Department of Precision and Microsystems Engineering

Manufacturability in automatic synthesis of planar rigid body spring mechanisms

A.O. Matser

Report no	: 2021.025
Coach	: F. G. J. Broeren MSc.
Professor	: Prof. dr. ir. J. L. Herder
Specialisation	: Mechatronic System Design
Type of report	: Master of Science Thesis
Date	: April 22, 2021

Master of Science Thesis

Manufacturability in automatic synthesis of planar rigid body spring mechanisms

by

A.O. Matser

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday April 22, 2021 at 14:00 AM.

Student number:	4387988	
Project duration:	August 28, 2020 – April 22, 2021	
Thesis committee:	Prof. dr. ir. J. L. Herder	TU Delft, chair
	F. G. J. Broeren, MSc.	TU Delft, supervisor
	Dr. ir. V. van der Wijk	TU Delft
	ir. P. R. Kuppens	external member

TU Delft

Preface

Hey there, dear reader! Thank you for opening my Master thesis, I hope you will read this with the same amount of joy I had during the course of my thesis. The story of this project started, which feels like a lifetime ago, at the end of 2019. That December was a ordinary dark and dreary Dutch month, when parties were still part of regular thursdays and when travels to the warm and sunny southern Europe were still a possibility.

I had talked to Reinier about the work he had done during his Masters and during his PhD. I loved the broad possible application area of this research, thus I wished to take it further and make the automatically evolved mechanism manufacturable. So thank you Reinier, for the inspiration for this work and the guidance during my literature research.

In the summer, when Reiniers supervision came to an end, Freek stepped up as my supervisor for the research side of my thesis. Thanks Freek, for your help whenever I was stuck, for your feedback and for the helpful discussions. Also, I would like to thank Just for his supervision and reminding me to take a step back from time to time and showing me new aspects of my work.

Finally, I would like to thank all the people who made life very enjoyable outside of the Masters research as well. Sanne, Eco, Joyce, family, friends and roommates, thanks for being a part of my life, for taking my mind of work and for listening to me ramble. But for now, please enjoy!

Abel
Delft, April 2021

Abstract

For more than 130 years there have been efforts to automate human services [21]. One such automation is the use of computer-based solving methods to aid engineers in their search for their ideal goal, such as automated planar mechanism design [9, 14–17, 19, 20, 24, 25, 30, 41]. Contrary to creating mechanisms manually, automatic methods are able to create and evaluate mechanisms in quick succession. However, computers can often reach intricate designs which are not directly feasible in the physical domain.

These programs do not focus on being able to manufacture the mechanisms [26], thus, the designs often have features that make it difficult or even impossible to create physically. Examples are inadequate spacing between joints or a mechanism consisting of many elements for which a good layer assignment is hard to find.

To ensure these designs are physically feasible, a manufacturability check can be implemented. Sen et al. [33] created a method to obtain manufacturable planar mechanisms for kinematic designs consisting of rigid bodies and revolute joints. This method identifies structural, kinematic or geometric infeasibilities for all layer assignments. When a mechanism is fully feasible, it can be manufactured.

However, this method is still limited. It is only applicable for mechanisms consisting of rigid bodies and revolute joints. This restricts the application space of possible designs. Also, this method is not tested with physical examples.

In this report, an extension of automatically created manufacturable mechanisms is developed. Firstly, it extends the method such that spring elements in a design can be checked for manufacturability and surrounding links can adapt their shape to accommodate for the springs, resulting in a wider field of applications. Secondly, a method is introduced to simplify the steps to manufacture the theoretical design. Several mechanism prototypes are built using the available methods and their manufacturability is evaluated. Lastly, this research can be used to find the layer assignment with the least number of layers, resulting in thinner mechanisms for space-constrained applications.

Contents

1 Introduction

Specially designed mechanisms make our live better on a daily basis. Some directly, such as mechanisms which can be used for human gait rehabilitation [1], others through the devices we use, for example an airplane shape changing wing [42], but also in the industry, where robots based on parrallel or serial mechanisms perform tasks quickly, precisely and efficiently.

These mechanisms need to be designed carefully with their respective application in mind. An engineer requires intuition and experience to design a good mechanism, which subsequently needs to be analyzed and synthesized correctly. These are complex tasks, which cost a lot of time.

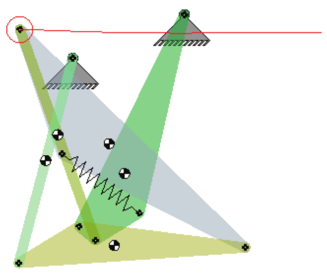
Nowadays, computer-based solving methods can aid an engineer to automatically create planar mechanism designs. These methods are most prevalently used to create the kinematics necessary to follow a given path [9, 14–17, 19, 20, 24, 25, 30, 41]. Other methods focus on the function generation [14] or the (dynamic) balancing of mechanisms [5, 7].

In the optimization method developed by Kuppens and Wolfslag [20], mechanisms that include springs are automatically designed. The springs in this method alter the dynamic properties of a mechanism to obtain the desired kinematic path.

Springs can also be used to improve mechanism designs in terms of efficiency and task execution speed [29]. Plooij and Wisse [29] use a spring to reduce the required energy to move a tool in repetitive tasks. By learning from example, engineers around the world can use springs to reduce the required actuation power of robots in an effort to reduce the environmental footprint.

Nevertheless, incorporating springs to improve mechanism performance is not straightforward. Manually evaluating all viable solutions, when it is possible to change the location of the attachment points of a spring, its initial length, and its stiffness, becomes near impossible. However, it is possible to create automatic methods which add springs to an existing rigid body mechanism. For example, an optimization goal could be to obtain a gravitationally balanced mechanism, which is what Kuppens worked on (unpublished).

However, transforming the automatically created kinematic designs to the physical domain is not trivial. Looking at mechanisms (Fig. 1.1a-b) created in [20], it is not apparent at once how to how to build these designs.


(a) Evolved straight line mechanism [19]

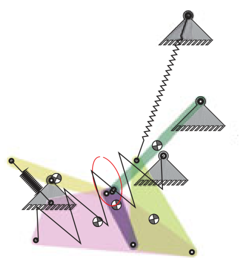

(b) Evolved ellipse-tracing mechanism [20]

Figure 1.1: Mechanisms automatically created by Kuppens [19, 20]

Previously, Sen et al. [33] created a method to obtain manufacturable planar mechanisms for kinematic designs consisting of rigid bodies and revolute joints. This method identifies whether a mechanism has structural, kinematic or geometric infeasibilities. When a mechanism is fully feasible it can be manufactured.

This method can be used by automatic mechanism designing methods to verify if kinematic designs can be build. If the design can not be build, the computer can stop or alter the optimization of that type of mechanism, so processing power can be used effectively. Additionally, this method [33] can create the manufacturable design, which can be build to evaluate a design physically.

However, the work by Sen et al. [33] is incomplete. Firstly, it is only applicable for mechanisms consisting of rigid bodies and revolute joints, thus springs can not be used. Secondly, this method does not directly produce physical examples, solely 3D renderings of mechanisms, making it impossible to physically evaluate a designed mechanism.

Research goals

For this thesis, the research goal is to create a method in which automatically generated rigid body spring mechanisms can be tested for manufacturability, and if they are not directly manufacturable, adapt for these changes by changing the link outlines.

This goal can be divided in three sub-goals:

1. Improve upon the geometric design of interference-free planar linkages by Sen et al. [33], by adding a new element to the planar linkages: springs.

2. Extend the mechanism engine developed by Kuppens [19], to automatically evaluate kinematic designs and produce manufacturable designs.

3. Investigate if automatically generated manufacturable mechanisms are manufacturable and interference free.

To reach these goals, we carefully analyze the mechanism engine developed by Kuppens [19] in chapter 2 and discuss the missing features for our goals. In chapter 3 the feasibilities and adaptations required for manufacturable mechanisms are discussed. Chapter 4 includes the changes necessary for manufacturability to account for springs, leading to the fulfillment of the second sub goal. Next, we use the obtained requirements for the representation of manufacturable mechanisms and incorporate these into the changes we make to the mechanism engine as discussed in chapter 5. This completes the first sub goal. The obtained methods are put into practice by manufacturing prototypes in chapter 6. Lastly, chapter 7 contains the discussion and recommendations and chapter 8 finishes this research with the conclusion.

2 Engine

This chapter provides an introduction to the mechanism engine. The engine contains a method for mechanism representation, simulation, evaluation and optimization. Combined, it represents, synthesizes, and improves mechanism designs. The mechanism engine used as a basis for this paper consists of elements initially developed by Kuppens and Wolfslag [20]. All engine elements are consecutively discussed in this chapter.

2.1. Representation

As a topology representation method adapted graph theory has been used, as described by Kuppens and Wolfslag [20]. Adapted graph theory encodes mechanisms into a DNA, which contains all the information to represent it. Graphs can be represented compactly by incidence strings and coupled with edge labels, their types can be distinguished. Element parameters are stored in separate matrices, and the location of the bodies over time in a state space matrix. Appendix A.1 shows a mechanism synthesis example using these methods.

2.1.1. Graph theory

Graph theory has been widely used for mechanism representation in the past [6, 8, 20, 27, 31, 34–38, 40], even though it was a mathematic concept at first [4]. Drawn graphs consist of vertices and edges, but can also be given in matrix form. Graphs embed the relationships between objects and can be used to show these relationships in physical and abstract systems. A simple example of a drawn graph is a railway system map, where the vertices (also called nodes) depict the train stations and the edges show the connecting railway lines between the stations.

To represent mechanism topology with graphs, vertices depict bodies and edges depict the connection between the bodies [36]. Figure 2.1 shows a graph representation on the left for the mechanism shown on the right. In the graph, the vertices and edges are labeled so elements can be distinguished.

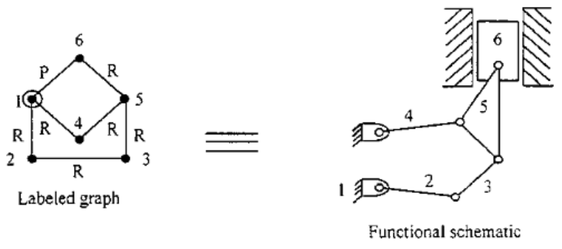


Figure 2.1: Labeled graph of Stephenson chain and corresponding functional schematic. P stands for prismatic joint, R for revolute joint. Adapted from [31]

Drawn graphs can also be represented by incidence matrices, which computers can more easily interface with. Incidence matrices show which edges (columns) and vertices (rows) are connected. Table 2.1 stores the incidence of the Stephenson chain (as shown in fig. 2.1) with ones when edges and vertices which are connected and zeros (left blank for legibility) if unconnected.

Table 2.1: Incident matrix Stephenson chain of figure 2.1. With R: revolute joint and P: prismatic joint

	R	R	R	R	R	R	P
(Ground) 1	1	1					1
2	1		1				
3			1	1			
4		1			1		
5				1	1	1	
6						1	1

Kuppens and Wolfslag [20] use the same graphs and incidence matrices to store the mechanisms and further define the connecting edges. Their edges exist of hinges, prismatic joints and spring forces, which are labeled with H, P and S respectively.

They use this as a basis for their more compact method of mechanism representation. Incidence strings and edge labels are used to automatically design dynamical mechanisms with an evolutionary algorithm. This, combined with their separated physical quantities (see ch. 2.1.2), allows for both the topology and the parameters to be optimized simultaneously and efficiently.

To translate an incidence matrix into an incidence string, a pattern containing all possible column configurations, as shown in figure 2.2, is used. To create the string, the column with the correct connections is found. The index of the column is noted and put in the string. For example, the first column of the incidence matrix in table 2.1 is equal to the first column of the pattern in figure 2.2, resulting in index 1. The second column of the incidence matrix is equal to the fourth row of the pattern. Continuing this for the remaining columns, this results in incidence string $I = [1,\ 4,\ 3,\ 9,\ 10,\ 15,\ 11]$.

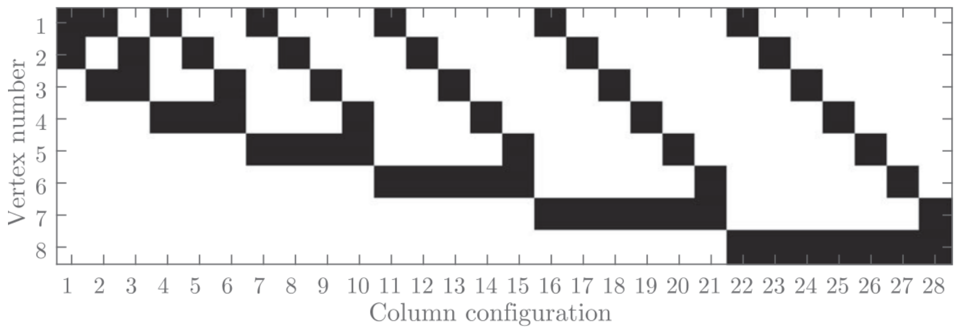


Figure 2.2: Pattern generated for up to 8 vertices. Each black square is a 1 in a column. Taken from [20]

This incidence string captures that elements are connected, but not how they are connected. To include this, Kuppens and Wolfslag [20] introduce the edge label. The edge label has the same size as the incidence string, as it also describes all edges. The edge labels are also translated, but in a different manner. Revolute joints - or hinges in [20] - are encoded as a 1, springs as a 2 and prismatic joints as a 3. Thus, the labeled graph in figure 2.1 and table 2.1 results in edge label $E = [1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 3]$.

2.1.2. Physical quantities

In total, Kuppens and Wolfslag [20] make use of rigid bodies, hinges, springs and prismatic joints. Solely using the topology does not yet embed all information of the mechanism. The starting position of the elements and their parameters still needs to be stored. Therefore, Kuppens and Wolfslag [20] make use of separate matrices for each element type. Finally, an outline is created automatically, based on the preceding matrices.

Bodies (referred to as masses M, and later in this research also as links) are defined by the initial location of the center of mass (CoM) x and y [m] and the mass m [kg], captured in M_{par} (Eq. 2.1, with index i). One exception for bodies is the ground, since it is static, its mass and CoM are irrelevant. Thus, the matrix can also be filled in with NaN values: $M_{\text{par,ground}} = [\text{NaN},\ \text{NaN},\ \text{NaN}]$.

$$M_{\text{par},i} = \left[x,\ y,\ m\right] \tag{2.1}$$

Hinges are defined by the initial location x and y [m], the spring stiffness k [Nm/rad], the damping ratio c [-] and the preload [Nm] captured in the matrix H_{par} (Eq. 2.2). In this research, hinges do not contain springs nor dampening elements.

$$H_{\text{par},i} = \left[x,\ y,\ k,\ c,\ \text{preload}\right]^T \tag{2.2}$$

Springs are defined by the initial locations of both ends of the spring (spring hinges) x_0, y_0, x_1, and y_1 [m], the zero-length L_0 [m] and the spring stiffness k [N/m] (Eq. 2.3).

$$S_{\text{par},i} = \left[x_0,\ y_0,\ x_1,\ y_1,\ L_0,\ k\right]^T \tag{2.3}$$

Kuppens also made an effort to include prismatic joints (Eq. 2.4) and prismatic motors (linear motors) in his unpublished model, but these methods are not included in the scope of this research, and are therefore not integrated with the new manufacturability method.

The prismatic joints are defined by the initial location of the hinge x and y [m] sliding over the linear element and the angle θ [rad] of the linear element.

$$P_{\text{par},i} = \left[x,\ y,\ \theta\right]^T \tag{2.4}$$

Finally, the location and velocity also need to be stored. This is done by storing the location of the bodies' CoM and their derivative, as shown in equation 2.5. Every row i is a single point in time. Using this xss (x for location, ss for state-space), the location of hinges and spring hinges can be derived and used for calculation such as the spring force.

$$xss_i = \left[x_1,\ y_1,\ \theta_1,\ \cdots,\ x_n,\ y_n,\ \theta_n,\ \dot{x}_1,\ \dot{y}_1,\ \dot{\theta}_1,\ \cdots,\ \dot{x}_n,\ \dot{y}_n,\ \dot{\theta}_n,\ \right] \tag{2.5}$$

The link outlines are automatically created after gaining all relevant information on a body. These stored vertices consist of the location of connected (regular, spring or prismatic) hinges and (optionally) the location of the bodies' CoM. The vertices are sorted such that they are arranged in a clockwise direction around the mean of all vertices. Next, a circle of a set radius is drawn around each vertex. This circle is used to find the tangent lines to the succeeding vertex circle. Finally, the outer lines and circle sections are used to create the complete outline. For example, in figure 2.3, the third body has adapted its outline to include the theoretical CoM and the two hinges.

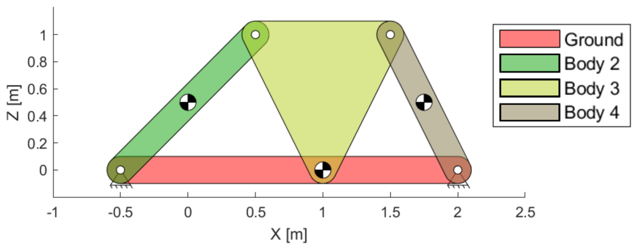


Figure 2.3: Roberts mechanism, depicting the inclusion of CoM in the outline of body 3

2.2. Simulation

To simulate the mechanism movement, a dynamical model can be used. The dynamical model described in [20] is used. In this model, the motions that occur are the effect of conservative forces (gravity $F_{g,i}$ and linear spring forces $F_{s,i}$).

To obtain motion that follows the desired trajectory the mechanism has to be designed with elements that force this movement. For example, the mechanism in figure 2.3 is not a symmetric Roberts mechanism. The second body is slightly longer than the fourth body to ensure the mechanism moves due to its instability. If they would have the same length, the mechanism would be in an unstable equilibrium and would require an initial velocity or external force to gain motion.

Equations of Motion

This dynamical model makes use of equations obtained through virtual work, together with Lagrange multipliers to add constraints [32]. This results in differential-algebraic equation (DAE), described in equation 2.6.

$$\begin{bmatrix} M_{ij} & D_{k,i} \\ D_{k,j} & 0_{kk} \end{bmatrix} \begin{bmatrix} \ddot{x}_j \\ \lambda_k \end{bmatrix} = \begin{bmatrix} f_i \\ -D_{k,pq}\dot{x}_p\dot{x}_q \end{bmatrix} \tag{2.6}$$

In this DAE, M_{ij} is the diagonal mass matrix, $D_{k,j}$ the Jacobian of the constraint vector D_k, x_j the global coordinates, λ_k the Lagrange multipliers, f_i the force vector and $D_{k,pq}\dot{x}_p\dot{x}_q$ the convective acceleration terms, where $x_j = [x_1, y_1, \theta_1, \cdots, x_n, y_n, \theta_n]^T$. Generalized methods can be difficult to find automatically, so this method - which uses CoM coordinates - is used [20].

This equation is solved numerically, thus it has to be written explicitly for $\dot{X}_1$ and $\dot{X}_2$ and Lagrange multiplier λ_k to make it a first order system [19], as shown in equation 2.7, where $X_1 = x_j$ and $X_2 = \dot{x}_j$.

$$\begin{bmatrix} \dot{X}_1 \\ \dot{X}_2 \\ \lambda_k \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & \begin{bmatrix} M_{ij} & D_{k,i} \\ D_{k,j} & 0_{kk} \end{bmatrix} \end{bmatrix}^{-1} \begin{bmatrix} X_2 \\ f_i \\ -D_{k,pq}\dot{x}_p\dot{x}_q \end{bmatrix} \tag{2.7}$$

Forces

Hinge constraints, prismatic constraints and spring forces are also incorporated in the model. For the hinge constraints and spring forces, the distance ΔP between points $\mathbf{p}_1$ and $\mathbf{p}_2$ located on two bodies is necessary. In equation 2.8, $\mathbf{x}_1 = \left[x_1, y_1\right]^T$ is the current CoM position of the first body, θ_1 its current angular position, and $\mathbf{x}_{c1} = \left[x_{c_1}, y_{c_1}\right]^T$ is the initial CoM position of the first body. $\mathbf{x}_2$, θ_2, and $\mathbf{x}_{c2}$ are defined analogously and $R(\theta)$ is the 2D rotation matrix in equation 2.9.

$$\begin{aligned} \Delta P\left(\mathbf{x}_1, \mathbf{x}_2, \theta_1, \theta_2\right) = & R\left(\theta_1\right)\left(\mathbf{p}_1 - \mathbf{x}_{c1}\right) + \mathbf{x}_1 \\ & - R\left(\theta_2\right)\left(\mathbf{p}_2 - \mathbf{x}_{c2}\right) - \mathbf{x}_2 \end{aligned} \tag{2.8}$$

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{2.9}$$

To fix the horizontal and vertical location of adjacent bodies coupled by hinges, two constraints of the form $\Delta P = 0$ are added to D_k with $\mathbf{p}_1 = \mathbf{p}_2 = \mathbf{x}_h$. When a body is connected to a ground, $\mathbf{x}_2 = \mathbf{x}_{c2} = \mathbf{x}_h$.

The forces in the system are given by $f_i = G_i + S_i + H_i$, where G_i are the gravity forces (Eq. 2.10), S_i the spring forces (Eq. 2.12) and H_i the hinge forces (Eq. 2.13). The spring force depends on its stiffness k and elongation from its zero-length L_0 (Eq. 2.12). The hinge rotational force (Eq. 2.13) depends on the torque (Eq. 2.15) and the damping ratio (Eq. 2.17), which depend on the rotation of the hinge in body i relative to adjacent body j.

$$G_i = \begin{bmatrix} 0 & -m_1 g & 0 & \cdots & 0 & -m_n g & 0 \end{bmatrix}^T \tag{2.10}$$

$$E_s\left(x_j\right) = \frac{1}{2}k\left(\Delta P - L_0\right)^2 \tag{2.11}$$

$$S_i = -\frac{\partial E_s}{\partial x_j} \tag{2.12}$$

$$H_i = H_i^{\mathrm{s}} + H_i^{\mathrm{d}} \tag{2.13}$$

$$E_s(x_j) = \frac{1}{2}k(\Delta\theta - \theta_c)^2 \tag{2.14}$$

$$H_i^{\mathrm{s}} = -\frac{\partial E_s}{\partial x_j} \tag{2.15}$$

$$E_f(x_j) = \frac{1}{2}k(\Delta\dot{\theta} - \dot{\theta}_c)^2 \tag{2.16}$$

$$H_i^{\mathrm{f}} = -\frac{\partial E_f}{\partial \dot{x}_j} \tag{2.17}$$

Solver

The Runge-Kutta fourth order method (RK4) is used to numerically integrate equation 2.7. RK4 is set at a fixed step-size of $h = 0.05$, uses the the Butcher tableau in table 2.2, and starts with initial conditions:

$$x_j(0) = \begin{bmatrix} x_{c1} & y_{c1} & \theta_{c1} & \cdots & x_{cn} & y_{cn} & \theta_{cn} \end{bmatrix} \tag{2.18}$$

$$\dot{x}_j(0) = \mathbf{0} \tag{2.19}$$

and is run until T = 5 seconds has been reached. To safeguard against RK4 hogging valuable resources, a few tests are included to ensure that the evaluated mechanism can be used. It checks for NaN-values, which are indicative of singularities. It also checks if the difference between the current state and the previous state does not exceed 10 m for x_j and 10 m/s for $\dot{x}_j$. If either fails, the integration is terminated.

