Automated Selection of Modelling Coordinates for Forward Dynamic Analysis of Multibody Systems

by

Mathieu Serge Léger

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2006

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Mathieu Léger

Abstract

Modelling mechanical systems using symbolic equations can provide many advantages over the more widely-used numerical methods of modelling these systems. The use of symbolic equations produces more efficient models, which can be used for many purposes such as real-time simulation and control. However, the number, complexity, and computational efficiency of these equations is highly dependent on which coordinate set was used to model the system.

One method of modelling a mechanism's topology and formulating its symbolic equations is to model the system using a graph-theoretical approach. This approach models mechanisms using a linear graph, from which spanning trees can be used to define a mechanism's coordinate set. This report develops two tree selection algorithms capable of estimating the tree set, and hence coordinate set, that produces models having the fastest forward dynamic simulation times.

The first tree selection algorithm is a heuristic-based algorithm that tries to find the coordinate set containing the minimal possible number of modelling variables. Most of this algorithm's heuristics are based on tree selection criteria found in the literature and on observations of a series of benchmark problems. It uses the topology information provided by a system's graph to find the coordinates set for the given system that produce very low simulation times of the system.

The second tree selection algorithm developed in this report also uses graph theory. It bases most of its heuristics on observations of one of the methods developed to obtain a mechanical system's symbolic equations using graph theory. This second algorithm also makes use of, and improves upon, a few of the heuristics developed in the first tree selection algorithm.

A series of examples for both algorithms will demonstrate the computational efficiency obtained by using the modelling variables found by the automated tree selection algorithms that are proposed in this report.

Acknowledgments

First, I would like to thank my supervisor, Prof. John McPhee, for the many insightful conversations we had during the development of this thesis. I could not have imagined a better advisor and mentor. His vast knowledge and perceptiveness were invaluable throughout my degree. I would also like to thank him for his generous financial support.

I am also very grateful for the financial support provided to me by the Natural Sciences and Engineering Research Council.

I would like to thank the members of my committee, Prof. Gordon Savage and Prof. Paul Calamai, for finding the time to read my thesis and provide their valuable comments.

Part of the thesis work was done while I was a visiting scholar at "L'Ecole d'ingénieurs et d'architectes de Fribourg" in Switzerland. I would like to thank Prof. Jacques Bersier for providing this great opportunity as well as his help, assistance, and financial support during this time. I would also like to thank Prof. Jean-Daniel Luthi and Prof. Olivier Grandjean for helping me with my research. Finally, I am grateful to everyone in Fribourg that made my stay there such a pleasant and enriching experience.

Furthermore, I would like to thank Chad Schmitke for his insightful responses to my many questions. I would also like to thank Kevin Morency, Adel Izadbakhsh, and Kiumars Jalali for all their help and for all the fun times we had working together. I also cannot forget to thank all my friends for getting me away from this research from time to time, and for making the last two years so pleasant. I would also like to thank Shirley Fleischer for our great discussions, and her wise words.

I am also very grateful to my family. I would particularly like to thank my parents, who have done all they could and more to get me to this point. Their constant encouragement and support throughout the years have been simply outstanding, and it is thanks to them that I am who I am today.

Finally, I would like to thank Myriam Fleischer for being the very special person she is. I am so grateful to her for taking the time for the arduous task of proofreading my thesis and for the incredible amount of patience she had with me during the last few months. I would also like to thank her for all her wonderful words of encouragement as well as her constant help and support. She has made the last two years the best years of my life. I simply can't thank you enough Myriam!

Contents

List of Tables

List of Figures

Chapter 1

Introduction and Literature Review

1.1 Background

There exist a number of techniques for automatic modelling of the kinematics and dynamics of multibody systems. Most techniques, such as the ones used by software packages such as ADAMS, DADS and Working Model, rely heavily on numerical methods to model such systems. These techniques model a mechanical system in a relatively simple, generalized, and inefficient manner, such as using Absolute Coordinates. This results in the need to use very large numerical matrices that must be re-evaluated for each time step in order to simulate a mechanical system. Because of this, these numerical methods are often slow and cumbersome. Furthermore, they keep the system model hidden from the user, who is only provided with numerical results.

Another technique that can be used to model a mechanical system is to find the system's governing equations in a symbolic form (e.g. hand derivation). Mechanical systems are modelled with two types of equations, kinematic and dynamic equations. Kinematic equations are non-linear algebraic equations used to describe the constraints existing between the different coordinates of the system. Dynamic equations, on the other hand, are differential equations that keep track of the forces within the system. Together they form a set of DAEs (Differential Algebraic Equations). Unlike the numerical models presented above, these equations only need to be formulated once and are valid at every time step,

which can lead to models that are much more efficient. In this case and throughout this report, an *efficient model* will be considered as a model for which the simulation time is minimized when solved using a standard numerical approach.

The generation of symbolic equations can be simplified and multiplications by zero or one can be eliminated from the equations. Furthermore, a mechanism's symbolic equations can be sent to code generation optimization routines, later referred to as code optimization, capable of restructuring the computation sequence of the procedure code containing the equations so that repeated terms only need to be evaluated once. These techniques can help increase the computational efficiency of the mechanism's model and hence helps reduce its simulation time.

The added efficiency of using symbolic equations to model mechanical systems makes them essential for real-time simulations. Furthermore, mechanism models that are composed of symbolic equations are rather small in size compared to their numerical counterparts that have to save large matrices of numerical values. The combination of fast simulation time and minimal storage makes symbolic models ideal for implementation within a mechanism's control algorithm. Furthermore, these same characteristics make it possible to create much more complex and complete models capable of simulation times that are equal to numerical models where many features such as friction and contact models are omitted for the sake of model efficiency.

Finally, since the system's equations are not hidden from the user, he can gain further insight on the model's characteristics by observing and dissecting its equations. In addition, since symbolic models are used in most physics and dynamics courses at school and university, the pedagogical aspect of these equations can not be overlooked.

Traditionally, these symbolic models were derived by hand - a process that was both tedious and error-prone. This is especially true in the field of robotics where a manipulator's symbolic equations can prove quite useful. Many methods of deriving symbolic equations of individual complex manipulators, especially parallel manipulators, have been developed over the years, such as the Stewart-Gough model developed by Liu et al. [17].

To eliminate the need of such complex hand derivations, methods of automating the formulation of a mechanism's symbolic equations have been developed [1, 15, 16, 19, 31]. Many of these methods are based on the use of graph methods such as Bond Graphs and

Linear Graphs. This report will focus on the latter of these techniques.

Since one of the primary objectives of modelling mechanical systems using its symbolic equations is to generate efficient simulations, a large amount of care must be taken to ensure that every possible method of model simplification is considered when generating such models. Methods of equation simplification and code optimization can be performed after the formulation of a given model's symbolic equations. Moreover, one can also focus one's attention on methods that can be used prior to the formulation of the mechanism's symbolic equations to ensure that these equations will be generated in the most efficient form possible.

When generating a set of symbolic equations used to model mechanical systems, a set of coordinates or modelling variables, in which these equations will be formulated, must be chosen. The chosen coordinates can be directly related to the number and complexity of the equations selected. Because of this, it is extremely important to choose coordinates wisely. The present report will focus on the automated selection of a mechanism's coordinate set so that the simulations of this mechanism can be very efficient.

1.2 Literature Review

Most of the commercial software used to model mechanical systems, such as DADS, Adams and Working Model, use *Absolute Coordinates* [8] to model these mechanical systems. In this coordinate set, the position and orientation of each body is described relative to the global (also called ground) reference frame. For general spatial systems, this results in each body's translation being modelled by its three Cartesian Coordinates (x_i, y_i, z_i) relative to the global reference frame. Its rotation is modelled by three rotations relative to the global reference frame (ζ_i, η_i, x_i), which can take a few different forms, such as the various types of Euler angles.

This coordinate set is generally used because the formulation of kinematic and dynamic equations is easily programmable when the system is modelled using these coordinates and it does not require intelligent coordinate selection.

However, this coordinate set models mechanisms using a very large number of modelling variables. Mechanical models require the use of one dynamic equation for each modelling

variable and require $n - DOF$ number of kinematic equations, where n is the number of variables used to model the system and DOF represents the system's degrees of freedom. The degrees of freedom of a system is the number of independent displacements that have to be specified in order to locate all parts of the mechanism. Modelling a system using the large number of modelling variables required by Absolute Coordinates results in the mechanism being modelled using a very large number of dynamic and kinematic equations. Solving all of these equations simultaneously is slow and inefficient.

Other methods of modelling mechanical systems use *Joint Coordinates*. This coordinate set describes the position and orientation of a body relative to its adjacent body in the system, which results in the variables of the various joints in the system being used to model the system. Joint Coordinates often prove more efficient than Absolute Coordinates, since it models the system using fewer coordinates, thus necessitating fewer equations. For example, when using Joint Coordinates to model serial mechanisms, the number of modelling variables equals the system's DOF. However, the equations generated using Joint Coordinates are generally more complex in nature than the many equations used in the case of Absolute Coordinates. The equations generated using Joint Coordinates are also more highly-coupled.

In systems having closed-loops, the values of some joint variables depend on the values of other joint variables. Wittenburg [31] used the concept of cut-joints to deal with this problem. In this method, a joint variable is removed from the modelling variables of each closed-loop in the system. Then, all the constraints and all the inertial joint forces are re-introduced in the system thus generating a series of kinematic equations (also called constraint equations) to be solved simultaneously with the system's dynamic equations.

To insure the selection of appropriate cut-joints from a mechanism's model, Wittenburg [31] created a *linear graph* model of the system. A linear graph is a series of nodes connected together by edges. In this graph, he depicted the mechanism's bodies as nodes and depicted each of the system's joints as edges relating these nodes together. The use of graph theory to model mechanical systems was first proposed by Andrews and Kesavan [1], who used graph theory to model multidimensional particle mass systems. This theory was improved by many subsequent authors and was eventually generalized to model spatial multibody systems of rigid bodies [16, 19, 31]. In graph theory, a *spanning tree*, often shortened to

simply *tree*, is a series of edges that connect to every node but do not form a closed-loop. Every edge not found in the tree is considered in the co-tree. Wittenburg stated that the Joint Coordinates generated by the variables found in each edge of a system graph's valid tree constitutes a set of suitable Joint Coordinates, with the edges in the cotree being appropriate cut-joints.

Wittenburg also established a criterion for coordinate selection that stated that *the systems using the fewest number of modelling variables, and hence the fewest number of system equations, will result in the most efficient coordinate set, since each equation will add extra complexity to the model.* He used this criterion to establish a method of selecting appropriate cut-joints/cotree edges by proposing that the joint having the fewest number of constraints should be used as cut-joints, since this would result in fewer kinematic equations used to model the system. However, Wittenburg did not propose any criteria for choosing between joints with a similar number of constraints.

Li and Andrews [16] also used a graph-theoretical approach to model mechanical systems. However, as opposed to Wittenburg, they used separate graphs for translation and rotation, and hence separate trees, to model mechanical systems. They also added body edges, which related each body node to the ground, in their graphs. This had a large effect on coordinate selection, since it was now possible to select body edges in the tree and hence model the systems using Absolute Coordinates. Furthermore, it allowed different cut-joints to be selected for translation and rotation.

As a coordinate selection criterion, Li and Andrews also used the criterion of selecting a minimal set of modelling coordinates to model the system. They provided a list of edges to be placed in the tree and cotree of each of the two graphs. This list took advantage of the dual trees. For example, it placed the revolute joints in the translational tree, as they do not include modelling variables in translation. It placed these revolute joints in the rotational graph's cotree because, in this case, it would include one modelling variable to the system if it were not used as the cut-joint.

However, Li and Andrews's tree and cotree lists only deal with planar systems and are rather strict, requiring certain edges to always be in the tree or cotree with sometimes conflicting results. It also strayed from the use of Joint Coordinates by always placing body edges in the rotational tree and revolute joints in the rotational cotree.

Later, Huston et al. [9] analysed the efficiency of the use of Absolute Coordinates to model multibody systems. They found that models of planar systems and spatial systems composed of purely spherical joints could be simulated more efficiently if they were modelled using Absolute Coordinates for rotation and Joint Coordinates for translation. This set of coordinates is called *Absolute Angular Coordinates*. The improved efficiency obtained by Absolute Angular Coordinates justifies Li and Andrews's choice of using body rotation variables instead of the rotation variables of revolute joints to model planar systems. It also shows that the selection of the most efficient set of coordinates is more complex than simply selecting appropriate Joint Coordinates because other coordinate types can prove more efficient. Finally, Huston et al.'s findings agree with the general concept that modelling systems with the fewest possible modelling variables produces more efficient simulations since the use of Absolute Coordinates in the circumstances they described also use a minimal set of coordinates.

In this same year, Fayet and Pfister [5] established a more general set of modelling coordinates for open-loop systems called *Indirect Coordinates*. In this set of coordinates, the motion of each of the system's frames can be described relative to any other frame in the system. The "joints" used to measure the relative motion of two frames where no actual joint exists were called virtual joints.

Fayet and Pfister showed that for three specific system geometries, the use of some Indirect Coordinates can result in very efficient system models. They first showed that if two revolute joints have parallel axis, it is best to measure the rotation of the distal frame of the distal joint relative to the proximal frame of the proximal joint, where the term *distal* signifies farther from the ground, while the term *proximal* signifies closer to ground. Secondly, they extended this rule to two prismatic joints with parallel axes. In this case as well, it is preferable to measure the displacement of the distal frame of the distal joint relative to the proximal frame of the proximal joint. Finally, Fayet and Pfister noted that it is better to model the distal body attached to a spherical joint using Absolute Angular Coordinates as Huston et al. [9] had concluded. Indirect Coordinates were later extended to closed-loop systems by Redmond and McPhee [25].

Once again, the situation in which Fayet and Pfister demonstrated the added efficiency of Indirect Coordinates fell within the concept of selecting a minimal set of modelling

coordinates.

There also exists a coordinate set called *Natural Coordinates* [3], as well as its closely related *Point Coordinates* [23]. These coordinates use the Cartesian Coordinates of points for planar mechanisms or unit vectors in spatial mechanisms, placed at various strategic points on the mechanism's bodies to model the system in purely Cartesian Coordinates. In an analysis done by Unda et al. [29], planar mechanisms modelled using Natural Coordinates are shown to produce more efficient simulations than those modelled using Absolute Coordinates. However, the use of purely Cartesian Coordinates can prove cumbersome when dealing with angular quantities. These coordinate sets usually model mechanical systems with a number of modelling coordinates that is higher than those of Joint Coordinates and lower than those of Absolute Coordinates.

In addition, there exist some authors [12, 20] who have proposed coordinate sets using velocity transformations. These methods express the system's symbolic equations relative to velocities, found through the use of velocity transformations, that are not the direct time derivatives of the position variables used to model the system. These coordinate sets can be shown to produce more efficient simulations if the velocity transformations are selected wisely.

McPhee [19] also developed a graph-theoretical approach to model multibody systems. As with Li and Andrews [16]'s technique, McPhee used two graphs to model mechanical systems, one for rotation and one for translation. McPhee also established a list of edges that should be found in the trees of planar mechanisms. Unlike Li and Andrews's lists, McPhee's list did not force any edge in the tree or cotree and simply provided lists of the graph's edges in the order in which they should be selected in the trees. The tree selection preference lists developed by McPhee were also based on the selection of a coordinate set having a minimal set of modelling variables. As with all previous coordinate selection criteria, no guidelines are given in order to distinguish between coordinate sets having the same number of modelling variables to the system.

McPhee [19] introduces the term *Branch Coordinates* to refer to the coordinate set obtained by the selection of appropriate trees from the system's graphs. He later generalizes his tree edge preference list to spatial systems [18] and Redmond and McPhee [25] demonstrate the possibility of adding virtual joints to mechanisms modelled using a graph-

theoretical approach. This allows the selection of Branch Coordinates corresponding to the Indirect Coordinates proposed by Fayet and Pfister.

The only fully automated coordinate selection algorithm was developed by Kim and Vanderploeg [12] who, like many others, looked into the selection of the system's coordinates using graph theory to enforce a minimal modelling variables approach to coordinate selection. To every edge representing a joint in the graph, Kim and Vanderploeg gave a weight corresponding to the number of modelling variables of the edge. They then used a minimal spanning tree algorithm developed by Kevin and Whitney [11] to find the tree with the minimal weight.

However, this algorithm is very limited. First of all, as with Wittenburg's tree selection criterion, Kim and Vanderploeg's algorithm selects a coordinate set having the fewest modelling variable. However, Kim and Vanderploeg's algorithm is incapable of distinguishing which coordinate set is the most efficient among coordinate sets composed of the same number of modelling variables and in such cases it simply selects one at random. This problem is not trivial since many mechanical systems can be modelled with a large number of possible coordinate sets each having the same low number of modelling variables.

Secondly, Kim and Vanderploeg's algorithm was used strictly under the context of Joint Coordinates. This is because their graph edges only represent joints and as such they do not relate every body to the ground with an edge as done by Li and Andrews [16] and McPhee [19]. Also, there are no virtual joints present in Kim and Vanderploeg's models. Furthermore, one single graph was used in their coordinate selection algorithm. This does not allow a separation of the coordinate selection process into rotation and translation such as the ones proposed by Li and Andrews, and McPhee.

1.3 Thesis Goals and Structure

From the analysis of the previous section, it is clear that there is no coordinate type that always produces the most efficient simulations and that different mechanical systems will benefit differently from each coordinate set mentioned.

Also, much of the literature deals with the advantages and disadvantages of one particular type of coordinate set. There is very little documentation of how various coordinate

sets can be used in conjunction, or how the formation of a hybrid coordinate set affects the system equations.

However, one coordinate selection criterion that seems to be generally accepted is the fact that modelling a system with the minimal set of modelling coordinates will require fewer equations and will hence provide more efficient simulations. This concept was even the basis for the only known fully automated coordinate selection algorithm [12] developed thus far. However, it can be noted that this criterion has not been extensively studied and no mathematical proof of its validity has been established.

Furthermore, no general criteria is ever given in order to select between sets of coordinates comprised of the same number of modelling variables. However, the studies made of various specific coordinate sets, as well as comparison studies between these coordinate sets [5, 9, 25, 29], could probably shed some light on this problem and help to establish such criteria.

Linear graph theory has been widely used when trying to establish coordinate sets. This is because linear graph theory provides information on a model's topology. Topology refers to how a system's various components are connected together. This topology information proves to be crucial to proper coordinate selection. For example, graph theory is capable of identifying closed kinematic chains within a system. It also provides methods, such as tree selection [12], capable of properly identifying the number of closed chains and indicating potential locations one can divide these chains for proper system analysis.

Another great advantage of methods using linear graph theory to model mechanical systems is that the formulation of the equations can be automated by using graph-theoretical techniques. This saves countless time and energy that would be required to formulate the symbolic equations of many mechanical systems for many different coordinate sets required when establishing tree selection criteria and coordinate selection algorithm validation. Finally, these methods are often very flexible and allow the possibility of modelling systems using many different coordinate sets as well as allowing mechanisms to be modelled using hybrid coordinate sets.

From the graph-theoretical methods capable of modelling mechanical systems, McPhee's method [19, 25, 26] seems to provide the most broad and flexible coordinate selection properties. This method is presently capable of modelling systems using Absolute

Coordinates, Joint Coordinates, Absolute Angular Coordinates, Indirect Coordinates as well as combinations of these coordinates [25].

The goal of this research will be to develop a tree selection algorithm capable of quickly and efficiently selecting coordinates that are as close to the system's optimal coordinates as possible. Here, the term *optimal* coordinates shall be used to define the coordinates that provides the most efficient simulations (simulations having the lowest solution times) for a given model and which are included within the coordinate sets considered in this report.

These new tree selection algorithms will be developed using a graph-theoretical approach similar to Kim and Vanderploeg's [12] coordinate selection algorithm. In other words, the coordinates selection shall be done by the selection of minimal spanning trees from the mechanism's graphs. This shall be done within the context of McPhee's graph-theoretical approach to modelling mechanical systems [19, 25] since it seems to provide the broadest and most flexible coordinate selection properties.

The selection of McPhee's graph-theoretical approach will not allow the selection of purely Cartesian coordinate sets such as Natural and Point Coordinates. Furthermore, velocity transformation methods shall be beyond the scope of this report; though, these are included within McPhee's approach by the selection of different trees at the position and velocity levels.

The major types of analyses that are typically performed on models of mechanical systems are kinematic, static, inverse dynamic, and forward dynamic simulations. Of these, the forward dynamic analysis, which requires the calculation of the mechanism's motion when forces are applied to the mechanism, is the slowest and most computationally intensive. This is because it requires the simultaneous solution of the system's complete set of DAEs. For these reasons, the coordinate selection algorithm developed in this report will focus on improving the simulation efficiency for forward dynamic simulations.

This report will be divided in five chapters of which the present chapter is the first. Chapter 2 will present an overview on the use of graph-theoretical methods of formulating mechanical system's symbolic equations. In this chapter, a particular attention will be placed on the methods used by McPhee [19], since this is the equation formulation that shall be utilized throughout this report.

The two following chapters will each present a proposed tree selection algorithm. Chapter 3 shall present a method based on tree selection criteria found in the literature as well as heuristics based on simulation observations of multiple benchmark problems and will be called the *Variable-Based Coordinate Selection*. This method shall use the concepts of Kim and Vanderploeg's tree selection algorithm [12] as a basis for its development. Chapter 4 will present a second tree selection algorithm based on the equation formulation procedure used by McPhee [19] to formulate a mechanism's symbolic equations using graph theory. This tree selection method shall be called the *Formulation-Based Coordinate Selection*.

Chapter 5, the final chapter of this report, will present conclusions and discuss the results obtained in this report. It will also contain suggestions for future work in the field of coordinate selection.

Chapter 2

Multibody System Dynamics

This chapter presents a short overview of how the symbolic equations of multibody systems are obtained using graph theory. First a brief description of graph theory as applied to mechanical systems is presented. To illustrate this concept, an example will be presented in parallel with the theory. Finally a brief description of a computer implementation of the symbolic equation formulation method for mechanical systems discussed in this chapter will be presented.

2.1 Graph Theory Applied to Mechanical Systems

In this report, mechanical systems are modelled using graph theory [19]. Graph theory is a method that is used to describe a system's topology and then uses this description to systematically obtain the system's symbolic equations. To accomplish this, the topology of the system is described using *nodes* and *edges*. Nodes are schematically represented by dots and, in mechanical systems, represent body-fixed reference frames. Edges are schematically represented by arrows that connect two nodes together. In mechanical systems, an edge represents a component of the system and defines the allowable motions and forces between the two reference frames it connects.

