P_k has a relation with nine parameters defining the mechanism.

For a perfect curve matching, $T_k = P_k$ for all $k \in [-p, p]$. This leads to

$$T_0 = C_0 re^{i\alpha}e^{i\theta_1} + A_0 \tag{2.27}$$

$$T_1 = C_1 re^{i\alpha}e^{i\theta_1}e^{i\phi_0} + l_2 e^{i(\theta_1+\phi_0)} \tag{2.28}$$

$$T_k = C_k re^{i\alpha}e^{i\theta_1}e^{ik\phi_0}|_{k\neq 0,1} \tag{2.29}$$

Thus, the problem of path synthesis involves finding ten optimum design parameters for a four-bar mechanism, namely

$$S = \left\{ l_2, \frac{l_2}{l_1}, \frac{l_3}{l_1}, \frac{l_4}{l_1}, x_0, y_0, \theta_1, \phi_0, r, \alpha \right\} \tag{2.30}$$

The design parameters can be modified such that $T_k|_{k\neq 0,1}$ depends on six variables instead of seven. Thus the new design variables are

$$S = \left\{ l_2, \frac{l_2}{l_1}, \frac{l_3}{l_1}, \frac{l_4}{l_1}, x_0, y_0, \theta_1, \phi_0, \mathbb{C}, \mathbb{S} \right\} \tag{2.31}$$

where

$$\mathbb{C} = r\cos\alpha + \theta_1, \quad \mathbb{S} = r\sin\alpha + \theta_1 \tag{2.32}$$

It is observed that four of the design variables i.e. $\{l_2, x_0, y_0, \theta_1\}$ only exist in the expressions for T_0 and T_1. As a result, a tenth dimensional search space has been reduced into a sixth dimensional search space. The four remaining variables can be fitted exactly using the complex equations for fitting T_0 and T_1. The objective now is to search for optimal $\left\{\frac{l_2}{l_1}, \frac{l_3}{l_1}, \frac{l_4}{l_1}, \phi_0, \mathbb{C}, \mathbb{S}\right\}$.

To find a least square solution to Eqn. 2.29, the minimum for $\mathbb{C}$ and $\mathbb{S}$ is calculated by minimizing the following error function

$$I = \sum_{k\neq 0,1} |C_k r e^{i(\alpha+\theta_1+k\phi_0)} - T_k|^2 \tag{2.33}$$

$$= \sum_{k\neq 0,1} [(A_k \cdot \mathbb{C} - B_k \cdot \mathbb{S} - T_k^x)^2 + (A_k \cdot \mathbb{S} + B_k \cdot \mathbb{C} - T_k^y)^2] \tag{2.34}$$

where $T_k = T_k^x + iT_k^y$ and $C_k e^{ik\phi_0} = A_k + iB_k$. Setting partial differentials to zero and analytically minimizing the error function leads to

$$\mathbb{C} + i\mathbb{S} = \frac{\sum_{k\neq 0,1} T_k C_k^* e^{-ik\phi_0}}{\sum_{k\neq 0,1} |C_k|^2} \tag{2.35}$$

As a result, the least square solution to Eqn. 2.29 can be found by optimization of four design variables namely $\left\{\frac{l_2}{l_1}, \frac{l_3}{l_1}, \frac{l_4}{l_1}, \phi_0\right\}$. The optimization space has thus been reduced to a mere four design variables instead of original ten design variables.

During the discussion, the constraint imposed by the feasibility condition has been ignored. Thus, to generate valid mechanisms, an extra relation needs to be satisfied. Rewriting the feasibility condition in terms of four design variables, we get

$$\left(\frac{l_3}{l_1} - \frac{l_4}{l_1}\right)^2 \leq 1 + \left(\frac{l_2}{l_1}\right)^2 - 2\left(\frac{l_2}{l_1}\right)\cos\phi \leq \left(\frac{l_3}{l_1} + \frac{l_4}{l_1}\right)^2 \tag{2.36}$$

All ϕ in range $[\phi_0,\ \phi_0 + \Phi_{max}]$ must satisfy the inequality for the mechanism to be meaningful. Direct search method has been used as the optimization method in work by Wu et al. [24]. This completes the review of Fourier based path synthesis algorithm.

2.2.1.4 Implementation summary

```
Input: $n$ Path points
Step 1- Find task path Fourier description($T_k$);
    No. of harmonics, domain limits and parameter distribution
  required;
Step 2- Find optimal $S_1 = \{l_{21}, l_{31}, l_{41}, \phi_o, \mathbb{C}, \mathbb{S}\}$;
    Search space is $\{l_{21}, l_{31}, l_{41}, \phi_o\}$;
    Adding $\phi_0$ to task parameter gives mechanism parameter;
    Check the feasibility condition at all $\phi$ in domain;
    Find values of $e^{i\lambda}$ at parameters analytically;
    Find Fourier descriptors($C_k$) of coupler angle $\lambda$ numerically;
    Find Fourier descriptors($P_k$) of coupler point;
    Find $\{\mathbb{C}, \mathbb{S}\}$ such that $|P_k - T_k|$ is minimized;
Step 3- Find $S_2 = l_2, x_0, y_0, \theta_1$ using $T_0, T_1$;
Output: Optimum mechanism
```

Algorithm 1: Pseudo-code for Fourier based path synthesis

This section summarizes the above technical details and expresses it in Pseudo-Code structure in Algo. 1 for enhanced detail.

2.2.2 Search for a better parametrization

The standardized implementations of DFT assume a uniformly sampled data i.e. the time parameter attached to each parameter is uniform. As a consequence, the Fourier curve fitted carries over time information to the path. In a purely geometric sense, this behavior is undesired. Although the problem has been mentioned in [26] no work has been done to find a parametrization independent representation until very recently in [27] where an approach to find an optimum parametrization has been proposed to eliminate time dependency. The approach tries to use arc length parametrization to capture the geometric features and eliminate the time parameter. However, its impossible to arc-length parametrize an unknown coupler curve which is the path synthesis problem. Thus, the author numerically "guess" the arc length parametrization using the Task point polygon and thus fail to get a time-independent parametrization.

However, time parameter is not necessarily an evil entity which needs to be eliminated. The time dependence of Fourier descriptors can be exploited to give mechanism designers increased flexibility. The time parameter decides when the synthesized mechanism moves over each point. By changing the time distribution, finer control on motion of coupler point can be established.

For example, uniform parametrization results in a target curve where the end effector takes equal time to pass through each target point. Thus, the uniform parameter is desired in cases where the relative time between each target point is required to be same.

Similarly, for cases where the constant speed of end effector over the target path is required, an arc length parametrization would be desired. Analytically, it is not possible to get the arc length parametrization before the synthesis process but a good approximation is the chord length parametrization.

Thus, uniform and chord length parametrization are two important parameterizations which have substantial design use-case. In next section, a generalized parametrization incorporating both of these cases is presented.

2.2.2.1 Family of parameterizations

In interpolation path finding problems, Chord length based parameterizations have been used by geometers for a long time. In the case of path synthesis, formulation in Eqn. 2.6 to define open task curve is valid for any parametrization, uniform or nonuniform, over a specified domain $[0, \theta]$. This resultant curve interpolating the input path points can be calculated using a generalized parametrization called centripetal parametrization. It can be represented as

$$t_0 = 0 \tag{2.37}$$

$$t_k = \frac{\sum_{i=1}^{k} |D_i - D_{i-1}|^\alpha}{L} \tag{2.38}$$

$$t_n = 1 \tag{2.39}$$

where t_k represents the parameter for each input path point, D_i represents the cumulative chord length till i^{th} point and L represents the total cumulative chord length till last point. α is the parametrization control variable such that $\alpha \in \mathbb{R}$ and $\alpha \geq 0$. On varying the control variable α, multiple parametrization can be generated. It must be noted that uniform and chord length parametrization are special cares at $\alpha = 0$ and $\alpha = 1$. It is interesting to see the effect these parameterizations have on the task curve.

A sample case, for data listed in Table 2.4, has been shown in Fig. 2.4, 2.5, 2.6 where least square fitted trigonometric curve is generated for five sample input points using different parametrization techniques. The wiggle in the first two cases is very apparent in the first two curve. The third curve is relatively better than the other two. This begs the question, for which value of α does the best parametrization exists.

Table 2.4: Input Path Point Data to analyze effect on Task path under different parameterizations

X	Y
-6.065	1.541
-4.298	2.218
-2.983	2.519
-0.589	2.444
4.436	-1.779

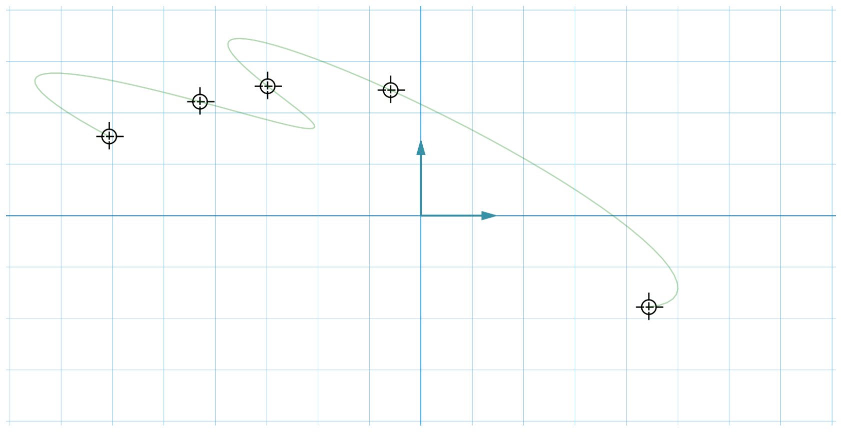


Figure 2.4: Task curve calculated using Uniform Parametrization ($\alpha = 0$)

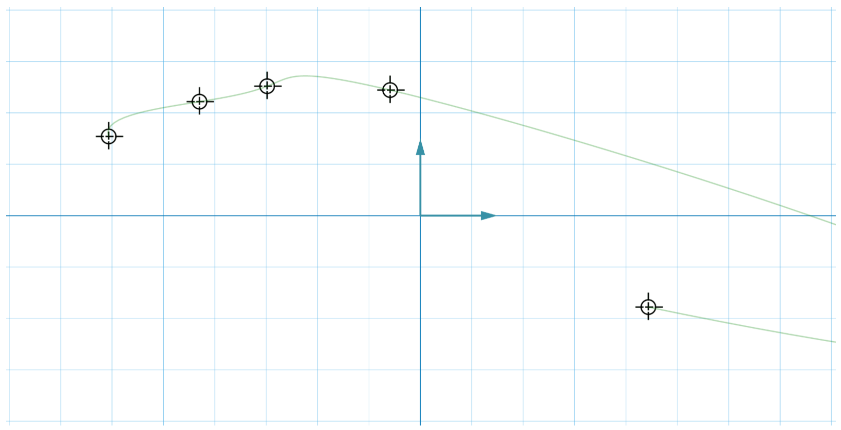


Figure 2.5: Task curve calculated using Chordal Parametrization ($\alpha = 1$)

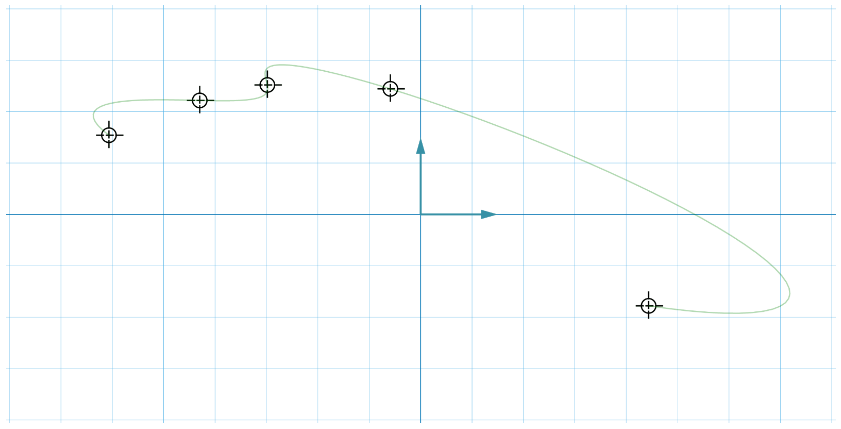


Figure 2.6: Task curve calculated using Centripetal Parametrization ($\alpha = .5$)

Thus, the problem now is to select the best parametrization from the family. Objective it to find specific value of α such that wiggle in the curve is minimized. This can be done by minimizing the total arc length of the curve. Analytically, arc length (L_a) is defined as

$$L_a = \int \sqrt{1 + \left(\frac{dz}{dt}\right)^2} dt \tag{2.40}$$

where z is the task curve and t is the time parameter. Closed form solution for the above integration is not possible.

Fortunately, L_a can also easily be computed numerically by dividing the curve into many intervals and approximate these lengths using a line segment. When the curve is divided into a large number of intervals, a close approximation can be calculated.

```
Input: Initial Parametrization ($\alpha_0$)
while $Residue > \epsilon$ do
    Move to better $\alpha_i$;
    Find new Task curve $z_i = z(\alpha_i)$;
    Calculate Arc length $L_{a,i} = L_a(z_i)$;
    Calculate Residue=$\|L_{a,i} - L_{a,i-1}\|$;
end
Output: Minimum parametr1ization ($\alpha$)
```

Algorithm 2: Pseudo-code for calculating the optimum parametrization

Thus, finding the best parametrization is a one-dimensional minimization problem with objective function equal to arc length. The algorithm can be summarized in Algo. 2.

Fig. 2.7 displays the Task curve for optimal parameterization with minimum arc length is at $\alpha = .4875$ for the sample test-case. Fourier descriptors for each curve are described in Table 2.5.

2.2.2.2 Design-centric Constraints

Design problems come in many flavors. Usually, large variation in end effector speeds is not the designers intent. The larger the changes in speed, the larger are the forces induced on links which make Dynamic analysis indispensable

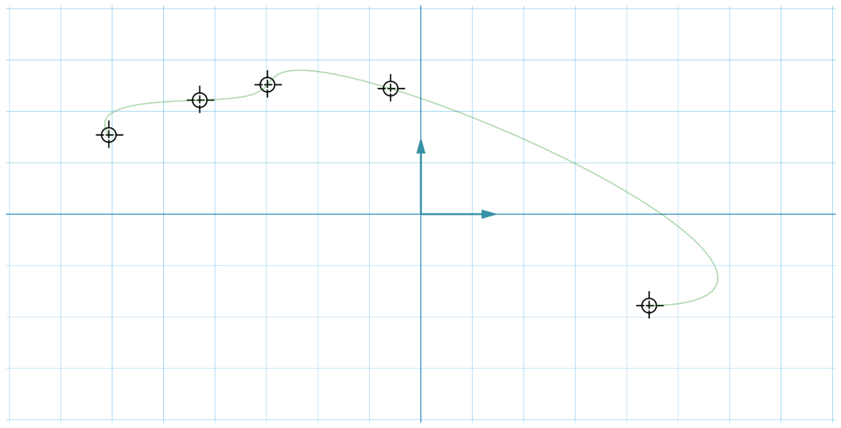


Figure 2.7: Task curve calculated using Optimum Parametrization ($\alpha = .4875$)

Table 2.5: Task path Descriptor Data for Parameterizations in Fig. 2.4, 2.5, 2.6, 2.7

	Task path $\alpha = 0$		Task path $\alpha = 1$	
Descriptor	Complex Value	Magnitude	Complex Value	Magnitude
p=−2	$-1.953 - 0.08941i$	1.9552	$-0.4244 + 1.532i$	1.5893
p=−1	$-1.355 - 2.423i$	2.7763	$-4.508 - 2.595i$	5.2015
p=0	$-1.91 + 1.404i$	2.3700	$1.727 + 0.4681i$	1.7890
p=1	$0.1551 + 1.477i$	1.4847	$-1.892 + 3.221i$	3.7360
p=2	$-1.002 + 1.174i$	1.5433	$-0.9668 - 1.085i$	1.4533

	Task path $\alpha = .5$		Task path $\alpha = .4875$	
Descriptor	Complex Value	Magnitude	Complex Value	Magnitude
p=−2	$-1.159 + 0.9341i$	1.4888	$-0.7192 + 0.9789i$	1.2147
p=−1	$-2.539 - 2.55i$	3.5983	$-2.759 - 1.96i$	3.3841
p=0	$-0.7293 + 1.054i$	1.2816	$-1.033 + 1.025i$	1.4555
p=1	$-0.4324 + 2.173i$	2.2154	$-0.6791 + 1.752i$	1.8792
p=2	$-1.205 - 0.06939i$	1.2075	$-0.8745 - 0.2552i$	0.9110

in these cases. These large forces can even compromise the rigidity of links making the kinematic analysis useless.

However, uniform speed over the complete motion of end effector is not the ideal solution to this problem. Some degree of speed change is important in mechanisms like quick return mechanism. Thus, the possibility of specifying speed restrictions would greatly benefit the designer.

The ratio of coupler point's maximum speed to minimum speed has been used as control variable denoted as S_r. S_r is always greater than one. The user's main interest while designing is to reject mechanisms which do not satisfy S_r for the design problem. For example, the user might want relatively uniform speed mechanisms with $V_r < 2$. This helps the user narrow down on the particular parametrization which has less wiggle and satisfies speed criteria.

Rewriting the formulation for target path Eqn. [2.5] here for reference,

$$z(t) = \sum_{m=-p}^{p} \alpha_m e^{\omega_o m i t} \qquad \forall \quad t \in [0, T], T < 1. \tag{2.41}$$

Differentiating the above path curve gives the velocity function as

$$v(t) = \sum_{m=-p}^{p} m\beta_m e^{\omega_o m i t} \tag{2.42}$$

where $\beta_m = i\omega_o\alpha_m$. To analytically find the maximum and minimum velocities, roots of derivative of Eqn. [2.42] are needed. Then, values of their second derivative at the roots will determine if its a maximum or minimum. Unfortunately, root finding problem for any function is an optimization problem in itself.

To avoid the use of optimization, max and min speeds can be found out by sampling the curve and directly finding the values of velocity using Eqn. [2.42]. The magnitude of this complex velocity gives us speed. By taking a large number of samples, a good approximation of S_r can be achieved.

The designer specifies a lower speed ratio bound ($S_{r,min}$) and upper speed ratio bound ($S_{r,max}$). Mechanisms of interest lie exclusively in this region. If no limitations are enforced by the user, then taking $S_{r,min} = 1$ and $S_{r,max} = \infty$ includes all the possible parameterizations.

Penalty functions have been used to convert a constrained optimization problem to an unconstrained one. Literature describing different types of internal and external penalties is rich. The quadratic penalty function is used to include these constraints within the objective function to find the relevant mechanisms. The inequality constraints are

$$S_{r,min} \leq S_r(z) \leq S_{r,max} \tag{2.43}$$

which can be rewritten as following inequalities.

$$g_1(z) : S_r(z) - S_{r,min} \geq 0 \tag{2.44}$$

$$g_2(z) : S_{r,max} - S_r(z) \geq 0 \tag{2.45}$$

Thus, the final objective function to minimize becomes

$$f(z) = L_a(z) + P(max(0, -g_1(z)) + max(0, -g_2(z)))^2 \tag{2.46}$$

where z is task curve, L_a is arc length, g_1, g_2 are constraints and P is the penalty. Tweaking P can lead to good results numerically.

Thus, a method to find a target Fourier curve which has minimum wiggle and is constrained by min- max speed has been outlined above. Using this methodology, the influence of time parametrization on the path is effectively minimized. If the design requirement is of uniform time or speed parametrization, α can directly be set to 0 or 1.

2.2.3 Optimization Algorithms

To implement path synthesis, selection of optimization algorithm [28] is a pivotal step. A fast, accurate and simple optimization is desirable. Unfortunately, no such ideal algorithm exists and trade-offs need to be made. Speed is the most important attribute in practical scenarios. A fast method which gives approximate solution is desirable to a slow and exact method. Thus, although a global minimum is desirable, local minima can be good prospective solutions.

2.2.3.1 Direct Search Method

One of the easiest local optimization methods available is the direct search method. It requires no differentiability condition on the defined error function. A random point in search space can be taken as the starting point. The algorithm then searches in each coordinate direction one after another to find the minimum error. This is repeatedly done until the algorithm converges to a minimum. Although this algorithm is very simple to understand and implement, the convergence is slow and it reevaluates the objective function a large number of times.

2.2.3.2 Nelder-Mead Method

Another local optimization method called Nelder-Mead or downhill-simplex method also solves multidimensional unconstrained optimization without derivatives. What makes this particular algorithm attractive is that it carries out the error function evaluation less number of times and tries to reuse information from previous iterations. Also, it converges at a faster rate as it is not constrained in any specific direction. Thus, Nelder-Mead optimization is a better method even though it's more complex and harder to implement.

Both direct search and Nelder-Mead methods find local minima which are not the best possible solution. Running these methods with different start points in optimization space will give us different minima. The best of these discovered local minima is usually a pretty good practical solution.

2.2.3.3 Monte-Carlo Method

To find the global minima, a global optimization method like Direct Monte-Carlo sampling can be used. This is a brute force method in which, random points from optimization space sampled and the error function is evaluated. The more points are sampled, the better the probability of discovering the minima. However, it's impractical as it requires too much time to converge. This simple method can be used in combination with local optimization method to increase the convergence rate.

2.2.3.4 Simulated Annealing

Another global optimization method is simulated annealing. A temperature variable varies from high to low over the process of simulation. The higher the temperature, the higher is the probability of selecting a worse solution in the neighborhood. The pseudo-code has been summarized in Algo. 3.

As is apparent, implementation of simulated annealing requires setting variables like initial temperature, cooling rate, cooling steps, max steps per temperature change, the definition of neighborhood and acceptance function. Specification of each need to be discussed for the syntheses problem.

```
Input: Some solution ($x_0$)
Set current solution ($x \leftarrow x_0$);
Set current temperature ($t \leftarrow t_0$);
for Cooling steps ($i \leftarrow 1 : Step_c$) do
    Calculate current error ($E \leftarrow Cost(x)$);
    for Temperature change steps ($j \leftarrow 1 : Step_t$) do
        Find random solution in neighborhood
         ($x_{new} \leftarrow Neighborhood(x)$);
        Calculate error for new solution ($E_{new} \leftarrow Cost(x_{new})$);
        if Acceptance criteria fullfilled
         ($Prob_{move}(E, E_{new}) \geq random(0,1)$) then
            Move to new solution ($x \leftarrow x_{new}$);
        else
            break;
        end
    end
    Lower temperature ($t \leftarrow t \times$ cooling rate) ;
end
Output: Minimum solution ($x$)
```

Algorithm 3: Pseudo-code for Simulated Annealing

The cooling schedule of simulated annealing is fully specified by initial temperature, cooling rate, and cooling steps. The effective cooling schedule is essential to reduce the amount of time required to find the optimum solution. The initial value of temperature must be high enough so that any new solution generated in the neighborhood should be accepted with a certain probability close to 1. This ensures free movement infeasible solution space and that the final solution is independent of the start point. The cooling rate is usually taken as an exponential function of multiplicative monotonic form $t_k = t_0 \times C$ where C is the cooling rate, t_0 is the initial temperature and t_k is the temperature at k^{th} step. The value of C is problem specific and fluctuates between .8 and .999 for practical scenarios. The total number of cooling steps for the algorithm depends on when the temperature becomes almost zero. At zero temperature, the algorithm stops selecting worse solutions and thus it finds the local optimum point in the region. Cooling steps also help bound the runtime of the algorithm by limiting the iterations to a specific number. For this problem, (initial temperature, cooling rate, cooling steps) were taken as $(10^4, .8, 10^2)$;

The neighborhood functions are usually uniform distributions with probabilities proportional to the size of the neighborhood. It is typically not temperature dependent. If the neighborhood size is small compared to total solution space, movement across the solution space is too slow. However, if the neighborhood size is large, the algorithm is unable to focus on a specific area where the optimal point might exist. Thus, neighborhood size is extremely problem specific. For path synthesis problem, the neighborhood of optimization variables $\phi_0, \frac{l2}{l1}, \frac{l3}{l1}, \frac{l4}{l1}$ have been defined as

$$\phi_{new} = \phi_0 + \phi_{size} \times (2 \times rand(0,1) - 1) \tag{2.47}$$

$$l21_{new} = l21 + l21_{size}^{(2\times rand(0,1)-1)} \tag{2.48}$$

$$l31_{new} = l31 + l31_{size}^{(2\times rand(0,1)-1)} \tag{2.49}$$

$$l41_{new} = l41 + l41_{size}^{(2\times rand(0,1)-1)} \tag{2.50}$$

where $(\phi_{size}, l21_{size}, l31_{size}, l41_{size})$ represent the size of neighborhood. For this problem, $(\frac{\pi}{10}, 1.5, 1.5, 1.5)$ specifies the sizes of each optimization variable. These values have been used in our implementation but are in no ways the best possible values. Thus, for our implementation, the neighborhood size remains constant and does not reduce with temperature.

At each temperature, the algorithm needs to run many times to get a good estimate of the neighborhood. This ensures that we move to a better point if it exists in the neighborhood. Another sensible heuristic is to ensure possible movement across solution space within each temperature state to ensure the algorithm spans the design space in totality. This is done by calculating maximum steps per temperature change according to the neighborhood size. It makes sense that the smaller the size, the larger the number of steps for a specific temperature to move from one extreme to another in solution space. These steps per temperature state vary from 100 to 1000 in practical problems. For path synthesis, this has been taken only as 50. This is mostly to speed up calculation at the cost of accuracy.