Table 2.2: Butcher tableau for Runge-Kutta 4th method

0	0	0	0	0
1/2	1/2	0	0	0
1/2	0	1/2	0	0
1	0	0	1	0
	1/6	1/3	1/3	1/6

2.3. Optimization

Since mechanisms can be applied to a variety of situations, several possible optimizations goals for mechanisms exist. One option, as outlined in [18], is to optimize a mechanism to follow a predefined path. This path can be a straight line, an ellipse, or even approximating a human gait to build a mechanism for protheses. The evolutionary algorithm used by [18] proves to be capable of designing mechanisms that track straight lines and ellipses.

Another possibility is to optimize a kinematic mechanism - designed to follow a certain path - to gravitationally balance it, which is a method Kuppens worked on. This section focusses on this gravity balancing optimization, as this is one of the optimization areas wherein verifying the manufacturability of planar rigid body spring mechanisms is useful.

The process starts with a mechanism consisting of rigid bodies, hinges and prismatic joints, thus without springs (also referred to as PHDNA, for the Prismatic joints, Hinges and bodies are encoded in a DNA). The path of this mechanism depends on the rigid bodies, so if parameters of the rigid bodies change, the path changes as well. Thus, these parameters stay fixed in this optimization.

The second ingredient to optimize towards a gravitationally balanced mechanism are springs. Springs can be added to the PHDNA freely, without adjusting the followed path. However, they do add to the total energy in the system, which can be used to obtain constant potential energy. Now, the mechanism can move in its path without operating energy. This ingredient is captured in a spring DNA (SDNA).

Initialization

An initial SDNA is created randomly. First, the number of vertices in the PHDNA is stored in nVtx_{max}. Since every vertex is associated to a body, this is equal to the number of bodies in the PHDNA. Secondly, the predefined parameter nS_{max} is used, which sets the maximum number of springs for the SDNA.

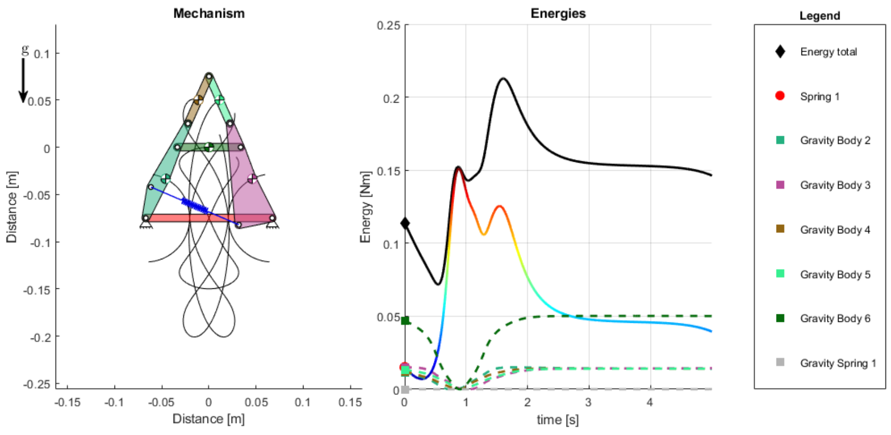
A spring incidence string I_s is created with a random length up to nS_{max}. I_s is subsequently filled with random integer values between 0 and nVtx_{max}. Also, an edge label with the same length as I_s is created: $E_s = \mathbf{2}$, to signify all edges are springs. Next, the spring parameters within S_{par} are randomly assigned with random values between 0 and 2. Together, these elements form the SDNA. This SDNA describes between which bodies springs are added with I_s and its parameters are stored in S_{par}.

Optimization

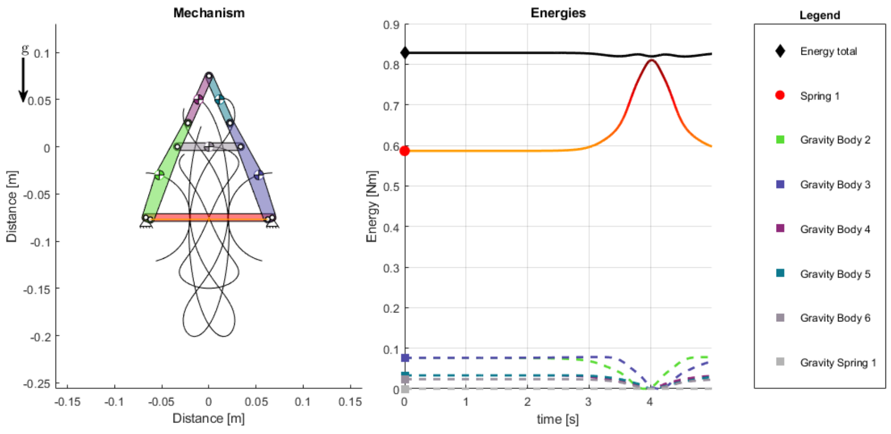
For the optimization both a PHDNA and an SDNA are used. The SDNA is added to the PHDNA, creating a mechanism including springs, called the DNA. This DNA comprises of the concatenated incidence strings and edge labels and reusing M_{par} and S_{par}. This DNA is evaluated by obtaining the energy of the system at every timestep. The energy is the sum of the gravity and spring energy: $E_t = E_g + E_s$. To obtain the gravity energy:

$$E_{g,j} = m_j g y_j \tag{2.20}$$

is evaluated for every body, with m_j the mass of the jth-body as defined in M_{par} and y_j its vertical location as obtained with equation 2.7. To obtain the spring energy, equation 2.11 is evaluated at every time step.



(a) Before optimizing the spring for gravitational balancing.



(b) After optimizing the spring (now situated below the red body) for gravitational balancing.

Figure 2.4: A-frame spring mechanism on the left with its energy graph on the right. Paths of all CoM are depicted as black lines on the left and their gravity energy is depicted with striped lines on the right. The spring energy corresponds to the rainbow colored line on the right. Colors in the legend match with the color of the elements. Note: the gravity energy of the spring is not calculated.

When the resulting total energy (Fig. 2.4a) is not approximately constant, an optimization is necessary. This is evaluated by taking the total energy's gradient and then taking the sum of squares to obtain the fitness:

$$\text{fitness} = \dot{\mathbf{E}}_{\text{total}}^T \dot{\mathbf{E}}_{\text{total}} \tag{2.21}$$

The optimization is done with the interior-point method. This method solves linear and nonlinear convex optimization problems and works fastest with continuous variables. The mechanism shown in figure 2.4a optimizes towards the gravitationally balanced mechanism in figure 2.4b.

2.4. Conclusion

To take manufacturability into account, the preceding engine is incomplete. Three aspects need to be improved:

Firstly, the created mechanisms are purely 2D: no layer configuration is assigned and the bodies do not have a defined depth. Without body depth, the mechanism weight can not be based on real values and has to be manually defined. Also, without a layer configuration the body interference can not be evaluated. Therefore, it is unknown if given mechanisms can make the movements they are supposed to. These extensions to the representation are discussed in section 5.1.

Secondly, the information for each element is hardcoded into the engine. This results in mechanisms elements like the hinge diameter or the width of a body to have a fixed value. When one is able to set the physical parameters in a centralized way as shown in section 5.2, multiple parameter combinations can be configured so that the engine can work for different application fields. This allows this engine to work just as well for mechanisms the size of micro-electromechanical systems as for mechanisms the size of cranes.

Lastly, the spring optimization method is difficult to use with off-the-shelve springs. Off-the-shelve spring parameters can not be chosen with continuous variables, but exist in a discrete catalogue. This results in a finite number of variations. A method to directly use springs from the manufacturer is included in section 5.3.

3 Manufacturability

Sen et al. [33] created a method to automatically evaluate and adapt rigid body mechanisms for manufacturability. This is helpful in the process of computerized, automatic mechanism design, since it can evaluate if a kinematic design is feasible or if it should not be used. However, the application space is limited when the only possible parts are hinges and rigid bodies. Therefore, manufacturability of mechanisms with springs is investigated.

This chapter recaps the existing method creating ’interference-free’ planar linkages [33]. The method to ensure manufacturability for mechanisms with spring elements is outlined in the next chapter. Chapter 4 also includes a method dedicated to prevent double adaptations which lead to infeasible mechanisms.

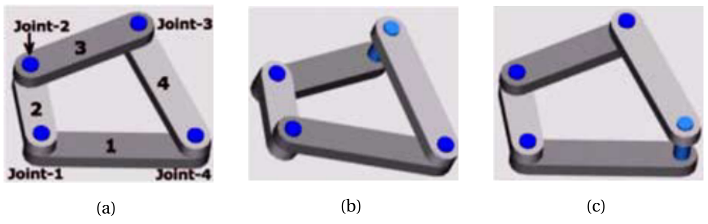


Figure 3.1: Mechanisms with full rotatability. Adapted from [33]

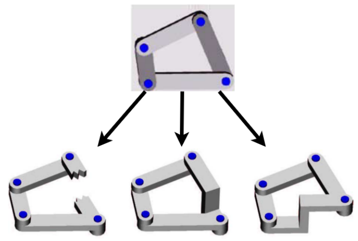


Figure 3.2: Mechanism with a bent link. Adapted from [33]

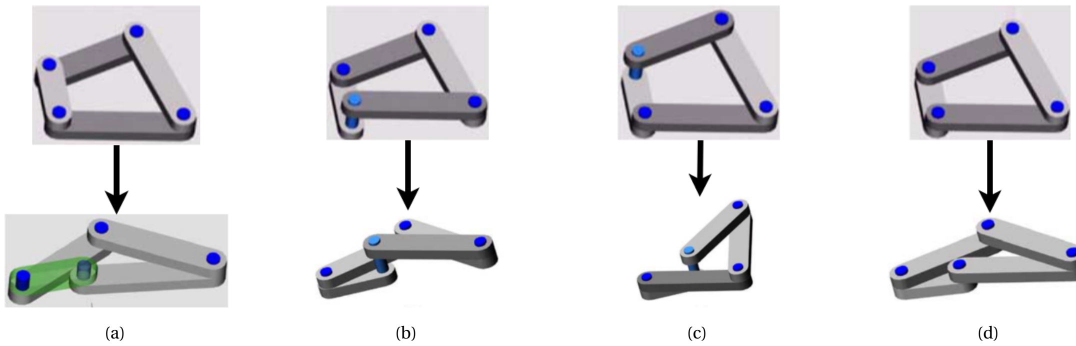


Figure 3.3: Mechanisms with interference in motion. Adapted from [33]

In mechanisms, the moving bodies can form obstacles and impede each others motions. This motion obstruction prevents the mechanism from completing its designed range of motion. Sen et al. [33] identifies three types of infeasibilities: structural, kinematic and geometric. When a mechanism does not contain any of these infeasibilities, it can successfully complete its designed range of motion and be fabricated. This method is limited to mechanisms consisting of rigid bodies and joints[1].

A four-bar Grashoff-type mechanism is used as example by Sen et al. [33]. A four bodied mechanism can have eight distinct layer assignments, which leads to eight different configurations (Fig. 3.1-3.3). In this mechanism, shown with labels in figure 3.1a, the first body is fixed and acts as a ground, and the second body

[1]In this section and in [33], links refer to rigid bodies whereas joints refer to revolute joints.

is designed to make a full rotation. Three distinct outcomes are possible for all possible layer assignments: first, the mechanism can complete its designed range of motion (Fig. 3.1). Second, the bodies are assigned in such a way that one body needs to be in two layers (Fig. 3.2) , resulting in a body with an S-shape. A body always has hinges connecting it to another body in a different layer, which creates an out-of-plane arm for the out-of-plane hinge forces, resulting in less stiff bodies. Therefore, the creation of an S-shape is deemed a structural infeasibility. Third, interference during movement occurs (Fig. 3.3), hindering it to complete its range of motion: a kinematic infeasibility.

To overcome these infeasibilities, two design steps have to be taken: structural synthesis (Ch. 3.1) and geometric synthesis (Ch. 3.2).

3.1. Structural synthesis

In structural synthesis all bodies are assigned to a layer, which is called the layer assignment. For a mechanism with a single degree of freedom, the maximum number of distinct layer assignments can be calculated with equation 3.1 [33]. A mechanism with four revolute joints (Fig. 3.1a) leads to eight separate layer assignments, while for a mechanism with seven revolute joints (Fig. 3.4a) 64 assignments are possible. The possible number of assignments increases by a power of two with the number of joints in the system (Tab. 3.1). Manually creating and evaluating all assignments for mechanisms with more than four joints is time-consuming, so in this section an automatic method is described which reduces all assignments to the structurally feasible assignments.

$$\lambda = 2^{j-1} \qquad \text{with } j \text{ number of joints} \tag{3.1}$$

To obtain these layer assignments, Sen et al. [33] makes use of graph theory, in specific the adjacency matrix describing the mechanism. The undirected link (or body) adjacency matrix (**Lu**) is obtained (Fig. 3.4e). In this matrix the connection between different links is stored with a non-zero value at their intersection. Since adjacent links can not be in the same layer, they need to be placed higher or lower than their adjacent counterpart. To store this information, a directed adjacency matrix (**L**) is created (Fig. 3.4g) for every possible assignment. When link a is higher than adjacent link b, $\mathbf{L}_{b,a} = 1$, while the transposed value is equal to its negative counterpart: $\mathbf{L}_{a,b} = -\mathbf{L}_{b,a}$. When this is done for all links, the complete layer stacking is obtained.

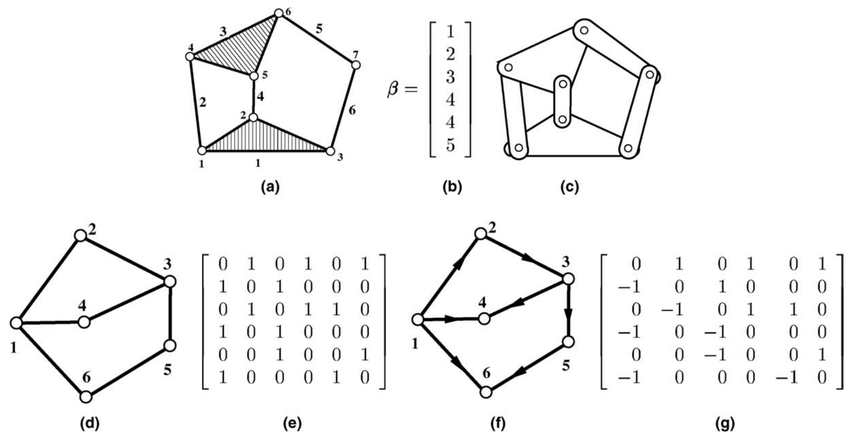

(a)
(b)
(c)
(d)
(e)
(f)
(g)
$\beta = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 4 \\ 5 \end{bmatrix}$
$\begin{bmatrix} 0&1&0&1&0&1 \\ 1&0&1&0&0&0 \\ 0&1&0&1&1&0 \\ 1&0&1&0&0&0 \\ 0&0&1&0&0&1 \\ 1&0&0&0&1&0 \end{bmatrix}$
$\begin{bmatrix} 0&1&0&1&0&1 \\ -1&0&1&0&0&0 \\ 0&-1&0&1&1&0 \\ -1&0&-1&0&0&0 \\ 0&0&-1&0&0&1 \\ -1&0&0&0&-1&0 \end{bmatrix}$
Figure 3.4: Different representations of the Stephenson chain (a) kinetic chain, (b) layer assignment vector, (c) construction, (d) the graph, (e) adjacency matrix (**Lu**), (f) a directed graph and (g) directed adjacency matrix (**L**). Taken from [33]

Table 3.1: Variation of number of layer assignments with links for DOF = 1 [33]

n	j	$\lambda = 2^{j-1}$
2	1	1
4	4	8
6	7	64
8	10	512
10	13	4096
12	16	32768

With this method, all distinct relative layer stackings are obtained. The upper triangular form ($\mathbf{L}_{ut}$) is used to obtain all different assignments with 1's and -1's. Next, the assignments (i) which are equal to the negative of another option (j) ($\mathbf{L}_{ut,i} == -\mathbf{L}_{ut,j}$) are removed, since these give the same construction order, but mirrored about the lowest plane. Lastly, the complete directed matrices ($\mathbf{L}_j = \mathbf{L}_{ut,j} - \mathbf{L}^{\mathrm{T}}_{ut,j}$) are obtained.

When all these unique layer assignments are obtained, the structurally infeasible options are removed with the algorithm described in [33]: *"A layer assignment is considered feasible or consistent if we can assign a layer index to each link such that all the edges in the corresponding directed graph goes from a link with lower layer index to a link with higher layer index"*. With this rule the assignment shown in figure 3.2 is judged structurally infeasible.

3.2. Geometric synthesis

The remaining structurally feasible layer assignments need to be evaluated to ensure the designed range of motion can be obtained. There are three possible outcomes for the remaining configurations:

1. The desired range of motion can be obtained without changing the link outline.
2. The desired range of motion be obtained when changing the link outline.
3. The desired range of motion not be obtained.

To evaluate in which category the remaining configurations belong, every layer needs to be evaluated separately. The first step of layer evaluation is done by obtaining the swept area of every moving element in a layer. When the relative swept areas of links overlap, full motion can not be achieved directly. The moving elements in a layer consist of:

1. The swept areas of links designed in the layer.
2. The swept areas of joint pins passing through the layer.
3. The swept areas of joints in the links in the layer.

3.2.1. Swept areas

To obtain the swept areas, the position of links and joints throughout the range of motion need to be known. The continuous swept area of a link is shown in figure 3.5b, but is computationally costly to obtain. Two methods can be used to approximate this swept area: the explicit- and implicit swept area [33].

The first option to approximate the continuous sweep is with the explicit sweep. For the explicit sweep, the continuous sweep is discretized to 'n' intervals [33] (Fig. 3.5a). When using 'm' points to represent the joints, the computational complexity is of order $\mathcal{O}(m \times n^2)$, which approximates $\mathcal{O}(n^4)$ [33]. The precise method to obtain this data is described in [33].

The second option to approximate the continuous sweep is with the implicit sweep. The implicit sweep is handled differently for joints and links. The swept area of a joint pin is implicitly represented by a curve at a set distance from the joint center.

The implicit swept area for a link is obtained differently than a joint. The implicit swept area of links makes use of discretized intervals to capture the points representing the link outlines, similar to the the explicit sweep. However, the difference between the implicit- and the explicit swept area of links is in the step size. In the implicit swept area, bigger step sizes are possible. An example of the discrete steps obtained for the swept area of links is shown in figure 3.5a. These steps do not match with the real, continuous swept area (Fig. 3.5b). To obtain an approximation of the swept area three methods are proposed.

1. The union of individual positions (Fig. 3.5c)
2. The convex hull of all of them (Fig. 3.5d)
3. The convex hull of two successive link geometries and the union of all of them (Fig. 3.5e)

Method 1 is the fastest with a processing cost of order $\mathcal{O}(n)$, method 2 slower with an order of $\mathcal{O}(n \log n)$ and the third method the slowest with order $\mathcal{O}(n^2 \log n)$ [33]. Sadly, the fastest method does not account for all swept areas and the second method overestimates the swept areas. The third and slowest method is the most accurate and can safely be used to obtain the most reliable results when using enough intervals. A comparison of these methods is given in appendix B.1.

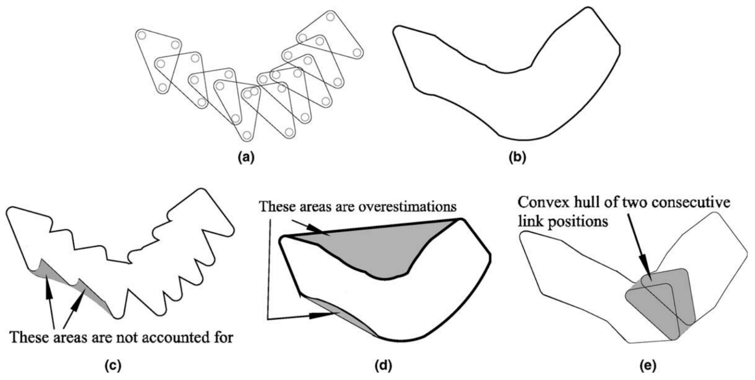

(a)
(b)
These areas are not accounted for
(c)
These areas are overestimations
(d)
Convex hull of two consecutive link positions
(e)

Figure 3.5: Implicit swept area by three approaches: (a) successive link positions, (b) actual swept area, (c) union of link positions, (d) convex hull of link positions and (e) union of convex hull of two successive link positions. Taken from [33]

3.2.2. Kinematic and Geometric infeasibilities

The structurally feasible configurations still need to be evaluated for kinematic and geometric infeasibilities. With the previously obtained swept areas of joints in the layer and joints passing through the layer, the configurations are evaluated.

When assessing the remaining infeasibilities for a link, the elements in the same layer as the inspected link create the infeasible area. This infeasible area consists of the swept area relative to the currently inspected link. The infeasible area has an outer (B_o) and inner edge (B_i) (Fig. 3.6).

Next, the joints connected to the currently inspected link need to be classified. If a joint is outside all B_o, it is classified OUT. If a joint lies within the infeasible area inbetween B_o and B_i, it is classified ON. If a joint lies within B_i, it is classified IN.

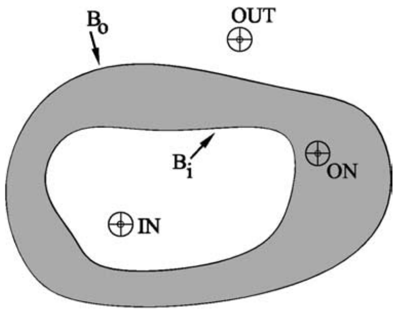


Figure 3.6: Joint classification. Taken from [33]

These classifications can be used to determine which situation is applicable to a configuration: a feasible geometry, a geometric infeasibility, or a kinematic infeasibility. The links are feasible if either:

1. All joints are OUT.
2. All joints are IN.
   (a) Inside the same B_i.

and if the center of the joints are at an appropriate distance from the infeasible area. If not, the joint interferes with that area and the link is geometrically infeasible. The link is kinematically infeasible if:

2. All joints are IN
   (b) Inside different B_i.
3. Some joints are IN, others are OUT.
4. One or more joints are ON.

As an example, take a look at the example in figure 3.7a. The inspected link lies in the middle and has two connected circular joints. Around the link lies an infeasible area, but it is within an appropriate distance from the center of the joints, thus this link has a feasible shape. In the second example, shown in figure 3.7b, the same is true so it is also geometrically feasible. However, the dotted link shape is currently geometrically infeasible. The infeasible area overlaps the dotted link shape, thus this shape used as an outline for the link is infeasible. The link shape needs to be changed to become geometrically feasible. In the last example, shown in figure 3.7c, the joints of the link lie within different infeasible areas, thus resulting in a kinematically infeasible shape.

