PMKS+ Web

A Major Qualifying Project
submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
degree of Bachelor of Science

Written By:
Nicole Burgess (CS)
Tyler Evans (CS)
Robert Eskridge (CS)

Advisors:
Professor David C. Brown (CS)
Professor Pradeep Radhakrishnan (MME)

May, 2024

Abstract

Students learning to design and analyze planar mechanisms often have the additional struggle of needing to learn sophisticated professional software such as SolidWorks. PMKS+ is a web application that aims to reduce this struggle by including only the functionality needed for novice students. PMKS+ has existed for years but the complexity of its implementation complicated further development. This project aimed to complete a refactor of PMKS+ focused on adhering to software design principles. We selected a new architecture for the program, allowing us to re-implement a majority of the existing features while adding support for multiple mechanisms, along with new mechanism manipulation functionality. Additionally, we analyzed the existing interface to improve system usability in the new implementation while adhering to standard HCI principles. Our comparative evaluation showed an improvement in user experience as well as a drastic improvement in code structure and documentation.

Acknowledgements

Our team would like to thank our advisors Professor David C. Brown (CS) and Professor Pradeep Radhakrishnan (ME) for their help in guiding us through this project. We greatly appreciate their help in analyzing the UI of PMKS+ to help guide us in redesigning the User Interface. We also would specifically like to thank Professor Radhakrishnan for his expertise in our application and taking the time to explain the concepts necessary to completing the project.

Our team would also like to thank Alex Galvan for his help in understanding the concepts that we were unfamiliar with regarding Mechanical Engineering. We also appreciate the help by Kohmei Kadoya for all of his advice on the direction to take our project. Finally we would like to thank each of the students who participated in each of our surveys for taking the time to give us constructive feedback on our application.

Table of Contents

List of Figures

Authorship

Section	Author(s)
Abstract	Tyler, Nicole
Acknowledgements	Nicole
Introduction	Robbie
Research/Problem Statement	Robbie
Goals/Requirements	Tyler, Robbie, Nicole
Planar Mechanisms	Tyler
Mechanism Synthesis	Nicole
Human Computer Interaction	Tyler
2019 MQP	Tyler
2020 MQP	Nicole
2022 MS Thesis	Tyler
2023 MS Thesis	Nicole
Competitor Application Comparison	Nicole
Understanding the Project	Nicole
Developing the Goals and Requirements	Nicole
A Term Student Survey	Robbie
Analyze Current UI: Tutorial, Side Navigation Tab, Synthesis	Nicole
Analyze Current UI: Edit, Analysis	Robbie
Refactoring Previous PMKS Codebase	Tyler
Three Position Synthesis Design	Nicole
Edit Panel Design - Links	Nicole
Edit Panel Design - Joints	Robbie
Edit Panel Design - Compound Links	Robbie
Analysis Panel Design	Robbie

Tab Design	Nicole
Refactor Development	Tyler
HTML Blocks	Nicole
Using Switch Cases	Robbie
Grid	Robbie
Top Bar	Nicole
Synthesis Panel	Nicole
Edit Panel - Links, Joints	Nicole
Edit Panel - Compound Links	Robbie
Analysis Panel	Robbie
User Comfort	Nicole
Class Diagram Comparison	Tyler
Written Code Comparison	Tyler
Developer Survey	Nicole
Restoring PMKS+	Robbie
Additional Features	Nicole, Robbie
Conclusions	Nicole

1. Introduction

PMKS+ (Planar Mechanism Kinematics Simulator Plus) is a web application designed to create and analyze four bar linkages. PMKS+ has been through several iterations of development. In 2019, development of the tool was started, and has been ongoing every year since. Within the scope of the PMKS+ project, the emphasis on planar mechanisms is deliberate and strategic. The software offers students in Mechanical Engineering (ME) and Robotics Engineering (RBE) a simplified, 2D representation of these mechanisms. By focusing on two dimensions, PMKS+ eliminates the complexities introduced by the third dimension, physical materials, and the intricacies of professional software. This approach ensures that students can grasp foundational concepts without being overwhelmed, serving as a stepping stone before they transition to more advanced tools. For an overview of this report's contents, see the Table of Contents.

2. Problem Statement

The existing PMKS+ has several problems that give a novice user difficulty in creating mechanisms, analyzing mechanisms, and using the tool to its fullest extent in the classroom. The front end and user interface of the current program is burdened with inconsistencies, bugs, and confusing language across several different components. The backend of the program does not adhere to design principles, strict architecture, or industry standard practices for software development. Broad issues like these breed smaller issues for the user interacting with the program and for developers attempting to add new features or fix bugs in existing features. Our team seeks to address these issues while maintaining the spirit and look of the original implementation, to ensure that students enjoy a seamless transition between systems during the course of their work.

3. Goals and Requirements

The goals of this project were to: refactor and simplify the code to allow for easier future development; improve the user interface for novice users; add useful functionality that is currently missing.

The most significant goal of this project was to recreate a codebase that will allow new developers to quickly and easily be onboarded to the project. By significantly decreasing the learning curve, we will enable future developers to start much quicker and make more progress early on. In addition to this, we also analyzed the user interface that exists in 2023 PMKS+ Master's Thesis (PMKS+2023) to improve the user experience and add new program functionality (Kadoya, 2023).

Currently, the codebase contains a lot of complicated logic that lacks the structure required for a piece of software of such scope. Due to this absence of structure, it takes a significant amount of time to understand the dependencies between separate components of the application. A large amount of time can be saved in a rolling project like this by adhering to software architectural principles taught at WPI, which is an element that PMKS+2023 lacks.

The current user interface (UI) that exists has issues with the content and with following the Human Computer Interaction (HCI) principles. There are many components that need to be added in order to make the application more useful to students. Analyzing the UI would give an opportunity to understand where the application is lacking. We wanted to adhere to standard HCI principles to ensure that our application is accessible, that it reduces the cognitive load, and is consistent. Based on this analysis, we created mockups to continue to analyze and eventually implement into the application. These mockups gave ourselves and will give any future teams a roadmap into potential looks of the project.

Lastly, it was important that we re-implemented the features that currently existed in the application into the new, refactored codebase. During our UI analysis we discussed the new features that should be added. The reason for this is because we want to have an application that retains current functionality and is viable for classroom use. We also wanted to be able to increase the functionality so the application is more useful and powerful for students.

In summary, the project goals were to:

1. Refactor the codebase and give it more structure
2. Analyze the current user interface to improve the user experience
3. Create and implement a list of application features

In support of these goals, we established the following testable requirements:

- Simplify the logic of the backend code

- Add documentation to the entire codebase
- Create mockups based on feedback
- Adhere to HCI principles
- Create list of features that exist on the current application
- Create a list of features that need to be added
- Implement the ability to create and analyze multiple mechanisms

4. Literature Review

This chapter presents an overview of what planar mechanisms are and how they are synthesized to create new mechanisms. As the project members' backgrounds are all in Computer Science, we felt it was necessary to understand these concepts. We discuss the importance of referencing Human Computer Interaction to ensure the accessibility and ease of use of our application. This section also examines the previous work done on the application to understand the methods used previously. Lastly, we examined other applications which have similar capabilities to PMKS+ that are already used by students.

4.1 Planar Mechanisms

Planar mechanisms operate within two dimensions and serve as foundational constructs in the fields of Mechanical Engineering and Robotics. These mechanisms are composed of links (rigid bodies that form the backbone of the mechanism) and joints (the connections that determine how these links interact and move relative to each other). In PMKS+2023 (shown in figure 4.1.1), the blue bars are the links that are connected by joints which are represented by the pale yellow circles. The left and right joints are grounded, represented by the black lines extended downward from the joint. The left joint is also labeled an input, represented by the black square surrounding the joint. This mechanism is a four-bar linkage which is the simplest form of a moveable mechanism.

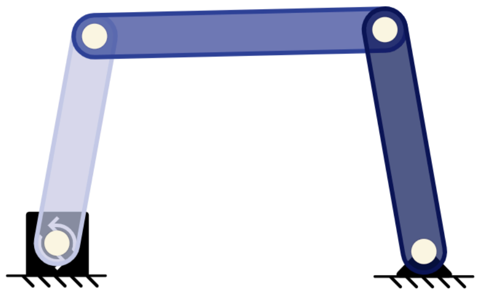


Figure 4.1.1 Four-Bar Linkage Created Using PMKS+2023

Joints are categorized into Lower Pairs and Higher Pairs. Lower pairs, such as the Revolute Pair and the Prismatic Joint, have surface contact between their elements, allowing for specific movements like rotation or sliding. In contrast, higher pairs, exemplified by cam joints and gear teeth contacts, are characterized by point or line contact between elements. These higher pairs are not handled by PMKS+2023 currently (Waldron et al., 2016).

A pivotal concept in planar mechanisms is the Degrees of Freedom (DoF), which represents the number of independent movements a mechanism can perform. The Mobility Formula, or Kutzbach–Grübler's equation, plays a crucial role in determining a mechanism's degrees of freedom, taking into account the number of links, joints, and their individual degrees of freedom (Waldron et al., 2016).

4.2 Mechanism Synthesis

Mechanism synthesis is a tool used in Mechanical Engineering to determine the configuration of linkages that would allow a mechanism to move through certain specified positions. There are three types of mechanism synthesis: three position synthesis, path synthesis, and function synthesis. In PMKS+2024, the current goal was to focus on the first two: namely, position and path syntheses.

Three position synthesis allows the user to input multiple positions that the user wants the coupler link of the mechanism to pass through. These positions are specified using x and y coordinates and angle. With these positions, the app creates a four-bar mechanism to pass through all the specified positions. A mechanism that could be created is shown in figure 4.2.1. In the figure below, the green positions represent the positions that the mechanism's coupler would pass through. It uses equations to calculate the lengths of input and follower links required to allow the mechanism to pass through those positions (Shete et al., 2017).

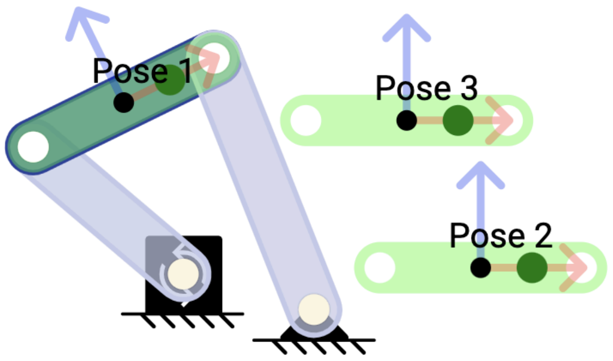


Figure 4.2.1 Mechanism Created through Three Position Synthesis

Path synthesis is very similar to three position synthesis, the difference is that path synthesis uses a series of points to follow instead of positions of a link to generate a mechanism. This point could be followed by either a joint or a tracer point defined by the user. The user would be able to input multiple points to create a path. This path could be a straight line or curves. There are a couple of different ways that a user could do this. The first is to draw the path using the cursor and the other is to input specific points that the path would follow. The two methods have different pros and cons. The drawing method is significantly easier for a user. However, if they attempt to draw a straight line and, due to user error, they have a slight curve in the line, the path would then be incorrect and not what they intended. Inputting points on a path allows the user to be more exact and precise with their paths. However this is a very tedious process for the user. When inputting various points, the user is able to specify whether the connection between points is curved or straight. The user also is able to specify a boundary that the mechanism has to stay within; for instance in case a wall exists that a mechanism would need to avoid. There could be either many or no solutions based on the path specified by the user. As with three position synthesis, the mechanism has to be able to be implemented in a real life example and certain paths may not be achievable (Torres-Moreno et al., 2021).

4.3 Human Computer Interaction

The study and implementation of Human Computer Interaction (HCI) within the PMKS+ software is of paramount importance, especially given the software's target audience of college students learning about planar mechanisms. HCI focuses on understanding and optimizing the interaction between users and computer systems, ensuring that software is both usable and effective. The principles discussed in the next sections are from the book “Designing Interactive Systems: A Comprehensive Guide to Hci, Ux & Interaction Design” by David Benyon (Benyon, 2013).

4.3.1 Usability and User Experience (UX)

For novice users, the interface of PMKS+ must be intuitive and user-friendly. This means that the design should prioritize ease of use, minimizing the learning curve. Features should be easily discoverable, and the flow from one task to another should be logical. Given that this software will be used as a learning tool, any barriers to usability could hinder their learning process. Moreover, a positive user experience (UX) can enhance user satisfaction, encouraging students to engage more deeply with the software and, by extension, the subject matter (Benyon, 2013).

4.3.2 Feedback Mechanisms

In the realm of HCI, feedback is crucial. When students interact with the software, whether they are building, simulating, or analyzing planar mechanisms, they should receive immediate and clear feedback on their actions. For instance, if they make an error in constructing a mechanism, the software should not only highlight the error but also provide guidance on how to correct it. This feedback aids in the learning process, allowing students to understand their mistakes and learn from them (Benyon, 2013).

4.3.3 Cognitive Load

From an HCI perspective, it is essential to consider the cognitive load that the software places on its users. Given that students are using PMKS+ as a learning tool, the software should aim to reduce unnecessary cognitive strain. This can be achieved by presenting information in small, digestible chunks, using visual aids where possible, and ensuring that the interface is not cluttered. A reduced cognitive load allows students to focus on the core learning objectives without being overwhelmed. It also reduces the amount of introduction required to use the program (Benyon, 2013).

4.3.4 Consistency and Predictability

Consistency in design and interaction patterns ensures that once users learn to perform a task, they can predict how similar tasks will function. This predictability enhances usability, as users don't have to relearn interactions for different parts of the software (Benyon, 2013).

4.4 Previous Work on PMKS+

This section analyzes the work done on PMKS+ on four projects; the 2019 MQP, the 2020 MQP, the 2022 Master's Thesis, and the 2023 Master's Thesis.

4.4.1 2019 Major Qualifying Project (MQP)

Historically, until 2019, PMKS, built on the Silverlight platform, primarily focused on the kinematics of mechanisms. The 2019 MQP (PMKS2019) report detailed the methodologies and technical decisions involved in transitioning the original PMKS to a web platform (Appikatla et al., 2019). Appikatla and team focused on two main objectives: ensuring multi-browser compatibility and introducing a modern user interface. Their decision to transition from Silverlight to AngularJS was in line with current trends in web development, which prioritize adapting legacy systems to contemporary technological standards.

4.4.1.1 The Process and its Challenges

The team conducted a literature review to inform their approach to interface development. Based on their findings, they adopted an iterative UI design process that underwent seven iterations, with each iteration refined according to user feedback. This methodology highlighted the role of user feedback in optimizing interface design.

The report provided insights into the technical challenges faced during the PMKS transition. Some of these challenges included migration options, system design considerations, and details about the simulator class which is used to animate the linkages. Their choice to use a web app platform was based on specific technical justifications, emphasizing the advantages of web-based applications, mainly pointing to the convenience of a web-based application.

4.4.1.2 Testing

Appikatla et al. prioritized testing in their development process. They conducted tests for browser compatibility and software functionality. Additionally, their user evaluations, which included individual assessments and user feedback, provided data on the application's performance and identified areas for improvement

4.4.2 2020 Major Qualifying Project (MQP)

The 2020 MQP (PMKS+2020) recognized the need for new features to be added to PMKS2019 (Dowd et al., 2020). The team began by analyzing what had been done in PMKS2019 and researching the best practices and industry standards for the user interface.

4.4.2.1 Designing the User Interface

The team first designed the user interface and went through multiple iterations. They designed mockups of each feature including how the user would access these features. Some key features added to the iterations were: the ability to change the speed and direction of the mechanisms; allowing the user to edit link information within the tables which display the coordinates and values representing the link; and improving the functionality and benefits of the tables overall. Overall, the PMKS+2020 focused on making PMKS+ more user friendly, such as improving the ability to create shapes and the ease of adjusting the shapes to be exactly what the user wants.

4.4.2.2 Features Added

Features were added that were significantly more difficult to implement. The first was creating links using Scalable Vector Graphics (SVGs). These SVGs have bounds and a list of points is generated to create the link. In addition to this, the team also was able to encode all the elements created by the user into a URL so that the mechanism can be shared by providing a URL. They also added the ability to save the mechanism: when it is saved it creates a CSV (comma-separated values) file that displays all the information similar to the way the URL encodes the same data.

4.4.2.3 Testing

The team ensured the functionality of the program by thoroughly testing the application. They used the browsers: Chrome in a Mac environment; Safari in a Mac environment; and Edge in a Windows environment. The team tested every functionality with multiple types of linkages and the individual features. They also tested the functionality of PMKS+ in comparison to Working Model, another application used to create linkages. The results did not show that PMKS+ was better, however it did show that it is comparable to a paid program such as Working Model. People responded that PMKS+ was easier to use and looked better, however the program would crash frequently causing PMKS+ to ultimately perform worse.

4.4.3 2022 Master’s Thesis

The 2022 Master's thesis (PMKS+2022) provided a comprehensive overview of the advancements achieved and challenges faced during the re-development of the Planar Mechanism Kinematic Simulator (PMKS) (Galvan, 2022). This work was particularly relevant as it laid the foundation for our current project, aiming to continue and expand upon PMKS+2020.

4.4.3.1 Analysis of Competitor Applications

Galvan did a review of similar software and course expectations. He noted that the software required the following features: position analysis using circle intersection technique, velocity analysis using instant center technique and loop equation technique, acceleration analysis using loop equation technique, force analysis using static equilibrium technique, and force analysis using Newton's second law. Galvan explored the possibilities of integrating different methods of synthesis such as three position synthesis and path synthesis. The complete list of desirable features was additionally supplemented by general notes about usability and flow.

4.4.3.2 Integrating New Features

PMKS+2022 expanded its capabilities by integrating methods based on static equilibrium and Newton’s second law. These methods were employed to determine reaction forces at joints and the input torque in 1 DOF (degree of freedom) planar mechanisms with revolute joints. Additionally, the instant center method was incorporated to ascertain instant centers and velocities of links in mechanisms, both revolute (joints that rotate) and prismatic (joints that move linearly). A significant addition was the stress analysis of links and joints.

PMKS+2022 aimed to achieve several goals, both in terms of analysis and interface improvements. On the analysis front, Galvan focused on loop kinematics, determining angular velocity and acceleration using loop equations, and evaluating joint forces and motor torque. The interface was enhanced to add the center of mass locations, force information, and unit systems. Features were integrated such as downloading analysis data, displaying equations for student verification, and showcasing charts for various analyses. Galvan also recognized the need to display loops to users and planned to showcase the equations used for position analysis in future iterations. However, Galvan identified areas that required further refinement and testing. For instance, force analysis involving grounded sliders needed more validation before being introduced in courses. The student evaluated the User Interface and based on the evaluation, were especially concerned with the display of instant centers.

The backend was enhanced to allow multiple inputs, facilitating the operation of multiple linkages simultaneously. Several analysis modules were integrated, including position analysis using the circle intersection method, identification of vector loops, as well as velocity and acceleration analyses using the loop equation technique. The user interface saw upgrades such as data filtering, data plotting, equation showcasing, and the display of labels for joints, links, and forces.

4.4.4 2023 Master's Thesis

The 2023 Master's thesis (PMKS+2023) explained the process taken to get to the current version that we have of PMKS+ (Kadoya, 2023). Like PMKS+2022, Kadoya began by discussing the previous work done. It listed the main features as the ability to create and edit planar linkages, performing kinematic and force analysis on joints and links, as well as exporting, saving and sharing linkages. The thesis also recognized the main goals of PMKS+ as an educational tool that is also flexible, can be used in a collaborative environment and is accessible. Students are the main users of PMKS+ and should be able to access the web app for free, use it to help them with homework, to learn, and to collaborate with other students.

4.4.4.1 Recognizing Current Needs

The thesis recognized some problems with PMKS+2022. Some of these issues focused on the user interface and the ease of use of the program, others focused on smaller bugs and issues in the web app. Specifically the user needed to find features that were embedded inside multiple nested tabs and it was especially difficult to create linkages. Another issue was the lack of space on the grid as there was a limit to how much a user could zoom in or out, so the size of linkages that could be created were limited.

The essential and primary needs of the users were identified as the ease of use of the user interface, the accessibility of the web app, kinematic analysis, force analysis, linkage synthesis, and an educational focus. With these needs identified the student made goals to improve the display of new features to make them more discoverable, address unintuitive quirks so as to make it more user friendly, modernize the UI, and make it easier to create linkages.

4.4.4.2 Design Process and Testing

The process for making the application more user friendly included creating prototypes and mockups of the UI, incorporating style conventions from similar engineering applications, determining the typical use cases, and testing the application throughout the process. The feedback received through the testing was positive overall and students testing the application were able to complete tasks faster than they could in PMKS+2022.

4.4.4.3 Implementing Changes

Some of the solutions to the issues with PMKS+2022 were addressed and solved. The new features and solutions included improved ability to select and deselect objects, as well as the way that is displayed. There is now an improved three-tab navigation allowing users to more easily find features, the analysis graphs were improved to see multiple components in one graph, and the ability to view kinematic force values was added. The users could make infinitely many link shapes with improved welding capabilities. There is a better and more dynamic grid allowing users to build larger linkages without losing the ability to see them clearly. Help options were also added to include a feedback form, tool tips, and new icons. Additionally, the backend was improved to cut down on code complexity and function length. Specific focus was placed on making the content delivery network more effective. Kadoya also added dependency injection, outsourcing to open source libraries, and reusable components were added to make it easier for future developers to use.

4.5 Competitor Application Comparison

PMKS+ is not the only application on the internet used for designing planar mechanisms. There is also MotionGen Pro, SAM, and Working Model (Purwar et al, 2017; Mad-croc nederland, n.d; Working Model, n.d). Each of these applications have their drawbacks and advantages, but some do not even compare to PMKS+ as the application's capabilities are not as advanced as the capability of PMKS+. The first thing to note is that of the four aforementioned applications, only MotionGen is free and available online, only requiring one to make an account, while the rest require you to download and pay for the application. Only SAM and MotionGen have the ability to synthesize mechanisms as well as easily create and analyze mechanisms. The others do not offer these features and are much more complicated to use. Many of these have a lot of icons that are not easily identifiable, making them not suitable for a novice user, the primary users of PMKS+. However, many of these other applications have the ability to download the data of the mechanisms created, which was not currently offered in PMKS+2023.

Overall, SAM and MotionGen Pro are the most comparable to PMKS+, as many of the other applications available are not as user friendly and are not as easily accessible due to their cost. SAM has a version of three position synthesis however it is a lot more involved and complex than it is made to be in PMKS+. There are minimal instructions and the next steps are somewhat unclear to someone who has never used the application before. Both SAM and MotionGen have the ability to simulate the mechanisms as well. MotionGen Pro also includes kinematic equations and force analysis. MotionGen is also not as user friendly as PMKS+ however it is more user friendly than the other applications. MotionGen includes a method of path synthesis which is easy to use. The user is just required to draw a path and the app outputs all the potential mechanisms that could be created. MotionGen also includes tutorials and examples of mechanisms that can be created. The user interface makes it very easy to draw shapes, add grounds, inputs and also to delete an entire mechanism instead of deleting each link individually as in PMKS+.

This chapter provides the knowledge necessary to understand the goals of the project and gives baseline knowledge about planar mechanisms. We also highlighted the importance of HCI principles within PMKS+. The next chapter discusses our methodology and the steps taken to produce our final application.

5. Methodology

This chapter describes how we set out to accomplish our goals. We discuss the reasoning behind understanding the previous work on the project as well as the previous surveys, and how we developed our goals and requirements for the project. We explain our procedure for analyzing the existing UI in PMKS+2023. We also explain the process for redesigning the backend of the program.

5.1 Understanding the Project

To begin our understanding of the project, we needed to familiarize ourselves with multiple different subjects. Our goal was to put together an understanding of planar mechanisms, where and how PMKS+ is used, the tools employed to develop the application, and how the application was developed.

5.1.1 Literature Review

The first of these subjects to learn was planar mechanisms themselves. We met with both our advisor and a previous student from the project to gain a rudimentary understanding of how mechanisms are constructed and analyzed. Additionally, we read provided material that covered several of the basics, such as joints, links, inputs, and a handful of the classic linkages. Lastly, we met with our advisor Professor Radhakrishnan, and read some available resources to understand the algorithms used to analyze mechanisms. These included methods to: calculate degrees of freedom; synthesize mechanisms; determine required loops; calculate position, velocity and acceleration data; and create force body diagrams.

During this time, we also discussed the use cases of PMKS+ with our advisor. We wanted to understand how the application was employed to help aid students in the classroom. In essence, we discovered PMKS+ was used primarily as an intermediate piece of software between more rudimentary software that can only simulate linkage motion, and expert level software such as SolidWorks, which has a steep learning curve. The goal was to teach students how to construct mechanisms while providing analysis of these mechanisms and displaying the methods used to calculate the analysis. PMKS+ also allowed students to export data, and share their mechanisms in the form of a generated URL to submit as homework.

After familiarizing ourselves with the necessary background knowledge of the project, we shifted our focus to understanding the system as it is implemented in PMKS+2023. We began by reading the previous Major Qualifying Project reports written by the students that had built PMKS+ to understand their approach to tackling the project (shown in section 4.4). From there, we met with the most recent team that had worked on PMKS+ so they could walk us through how the codebase was organized, and present the tools they used to create the application. These tools included the Angular Javascript framework with Typescript as the language for the project itself. HTML and SCSS handled the construction of the front end. Github was used for issue tracking and distributed version control. Additionally, several external libraries were imported to handle creating graphs, as well as zooming and panning of the screen.

5.1.2 Understanding the Code Structure

Once we had grasped the fundamentals of the project, we moved onto developing an understanding of the complete application. To do this, we began performing minor bug fixes throughout the application. We used the GitHub issues page (https://github.com/orgs/PMKS-Web/projects/1) from previous projects to determine which bugs we could reasonably tackle given our still basic working knowledge of the project. Concurrently with our bug fixing, we created a UML class diagram of the model in the application (shown in figure 5.1.2.1).

By beginning to work within the codebase of PMKS+2023 and building a class diagram of the mechanism, we were able to formulate a mental model of the responsibilities of and relationships between different classes within the code. In order to further our understanding of the project, we met multiple times with the previous team of WPI students including Alex Galvan, Kohmei Kadoya, and Ansel Chang. These meetings gave us the largest insight into the current issues facing PMKS+2023 as a codebase and several possible solutions to remedy them.

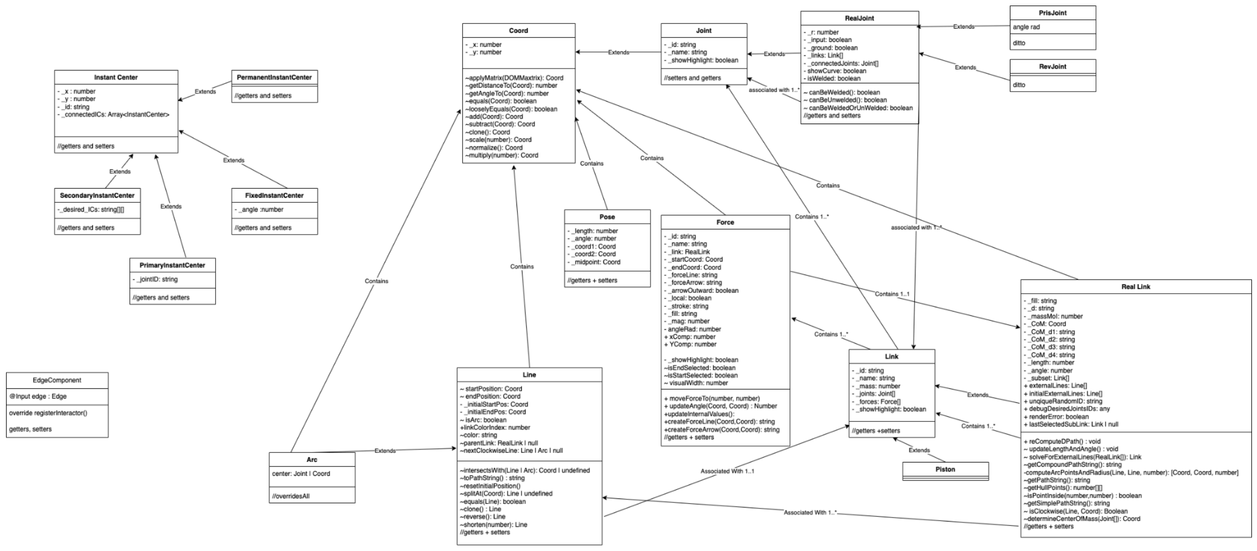


Figure 5.1.2.1 PMKS+2023 Mechanism Class Diagram

5.2 Developing Goals and Requirements

By this time at the end of A term, we decided to develop the concrete goals and requirements for our project. We had initially anticipated that a majority of our work would revolve around providing new functionality for PMKS+ as requested by our advisor Professor Radhakrishnan. These included several new features for mechanism synthesis, adding support to simulate and analyze multiple mechanisms at once, updates to the user interface design, as well as adding several other small features, and fixing many bugs.

It was at this point however, after speaking with the previous team, attempting to work within the project ourselves, and creating a class diagram of the model, we made the major decision to shift our project focus to performing a refactor of the entire project. This decision was not made lightly. The previous codebase was simply too complex from a developer standpoint to make efficient contributions to the project. By pivoting away and refactoring the project on our own, we could establish architectural practices that future developers could utilize when making further additions to PMKS+. We devised three major goals for our project at this point. Firstly, we wanted to rewrite the application while implementing as much of the current functionality as possible. Secondly, a rewrite would provide us with the opportunity to evaluate the user interface and make changes as necessary. Thirdly, while refactoring the application we would formulate a design that would either directly implement, or simplify the implementation of the additional features requested by our advisor.

With this set of goals in mind, we developed a plan for the rest of our project. One team member would begin refactoring the code, gaining feedback from the previous team for the feasibility of different designs, and begin implementation. The other two members would meet with our advisors to discuss issues with the current user interface implementation and develop mockups within Figma of an improved user interface (Figma, 2023). Once the evaluation and proposed updates to the UI were finalized, these two would begin to implement the new UI in the refactored codebase while our third member continued work on building the simulator, algorithms for analysis, and other functionality. Our goal was to give the team member in charge of refactoring enough time to produce a minimum viable product that the other two members could then use to implement the user interface and connect functionality to as it was completed.

5.3 Survey

Over the course of the year, we conducted multiple surveys to gauge how users felt about the application as well as gauge how other developers felt about the structure of the codebase. In this section, we describe the process for creating and administering these surveys.

5.3.1 A Term 2023 Student Survey Structure and Goals

The goal of the A Term 2023 survey was to assess how students could handle the features of PMKS+2023 based on their natural intuition and the tools provided by PMKS+. PMKS+ has always been, first and foremost, a tool for teaching and learning. Students were given the survey at the end of A Term, after a class that used PMKS+2023 or related tools. These students specifically came from the Mechanical Engineering and Robotics Engineering disciplines, as these are the disciplines that most often require the use of tools similar to PMKS+.

The survey was divided into a few sections. There was a section for replicating a four bar linkage based on a given example, a section for analyzing and editing the four bar linkage, a section for utilizing the existing Synthesis tab, and a section for both describing potential new features and other more abstract feedback. Each of the first three sections had a quantitative scale from 1-5 of difficulty of accomplishing tasks, qualitative opinion responses about the flow of tasks, and general questions about the task. The final section had mockup examples of future features and how each user felt about each potential new feature. These potential features included Free Body Diagrams, the ability to download kinematic data, UI changes to analysis and synthesis tabs, and other changes.

By aggregating the responses to the quantitative difficulty questions, and sorting the qualitative responses into separate buckets, we were able to assess how students felt about the overall ease of use of existing features, and where there was room for improvement. The results informed our design decisions and priorities as we progressed through our project. The questions are shown in Appendix A and the results are discussed in Appendix B.

5.3.2 Developer Survey

The main motivation for refactoring the PMKS+2023 codebase was to optimize and create code that was more understandable. To establish this we created a survey where survey participants would look at the codebase of PMKS+2023 as well as the new codebase created (PMKS+2024). The participants' requirements were that they had to be a Computer Science major or minor, needed to be at least a junior or to have taken CS3733, software engineering, and were familiar with web development. We focused vastly on the Edit panels. We had opened both codebases with each file related to the Edit panel open and gave the computer to the participants. One team member led the discussion on the code while another took notes on a separate computer. We asked questions about usability, how easy it is to find certain components in each codebase, and for general comments on the code. We surveyed four computer science students and to reduce bias, for half of the participants we showed PMKS+2023 first and the other half were shown PMKS+2024 first. The results are discussed in section 9.2.3.

5.3.3 D Term Student Survey

In order to evaluate the effectiveness of our application, we conducted a user survey with students from ME4320 in mid D term. We gave a similar task as the survey we had given in A term. All of the students taking the D term survey, did not participate in the A term survey. We created two surveys, one using PMKS+2024 and one using PMKS+2023, and randomly distributed the surveys to reduce the bias in using the application. Each survey had the same questions, only modified to suit the two PMKS+ versions. We gave a similar task as the survey we had given in A term. The majority of the students taking the D term survey, did not participate in the A term survey, but to prevent bias, we anonymized the survey. The tasks included: making a four-bar mechanism, getting kinematic data from the analysis graphs and translating the whole mechanism by 5 cm to the right. These tasks are all things that can be done in both systems but are slightly different.

The difference between this survey and the A term survey was that this one aims to be more quantitative because we asked the students to rank the difficulty of each task. The last part of the new survey included the questions from the System Usability Scale (Thomas, 2019). The purpose of using this scale is that it is a quick and easy way to get quantitative data that we could use to be able to compare the two applications. The scale includes 10 questions that require an answer between 1-5, 1 being strongly disagree and 5 being strongly agree. To get a final score, responses are then input into a formula where every odd numbered question has 1 subtracted from its score and every even number's score gets subtracted from 5. After this, all the new values are added and multiplied by 2.5 to get a score out of 100. According to the System Usability Scale, a passing score is a 68 out of 100, which means the system is usable but needs work. The results are discussed in section 9.1. The questions given are shown in Appendix C and the results are shown in Appendix D.

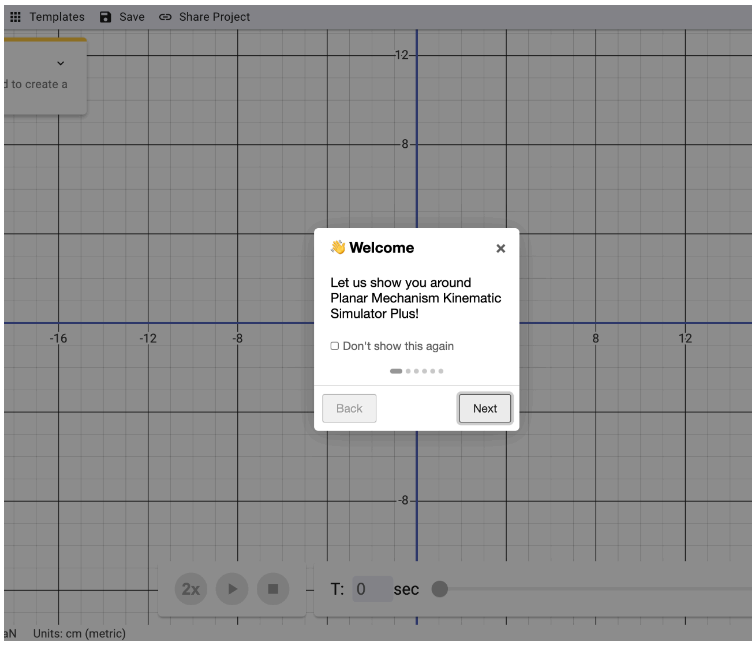
5.4 Heuristic Evaluation of PMKS+2023

There are several different sections of the user interface utilized in PMKS+2023. We chose to conduct a heuristic evaluation of each of these in succession, taking notes on how the interface prompted users to complete tasks that would be required of them. These notes can be found in Appendix J. Each section below was important enough to have its own scrutinized review. Special care was taken on the Edit, Analysis, and Synthesis panels, as these panels are where users spend a majority of their time. The Edit tab allows users to create mechanisms and edit their attributes while the Analysis tab allows users to view kinematic data of the mechanism they created through graphs. The Synthesis tab allows users to create mechanisms based on the path of a link or a joint. The feedback from this section was integrated into our new designs which can be viewed in chapter 6.

5.4.1 Tutorial Review

In evaluating the PMKS+2023, we began with analyzing the tutorial. The tutorial is a guide through PMKS+ and explains some of the tools that are available to the user. There were a lot of issues with how the tutorial looked, the information presented, and the animations used.

The information was presented in a way that personified the application. There were emojis placed at the top in the first and last section of the tutorial. The text was written with emotion as if the application was a person; for example the first slide says "Let us show you around…" (as shown in figure 5.4.4.1). The text that currently exists in the tutorial was just a placeholder and much of it needed to be re-written.


Welcome

Let us show you around Planar Mechanism Kinematic Simulator Plus!

☐ Don't show this again

Figure 5.4.1.1 PMKS+2023 Tutorial welcome page

In addition to the issues with the text within the tutorial, there were also visual issues. The first was the progress bar. The bar was very small and not very representative of the length of the tutorial. The progress bar currently includes small circles with a wider oval when the section of the tutorial is active. A better solution to more visibly see progress would be to include a bar at the bottom of the tutorial window that is extended as the user progresses through the tutorial.

The next visual issue is that during the entire tutorial, the Edit panel is active. The user is able to open other tabs during the tutorial, but this creates a bug which changes where the tutorial window is displayed. The Edit panel being open automatically can be confusing to the user as they may assume that the application will always have the Edit tab open, which in reality is not the case and the user may only edit if the edit tab is open. Since the edit tab is always open during the tutorial, the tutorial does not actually show the user what each tab looks like and it implies that the whole tutorial is about the edit tab. The tutorial should either show all of the main tabs, or none of them. But, it should at least highlight them so that users know what each tab does, not just their names.

The location and shading of the tutorial and application were also inconsistent. In the tutorial mode, anything that the tutorial did not focus on was grayed out. However, when the animation bar is shown, the empty space to the left of the animation bar was not grayed out. The tutorial window had arrows that pointed to the part that the tutorial is describing. As shown in figure 5.4.1.2, the arrows were very small and blended in with the white background and were not always pointing to what the tutorial is showing.

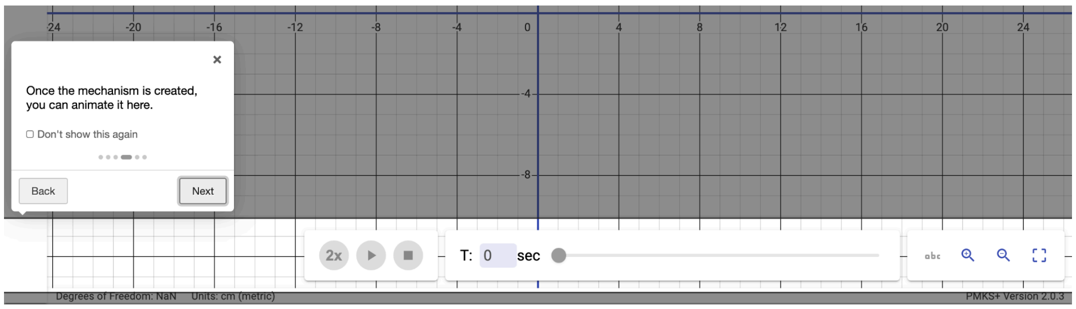

Once the mechanism is created, you can animate it here.

Figure 5.4.1.2 PMKS+2023 Tutorial Inconsistencies

One vital part that is missing from the tutorial, is that it never explains how to create a mechanism. Currently, the tutorial only highlights the features and does not provide any usage instructions. It would be a nice feature to showcase the steps to create a mechanism.

5.4.2 Side Navigation Tab Review

The current tab design (shown in figure 5.4.2.1) had some flaws. The first is that the colors should be inverted. The active tabs are blue while the panel that corresponds with the active tab is white. The colors should have more correlation so if the active tabs are white, it makes more sense for its content (the panel) to also be white. This is because the foreground is what is active and therefore these colors should match to show correlation (Benyon, 2013). Another flaw was that the tab icons are not descriptive. The only tab that is familiar is the Edit tab which has a pencil icon which is very standard in applications. The issue is that Synthesis and Analysis do not have standard icons to represent their tabs. The current design is what users preferred overall based on our survey results shown in Appendix B, but it may make the learning curve steeper because novice users may not understand the purpose of these tabs. We came up with three solutions to this: use the current design but change the icons to be more descriptive; change the icons to use text only; or a combination of both. These solutions are discussed in section 6.4.

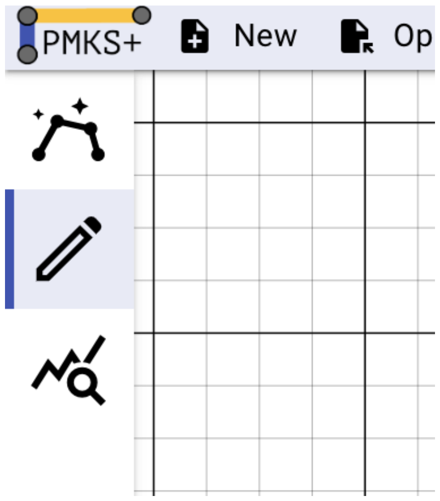
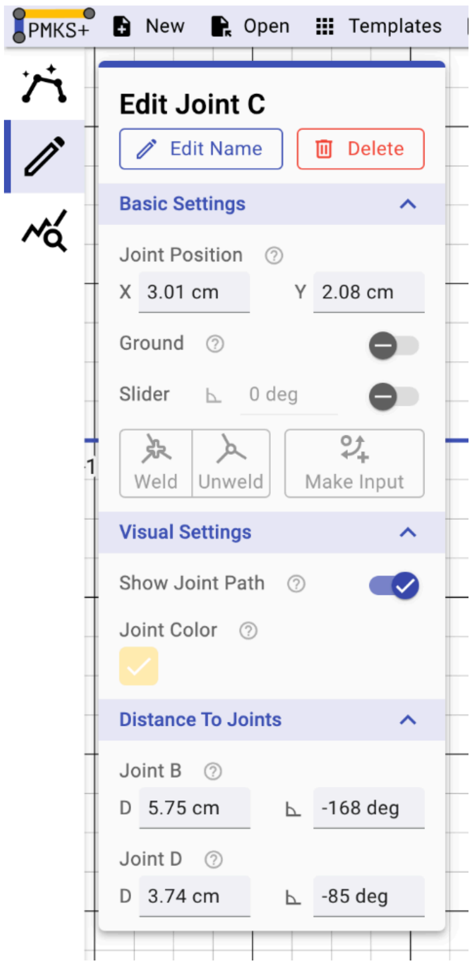


Figure 5.4.2.1 PMKS+2023 Side Navigation Icons with Active Edit Panel

5.4.3 Edit Panel Review

The Edit panel as it exists in PMKS+2023 needed several key features to be refined. The main needs for the panel include improving the user experience and intuitiveness of the design of the panel including its layout, iconography, and readability. There are two Edit panels in PMKS+2023: editing joints (figure 5.4.3.1), and the other for editing links (figure 5.4.3.2).

The layout of the Joint Edit panel was not very intuitive. The wording of the three main dropdowns were Basic Settings, Visual Settings, and Distance to Joints. This leads a user to believe having a “Basic Setting” would mean having an “Advanced Setting,” which is not the case. In addition, though Visual Settings was its own dropdown, there was only a single option inside of it (to show joint paths). Additionally, some critical features, like the ability to set speed in rotations per minute (rpm) and direction on an input joint, did not exist. This is critical because these are in the Settings panel which is set globally but, with our goal of adding multiple mechanisms, there could be a need to independently set these values. We chose to clean up the panel by more effectively grouping each action into their dropdowns, and by changing the layout within each dropdown to more effectively convey what information was available and where to access it. The layout of the Link Edit panel suffered from many of the same problems, albeit on a lesser scale. The grouping of features was more intuitive in the Link panel than the Joint panel, so we simply cleaned up the layout and streamlined the interactions within each dropdown to solve this problem.



Edit Joint C

Basic Settings

Joint Position
X 3.01 cm
Y 2.08 cm
Ground
Slider 0 deg

Visual Settings

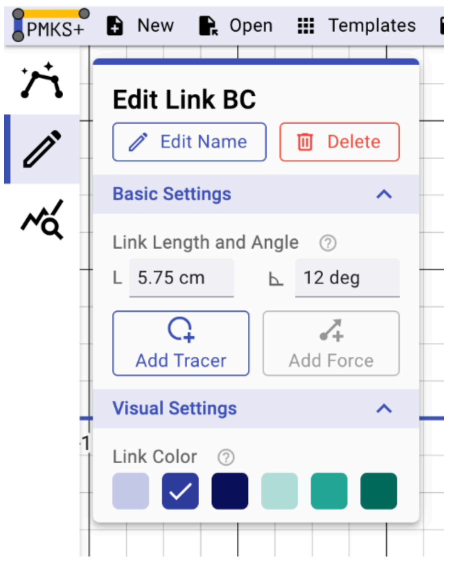
Show Joint Path ✓
Joint Color

Distance To Joints

Joint B
D 5.75 cm
-168 deg

Joint D
D 3.74 cm
-85 deg

Figure 5.4.3.1 PMKS+2023 Joint Edit Panel



Edit Link BC

Link Length and Angle

L 5.75 cm

12 deg

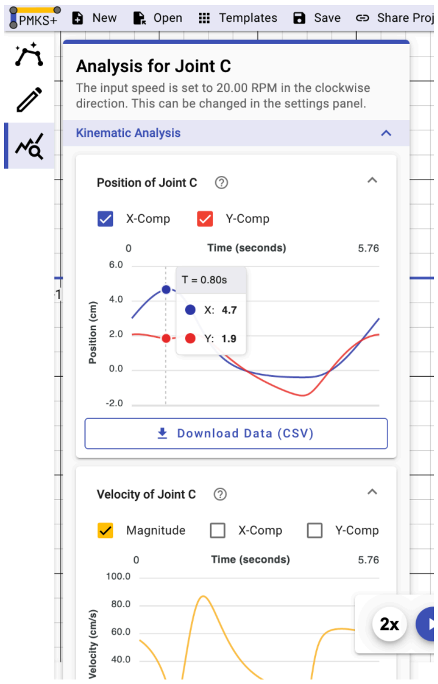
Figure 5.4.3.2 PMKS+2023 Link Edit Panel

The panel included gray text that was difficult to read. There were also instances where the meaning of text was not clear. The letter ‘D’ was used to denote distance, while angles were denoted using an icon. Icons became our preferred way to showcase what an interactable element was capable of doing or meant for. Inconsistencies in lettering and text-based information made the panel much less intuitive. There was much less direct feedback provided to the user about what the changes they were making in the Edit panel were actually doing, such as an angle displayed on screen or a link length. These would need to be added to the panel, to make the user more comfortable in editing the different components of their four bar linkages. The mockups created based on this feedback can be found in section 6.2.

5.4.4 Analysis Panel Review

The Analysis panel in PMKS+2023 (shown in figure 5.4.4.1) only had one option for analysis: kinematic analysis. Within this, the user could view graphs for position, velocity, and acceleration for the selected object (joint or link) within their linkage. While analyzing the current UI and the responses to surveys conducted on students, we found several issues that would need to be addressed in a revamped iteration of PMKS+.

The first was the ease of data access. Users wanted to see the important bits of mechanism data at a specific time step. We determined a data summary screen would be best to provide this functionality. Additionally, students wanted to be able to view multiple graphs at a time, or to know what analytical graphs were available to them. Several possibilities were developed to allow the user easier access to this information. We experimented with radio buttons of graphs, dropdown menus, and more, in an effort to reduce the clutter users reported experiencing. We also noted that adding more kinds of analysis to the panel could confuse users. As stress analysis and force analysis were both planned as future features, we needed to find a way to integrate them with the current UI without adding more complication for the user.


Analysis for Joint C

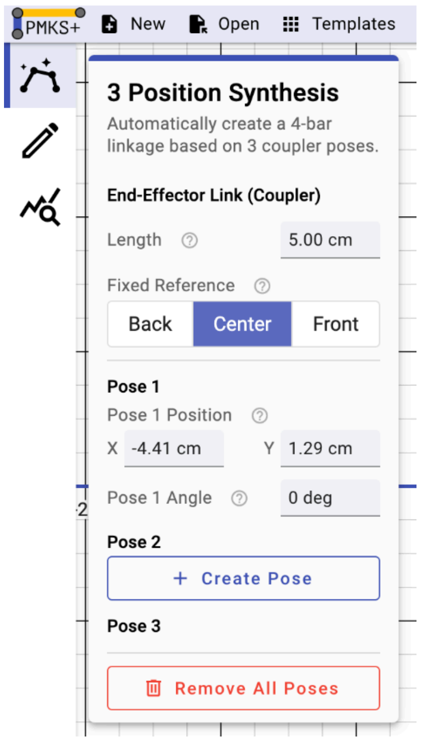
The input speed is set to 20.00 RPM in the clockwise direction. This can be changed in the settings panel.

Figure 5.4.4.1 PMKS+2023 Analysis Panel

As with many of the other panels, there was inconsistency with text and iconography that we needed to address. Differences in angle displays from the Edit panel to the Analysis panel were one example of this. The graphs themselves were not very intuitive, as axes were not clearly labeled, and many mechanisms had graphs whose axes did not scale in a readable way. On the graphs, users were unable to intuitively read points at specific time steps without attempting to pause the simulated mechanism in motion. All of these combined to make a panel that was able to display information accurately but only if a user knew where to look and how to manipulate it. As the stated goal for our project was to improve the experience of the novice user, new alternatives had to be devised for the Analysis panel which are discussed in section 6.3.

5.4.5 Synthesis Panel Review

Much of the PMKS+2023 Synthesis panel (shown in figure 5.4.5.1) design needed work and the text needed to be re-written to use standard terminology. There were also issues with the way the couplers were displayed on the grid. The main text changes were as follows : “3” needs to be “Three”, “Pose” should say “Position”, “end-effector link” should be replaced by “Coupler”, and all coordinates should be lowercase. The length and angle text boxes should have their standard icons next to the text boxes. There needs to be an option that allows the user to generate a four-bar or a six-bar mechanism. Note that three position synthesis can only generate a four-bar mechanism. Changing that to a six-bar would only require changing the location of the crank. Allowing users to generate a six-bar mechanism would allow the input to rotate a full 360º. All of the buttons need to be displayed more rounded to signify more clearly that they are buttons and can be clicked. When the user creates a position, each position is initially created with its reference point at the origin. This places new positions on top of each other which makes it hard for the user to see that there are multiple positions there. Also, the “Create Pose” button should not say create and should say “Specify Position” because you are not creating a position, you are specifying the position that the coupler should pass through.



3 Position Synthesis

Automatically create a 4-bar linkage based on 3 coupler poses.

End-Effector Link (Coupler)

Length 5.00 cm

Fixed Reference: Back | [✓ Center] | Front

Pose 1

Pose 1 Position: X -4.41 cm, Y 1.29 cm

Pose 1 Angle 0 deg

Pose 2

Pose 3

Figure 5.4.5.1 PMKS+2023 Synthesis Panel

The coupler that is positioned on the grid as part of Three Position Synthesis in PMKS+2023 (shown in figure 5.4.5.2) displayed a lot of information and had the potential to confuse users. They have blue and red arrows, a black circle and a green circle all inside the small coupler. If the coupler is smaller, all of these elements are not responsive to the size and will appear outside of the coupler, which could be confusing. These elements are confusing to the user as they are not descriptive or necessary. The red and blue arrows allow the user to move the coupler vertically or horizontally in a straight line if they are selected. If the user selects the coupler any other way, the coupler will move freely. The green circle allows the user to rotate the coupler about the reference point, which is represented by the black circle. Since all of these values can also be changed by inputting the values in the text boxes in the panel, these arrows and circles are unnecessary.