Simulated annealing prevents itself from getting stuck in local minima by occasionally selecting a worse candidate solution than the present one. The probability of selecting the new position in design space is governed by the acceptance probability function. It takes in the errors of old and new positions and gives us the feasibility/probability of the move. The usual acceptance criteria can be defined as

$$P(\text{acceptance}) = \begin{cases} 1 & \text{if neighbor cost}(c^*) \leq \text{solution cost}(c) \\ e^{-\frac{c^*-c}{t}} & \text{if neighbor cost}(c^*) > \text{solution cost}(c) \end{cases} \tag{2.51}$$

Once the acceptance probability is calculated, it's compared to a randomly-generated number between 0 and 1. If the acceptance probability is larger than the random number, the new point is deemed feasible and is made the new solution. Thus, according to the equation, the acceptance probability of a better solution is always 1. The worse the error for the new solution, the lesser is the probability of move. The probability of accepting a worse solution also decreases with a decrease in temperature. When the temperature reaches zero, the probability to accept a bad solution is zero and only better solutions are accepted.

The objective function is problem specific and calculates a cost/error associated with any feasible point in design space. In case of path synthesis, Eqn. 2.33 defines the error function. Within the objective function, the constraint Eqn. 2.36 is also checked and is defined infinite for infeasible parameters.

Thus, all the details to solve the path synthesis problem using simulated annealing has been detailed. It must be noted, given the extremely fast runtime of the implementation, the solution found might not be a minimum at all. However, since the algorithm converges onto the region gradually, it can be said, with some confidence, that the global minima are in the region of solution returned by the algorithm. Nelder Mead local search algorithm can be applied to the result to converge quickly to a minimum in the region. The focus of this implementation is the fast calculation of an approximate solution.

It must be mentioned that the rate of convergence to a result for the algorithm is extremely sensitive to the parameters defining simulated annealing optimization. Being a heuristic algorithm, the guidelines given above, with a healthy dose of hit and trial, were used to find a set of parameters which works satisfactorily. However, the assurance of these parameters being the best for our targeted speed and accuracy cannot be proved. The optimum parameters for intended accuracy-speed tradeoff can possibly even vary for different inputs paths which makes fixing them, an even harder problem.

2.2.4 Summary of Algorithm

A brief overview of complete path synthesis algorithm has been presented in this section.

```
Input: Set of Path points
Calculate Task curve Fourier descriptors;
Find the optimum 4 parameters by minimizing error;
Calculate the remaining parameters;
Output: Four-bar mechanism parameters
```

Algorithm 4: Pseudo-code for Path Synthesis

2.2.5 Results

In this section, the proposed algorithm's performance is tested for the existing uniform parameterization and the newly proposed optimum parametrization for Fourier based Path Synthesis algorithm.

2.2.5.1 Synthesizing Existing Mechanisms

To test the ability of Path Synthesis to create existing mechanisms, a sample mechanism is taken, points from its coupler path are sampled and then a mechanism is synthesized which goes through the sampled points. Ideally, we should get the exact same mechanism. However, a similar mechanism is also acceptable since the premise is of approximate synthesis.

A sample mechanism as visualized in Fig. 2.8 has been used to generate coupler path positions. The mechanism has been defined using the position of its fixed pivots, moving pivots and coupler coordinates as shown in Table 2.6. Coupler curve of this mechanism is sampled and used as input for Path Synthesis algorithm. The least square curves are approximated with seven Fourier Descriptors for these test cases.

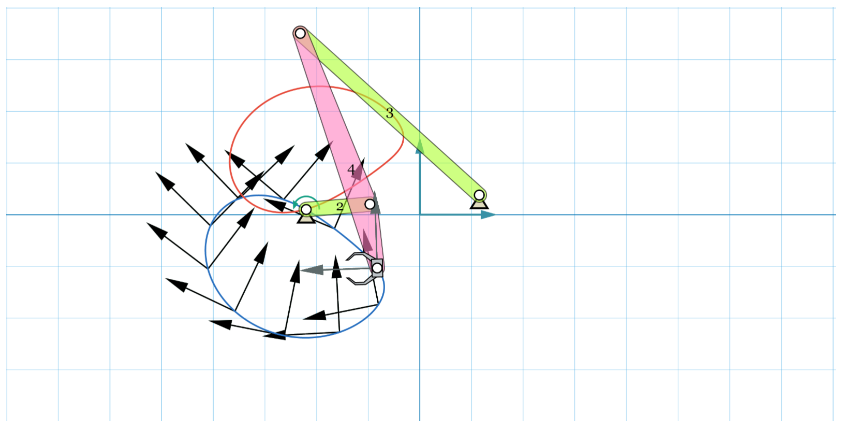


Figure 2.8: Path data Generator mechanism used to analyze effect on Path Synthesis under different parameterizations

Table 2.6: Path data Generator Mechanism design parameters as shown in Fig. 2.8

Point	X	Y
Link 2 Fixed Pivot	−2.20	0.10
Link 2 Moving Pivot	−0.97	0.20
Link 3 Fixed Pivot	1.15	0.38
Link 3 Moving Pivot	−2.32	3.51
Coupler Point	−0.825	−1.033

Test case 1

Coupler curve data can be sampled randomly or uniformly over the input angle intervals. First, we compare the use of optimum and uniform parameterizations for uniformly sampled data. The sampled data used for testing is given in Table 2.7. Fig. 2.9 represents the uniform parameterization case while Fig. 2.10 represents the optimum parameterization case. Table 2.8 and Table 2.9 contains the Fourier descriptor and Synthesized mechanism data for uniform parametrization while Table 2.10 and Table 2.11 shows same for optimum parameterization. It can be observed that both uniform and optimum parametrization yields good results. They approximate the original mechanism well when Coupler path has been sampled uniformly.

Table 2.7: Test case 1: Input uniformly sampled point data to analyze effect of Path Synthesis under different parameterizations

X	Y
−0.8253	−1.0329
−1.6554	−0.2721
−2.6392	0.292
−3.5027	0.3088
−4.0549	−0.2141
−4.1022	−1.0521
−3.5805	−1.8708
−2.625	−2.3408
−1.5557	−2.2742
−0.8016	−1.7382

Test case 2

Second, we compare the use of optimum and uniform parameterizations for non-uniformly sampled data. The sampled data used for testing is given in Table 2.12. Fig. 2.11 represents the uniform parameterization case while Fig. 2.12 represents the optimum parameterization case. Table 2.13 and Table 2.14 contains the Fourier descriptor and Synthesized mechanism data for uniform parametrization while Table 2.15 and Table 2.16 shows same for optimum parameterization. In this case, the results are drastically different. Optimum parameterization yield way better result than uniform parameterization. Also, the uniform parameterization result is fundamentally different

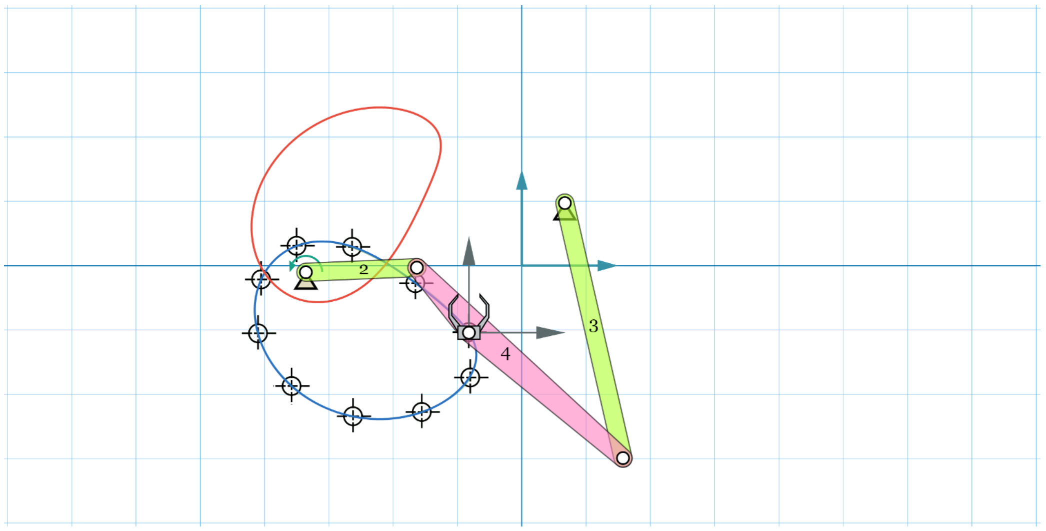


Figure 2.9: Test case 1: Synthesized mechanism for uniformly sampled input using uniform parametrization

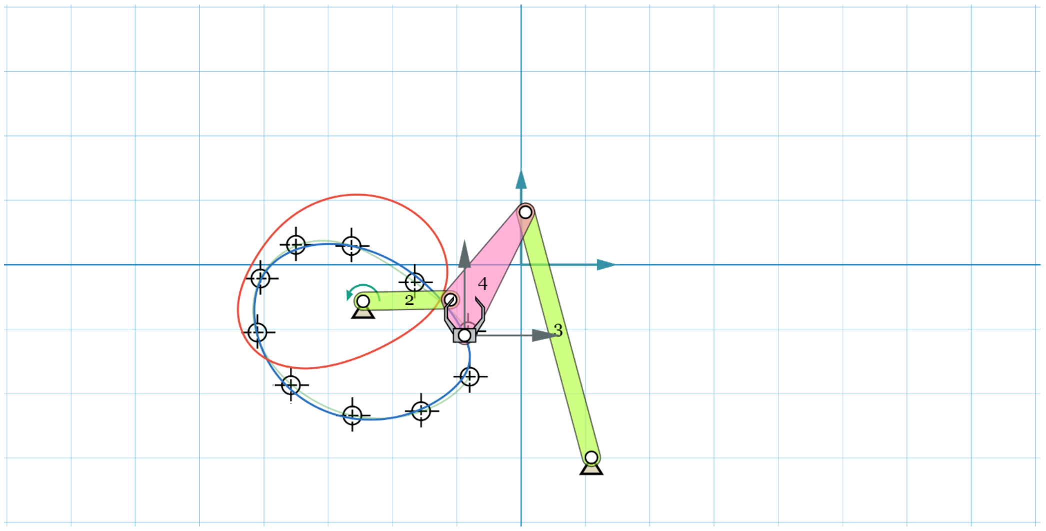


Figure 2.10: Test case 1: Synthesized mechanism for uniformly sampled input using optimum parametrization

Table 2.8: Test case 1: Task path and Coupler path Descriptor Data for Fig. 2.9

	Task path		Coupler path	
Descriptor	Complex Value	Magnitude	Complex Value	Magnitude
p=−3	$0.02016 - 0.0138i$	0.0244	$0.02259 - 0.01406i$	0.0266
p=−2	$0.0526 - 0.05623i$	0.0770	$0.05239 - 0.05593i$	0.0766
p=−1	$0.1256 - 0.2511i$	0.2808	$0.1254 - 0.2513i$	0.2808
p=0	$-2.544 - 1.017i$	2.7399	$-2.544 - 1.017i$	2.7399
p=1	$1.488 + 0.252i$	1.5096	$1.488 + 0.252i$	1.5096
p=2	$0.01475 + 0.04111i$	0.0437	$0.01394 + 0.04085i$	0.0432
p=3	$0.004255 + 0.009628i$	0.0105	$0.003527 + 0.005555i$	0.0066

Table 2.9: Test case 1: Synthesized mechanism parameters Data for Fig. 2.9

l_1	l_2	l_3	l_4	x_0	y_0	θ_1	r	α	ϕ_0
4.169	1.728	4.363	4.066	−3.358	−0.103	0.261	1.295	−0.150	6.064

Table 2.10: Test case 1: Task path and Coupler path Descriptor Data for Fig. 2.10

	Task path		Coupler path	
Descriptor	Complex Value	Magnitude	Complex Value	Magnitude
p=−3	0.01488 − 0.01291i	0.0197	−0.001126 − 0.01555i	0.0156
p=−2	0.05154 − 0.03428i	0.0619	0.04002 − 0.02767i	0.0487
p=−1	0.1264 − 0.1975i	0.2345	0.1269 − 0.1946i	0.2323
p=0	−2.467 − 1.059i	2.6852	−2.467 − 1.059i	2.6852
p=1	1.481 + 0.2157i	1.4965	1.481 + 0.2157i	1.4965
p=2	−0.04032 − 0.004176i	0.0405	−0.06052 − 0.008252i	0.0611
p=3	0.004025 + 0.0632i	0.0633	0.003789 + 0.002891i	0.0048

Table 2.11: Test case 1: Synthesized mechanism parameters Data for Fig. 2.10

l_1	l_2	l_3	l_4	x_0	y_0	θ_1	r	α	ϕ_0
4.303	1.362	1.799	3.939	−2.459	−0.564	−0.601	0.597	−2.050	0.617

from the initial mechanism which was a crank-rocker mechanism. Thus, at-least for existing mechanisms, we can conclude that optimum parameterization is way better at handling generalized input path data that uniform parameterization.

Table 2.12: Test case 2: Input non-uniformly sampled point data to analyze effect of Path Synthesis under different parameterizations

X	Y
−0.8253	−1.0329
−1.2488	−0.6005
−1.7546	−0.1982
−2.3253	0.1599
−2.6153	0.2839
−2.9181	0.3599
−3.2869	0.3634
−3.5212	0.3017
−3.9349	−0.0109
−4.0380	−0.1780
−4.1510	−0.6345
−4.1444	−0.8304
−4.1023	−1.05203
−4.0594	−1.1848
−3.9470	−1.4231
−3.5411	−1.9049
−3.2360	−2.1158
−2.5438	−2.3554
−1.8641	−2.3523
−1.2110	−2.1151
−0.8494	−1.8050
−0.6925	−1.4424

It must be noted that the mechanisms found using optimization techniques are guaranteed local optimum. However, it is uncertain if these are actually the global minimum in search space. Simulated Annealing only guarantees global minimum asymptotically over a long time. As mentioned earlier, faster solutions have been preferred in our implementation.

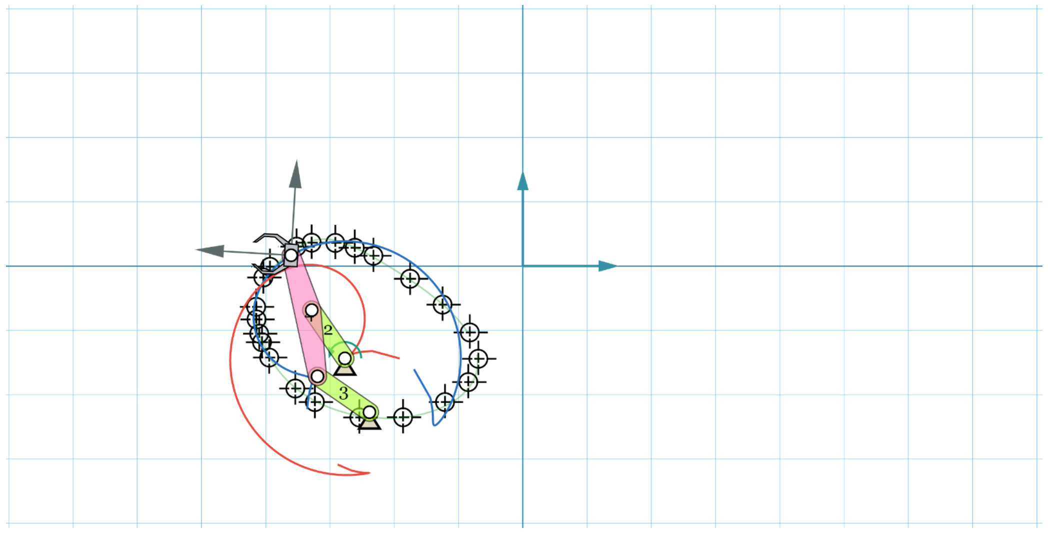


Figure 2.11: Test case 2: Synthesized mechanism for non-uniformly sampled input using uniform parametrization

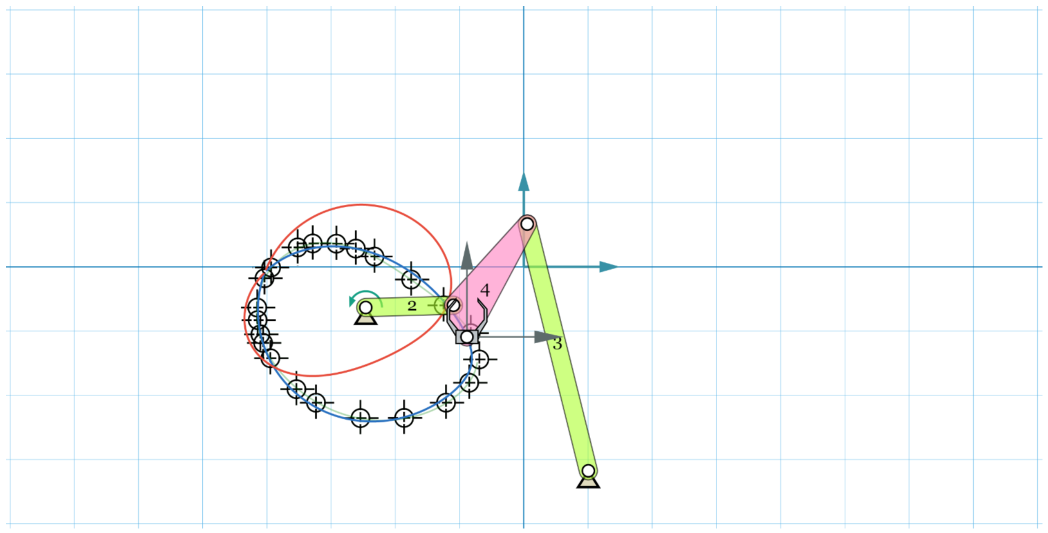


Figure 2.12: Test case 2: Synthesized mechanism for non-uniformly sampled input using optimum parametrization

Table 2.13: Test case 2: Task path and Coupler path Descriptor Data for Fig. 2.11

	Task path		Coupler path	
Descriptor	Complex Value	Magnitude	Complex Value	Magnitude
p=−3	$0.02795 - 0.03086i$	0.0416	$-0.04315 - 0.01576i$	0.0459
p=−2	$0.03278 - 0.126i$	0.1302	$-0.007658 - 0.07339i$	0.0738
p=−1	$0.174 - 0.1808i$	0.2509	$0.1788 - 0.1024i$	0.2060
p=0	$-2.789 - 0.8947i$	2.9292	$-2.789 - 0.8947i$	2.9292
p=1	$1.454 - 0.01518i$	1.4542	$1.454 - 0.01518i$	1.4542
p=2	$0.2398 + 0.1194i$	0.2679	$0.2584 + 0.07961i$	0.2704
p=3	$0.0018 + 0.09918i$	0.0992	$-0.02167 + 0.1402i$	0.1419

Table 2.14: Test case 2: Synthesized mechanism parameters Data for Fig. 2.11

l_1	l_2	l_3	l_4	x_0	y_0	θ_1	r	α	ϕ_0
1.730	1.693	1.971	2.108	−3.328	−0.854	−0.979	0.897	0.067	1.290

Table 2.15: Test case 2: Task path and Coupler path Descriptor Data for Fig. 2.12

	Task path		Coupler path	
Descriptor	Complex Value	Magnitude	Complex Value	Magnitude
p=−3	$0.01283 - 0.01092i$	0.0169	$-0.009025 - 0.01366i$	0.0164
p=−2	$0.053 - 0.03109i$	0.0615	$0.04745 - 0.02206i$	0.0523
p=−1	$0.1239 - 0.1935i$	0.2297	$0.1211 - 0.1896i$	0.2250
p=0	$-2.464 - 1.06i$	2.6826	$-2.464 - 1.06i$	2.6826
p=1	$1.478 + 0.2214i$	1.4941	$1.478 + 0.2214i$	1.4941
p=2	$-0.04315 - 0.004513i$	0.0434	$-0.06751 - 0.01835i$	0.0700
p=3	$-1.967\text{e-}4 + 0.06792i$	0.0679	$0.006821 + 0.01176i$	0.0136

Table 2.16: Test case 2: Synthesized mechanism parameters Data for Fig. 2.12

l_1	l_2	l_3	l_4	x_0	y_0	θ_1	r	α	ϕ_0
4.325	1.369	1.747	3.952	−2.464	−0.615	−0.634	0.552	−1.980	0.658

2.2.5.2 Synthesizing Unknown Mechanisms

Synthesizing mechanism using path points which are known to fall on an existing mechanism is a relatively easier problem. Our algorithm should be able to deal well with cases where it is unknown if any four- bar mechanism exists which can fulfill path constraints. Thus, this section deals with path synthesis for random data-points which may not satisfy any mechanism.

The number of descriptors being used to describe the Task path controls if Interpolation or approximation of path points will take place. When number of descriptors being considered is equal to the path point input, Task Path is an interpolation curve. When number of descriptors is less than path point input, Task Path is an approximation curve. The optimum parameterization directly affects the Task path and hence has different effect when Task curve is an interpolation curve than when it is an approximation curve.

Test case 3

First, the case when Task curve interpolates user-inputted path points is considered. The sampled data used for testing is given in Table 2.17. Fig. 2.13 represents the uniform parameterization case while Fig. 2.14 represents the optimum parameterization case. Table 2.18 and Table 2.19 contains the Fourier descriptor and Synthesized mechanism data for uniform parametrization while Table 2.20 and Table 2.20 shows same for optimum parameterization. Better results are observed using Optimum parameterization.

Table 2.17: Test case 3: Input point data to analyze effect of Path Synthesis under different parameterizations

X	Y
−6.0652	1.5414
−4.2982	2.2180
−2.9825	2.5188
−0.5890	2.4436
4.4361	−1.7794

Test case 4

Another case when Task curve interpolates user-inputted path points is considered. The sampled data used for this case is given in Table 2.22. Fig. 2.15

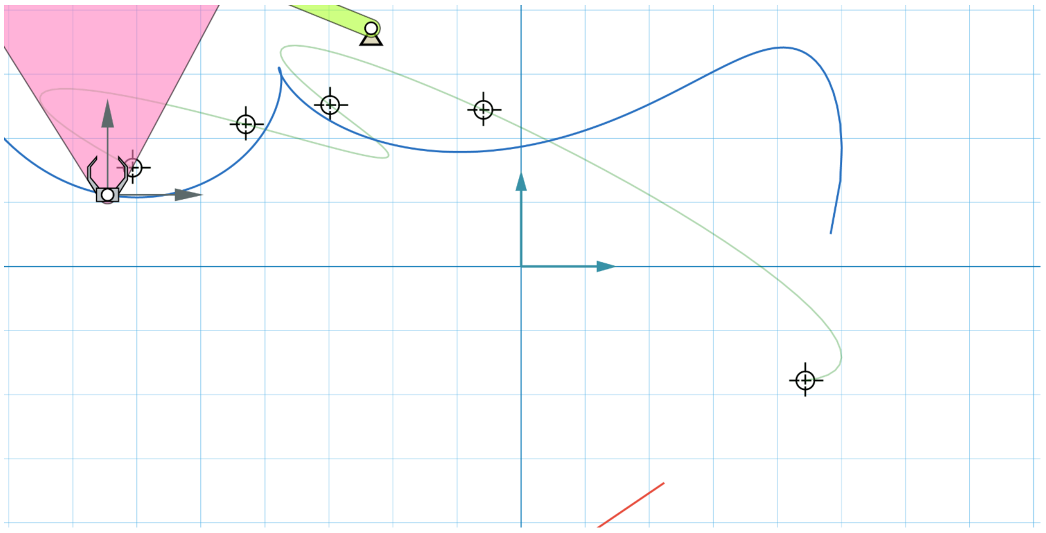


Figure 2.13: Test case 3: Synthesized mechanism using uniform parametrization

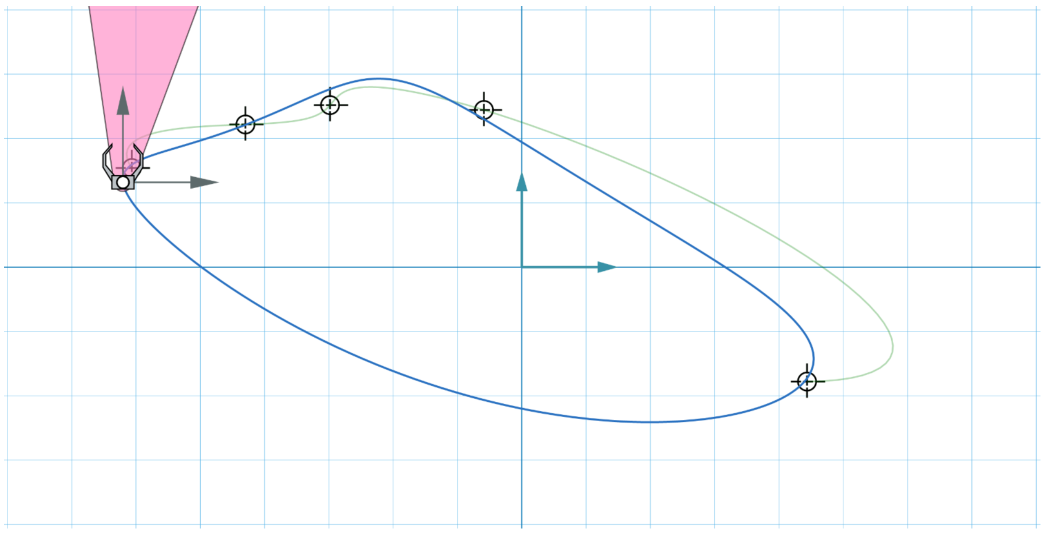


Figure 2.14: Test case 3: Synthesized mechanism using optimum parametrization

Table 2.18: Test case 3: Task path and Coupler path Descriptor Data for Fig. 2.13

	Task path		Coupler path	
Descriptor	Complex Value	Magnitude	Complex Value	Magnitude
p=−2	$-1.953 - 0.08941i$	1.9552	$-1.927 + 0.001045i$	1.9267
p=−1	$-1.355 - 2.423i$	2.7763	$-1.578 - 2.342i$	2.8244
p=0	$-1.91 + 1.404i$	2.3700	-1.91 + $1.404i$	2.3700
p=1	$0.1551 + 1.477i$	1.4847	$0.1551 + 1.477i$	1.4847
p=2	$-1.002 + 1.174i$	1.5433	-1.197 + $0.5792i$	1.3294

Table 2.19: Test case 3: Synthesized mechanism parameters Data for Fig. 2.13

l_1	l_2	l_3	l_4	x_0	y_0	θ_1	r	α	ϕ_0
4.038	4.498	7.132	8.315	−6.277	4.621	−0.226	7.419	0.965	0.969

Table 2.20: Test case 3: Task path and Coupler path Descriptor Data for Fig. 2.14

	Task path		Coupler path	
Descriptor	Complex Value	Magnitude	Complex Value	Magnitude
p=−2	$-0.7192 + 0.9789i$	1.2147	$-0.6846 + 0.7712i$	1.0312
p=−1	$-2.759 - 1.96i$	3.3841	$-2.783 - 1.936i$	3.3898
p=0	$-1.033 + 1.025i$	1.4555	$-1.033 + 1.025i$	1.4555
p=1	$-0.6791 + 1.752i$	1.8792	$-0.6791 + 1.752i$	1.8792
p=2	$-0.8745 - 0.2552i$	0.9110	$-1.024 - 0.294i$	1.0651

Table 2.21: Test case 3: Synthesized mechanism parameters Data for Fig. 2.14

l_1	l_2	l_3	l_4	x_0	y_0	θ_1	r	α	ϕ_0
13.427	4.249	6.608	15.355	−5.931	7.017	0.509	9.316	0.240	0.286

represents the uniform parameterization case while Fig. 2.16 represents the optimum parameterization case. Table 2.23 and Table 2.24 contains the Fourier descriptor and Synthesized mechanism data for uniform parametrization while Table 2.25 and Table 2.25 shows same for optimum parameterization. Better results just like the previous case are observed using Optimum parameterization. This can be attributed to the smoother Task path being generated.