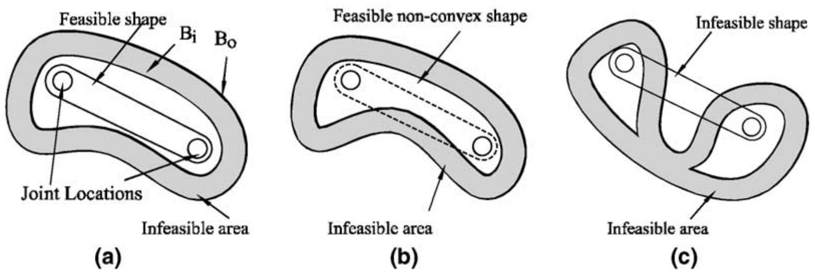


Figure 3.7: Determination of feasible geometry with all joints IN: (a) feasible nominal geometry, (b) modified nominal geometry and (c) infeasible geometry. Adapted from [33]

3.2.3. Feasible geometry

To obtain the geometrically feasible shape for cases similar to the second example in figure 3.7b, the link shape needs to be changed. First, one starts with a given outline of the link, for example Sen et al. [33] uses a convex hull (the nominal geometry) around a link with three joints (Fig. 3.8a). Next, the infeasible areas in the same layer as the inspected link are evaluated one by one with three possible outcomes:

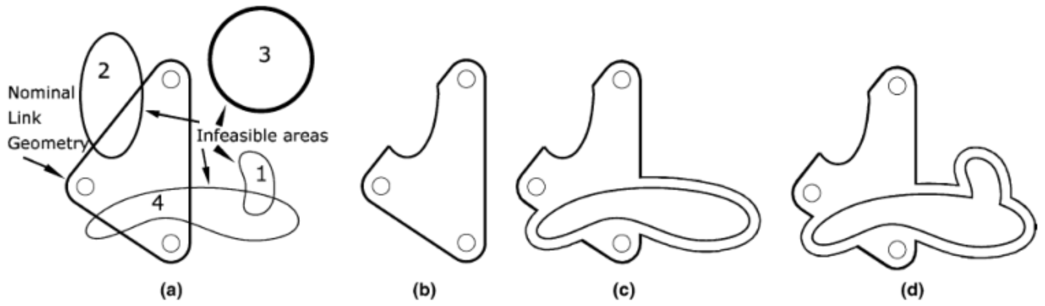


Figure 3.8: Determination of feasible geometry with all joints OUT: (a) nominal geometry with infeasible areas, (b) step 1, (c) step 2 and (d) obtained feasible geometry for link. Taken from [33]

At first, when an infeasible area lies partly inside the link geometry, the infeasible area in question is subtracted from the link geometry if the resulting geometry is still one body. This happens in figure 3.8b, where infeasible area 2 is overlapping a part of the nominal link geometry.

Secondly, if an infeasible area causes the link geometry to be split in multiple separate shapes, a different method is followed. First, a buffered shape is created, consisting of the infeasible area with an offset around its outer edge, creating a small buffer. This buffered shape is added to the link geometry. Next, the original infeasible area is subtracted from the new link geometry, which results in figure 3.8c. If infeasible area 4 would have been subtracted from the original geometry directly, the link would have been split in two parts. Thirdly, if an infeasible area does not overlap the link geometry, no action needs to be undertaken.

Finally, when all infeasible areas in the layer of the inspected link have been taken into account, another run commences. This new iteration checks if previously not overlapping infeasible areas are overlapping now (for example, area 1 has to be adapted for, as shown in figure 3.8d). This procedure is repeated until none of the infeasible areas overlap the link geometry.

3.3. Implementation

The first step towards creating manufacturable designs for spring mechanisms, is to implement the method as described in the previous sections by Sen et al. [33]. This method allows for the creation of manufacturable rigid body mechanisms and is a good basis towards an expansion including springs. To summarize, the implemented steps are (in the order of execution):

1. Obtain outlines of:
   (a) all joints at their absolute location.
   (b) all joints relative to all other links.
2. Obtain all structurally feasible layer configurations
3. Fill configurations information with the swept area of itself and the infeasible area - using the previously obtained outlines.
4. Evaluate if the layer configurations are kinematically feasible
   (a) When all configurations are kinematically infeasible stop the process.
   (b) When a configuration is geometrically feasible, repeat step 1 and 3 for link outlines instead of only joint outlines.
5. Obtain geometrically feasible outlines.
6. Choose the best configuration to present to the user.

3.3.1. Obtain outlines

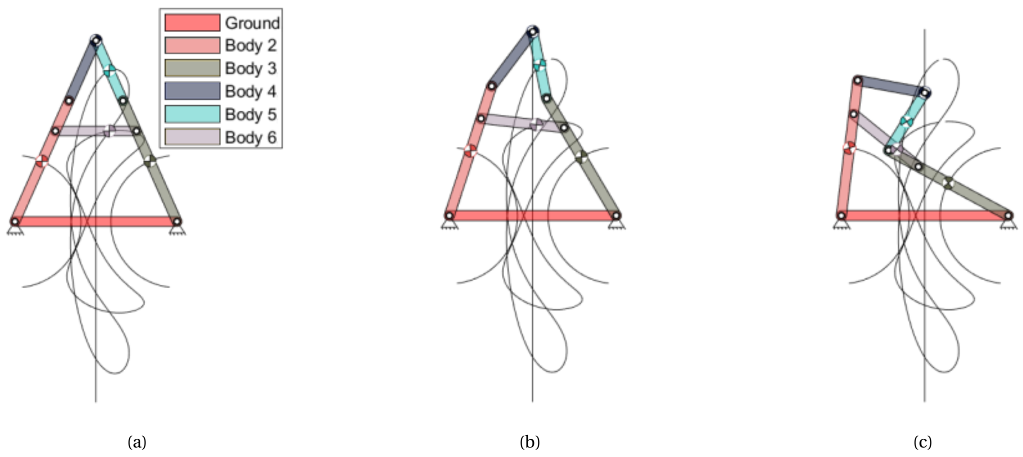


Figure 3.9: Movement of Harts A-frame. The paths of the (manually determined) CoMs are traced with a black line. Full movement in appendix B.2

In this section Harts A-frame [10] (Fig. 3.9) is used as an example. To start, the swept area of the hinges and the links are obtained. Next, the inverted (or relative) swept area is obtained. Finally, two geometrically feasible link outlines are obtained.

The resulting swept area of hinges are shown in figure 3.10a-b. This swept area is the same as the swept area relative to the ground link, as the ground link does not move. The swept areas calculated in this step can be used to evaluate if layer configurations are kinematically feasible.

The swept area of hinges are calculated with the initial outline of a hinge $\begin{bmatrix}\mathbf{x}_{\text{old}}\\ \mathbf{y}_{\text{old}}\end{bmatrix}$. This hinge outline consists of 314 points on a circle with a fixed radius from the center of the hinge position. Subsequently, this outline is moved according to the mechanisms movement at different time steps (Eq. 3.2). The total swept area is obtained using to the theory discussed in section 3.2.1.

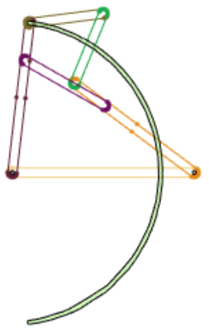
$$\begin{bmatrix}\mathbf{x}_{\text{new}}\\ \mathbf{y}_{\text{new}}\end{bmatrix} = \begin{bmatrix}\cos(-\theta_{\text{old}}+\theta_{\text{new}}) & -\sin(-\theta_{\text{old}}+\theta_{\text{new}})\\ \sin(-\theta_{\text{old}}+\theta_{\text{new}}) & \cos(-\theta_{\text{old}}+\theta_{\text{new}})\end{bmatrix}\begin{bmatrix}\mathbf{x}_{\text{old}}-x_{\text{old}}+x_{\text{new}}\\ \mathbf{y}_{\text{old}}-y_{\text{old}}+y_{\text{new}}\end{bmatrix} \tag{3.2}$$

with $\begin{bmatrix}\mathbf{x}_{\text{new}}\\ \mathbf{y}_{\text{new}}\end{bmatrix}$ as the outline of a hinge translated and rotated to a new location, θ_{old} as the previous rotational position of the link, x_{old} and y_{old} as the previous center of mass of the link, and θ_{new}, x_{new}, and y_{new} as the new location data.

The moved outlines $\begin{bmatrix}\mathbf{x}_{\text{new}}\\ \mathbf{y}_{\text{new}}\end{bmatrix}$ are obtained for time points within the time range obtained by the simulation described in chapter 2.3. The standard time range is from 0 seconds to 5 seconds, with the location of the links calculated for every 0.01 seconds. Even though we have the location data for every 0.01 seconds, every

0.05 seconds an outline is obtained. This is done because the computation time is increased five fold, while the improvement in results are not directly necessary (For more details, see appendix B.1).

Alternatively, a faster method to obtain the swept outline for hinges can be used. Since the hinges are stored as simple circles in 2D, the rotation does not matter. With the location of the center of the hinges at every time step ($\Delta t = 0.01s$), the line connecting these points can be used to create a shape instantly. Using Matlab's `polybuffer` command, the total swept area of the hinge outlines can be obtained.


(a) Swept area of the hinge connecting body 2 and body 4

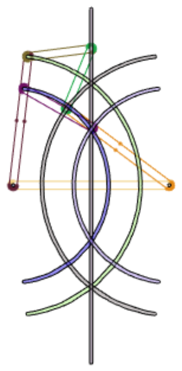
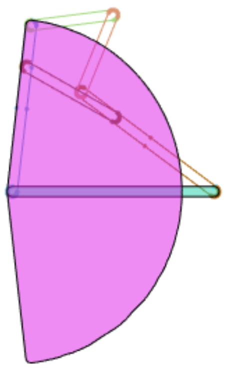

(b) Swept area of all hinges

Figure 3.10: Hinge outlines

Then, the swept area of the links are also obtained with equation 3.2. This is only done if a kinematically feasible mechanism exists, in an effort to minimize wasted calculation cost and reduce the time to evaluate kinematically infeasible mechanisms.


(a) Swept area of the ground (turquoise) and body 1 (pink)

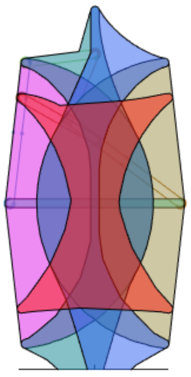

(b) Swept area of all bodies

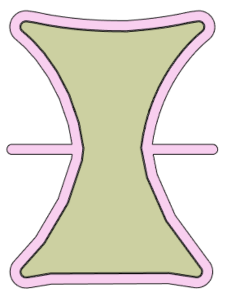
Figure 3.11: Link outlines

To obtain the inverted swept area, some more clarification is necessary. A quick reminder what the definition of an inverted swept area is: the swept area of a link relative to another link. For example, all swept areas inverted with respect to the ground - or a fixed link - consist of the same shape as the original swept area. Only once a swept area is obtained relative to a moving link, the inverted swept area looks different than the original swept area. Previously, the outline points $\begin{bmatrix}\mathbf{x}\\\mathbf{y}\end{bmatrix}$ during all time steps were obtained. These are inverted per time step:

$$\begin{bmatrix}\mathbf{x}_{\text{inverted}}\\\mathbf{y}_{\text{inverted}}\end{bmatrix} = \begin{bmatrix}\cos(-\theta_{\text{CoM}}) & -\sin(-\theta_{\text{CoM}})\\\sin(-\theta_{\text{CoM}}) & \cos(-\theta_{\text{CoM}})\end{bmatrix}\begin{bmatrix}\mathbf{x}-x_{\text{CoM}}\\\mathbf{y}-y_{\text{CoM}}\end{bmatrix} \tag{3.3}$$

with $\begin{bmatrix}\mathbf{x}\\\mathbf{y}\end{bmatrix}$ as the coordinates of all outline points of the link, θ_{CoM} as the current rotational position of the link and x_{CoM} and y_{CoM} as the current center of mass of the link.

Figures 3.12a-b are examples for the use of the inverted swept area. The infeasible area shown in these figures consists only of the swept areas of other links in the same layer, the inverted swept area of those links. In figure 3.12a, the infeasible area consists of the red area shown in figure 3.11b, which is the swept area of the sixth body in figure 3.9. In figure 3.12b, the infeasible area is created by the movement of the second body relative to the third body.


(a) Body 1 (pink, adapted geometry) and infeasible area (green).

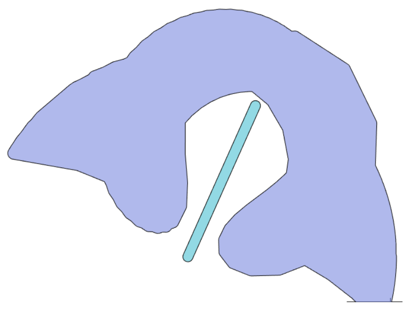
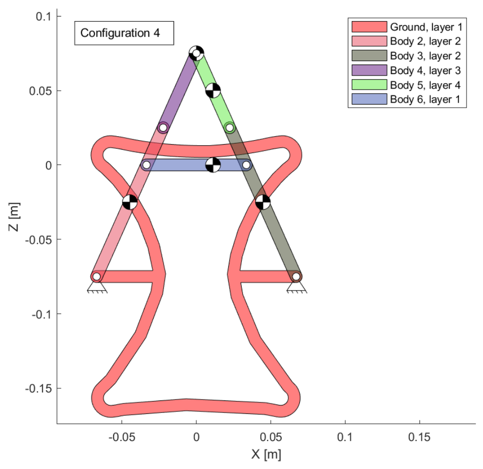

(b) Body 2 (turquoise) and infeasible area (purple).

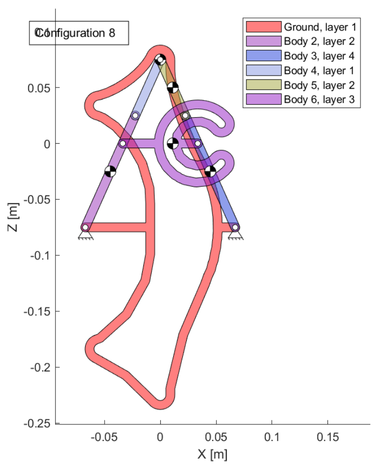
Figure 3.12: Harts A-frame mechanism: link and infeasible area as obtained for configuration 4, shown in figure 3.13a.

3.3.2. Geometrically feasible outlines

After obtaining the layer configurations, the link outlines and the infeasible areas, the geometrically feasible outlines can be obtained. Two examples are shown in figure 3.13. In the first example, shown in figure 3.13a, the ground link has adapted its shape to the infeasible area shown before in figure 3.11b and figure 3.12a. This happens because the ground is in the same layer as the sixth body. In figure 3.13b the ground again adapts its shape, but now for the swept area of the fourth body, as they are now in the same layer. Additionally, the sixth body adapts its shape to the swept area of the joint connecting body 3 and 5, because the joint passes through the layer of the sixth body.



(a) Configuration for which the shape of the ground is changed.



(b) Configuration for which the shape of the ground and the sixth body are changed.

Figure 3.13: Possible configurations for the A-frame mechanism

3.3.3. Best configuration

Multiple fitness functions can be used to obtain the best configuration. One possible goal can be to obtain the configuration with the least number of layers, resulting in the thinnest mechanism. Another goal can be to minimize the length of the pin joints, which leads to fewer problems with skewed axes due to bigger moments. Another option is to obtain the configuration with the least number of adapted links, or similarly, to aim for the configuration where the link surface area has changed the least. This can be calculated with equation 3.4:

$$A_{\text{change}} = \sum_{i=1}^{n} \left| \frac{A_{\text{new,i}}}{A_{\text{old,i}}} - 1 \right| \tag{3.4}$$

with $A_{\text{new,i}}$ [m^2] the area of the geometrically feasible outline of link i and $A_{\text{old,i}}$ [m^2] the area of link outline i before any changes. A_{change} [-] is the sum of the percentage of change over all links.

The fitness function given in equation 3.5 is a combination of multiple fitness goals, and is used to obtain the best configuration. It depends on the layer assignment vector β, the number of pierced layers by long joints and the total changed area of links. This fitness function gives the mechanism configuration shown in figure 3.13a a better score than the mechanism configuration shown in 3.13b.

$$f = \max(\beta) + \text{nPiercingJoints} + A_{\text{change}} \tag{3.5}$$

3.4. Conclusion

In summary, we find that Sen et al. [33] has found a method to evaluate kinematic designs for feasibility. To do this, he first obtains all structurally feasible layer assignments. Next, the kinematic feasibility is evaluated, after which the links can be adapted to obtain a geometrically feasible mechanism. These steps have been implemented in this research. Additionally, an implementation to obtain the best configuration from all feasible configurations is given.

4 Manufacturability extension

Adding springs to a rigid body mechanism changes their behavior. With springs, the mechanism is not solely designed to follow a kinematic path, but the spring forces are also incorporated into its behavior. For example, springs can be used to tune the natural dynamics of a mechanism, thereby minimizing energy consumption [13] or springs can be used to obtain energy free systems. Energy free systems can move quasi-statically without operating energy, since they have a constant potential energy over a desired range of motion [11]. A few examples are gravity compensation [3], vibration isolation systems [23] and statically balanced parallel robots [22, 39].

This chapter focusses on the manufacturability for mechanisms with spring elements by extending the theory from chapter 3. First, for the structural synthesis, the possible layer assignments for mechanisms with springs is described. Next, for the geometric synthesis, the swept area of springs is obtained and lastly, a new rule for feasibility is introduced.

4.1. Structural synthesis

To fulfill the structural synthesis for mechanisms including springs, two additions are made. Firstly, the layer assignment is expanded to include springs next to links. Secondly, the layer assignment algorithm is adjusted to evaluate more possible spring layer assignments.

4.1.1. Layer assignment vector

The layer assignment vector originally captures in which layer a link is situated. In the layer assignment vector β, also seen in figure 3.4b, Sen et al. only deals with links, which take up space a single layer, while disregarding joints, which connect elements between different layers. When adding springs and prismatic joints, they can not be handled the same way as links nor joints. Mechanism graph theory as used by Kuppens would describe them as a joint, but since they are also structural elements which takes up space in a layer, this space can not be disregarded.

Springs should be placed in a layer and the area they sweep during their movement should be free outside infeasible areas. Thus, they should obtain a place in the layer stacking algorithm (partly described in Ch. 4.2.3). However, to connect the springs to links in different layers a connection point is also necessary. If the spring rotates in any form, this connection should rotate so the spring can always have a straight connection between both connection points. Thus, the spring element is represented in two parts:

1. The spring link - taking up space in a single layer
2. The spring hinges - the connection points of the spring with elements in other layers

If one would like to evaluate prismatic joints for manufacturability, they have to be changed in a similar way. Prismatic joints are not just static joints which connect elements between layers. They also take up space in a layer, and thus should be taken into account when obtaining the layer information of a mechanism.

To conclude, the layer assignment vector needs to be adjusted to include elements which take up space in a layer. From here on, these elements are referred to as members. Previously the vector had a size equal to the number of rigid bodies in the mechanism, but the proposed change increases this by the number of springs and prismatic joints.

4.1.2. Layer assignment algorithm

The layer assignment method is initially created by Sen et al. [33]. All possible relative layer assignment are achieved with the directed adjacency matrix **L**. This matrix stores whether links are higher or lower than connected links. This is used to create possible layer assignments, but not all possible layer assignments are found.

A new method for the layer assignment is necessary because the current method does not work for mechanisms with springs, when both ends of a spring are connected to the same links as another member. In other words, when the adjacent links of the spring and the member are the same. We refer to this as members being kinematically parallel to each other. When members are kinematically parallel, it is possible that the members are placed in the same layer.

When kinematically parallel rigid bodies are situated at the same side of their kinematically parallel body, the body and their hinges interfere with each other even if one of them is put in a different layer. When a spring and a body are initially in the same layer they are often also kinematically infeasible. However, when changing the layer the spring resides in, the configuration can become kinematically feasible.

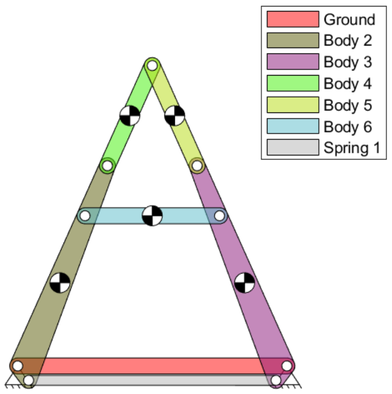


Figure 4.1: Harts A-frame mechanism [10] with spring. The spring is shown as a grey outline without CoM sign.

Take the mechanism in figure 4.1 for example. The spring is is kinematically parallel to the ground and the sixth body. Coincidentally, these links are also geometrically parallel. With the current method, links are assigned to the lowest layer index while adhering to the directed adjacency matrix **L**. This means that the spring is assigned to the same layer as the kinematically parallel bodies. When they are in the same layer, they interfere, but if the spring is put in another layer, they do not interfere.

A solution to this problem is shown in figure 4.2. The spring is put in layer 1, and the rest of the mechanism is moved above it, resulting in a manufacturable mechanism (See chapter 6 Prototypes).

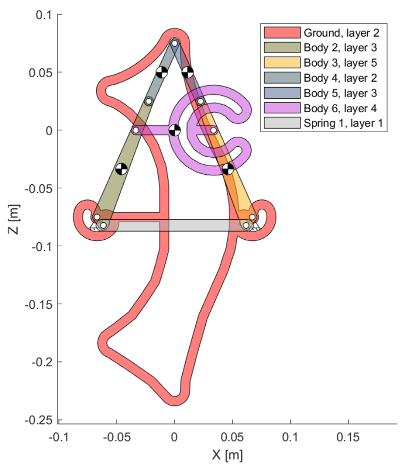


Figure 4.2: Manufacturable A-frame spring mechanism (with spring shown as a grey outline)

This method is captured in the extended method, which for every link assignment finds additional structurally possible spring layer permutations. To be structurally feasible, the directed adjacency matrix **L** created in chapter 3.1 is adhered to. This can be captured with the following rule:

To obtain additional structurally feasible assignments, the originally obtained assignments must be re-evaulated one-by-one. New assignments should be created for every original assignment, in which the spring is moved to different layers. The spring is moved to all layers above, below, or between adjacent links to obtain the possible alterations. When there are multiple springs in a mechanism, this task is done recursively.

This rule can be translated into the following algorithm which creates the spring layer configuration, using these steps for spring i when given configuration[1] a:

Algorithm 1 Create spring layer configuration

```
1:  Get layer indices of adjacent layers
2:  Create possible new layer indices for spring
3:  if Spring is below adjacent layers then
4:      Possible layer indices reach from current index up to the minimum layer - 1
5:  else if Spring is above adjacent layers then
6:      Possible layer indices reach from current index up to the maximum layer + 1
7:  else if Spring is between adjacent layers then
8:      Possible layer indices reach from the lowest adjacent layer index + 1 up to the highest adjacent layer
        index - 1
9:  end if
10: Create new configuration from every possible new layer index, leading to $a_1$ ... $a_n$
11: If currently evaluated spring is not the last spring, evaluate the newly created layer configuration $a_k$ from
    line 1 for spring $i+1$
12: Remove all duplicate layer configurations
13: return Remaining layer configurations
```

This method ensures a multitude of new structurally feasible configurations are tested for kinematic feasibility. However, this method is not exhaustive. Currently, the method only changes the layer in which a spring resides, but to be complete, the layer in which other members reside should also be changed.