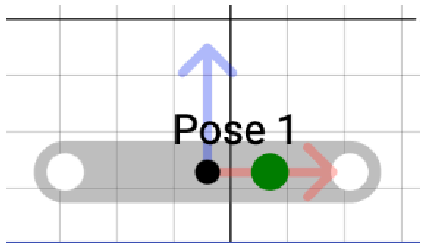


Figure 5.4.5.2 PMKS+2023 Synthesis Coupler

There are also features that need to be added to the Synthesis panel. In the Edit panel, if you hover over the angle text box, the value of the angle appears on the grid. Adding this to the Synthesis panel would allow for more consistency across the application. Another required feature is the ability to delete individual coupler positions instead of having one button to remove all positions. There also needs to be a popup that appears to confirm that you would like to delete the specific position. Another new button that should be added is a “generate mechanism” button. The current application synthesizes a four-bar mechanism as soon as the third position is added even before the user has updated the position of the coupler. The “Generate Mechanism” button would allow a user to prepare their positions in their desired coordinates, before they start to see a mechanism appear. The mockups created based on this feedback can be found in section 6.1.

5.4.6 Other Application Synthesis UI Review

In order to create our redesign, we needed to analyze the way other applications presented information. We looked specifically at how SAM, MotionGen and PMKS+2022 presented path synthesis (Mad-croc nederland, n.d; Purwar et al, 2017; Galvan, 2022). Each system had their own way of inputting the data. SAM allows the user to upload a text file with a list of values for a path and it then creates a path based on those values. The system then makes 100 iterations to attempt to find the mechanism that most closely matches the path created. PMKS+2022 allows you to click on the screen to input the points for the path. Then the system approximates a path using those points and outputs a list of mechanisms that can be created. This was purely a front-end feature that had no corresponding code in the back end. MotionGen allows the user to simply draw a path as well as input a file with the path listed as coordinates. The system then outputs multiple mechanisms that match the path as closely as possible. In SAM, you can specify the area in which you want the input or the ground to be placed. MotionGen allows you to draw constraints to create a mechanism within a specified area. Based on this information we used Figma to create mockups (shown in section 6.1) that would best represent synthesis to the user (Figma, 2023). We created multiple designs using inspiration from MotionGen and SAM.

5.5 Refactoring the PMKS+2023 Codebase

In tandem with reimagining the front end UI of PMKS+, our team began designing and writing the full refactor of the codebase. This refactor would completely restructure the codebase to improve the codebase without changing its functionality. Although the application was functional in its existing state, the implementation was cumbersome at best. The refactor needed to address several major issues found to be prevalent in the previous version. As a whole the design and implementation of the refactor needed to accomplish several things. The first of these goals was deciding on an architecture for the entire application. Additionally we needed to adhere to the SOLID design principles: Single-Responsibility, Open-closed, Liskov Substitution, Interface Segregation, and Dependency Inversion (Chris, 2023). Lastly, the refactor must maintain best code practices (documentation, helper functions, well named variables/methods, and limited nesting) to improve the onboarding of future development teams, and simplify the addition of new features. The link to the codebase (Github link) and instructions for how to run the code can be found in Appendix K.

5.5.1 Comprehensive Initial Evaluation and Strategic Planning

We began with an exhaustive evaluation of the PMKS+2023 codebase. We engaged in collaborative sessions with the students from PMKS+2022 and PMKS+2023 to acquire a deep understanding of the existing system's architecture and style of code writing. This process was critical for identifying the issues and pitfalls that previous development teams encountered during their development.

Our first discovery was the lack of a chosen architecture for the application. This resulted in a small number of classes which contained a bulk of the logic for drawing to the screen, interpreting user input, and updating the state. Additionally, these responsibilities were divided between all of these classes. This resulted in instances where the function to add a component to a mechanism and the function to remove that same component were in different classes. We also found that the code representing the state (the mechanism) itself was missing clearly defined class relationships. The classes representing joints and links contained identical state tracking that complicated updates to the model dramatically. Due to this complication of the state, we found many functions exceeding a length of 300 lines of code, with nesting depths ranging from six to nine. In programming, nesting is the insertion of code within a control flow line of code. An example of a nesting depth of two could be an if statement within a for-loop. The for-loop is the first nest while the if statement is the second, giving the code within the if statement a nesting depth of two. Each additional layer is multiplicative in respect to the number of possible execution paths. This in turn requires a developer to remember each layer that results in the execution of the code they are analyzing. Lastly, the code itself lacked documentation, many methods/variables were poorly named, and the comments we did find consisted of several "TODO"s and sections of obsolete code.

Given how prevalent these issues were throughout the application, our team decided to rebuild PMKS+ from the ground up. Keeping in mind how long it has taken the application to reach its current feature set, we decided to operate our project in sprints (a set time interval in which a particular set of tasks is completed). By utilizing sprints, we would ensure that at the end of the project we would have a working application with some subset of the current features. We then defined a minimum viable product for the end of our second term. The feature set was limited to recreating the central grid of the application. This included a grid which could pan and zoom, dragging elements around the screen, and a right-click menu for adding and modifying mechanism components.

5.5.2 Architecture Selection Process

Once we had created a plan for the refactor, the team member responsible for implementing the minimum viable product began designing the new architecture. Several factors were considered during the architecture selection process. We wanted to choose an architecture with which most future developers from WPI would already be familiar. This limited our choices to architectures taught within the Software Engineering and Design classes offered here at WPI. Additionally, the architecture needed to clearly map onto the piece of software we were building, hence we ruled out most architectures that primarily dealt with databases and external servers.

These requirements led the team to choose the MVC (Model, View, Controller) architecture. This architecture creates a division of responsibility into three clear buckets. The Model code is responsible for handling the application's internal representation of state. For PMKS+2024, that was the current state of the mechanism, as well as the algorithms that calculate the analysis of the state. The View section of the code is responsible for reading the Model, representing it to the user, and passing user input to the Controller. Finally, the Controller section of the code handles user input sent from View, translates it to a desired change in state, and communicates to the Model what needs to be updated.

5.5.3 Implementing an Improved Model for State

With an architecture decided, and a minimum viable product outlined, we began the refactor by designing and implementing a new model. The design phase consisted of two stages. The first was simplifying the representation of the mechanism. The process took several meetings with the previous team that worked on PMKS+. During these meetings, we discussed each of the classes, their variables, and their responsibilities at length. We removed as much unnecessary code as possible while still maintaining the minimum amount of data required to represent a mechanism fully and allow for additional functionality in future.

Some of this trimming removed what we dubbed "nonessential characteristics"– this was data that could be quickly calculated based on the values of other properties, or were not relevant to what that component represents from a mechanical engineering point of view. These included centers of mass, whether a component is currently highlighted on the screen, color, and SVG data containing how to draw an element.

The other major source of removing complexity was a consolidation of and transferring of class responsibility. Most of this consolidation occurred in the mechanism class. We concluded that any functions or data that required multiple components to calculate should be moved to the mechanism class. As an example, joints should not have properties that contain data for which links it is a part of and what other joints are connected to it. It is much simpler for the mechanism class to implement this property as a method that takes the ID of a joint as a parameter.

Since the mechanism class represents the entire model, we wanted to force future developers to make any changes to the state through the Mechanism class only. To accomplish this, we took every instance where the state is changed in the old system and created a method for it in the new Mechanism class. This way, all changes to the state are consolidated into a single interface that takes requests for a state change, checks if that change is valid, then performs the change. This is in contrast to PMKS+2023 where alterations to the state existed across multiple classes.

We also devised a new class to handle the complicated functionality of welding joints. When a joint is part of multiple links, it can be welded. Once welded, all links attached to this joint cannot move relative to each other and act as one link (Figure 5.5.3.1). However, once a joint is unwelded, the links become separate again. By creating a new compound link class, we were able to both simplify their representation in the model, and fix a bug in the previous system that prevented links from becoming unwelded.

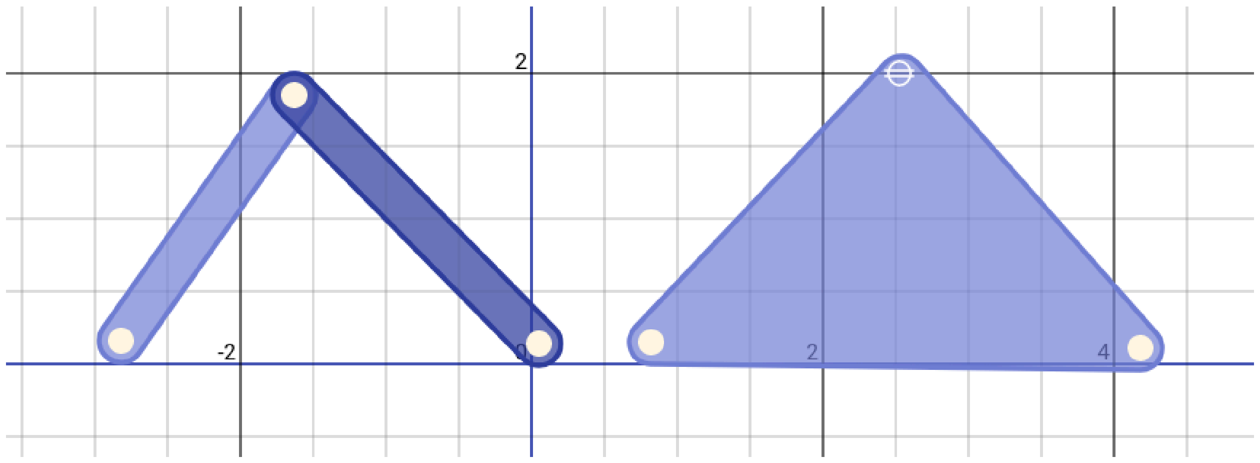


Figure 5.5.3.1 Link welding Reference

During the course of our work on the codebase, the class diagram for simulating the model existed as laid out in Figure 5.5.3.2:

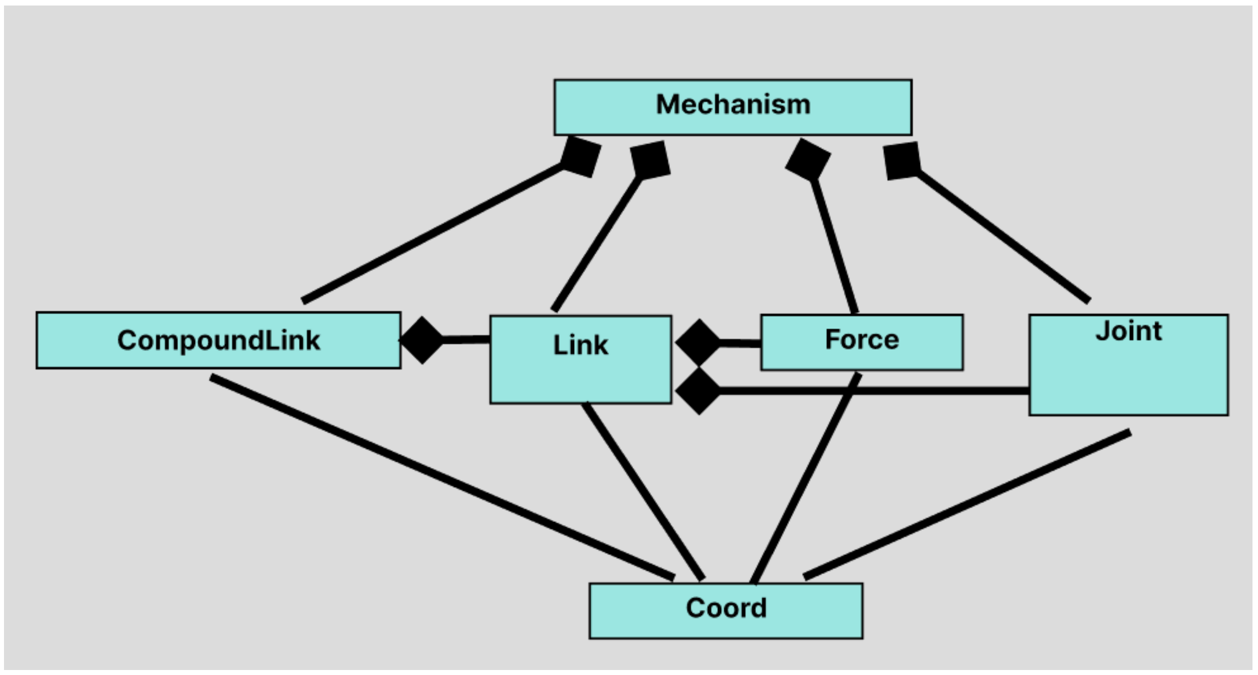


Figure 5.5.3.2 New Model Class Diagram

5.5.4 Developing Controllers for the Model

Once a new model representing mechanisms was completed, the team member moved onto designing and implementing the controllers for the minimum viable product. The outline for the controller design was quite simple. A parent class would handle initial interactions with the screen/grid, and then after performing some checks, pass that input to the controller for the particular component the input came from.

The parent class became necessary to ensure the smooth operation of different functionality. The class contains access to all controllers, allowing it to track which components are currently selected. It also handles entering and exiting unique modes of interacting, such as creating a link. Finally, this parent controller class determined when and where to show the right click context menu.

The controllers for each component were implemented quite simply. Each one extends a shared abstract controller. Additionally, they make calls to the model during dragging, generate the context menu available for their respective component, and are responsible for modifying the state after a "click-capture". These click-captures are interactions that require storing two mouse inputs, the most simple example is adding a link to a grid. The interaction involves the user right clicking on the grid and the parent controller then shows the context menu with the options generated by the grid controller (Figure 5.5.4.1 a, b). The grid controller also stores the location of the initial right click. The user then selects the "add link" option, and the grid controller creates a LinkFromGrid click capture. Once the user selects a second point on the screen, the parent controller calls the on-click function of the click-capture, which sends a request to update the state (Figure 5.5.4.1 c, d).

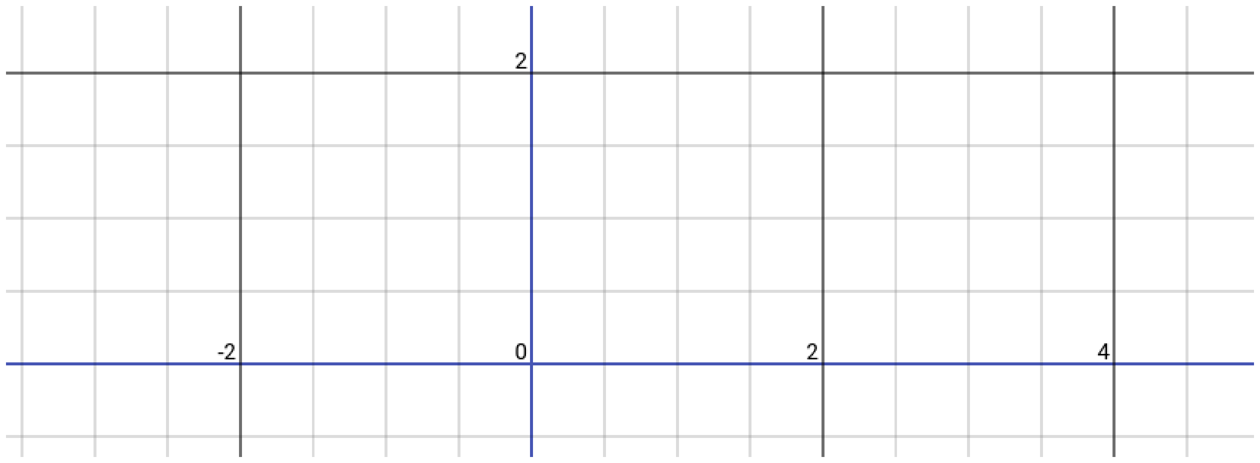


Figure 5.5.4.1 a) Grid

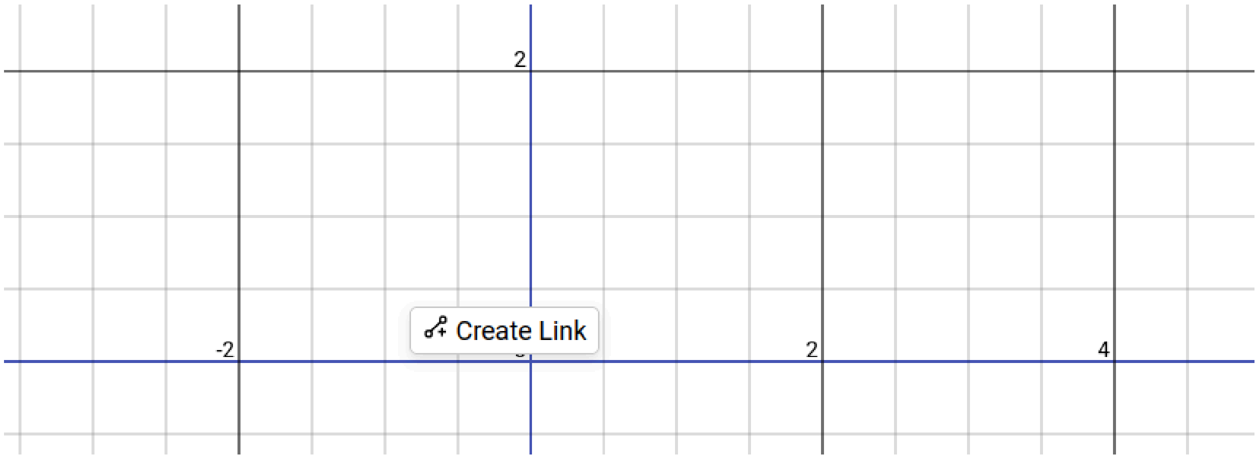


Figure 5.5.4.1 b) Context Menu

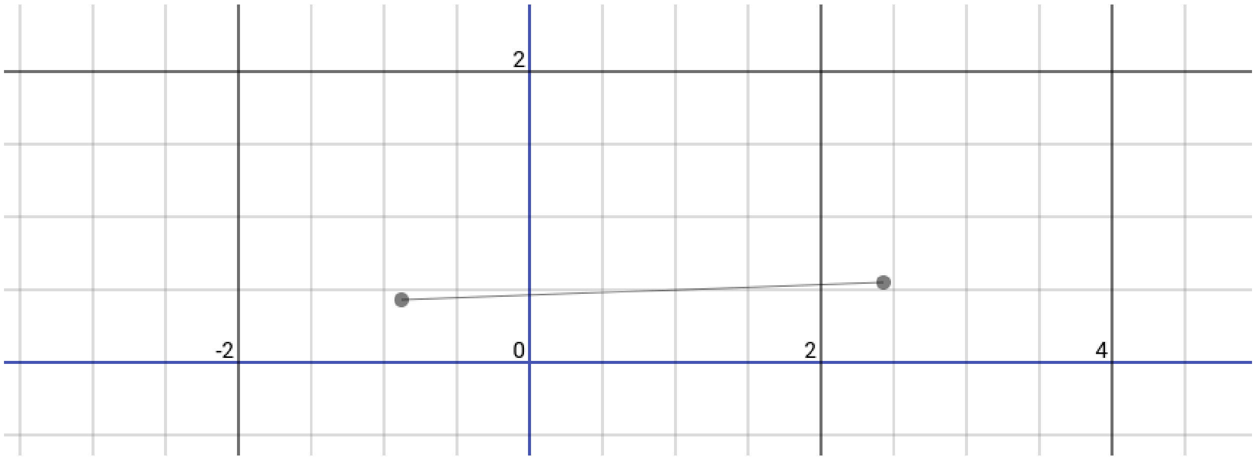


Figure 5.5.4.1 c) Dragging to Create Link

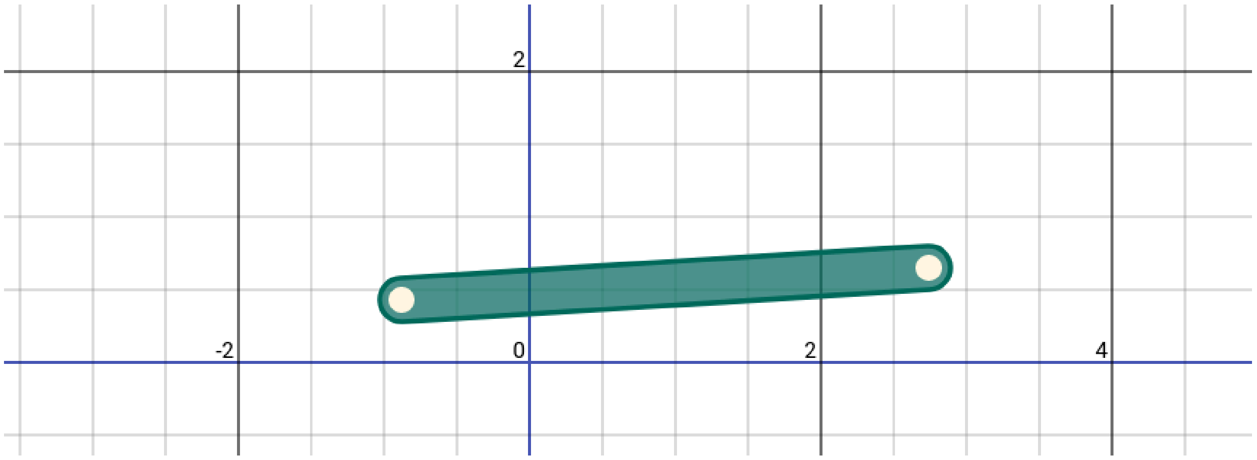


Figure 5.5.4.1 d) Created Link

Once completed the full class diagram of the application at this point was as shown in Figure 5.5.4.2:

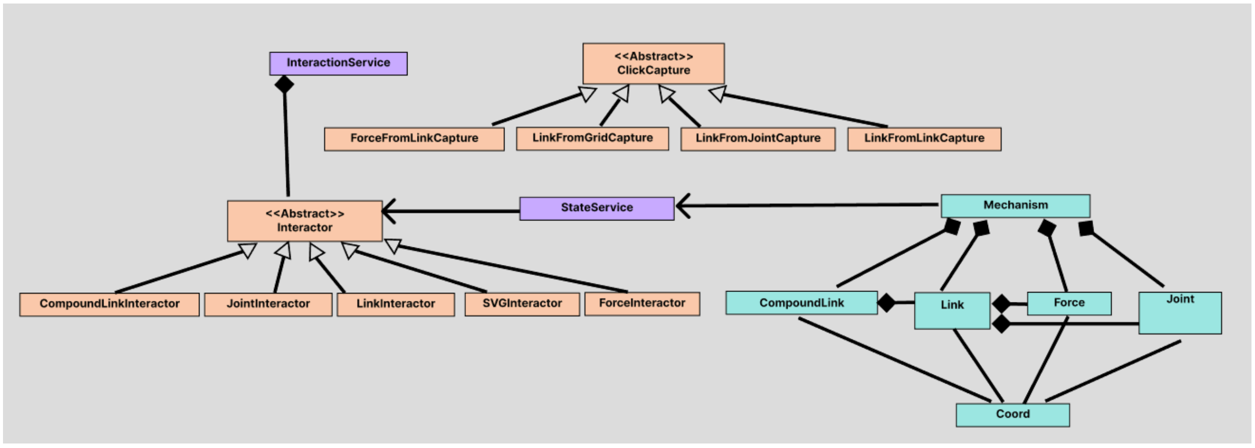
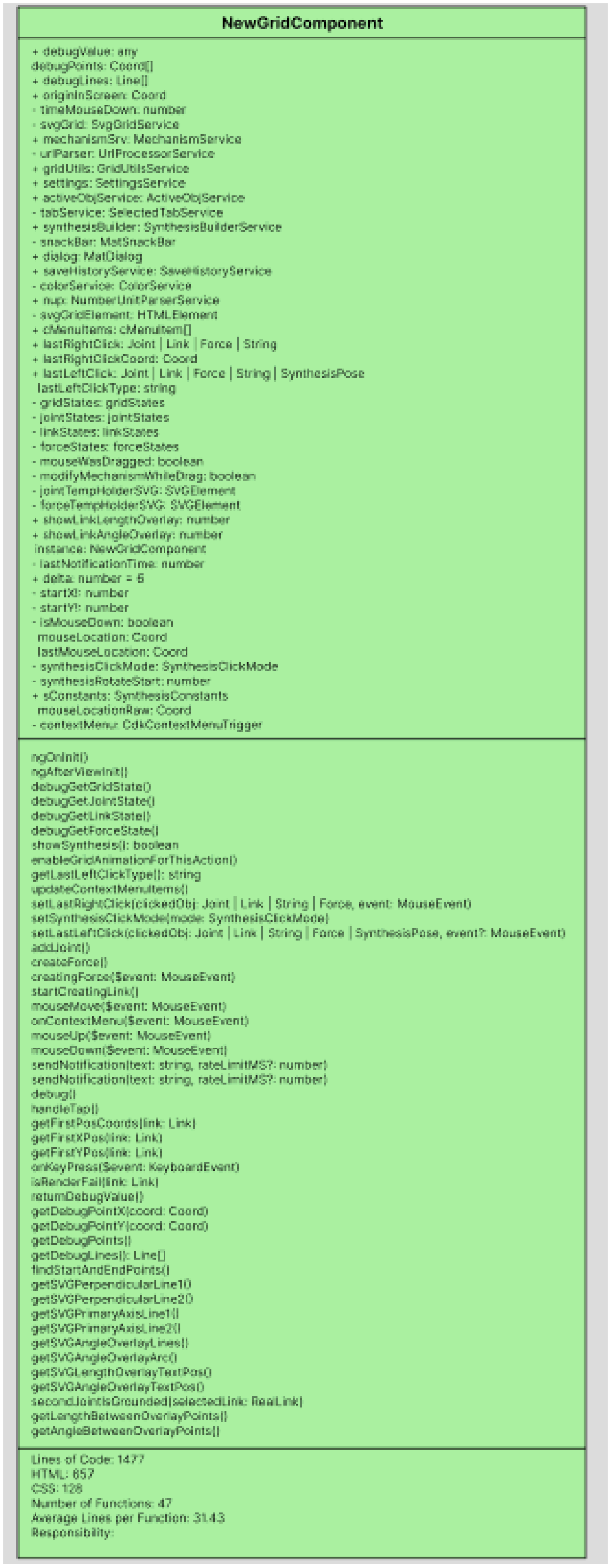


Figure 5.5.4.2 Full Application Class Diagram

5.5.5 Devising the Architecture for the View

With a first pass of the model and controllers completed, we began designing the architecture for the view section of the application. In general, the organization of the user interface within the PMKS+2023 was well divided between different components based on responsibility. However, we found that three components made much more sense to be divided into separate classes. The first two were the panels for editing the mechanism and for analyzing the mechanism. These were addressed earlier. The last class however, was the New-Grid-Component, which was responsible for handling panning and zooming updates, drawing the gridlines on the SVG, displaying joints, links, compound links, forces, and the context menu. Additionally, this class tracked and handled all inputs on the grid, and several different updates to the state. The full class is shown below in Figure 5.5.5.1 (a clear version is available here).



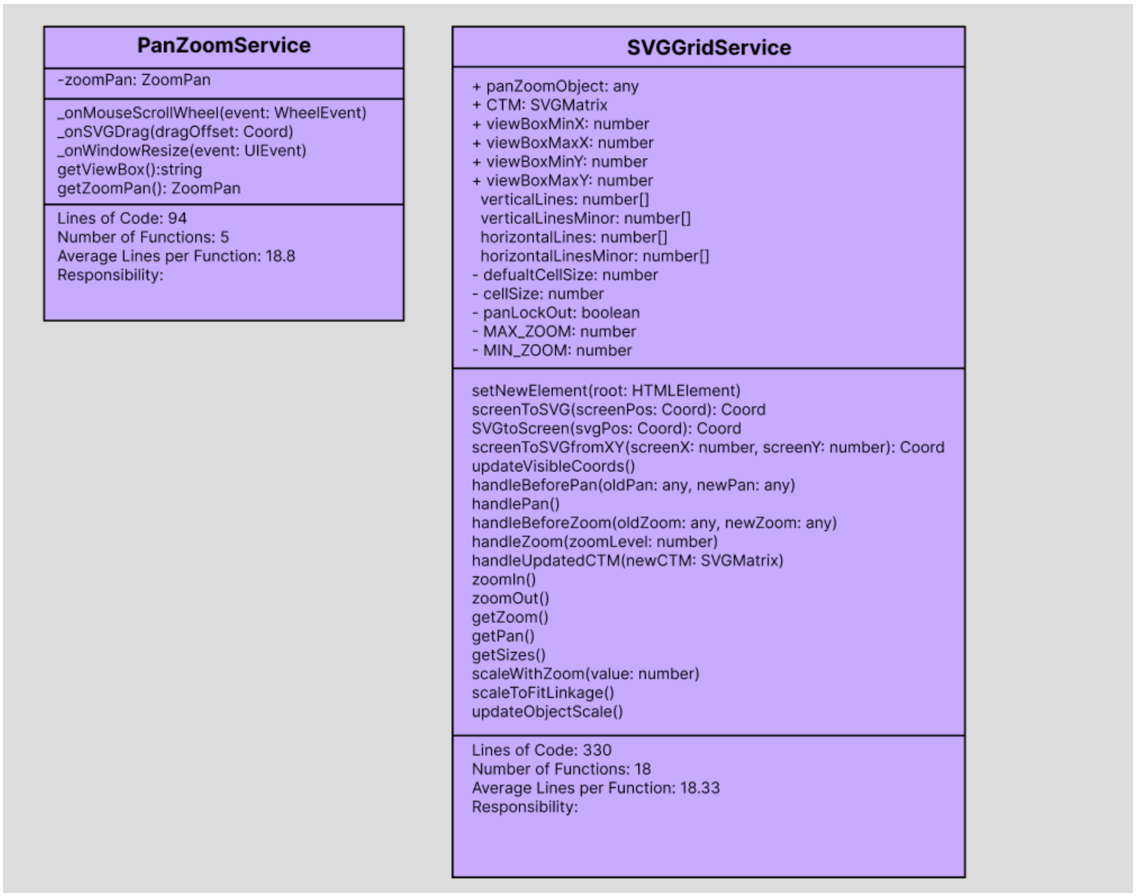
```
NewGridComponent

+ debugValue: any
debugPoints: Coord[]
+ debugLines: Line[]
+ originInScreen: Coord
- timeMouseDown: number
- svgGrid: SvgGridService
+ mechanismSrv: MechanismService
- urlParser: UrlProcessorService
+ gridUtils: GridUtilsService
+ settings: SettingsService
+ activeObjService: ActiveObjService
- tabService: SelectedTabService
+ synthesisBuilder: SynthesisBuilderService
- snackBar: MatSnackBar
+ dialog: MatDialog
+ saveHistoryService: SaveHistoryService
- colorService: ColorService
+ nup: NumberUnitParserService
- svgGridElement: HTMLElement
+ cMenuItems: cMenuItem[]
+ lastRightClick: Joint | Link | Force | String
+ lastRightClickCoord: Coord
+ lastLeftClick: Joint | Link | Force | String | SynthesisPose
 lastLeftClickType: string
- gridStates: gridStates
- jointStates: jointStates
- linkStates: linkStates
- forceStates: forceStates
- mouseWasDragged: boolean
- modifyMechanismWhileDrag: boolean
- jointTempHolderSVG: SVGElement
- forceTempHolderSVG: SVGElement
+ showLinkLengthOverlay: number
+ showLinkAngleOverlay: number
 instance: NewGridComponent
- lastNotificationTime: number
+ delta: number = 6
- startX!: number
- startY!: number
- isMouseDown: boolean
 mouseLocation: Coord
 lastMouseLocation: Coord
- synthesisClickMode: SynthesisClickMode
- synthesisRotateStart: number
+ sConstants: SynthesisConstants
 mouseLocationRaw: Coord
- contextMenu: CdkContextMenuTrigger

ngOnInit()
ngAfterViewInit()
debugGetGridState()
debugGetJointState()
debugGetLinkState()
debugGetForceState()
showSynthesis(): boolean
enableGridAnimationForThisAction()
getLastLeftClickType(): string
updateContextMenuItems()
setLastRightClick(clickedObj: Joint | Link | String | Force, event: MouseEvent)
setSynthesisClickMode(mode: SynthesisClickMode)
setLastLeftClick(clickedObj: Joint | Link | String | Force | SynthesisPose, event?: MouseEvent)
addJoint()
createForce()
creatingForce($event: MouseEvent)
startCreatingLink()
mouseMove($event: MouseEvent)
onContextMenu($event: MouseEvent)
mouseUp($event: MouseEvent)
mouseDown($event: MouseEvent)
sendNotification(text: string, rateLimitMS?: number)
sendNotification(text: string, rateLimitMS?: number)
debug()
handleTap()
getFirstPosCoords(link: Link)
getFirstXPos(link: Link)
getFirstYPos(link: Link)
onKeyPress($event: KeyboardEvent)
isRenderFail(link: Link)
returnDebugValue()
getDebugPointX(coord: Coord)
getDebugPointY(coord: Coord)
getDebugPoints()
getDebugLines(): Line[]
findStartAndEndPoints()
getSVGPerpendicularLine1()
getSVGPerpendicularLine2()
getSVGPrimaryAxisLine1()
getSVGPrimaryAxisLine2()
getSVGAngleOverlayLines()
getSVGAngleOverlayArc()
getSVGLengthOverlayTextPos()
getSVGAngleOverlayTextPos()
secondJointIsGrounded(selectedLink: RealLink)
getLengthBetweenOverlayPoints()
getAngleBetweenOverlayPoints()

Lines of Code: 1477
HTML: 657
CSS: 128
Number of Functions: 47
Average Lines per Function: 31.43
Responsibility:
```

Figure 5.5.5.1 New-Grid-Component Class Diagram

Our primary improvement to the architecture of these classes was dividing them into child components responsible only for representing a single aspect of the state. We also transferred much of the system logic present in these components to either our controller system or the new model representation.

5.5.6 Creating Views for the State

With a redesigned model, and a controller system in place, we began implementing the View portion of the minimum viable product. The first component to implement was the parent SVG component for the entire grid. This component contains the viewbox for the entire grid, and three child components. One component was solely responsible for drawing gridlines, another represented the parent component of the mechanism, and the third represented the context menu. Additionally, this component uses a custom written service to update the viewbox during dragging, zooming, and window resizing. This service was under 100 lines of code, and written from scratch, as opposed to the previous almost 400 lines of code utilizing an external library (Figure 5.5.6.1).



PanZoomService

```
-zoomPan: ZoomPan
```

```
_onMouseScrollWheel(event: WheelEvent)
_onSVGDrag(dragOffset: Coord)
_onWindowResize(event: UIEvent)
getViewBox():string
getZoomPan(): ZoomPan
```

Lines of Code: 94
Number of Functions: 5
Average Lines per Function: 18.8
Responsibility:

SVGGridService

```
+ panZoomObject: any
+ CTM: SVGMatrix
+ viewBoxMinX: number
+ viewBoxMaxX: number
+ viewBoxMinY: number
+ viewBoxMaxY: number
  verticalLines: number[]
  verticalLinesMinor: number[]
  horizontalLines: number[]
  horizontalLinesMinor: number[]
- defualtCellSize: number
- cellSize: number
- panLockOut: boolean
- MAX_ZOOM: number
- MIN_ZOOM: number
```

```
setNewElement(root: HTMLElement)
screenToSVG(screenPos: Coord): Coord
SVGtoScreen(svgPos: Coord): Coord
screenToSVGfromXY(screenX: number, screenY: number): Coord
updateVisibleCoords()
handleBeforePan(oldPan: any, newPan: any)
handlePan()
handleBeforeZoom(oldZoom: any, newZoom: any)
handleZoom(zoomLevel: number)
handleUpdatedCTM(newCTM: SVGMatrix)
zoomIn()
zoomOut()
getZoom()
getPan()
getSizes()
scaleWithZoom(value: number)
scaleToFitLinkage()
updateObjectScale()
```

Lines of Code: 330
Number of Functions: 18
Average Lines per Function: 18.33
Responsibility:

Figure 5.5.6.1 Side-By-Side Comparison of Two Services

The gridlines component was written next, and was very straight forward. By utilizing the values of the viewbox, it draws the major and minor gridlines, axes, and labels. From here, we began to draw the state of the mechanism onto the grid. The parent component for the mechanism simply iterates over all of the joints, links, compound links and forces, and creates a new child component to handle drawing each one.

The view representing a joint was created next. This component takes a single joint as its input, creates its own controller, registers that controller with parent controller, and draws the joint on the SVG after converting its coordinate position in meters (the industry standard) to a position on the screen. In addition to displaying the joint as a circle on the screen, the component also displays different images on the screen to represent different combinations of joint states.

The views for both the link and compound link were constructed in a similar manner. The one notable exception being the utilization of a service dedicated to generating the string that represents the SVG path of the link. The methods for this generation existed in the PMKS+2023 within the link class of the model. These were simplified, had helper functions created and commented, before being extracted into service within the refactored code.

During this stage of development, the minimum viable product was delivered. Almost all of the functionality from the grid apart from forces had been created, with a new controller system, and model implemented. Mechanisms could be created, modified, and dragged around with a new multi-select feature. Additionally, the toolbar, and side navigation components had been added as templates for the rest of the team. At this point, several previous bugs regarding the properly modifying state were also nonexistent in the refactor. Figure 5.5.6.2 shows the class diagram of the refactored PMKS+ at this state of development.

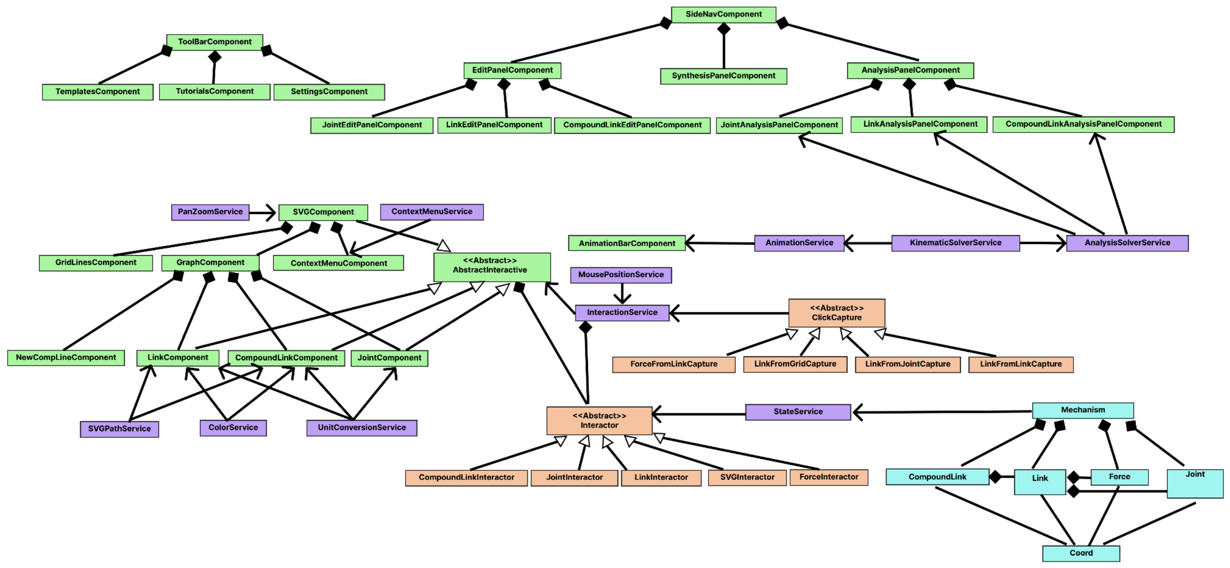


Figure 5.5.6.2 PMKS+2024 Class Diagram

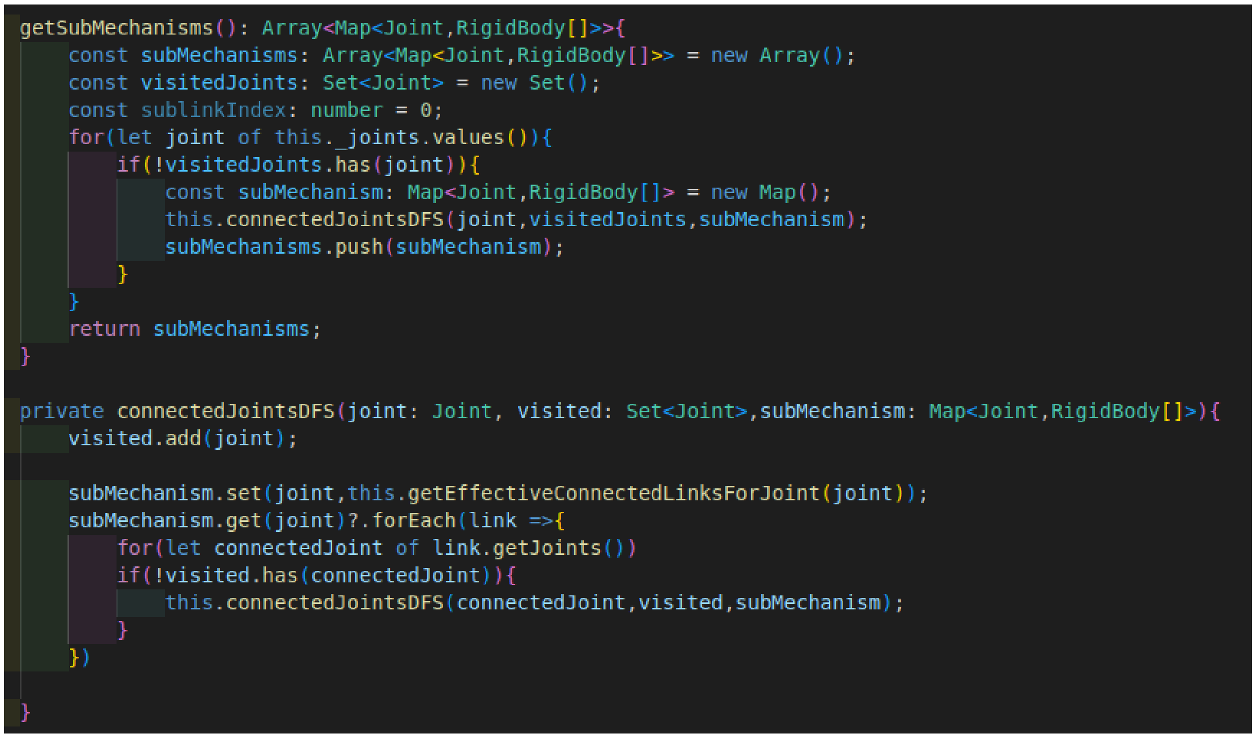
5.5.7 Refactoring the Simulator

With the completion of this new application containing a small subset of the features from the original, we moved onto refactoring the classes responsible for simulating the mechanisms. We devised an order of completion to provide the most crucial features first followed by other functionality as time permitted. Additionally, we knew that the support for simulating multiple mechanisms would reside within these simulator classes and aimed to add that support during our development.

5.5.8 Refactoring the Position Solver

The first of the simulator classes to be refactored was the position solver. We first met with our advisor Professor Radhakrishnan to understand how the algorithms used to determine positions were performed. We then conducted several meetings with Alex Galvan, the student who wrote the previous version of the simulator, to understand how these algorithms were translated into code within the existing PMKS+2023 system. From these meetings, we divided the position solver into four steps, generating independent mechanisms to be simulated, verifying that these mechanisms were valid, determining the order joints in these mechanisms will be solved, and finally generating the positions of the joints for each mechanism.

The largest difficulty in completing the first step was determining the optimal data structure to represent each mechanism to be simulated. This decision was crucial, as we needed a representation that easily allowed for retrieval of data needed to perform our algorithms. Our team opted for a hashmap containing Joints as keys with an array of RigidBodies, an interface implemented by the link and compound link classes with a getJoints method, as the value. This data structure allowed us to represent the mechanism as a form of graph, with the joints acting as vertices, and the rigidbodies becoming a pseudo-edge that can connect multiple vertices. We then built a simple depth first search algorithm within the mechanism class that generated the set of disconnected graphs (Figure 5.5.8.1). Each of these graphs was a mechanism to be simulated separately.



```
getSubMechanisms(): Array<Map<Joint,RigidBody[]>>{
    const subMechanisms: Array<Map<Joint,RigidBody[]>> = new Array();
    const visitedJoints: Set<Joint> = new Set();
    const sublinkIndex: number = 0;
    for(let joint of this._joints.values()){
        if(!visitedJoints.has(joint)){
            const subMechanism: Map<Joint,RigidBody[]> = new Map();
            this.connectedJointsDFS(joint,visitedJoints,subMechanism);
            subMechanisms.push(subMechanism);
        }
    }
    return subMechanisms;
}

private connectedJointsDFS(joint: Joint, visited: Set<Joint>,subMechanism: Map<Joint,RigidBody[]>){
    visited.add(joint);

    subMechanism.set(joint,this.getEffectiveConnectedLinksForJoint(joint));
    subMechanism.get(joint)?.forEach(link =>{
        for(let connectedJoint of link.getJoints())
        if(!visited.has(connectedJoint)){
            this.connectedJointsDFS(connectedJoint,visited,subMechanism);
        }
    })

}
```

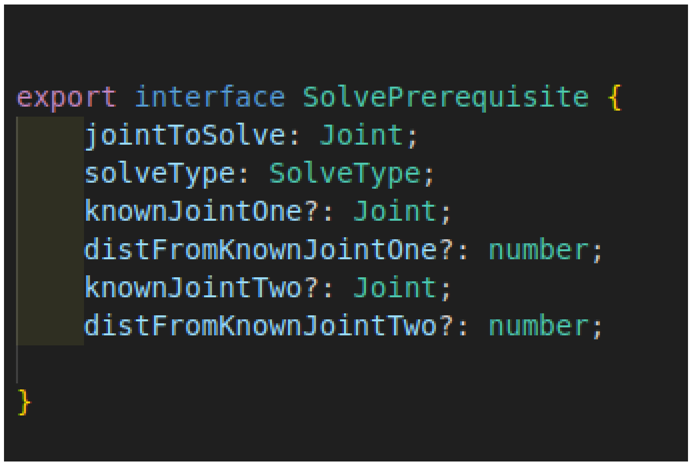
Figure 5.5.8.1 Depth First Search algorithm

The second step for solving the positions was calculating the validity of each independent mechanism. This check included three phases that guaranteed a mechanism was formed correctly. One phase was ensuring the mechanism contained exactly one joint which was an input and grounded. Another was guaranteeing the degrees of freedom for the mechanism was one by using both Kutzback Equation and Grubler's Equation. Lastly, the input joint needed to have exactly three links between itself and the closest grounded joint.

The third part of the algorithm was determining the order the joint data needed to be solved. This step took the team the longest to complete, requiring multiple iterations to maintain a relatively low complexity. The difficulty lay within the conditions required to solve joints in different scenarios. A total of five different cases existed that contained very specific criteria:

1. Grounded revolute joints whose position did not change (including revolute inputs).
2. Prismatic joints that were the input for the mechanism.
3. Ungrounded revolute joints connected directly to a revolute input joint.
4. Prismatic joints connected to at least one joint that is solved.
5. Ungrounded revolute joints connected to at least two joints that are solved.

Given these five types of methods to solve the position for a joint, with some requiring that other joints must be solved first, we constructed a new data structure to better store this information. This structure included the joint itself, the method used to solve the joint, and potentially null fields to store references to the joints that were solved first. Figure 5.5.8.2 shows the data structure as exported in the code.



```
export interface SolvePrerequisite {
    jointToSolve: Joint;
    solveType: SolveType;
    knownJointOne?: Joint;
    distFromKnownJointOne?: number;
    knownJointTwo?: Joint;
    distFromKnownJointTwo?: number;

}
```

Figure 5.5.8.2 SolvePrerequisite Interface

The final step was to calculate all of the positions for the joints. Since we had a simple data structure that stored the order the joints will be solved for each timestep, and the requisite information to solve each joint, the implementation went quite smoothly. We imported and modified the functions used in PMKS+2023 for each case, while developing a simple loop that determined when the mechanism should switch directions, and when to finish calculating the positions. The last thing to determine was how to represent the position data once it was calculated. We settled on a 2D array of coordinates with rows representing time steps, and columns representing each joint.

5.5.9 Building an Animation System

The only prerequisite blocking us from building an animation system was obtaining the position data used to create the animations. Our team built this system by looping over the calculated position data and updating the positions of the joints in each mechanism before timing out until the next frame should be drawn. We also partially implemented the animation controls from the previous system, with play, pause and stop functionality. At this point, we were nearing the end of our development time within the project, and felt it necessary to move onto building the code for calculating the velocities and accelerations of joints and links.

5.5.10 Refactoring Kinematics Solver

Our final major piece of refactored code was the kinematics solver. We built this solver using the graphical method as it allowed us to employ both the previously calculated solve orders, and the same five cases for solving as our position solver. With this, we were able to loop over the positions of each mechanism, solving in the same order as our position solver, and write functions that handled generating velocity and acceleration data for each case. A similar method was employed for generating link kinematics data by working with the solutions of each joint within the link.

This chapter provides the path taken to create our final product. We discussed our methods for understanding the project and the work that had already been done on the project. We also discussed the surveys taken to get feedback about the existing application as well as our application. Finally we discussed the methods used to re-design the UI and backend of the application. The next chapter discusses the design choices made in re-designing the UI.

6. UI Development and Design

This chapter bridges the gaps between what we learned during our preliminary research into PMKS+2023, our UI review previously discussed, and our final implementation. We include examples of the design decisions we were faced with, and the pros and cons of each potential decision. To understand the content of this section, it is important to note the distinction between a tab and a panel. A tab refers to the button in the side navigation used to open a panel. A panel refers to the content displayed when clicking on a tab. The final implementation is discussed and shown in section 8.

6.1 Three Position Synthesis Panel Design

In redesigning the Three Position Synthesis panel, we originally decided to keep the tab design very similar to the way it was in PMKS+2023. However, after meeting and discussing the current UI with advisors Professor Brown and Professor Radhakrishnan, we decided there were things that needed to be added. In figure 6.1.1, the coupler is shown to be cluttered with arrows whose purposes are unclear to the user. To fix this, we removed all of the arrows and dots. The black dot on the left is the reference point. We replaced this with the standard icon for reference points, which is a circle with a cross through (figure 6.1.2). The purpose of the green dot in the middle was to rotate the coupler and you could click on the coupler to move it on the screen. We decided to make the interface more similar to how you would edit in the edit tab. To do this, we added a yellow outline to the coupler when you click to drag (shown in figure 6.1.2 a). In order to rotate the coupler, you can click a joint that is not a reference point which will highlight that joint in yellow and allow you to drag it to the desired angle (shown in figures 6.1.2 b and c). The ability to manipulate the values manually by entering the values into the text boxes is still allowed.