Some of the components, or edge types, used to model mechanical systems are as follows:

Rigid Body Elements (m): Body elements start at the ground node and end at the node representing a reference frame at the center of mass of a rigid body. The body element contains the information necessary to describe the body, such as its mass and inertia matrix.

Rigid Arm Elements (r): Arm elements are used to define new reference frames, relative to the mass center frame, at fixed locations on a rigid body. They start at the center of mass node of the body and end at the desired node (reference frame).

Joint Elements: Joint elements define the allowable motions between two bodies comprising a kinematic pair. There is a different edge type for each different joint. The joint elements considered in this report are: revolute joints (h), prismatic joints (s), universal joints (u), spherical joints (b), planar joints (p), cylindrical joints (c), weld (0- Degree of Freedom (DOF)) joints (w), free (6-DOF) joints (fr), and XYZ (3-DOF) translational joints (tr).

Motion Drivers (md): This component is used to define a prescribed motion between two nodes.

Force/Torque Drivers (fd/td): This component is used to define forces and torques between two nodes.

Spring-Damper-Actuators (SDA): This component represents a spring, damper, and force/torque actuator that act in parallel between two reference frames.

An example of how a mechanism's graph is created is depicted in Figures 2.1 and 2.2, where Figure 2.2 presents the graph of the slider-crank mechanism shown in Figure 2.1. The graph edges m_1-m_3 represent the three rigid bodies in the mechanism. The edges r_4-r_7 are used to define the positions, relative to the body's center of mass frame, where the joints are connected to each of the bodies. The edges h_8-h_{10} characterize the three revolute joints in the system, while the edge s_{11} represents the prismatic joint connecting the third body to the ground. Finally, the edge td_{12} represents the torque applied to the first body, the crank.

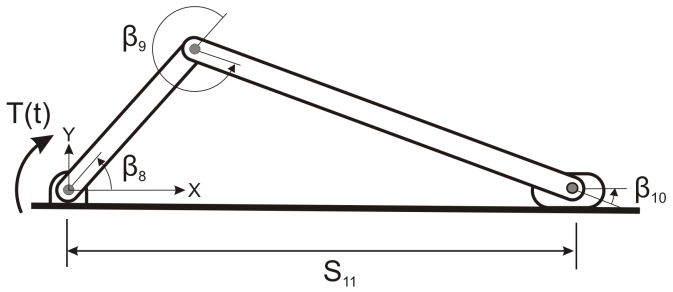


Figure 2.1: Slider-crank mechanism.

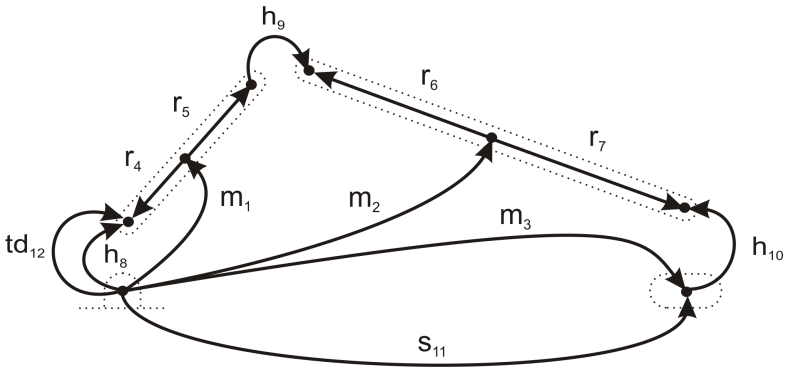


Figure 2.2: Linear graph of the slider-crank mechanism.

There are two important types of subsets used in graph theory. The first type is called a *circuit* and is composed of a series of edges connected together to form a closed-loop. In the graph found in Figure 2.2, the edges m_1, r_4 and h_8 form a circuit.

The second subgraph of importance is called a *cutset*. A cutset consists of a series of edges that, if removed, would separate the graph in two unconnected parts. A cutset also has the restriction that no subset of this subgraph possesses the property of separating the graph in two unconnected parts. The subgraph created by the edges m_2, h_9 and h_{10} in the graph found in Figure 2.2 are a good example of a cutset. If these edges were removed the graph would be divided into two unconnected sections because the section consisting of edges r_6 and r_7 would be separated from the rest of the graph.

In graph theory there are *across variables* and *through variables* associated with each edge. In the case of a mechanical system, the across variables are vectors representing the relative position, velocity, and acceleration of the two nodes that the edge connects. These variables are divided into the translational and rotational domains. In the translational domain, the across variables are expressed as the vectors $\vec{r}$ for displacement, $\vec{v}$ for velocity, and $\vec{a}$ for acceleration.

In rotation, there exists one notable exception to the use of vectors as across variables. In this case, due to the fact that rotations are non-commutative, rotation matrices are used to describe rotation. In this report, a rotation matrix is expressed by the symbol $[R(\theta_1, \theta_2, \theta_3)]$, where θ_1, θ_2, and θ_3 are the variables representing each of the three possible rotations. In the case where an edge allows fewer than three rotations, the symbols $[R(\theta, \widehat{u})]$ and $[R(\theta_1, \theta_2, \widehat{u}_1, \widehat{u}_2)]$ are used to represent rotation matrices that define rotations about one and two axes respectively. Here, the unit vectors about which the rotations occur are expressed by $\widehat{u}$. These rotation matrices can represent body-fixed or space-fixed rotations. Rotational velocities and accelerations, on the other hand, are commutative and can therefore be expressed in vector form as $\vec{\omega}$ and $\vec{\alpha}$ respectively. Finally, the vector space spanned by the time-variant across variables is called the *motion space* of the edge (e.g. $\widehat{u}$ for rotation about a single axis).

Through variables are vectors representing the relative forces and torques between the two nodes that the edge connects. These can also be divided into the translational and rotational domains as force vectors $\vec{F}$ and torque vectors $\vec{T}$ respectively. The *reaction space*

of the edge can be defined as the vector space spanned by these through variables.

The through and across variables are regulated by two postulates. The first postulate, known as the *Circuit Postulate*, stipulates that the directed sum of the across variables in any circuit must equal zero, where the term *directed sum* refers to the fact that each edge's orientation is taken into account during the summation. Once again, there is one important exception to this postulate in the case of rotations. In this case, since rotations are not expressed by vectors, instead of adding the rotation matrices in a circuit, the product of these matrices in the order in which they appear in the circuit is used instead. An example of the application of the circuit postulate in the context of displacements on a circuit of the slider-crank graph (Figure 2.2) is presented in equation (2.1). When dealing with rotations, the application of the circuit postulate on this same circuit changes to the form shown in equation (2.2), where $\vec{r}_1$, $\vec{r}_4$, $\vec{r}_8$ represent the displacement vectors associated to the edges r_1, r_4 and r_8 respectively. The rotation matrices of the edges r_1, r_4 and r_8 are represented by $[R_1]$, $[R_4]$ and $[R_8]$ respectively and $[I]$ represents an identity matrix. The rotation matrix $[R_8]$ is transposed in equation (2.2) due to the fact that edge r_8 point in the opposite direction than the edges r_1 and r_4 in the circuit (shown in equation (2.1) by the minus sign before the vector $\vec{r}_8$). In mechanical systems the Circuit Postulate corresponds to the summation of vector displacements (or velocities, or accelerations) that should be equal to zero for any given closed kinematic chain. The equation obtained by applying the Circuit Postulate to a circuit subset found in the graph is called a *circuit equation*.

$$\vec{r}_1 + \vec{r}_4 - \vec{r}_8 = 0 \tag{2.1}$$

$$[R_1][R_4][R_8]^T = [I] \tag{2.2}$$

The second postulate is called the *Vertex Postulate* and states that the directed sum of the through variables at each node must equal zero. From the point of view of mechanical systems, this postulate simply ensures that at every node in the model, the sum of the forces, as well as the sum of the torques, will equal zero. The Vertex Postulate can be generalized to stipulate that the directed sum of the through variables of the edges in a cutset equal zero. This wider definition of the Vertex Postulate allows one to get the dynamic equilibrium for any body or combination of bodies of a mechanical graph. The

equation obtained by applying the Vertex Postulate to a cutset in the graph is called a *cutset equation*.

Finally, each edge is assigned *terminal equations*, determined from experiments, that regulate the edge's across and through variables and their relationship to each other. The terminal equations associated with four of the most commonly-used components are presented in Appendix B.

2.2 Spanning Trees

The two postulates presented above, combined with the terminal equations of each edge, are sufficient to form a set of equations capable of obtaining a system's symbolic equations. However, to automate the extraction of a system's symbolic equations, some type of formulation procedure, which organizes the application of each postulate and terminal equation, must be used. There exist a few different formulation procedures such as the *nodal formulation* [13], the *branch-chord formulation* [19] and the *tableau methods of formulation* [15]. In this report, the branch-chord formulation procedure, which is based on the selection of a *spanning tree*, will be used. This will be augmented by orthogonal projection techniques to eliminate non-working constraint reactions, which was developed by McPhee [19] and will be presented in detail in Section 2.3. A graph-theoretical approach based on analytical mechanics (i.e. virtual work) was also developed by Shi and McPhee [26]. This method is especially useful when trying to model mechanisms with flexible bodies. However its development shall not be demonstrated in this report.

A spanning tree, simply referred to as a *tree*, is a series of edges, called *branches*, that reach all the nodes in the graph without forming a closed-loop. All edges not included in the tree are called *chords* and form the *cotree*. In mechanical systems there are two trees, one for translation and one for rotation. An example of the trees that can be selected for the slider-crank depicted in Figure 2.1 are shown in Figures 2.3 and 2.4, where the branches are depicted in bold. When the system's governing equations are derived using a certain tree, the equations of motion of this system will be expressed relative to the branch across variables and the chord through variables, which together are called the *primary variables*. In contrast, the branch through variables and the chord across variables are called

secondary variables. This implies that, when using graph theory to model a mechanical system, optimal coordinate selection is accomplished by selecting the optimal trees for the system.

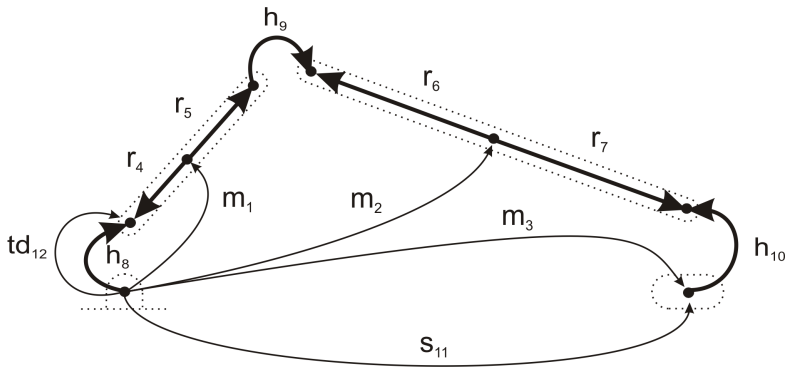


Figure 2.3: A translational tree of the slider-crank mechanism.

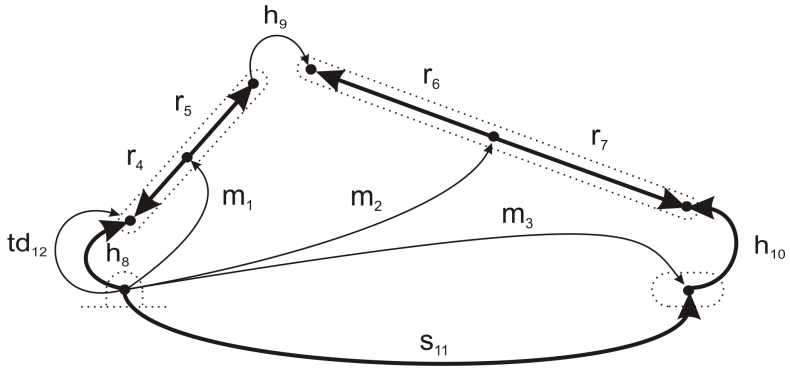


Figure 2.4: A rotational tree of the slider-crank mechanism.

Once a tree is established for a given graph, a new type of circuit and cutset, called *fundamental circuit* (*f-circuit*) and *fundamental cutset* (*f-cutset*), can now be defined. An f-circuit is a circuit that consists of one single chord and a series of branches. There exists one unique f-circuit for each chord in the graph. The equation obtained by using the f-circuit equation to express the chord's across variables as a function of branch across variables is called the *branch transformation equation*. On the other hand, an f-cutset is a cutset composed of a series of chords and one single branch. There exists one unique

f-cutset for each branch in the graph. The equation obtained by using the f-cutset equation to express the branch's through variables as a function of chord through variables is called the *chord transformation equation*.

A typical method used to select a tree is to use a weighted graph and to find the tree having the minimal weight, called a *minimal spanning tree*. A weighted graph is a graph for which each edge is assigned a weight based on how advantageous it is to put the given edge in the tree. The edges with the lowest weights are the ones that are the most advantageous to put into the tree. When modelling mechanical systems, there will be two different weighted graphs constructed for each system, one for translation and one for rotation. This will enable the selection of distinct translational and rotational trees.

There exist three major minimal spanning tree algorithms that are commonly-used and upon which many subsequent tree selection algorithms are based. These algorithms are Borůvka's algorithm [22], Kruskal's algorithm [14] and Prim's algorithm [24].

Borůvka's algorithm [22] is based on the premise that for every node in the system, the edge with the lowest weight attached to the given node is selected as a branch. This is done without taking into account if this edge was already placed in the tree by other nodes. If this does not immediately result in a valid tree for the graph, all the nodes related to each other by tree edges form what are called *super nodes*. Then the lowest weight edge connecting super nodes to each other is placed in the tree. This process continues until a final tree is selected.

A simple example of this process can be seen in Figure 2.5(a), where the number found next to each edge corresponds to this edge's weight. In the first step of Borůvka's algorithm, the node A adds edge BA to the tree since this is the minimal weight edge to which it is connected. In a similar fashion node B adds the same edge to the tree, node C adds edge CB and nodes D and E add edge DE. The resulting tree is shown in bold in Figure 2.5(a). However, this is not a valid tree since a tree must be a connected graph. The term *connected graph* refers to a graph where *each node found in the graph can be related to any other node in the graph by a series of the graph's edges*. In order to find a valid tree, nodes A, B, and C, which are connected together by tree edges, are grouped into super node F and nodes D and E are grouped together to form the super node G as shown in Figure

2.5(b). Super node F then adds edge CD to the tree, since the edge AC is considered part of the super node F. Finally the super node G adds the same edge to the tree, thus resulting in the minimal weight tree of the graph.

(a) (b)

Figure 2.5: A tree selection example.

Kruskal's algorithm [14] takes a different approach to tree selection. This algorithm progressively adds to the tree the lowest weighted edge not already in the tree whose two nodes are not found in the same tree section (a series of branches forming a connected graph). This is done repeatedly until a valid tree is selected.

Once again referring to Figure 2.5(a), the first edge that would be added to the tree would be edge BA since it has the lowest weight. The next edge added would be edge BC. At this stage it is important to note that edge AC can no longer be placed in the tree because both its nodes are part of the same tree section composed of the nodes A, B, C and their connecting tree edges. Because of this, the next edge selected in the tree will be the edge DE, which will provide the tree demonstrated in bold in Figure 2.5(a). Finally, the last edge to be added is the edge CD, since it has the lowest weight and has nodes in two different tree sections.

Prim's algorithm [24] starts by placing one of the graph's nodes in the tree. From there, the *potential edges* are found. The potential edges are edges already connected to one node in the tree and one node not yet in the tree. Of these potential edges, the one with the lowest weight is placed in the tree (as well as its nodes). This process continues until a valid tree for the graph is found.

Taking the same graph used to demonstrate the two other algorithms found in Figure 2.5(a), the first step is to place the node A in the tree. This node was selected randomly

since any of the graph's nodes can be placed in the tree in this first step. From there the potential edges consist of edges AB, AC and AD of which the edge AB is placed in the tree. Now the potential edges are BC, AC, and AD of which edge BC is placed in the tree. This same process is repeated adding the edge CD and finally the edge DE to the tree.

Minimal spanning tree algorithms require the assignment of appropriate weights to each edge of the graph. This aspect of tree selection proves to be complex in the case of mechanical systems, and will be a major focus of subsequent chapters of this report.

There exist other minimal spanning tree methods not presented above, some of which have improved solution time, especially for large graphs. Graham and Hell [7] give a particularly good overview of these methods spanning from Borůvka's's algorithm, the first known minimal spanning tree algorithm, to the invention of Fibonacci heaps. However, in the case of mechanical systems, whose graphs are relatively small, the solution time of such systems using the three algorithms presented above is already sufficient for our needs.

There are also other tree selection methods than the minimal spanning tree that can be used to select valid trees from graphs. For example there exist algorithms, such as Dijkstra's algorithm [4], which are capable of finding the shortest path between every node in the graph and, while doing so, they find a valid tree whose branches are the edges used by all these paths. For the time being, we shall not look into the details of other tree selection methods and algorithms. If, during the establishment of an appropriate tree selection algorithm for mechanical system, a certain tree characteristic requires a tree selection method with specialized features, these algorithms shall be discussed at this time.

2.3 Generating the System's Equations

Once a tree is selected, the formulation of the system's symbolic equations can be automated by four steps, developed by McPhee [19] and described below.

Step I

The first step is to obtain the dynamic equations of the system. These equations are generated by projecting the f-cutset equations of the bodies, passive joints, and the forces/tor-

ques in the trees onto their motion space. By projecting these equations onto their motion space, the reaction forces and torques of passive kinematic constraints are eliminated. This will generate one dynamic equations for each of the n coordinates in $\{q\}$, where $\{q\}$ represents the column matrix containing a given system's modelling variables. In the case of the trees selected in Figures 2.3 and 2.4, the coordinates of the systems will take the form $\{q\} = [\beta_8, \beta_{10}]^T$. Generally, the dynamic equations equations generated in this step will not correspond to the final form of the dynamic equations as they usually still include secondary variables that will be eliminated in subsequent steps of the equation formulation procedure.

To illustrate this process, the result of the first step when applied to the slider-crank depicted in Figure 2.3 and 2.4, is given in the following equations:

$$(\vec{T}_1 + \vec{T}_8 - \vec{T}_9 + \vec{T}_{12}) \cdot \widehat{k}_G = 0 \tag{2.3}$$

$$(\vec{T}_2 + \vec{T}_9 + \vec{T}_{10}) \cdot \widehat{k}_G = 0 \tag{2.4}$$

where $\vec{T}_i$ represents the torque vector of the i^{th} edge and $\widehat{k}_G$ represents a unit vector along the ground reference frame's Z axis.

These two equations were obtained from the f-cutset equations of the edges h_8 and h_{10} in rotation. In this case, none of the translational branches generate dynamic equations since none of them have any translational motion space. After substitution of the terminal equations presented in Appendix B, the equations (2.3) and (2.4) become:

$$(-\vec{\vec{J}}_1 \cdot \vec{\boldsymbol{\alpha}}_{\mathbf{1}} - \vec{\boldsymbol{\omega}}_{\mathbf{1}} \times \vec{\vec{J}}_1 \cdot \vec{\boldsymbol{\omega}}_{\mathbf{1}} - l_4\widehat{\imath}_1 \times \vec{\boldsymbol{F}}_{\mathbf{4}} - l_5\widehat{\imath}_1 \times \vec{\boldsymbol{F}}_{\mathbf{5}} + \vec{\boldsymbol{T}}_{\mathbf{8}} - \vec{T}_9 + \vec{T}_{12}(t)) \cdot \widehat{k}_G = 0 \tag{2.5}$$

$$(-\vec{\vec{J}}_2 \cdot \vec{\boldsymbol{\alpha}}_{\mathbf{2}} - \vec{\boldsymbol{\omega}}_{\mathbf{2}} \times \vec{\vec{J}}_2 \cdot \vec{\boldsymbol{\omega}}_{\mathbf{2}} - l_6\widehat{\imath}_2 \times \vec{\boldsymbol{F}}_{\mathbf{6}} - l_7\widehat{\imath}_2 \times \vec{\boldsymbol{F}}_{\mathbf{7}} + \vec{T}_9 + \vec{\boldsymbol{T}}_{\mathbf{10}}) \cdot \widehat{k}_G = 0 \tag{2.6}$$

where $\vec{\vec{J}}_i$ represents the inertia dyadic of the body represented by the edge m_i and $\vec{\omega}_i$, $\vec{\alpha}_i$ represent the rotational velocity vector and rotational acceleration vector of the i^{th} edge. The terms l_i represent the length of the arm element r_i, while the term $\vec{F}_i$ represents the force vector associated to the i^{th} edge. The terms $\widehat{\imath}_i$ represents the unit vector along the the i^{th} edge's end node's local X axis. Finally, $\vec{T}_{12}(t)$ represents the user-defined torque of the torque driver represented by the edge td_{12}.

The symbols depicted in bold are the secondary variables. This was done in order to help clarify the substitutions of secondary variables that will be performed throughout the equation formulation procedure. As mentioned previously, the equations (2.5) and (2.6) still include secondary variables such as the body rotational velocities and accelerations as well as the arm forces.

Step II

The second step consists of acquiring the system's kinematic equations, which are generated by obtaining the fundamental circuit equations for each chord having active or passive constraints and projecting them onto their reaction space. The number of kinematic equations will be equal to the number of modeling variables n minus the number of degrees of freedom possessed by the mechanical system. Once the terminal equations are substituted, these equations correspond to nonlinear algebraic equations that are functions of time. As with the dynamic equations, these equations generally still contain secondary variables that will be eliminated in the next step of the equation formulation procedure.

The kinematic equation for the slider-crank depicted in Figure 2.3 is obtained by using the fundamental circuit equation of s_{11} in translation and is given in equation (2.7). This fundamental circuit equation should also be projected onto the prismatic joint's second reaction vector ($\widehat{k}_G$). However, since this mechanism is planar, projecting the translational fundamental circuit equation onto $\widehat{k}_G$ will give a kinematic equation that can be simplified to $0 = 0$. This is also true for the projection of the fundamental circuit equation of h_9 in the rotational cotree onto its reaction space.

$$(\vec{r}_4 - \vec{r}_5 + \vec{r}_6 - \vec{r}_7 - \vec{r}_8 - \vec{r}_9 + \vec{r}_{10}) \cdot \widehat{\jmath}_G = 0 \tag{2.7}$$

where $\widehat{\jmath}_G$ represents a unit vector along the ground reference frame's Y axis.