For example, using the current method on the A-frame spring mechanism in figure 4.1, a possible solution results in the spring being placed in the lowest layer (layer 1), beneath the red ground link (layer 2) - which is usually situated in the lowest layer - and the rest of the mechanism, as seen in figure 4.2. However, it does not explore the option in which the red link is the lowest layer (layer 1), and the spring is placed one layer higher (layer 2).

This complete method is not explored, since it increases the total number of configurations very quickly. Currently, a 9-jointed mechanism such as the A-frame spring mechanism results in 256 layer assignments, as calculated with equation 3.1, of which 34 are structurally feasible. With the current method, 29 additional structurally feasible configurations are created to accommodate for a single spring.

However, when two springs are used with the same A-frame mechanism as a base, 1024 layer assignments are found, of which 65 are structurally feasible. With these 65 assignments, 120 additional variations are obtained to account for the springs in the mechanism.

To conclude, the number of configurations created will quickly increase when evaluating more members. This is due to the recursive nature of this method: the number of variations found increases by a power of 2 ($\mathcal{O}(2^n)$) for the number of springs (or members) considered in this method, for every structurally feasible assignment.

4.2. Geometric synthesis

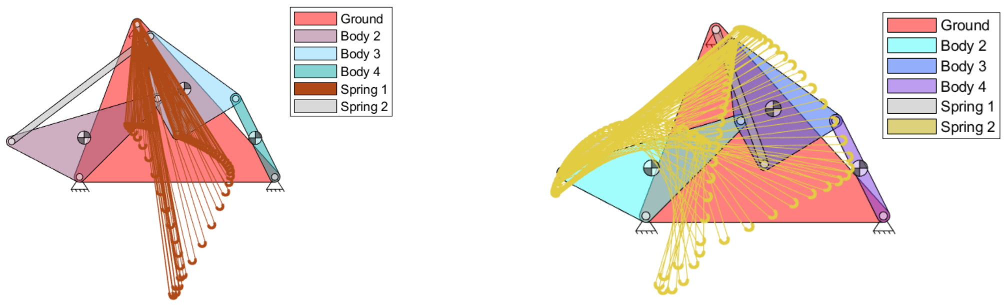
A change and two additions are implemented in the geometric synthesis, which will be discussed in this section. Firstly, with respect to the original geometrical synthesis as outlined in section 3.2, the swept area of springs is used differently. Secondly, the additional method to obtain the geometric feasibility of a spring is given. Lastly, to prevent broken configurations in which multiple link outlines adapt to each other, another additional method is introduced.

[1]Configuration and assignment can be used interchangeably

4.2.1. Swept area spring

The method proposed by Sen et al. to obtain the swept areas relative to a body is used as a basis to obtain the swept area relative to springs. The definition for a swept area is given by Sen et al. [33]: *"When a planar object undergoes motion, the totality of all the locations occupied by the points on the object is called its swept area for the given motion"*.

The swept area of a spring is obtained in the same way, with the positions of the springs throughout time shown in figure 4.3. With the methods discussed in chapter 3.2.1, we can combine the positions to produce the swept area. However, we can already see springs do not move in the same way as rigid bodies do: the length of springs changes.



(a) Positions of spring 1 over time (b) Positions of spring 2 over time

Figure 4.3: Roberts mechanism with two springs and their positions over time

To obtain the infeasible area of elements in a layer with a spring which is based on the relative swept area [2], the CoM of the spring is set at a fixed location: at the center of the spring, the same as for a body. This fixed location ensures that the CoM is always at the right location and orientation. Now the links move respectively to that CoM to create the relative swept area. However, with a spring, solely fixing the CoM is not sufficient. Since the spring length is not static, the distance from the center of the spring ends change with time. When taking the swept area of spring hinges relative to their spring center, this spans the entire area where the spring hinges have moved, as shown in figure 4.4c.

If that path is compared to the relative swept area of other hinges and they overlap, there is no guarantee that the path of the spring is interference-free. This is due to the fact that the spring hinges are not at all locations of the swept area at all times - which would be the case when evaluating the situation shown in figure 4.4c for kinematic infeasibilities for spring 2. On the contrary, a spring hinge is at only one position at a certain time, encompassing only a certain part of the red oval in the original swept area of figure 4.4c, as shown in figure 4.5a and enlarged in figure 4.5b.

A layer configuration in which a hinge overlaps with the original swept area of a spring hinge could be kinematically feasible, as long as the spring hinge is currently not in the same location as the other hinge. Therefore, we require a method which can compare swept areas relative to spring hinges as if the spring hinges are always in the same location.

Solution

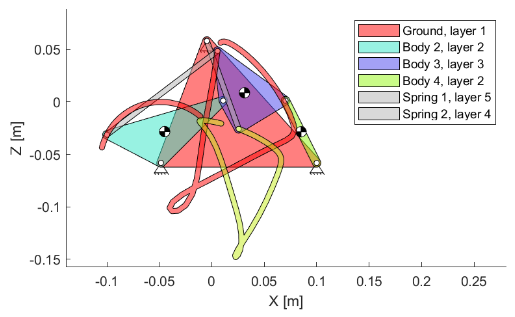
To mitigate this issue, a solution has been found. When obtaining the inverted swept area, an extra step is necessary where the swept areas are scaled:

1. Translate the fixed springs CoM to coordinates (0,0) and translate the infeasible area likewise.
2. Rotate the fixed spring to $\theta = 0$ and rotate the infeasible area by the same angle.
3. Scale the fixed spring over the x-axis so that the center of the spring ends are always at the same location. Scale infeasible area by the same ratio.

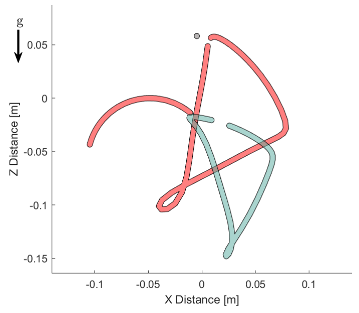
Step 1 and 2 are unchanged from equation 3.3, whereas the third step is the new theory, creating the solution.

Previously the relative swept area of spring hinges would be at varying locations as shown in figure 4.4c. With this added scaling step, the spring hinges are centered at one location (Fig. 4.4d). Looking at an enlarged hinge in figure 4.5d), the outer shape becomes a flatter oval and the inner shapes are scaled into less wide ovals.

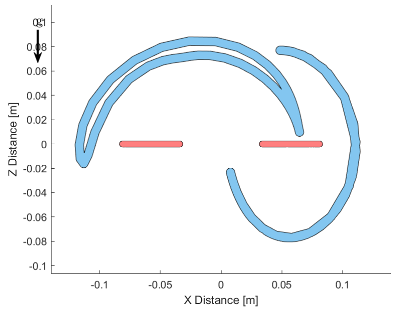
[2]Relative swept area and inverted swept area can be used interchangeably



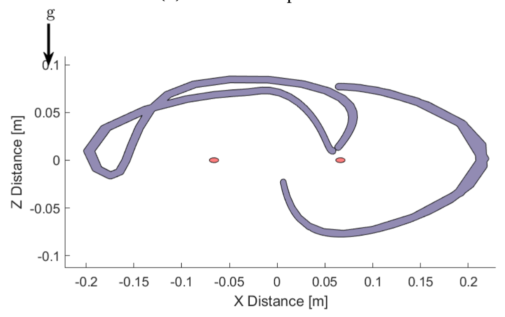
(a) Roberts mechanism and absolute swept areas



(b) Absolute swept areas



(c) Original method for relative swept areas, in which the hinges of spring 2 are spread out over a range of locations



(d) New method for relative swept areas, in which the hinges of spring 2 are centered around a single location

Figure 4.4: Swept area of the hinges in layer 4 of the Roberts mechanism in figure 4.4a. The swept area of the hinges of spring 2 are red, the hinges of spring 1 passing through layer 4 are in the other colors.

This scaled shape has a downside. Situations occur where the algorithm thinks that an infeasible area overlaps a hinge, while it does not. This happens when an infeasible area partly overlaps with the combined hinge outline due to scaling. Effectively, the algorithm thinks the hinge is not at an appropriate distance from the infeasible area. This lack of distance results in a geometric infeasibility. This would also happen in the unscaled, original method.

There is a work around to prevent this. Instead of combining the discrete steps to form one outline, every time step is evaluated separately. However, this results in two other downsides: the increase of required computing effort - by a factor of the number of time steps and when the discrete outlines are not combined as discussed in section 3.3, the gaps between discrete points are not taken into account. These gaps are visible in figure 4.5c between the unconnected grey or gold circles.

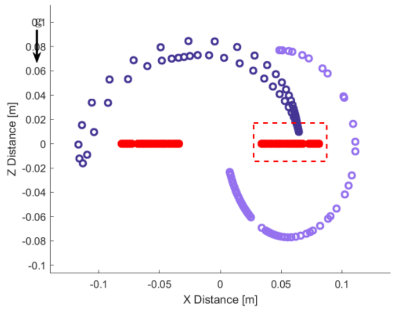
4.2.2. Geometric feasibility spring

Next, the geometric feasibility is evaluated as well. As with the swept area calculations, this step is done different for springs than for bodies. To check for geometric feasibilities, the convex hull (nominal geometry) of the two spring ends is taken and a buffer around this area is added to account for material allowances (Fig. 4.6). This buffered hull should encompass more than the whole region where the spring can be located to create a safe margin. If any other joint in this layer is in this area, the spring can not be created and the configuration is geometrically infeasible.

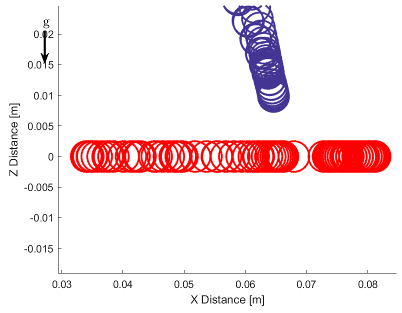
An example of a geometric infeasibility for a spring is shown in figure 4.6a, where other hinges intersect the convex hull. Opposingly, an example of a geometric feasibility for a spring is shown in figure 4.6b, where other hinges do not intersect the convex hull. From these figures we can conclude that the hinges of spring 1 (purple swept area in figure 4.6b) do not interfere with the convex area of the hinges of spring 2. The other way around, this is not true. Thus, the manufacturability method chooses to put spring 1 in layer 5 (with its hinges passing through layer 4), and spring 2 in layer 4 (Fig. 4.4a), which is a feasible configuration.

For links, if the swept area of another link intersects a fixed link, the fixed link needs to adapt its shape to allow for the intersecting swept area to pass through. That same comparison is also done the other way around: take the swept area of the fixed link and compare it to the fixed outline of the previously swept link.

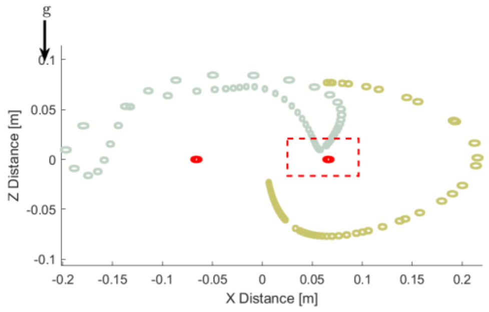
For springs this is different, as they can not adapt to an intersecting swept outline. Thus, when an outline intersects the springs outline, the spring does not change, but the other outline has to adapt to the springs outline. Since only a single link is changed in this case, this also does not lead to doubly adapted links, which



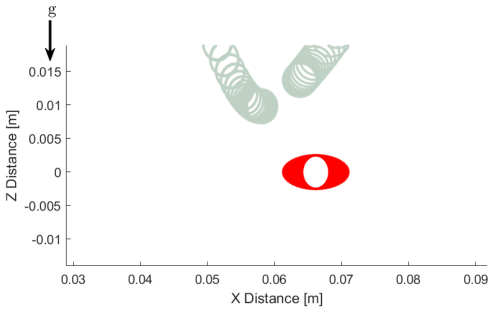
(a) Overview of the spring hinge outlines generated with the original relative method



(b) Detailed view of figure 4.5a

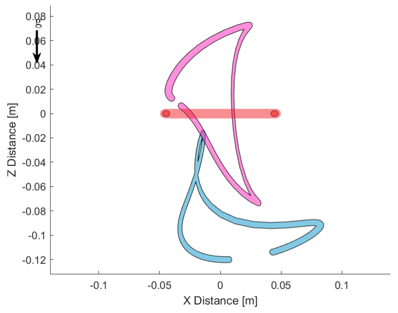


(c) Overview of the spring hinge outlines generated with the new relative method

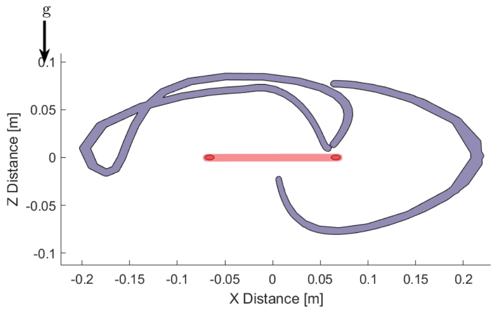


(d) Detailed view of figure 4.5c

Figure 4.5: Outlines of spring hinges drawn at every generated location relative to spring 2. Hinges of the spring 2 are drawn in red.



(a) Spring 2 hinges (pink and blue) relative to spring 1 (red)



(b) Spring 1 hinges (purple) relative to spring 2 (red)

Figure 4.6: Relative swept areas spring hinges in the Roberts mechanism from figure 4.4a

will be discussed in the following section.

4.2.3. Interference of multiple changed links per layer

To create the geometrically feasible outlines, link shapes are adapted to the infeasible area as described in chapter 3.2. In previous work [33], when multiple links in a layer were interfering with each other, that could cause the outline of more than one link to be changed, as shown in figure 4.7a.

This occurs in the method used by Sen et al. [33], since all links are studied one-by-one when creating the possible geometries. This results in the unwanted first situation listed below. The second situation is an incomplete solution to the first situation. Finally, a solution is given to solve the problem of interference of multiple changed links in the same layer.

Situation 1 (Fig. 4.7a): The infeasible areas created by other links in a layer stem from the initial outline of a link. This works for configurations in which links in a layer do not interfere with each other. However when they do interfere with each other, both links obtain a new outline to deal with the infeasible area created by the initial shape of the other link. Naturally, when both links are changed, they may interfere with each other again, but now with the changed shapes, like in figure 4.7a.

Situation 2 (Fig. 4.7b): The infeasible areas created by other links in a layer are created from the initial

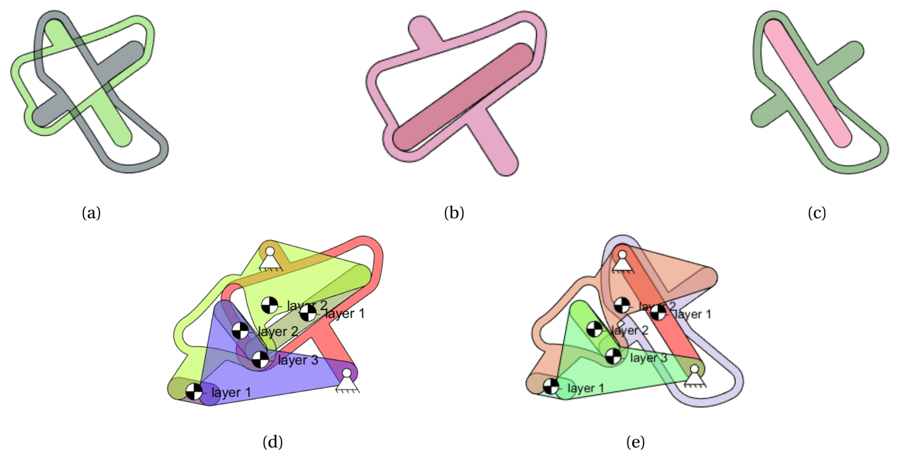


Figure 4.7: Doubly adapted configuration (a) and the resulting fixed configurations (b, c). Configuration within the full mechanisms (d, e)

outline of a link. When encountering this situation, the first link is changed, while the other interfering link is left unadjusted, like in figure 4.7b, resulting in the mechanism shown in figure 4.7d.

This leads to a feasible design in simple cases, but with multiple links in a layer it could behave unexpectedly. Another reason to not use this method, is that not all possible configurations are explored. For example, the solution shown in figure 4.7c, resulting in the mechanism shown in figure 4.7e, is not found. It could be that a configuration which depends on changing the second (or later) link is better than the configuration found by changing the first link.

Solution: At first follow situation 1: create the new outlines from the initial infeasible areas. After obtaining all configurations in this way, check every configuration for layers with multiple changed links .

If there are multiple changed links in a layer, copy the current configuration $2^{n_{\text{links changed}}}$ times. The new copies are adjusted by reverting the changed links to their original shapes. To examine all possible combinations of old and changed links, all $2^{n_{\text{links changed}}}$ new options are necessary.

Finally, check if the current configurations are still geometrically feasible. If a change to a design occurs because of an interfering infeasible area, render that design infeasible. With this method all possible configurations are explored and no feasible options are overlooked.

A downside to this alteration is that additional calculations are introduced. The processing power is increased by an order of $\mathcal{O}(2n)$ for every layer in which multiple links are changed. Although costly, this increase in cost is necessary to obtain a feasible design.

4.3. Conclusion

To conclude, in this chapter we add extensions to the structural and geometric synthesis. Firstly, we add spring elements to the layer assignment vector, after which we obtain more layer assignment vectors. These new assignments are necessary, since not all options are explored. In the geometric synthesis, we obtain the swept area of elements relative to springs by scaling, which minimizes the possibility of false overlaps. We also introduce a new method to evaluate the geometric feasibility of springs and finally a method which deals with multiple changed links in the same layer.

5
Engine extension

The pre-existing engine handles kinematic designs, but to ensure mechanisms are manufacturable, additional features are required. To evaluate the manufacturability of rigid body spring mechanisms, its elements need to be represented in a member-hinge matrix. Additionally, a mechanism is not built in a single plane and thus requires a representation for a physical model. Finally, the spring optimization used in chapter 2.4 is adjusted to allow for the use of off-the-shelve springs.

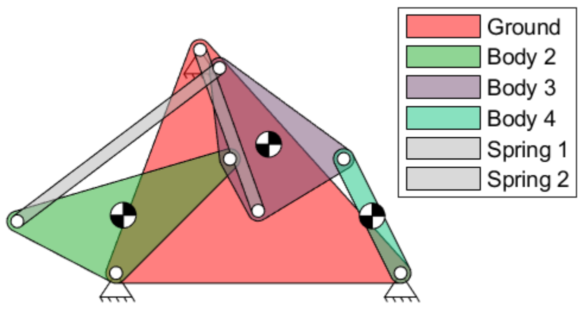


Figure 5.1: Roberts mechanism with springs

5.1. Representation

In previous work on manufacturability [33], the only elements used were links and revolute joints. Links are rigid constructional elements, also referred to as bodies, and revolute joints are also known as hinges. In this work, springs are introduced and in future work prismatic joints can also be introduced. The blanket term used in this research for rigid bodies and springs is members. To incorporate these into a design, we need to examine how members and joints can fit into the current representation.

5.1.1. Member-hinge matrix

Link-joint matrices are incidence matrices with links in the rows and the joints in the columns. This matrix is used to obtain feasible geometries by obtaining the swept areas from the links in the rows and the swept areas of the joints in the columns. Originally, link-joint matrices are used only with rigid bodies and revolute joints. A member-hinge matrix is the replacement for the link-joint matrix. The member-hinge matrix has the ability to account for springs and makes use of an unambiguous term to describe the type of joint. Each hinge in this matrix represents a pin, around which two members are rotating. Incidence matrices in the automatic creation of mechanisms by Kuppens [20] have links in the rows and hinges, springs and prismatic joints in the columns.

Where Sen et al. currently do not allow for springs, Kuppens and Wolfslag do, as shown in table 5.1. Even though both methods represent the links and joints with graph theory, we need to differentiate in how it is used. Sen et al. uses graph theory to create a feasible geometry and Kuppens and Wolfslag to create a kinematic design.

	H1	H2	H3	H4	S1	S2
Ground	1	0	0	1	1	0
Body 2	1	1	0	0	0	1
Body 3	0	1	1	0	1	1
Body 4	0	0	1	1	0	0

Table 5.1: Kuppens incidence matrix of Roberts mechanism

Thus, we want to convert Kuppens incidence matrix to a form in which it can be used to obtain feasible geometries. For this manufacturability, all elements which are situated within a single layer (members) should be in the rows while all elements passing through multiple layers (hinges) should be in the columns.

To do this, a single spring from Kuppens incidence matrix is transformed into two elements in the member-hinge matrix. The spring itself lays within a single layer, thus is added as a row entry. Next, the spring has two ends, which are connected to a link. These spring ends will be referred to as spring hinges and in the member-hinge matrix with $S_{a,b}$, with a being the index of the spring and b being the index of the end of the spring.

For example, take a look at the Roberts spring mechanism (Fig. 5.1). The incidence matrix in the notation of Kuppens and Wolfslag (Tab. 5.1) and the new member-hinge incidence matrix (Tab. 5.2) are given. In the new matrix, two rows are added to accommodate for the springs and the two columns previously used to describe the spring have been replaced with four columns describing the incidence of the hinges of the two springs.

	H1	H2	H3	H4	S1,1	S1,2	S2,1	S2,2
Ground	1	0	0	1	1	0	0	0
Body 2	1	1	0	0	0	0	1	0
Body 3	0	1	1	0	0	1	0	1
Body 4	0	0	1	1	0	0	0	0
Spring 1	0	0	0	0	1	1	0	0
Spring 2	0	0	0	0	0	0	1	1

Table 5.2: Member-hinge incidence matrix of Roberts mechanism

5.2. Physical parameters

A set of design parameters are required to transform the mechanism from kinematic design to manufacturable design. These parameters are used to obtain physical quantities such as the link weight, the link outline, but also the necessary pin length which needs to be exported to the Bill of Materials (see App. B.3.2). In this section the parameters are explained after outlining the prototype design options.

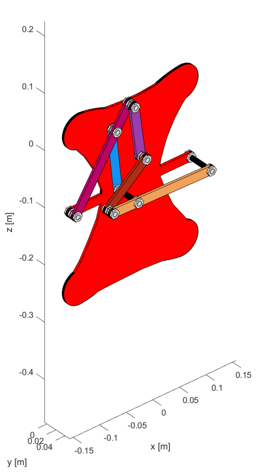
5.2.1. Prototype design options

With the prototypes, we explore two manufacturing options. In the first option, the axes are made of threaded rods and locknuts are used to set the links at the correct height. An example of this is discussed in chapter 6 and a render of a manufacturable mechanism is shown in figure 5.2.

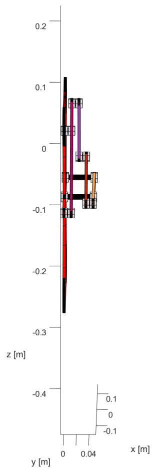
We initially chose to follow this path as it adheres to the spirit of this research: obtaining a prototype after an optimization as quickly as possible. This method only uses standard, easily accessible parts, which can be cut to length with a saw. Next to laser cutting the links, they can also be 3D printed or (manually) cut from wood. This allows engineers, even with minimal access to specialized tools to quickly prototype a mechanism.