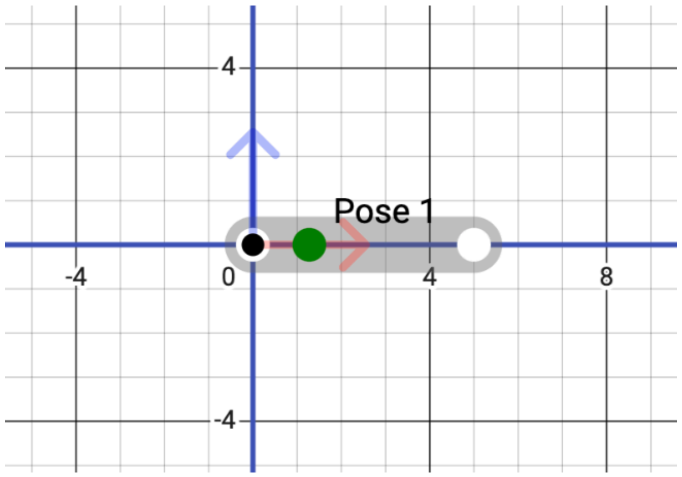
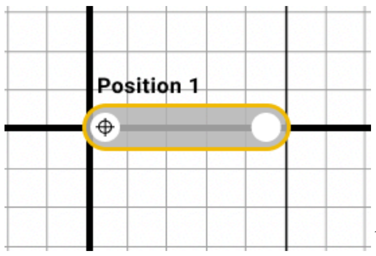
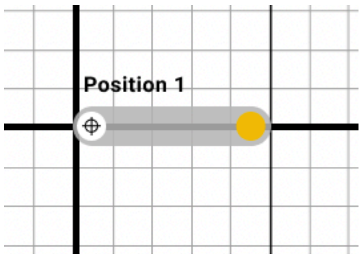


Figure 6.1.1 PMKS+2023 Three Position Synthesis Coupler Display

a) 

b) 

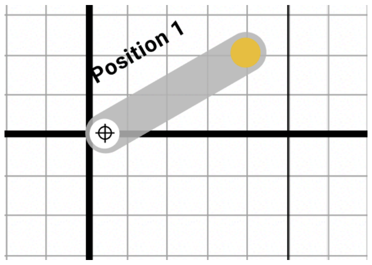
c) 

Figure 6.1.2 a) Proposed Three Position Synthesis Coupler Display: Selected Coupler

Figure 6.1.2 b) Proposed Three Position Synthesis Coupler Display: Selected Angle

Figure 6.1.2 c) Proposed Three Position Synthesis Coupler Display: Selected Angle After Rotation

Another concept that was added was the ability to delete individual coupler positions. This was not a feature that was available in the old version. We came up with 3 options to do this. The first was to create a context menu, similar to that which can be accessed in the edit tab. This context menu could be reached by right-clicking on the desired position. It is displayed with one option:"Delete Position" as shown in figure 6.1.3. The second was to put a trash icon next to each position on the tab (shown in figure 6.1.4). This would allow the user to be able to quickly delete any of the three positions. The last option was to have a button with a dropdown that would allow the user to select which position they would like to delete and if there are only two positions specified, then the third position would be grayed out (shown in figure 6.1.5). In the first two options there is also an option to delete all specified positions which would allow the user to easily start over. We inevitably decided it was best to include both the context menu and option 1 but were unable to implement this.

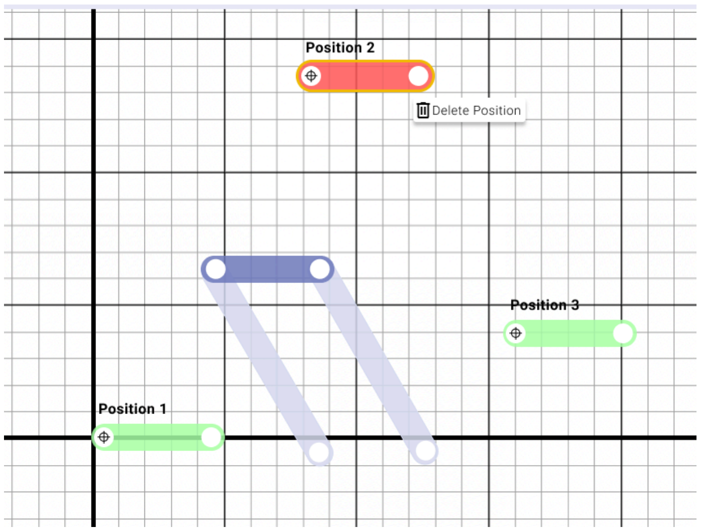
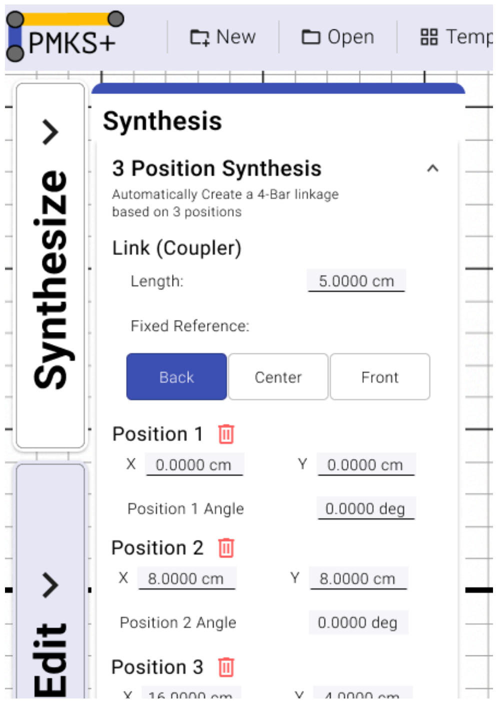


Figure 6.1.3 Proposed Three Position Synthesis Delete Position Context Menu



Synthesis

3 Position Synthesis

Automatically Create a 4-Bar linkage based on 3 positions

Link (Coupler)

Length: 5.0000 cm

Fixed Reference: Back

Position 1

X 0.0000 cm Y 0.0000 cm

Position 1 Angle 0.0000 deg

Position 2

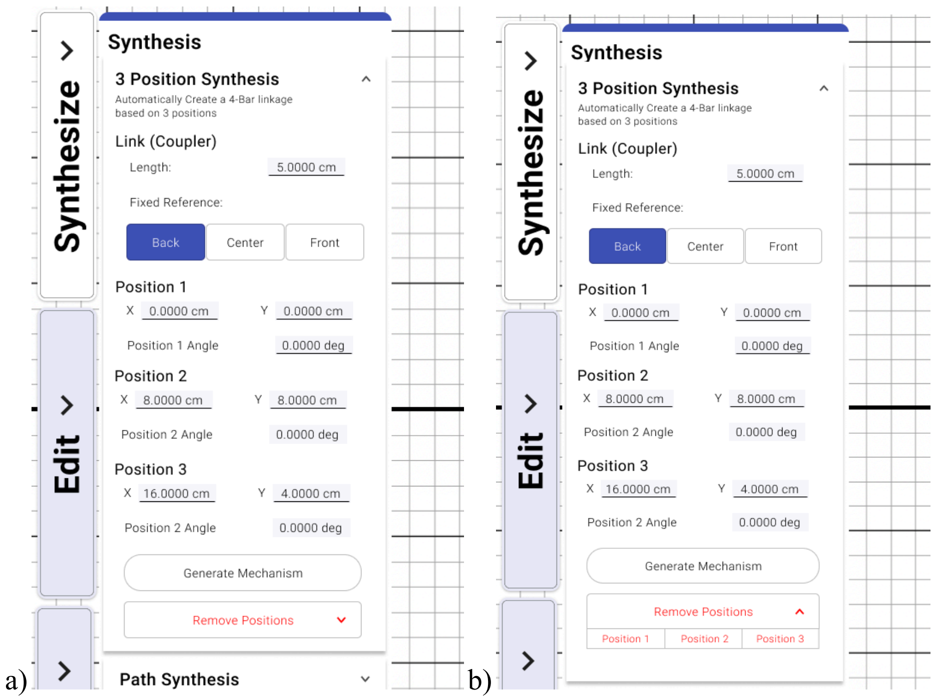
X 8.0000 cm Y 8.0000 cm

Position 2 Angle 0.0000 deg

Position 3

X 16.0000 cm Y 4.0000 cm

Figure 6.1.4 Proposed Three Position Synthesis Delete Position Option 1.



Synthesis

3 Position Synthesis

Automatically Create a 4-Bar linkage based on 3 positions

Link (Coupler)

Length: 5.0000 cm

Fixed Reference: Back

Position 1

X 0.0000 cm Y 0.0000 cm

Position 1 Angle 0.0000 deg

Position 2

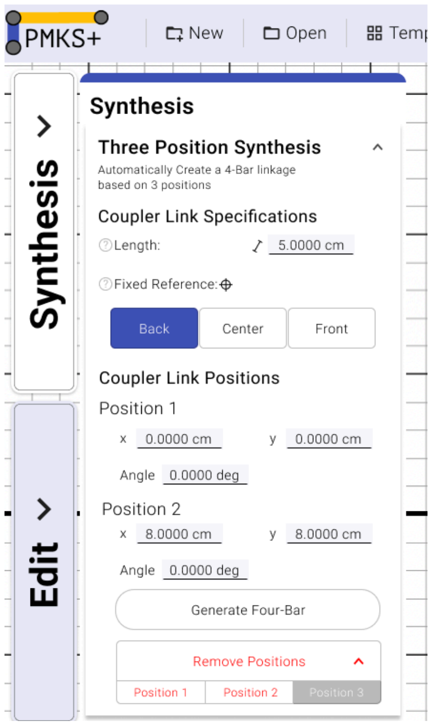
X 8.0000 cm Y 8.0000 cm

Position 2 Angle 0.0000 deg

Position 3

X 16.0000 cm Y 4.0000 cm

Position 2 Angle 0.0000 deg

c)

PMKS+
New
Open
Temp
Synthesis
Edit

Synthesis

Three Position Synthesis
Automatically Create a 4-Bar linkage based on 3 positions

Coupler Link Specifications

Length: 5.0000 cm

Fixed Reference:

Back | Center | Front

Coupler Link Positions

Position 1

x 0.0000 cm y 0.0000 cm

Angle 0.0000 deg

Position 2

x 8.0000 cm y 8.0000 cm

Angle 0.0000 deg

Generate Four-Bar

Remove Positions

Position 1 | Position 2 | Position 3

Figure 6.1.5 a) Proposed Three Position Synthesis Delete Position Option 2: Collapsed Menu

Figure 6.1.5 b) Proposed Three Position Synthesis Delete Position Option 2: Expanded Menu

Figure 6.1.5 c) Proposed Three Position Synthesis Delete Position Option 2: Position 3 Removed

6.2 Edit Panel Design

The Edit panel is one of the most important components of PMKS+. This is where the user spends a majority of their time. Creating linkages must be simple yet powerful, to allow novice users to easily complete assignments and allow experienced users to create complex linkages. The user needs to be able to edit links and joints separately. Additionally, some consideration must be made for compound links, as those can create critical components of any linkage. All three of these cases have to be accounted for in the broad “Edit panel”. Each of these panels (joint, link, and compound link Edit panels) had different necessities and potential interactions. For example, there would be no need to change the grounding of a joint when clicking on a link. The design of each subpanel is elaborated on below.

6.2.1 Links

In the PMKS+2023, the Edit panel was very basic (shown in figure 6.2.1.1) and did not include much information that was wanted by the user. The A-Term survey results described in Appendix B showed that many students wanted to be able to see more information about the link, including which joints were components of the link. The new Edit Link panel design consists of collapsable sections similar to those found in the synthesis and analysis panels. This allows the user to only look at what they need. There are three collapsible sections on this panel. The first section is labeled "Basic Settings," which consists of the link's length, angle, center of mass location, and buttons to add a tracer or force. Another change that was incorporated was the addition of the length icon replacing the "L" in PMKS+2023.The second is "Components," which consists of the joints that are components of that link. The third is "Visual Settings" which, like before, displays the link color choices. These changes are all shown in figure 6.2.1.2

Another key change was the edit and delete button placement and design. The new design is much more compact and consists only of icons representing the actions. These buttons can be hovered over to display their purpose. The pencil icon, when hovered over, has a popup that says "Edit Selected Link Name" and the trash icon, when hovered over, has a popup that says "Delete Selected Link".

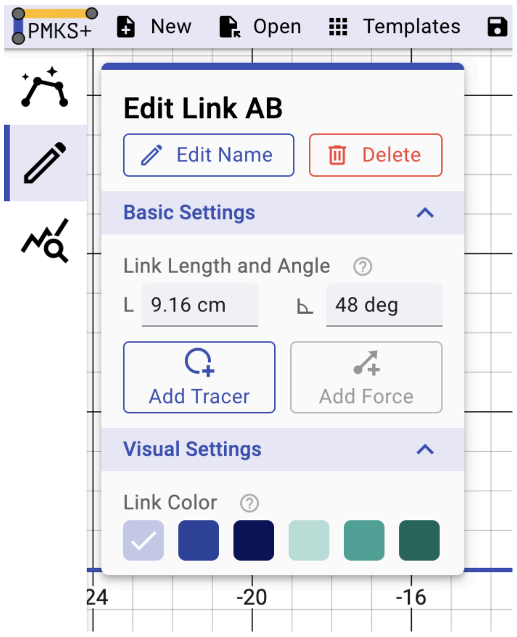
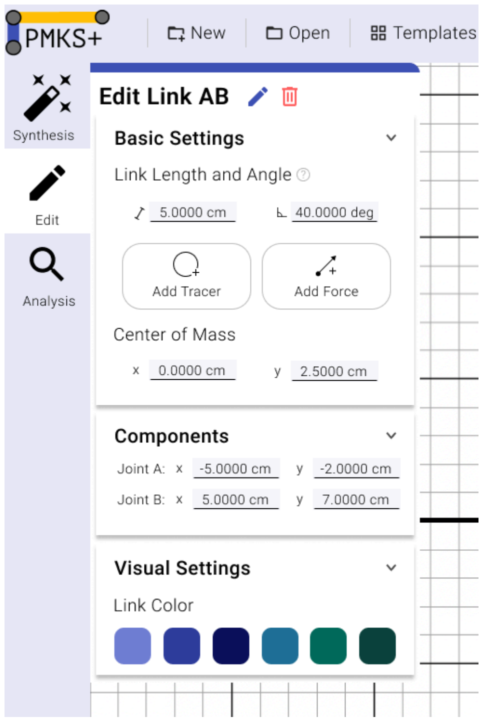


Figure 6.2.1.1 PMKS+2023 Link Edit Panel



Edit Link AB

Basic Settings

Link Length and Angle

5.0000 cm   40.0000 deg

Center of Mass

x 0.0000 cm   y 2.5000 cm

Components

Joint A: x -5.0000 cm   y -2.0000 cm

Joint B: x 5.0000 cm   y 7.0000 cm

Visual Settings

Link Color

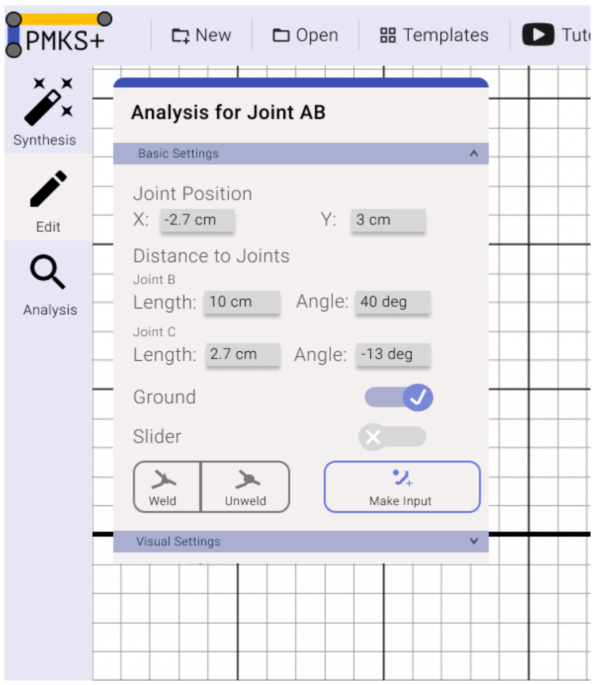
Figure 6.2.1.2 Proposed Link Edit Panel

We also changed the color palette used for links to include a more gradual change. When creating new links, the color was decided based on their ID in the backend and because of this, there is a drastic change between the darkest blue and the lightest green. To fix this, we used a darker color in between the dark blue and green shown above in figure 6.2.1.2.

6.2.2 Joints

Creating a more intuitive Edit panel for joints was straightforward. As we found in analyzing PMKS+2023, users appreciated intuitive and effective grouping of tools available to them. By separating out basic settings in the Edit panel such as position and distance to other joints, the user can more accurately assess their options for manipulating their mechanism (Figure 6.2.2.1). Further, by grouping tools such as welding, sliding, and grounding into the Advanced Settings dropdown, the user is more comfortable working their way up to more advanced concepts (Figure 6.2.2.2). Many of these editable qualities are also available by right clicking joints, which allows the user to engage with editing in whichever way is more comfortable and intuitive for them.

Several icons were developed to show the user what it was they were interacting with in a more productive way. Angles and lengths have standard icons that are used across mechanical and robotics engineering. This allows a student being introduced to this program to use any prior knowledge of general presentation of four-bar linkages to create mechanisms more quickly and intuitively.



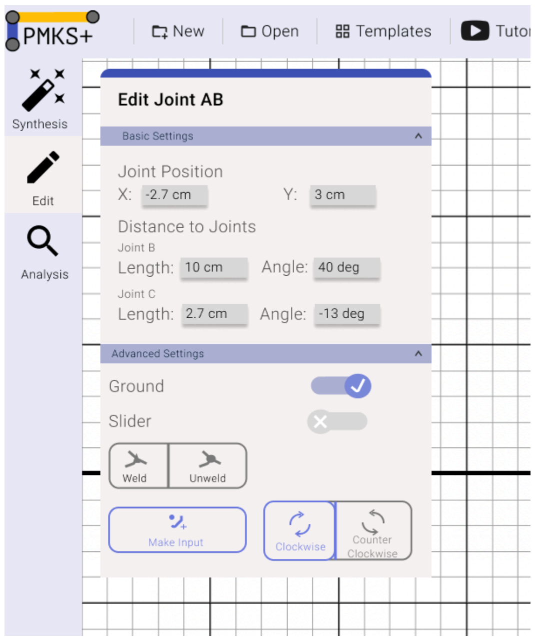
Analysis for Joint AB

Joint Position
X: -2.7 cm Y: 3 cm

Distance to Joints
Joint B
Length: 10 cm Angle: 40 deg
Joint C
Length: 2.7 cm Angle: -13 deg

Ground ✓
Slider ✗

Figure 6.2.2.1 Proposed Joint Edit Panel



Edit Joint AB

Joint Position
X: -2.7 cm Y: 3 cm

Distance to Joints
Joint B
Length: 10 cm Angle: 40 deg
Joint C
Length: 2.7 cm Angle: -13 deg

Ground ✓
Slider ✗

Figure 6.2.2.2 Proposed Joint Edit Panel Advanced Settings Menu

6.2.3 Compound Links

Where links are rigid bodies made by connecting two joints, compound links are rigid bodies made by welding two or more links together. The Edit panel for compound links in PMKS+2023 (shown in figure 6.2.3.1) was largely nonfunctional; no editing could be done, and the values displayed within any dynamic input fields were largely incorrect. To access the Compound Link Edit panel, the user first needed to navigate to settings, enable the Experimental Weld feature, weld two links together to create a compound link, and then click the newly created compound link. The screen would display each of the components, or sub-links, that were welded into the compound link. We sought to improve upon the usability of this feature and enable welding by default, to make the process of creating compound links more intuitive. It was also critical that we could display relevant and accurate information about the compound link and each of its sub-links.

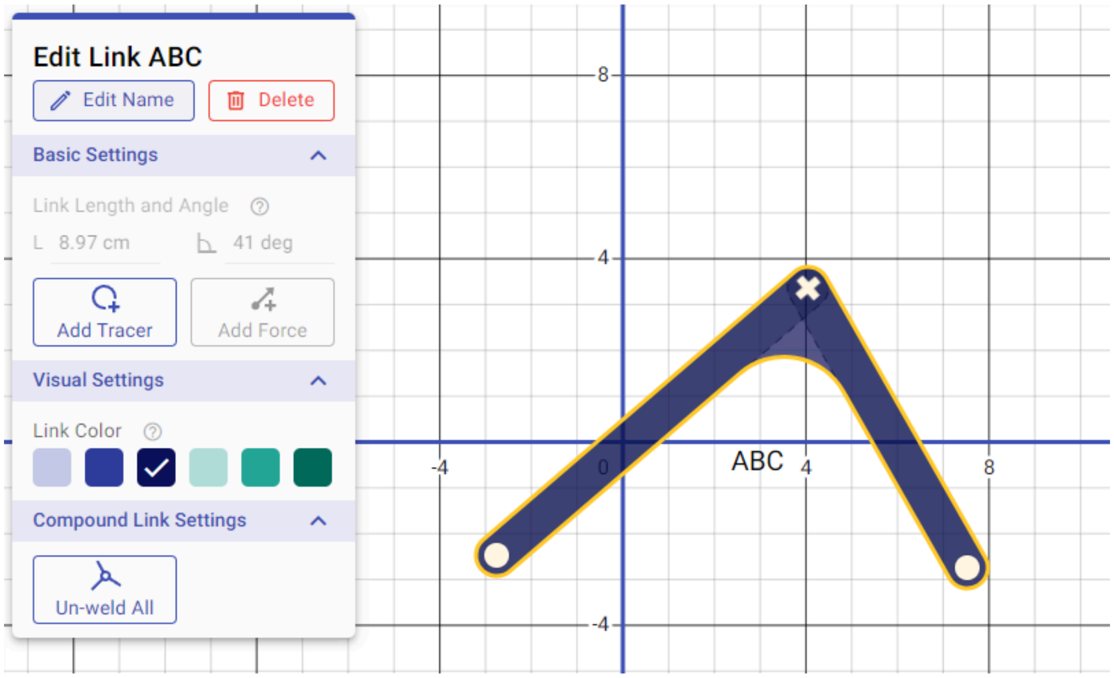


Figure 6.2.3.1 PMKS+2023 Compound Link Edit Panel

The reference joint on a compound link was necessary for a few reasons. Users had a need to keep the position of one or more joints consistent when editing compound links. Reference joints were a natural solution to allow users the ability to consistently see the position values of specific joints. Additionally, there was a need for users to be able to rotate compound links around a specific joint while maintaining the other properties of the compound link. Rotating compound links did not get implemented in PMKS+2024, but the basic tools exist for a future team to create this feature.

The sub-links of a compound link are stored, and their data is readily available, for the cases where users need to manipulate a specific component of a compound link, and where a user needs to unweld a compound link back into its previous components. These sub-links are drawn when hovering over a compound link. The position and angle values within each sub-link are displayed on the Edit panel, although they cannot be edited directly. The joints that make up each sublink (and thus the compound link) are also visible in the panel. This is all in an effort to keep the user informed about what makes up their compound link, and consequently how their mechanism is structured. Overall, our Compound Link Edit panel greatly improves the usability and usefulness compared to PMKS+2023.

6.2.4 Locking Links

Another new development in our program is the addition of "locking" links. Users may want to guarantee that specific links cannot be altered by editing other links or joints in their mechanism. We introduced the ability to lock specific links through their respective Edit panel. By selecting the Lock option, every attribute of a link or compound link becomes fixed. No action can be taken to edit any specific joint or link, and nothing can move a link or any of its joints on the grid. When a link is unlocked by using the Unlock button or context menu option, all of the attributes of the link become available for modification.

In order to make the process easy for the user and guarantee that they are aware of this feature, we added an option to our context menu allowing the locking / unlocking of joints as shown in figure 6.2.4.1. This option is directly next to other fundamental features such as adding links and welding joints. Additionally, we added a "Locked" icon to our grid, which displays over the center of mass of locked links and compound links as shown in figure 6.2.4.2. We were initially concerned about users locking pieces of their mechanism and then moving on, forgetting why they were unable to manipulate their mechanism. The lock icon acts as a visual clue to ensure a smooth workflow when working with locks.

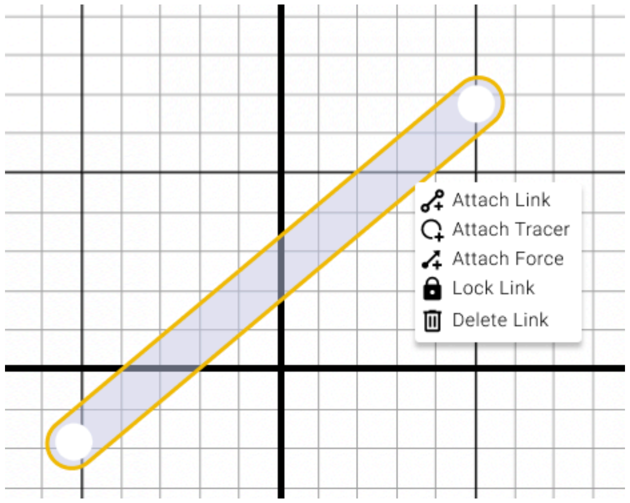


Figure 6.2.4.1 Proposed Locking Link Context Menu

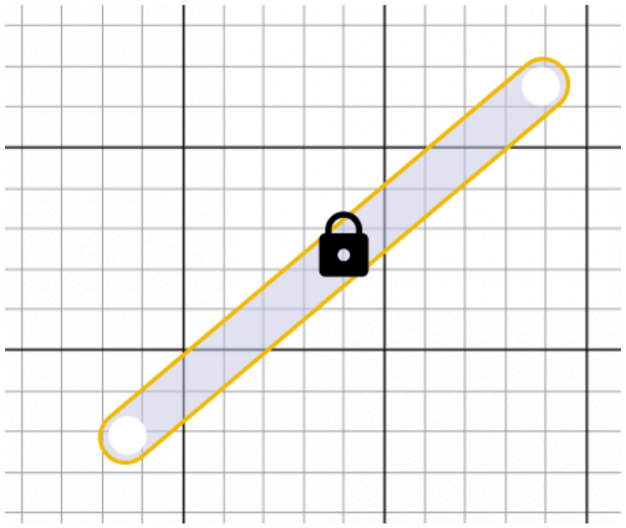


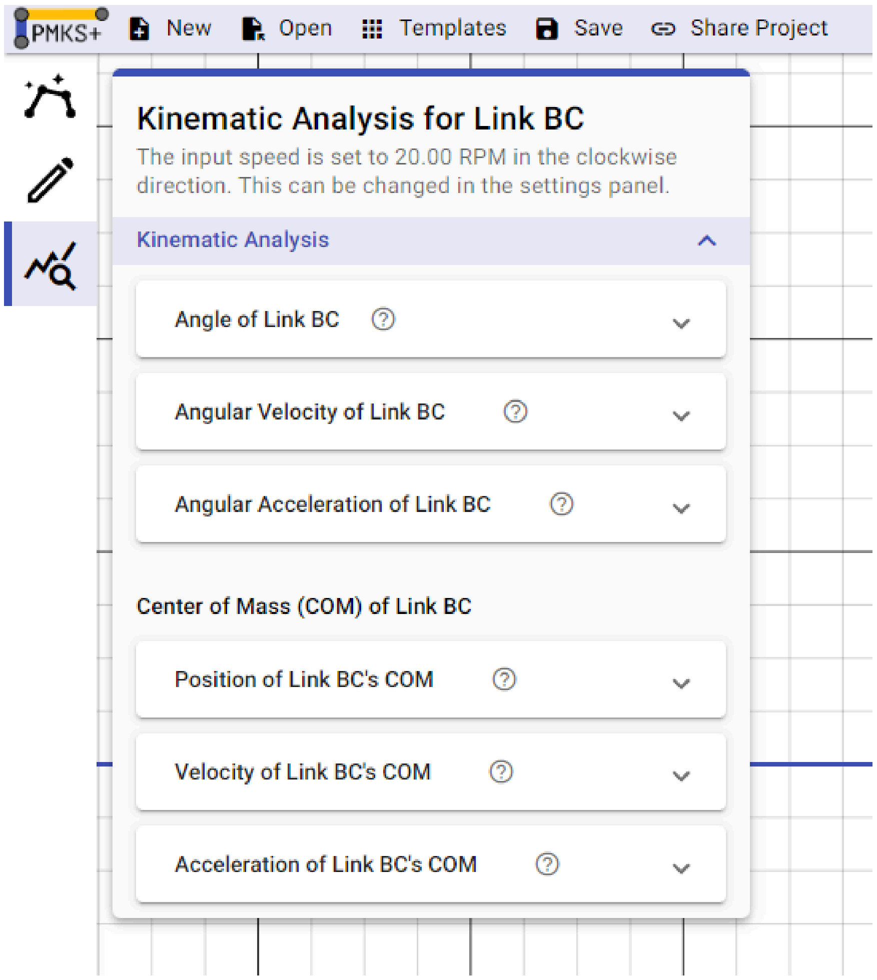
Figure 6.2.4.2 Proposed Locked Link

6.3 Analysis Panel Design

There were several considerations we had to take into account when redesigning the Analysis panel. This panel is the tool primarily used in homework assignments for which students would be using PMKS+. To ensure that we had a solid grasp of what we wanted the Analysis panel to evolve into, we first needed to understand what users currently could accomplish with the PMKS+ Analysis panel. Additionally, we needed to gather user feedback on what they would like to be able to do that would make the tool more versatile and intuitive. It should be mentioned here that one of our motivations affecting the design of the Analysis panel was future proofing it. PMKS+2023 only accounts for kinematic analysis of joints and links. In the future, it is assumed this tool will be able to handle force analysis, generate free body diagrams, and perform stress analysis. These considerations played a part in our design and development but were not as important to us as the kinematic analysis.

6.3.1 Strengths of PMKS+2023 Analysis

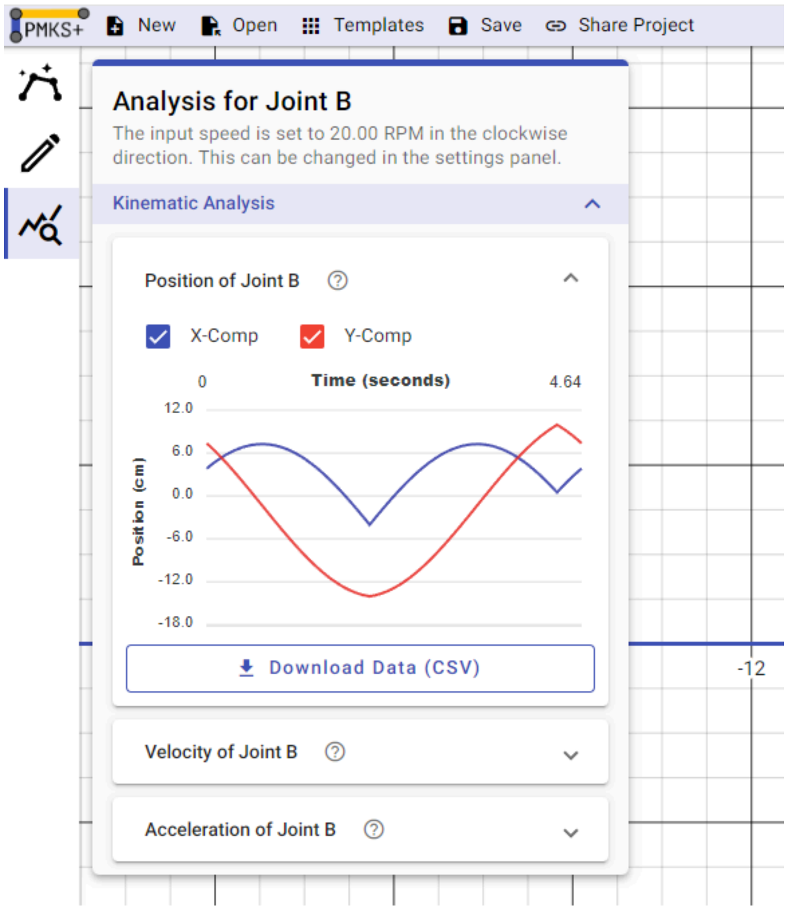
The Analysis panel of PMKS+2023 (shown in figure 6.3.1.1) revolves around a simple user experience loop. The user selects an element on screen, either a link or a joint. The panel will display several options for graphs that the user can view. For joints, these graphs display position, velocity and acceleration. For links, the graphs display the same for the center of mass, in addition to the angular velocity and acceleration of the link.


PMKS+
New
Open
Templates
Save
Share Project
Kinematic Analysis for Link BC
Kinematic Analysis
Angle of Link BC
Angular Velocity of Link BC
Angular Acceleration of Link BC
Center of Mass (COM) of Link BC
Position of Link BC's COM
Velocity of Link BC's COM
Acceleration of Link BC's COM

The input speed is set to 20.00 RPM in the clockwise direction. This can be changed in the settings panel.

Figure 6.3.1.1 PMKS+2023 Analysis Panel

Upon opening a graph, users are presented a simple, intuitive line graph displaying the data (shown in figure 6.3.1.2). The graphs have all the hallmarks of complete analytical tools, with scales, a legend, and color coded data. The mouse hovering over the graph will tell users about all of the data points at that position on the x-axis. All of these components blend together to create a solid user experience when viewing any of the graphs. Changing between joints and links is also simple, requiring only a click on the desired link for the new data to display.



Analysis for Joint B

The input speed is set to 20.00 RPM in the clockwise direction. This can be changed in the settings panel.

Figure 6.3.1.2 PMKS+2023 Analysis Graph

We sought to maintain this ease of use, specifically, as the graphs looked and felt very intuitive to novice users. We also sought to maintain the wealth of information available to the user, albeit with a different presentation. Users appreciated that they had the options laid out for them to view relevant data at just a click. We were concerned that by changing the layout of this panel, we could be inadvertently "hiding" data from users who did not know what to look for.

6.3.2 Limitations of PMKS+2023 Analysis

In contrast to the strengths of the graphs and viewing data, there were certain limitations of PMKS+2023 that presented issues for novice users. The first and most prominent feedback we received from students in our A term 2023 class survey was about the inability to view more than one set of data at once. Even if not required for a specific problem, it can make the user's understanding much more intuitive to see position, velocity, and acceleration, for example. We had a few approaches for how to counter this problem, while maintaining the flow and usability of the panel.

The first proposed solution was a dropdown menu in the Analysis panel, containing each of the potential graphs. The strength of this approach is that it would allow users to switch graphs as seamlessly as the previous version allowed switching joints and links, with a single click. The user could take their pick of which variables they wanted to compare, before switching to another graph quickly and efficiently. Dropdowns are used in many modern tools to show users the list of options they have available to them.

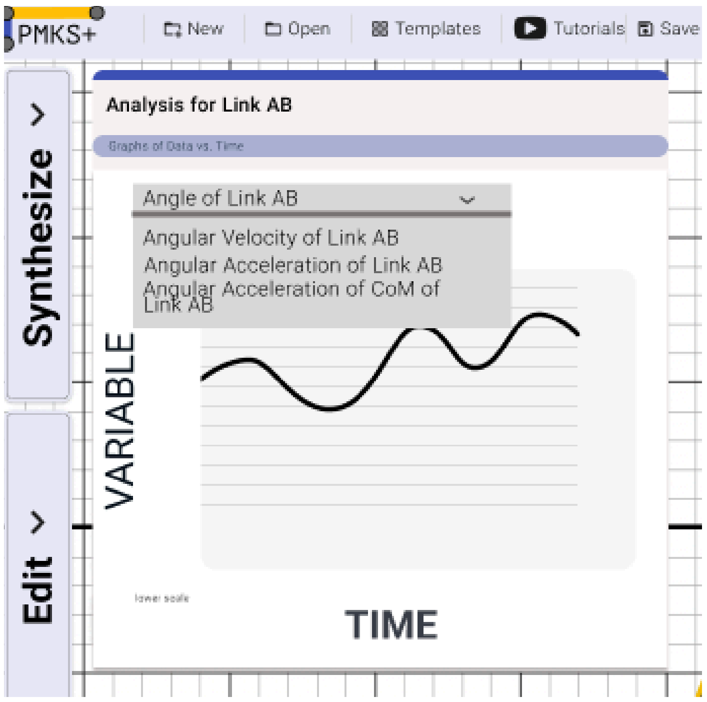
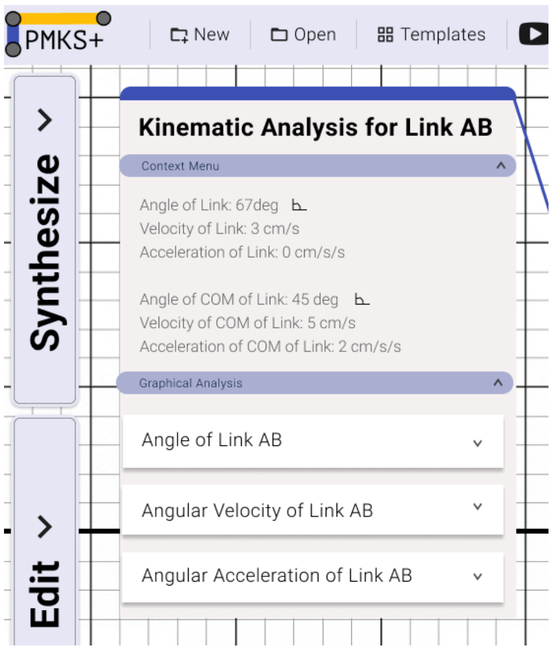


Figure 6.3.2.1 Proposed Data Summary Dropdown

There are several drawbacks to the dropdown menu approach. The most important drawback is that a dropdown menu does not solve the problem of only displaying one set of data at the time. The improvement in user experience is negligible compared to the drawback of not completely solving the initial issue. While it may be more intuitive, we discarded it as an option.

The second proposed solution was a data summary on the panel that would display relevant information to users at a glance. A major strength of this approach is that users would be able to see almost everything they would need for a task at any given time. This gives users a much broader view of the data available to them. By having a complete idea of the capabilities of the application, users can have a clearer picture of the ways that they can manipulate the data provided to them.



Kinematic Analysis for Link AB

Angle of Link: 67deg
Velocity of Link: 3 cm/s
Acceleration of Link: 0 cm/s/s

Angle of COM of Link: 45 deg
Velocity of COM of Link: 5 cm/s
Acceleration of COM of Link: 2 cm/s/s

Figure 6.3.2.2 Proposed Data Summary Panel

A drawback of this proposed solution is available screen space. The data summary takes up a majority of the space we were planning to use for the Analysis panel. This can create problems, as we determined in the survey, students disliked when their graphs were not easily accessible. The data summary was designed with the intent to solve this problem. By displaying data most commonly used in problems, users would be expected to solve with this tool.

The third solution was a popup graph feature (shown in figure 6.3.2.3). By selecting one of the sets of data, a graph popout would be created. The user could move this graph around the screen and open another. This would solve the problem of users not being able to see multiple sets of data at a time. Additionally, users could select where to place their graphs according to personal preference. This solution would be the most intuitive and fluid of the three.

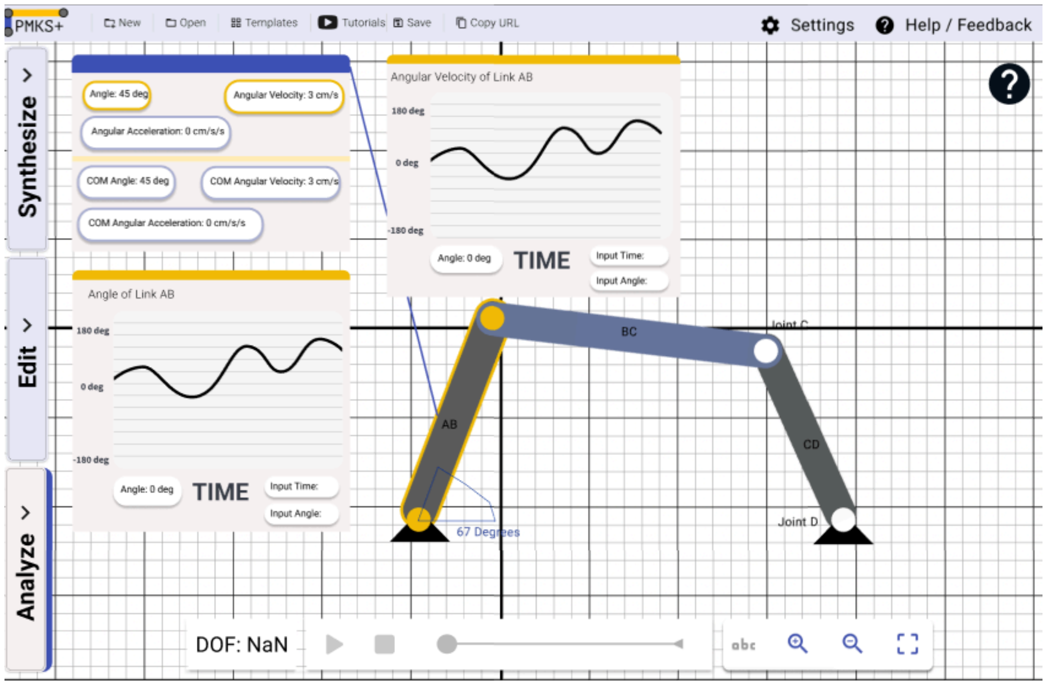


Figure 6.3.2.3 Proposed Analysis Context Menu With Multiple Graphs Selected

The biggest drawback of this solution was screen space. By opening multiple graphs from a panel on the left and moving them around the screen, the application could become cluttered. We did not want to overwhelm newer users with information, so popout graphs were discarded.

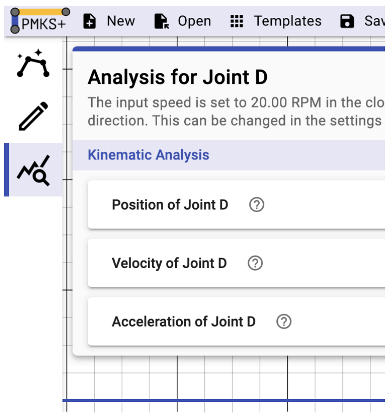
6.4 Tab Design

The tabs are the first thing a user needs to look at when opening the application. It is the side navigation that allows users to switch between the three modes of the system: Synthesis, Edit, Analysis (explained above in section 5.4.2).

The first tab design (shown in design a of figure 6.4.1) from PMKS+2023 had several flaws in the design. The primary users of PMKS+ are novice users with no experience. The current issues described in section 5.4.2 as well as the feedback from our initial survey, described in Appendix B, showed that experienced users liked the tabs with icons while inexperienced users liked the tabs with labels. Another issue was the tab color inconsistencies. Due to differences in preferences with different experience levels, one of the options was to combine the two (shown in design b of figure 6.4.1). This option has the three tabs which include an icon and the tab label below the icon. When you are in the tab, the tab changes from light blue to white which is the same background as the contents of the tab which better displays to the user which tab they are actively in.

The other two options were based on options created by Kohmei Kadoya in the 2023 thesis. These options consist of text rotated 90 degrees with wide tabs. The first of these two (shown in design (a) of figure 6.4.2) has similar issues to the previous design. There are unconnected tabs due to the panels for each tab being located in the same position at the top of the window. Specifically, the Analysis tab is at the bottom of the window, while the panel is placed at the top. The next option (shown in design (b) of figure 6.4.2) would eliminate this issue by connecting the panel to the tab and changing the placement of the panel based on the tab. The Synthesis panel would be positioned at the top, in-line with the synthesis tab, the Edit panel would be positioned in the middle, in-line with the Edit tab and the Analysis panel would be positioned in the lower third of the window, in-line with the analysis tab. However, in order to display all the necessary information in the Analysis panel without the user needing to scroll, the panel would shift up as more information is shown or added. A similar issue would occur if the Edit panel were to extend beyond the bottom of the window, it would shift up past the top of the tab in order to display all of the information without the user needing to scroll. This would cause inconsistencies with the way that information is positioned and displayed on the panels.

a)



The input speed is set to 20.00 RPM in the clo
direction. This can be changed in the settings

b)

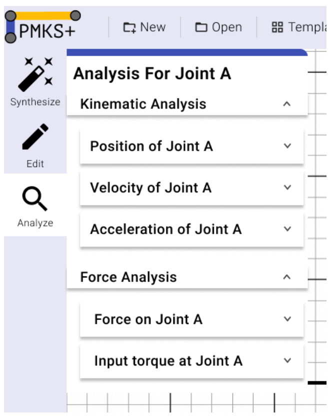
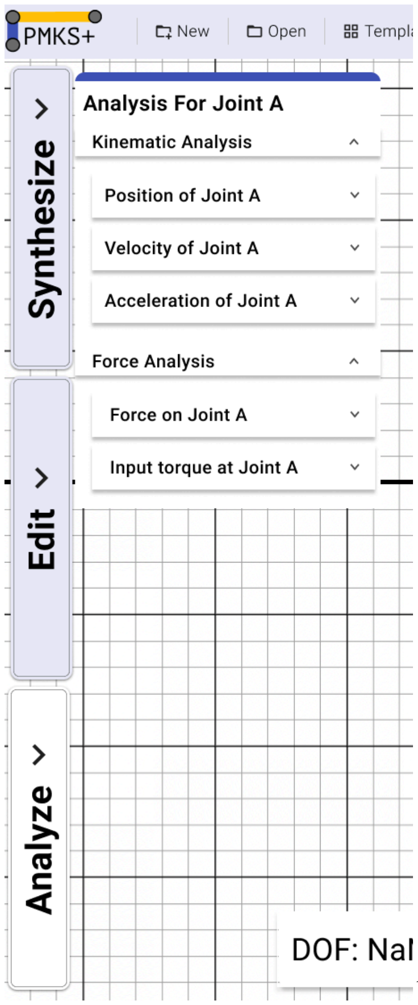
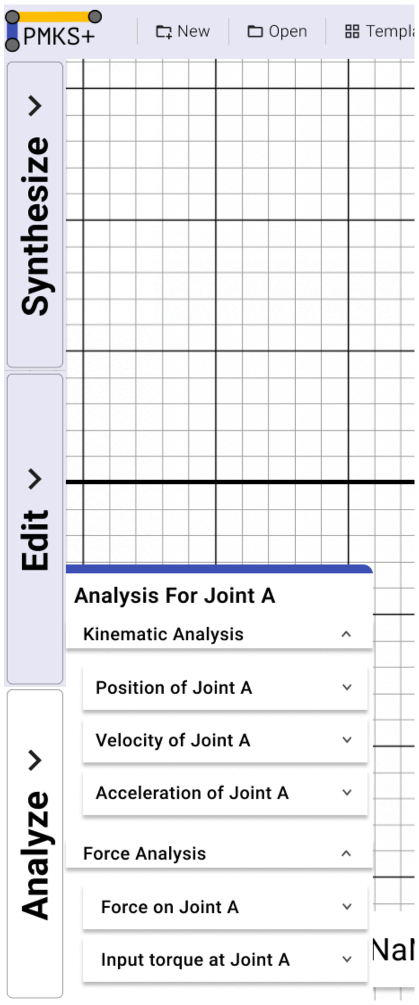


Figure 6.4.1 a) Proposed Tab Designs: with Icons

Figure 6.4.1 b) Proposed Tab Designs: with Icons and Text



a)


b)

Figure 6.4.2 a) Proposed Tab Designs with Text Only: Panel Disconnected from Tab

Figure 6.4.2 b) Proposed Tab Designs with Text Only: Panel Connected to Tab

This chapter discussed how we took the feedback discussed in section 5.4 to create our mockups that helped us to implement the re-designed UI. The next chapter discusses the design choices made to implement our Model-View-Controller architecture on the backend.

7. Refactor Development and Design

This chapter discusses the design phase of the refactored codebase that took place after our decision to recreate PMKS+ using the Model-View-Controller (MVC) architecture. We divide this section by the three pillars within the architecture, and approach our discussion of each from a top-down perspective.

7.1 Model Design

The composition of our model depended heavily on identifying the key components needed to represent a mechanism. These were joints, rigid bodies, and forces during our project. By limiting our mechanisms to these three subcomponents, we ensured the creation of a simple model with limited complexity.

Joints were the foundation on which the rest of our model would rely. In our pursuit to simplify their representation, we decided to represent them with a single class in contrast to the multiple classes of the previous system. We instead opted to use an 'enum' to differentiate the types of joints. This left us with a joint class containing a coordinate representing position, several boolean fields representing inputs, grounded and welded, along with an input speed and angle that would only apply given other fields were true.

Links and compound links were designed next. Links contained two lists of joints and forces within them, a mass, and center of mass position. Compound links replaced the two lists with a single list of links that made up the compound link. These compound links were a combination of multiple links into one, but with the ability to come apart back into separate links if needed. Two links would become a compound link through a shared link between them that became welded.

The design of the mechanism class itself was quite straight forward. Every method represented a change in state that could be made in the current system. Thus we removed getters and setters and replaced them with actions taken by the user to alter the mechanism. Alterations to the subcomponents themselves also were done through the parent mechanism class to account for scenarios where a change in one state would require changes in other components. This is most common in welding a joint. When a joint is welded, a new compound link must be created, and the mechanism class performs that task.

7.2 View Design

When designing the view for the refactor, we divided the application into four main components. These were the central grid that displayed the mechanism, the side navigation tabs/panels, the toolbar, and the animation bar. For the side navigation and the toolbar, all of the child components were created as panels. These panels were then populated with information and functionality for the user. The animation bar was designed to replicate the functionality and style of the previous version, but our team lacked the time to fully complete its implementation.

The central grid of the application was the most important aspect of our redesign for the view. We divided the grid into a component responsible for drawing the grid lines and a secondary component that handles drawing the mechanism to the screen. For the mechanism, we employed child components that directly corresponded to the classes within our model. These included links, compound links, joints, and forces. By dividing the mechanism into the individual parts of the mechanism, we drastically reduced the code needed to draw the mechanisms.

7.3 Controller Design

The controller design was divided into four aspects. We created a parent controller that handled all interactions with the grid. This class would delegate interactions to other controllers responsible for the components. These other controllers contained the three remaining responsibilities for controllers. These were: updating the position of the component while dragging; determining the available context menu options; and entering modes for adding links. By having a controller for each part of a mechanism, we created a one to one mapping for every subcomponent within our model, to every component drawn to the screen, and to every controller responsible for updating the mechanism. This heavily simplified the process of identifying code responsibility, and reduced class sizes by over 50%. With this architecture in place, adding additional subcomponents (new rigid bodies) becomes trivial, and the time to develop additional functionality is reduced by the simplicity of the refactored system.

7.4 Services Design

The final aspect of design for our project was recognizing the need for services classes. These classes contained functionality that was needed by multiple other parts of the program. These services included the kinematics solvers, animation handlers, and various other functions needed throughout the program. By separating these functions into multiple classes, we reduced the overhead created by a single large utilities class.

This chapter discussed how the model, view, and controller elements were implemented to re-design and give structure to the backend of the codebase. The next chapter discusses the design decisions and the final implementation of our UI.