Table 2.22: Test case 4: Input point data to analyze effect of Path Synthesis under different parameterizations

X	Y
−5.9649	2.0677
−5.9148	2.5815
−2.6817	2.4436
−1.0276	1.2907
0.4511	0.1378
0.6140	−0.7143
−0.1504	−1.7419
−1.0025	−2.0677
−3.1830	−2.2180

Table 2.23: Test case 4: Task path and Coupler path Descriptor Data for Fig. 2.15

	Task path		Coupler path	
Descriptor	Complex Value	Magnitude	Complex Value	Magnitude
p=−4	$-0.1434 + 0.2558i$	0.2933	$-0.1243 + 0.09496i$	0.1564
p=−3	$-0.3389 + 0.01849i$	0.3394	$-0.3102 + 0.1065i$	0.3280
p=−2	$-0.6863 - 0.1632i$	0.7055	$-0.8236 - 0.05511i$	0.8255
p=−1	$-2.687 - 0.2777i$	2.7011	$-2.63 - 0.3277i$	2.6502
p=0	$-2.04 + 0.2058i$	2.0500	$-2.04 + 0.2058i$	2.0500
p=1	$-0.3293 + 0.8554i$	0.9166	$-0.3293 + 0.8554i$	0.9166
p=2	$0.09096 + 0.4632i$	0.4721	$0.2565 + 0.5486i$	0.6056
p=3	$3.19\text{e-}4 + 0.3465i$	0.3465	$-0.01048 + 0.1059i$	0.1064
p=4	$0.1681 + 0.3633i$	0.4003	$-0.02299 + 0.09183i$	0.0947

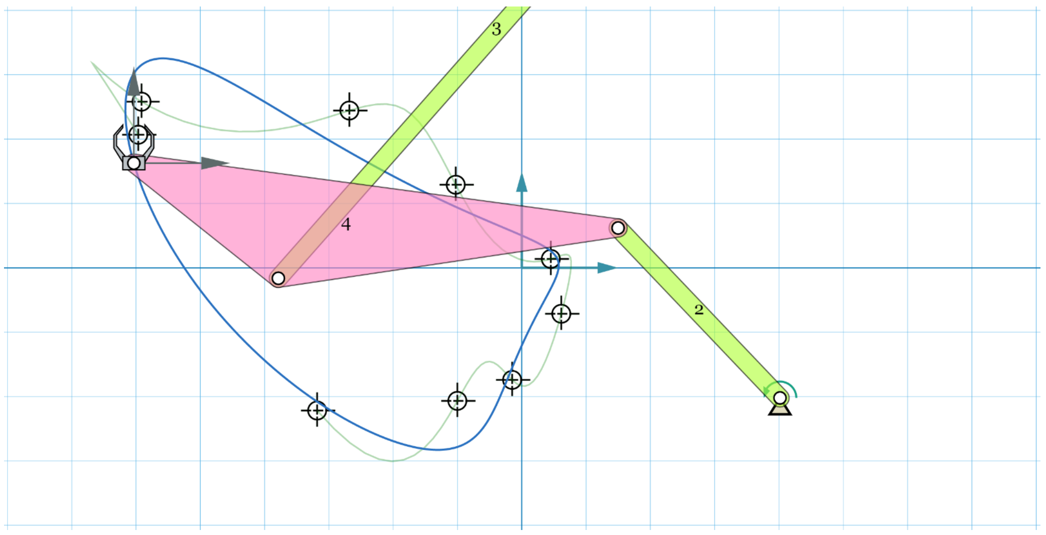


Figure 2.15: Test case 4: Synthesized mechanism using uniform parametrization

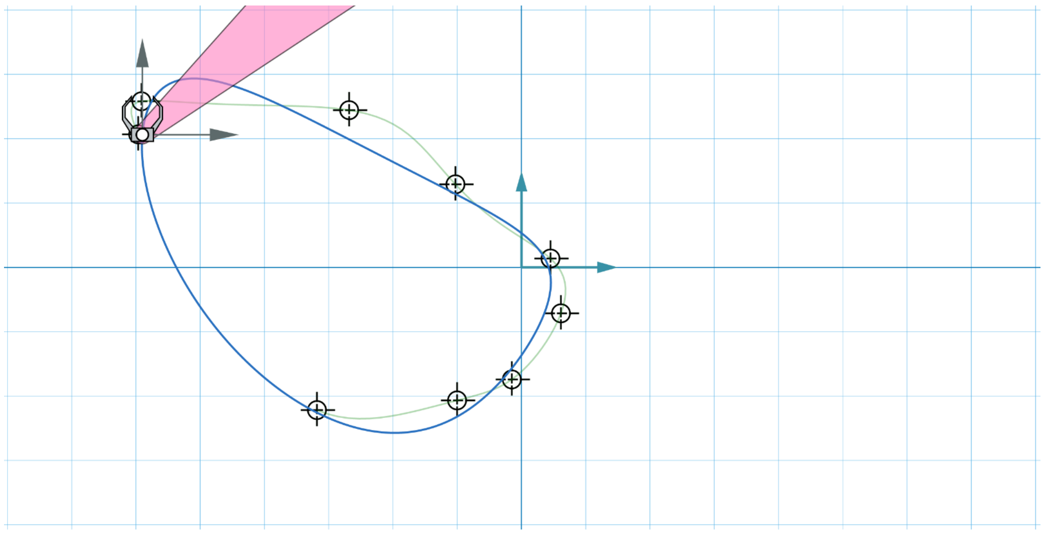


Figure 2.16: Test case 4: Synthesized mechanism using optimum parametrization

Table 2.24: Test case 4: Synthesized mechanism parameters Data for Fig. 2.15

l_1	l_2	l_3	l_4	x_0	y_0	θ_1	r	α	ϕ_0
9.601	3.649	5.341	10.261	4.016	-2.022	1.676	7.598	-0.280	0.657

Table 2.25: Test case 4: Task path and Coupler path Descriptor Data for Fig. 2.16

	Task path		Coupler path	
Descriptor	Complex Value	Magnitude	Complex Value	Magnitude
p=-4	$0.0706 + 0.02878i$	0.0762	$0.001212 + 0.03129i$	0.0313
p=-3	$-0.09564 + 0.06566i$	0.1160	$-0.05046 + 0.1282i$	0.1378
p=-2	$-0.3552 + 0.1578i$	0.3887	$-0.4064 + 0.2881i$	0.4982
p=-1	$-2.693 + 0.4287i$	2.7273	$-2.659 + 0.4019i$	2.6891
p=0	$-2.334 + 0.2155i$	2.3435	$-2.334 + 0.2155i$	2.3435
p=1	$-0.4451 + 0.6047i$	0.7509	$-0.4451 + 0.6047i$	0.7509
p=2	$-0.02 + 0.2612i$	0.2619	$0.03202 + 0.3735i$	0.3749
p=3	$-0.0888 + 0.1588i$	0.1819	$-0.02017 + 0.02214i$	0.0299
p=4	$-0.003836 + 0.1466i$	0.1466	$-0.01474 + 0.005768i$	0.0158

Table 2.26: Test case 4: Synthesized mechanism parameters Data for Fig. 2.16

l_1	l_2	l_3	l_4	x_0	y_0	θ_1	r	α	ϕ_0
6.324	2.001	5.582	4.732	4.866	12.652	-1.574	14.897	0.251	0.616

Test case 5

Let us now focus our attention on the second scenario where Task curve approximates the given user-defined Path points. The sampled data used for this case is given in Table 2.27. Just five Fourier descriptors have been used to solve this Path Synthesis problem. Fig. 2.17 represents the uniform parameterization case while Fig. 2.18 represents the optimum parameterization case. Table 2.28 and Table 2.29 contains the Fourier descriptor and Synthesized mechanism data for uniform parametrization while Table 2.30 and Table 2.30 shows same for optimum parameterization. Very similar synthesized mechanisms results are observed for both parameterizations. This can be attributed to relatively small change in the Task curve geometry by using a different parameterization.

Table 2.27: Test case 5: Input point data to analyze effect of Path Synthesis under different parameterizations

X	Y
−3.0075	1.3910
−1.8045	2.3935
−1.4035	1.9799
−0.1504	1.5915
0.8521	0.8897
1.0777	−0.6642
0.8521	−1.4411
−0.0501	−1.6165

Table 2.28: Test case 5: Task path and Coupler path Descriptor Data for Fig. 2.17

	Task path		Coupler path	
Descriptor	Complex Value	Magnitude	Complex Value	Magnitude
p=−2	$-3.143 + 1.155i$	3.3483	$-3.165 + 1.169i$	3.3743
p=−1	$6.428 - 0.1827i$	6.4308	$6.414 - 0.1735i$	6.4166
p=0	$-10.39 + 5.06i$	11.5580	$-10.39 + 5.06i$	11.5580
p=1	$3.915 - 7.604i$	8.5528	$3.915 - 7.604i$	8.5528
p=2	$0.2588 + 3.031i$	3.0418	$0.2724 + 3.031i$	3.0429

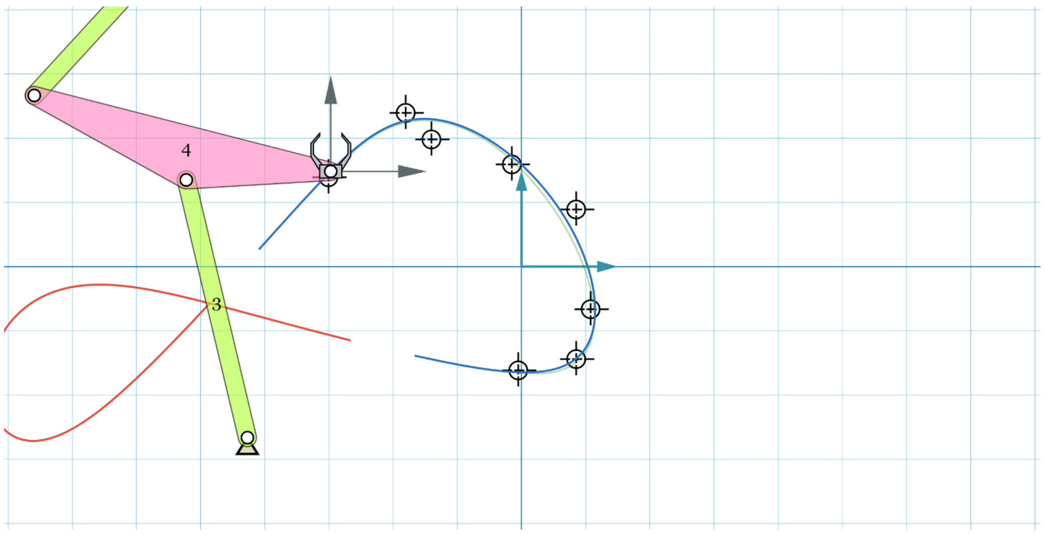


Figure 2.17: Test case 5: Synthesized mechanism using uniform parametrization

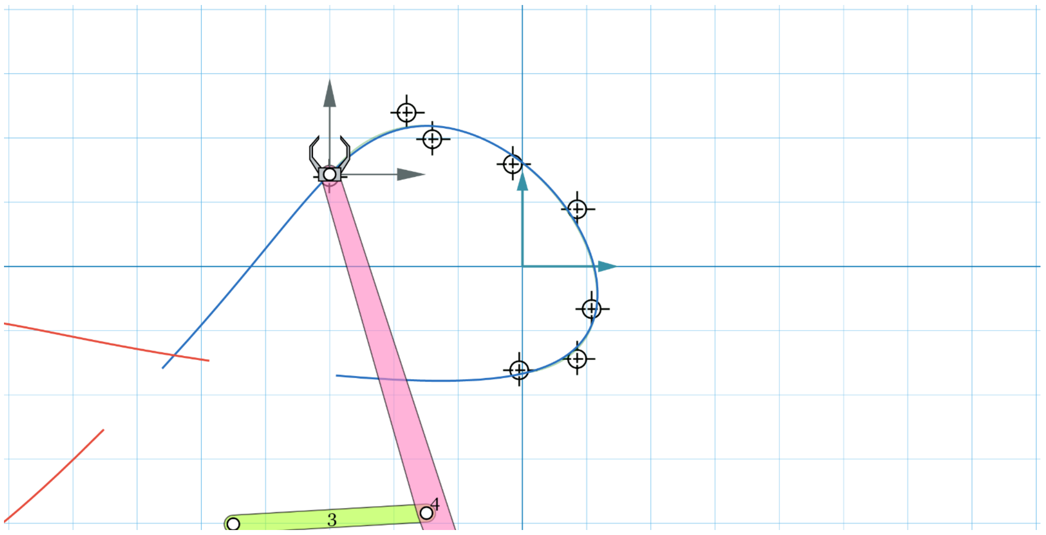


Figure 2.18: Test case 5: Synthesized mechanism using optimum parametrization

Table 2.29: Test case 5: Synthesized mechanism parameters Data for Fig. 2.17

l_1	l_2	l_3	l_4	x_0	y_0	θ_1	r	α	ϕ_0
7.128	3.646	3.434	2.256	-0.202	-8.516	2.178	7.262	-0.310	5.498

Table 2.30: Test case 5: Task path and Coupler path Descriptor Data for Fig. 2.18

	Task path		Coupler path	
Descriptor	Complex Value	Magnitude	Complex Value	Magnitude
p=-2	$-3.076 + 1.468i$	3.4085	$-3.076 + 1.468i$	3.4085
p=-1	$4.407 - 1.129i$	4.5496	$4.407 - 1.129i$	4.5496
p=0	$-6.333 + 3.794i$	7.3830	$-6.333 + 3.794i$	7.3830
p=1	$2.025 - 4.318i$	4.7693	$2.025 - 4.318i$	4.7693
p=2	$-0.01952 + 1.614i$	1.6142	$-0.01956 + 1.614i$	1.6142

Table 2.31: Test case 5: Synthesized mechanism parameters Data for Fig. 2.18

l_1	l_2	l_3	l_4	x_0	y_0	θ_1	r	α	ϕ_0
9.115	4.059	5.117	3.013	-1.073	-12.459	1.956	10.594	-0.040	5.555

Test case 6

Another case of the scenario where Task curve approximates the given user-defined Path points is taken. The sampled data used for this case is given in Table 2.32. Just five Fourier descriptors have been used to solve this Path Synthesis problem. Fig. 2.19 represents the uniform parameterization case while Fig. 2.20 represents the optimum parameterization case. Table 2.33 and Table 2.34 contains the Fourier descriptor and Synthesized mechanism data for uniform parametrization while Table 2.35 and Table 2.35 shows same for optimum parameterization. Again, the synthesized mechanism are not drastically different like the previous case.

Table 2.32: Test case 6: Input point data to analyze effect of Path Synthesis under different parameterizations

X	Y
−4.2105	0.3258
−3.4586	1.5288
−2.6065	1.6792
−1.3033	1.8672
0.1003	1.2657
1.2030	0.6266
1.8672	−1.2531
1.6040	−1.7920
0.8521	−2.3308
0.0752	−2.2431
−1.2782	−1.8296
−1.5288	−1.2782
−1.8546	−0.6767
−2.3058	−0.3509
−3.1830	−0.3759

2.2.5.3 Observations

After analyzing all the example data, a couple of observations can be made. When Task curve interpolates the path points, optimal parameterization is very effective. This happens because interpolation curves extremely sensitive to different parameterizations. However, when Task curve approximates

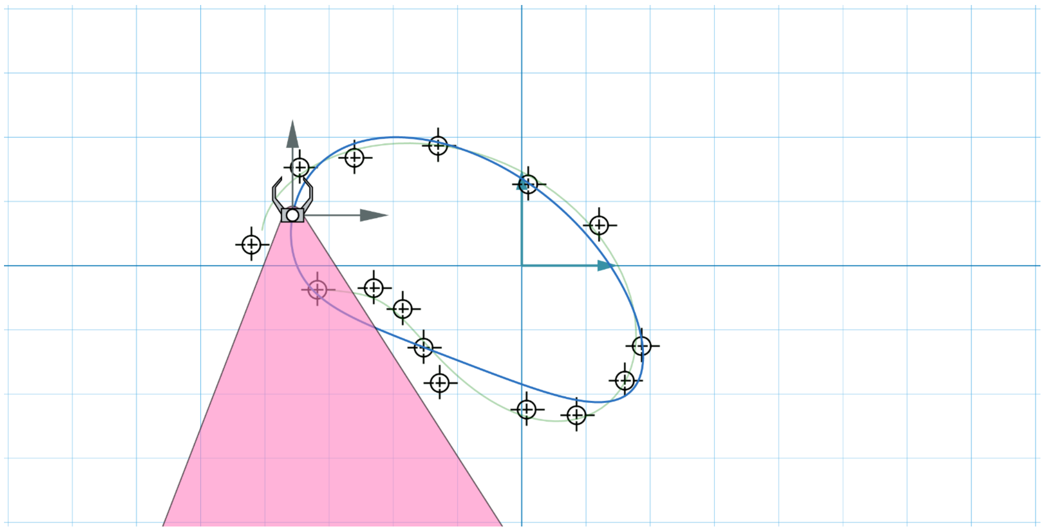


Figure 2.19: Test case 6: Synthesized mechanism using uniform parametrization

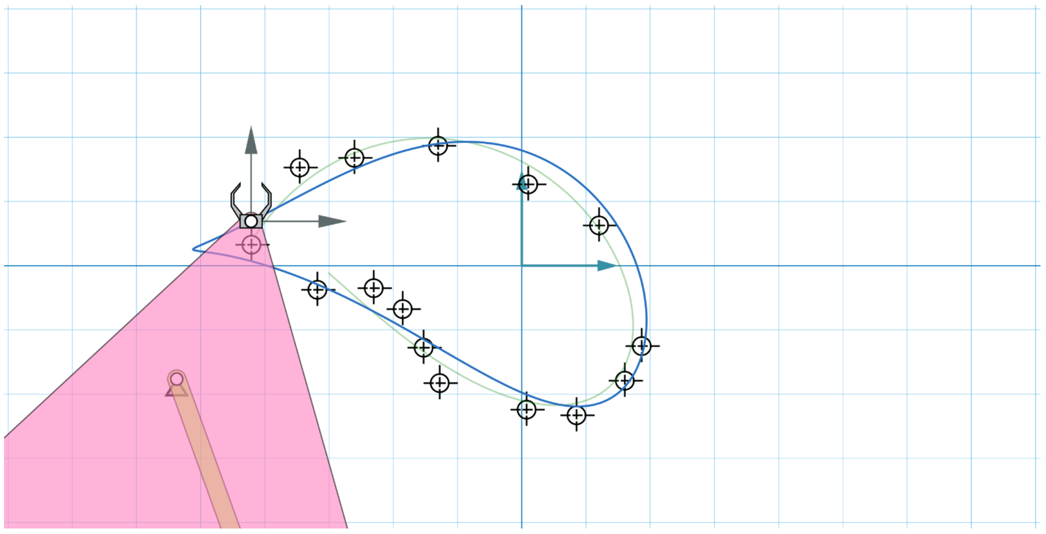


Figure 2.20: Test case 6: Synthesized mechanism using optimum parametrization

Table 2.33: Test case 6: Task path and Coupler path Descriptor Data for Fig. 2.19

	Task path		Coupler path	
Descriptor	Complex Value	Magnitude	Complex Value	Magnitude
p=−2	$-0.4317 - 0.596i$	0.7359	$-0.3561 - 0.5088i$	0.6210
p=−1	$-1.586 + 1.403i$	2.1173	$-1.585 + 1.413i$	2.1233
p=0	$-1.332 - 0.3022i$	1.3659	$-1.332 - 0.3022i$	1.3659
p=1	$-0.707 + 0.08172i$	0.7117	$-0.707 + 0.08172i$	0.7117
p=2	$0.01146 - 0.0414i$	0.0430	$0.2299 + 0.0727i$	0.2412

Table 2.34: Test case 6: Synthesized mechanism parameters Data for Fig. 2.19

l_1	l_2	l_3	l_4	x_0	y_0	θ_1	r	α	ϕ_0
4.552	1.735	14.385	14.385	−8.772	−8.688	0.761	11.677	1.491	4.563

Table 2.35: Test case 6: Task path and Coupler path Descriptor Data for Fig. 2.20

	Task path		Coupler path	
Descriptor	Complex Value	Magnitude	Complex Value	Magnitude
p=−2	$-0.5691 - 0.6445i$	0.8598	$-0.5687 - 0.6448i$	0.8597
p=−1	$-1.162 + 1.772i$	2.1195	$-1.162 + 1.773i$	2.1195
p=0	$-1.906 - 0.04384i$	1.9062	$-1.906 - 0.04384i$	1.9062
p=1	$-0.7783 - 0.5533i$	0.9549	$-0.7783 - 0.5533i$	0.9549
p=2	$0.2387 - 0.1034i$	0.2601	$0.2392 - 0.1031i$	0.2605

Table 2.36: Test case 6: Synthesized mechanism parameters Data for Fig. 2.20

l_1	l_2	l_3	l_4	x_0	y_0	θ_1	r	α	ϕ_0
8.044	3.120	24.414	25.420	−12.920	−4.554	0.354	12.282	1.578	4.261

the path points, optimal parameterization is less effective. This can be attributed to the fact that approximation curve are less sensitive to different parameterization. Thus, the synthesis problems where optimal parameterization is most advantageous are the ones with number of points being equal to Fourier descriptors being considered.

2.3 Frenet frame based Path Approximation

Motion synthesis problem can be described as calculating a mechanism whose coupler passes through a set of user-defined poses. A pose contains path and orientation and is mathematically defined as (x, y, θ).

Another way to solve the path synthesis problem is to reformulate it as a motion synthesis problem and solve that. This can be accomplished by attaching an adequate orientation to each of the inputted path points and consequently turning them into poses. In doing this, additional constraints are being applied to the existing problem. Motion synthesis turns out to be a mathematically less complex problem as each dyad can be calculated separately effectively halving the number of unknowns in the equations. Thus, if the applied extra orientation constraints make physical sense, the path synthesis problem can be solved with greater speed and ease. Another great advantage of the motion synthesis algorithm is that it gives multiple solutions instead of single solution due to which the user has enhanced flexibility.

Thus, Path synthesis algorithm solve nonlinear system of equations in real space while Motion synthesis algorithm solve a linear system of equations in quaternion space. Consequently, the motivation to convert path synthesis problem into a motion synthesis problem is enhancement in computation speed.