The second option we explore is more compact, but requires more resourcefulness. This option makes use of round rods, onto which the members are clamped in position with circlips and nylon washers, as shown in figure 5.3. Nylon washers are used to reduce friction between members and circlips.

To attach the circlips to the round rods, slots are made using a lathe. The circlips are not as common as locknuts, which, next to requiring a lathe, makes this prototyping method less accessible than the first option.



(a) Isometric view



(b) Side view

Figure 5.2: Harts A-frame, manufacturable mechanism clamped with nuts. The gap in the ground (red shape) is not shown correctly in this 3D view.

5.2.2. Parameters

Table 5.3: General parameters

Attribute	**Type**	**Default**	**Description**
Htype	string	’circlip’	Choose between ’circlip’ and ’locknut’. If configured, chooses the relevant nut thickness and weight.
pinDiameter	number	4e-3 m	Diameter of the pins acting as axes.
thicknessClearance	number	1e-3 m	How much space should be between elements stacked on top of each other (In figure 5.3 shown with clearance (E), the space between both circlips).
jointClearance	number	1e-3 m	How much space should be between two joints in the same layer. This is used for pins (rod piercing layers), nuts (to fix links to a layer) and spring attachments (to attach springs to links).

The general parameters, given in table 5.3, are the most important parameters. The pinDiameter is used directly to obtain the swept area of hinges passing next to each other.

Table 5.4: Density parameters

Attribute	**Type**	**Default**	**Description**
densityPin	number	7820 kg/m3	Material density of the pins acting as axes.
densityLink	number	7820 kg/m3	Material density of the links used for rigid bodies.

The density parameters are configured in table 5.4. The densities are used to obtain the correct weight of links and pins. Since the weight of pins is not included (yet) by the engine in the dynamical analysis, a workaround is created. The weight of half the pin is added to each of the links it is attached to. Subsequently, all these new masses are added to the mass of the link to find the total mass and the combined center of mass is also calculated.

In an effort to make it easier to use the engine, a method using the parameters of table 5.5 is created to add a toolhead (see appendix A.2). This toolhead changes the mechanism in such a way that (1) the mechanism is forced to create a shape at a certain point, making it easier to create link shapes such as a Roberts mechanism - in which the link shape is not the convex shape between connected joints -, (2) a point mass is added to the dynamical evaluation and (3) when the manufacturing files are created mounting holes are already in place.

The body parameters listed in table 5.6 are used to create the outlines for the rigid body elements. These are ’GM’ parameters, which stand for Ground and Mass, which is what these elements are defined with in the code.

The hinge parameters listed in table 5.7 are chosen according to the chosen Htype. The default option is configured for circlips, for which all data is extracted from DIN6799. Washer information is extracted from

Table 5.5: Toolhead parameters

Attribute	**Type**	**Default**	**Description**
TooliL	integer	[]	Index of member which is a tool, this index is added to GMLocked.
ToolDiameter	number	40e-3 m	Diameter around tool anchor point, which encompasses the tool body.

Table 5.6: Body parameters

Attribute	**Type**	**Default**	**Description**
GMFileName	string	[]	When importing outlines, the file name which contains points for a rigid body.
GMLocked	boolean 1xnGM	[]	Index of rigid bodies of which the outlines can not be changed from outlineInitial.
GMthickness	number	4e-3 m	Thickness of a rigid body (e.g. thickness of a perspex plate).
GMendDiameter	number	8e-3 m	Diameter of the ends of links. Is the same as the minimal link width.
GMjointDiameter	number	4e-3 m	Diameter for hole through which the pins go. This number can be increased to allow for the use of bushings or bearings.

Table 5.7: Hinge parameters, configured for circlips (variable name is still nut) in a 3.2mm groove according to DIN6799

Attribute	**Type**	**Default**	**Description**
HnutDiameter	number	9.3e-3 m	Nut outer diameter
Hthickness	number	0.6e-3 m	Nut thickness
Hthickness_before	number	0.9e-3 m	Washer thickness
Hthickness_after	number	1.2e-3 m	Clearance to end of pin
HnutMass	number	0.088e-3 m	Nut mass

DIN125. The information for the locknuts is extracted from DIN985.

The parameters are perhaps more easily explained by looking at figure 5.3. HnutDiameter is the largest diameter of either the washer (C) or the nut (or circlip (B) in this example). Hthickness is the height of the nut (B). Hthickness_before describes the height of the washer (C). Hthickness_after is the required clearance at the end of the pin (A). HnutMass is used in the hacky method to calculate the link mass.

Finally, the spring parameters listed in table 5.8 are configured. One would expect Sdiameter and Sthickness to be the same. However, if one does not choose to attach the spring to the hinge with a spring end loop, but an alternative device (e.g. Fig. 5.4), one may alter this height.

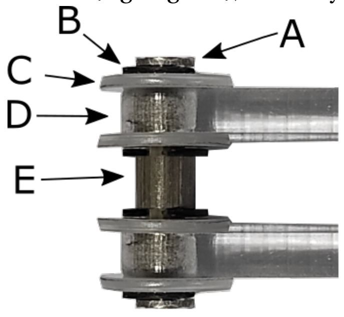


Figure 5.3: Pin (A), circlip (B), washer (C), perspex link (D), and the clearance between the links (E)

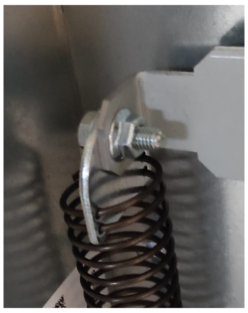


Figure 5.4: Spring attachment

Table 5.8: Spring parameters

Attribute	Type	Default	Description
Sdiameter	number	5e-3 m	Diameter of spring used
Sthickness	number	10e-3 m	Thickness of spring in layer (probably the size of the hinge ends - where it attaches to a pin)

5.3. Spring optimization

One of the many mechanism optimization methods available, is spring optimization towards gravitationally balanced mechanisms. To make sure this method, as initially created by Kuppens, also produces manufacturable results, we need to change it. Ideally, this optimization converges towards a design which is 1) geometrically feasible and 2) uses off-the-shelve springs.

In this section the process of including off-the-shelve springs in the optimization is described. To do this, we first take four off-the-shelve springs and test their force-extension behavior. Next, we analyse the calculation of energy of the current model and how it should be adapted to allow for the tested springs. Then, new constraints are added to increase the chance of manufacturability. Lastly, we show some mechanisms created with this optimization method.

5.3.1. Off-the-shelve tests

Using off-the-shelve springs in an optimization situation is not trivial. The dynamic properties of springs depend on three main characteristics: the free length, the spring stiffness and the pre-tension. When using off-the-shelve springs from a catalogue of discrete springs, these variables are discrete values. However, the used interior-point method requires continuous variables for quicker optimization.

Thus, for manufacturing we use springs which are not hindered by these discrete steps. We bought four springs of 500 mm length. These springs can be used to simplify the optimization method by varying the initial length to obtain different spring stiffness coefficients.

To test spring behavior when it is cut into multiple elements, the four springs are tested in a tensile bench. Each spring is tested at its initial length of 500 mm, at 400 mm and at 100 mm. We expect the springs to adhere to Hooke's law after an initial phase in which a preload needs to be overcome. A graph is depicting the test result of one spring type and the new approximation is shown in figure 5.5, the test and approximation results can be found in appendix A.4.

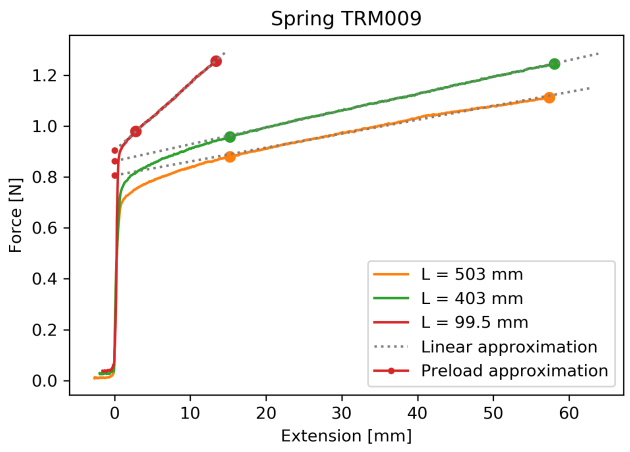


Figure 5.5: TRM009

When analyzing these results (Tab. 5.9), we found that the equation to calculate spring stiffness based on their length, derived from Hooke's law, holds true:

$$k = k_{500} \cdot \frac{0.500}{L_s} \tag{5.1}$$

where k is the spring stiffness of the cut spring [N/m], k_{500} is the spring stiffness of the spring at 500 mm, and L_s is the length to which it is cut. Thus, if the spring is cut shorter, its stiffness increases.

Table 5.9: Calculated spring stiffness TRM009

Spring type	**Length [mm]**	**Stiffness [N/mm]**	**Stiffness 500mm [N/mm]**	**Preload Fv [N]**
TRM009	503	0.0055	0.0055	0.806
TRM009	403	0.0066	0.0053	0.8622
TRM009	99.5	0.0263	0.0052	0.9042

In these calculations the springs behavior is simplified. We approach the spring stiffness as a constant, with a fixed preload for every spring type - independent of its length. First, the spring constant is obtained with a linear approximation of the spring force-extension behavior between 30% and 100% of the measured spring extension.

After this, the preload has to be approximated. The preload is the force required to stretch the springs from zero extension to a bit of elongation, seen in figure 5.5 as the sharp increase in force. In our simplified approach, we model this preload as constant for any length of spring of the same type. We use the average preload from the preload approximations determined in table 5.9.

To conclude, our approach needs to differ from the original spring force equation 2.11, since the preload has to be included in this equation.

5.3.2. Calculation

From the original spring energy equation, as given in eq. 2.11, we can derive spring force equation 5.2, with k as spring stiffness, ΔP the distance between spring ends and zero-length L_0:

$$F_s(x_j) = k(\Delta P - L_0) \tag{5.2}$$

As discussed in the previous section, this still lacks the preload F_v we find in off-the-shelve springs. That is why we use a new spring force equation 5.3, resulting in the spring energy equation 5.4. These new equations are used in the optimization towards gravity balanced mechanisms.

$$F_s(x_j) = k(\Delta P - L_0) + F_v \tag{5.3}$$

$$E_s(x_j) = \frac{1}{2}k(\Delta P - L_0)^2 + F_v \cdot (\Delta P - L_0) \tag{5.4}$$

5.3.3. Constraints

One of the variables in the optimization is the location of the spring ends. With the theory of chapter 3, we know that a mechanism is kinematically infeasible if hinges connected to the same link are too close together. Thus, we have to introduce a new constraint function.

The constraint is necessary to make sure the infeasible area around a hinge does not overlap with the area used by another hinge in the same layer. Thus, we have to create a buffer around other hinges in the same layer to constrain the optimization.

$$c = -\left((x_s - x_h)^2 + (y_s - y_h)^2 - r_h^2\right), \quad \text{with } c \leq 0 \tag{5.5}$$

In equation 5.5; x_s and y_s gives the location of the spring hinge and x_h and y_h gives the location of the adjacent hinge. r_h is the radius of the infeasible area around a hinge plus (optionally) the minimum width of a link. A constraint is created for every hinge connected to the same link.

5.3.4. Resulting designs

We start with the PHDNA of the A-frame and a single spring is generated in an SDNA. The spring is connected to body 2 and 3 and its location and stiffness are optimized as described in this chapter. In chronological steps, the previously shown figure 2.4a shows us the energies of the mechanism before optimizing, figure 2.4b shows the optimal spring placement and figure 5.6 shows the configuration which is kinematically feasible.

The mechanism in figure 5.6 differs from the mechanism in figure 2.4b in its spring hinge locations. The spring hinge locations were originally too close to hinges connected to adjacent links, thus triggering a kinematic infeasibility. To combat this, constraint equation 5.5 is added to the spring optimization method.

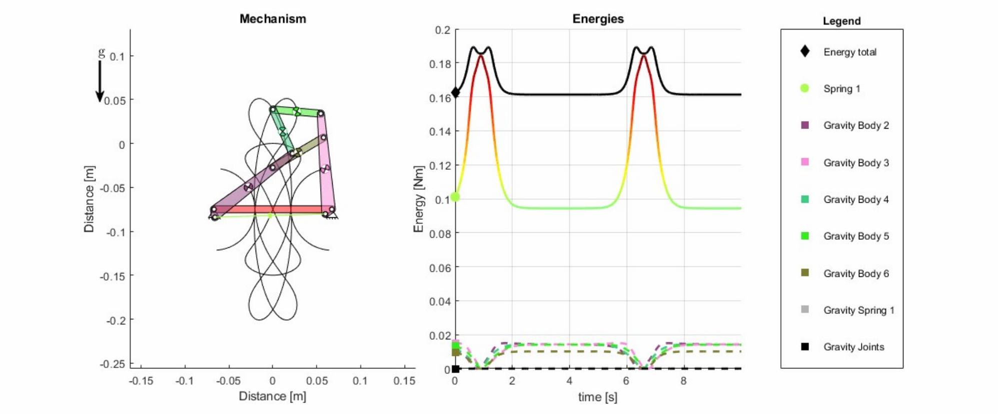


Figure 5.6: Harts A-frame with spring after optimizing with constraints, ressulting in a kinematically feasible gravitationally balanced mechanism

To conclude, we used the extended spring optimization algorithm to find a kinematic design which uses of an off-the-shelve spring. Next, the design is evaluated and adapted for manufacturability, so it can be prototyped, which will be discussed in the following chapter.

5.4. Conclusion

To conclude, in this chapter we extend the mechanism engine developed by Kuppens and Wolfslag [20]. We do this so that mechanisms represented with adapted graph theory can be evaluated and adapted for manufacturability. This is done with a newly developed member-hinge matrix, a member layer assignment vector and by adding thickness information to members.

Additionally, the spring optimization method is extended to use off-the-shelve springs when optimizing mechanisms towards spring-based gravity-balanced mechanisms. The springs are optimized such that a certain off-the-shelve spring has to be cut to length. This spring can now be directly attached to the optimized gravity-balanced mechanism.

6 Prototypes

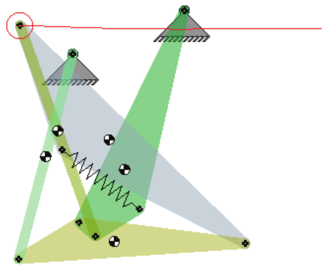
The final of the three research goals is: *"Investigate if automatically generated manufacturable mechanisms are manufacturable and interference free"*. Thus, this chapter discusses manufacturing and validating the manufacturable mechanisms. The manufacturing method and the freedom of movement is validated.

Two pre-existing designs and two new designs are chosen to be evaluated. First, two automatically evolved designs by Kuppens (shown in figure 6.1) are evaluated, as his work was the inspiration for the current research. Then, a Harts A-frame is constructed to evaluate if another construction method is suitable. At last, two Roberts mechanisms with additional springs are evaluated, showcasing the true value of the extended manufacturability method. One of the two Roberts mechanisms is manufactured without any adaptations and the other is manufactured such that less layers are required and in both mechanisms the two springs can move interference-free.

6.1. Automatically evolved mechanisms

Kuppens work [19, 20] was the inspiration for this research, therefore two kinematic designs are chosen from his work. A straight line approximating mechanism, as shown in figure 6.1a, is taken from [19] and an ellipse-tracing mechanism, ass shown in figure 6.1b, is taken from [20]. These are recreated using the adapted graph theory, simulated and evaluated and adapted for manufacturability.

These mechanisms were the initial inspiration for this thesis, as at first glance they seem simple enough to manufacture quickly and easily. However, when one starts to think of a solution for how the links should be arranged, it is harder than it looks. Using equation 3.1, we can calculate that there are at least 256 possible configurations for the straight line mechanism and 2048 possible configurations for the ellipse-tracing mechanism.


(a) Evolved straight line mechanism [19]

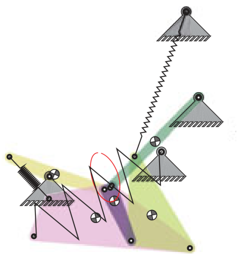
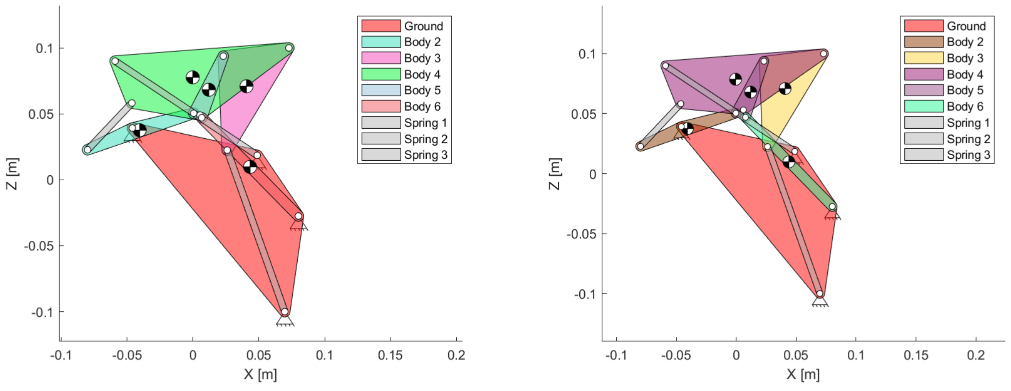

(b) Evolved ellipse-tracing mechanism [20]

Figure 6.1: Mechanisms automatically created by Kuppens

When both designs are tested for manufacturability, we find that the ellipse-tracing mechanism is not manufacturable. The joints circled by the red ellipse in figure 6.1b, also seen in figure 6.2a, overlap, causing a kinematic infeasibility.

Two possible solutions exist. First, one could take smaller joints, so that the joints do not overlap anymore. This should work fine, but for this mechanism the relative size difference between the joints and the mechanism would be too big. Secondly, we can adjust the joint location. This also changes the path the mechanism follows - for which it was initially evolved - so it should be avoided in general.



(a) Mechanism with joints at original location (b) Mechanism with joints moved from original location

Figure 6.2: Recreated ellipse-tracing mechanisms

In order to to further test the manufacturability of this mechanism, the joints are moved to make sure they do not overlap anymore (Fig. 6.2b). Even after this adaptation, the mechanism is still not manufacturable. There is no possible layer assignment which results in a geometrically feasible mechanism. This is likely due to the use of three springs, that cover a big area which is in the way of other elements. An example of a kinematic infeasibility for the third body is shown in figure 6.3. Parts of the infeasible area (shown in purple) are overlapping the area of the joints (shown in red), creating the kinematic infeasibility rendering this configuration infeasible and thus not manufacturable.

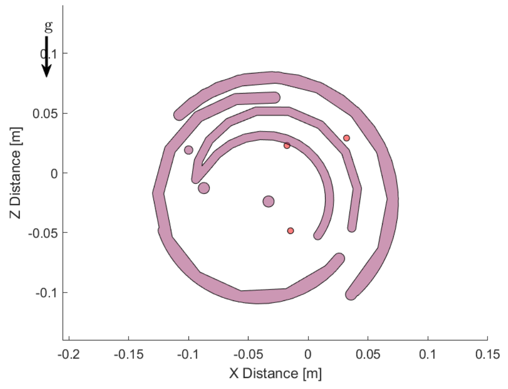


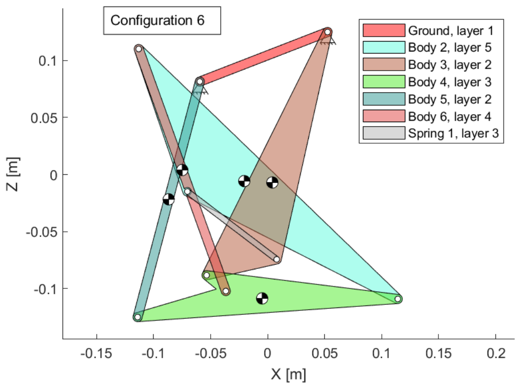
Figure 6.3: Area of the joints of body 3 in red, and their infeasible area in purple

Method

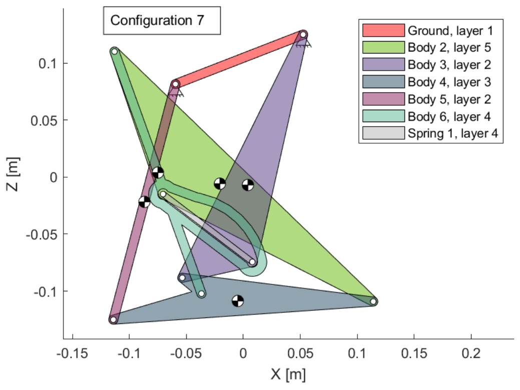
From the manufacturable mechanism shown in figure 6.1a, nine manufacturable designs are produced (see appendix C.1). Out of the nine options, the best configuration, shown in figure C.2, is automatically chosen using equation 3.5.

The chosen option looks very similar to the original design. However, the shape of one link was adapted to allow a joint to pass through and we know how to stack all members. Looking carefully at the sixth body in figure 6.4a, one can notice that there is a circular cutout at one of the spring ends. An alternative manufacturable configurations is shown in figure 6.4b. In this configuration, the fourth body adapts its shape to account for the swept are of the spring in the same layer.

The first design is chosen to be manufactured with plexiglass, washers, locknuts and threaded rod. To manufacture the manufacturable design, the mechanism outlines are converted into manufacturing drawings as described in appendix B.3.1. These manufacturing drawings are sent to the laser cutter, which cuts the link shapes from plexiglass.



(a) Best configuration



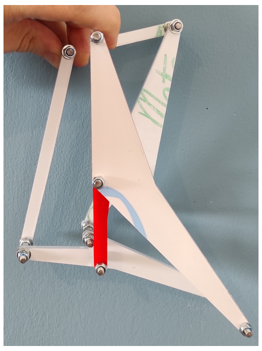
(b) Alternative configuration in which body 4 adapts its shape for the swept area of the spring

Figure 6.4: Possible configurations

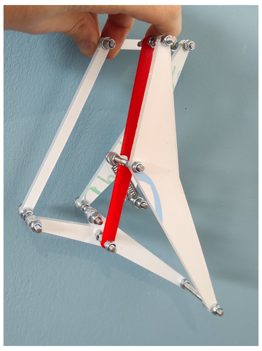
The threaded rod acting as axes then needs to be cut to size by hand. The required element lengths are automatically generated to the Bill of Materials (See also app. B.3.2) and presented to the user in a table. This table also gives the user the number of required locknuts and washers.

Results

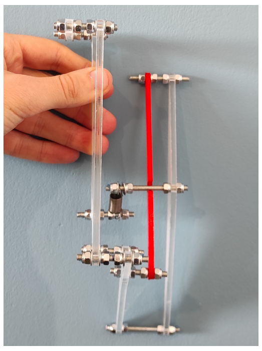
The manufactured mechanism is shown in figure 6.5. It shows the mechanism with white plexiglass links and a single red 3D printed element.