8. Frontend Implementation

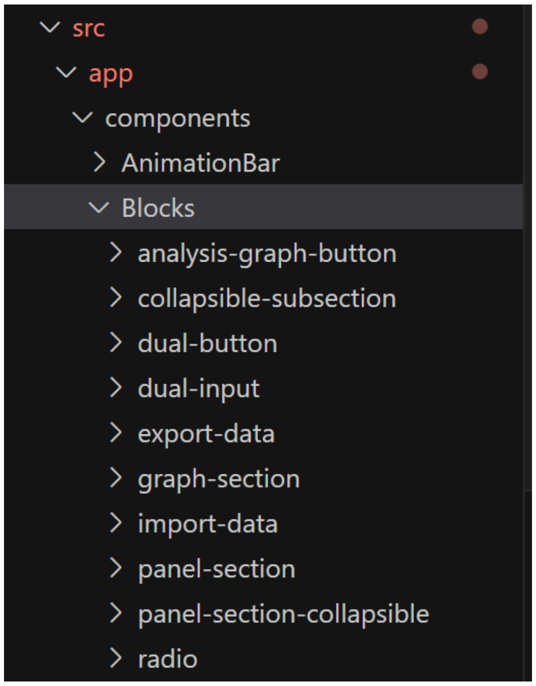
This chapter contains details of our implementation of the front end. There are two areas of focus of our frontend implementation. The first focus is on interesting or novel architectural choices made by the 2024 team. The secondary focus is placed on the unique architectural design choices we made that contrast with the design choices of PMKS+2023.

8.1 Design Decisions

In this section, we discuss the methods used to make PMKS+2024 have a cohesive look and common structure to allow for new developers to more easily add additional features.

8.1.1 HTML Blocks

We took inspiration from PMKS+2023 to build a modular set of front end components that we call HTML blocks. Instead of dumping a load of code into each file, we instead separated out specific instances of code that could be used in multiple places in the front end (shown in figure 8.1.1.1). These blocks often contained data input or output components that needed dynamic data. By creating modular blocks, we could substitute the “hard coding” done in PMKS+2023 with HTML elements of our own that can be modified.



```
src
  app
    components
      AnimationBar
      Blocks
        analysis-graph-button
        collapsible-subsection
        dual-button
        dual-input
        export-data
        graph-section
        import-data
        panel-section
        panel-section-collapsible
        radio
```

Figure 8.1.1.1 PMKS+2024 Blocks

Each block is a different element of the user interface. A key benefit to blocks is that once we set the style for the block, we never have to change it again. This creates a more uniform and intuitive interface for users. For example, our collapsible sub-sections block is used in each of the Edit, Analysis, and Synthesis panels, and allows each panel to expand information the same way. We created button blocks for two and three buttons positioned next to each other. The dual input block is used to create a section with two inputs; this is used in the Edit panels. When using the blocks, we are able to use different functions to pass in values based on the needs of the application.

8.1.2 Using Switch Cases

Something we as developers found confusing with PMKS+2023 was its Edit panel. Despite using extremely similar code and layouts for Joint and Link Edit panels, the file was hard coded to contain all of the data itself. This left us sifting through hundreds of lines of code to find individual HTML elements. To combat this in our refactor, we decided to use a system of cascading switch cases in our frontend view. To start, when selecting a panel on the side navigation, our program switches from a default view to the view of one of the panels. When selecting a component of a mechanism while on one of those panels, another switch case is triggered, displaying the relevant information and HTML components. This dynamic implementation of the different panels gives future developers a more natural sense of how our program flows than the previous iteration.

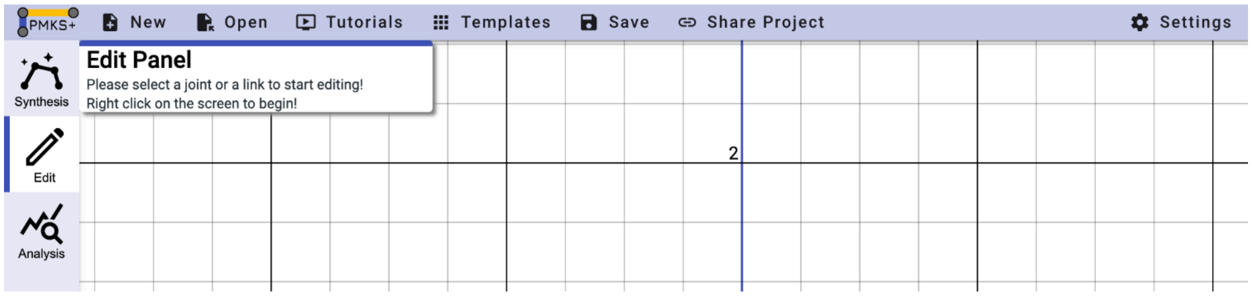
8.2 Grid

The grid is the first thing users see when opening PMKS+2024. It is an empty canvas on which users can create their mechanisms. It is very similar to the grid space of PMKS+2023. The lines on our grid correspond to units of measurement, which is in centimeters by default. The blue grid lines correspond to when the x and y axis are 0, converging on the origin. There are major lines at certain multiples of 2, depending on the level of zoom.

We implemented several distinct improvements over the grid space in the 2023 version. The most important of these is a click-and-drag function for components on the grid. By clicking on a link or a joint, the user can move either that link or that joint, and not the entire grid. This allows users greater freedom in manipulating their mechanisms, and is more intuitive to novice users. Additionally, we implemented the ability to select multiple components on the screen at once. By holding the Shift key and clicking another joint or link, the user can curate the selected objects at any given time. This selection can then be dragged around the screen, similarly to a single link or joint. Using Shift + clicking to select multiple components is intuitive to the user, as it is standard practice across several similar types of programs.

8.3 Top Bar

The top bar in PMKS+2024 (shown in figure 8.3.1) mirrors the implementation in PMKS+2023. The top bar is only partially implemented. Currently the New button has the same implementation and opens a new window of PMKS+. The Tutorials, Templates, and Settings tabs have not been fully implemented; they are partially functional but do not yet look the way they should.



Edit Panel

Please select a joint or a link to start editing!
Right click on the screen to begin!

Figure 8.3.1 PMKS+2024 Top Bar

The top bar tab with the most functionality is the templates (shown in figure 8.3.2). PMKS+2023 implemented templates by using the encoded links to open a new window with the selected mechanism. Due to PMKS+2024 not having all of the functionality of PMKS+2023, there is no way to implement that in this way. Instead, we used functions in the mechanism class to be able to create a new mechanism. This is called using the State Service that displays the mechanism to the user in the same window. The reasoning for this was because of the new functionality of multiple mechanisms. The current implementation is functional, but does not look the way it is intended. However, the functionality of using multiple mechanisms in one window does not work yet due to the way the functions are determining which joints are grounded, sliders, or inputs.

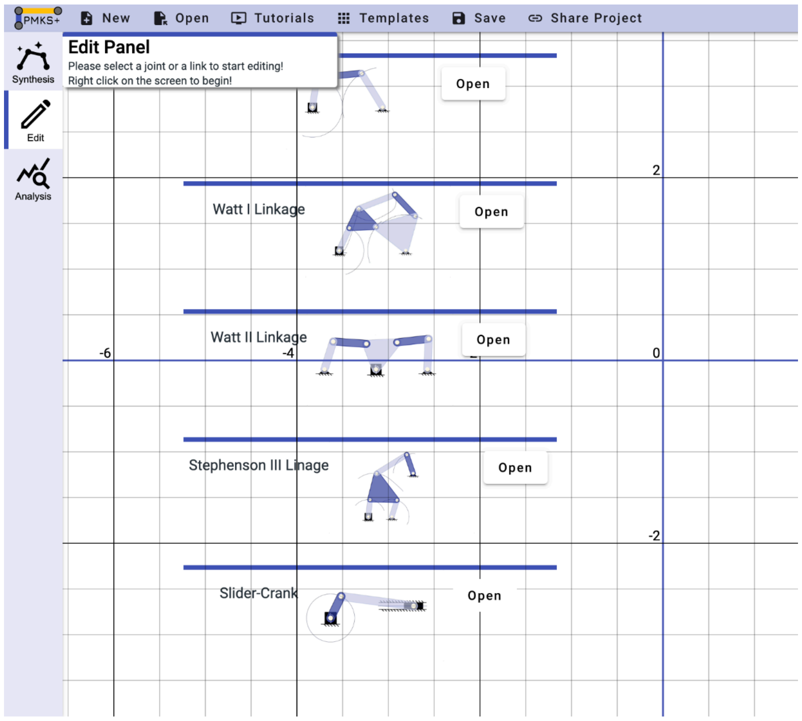
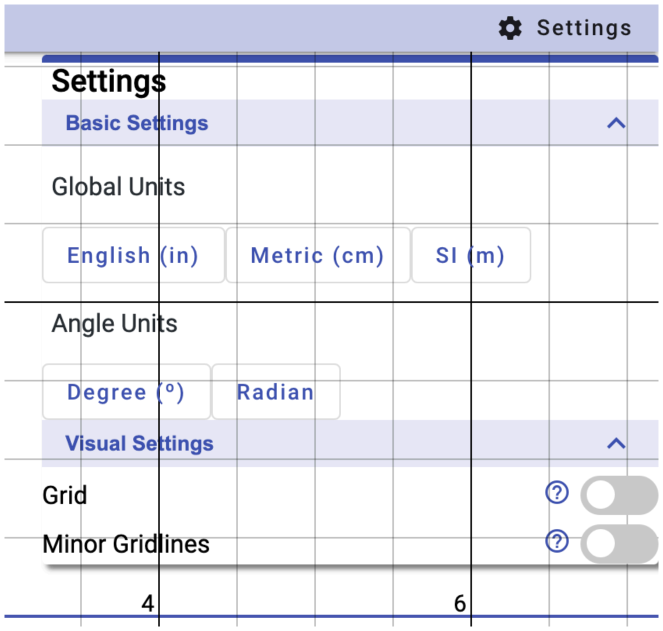


Figure 8.3.2 PMKS+2024 Templates

We decided to combine the Settings and Help/Feedback tabs that existed in PMKS+2023 to be just Settings. The Settings panel is currently implemented but not functional as shown in figure 8.3.3. Like with the Templates panel, the Settings panel is not styled properly and multiple bugs exist. The functionality to change units (e.g., metric) on the backend does not exist, so therefore the content is only a placeholder for later implementation.


Settings

Settings

Basic Settings

Global Units

English (in) | Metric (cm) | SI (m)

Angle Units

Degree (°) | Radian

Visual Settings

Grid

Minor Gridlines

Figure 8.3.3 PMKS+2024 Settings

Lastly, we added the Tutorials tab, which did not exist in PMKS+2023. The intention was that this tab would include video tutorials of common questions and issues that users may have including: welding joints, making a mechanism, animating a mechanism, analyzing mechanism data, deleting links or joints, using synthesis, etc. This tab currently only includes one video placeholder and is not functional.

8.4 Synthesis Panel

The Three Position Synthesis panel was the only type of synthesis that existed in PMKS+2023. In PMKS+2024, we were able to re-implement the panel with the changes that were discussed in sections 5.4.5 and 6.1 (shown in figure 8.4.1). We also added a placeholder to create future types of synthesis in addition to Three Position Synthesis. While the panel is implemented, it is not functional. The backend does not include the functionality that would allow mechanisms to be synthesized. The current panel uses placeholders to display what the functionality of the panel would eventually look like.

We used a series of Dual-Input-Blocks to display the position data to the user and allow them to change their values. We also decided to include a combination of each method proposed in section 6.1 to delete positions. We placed trash icons next to each position to allow users to delete specific positions as well as a button to remove all positions. We also decided to include a "Generate Four-Bar" button so a mechanism isn't created before a user is ready. After a mechanism is created, the "Generate Six-Bar" button is displayed to give users the option to convert their mechanism to a six-bar mechanism. The panel still needs basic styling and spacing before it will be ready for users to synthesize mechanisms.

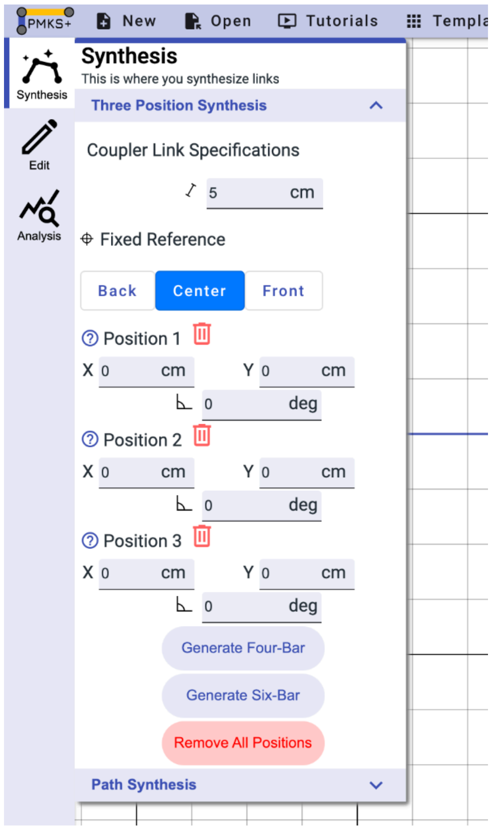


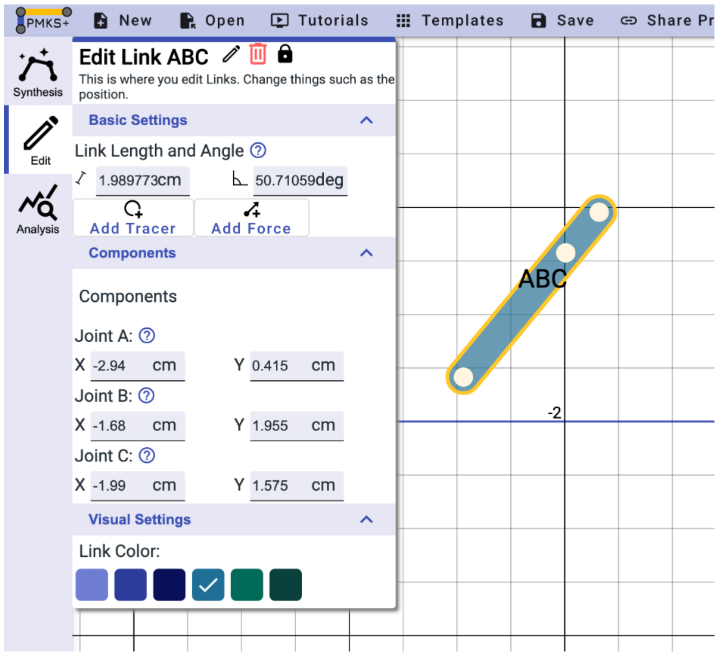
Figure 8.4.1 PMKS+2024 Synthesis Panel

8.5 Edit Panel

Each Edit panel was designed using the same HTML blocks mentioned above in section 8.1.1. Each panel as it currently exists still needs minor styling changes but maintains the majority of the functionality, if not more, of the functionality that exists in PMKS+2023.

8.5.1 Link Edit Panel

We used the designs described in section 6.2.1 to implement the Link Edit panel. Almost all of the functionality of the Link Edit panel that existed in PMKS+2023, exists in PMKS+2024 (shown in figure 8.5.1.1). The only thing that is missing is the "Add Force" button whose functionality is not implemented in the backend. It is currently a placeholder button that does not do anything but will be easy to connect to once the functionality exists in the system. In addition to including everything that existed previously, we also added the ability to view the coordinates of each component that makes up the link, including joints and tracer points.



Edit Link ABC

This is where you edit Links. Change things such as the position.

Basic Settings

Link Length and Angle

1.989773cm 50.71059deg

Components

Components

Joint A:

X -2.94 cm Y 0.415 cm

Joint B:

X -1.68 cm Y 1.955 cm

Joint C:

X -1.99 cm Y 1.575 cm

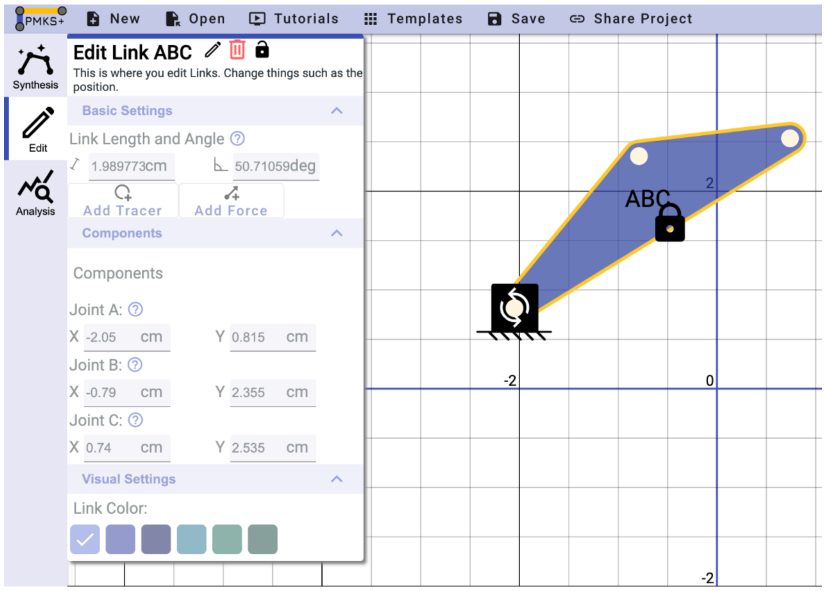
Visual Settings

Link Color:

Figure 8.5.1.1 PMKS+2024 Link Edit Panel

Another feature that was added was the ability to lock links. This was described in section 6.2.4. When a link is locked a large lock icon is displayed at the center of mass of that link (see figure 8.5.1.2). The panel becomes grayed out and the only pieces that are able to be used or modified are the edit title button, delete link button, and unlock button. In order to modify any attributes of the link, the user would need to unlock the link first.

PMKS+2024's implementation of the Compound Link Edit panel (shown in figure 8.5.1.3) is based on the Link Edit panel previously discussed. We organized some elements of the UI better when compared to the PMKS+2023 implementation and increased the panel's functional capabilities. The user is now able to view the different subcomponents of the compound link. They can drag and manipulate the link, and can edit its values to be whatever they require. It has primarily the same functionality as the regular Link Edit panel.



Edit Link ABC

This is where you edit Links. Change things such as the position.

Basic Settings

Link Length and Angle

1.989773cm 50.71059deg

Add Tracer Add Force

Components

Components

Joint A: X -2.05 cm Y 0.815 cm

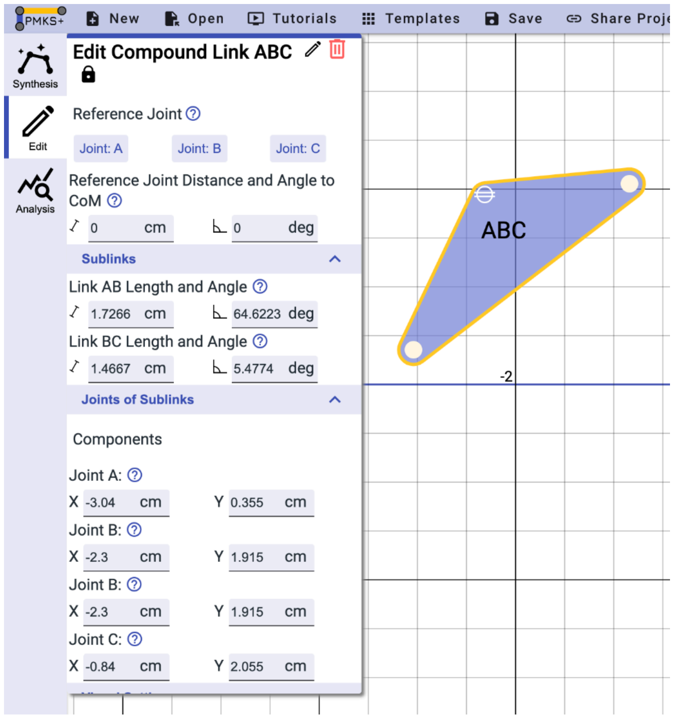
Joint B: X -0.79 cm Y 2.355 cm

Joint C: X 0.74 cm Y 2.535 cm

Visual Settings

Link Color:

Figure 8.5.1.2 PMKS+2024 Locked Link Edit Panel



Edit Compound Link ABC

Reference Joint

Joint: A Joint: B Joint: C

Reference Joint Distance and Angle to CoM

0 cm 0 deg

Sublinks

Link AB Length and Angle

1.7266 cm 64.6223 deg

Link BC Length and Angle

1.4667 cm 5.4774 deg

Joints of Sublinks

Components

Joint A: X -3.04 cm Y 0.355 cm

Joint B: X -2.3 cm Y 1.915 cm

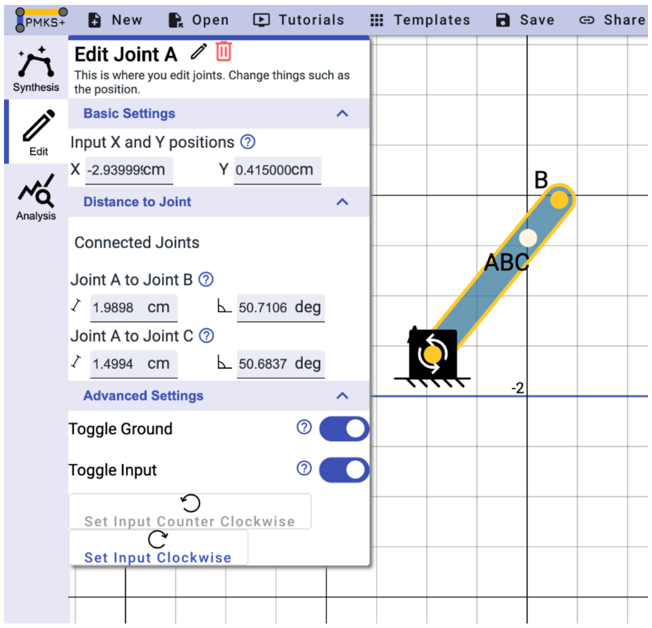
Joint B: X -2.3 cm Y 1.915 cm

Joint C: X -0.84 cm Y 2.055 cm

Figure 8.5.1.3 PMKS+2024 Compound Link Edit Panel

8.5.2 Joint Edit Panel

Similar to the Link Edit panel, the Joint Edit panel also contains all the functionality from PMKS+2023 (shown in figure 8.5.2.1). We changed each of the buttons to use toggles to more visibly show their state to the user. The input direction functionality is not implemented in the backend so the buttons are just placeholders for future implementation. We also decided to remove the “Visual Settings” drop down because of its lack of purpose and move the toggles to the “Advanced Settings” dropdown as they are not as frequently used.



Edit Joint A

This is where you edit joints. Change things such as the position.

Basic Settings

Input X and Y positions

X -2.93999cm Y 0.415000cm

Distance to Joint

Connected Joints

Joint A to Joint B

1.9898 cm 50.7106 deg

Joint A to Joint C

1.4994 cm 50.6837 deg

Advanced Settings

Toggle Ground

Toggle Input

Set Input Counter Clockwise

Set Input Clockwise

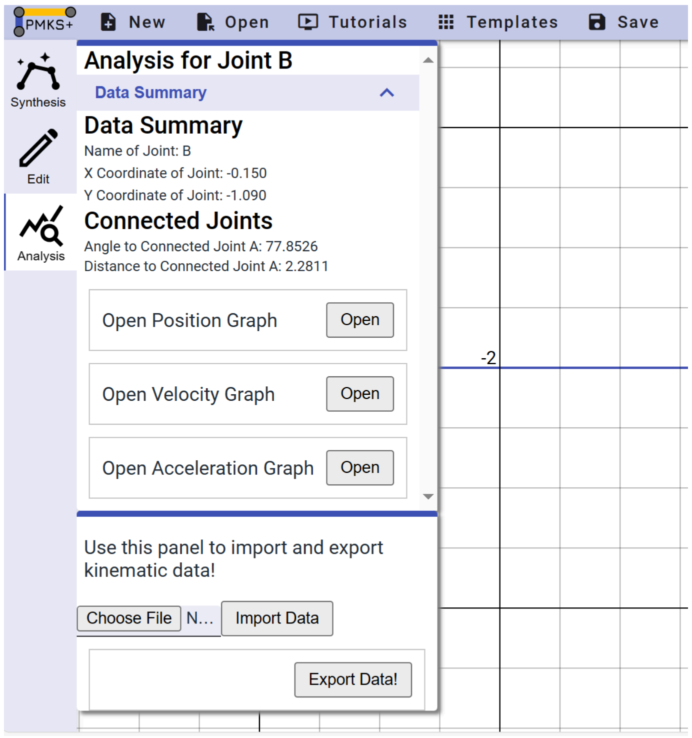
Figure 8.5.2.1 PMKS+2024 Joint Edit Panel

8.6 Analysis Panel

Our Analysis panel is split into three components. There are separate views for joint, link, and compound link analysis. We addressed the concerns presented in our UI analysis section as well as we could and implemented the mockups we discussed in that section.

8.6.1 Joint Analysis Panel

The first Analysis panel we developed was the one for joints. We addressed the major concerns about PMKS+2023’s Analysis panel described in our UI design phase description. While roughly similar, the panel deviates in some critical ways. The most important is the data summary, showcased in figure 8.6.1.1. This summary evolved from the figma mockups showcased in previous sections. We consulted with our advisors to find the information that would be most useful to a mechanical engineering student working on assignments. The summary is divided into general joint information, and information relating to connected joints such as relative angle or distance.



Analysis for Joint B

Data Summary

Data Summary

Name of Joint: B

X Coordinate of Joint: -0.150

Y Coordinate of Joint: -1.090

Connected Joints

Angle to Connected Joint A: 77.8526

Distance to Connected Joint A: 2.2811

Open Position Graph Open

Open Velocity Graph Open

Open Acceleration Graph Open

Use this panel to import and export kinematic data!

Choose File N... Import Data

Export Data!

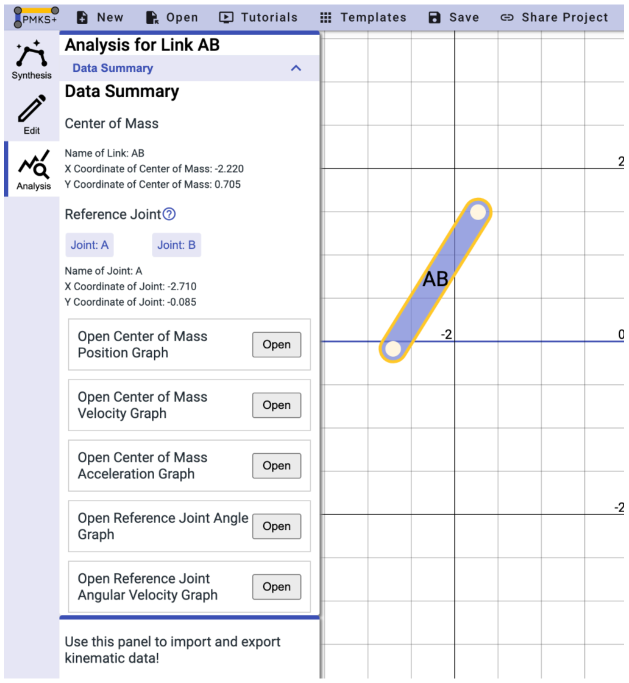
Figure 8.6.1.1 PMKS+2024 Joint Analysis Panel

To actually analyze the data, users select the type of data graph they wish to open. For joints, the user can open graphs that display data for position, velocity, and acceleration. These graphs are elaborated on in more detail in section 8.6.4.

We also have an export data component in our Analysis panel. This exports data automatically from the application into a CSV spreadsheet that the user can then manipulate in any way they wish. As one of the most requested features from users in our A term survey, we needed to give users the flexibility of having every piece of kinematic data available. Users can also select a dataset to import kinematic data from. This successfully logs each position frame of a mechanism, however, it does not display on the grid.

8.6.2 Link Analysis Panel

Seen in figure 8.6.2.1, the Link Analysis panel is similar to the Joint Analysis panel. The first component of the panel is a data summary, and the second is a selection of graphs. The data summary contains data about the link's center of mass (CoM) and a section dedicated to a reference joint the user can specify. The reference joint can be any of the joints that make up the selected link. These pieces of information update as the mechanism changes, and display appropriately. Changing the selected reference joint also updates the data displayed on screen to the appropriate values.



Analysis for Link AB

Data Summary

Data Summary

Center of Mass

Name of Link: AB
X Coordinate of Center of Mass: -2.220
Y Coordinate of Center of Mass: 0.705

Reference Joint

Joint: A Joint: B

Name of Joint: A
X Coordinate of Joint: -2.710
Y Coordinate of Joint: -0.085

Open Center of Mass Position Graph

Open Center of Mass Velocity Graph

Open Center of Mass Acceleration Graph

Open Reference Joint Angle Graph

Open Reference Joint Angular Velocity Graph

Use this panel to import and export kinematic data!

Figure 8.6.2.1 PMKS+2024 Link Analysis Panel

There are six new graphs available to the user in the Link Analysis panel. The first three are similar to the three from the Joint Analysis panel. They display the position, velocity, and acceleration data for the link's CoM (Center of Mass). This data is useful to the user for a variety of reasons and was available in the PMKS+2023 application. Additionally, the panel displays three graphs for angular kinematics; these are angular velocity, angular acceleration, and angle. The angular kinematic graphs refer to the specific reference joint previously selected by the user.

8.6.4 Graphs

We chose to use chart.js to display our graphs (Chart.js, n.d). This library gave us a lot of flexibility in the styling and modularity of the graph component of our Analysis panel. By formatting a dataset when calculating it, we can feed almost anything we require into the analysis graph HTML block we created. We utilize this to great effect in each Analysis panel, as we do not need to hard code any extra graph types for each panel. By using helper functions in the analysis solver service and consistent styling in our graph block, we created an intuitive graphical interface for users to analyze their data.

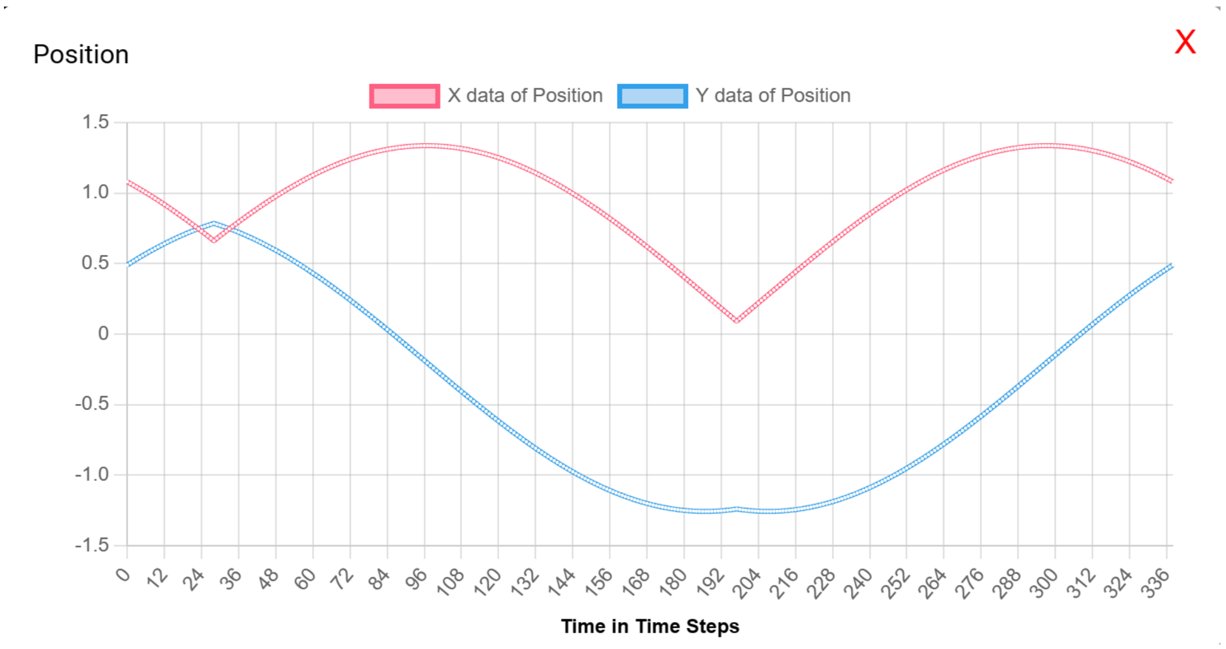


Figure 8.6.4.1 PMKS+2024 Graph

Each graph displays on a two dimensional line chart, with the x-axis being the timestep of the mechanism and the y axis being the data fed to the chart. Each graph for position, velocity, and acceleration is fully functional. As seen in figure 8.6.4.1, the first dataset fed to the chart (in this case the X component of a joint's position) is displayed in a pink-red color. The second dataset fed to the chart (the Y component of the joint's position) is displayed as a blue color. When hovering over one of the lines, the data is displayed in a text box next to the data point shown in figure 8.6.4.2.

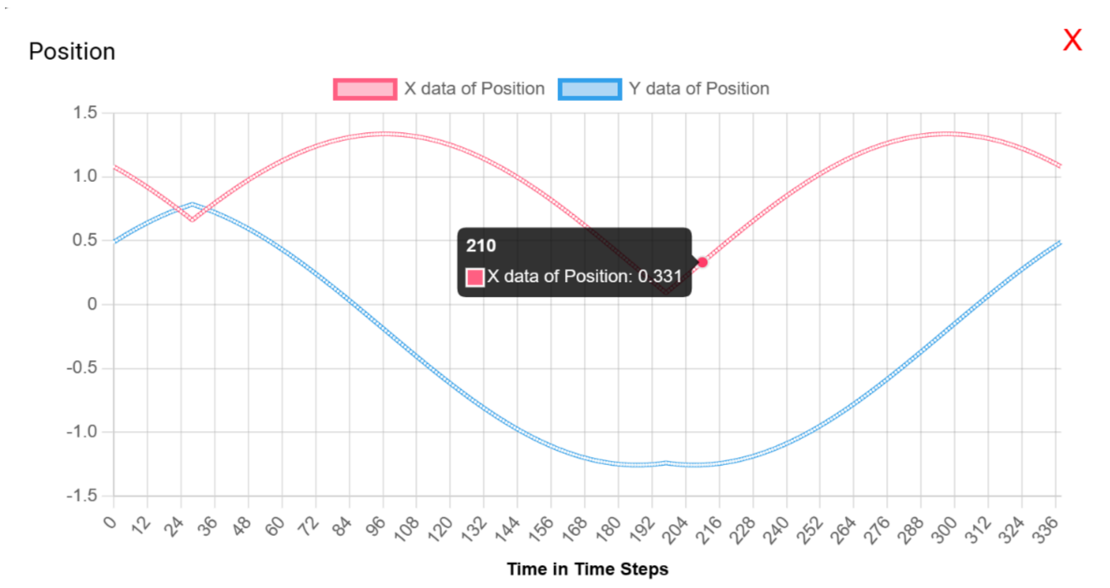


Figure 8.6.4.2 PMKS+2024 Graph Being Hovered Over

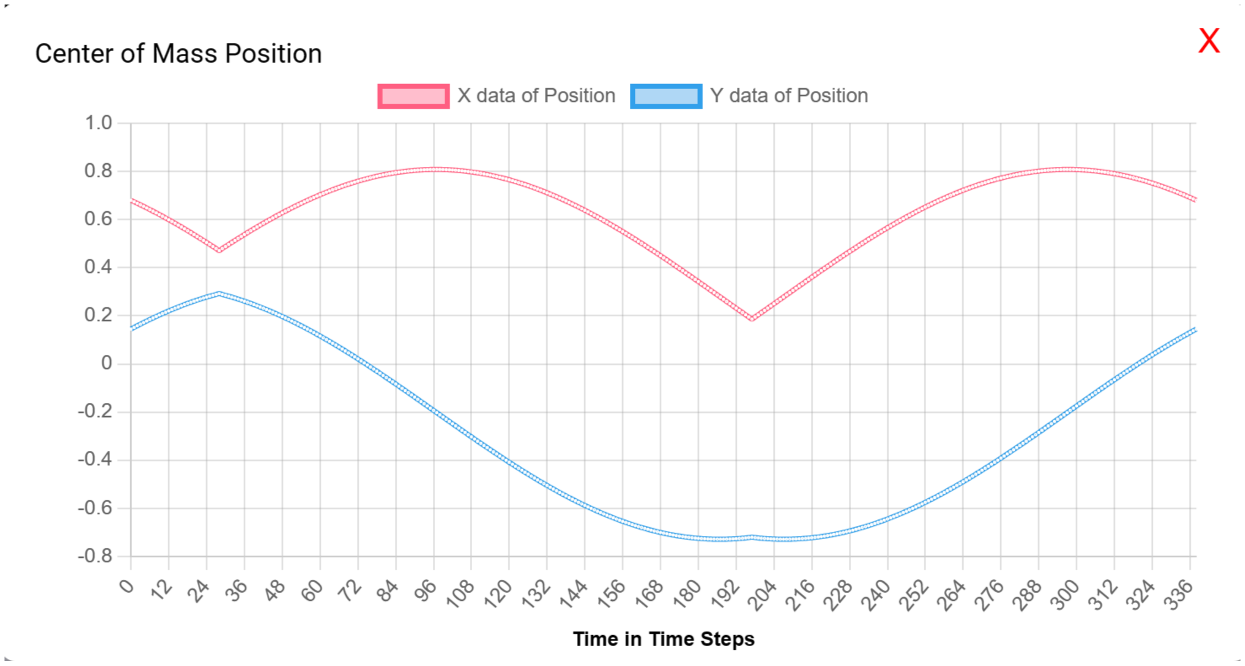


Figure 8.6.4.3 PMKS+2024 Position Graph

The angular kinematic graphs are not fully functional. Figure 8.6.4.4 shows a graph of the angle of a reference joint in radians against time. This is generally consistent with what we expect such a graph to look like. Figure 8.6.4.5 shows the same reference joint's angular velocity data versus time. As we can see, it is an incomprehensible set of data, and certainly not a correct display. The scale of the x-axis is separated by a margin of 10^-15, which is an unrealistic level of detail and indicative of an underlying calculation error. This will be addressed in future work.

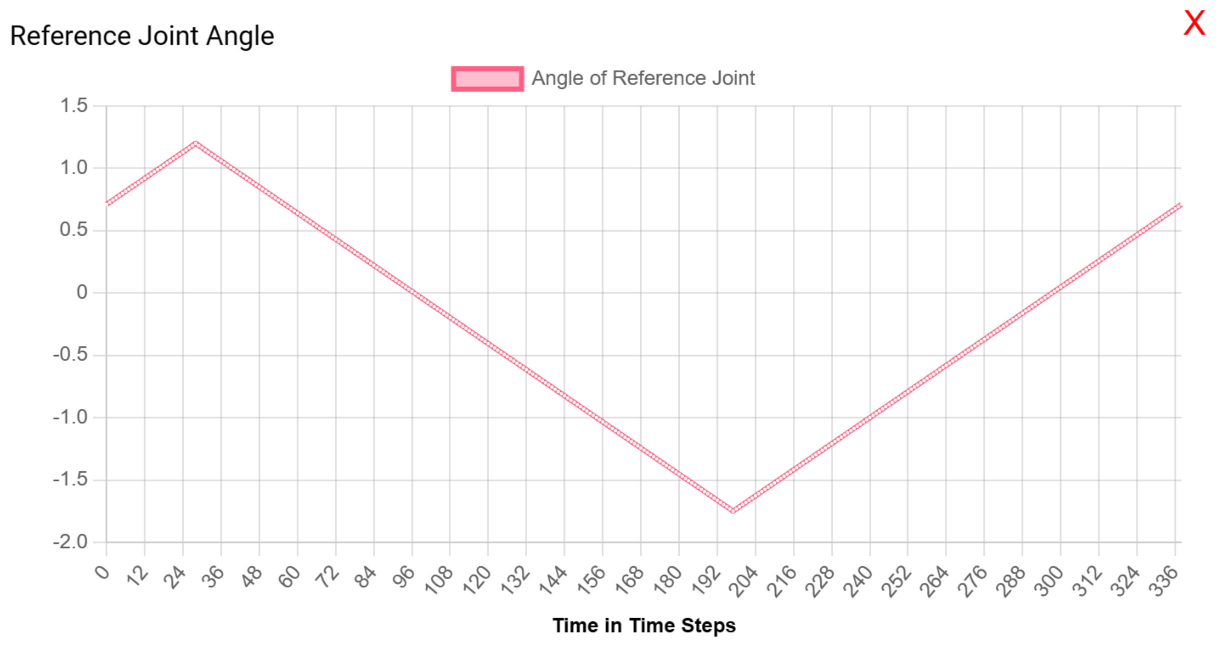


Figure 8.6.4.4 PMKS+2024 Angle vs Time Graph

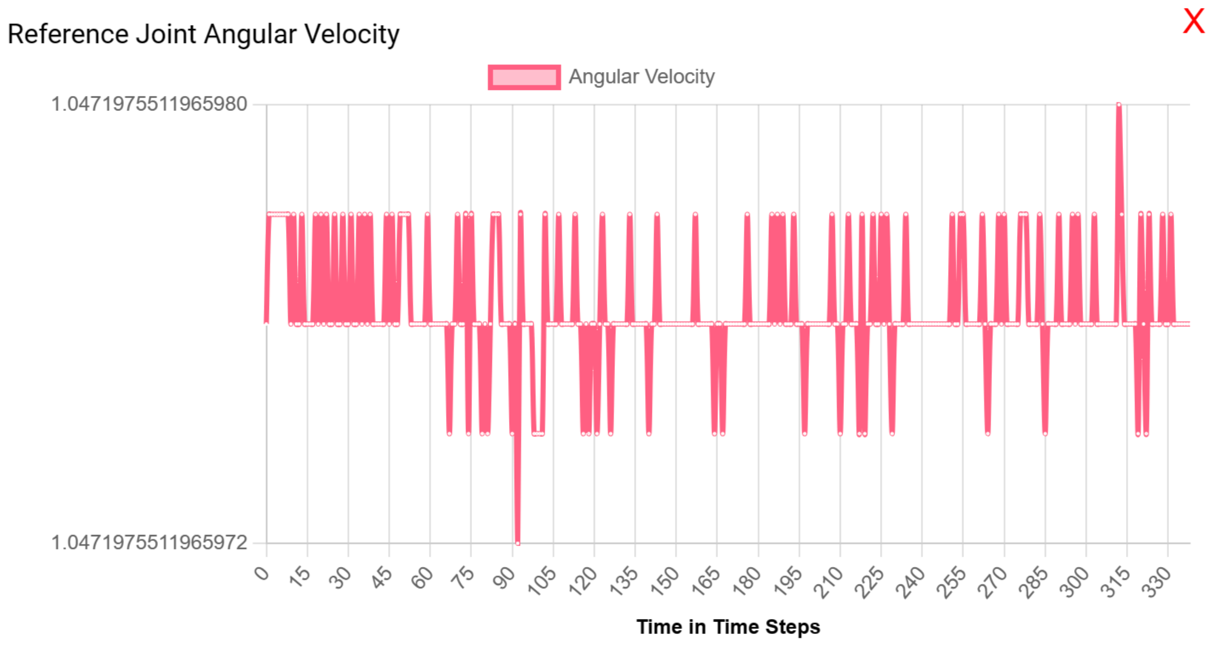


Figure 8.6.4.5 PMKS+2024 Angular Velocity Broken Graph

In other cases, the angular velocity and acceleration look correct. Figure 8.6.4.6 shows a graph of angular velocity for a different linkage in the same mechanism. The scales look correct for a velocity graph of a linkage in this configuration and the units seem to be accurate. The spikes in velocity correspond to when the mechanism switches directions, which is something a future team can account for in their graphing.

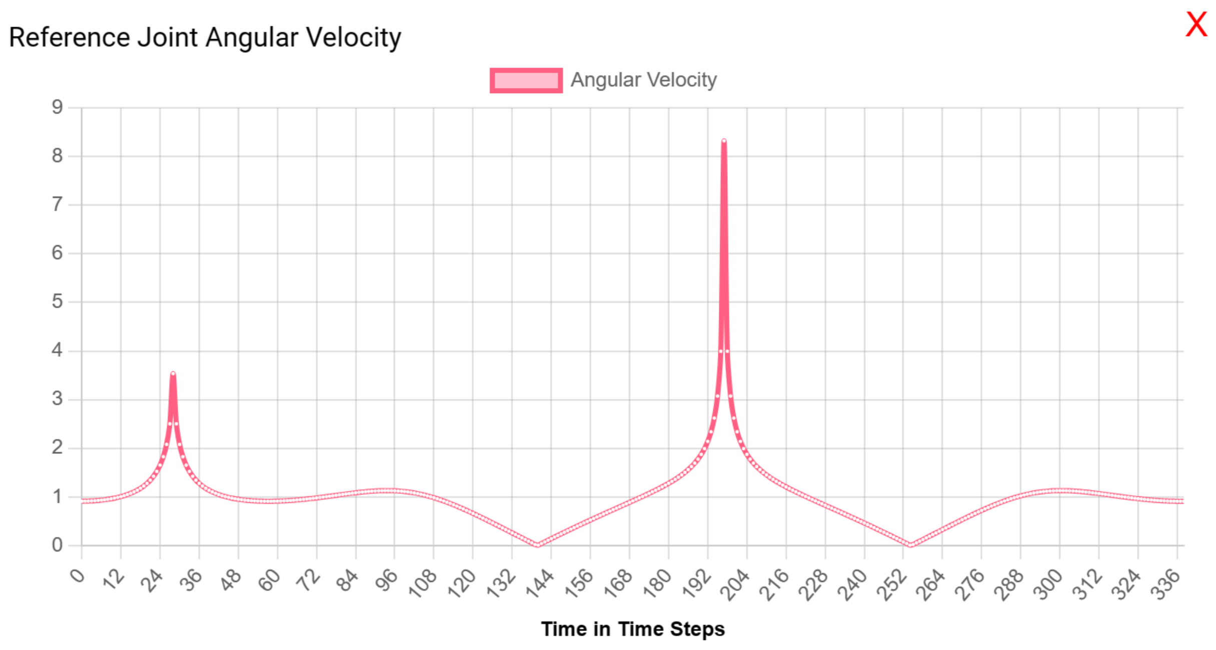


Figure 8.6.4.6 PMKS+2024 Angular Velocity Correct Graph

Overall, when fed appropriate data, our graphs are consistent in presentation with the graphs offered in PMKS+2023. The strength of our implementation comes from the modularity—any correctly calculated dataset can be fed into the same graph object and output a consistent view.

This chapter discussed the current state of PMKS+2024 as well as the design choices taken. The next chapter discusses how usable PMKS+2024 is compared to PMKS+2023 as well as the achievements made with the reduction of code complexity.

9. Evaluation

It was important for us to determine what our system improved over the previous iterations of PMKS+. Most importantly, we wanted to know what novice users would think about the usability of our program versus PMKS+2023 specifically. To assess this, we conducted user surveys of select members of upper level classes to compare the usability of the old version versus our newer version. Additionally, we wanted to know the developer usability; the ability for a fresh developer to take over this codebase. For that, we created UML class diagrams to visually showcase the previously mentioned architectural changes we made. We also surveyed Computer Science students to assess whether they would feel more comfortable working in the PMKS+2023 versus the new PMKS+2024.

9.1 System Usability

After giving students the two surveys, one using PMKS+2023 and one using PMKS+2024, we evaluated the usability of PMKS+2024 by comparing the System Usability Scores (Thomas, 2019). As shown in figure 9.1.1, the scores ranged from 50 to 100 with an average score of 82.7 for PMKS+2023 and 70.2 for PMKS+2024. According to the System Usability Scale, scores of 68 or higher are considered to be a usable system so our average score of 70.2 shows that the system is usable but still needs work.

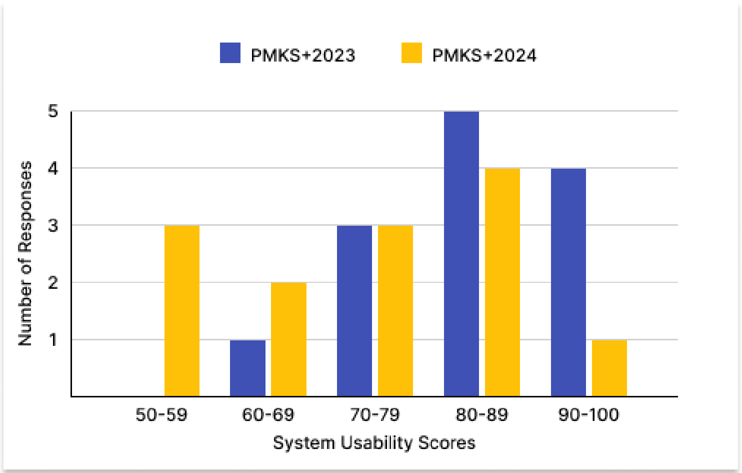


Figure 9.1.1 D Term Survey: System Usability Scale Results

Of the 26 students who took the survey, 19 students agreed that the locking link attributes feature was useful and two students specified that they would also like the ability to lock the attributes of joints.

This was a feature that we had originally looked into and decided there may not be a need for it. However, it could be something that could be re-addressed in the future.

The main task of the survey was to create a mechanism. We asked the students to rank the difficulty of this task and the majority of students using both surveys indicated the level of difficulty on a scale of 1-5 (1: very easy, 5: very difficult) as being either 1 or 2, with an average of 2.5 for PMKS+2024 and 1.9 for PMKS+2023, as shown in figure 9.1.2. The majority of the students were also able to make the mechanism in under five minutes in both systems. This means we were able to successfully recreate this feature in PMKS+2024.