2.3.1 Calculating orientations

The path synthesis problem dealt till now has been directly related to the general shape of coupler curve rather than a small number of precision points. Consequently, an interpolation curve passing through the inputted path points, describing the target shape, is already an integral part of the algorithm. In the previous section, this target curve has been an approximation using a limited number of Fourier basis functions. However, any other curve like a Bezier or b-spline curve is an equally valid possible target path.

Frenet-Serret frame [29] is a moving reference frame of orthonormal vectors which are used to describe a time-parametrized curve locally at each point on it. The orthonormal vectors are denoted by T, N, B representing tangent, normal and binormal vectors. The osculating plane is defined as the plane spanned by the vectors T and N. If a three-dimensional Euclidean space curve $r(t)$ is parametrized by time t belonging to the interval $[a, b]$, then T, N, B are defined as

$$T(t) = \frac{r'(t)}{\|r'(t)\|} \tag{2.52}$$

$$N(t) = \frac{T'(t)}{\|T'(t)\|} \tag{2.53}$$

$$B(t) = \frac{T(t) \times N(t)}{\|T(t) \times N(t)\|} \tag{2.54}$$

In the path synthesis problem of planer mechanisms being discussed, the task curve is strictly two-dimensional. As a result, The binormal vector always points out of the osculating plane and its speed of rotation (torsion) is zero. The orientation of the local frame can be determined by finding the angle at which the tangent exists. This step extracts the orientation data at each path point and combines with it to become pose data. Once path data has been transformed to pose data using the target curve, path synthesis transforms to motion synthesis.

2.3.1.1 Fourier Curve

For an approximation curve the least square fitted to path points using limited Fourier basis functions described by

$$T = \sum_{k=-n}^{n} T_k e^{ik\phi}, \tag{2.55}$$

the tangent vector is

$$\frac{dT}{d\phi} = T' = \sum_{k=-n}^{n} ikT_k e^{ik\phi} = \sum_{k=-n}^{n} T'_k e^{ik\phi} \tag{2.56}$$

and the orientation is

$$\theta_t = arg(T'_t) \tag{2.57}$$

Using these relationships, orientation data for each path point can be calculated and combined to give pose data.

2.3.1.2 B-spline Curve

A B-spline can be used to represent the target curve for path synthesis problem. To calculate it, a B-spline global interpolation algorithm is used which first finds parameters for input path points using the centripetal method. Once a set of parameters is obtained, knot vector is computed using the averaging method. After that, the set of control points can be calculated by solving a system of equation using input data points and B-spline basis function values at calculated parameters. As a result, a target b-spline which interpolates input path points is calculated. It is described as

$$C(u) = \sum_{i=0}^{n} N_{i,p}(u) P_i \tag{2.58}$$

where $N_{i,p}$ are B-spline basis functions of degree p and P_i represents control points. It is characterized by a knot vector $U = \{u_0, u_1, \ldots, u_m\}$ and is of degree p.

The derivative of a B-spline curve is given by the equation

$$C'(u) = \sum_{i=0}^{n-1} N_{i+1,p-1}(u) Q_i \tag{2.59}$$

where Q_i are defined as

$$Q_i = \frac{p}{u_{i+p+1} - u_{i+1}} (P_{i+1} - P_i) \tag{2.60}$$

Therefore, the derivative of a B-spline curve is another B-spline curve of degree $p - 1$ on the original knot vector with a new set of n control points $Q_0, Q_1, \ldots, Q_{n-1}$. Once the tangent vector is known, the orientation can be calculated as $\tan^{-1}(y/x)$

2.3.2 Motion Synthesis Algorithm Review

Now that the path synthesis problem has been reformulated as motion synthesis problem, solution mechanisms can be achieved by calculating the dyads. Algebraic fitting based motion synthesis algorithm [30–33] has been used in our implementation. This goal of this approach is to map the poses into image space and fit geometric constraint manifolds described using image space coordinates to calculate the least square solution dyads.

First, using kinematic mapping, each of the user-defined pose $\{d_1, d_2, \phi\}$ is mapped to quaternion space defined by a four-dimensional vector $\mathbf{Z} = \{Z_1, Z_2, Z_3, Z_4\}$ is Image Space. The relations mapping real space to image space are

$$Z_1 = \frac{1}{2}(d_1 \cos\frac{\phi}{2} + d_2 \sin\frac{\phi}{2}) \tag{2.61}$$

$$Z_2 = \frac{1}{2}(-d_1 \sin\frac{\phi}{2} + d_2 \cos\frac{\phi}{2}) \tag{2.62}$$

$$Z_3 = \sin\frac{\phi}{2} \tag{2.63}$$

$$Z_4 = \cos\frac{\phi}{2} \tag{2.64}$$

The geometric motion constraint which every dyad needs to satisfy is given by the G-constraint manifold described as

$$\begin{aligned} q_1(Z_1^2 + Z_2^2) + q_2(Z_1Z_3 - Z_2Z_4) + q_3(Z_2Z_3 + Z_1Z_4) \\ +q_4(Z_1Z_3 + Z_2Z_4) + q_5(Z_2Z_3 - Z_1Z_4) + q_6Z_3Z_4 \\ +q_7(Z_3^2 - Z_4^2) + q_8(Z_3^2 + Z_4^2) = 0, \end{aligned} \tag{2.65}$$

where $q_i(i = 1, 2, \cdot, 8)$ represents a dyad in Image space. Thus, the coefficients for G-manifold defined for each pose can be calculated using

$$A_{i1} = Z_{i1}^2 + Z_{i2}^2 \tag{2.66}$$

$$A_{i2} = Z_{i1}Z_{i3} - Z_{i2}Z_{i4} \tag{2.67}$$

$$A_{i3} = Z_{i2}Z_{i3} + Z_{i1}Z_{i4} \tag{2.68}$$

$$A_{i4} = Z_{i1}Z_{i3} + Z_{i2}Z_{i4} \tag{2.69}$$

$$A_{i5} = Z_{i2}Z_{i3} - Z_{i1}Z_{i4} \tag{2.70}$$

$$A_{i6} = Z_{i3}Z_{i4} \tag{2.71}$$

$$A_{i7} = Z_{i3}^2 - Z_{i4}^2 \tag{2.72}$$

$$A_{i8} = Z_{i3}^2 + Z_{i4}^2 \tag{2.73}$$

where i is the pose index ranging from $i = (1, 2, \cdot, n)$. Consolidating all the G-manifold equations results in the following over-constrained homogenous linear system on equation

$$Aq = 0 \tag{2.74}$$

where

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} & \cdots & \cdots & \cdots & A_{18} \\ A_{21} & A_{22} & A_{23} & A_{24} & \cdots & \cdots & \cdots & A_{28} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ A_{n1} & A_{n2} & A_{n3} & A_{n4} & \cdots & \cdots & \cdots & A_{n8}, \end{bmatrix} \tag{2.75}$$

$$q = \begin{bmatrix} q_1 & q_2 & \cdots & q_8 \end{bmatrix}^T \tag{2.76}$$

The least square solution to this homogeneous system of equation can easily be found out using the full singular value decomposition of G-manifold coefficient matrix A. The last column of the right singular vector, which corresponds to the smallest singular value, is the least square solution. However, to generate a mechanism, at least two dyads are required. Thus, the right singular vector corresponding to the second-most and third-most smallest singular value is also taken as a possible solution. Space spanned by these three orthonormal vectors reflect a family of possible dyads which can be used for given poses.

For dyads to make physical sense, following extra C-manifold constraints are required to be satisfied

$$q_1q_6 + q_2q_5 - q_3q_4 = 0 \tag{2.77}$$

$$2q_1q_7 - q_2q_4 - q_3q_5 = 0 \tag{2.78}$$

An analytic solution to the minimization problem to satisfy the constraints gives a quartic equation. As a result, upto 4 unique dyads are generated which satisfy the initial motion synthesis problem. Combining any of the two dyads results in a solution four-bar mechanism.

As a result, the path synthesis problem is solved and prospective solutions are generated. However, solutions received are characterized by branch defects i.e. reaching all the path points in a given assembly mode might be impossible. This reduces the application of solutions in practical problems to some extent.

2.3.3 Results

Fig 2.21 shows an example of Frenet frame based path synthesis using Fourier based curve. The user-defined path input for this example is given in Table 2.37. Our algorithm finds a defect free mechanism for this path problem.

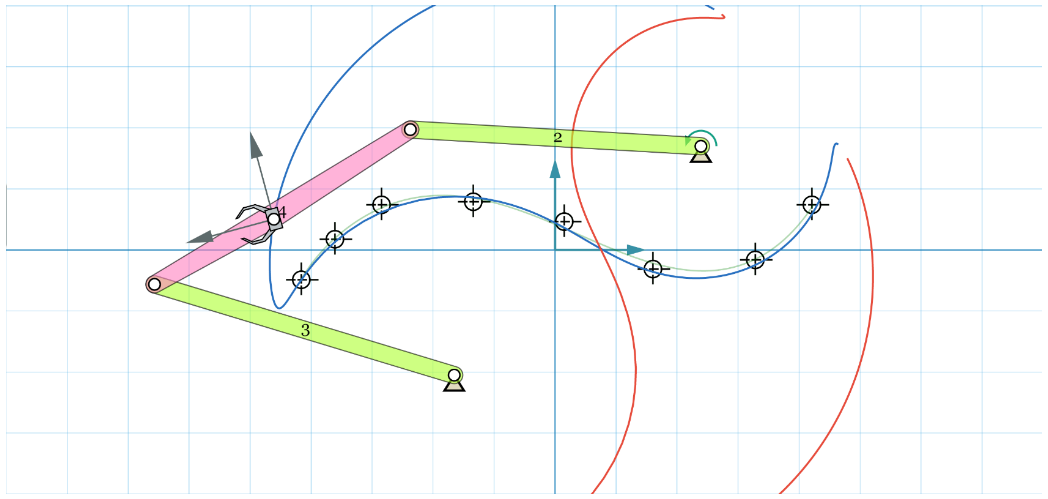


Figure 2.21: Motion synthesis using Frenet frame for Fourier based curve

Table 2.37: Input point data to analyze Frenet frame based Path Synthesis using Fourier Curves

X	Y
-4.1604	-0.4887
-3.6090	0.1754
-2.8446	0.7393
-1.3409	0.7895
0.1504	0.4637
1.6040	-0.3133
3.2832	-0.1629
4.2105	0.7393

Fig 2.22 shows an example of Frenet frame based path synthesis using B-spline curve. The user-defined path input for this example is given in Table 2.38. Our algorithm is able to find a good prospective solution. However, the synthesized mechanism does suffers from branch defect.

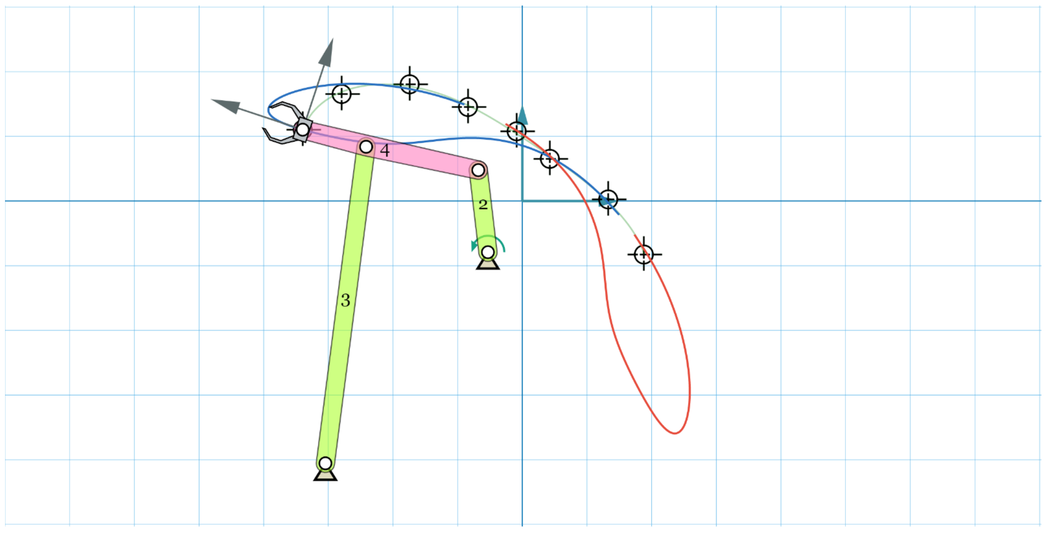


Figure 2.22: Motion synthesis using Frenet frame for Bspline curve

Table 2.38: Input point data to analyze Frenet frame based Path Synthesis using B-spline Curves

X	Y
-3.3960	1.1028
-2.7945	1.6541
-1.7419	1.8045
-0.8396	1.4536
-0.0877	1.0777
0.4261	0.6516
1.3283	0.0251
1.8797	-0.8271

It must be noted that giving extremely random input points rarely generates a mechanism which can be attributed to two main reasons. Firstly, the might not exist any mechanism which passes through those input points. Secondly, as the orientation data is not representative in any ways of four bar motion (represents physical motion), the algorithm might fail to generate an existing mechanism with same path points but different orientations.

2.4 Better shape estimation for limited data

In problem instances where the input path point data is less, the least square approximation curve can wiggle unnecessarily. As a result, target path shape can be quite different than what the user's intrusion might be. This wriggle can always exist in cases where the input points are equal to the number of Fourier basis functions being used to describe the approximation curve. The oscillations in interpolation curve have been studied as Runge's phenomenon for polynomial functions [34]. Thus, the understanding is that more sample points can help better approximate the target shape only if the order of the approximating curve remains same. If the order is also increased, the oscillations exacerbate. Consequently, the only way to minimize the wriggle is to increase the input path data before the approximation is done in a manner which is reflective of user intuition.

Interpolating subdivision algorithms [35,36] are handy recursive methods to generate more intermediate points for input. Depending on the selection of averaging mask, a variety of curve geometries can be recursively generated. The curve selected is the cubic B-spline curve and Dyn-Levin-Gregory subdivision algorithm is used to achieve this. The cubic B-splines reflect the mathematical properties of a physical spline made of some flexible material, anchored at a set of points. As a result, the choice of cubic B-splines encodes the curve shape in accordance with user intuition and his real-world experiences. Thus, extra data is generated according to the best physically relevant shape which the user might have in his mind.

The process of subdivision is implemented in two steps. First, new intermediate midpoints are generated for the given control polygon. Then, the generated points are moved to their final position using an averaging mask. The averaging mask used for Dyn-Levin-Gregory subdivision [35] is

$$r = \frac{1}{16}[-2, 5, 10, 5, -2] \tag{2.79}$$

As a result, the input data of n data-points is converted to $2n-1$ data-points after each instance of recursion.

An added advantage of using the subdivision algorithm is that target path curves defined by B-splines can be easily approximated by running the recursion enough number of times. As a result, just a small number of control points are required to calculate an approximate trigonometric curve descriptors representing the described cubic B-spline curve.

In Fig. 2.23, 2.24 an example of path synthesis problem is displayed which benefits from interpolating subdivision algorithm. In the first figure, it can be observed that synthesized mechanism is unable to pass through the given data points. However, in the second figure, using one level of interpolation points help generate a better mechanism. This can be attributed to the minimization of inherent wiggle in the target curve on using path points greater than the number of harmonics being used to describe it. Fig. 2.25, 2.26 represent another such problem. Thus, from the examples, it is apparent that the interpolating strategy is helpful in cases where target curves wiggle unnecessarily.

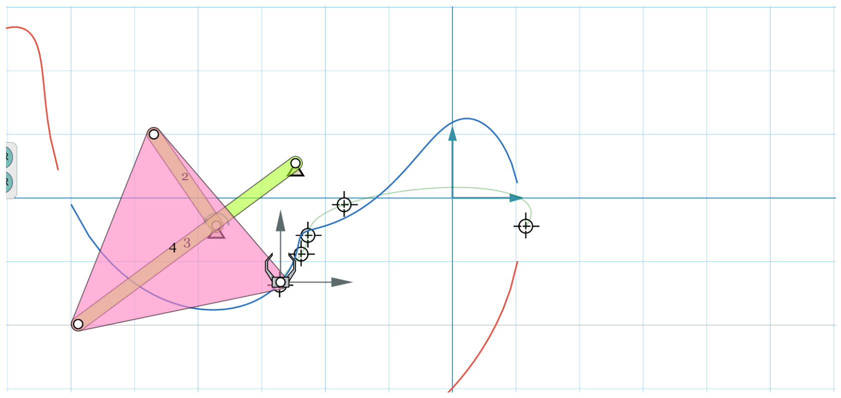


Figure 2.23: Example 1: Path synthesis without interpolation data points

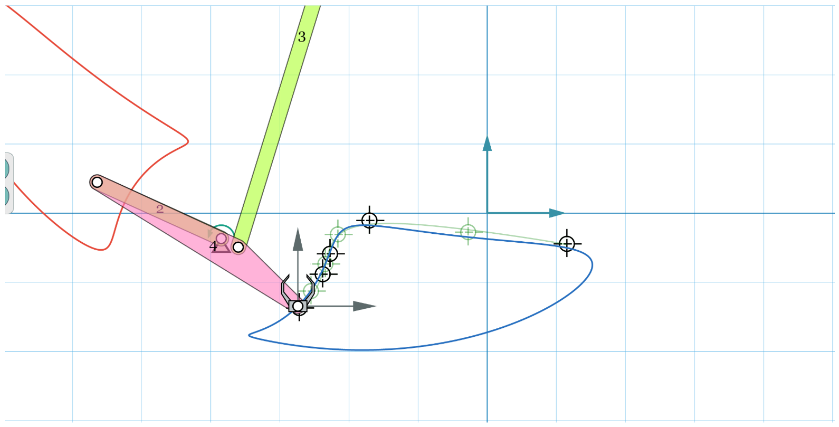


Figure 2.24: Example 1: Path synthesis with interpolation data points

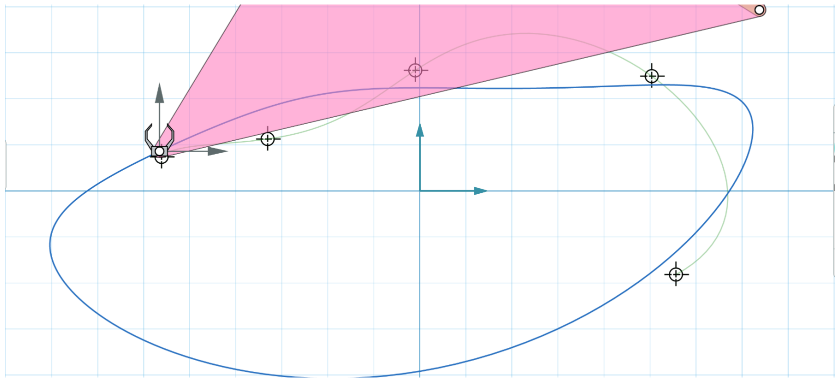


Figure 2.25: Example 2: Path synthesis without interpolation data points

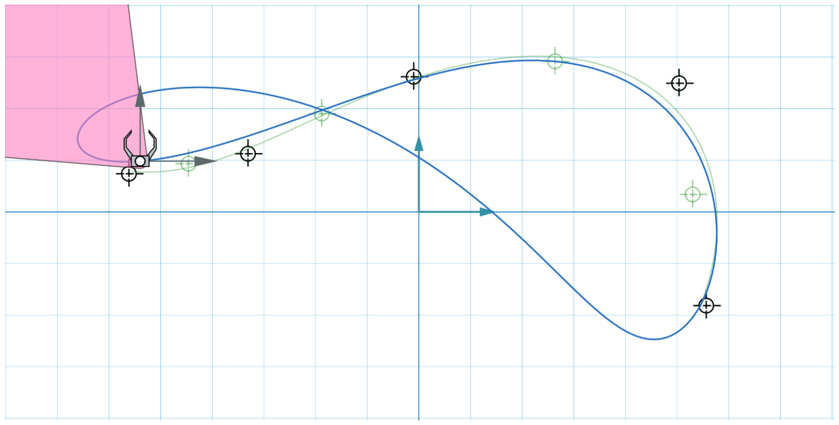


Figure 2.26: Example 2: Path synthesis with interpolation data points

Chapter 3

Mixed Synthesis

3.1 Literature review

In Chapter 2, the path problem has been transformed into a motion synthesis problem by generating orientation data. The orientations were generated in accordance with the velocity vector i.e. the coupler always remain oriented in the direction it is moving. This method is a heuristic and does not depend on the actual relation between the path and orientation for a mechanism. In this chapter, the analytic relation between the orientation data and path data using the harmonic breakdown of the loop closure equation is explored. This relation enables to attack a new set of problems which is a hybrid of path and motion synthesis, aptly called Mixed Synthesis.

It must be noted that Mixed Synthesis in our case does NOT refer to the Mixed exact- approximate path or motion synthesis where we have a set of precision and approximate constraints. Our definition of Mixed refer to a mixture of Path points and Poses.

Work on Mixed synthesis is pretty recent with its first study in [37] where it is termed as Alt-Burmester problem. The continuation of this study is presented in [38] where all possible cases for precision cases have been analyzed. However, the precision point and pose cases are restricted to just upto 9 points and 5 poses, thus ignoring a vart population of possible mixed problems. A graphical based approach has been presented in [39] to solve the mixed problem using sketcher tool inbuilt in modern CAD software. The Fourier approximation based approach proposed in this thesis can handle any number of path points or poses.

Sections 3.2, 3.3, 3.4 highlight the relationship between path and orientation Fourier descriptors for a four bar motion as shown in [40]. The four bar design variables used to specify the motion are $\{l_1, l_2, l_3, l_4, x_0, y_0, \theta_1, r, \alpha\}$ where l_1, l_2, l_3, l_4 are the link lengths, x_0, y_0 are the actuating fixed pivot coordinates, θ_1 is the fixed link orientation and r, α are coupler point parameters. δ is the angle at which moving frame has been attached to the coupler point. These mechanism parameters have been visualized in Fig. 3.1.

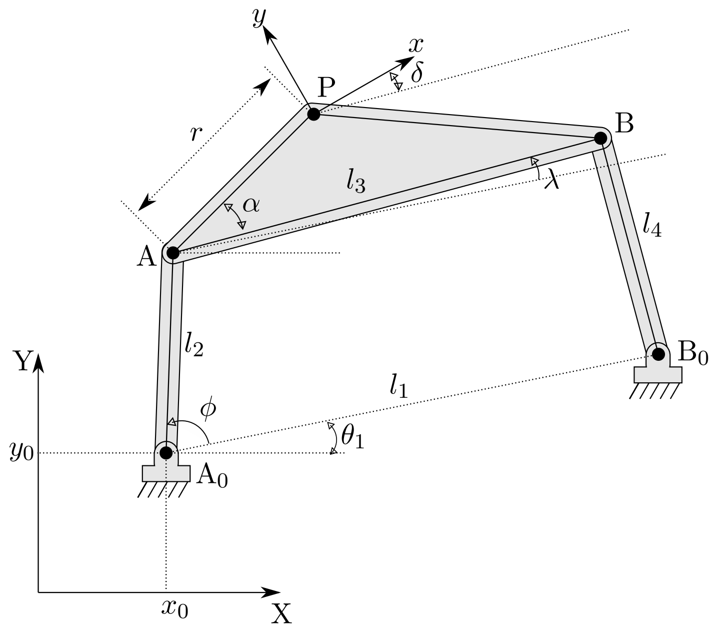


Figure 3.1: Visualization of Motion synthesis mechanism parameters

3.2 Harmonic representation of Coupler angle

The analytic equation defining the coupler angle (λ) for a four bar mechanism is represented as

$$e^{j\lambda} = \sum_{k=-\infty}^{\infty} C_k e^{jk\phi} = \sum_{k=-\infty}^{\infty} C_k e^{jkt} e^{jk\phi_0} \tag{3.1}$$

where C_k are the harmonic descriptors of coupler angle.