(a) Front



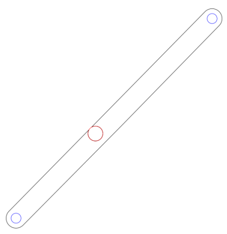
(b) Angled



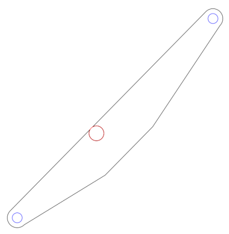
(c) Side

Figure 6.5: Manufactured straight line mechanism [19]

The red link (body 6) in figure 6.5 was initially also made from laser cut plexiglass, but due to the hole in the middle, it was fragile. This fragility caused it to break and a new link had to be created. The link shape was manually changed to ensure a new, more rigid link shape could be laser cut, or in this case, 3D printed.



(a) Original link shape



(b) Manually adapted link shape

Figure 6.6: Manufacturing drawings body 6 . Black and red outlines are the link and infeasible area outlines, respectively.

Discussion

Using the method outlined in this research, it is possible to go from kinematic design to a physical mechanism quickly, since the manufacturable design is automatically generated. We can also see that bodies successfully adapt their shape to the swept area of springs and spring hinges passing through the layer. However, there are two drawbacks to the manufacturing method:

1. The created link outlines are not checked for a minimum width. Thus, links with small widths can be designed and sent to manufacturing. For these links we currently still need a human to alter the design.
2. Designing mechanisms with nuts and threaded rod proved to be a bad choice because tightening 36 nuts is tedious work. Also, (lock)nuts have a significant height, resulting in long axes. When the axes and the axes holes are not very well fitting, the axes will be skewed. This results in links being skewed and the mechanism as a whole being skewed as well.

6.2. Harts A-frame

For the second prototype, Harts A-frame [10] is created. The path of the uppermost joint of the frame follows a straight, vertical line, see appendix B.2. To this mechanism a single spring is added and its location and stiffness are optimized to obtain a gravitationally balanced mechanism as discussed in chapter 5.3.4.

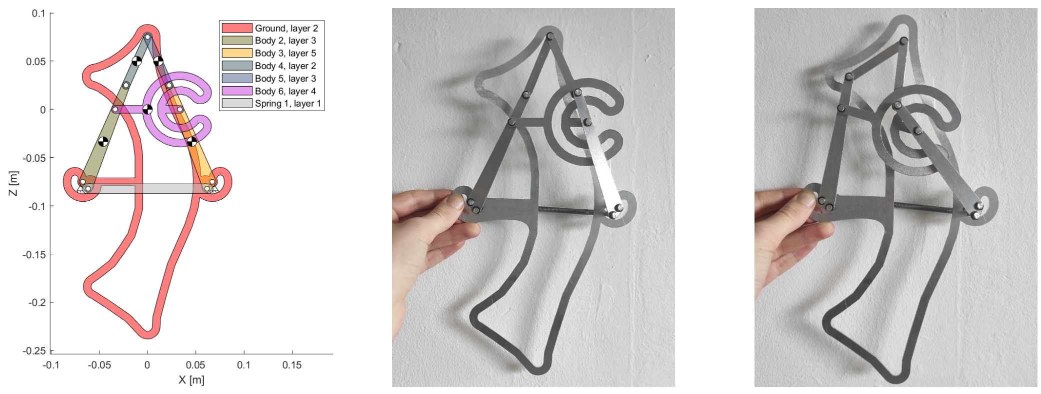
The purpose of this prototype is to evaluate another manufacturing method. This method uses lasercut aluminium sheet and a round rod of aluminium as axis, with circlips confining links to their height.

Method

The mechanism is manufactured with the best option from the manufacturable mechanism designs, as shown in figure 6.7a. For the other options, see appendix C.2. In this design it was opted to use an 1.5mm thick aluminium sheet with aluminium rods as axes and circlips holding the links at their height. Additionally, the ground link was enlarged slightly to make sure it is stiff enough, which can be seen in figure 6.7b.

Results

The manufactured A-frame follows the expected kinematic movement (Fig. 6.7b-c).



(a) Best configuration for A-frame mechanism (b) Initial link locations (c) Moved link locations

Figure 6.7: Manufactured A-frame

Discussion

There are two drawbacks to the manufactured design:

1. The gravitationally balanced aspect does not really come to fruition. Because the tension spring contracts the distance between two axes, the axes become slanted. As they are now skewed, the length of the spring is different and thus the energy is also different than calculated.
2. The links also become skewed, due to the spring slanting the axes. The skewed links are in each others way which hinders the free movement of the mechanism. This needs to be manually counteracted when moving the mechanism to ensure it follows the entire kinematic movement.

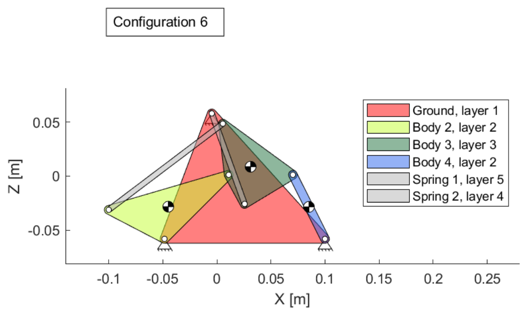
6.3. Roberts spring mechanism

Finally, two Robert spring mechanisms are build. Both are based on the same kinematic design, but differ in their manufacturable design. The first design does not have a single link which has adapted its shape to take the infeasible area created by other elements into account, whereas the second design is nearly unrecognizable due to the two adapted link shapes.

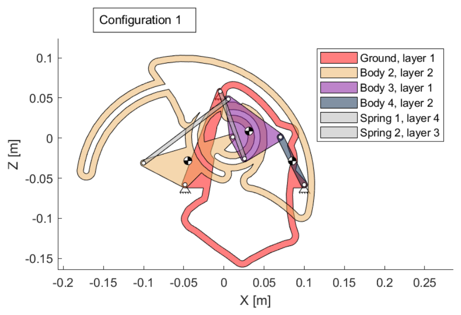
These manufactured mechanisms show the different possibilities which can exist when manufacturing a kinematic design. Is a simple, straightforward design allowed, or is a more complicated design required? The first mechanism consists of a higher number of layers, while the other mechanism combines more complex links in the same layer, which leads to a thinner mechanisms, as shown in figure 6.8b.

Method

The mechanism is based on the Roberts mechanism with two springs. These springs were automatically generated, but not optimized towards a certain goal. With the manufacturability method, we found that four possible designs could be manufactured, see appendix C.3 for all manufacturable mechanisms. The automatically chosen design, using equation 3.5, leads us to the simple configuration as shown in figure 6.8a and we manually choose the thinnest configuration shown in figure 6.8b.



(a) The best configuration consists of simple link shapes

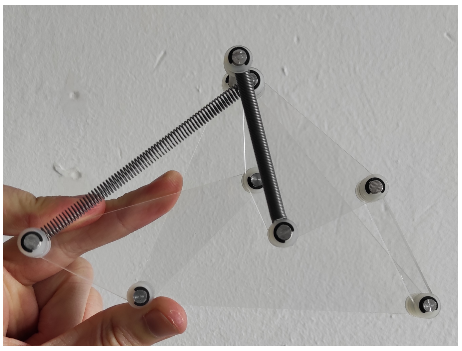


(b) The manually chosen thin configuration requires heavily adapted link shapes

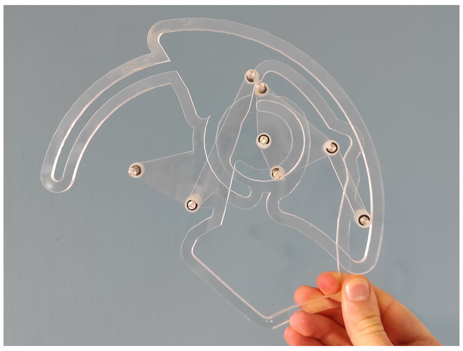
Figure 6.8: Two chosen configurations of the Roberts mechanism

Results

The mechanisms are manufactured and follow the movement as expected. However, it was chosen to not attach springs to the thin configuration, as will be explained in the discussion below.



(a) Simple configuration



(b) Thin configuration (without attaching springs)

Figure 6.9: Manufactured Roberts mechanisms

Discussion

Both mechanisms allow for the full range of desired motion and can complete this without interference. In the prototype of the simple mechanism the springs are attached. This shows that they can be placed in layer 4 and 5, which is the same layer configuration as seen in the thin configuration. However, we experience a lot of force created by the springs, which results in the inability of smoothly moving the mechanism with its skewed axes. Therefore, it was decided not to attach the springs to the thin configuration so the movement can easily be showcased.

6.4. Conclusion

In this chapter, we have shown the steps to obtain four manufactured mechanisms. An automatically evolved mechanism, Harts A-frame and two Roberts spring mechanism configurations are created and all these mechanisms can follow their desired path interference-free. The first prototype demonstrates that automatically created rigid body spring mechanism designs can be evaluated and can produce manufacturable mechanisms. The second prototype demonstrates that different manufacturing methods can be used. Lastly, the Roberts prototypes incorporate multiple springs, which can potentially interfere with other members. Since no interference is found in any of the four manufactured prototypes, they demonstrate the successful evaluation of potential spring interference.

7 Discussion and recommendations

In this chapter, insights will be given on the achieved results and on possible further research. First, the achievement of obtaining interference-free rigid body spring mechanisms is discussed. Next, we unexpectedly obtain thinner-than-conventional mechanisms with this method. Moreover, the variety of mechanism elements is discussed after which the speed of the manufacturability evaluation is assessed. Then, the method of spring optimization is studied. Lastly, a possible new feature is discussed: the inclusion of a toolhead.

Spring interference
With this research, we can now automatically evaluate and adapt for manufacturable rigid body spring mechanisms. This step has been reached by expanding on the theory by Sen et al. using the basis of the engine developed by Kuppens and Wolfslag.

To reach this goal, the method by Sen et al. [33] was rebuild to work with the engine. By implementing this method, a thorough understanding of the ins and outs of the method was obtained, which also allowed us to gain insight into an incomplete part of Sens method.

After validating that the implemented method is applicable for rigid body mechanisms, work started on spring interference. There was a clear vision on how to tackle the integration, as a spring works similar to a body for the majority of the methods. The biggest addition is the method to obtain the relative swept area for springs, in which the spring hinges remain at one location. Even though this method seldom triggers false negatives when infeasible areas are overlapping with an outer part of the oval, this method works well.

After observing the relative swept areas of spring hinges, an additional geometric feasibility rule was created. This rule detects geometric infeasibilities in early stages of mechanism evaluation, preventing costly evaluations in later stages.

Thinner mechanisms
An unforeseen result was the creation of thin mechanisms due to the reduction in the number of layers. Since the method in this research creates a variety of possible configurations, less conventional configurations are also obtained. In these configurations, links sometimes move within another link (as seen in fig. 6.7a and fig. 6.8b). These configurations resulted in manufacturable mechanisms which consist of less layers than a more straightforward solution (as seen in fig. C.3c and fig. 6.8a).

Thinner mechanisms can be applied in new areas where the allowed thickness is minimal. However, these configurations are often not automatically selected by their fitness. The increased area of the link shapes result in a less favorable score than straightforward solutions. The fitness equation can be adapted to select mechanisms which are thinner, at the expense of links that are changed in size.

Variety of mechanism elements
The current method to evaluate and adapt for manufacturability in automatically synthesized mechanisms is limited by the variety of implemented elements. In this research, springs are added as a possible element, but this does not complete the full scale of all mechanism element options.

Since as a basis we use the representation by Kuppens [19], we are limited to the elements available in his work. Thus, the first steps to become all-inclusive should be to take full advantage of Kuppens work.

In his work, he incorporates rigid bodies, revolute joints, prismatic joints and springs. Prismatic joints are still missing in this research. To include prismatic joints, similar changes as for springs are required. The layer assignment vector should include prismatic joints, the layer assignment algorithm needs to be adapted for prismatic joints, new feasibility rules should be obtained and the member-hinge matrix has to include prismatic joints. Suggestions to implement prismatic joints are documented in appendix A.3.

However, in the current research the work on prismatic joints has not been completed or implemented due to time constraints. As a result of these constraints, no work thus far has been undertaken to include torsion springs and dampers. These torsional elements are defined as parameters of hinges within the mechanism represention by Kuppens [19].

Speed

Currently, mechanism manufacturability evaluation and adaptation occurs after mechanism synthesis, however, ideally, every automatically synthesized mechanism is manufacturable. This can be done by including the manufacturability evaluation within an optimization. To include this for mechanism optimization algorithms, the manufacturability method needs to be very fast. Sadly, the current method to check and adapt for manufacturability is too slow.

The method takes approximately between 5 and 60 seconds on an Intel i7-9750H CPU using 6 Matlab workers. Up to 80% of this time is used for obtaining the swept areas in 6 parrallel loops. The remaining 20% is spent on the structural and geometric synthesis. In an early effort to reduce computation time, member outlines are only generated once every five time steps, as described in section 3.3.

We believe for this method to be useful in a full mechanism optimization algorithm where thousands of options are evaluated, it should run well within one second. To achieve a runtime lower than one second, several options can be explored:

1. Compile (mex) time-costly functions in Matlab
2. Write program in a pre-compiled, faster language like C++ [2]
3. Write a more efficient program

The first option would seem the easiest option, as currently the program is already written in Matlab. However, to compile functions, the functions and objects need to be compatible with Matlabs compilation engine. One of the objects used (`polyshape`) is - at the time of writing - not compatible with Matlab compilations. Because this object is not directly compatible, a different object needs to be used. This new object could be custom written, or perhaps the deprecated `polybool` could be used.

The second option requires for sections of the code to be rewritten in a different programming language. Currently, obtaining the swept areas takes up to 80% of the computation time, so it would be wise to start by rewriting this part. Lastly, the third option has not been explored, since the code has been written to the best of our ability. One possible improvement, given in chapter 3.3.1, could be to obtain the swept areas of hinges with their position instead of translating and rotating its outline for every time step. Apart from this, there do not seem to be more significant speed improvements possible when using Matlab with the same objects.

Layer assignment

The layer assignment method is initially created by Sen et al. [33]. All possible relative layer assignment are achieved with the directed adjacency matrix **L**. This matrix stores whether links are situated higher or lower than connected links. This is used to create possible layer assignments, but not all possible layer assignments are found.

A method to find the unexplored options is created with the extended layer assignment algorithm in chapter 4.1.2. The previously unexplored assignments are the options in which a member is not directly below or above adjacent links, but moved one or more layers. This research finds that moving springs to a different layer, while adhering to the initial directed adjacency matrix, changes the feasibility of rigid body spring mechanisms. Since this method is necessary to obtain feasible designs, this new layer assignment method was created for springs.

However, it has not been tested if the feasibility of mechanisms changes if rigid bodies would be allowed to change layers. Potentially, the link outlines could become more simple, resulting in better mechanisms. Therefore, the current adapted method should be evaluated when used to change the layer of links.

Spring optimization
With this research, rigid body spring mechanisms can be evaluated and adapted for manufacturability. This work was put into practice by building three prototypes, of which one was optimized towards a gravitationally balanced mechanism, described in section 6.2. To obtain this mechanism, a single spring type was used and optimized towards the best result by varying its initial length and hinge locations.

This optimization method varies the spring stiffness indirectly by changing its initial length. Ideally, this method could also choose different spring types, which - for the same initial length - have a different spring stiffness. This still limits this algorithm, and thus does not freely choose from off-the-shelve mechanisms, it merely selects the ideal configuration of the spring a user has on hand.

Toolhead
Mechanisms manufactured with the method described in this research are usually not directly ready to be used for an application. More elements are often necessary to make the mechanism useful, for example a pen, a gripper or even a drill. To use such a tool, the mechanism should be manufacturable with the element in mind. The mechanism should include mounting points for the tool, take the swept area of the tool into account and consider the mass of the tool.

In an effort to make the manufacturable mechanisms more useful, a start has been made at implementing such a tool in an appendix A.2. Unfortunately, this method only works when using a single hinge to attach a tool body. The number of hinges is limited by the simulation, which likely can not deal with an overconstrained mechanism.

Conclusion

Concluding the discussion and recommendations, we find that we can now obtain mechanisms without spring interference. Additionally, the configurations found can consist of fewer layers than conventional solutions. Furthermore, recommendations for future improvements are given. The time it takes to evaluate and adapt for a manufacturable mechanism could be decreased. Additionally, by adapting the layer assignment method, all possible layer assignments could be obtained. This theory is currently only applied when assigning layers to springs. Finally, off-the-shelve springs can be optimized towards gravitationally balanced mechanisms and an effort to attach toolheads to mechanisms is discussed.

8 Conclusions

Concluding this research, we find that the discussed methods result in the ability to succesfully evaluate whether planar rigid body spring mechanisms can be manufactured. Furthermore, if these mechanisms can not directly be manufactured, they can be adapted to become manufacturable. To reach this goal, three pre-identified subgoals have been accomplished.

Firstly, interference-free planar linkages with springs can be created. This goal is reached by implementing and extending the theory by Sen et al. [33]. For the structural synthesis, this extension allows for springs in the layer assignment vector and calculates additional layer assignments leading to new, feasible solutions.

For the geometrical synthesis, this extension obtains an adapted swept area of springs and inverted swept areas of members in the same layer by rotating and scaling the swept area of those members. This adapted swept area allows the algorithm to reliably check if interference with springs occurs. Also, a new geometric feasibility check for springs is introduced. Rigid bodies can adapt their shape for an infeasible area, in contrast to a spring's body, which can not change. This new check evaluates if there is an infeasible area between the spring hinges which would result in a geometric infeasibility.

Secondly, this algorithm is coupled to an extended version of the mechanism engine developed by Kuppens and Wolfslag [20]. With this extended engine, mechanisms represented with adapted graph theory can be evaluated and adapted for manufacturability. This is done with a newly developed member-hinge matrix, a member layer assignment vector and by adding thickness information to members. Furthermore, the method by Kuppens [19] is extended by using off-the-shelve springs when optimizing mechanisms towards spring-based gravity-balanced mechanisms. The springs are optimized such that for a given off-the-shelve spring its optimal length is determined. This spring can now be directly attached to the optimized gravity-balanced mechanism.

Thirdly, four automatically generated prototypes are produced and are able to move interference-free. The first prototype demonstrates that automatically created rigid body spring mechanism designs can be evaluated and are able to become manufacturable mechanisms. The second prototype demonstrates that different manufacturing methods can be used. The third mechanism is built in two configurations. The thin configuration shows that a mechanism can be built with fewer layers than a conventional configuration. Since no interference is found in any of the four manufactured prototypes, they demonstrate the successful evaluation of potential spring interference and thereby the completion of this research.

With this research, engineers gain a tool which can obtain manufacturable mechanisms from kinematic mechanism designs. These designs can originate from automatic methods creating complete mechanisms, optimization methods to obtain a spring-based gravity-balanced design or even manually created designs. The methods in this research make it possible for a computer to evaluate more mechanism configuration options than realistically possible by manual labor. Automating the evaluations of mechanisms configurations frees time for the user to focus on other aspects of his work. Therefore, this work can save time for mechanism designers.

Bibliography

[1] Pedro Alves, Francisco Cruz, Luís F. Silva, and Paulo Flores. Synthesis of a Mechanism for Human Gait Rehabilitation: An Introductory Approach. In Paulo Flores and Fernando Viadero, editors, *Mechanisms and Machine Science*, volume 24 of *Mechanisms and Machine Science*, pages 121–128. Springer International Publishing, Cham, 2015. doi: 10.1007/978-3-319-09411-3_13.

[2] Tyler Andrews. Computation time comparison between matlab and c++ using launch windows. 2012. URL `https://www.researchgate.net/publication/303945839_Computation_Time_Comparison_Between_Matlab_and_C_Using_Launch_Windows`.

[3] Rogier Barents, Mark Schenk, Wouter D. Van Dorsser, Boudewijn M. Wisse, and Just L. Herder. Spring-to-spring balancing as energy-free adjustment method in gravity equilibrators. *Journal of Mechanical Design, Transactions of the ASME*, 133(6):1–10, 2011. ISSN 10500472. doi: 10.1115/1.4004101.

[4] Norman Biggs, Edward K. Lloyd, and Robin J. Wilson. *Graph theory 1736 - 1936*. Clarendon Press, 1986.

[5] Kailash Chaudhary and Himanshu Chaudhary. Optimal dynamic design of planar mechanisms using teaching-learning-based optimization algorithm. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 230(19):3442–3456, 2016. ISSN 20412983. doi: 10.1177/0954406215612831.

[6] L. Dobrjanskyj and F. Freudenstein. Some applications of graph theory to the structural analysis of mechanisms. *Journal of Manufacturing Science and Engineering, Transactions of the ASME*, 89(1):153–158, 1967. ISSN 15288935. doi: 10.1115/1.3609988.

[7] Selçuk Erkaya. Investigation of balancing problem for a planar mechanism using genetic algorithm. *Journal of Mechanical Science and Technology*, 27(7):2153–2160, 2013. ISSN 1738494X. doi: 10.1007/s12206-013-0530-z.

[8] F. Freudenstein and A. T. Yang. Kinematics and statics of a coupled epicyclic spur-gear train. *Mechanism and Machine Theory*, 7(2):263–275, 1972. ISSN 0094114X. doi: 10.1016/0094-114X(72)90008-0.

[9] Sang Min Han, Suh In Kim, and Yoon Young Kim. Topology optimization of planar linkage mechanisms for path generation without prescribed timing. *Structural and Multidisciplinary Optimization*, 56(3):501–517, 2017. ISSN 16151488. doi: 10.1007/s00158-017-1712-6.

[10] Harry Hart. On some Cases of Parallel Motion. *Proceedings of the London Mathematical Society*, s1-8(1):286–291, nov 1876. ISSN 1460244X. doi: 10.1112/plms/s1-8.1.286.

[11] Just L Herder. Energy-free Systems. Theory, Conception and Design of Statically balanced Spring Mechanisms, 2001.

[12] Inkscape Project. Inkscape. URL `https://inkscape.org`.

[13] Amir Jafari, Nikos G. Tsagarakis, and Darwin G. Caldwell. Exploiting natural dynamics for energy minimization using an actuator with adjustable stiffness (AwAS). *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4632–4637, 2011. ISSN 10504729. doi: 10.1109/ICRA.2011.5979979.

[14] Bum Seok Kim and Hong Hee Yoo. Unified synthesis of a planar four-bar mechanism for function generation using a spring-connected arbitrarily sized block model. *Mechanism and Machine Theory*, 49:141–156, 2012. ISSN 0094114X. doi: 10.1016/j.mechmachtheory.2011.10.013.

[15] Bum Seok Kim and Hong Hee Yoo. Unified mechanism synthesis method of a planar four-bar linkage for path generation employing a spring-connected arbitrarily sized rectangular block model. *Multibody System Dynamics*, 31(3):241–256, 2014. ISSN 13845640. doi: 10.1007/s11044-013-9371-x.

[16] Suh In Kim and Yoon Young Kim. Topology optimization of planar linkage mechanisms. 2014. doi: 10.1002/nme.