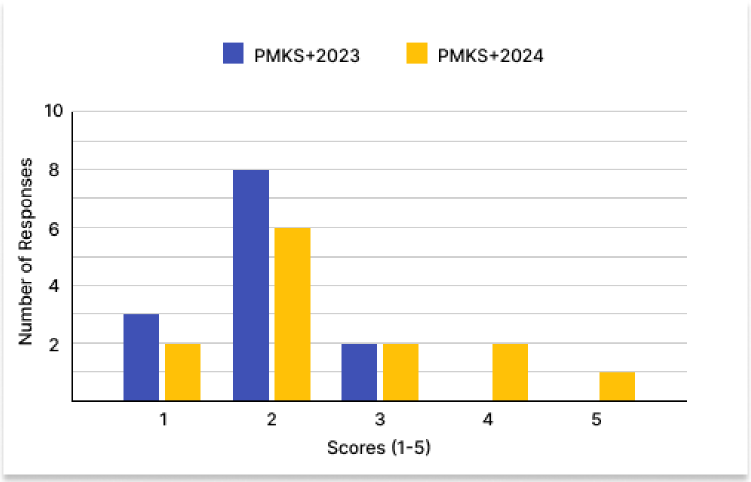
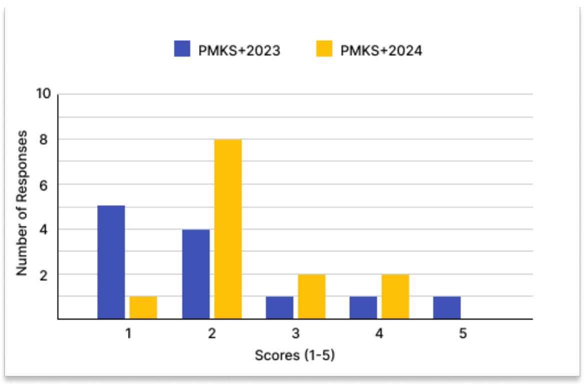
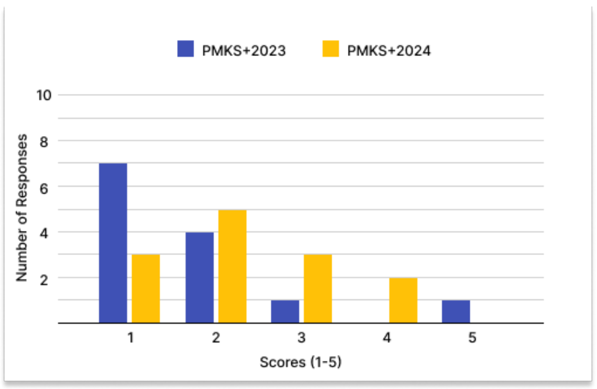


Figure 9.1.2 D Term Survey: Mechanism Creation Difficulty Results

Students were asked to gather analysis data and we asked them to rank the difficulty in accessing that information. We asked them to find three values: a) Position of Joint B, b) Maximum Velocity of Joint B, and c) Center of Mass Acceleration of Link AB, then rank the difficulty (see figure 9.1.3 below). As with creating a mechanism, the majority of students indicated that the level of difficulty was either 1 or 2 as shown in figure 9.1.3. For finding the position of joint B the average difficulty for PMKS+2023 was 2 and PMKS+2024 was 2.3; for finding the maximum velocity of joint B, the average difficult for PMKS+2023 was 1.7 and PMKS+2024 was 2.3; for finding the CoM acceleration of link AB the average difficulty for both systems was 2.2. However, while many students did not indicate they had a difficult time finding the values asked for, many students in both systems were not able to find the correct value. This may show that there is a general disconnect in this area of PMKS+2023 and PMKS+2024.

a) 

b) 

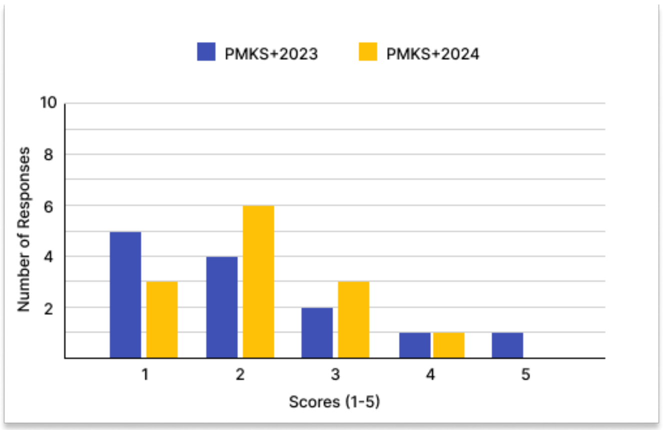
c) 

Figure 9.1.3 a) D Term Survey: Analyze Mechanism Difficulty Results: Position of Joint B

Figure 9.1.3 b) D Term Survey: Analyze Mechanism Difficulty Results: Maximum Velocity of Joint B

Figure 9.1.3 c) D Term Survey: Analyze Mechanism Difficulty Results: Center of Mass Acceleration of Link AB

One feature whose responses were surprising was the ability to shift a whole mechanism by selecting each element while holding shift, which is not a feature that exists in PMKS+2023. We asked students to translate the whole mechanism to the right by 5 cm. Our hope was that the students using PMKS+2024 would have an easier time than the students using PMKS+2023. To our surprise, students using both systems had a wide range of responses when indicating the difficulty of the task, as shown in figure 9.1.4. The average difficulty for translating the mechanism for PMKS+2023 was 2.5 and PMKS+2024 was 3.

Overall, the comments indicated that both systems are fairly straightforward and that they generally like the look and feel of the system. We believe the discrepancy in the scores is mainly due to the time spent creating both systems. PMKS+2023 has been worked on for a year while PMKS+2024 has only been worked on for five months and many of the features in PMKS+2024 still have bugs and have not been completely polished. However, the scores show that we made substantial progress in the limited time that we spent refactoring PMKS+2023 and that we were able to present a usable system in that timeframe. Based on these results, we believe that, with more time, future students on the project could make quick progress. We had anticipated this discrepancy as we were aware of potential bugs in the system. Many students focused their comments on those bugs and these comments can be found in Appendix D. Many students indicated they would in the future like to see an undo/redo feature added as well as an auto locking feature to ensure link and joint attributes cannot change once they are set.

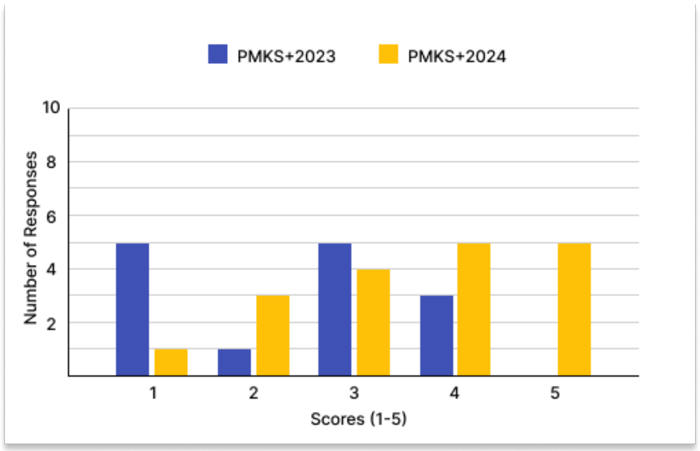


Figure 9.1.4 D Term Survey: Mechanism Translation Difficulty Results

9.2 Codebase Comparison and Developer Experience

Our second area of evaluation was focused on assessing the readability, documentation, organization, and maintainability of our refactored codebase in comparison to the previous version. Through a combination of class diagrams, a developer survey, and direct comparison of code containing the same responsibilities, we show that the refactor we performed dramatically reduced the complexity of PMKS+. This reduction in complexity will simplify the future implementation of new features in PMKS+, and hasten the onboarding of future development teams.

9.2.1 Class Architecture Comparison

We began our code evaluation with a top level view of each of the applications. Shown below in Figure 9.2.1.1 are simplified UML class diagrams with variables and functions removed for readability. The classes are divided into four categories. Blue classes are responsible for representing the state of the mechanism and processing changes to it. Orange classes are controllers that interpret user input sent from the grid. Green classes represent components of the user interface such as the panels and the grid. Lastly, purple classes are services representing utility functions, algorithms for analysis computation, and auxiliary classes that track non-model related state.

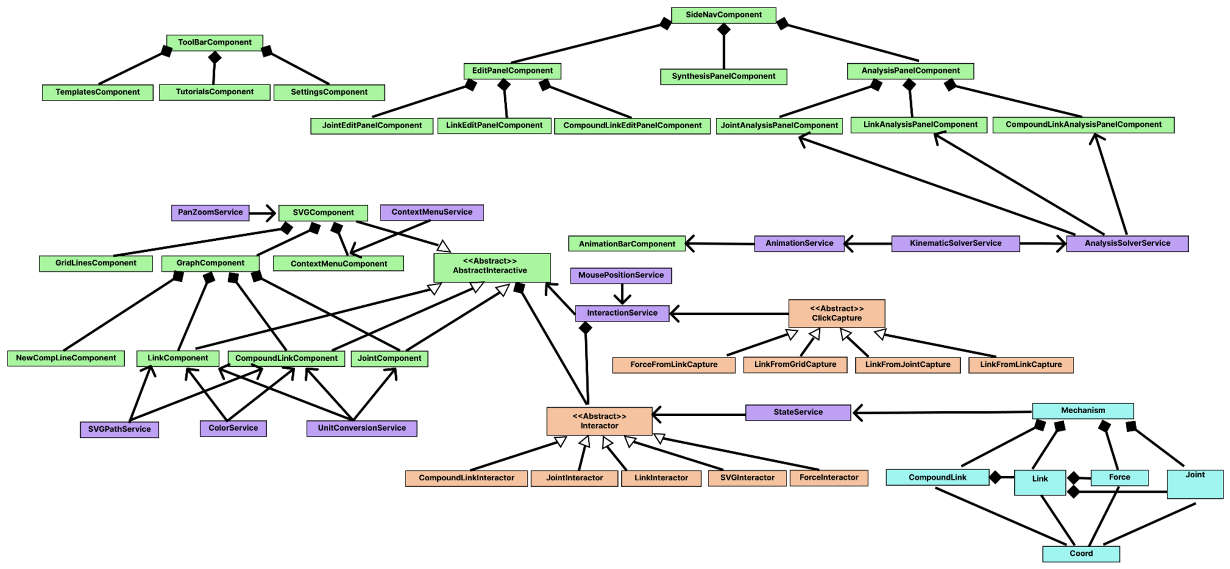


Figure 9.2.1.1 PMKS+2024 Full Class Diagram (Full version at following Link)

Even when disregarding the classes we have not yet implemented, it is clear how much more organized our refactor is in comparison to the PMKS+2023. Much of the interdependence and shared state between classes has been removed. Each individual class now has a single responsibility opposed to the shared responsibilities between classes. Furthermore, our implementation of the Model View Controller architecture decoupled code that was previously responsible for drawing to the screen, handling user input, and updating state into separate classes with greater modularity.

These improvements are most clear when analyzing the full functionality of the grid between the two applications. The code responsible for drawing to the grid, handling user input, and updating the model was previously shared between only four classes in the previous version. These classes were the NewGridComponent, SVG-Grid-Service, and GridUtilsService and Mechanism service with a total of 3700 lines of code between them. In contrast, an almost identical set of functionality was provided in the minimal viable product of our application containing 2600 lines of code split between 27 different classes. Shown below in figure 9.2.1.2 is each of the full UML classes of the previous version.

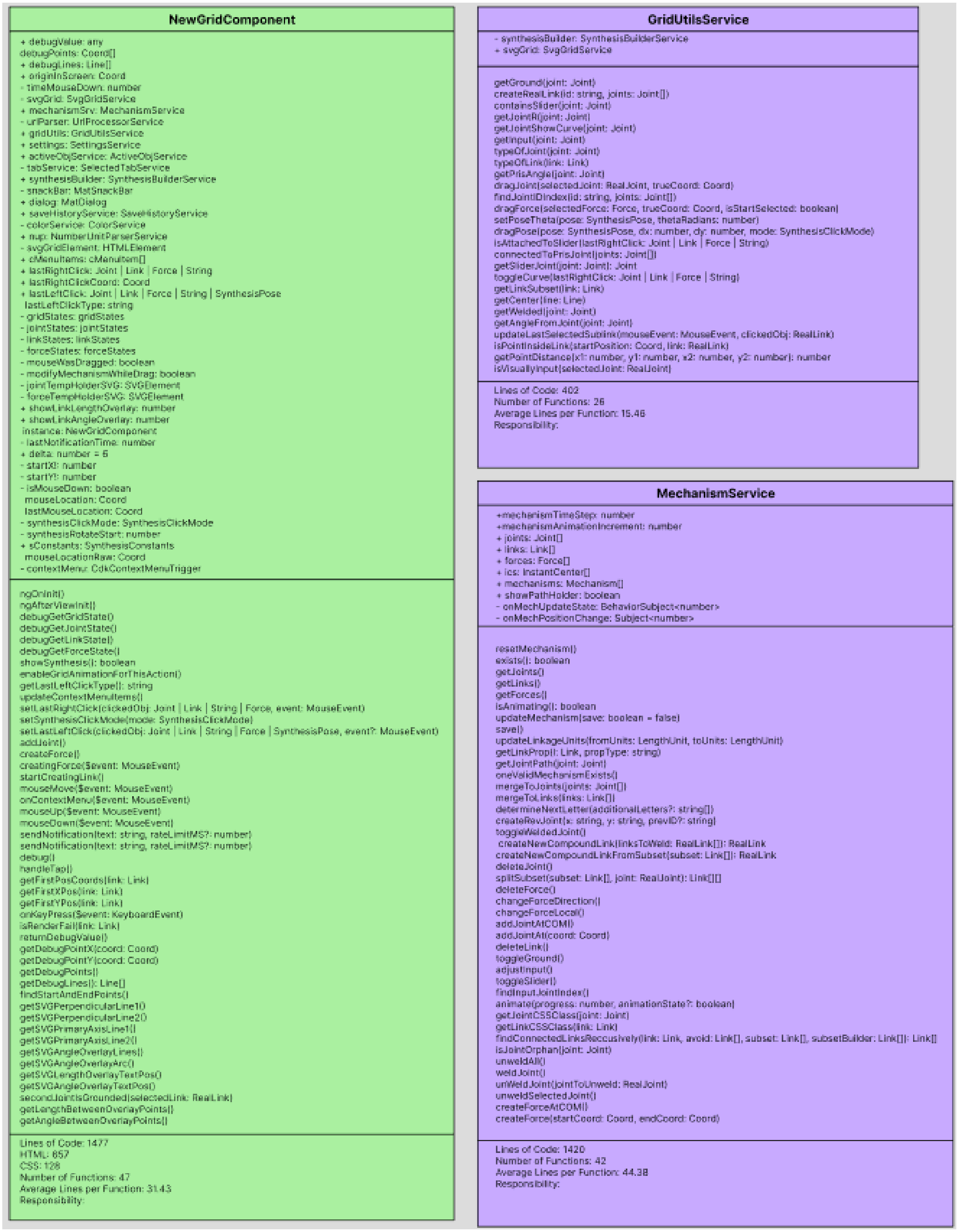

NewGridComponent
+ debugValue: any
debugPoints: Coord[]
+ debugLines: Line[]
+ originInScreen: Coord
- timeMouseDown: number
- svgGrid: SvgGridService
+ mechanismSrv: MechanismService
- urlParser: UrlProcessorService
+ gridUtils: GridUtilsService
+ settings: SettingsService
+ activeObjService: ActiveObjService
- tabService: SelectedTabService
+ synthesisBuilder: SynthesisBuilderService
- snackBar: MatSnackBar
+ dialog: MatDialog
+ saveHistoryService: SaveHistoryService
- colorService: ColorService
+ nup: NumberUnitParserService
- svgGridElement: HTMLElement
+ cMenuItems: cMenuItem[]
+ lastRightClick: Joint | Link | Force | String
+ lastRightClickCoord: Coord
+ lastLeftClick: Joint | Link | Force | String | SynthesisPose
lastLeftClickType: string
- gridStates: gridStates
- jointStates: jointStates
- linkStates: linkStates
- forceStates: forceStates
- mouseWasDragged: boolean
- modifyMechanismWhileDrag: boolean
- jointTempHolderSVG: SVGElement
- forceTempHolderSVG: SVGElement
+ showLinkLengthOverlay: number
+ showLinkAngleOverlay: number
instance: NewGridComponent
- lastNotificationTime: number
+ delta: number = 6
- startX!: number
- startY!: number
- isMouseDown: boolean
mouseLocation: Coord
lastMouseLocation: Coord
- synthesisClickMode: SynthesisClickMode
- synthesisRotateStart: number
+ sConstants: SynthesisConstants
mouseLocationRaw: Coord
- contextMenu: CdkContextMenuTrigger
ngOnInit()
ngAfterViewInit()
debugGetGridState()
debugGetJointState()
debugGetLinkState()
debugGetForceState()
showSynthesis(): boolean
enableGridAnimationForThisAction()
getLastLeftClickType(): string
updateContextMenuItems()
setLastRightClick(clickedObj: Joint | Link | String | Force, event: MouseEvent)
setSynthesisClickMode(mode: SynthesisClickMode)
setLastLeftClick(clickedObj: Joint | Link | String | Force | SynthesisPose, event?: MouseEvent)
addJoint()
createForce()
creatingForce($event: MouseEvent)
startCreatingLink()
mouseMove($event: MouseEvent)
onContextMenu($event: MouseEvent)
mouseUp($event: MouseEvent)
mouseDown($event: MouseEvent)
sendNotification(text: string, rateLimitMS?: number)
sendNotification(text: string, rateLimitMS?: number)
debug()
handleTap()
getFirstPosCoords(link: Link)
getFirstXPos(link: Link)
getFirstYPos(link: Link)
onKeyPress($event: KeyboardEvent)
isRenderFail(link: Link)
returnDebugValue()
getDebugPointX(coord: Coord)
getDebugPointY(coord: Coord)
getDebugPoints()
getDebugLines(): Line[]
findStartAndEndPoints()
getSVGPerpendicularLine1()
getSVGPerpendicularLine2()
getSVGPrimaryAxisLine1()
getSVGPrimaryAxisLine2()
getSVGAngleOverlayLines()
getSVGAngleOverlayArc()
getSVGLengthOverlayTextPos()
getSVGAngleOverlayTextPos()
secondJointIsGrounded(selectedLink: RealLink)
getLengthBetweenOverlayPoints()
getAngleBetweenOverlayPoints()
Lines of Code: 1477
HTML: 657
CSS: 128
Number of Functions: 47
Average Lines per Function: 31.43
Responsibility:
GridUtilsService
- synthesisBuilder: SynthesisBuilderService
+ svgGrid: SvgGridService
getGround(joint: Joint)
createRealLink(id: string, joints: Joint[])
containsSlider(joint: Joint)
getJointR(joint: Joint)
getJointShowCurve(joint: Joint)
getInput(joint: Joint)
typeOfJoint(joint: Joint)
typeOfLink(link: Link)
getPrisAngle(joint: Joint)
dragJoint(selectedJoint: RealJoint, trueCoord: Coord)
findJointIDIndex(id: string, joints: Joint[])
dragForce(selectedForce: Force, trueCoord: Coord, isStartSelected: boolean)
setPoseTheta(pose: SynthesisPose, thetaRadians: number)
dragPose(pose: SynthesisPose, dx: number, dy: number, mode: SynthesisClickMode)
isAttachedToSlider(lastRightClick: Joint | Link | Force | String)
connectedToPrisJoint(joints: Joint[])
getSliderJoint(joint: Joint): Joint
toggleCurve(lastRightClick: Joint | Link | Force | String)
getLinkSubset(link: Link)
getCenter(line: Line)
getWelded(joint: Joint)
getAngleFromJoint(joint: Joint)
updateLastSelectedSublink(mouseEvent: MouseEvent, clickedObj: RealLink)
isPointInsideLink(startPosition: Coord, link: RealLink)
getPointDistance(x1: number, y1: number, x2: number, y2: number): number
isVisuallyInput(selectedJoint: RealJoint)
Lines of Code: 402
Number of Functions: 26
Average Lines per Function: 15.46
Responsibility:
MechanismService
+mechanismTimeStep: number
+mechanismAnimationIncrement: number
+ joints: Joint[]
+ links: Link[]
+ forces: Force[]
+ ics: InstantCenter[]
+ mechanisms: Mechanism[]
+ showPathHolder: boolean
- onMechUpdateState: BehaviorSubject<number>
- onMechPositionChange: Subject<number>
resetMechanism()
exists(): boolean
getJoints()
getLinks()
getForces()
isAnimating(): boolean
updateMechanism(save: boolean = false)
save()
updateLinkageUnits(fromUnits: LengthUnit, toUnits: LengthUnit)
getLinkProp(l: Link, propType: string)
getJointPath(joint: Joint)
oneValidMechanismExists()
mergeToJoints(joints: Joint[])
mergeToLinks(links: Link[])
determineNextLetter(additionalLetters?: string[])
createRevJoint(x: string, y: string, prevID?: string)
toggleWeldedJoint()
createNewCompoundLink(linksToWeld: RealLink[]): RealLink
createNewCompoundLinkFromSubset(subset: Link[]): RealLink
deleteJoint()
splitSubset(subset: Link[], joint: RealJoint): Link[][]
deleteForce()
changeForceDirection()
changeForceLocal()
addJointAtCOM()
addJointAt(coord: Coord)
deleteLink()
toggleGround()
adjustInput()
toggleSlider()
findInputJointIndex()
animate(progress: number, animationState?: boolean)
getJointCSSClass(joint: Joint)
getLinkCSSClass(link: Link)
findConnectedLinksReccusively(link: Link, avoid: Link[], subset: Link[], subsetBuilder: Link[]): Link[]
isJointOrphan(joint: Joint)
unweldAll()
weldJoint()
unWeldJoint(jointToUnweld: RealJoint)
unweldSelectedJoint()
createForceAtCOM()
createForce(startCoord: Coord, endCoord: Coord)
Lines of Code: 1420
Number of Functions: 42
Average Lines per Function: 44.38
Responsibility:

Figure 9.2.1.2 PMKS+2023 "God" Classes (View in full at following Link)

9.2.2 Written Code Comparison

The secondary portion of our code comparison revolved around how the code itself is written, comparing PMKS+2023 and PMKS+2024. This includes several factors such as function length and complexity, naming conventions, and documentation. In essence this is a comparison of the micro-level details as opposed to the macro-level details of the previous section.

In regards to function length and complexity, PMKS+2023 had several issues. A large number of functions had reached over one hundred lines of code in length, and would typically contain a level of nesting between four and nine. The most egregious examples were functions with a length close to 600 lines of code, with a nesting close to nine. By comparison, PMKS+2024 strictly kept our function length below 100 lines, and typically ranged between 20 and 50 lines. We also strictly adhered to a nesting depth of three or less. An example of this would be an if statement within a for loop, which is then within another if statement. To achieve this, we used helper functions extensively, along with guard clauses (which exit out of functions if the conditions are not met) to reduce the complexity of our functions. In essence, our programming practices were based on those found in "Clean Code: A Handbook of Agile Software Craftsmanship" by Robert C. Martin (Martin, 2009).

We also tackled the issue of naming conventions. PMKS+2023 contained inconsistent naming schemes within functions such as “addJoint” and “createLink” which generated confusion when working with the code. When developers have a set of functions that act almost identically, such as adding a subcomponent to a mechanism, this inconsistency forces the developer to remember individual function names as opposed to verb-noun pairs which are interchangeable. This issue was fixed within PMKS+2024 by using consistent naming conventions for functions. Additionally, variable names within the simulator classes would follow the naming conventions used within Mechanical Engineer textbooks, such as “x1”, “y2”, “theta” and so on. For simpler functions this convention worked well but we found readability was difficult at higher levels of abstraction. We instead used more descriptive naming conventions within our code, replacing variables that were previously written as “r1” and “r2” with “distFromKnownJointOne” and “distFromKnownJointTwo”, and with other variables written in a similar fashion. We believe this naming convention, though more cumbersome to type, drastically increases the readability of our code for the new teams that will work on this project in future MQPs.

Lastly, we attempted to improve code documentation for future developers. PMKS+2023 lacked a great deal of expected documentation within the code. Functions themselves were rarely documented correctly and a majority of the comments within the code itself consisted of “TODO”s without clear indication whether they had been completed or not. Within PMKS+2024, we correctly implemented documentation for a majority of our functions and for more complex functions we included comments explaining the steps taken within. All of these changes and additions can be viewed within the codebase on Github through the link found in appendix K.

9.2.3 Developer Survey

In general, the four participants of our developer survey, described in section 5.3.2, had similar things to say about the code. Typically, they had more to say once they were shown the second version, whether that was PMKS+2023 or PMKS+2024, because they had something to compare it to. For PMKS+2023, participants generally were overwhelmed by the length of the files and functions within the files and the density of the functions. They also noted that in general programming practices, components that are not related to each other should be in separate files. For PMKS+2024, participants generally liked the separation of files, the short files and functions, and felt the naming conventions made the code easy to read and understand.

After showing the code, we asked more specific questions about their preferences. All but one of our participants said they preferred to work in a codebase with many short files. The one who preferred working with fewer long files said that it depended on the scope of the project but generally preferred fewer files. Each of our participants said if given the option to work with one codebase, they would choose to work with PMKS+2024 due to its abstraction and readability. Finally, we asked participants if they were to be placed on this project, how easy they felt it would be to learn and begin coding in each codebase. Overall participants were in agreement that PMKS+2024 would be easier to start working on.

This chapter discussed how we evaluated our codebase and the effectiveness of the work we did. We did this through surveys and an evaluation of our codebase. The next chapter discussed the future work needed to restore PMKS+2024 with functionality that existed in PMKS+2023 as well as additional features that could be added to PMKS+2024.

10. Future Work

As extensive as our project was, there still remains a lot of work to be done for any future teams. The following are the recommendations to assist future teams in where to begin their project. There are two areas the team believes can be improved upon by future project groups: firstly, getting PMKS+2024 into a state of equivalent presentability and functionality as PMKS+2023, and secondly, providing additional features that were deemed out of scope for this project.

10.1 Restoring PMKS+

There are a few features from PMKS+2023 that still need to be implemented in the refactor. Adding these features proved to be challenging for us, so to ensure that our project was completed on time we opted to focus our efforts elsewhere. The primary feature missing from our refactor is the Three Position Synthesis panel. Additionally, during development of our refactor, PMKS+2023 was updated to include force creation. Integrating that in the refactor will go hand in hand with correcting our calculations for kinematic analyses.

The UI for Three position synthesis inputs is already completed (shown in section 6.1). We utilized our HTML blocks and basic styling to make it look consistent with the rest of the program. There was no connection to the data or mechanisms in the backend. A future team would need to focus on this if they want to redevelop the refactor to the state of PMKS+2023. Drawing the poses on the screen, generating a mechanism, and allowing the input fields to change the poses are all things that need to be correctly implemented.

Forces and force analyses are the next major features that were fully implemented in PMKS+2022 but not PMKS+2023. In the 2023 version, forces can be enabled as an experimental feature. By attaching a force to a link, shown in figure 10.1.1, the user enables force analysis for themselves. This is a new dropdown in the Analysis panel, with graphs for force and axial stress. Figure 10.1.2 shows the 2023 implementation of this feature. Note that these were never tested but existed as a proof of concept only. The refactor's mechanism and kinematic calculator have been programmed with the development of this feature in mind; in the same way a user can create a new joint, they can create a new force. The data structures and stub functions exist in the important files of our simulator. We intended this to let future developers utilize the same development methods as our team, to create a clean and consistent codebase.

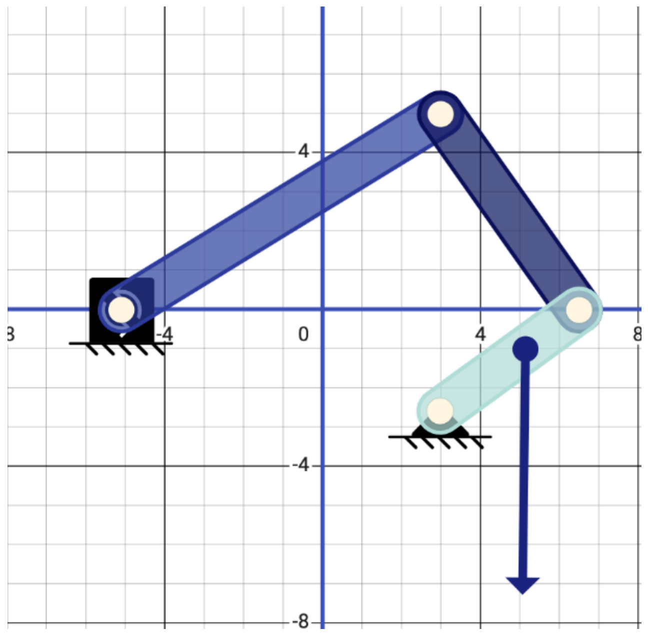
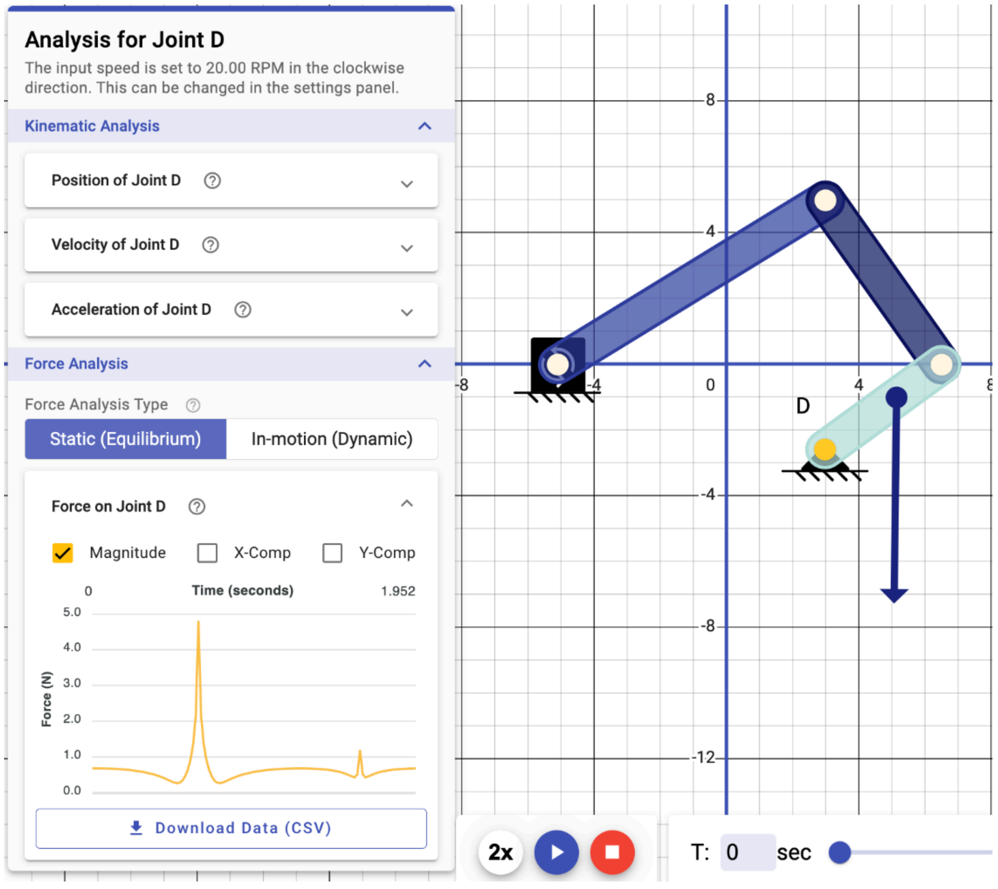


Figure 10.1.1 PMKS+2023 Mechanism with Force Attached to a Link


Analysis for Joint D

The input speed is set to 20.00 RPM in the clockwise direction. This can be changed in the settings panel.

Kinematic Analysis

Position of Joint D

Velocity of Joint D

Acceleration of Joint D

Force Analysis

Force Analysis Type

Static (Equilibrium) | In-motion (Dynamic)

Force on Joint D

Magnitude ✓ X-Comp Y-Comp

Figure 10.1.2 PMKS+2023 Force Analysis Panel

Importantly, our angular kinematic calculations are not mathematically correct. As discussed in section 8.6.4, the angular velocity and acceleration graphs can sometimes be accurate and sometimes be completely indecipherable. Users have a need to access this data, so fixing its calculation and display should be a high priority for future teams hoping to restore PMKS+ to its current working state. By fixing these calculations, the export data component will also need to accommodate the parsed and cleaned datasets of angular velocity and acceleration.

There are a few minor features that exist in PMKS+2023 that are not currently reflected in the refactor. Right now there is not a way for users to provide direct feedback to the development team. PMKS+2023 had feedback go directly into one of the students' emails. We decided that that solution was not future proof, but creating and maintaining a database of feedback associated with PMKS+2023 was out of our scope. Additionally, the ability to change which units are being used for calculation will be extremely useful after forces have been implemented. Everything is set to the unit type we programmed in; centimeter for distance, radian for angles, and kinematic frame timesteps for time. The possibility of changing between units in calculations will create more learning opportunities for students using this tool.

PMKS+2023 utilizes encoding to save a mechanism. The app feeds position data into a tool that encodes the data into a URL. This data can be decoded by the program into a mechanism when sharing URLs with other browsers, for example between professors and students. Our team does not have an encoding process in PMKS+2024. We considered a method to store linkages such as creating an account and having a collection of saved mechanisms, but chose not to pursue this path of development. Another feature is the undo and redo buttons recently added to PMKS+2023. Our team experimented with these at the beginning of our development process, but without encoding and decoding functionality it was difficult for us to implement these features efficiently.

10.2 Additional Features

There are several new features that we discussed in meetings, or attempted to implement but which landed outside the scope of our project. Some of these features include the addition of: stress analysis; modifying the material links and joints are made of; free body diagrams; an additional form of synthesis; and uploading position data from a CSV file.

A free body diagram is something that would naturally follow after adding forces and force analyses. Students use free body diagrams for a variety of applications, both homework and project related. The end goal is for PMKS+ to intuitively create a free body diagram that the user could export as an image. All relevant information about forces and masses would be condensed into a few shapes for a student to easily comprehend.

Uploading position states in the form of a CSV file will act as an additional way to save and share mechanisms and linkages. Rather than encoding, which relies on a URL being sent from peer to peer, a CSV can be more easily manually input, and can be shared with more ease through a tool such as sharepoint or onedrive. Uploading a CSV full of data is another alternative for dissemination of mechanisms by faculty to students.

An additional kind of synthesis was proposed to us that we were unable to implement: path synthesis. Path synthesis had been an experimental feature implemented in PMKS+2022 which only included a basic UI. Path synthesis involves generating a mechanism where a joint or a tracer point traces the path specified by the user by specifying a series of points. This path is then used as a guide for positions that a mechanism should follow during its animation. We created initial mockups for potential UIs shown in Appendix F.

An additional feature that our group feels could be beneficial in the future would be to connect the application to a database. This would allow users to be able to make accounts and store their mechanisms within PMKS+. This would easily pair with the re-implementation of encoding and decoding that existed in PMKS+2023. The database would store user information and each user account could have a list of their mechanisms encoded into strings that would be decoded once the file is opened.

There are also small additions that would be beneficial to the application. Once the application is polished and restored with the functionality listed in section 10.1, video tutorials should be created and added to the “Tutorials” tab in the top bar. One feature that is often requested by users is the ability to “snap to grid”. This would allow for users to be able to make perfect right angles or place links to align with grid lines. Lastly, we recommend that the animation service is decoupled from the state. In PMKS+2024, the animation creates a new instance of the state and re-draws the SVGs in their new positions on the screen. If this was decoupled, animation would be separate from the state and would ensure that no editing could occur while animating the mechanism. This is important because if edits are made while animation is occurring, it could disrupt the application and cause bugs to occur.

This chapter discussed the future work needed to restore and add more functionality to PMKS+2024. The next chapter discusses the conclusions made both as a team as well as our individual reflections on the project.

11. Conclusions

In chapter 3, we discussed our goals and requirements for the project. The requirements were created to be achievable and testable. These requirements included:

- Simplify the logic of the backend code
- Add documentation to the entire codebase
- Create mockups based on user & expert feedback
- Adhere to HCI principles
- Create a list of features that exist in the current application
- Create a list of features that need to be added
- Implement the ability to create and analyze multiple mechanisms

As shown in section 9.2, we significantly decreased the length and complexity of the entire codebase. Using the Model-View-Controller (MVC) architecture, we added structure to the codebase. We also added a significant number of comments to the codebase to ensure each function's purpose could be easily understood. This simplification will allow future developers to easily understand the codebase and be able to quickly add new functionality.

As shown in section 5.4, we conducted a heuristic evaluation on PMKS+2023 in order to understand the problem areas and where the standard HCI principles were not followed. Specifically these principles were lacking with respect to feedback to the user, and the standard representation of foreground and background. We created the mockups shown in chapter 6, as well as those shown in Appendices E to I, to address each of the problem areas.

Sections 10.1 and 10.2 indicate a list of functionality that exists in PMKS+2023 and functionality that still needs to be added to PMKS+. As shown in section 5.2, we also created a list, based on the previous work done on PMKS+, as well as the Github repository for PMKS+, of features that need to be added or fixed in PMKS+. The full list can be found in Appendix L.

As shown in section 5.5, we were able to implement the functionality to analyze and animate multiple mechanisms. Each mechanism animates simultaneously and does not yet allow for a user to select only specific mechanisms to animate, but it sets the foundation to implement that in the future.

11.1 Reflections

Tyler Evans

Working on this project has been unlike anything I've accomplished before at WPI. My time with PMKS+ has taught me how much I've learned in the last four years of studying Computer Science. Specifically in the realm of software design and development, being able to take an existing product and refactor it into an application a fraction of the size, with comparable functionality, has been a pleasant surprise. Additionally, to do so without any prerequisite knowledge of the tech stack, or planar mechanisms, has been an accomplishment I will continue to be proud of for years. I'm thrilled to have taken the skills and generalized knowledge I've acquired over my years at WPI and seen the fruits of that labor in this project. The speed at which our team was able to learn and develop this application with Angular has given me the determination needed to pursue a career in software development.

Nicole Burgess

This project has taught me a lot about my own abilities. I was able to showcase my ability to design, which is not something I've had a lot of experience with in my previous classes. However, I found that I really enjoyed it. This project allowed me to further develop the skills that I had gained working with HTML, CSS and Typescript in my Webware (CS4241) and Design of Software Systems (CS509) courses. This project had a significant learning curve that we had to overcome at the beginning of the project. I didn't have experience working on an existing project, let alone a project using a framework I was unfamiliar with. It helped me to learn more about how to use Angular and taught me about the importance of well written code when working on a long-term team-based project.

Robert Eskridge

This project was an undertaking unlike anything I've done before at WPI. In CS 3733 (Soft Eng), my team of ten people worked together to make a simple desktop application with limited functionality. In CS 542 (Graduate Database Management Systems), my team of three people worked together to make an extremely simple web app with a database attached to it. Here in this MQP, we worked as a team of three to create a robust web application nearly from scratch, as we outlined in the above report. Working as a team to overcome great obstacles was something I learned on my IQP. Working as a team to handle confusing new CS topics was something I learned in CS3733. I've never had to work with the Angular framework, or HTML, or Typescript. I've certainly never had to work with four bar linkages and the complex kinematics behind them. Using the skills I've acquired over my time at WPI, I knew exactly where to look to get the resources I needed to succeed. Whether those resources came from my own knowledge, or the expertise of my teammates, or the advisors, our team had all of the ability this project demanded of us.

12. References

Appikatla, P., Cecil, G., Taylor, M., & Tsiakmakis, D. (2019). *PMKS+: Recreating a Legacy Application*. Unpublished Major Qualifying Project. Worcester Polytechnic Institute.

Benyon, D. (2013). *Designing Interactive Systems: A Comprehensive Guide to Hci, Ux & Interaction Design* (3rd ed.). Pearson Education.

Chart.js. Chart.js | Chart.js. (n.d.). https://www.chartjs.org/docs/latest/

Chris, K. (2023, February 22). Solid design principles in software development. freeCodeCamp.org. https://www.freecodecamp.org/news/solid-design-principles-in-software-development/#:~:text=SOLID%20is%20an%20acronhttps://www.freecodecamp.org/news/solid-design-principles-in-software-development/#:~:text=SOLID%20is%20an%20acronym%20that,Liskov%20Substitution%20Principle%20(LSP)ym%20that,Liskov%20Substitution%20Principle%20(LSP)

Dowd, T., Zhang, H., & Dutile, R. (2020). *PMKS+: Enhancements*. Unpublished Major Qualifying Project. Worcester Polytechnic Institute.

Fimga [Computer software]. (2023). Retrieved from https://www.figma.com/

Galvan, A. (2022). *PMKS+: A Tool to Analyze and Synthesize Planar Mechanisms.* Unpublished Master's Thesis. Worcester Polytechnic Institute.

Kadoya, K. (2023). *Improving the User Experience and Features of PMKS+: A Web-Based Linkage Analysis Tool for Education.* Unpublished Master's Thesis. Worcester Polytechnic Institute.

Lyu, Z., Purwar, A., and Liao, W. (December 15, 2023). *A Unified Real-Time Motion Generation Algorithm for Approximate Position Analysis of Planar N-Bar Mechanisms.* ASME. J. Mech. Des. June 2024; 146(6): 063302. https://doi.org/10.1115/1.4064132

Mad-croc nederland. SAM - The Ultimate Mechanism Designer Software. Artas Engineering. (2020). https://www.artas.nl/en/

Purwar, A., Deshpande, S., and Ge, Q. J. (March 9, 2017). *MotionGen: Interactive Design and Editing of Planar Four-Bar Motions for Generating Pose and Geometric Constraints.* ASME. J. Mechanisms Robotics. April 2017; 9(2): 024504. https://doi.org/10.1115/1.4035899

Martin, R. C. (2009). *Clean code: A Handbook of Agile Software Craftsmanship*. Prentice Hall.

Shete, S., Kumar, N., Nalawade, P., & Tripathi, P. (2017, May). *Review of synthesis of four bar mechanism.* International Research Journal of Engineering and Technology (IRJET). https://www.irjet.net/archives/V4/i5/IRJET-V4I5297.pdf

SOLIDWORKS 3D CAD. SOLIDWORKS. (2023, August 9). https://www.solidworks.com/product/solidworks-3d-cad

Thomas, N. (2019, September 13). How to use the system usability scale (SUS) to evaluate the usability of your website. Usability Geek. https://usabilitygeek.com/how-to-use-the-system-usability -scale-sus-to-evaluate-the-usability-of-your-website/

Torres-Moreno, J. L., Cruz, N. C., Álvarez , J. D., Redondo, J. L., & Giménez-Fernandez, A. (2021, November 19). *An open-source tool for path synthesis of four-bar mechanisms*. Mechanism and Machine Theory. https://www.sciencedirect.com/science/article/pii/S0094114X21003438

Working Model 2D. Working Model 2D - 2D Kinematics & Dynamics Software - Engineering Simulait. (n.d.). https://www.design-simulation.com/WM2D/Index.php

Waldron, K. J., Kinzel, G. L., Agrawal, S. K. (2016). *Kinematics, Dynamics, and Design of Machinery*. United Kingdom: Wiley.

Appendices

Appendix A: A Term 2023 Survey Questions

PMKS+ User Testing

This user evaluation is used to test the effectiveness of the PMKS+ software. This software is proposed to be used in the ME-RBE 4322/ME 3310/ES2503/RBE 2001 courses in the future. Data and conclusions made from these evaluations will be used to assess areas of strengths and weaknesses in the system. All responses will be anonymized before summarizing data.

Email*

Short answer text

Name*

Short answer text

Major*

ME

RBE

Other:

Year of Graduation*

2023

2024

2025

Other:

Select the courses you've had prior to ME/RBE 4322*

ES 2501

ES 2502

ES 2503

ME 3310

ME 3320

ME 3311

RBE 2001

RBE 3001

ME 4320

Other:

Do you have experience with any of the tools below?*

SolidWorks

20Sim

Autodesk Inventor

Working Model

ADAMS

ANSYS

MATLAB

Linkages

SAM

Other:

Go to the following link to access PMKS+:https://app.pmksplus.com Please keep this URL somewhere you can copy it from, as you will be asked to close and reopen the application several times.

Step 1:

When PMKS+ loads for the first time, a tutorial shows up (if not, please use Incognito Mode / Private Browsing Mode to load PMKS+). Please go through the tutorial and answer the following questions.

Did the tutorial give you a good overview of the software's features?*

Yes

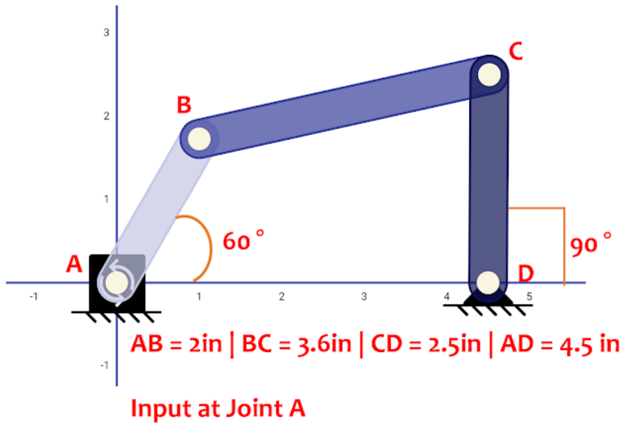
No

Partially

Do you have suggestions to improve the tutorial?

Short answer text

Create the following mechanism using PMKS+ with an input speed of 10rpm in the clockwise direction. Note: Please make sure to set the appropriate unit system from the "Settings" Panel.



How easy was it to create the mechanism?*

Very easy 1 2 3 4 5 Very difficult

How long did it take to create the mechanism?*

Less than 1 min

> 1 min but < 5 min

> 5 min but < 10 min

> 10 min

When the input link makes 0-deg with respect to the horizontal axis, can you specify the angular velocity of the coupler link BC at that position of the input link?*

Short answer text

How easy was it to do the above task?*

Very easy 1 2 3 4 5 Very difficult

Do you have any additional feedback related to the process of getting the angular velocity of link BC?

Short answer text

Please share the URL of the mechanism you created within PMKS+*

Please change the names of links to the following:

AB to "Input"

BC to "Coupler" and

CD "Follower" respectively.

(Note: You will need to press the stop button on the animation bar before doing this task)

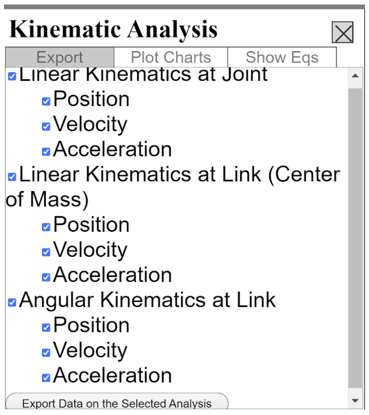
How easy was it to do the above task?*

Very easy 1 2 3 4 5 Very difficult

Please share the URL of the updated linkage here*

Your answer

The earlier version of PMKS+ had the option to download the complete kinematic data into an Excel Spreadsheet. Would you prefer that a similar feature be available in the new version?*



Kinematic Analysis

Export | Plot Charts | Show Eqs

- ☑ Linear Kinematics at Joint
  - ☑ Position
  - ☑ Velocity
  - ☑ Acceleration
- ☑ Linear Kinematics at Link (Center of Mass)
  - ☑ Position
  - ☑ Velocity
  - ☑ Acceleration
- ☑ Angular Kinematics at Link
  - ☑ Position
  - ☑ Velocity
  - ☑ Acceleration

Export Data on the Selected Analysis

Yes

No

Maybe

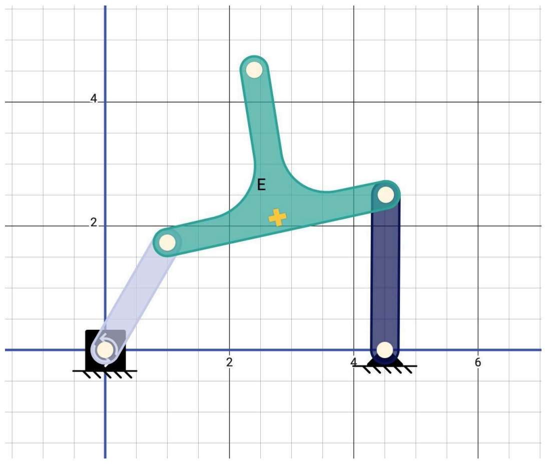
Other:

Please elaborate on the reasons for your choice on the previous question*

Short answer text

Go to the "Settings" panel, and under "Experimental Features", please click on "Enabled Welded Joints."

Change the shape of the coupler link from a straight bar to a T-shaped link as shown below



How easy was it to do the above task?*

Very easy	1	2	3	4	5	Very difficult

Please share the URL of the linkage with the T-shaped coupler link *

The three position synthesis technique is used to generate a four-bar mechanism with revolute joints. You begin by specifying the three positions that the coupler link is supposed to reach.

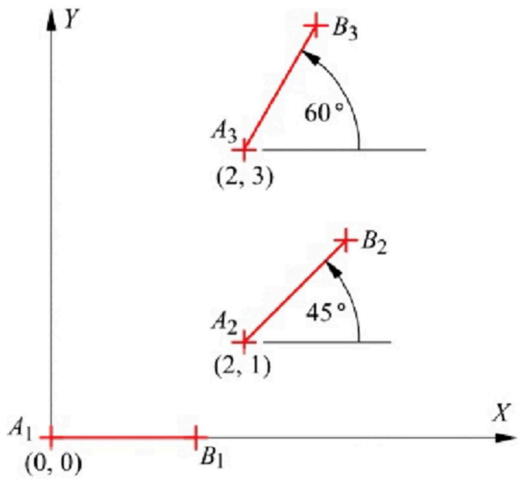
Note:

* Click on "New" before you do this task.

Shown are three positions (referred to as Poses). The length of each link segment is 1.50in. Use the Synthesis feature to generate a four-bar linkage solution.

Note:

* You may need to change the location of the fixed reference.
* Also, make sure to set the units under the "Settings" Panel before doing this task.



Please upload a screenshot of the four-bar linkage synthesized by PMKS+*

Is the generated four-bar linkage capable of reaching all the three poses? *

Yes

No

Partially

Don't Know

Other:

If you selected "No" or "Partially" for the previous question, please indicate how you came to that conclusion.

Short answer text

Change the coordinate of A3 in Pose 3 (represented by line A3 B3) from (2,3) to (-2,3) and what do you infer about the synthesized four-bar linkage? *

Short answer text

Please share a screenshot of the synthesized four-bar linkage with the changed third pose.*

Would you prefer that PMKS+ provide more information about the synthesized mechanism or the three-position synthesis itself? *

Root Mean Squared Error between the given positions and the positions reached by the synthesized four-bar mechanism

Construction lines showing three-position synthesis

Other:

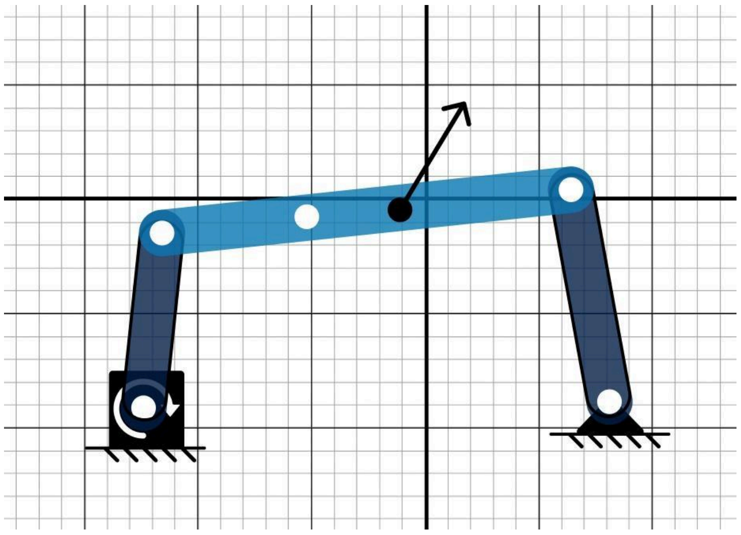
In the "Settings" panel, under "Visual Settings," you will see an option for "Object Scale."