3.3 Harmonic representation of Path

The analytic equation which defines the path (P) of coupler point for a four bar mechanism is

$$P = A_0 + l_2 e^{i\theta_1} e^{i\phi} + r e^{i\alpha} e^{i\theta_1} e^{i\lambda} \tag{3.2}$$

Being a periodic function, it can also be represented as

$$\mathbf{P} = \sum_{k=-\infty}^{\infty} P_k e^{jk\omega t} \tag{3.3}$$

where P_k are the harmonic descriptors for path and are defined as

$$P_0 = C_0 \mathrm{r} e^{j(\alpha+\theta_1)} + (jy_0 + x_0) \qquad k = 0 \tag{3.4}$$

$$P_1 = C_1 e^{\mathrm{j}\phi_0} \mathrm{r} e^{j(\alpha+\theta_1)} + l_2 e^{\mathrm{j}\theta_1} e^{\mathrm{j}\phi_0} \qquad k = 1 \tag{3.5}$$

$$P_k = C_k e^{\mathrm{jk}\phi_0} \mathrm{r} e^{j(\alpha+\theta_1)} \qquad k \neq 0, 1 \tag{3.6}$$

3.4 Harmonic representation of Orientation

The orientation (ζ) at the coupler point for a four bar mechanism can be defined as

$$\zeta = \delta + \lambda + \theta_1 = \arg(e^{j(\delta+\lambda+\theta_1)}) \tag{3.7}$$

As λ varies periodically and other variables remain constant, the orientation can be decomposed harmonically as

$$e^{j(\delta+\lambda+\theta_1)} = \sum_{k=-\infty}^{\infty} C_k^* e^{jk\omega t} \tag{3.8}$$

where C_k^* are the harmonic descriptors for orientation and defined as

$$C_k^* = C_k e^{j(\delta+\theta_1)} e^{k\phi_0} \tag{3.9}$$

3.5 Relationship between path and orientation descriptors

Finding the relationship between the harmonic descriptors of path(P_k) and orientation(C_k^*) results in the following relationship

$$C_0^* = (P_0 + z_2) z_1 \tag{3.10}$$

$$C_1^* = (P_1 + z_3)z_1 \tag{3.11}$$

$$C_k^* = P_k z_1 \tag{3.12}$$

where

$$z_1 = \frac{e^{j(\delta-\alpha)}}{r} \tag{3.13}$$

$$z_2 = -(x_0 + jy_0) \tag{3.14}$$

$$z_3 = -(l_2 e^{j\theta_1} e^{j\phi_0}) \tag{3.15}$$

Using the above relationship, the orientation at coupler point can be defined exclusively using path harmonic descriptors as follows

$$e^{j\zeta(t)} = z_1 \left(z_2 + z_3 e^{j\omega t} + \sum_{k=-\infty}^{\infty} P_k e^{jk\omega t} \right) \tag{3.16}$$

Thus, for n path points, the system of equation describing orientation at each path point turns out to be

$$\begin{bmatrix} e^{j\zeta_1} \\ e^{j\zeta_2} \\ \vdots \\ e^{j\zeta_n} \end{bmatrix} = \begin{bmatrix} 1 & e^{j\omega t_1} & \sum_{k=p}^{-p} P_k e^{jk\omega t_1} \\ 1 & e^{j\omega t_2} & \sum_{k=p}^{-p} P_k e^{jk\omega t_2} \\ \vdots & \vdots & \vdots \\ 1 & e^{j\omega t_n} & \sum_{k=p}^{-p} P_k e^{jk\omega t_n} \end{bmatrix} \begin{bmatrix} z_1 z_2 \\ z_1 z_3 \\ z_1 \end{bmatrix} \tag{3.17}$$

Thus, the orientations at different points of a four bar coupler path is dependent on Path descriptors and three complex variables $\{z1, z2, z3\}$ which are dependent on design parameters.

3.6 Solving for unknown orientations

For a given general mixed synthesis problem, with n poses and m path points, a smooth task path with low harmonic Fourier descriptor can easily be calculated using optimum parametrization discussed before. As it has been observed [40], coupler path for a four-bar mechanism has significant magnitude only for the lower harmonics. Thus, the fitted task path is a good prospective four bar coupler curve. By using task path descriptors as coupler path descriptors, an "Exact solution to an approximate question" is being calculated.

There are multiple ways the relation given in Eqn. 3.17 can be used to convert mixed synthesis into motion synthesis problem. The aim is to find the values of $\{z_1, z_2, z_3\}$ so that all the unknown orientations can be found out. Additional external constraints which will define unique values for unknown orientations are required.

One way is using three orientations to constrain z_1, z_2, z_3. Physically, it makes sense that the user might know orientations at the initial and final position and an additional location in between the motion.

Location of actuating fixed pivot and two orientations is another way of constraining the unknown orientations. The user can specify the orientations at first and the last point and choose the crank fixed pivot according to his convenience.

If the user has coupler link constraints, $\{r, \alpha, \delta\}$ can be fixed to find the value of z_1. Subsequently, only two more orientations are required to fully define the system of equation.

If there is a physical application where constraining variables like l_2, θ_1, ϕ_0 is a part of the problem statement, the system of equation can be constrained in some other way according to the problem. l_2 can be fixed to defined to scale the mechanism according to the task. θ_1 defines the fixed link angle and can be selected to geometrically constrain fixed pivots at specified angle from the horizontal.

However, it is improbable that the user has a limited choice of most or all mechanism parameters because that would imply a four bar mechanism has already been mostly selected. Over-specifying a mechanism even before the synthesis problem drastically decreases the space of possible paths that can be synthesized.

To summarize, the three possible mixed constraints are

1. Specify actuating fixed pivot i.e. $\{x_0, y_0\}$
2. Specify coupler parameters i.e. $\{r, \alpha, \delta\}$
3. Specify scale, orientation and initial angle i.e. $\{l_2, \theta_1, \phi_0\}$

When the number of orientations known is more than 3, a least square solution for z_1, z_2, z_3 can easily be calculated by finding complex Singular Value Decomposition. Since complex SVD is not as widespread in use as its real counterpart, libraries for complex SVD computation on a web-based platform are lacking. Thus, complex system of equation in Eqn. 3.17 is reduced to an equivalent real system of equation in accordance with [41]. The $K1$ formulation has been used in the implementation. According to the formulation, a complex system of equation

$$(A + iB)(x + iy) = b + ic \tag{3.18}$$

can be written as a real system of equation

$$\begin{bmatrix} A & -B \\ B & A \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ c \end{bmatrix} \tag{3.19}$$

Finding least square solution to this equivalent real system of equations gives the solution to original complex problem and values of z_1, z_2, z_3 can be calculated in over-constrained cases.

Once the values of z_1, z_2, z_3 are known, orientations at all the unknown path points can be found out and the motion synthesis algorithm can be used to calculate dyads. It must be noted that with constraints on orientation angle at some path points, the problem is essentially a hybrid of path and motion synthesis. Thus, our problem definition now incorporates both poses and path points as input and is called as mixed synthesis.

3.7 Generalized framework for Motion, Path, and Mixed Synthesis

The power of methodology outlined above is that it can handle both Motion and Mixed synthesis problems within it. A case by case discussion covering various permutations of $(0, 1, \cdots, m)$ path points and $(0, 1, \cdots, n)$ poses is displayed in Table 3.1

In the table, MOC = Motion Synthesis constraint [30], MIC = Mixed Synthesis Constraint, FD = Fully Defined. The * refers to conditions where a Fourier task curve with just four points needs to be fitted and would have unsymmetrical descriptors.

Motion Synthesis constraints refer to the geometric constraints outlined in [30] which are used to specify the position of fixed or moving pivots using line or point constraints. Mixed Synthesis Constraint have been described in detail in the previous section and involves constraints on actuating pivot, coupler dimensions, and other mechanism parameters. Fully Defined entails

Table 3.1: Case by case Mixed Synthesis using generalized framework

		Path Points					
		0	1	2	3	4	m
Poses	0	X	X	X	X	FD*	FD
	1	X	X	X	2 MIC*	2 MIC	2 MIC
	2	X	X	1 MIC*	1 MIC	1 MIC	1 MIC
	3	2 MOC	1 MOC*	FD	FD	FD	FD
	4	1 MOC	FD	FD	FD	FD	FD
	5	FD	FD	FD	FD	FD	FD
	n	FD	FD	FD	FD	FD	FD

that no extra constraints are needed to exactly or least square solve the mixed synthesis Eqn. 3.17.

Except for the cases involving zero poses, Eqn. 3.17 can be used to solve the generalized problem. When the synthesis problem has zero pose, the usual path synthesis algorithm is used. It must be noted that except for the case where there are no poses, the synthesis is both Type and Dimensional in nature. However, cases with only path points are branch defect free.

The algorithm described to solve the Generalized Path-Pose Mixed problem can be summarized as follows

```
Input: Path points and Poses
if n(Path points+poses)$\geq$ 4 then
    Calculate Task curve fourier descriptor using modified
     reparemetrization
end
if n(Pose)=0 then
    Fourier based Path synthesis
end
else
    Combined Motion+Path synthesis
end
Output: Synthesised mechanism
```

Algorithm 5: Pseudo-code for Unified Motion, Path and Mixed Synthesis

The only currently known facts characterizing descriptors of Four-bar motion and path are

- Magnitude of higher harmonics decrease very fast
- Ratio between the Path and orientation Fourier descriptors is constant

Both of these rules are being exploited in the proposed algorithm. Thus, this attempt can be claimed to be the best that can be achieved with the existing state of knowledge.

3.8 Results

Fig. 3.2, 3.3 shows a mixed synthesis problem which has 3 poses and 10 path points. Coincidentally, the mechanism synthesized does not suffer from branch defect. The point and pose data, which define this mixed problem, are displayed in Table 3.2.

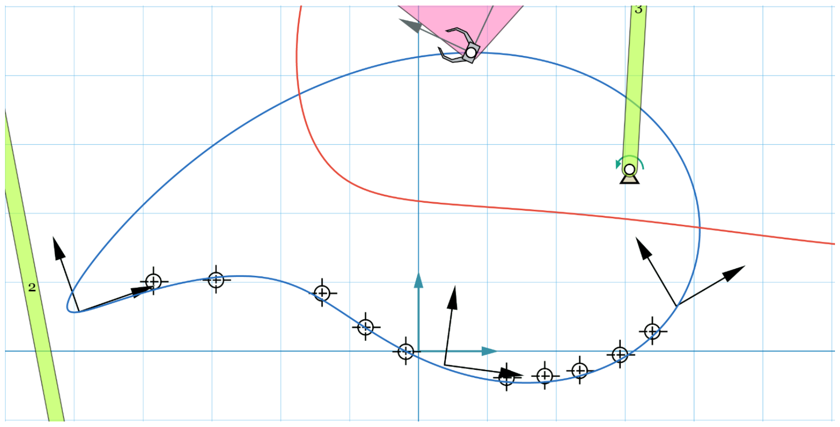


Figure 3.2: Example 1: Mixed synthesis for Three poses and ten Path points

Fig. 3.4, 3.5 shows another example of mixed synthesis problem having 3 poses and 4 path points. The synthesized mechanism for this problem suffers from a branch defect. The point and pose data, which define this mixed problem, are displayed in Table 3.3.

One of the major advantages of mixed synthesis is the additional flexibility it imparts to users. An over constrained motion problem can be solved in a least square manner using existing algorithms. However, the solution provided are usually so bad, they don't approximate the motion at all. An

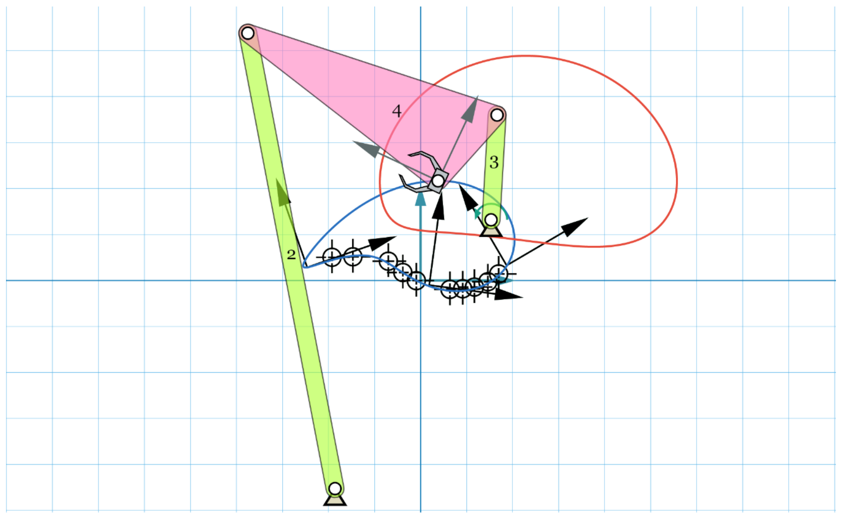


Figure 3.3: Example 1: Mixed synthesis- zoomed out

Table 3.2: Example 1: Mixed Synthesis problem Input constraint data

Constraint	X	Y	ζ
Pose 1	-9.853	1.139	19.23
Point 2	-7.697	2.016	
Point 3	-5.881	2.063	
Point 4	-2.802	1.678	
Point 5	-1.539	0.692	
Point 6	-0.369	-0.0153	
Pose 7	0.754	-0.400	352.37
Point 8	2.555	-0.769	
Point 9	3.664	-0.723	
Point 10	4.680	-0.569	
Point 11	5.850	-0.107	
Point 12	6.789	0.569	
Pose 13	7.482	1.3086	30.37

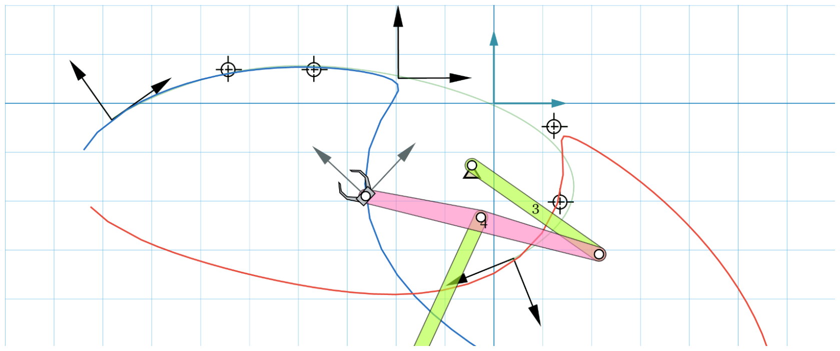


Figure 3.4: Example 2: Mixed synthesis for Three poses and four Path points

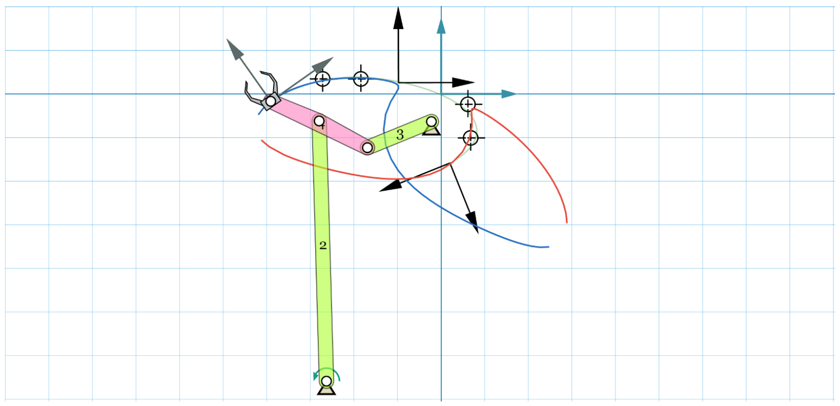


Figure 3.5: Example 2: Mixed synthesis- zoomed out

Table 3.3: Example 2: Mixed Synthesis problem Input constraint data

Constraint	X	Y	ζ
Pose 1	-7.819	-0.338	35.21
Point 2	-5.438	0.689	
Point 3	-3.684	0.689	
Pose 4	-1.954	0.513	0.23
Point 5	1.228	-0.476	
Point 6	1.353	-2.017	
Pose 7	0.401	-3.170	201.94

example of such a over constrained motion problem is shown in Fig. 3.6. It can be observed how that the solution is useless. Now, with the help of mixed analysis, relaxing the problem by neglecting most of orientation data reduces the problem to the first example discussed in Fig. 3.2. This mixed problem has a good approximate solution. The difference in the usefulness of mechanism using different approaches is huge. Thus, if a designer has flexibility over orientations, he can now use the Mixed synthesis framework to find good possible mechanisms to his problems.

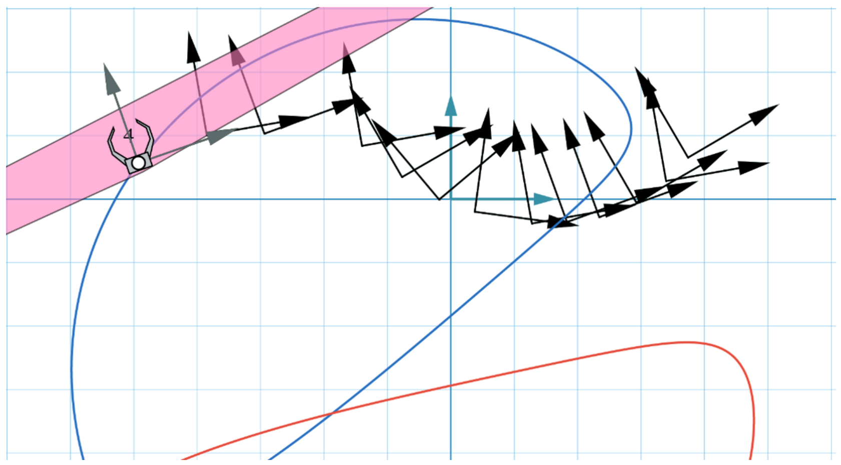


Figure 3.6: Example of over-constrained Motion Synthesis problem for thirteen Poses

Chapter 4

Coupled Multi-Degrees of Freedom Mechanisms

The focus till now has been to synthesize closed-loop mechanisms. Synthesizing another class of mechanisms to solve the path problem will now briefly be explored.

4.1 Single degree-of-freedom coupled serial chain mechanisms

A number of mechanisms have been used for traversing a specified path. One of the prominent ones, as discussed before, is the four-bar linkage. It can generate a rich family of paths using a single actuator. However, it can prove to be unsuitable in a cluttered environment caused by interference of the links with each other and with the environment. In such environments, serial chain manipulators are the preferred mechanism type but their requirement of multiple articulations and coordinated control makes them a more complicated mechanism to implement practically. Tendon-driven serial-chain manipulators permit the relocation of the actuators to the base of the manipulator, thereby reducing the inertia of the moving parts, but still require at least as many actuators as degrees of freedom. It would be preferred if a serial chain could be actuated using only a single input as it would simplify the control scheme.

Single Degree-of-freedom Coupled Serial Chain (SDCSC) [42–44] mechanisms offer such an alternative. Such mechanisms are constructed by me-

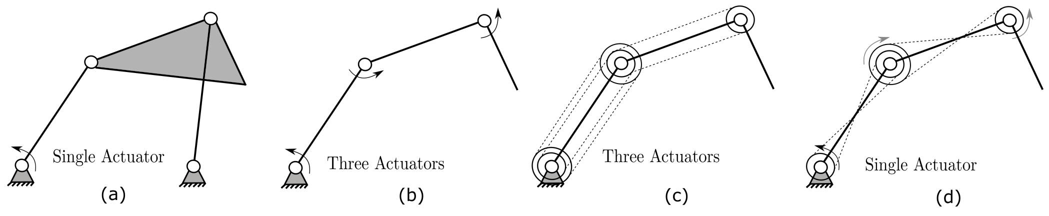


Figure 4.1: Different mechanism configurations: (a) single degree-of-freedom four-bar linkage; (b) multi-degree-of-freedom conventional, serial-chain linkage; (c) tendon-driven serial chain linkage; and (d) single-degree-of-freedom coupled serial chain mechanism

chanically coupling the rotations of the links of an n-link, n-d.o.f. serial chain manipulator using cable and pulley drives or by gear-trains. Each coupling between two successive joint rotations reduces one degree-of-freedom and repeated coupling reduces the overall degrees of freedom of the manipulator to one. Thus, The resulting SDCSC mechanisms combine the simplicity of single-degree-of-freedom control of closed-loop linkages with the modularity, compactness and reduced interference of serial chains. SDCSC mechanisms can be used to describe any complex path trajectory exactly if the total number of links and their size is not a restriction. Fig. 4.1 displays the different classification of mechanisms for more clarity.

4.2 Synthesis Algorithm

A SDCSC mechanism is characterised by the total number of coupled links in it. The trajectory of the end effector can be fully defined by defining length, rpm and initial position of each link. Thus, to synthesize SDCSC mechanism for a given path, all these parameters need to be computed.

Firstly, the set of path point which the user inputs are used to fit a trigonometric curve characterized by Fourier basis. The coefficients whose linear combination with basis functions representing this path are loosely termed as its Fourier descriptors. Exact methodology to calculate these descriptors has been discussed in previous sections. The number of harmonics considered is dependent on the approximation requirement for the problem.

These calculated Fourier coefficients can easily be mapped to SDCSC. The important observation here is that the Fourier Basis functions represent the circular rotation at a specific rpm which represents a revolute link in the physical world. The summation of basis represents the superimposition of the motions of each link hence creating a serial chain. The complex Fourier coefficient attached to each Fourier basis thus represents the dimensions of that revolute link. The total number of Fourier coefficients is the quantity of links in the resultant coupled serial chain. This is in accordance with the intuition that more the links in the mechanism, better the given path can be approximated.

Representing the above mathematically, let a Fourier descriptor for k^{th} harmonic be $\alpha_k = a_k + ib_k$, then length on k^{th} link $l_k = \|\alpha_k\|$ and initial phase $\phi_k = arg(\alpha_k)$. Also, the (x, y) position of one end relative to another is $(Re(\alpha_k), Im(\alpha_k))$. The rpm for nth harmonic is determined as n times rpm of base harmonic. Thus, we have calculated link length, phase and rpm for each link in synthesized SDCSC.

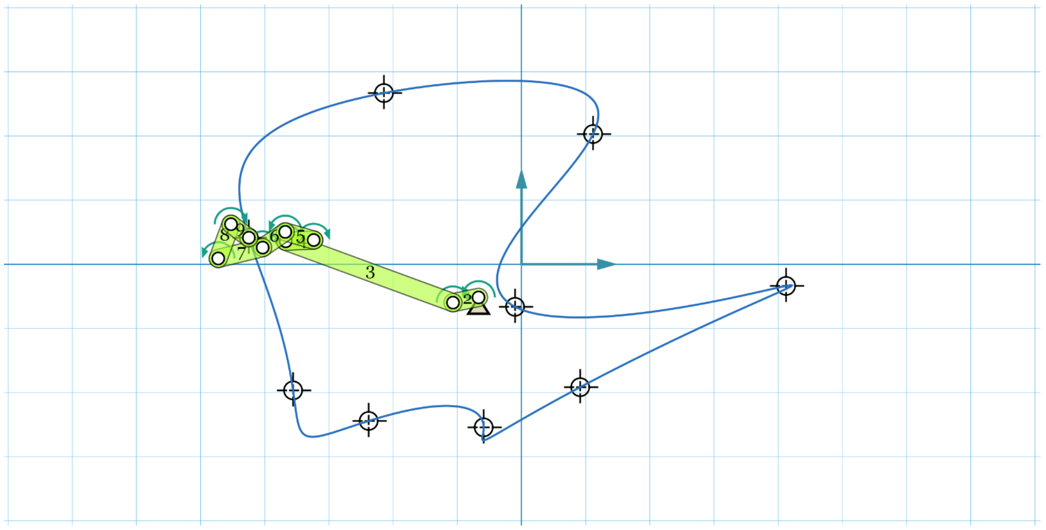


Figure 4.2: Synthesis of a SDCSC with eight revolute joints using uniform parameterization

Fig. 4.2 shows an example of synthesized 4R serial coupled mechanism. The input path point data is given in Table 4.1. The rpms of links alternate between clockwise and anticlockwise for subsequent links. The ratios of rpm for each link is defined. Thus, the absolute value of each rpm is dependent on the value of rpm for the first link. The target path for this curve was approximated using 5 Fourier basis namely base, first and second harmonics.

Table 4.1: SDCSC Synthesis problem Input constraint data

X	Y
−4.2481	0.4135
−2.1429	2.6692
1.1153	2.0301
−0.1003	−0.6642
4.1228	−0.3383
0.9148	−1.9173
−0.5890	−2.5439
−2.3810	−2.4436
−3.5589	−1.9674

4.3 Synthesis using Optimum Parametrization

Using optimal parametrization proposed previously to derive the Task curve which has the smoothest path, it is possible to get a considerably better mechanism. The example displayed in Fig. 4.2 has been parameterized using the uniform scheme. Optimum parameterization is used to find a better task path for the same input and the resultant mechanism is displayed in Fig. 4.3. Task curve descriptors are given in Table 4.2. We notice that the minimized arc length task path received from the optimum parameterization is smoother than the one obtained from uniform parameterizations. Thus, optimum parameterization helps reduce the unintended waviness of task curve successfully.