[17] Yoon Young Kim, Jung Hun Park, Jin Sub, Hyun Sang, and Jun Nam. Automatic Synthesis of a Planar Linkage Mechanism With Revolute Joints by Using Spring-Connected Rigid Block Models. 129, 2007. doi: 10.1115/1.2747636.

[18] P. Reinier Kuppens. Literature Survey: Automated mechanism design with artificial evolution. 2015.

[19] P. Reinier Kuppens. Automated Robot Design With Artificial Evolution. 2016.

[20] P. Reinier Kuppens and Wouter J. Wolfslag. A String-Based Representation and Crossover Operator for Evolutionary Design of Dynamical Mechanisms. *IEEE Robotics and Automation Letters*, 3(3):1600–1607, 2018. ISSN 23773766. doi: 10.1109/LRA.2018.2800102.

[21] Mary C. Lacity and Leslie P. Willcocks. A new approach to automating services. *MIT Sloan Management Review*, 58(1):41–49, 2016. ISSN 15329194. doi: 10.7551/mitpress/11633.003.0015.

[22] T. Laliberte, C.M. Gosselin, and Martin Jean. Static balancing of 3-DOF planar parallel mechanisms. *IEEE/ASME Transactions on Mechatronics*, 4(4):363–377, 1999. ISSN 10834435. doi: 10.1109/3516.809515.

[23] Thanh Danh Le and Kyoung Kwan Ahn. A vibration isolation system in low frequency excitation region using negative stiffness structure for vehicle seat. *Journal of Sound and Vibration*, 330(26):6311–6335, 2011. ISSN 0022460X. doi: 10.1016/j.jsv.2011.07.039.

[24] Hod Lipson. How to Draw a Straight Line Using a {GP}: Benchmarking Evolutionary Design Against 19th Century Kinematic Synthesis. *Late Breaking Papers at the 2004 Genetic and Evolutionary Computation Conference*, 2004. URL `http://www.cs.bham.ac.uk/{~}wbl/biblio/gecco2004/LBP063.pdf`.

[25] Hod Lipson. Evolutionary synthesis of kinematic mechanisms. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*, 22(3):195–205, 2008. ISSN 08900604. doi: 10.1017/S0890060408000139.

[26] Abel O. Matser. Manufacturability in automatic synthesis of planar rigid body spring mechanisms. 2020.

[27] D. G. Olson, T. R. Thompson, D. R. Riley, and A. G. Erdman. An algorithm for automatic sketching of planar kinematic chains. *Journal of Mechanical Design, Transactions of the ASME*, 107(1):106–111, 1985. ISSN 10500472. doi: 10.1115/1.3258672.

[28] Sergiy Plankovskyy, Yevgen Tsegelnyk, Olga Shypul, Alexander Pankratov, and Tatiana Romanova. Cutting irregular objects from the rectangular metal sheet. In Mykola Nechyporuk, Vladimir Pavlikov, and Dmitriy Kritskiy, editors, *Integrated Computer Technologies in Mechanical Engineering*, pages 150–157, Cham, 2020. Springer International Publishing. ISBN 978-3-030-37618-5.

[29] Michiel Plooij and Martijn Wisse. A novel spring mechanism to reduce energy consumption of robotic arms. *IEEE International Conference on Intelligent Robots and Systems*, pages 2901–2908, 2012. ISSN 21530858. doi: 10.1109/IROS.2012.6385488.

[30] R. Sancibrian, F. Viadero, P. García, and A. Fernández. Gradient-based optimization of path synthesis problems in planar mechanisms. *Mechanism and Machine Theory*, 39(8):839–856, 2004. ISSN 0094114X. doi: 10.1016/j.mechmachtheory.2004.02.012.

[31] Linda C. Schmidt, Harshawardhan Shetty, and Scott C. Chase. A graph grammar approach for structure synthesis of mechanisms. *Journal of Mechanical Design, Transactions of the ASME*, 122(4):371–376, 2000. ISSN 10500472. doi: 10.1115/1.1315299.

[32] Arend L. Schwab, Richard Q. van der Linde, and Pier H. de Jong. Multibody Dynamics reader. 2018.

[33] Dibakar Sen, Sanjib Chowdhury, and Shashank Ram Pandey. Geometric design of interference-free planar linkages. *Mechanism and Machine Theory*, 39(7):737–759, 2004. ISSN 0094114X. doi: 10.1016/j.mechmachtheory.2004.02.007.

[34] Wayne J. Sohn and Ferdinand Freudenstein. An application of dual graphs to the automatic generation of the kinematic structures of mechanisms. *Journal of Mechanical Design, Transactions of the ASME*, 108 (3):392–398, 1986. ISSN 10500472. doi: 10.1115/1.3258745.

[35] Lung-Wen Tsai. An Application of the Linkage Characteristic Polynomial to the Topological Synthesis of Epicyclic Gear Trains. *Journal of Mechanisms, Transmissions, and Automation in Design*, 109(3):329–336, sep 1987. ISSN 0738-0666. doi: 10.1115/1.3258798.

[36] Lung-Wen Tsai. *Enumeration of Kinematic Structures According to Function.* 2001. ISBN 0849309018.

[37] Lung-Wen Tsai, Dar-zen Chen, and Ta-wei Lin. Dynamic Analysis of Geared Robotic Mechanisms Using Graph Theory. *Journal of Mechanical Design*, 120(2):240–244, jun 1998. ISSN 1050-0472. doi: 10.1115/1.2826964.

[38] John J. Uicker Jr, Gordon R. Pennock, and Joseph E. Shigley. *Theory of Machines and Mechanisms.* Oxford University Press, fifth edition, 2017. ISBN 9780190264482.

[39] Jiegao Wang and Clément M. Gosselin. Static balancing of spatial four-degree-of-freedom parallel mechanisms. *Mechanism and Machine Theory*, 35(4):563–592, 2000. ISSN 0094114X. doi: 10.1016/S0094-114X(99)00029-4.

[40] L. S. Woo. Type Synthesis of Plane Linkages. *Journal of Engineering for Industry*, 89(1):159–170, feb 1967. ISSN 0022-0817. doi: 10.1115/1.3609989.

[41] Jeonghan Yu, Sang Min Han, and Yoon Young Kim. Simultaneous Shape and Topology Optimization of Planar Linkage Mechanisms Based on the Spring-Connected Rigid Block Model NOT COPYEDITED. *Journal of Mechanical Design, Transactions of the ASME*, 142(1), 2020. ISSN 10500472. doi: 10.1115/1.4044327.

[42] Kai Zhao, James P. Schmiedeler, and Andrew P. Murray. Design of Planar, Shape-Changing Rigid-Body Mechanisms for Morphing Aircraft Wings. *Journal of Mechanisms and Robotics*, 4(4):1–10, 2012. ISSN 19424302. doi: 10.1115/1.4007449.

A Engine

A.1. Creating kinematic design

To recreate the Roberts mechanism shown in figure A.1, the topology is used as the starting point. The ground (red) is assigned as the first body and the remaining bodies get their index in a clockwise manner. In the same way we distinguish the hinges: starting with the hinge in the lower left corner as the first, the rest get their index in a clockwise manner counting up.

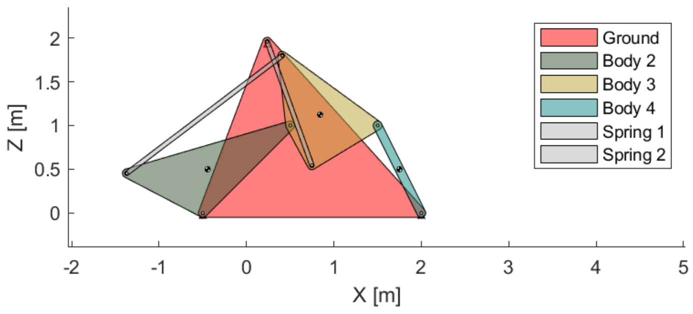


Figure A.1: Roberts mechanism with two springs

Table A.1: Incidence matrix Roberts mechanism of figure 2.3

	H1	H2	H3	H4	S1	S2
Body 1	1	0	0	1	1	0
Body 2	1	1	0	0	0	1
Body 3	0	1	1	0	1	1
Body 4	0	0	1	1	0	0

The pattern in figure 2.2 is used to create the incidence string. This incidence matrix results in $I = [1, 3, 6, 4, 2, 3]$. Together with edge label $E = [1, 1, 1, 1, 2, 2]$ for a mechanism with four hinges and two springs as connectors. We create the mechanism according to the representation of Kuppens [19], as the extended representation, the manufacturable design discussed in this research, is automatically derived from the original representation, the kinematic design.

Next, the physical quantities need to be added. We assume that all bodies have a mass of 1 kg, and that the hinges are regular revolute joints, resulting in mass matrix A.1 and hinge matrix A.2 respectively. The springs in this example were randomly determined, resulting in spring matrix A.3. The other available elements are not utilized, so for Matlab to process them correctly empty containers need to be created, as stated in the equations in A.4. The last elements Matlab requires are initialized in eq A.5. Their values are automatically generated.

$$M_{\text{par}} = \begin{bmatrix} \text{NaN} & \text{NaN} & \text{NaN} \\ 0 & 0.5 & 1 \\ 1 & 0 & 1 \\ 1.75 & 0.5 & 1 \end{bmatrix} \tag{A.1}$$

$$H_{\text{par}} = \begin{bmatrix} -0.5 & 0 & 0 & 0 & 0 \\ 0.5 & 1 & 0 & 0 & 0 \\ 1.5 & 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{A.2}$$

$$S_{\text{par}} = \begin{bmatrix} 0.23 & 1.95 & 0.75 & 0.55 & 0.005 & 4.4 \\ -1.37 & 0.45 & 0.41 & 1.80 & 0.0003 & 0.68 \end{bmatrix}^T \tag{A.3}$$

$$\begin{aligned} P_{\text{par}} &= \text{empty 3 x 0 matrix} \\ Pm_{\text{par}} &= \text{empty 5 x 0 matrix} \end{aligned} \tag{A.4}$$

$$\begin{aligned} \text{fitness} &= [\,] \\ t &= [\,] \\ xss &= [\,] \end{aligned} \tag{A.5}$$

A.2. Toolhead

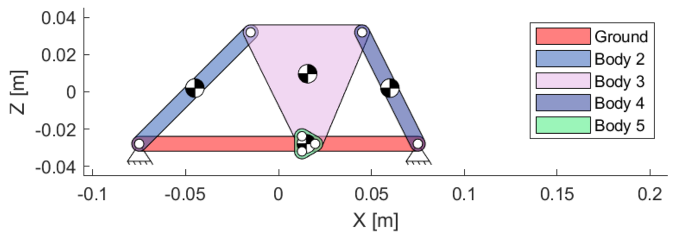


Figure A.2: Roberts mechanism with toolhead attached with three hinges

As discussed in chapter 7, mechanisms manufactured with the method described in this research are not often directly ready to be used for a specific application. Additional elements are often necessary to make the mechanism useful. Such an element could be a pen, a gripper or even a drill. However, to use such a tool the mechanism should be manufacturable with the element in mind. The mechanism should include mounting points for the tool, take the swept area of the tool into account and consider the mass of the tool.

In an effort to make the manufacturable mechanisms more useful, we made a start at implementing such a tool with the function `createToolDNA` (Alg. 2). This function alters a given DNA to include a toolhead.

Algorithm 2 Create spring layer configuration

```
1: function CREATETOOLDNA(DNA, iGM, MparTool, nHNew, alphaStart)
2:     Create new empty DNA toolDNA
3:     Obtain incidence string and edge label for tool - depends on iGM
4:     Fill toolDNA with incidence string, edge label and MparTool
5:     Determine hinge locations and create Hpar - depends on MparTool, nHNew, alphaStart
6:     Fill toolDNA with Hpar
7:     return toolDNA
8: end function
```

With:

- *DNA*: structure onto which this tool will be attached
- *iGM*: index of body onto which the tool will be attached
- *MparTool*: [x y m] location and mass of tools' Center of Mass

- *nHNew* (optional): number of hinges attaching tool to body. If *nHNew* is 1, the hinge will be created at the CoM, otherwise the specified number of hinges will be drawn around the CoM at a radius depending on the dimensions the DNA.
- *alphaStart* (optional): at which degree on the unity circle the first hinge should be placed.

This function returns the toolhead DNA and is built to be attached to iGM with *nHNew* hinges, located around the CoM defined in *MparTool*. Distance from the CoM depends on the dimensions of the DNA. To combine the DNA and the toolDNA use `combineToolDNA`(*DNA,toolDNA*).

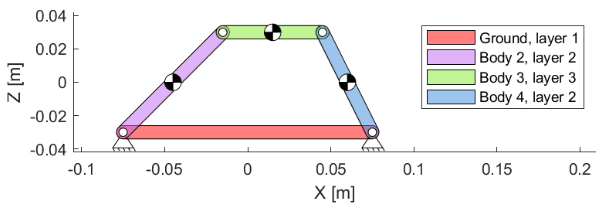


Figure A.3: Unmodified Roberts mechanism

An example is given with a Roberts mechanism . The unmodified Robert mechanism is shown in figure A.3 and mechanisms including a toolhead are shown in figure A.4 and figure A.2. In this implementation, the tool, shown as body 5 in figure A.4, encompasses a single layer and the body shape depends on the hinge locations. Possibly, the tool shape can be manually imported from a DXF file.

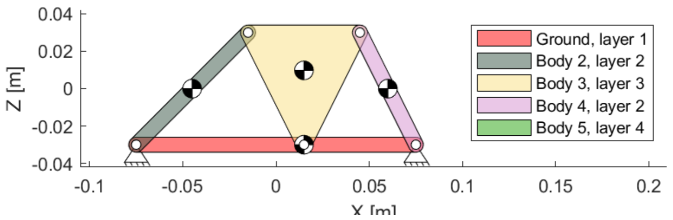


Figure A.4: Roberts mechanism with toolhead (CoM sign at the lower triangle of body 3)

Sadly, this implementation only works when using a single hinge. When using more than one hinge, as shown in figure A.2, the simulation exits without results. We believe this is because the mechanism becomes overconstrained. In some cases the simulation did work, but then the mechanism would slowly, unnaturally, gain more energy throughout its movement. We believe this happens because the hinge constraint equations could add energy when the mechanism is overconstrained.

To conclude, this method can be used to obtain the manufacturability for mechanisms containing a simple toolhead in a single layer. This toolhead can only be attached with one hinge, which limits the possible applications.

A.3. Prismatic joint

As discussed in chapter 7, more mechanism elements could be included in the methods discussed in this research. One of these can be prismatic joints. To include prismatic joints similar changes as for springs are required. A spring is split in three elements: the spring which situates in a single layer and two hinges which connect the spring to other links.

For a prismatic joint something similar needs to be done. First, a prismatic joint, or a slider, requires at least two mounting points (for example at either ends of figure A.5). In Kuppens work, these mounting points are rigidly connected to the ground. Alternatively, they can also be attached with hinges or attached to other links. Secondly, the sliding rails need to be represented. This is the rigid element between the sliding element and the attachment points, and is only situated in a single layer. Thirdly, the sliding element needs to be represented. This could be connected rigidly, or with a hinge, to another member.

For example, take the piston crank mechanism in figure A.6. For the Kuppens representation the incidence matrix in table A.2 is used. To split this single prismatic joint into more elements, the member-hinge matrix in table A.3 is created. In this member-hinge matrix column P1,1 and P1,2 signify the mounting points

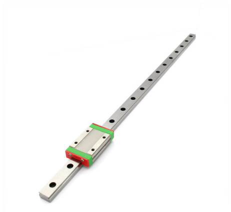


Figure A.5: A possible implementation of a prismatic joint: a linear guide.

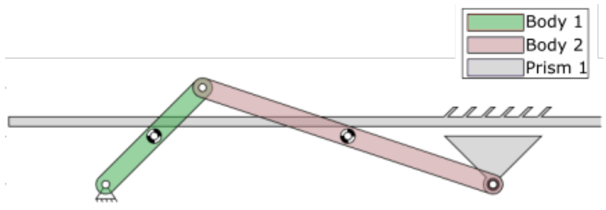


Figure A.6: Piston crank mechanism

connecting the linear guide to the ground, and column P1,3 signifies the slider connecting the linear guide to the the second body.

Table A.2: Original incidence matrix of the piston crank mechanism

	H1	H2	P1
Ground	1	0	0
Body 1	1	1	0
Body 2	0	1	1

Table A.3: Member-hinge matrix of the piston crank mechanism

	H1	H2	P1,1	P1,2	P1,3
Ground	1	0	1	1	0
Body 1	1	1	0	0	0
Body 2	0	1	0	0	1
Prismatic joint 1	0	1	1	1	1

Next, the feasibility of all elements should be determined. The swept area of P1,1 to P1,3 could evaluated similar as hinges, as these elements are kinematically defined. Similarly, the outline of a linear guide can not be changed. This means the new geometric feasibility rule for springs, as discussed in section 4.2, can also be used for prismatic joints.

A.4. Spring tests

Four types of springs are tested: TRM007, TRM009, TRM015 and TRM029. These springs are obtained from Tevema, a Dutch spring supplier. To test spring behavior when it is cut into multiple elements, the four springs are tested in a tensile bench.

The springs are attached to two pieces of laser cut PMMA by threading the spring through the holes, as shown in figure A.7. The distance is measured between the insides of the PMMA mounting pieces, which is equal to the length of spring which can elongate.

The spring is subsequently mounted in a tensile bench without elongating the springs, so there is no tension in the springs. A test is started to elongate the springs to a length which was subjectively determined. The goal was to test the spring force-deflection without plastically deforming. Also, we wanted to reach a linear behavior for the spring stiffness.

Sadly, this did not happen for spring TRM007 (Fig. A.8a). In the 400 mm test the spring behaves as expected, with a steep rise at the beginning for the preload, after which the slope became constant. However, for the lengths of 100 and 500 mm, this did not happen. Here, the steep slope resembles more of a gradual climb. This is probably due to the plastic deformation of one side of the spring before testing. This deformed side influenced the results of the 100 mm and 500 mm test.

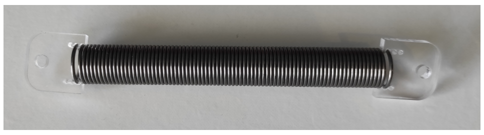
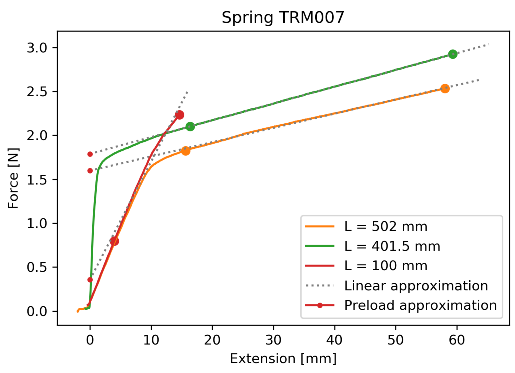
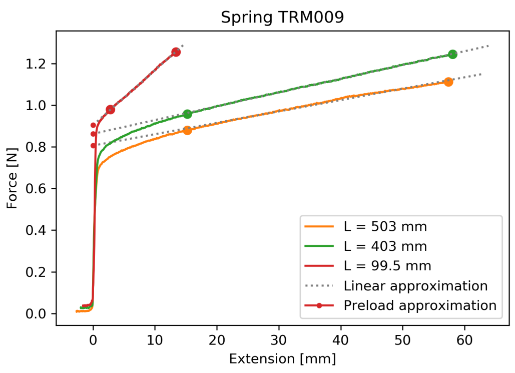


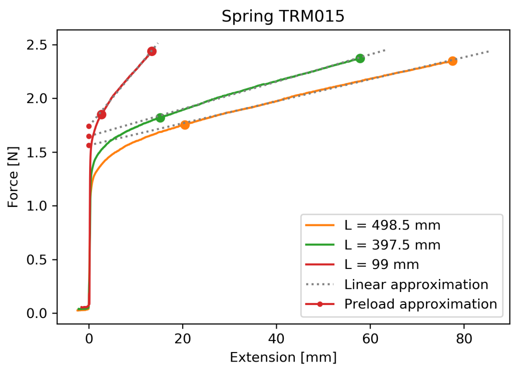
Figure A.7: Spring attached to two mounts for tensile testing



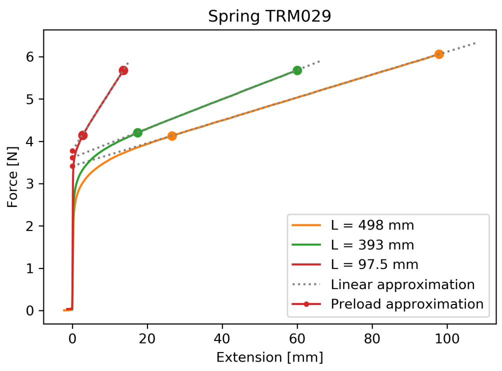
(a) TRM007



(b) TRM009



(c) TRM015



(d) TRM029

Figure A.8: Test results of all four springs

Table A.4: Calculated spring stiffness

Test	**Spring ID**	**Length [mm]**	**Stiffness [N/mm]**	**Stiffness 500mm [N/mm]**	**Preload Fv [N]**
tra05	TRM007_502	502	0.0162	0.0163	1.5971
tra08	TRM007_401.5	401.5	0.0191	0.0154	1.7882
tra12	TRM007_100	100	0.1344	0.0269	0.355
tra02	TRM009_503	503	0.0055	0.0055	0.806
tra06	TRM009_403	403	0.0066	0.0053	0.8622
tra11	TRM009_99.5	99.5	0.0263	0.0052	0.9042
tra03	TRM015_498.5	498.5	0.0102	0.0102	1.563
tra07	TRM015_397.5	397.5	0.0127	0.0101	1.6476
tra10	TRM015_99	99	0.0526	0.0104	1.7403
tra01	TRM029_498	498	0.027	0.0269	3.4142
tra04	TRM029_393	393	0.0345	0.0271	3.6116
tra09	TRM029_97.5	97.5	0.1399	0.0273	3.7747

B

Manufacturability

B.1. Obtaining the swept area

In this section we compare the different methods to obtain the swept area and draw a conclusion as to which method will be used in our model.

B.1.1. Method

Sen et al. [33] lists three methods to obtain the swept area are given:

1. The union of individual positions.
2. The convex hull of all of them
3. The convex hull of two successive link geometries and the union of all of them

Two different operations can be distinguished: the union and the convex hull. In Matlab, the `convhull` operation exists for a set of points and for a polyshape (special type). To create a union in Matlab, an operation only exists for polyshapes. Therefore, the methods that allow for the use of points will be tested with the points type, while the others may also use the polyshape type.

Furthermore, it will be tested whether the precision has a negative effect on the result. The precision is the number of shapes to obtain in the time range. The default time range is from 0 to 5 seconds, with steps of 0.01 seconds. When the precision is 1, the time between shape calculation is the lowest step size of 0.01 seconds, while a precision of 5 results in a step size of 0.05 seconds.

B.1.2. Results

A single run, without drawing the plots, with precision = 5, results in the following runtimes:

1. Using polyshapes: 0.142330 seconds
2. - Using points: 0.000405 seconds
   - Using polyshapes: 0.266028 seconds
3. Using combination of points and polyshapes: 0.086934 seconds.