The corresponding terminal equations can be substituted into equation (2.7) to obtain:

$$((l_4 - l_5)\widehat{\imath}_1 + (l_6 - l_7)\widehat{\imath}_2) \cdot \widehat{\jmath}_G = 0 \tag{2.8}$$

Step III

The third step in the equation formulation procedure is to eliminate any secondary variables appearing in the equations found in the first two steps. To do this, the branch and chord transformations are substituted into the equations. This process can be divided into three parts that must be performed in a specific order that is described below:

First, the rigid arm forces are found by using the chord transformation equations of the arm components in the translational graph. These equations are substituted into the dynamic equations. The rigid arm forces for the slider-crank depicted in Figure 2.3 are:

$$\vec{F}_4 = \vec{F}_1 + \vec{F}_2 + \vec{F}_3 + \vec{F}_{11} \tag{2.9}$$
$$\vec{F}_5 = -\vec{F}_2 - \vec{F}_3 - \vec{F}_{11} \tag{2.10}$$
$$\vec{F}_6 = \vec{F}_2 + \vec{F}_3 + \vec{F}_{11} \tag{2.11}$$
$$\vec{F}_7 = -\vec{F}_3 - \vec{F}_{11} \tag{2.12}$$

The terminal equations associated with each of the chord forces are then substituted, and the secondary variables are highlighted in bold, giving:

$$\boldsymbol{\vec{F}_4} = -m_1\boldsymbol{\vec{a}_1} - m_2\boldsymbol{\vec{a}_2} - m_3\boldsymbol{\vec{a}_3} + \vec{F}_{11} \tag{2.13}$$
$$\boldsymbol{\vec{F}_5} = m_2\boldsymbol{\vec{a}_2} + m_3\boldsymbol{\vec{a}_3} - \vec{F}_{11} \tag{2.14}$$
$$\boldsymbol{\vec{F}_6} = -m_2\boldsymbol{\vec{a}_2} - m_3\boldsymbol{\vec{a}_3} + \vec{F}_{11} \tag{2.15}$$
$$\boldsymbol{\vec{F}_7} = m_3\boldsymbol{\vec{a}_3} - \vec{F}_{11} \tag{2.16}$$

where m_i represent the mass of the body depicted by the edge m_i and $\vec{a}_i$ represents the translational acceleration vector of the i^{th} edge.

Secondly, the translational branch transformation equations are substituted in the dynamic and kinematic equations. The translational acceleration branch transformation equations for the slider-crank of Figure 2.3 are:

$$\vec{a}_1 = -\vec{a}_4 + \vec{a}_8 \tag{2.17}$$
$$\vec{a}_2 = -\vec{a}_4 + \vec{a}_5 - \vec{a}_6 + \vec{a}_8 + \vec{a}_9 \tag{2.18}$$
$$\vec{a}_3 = -\vec{a}_4 + \vec{a}_5 - \vec{a}_6 + \vec{a}_7 + \vec{a}_8 + \vec{a}_9 - \vec{a}_{10} \tag{2.19}$$
$$\vec{a}_{11} = -\vec{a}_4 + \vec{a}_5 - \vec{a}_6 + \vec{a}_7 + \vec{a}_8 + \vec{a}_9 - \vec{a}_{10} \tag{2.20}$$

The appropriate terminal equations are then substituted in the equations and the secondary variables are highlighted in bold, giving:

$$\vec{\boldsymbol{a}}_{\mathbf{1}} = -\vec{\boldsymbol{\alpha}}_{\mathbf{1}} \times l_4\widehat{\imath}_1 + \vec{\boldsymbol{\omega}}_{\mathbf{1}} \times (\vec{\boldsymbol{\omega}}_{\mathbf{1}} \times l_4\widehat{\imath}_1) \tag{2.21}$$
$$\vec{\boldsymbol{a}}_{\mathbf{2}} = -\vec{\boldsymbol{\alpha}}_{\mathbf{1}} \times (l_4 - l_5)\widehat{\imath}_1 + \vec{\boldsymbol{\omega}}_{\mathbf{1}} \times (\vec{\boldsymbol{\omega}}_{\mathbf{1}} \times (l_4 - l_5)\widehat{\imath}_1) - \vec{\boldsymbol{\alpha}}_{\mathbf{2}} \times l_6\widehat{\imath}_2 + \vec{\boldsymbol{\omega}}_{\mathbf{2}} \times (\vec{\boldsymbol{\omega}}_{\mathbf{2}} \times l_6\widehat{\imath}_2) \tag{2.22}$$
$$\vec{\boldsymbol{a}}_{\mathbf{3}} = -\vec{\boldsymbol{\alpha}}_{\mathbf{1}} \times (l_4 - l_5)\widehat{\imath}_1 + \vec{\boldsymbol{\omega}}_{\mathbf{1}} \times (\vec{\boldsymbol{\omega}}_{\mathbf{1}} \times (l_4 - l_5)\widehat{\imath}_1) - \vec{\boldsymbol{\alpha}}_{\mathbf{2}} \times (l_6 - l_7)\widehat{\imath}_2 + \vec{\boldsymbol{\omega}}_{\mathbf{2}} \times (\vec{\boldsymbol{\omega}}_{\mathbf{2}} \times (l_6 - l_7)\widehat{\imath}_2) \tag{2.23}$$
$$\vec{\boldsymbol{a}}_{\mathbf{11}} = -\vec{\boldsymbol{\alpha}}_{\mathbf{1}} \times (l_4 - l_5)\widehat{\imath}_1 + \vec{\boldsymbol{\omega}}_{\mathbf{1}} \times (\vec{\boldsymbol{\omega}}_{\mathbf{1}} \times (l_4 - l_5)\widehat{\imath}_1) - \vec{\boldsymbol{\alpha}}_{\mathbf{2}} \times (l_6 - l_7)\widehat{\imath}_2 + \vec{\boldsymbol{\omega}}_{\mathbf{2}} \times (\vec{\boldsymbol{\omega}}_{\mathbf{2}} \times (l_6 - l_7)\widehat{\imath}_2) \tag{2.24}$$

The position and velocity-level branch transformation equations are acquired in a similar fashion.

Finally, the rotational branch transformation equations are substituted in the dynamic and kinematic equations. The rotational velocity branch transformation equations for the slider-crank, depicted in Figure 2.4, are:

$$\vec{\omega}_1 = -\vec{\omega}_4 + \vec{\omega}_8 \tag{2.25}$$
$$\vec{\omega}_2 = -\vec{\omega}_7 + \vec{\omega}_{10} + \vec{\omega}_{11} \tag{2.26}$$
$$\vec{\omega}_3 = \vec{\omega}_{11} \tag{2.27}$$
$$\vec{\omega}_9 = \vec{\omega}_4 - \vec{\omega}_5 + \vec{\omega}_6 - \vec{\omega}_7 - \vec{\omega}_8 + \vec{\omega}_{10} + \vec{\omega}_{11} \tag{2.28}$$

After substituting the terminal equations, and highlighting the secondary variables in bold, these equations become:

$$\vec{\boldsymbol{\omega}}_{\mathbf{1}} = \frac{d}{dt}\beta_1(t)\widehat{k}_G \tag{2.29}$$

$$\vec{\boldsymbol{\omega}}_{\mathbf{2}} = \frac{d}{dt}\beta_3(t)\widehat{k}_{11} \tag{2.30}$$

$$\vec{\boldsymbol{\omega}}_{\mathbf{3}} = \vec{0} \tag{2.31}$$

$$\vec{\boldsymbol{\omega}}_{\mathbf{9}} = -\frac{d}{dt}\beta_1(t)\widehat{k}_G + \frac{d}{dt}\beta_3(t)\widehat{k}_{11} \tag{2.32}$$

where $\beta_1(t)$ and $\beta_3(t)$ represent the rotation of the revolute joints h_8 and h_{10}, as depicted in Figure 2.1. The terms $\widehat{k}_{11}$ and $\widehat{k}_G$ represent the unit vector about the local z axis of the edge h_{11}'s end node and the unit vector about the ground frame's Z axis respectively.

Step IV

The fourth and final step in formulating the system's symbolic equations is to evaluate the dot products and simplify and reorganize the equations so that they can be presented in a standardized form. In this standardized representation, the kinematic equations are written as a column matrix:

$$\{\Phi(q,t)\} = 0 \tag{2.33}$$

While the dynamic equations are generated in the form:

$$[M]\{\ddot{q}\} + \{\Phi\}_q^T\{\lambda\} = \{Q(q,\dot{q},t)\} \tag{2.34}$$

where $[M]$ is a symmetric $n \times n$ mass matrix, $\{\Phi_q\}$ is the Jacobian of the constraint equations (in equation (2.33)), $\{\lambda\}$ are unknown Lagrange multipliers associated with cotree joint reactions, and $\{Q\}$ contains forcing and quadratic velocity terms.

After evaluating the dot products and simplifying and reorganizing the equations for the slider-crank depicted in Figure 2.1, the following kinematic equation is obtained:

$$\Phi(\beta_1,\beta_2,t) = -(l_4 - l_5)\sin(\beta_1(t)) - (l_6 - l_7)\sin(\beta_3(t)) = 0 \tag{2.35}$$

Applying the same steps to the dynamic equations gives the following entries for equation (2.34):

$$M_{[11]} = Izz_1 + m_1 l_4^2 + (m_2 + m_3)(l_4 - l_5)^2 \tag{2.36}$$

$$M_{[12]} = M_{[21]} = (l_4 - l_5)(m_2 l_6 + m_3(l_6 - l_7))\cos(-\beta_1(t) + \beta_3(t)) \tag{2.37}$$

$$M_{[22]} = Izz_2 + m_2 l_6^2 + m_3(l_6 - l_7)^2 \tag{2.38}$$

$$\{q\} = \begin{Bmatrix} \beta_1 \\ \beta_2 \end{Bmatrix} \tag{2.39}$$

$$\{\Phi\}_q^T = \begin{Bmatrix} (l_4 + l_5)\cos(\beta_1(t)) \\ (l_6 + l_7)\cos(\beta_3(t) \end{Bmatrix} \tag{2.40}$$

$$\lambda = F_{11} \tag{2.41}$$

$$Q_{[1]} = (l_4 - l_5)((m_2 + m_3)l_6 - m_3 l_7)\sin(\beta_3(t) - \beta_1(t))(\frac{d}{dt}\beta_3(t))^2 + T_{12}(t) \tag{2.42}$$

$$Q_{[2]} = -(l_4 - l_5)((m_2 + m_3)l_6 - m_3 l_7)\sin(\beta_3(t) - \beta_1(t))(\frac{d}{dt}\beta_1(t))^2 \tag{2.43}$$

where $M_{[ij]}$ represents the value of the mass matrix's i^{th} row and j^{th} column, $Q_{[i]}$ represents the value of the $\{Q\}$'s i^{th} value, Izz_i represents the moment of inertia about the line through the centroid, parallel to the Z axis of the body represented by the edge m_i.

One can observe that, in this case, the planar nature of the slider-crank allowed for its symbolic equations to be greatly simplified, generating the compact model found in equations (2.35) to (2.43).

When modelling a system using the graph-theoretical approach presented previously, Branch Coordinates are inevitably used to model the system. The symbolic equations of a mechanism modelled in this fashion will be expressed in terms of the graph's primary variables (branch across variables and chord through variables). However, once the primary variables are known, if the values of secondary variables are required during the analysis of the model, these values can quickly be obtained through back-substitution using the branch and chord transformation equations.

For this reason, even if one variable is of particular interest in the system's analysis, it does not necessarily need to be explicitly found in the system's coordinates. Hence, the

most efficient set of modelling coordinates can always be used to model the system, which should help to increase model efficiency in many circumstances.

2.4 Computer Implementation

In order to test the validity of the tree selection algorithms developed in this report, these algorithms will be incorporated in DynaFlexPro, a Maple package that uses graph-theoretical methods to generate a system's equations of motion in symbolic form. It uses a Java graphical user interface called ModelBuilder to allow the user to describe the system that is being modelled. ModelBuilder then generates the graph model of the mechanism that is sent to the DynaFlexPro Maple package. DynaFlexPro uses this graph model to generate system equations using the methods described in the previous sections. These equations can then be solved within Maple or exported as optimized simulation code (e.g. C procedure or Matlab S-function).

In DynaFlexPro, it can take a certain amount of time to formulate a system model. For example, the 3-DOF spatial parallel manipulator presented in Appendix A.12 takes an average of 19.0 s to formulate and simplify the system's symbolic equations using the system's optimal coordinate set using a Pentium 4 of 1.80 GHz with 768 MB of RAM. However, other coordinate sets take up to 99.3 s for the exact same formulation and simplification process. Furthermore, using another coordinate sets, this same model's symbolic equations could not even be formulated and simplified within one hour. Since one of the goals of this report is to develop algorithms that are quick and efficient, it was deemed infeasible to use an optimization method to evaluate various fully-formulated system models since formulating many system models would be too time-consuming. Because of this, the tree selection algorithms in this report were developed in a pre-processing environment of ModelBuilder in which the system models did not have to be formulated.

ModelBuilder will enable the selection of the trees of the system's graph before it is sent to the DynaFlexPro equation generating routines. This will enable the user to have a visual outlook on the coordinate selection process, thus allowing much more interaction with the user so he can clearly see and understand which coordinates are being selected and their relation to the model.

By implementing the algorithm in ModelBuilder, which is built using the Java language, rather than implementing it directly in the DynaFlexPro Maple package, one can generate a much quicker tree selection algorithm due to the low-level programming language involved.

However, this choice is not without its drawbacks. By placing the tree selection algorithm in this pre-processing environment, some model information could prove difficult to obtain. Since the main function of ModelBuilder is to generate the system's graph, the topology of the mechanism will be easily accessible in this environment. However, the mechanism's geometry will be much more difficult to extract. Furthermore, the purely numerical nature of Java will make it difficult to predict the equation simplification and code optimization that can be performed on system equations in symbolic environments such as Maple.

A major part of the coordinate selection criteria proposed in the literature thus far are based in a large part on mechanism topology. Consequently, the pre-processing environment provided by ModelBuilder should be sufficient for the implementation of a tree selection algorithm and its advantages to the user seems to greatly outweigh the possible disadvantages it may entail.

When trying to determine the efficiency of certain models, the *complexity* of the mechanical system's symbolic equations can also be used. Though the term *efficiency* is still defined as the mechanism's simulation time, this extra criterion can help shed some light on certain mechanism properties. Using Maple, it is possible, through the use of a *cost* function, to estimate the complexity of a given set of equations by finding the sum of all the additions, multiplications, and function calls used to define the set of equations.

This was used as a measure of efficiency by Redmond and McPhee [25], but was shown to sometimes improperly predict the actual efficiency of the mechanism's simulations. However, Redmond and McPhee had estimated the equation cost prior to the code optimization of the equations. In this report, the equation complexity was estimated using Maple's cost function after the optimization process was completed. This results in the relative difference in solution time between different models and coordinate sets to be relatively accurately reflected within this estimation of equation complexity. For this reason, the system model's estimated complexity shall always be included in the efficiency analysis of different models in this report.

Chapter 3

Variable-Based Tree Selection

A major part of the literature dealing with coordinate selection states that the fewer the modelling variables, the more efficient the model's simulations will be [12, 16, 19, 31]. Other authors do not state this directly, but their analysis and/or proposed coordinate set fits this description [5, 9, 25]. The only known coordinate selection algorithm, developed by Kim and Vanderploeg [12], uses this single criterion for coordinate selection. However, no research seems to have been made on how to select between coordinates systems having the same number of variables.

Since McPhee's [19] graph-theoretical method will be used to obtain the symbolic equations of mechanical systems, and that coordinate selection in this method is the result of the spanning tree selection, the goal of the algorithms developed in this report will be expanded from automated coordinate selection to automated tree selection. As will be discussed later, tree selection has more influence on the system's equation than simply coordinate selection. These influences will also be taken into account in this tree selection method.

The tree selection algorithm in this chapter will be based on the notion of minimal coordinate selection and will also expand this theory by proposing criteria capable of distinguishing between different coordinate sets having the same number of variables.

In order to verify the extra heuristics needed to automate the selection of a mechanical system's trees, a series of benchmark problems were established. A major focus of these benchmark problems was placed on closed-loop systems, since these systems require special

care during tree selection due to their closed kinematic chains. This makes them particularly challenging from the point of view of coordinate selection and hence tree selection. Each of these benchmark problems were then solved using various trees that model the system with a similar number of modelling variables using both traditional coordinate set types (i.e. Joint and Absolute Coordinates), and hybrid types. Their solution time, as well as their equation complexity, were compared in order to gain insight on the factors affecting tree selection. The benchmark problems and their simulation results can be found in Appendix A.

First, the series of heuristics developed for this tree selection algorithm will be presented in Section 3.1, followed by a description of the algorithm itself in Section 3.2 and a detailed analysis of the application of this algorithm on two benchmark problems in Section 3.3. Finally, the shortcomings of this algorithm will be presented in Section 3.4.

3.1 Heuristics

Based on the results of the benchmark problems when modelled using various tree selections that result in a minimal set of coordinates (Appendix A), as well as information from the literature, a series of heuristics were developed for automated tree selection. These heuristics are presented in order of importance.

3.1.1 Absolute Branch and Chord Heuristics

Since we are using McPhee's [19] graph-theoretical method of obtaining a mechanical system's symbolic equations, it is important to be aware of this method's tree selection limitations. According to McPhee [19]: "The only restriction on the selection of trees in this current work is that all rigid-arm elements must be in the translational tree". This limitation is not very significant and, as will be shown in this section, is actually the basis for a useful set of heuristics.

We know that there will be a dynamic equation generated for each of the n Branch Coordinates. This gives a system of n equations with $n + m$ variables, where m is the number of chord through variables whose value through time is also unspecified. The m

kinematic equations contain the same Branch Coordinates, thus providing a system of $n + m$ equations with $n + m$ unknown variables. Note that $n - m = DOF$, where DOF refers to the degree of freedom of the system.

From this brief overview of the equation formulation procedure of Section 2.3, McPhee [19] deduces that *edges whose across variables are known functions, such as motion drivers and arm elements, should always be placed in the tree.* This will be known as *the absolute branch heuristic.* If these elements are placed in the tree, no dynamic equation will be generated by these edges. On the other hand, if these same edges were placed in the cotree, they would generate three kinematic equations since the forces associated to motion drivers and arms are unknown. To make matters worse, if these edges are placed in the cotree, other edges, probably containing a few unknown across variables, must be placed in the tree in their place. This would generate more dynamic equations. This heuristic also clearly shows that the limitation of McPhee's graph-theoretical symbolic equation formulation method for mechanical systems does not negatively affect the system's tree selection and that this limitation is automatically avoided by using the *absolute branch heuristic*

An example of this fact can be demonstrated by a simple pendulum whose graph is illustrated in Figures 3.1(a) and 3.1(b), where the trees are shown in bold. A model whose trees both consist of the tree illustrated in bold in Figure 3.1(a), with both arms selected in the tree, would be modelled using only one dynamic equation due to the revolute joint in the rotational tree. If this same pendulum was modelled with both trees taking the form of the tree in bold in Figure 3.1(b), the situation would be different. This model's trees would contain an arm in the cotree, forcing the body element to be placed in the trees. The pendulum generated in this fashion would generate seven dynamic equations (six due to the translation and rotation of the body element and one due to the rotation of the revolute joint) and six kinematic equations. In order to keep this report concise, all of these equations were not transcribed; however the fact remains clear that the first model of the pendulum will be much more compact and simple than the second model.

McPhee [19] uses the same thought process to elaborate what will be called the *absolute chord heuristic* that states that *edges whose through variables are known functions, such as force and torque drivers and spring-damper-actuators (SDAs), should always be placed in the cotree.* The SDAs, also like drivers, allow six degree of freedom movements. Hence,

(a) All Arms in the Rotational Tree

(b) Only One Arm in the Rotational Tree

Figure 3.1: Pendulum with different rotational trees.

if they are placed in a tree, they will generate six dynamic equations, three in translation and three in rotation.

3.1.2 Minimal Variable Heuristic

This heuristic, called the *Minimal Variable Heuristic*, states that *minimizing the number of modelling variables will result in more efficient simulations*, as proposed by many authors [12, 16, 19, 31]

The logic behind this heuristic can be explained by knowing, as stated earlier, that modelling mechanical systems using n variables results in the use of n dynamic equations and $n - DOF$ kinematic constraint equations. To simulate a mechanism, all of these equations must be solved simultaneously. When $n = DOF$, the equations consist of a set of ordinary differential equations (ODEs). However, when $n > DOF$ (this is inevitable with most closed-loop systems) the equations consist of a set of differential-algebraic equations (DAEs) for which numerical solution methods are more complex and time-consuming. From these observations, it can be theorized that the fewer the variables that are used to model the mechanism, the fewer the equations that must be solved simultaneously, which will very likely lower the solution time.

3.1.3 Minimal Variable Chain Heuristic

To the best of the author's knowledge, there exist no universal criteria capable of providing insight on how to predict the optimal set of coordinates from different sets comprised of the same number of modelling variables. However, Huston [9] does show that if, in such circumstances, rotations expressed relative to the global reference frame are used, efficient simulations are produced. Furthermore, Fayet and Pfister show that whenever a body's translation along an axis or rotation about an axis can be modelled, using a virtual joint, relative to a body that is more closely related to the ground than its adjacent body, and that this process does not add any extra modelling variables, this model will produce a more efficient model.

Combining these facts with observations made from various benchmark problems, it is possible to formulate a general heuristic, called the *Minimal Variable Chain Heuristic*, that states that *for a given body, the more variables present in the chain of bodies relating this body to the ground, the more complex will be the equations describing the motion of this body.*

This can be explained by considering that if there exists a long chain of modelling variables relating a given body to the ground, every time the absolute position, orientation, velocity or acceleration of this body appears in the system's equations, they will be expressed according to this long series of variables. By reducing the number of modelling variables relating every body to the ground, the system's equations will reduce in complexity.

For example, let us look at a planar five-bar mechanism, depicted in Figure 3.2, whose graph is depicted in Figure 3.3. In this graph, the edges h_{14} to h_{18} represent the five revolute joints of the system. The edges m_1 to m_4 represent the four bodies of the system and link the ground node to the center of mass of each body. Finally, the edges r_5 to r_{13} are the arm elements that define the location of each revolute joint, relative to the body's center of mass.

If we look at the rotational tree of this system, since the body edges allow three degrees of freedom in rotation, four revolute joints must be selected in order to obtain a valid tree if one wants to model the system with the fewest variables as possible. There are many possible trees allowing four revolute joints in the tree. If we model the system using the

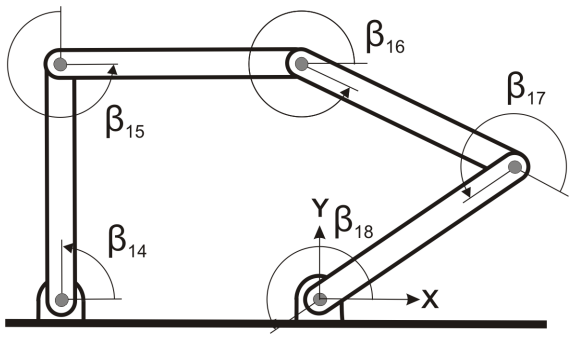


Figure 3.2: A five-bar mechanism.