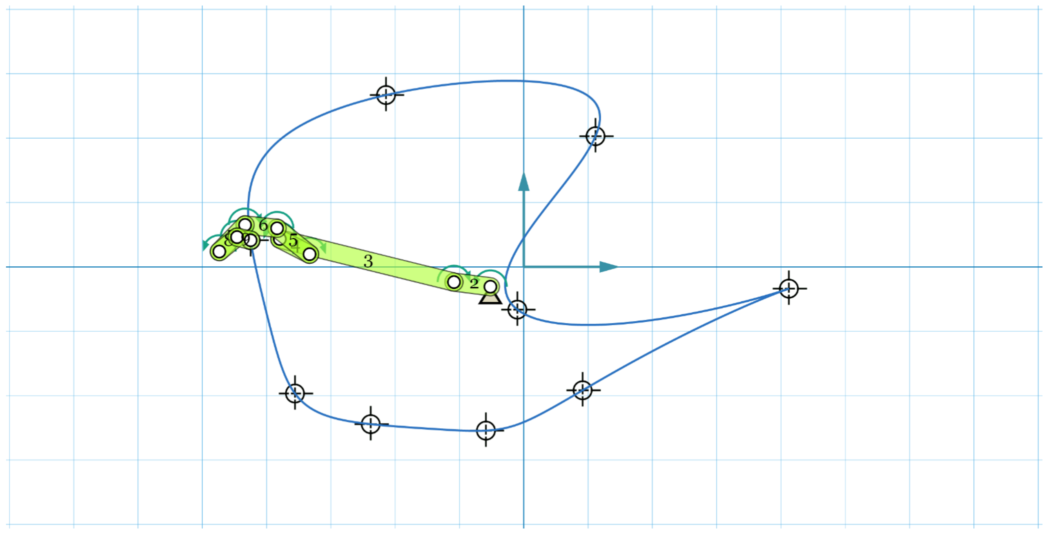


Figure 4.3: Synthesis of a SDCSC with eight revolute joints using optimum parameterization

Table 4.2: Task path Descriptor Data when using different Parameterizations for SDCSC synthesis

	Task path $\alpha = 0$		Task path $\alpha = optimum$	
Descriptor	Complex Value	Magnitude	Complex Value	Magnitude
p = −4	$0.2779 - 0.2105i$	0.3486	$0.2115 - 0.04888i$	0.2171
p = −3	$-0.6928 - 0.1695i$	0.7132	$-0.4002 - 0.4132i$	0.5752
p = −2	$-0.4429 + 0.1275i$	0.4609	$-0.5051 + 0.4028i$	0.6461
p = −1	$-2.607 + 0.9566i$	2.7766	$-2.709 + 0.6696i$	2.7902
p = 0	$-0.6691 - 0.5177i$	0.8460	$-0.5194 - 0.3082i$	0.6040
p = 1	$-0.3981 - 0.0814i$	0.4063	$-0.5667 + 0.07181i$	0.5712
p = 2	$0.4362 + 0.01954i$	0.4367	$0.4628 - 0.2367i$	0.5198
p = 3	$-0.3517 - 0.2449i$	0.4286	$-0.4979 + 0.05205i$	0.5007
p = 4	$0.1989 + 0.5339i$	0.5698	$0.2756 + 0.2242i$	0.3553

Chapter 5

Software Review

Many existing kinematic design and synthesis softwares are being used by professionals and researchers. However, none of them seem to have widespread adoption. Some of the major ones have been discussed in this section. Special emphasis has been placed on determining the extent of ability, to which a software can simulate and synthesize mechanism.

5.1 Autodesk ForceEffect Motion

ForceEffect [45] is a mechanical design application developed by Autodesk Inc. It was available for iOS, Android, and Desktop platforms. It had two versions called Mechanical and Motion. Fig. 5.1 displays the dashboard of Autodesk ForceEffect Motion. Autodesk retired the app in 2016. It used to provide rich functionality for the simulation and analysis of multi-link mechanisms. However, it lacked any synthesis functionality.

Thus its abilities can be summarized as

- Analysis
    1. Static load analysis
    2. N-bar one degree of freedom mechanisms simulation (revolute and prismatic joints).
    3. Graphing the position, velocity, and accelerations.
- Synthesis
    1. No synthesis ability.

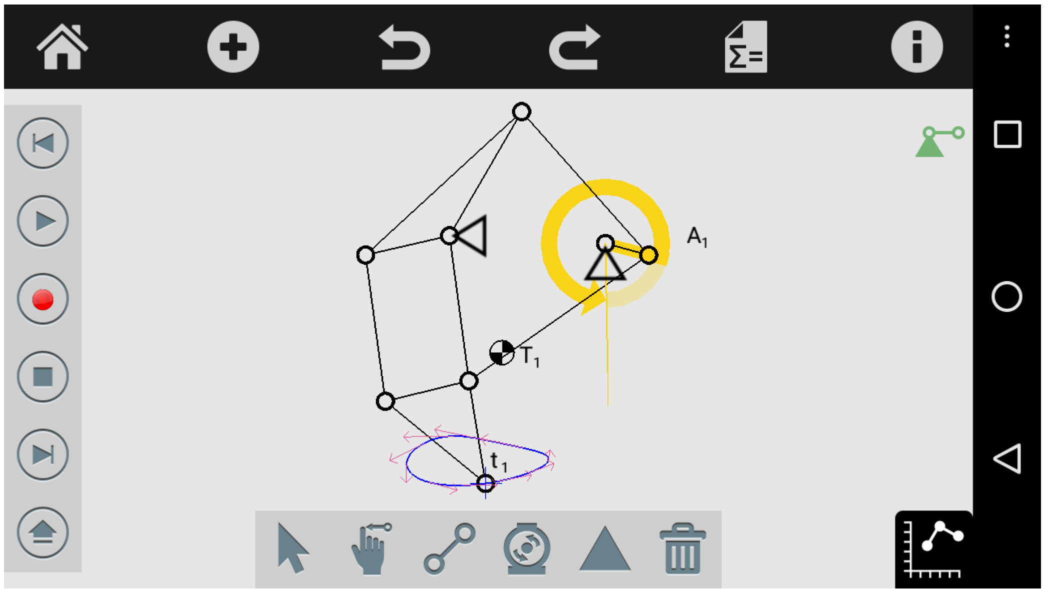


Figure 5.1: Autodesk ForceEffect Motion Dashboard

5.2 Linkage

Linkage [46] is a Windows program developed by David Rector. Its focus is on quick prototyping of linkage mechanisms. It has an intuitive UI and feels like a well-built CAD software. Fig. 5.2 displays the dashboard of Linkage. The strength of Linkage lies in its ability to simulate n-bar mechanisms. However, due to the use of dyadic decomposition, it cannot be used to simulate mechanisms with more complex topologies like involvement of triads. As the purpose of software is a quick analysis of mechanisms, it lacks the ability to do path, motion or function synthesis.

Thus, its abilities can be summarized as

- Analysis

  1. N-bar multi-degree of freedom mechanisms simulation (revolute and prismatic joints). However, only dyadic decomposable mechanisms can be simulated.
  2. Can simulate gears and chains
  3. Supports multiple simultaneous simulations

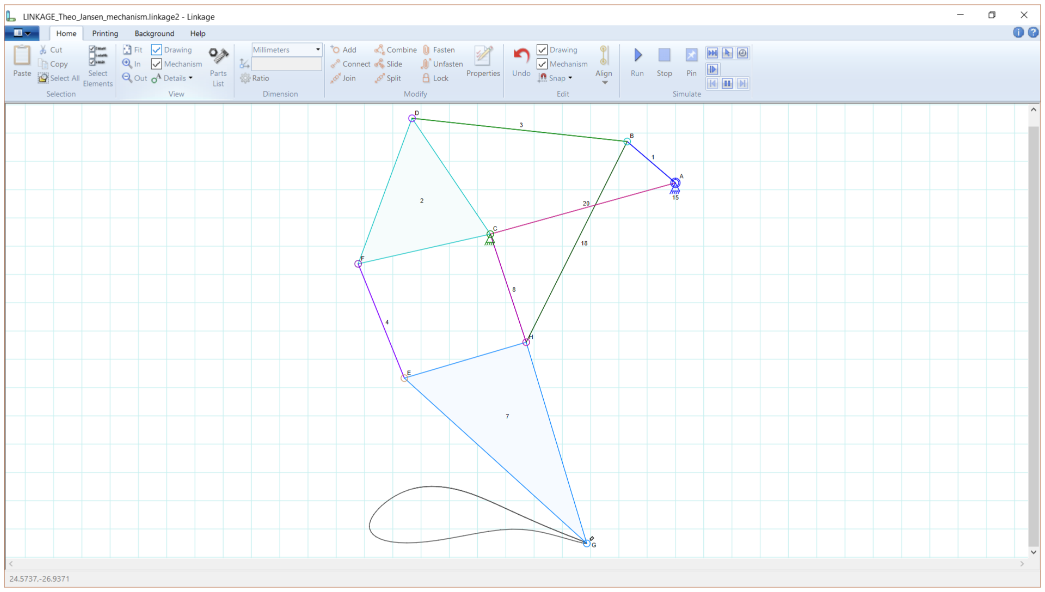


Figure 5.2: Linkage Dashboard

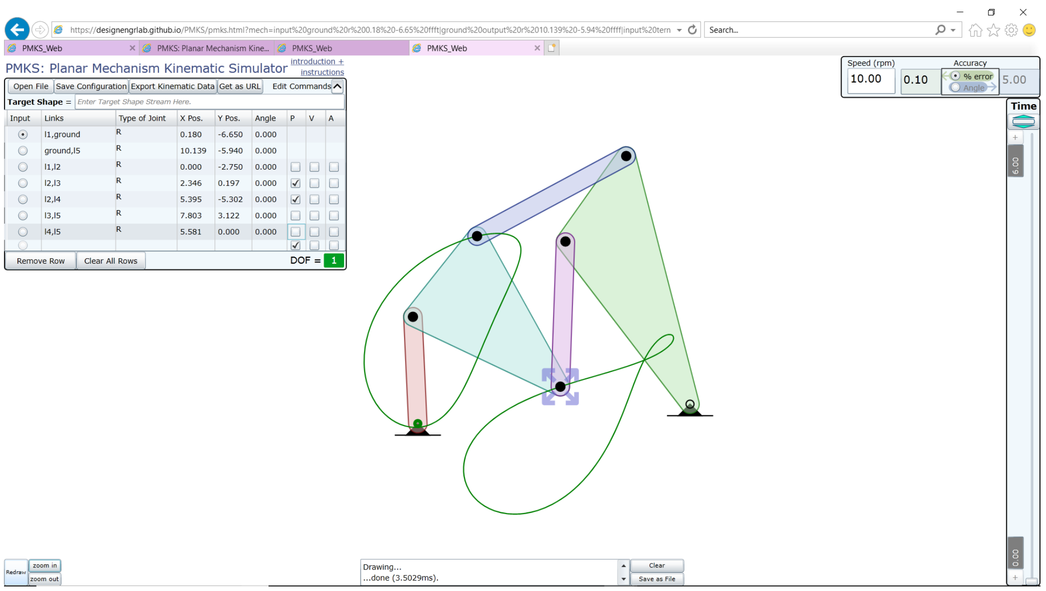
- Synthesis
    1. No synthesis ability.

5.3 Planar Mechanism Kinematic Simulator (PMKS)

PMKS [47] is a web application developed by Design Engineering Lab at Oregon State University. Fig. 5.3 displays the dashboard of PMKS. Its purpose is to simulate one degree of freedom planar mechanisms with multiple rigid bodies. It can calculate quick and accurate results for the position, velocity, and acceleration. It can handle non-dyadic mechanisms using a novel approach. Synthesis of mechanism is not the focus of this application.

Thus, its abilities can be summarized as

- Analysis
    1. N-bar one degree of freedom mechanisms simulation (revolute and prismatic joints).


PMKS: Planar Mechanism Kinematic Simulator

Target Shape = Enter Target Shape Stream Here.

Input	Links	Type of Joint	X Pos.	Y Pos.	Angle	P	V	A
●	l1,ground	R	0.180	-6.650	0.000			
○	ground,l5	R	10.139	-5.940	0.000			
○	l1,l2	R	0.000	-2.750	0.000	☐	☐	☐
○	l2,l3	R	2.346	0.197	0.000	✓	☐	☐
○	l2,l4	R	5.395	-5.302	0.000	✓	☐	☐
○	l3,l5	R	7.803	3.122	0.000	☐	☐	☐
○	l4,l5	R	5.581	0.000	0.000	☐	☐	☐
○						✓	☐	☐

DOF = 1

Speed (rpm) 10.00

Accuracy 0.10 [✓ % error] Angle 5.00

Drawing...
...done (3.5029ms).

https://designengrlab.github.io/PMKS/pmks.html?mech=input%20ground%20r%200.18%20-6.65%20ffft|ground%20output%20r%2010.139%20-5.94%20ffff|input%20tern
Search...
PMKS_Web
PMKS: Planar Mechanism Kine...
PMKS_Web
PMKS_Web
introduction + instructions
Open File
Save Configuration
Export Kinematic Data
Get as URL
Edit Commands
Remove Row
Clear All Rows
Time
6.00
0.00
Redraw
zoom in
zoom out
Clear
Save as File

Figure 5.3: Planar Mechanism Kinematic Simulator Dashboard

2. Calculates position, velocity and acceleration data.

- Synthesis

  1. No synthesis ability.

5.4 Synthesis and Analysis of Mechanisms (SAM)

SAM [48] is a feature-rich PC software developed by Artas Engineering. Fig. 5.4 displays the dashboard of SAM. Although its UI seems dated and nonintuitive, it boasts both mechanism analysis and synthesis capabilities. It has functionality to supports n-bar simulation for planar linkages with both Revolute and Prismatic joints. SAM offers a set of design wizards to synthesize four-bar mechanisms for motion and function generation. A wizard for path synthesis is not present.

The software does have the ability to uses Simplex Method or evolutionary algorithm for the local optimization. It lacks global optimization methods. Using these methods, path optimization of existing mechanism can possibly be modeled. However, it is up to the user to define the objective function.

Thus, features of SAM can be summarized as

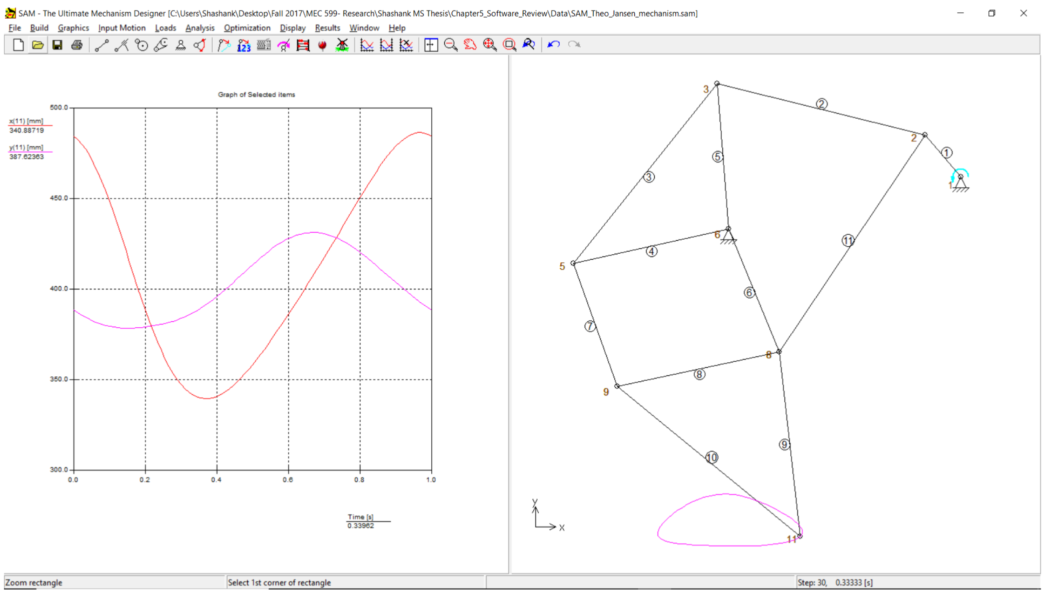

SAM - The Ultimate Mechanism Designer [C:\Users\Shashank\Desktop\Fall 2017\MEC 599- Research\Shashank MS Thesis\Chapter5_Software_Review\Data\SAM_Theo_Jansen_mechanism.sam]
File Build Graphics Input Motion Loads Analysis Optimization Display Results Window Help
Graph of Selected items
x(11) [mm]
340.88719
y(11) [mm]
387.62363
500.0
450.0
400.0
350.0
300.0
0.0
0.2
0.4
0.6
0.8
1.0
Time [s]
0.33962
Zoom rectangle
Select 1st corner of rectangle
Step: 30, 0.33333 [s]

Figure 5.4: Synthesis and Analysis of Mechanisms Dashboard

- Analysis

  1. N-bar multi degree of freedom mechanisms simulation upto 10 simultaneous inputs (revolute and prismatic joints).
  2. Detailed graphing of position, velocity and acceleration data.
  3. Can simulate gears and chains
  4. Force analysis

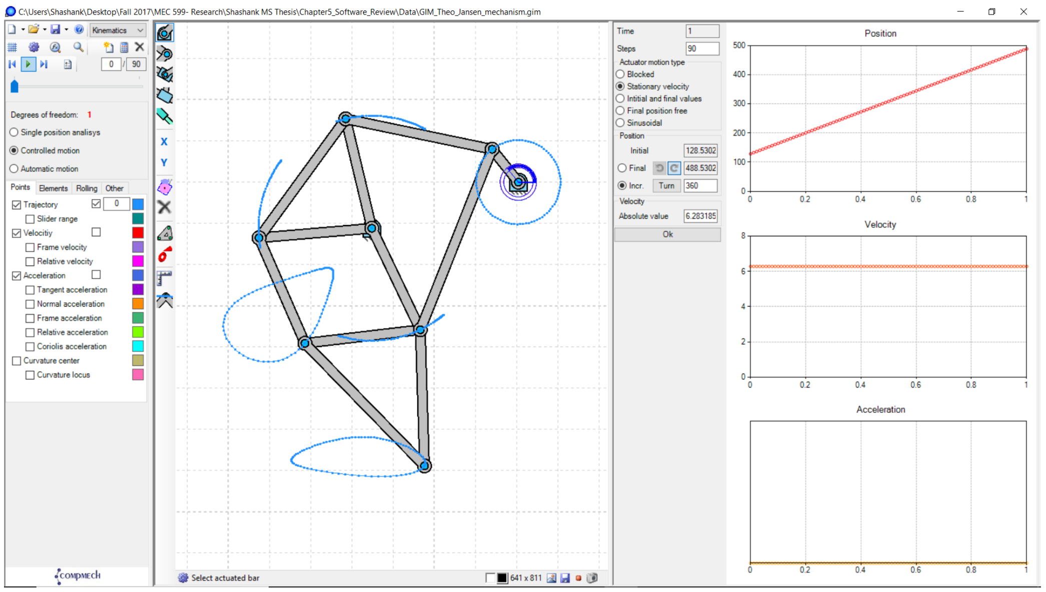
- Synthesis

  1. Path synthesis possible but difficult.
  2. Motion synthesis of four-bar mechanism for 3 poses using graphical method.
  3. Function generation of four-bar mechanism for ≥ 3 input-output angle pair.
  4. Exact and approximate straight line mechanisms

5.5 GIM

GIM [49] has been developed by COMPMECH research group in the University of the Basque Country UPV/EHU. Fig. 5.5 displays the dashboard of GIM. Mechanisms with n-ary elements joined by revolute and prismatic pairs can be simulated in GIM, The position problem is solved iteratively using a numerical method.

Although the focus of GIM is analysis of mechanisms, it does have some synthesis capabilities. It uses graphical method to do 3,4,5 precision point synthesis. Motion synthesis for 3,4 pose is also possible in the software. Function generation for three input-output angles can be done. All of these synthesis options are very limited and doesn't give the user freedom to specify an arbitrary number of inputs. Also, it is not freely distributed which restricts its usage.


C:\Users\Shashank\Desktop\Fall 2017\MEC 599- Research\Shashank MS Thesis\Chapter5_Software_Review\Data\GIM_Theo_Jansen_mechanism.gim

Kinematics

0 / 90

Degrees of freedom: 1

( ) Single position analisys
(●) Controlled motion
( ) Automatic motion

Points | Elements | Rolling | Other

☑ Trajectory ☑ 0 🟦
☐ Slider range 🟩
☑ Velocitiy ☐ 🟥
☐ Frame velocity 🟪
☐ Relative velocity 🟪
☑ Acceleration ☐ 🟦
☐ Tangent acceleration 🟪
☐ Normal acceleration 🟧
☐ Frame acceleration 🟩
☐ Relative acceleration 🟩
☐ Coriolis acceleration 🟦
☐ Curvature center 🟨
☐ Curvature locus 🟥

Time: 1
Steps: 90

Actuator motion type
( ) Blocked
(●) Stationary velocity
( ) Intitial and final values
( ) Final position free
( ) Sinusoidal

Position
Initial: 128.5302
( ) Final: 488.5302
(●) Incr. Turn: 360

Velocity
Absolute value: 6.283185

Ok

Select actuated bar

641 x 811

Figure 5.5: GIM Dashboard

- Analysis
    1. N-bar multi degree of freedom mechanisms simulation(revolute and prismatic joints).
    2. Verbose graphing of position, velocity and acceleration data.

3. Can simulate cams, gears and chains
4. Static analysis

- Synthesis

  1. Path synthesis of four bar mechanism for 3,4,5 precision points.
  2. Motion synthesis of four bar mechanism for 3,4 poses.
  3. Function generation of four bar mechanism for 3 input-output angle pair.

5.6 Limitations of existing software

After analyzing the most prevalent kinematic softwares, these are the common drawbacks observed

- None of the softwares is able to do n-point path synthesis for four-bar mechanisms.
- None of the softwares is available across all platforms i.e. on PC, Mac, and Mobile. Most of them require initial setup and installation.
- Majority of softwares have not been updated in a long time. They have old GUI's which is difficult to understand and use. This leads to a steep learning curve for newcomers.

To the best of authors knowledge, this thesis outlines the first attempt at n-point Path Synthesis using web-based implementation. Also, this is the first attempt to implement Generalized Mixed Synthesis.

Chapter 6

Application Design Guiding Principles

This section outlines the design principles which have guided the course of application development and their immense importance. The goal is to create a kinematic analysis and syntheses software which is accessible everywhere, modular, scalable, and fast. It also needs to be easy to understand and use.

The software has been created upon MotionGen, a four-bar analysis and motion synthesis software. Henceforth, this newly developed software implementation of the previous MotionGen is referred to as MotionGen 2.0. Finer details comparing features of old software to new software has been discussed in next section. This section delves into the design philosophy incorporated.

6.1 Design Paradigm: MVC

The Model-View-Controller design pattern (MVC) is a global high-level pattern which acts upon the structure of software and basically separates the core logic from the rest of user interface. Using MVC design on an object-oriented programing language is especially beneficial to a codes structure. MVC self-enforces a structure on the codebase, making it more reusable and its communication channels better defined. As a result, the software is extremely adaptable to changes i.e. new functionality can easily be added or old obsolete functionality removed. Thus, MVC facilitates better maintenance and a robust program structure.

The MVC design pattern directs the programmer to group all objects into three types of objects: model, view, and controller. This categorization is done based on an object's role in software. Once the classification is done, MVC architecture has a set of well-defined guidelines which needs to be followed to standardize the exchange of data between these three individual types of object types. Thus, to follow the MVC framework, it is of paramount importance that the application designer chooses each system object to fall into one of three basic object types.

6.1.1 Model

The model-objects is tasked with managing the core functionality of the application and stores all the data which defines the state of the application at any given point in time. It communicates with view objects who query it for information related to present system state. The controller object can instruct the model object to change its state based on user intent.

For MotionGen2.0, the salient functionality is twofold, namely analysis and synthesis. Model objects containing analysis data include Links, Joints, and Linkages while the synthesis data is held in Points, Lines, Poses, and Path. Multiple instances of these model objects represent the state of workspace at any given point in time.

6.1.2 View

The view-objects is tasked with managing the display or communication of information to the user. It communicates with model objects and gets the necessary data required by the user by querying state data. It then presents this information usually in an audio-visual manner.

For MotionGen 2.0, HTML layered canvas acts as view objects on which graphics are drawn. These graphics are dependent on the data obtained from the models representing system state.

6.1.3 Controller

The controller sole purpose is to decipher the mouse, keyboard and touch inputs from the user. It then processes this information and instructs model and view objects to make appropriate changes. Thus, it provides the interface between the user and the application.

In MotionGen 2.0, the controller is tasked with deciphering touch, mouse or keyboard input and deciding which action to carry out. This includes changes to model objects or view objects.

6.1.4 Passive model

There are two types of MVC models, passive and active. In a passive model, only the controller objects can instruct the model object to change. However, in an active model, both view and controller objects have the ability to manipulate the model object.

Passive model is being used as it is simpler and does not require an extra observer class to keep track of all changes happening to system state. Thus, when the controller modifies the model object, it handles the responsibility of instructing the view object to update itself accordingly. The model in this scenario is completely independent of the view and the controller, which means that there is no way for the model to report changes in its state.

Fig. 6.1 displays the flow of information through the software. Note that model objects don't push data themselves. It needs to be pulled by the view or pushed by the controller. As a result, the model has the capability to be a self-sufficient unit while view and controller are dependent on the model for information. This separation is what empowers the MVC approach and allow independent programming and testing of core logic from user interface.

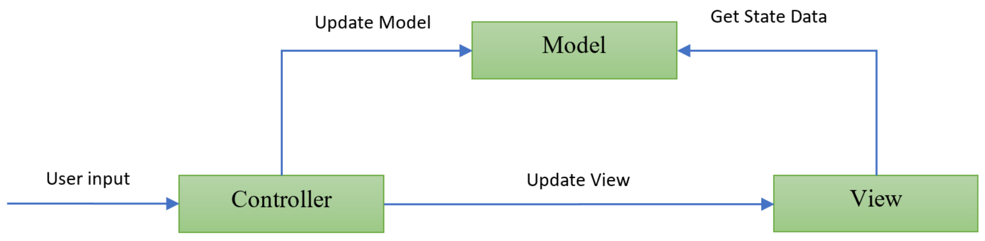


Figure 6.1: Passive MVC Architecture

6.2 Design Paradigm: RESTful web service

The Internet is the glue that binds diverse devices ranging from PC to mobile together. It is a neutral platform which is accessible to all. To reach out to a maximum number of users, it is imperative to embrace the web as the development platform.

RESTful web services are built to work best on the Web. REST was first introduced by Roy Fielding in year 2000 [50]. Representational State Transfer (REST) is a web service architectural style which proposes certain guidelines to enhance the performance, scalability, and modifiability of the service on web.