Are you able to infer its use? Please explain*

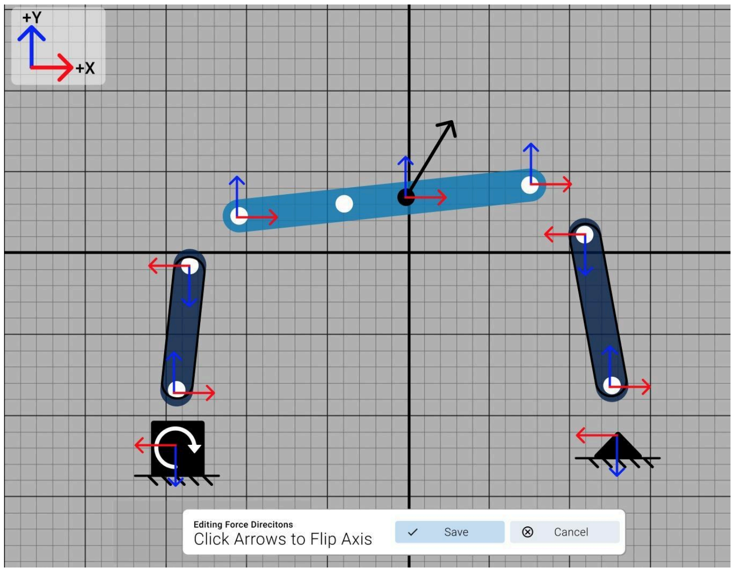
Short answer text

This section will request your feedback on the proposed Free Body Diagram display within PMKS+

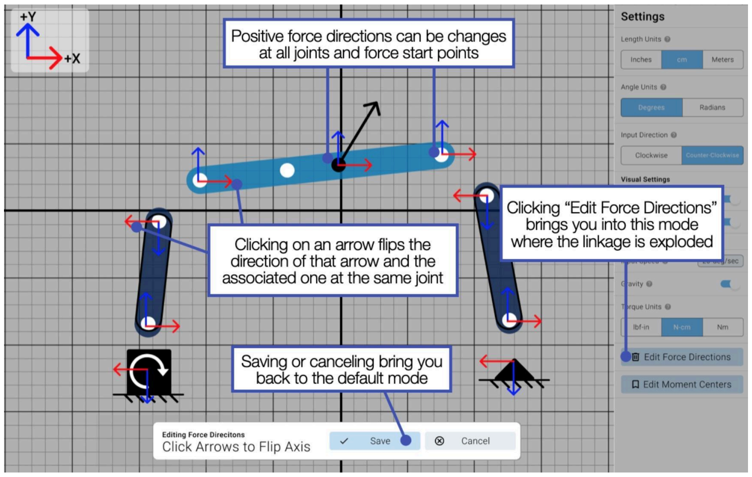
We are looking to incorporate FBDs into PMKS+. Consider the following four-bar mechanism with an applied load.



The FBDs will be displayed as shown below with the ability to change force directions and moment references

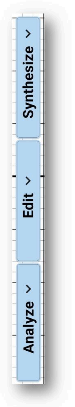


A brief explanation of the proposed UI is given below:

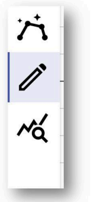

Positive force directions can be changes at all joints and force start points
Clicking "Edit Force Directions" brings you into this mode where the linkage is exploded
Clicking on an arrow flips the direction of that arrow and the associated one at the same joint
Saving or canceling bring you back to the default mode
Editing Force Direcitons
Click Arrows to Flip Axis
Save
Cancel
Settings
Edit Force Directions
Edit Moment Centers

Please provide your thoughts on the FBD display and the ability to change force directions*

Shown below are two images. "Option A" displays the three main features of PMKS+ in words (Synthesize, Edit, Analyze) while "Option B" uses icons for these three features.

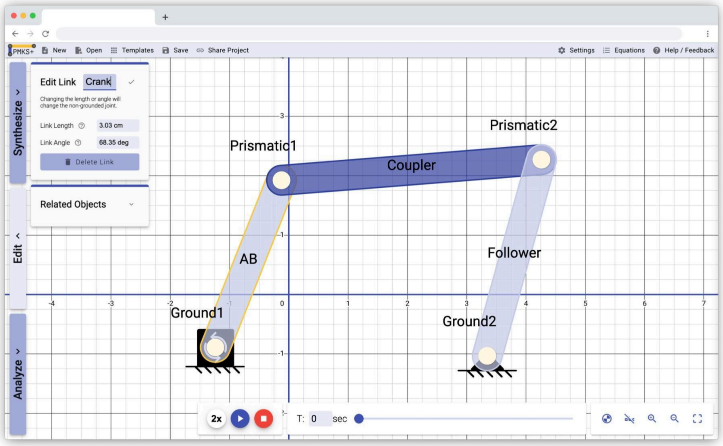


Option A

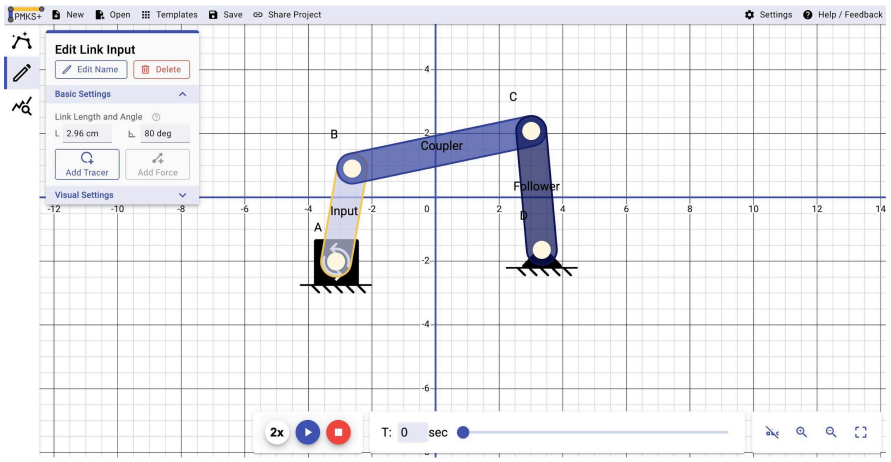


Option B

Option A appears as shown below



Option B (that is available on the website and is the current version) appears as shown below



Which one do you prefer? Please indicate your preference *

Option A (with words representing the three main features of PMKS+)

Option B (using icons to represent the three main features)

It does not matter

Other:

If you would like to elaborate on your choice, feel free to use this space

Short answer text

What are some features you would prefer to be integrated into the system?*

Show kinematics and force equations

Perform Stress Analysis

Generate CAD Models of the links

Generate MATLAB script for analysis

Integrate different synthesis methods (path, function, etc.)

Show instant centers

Other:

Any other comments

Short answer text

Appendix B: A term Survey Results

Full Results Found Here: Aggregated Survey Notes / Responses

ME/RBE 4322 PMKS+ Survey A Term 2023

Above is a summary of the results received along with the raw data from the survey.

Forty-five students took the A Term 2023 survey. Each of these students were pursuing either a major or minor in Mechanical Engineering. Eleven percent were additionally pursuing a Robotics Engineering degree. This is an expected skew, as the survey was distributed to students in the Modeling and Analysis of Mechatronic Systems class (ME4322). All but one respondent had experience in SolidWorks, and over half had experience in MATLAB. This experience is relevant as it informs the expected behavior of PMKS+ from the user perspective.

Fifty-nine percent of respondents, using a five point scale, found it very easy (1) or somewhat easy (2) to replicate the given four bar linkage, and an additional thirty percent found it neither easy nor difficult (3). This implies that the act of simply creating a four bar linkage is intuitive and flows naturally for the average user. The first "competency question" was finding the angular velocity of a specific selected joint. The respondents who found it less difficult to create the linkage (1-3) tended to get the correct answer, while the respondents that found it more difficult (4-5) tended to get wrong answers. This seemed to imply that the students who were experienced in creating linkages, either from other software or from PMKS+, were able to assess the various tools available to them in PMKS+ to find angular velocity much easier than the students who were less familiar with PMKS+ and similar tools.

When finding angular velocity, the survey asked students to find the value at a specific time. The most effective and obvious way of doing so was to pause the simulation as close to the desired time as possible. Twenty-nine percent of students specifically gave feedback requesting the addition of some functionality that would allow the user to input a specific time into the simulation. While the difference in values was negligible from those who paused at and around the correct moment, the user still experienced doubt about the validity of their answer due to the hit-or-miss nature of the selection method. Ten percent of students expressed interest in a tool that would allow them to see the angle of a specific joint on screen as the simulation moved, again for the purpose of assisting the selection of specific moments during the simulation.

Ninety-seven percent of respondents selected that they would like to see a feature allowing the download of kinematic data into a csv file. The reasoning for this is 27 responses specified that students had an easier time doing data analysis from a CSV file than from the PMKS+ tools. 16 responses indicated that manipulating the data (i.e. changing values to accommodate theoretical situations, or to do calculations not currently supported in PMKS+). Fourteen responses indicated that CSV file data formats were easier to view than PMKS+ visualizations.

Multiple needs could be inferred from these responses. The first is the need for an export feature, as almost every student was solidly in favor of one. The second is the need to redesign the way data is presented in PMKS+, because a sizable number of students would prefer to create their own data, rather than utilize the existing tools to show their data.

In terms of the UI side navigation, sixty-four percent of respondents wanted only icons to represent the three states of PMKS+. Twenty percent indicated a preference for words, and fifteen percent had no preference. The respondents who opted for only icons and offered rationale were primarily students of intermediate ability. These students primarily cited a desire to reduce clutter as their biggest reason for choosing icons only. By contrast, a majority of students who desired words for the UI side navigation tabs were novice users. These students primarily cited a confusing interface to navigate, and that words on their screen would help them in figuring out where to access certain tools and features.

Appendix C: D term Survey Questions

PMKS+ Usability

The objective of this MQP is to recreate PMKS+, a tool for making planar mechanisms for novice users, and make it less complex and more user friendly.
The purpose of this survey is to evaluate the usability of the application and the effectiveness of new features that have been implemented.

PMKS+ MQP Students: Nicole Burgess (CS), Robbie Eskridge (CS), Tyler Evans (CS)
Advisors: Profs. David Brown (CS) and Pradeep Radhakrishnan (ME)
Contact Info: gr-pmksplus@wpi.edu

All responses will be anonymized before summarizing data

User Information

Email*

Short answer text

Name*

Short answer text

Major*

ME
RBE
Other…

Graduation Year*

2023
2024
2025
2026
Other…

Select the courses you've had*

ES2501
ES2502
ES2503
ME3310
ME3320
ME3311

RBE2001

RBE3001

ME4320

ME/RBE 4322

Other…

Do you have experience with any of the tools below?*

SolidWorks

20Sim

Autodesk Inventor

Working Model

ADAMS

ANSYS

MATLAB

Linkages

SAM

PMKS

MotionGen

Other…

Video Tutorial

Please watch the short video below to familiarize yourself with the software (PMKS+)

PMKS+ Tutorial

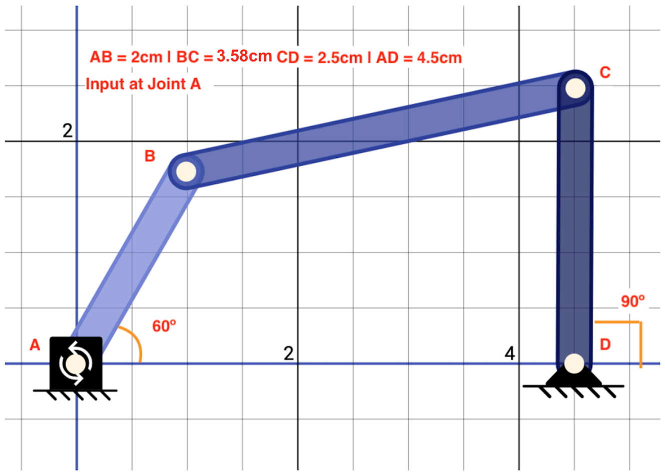
Please open the web application found at this link: PMKS+ Refactor PMKS+2023

If you need to revisit the tutorial, it can be found at this link: https://youtu.be/VcR1cYgOrDQ (Refactor) https://youtu.be/TQec4-XcfLs (PMKS+2023)

Once you open the application and watch the tutorial, please perform the following task and complete the survey afterwards.

****Note: Please focus on the functionality of the application rather than the look of the application****

Create the following mechanism (due to rounding in the system, values may be off by +/- .1)



On a scale of one to five, rank the difficulty of creating the mechanism.*

Very easy	1	2	3	4	5	Very difficult

Did you make use of the locking feature? Was it useful? If not, would you find a locking feature useful?*

Short answer text

Approximately how long did it take you to make the mechanism*

Less than 1 min

> 1 min but < 5 min

> 5 min but < 10 min

> 10 min

What is the position of Joint B at time stamp 74 (time stamp 4 PMKS+2023)? (x,y)*

Short answer text

On a scale of one to five, rank the difficulty of finding the position of Joint B.*

Very easy	1	2	3	4	5	Very difficult

What is the maximum velocity of joint B in the Y direction?*

Short answer text

On a scale of one to five, rank the difficulty of finding the maximum velocity of Joint B.*

Very easy	1	2	3	4	5	Very difficult

What is the center of mass acceleration of link AB at time 303 (7.2 sec PMKS+2023) in the X direction?*

Short answer text

On a scale of one to five, rank the ease of finding the center of mass acceleration of link AB.*

Very easy	1	2	3	4	5	Very difficult

Now, take the mechanism you created and translate the whole mechanism 5 cm to the right. On a scale of one to five, rank the difficulty of this task.

For example Joint A would be at (5,0) instead of (0,0) and Joint D would be at (9.5,0) instead of (4.5,0)*

Very easy 1 2 3 4 5 Very difficult

Do you have any comments about how easy/hard the application is to use?

Long answer text

Overall, do you have any comments about the look and feel of using the application?*

Long answer text

Please include a screenshot of the mechanism you created.*

System Usability Scale

Please respond to these questions based on how much you agree with the presented statements.

I think I would use this system frequently.*

Never 1 2 3 4 5 Very Frequently

I found the system unnecessarily complex.*

Not complex 1 2 3 4 5 Very Complex

I thought the system was easy to use.*

Hard 1 2 3 4 5 Easy

I think that I would need the support of a technical person to be able to use this system.*

No Support Needed 1 2 3 4 5 Would need tech support

I found that the various functions in this system were well integrated.*

Not well integrated 1 2 3 4 5 Well integrated

I thought there was too much inconsistency in this system.*

No inconsistencies 1 2 3 4 5 Very inconsistent

I would imagine that most people would learn to use this system very quickly.*

Hard to learn 1 2 3 4 5 Easy to learn

I found the system very cumbersome/complex.*

Not cumbersome 1 2 3 4 5 very cumbersome

I felt very confident using the system.*

Not confident 1 2 3 4 5 Very confident

I needed to learn a lot of things before I could begin working with this system.*

Don't need to learn a lot 1 2 3 4 5 Need to learn a lot

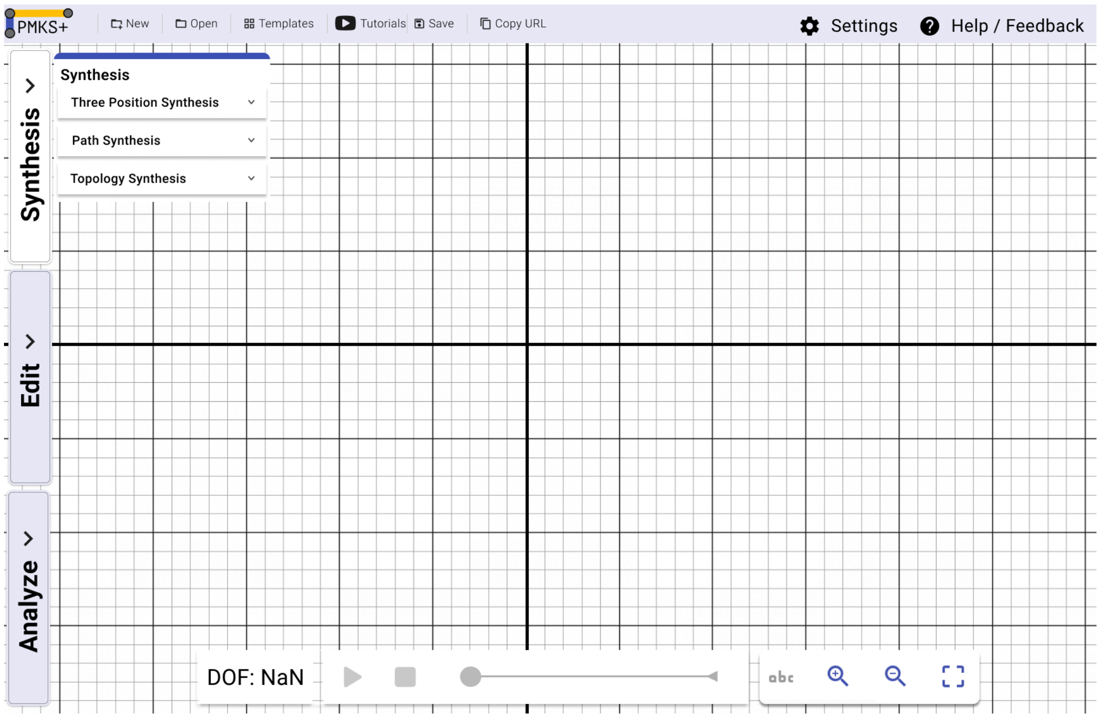
Appendix D: D term Survey Results

The link below includes 3 spreadsheets. One for using PMKS+2023, one for using PMKS+2024, and one for the analysis of both.

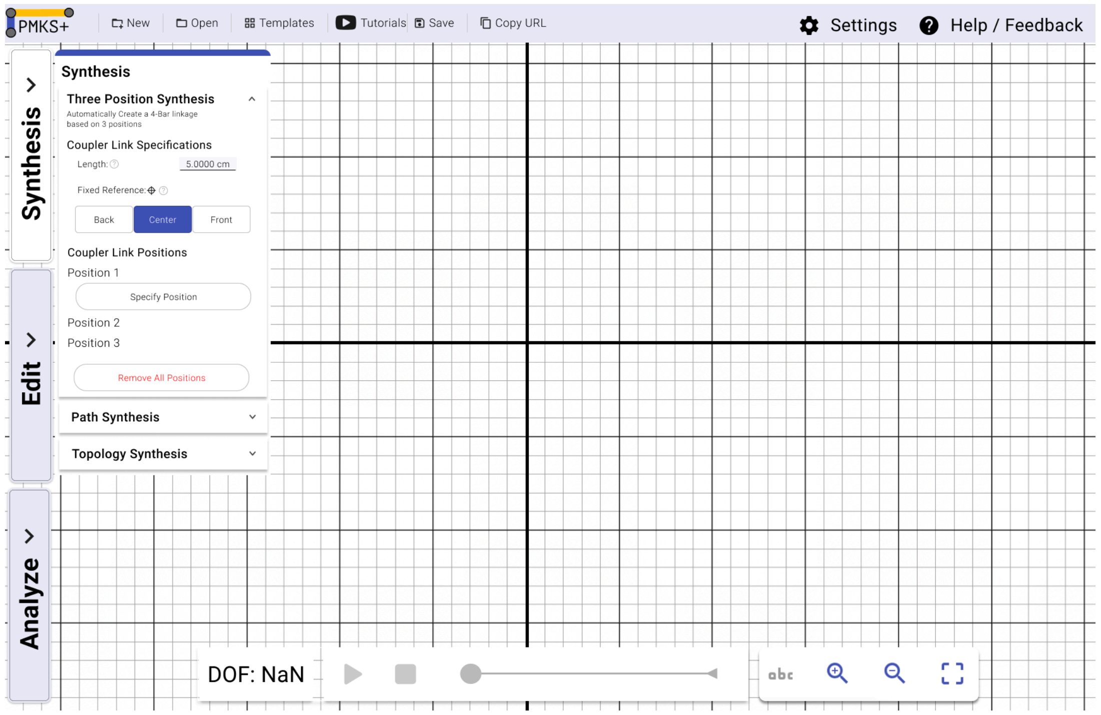
PMKS+ Usability (Responses)

Appendix E: Three Position Synthesis Mockups

Open Synthesis Panel

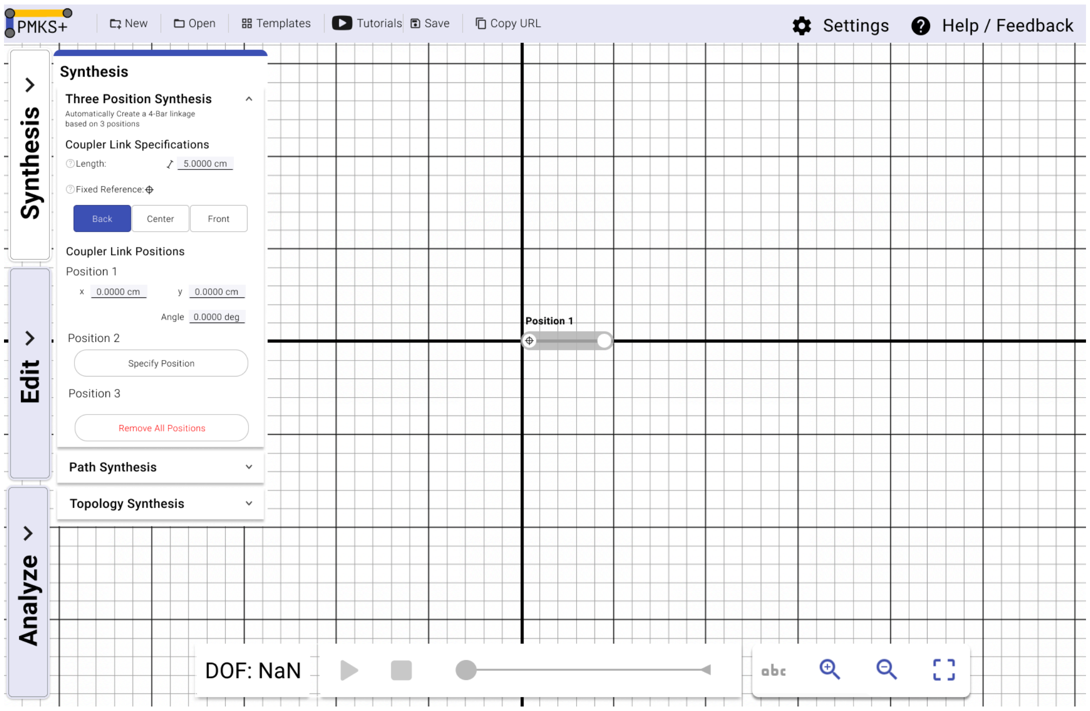


Open Three Position Synthesis



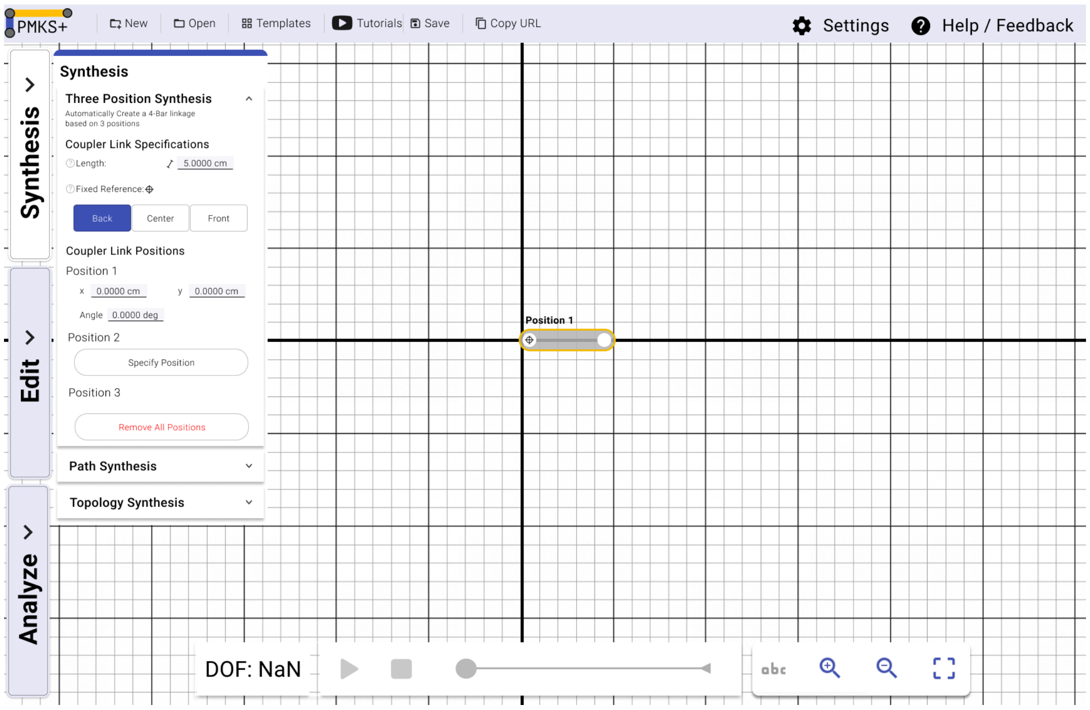
Length: 5.0000 cm

Fixed Reference: Back [✓ Center] Front



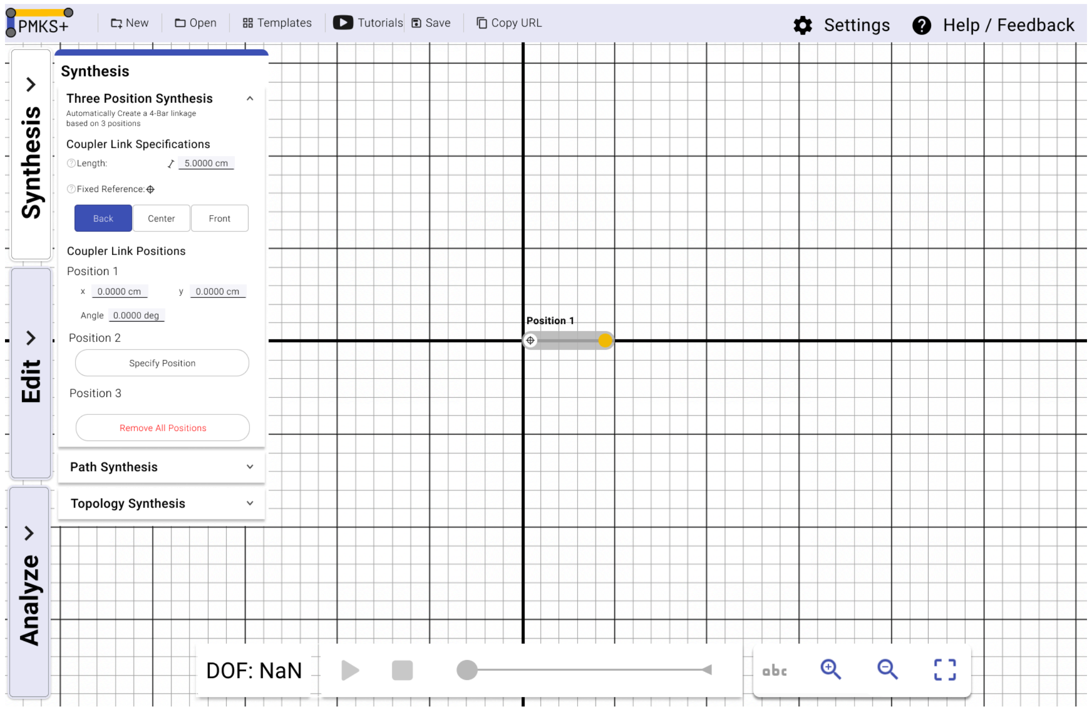
Length: 5.0000 cm
Fixed Reference: Back
Position 1 x 0.0000 cm y 0.0000 cm Angle 0.0000 deg
Position 2
Position 3

Click on position to select



Length: 5.0000 cm
Fixed Reference: Back
Position 1 x 0.0000 cm y 0.0000 cm Angle 0.0000 deg
Position 2
Position 3

Click on joint to drag angle



Synthesis

Three Position Synthesis
Automatically Create a 4-Bar linkage based on 3 positions

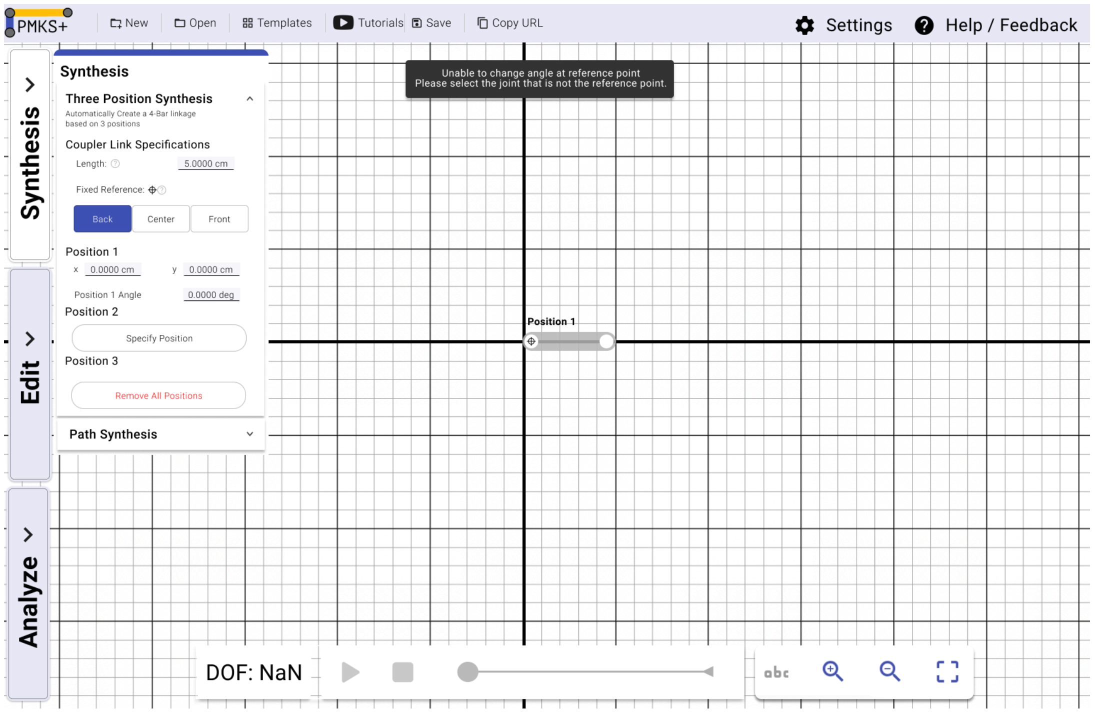
Coupler Link Specifications
Length: 5.0000 cm
Fixed Reference:
Back | Center | Front

Coupler Link Positions
Position 1
x 0.0000 cm y 0.0000 cm
Angle 0.0000 deg
Position 2
Specify Position
Position 3
Remove All Positions

Path Synthesis

Topology Synthesis

Click on reference joint error



Unable to change angle at reference point
Please select the joint that is not the reference point.

Synthesis

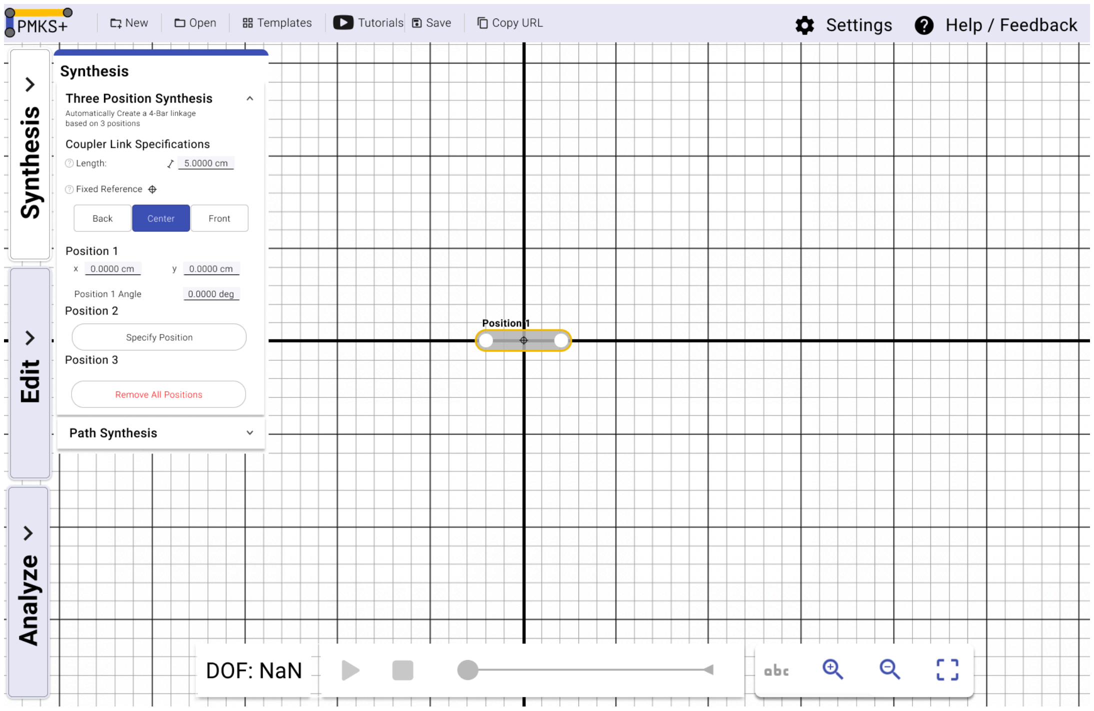
Three Position Synthesis
Automatically Create a 4-Bar linkage based on 3 positions

Coupler Link Specifications
Length: 5.0000 cm
Fixed Reference:
Back | Center | Front

Position 1
x 0.0000 cm y 0.0000 cm
Position 1 Angle 0.0000 deg
Position 2
Specify Position
Position 3
Remove All Positions

Path Synthesis

Center fixed reference



Synthesis

Three Position Synthesis

Automatically Create a 4-Bar linkage based on 3 positions

Coupler Link Specifications

Length: 5.0000 cm

Fixed Reference

Back | Center | Front

Position 1

x 0.0000 cm y 0.0000 cm

Position 1 Angle 0.0000 deg

Position 2

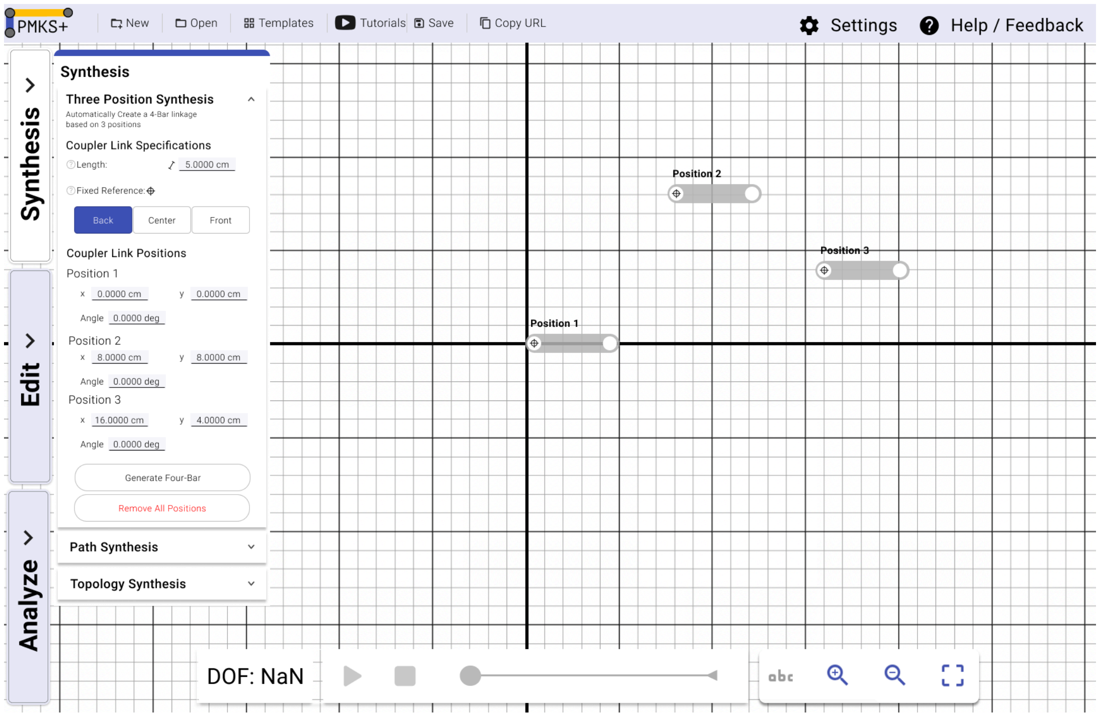
Specify Position

Position 3

Remove All Positions

Path Synthesis

Specify all positions



Synthesis

Three Position Synthesis

Automatically Create a 4-Bar linkage based on 3 positions

Coupler Link Specifications

Length: 5.0000 cm

Fixed Reference:

Back | Center | Front

Coupler Link Positions

Position 1

x 0.0000 cm y 0.0000 cm

Angle 0.0000 deg

Position 2

x 8.0000 cm y 8.0000 cm

Angle 0.0000 deg

Position 3

x 16.0000 cm y 4.0000 cm

Angle 0.0000 deg

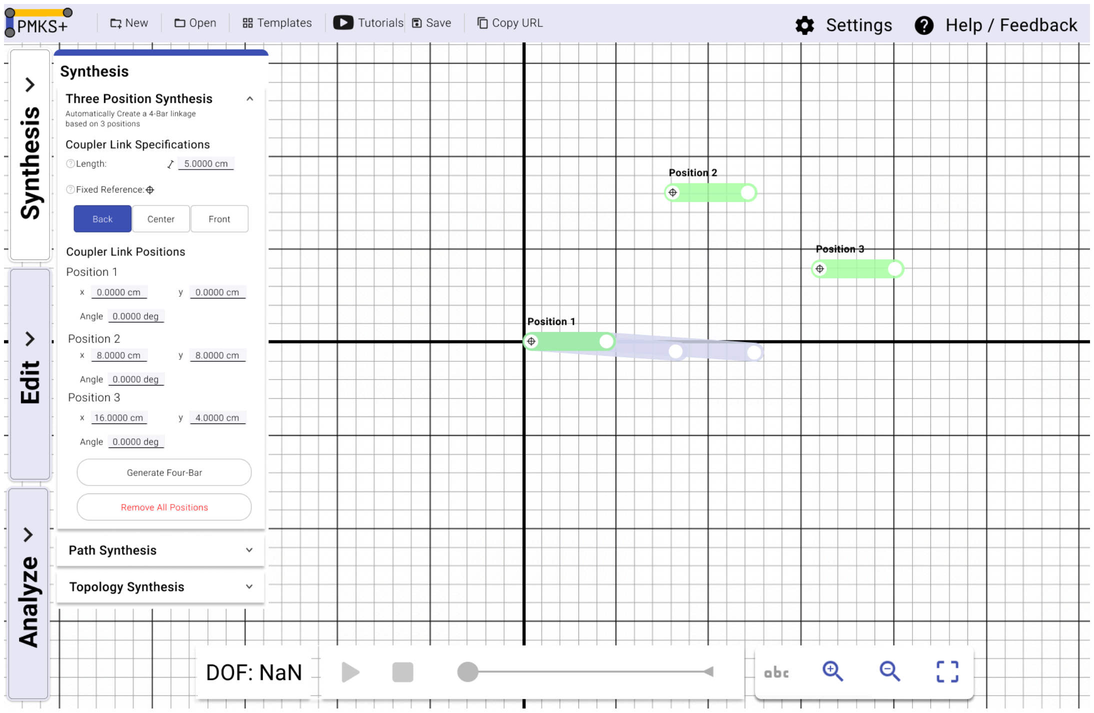
Generate Four-Bar

Remove All Positions

Path Synthesis

Topology Synthesis

Generate mechanism (all positions are valid)



Synthesis

Three Position Synthesis

Automatically Create a 4-Bar linkage based on 3 positions

Coupler Link Specifications

Length: 5.0000 cm

Fixed Reference: [✓ Back] Center Front

Coupler Link Positions

Position 1

x 0.0000 cm y 0.0000 cm

Angle 0.0000 deg

Position 2

x 8.0000 cm y 8.0000 cm

Angle 0.0000 deg

Position 3

x 16.0000 cm y 4.0000 cm

Angle 0.0000 deg

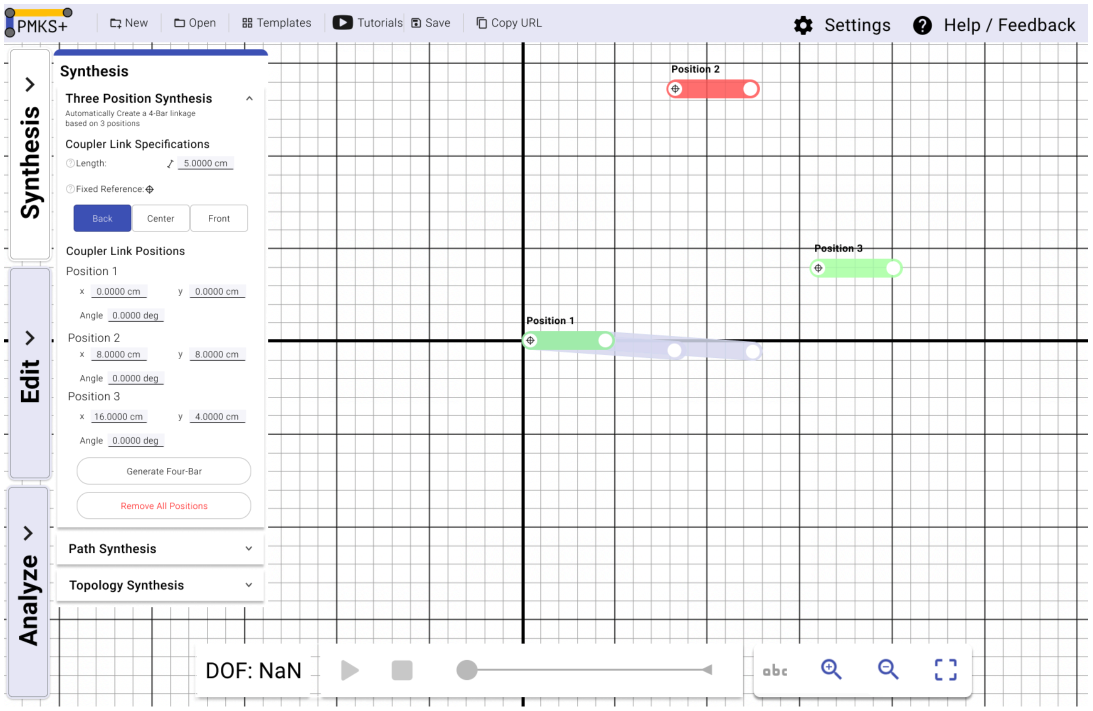
Generate Four-Bar

Remove All Positions

Path Synthesis

Topology Synthesis

Generate mechanism (not all positions valid)



Synthesis

Three Position Synthesis

Automatically Create a 4-Bar linkage based on 3 positions

Coupler Link Specifications

Length: 5.0000 cm

Fixed Reference: [✓ Back] Center Front

Coupler Link Positions

Position 1

x 0.0000 cm y 0.0000 cm

Angle 0.0000 deg

Position 2

x 8.0000 cm y 8.0000 cm

Angle 0.0000 deg

Position 3

x 16.0000 cm y 4.0000 cm

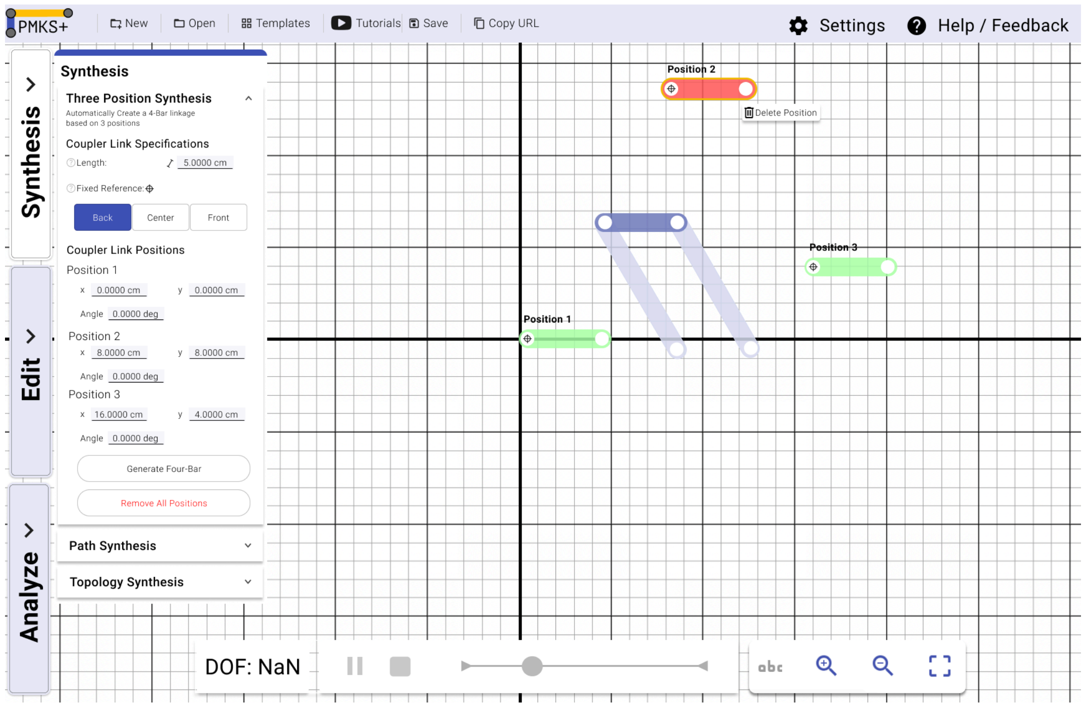
Angle 0.0000 deg

Generate Four-Bar

Remove All Positions

Path Synthesis

Topology Synthesis



Synthesis

Three Position Synthesis

Automatically Create a 4-Bar linkage based on 3 positions

Coupler Link Specifications

Length: 5.0000 cm

Fixed Reference: Back | Center | Front

Coupler Link Positions

Position 1

x 0.0000 cm y 0.0000 cm

Angle 0.0000 deg

Position 2

x 8.0000 cm y 8.0000 cm

Angle 0.0000 deg

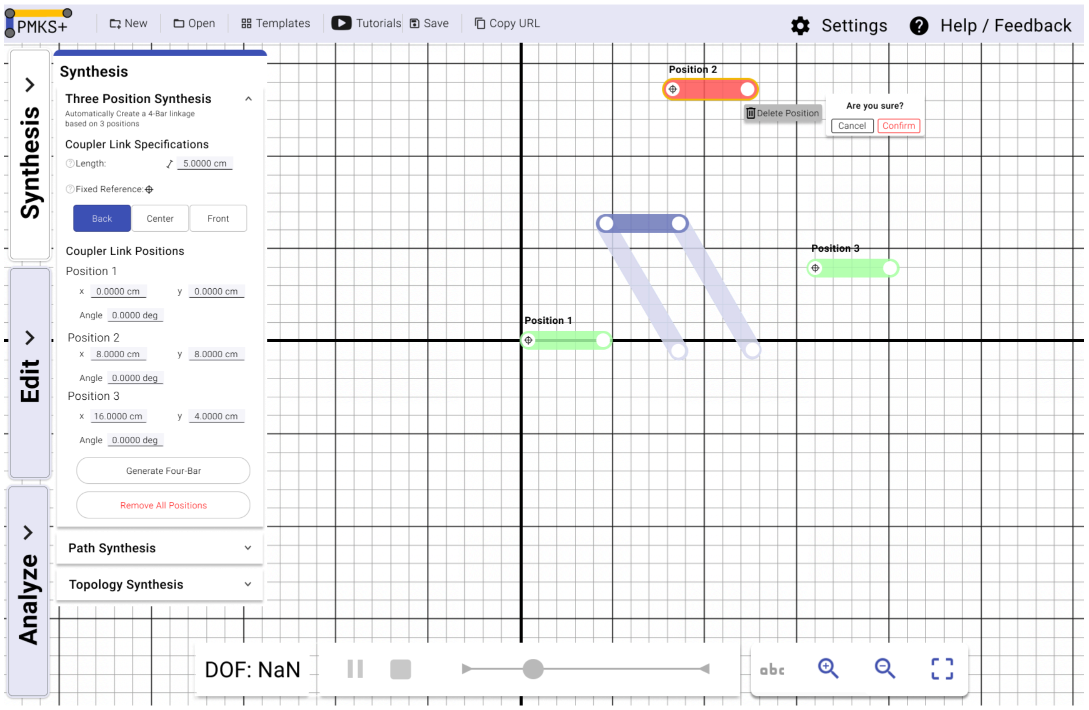
Position 3

x 16.0000 cm y 4.0000 cm

Angle 0.0000 deg

Generate Four-Bar

Remove All Positions



Synthesis

Three Position Synthesis

Automatically Create a 4-Bar linkage based on 3 positions

Coupler Link Specifications

Length: 5.0000 cm

Fixed Reference: Back | Center | Front

Coupler Link Positions

Position 1

x 0.0000 cm y 0.0000 cm

Angle 0.0000 deg

Position 2

x 8.0000 cm y 8.0000 cm

Angle 0.0000 deg

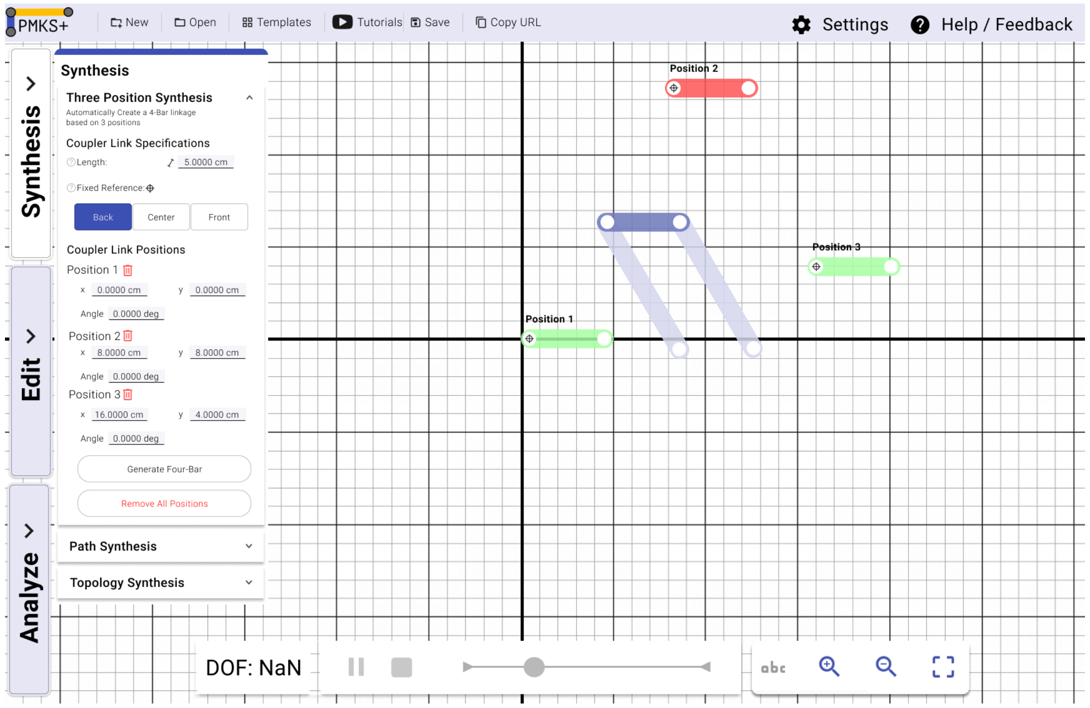
Position 3

x 16.0000 cm y 4.0000 cm

Angle 0.0000 deg

Generate Four-Bar

Remove All Positions


PMKS+ New Open Templates Tutorials Save Copy URL Settings Help / Feedback

Synthesis

Three Position Synthesis

Automatically Create a 4-Bar linkage based on 3 positions

Coupler Link Specifications

Length: 5.0000 cm

Fixed Reference:

Back Center Front

Coupler Link Positions

Position 1

x 0.0000 cm y 0.0000 cm

Angle 0.0000 deg

Position 2

x 8.0000 cm y 8.0000 cm

Angle 0.0000 deg

Position 3

x 16.0000 cm y 4.0000 cm

Angle 0.0000 deg

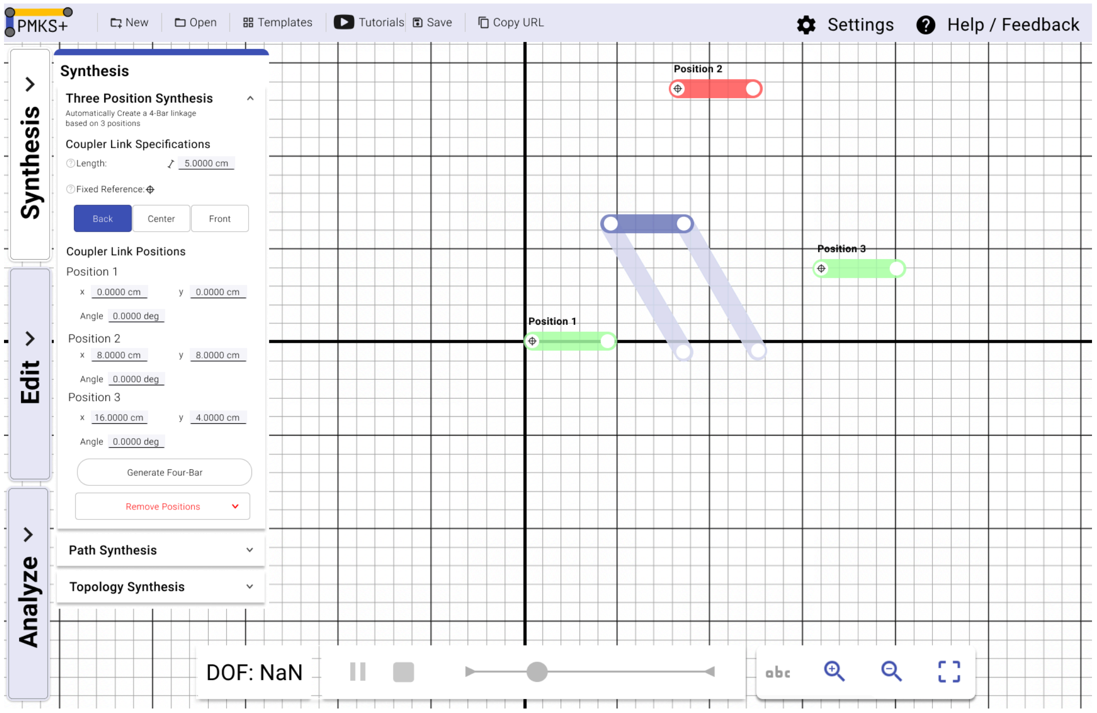
Generate Four-Bar

Remove All Positions

Path Synthesis

Topology Synthesis

DOF: NaN


PMKS+ New Open Templates Tutorials Save Copy URL Settings Help / Feedback

Synthesis

Three Position Synthesis

Automatically Create a 4-Bar linkage based on 3 positions

Coupler Link Specifications

Length: 5.0000 cm

Fixed Reference:

Back Center Front

Coupler Link Positions

Position 1

x 0.0000 cm y 0.0000 cm

Angle 0.0000 deg

Position 2

x 8.0000 cm y 8.0000 cm

Angle 0.0000 deg

Position 3

x 16.0000 cm y 4.0000 cm

Angle 0.0000 deg

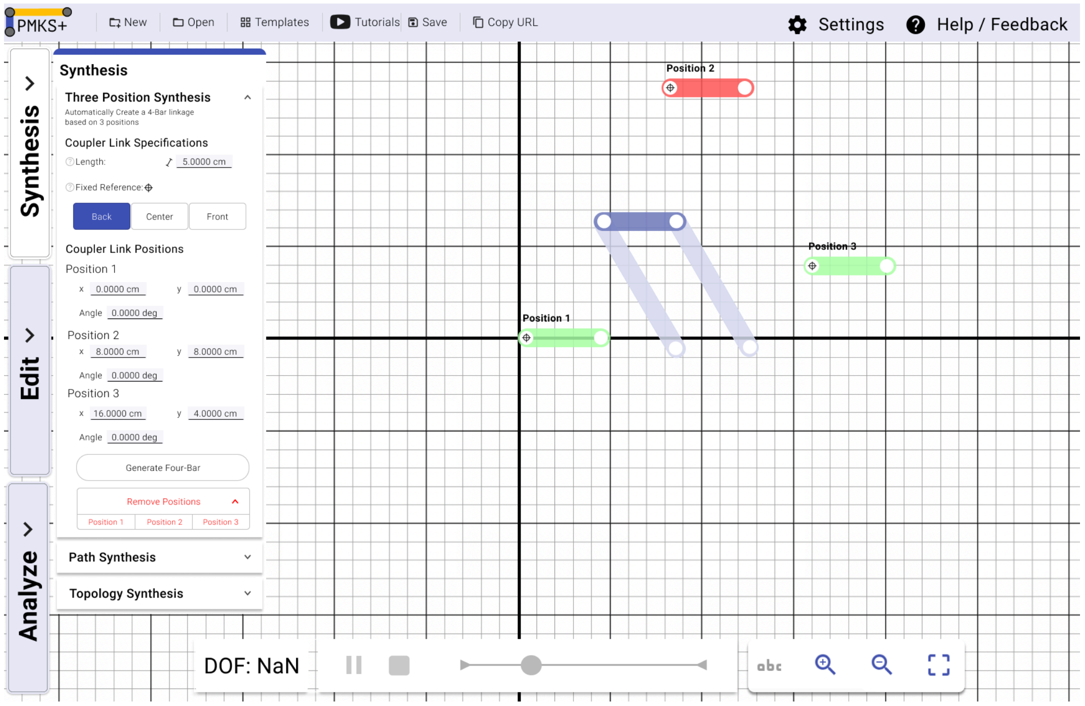
Generate Four-Bar

Remove Positions

Path Synthesis

Topology Synthesis

DOF: NaN



Synthesis

Coupler Link Specifications

Length: 5.0000 cm

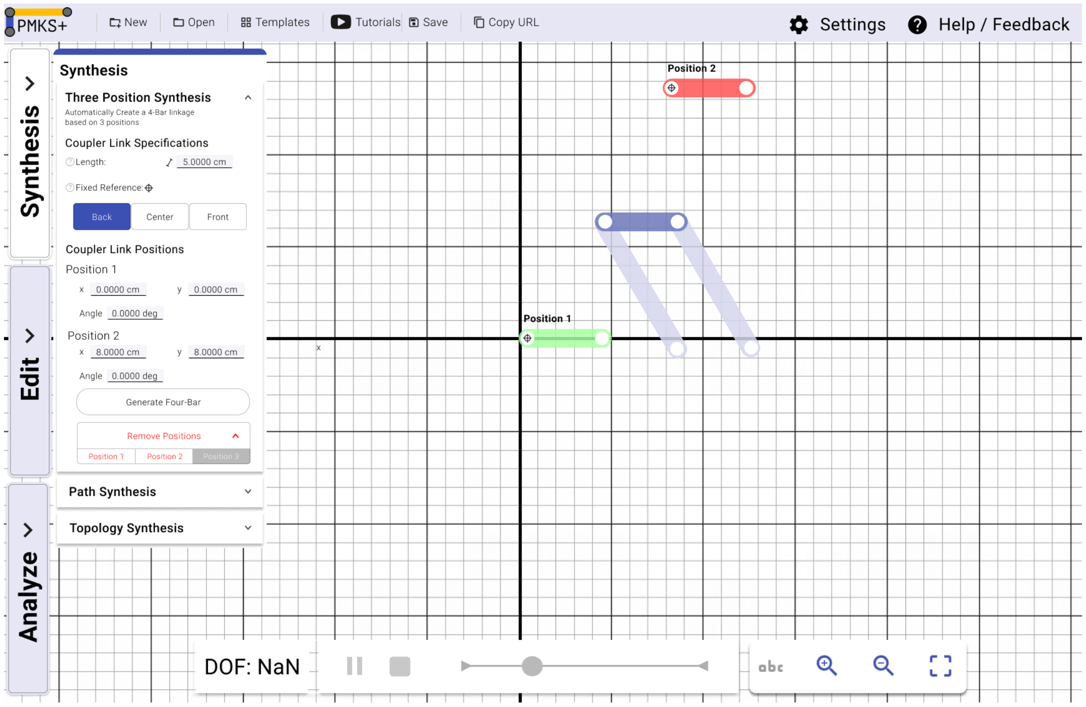
Fixed Reference: Back

Coupler Link Positions

Position 1: x 0.0000 cm, y 0.0000 cm, Angle 0.0000 deg

Position 2: x 8.0000 cm, y 8.0000 cm, Angle 0.0000 deg

Position 3: x 16.0000 cm, y 4.0000 cm, Angle 0.0000 deg



Synthesis

Coupler Link Specifications

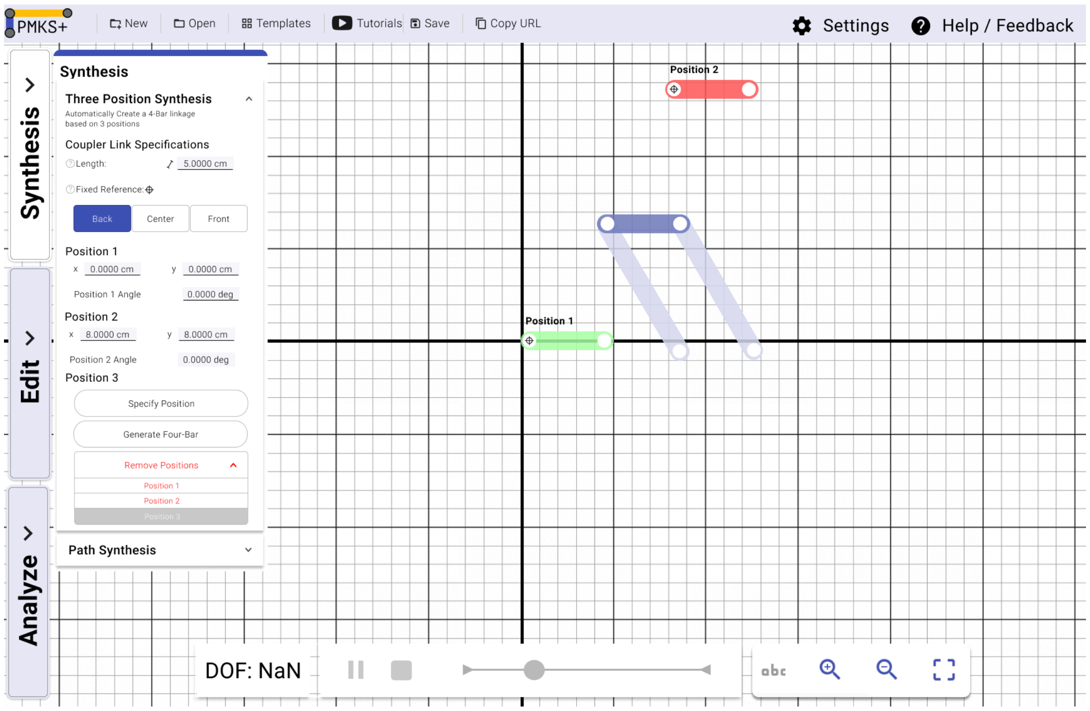
Length: 5.0000 cm

Fixed Reference: Back

Coupler Link Positions

Position 1: x 0.0000 cm, y 0.0000 cm, Angle 0.0000 deg

Position 2: x 8.0000 cm, y 8.0000 cm, Angle 0.0000 deg



Synthesis

Three Position Synthesis

Coupler Link Specifications

Length: 5.0000 cm

Fixed Reference: [✓ Back] Center Front

Position 1

x 0.0000 cm y 0.0000 cm

Position 1 Angle 0.0000 deg

Position 2

x 8.0000 cm y 8.0000 cm

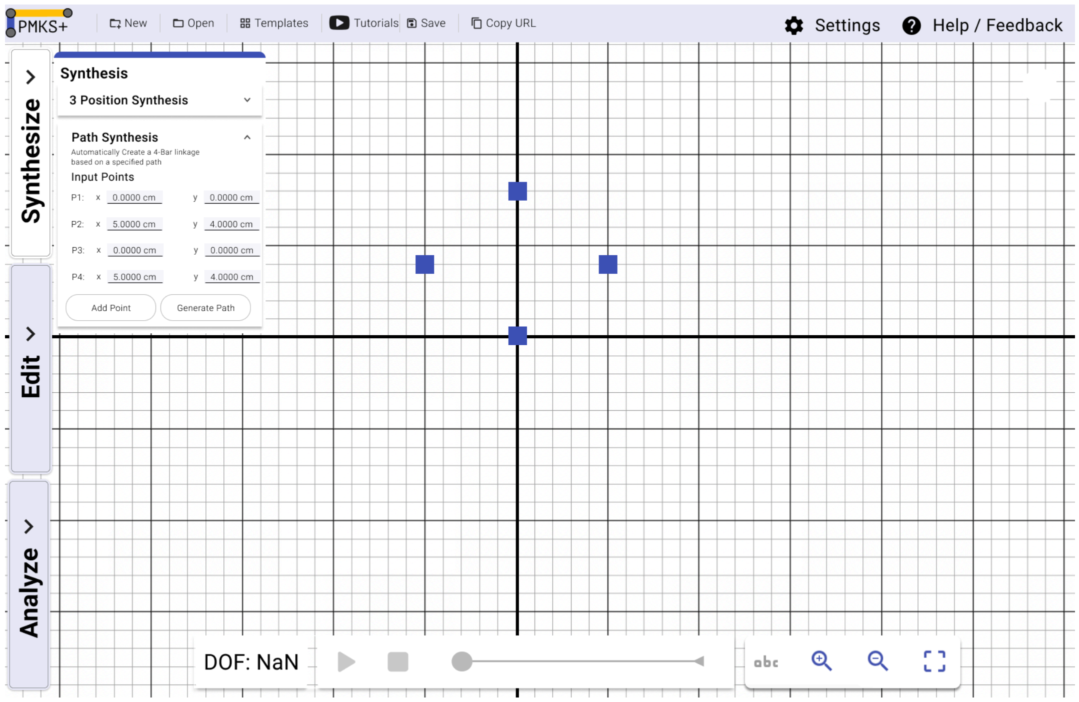
Position 2 Angle 0.0000 deg

Position 3

DOF: NaN

Appendix F: Path Synthesis Mockups

User inputs positions by inputted points



Synthesis

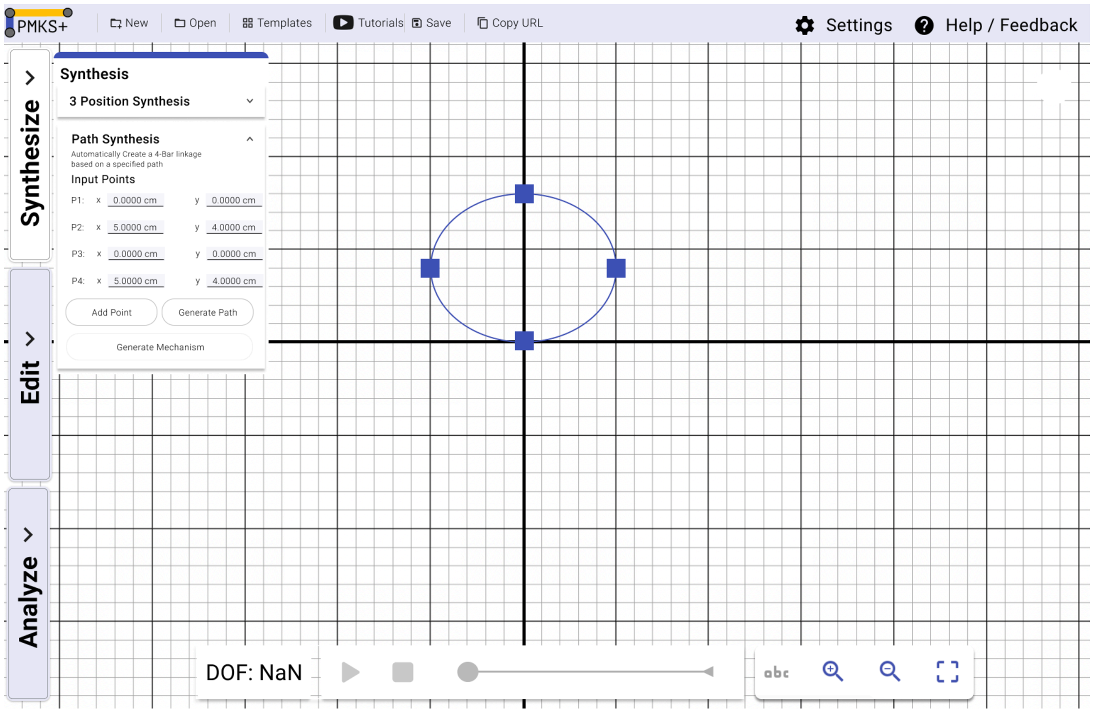
3 Position Synthesis

Path Synthesis

Automatically Create a 4-Bar linkage based on a specified path

Input Points

P1:	x	0.0000 cm	y	0.0000 cm
P2:	x	5.0000 cm	y	4.0000 cm
P3:	x	0.0000 cm	y	0.0000 cm
P4:	x	5.0000 cm	y	4.0000 cm



Synthesis

3 Position Synthesis

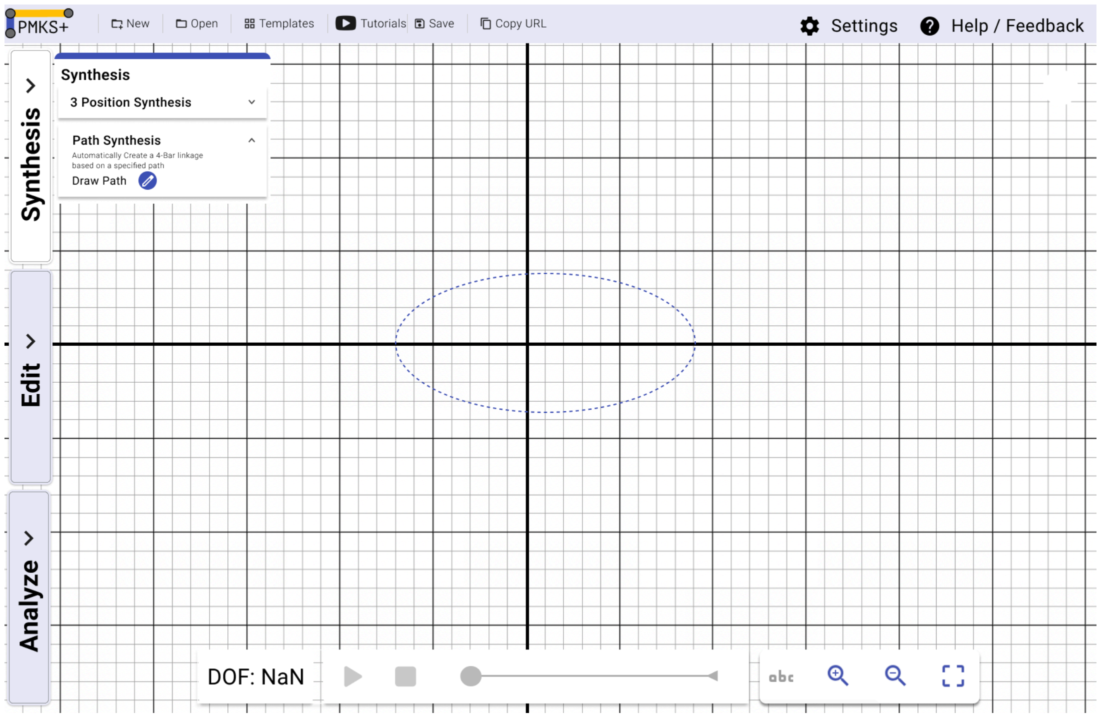
Path Synthesis

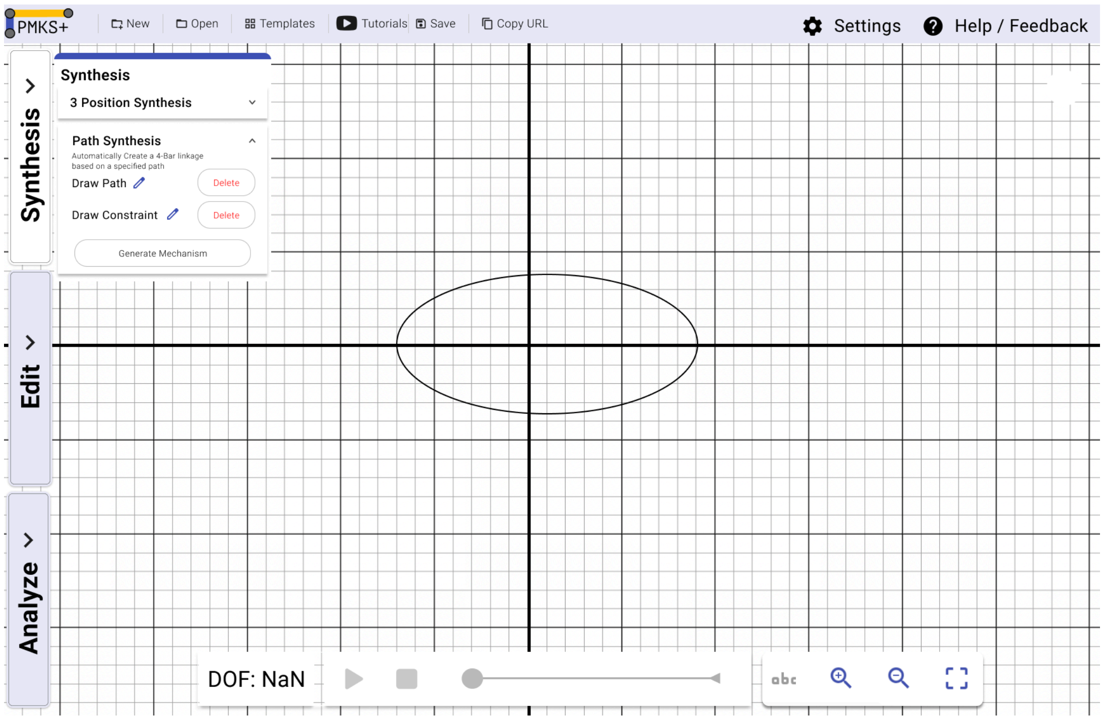
Automatically Create a 4-Bar linkage based on a specified path

Input Points

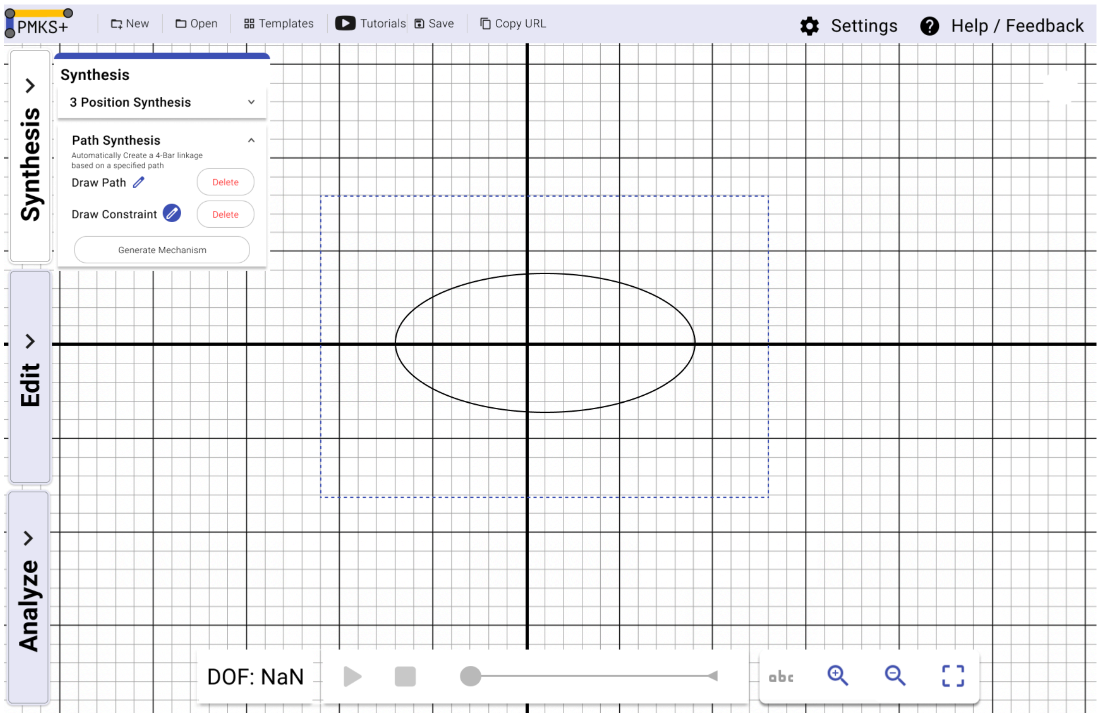
P1:	x	0.0000 cm	y	0.0000 cm
P2:	x	5.0000 cm	y	4.0000 cm
P3:	x	0.0000 cm	y	0.0000 cm
P4:	x	5.0000 cm	y	4.0000 cm

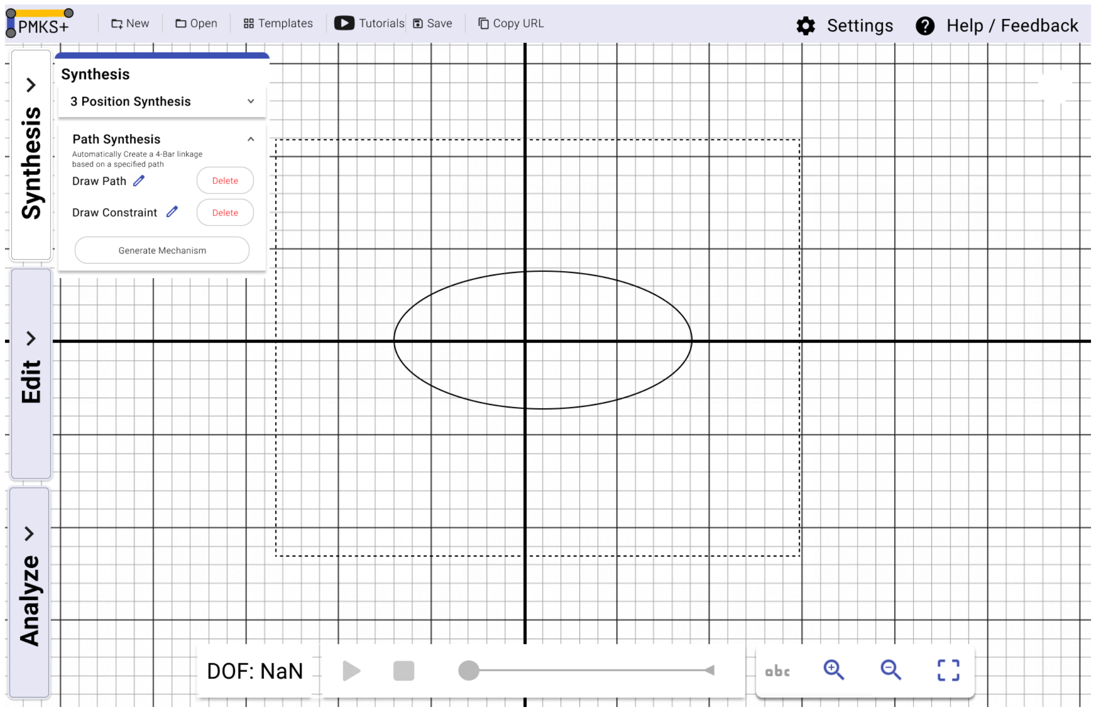
User draws path with mouse



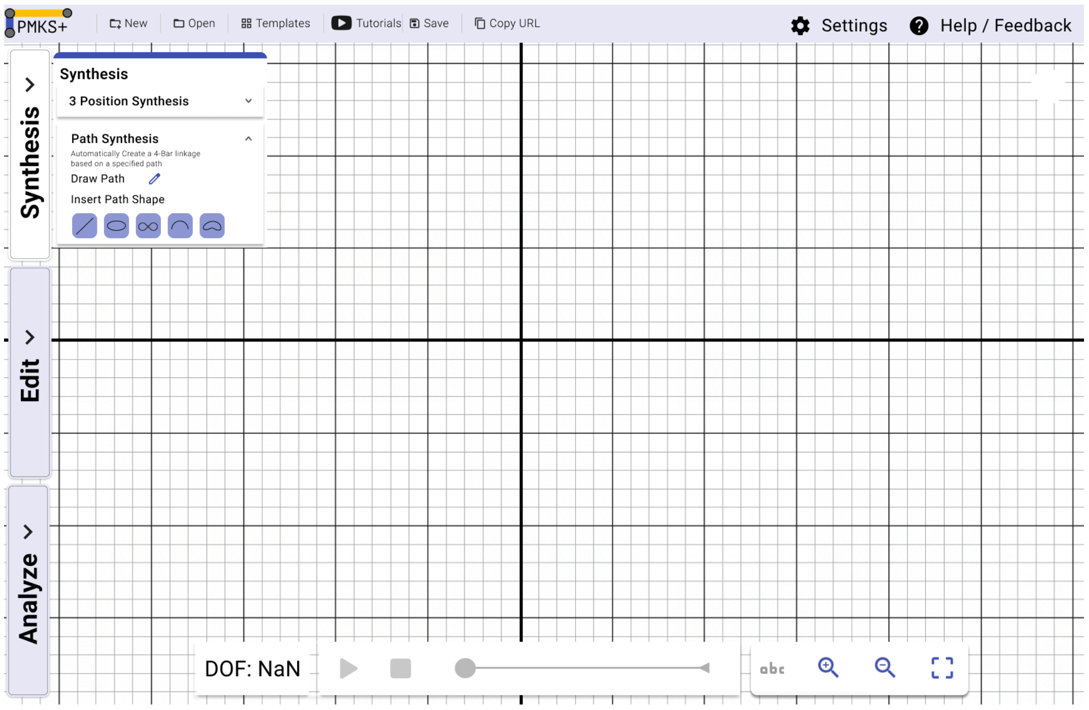


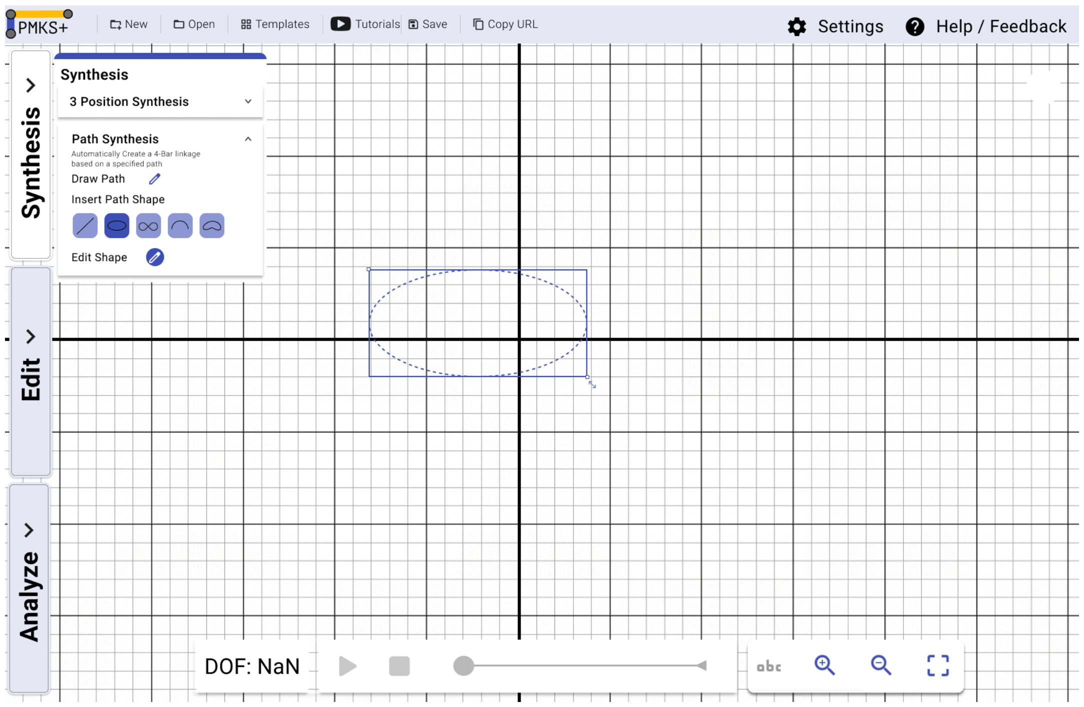
User draws constraint (box) to keep mechanism contained inside

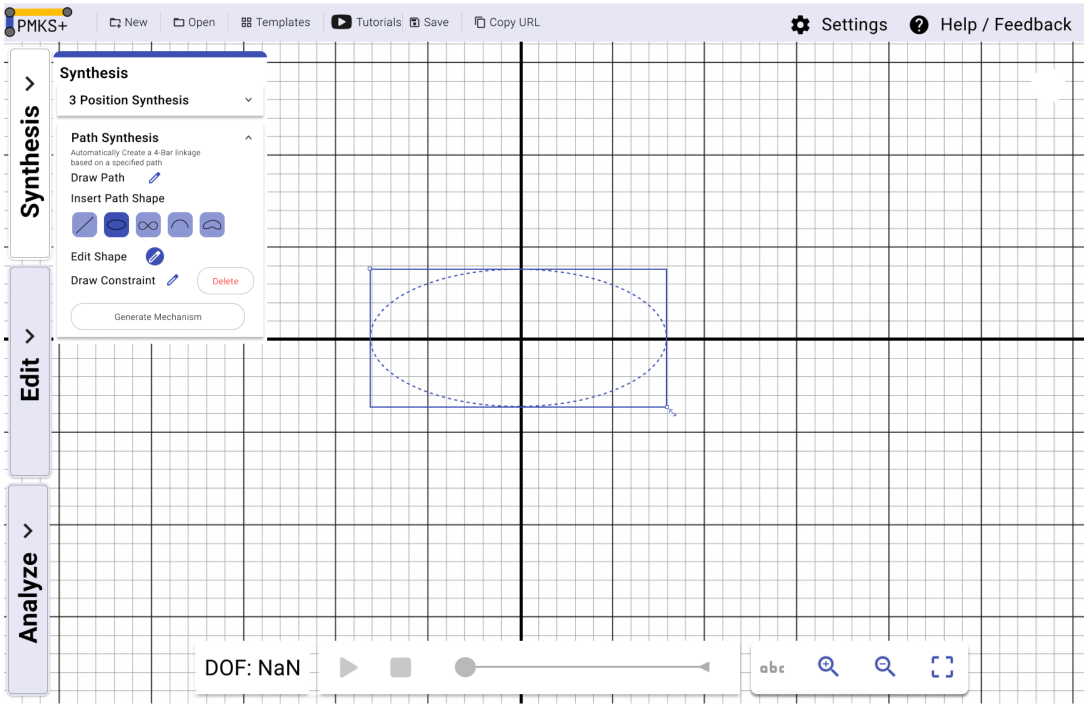




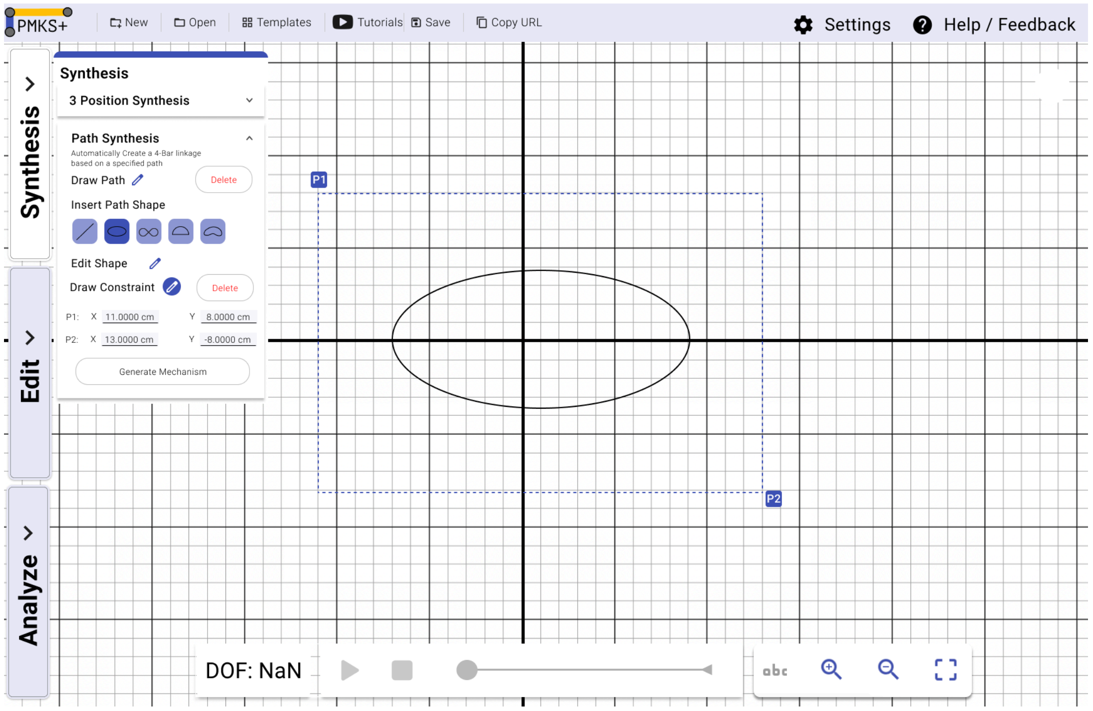
User can choose a shape to make the path

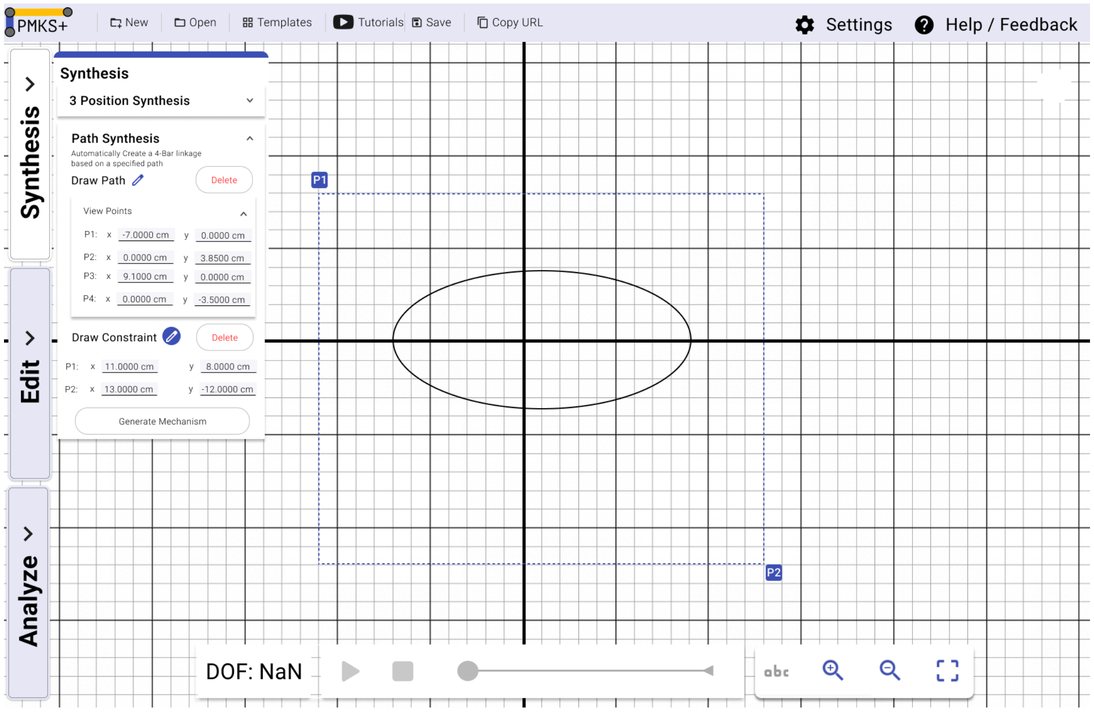

PMKS+
New
Open
Templates
Tutorials
Save
Copy URL
Settings
Help / Feedback
Synthesis
3 Position Synthesis
Path Synthesis
Automatically Create a 4-Bar linkage based on a specified path
Draw Path
Insert Path Shape
Synthesis
Edit
Analyze
DOF: NaN
abc


PMKS+
New
Open
Templates
Tutorials
Save
Copy URL
Settings
Help / Feedback
Synthesis
3 Position Synthesis
Path Synthesis
Automatically Create a 4-Bar linkage based on a specified path
Draw Path
Insert Path Shape
Edit Shape
Synthesis
Edit
Analyze
DOF: NaN
abc



User selects the constraint to edit the points by dragging P1 or P2





Synthesis

3 Position Synthesis

Path Synthesis

Automatically Create a 4-Bar linkage based on a specified path

Draw Path — Delete

View Points

P1:	x	-7.0000 cm	y	0.0000 cm
P2:	x	0.0000 cm	y	3.8500 cm
P3:	x	9.1000 cm	y	0.0000 cm
P4:	x	0.0000 cm	y	-3.5000 cm

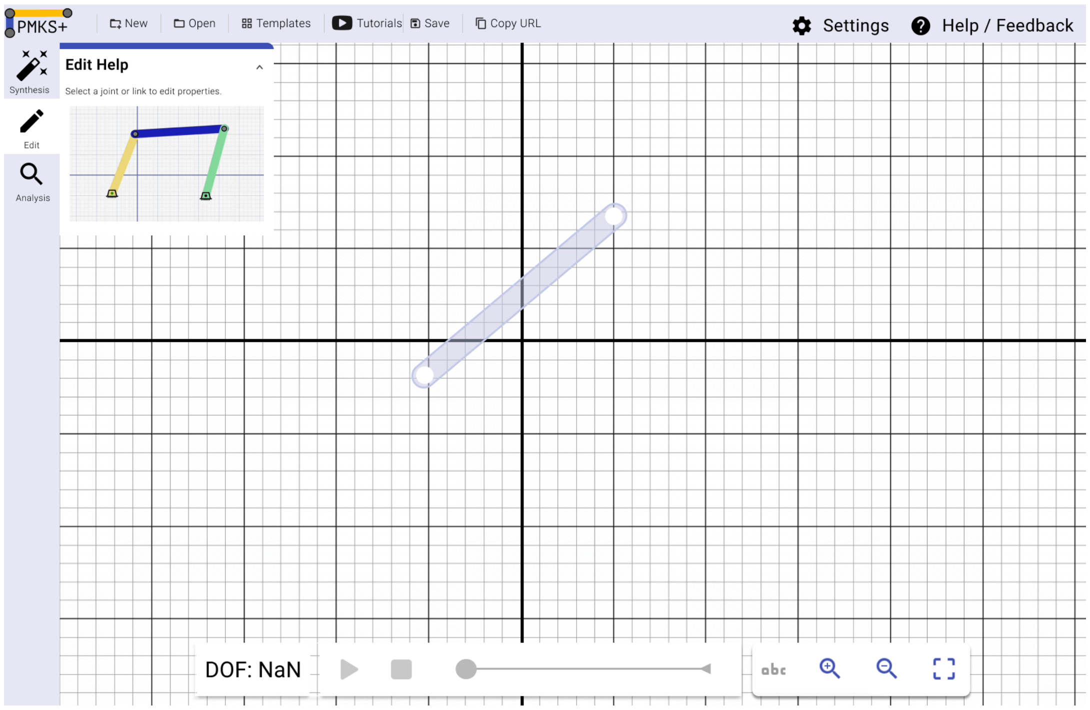
Draw Constraint — Delete

P1:	x	11.0000 cm	y	8.0000 cm
P2:	x	13.0000 cm	y	-12.0000 cm

Generate Mechanism

Appendix G: Link Edit Panel Mockups

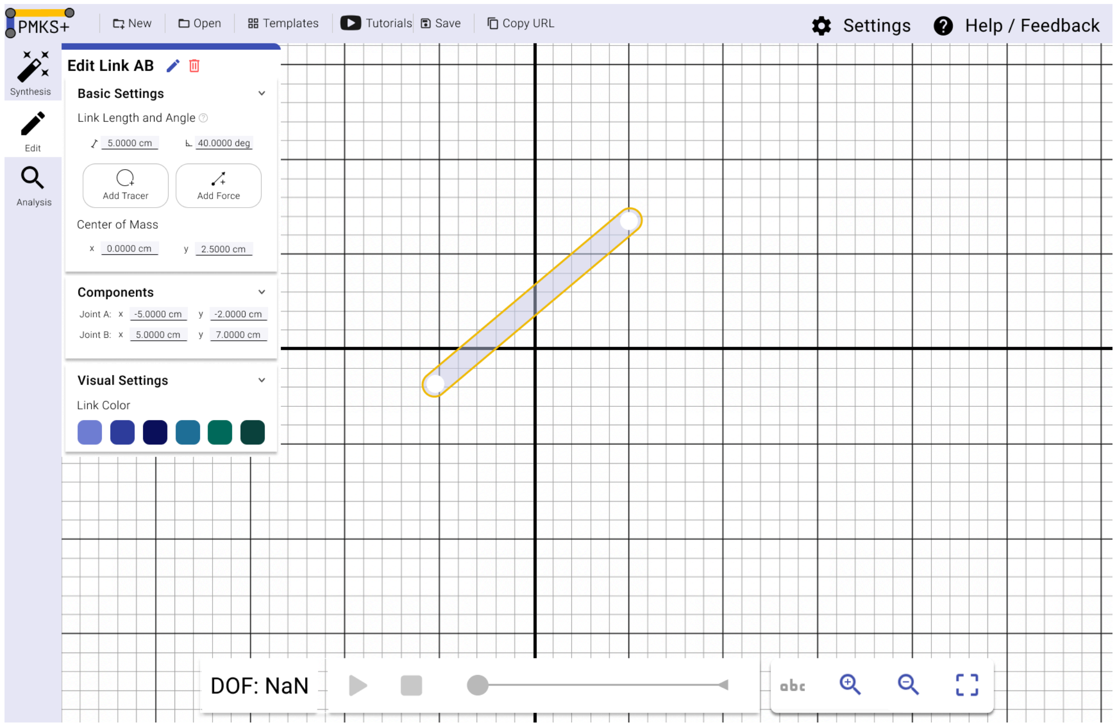
Nothing selected



Edit Help

Select a joint or link to edit properties.