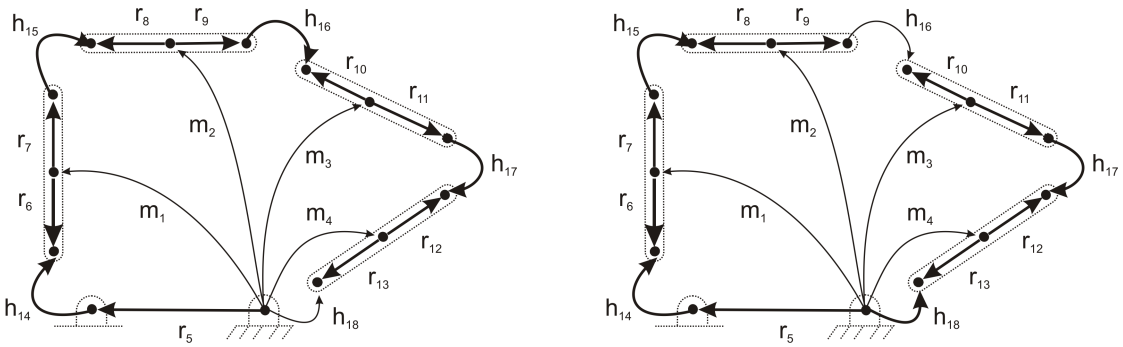


Figure 3.3: Pendulum with different rotational trees.

tree depicted in bold in Figure 3.3(a) the modelling coordinates will be β_{14} to β_{17}, which correspond to the rotation of h_{14} to h_{17}. Notice that the body m_4 is related to the ground using all four variables. Suppose we want to know the position in the global X axis of the center of mass of body m_4, represented by the symbol x_4. The resulting equation is given in equation (3.1).

$$\begin{aligned} x_4 &= (l_6 + l_7)\cos(\beta_{14}) + (l_8 + l_9)\cos(\beta_{14} + \beta_{15}) + \\ &\quad (l_{10} + l_{11})\cos(\beta_{14} + \beta_{15} + \beta_{16}) + l_{12}\cos(\beta_{14} + \beta_{15} + \beta_{16} + \beta_{17}) \end{aligned} \tag{3.1}$$

where l_i represent the length of the arm element r_i.

Now suppose we use the tree depicted in Figure 3.3(b) to model the rotation of the system. In this case, the modelling variables are β_{14}, β_{15}, β_{17}, and β_{18}, representing the rotation of the revolute joints h_{14}, h_{15}, h_{17}, and h_{18}. Notice that every body in the mechanism now has a maximum of two variables relating it to the ground. If we again try to model the position in the global X axis of the center of mass of the body m_4, we obtain equation (3.2). This equation is clearly much less complex than the one obtained using the the tree in Figure 3.3(a). Imagining that this body's position, velocity, and acceleration will be defined in a similar manner, it becomes clear that the second tree depicted in Figure 3.3(b) will produce a much more efficient model than would the tree depicted in Figure 3.3(a).

$$x_4 = L_5 + L_{13}\cos(\beta_{18}) \tag{3.2}$$

The full tree selection results for this benchmark problem can be found in Appendix A.2.

3.1.4 Minimal Joint Chain Heuristic

It is important to extend our heuristics to select proper trees if many possibilities still exist after the enforcement of the heuristics presented thus far. For example, all of the edges in the translational graph of the five-bar mechanism depicted in Figure 3.3 have no translational across variables other than the three provided by each body edge. Because of this, the Minimal Variable and Minimal Variable Chain Heuristics will only allow any

tree consisting of only arm edges and revolute joints to be selected. Since there are many such possibilities, we must establish which is preferable.

Clearly, in the case of this planar five-bar mechanism, the selection of any translational tree consisting of only arm edges and revolute joints will not affect the choice of modelling coordinates, since none are added to the system in any of those cases. However, the translational tree will be used for more than just coordinate selection as it plays an important role in the symbolic equation formulation procedure described in Section 2.3. For example, Step III of the equation formulation procedure uses the translational tree in its first two parts and uses the rotational tree in the last part. Step III is applied to all the system's equations, no matter if they are dynamic or kinematic equations generated by the translational or rotational tree. This means that both trees always play a role in generating the model and have a certain influence whether they contain modelling variables or not.

Through observations of the benchmark problems presented in Appendix A, it can be seen that even when no modelling variables are present in a tree, it is still *better for the chain of edges relating each body to the ground to contain the fewest number of joints as possible*. This extension to the Minimal Variable Chain Heuristic will be called the *Minimal Joint Chain Heuristic*.

In the case of the five-bar mechanism depicted in Figure 3.2, the Minimal Joint Chain Heuristic will ensure that the mechanism's translational tree takes the form of the tree depicted in the Figure 3.3(b). This tree will ensure that each body is connected to the ground by a maximum of two revolute joints in the tree.

3.1.5 Tree Similarity Heuristic

Even with the heuristics developed thus far, there still exist cases where many trees are possible. A good example of this can be seen in the four-bar mechanism in Figure 3.4 whose graph is depicted in Figure 3.5. The four revolute joints are represented by the edges h_{11} to h_{14} and the bodies are represented by the edges m_1 to m_3. These three edges relate the ground nodes to each body's center of mass. Finally, the edges r_4 to r_{10} represent the location where each joint is connected, relative to each body's center of mass. For this mechanism, there are two rotational and two translational trees capable of satisfying all the heuristics presented thus far. These two trees are the same in both rotation and

translation and are presented in bold in Figures 3.5(a) and (b) and are denoted by the names *Tree A* and *Tree B*.

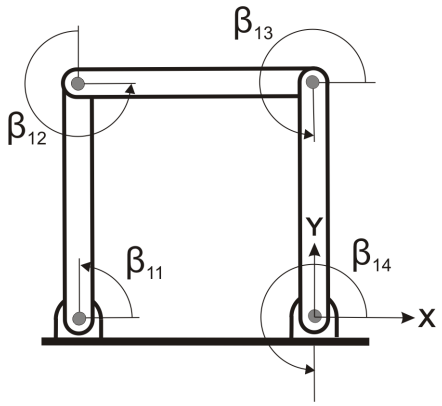
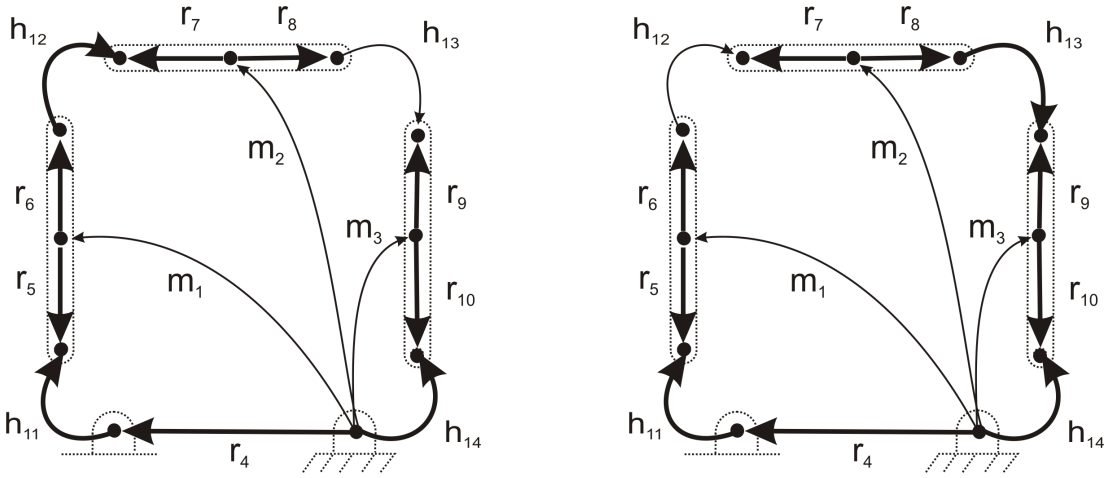


Figure 3.4: A four-bar mechanism.



(a) *Tree A* (b) *Tree B*

Figure 3.5: Two trees of the four-bar mechanism.

Clearly, these two trees satisfy the Minimal Variable and Minimal Variable Chain Heuristics in rotation since they both contain three modelling variables (one for each of the three revolute joints in the tree) and the highest number of variables relating any body

to the ground equals two, which is clearly the minimum in this case, considering that a valid tree must be selected.

The same can be said for the translational graph, in which case both trees would generate no translational across variables, hence easily satisfying the Minimal Variable and Minimal Variable Chain Heuristics. Furthermore both relate each body to the ground node via a maximum of two joints, thus each satisfying the Minimal Joint heuristic as much as possible.

When trying the different combinations of trees for this system, whose results can be found in Appendix A.1, as well as those for other systems allowing multiple trees even after applying all the heuristics mentioned thus far, it was determined that *the more similar the translational and rotational trees are to each other, the more efficient were the models' simulations*. This shall be called the *Tree Similarity Heuristic*.

At this point in time, it is unclear why the Tree Similarity Heuristic affects the final equations. However, a more detailed look at this phenomenon will be pursued in Section 4.1.6 when establishing heuristics for the second tree selection algorithm. This was only done at this later stage since the present tree selection algorithm was mostly derived from observations of benchmark problems, while the other tree selection algorithm of Chapter 4 will take a detailed look into the equation formulation procedure presented in Section 2.3 to find tree selection heuristics. This detailed analysis will provide better methods to determine the exact cause of this observation.

3.2 Algorithm

As discussed in Section 2.2, when modelling mechanical systems using graph theory, the selection of modelling coordinates is accomplished by selecting trees from the graph model of the mechanism. A typical method used to select a tree is to use a weighted graph and to find the minimal spanning tree. The following section will discuss how each of the heuristics presented in Section 3.1 are included in the edge weighting process and how a minimal spanning tree algorithm will be applied to find spanning trees. This tree selection algorithm will be called the *Variable-Based Tree Selection Algorithm*.

3.2.1 Edge Weights

The weight assigned to each edge is expressed according to five distinct factors. Each of these factors enforces one of the heuristics presented in Section 3.1. These factors will have distinct values for rotation and translation.

Absolute Factor: This factor enforces the Absolute Branch and Chord Heuristics. In order to enforce the Absolute Branch Heuristic, the Absolute Factor for motion drivers and arm elements will be equal to zero. This will ensure that these edges have the lowest possible weight and are selected in the tree before other edges. Likewise, the Absolute Chord Heuristic is implemented by assigning an Absolute Factor of two to force/torque drivers and spring-damper-actuators (SDAs) to ensure that these have a high weight and are not selected in the tree. All other edges are assigned an Absolute Factor of one.

Edge Type Factor: This factor enforces the Minimal Variable Heuristic by assigning to this factor the number of across variables the given edge type has within the tree that is being selected. The Edge Type Factor of all the edge types are presented in Appendix C.

Distance Factor: This factor will enforce the Minimal Variable Chain Heuristic by ensuring that the edges with the fewest Branch Coordinates relating them to the ground will be placed in the tree. The first step in determining this factor is to assign the weight of each edge in each of the graphs to be equal to its Edge Type Factor. Subsequently, Dijkstra's algorithm [4], whose pseudo-code is presented in Figure D.2 of Appendix D, is used to find the shortest path weight, or SPW, associated with each node. The shortest path is the series of edges connecting a given node to the ground node such that the sum of the weights of its constituent edges is minimized. Since the edge weights (equal to the Edge Type Factors) are proportional to the number of across variables of each edge, the SPW of each node will be proportional to the minimum number of across variables relating it to the ground node.

Once the algorithm has executed, the SPW of the two nodes of each edge are compared and the lower of these two shortest path weights is assigned as the edge's

Distance Factor.

Joint Factor: This factor will enforce the Minimal Joint Chain Heuristic. In this case, each edge will be given a weight of one if it is a joint and a weight of zero if it is an arm and a very high weight if it is a body edge. Then, similar to the method used to find the Distance Factor, Dijkstra's algorithm is once again used to find the shortest path weight, or SPW, associated with each node. Then, each edge's Joint Factor is assigned the lowest of its two node's SPW. This provides the minimal number of joints that can relate each edge to the ground.

Similarity Factor: This factor enforces the Tree Similarity Heuristic by ensuring that the translational and rotational trees are as similar as possible. In this case, when the first tree is selected, the Similarity Factor is set to one for all edges. When the second tree is weighted, the Similarity Factor of the edges that were included in the first tree is set to zero, and is set to one for the rest of the edges.

3.2.2 Tree Selection

Once all the weighting factors of each of the edges are found, the final edge weight of each of these edges is calculated. Since some heuristics, and hence weighting factors, are more important than others, some of the factors need to be multiplied by priority factors in order to have a bigger effect on the final edge weight. The equation needed to find the edge's final weight is presented in equation (3.3). This shows that the Distance Factor as well as the Joint Factor are provided two digits in the final weight because their value can get relatively large for edges in a complex system. These two digits will allow these factors to range from 0 to 99 before catching up to the next-level factor.

$$\begin{aligned} \text{Final Edge Weight} \;=\; & 1000000 \times (\text{Absolute Factor}) + \\ & 100000 \times (\text{Edge Type Factor}) + \\ & 1000 \times (\text{Distance Factor}) + \\ & 10 \times (\text{Joint Factor}) + \\ & (\text{Similarity Factor}) \end{aligned} \tag{3.3}$$

Once the final weight of every edge of the given tree is found, Prim's algorithm [24], briefly described in Section 2.2, is used to find the minimal spanning tree. Any other minimal spanning tree algorithm could have been used in this case, and Prim's algorithm was only chosen because of its simplicity and its wide-spread use. A pseudo-code of this algorithm is presented in Figure D.1 of Appendix D.

This tree selection process is done for both the rotational and translational trees, where each weighting factor, and hence final weight, are different for rotation and translation. In general, a mechanical system's rotations add more complexity to this mechanism's symbolic equations than its translations, due to the added trigonometry associated to these rotations. Because of this, the rotational tree is chosen as the first tree to be selected by the Variable-Based Tree Selection Algorithm.

3.3 Examples

Two examples, a spatial serial manipulator and a planar 3-RRR parallel manipulator, are presented in this section to demonstrate the validity of the proposed Variable-Based Tree Selection Algorithm and its ability to produce efficient mechanism simulations.

3.3.1 Spatial Serial Manipulator

This example will show that the Variable-Based Tree Selection Algorithm, discussed in Section 3.2, is capable of selecting very efficient modelling variables when many coordinate sets with the same number of modelling variables exist for a given system. It will also demonstrate the algorithm's ability to select Indirect Coordinates, provided that appropriate virtual joints are added to the system model.

Fayet and Pfister [5] used the spatial serial manipulator shown in Figure 3.6(a) as an example of a system that is best modelled using Indirect Coordinates. From this figure, it can be observed that the axes of rotation $\widehat{u}_{13}$ and $\widehat{u}_{14}$ of the revolute joints h_{13} and h_{14} are parallel. According to Fayet and Pfister's rules for selecting Indirect Coordinates, described in Section 1.2, measuring the rotation of body m_3 relative to body m_1 and measuring the rotation of body m_4 relative to the ground is the method of modelling the given system

that will produce the most efficient simulations.

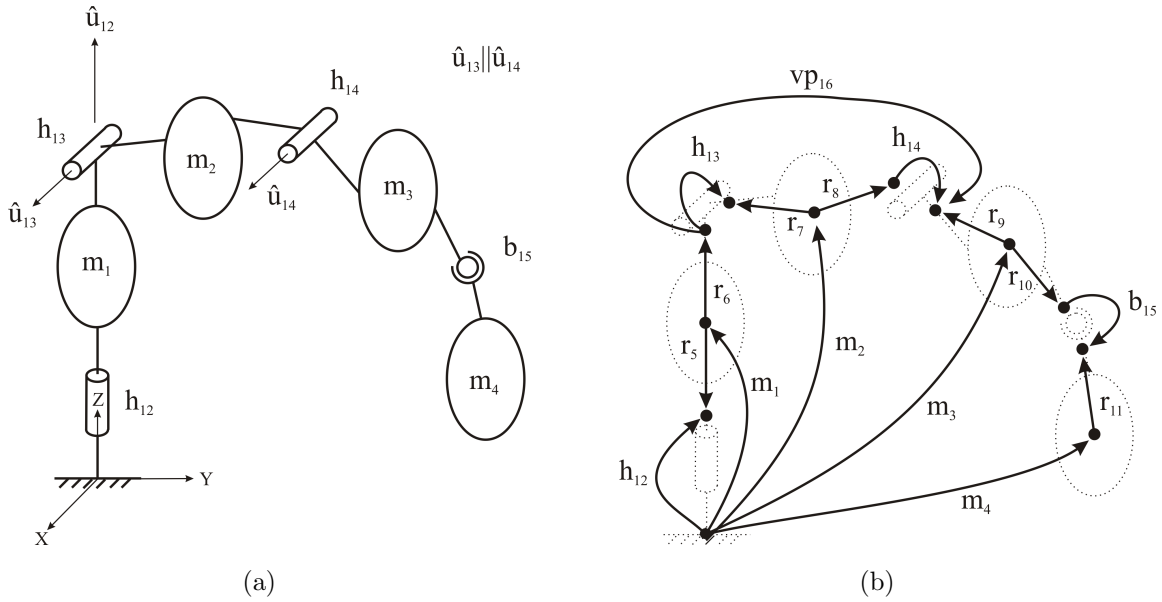
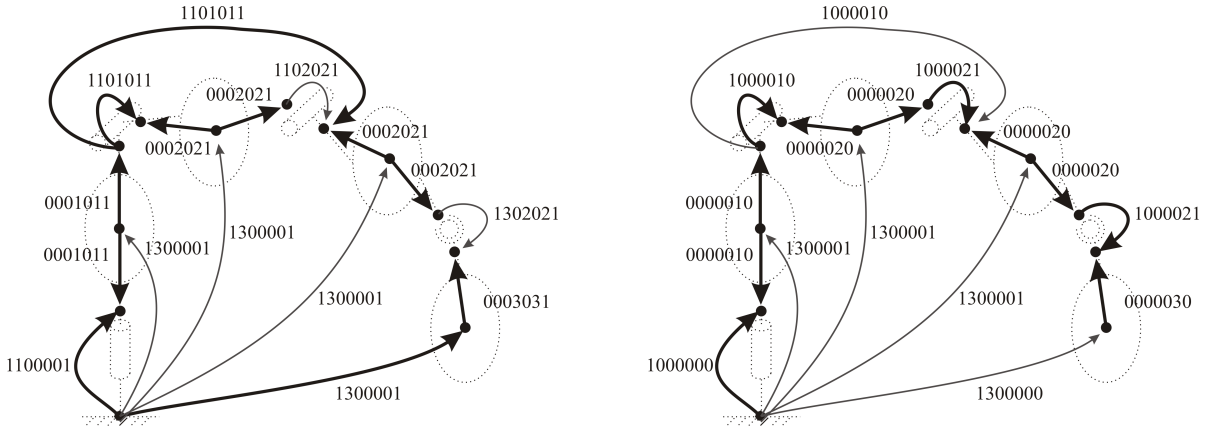


Figure 3.6: The spatial serial manipulator and its graph.

McPhee and Redmond [25] later modelled this mechanism using a graph-theoretic approach and confirmed Fayet and Pfister's findings while proving that Indirect Coordinates could be exploited in a graph-theoretic formulation. The graph used by McPhee and Redmond, shown in Figure 3.6(b), was recreated and submitted to the Variable-Based Tree Selection Algorithm in order to see if the algorithm would automatically select the Indirect Coordinates. The relative motion of non-adjacent bodies, as proposed by Fayet and Pfister, was modelled by a virtual planar joint vp_{16} between body m_1 and body m_3. No joints were added to simulate the movement of body m_4 relative to the ground since the rigid body element (m_4) of this edge already provides the appropriate Branch Coordinates. The resulting weighted graphs for rotation and translation, as well as the tree selection in bold, are shown in Figure 3.7(a) and 3.7(b). These trees were selected within an average of 15 milliseconds on a Pentium 4 of 1.80 GHz with 768 MB of RAM.

Using these two trees, the model is generated using the Indirect Coordinates: β_{12}, β_{13}, β_{16}, ζ_4, η_4, and ξ_4. The coordinates β_{12}, β_{13}, and β_{16} refer to the rotational variables of the



(a) Rotational Tree (b) Translational Tree

Figure 3.7: The spatial serial manipulator's trees.

revolute joints h_{12}, h_{13} and the planar joint vp_{16}, respectively. The variables ζ_4, η_4, and ξ_4 refer to the rotation of body m_4 relative to the ground.

A coordinate set typically used to model this system would be the Joint Coordinates β_{12}, β_{13}, β_{14}, ζ_{15}, η_{15} and ξ_{15}, which are the across variables for revolute joints h_{12} to h_{14} and the spherical joint b_{15}. This coordinate set has the same number of modelling variables as those selected by the Variable-Based Tree Selection Algorithm.

Fayet and Pfister [5] and McPhee and Redmond [25] have shown that the coordinate set found by the Variable-Based Tree Selection Algorithm will generate the model with the most efficient simulations when compared to the previously mentioned Joint Coordinates. In order to reconfirm this, the same example was also used as a benchmark problem and is presented in Appendix A.10, where McPhee and Redmond's model dimensions and inertia properties were used as the modelling parameters. This appendix models this mechanism using many coordinate sets and shows that the Joint Coordinates described above result in an average solution time that is 345% higher than the coordinate set found using the Variable-Based Tree Selection Algorithm when virtual joints are added to the model's graph.

The coordinates selected by the Variable-Based Tree Selection Algorithm show that

this algorithm takes full advantage of the benefits of Indirect Coordinates. It also shows that the concept of Absolute Angular Coordinates [9] is also fully utilized since the rotation of the body m_4 was described relative to the ground as opposed to relative to its adjacent body due to the spherical joint relating these two bodies.

3.3.2 Planar 3-RRR Parallel Manipulator

Joint Coordinates

The 3-RRR parallel manipulator is depicted in Figure 3.8, and its graph is depicted in Figure 3.9. It is comprised of three legs, each having three revolute joints. The joints connected to the ground (h_{25}, h_{28} and h_{31}) are generally referred to as the ankle joints. The revolute joints connected to the platform (h_{27}, h_{30} and h_{33}) are called the hip joints. Lastly, the revolute joints at the center of the legs (h_{26}, h_{29} and h_{32}) are called knee joints.