A single run, without drawing the plots, with precision = 1, results in the following runtimes:

1. Using polyshapes: 1.296410 seconds
2. - Using points: 0.024115 seconds
   - Using polyshapes: 58.491201 seconds
3. Using combination of points and polyshapes: 0.661838 seconds

A single run, with drawing the plots, with precision = 5, results in the following runtimes:

1. Using polyshapes: 0.138741 seconds
2. - Using points: 0.001713 seconds
   - Using polyshapes: 0.266551 seconds
3. Using combination of points and polyshapes: 0.096538 seconds.

In figure B.1 and B.2, the effect of the precision on the contour of the swept area is visualized. In these figures, the red contour depicts all timesteps the link has been through. In addition the four different methods are displayed, in which the most outer contour is the convex hull of all positions (method-2). Method-1 is the outer line of the union of all red mechanism contours. The third and most accurate method [33], method-3 is harder to discern.

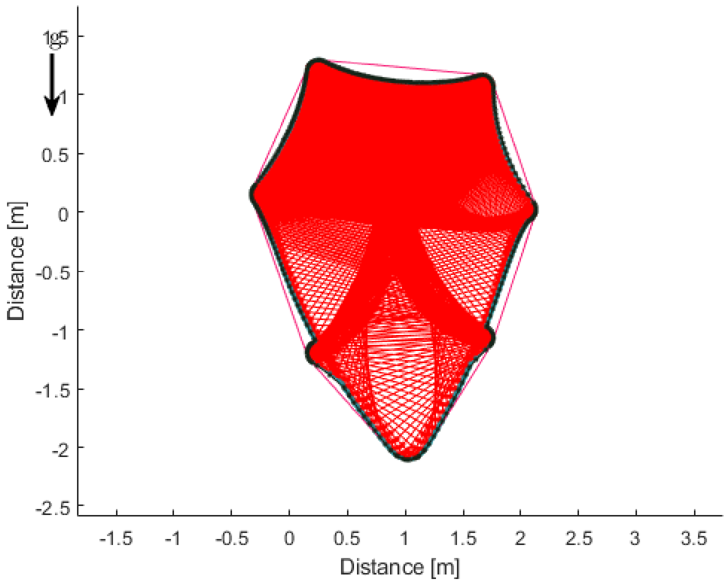


Figure B.1: Swept area with precision value 1. With method-1 underneath the black line, method-2 pink, method-3 underneath the black line.

B.1.3. Conclusion and recommendations

In a direct comparison between points and polyshapes in method-2, using points is significantly faster. However, in the most accurate method according to Sen et al. [33] - method-3 - the speed is lower, but still acceptable.

When looking at the results of figure B.1, method-1 and method-3 both closely follow the outlines, due to the high availability of link outlines. However, method-2 (in pink) always takes up too much area. When looking at the results of figure B.2, in the locations with lots of shapes near to each other, method-3 still functions great with this lower precision (see upper parts of the figure B.2, where the downward curves are identical to the outlines). However, when less data is provided, the methods to combine the points is more prone to errors (best visible in the lower part of figure B.2). For example, on the lower left corner of figure B.2, method-3 fails to make an inward arc. In the same place, method-1 does make the inward arc, but fails to account for the movement between two positions, resulting in a too small swept area.

To conclude, the third method is slower than other possible methods, but still preferred over method-1 and method-2. Sen et al. [33] recognizes method-3 as the best option, because it gives a good approximation of the swept area of the links. Even when it is mistaken, it creates a swept area that is too big instead of too small, which is a safe error margin. A precision of 5 is used to decrease the computation time, while still obtaining good results.

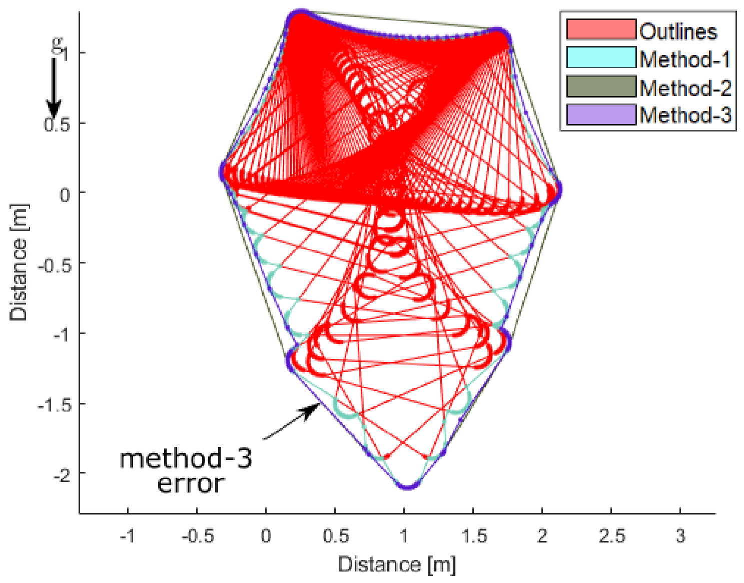
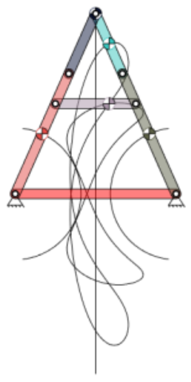
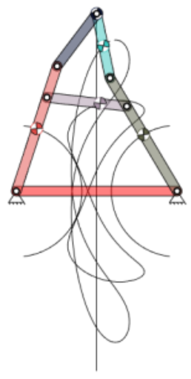


Figure B.2: Swept area with precision value 5.

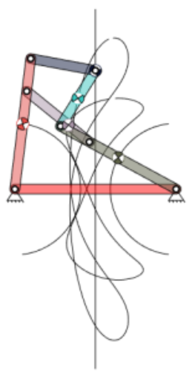
B.2. Harts A-frame movement



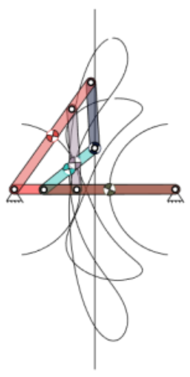
(a)



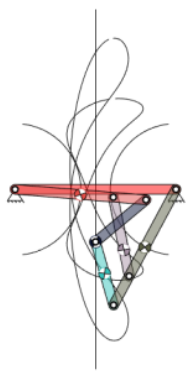
(b)



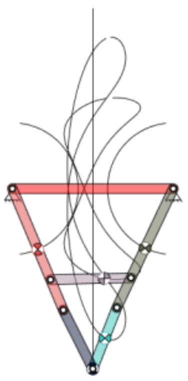
(c)



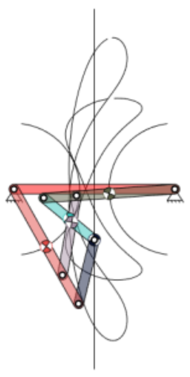
(d)



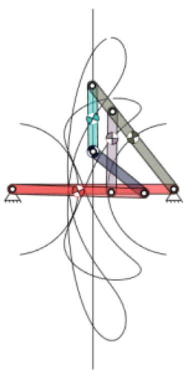
(e)



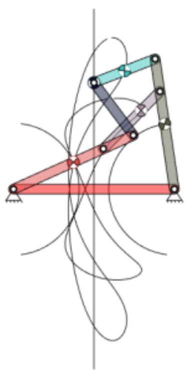
(f)



(g)



(h)



(i)

Figure B.3: Harts A-frame. The paths of the (manually determined) CoMs are traced with black lines

B.3. Export mechanism

To evaluate the third research goal: *"Investigate if automatically generated manufacturable mechanisms are manufacturable and interference free"*, the manufacturable mechanisms need to be physically built. To make this process just as easy and care-free as the creation of manufacturable designs, two methods are required. The first method exports the link outline shape to an AutoCAD Drawing Exchange Format (DXF) and the second creates the Bill of Materials (BoM).

B.3.1. Drawings

To quickly build link shapes, they need to be manufacturable without involving a lot of manual labor. A lasercutter or 3D printer requires minimal manual labor and are therefore suitable options.

A laser cutter uses DXF files to cut shapes according to the lines in the drawing. The cutting order is carried out in the same order as the order in which the outlines are drawn. However, when an external contour is cut first, the workpiece may tilt, slip, or possibly fall through the grates of the cutting table. Therefore, internal contours should be cut first and should be indicated differently so the machine operater is less likely to make a mistake.

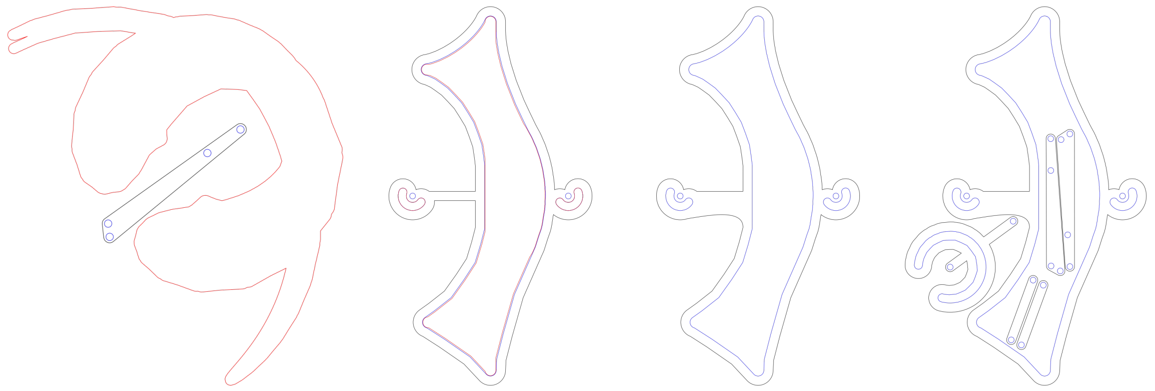
To obtain these manufacturing files, the following algorithm is used for every link:

Algorithm 3 Export link shape to Autocad DXF

```
1: Get link external contour, link internal contour and surrounding infeasible area
2: Draw link external contour in black
3: Draw link internal contour in blue
4: Draw infeasible area in red
5: return Drawing
```

This results in the drawings of figure B.4a and B.4b. In figure B.4b we notice that the link shape is likely too weak to be used in a mechanism. Therefore, we change the shape, while staying outside the (red) infeasible area, resulting in figure B.4c. Changing the shape can easily be done in a CAD program or a free vector editor such as Inkscape [12]

These files can be sent to a lasercutter directly, but for optimal material usage they need to be placed efficiently. Currently, this is done manually (Fig. B.4d), but this could be optimized with methods such as the one created by Plankovskyy et al. [28].



(a) Body 2 and infeasible area (red) (b) Body 1 and infeasible area (red) (c) Body 1 manually changed to be stronger (d) Final layout for cutting of all bodies

Figure B.4: Manufacturing drawings A-frame

B.3.2. Bill of Materials

To quickly and effortlessly get an overview of the required materials for the project, a Bill of Materials is created. For the mechanisms created with the methods in this research, one needs to know the number of circlips (or locknuts), washers, rods, springs and bodies. An example is given in table B.1, which could also store the location where the notches for the circlips should be chiselled.

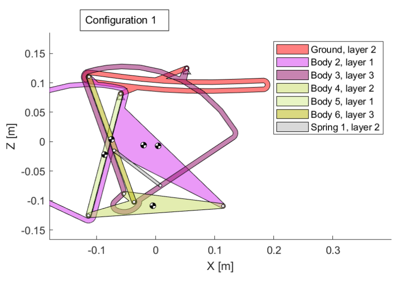
Table B.1: Bill of Material of the A-frame mechanism

Item	**Amount**	**Diameter [mm]**	**Length [mm]**
Circlip	36	5	
Washer	36	5	
Body	7		
Rod	1	5	9.6
Rod	1	5	18.2
Rod	1	5	9.6
Rod	1	5	13.9
Rod	1	5	9.6
Rod	1	5	9.6
Rod	1	5	9.6
Rod	1	5	22.4
Rod	1	5	31.0
Spring TRM009	1		74.0

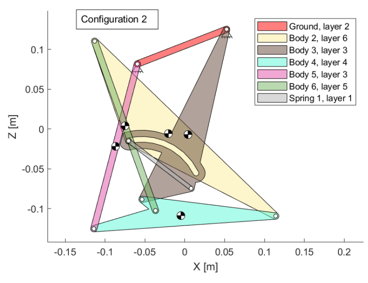
C Prototypes

In this section the manufacturable configurations are shown for mechanisms of which a prototype has been built.

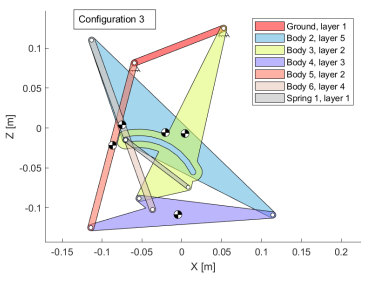
C.1. Automatically evolved mechanism



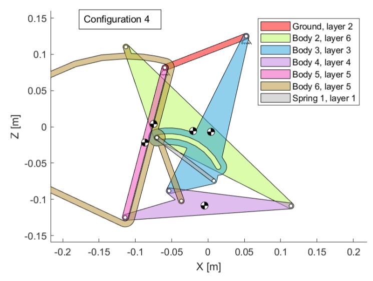
(a) Configuration 1



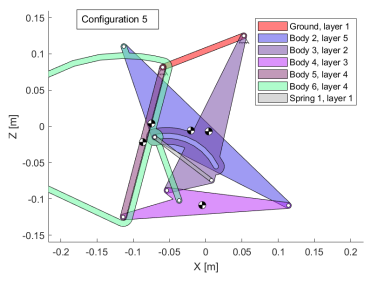
(b) Configuration 2



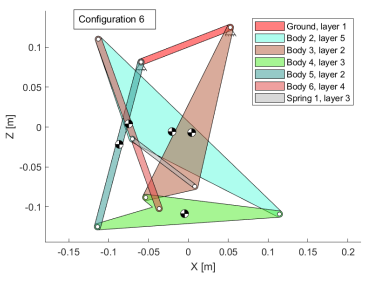
(c) Configuration 3



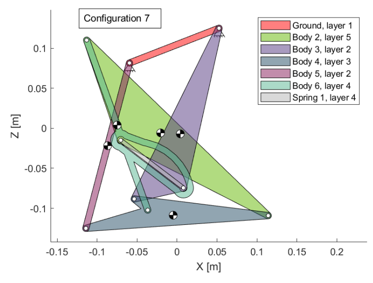
(d) Configuration 4



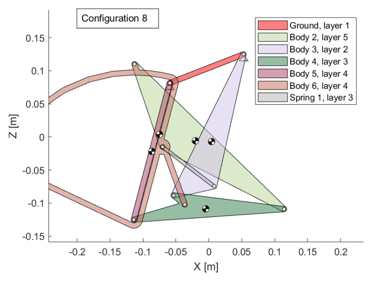
(e) Configuration 5



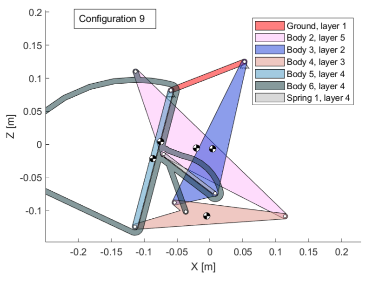
(f) Configuration 6



(g) Configuration 7



(h) Configuration 8



(i) Configuration 9

Figure C.1: Manufacturable configurations automatically evolved mechanism

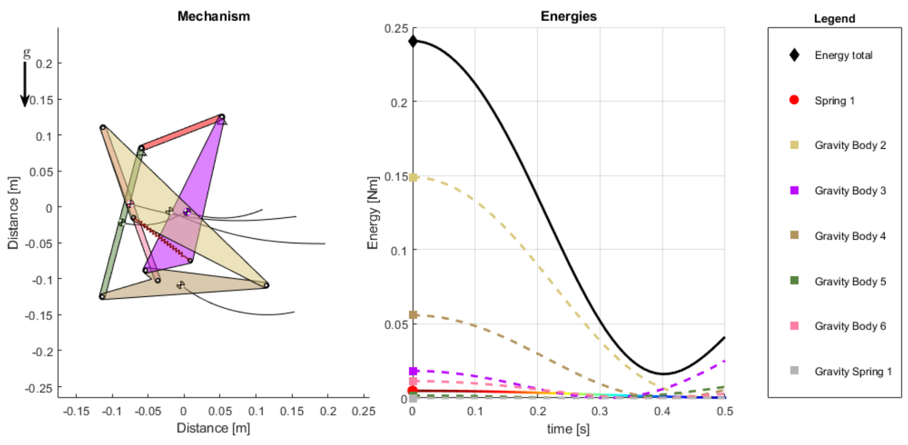
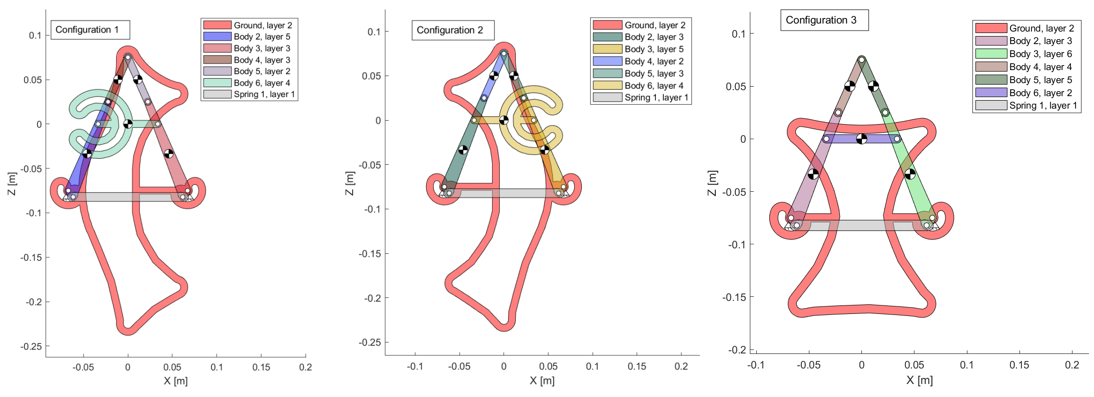


Figure C.2: Automatically evolved mechanism with outlines of CoMs shown with black lines

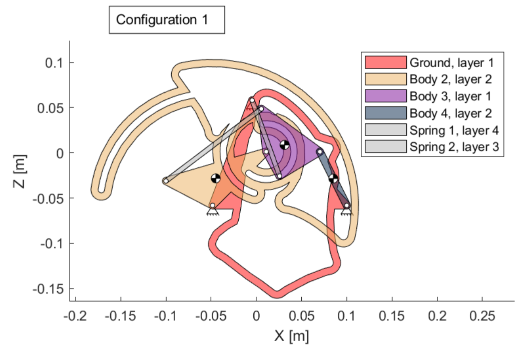
C.2. Harts A-frame



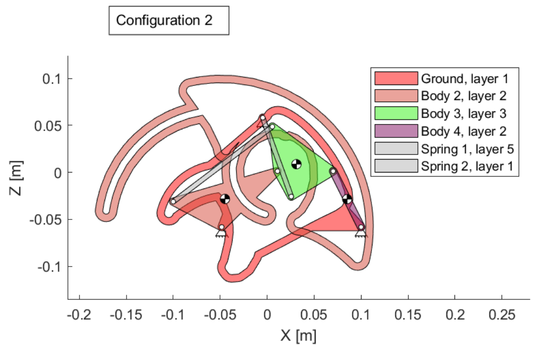
(a) Configuration 1 (b) Configuration 2 (c) Configuration 3

Figure C.3: Manufacturable configurations A-frame

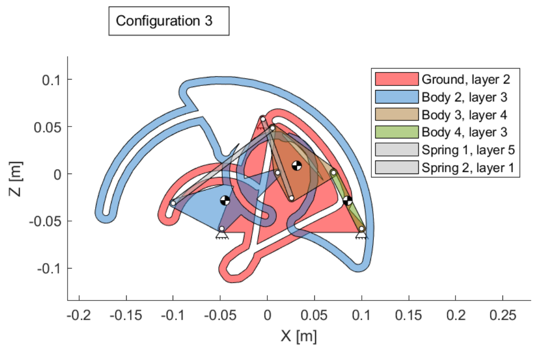
C.3. Roberts spring mechanism



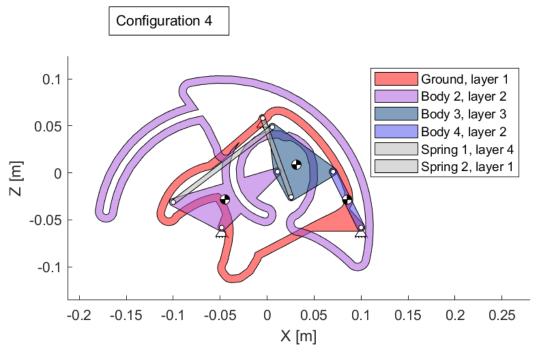
(a) Configuration 1



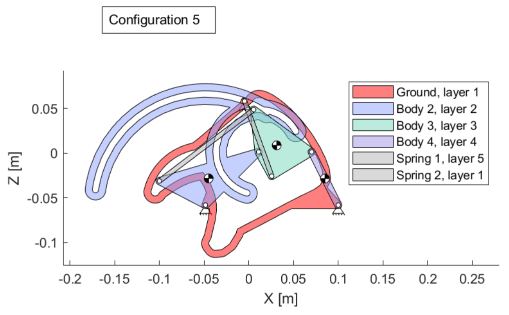
(b) Configuration 2



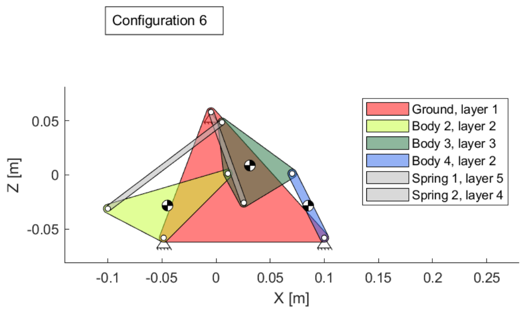
(c) Configuration 3



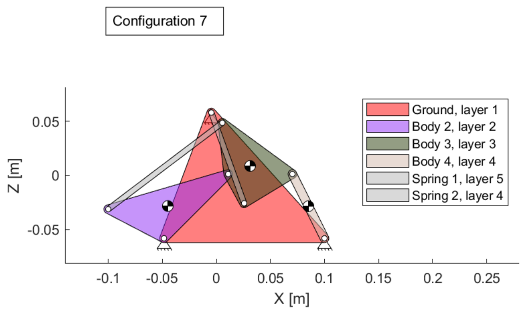
(d) Configuration 4



(e) Configuration 5



(f) Configuration 6



(g) Configuration 7

Figure C.4: Manufacturable configurations Roberts spring mechanism

D
Errata

1. **Title page** Added committee members
2. **p. i** Fix spelling in ’Thanks you’
3. **p. 16** Fix reference of sixth body to second body.
4. **p. 27** Fix body index in table 5.1 and table 5.2.
5. **p. 60** Fix wrong spring parameters for obtaining manufacturable designs. Obtained five additional manufacturable designs.