Select Link



Edit Link AB

Basic Settings

Link Length and Angle

5.0000 cm   40.0000 deg

Center of Mass

x 0.0000 cm   y 2.5000 cm

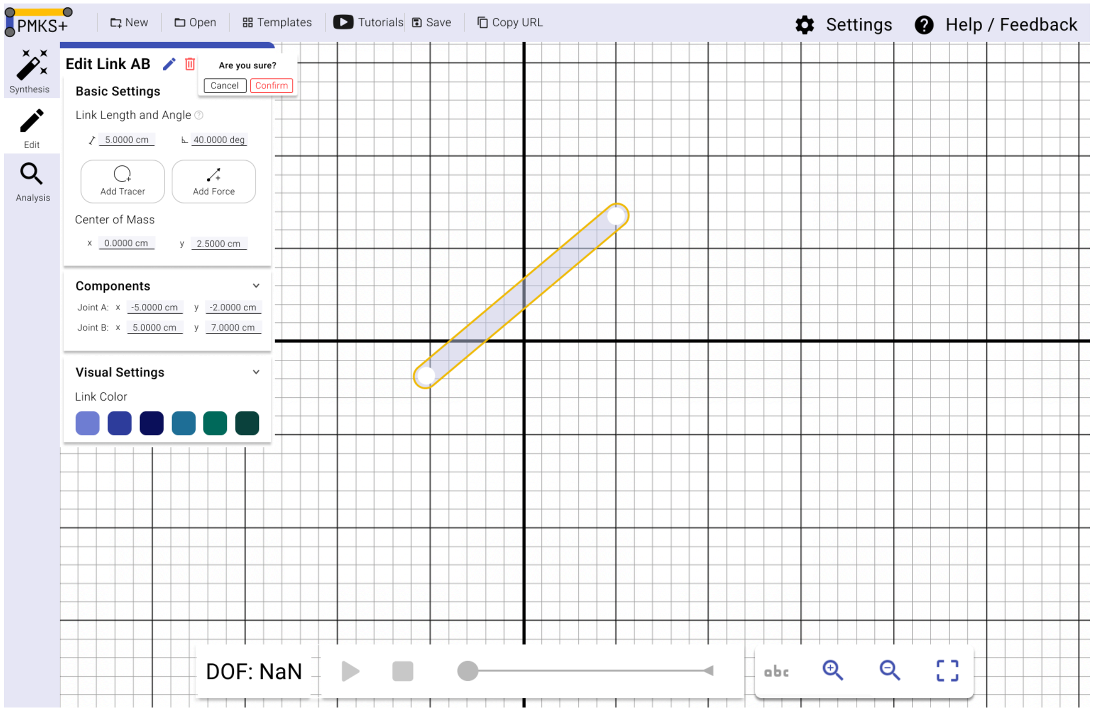
Components

Joint A: x -5.0000 cm   y -2.0000 cm

Joint B: x 5.0000 cm   y 7.0000 cm

Visual Settings

Link Color



Edit Link AB

Basic Settings

Link Length and Angle

5.0000 cm 40.0000 deg

Center of Mass

x 0.0000 cm y 2.5000 cm

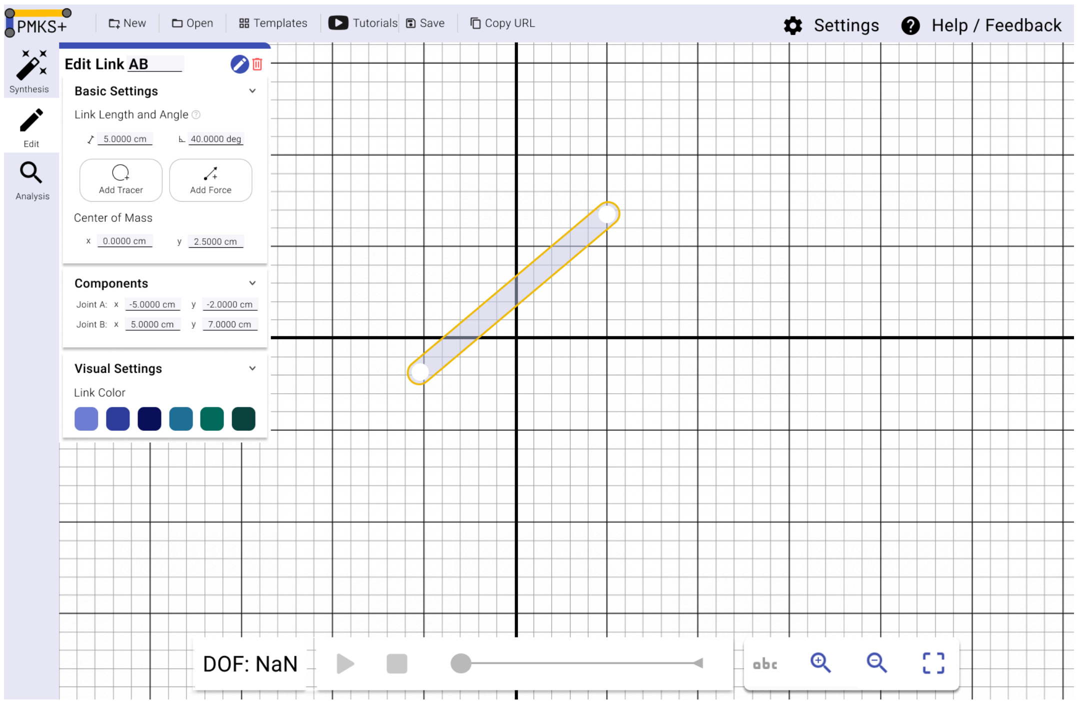
Components

Joint A: x -5.0000 cm y -2.0000 cm

Joint B: x 5.0000 cm y 7.0000 cm

Visual Settings

Link Color



Edit Link AB

Basic Settings

Link Length and Angle

5.0000 cm 40.0000 deg

Center of Mass

x 0.0000 cm y 2.5000 cm

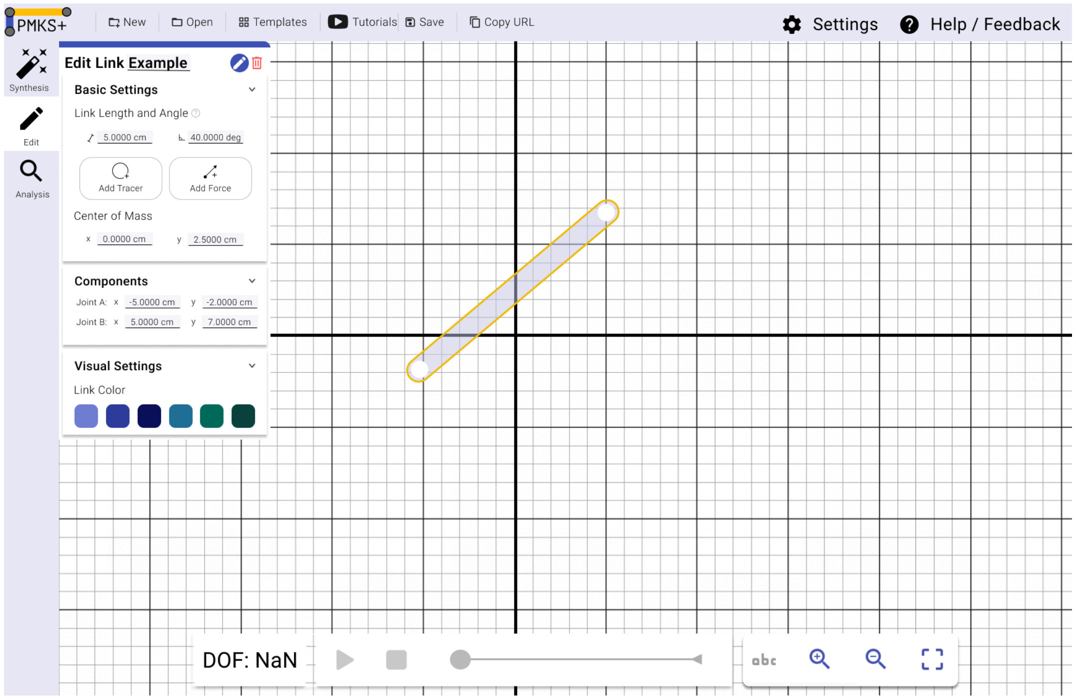
Components

Joint A: x -5.0000 cm y -2.0000 cm

Joint B: x 5.0000 cm y 7.0000 cm

Visual Settings

Link Color


PMKS+ New Open Templates Tutorials Save Copy URL Settings Help / Feedback

Synthesis Edit Analysis

Edit Link Example

Basic Settings

Link Length and Angle

5.0000 cm 40.0000 deg

Add Tracer Add Force

Center of Mass

x 0.0000 cm y 2.5000 cm

Components

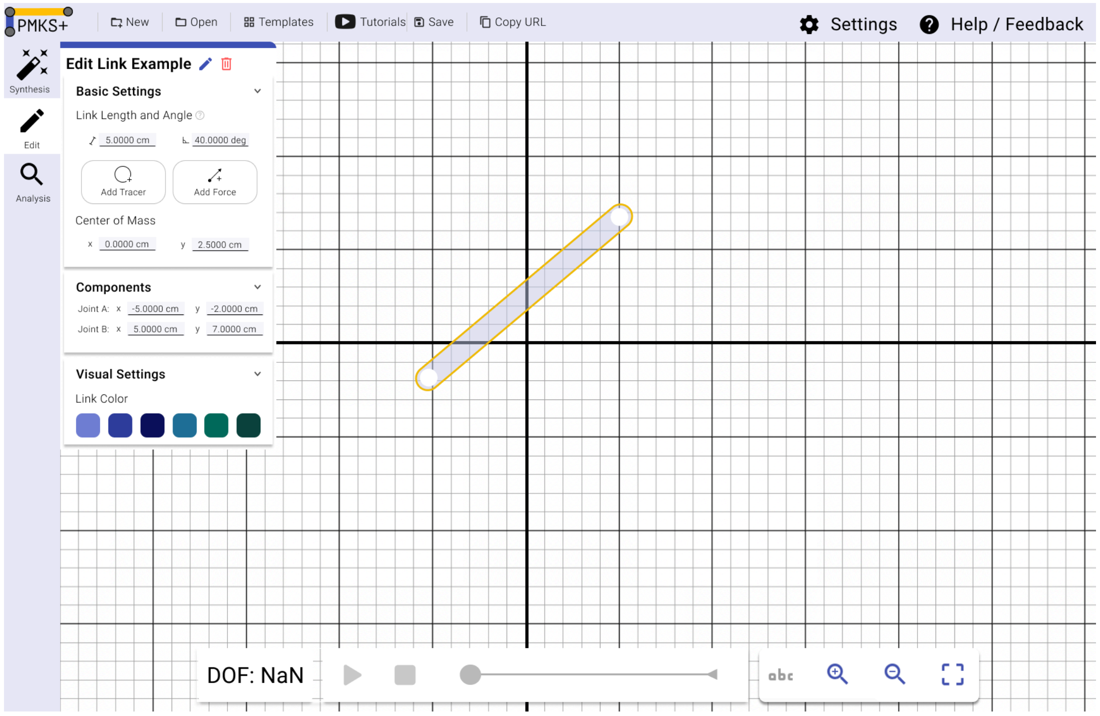
Joint A: x -5.0000 cm y -2.0000 cm

Joint B: x 5.0000 cm y 7.0000 cm

Visual Settings

Link Color

DOF: NaN


PMKS+ New Open Templates Tutorials Save Copy URL Settings Help / Feedback

Synthesis Edit Analysis

Edit Link Example

Basic Settings

Link Length and Angle

5.0000 cm 40.0000 deg

Add Tracer Add Force

Center of Mass

x 0.0000 cm y 2.5000 cm

Components

Joint A: x -5.0000 cm y -2.0000 cm

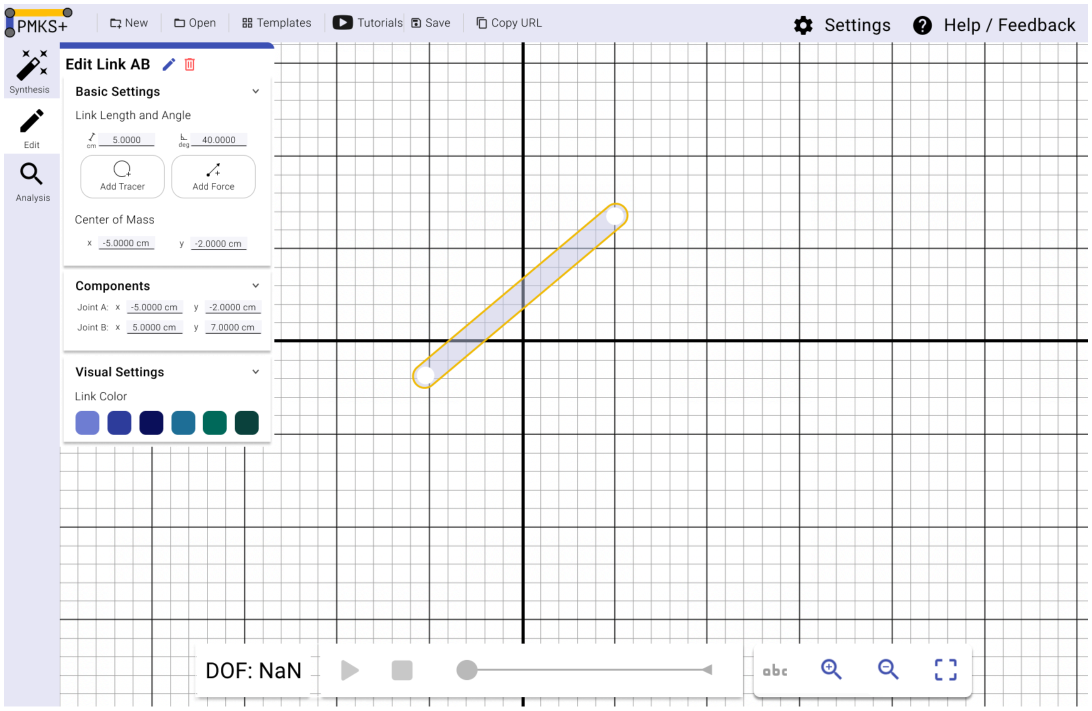
Joint B: x 5.0000 cm y 7.0000 cm

Visual Settings

Link Color

DOF: NaN

Potentially place units underneath labels



Edit Link AB

Basic Settings

Link Length and Angle

cm 5.0000 deg 40.0000

Center of Mass

x -5.0000 cm y -2.0000 cm

Components

Joint A: x -5.0000 cm y -2.0000 cm

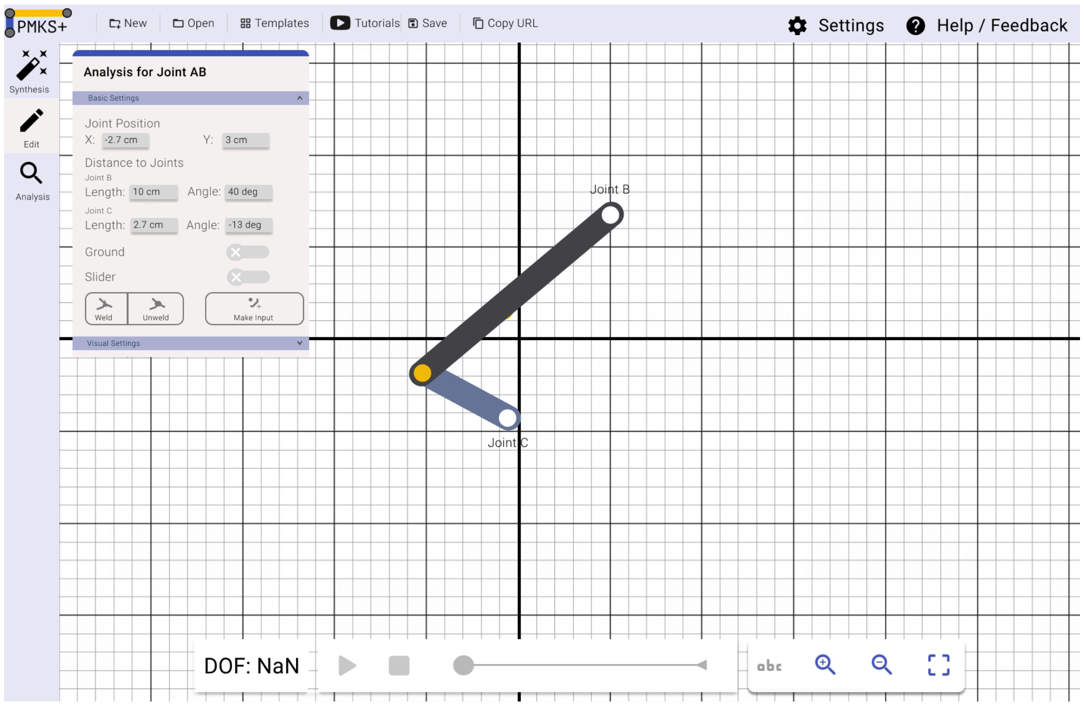
Joint B: x 5.0000 cm y 7.0000 cm

Visual Settings

Link Color

Appendix H: Joint Edit Panel Mockups

Select a joint to view basic settings (Heading is incorrect)



Analysis for Joint AB

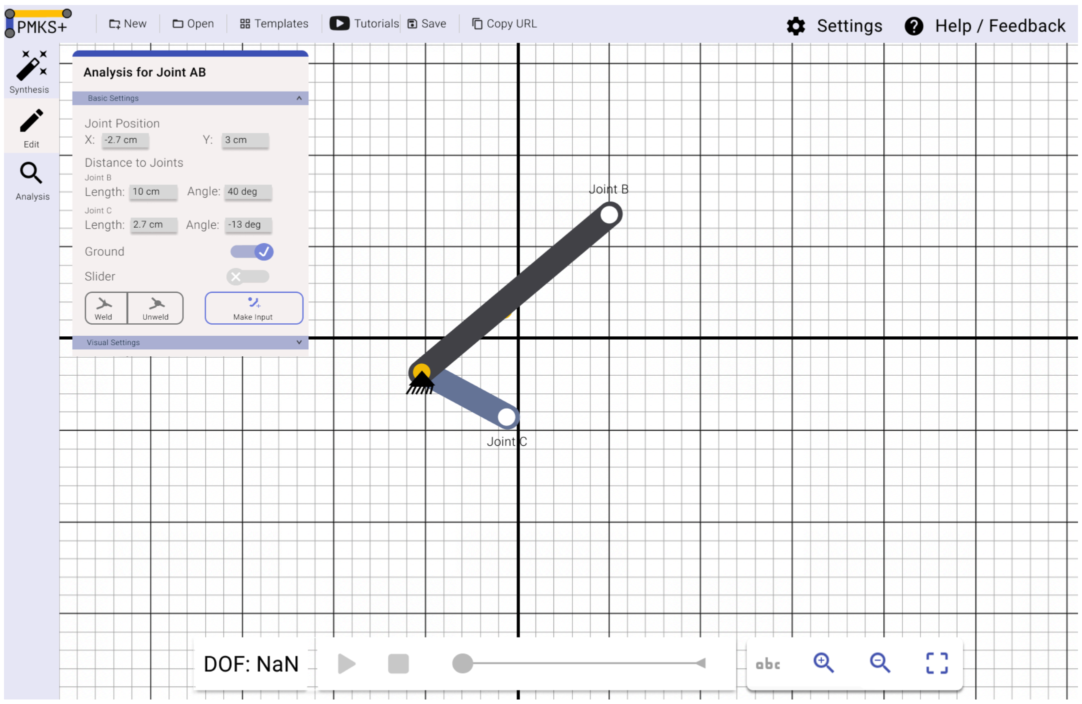
Joint Position
X: -2.7 cm Y: 3 cm

Distance to Joints
Joint B
Length: 10 cm Angle: 40 deg
Joint C
Length: 2.7 cm Angle: -13 deg

Ground
Slider

DOF: NaN

Ground enabled for joint, make input button now available



Analysis for Joint AB

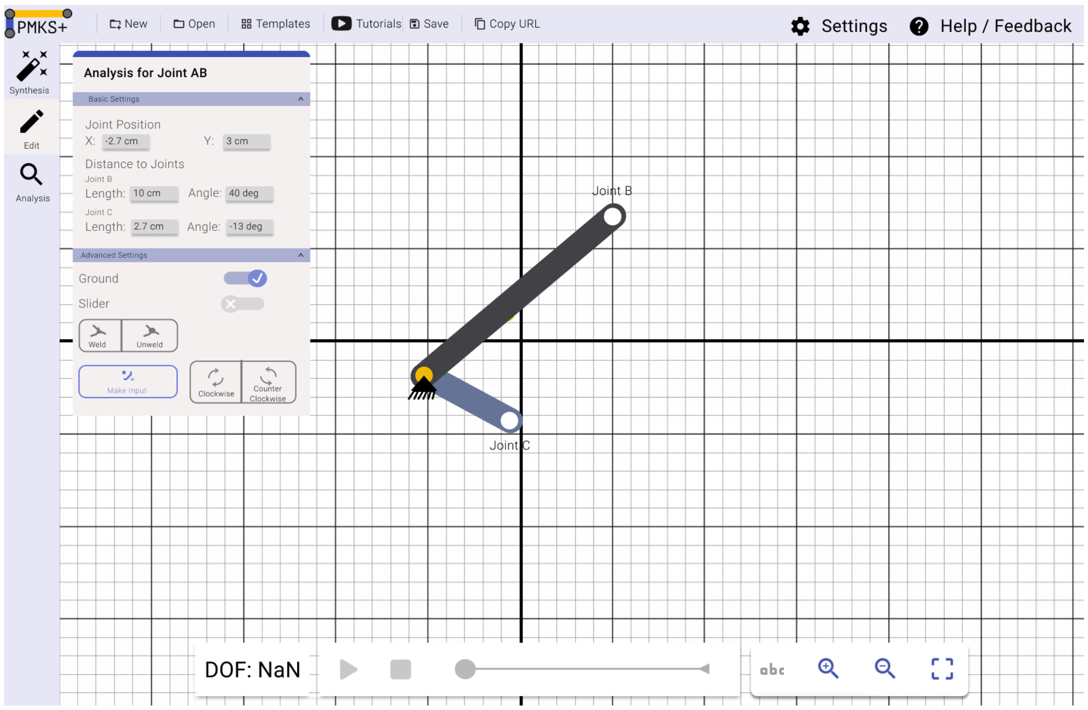
Joint Position
X: -2.7 cm Y: 3 cm

Distance to Joints
Joint B
Length: 10 cm Angle: 40 deg
Joint C
Length: 2.7 cm Angle: -13 deg

Ground ✓
Slider

DOF: NaN

Joint is made an input, direction buttons appear once joint is made to be an input



Analysis for Joint AB

Basic Settings

Joint Position
X: -2.7 cm Y: 3 cm

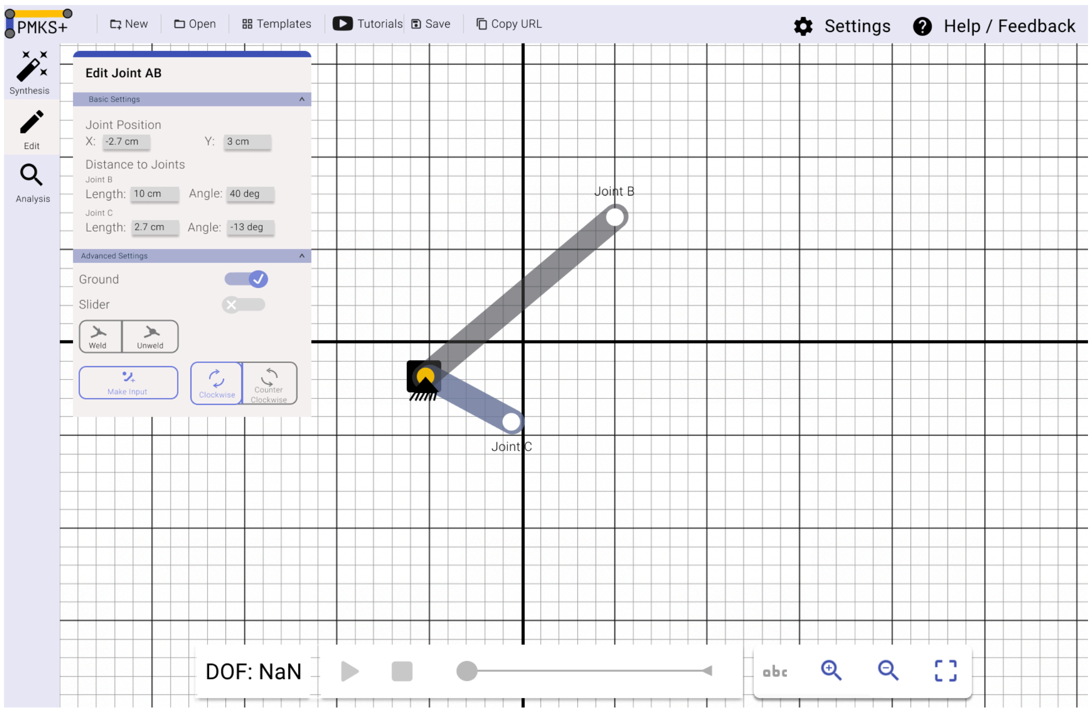
Distance to Joints
Joint B
Length: 10 cm Angle: 40 deg
Joint C
Length: 2.7 cm Angle: -13 deg

Advanced Settings

Ground ✓
Slider ✗

DOF: NaN

Select input direction



Edit Joint AB

Basic Settings

Joint Position
X: -2.7 cm Y: 3 cm

Distance to Joints
Joint B
Length: 10 cm Angle: 40 deg
Joint C
Length: 2.7 cm Angle: -13 deg

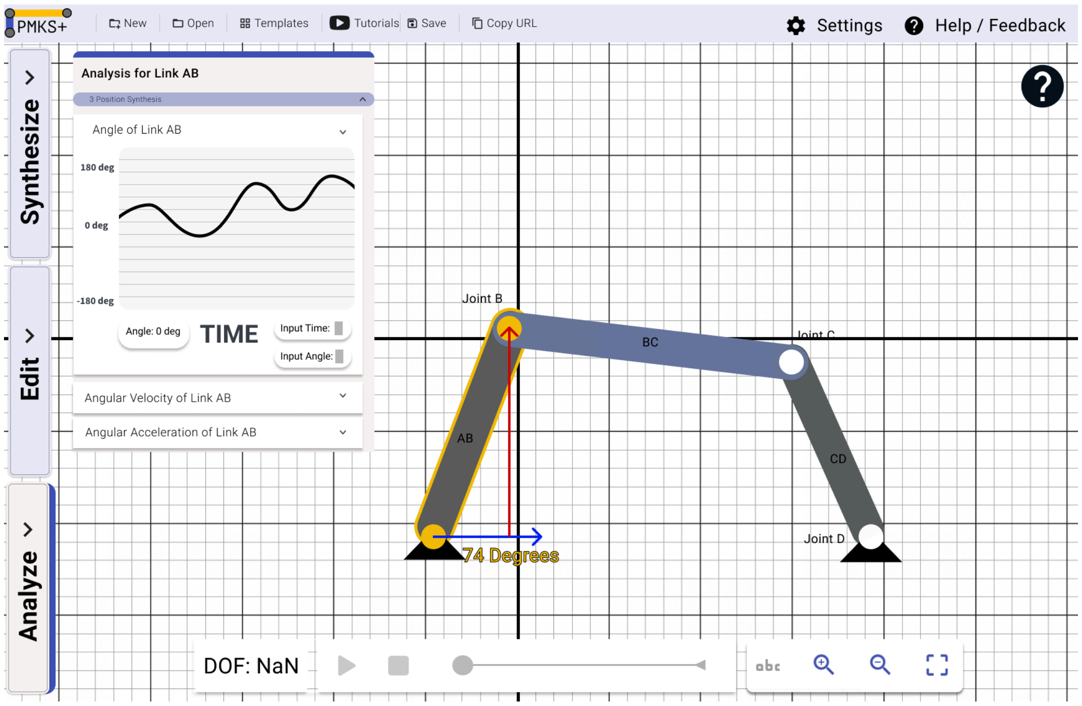
Advanced Settings

Ground ✓
Slider ✗

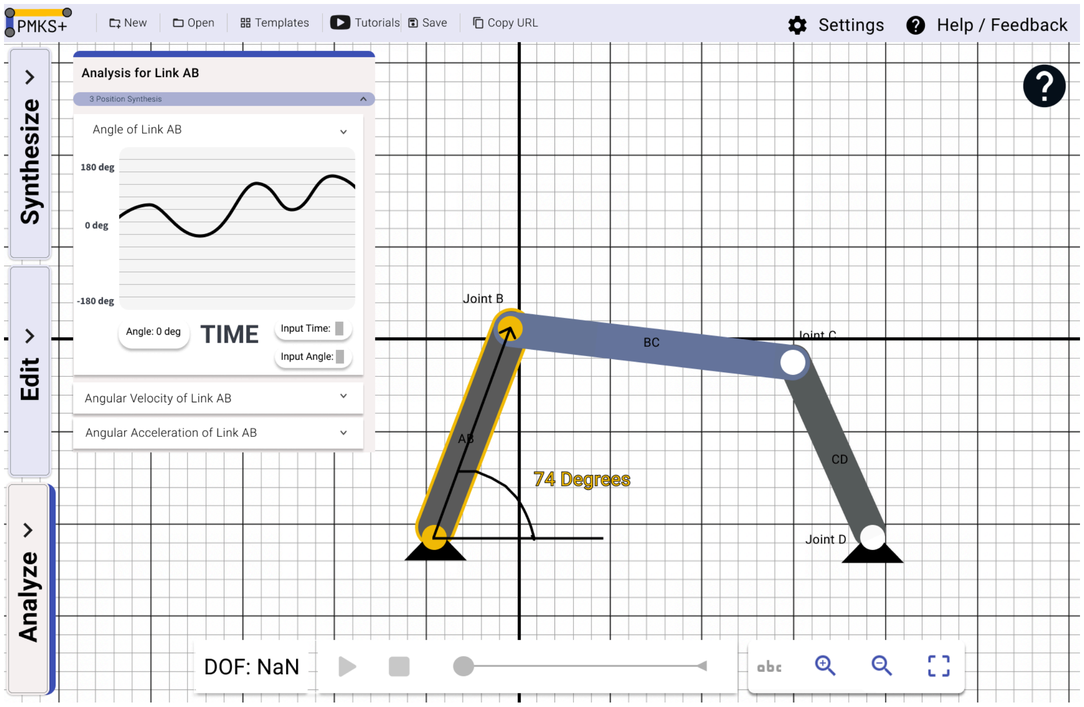
DOF: NaN

Appendix I: Analysis Panel Mockups

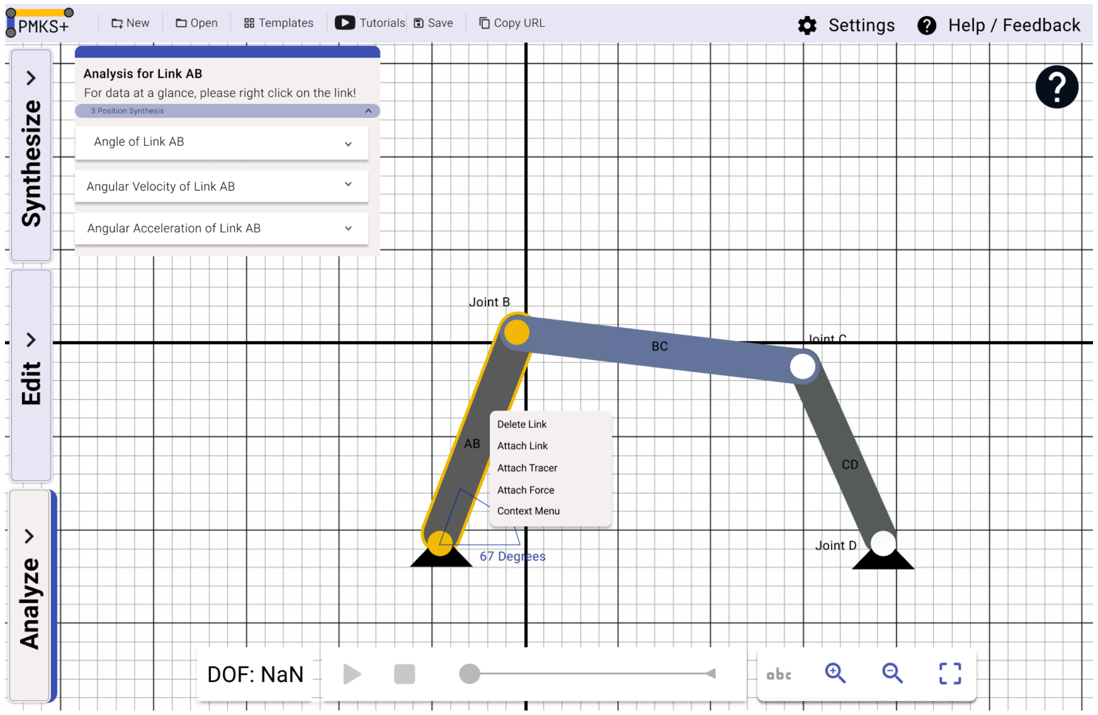
Click on link to display panel and angle of link



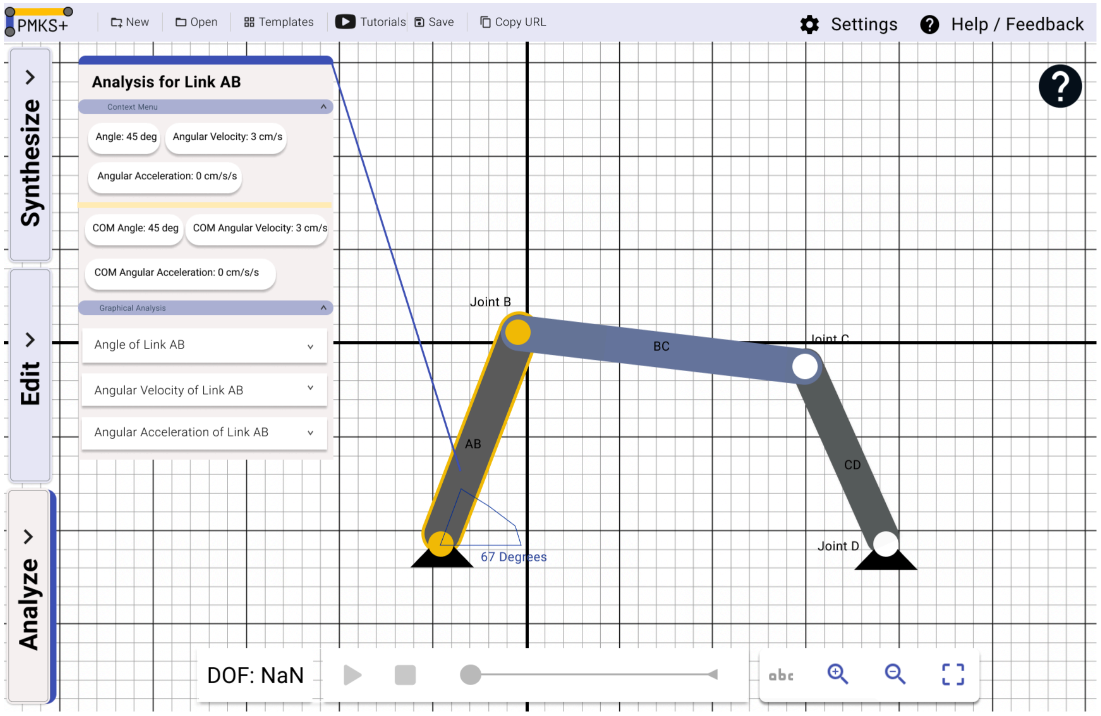
Another method to view angle



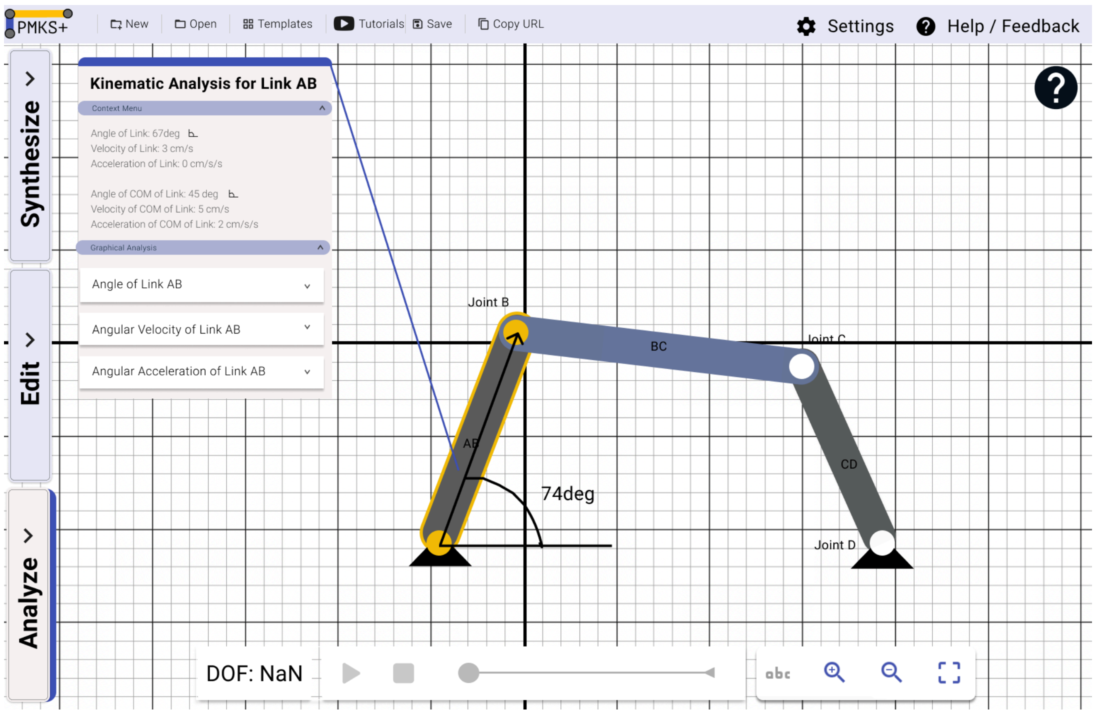
Click on link for context menu



Select context menu to show data summary



Another visual for data summary


Kinematic Analysis for Link AB

Context Menu

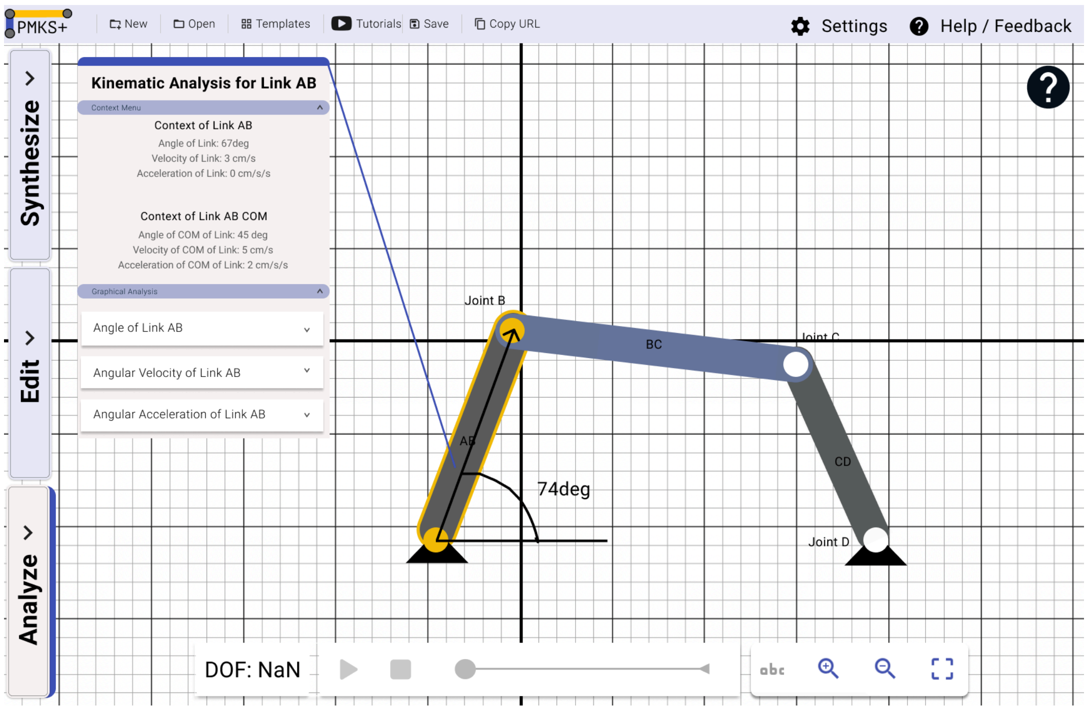
Angle of Link: 67deg
Velocity of Link: 3 cm/s
Acceleration of Link: 0 cm/s/s

Angle of COM of Link: 45 deg
Velocity of COM of Link: 5 cm/s
Acceleration of COM of Link: 2 cm/s/s

Graphical Analysis

Angle of Link AB
Angular Velocity of Link AB
Angular Acceleration of Link AB

Another visual for data summary


Kinematic Analysis for Link AB

Context Menu

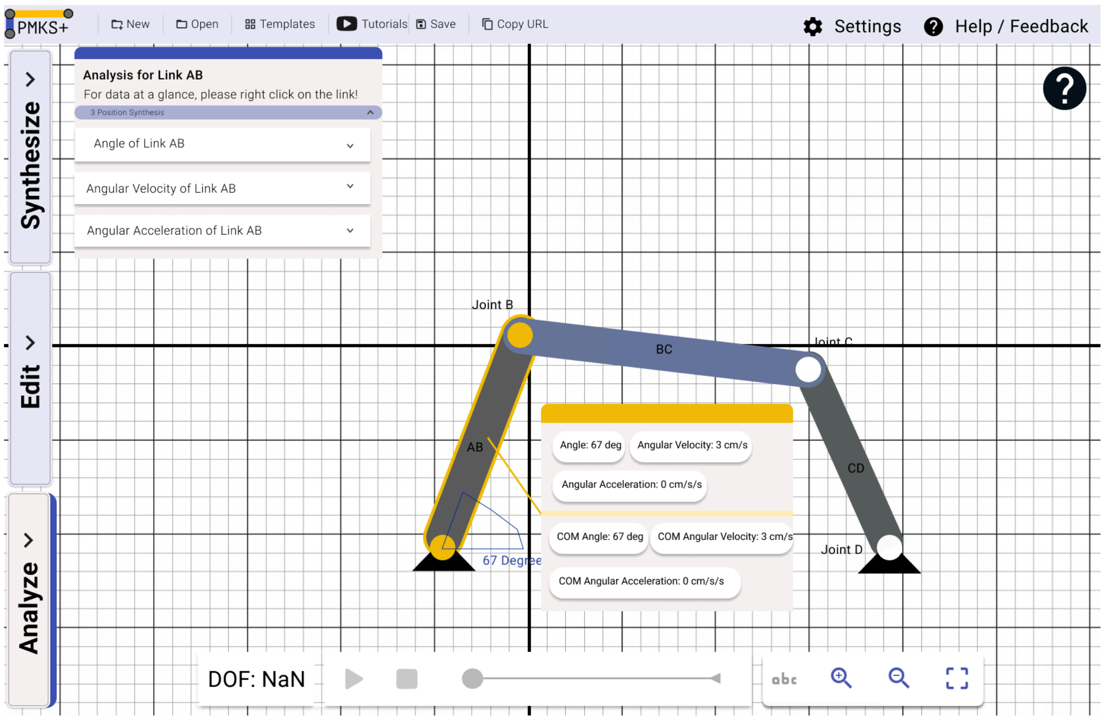
Context of Link AB
Angle of Link: 67deg
Velocity of Link: 3 cm/s
Acceleration of Link: 0 cm/s/s

Context of Link AB COM
Angle of COM of Link: 45 deg
Velocity of COM of Link: 5 cm/s
Acceleration of COM of Link: 2 cm/s/s

Graphical Analysis

Angle of Link AB
Angular Velocity of Link AB
Angular Acceleration of Link AB

Another visual for context menu data summary next to link



Analysis for Link AB

For data at a glance, please right click on the link!

3 Position Synthesis

Angle of Link AB

Angular Velocity of Link AB

Angular Acceleration of Link AB

Angle: 67 deg

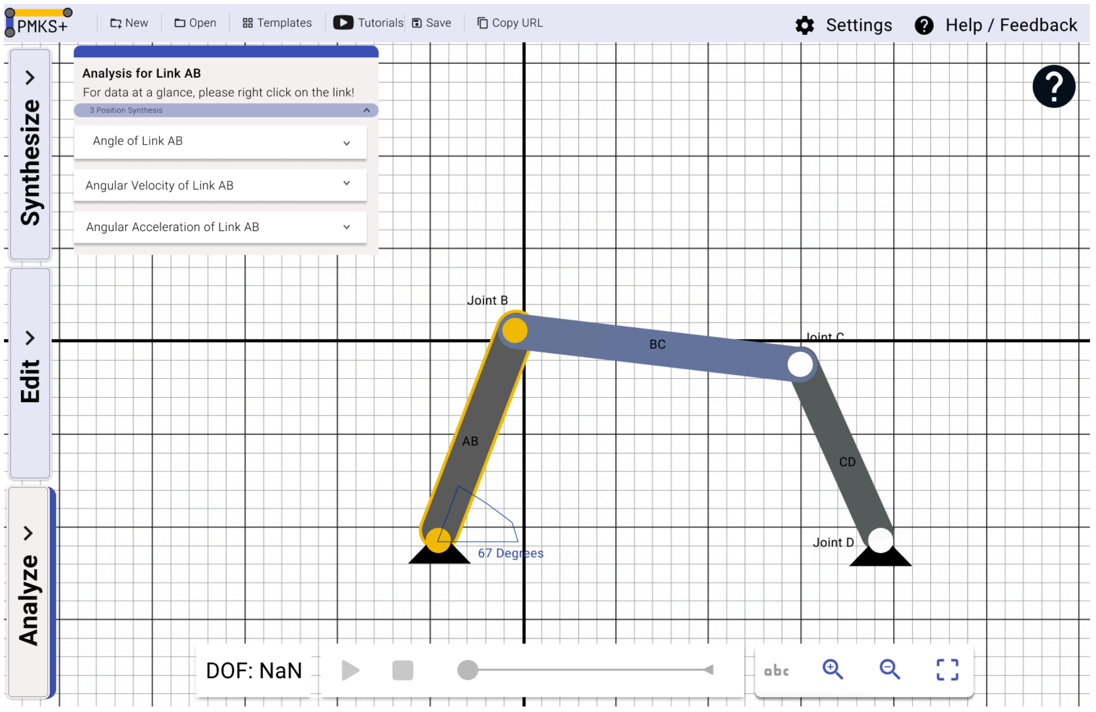
Angular Velocity: 3 cm/s

Angular Acceleration: 0 cm/s/s

COM Angle: 67 deg

COM Angular Velocity: 3 cm/s

COM Angular Acceleration: 0 cm/s/s



Analysis for Link AB

For data at a glance, please right click on the link!

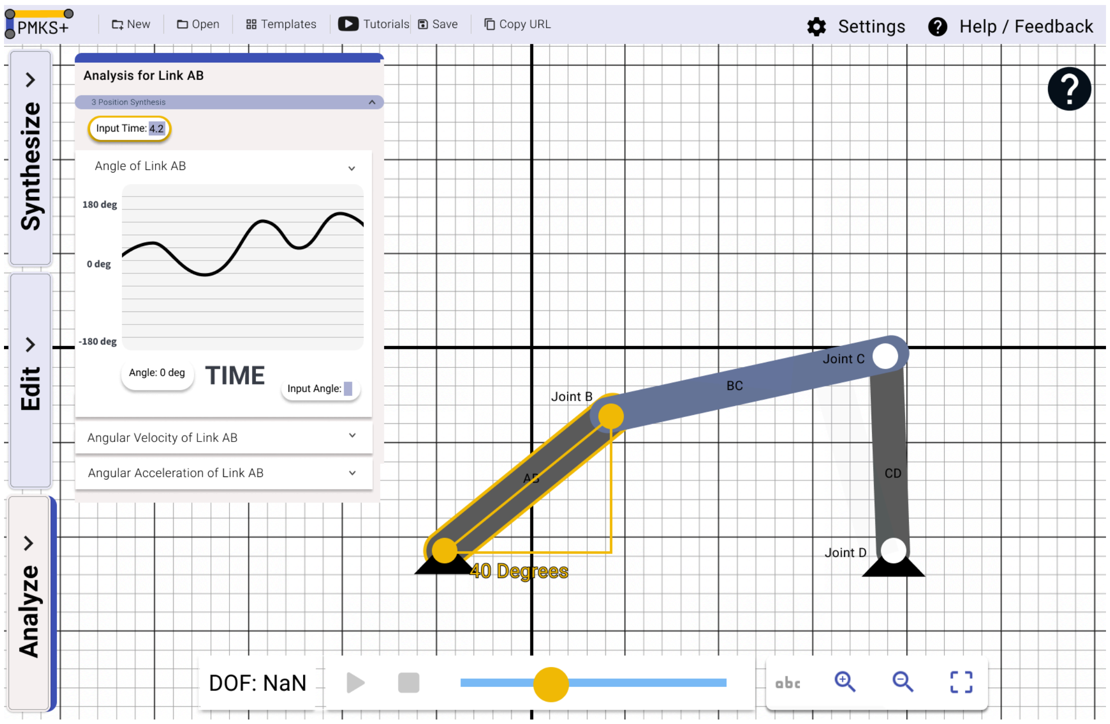
3 Position Synthesis

Angle of Link AB

Angular Velocity of Link AB

Angular Acceleration of Link AB

Input a time for the graph to highlight data at that time and show mechanism at that time



Appendix J: UI Discussion Notes

These document links include the notes taken by our team during our heuristic evaluation of PMKS+2023. They are separated out by section.

- Synthesis Panel
- Edit Panel
- Analysis Panel
- Forces and Equations
- General/other Notes

Appendix K: ReadMe for Future Developers

https://github.com/PMKS-Web/PMKS-Refactor

This project was generated with Angular CLI version 16.2.4.

Development server

Run `ng serve` for a dev server. Navigate to `http://localhost:4200/`. The application will automatically reload if you change any of the source files.

Code scaffolding

Run `ng generate component component-name` to generate a new component. You can also use `ng generate directive|pipe|service|class|guard|interface|enum|module`.

Build

Run `ng build` to build the project. The build artifacts will be stored in the `dist/` directory.

Running unit tests

Run `ng test` to execute the unit tests via Karma.

Running end-to-end tests

Run `ng e2e` to execute the end-to-end tests via a platform of your choice. To use this command, you need to first add a package that implements end-to-end testing capabilities.

Further help

To get more help on the Angular CLI use `ng help` or go check out the Angular CLI Overview and Command Reference page.

Appendix L: Feature List

The lists below can be found in more detail in our Feature List and UI Checklist document. The feature list was made in A term when we were originally working with PMKS+2023 only. The UI checklist is a list of elements that is mostly a summary of our heuristic evaluation of PMKS+2023 and is updated to show the elements included and not included in the current state of PMKS+2024.

A full list can be found at: Feature List UI Checklist

General Features and Bugs

- Weld Joints need different symbol
- Settings
- Three position synthesis
- Account for unit conversion
- Make sure users can't add force on top of a joint
- Dof displayed better to user, top toolbar larger
- Analysis Panel - Multiple minor graph formatting issues
- Ang vel, Ang accel should be synced with to Global units
- prevent users from assigning same names to joints, links, forces, etc.
- Issue when entering/changing joint coordinates
- Update Analysis left panel help text
- Change number of decimal places shown within textbox
- Remove symbols within context menus should have - (minus) sign
- Change icon used for add slider
- Analysis graphs crash when simulating single link
- Add force from welded link to a compound link
- Make sure force is deleted when you delete a compound link
- Position Analysis Differences: Bug in PMKS; Requires further study and solution
- introduce tutorial functionality like intro.js
- New User Welcome Screen
- A dictionary of terminology
- Create Help Videos
- Undo/Redo
- Update Save and Open
- Server-side URL Encoding
- Implement synthesis as part of the URL sharing and Save/Open

- Link synthesis and path synthesis need to be added
- UI implementation of adding lengths of links and the app responding with the possible mechanisms that can be created
- Copy link
- Implement multiple linkages
- Allow users to specify various conditions for synthesis
- Display force diagrams, equations on screen
- Download analysis data from the backend
- Develop optimization modules in the backend
- Equations and Loops Panel
- Implement Kinematic Analysis for instant centers with revolute joints
- Implement force analysis on slider crank
- Snap to 45 deg increments when holding down a hotkey
- Move force location to snap to COM or other tracer points
- Links need to snap together if one joint is dragged on top of another
- Grashoff indicator, or type of linkage indicator
- Multi-selection / Group selection / lasso selection of joins
- Center of mass locations change when adding tracers
- T=0 value bug in analysis graph (data leakage)
- Input velocity for Sliders (m/s)
- SVG Images (Ground, Input) become hidden when they become too small (by pressing 'Update Object Scale')
- Welded joint on a slider should be simulatable
- Put in logic for determining if four bar is parallel
- Add dynamic increment size of calculation tick so that big sliders don't fail by exceeding the maximum simulation
- Make sure that kinematics can be determined when you put an input joint on a prismatic joint
- Welded joints bugs
- Mechanism does not update within simulator properly (Not simulatable when it should be)
- Force analysis type radio needs to change requested analysis type
- Add joint friction
- Iterative Link Length and Angle Solver
- Multiple Forces-Related Features to Add
- Slider ground (not pill) does not adapt size based on joint path

- Force should showcase how angle is determined
- Implement stress analysis
- Allow old PMKS+ links to be opened in the new encoding
- Redesigned Templates Menu
- Improve context sensitive help
- Determine force analysis when torque is unknown and force is unknown

UI Features

Edit Panel - Links

- Make things right aligned
- When you hover on length in edit, this length symbol that appears should have arrow on both sides
- When the angle changes right now the angle on screen disappears, change it to show what the new angle is then disappear
- Units
- Add Pencil and Trash icon
- Center of mass should be displayed when masses and forces are enabled
- Lock icon over locked links

Edit Compound Links

- Locking welded shape
- If a user tries to edit length/angle of welded, display warning saying “Unweld to edit attributes”
- Want to know distance from for compound link ABC want to know distance from A to C (hypotenuse) want to display dotted line with angle similar to how you would see it on the other links
- Change weld icon
- Compound link can be unlocked
- When compound links are locked, allow rotation around joint

Edit Panel - Joints

- Toggle uses x and check
- Group basic settings and distance to joints
- Advanced settings include ground, slider, weld, tracer, make input
- When you change angle when box is hovered it would display the angle visual
- Input direction and speed
- Input arrow change based on direction

- Hover over angle for slider should show it like with joint angle
- Make things right aligned
- Units
- Use length symbol in distance to and angle symbol
- Hover over distance to and show the joint
- Pencil and trash icon
- Change welded joint symbol
- Unweld all should be closer to where it is when you look at the welded joint (middle not bottom)

Analysis

- Integrate stress analysis
- Rpm lowercase
- Redo axis
- Type desired position (angle
- Popup with info summary
- Export graphs
- Export data
- Import time into animation bar to show specific time
- Only one set of values in graphs
- Show only one graph at a time
- Optionally show equations

Three Position Synthesis

- Option to change 4-bar to 6-bar
- Length symbols
- Same UI as edit for dragging
- Delete individual positions
- Highlight text boxes as you change things
- Context menu to delete
- Default position is not the same
- Generate mechanism button

Equations

- Step by step expand equations then plug in numbers
- Add more labels
- Equations need to be on one line

General Notes

- Four decimal places
- No editing in other modes
- Top bar help/feedback together
- Animation bar needs to have input time and dof and time input would show on graph
- Tutorial with video examples
- Change tracer icon to welded joint icon
- Add to general settings whether to show joint path or not