Figure 3.10 and Figure 3.11 show the weighted graphs for rotation and translation obtained using the Variable-Based Tree Selection Algorithm. The weight of the arm elements were not included in these figures in order for them not to become too cluttered. This omission in the figure does not affect one's understanding of the tree selection algorithm since it is already clear that the Absolute Branch Heuristic assures their selection in the trees. These figures also show, using bold edges, the minimal spanning trees that were selected to model the mechanism. These trees were selected within an average of 16 milliseconds using the Variable-Based Tree Selection Algorithm.

First, one can observe that no body edges were selected in the rotational or translational trees. This fact shows that the Minimal Variable Heuristic was respected since the body edges each have three across variables in rotation and translation.

Since all the arms and none of the body elements are in the rotational tree, some, but not all, of the revolute joints must be selected in the tree. In the rotational tree, since all the revolute joints have one variables in rotation, the Minimal Variable Chain Heuristic placed the ankle joints in the tree first, followed by the knee joints. This is because the ankle joints each have zero variables relating them to the ground while the knee joints have one variable (from the ankle joint) relating them to the ground. This was followed by placing one hip joint in the tree. The hip joints each have two variables relating them

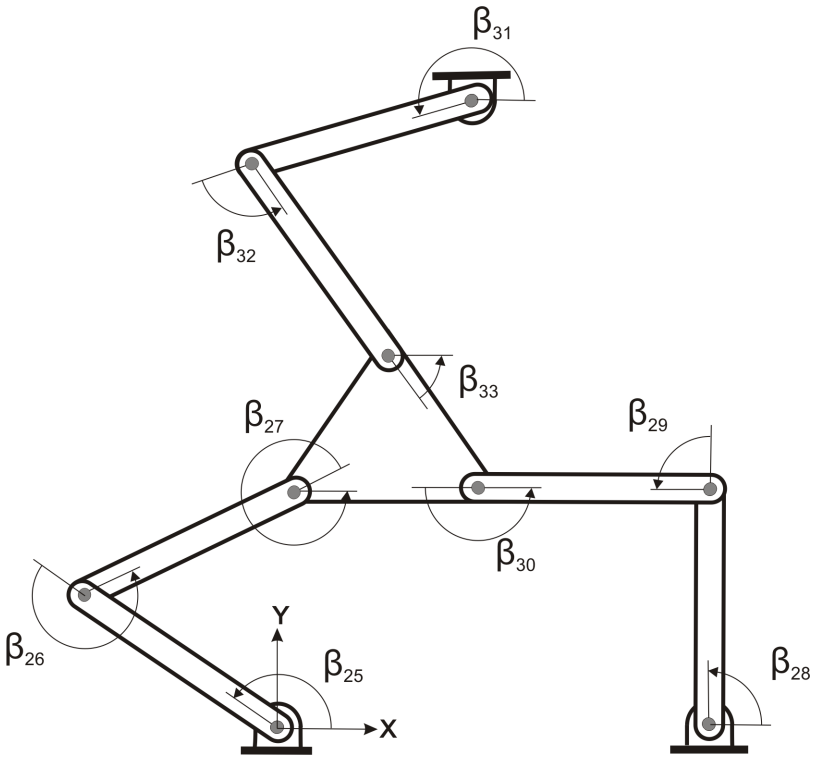


Figure 3.8: The planar 3-RRR parallel manipulator.

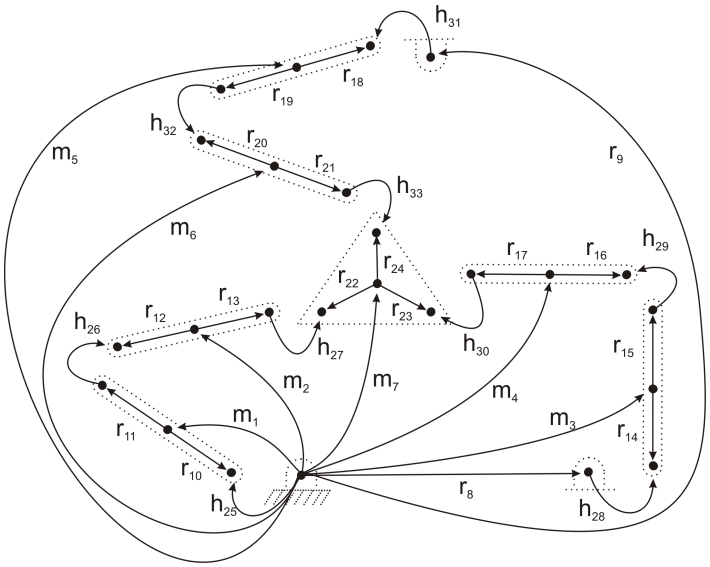


Figure 3.9: The planar 3-RRR parallel manipulator's graph.

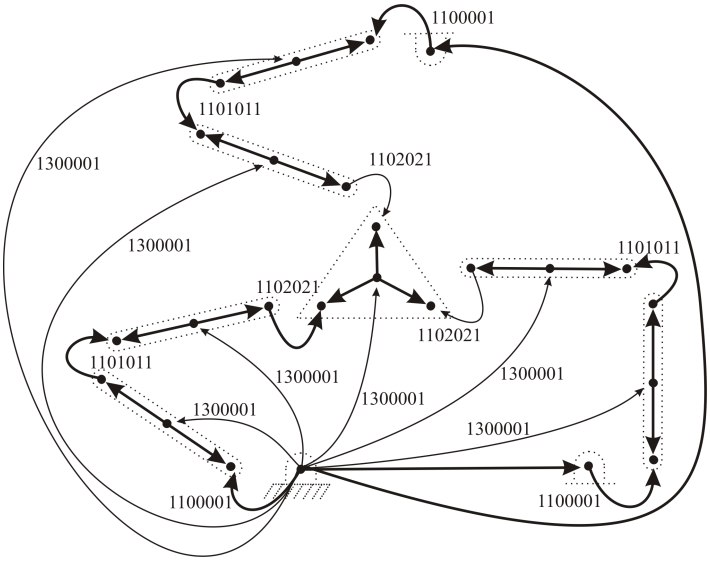


Figure 3.10: The planar 3-RRR parallel manipulator's rotational tree.

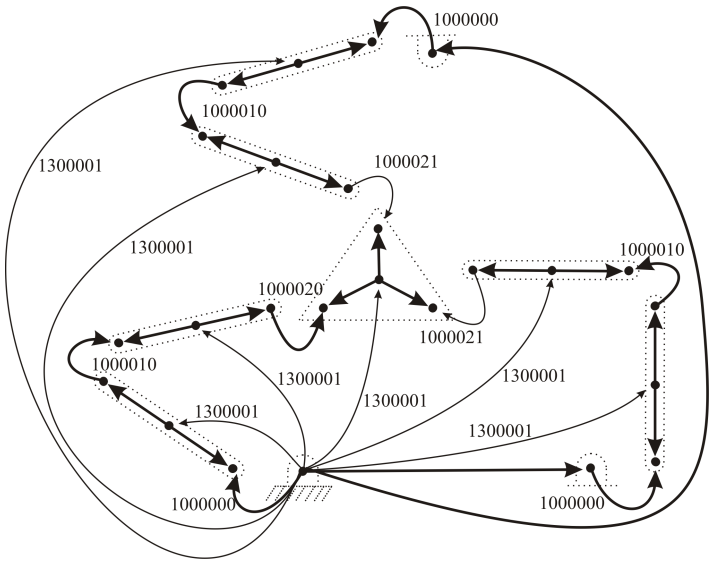


Figure 3.11: The planar 3-RRR parallel manipulator's translational tree.

to the ground. Only one such joint was placed in the rotational tree because a valid tree only required one of these joints to be placed in the tree. For the translational tree, the Minimal Joint Chain Heuristic had the same effect on the tree selection. Finally, the Tree Similarity Heuristic forced the same hip joint to be placed in the translational tree as the hip joint placed in the rotational tree.

This results in the mechanism being modelled using the coordinates β_{25} to β_{29}, β_{31}, and β_{32} representing the rotation of the revolute joints h_{25} to h_{29}, h_{31}, and h_{32} respectively. This coordinate system shall be called the *Algorithm Selected Coordinates*. It models the three DOF system using seven dynamic equations and four kinematic equations, as explained in the equation formulation procedure described in Section 2.3.

In order to verify the efficiency of the trees selected by the Variable-Based Tree Selection Algorithm, the mechanism was modelled with other trees that result in the same number of modelling variables. Because of this, all of these models will be modeled with the same number of dynamic and kinematic equations. Obviously, many tree containing all of the arm elements and seven different revolute joints will be a valid tree with the minimal possible number of modelling variables. First, we will suppose that the translational tree

still uses the tree in Figure 3.11 and the rotational tree contains all the arm elements, the three ankle joints (h_{25}-h_{28}, h_{31}), all three hip joints (h_{27}-h_{30}, h_{33}), and one knee joint (h_{26}). This new rotational tree shall be called the *Alternative Tree*. This model shall be called the *Minimal Variable Chain Violation tree set*, since it clearly violates the Minimal Variable Chain Heuristic.

A second tree set using the Alternative Tree as the translational tree and the rotational tree of Figure 3.10 shall also be used for comparison and will be called the *Minimal Joint Chain Violation tree set* since it violates the Minimal Joint Chain Heuristic.

A final model is created and given the name *Tree Similarity Violation tree set* since it violates the Tree Similarity Heuristic. This model's translation is modeled using the translational tree of Figure 3.11. In contrast, this model's rotational tree contains all of the branches of the rotational tree of Figure 3.10, except for the hip joint h_{27}, which is replaced by the hip joint h_{30}.

The geometry, inertial properties, and initial conditions of the manipulator are presented in Appendix A.7. The mechanism's symbolic model was developed for every different tree set. Then, the kinematic equations were twice differentiated and simplification and code optimization routines were performed on the resulting system equations in the Maple. The resulting sets of ODEs were solved using a Runge-Kutta Fehlberg method with an absolute error of 1×10^{-6} and a relative error of 1×10^{-5}. The mechanism was then simulated while starting at rest and falling for three seconds under the force of gravity, which is directed in the global minus Y direction. The results, which are taken from an average of ten simulations, are shown in Table 3.1.

The results clearly show that the trees that were selected by the Variable-Based Tree Selection Algorithm were the most efficient of all tree sets presented and the only model capable of real-time simulation in this case. The results also show the effect of some of the various tree selection heuristics presented in Section 3.1. For example, one can see a significant reduction in solution time, even for the seemingly-innocuous Tree Similarity Violation coordinate set. Furthermore, these results show a good correlation between the equation complexity and solution time. Finally, further simulations performed on this benchmark problem can be found in Appendix A.7.

One of the main reasons of using symbolic equations to model mechanical systems is

Table 3.1: Efficiency of Various coordinate sets

coordinate set	Sol. Time (s)	Complexity
Algorithm Selected Coordinates	1.74	2718
Minimal Variable Chain Violation	5.88	7610
Minimal Joint Chain Violation	8.47	11948
Tree Similarity Violation	5.74	7798

to diminish the model's simulation time. The simulation times presented in Table 3.1 are relatively high due to the use of the high-level programming language found in the Maple software used to numerically solve the symbolic equations of the models. If these models were exported to a more efficient programming language such as C procedure or Matlab S-function, faster simulation times could be achieved. It is possible to perform such exports in Maple (and, as such, in DynaFlexPro) and this method was used for certain benchmark problems presented in the Appendix A. These models where exported to Matlab S-functions and were then solved in Matlab/Simulink using various ODE solvers. The results showed that a ten-fold reduction in simulation solution time could be achieved using this method. It also showed that the relative efficiency between different coordinate sets used to model the same system stayed relatively the same as the Maple solution when the Matlab S-function was solved with various ODE solvers.

However, at the time these simulations were made, the act of exporting the system model in Maple to a Matlab S-function was in its infant stages and required a lot of time and effort. In order to not extend the completion time of this report, it was deemed unnecessary to complete this process for every coordinate set of every benchmark problem, since the relative solution time between these coordinate sets, which is what we are looking at in this report, would stay the same as in Maple.

Absolute Angular Coordinates

As mentioned earlier, the Variable-Based Tree Selection Algorithm is capable of selecting the best possible trees using the best possible coordinate type, provided that the appropriate virtual joints are added to the system.

When modelling planar systems, the most efficient coordinates are often Absolute Angular Coordinates. In order to be able to model the planar 3-RRR parallel manipulator using Absolute Angular Coordinates, virtual joints allowing rotations about only the global Z axis need to be added between the ground and every body in the system. In this case, the needed virtual joints can be simulated by adding planar joints between every body's center of mass and the ground node. The new system graph for one of the legs of the 3-RRR manipulator containing these extra planar joints (in bold) is illustrated in Figure 3.12.

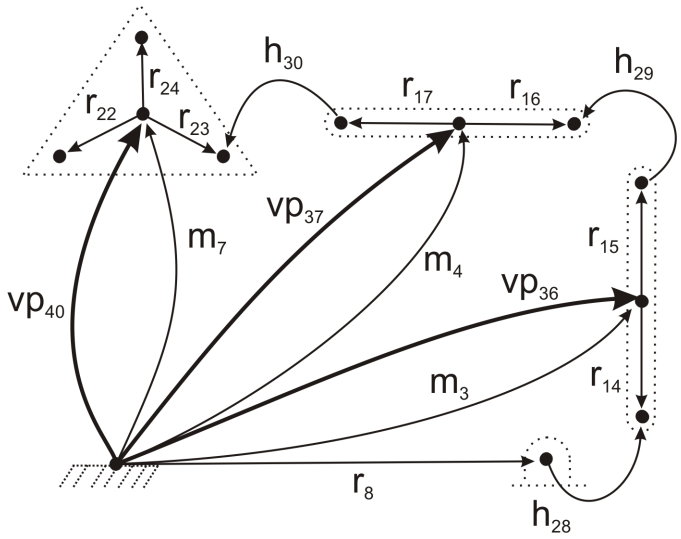
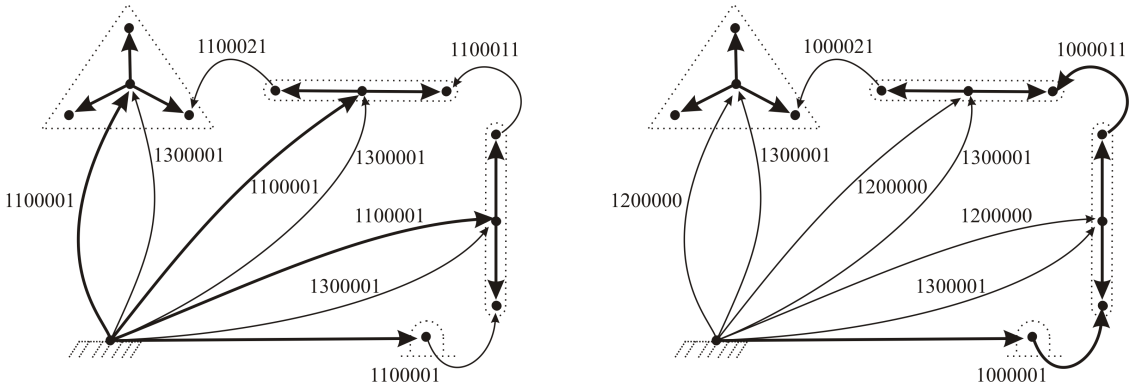


Figure 3.12: The planar 3-RRR parallel manipulator's graph with added virtual planar joints.

The Variable-Based Tree Selection Algorithm was applied to this graph and the resulting weighed graph and tree selections (in bold) are illustrated in Figure 3.13(a) and Figure 3.13(b). As with the previous weighted graphs, the arm weights are not shown in order to clarify the figures. As mentioned previously, only one of the manipulator's legs is shown

in order to clarify the illustration. The weighted graph and tree selection was the same for every leg except that in translation one of the hip joints is in the tree, giving the same translational tree as shown in Figure 3.11, where no planar joints had been added.



(a) Rotational Tree (b) Translational Tree

Figure 3.13: The trees when virtual joints are added to the model.

As predicted, the present tree selection algorithm selected the most appropriate trees containing all the virtual planar joints in the rotational graph that were necessary in order to obtain Absolute Angular Coordinates. This results in a model containing seven dynamic equations and four kinematic equations, which is the same number of equations used in the case of Joint Coordinates in the model that did not include the virtual joints. Furthermore, it is possible to note that symbolic processing detects redundant constraints, reduces them to $0 = 0$, and eliminates them.

Using these trees, the planar 3-RRR manipulator was simulated under the same conditions as the previous simulations. The results of this simulation are found in Table 3.2. It is clear from these results, that using Absolute Angular Coordinates for this system results in less complex equations and better simulation times. This example shows the versatility of the Variable-Based Tree Selection Algorithm to choose the best tree, given a well-modelled system graph.

Finally, further proof of the Variable-Based Tree Selection Algorithm's validity is found in the benchmark problems depicted in the Appendix A. A brief summery of these results

Table 3.2: Results Using the Absolute Angular Coordinates

coordinate set	Sol. Time (s)	Complexity
Absolute Angular Coordinates	1.26	1983

are presented in Table 3.3, where the column designated by *Optimal Tree Set* indicates whether or not the tree set providing the best simulation times of all the trees tested in the context of this report was found for the given mechanism using the Variable-Based Tree Selection Algorithm. The column designated by *% Longer Sim. Time* indicates, in percentage, how much longer the simulation time of the model using the tree set found by the Variable-Based Tree Selection Algorithm is compared to the simulation time of the optimal tree set.

3.4 Algorithm Shortcomings

Flaws in the Minimal Variable Heuristic

The Variable-Based Tree Selection Method presented in this chapter was developed around the premise that the coordinate sets containing the fewest number of modelling variables always produced the most efficient simulations, which is often stated in the literature. Because of this, the benchmarks were mostly tested with coordinate sets containing the minimal number of possible modelling coordinates. A particular attention was taken to establish heuristics capable of distinguishing between these minimal modelling variable coordinates sets. Within these confines, this tree selection method seemed to provide excellent results.

However, in the later stages of development of the Variable-Based Tree Selection Algorithm, certain benchmark problems were modelled with coordinate sets containing more than the minimal possible number of modelling variables. Some of these coordinate sets that contain slightly more modelling variables actually produced more efficient simulations.

Table 3.3: Results of the Variable-Based Tree Selection for Various Mechanisms

Mechanism	Appendix	Optimal Tree Set	% Longer Sim. Time
Planar 4-Bar Mechanism	A.1	yes	–
Planar 5-Bar Mechanism	A.2	yes	–
Three Bodies Attached with Two Revolute Joints	A.3	no	67
Planar Slider-Crank	A.4	yes	–
Planar Flexible Slider-Crank	A.5	no	44
Spatial Slider-Crank	A.6	no	21
Planar 3-RRR Parallel Manipulator	A.7	yes	–
Planar 3-RPR Parallel Manipulator	A.8	no	21
Peaucellier-Lipkin Straight-Line Mechanism	A.9	yes	–
Spatial Serial Manipulator	A.10	no	18
Stewart-Gough Platform	A.11	no	184
3-DOF Spatial Parallel Manipulator	A.12	no	162

Many of these situations occurred when extra translational modelling variables were added to the coordinate set. For example, in the spatial serial manipulator presented as an example in Section 3.3.2, a model producing more efficient simulations can be obtained by adding the edge m_4 in the translational tree and removing the edge b_{15} from this tree. This adds three translational variables to the coordinate set. Also, the spatial slider-crank in Appendix A.6 is simulated faster if its prismatic joint is placed in the translational tree than if it is not, even if this adds an extra modelling variable. Similarly, the Stewart-Gough platform of Appendix A.11 is modelled more efficiently when all six of its prismatic joints as well as the Cartesian Coordinates of its platform are used as modelling variables for the system. Other benchmark problems seem to possess this property as well, such as the benchmark problems presented in Appendices A.8 and A.12.

Furthermore, this phenomenon is not limited to extra translational modelling variables as shown in the 3-DOF spatial parallel manipulator presented in Appendix A.12. This benchmark problem is particularly interesting since its optimal coordinate set contains 15 modelling variables, which is much more than the minimal possible number of seven modelling variables. This shows that the optimal coordinate set does not need to contain a number of modelling variables that is close to the minimal possible number. In this case, the coordinate set containing the minimal number of modelling variables seems to be one of the most inefficient coordinate sets for the mechanism.

Looking at all the benchmark problems again after extensive testing of a much wider variety of coordinate sets, one can see that the concept of modelling a system using a minimal set of modelling variables does generally help in generating efficient simulations; however, a large number of the benchmark mechanisms treated in this report have an optimal coordinate set that contains more than the minimal number of modelling variables. This discredits the perceived universal nature of the Minimal Variable Heuristic. Since this heuristic is the main heuristic of this proposed tree selection method, this presents a major problem.

Many solutions to this problem were tested, such as using the Minimal Variable Chain Heuristic as the primary heuristic. This seems to greatly help with certain benchmark problems such as the 3-DOF spatial parallel manipulator presented in Appendix A.12, but produced poor results for other benchmark problems, primarily those containing a

high number of DOF as well as those requiring many extra translational variables in the optimal coordinate set. Furthermore, many methods were tried in order to estimate the translational variables that should be added to the system. Most of these methods became very elaborate and did not provide good results.

All of this showed that establishing heuristics based mostly on benchmark observations produced only limited success. In order to establish better and more universal heuristics, it seemed necessary to understand the underlying reasons why the Minimal Variable Heuristic is not universally applicable. Establishing heuristics based on how the equations were formulated rather than on simple benchmark observations seems to be a good option for future tree selection algorithms. This will be the premise used to establish tree selection heuristics in Chapter 4.

Priority Factors

The priority factor given to each weighting factor has a major effect on the tree selection process. Presently, the Edge Type Factor is given priority over the Distance Factor, however certain benchmark problems (e.g. the 3-DOF Spatial Parallel Manipulator presented in Section A.12) indicate that this might not always be the most efficient solution. Future research could be conducted to adapt the priority factor for different mechanism types.

Problematic Implementation of the Minimal Variable and Minimal Joint Chain Heuristics

In most cases, the method used to determine the Distance Factor and the Joint Factor produces valid results. However, certain situations can occur where these factors are not given appropriate values. This problem arises mostly when multiple edges, each having the same weight, are connected to the same node.

For example, the mechanism presented in Figure 3.14 has such a problem. This mechanism consists of three bodies that are connected by two revolute joints (h_4 and h_5), none of which connect the system to the ground. In this case, there are three body edges (m_1, m_2, and m_3) connected to the ground. These three edges are the only ones connecting the mechanism to the ground. In the rotational domain, when Dijkstra's algorithm is used to find the shortest path to the ground for each node in rotation, the shortest path for the

node labeled as N_1 is simply m_1, which has three modelling variables, giving this node a SPW of three. Similarly, the shortest path for the node N_2 is composed of the three variables from the edge m_2 and the shortest path for the node N_3 is the three variables from the edge m_3. However, it is obvious that edges m_1, m_2, and m_3 will not all be found in the optimal tree for this model, since they all have three across variables in rotation and the two revolute joints only have one such variable. This makes the SPW invalid since it considers all three body nodes to be in the tree.