RESTful directives focus on creation of a stateless service which can be used in a client/server framework easily. These stateless services offered through web are termed as web resources and they are accessible using unique Uniform Resource Identifiers (URIs). As a result, standardized interface to web resource can be achieved. The communication to these web resources is usually using HTTP which itself is a stateless protocol.

The set of rules which enable a web service to adopt the RESTful architectural style are as follows.

- **Addressability**: It is mandatory for each web resource exposed by the RESTful service to be identifiable on web. To accomplish this, URIs are employed as global resources addresses. Using these URIs, the client can make use of resources available with the web service.

- **Uniform interface**: The availability of web resources is established using a standardized stateless protocol like HTTP. Resources can subsequently be manipulated using a fixed set of four create, read, update, delete (CRUD) operations. For HTTP protocol, these turn out to be PUT, GET, POST, and DELETE requests.

- **Self-descriptive messages**: Requests to web resources need to be representation independent. This implies the server should be able to handle a variety of data formats, such as HTML, XML, plain text, JSON, etc.

- **Client-server architecture**: The motivation is to divide the application into manageable chunks. User interface concerns are separated from data storage and computation concerns. As a result, both the components can be developed independently of each other.

- **Stateless interaction**: Each interaction with the web service need to be self-sufficient and contain all the required data. Client-side information like the session state can be stored only on the client itself. Thus, the web service is unaware of any past or future interactions with client. If some information needs to be persisted, a database needs to be used and some kind of authentication system needs to be adopted.

- **Layered system**: The client connects to an intermediate web service which then handles the request and proxies it to relevant service. This operation happens automatically without the client knowing anything about it. This improves system scalability and stability. The intermediate server can also be used for load balancing.

Thus, creating RESTful web services helps transcend the boundaries imposed by platforms and devices. The client-server model enables dynamic updates to the web application. The client-server communication is standardized, making the application inherently scalable and modular.

In MotionGen2.0, resource identification is done by assigning different URLs to independent operations like path synthesis, motion synthesis, and mixed synthesis. Use of HTTP CRUD operations ascertains a standard interface. The server can handle requests containing XML, JSON or plain text files. Each query contains sufficient information for the server to process it independently enabling the server to be stateless. A multi-layer server architecture is established to handle static content requests, computation intensive requests and progress queries separately. This enhances the scalability of overall system. More details about MotionGen2.0's server back-end and client front-end have been presented in next chapter.

6.3 Stack selection: MEAN

A Web stack refers to a group of software which is used in tandem for the purpose of web development. Some of the main components of a minimal web stack are an operating system, web server, database server, and programming language. LAMP, which stands for Linux, Apache, MySQL, and PHP, is one traditionally used and well-established web stack. However, lately, newer stacks have gained usage over LAMP and offer many new advantages. MEAN is one such web stack which is of interest.

MEAN consists of JavaScript-based tools MongoDB [51] (database system), Express.js [52] (backend web framework), AngularJS citeangularjs (front end frame work), and Node.js [53] (runtime environment) used to develop web applications. MEAN stack consists of all the basic ingredients needed to create a RESTful web service.

MEAN stack has been chosen to develop MotionGen2.0. It offers numerous advantages over other web stacks. Most important advantages it has are

- It lets the developer write the entire code in JavaScript, be it back-end or front-end.
- It provides a more flexible noSQL database than conventional relational databases.
- All the MEAN stack technologies are open source and available for free.

Each of the technology in MEAN stack and its use in MotionGen2.0 has been discussed in a later section.

6.4 Mobile App: Apache Cordova

Mobile operating systems have conventionally been app driven instead of browser-based technologies. To target mobile users, one needs to create a mobile app catered to each mobile platform like ios, android, etc.

One way is to create native apps for each platform. This requires a different codebase for each platform in their programming language of choice. For example, applications in Android are developed using Java while those on iOS requires Objective-C or Swift. Maintaining these native apps is time intensive and redundant as updating a feature requires equal work on each platform. To get over this disadvantage, MotionGen2.0 for mobile has been used using Apache Cordova [54] framework.

Apache Cordova is an open-source mobile development framework which uses standard web technologies like HTML [55], CSS [56] and JavaScript [57] for cross-platform development. It effectively eliminates programming for a specific mobile platform and replaces it with standardized web development. Apache Cordova provides native application wrappers which run web-pages in the form of webViews. WebView can be thought of as emulated browsers within a native app. Thus, when a Apache Cordova application is launched, a WebView is loaded within the app and the code is executed on it. This is a huge advantage as it permits the use of web-based front-end code as the apps source code. This standardizes the development process and features can simultaneously be pushed to both web and mobile.

Apache Cordova began as PhoneGap [58], a mobile development framework created by Nitobi, and was later acquired by Adobe. Adobe contributed PhoneGap to Apache [59] to be developed as open source under the name Cordova. Currently, Adobe PhoneGap is the commercial version of Apache Cordova which Adobe offers with additional features like desktop installer and online compiler. Although these features are great to have, issues have been observed with PhoneGap's buggy integrating with Cordova every release cycle. In order to maintain stability and reliability, MotionGen 2.0 has been developed on Cordova and not PhoneGap.

Since apps developed on Apache Cordova are implemented through webViews, they require additional standards-compliant APIs to access device specific functions such as sensors, device data, etc. These plugins convert JavaScript functions into native functions. MotionGen 2.0 uses plugins for file access, camera, email, file open, file copy, file sharing, PDF export and GIF export [60–69].

6.5 User experience: Response Time Limits

Quick reactiveness of application to user interaction is of utmost importance to conserve workflow. As a result, one of the most important usability metric for any website is its speed. The time it takes a web application to respond to user input is measured as the response time.

As Nielsen [70] discusses, speed affects the usability of a website in two ways. Firstly, a user has a limited working memory where information quickly degrades if engagement is not continuous. Secondly, waiting for the website to respond does not play well with a user's psychological desire to have control over the machine. This leads to a situation when user detaches from his workflow and loses his focus.

This high sensitivity of users work efficiency to website response time was first looked into by Miller [71] and further explored by Nielsen. They propose three response-time limits established based on their experiences which are as follows.

- **0.1 second** is the response-time limit for having the user feel that the application is reacting instantaneously. Thus, to keep the user engaged, display of results is sufficient and no additional feedback is necessary.

- **1.0 second** is the response-time limit for the user's flow of thought to stay uninterrupted. Even though the user notices this reaction delay, no special feedback is usually necessary to keep him engaged. However, the user does lose the feeling of working directly with the software and becomes aware of each interaction.

- **10 seconds** is the response-time limit for keeping the user's attention focused on the task. A feedback is mandatory to communicate expected completion time, especially when response time is variable. This gives user the opportunity to engage with other tasks while waiting for the computer to finish

Thus, application response time is paramount to conserving a user's workflow. To implement path and mixed synthesis in MotionGen 2.0, optimization subroutines are used which are inherently time-consuming. Efforts have been made to minimize this response time with a least possible trade-off on accuracy of the calculations. Also, the progress of server, for a computationally intensive task, is communicated to the user in real-time to keep him informed and engaged.

6.6 User Interface: Responsive web design

Responsive web design (RWD) is an approach to web design which makes web pages adapt and render well on a variety of devices and window or screen sizes. As MotionGen 2.0 is targeted to work on a multitude of devices, it has been developed using the principles of RWD. To make sure MotionGen 2.0 looks and function at its best, UI has been tweaked appropriately for large screens (desktop), medium screens (tablets) and small screens (phones).

6.7 User Interface: Principles of display design

MotionGen 2.0's interface has been designed keeping user interaction as its focus. The interface is intended to provide intuitive controls to the user through the design process. Christopher Wickens [72, 73] defined 13 principles of display design in his book “An Introduction to Human Factors Engineering”. MotionGen 2.0’s User Interface(UI) has been designed by utilizing some of these principles to create an effective display design. These principles can be divided into four categories as

- **Perceptual principles**- Clear display on a variety of screen sizes while balancing similarity and redundancy of interactions and features.
- **Mental model principles**- Impress upon user’s intuition by icon, animation and feature design
- **Principles based on attention**- Keeping the clutter to a minimum and providing contextual feature access to user
- **Memory principles**- Pro-active aiding based on user intent while allowing consistency between other CAD softwares.

Application of these principles makes a huge impact on user productivity.

Chapter 7

MotionGen 2.0

Having discussed the guiding principles of application architecture, it is now time to discusses the implementation and functionality details related to MotionGen2.0.

7.1 Functionality

The new MotionGen2.0 features a totally new code-base and web-based architecture compared to old MotionGen, which was focused exclusively on mobile devices. Enormous new functionality in both domains, mechanism analysis and synthesis, have been added in MotionGen2.0. It can now simulate one and multi-degrees of freedom planar n-bar linkages. It also has path and mixed four- bar synthesis capabilities in addition to motion synthesis. These additions make MotionGen2.0 a cutting-edge platform for mechanism designers accessible easily through the web.

The analysis capability of MotionGen2.0 include simulation of

- Multi degree of freedom n-bar mechanisms having revolute joints with sequential multi-inputs.
- Single degree of freedom n-bar mechanisms having revolute joints.
- Single degree of freedom 4-bar mechanisms having prismatic or revolute joints.
- Serial chain mechanisms.

The synthesis capability of MotionGen2.0 include

- Motion Synthesis ($\geq$5+point/line constraints, Tolerance based)
- Path Synthesis ($\geq$5pts)
- Mixed Synthesis ($\geq$3pose+$\geq$2path)
- Serial coupled chain mechanisms Synthesis ($\geq$5pts)

The focus of this document is synthesis and architecture related enhancements offered in MotionGen2.0. For more details regarding algorithms used to synthesize mechanisms, refer Chapter 2,3,4. A detailed study of analysis related additions in MotionGen2.0 is available in [74]

Other MotionGen2.0 backbone functionalities, which enable the user to interact with analysis and synthesis algorithms are as follows

- Platform - Web, Appstore
- User interaction - Touch, mouse
- Input - Text, XML
- Output - Text, XML, Pdf, Gif
- Tracing - image, camera
- Sample example problems
- Workspaces for multiple simultaneous workflows

The figure below summarizes the capability of MotionGen2.0 in a nice graphical form.

All the code-base for MotionGen2.0 resides on the Server (also referred as back-end). The server sends user-facing code (also referred as front-end) on request by the user through the browser or app. Synthesis calculations are handled by the server while analysis computation is done locally on user's device. This delicate load-balance enables real-time simulation with scalable synthesis computation power. Details about technologies used in back-end and front-end are discussed in next sections.

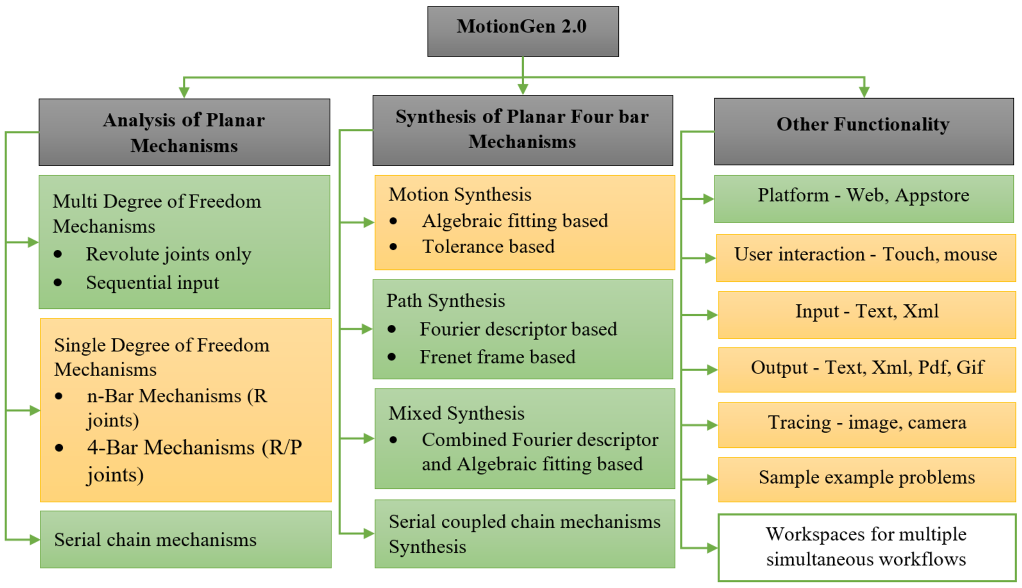

MotionGen 2.0
Analysis of Planar Mechanisms
Synthesis of Planar Four bar Mechanisms
Other Functionality
Multi Degree of Freedom Mechanisms
Revolute joints only
Sequential input
Single Degree of Freedom Mechanisms
n-Bar Mechanisms (R joints)
4-Bar Mechanisms (R/P joints)
Serial chain mechanisms
Motion Synthesis
Algebraic fitting based
Tolerance based
Path Synthesis
Fourier descriptor based
Frenet frame based
Mixed Synthesis
Combined Fourier descriptor and Algebraic fitting based
Serial coupled chain mechanisms Synthesis
Platform - Web, Appstore
User interaction - Touch, mouse
Input - Text, Xml
Output - Text, Xml, Pdf, Gif
Tracing - image, camera
Sample example problems
Workspaces for multiple simultaneous workflows

Figure 7.1: Functionality of MotionGen2.0 (Green: newly added, Yellow: updated, White: no change)

7.2 Back-end

7.2.1 Server Hardware

The hardware specification of server module are as follows

- Processor- 8x Intel Xeon CPU E5620 @ 2.40GHz
- Memory- 8161MB
- Storage- 2x ATA WDC WD1602ABKS-1 160GB, 1x ATA SAMSUNG HE161HJ 160GB

The storage devices have been setup in RAID configuration to mirror each other's data as a failsafe.

The software being used is Ubuntu Server 16.04.3 LTS which is an open source platform. It is the most used Linux server operating system because of its support and reliability.

7.2.2 Node.js

Node.js [53] is a server-side environment that allows Node developers to build servers and network applications with JavaScript for the first time. This means entire sites can be run on a unified JavaScript stackboth the client-side software and the server-side software.

Its an asynchronous, non-blocking, event-driven I/O system due to which, it can handle many requests concurrently. Its open source, cross-platform and actively developed. It is light-weight and efficient too.

MotionGen2.0 uses a two server setup. All client requests are handled by one server while the other is exclusively used for processing computationally complex and time-taking tasks.

The multi-threaded environment provided by multi-core CPU is being exploited in MotionGen2.0 using 'cluster' module of Node.js.

7.2.3 NPM

NPM [75] is the default package manager for Node.js platform. This centralized repo can be used to find the best framework and packages for the task at hand. Some of the packages being used in MotionGen2.0 are as follows

- **express** [76] is a web application framework that provides you with a simple API to build websites, web apps and backends.
- **http-proxy** [77] is a HTTP programmable proxying library that supports websockets.
- **socket.io** [78] is a framework which enables real-time bidirectional event-based communication between server and client.
- **socket.io-redis** [79] is a library which helps run multiple socket.io instances that can all broadcast and emit events to and from each other.
- **socket.io-emitter** [80] is a library which allows communication with socket.io servers easily from non-socket.io processes.
- **mongodb** [81] is the driver API which connects the server to Mongo daemon and database.

- **pm2** [82] is a production Runtime and Process Manager for Node.js apps with a built-in Load Balancer.
- **nodemon** [83] is a utility that will monitor for any changes in your source and automatically restart your server. Process manager helps to keep the application alive forever, restart on failure, reload without downtime and simplifies administrating.
- **fmin** [84] is a math library used to implement Nelder Mead optimization.
- **numeric** [85] is a math library used to compute SVD of a matrix. SVD for a complex system of equations is done by reducing it in higher dimensional real system [41].
- **mathjs** [86] is a well-documented math library used for matrix and complex number manipulation.
- **body-parser** [87] is a package which parses the body content with POST HTTP requests.
- **http-auth** [88] is a package which enables client-side authentication requirement
- **xmldom** [89] is a package which enables .xml related functionality
- **cordova** [90] is a framework which repackages the front-end into a mobile app.

7.2.4 Mongodb

MongoDB [51] is an open source, document-oriented database designed with both scalability and developer agility in mind. Instead of storing your data in tables and rows as you would with a relational database, in MongoDB you store JSON-like documents with dynamic schemas.

In MotionGen2.0, it is used to store server progress on computationally heavy tasks and store synthesis data to be used in future for machine learning purposes.

7.2.5 Back-end Architecture

The back-end adopts a multi layered architecture with various servers simultaneously running each dedicated to a specific task. Figure 7.2 visualizes how each of these components work in tandem to serve MotionGen2.0 from web.

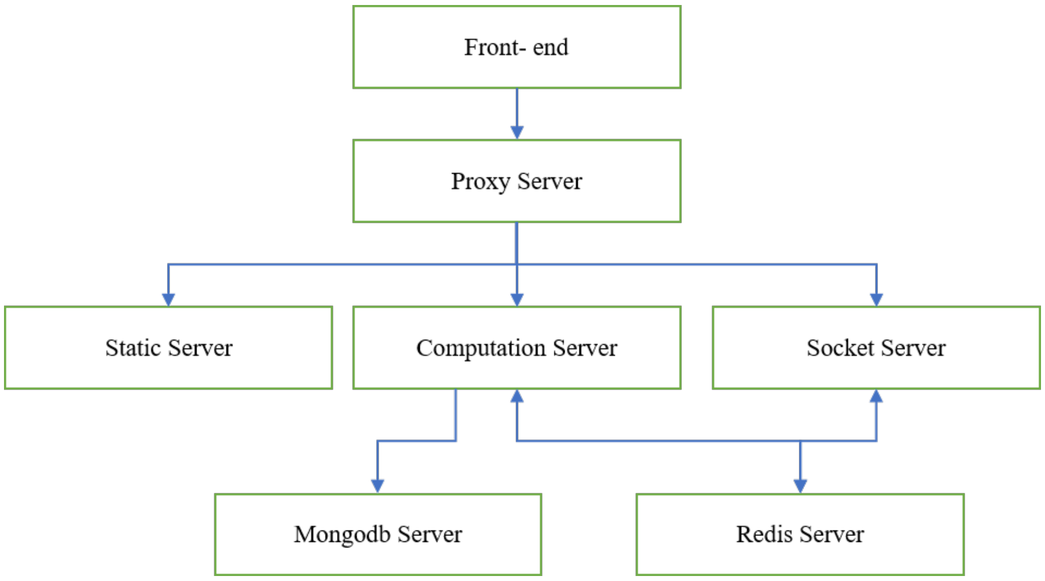


Figure 7.2: Back-end Architecture

The individual components of back-end architecture and their intended task are

- **Frontend** is the part of codebase served to the user which resides on their device. It is tasked with accessing server resources according to user requirements.
- **Proxy Server** is the server which handles all requests to the website and proxies them to the relevant route.
- **Static Server** is the server which hosts Frontend code and serves it to user device on request to website domain.
- **Computation Server** is the server which process all computationally tedious tasks including path, motion and mixed synthesis. It runs in a multi-threaded environment to take advantage of multi-core processing.

- **Socket Server** is the server which process websocket requests to enable realtime communication between server and clients.
- **Redis Server** is the server which acts as central communication hub between multithreaded cluster on Computation Server and single threaded Socket Server.
- **Mongodb Server** is the database being used to store the synthesis information being calculated on computation server.

7.3 Front-end

Front-end includes all the Static web content including HTML, CSS and js libraries which are served to the user. This is the part of code which actually resides on the user's device once it is transmitted from the server. Analysis computations are done on the front-end as the numeric complexity is lesser. This also subverts the movement of data to and fro from the server, making the process almost instantaneous and real-time.

Some of the libraries being used on the front end are as follows

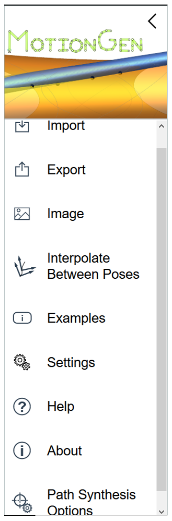
- **hammer.js** [91] is the library which enables simultaneous handling of touch and mouse based events on the front-end.
- **jquery.js** [92] is a library which simplifies how to traverse HTML documents, handle events, perform animations, and AJAX. All HTTP requests to the server are made using jquery.
- **Numeric.js** [85], **Math.js** [86], **Quartic.js** [93], **Decimal.js** [94] and **Complex.js** [95] act as the math engine and carry out all front-end calculations.

7.3.1 UI

The user interface for MotionGen2.0 has been designed to enhance user workflow. Contextual options have been added and less frequented options have been moved to side menu. Effectively, this has reduced display clutter drastically and the user can now focus exclusively on the task of designing or analyzing the mechanism. The core menus of MotionGen2.0 are as follows

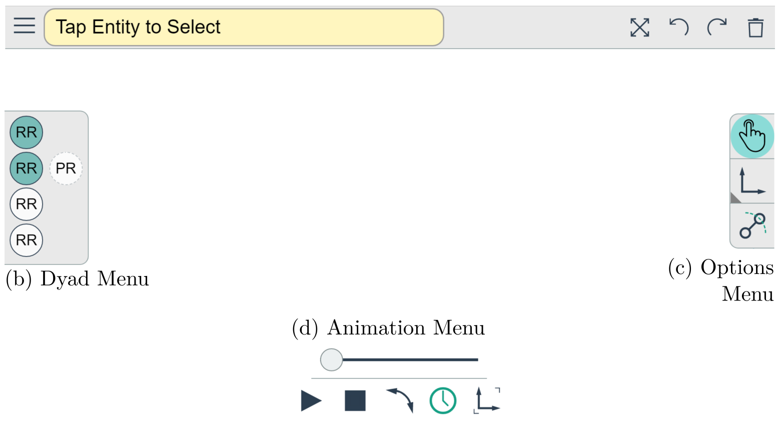
- **Top menu** houses the contextual bar and undo, redo, delete buttons.
- **Options menu** consist of three-tier menu for Linkage control, Synthesis, and Analysis. Each tier can easily be accessed by selecting any of its other member buttons. Long pressing buttons can make extra options available.
- **Animation menu** is tasked with controlling the motion of mechanism drawn or synthesized.
- **Dyad menu** consists of a variety of dyads if multiple solutions for a synthesis problem exists.
- **Side menu** contains all the other advanced functionality, neatly accessible a click away. It has menus for import, export, image, examples, settings and path synthesis options.

Fig. 7.3a and Fig. 7.3b shows how each of the mentioned menus actually looks on MotionGen 2.0.



(a) Side Menu

(a) Top Menu / Information Bar



(b) Dyad Menu

(c) Options Menu

(d) Animation Menu

(b) Dashboard Menus

Figure 7.3: MotionGen2.0 Menus

7.3.2 Front-end Architecture

The front-end adopts an MVC architecture with clear demarcation between input, processing and output tasks. Figure 7.4 visualizes how each of these components work together to display MotionGen2.0 and make it work on users device.

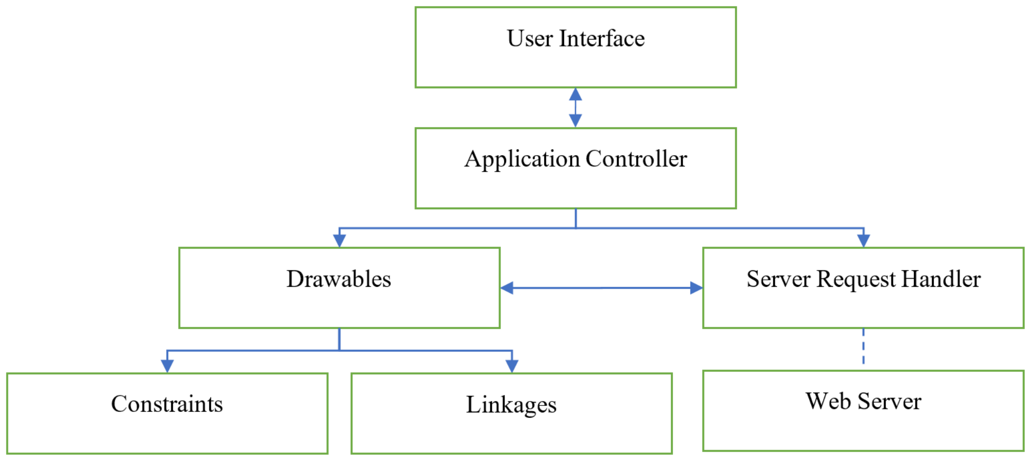


Figure 7.4: Front-end Architecture

The individual components of front-end and their function are

- **User Interface** consists of all the HTML content including the multi-layered canvas, menus, buttons, etc. Any input on the canvas layer triggers the Application Controller.
- **Application Controller** is the prominent communication hub of MotionGen 2.0. It listens for user input and responds by sending appropriate data to Drawables object or Server Request Handler object. It contains all event handlers and is the main entry point into the program.
- **Drawables** is an abstract class which stores the state of Client session at any given time. It responds to the Application controller and assigns information to Constraints or Linkage appropriately.

- **Server Request Handler** is an abstract class which deals with transfering computationally intensive tasks to web server. It works asynchronously with the server which is in contrast with the synchronous nature of most of the other operations in the application.
- **Constraints** is a class which consists of all the synthesis constraint info including Poses, Path points, Lines and Points. It assigns information based on the instructions that it receives from the Drawables class.
- **Linkages** is a class which consists of all the analysis data like the Links, Joints, Curves, Input Rpm's, etc. It also assigns information based on the instructions that it receives from the Drawables class.