For example, the Distance Factor in rotation of each revolute joint is assigned a value of three due to the SPW of three associated to all of the system's nodes, as discussed above. Let us assume that the rotational tree includes all the arm elements (r_4-r_9), both revolute joints h_{10} and h_{11} and the body edge m_1. In this case, the Distance Factor of three, assigned to the revolute joint h_{11}, is wrong since this joint is clearly related to the ground with four variables, one rotational variable from the revolute joint h_{10} and three from the body m_1.

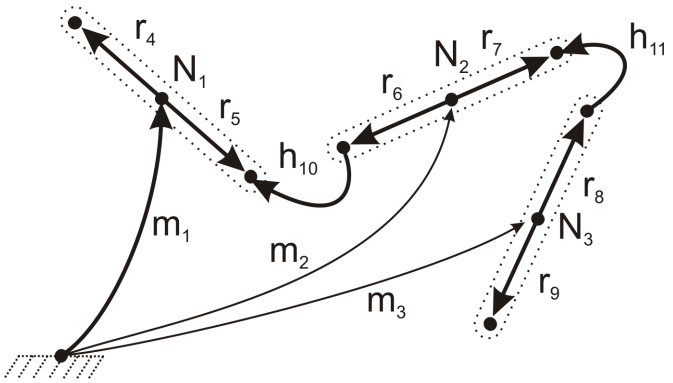


Figure 3.14: Three connected bodies not connected to the ground.

Here, the weights of all the other edges not labeled in the graph depend on the shortest path found for N_1, N_2 and N_3, which we just demonstrated are not always valid for every possible tree. The full tree selection results for this benchmark problem can be found in Appendix A.3.

This same problem can be shown for the Joint Factor that implements the Minimal Joint Chain Heuristic.

Edges with Similar Weights Having Different Effects on the Tree Selection

This algorithm shortcoming is closely linked with the previous algorithm shortcoming. In the mechanism presented in Figure 3.14, m_1, m_2, and m_3 will be given the same weight in rotation by the Variable-Based Tree Selection Algorithm. This means that Prim's algorithm will place one of the three edges at random in the tree. This is not desirable, because m_2 is obviously the best edge to put into the tree, because this choice would relate the bodies m_1 and m_3 to the ground by only four rotational variables (one rotational variable from a revolute joint and three from the edge m_2). If the body edges m_1 or m_3 are placed in the tree instead of m_2, one of the system's bodies will be related to the ground by five modelling variables (two variables provided by two revolute joints and three variables provided by the body edge in the tree).

This problem is particularly difficult to solve since it requires knowledge of the edges that will most likely be placed in the tree in the future.

Unidirectional Tree Similarity

Since the rotational tree is selected first in the Variable-Based Tree Selection Algorithm, the translational tree can be chosen to resemble the rotational tree. However, in the present algorithm, the rotational tree cannot be selected according to the translational tree since this tree is not yet selected when the rotational tree must be selected.

Flexible Body Support

Techniques have been developed to be able to model flexible bodies within a graph-theoretical approach to modeling multibody dynamics [26, 27]. The Variable-Based Tree Selection Method only supports rigid bodies and no special heuristics were developed to take into account the different properties of the flexible bodies as well as their effect on the system's symbolic equations, coordinate selection and tree selection.

Final Comments

The first algorithm shortcoming, which showed that the Minimal Variable Heuristic is not always optimal, is a major problem for the Variable-Based Tree Selection Algorithm. Since

no appropriate correction for this problem could be found using the present benchmark observations, it was deemed necessary to abandon the present tree selection method and to develop a new method from scratch. Because of this, even though some of the other algorithm shortcomings presented in this section seemed relatively simple to fix, these algorithm improvements were not pursued. Some of these shortcomings will reoccur during the formulation of the tree selection algorithm presented in Chapter 4 and will therefore be dealt with at that time.

Chapter 4

Formulation-Based Tree Selection

The Variable-Based Tree Selection presented in Chapter 3 was based on heuristics gathered mostly from criteria found in the literature and the observations of the symbolic equations as well as solution times of benchmark models. The main goal of this algorithm was to model a system using the fewest number of modelling variables in order to obtain the most efficient simulations of the system as possible. However, later simulations performed on the benchmark problems proved that this criterion sometimes does not provide models with the most efficient simulations. Simple benchmark observations are insufficient to get the thorough understanding of the equation structure required to develop the necessary heuristics.

To try and overcome this problem, a new tree selection method, called the *Formulation-Based Tree Selection*, is developed in this chapter. In this method, the heuristics will be based on the equation formulation procedure used to extract the system's equations of motion from its linear graph developed by McPhee [19]. These heuristics will be augmented by heuristics already developed for the Variable-Based Tree Selection that have worked well for the benchmark problems used in this study.

First, the formulation of the various heuristics used for the Formulation-Based Tree Selection shall be presented in Section 4.1, followed by the development of the Formulation-Based Tree Selection Algorithm in Section 4.2. Section 4.3 will present an example of the application of this algorithm. Finally, Section 4.4 will present the algorithm's shortcomings.

4.1 Heuristics

This section will re-analyse some of the heuristics developed for the Variable-Based Tree Selection as well as the equation formulation procedure presented in Section 2.3 and examine which aspects of this procedure affect the final system's equations the most. Based on these observations, a series of tree selection heuristics will be established. Finally, an overview of these heuristics as well as the general method and order in which they should be applied is presented in Section 4.1.7.

4.1.1 Absolute Branch and Chord Heuristics

The first step in establishing heuristics for the Variable-Based Tree Selection was to determine the tree selection limitations of McPhee's symbolic equation formulation procedure of mechanical systems using graph theory [19]. This limitation, which states that the arm elements must always be placed in the translational tree, must also be enforced within the Formulation-Based Tree Selection.

In Section 3.1.1, this limitation was combined with other observations made by McPhee to formulate two heuristics. The first heuristic, called the Absolute Branch Heuristic, stated that edges whose across variables are explicitly known functions, such as motion drivers and arm elements, should always be placed in the tree. The second heuristic, called the Absolute Chord Heuristic, stated that edges whose through variables are explicitly known functions, such as force and torque drivers and spring-damper-actuators (SDAs), should be placed in the cotree.

The first of these heuristics helps in avoiding the tree selection limitation of McPhee's equation formulation method. Furthermore, their conception was also based on logical observations. Out of all the benchmarks tested using many different trees, these two heuristics were never violated. Because of these facts, these two heuristics shall also be used within the Formulation-Based Tree Selection.

4.1.2 Rotational Dynamic Equations

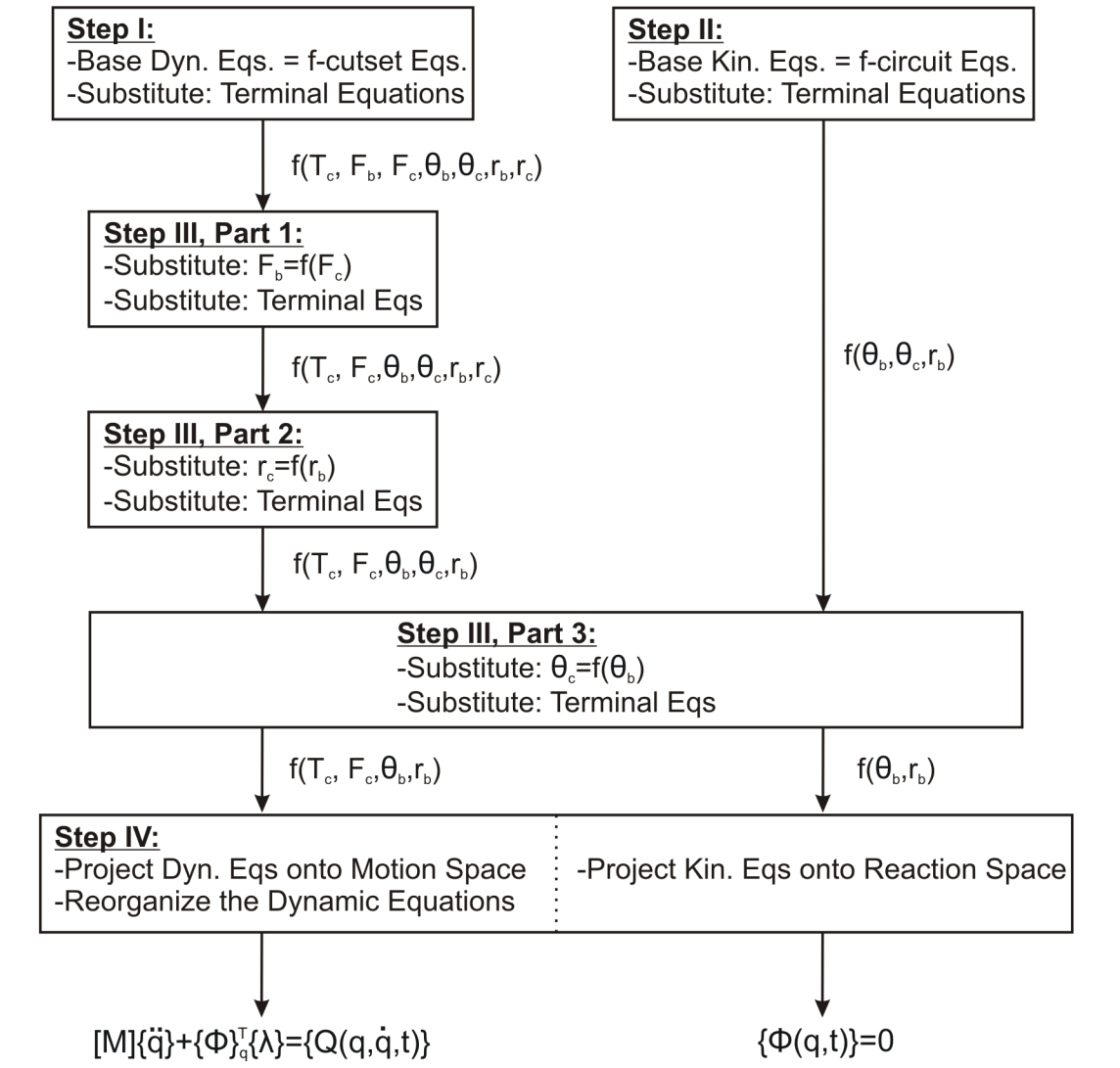
Analysis of the Dynamic Equations

Forward dynamic problems require the solution of both the dynamic and kinematic equations of the mechanical system. The dynamic equations of a system consist of differential equations that are often very complex in nature. Also, as mentioned in the previous section, there will be as many dynamic equations as across variables used to define the system (n). The kinematic equations are a series of m nonlinear algebraic equations, where $m = n - DOF$, with DOF being the degrees of freedom of the system. This implies that there are fewer kinematic equations than dynamic equations for a given mechanical model. This fact demonstrates that the simplification of a mechanical system's dynamic equations should be a priority of the tree selection algorithm.

To try and predict the complexity of the system's dynamic equations, each step of the equation formulation procedure, presented in Section 2.3 and summarized in Figure 4.1, used to generate these equations will be analysed to see its effect on the equation complexity.

The creation of a system's dynamic equations starts at Step I of the equation formulation procedure. This step generates the basic equations from which the dynamic equations will be formed. In this report, these equations will be called the *basic dynamic equations*.

Let us now go to Step IV, the final step in formulating the dynamic equations. This step adds a bit of complexity to the system's dynamic equations since it projects the equations onto their motion space (which is defined by q). This will require the evaluation of dot products. However, generally speaking the number and complexity of these dot products will be affected by the choice of the frame in which each terminal equation is expressed, which can not be predicted in the pre-processing environment. Moreover, the results of these dot products are extremely dependent on the mechanism's geometry, which can also prove difficult to treat in the pre-processing environment in which the tree selection algorithm must be developed. Furthermore, the complexity added by performing the dot products pales in comparison to the overall complexity of the dynamic equations and hence these dot products will generally have little effect on the overall complexity of the system's dynamic equations. For these reasons, the added complexity of dot products shall not be



Where:

f(a,b)=the equations are a function of *a* and *b*

Primary Variables:

T_c=Rotational Chord Through Variables
F_c=Translational Chord Through Variables
θ_b=Rotational Branch Across Variables
r_b =Translational Branch Across Variables

Secondary Variables:

T_b=Rotational Branch Through Variables
F_b=Translational Branch Through Variables
θ_c=Rotational Chord Across Variables
r_c =Translational Chord Across Variables

Figure 4.1: Equation formulation summary.

taken into consideration when establishing heuristics for dynamic equations.

In contrast, Step III of the equation formulation procedure has a major effect on the complexity of the system's dynamic equations. This step modifies the basic dynamic equations by substituting in expressions that eliminate secondary variables, and is composed of three parts. This adds complexity to these equations but does not generate any new dynamic equations. This complexity generally grows exponentially depending on how many of Step III's parts must be used in the substitution process, since variables of the first part are expanded by the second part, which in turn contain multiple variables that must each be substituted for, and are hence also expanded, in the third part. By observing which variables, found in the basic dynamic equations, are most likely to be expanded in Step III, and how many of this step's parts are required to perform this substitution, one can create tree selection heuristics that minimize the number of these variables found in the basic dynamic equations generated in step I.

Using this idea, the present section, as well as Section 4.1.3, will elaborate heuristics to help in the selection of efficient dynamic equations generated by the rotational and translational trees respectively.

Generation of the Rotational Dynamic Equations

Basic dynamic equations are generated in Step I and will be composed of various torque vectors. These torque vectors will be replaced by the torque terminal equations corresponding to the given torque vector's edge type. In step III of the equation formulation procedure, these torque terminal equations are expanded by various substitutions that result in the basic dynamic equations being expressed relative to only primary variables. These substitutions add a lot of complexity to the dynamic equations. By determining which torque terminal equation requires the most substitutions in step III, it will be possible to determine the relative complexity that each of these terminal equations can add to the system's final equations. This will help determine which torques should be avoided in the basic dynamic equations in order to produce the simplest dynamic equations.

Let us first look at the different torque terminal equations that can be found in the basic dynamic equations generated by the rotational tree. There are six edge types that can be found in these dynamic equations: joint elements, torque drivers, force drivers, rotational

SDAs, translational SDAs, and body elements. The terminal equations associated with the torque of each of these elements are presented in Table 4.1.

In this table, the vector $\vec{T}(t)$ represents the user-defined torque found in the torque driver and the rotational SDA. The joint torque $\vec{T}$ is expressed relative to $\widehat{n1}\ldots\widehat{n_n}$, which represents the rotational reaction space of the joint. In the torque expression for the rotational SDA, k is the spring stiffness, θ_{ini} is the undeformed spring's angle (the equivalent of unstretched length for a linear spring), θ is the angle of rotation for the spring, d is the damping constant and $\widehat{u}$ is the rotational unit vector along which the rotation takes place. In the rigid body's torque terminal equation, the term $\vec{\vec{J}}$ represents the body's inertia dyadic, $\vec{\omega}$ represents the body's rotational velocity vector, and $\vec{\alpha}$ represents the rotational acceleration vector. The summation $\sum \vec{r}_r \times \vec{F}_r$ represents the sum of the torques generated by each arm attached to the body (r_r), where $\vec{r}_r$ and $\vec{F}_r$ represent the displacement vector and force vector associated to one of the edges attached to the body respectively. Finally, the summation $\sum \vec{r}_t \times \vec{F}_t$ represents the sum of the torques generated by each joint allowing translation that is attached to the body or one of it's arms (r_t), where $\vec{r}_t$ and $\vec{F}_t$ represent the displacement vectors and force vectors associated to one of these joints.

Note that the motion drivers and arm elements are not mentioned in this list. This is because the absolute branch and chord heuristics states that both of these edges must be placed in the tree. Furthermore, the f-cutsets of these branches will not be used to generate dynamic equations as discussed in Section 2.3.

Torque Drivers:

First, let's consider the terminal equation of the torque driver. Since by definition this torque is already fully defined and torque driver elements must be in the cotree (as indicated by the Absolute Chord Heuristic presented in Section 4.1.1), Step III will not have to eliminate any secondary variables and the torque variables will stay as they are in the basic dynamic equation. This means that it is very advantageous to have torque elements in the basic dynamic equations since they only consist of one torque vector and will not be transformed into a complex equation in Step III. (It can be noted that it is possible for the user to enter a torque value that is dependent on other variables of the system, such as body position. This could require Step III to replace these variables if they are secondary variables. However, this possibility will not be considered in this report.)

Table 4.1: Torque Terminal Equations Found in Basic Dynamic Equations

Element Type	Torque Terminal Equations
Torque Driver	$\vec{T} = \vec{T}(t)$
Force Driver	$\vec{T} = \vec{0}$
Joint	$\vec{T} = \vec{T}(\widehat{n1} \dots \widehat{n_n})$
Rotational SDA	$\vec{T} = (k(\theta_{ini} - \theta) - d(\frac{d}{dt}\theta) + \vec{T}(t)) \cdot \widehat{u}$
Translational SDA	$\vec{T} = \vec{0}$
Rigid Body	$\vec{T} = -\vec{\vec{J}} \cdot \vec{\alpha} - \vec{\omega} \times \vec{\vec{J}} \cdot \vec{\omega} - \sum \vec{r}_r \times \vec{F}_r - \sum \vec{r}_t \times \vec{F}_t$

This concept can be illustrated by the slider-crank example presented in Section 2.1. When we look at the basic dynamic equations of the system (equations (2.5) and (2.6)) we see that the torque element $\vec{T}_{12}(t)$ is present in the first of the two equations. When looking at the final system equations, presented in equations (2.36) to (2.43), we see that the only change to this torque driver is its projection onto the motion space of the joint h_8. Its final value, taking the form $T_{12}(t)$, is only present once in equation (2.42). This shows that this term was not expanded in Step III of the equation formulation procedure and had minimal effect on the final system's equations complexity.

Force Drivers:

Let us now consider the force drivers. It will be seen later in our discussion that torques resulting from forces exerted on the various bodies of the system will be considered in the torque terminal equation of the body element. This means that the force driver element's torque terminal equation is null and the presence of this edge in the basic rotational dynamic equation (the basic dynamic equation generated by the rotational tree) will add nothing to the final version of these dynamic equations.

Joint Elements:

Joint elements can also be found in the basic dynamic equations. These fall under two categories, branch joints and chord joints. First, let us consider that the joint element is a

branch. In this case, the joint element is the basis for the f-cutset that generated the basic dynamic equation in which it is found. This f-cutset was also projected onto the branch joint's motion space to generate the basic dynamic equations. Since every joint element is by itself passive and frictionless, the reaction space spanned by the joint will be orthogonal to its motion space. In essence, this means that a joint cannot generate torque or forces along its motion space. Since the f-cutsets are projected onto the branch joint's motion space, this joint will not add any torque elements to the dynamic equations.

However, if the joint is a chord, its torque terminal equation consists of a simple torque vector and does not contain any secondary variables. Because of this, it will not require any substitution in Step III of the equation formulation procedure.

This does not entail that the joints have little or no effect on the final dynamic equations. The joints will have a major effect on the number of dynamic equations generated and will affect the complexity of the equations used in the substitutions of the equation formulation procedure's Step III. All of these factors will be discussed in further sections. For now, what is important to keep in mind is that the presence of a joint element in a given basic rotational dynamic equation will either add nothing to the basic equation, or will add a simple torque vector. This indicates that it is advantageous to have joints in the basic dynamic equations.

The effect of joint elements in the basic dynamic equations can partly be seen in the slider-crank example presented in Section 2.1. In equation (2.3) one finds the branch joint $\vec{T}_8$ and the chord joint $\vec{T}_9$. In this case both joint torques do not appear in the final dynamic equations described in equations (2.36) to (2.43). The reason $\vec{T}_9$ is not found in these equations, though it is a chord, is that it has the same axis of rotation as the joint $\vec{h}_8$ and when the dot products were executed, $\vec{T}_9$ was eliminated with $\vec{T}_8$. Had this joint had a different axis of rotation, $\vec{T}_9$ would be found in the final equations and would not have any serious impact on the complexity of the first dynamic equation in which it would be found.

Spring-Damper-Actuators:

The next element type that can be found in the basic dynamic equations is the spring-damper element. The Absolute Chord Heuristic assures that these elements will always

be placed in the cotree. Similarly to the force driver, the translational spring-damper in a rotational tree provides no torque and hence will not add anything to the rotational dynamic equations (the dynamic equations generated by the rotational tree).

In the case of a rotational spring-damper element, the situation is a bit different. Since rotational SDA elements must be in the cotree, its torque terminal equation will contain the secondary variables $\vec{\theta}$ and $d/dt\ \vec{\theta}$. In Step III of the equation formulation procedures, these variables will be substituted by functions containing only primary variables. Since no arm forces are present in this terminal equation, only the two last parts of Step III's substitution process will be needed. Because of this, rotational SDAs in the basic dynamic equation will affect the final equation's complexity more than the presence of drivers and joints. However it is important to note that SDA elements are almost always placed in parallel with a revolute joint. If this revolute joint is a branch, the substitutions made in section three of the equation formulation procedure will simply replace $\vec{\theta}$ and $d/dt\ \vec{\theta}$ with the rotational variables of the revolute joint, thus keeping the SDA's torque element simple and compact.

Rigid Body Elements:

The final element type that one can find in a basic rotational dynamic equation is the body element. Due to the exceptional complexity of flexible bodies, their effect on dynamic equations will be analysed specifically in Section 4.1.5 and only rigid bodies will be considered in this section. The rigid body's terminal equation contains many secondary variables, whether the body is in the tree or not. In order to systematically analyse this terminal equation, we will divide it into three sections and analyse them one at a time. The first section will relate to the torque generated by the body itself ($-\vec{\vec{J}}\cdot\vec{\alpha}-\vec{\omega}\times\vec{\vec{J}}\cdot\vec{\omega}$), the second section will consider the torque due to the arms ($-\sum\vec{r}_r\times\vec{F}_r$) and the final section will consider the torques due to joints that originate from the body in question and allow translation ($-\sum\vec{r}_t\times\vec{F}_t$).

The first section of the body's terminal equation will contain no secondary variables if the body is in the tree. However, even if there are no secondary variables to replace, this section can add a certain amount of complexity to the dynamic equations due to the cross products and dot products involved with the three rotations allowed by a body element. If the body is in the cotree, the terminal equation's first section contains two secondary

variables: $\vec{\omega}$ and $\vec{\alpha}$. These secondary variables are substituted using the last part of Step III in the equation formulation procedure. The functions used in the substitutions is affected by which joints are placed in the trees and how many joints connect a given body to the ground (both heuristics in the Variable-Based Tree Selection presented in Chapter 3) but will always add a considerable amount of complexity to the dynamic equations. In general chord body torques will add slightly more complexity to the rotational dynamic equations than branch body torques due to the two secondary variables $\vec{\omega}$ and $\vec{\alpha}$ in the chord body torques.

The second section of the body's torque terminal equation is extremely important to the creation of dynamic equations and affects the model's complexity drastically. This section is dependent on how many arms are attached to a given body. In general, two arms are connected to every body (in order to connect two joints at given points on the body for example). However, this number can be lower or substantially higher. Whether or not the body is in the tree, for each arm element attached to the body there is one secondary variable, the arm force ($\vec{F}_r$), in the body's torque terminal equation. These are always secondary variables since they are associated with the arm elements that must always be found in the tree. To substitute the arm forces by functions of primary variables, all three parts of Step III in the equation formulation procedure must be used. As mentioned earlier, the use of all three parts will result in the use of a very large and complex function of primary variables to be used in the substitution. Once again, the actual size and complexity added when performing Step III of the equation formulation procedure will depend on how many joints and which types of joints relate the body to the ground. However, no matter how simple or complex this series of joints may be, the functions used to substitute the arm forces found in the basic dynamic equations will be extremely complex and will add much more complexity to the final dynamic equation than any other variable found in the basic dynamic equation mentioned thus far.

To illustrate the complexity brought forth by the arm forces, it is possible to once again take a look at the slider-crank example presented in Section 2.1. Looking at this model's first basic dynamic equation (equation (2.3)), one can see that it contains a body torque T_1, two joint torques T_8 and T_9, and the torque driver T_{12}. In the system's final dynamic equations (equations (2.36) to (2.43)), the two joint torques were eliminated by

dot products, the torque driver's torque remains unchanged, and the first section of the body's torque terminal equation was simplified to $Izz_1\ddot{\beta}_1$, which is seen by the presence of Izz_1 in equation (2.36). The rest of this dynamic equation was completely generated by the two arm forces, $\vec{F}_4$ and $\vec{F}_5$, in the body m_1's torque terminal equation. It is also interesting to note that the body m_1 was related to the ground by a single revolute joint, which is one of the simplest possible configurations. Yet, even in this very simple form, the arm forces add major complexity to the model and are virtually the only source of complexity in the slider-crank model.

Finally, the third section of the body's terminal equation, representing the torque produced by joints allowing translation (e.g. prismatic and cylindrical joints) that originate from the given body, can add significant complexity to the model. However, this section is rarely needed since not all models have joints allowing translation that do not originate from the ground. However, if a model contains such joints and these are found in the tree, $\vec{F}_t$ become a secondary variable and the added complexity is the same as with an arm element. If the joint is a chord, the variables $\vec{r}_t$ becomes a secondary variable. The substitution of this secondary variable is relatively simple and will only require the last two parts of Step III in the equation formulation procedure.

The Body Torque Heuristic

The analysis presented above clearly shows that the body elements are the components that add the most complexity to rotational dynamic equations. Hence, all of this analysis comes down to one general and simple heuristic, called the *Body Torque Heuristic*, that can be stated: *the total number of body torque instances found in all the basic dynamic equations obtained from this tree must be minimal. Furthermore, branch body torques are slightly less important in this minimisation than chord body torques.*

It might seem surprising that the heuristic presented above does not explicitly address the issue of which joints should be placed in the tree, especially considering that the Variable-Based Tree Selection of Chapter 3 was entirely based on this premise. However, a more thorough analysis of this heuristic reveals that this concept is implied by the need to define a tree.

In order to demonstrate this, let us examine a model of a spatial slider-crank depicted

in Figure 4.2. The graph of this model can be seen in Figures 4.3 and 4.4, where two different rotational trees (*R-Tree A* and *R-Tree B*), shown in bold, are selected in each figure. The spatial slider-crank is formed by a revolute joint, depicted by the edge h_9, that connects a crank, depicted by the body m_1 and its two arms r_5 and r_6, to the ground. The location where the revolute joint connects to the ground is depicted by the edge r_4. The crank is then connected to a connecting rod, represented by the edges m_2 and its two arms r_7 and r_8, by a spherical joint corresponding to the edge b_{10}. Finally, a sliding block (m_3) is connected to the connecting rod via a universal joint (u_{11}) and connected to the ground by a prismatic joint (s_{12}).

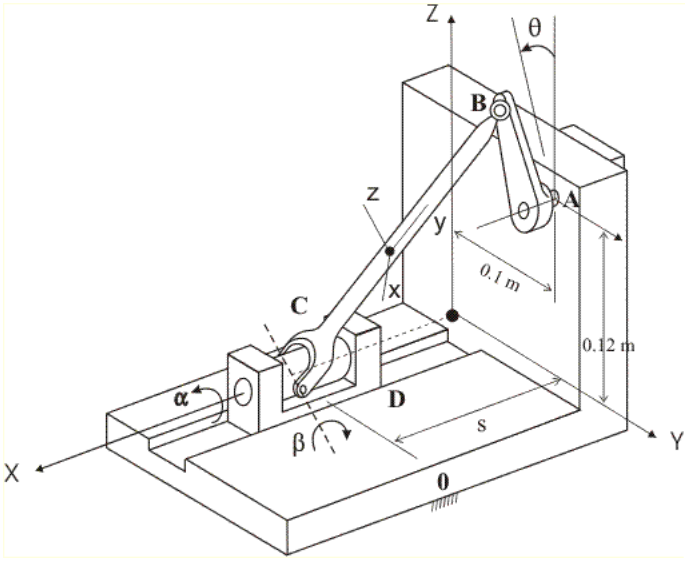


Figure 4.2: Spatial slider-crank mechanism.

The basic dynamic equations generated by *R-Tree A* are presented in equations (4.1) to (4.3). These equations were obtained with h_9 and u_{11}'s f-cutset equations depicted in Figure 4.3. The basic dynamic equations for *R-Tree B* are presented in equations (4.4) to (4.7), which were obtained by the tree's corresponding f-cutset equations for edges h_9 and b_{10}, depicted in Figure 4.4. The terms $\vec{T}_i$ in these equations refers to the torque vector of the i^{th} edge. The term $\widehat{u}_9$ represents the unit vector of the revolute joint h_9's rotational motion space. The term $\widehat{u1}_{11}$ and $\widehat{u2}_{11}$ represent the two unit vectors of the universal joint

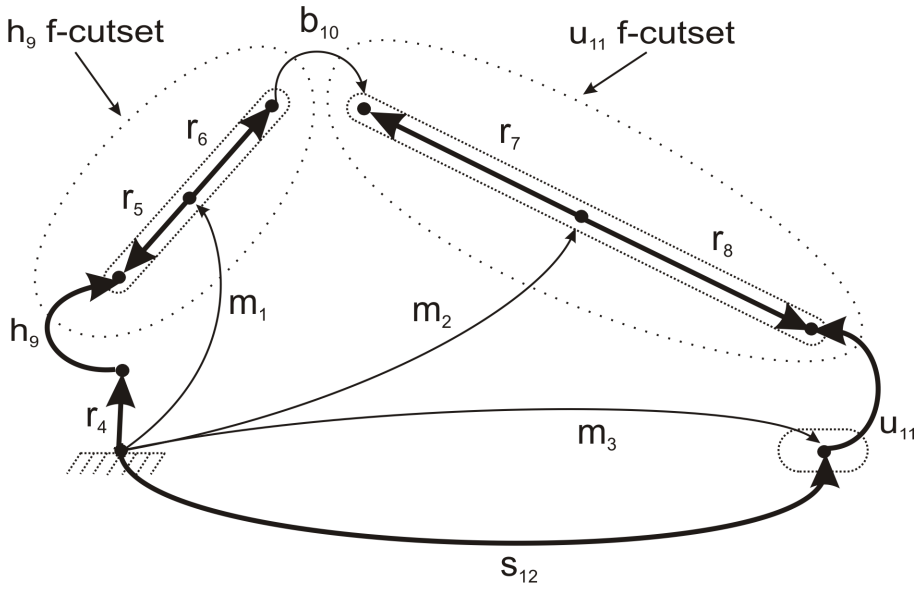


Figure 4.3: Rotational tree *R-Tree A* for the spatial slider-crank.

u_{11}'s rotational motion space, while $\widehat{u1}_{10}$, $\widehat{u2}_{10}$, and $\widehat{u3}_{10}$ represent the three unit vectors of the spherical joint b_{10}'s rotational motion space.

From these equations, one can see that the mechanism modelled using *R-Tree A* contains three instances of body torques in its basic rotational dynamic equations. $\vec{T}_1$ is present in equation (4.1) and $\vec{T}_2$ is present in both equations (4.2) and (4.3). In contrast, the model generated by *R-Tree B* contains five instances of body torques consisting of $\vec{T}_1$ and $\vec{T}_2$ in equation (4.4) and $\vec{T}_2$ appearing once in each of the three equations (4.5) to (4.7). According to the Body Torque Heuristic, the first tree is the better choice.

$$(\vec{T}_1 + \vec{T}_9 - \vec{T}_{10}) \cdot \widehat{u}_9 = 0 \tag{4.1}$$

$$(\vec{T}_2 + \vec{T}_{10} + \vec{T}_{11}) \cdot \widehat{u1}_{11} = 0 \tag{4.2}$$

$$(\vec{T}_2 + \vec{T}_{10} + \vec{T}_{11}) \cdot \widehat{u2}_{11} = 0 \tag{4.3}$$

$$(\vec{T}_1 + \vec{T}_2 + \vec{T}_9 + \vec{T}_{11}) \cdot \widehat{u}_9 = 0 \tag{4.4}$$

$$(\vec{T}_2 + \vec{T}_{10} + \vec{T}_{11}) \cdot \widehat{u1}_{10} = 0 \tag{4.5}$$

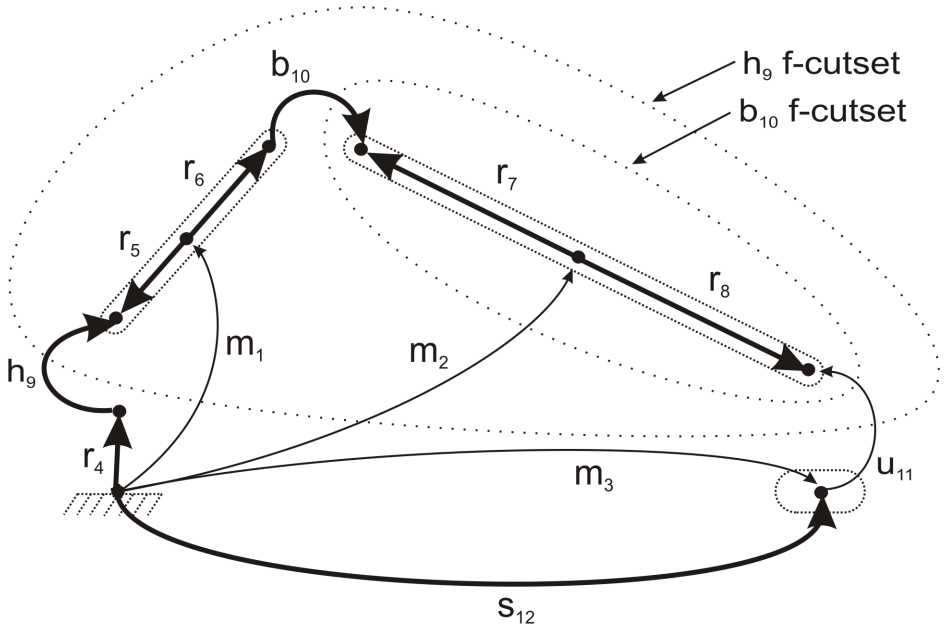


Figure 4.4: Rotational tree *R-Tree B* for the spatial slider-crank.

$$(\vec{T}_2 + \vec{T}_{10} + \vec{T}_{11}) \cdot \widehat{u2}_{10} = 0 \tag{4.6}$$

$$(\vec{T}_2 + \vec{T}_{10} + \vec{T}_{11}) \cdot \widehat{u3}_{10} = 0 \tag{4.7}$$

The optimal nature of the rotational tree *R-Tree A* is confirmed in Appendix A.6, where the spatial slider-crank is modelled using various tree combinations. In this analysis, the systems modelled using the rotational tree *R-Tree A* produce much simpler sets of equations that are solved much faster than systems modelled using *R-Tree B*.

It is now possible to look at what caused more body torques to appear in the second model. Equation (4.1) represents the basic dynamic equation generated by the revolute joint h_9 in *R-Tree A* and contains one body torque: $\vec{T}_1$. The basic dynamic equation generated by the same revolute joint using *R-Tree B*, equation (4.4), contains $\vec{T}_1$ but also contains the body torque $\vec{T}_2$. This torque is present because in *R-Tree B* the body element m_2 is connected to the ground by two joints allowing rotation. Because of this the f-cutset of the joint h_9 contains m_2. This illustrates that in certain situations, the Body Torque Heuristic insures that each body is connected to the ground with the fewest number of

across variables, which is the second heuristic of the Variable-Based Tree Selection Method and was presented in Section 3.1.3.

From this example we also can see that the f-cutset equations of an edge will include the edges of every body whose branch path to the ground node passes through the given edge. The term *branch path* in this statement refers to the series of branch edges relating a given node to another node. This observation will become important in Section 4.2.3 when the basic tree selection algorithm, used to enforce the Body Torque Heuristic, will be chosen.

In the *R-Tree A*, the universal joint u_{11} is placed in the tree and generates two basic dynamic equations (equations (4.2) and (4.3)), each containing one body torque. In *R-Tree B*, the spherical joint b_{10}, is placed in the tree. This joint generates three basic dynamic equations (equations (4.5) and (4.7)), each containing one body torque. This illustrates that when a joint generates more dynamic equations, it inadvertently adds more body torques to the system equations. In most circumstances, the use of the Body Torque Heuristic results in the system being modelled with the fewest number of rotational dynamic equations possible. The full tree selection results for this benchmark problem can be found in Appendix A.6.

The demonstration above shows that the Body Torque Heuristic enforces both the Minimal Variable and Minimal Variable Chain Heuristics used in the Variable-Based Tree Selection of Chapter 3. However, the Body Torque Heuristic does not give preference of one of these two heuristics over the other. Through the minimization of the body torques in the rotational basic dynamic equation, a low number of modelling variables is used and each body is related to the ground in a simple way; however, the importance of each of these factors is dictated by the model to be solved.

A possible criticism of the Body Torque Heuristic is that it takes a too general approach by simply minimizing all the body torques without distinguishing which of these torques are the most complex. This is a valid argument since, as mentioned in the analysis of the body torques, the complexity of the body torques is dependent on the complexity of the substitutions made in Step III of the equation formulation procedure. Looking at the three parts of Step III's substitutions, presented in Section 2.3, we can see that the first two steps are completely dependent on the translational tree and will be addressed in Section

4.1.3. In contrast, Step III is dependent on the rotational tree and is based on the use of branch transformation equations to substitute the chord across variables with branch across variables. This means that the complexity of these substitutions will be highly dependent on how many branch joints relate each body to the ground. This is already taken under consideration indirectly by the Body Torque Heuristic, as demonstrated with the spatial slider-crank example above. This means that the Body Torque Heuristic already simplifies each body torque in the basic rotational dynamic equations as much as possible and the heuristic used to select the translational tree will have to take care of the rest.

It is also important to note that the heuristics for tree selection should be kept general and fairly broad. There are many factors that affect the complexity of the final symbolic equations of a model and some of these factors, such as geometry and equation simplification and code optimization, are difficult or impossible to predict within the pre-processing environment in which the algorithm is to be developed, as well as the time limits upon which this project is based. Consequently, if the tree selection algorithm uses heuristics that take into account minute changes in equation complexity due to topology without taking into account geometry and equation code optimization, the added information will be useless and will most likely slow down the tree selection process. This is why using the the Body Torque Heuristic is deemed sufficiently precise. With more heuristics, one could determine which body torque is most complex, but it would be difficult to predict if this complex torque would or would not be drastically simplified during equation simplification or the generation of optimized code to express the system's equations.

4.1.3 Translational Dynamic Equations

Generation of the Translational Dynamic Equations

In order to generate the dynamic equations of the translational tree, one must initially apply Step I of the equation formulation procedure that generates the translational basic dynamic equations. These equations will contain various force vectors. As with the rotational tree, one can predict the final complexity of the dynamic equations created with the translational tree by examining each of the possible element types whose force vector can be found in the basic dynamic equations of the translational tree. By determining which of these

element types will require the most substitutions in Step III of the equation formulation procedure, it is possible to determine which element types should be minimized within these translational basic dynamic equations.

Table 4.2 lists the force vector terminal equations of each joint element that can be found in the basic dynamic equation of the translational tree. In this table, the vector $\vec{F}(t)$ represents the user-defined force found in the force driver and the translational SDA. The term $\vec{F}$ in the joint's terminal equation represents the joints force vector and $\widehat{n1}\ldots\widehat{n_n}$ represents the translational reaction space of the joint. In the translational SDA's terminal equation, k is the spring stiffness, L is the unstretched length of the spring, $|\vec{r}|$ and $|\vec{v}\cdot\widehat{u}|$ represents the amplitude of the rate of change of the spring's length respectively, d is the damping constant and $\widehat{u}$ is the translation vector along which the translation takes place. Finally, in the rigid body's terminal equation, m represents the body's mass and $\vec{a}$ represents the body's translational acceleration.

Table 4.2: Force Terminal Equations Found in Basic Dynamic Equations

Element Type	Force Terminal Equations
Torque Driver	$\vec{F}=\vec{0}$
Force Driver	$\vec{F}=\vec{F}(t)$
Joint	$\vec{F}=\vec{F}(\widehat{n1}\ldots\widehat{n_n})$
Rotational SDA	$\vec{F}=\vec{0}$
Translational SDA	$\vec{F}=(k(L-\lvert\vec{r}\rvert)-d\lvert\vec{v}\cdot\widehat{u}\rvert+\vec{F}(t))\cdot\widehat{u}$
Rigid Body	$\vec{F}=-m\vec{a}$

Drivers and Joints:

The analysis of each of these force terminal equations is similar to the analysis of each of these element's torque equations presented in Section 4.1.2. First, the force driver element, which is always a chord (as indicated by the Absolute Chord Heuristic presented in Section 4.1.1), will contain no secondary variables and will simply remain in the final dynamic equation as a force vector.

Similarly, the joint element, if in the cotree, will also contain no secondary variables and will simply end up as a force vector in the dynamic equations. If this joint is in the tree however, it will add no variable to the dynamic equation at all, since its force vector will be eliminated by dot products when the dynamic equation is projected onto the motion space of the vector.

Similarly, the torque driver and the rotational spring-damper-actuator have no force vector terminal equation and hence will not contribute to the complexity of the dynamic equations of the translational tree.

Spring-Damper-Actuator:

The translational spring-damper-actuator, which is also always found in the cotree (again due to the Absolute Chord Heuristic), will contain two secondary variables that consist of $\vec{r}$, $\vec{v}$ and $\widehat{u}$. The substitution of these secondary variables will add extra complication to the terminal equation. These substitutions will be executed by the two last parts of Step III of the equation formulation procedure. As with the torque SDA in rotation, one can note that the translational spring-damper is occasionally placed in parallel with a prismatic joint that stands a high chance of being placed in the tree. This would greatly simplify the substitution process, since the secondary variables can directly be substituted with the primary variables of the prismatic joint.

The rotational spring-damper-actuator, as with the torque driver, has a null vector for a force terminal equation and hence will not add any complexity to the final system equations.

Rigid Body:

Finally, the rigid body element, if placed in the tree, contains no secondary variables and its force terminal equation will simply remain unvaried in the final dynamic equation. Alternatively, if the rigid body is placed in the cotree, it will contain one secondary variable in the form of $\vec{a}$. This secondary variable will have to be substituted by using the two last parts of the equation formulation procedure's Step III. The resulting function used to replace the secondary variable $\vec{a}$ can quickly get very complex, depending on the joints selected in each tree. Also, functions used to substitute accelerations are generally much more complex than those used for position and velocity. A good demonstration of this can

be found by simply looking at the terminal equations of translational position, velocity, and acceleration of arm elements presented in Table B.2. These terminal equations are extensively used in the second part of the equation formulation procedure's Step III. This shows that even though at first glance the force terminal equations of translational SDAs may seem more complex than those of rigid body elements, the SDAs only contain secondary variables that are of a much less complex nature than the acceleration secondary variables needed for the rigid body element. This justifies the assumption that chord rigid bodies are the elements bringing the most complexity to dynamic equations generated from the translational tree.

Relation Between the Two Trees

Before formulating a final heuristic governing the selection of the translational tree, it is important to remember that on several occasions in the study of the dynamic equations created by the rotational tree, presented in Section 4.1.2, certain substitution complexities were highly dependent on properties of the translational tree. This means that there is a direct link between the rotational and translational trees that simply cannot be ignored when trying to select a system's trees.

Let us first take a look at how the translational tree's dynamic equations are related to the rotational tree. When looking at the steps one must go through in order to formulate the dynamic equations generated by the translational tree, one can notice two specific procedures that will require the rotational tree. First, during the last part of the equation formulation procedure's Step III, substitutions are made using the rotational branch transformation equations. As mentioned in Section 4.1.2, these equations are highly dependent on the joints found in the rotational tree, which are already taken into consideration within the Body Torque Heuristic. The next and final step where the translational dynamic equations are dependent on the rotational tree will be in Step IV, where the dot products are evaluated. Once again, these dot products are dependent on the joints found in the rotational tree and are also highly dependent on the geometry of the system, which will not be taken under consideration in this report. For these reasons and others mentioned previously in this report, the dot products will not be considered when establishing heuristics for generating dynamic equations. From these observations, one can gather that the effect

of the rotational tree on the translational dynamic equations is already optimized by the Body Torque Heuristic and, as such, poses no particular problem.