7.4 Case study

This section presents a step by step process which the user can follow to synthesize his own mechanisms. Motion, Path, and Hybrid Synthesis workflow have been demonstrated.

7.4.1 Motion Synthesis workflow

Follow the following workflow to synthesize motion using >5 poses.

- Clear the screen by tapping on 'Delete' icon at the top right corner of the screen.
- Select 'Add Pose' button under Synthesis group in Options Menu.
- Tap on the screen to add a pose
- While adding, touch and drag to set the required orientation of the pose
- Repeat until the all poses have been inputted. Once 5 poses are inputted, MotionGen2.0 automatically synthesizes four bar in real-time.
- Update, delete or add poses as necessary according to the problem constraints.

Motion Synthesis is also possible using 3 or 4 poses. The user just needs to select additional line or point constraints to get the solution mechanism. The user has the ability to select is a mechanism should follow a subset of poses exactly and approximate others. An example of Motion Synthesis on MotionGen 2.0 is shown in Fig. 7.5.

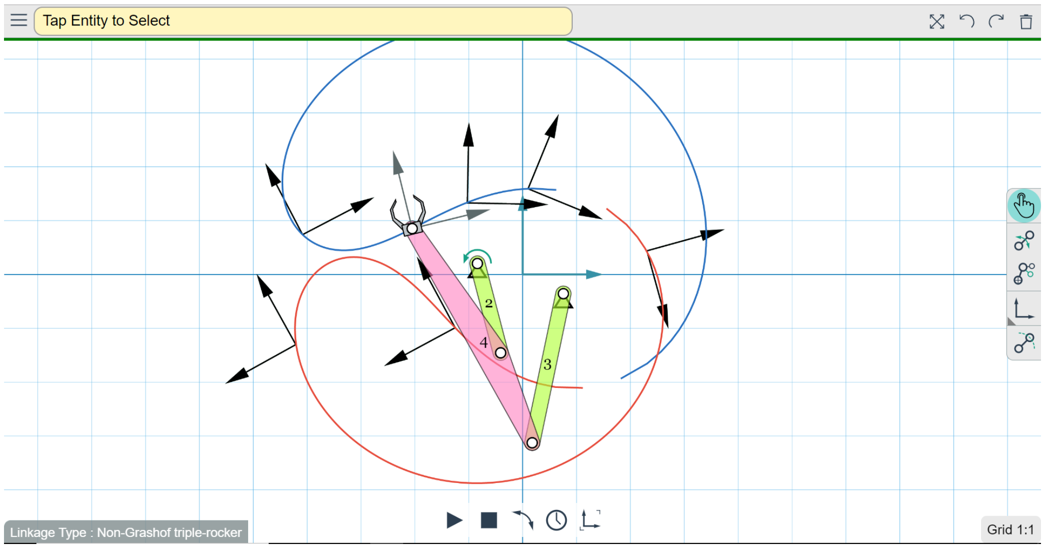


Figure 7.5: Instance of Motion Synthesis on MotionGen 2.0

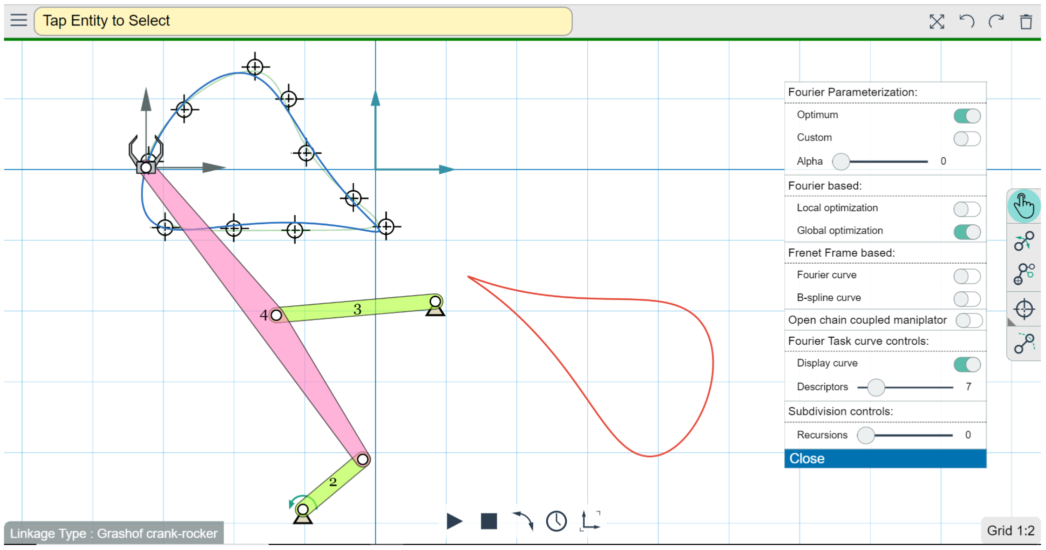
7.4.2 Path Synthesis workflow

Follow the following workflow to synthesize path using >5 path points.

- Clear the screen by tapping on 'Delete' icon at the top right corner of the screen.
- Long press 'Add Pose' button under Synthesis group in Options Menu.
- When the 'Add Path point' button is displayed, select it
- Tap on the screen to add a pose
- While adding, touch and drag to set the path point to the exact required position.

- Repeat until all poses have been inputted. Once 5 poses are inputted, MotionGen2.0 automatically synthesizes a Single degree of freedom coupled mechanism. This is the default synthesis option selected
- Go into the side menu and access 'Path Synthesis Options'
- In the pop-up dialog box, select local optimization or global optimization for Fourier descriptor based synthesis. Select Fourier curve or B-spline curve for Frenet frame based path synthesis.
- NOTE: The path synthesis is not instantaneous and can take the server upto 10 seconds to process your request.
- Update, delete or add poses as necessary according to the problem constraints.

Other advanced options like harmonic count, subdivision recursion, and parameter control can be controlled from the 'Path Synthesis Options' popup box. These options grant the user enhanced flexibility and control over the synthesis process. An example of Path Synthesis on MotionGen 2.0 is shown in Fig. 7.6.



Fourier Parameterization:
Optimum ✓
Custom
Alpha 0
Fourier based:
Local optimization
Global optimization ✓
Frenet Frame based:
Fourier curve
B-spline curve
Open chain coupled maniplator
Fourier Task curve controls:
Display curve ✓
Descriptors 7
Subdivision controls:
Recursions 0

Figure 7.6: Instance of Path Synthesis on MotionGen 2.0

7.4.3 Mixed Synthesis workflow

Follow the following workflow to synthesize mechanism using >2 path points and >3poses.

- Clear the screen by tapping on 'Delete' icon at the top right corner of the screen.
- If you want to input a path point, Long press 'Add Pose' button under Synthesis group in Options Menu and then select 'Add Path point' button. Then tap on the workspace.
- Similarly 'Add Pose' can be selected
- Note: The order in which poses and paths are inputted decides the Fourier descriptors of task path.
- Repeat until all poses have been inputted. Once the minimum number of required constraints are inputted, the server will automatically compute the mechanism and display it onscreen.
- NOTE: The mixed synthesis is not instantaneous and can take the server upto 10 seconds to process your request.
- Update, delete or add poses or path points as necessary according to the problem constraints.

The server automatically uses the modified re-parametrization to find smoothest path curve through the inputted constraints. The user has the flexibility of order in which he inputs poses and path points. This is very similar to practical situations where the user knows just some intermediate orientations. An example of Mixed Synthesis on MotionGen 2.0 is shown in Fig. 7.7.

This completes the overview of MotionGen 2.0

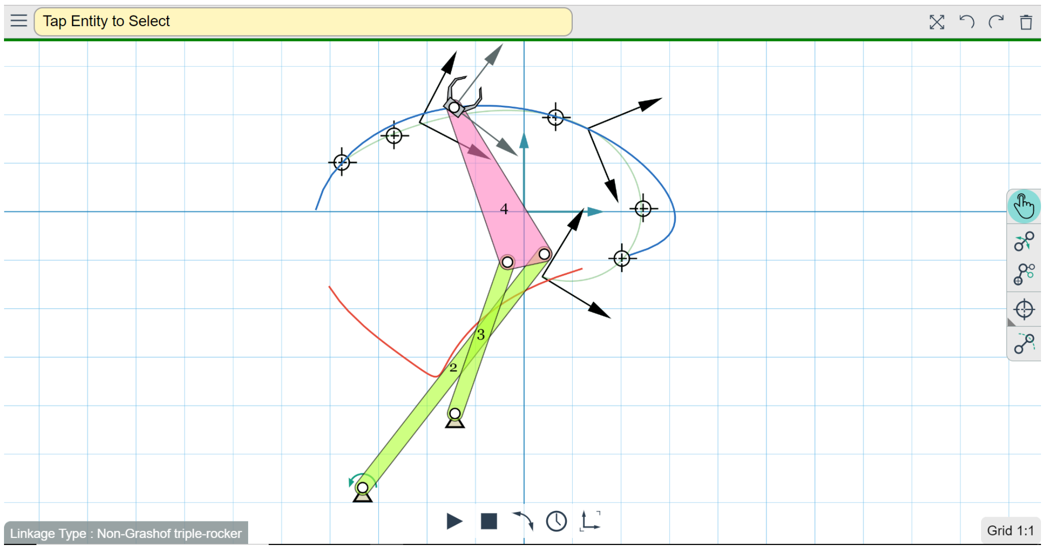


Figure 7.7: Instance of Mixed Synthesis on MotionGen 2.0

Chapter 8

Conclusion and Future Work

This thesis claims original contributions in following areas

- Introduces MotionGen2.0, the first ever web-based scalable application capable of performing Motion, Path, and Mixed Synthesis.
- A re-parameterization scheme to optimize Fourier parameters for Path synthesis.
- A Fourier based Mixed synthesis approach which unifies Path and Motion Synthesis.

Implementation of deterministic and stochastic optimization techniques has been done to solve the Path Synthesis problem. A framework to find the least square solution of a complex system of equation using SVD has also been implemented.

Various user-centric tweaks have been implemented to enhance the usability of existing software. Combined with a polished user interface and a powerhouse of capability, MotionGen2.0 aims to inspire professionals and students to design.

8.1 Future Work

The computation of synthesized mechanism is still not real-time and has immense scope for improvement. Use of Machine Learning to train a neural network which gives a good approximate mechanism is being looked into. This would cut down runtime immensely.

Eqn. 3.17 derived for mixed synthesis can be used in pure over-constrained motion approximation problem to interactively guide the user towards a better mechanism. Calculation of best pose orientation would be possible and this flexibility would enhance the design process.

MotionGen2.0 only supports Path and Mixed synthesis of RR dyads. An algorithm which does simultaneous type and dimensional synthesis for Path problem is an interesting prospect.

Bibliography

[1] Freudenstein, F., 1954, “An analytical approach to the design of four-link mechanisms”, Trans. ASME, **76(3)**, pp. 483–492.

[2] Hartenberg, R. S. and Denavit, J., 1964, Kinematic synthesis of linkages, McGraw-Hill.

[3] Blechschmidt, J. and Uicker, J., 1986, “Linkage synthesis using algebraic curves”, ASME J. Mech., Transm., Autom. Des, **108**, pp. 543–548.

[4] Suh, C. H. and Radcliffe, C. W., 1978, Kinematics and mechanisms design, Wiley.

[5] Sandor, G. and Erdman, A., 1984, “Advanced Mechanism Design, vol”, I and II Prentice-Hall Inc.

[6] Nolle, H. and Hunt, K., 1971, “Optimum synthesis of planar linkages to generate coupler curves”, Journal of Mechanisms, **6(3)**, pp. 267–287.

[7] Fox, R. and Gupta, K., 1973, “Optimization technology as applied to mechanism design”, Journal of Engineering for Industry, **95(2)**, pp. 657–663.

[8] Bagci, C. and Lee, I.-P. J., 1975, “Optimum synthesis of plane mechanisms for the generation of paths and rigid-body positions via the linear superposition technique”, Journal of Engineering for Industry, **97(1)**, pp. 340–346.

[9] Angeles, J., Alivizatoss, A., and Akhras, R., 1988, “An unconstrained nonlinear least-square method of optimization of RRRR planar path generators”, Mechanism and Machine Theory, **23(5)**, pp. 343–353.

[10] Sancibrian, R., Viadero, F., Garcıa, P., and Fernández, A., 2004, "Gradient-based optimization of path synthesis problems in planar mechanisms", Mechanism and machine theory, **39(8)**, pp. 839–856.

[11] Smaili, A. A., Diab, N. A., and Atallah, N. A., 2005, "Optimum synthesis of mechanisms using tabu-gradient search algorithm", Journal of Mechanical Design, **127(5)**, pp. 917–923.

[12] Smaili, A. and Diab, N., 2007, "Optimum synthesis of hybrid-task mechanisms using ant-gradient search method", Mechanism and Machine Theory, **42(1)**, pp. 115–130.

[13] Ullah, I. and Kota, S., 1997, "Optimal synthesis of mechanisms for path generation using Fourier descriptors and global search methods", Journal of Mechanical Design, **119(4)**, pp. 504–510.

[14] Martínez-Alfaro, H., 2007, "Four-bar mechanism synthesis for n desired path points using simulated annealing", Advances in Metaheuristics for Hard Optimization, Springer, pp. 23–37.

[15] Zhou, H. and Cheung, E. H., 2001, "Optimal synthesis of crank–rocker linkages for path generation using the orientation structural error of the fixed link", Mechanism and Machine Theory, **36(8)**, pp. 973–982.

[16] Marı, F. T. S., González, A. P., et al., 2003, "Global optimization in path synthesis based on design space reduction", Mechanism and machine theory, **38(6)**, pp. 579–594.

[17] Smaili, A. and Diab, N., 2007, "A new approach to shape optimization for closed path synthesis of planar mechanisms", Journal of Mechanical Design, **129(9)**, pp. 941–948.

[18] Watanabe, K., 1992, "Application of natural equations to the synthesis of curve generating mechanisms", Mechanism and Machine Theory, **27(3)**, pp. 261–273.

[19] Hoeltzel, D. and Chieng, W.-H., 1990, "Pattern matching synthesis as an automated approach to mechanism design", ASME J. Mech. Des, **112(6)**, pp. 190–199.

[20] Vasiliu, A. and Yannou, B., 2001, “Dimensional synthesis of planar mechanisms using neural networks: application to path generator linkages”, Mechanism and Machine Theory, **36(2)**, pp. 299–310.

[21] Galán-Marín, G., Alonso, F. J., and Del Castillo, J. M., 2009, “Shape optimization for path synthesis of crank-rocker mechanisms using a wavelet-based neural network”, Mechanism and Machine Theory, **44(6)**, pp. 1132–1143.

[22] Efrat, A., Fan, Q., and Venkatasubramanian, S., 2007, “Curve Matching, Time Warping, and Light Fields: New Algorithms for Computing Similarity between Curves”, Journal of Mathematical Imaging and Vision, **27(3)**, pp. 203–216, doi:10.1007/s10851-006-0647-0, URL https://doi.org/10.1007/s10851-006-0647-0.

[23] Dibakar, S. and Mruthyunjaya, T., 1999, “Synthesis of workspaces of planar manipulators with arbitrary topology using shape representation and simulated annealing”, Mechanism and Machine Theory, **34(3)**, pp. 391–420.

[24] Wu, J., Ge, Q., Gao, F., and Guo, W., 2011, “On the extension of a Fourier descriptor based method for planar four-bar linkage synthesis for generation of open and closed paths”, Journal of Mechanisms and Robotics, **3(3)**, p. 031002.

[25] Wu, J., Ge, Q., and Gao, F., 2009, “An efficient method for synthesizing crank-rocker mechanisms for generating low harmonic curves”, ASME Paper No. DETC2009-87140.

[26] McGarva, J. and Mullineux, G., 1993, “Harmonic representation of closed curves”, Applied Mathematical Modelling, **17(4)**, pp. 213–218.

[27] Li, X. and Chen, P., 2017, “A Parametrization-Invariant Fourier Approach to Planar Linkage Synthesis for Path Generation”, Mathematical Problems in Engineering, **2017(8458149)**, p. 16.

[28] Golinski, J., 1974, “An adaptive optimization system applied to machine synthesis”, Mechanism and Machine Theory, **8(4)**, pp. 419–436.

[29] Kreyszig, E., 1968, Introduction to differential geometry and Riemannian geometry, University of Toronto Press.

[30] Deshpande, S. and Purwar, A., 2017, “A Task-Driven Approach to Optimal Synthesis of Planar Four-Bar Linkages for Extended Burmester Problem”, Journal of Mechanisms and Robotics, **9(6)**, p. 061005.

[31] Ge, Q., Purwar, A., Zhao, P., and Deshpande, S., 2017, “A Task-Driven Approach to Unified Synthesis of Planar Four-Bar Linkages Using Algebraic Fitting of a Pencil of G-Manifolds”, Journal of Computing and Information Science in Engineering, **17(3)**, p. 031011.

[32] Ge, Q., Zhao, P., Purwar, A., and Li, X., 2012, “A novel approach to algebraic fitting of a pencil of quadrics for planar 4R motion synthesis”, Journal of Computing and Information Science in Engineering, **12(4)**, p. 041003.

[33] Ravani, B. and Roth, B., 1983, “Motion synthesis using kinematic mappings”, ASME J. Mech., Transm., Autom. Des, **105(3)**, pp. 460–467.

[34] Rudin, W. et al., 1964, Principles of mathematical analysis, volume 3, McGraw-hill New York.

[35] Dyn, N., Levin, D., and Gregory, J. A., 1987, “A 4-point interpolatory subdivision scheme for curve design”, Computer aided geometric design, **4(4)**, pp. 257–268.

[36] Dyn, N. and Levin, D., 1990, “Interpolating subdivision schemes for the generation of curves and surfaces”, Multivariate approximation and interpolation, Springer, pp. 91–106.

[37] Tong, Y., Myszka, D. H., and Murray, A. P., 2013, “Four-bar linkage synthesis for a combination of motion and path-point generation”, Ph.D. thesis, University of Dayton.

[38] Brake, D. A., Hauenstein, J. D., Murray, A. P., Myszka, D. H., and Wampler, C. W., 2016, “The complete solution of alt–burmester synthesis problems for four-bar linkages”, Journal of Mechanisms and Robotics, **8(4)**, p. 041018.

[39] Zimmerman, R. A., 2017, “Planar Linkage Synthesis for Mixed Motion, Path and Function Generation Using Poles and Rotation Angles”, ASME 2017 International Design Engineering Technical Conferences

and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. V05AT08A047–V05AT08A047.

[40] Li, X., Zhao, P., and Ge, Q., 2012, "A fourier descriptor based approach to design space decomposition for planar motion approximation", ASME Paper No. DETC2012-71264.

[41] Day, D. and Heroux, M. A., 2001, "Solving complex-valued linear systems via equivalent real formulations", SIAM Journal on Scientific Computing, **23(2)**, pp. 480–498.

[42] Krovi, V., Ananthasuresh, G., and Kumar, V., 2002, "Kinematic and kinetostatic synthesis of planar coupled serial chain mechanisms", TRANSACTIONS-AMERICAN SOCIETY OF MECHANICAL ENGINEERS JOURNAL OF MECHANICAL DESIGN, **124(2)**, pp. 301–312.

[43] Nie, X. and Krovi, V., 2005, "Fourier methods for kinematic synthesis of coupled serial chain mechanisms", Transactions of the ASME-R-Journal of Mechanical Design, **127(2)**, pp. 232–241.

[44] Liu, Y. and McCarthy, J. M., 2017, "Design of Mechanisms to Draw Trigonometric Plane Curves", Journal of Mechanisms and Robotics, **9(2)**, p. 024503.

[45] Inc, A., 2008, "Force Effect Motion", .

[46] Rector, D., "Linkage", URL http://blog.rectorsquid.com/linkage-mechanism-designer-and-simulator/.

[47] Campbell, M., "Planar Mechanism Kinematic Simulator", URL http://design.engr.oregonstate.edu/pmksintro.html.

[48] Engineering, A., "SAM (Synthesis and Analysis of Mechanisms)", URL http://www.artas.nl/en.

[49] Petuya, V., Macho, E., Altuzarra, O., and Pinto, C., 2011, "Educational Software Tools for the Kinematic Analysis of Mechanisms", Comp. Appl. Eng. Education, **6(4)**, pp. 261–266.

[50] Fielding, R. T. and Taylor, R. N., 2000, Architectural styles and the design of network-based software architectures, University of California, Irvine Doctoral dissertation.

[51] MongoDB, Inc., "Mongodb", URL https://www.mongodb.com/.

[52] The Nodejs Foundation, "Express.js", URL https://expressjs.com/.

[53] The Nodejs Foundation, "Node.js", URL https://nodejs.org/en/about/.

[54] Foundation, T. A. S., "Cordova", URL http://cordova.apache.org/docs/en/5.0.0/guide_overview_index.md.html#Overview.

[55] W3C, "HTML 5", URL http://www.w3.org/TR/html51/.

[56] W3C, "CSS 3", URL http://www.w3.org/Style/CSS/.

[57] W3C, "JavaScript", URL http://www.w3.org/standards/webdesign/script.

[58] Inc., A. S., "PhoneGap", URL http://phonegap.com/about/.

[59] Foundation, T. A. S., "ASF", URL http://www.apache.org/foundation/.

[60] Apache, "cordova-plugin-file", URL https://github.com/apache/cordova-plugin-file.

[61] Apache, "cordova-plugin-camera", URL https://github.com/apache/cordova-plugin-camera.

[62] Katzer, "cordova-plugin-email-composer", URL https://github.com/katzer/cordova-plugin-email-composer.

[63] Pwlin, "cordova-plugin-file-opener2", URL https://github.com/pwlin/cordova-plugin-file-opener2.

[64] Chaudhary, G., "Asset2SD", URL https://github.com/gkcgautam/Asset2SD.

[65] MrRio, "jsPDF", URL https://github.com/MrRio/jsPDF.

[66] MrRio, "jsPDF Image Plugin", URL https://github.com/MrRio/jsPDF/blob/master/plugins/addimage.js.

[67] W3C, "File API", URL http://www.w3.org/TR/FileAPI/.

[68] Verbruggen, E., "SocialSharing-PhoneGap-Plugin", URL https://github.com/EddyVerbruggen/SocialSharing-PhoneGap-Plugin.

[69] Kwok, K., "jsgif", URL https://github.com/antimatter15/jsgif.

[70] Group, N. N., 1993, "Response Times: The 3 Important Limits", URL https://www.nngroup.com/articles/response-times-3-important-limits/.

[71] Miller, R. B., 1968, "Response time in man-computer conversational transactions", Proceedings of the December 9-11, 1968, fall joint computer conference, part I, ACM, pp. 267–277.

[72] Wickens, C., Lee, J., Liu, Y., and Becker, S., 2004, An Introduction to Human Factors Engineering. Second ed., Pearson Prentice Hall.

[73] Wikipedia, 2017, "Human-computer interaction — Wikipedia, The Free Encyclopedia", URL https://en.wikipedia.org/w/index.php?title=Human%E2%80%93computer_interaction&oldid=791784316, [Online].

[74] Lodha, A., 2017, "Web-Based Design and Simulation of One- and Multi-Degrees of Freedom Planar n-Bar Linkages", .

[75] npm Inc., "npm", URL https://www.npmjs.com/about.

[76] TJ Holowaychuk, C., "Express", URL github.com/expressjs/express.

[77] Charlie Robbins, C., Jarrett Cruger, "Http-proxy", URL github.com/nodejitsu/node-http-proxy.

[78] darrachequesne, C., "Socket.io", URL github.com/socketio/socket.io.

[79] darrachequesne, C., "socket.io-redis", URL github.com/socketio/socket.io-redis.

[80] darrachequesne, C., "socket.io-emitter", URL github.com/socketio/socket.io-emitter.

[81] mbroadst, C., “MongoDB driver”, URL github.com/mongodb/node-mongodb-native.

[82] tknew, C., “pm2”, URL github.com/Unitech/pm2.

[83] remy, C., “nodemon”, URL github.com/remy/nodemon.

[84] benfrederickson, “fmin”, URL github.com/benfred/fmin.

[85] Loisel, S., “Numeric Javascript”, URL http://www.numericjs.com/.

[86] de Jong, J., “Math.js”, URL http://mathjs.org/index.html.

[87] dougwilson, C., “body-parser”, URL github.com/expressjs/body-parser.

[88] gevorg, C., “http-auth”, URL github.com/http-auth/http-auth.

[89] jindw, C., “xmldom”, URL github.com/jindw/xmldom.

[90] stevegill, C., “cordova”, URL github.com/apache/cordova-cli.

[91] Tangelder, J., “Hammer JS”, URL http://hammerjs.github.io/.

[92] The jQuery Foundation, “jQuery”, URL http://learn.jquery.com/about-jquery/.

[93] Lathoud, G., “JS Quartic”, URL https://github.com/glathoud/js.quartic/blob/master/solve_quartic.js.

[94] Mclaughlin, M., “Decimal JS”, URL http://mikemcl.github.io/decimal.js/.

[95] Lathoud, G., “Complex”, URL https://github.com/glathoud/js.quartic/blob/master/complex.js.