When looking at the effect the translational tree has on the dynamic equations generated by the rotational tree, the situation is quite different. Section 4.1.2 determined that the secondary variables $\vec{F}_r$ and $\vec{F}_t$, referring to the forces associated with arms and joints allowing translation found in the rigid body's torque terminal equation, are the variables that affect the complexity of the final rotational dynamic equations the most. The substitution process for these variables spans all three parts of the equation formulation procedure's Step III. The first of these three parts, which will affect the final dynamic equation the most, is based on using the translational chord transformation equations to express these forces as chord forces. Incidentally, Step I in generating the dynamic equations of the translational tree is based on the use of f-cutset equations of the translational tree. Since a chord transformation equation is simply a rearranged form of an f-cutset equation, we can see that the substitution for $\vec{F}_r$ or $\vec{F}_t$ will have the same order of complexity as one complete dynamic equation created by the translational tree. This also implies that the functions used to substitute $\vec{F}_r$ or $\vec{F}_t$ will be just as dependent on the translational tree's configuration as the dynamic equations generated by this same tree. Hence, when trying to find the optimal translational tree, one must not only try to simplify the dynamic equations created by this tree, but one must also minimize the functions that will be used to substitute the $\vec{F}_r$ and $\vec{F}_t$ variables present in the basic dynamic equations generated by the rotational tree.

The fact that the selection of the translational tree will also have to take into account properties obtained by the rotational tree, implies that *the rotational tree should be selected before the translational tree.* Another important observation is that, in general, the dynamic equations generated by the rotational tree contain much more $\vec{F}_r$ and $\vec{F}_t$ variables than the translational tree generates dynamic equations. This means that not only will the rotational tree's properties be necessary to the selection of the translational tree, but these properties will have more impact on the selection of a translational tree than the dynamic equations of the translational tree itself.

Body Force Heuristic

In Section 4.1.2, it was determined that to minimize the complexity of the dynamic equations generated by the rotational tree, one had to minimize the number of chord rigid body torques in the basic rotational dynamic equations. This train of thought can also be applied to the dynamic equations generated by the translational tree by minimizing the number of chord body forces in the basic translational dynamic equations. However, as discussed earlier, the translational tree also has a very big effect on the complexity of the dynamic equations generated by the rotational tree due to the forces $\vec{F}_r$ and $\vec{F}_t$ found in the rotational basic dynamic equations.

Let us examine the process required for the substitutions of the forces $\vec{F}_r$ and $\vec{F}_t$ and find a method to minimize the complexity added to the final equations by this process. As mentioned earlier, the first part of Step III's substitution process is to use the translational chord transformation equations to express these forces as chord forces. Since this process and the process used to obtain the translational dynamic equations are both based on elements in the f-cutsets, the chord forces that can be found in both these equations are the same and were presented in Table 4.2. Furthermore, the substitutions needed to eliminate the secondary variables from the force terminal equations in this table are the same for both the translational dynamic equations and the equations used to substitute $\vec{F}_r$ and $\vec{F}_t$. Because of this, the minimization of the substitutions replacing the forces $\vec{F}_r$ and $\vec{F}_t$ can also be achieved by minimizing the number of chord body forces found in their translational chord transformation equations.

Finally, one can summarise the selection of a translational tree in one simple heuristic, called the *Body Force Heuristic*. It can be stated as follows: *one must minimize the number of chord rigid body force instances needed to formulate every rotational and translational dynamic equation of the system .*

When applying this heuristic to the tree selection process, the dependence of the translational tree relative to the rotational tree becomes apparent. Because of this, the selection of the translational tree will not always minimize the number of translational modelling variables, which was one of the primary heuristics used in the Variable-Based Tree Selection presented in Chapter 3.

In order to better understand the Body Force Heuristic, let us look at the selection of a translational tree for the spatial slider-crank presented in Figure 4.2. In Section 4.1.2 the rotational tree *R-Tree A*, depicted in Figure 4.3, was shown to be an advantageous rotational tree choice for this mechanism. This analysis will assume that this rotational tree was chosen to model the spatial slider-crank.

The first step needed to enforce the Body Force Heuristic is to determine how many times each arm force is present in the rotational basic dynamic equations. These basic dynamic equations were formulated in Section 4.1.2 in the equations (4.1) to (4.3). The equations (4.8) to (4.10) present these same basic dynamic equations with the terminal equations substitutions completed and the arm forces depicted in bold.

$$(-\vec{\vec{J}}_1 \cdot \vec{\alpha}_1 - \vec{\omega}_1 \times \vec{\vec{J}}_1 \cdot \vec{\omega}_1 - l_5\widehat{\imath}_1 \times \boldsymbol{\vec{F}_5} - l_6\widehat{\imath}_1 \times \boldsymbol{\vec{F}_6} + \vec{T}_9 - \vec{T}_{10}) \cdot \widehat{k}_9 = 0 \quad (4.8)$$

$$(-\vec{\vec{J}}_2 \cdot \vec{\alpha}_2 - \vec{\omega}_2 \times \vec{\vec{J}}_2 \cdot \vec{\omega}_2 - l_7\widehat{\imath}_2 \times \boldsymbol{\vec{F}_7} - l_8\widehat{\imath}_2 \times \boldsymbol{\vec{F}_8} + \vec{T}_{10} + \vec{T}_{11}) \cdot \widehat{\imath}_{11} = 0 \quad (4.9)$$

$$(-\vec{\vec{J}}_2 \cdot \vec{\alpha}_2 - \vec{\omega}_2 \times \vec{\vec{J}}_2 \cdot \vec{\omega}_2 - l_7\widehat{\imath}_2 \times \boldsymbol{\vec{F}_7} - l_8\widehat{\imath}_2 \times \boldsymbol{\vec{F}_8} + \vec{T}_{10} + \vec{T}_{11}) \cdot \widehat{\jmath}_{11} = 0 \quad (4.10)$$

where $\vec{\vec{J}}_i$ represents the inertia dyadic of the body represented by the edge m_i and $\vec{\omega}_i$, $\vec{\alpha}_i$ represent the rotational velocity vector and rotational acceleration vector of the i^{th} edge. The terms l_i represent the length of the arm element r_i, while the term $\vec{F}_i$ represents the force vector associated to the i^{th} edge. The terms $\widehat{\imath}_i$ represents the unit vector along the the i^{th} edge's end node's local X axis. The term $\vec{T}_i$ represents the torque of the i^{th} edge. The term $\widehat{k}_9$ represents the revolute joint h_9's rotational motion space unit vector about its end node's local Z axis. Finally, the terms $\widehat{\imath}_{11}$ and $\widehat{\jmath}_{11}$ represents the universal joint u_{11}'s rotational motion space unit vectors which are about its end node's local X and Z axes respectively.

From these equations we can see that the arm forces $\vec{F}_5$ and $\vec{F}_6$ are each present once, while the arm forces $\vec{F}_7$ and $\vec{F}_8$ are each present twice in the basic dynamic equations generated by the rotational tree. This is because the body torque $\vec{T}_1$ appeared once in the equations (4.1) to (4.3), while the body torque $\vec{T}_2$ appeared twice in these same equations.

Let us consider the possible translational tree presented in Figure 4.5 called *T-Tree A*. In order to evaluate the Body Force Heuristic, one must first determine how many chord body forces are substituted into the dynamic equations generated by the rotational tree.

Table 4.3 gives each of the chord transformation equations that will be used to substitute the arm forces. This table also gives the number of chord body forces in each of these equations that are then multiplied by the number of times the given arm forces are found in the dynamic equations generated by the rotational tree. Adding all of these values together provides the total quantity of body forces in all the rotational dynamic equations, which in this case is eleven.

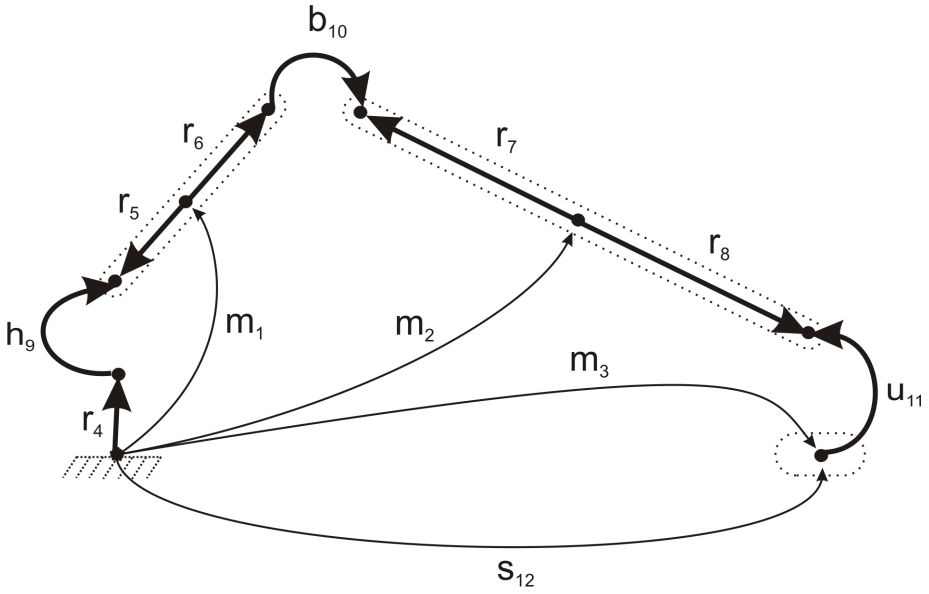


Figure 4.5: *T-Tree A* for the spatial slider-crank.

Now that we know the number of chord body forces used to formulate the rotational dynamic equations, we must also look at the number of these forces present in the translational dynamic equations. In the case of the *T-Tree A*, none of the edges in this translational tree allow a variation in translation in time. This means that this particular translational tree will not generate any translational dynamic equations. This leads to the conclusion that a total of eleven chord body forces will be used to generate all the equations of the system when using the rotational tree *R-Tree A* in Figure 4.3 and the translational tree *T-Tree A* in Figure 4.5.

Let us now consider the possibility of modelling the spatial slider-crank using the same rotational tree (*R-Tree A*), while using the translational tree called *T-Tree B* depicted in

Table 4.3: Number of Chord Body Forces Used in the Formation of the Rotational Dynamic Equations with *T-Tree A*

Arm Forces Substitution	# of Body Forces	# Times in Basic Rot. Dyn. Eq.	Total
$\vec{F}_5 = \vec{F}_1 + \vec{F}_2 + \vec{F}_3 + \vec{F}_{12}$	3	1	3
$\vec{F}_6 = -\vec{F}_2 - \vec{F}_3 - \vec{F}_{12}$	2	1	2
$\vec{F}_7 = \vec{F}_2 + \vec{F}_3 + \vec{F}_{12}$	2	2	4
$\vec{F}_8 = -\vec{F}_3 + \vec{F}_{12}$	1	2	2
Total Number of Body Forces in all the Rotational Dynamic Equations			11

Figure 4.6. In this case, Table 4.4, similarly to Table 4.6 for tree *T-Tree A*, shows the arm force substitutions for the *T-Tree B* and uses this information to determine that a total of three chord body forces will be used to generate the rotational dynamic equation when these trees are used to model the system.

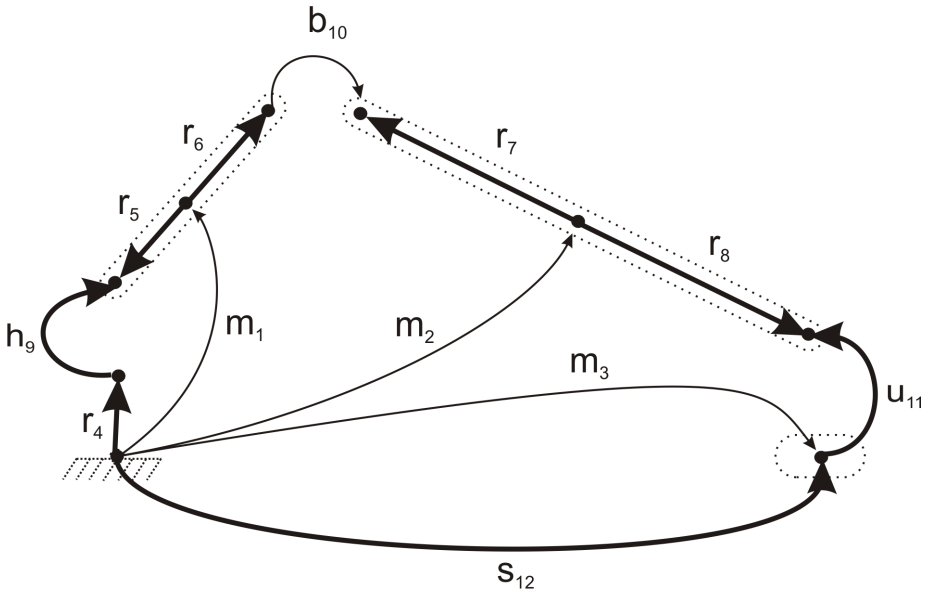


Figure 4.6: *T-Tree B* for the spatial slider-crank.

Table 4.4: Number of Chord Body Forces Used in the Formation of the Rotational Dynamic Equations with *T-Tree B*

Arm Forces Substitution	# of Body Forces	# Times in Basic Rot. Dyn. Eq.	Total
$\vec{F}_5 = \vec{F}_1 - \vec{F}_{10}$	1	1	1
$\vec{F}_6 = \vec{F}_{10}$	0	1	0
$\vec{F}_7 = -\vec{F}_{10}$	0	2	0
$\vec{F}_8 = \vec{F}_2 + \vec{F}_{10}$	1	2	2
Total Number of Body Forces in all the Rotational Dynamic Equations			3

Unlike the *T-Tree A*, the *T-Tree B* does generate a translational dynamic equation for the joint edge s_{12} in the tree. The basic dynamic equation used to generate this equation is presented by equation (4.11). This basic equation contains two chord body forces that combine with the three such forces used in the formulation of the rotational dynamic equations to give a total of five chord body forces used to generate all of the system's dynamic equations.

$$(\vec{F}_2 + \vec{F}_3 - \vec{F}_{10} + \vec{F}_{12}) \cdot \widehat{u}_{12} = 0\cdot \tag{4.11}$$

This predicts that, in this case, *T-Tree B* should produce a model whose simulations are more efficient than the ones obtained with *T-Tree A*. At first glance, these results may seem puzzling and unintuitive since *T-Tree B* will require more dynamic equations to model the system. However, the Body Force Heuristic predicts that all the dynamic equations generated by the translational tree *T-Tree B* will have a much lower complexity than those generated using *T-Tree A*. The lowered complexity of each equation will be such that even the addition of an extra dynamic equation with *T-Tree B* will still result in a lower total complexity of the dynamic equations.

The validity of these assumptions are demonstrated in Appendix A.6, where the spatial slider-crank is modelled with both of these two tree combinations, as well as a few others. The results presented in this Appendix clearly show that a lowered solution time is obtained

when using the rotational tree *R-Tree A* in Figure 4.3 and the translational tree *T-Tree B* in Figure 4.5.

This spatial slider-crank demonstration also shows the strong influence the rotational tree has on the selection of the translational tree, since the chord body forces used in the formulation of the rotational dynamic equations had a greater impact than the chord body forces that were used in the formulation of the translational dynamic equation.

4.1.4 Kinematic Equations

Kinematic equations are nonlinear algebraic equations. These equations are generally less complex than the dynamic equations that are in the form of differential equations. In this report the trees are selected to optimize the simulation times of mechanical models used for forward dynamics, a method that requires the solution of both the dynamic and kinematic equations. Because of this, the relative simplicity of the kinematic equations makes them less important in the tree selection process than the dynamic equations. Furthermore, the complexity of these equations is much more difficult to predict since their relative simplicity makes them much more dependent on the evaluation of dot products within the basic equations. These dot products are affected by both the joints found in the rotational tree and in the geometry of the model. The joints found in the rotational tree are intelligently selected by the Body Torque Heuristic presented in Section 4.1.2. However, given the pre-processing environment in which the trees must be selected, it is almost impossible to predict the effect of geometry in the model.

A mechanical system has $n - DOF$ kinematic equations, where n is the number of modelling variables and DOF is the number of degrees of freedom. Certain methods exist to predict the degrees of freedom of a system, such as the well-known Gruebler's equation. However, it is important to note that this method, as well as all other known DOF prediction methods, do not always adequately take into account geometry and often give false results for even the simplest mechanisms. Also, the DOF estimation method only helps in predicting the number of kinematic equations and does nothing to help in determining their complexity.

Since the kinematic equations are based on chords with active or passive constraints

whose f-circuit equations are projected onto their reaction space, one could simply count the number of kinematic equations generated in this fashion by the equation formulation procedure. This could help determine the total number of kinematic equations and even estimate their complexity. However, some of these equations could be redundant and would not be present in the final system model, thus modifying the DOF count of the system. This is very common, especially for planar systems. These simplifications are almost impossible to predict in the pre-processing environment in which this tree selection algorithm will be developed.

Because of all of these factors, the kinematic equations will only be treated in a simple fashion in this tree selection algorithm. Since each kinematic equation obviously adds complexity to the model, one clear heuristic, which shall be called the *Kinematic Equation Heuristic*, can be established for kinematic equations. This heuristic can be stated as follows: *the number of kinematic equations in a mechanism should be minimized.*

It is also important to predict the general complexity added to the system by these kinematic equations. This topic is very complex and dependent on many factors mentioned previously. For these reasons, the addition of extra heuristics concerning kinematic equations shall be discussed mostly in Section 4.2.4, where the tree selection method will be established, and can be considered in this decision.

Simply stating the Kinematic Equation Heuristics is insufficient however, since it is just as important to determine what priority this heuristic should be given in the tree selection process. Despite the fact that the kinematic equations are generally less complex than the dynamic equations, their impact cannot be neglected. As mentioned above, the number of kinematic equations (m) is equal to the number of dynamic equations minus the DOF of the system ($n - f$). Since the addition of dynamic and kinematic equations to the system model are so interlinked, the major heuristics enforcing dynamic equations simplification, namely the Body Torque and Body Force Heuristics (the heuristics regulating the number of dynamic equation of the system), must work in parallel with the heuristic concerning kinematic equations. The interaction between these heuristics is crucial to tree selection.

4.1.5 Flexible Bodies

Due to the problems facing the Variable-Based Tree Selection of Chapter 3, this algorithm was not expanded to take into account the effects of flexible bodies on coordinate and tree selection. This section will look into this issue and establish a heuristic to better account for the presence of flexible bodies in a mechanical system's model.

The method used to model flexible beams in graph theory are presented by Shi et al. [27]. The analysis of the effect of flexible bodies in the graph has been separated from the general analysis of the model in previous sections of Chapter 2 as well as this chapter since the terminal equations of this component are of a much more complex nature than the other elements previously mentioned. Due to the exceptional complexity of the flexible body relative to other element types, it is not necessary to analyse its terminal equations in great detail as their effect on the system equations are very pronounced, and as such, fairly easy to predict.

There are two important observations one can make on the modeling of flexible bodies. The first observation is that the flexible body element ends at the beginning of the flexible beam it models. This is represented in Figure 4.7, where the flexible body element is depicted by fm_1 and the flexible arm is depicted by the edge fr_2.

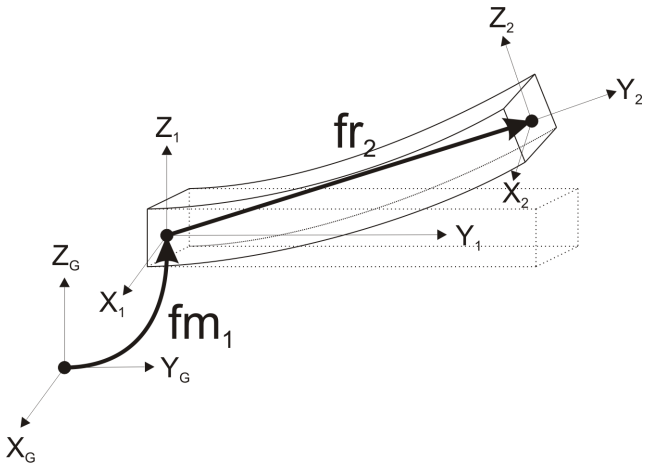


Figure 4.7: Graph edges of the flexible body.

Secondly, Flexible beams are modeled by using a set of elastic coordinates to discretize

a Rayleigh-Ritz discretization of the flexible beam. The polynomial shape function used, as well as the number of elastic coordinates is chosen by the user. This indicates that the functions describing the position, velocity, and accelerations in both rotation and translation will become a complex function of elastic coordinates. These functions will be so complex that they are usually what adds the most complexity to the system's symbolic equations.

The position, velocity, and acceleration between a flexible beam's base and end nodes are modeled by the flexible arm edges. Within the graph-theoretical approach to modeling mechanical systems used in this report, these flexible edges must be placed in the tree (if one tries to place the flexible arm elements in the cotree, DynaFlexPro will not be able to model the system). Consequently, when modeling systems containing flexible bodies, it is inevitable for the symbolic equations of these systems to contain the complex functions describing the deformations between the flexible body's base and tip reference frames. Therefore, to model systems producing efficient simulations, one must minimize the occurrences of these complex functions in the system's equations.

The addition of flexible arm across variables to the system's symbolic equations occurs at four specific points during the equation formulation procedure. The first such occurrence can be found in Step II of the equation formulation procedure where the f-circuit equations are used to form the kinematic equations. The second and third use of the arm across variables occurs in the equation formulation procedure's Step III. In the two last parts of this step, the translational and rotational branch transformation equations are used to substitute the chord across variables in the dynamic and kinematic equations. Finally, the flexible arm across variables can be added to the system when the dot products found in the kinematic and dynamic equations are executed in Step IV.

Let us start by analysing Step III of the equation formulation procedure. This step will have an very big effect on the number of times the deformations between the flexible body's base and tip reference frames will be found in the final system equations. This is because this Step of the equation formulation procedure is repeated so often in the equation formulation procedure. These steps are based on the branch transformation equations of each tree. In order to minimize the presence of flexible arm across variables in these equations, the flexible arms must be *leaf edges* of the tree, meaning that they must be at

the end of a tree section. Leaf edges can also be defined as *edges containing one node, called a leaf node, that only has the leaf edge as an adjacent branch.* The across variables of the leaf edges of the tree will only be found in the branch transformation equation of chords connected to the leaf edge's leaf node. In the case of mechanical systems, flexible bodies, SDAs and force/torque drivers are the only chords that can be attached to a flexible arm's leaf node and whose force and torque terminal equations can include across variables to be substituted in Step III of the equation formulation procedure. By selecting the flexible arm's end node as the leaf node, the flexible body's edge, which is generally a chord due to its six degrees of freedom, will not be one of the chords attached to the leaf node.

These observations can be expressed in a heuristic called the *Flexible Body Heuristic*, which states that: *The flexible arms should always be a leaf edge with its end node as the leaf node.* By following this heuristic for both tree types, no arm across variables will be used in Step III's substitution process unless SDAs or force/torque are connected to the flexible arm's end node, which one can avoid by orientating the flexible body so that these edges connect to the base reference frame of the flexible body.

Now let us look at the two other places where arm across variables can be added to the system's equations. In Step II of the equation formulation procedure, the kinematic equations are formed using the f-circuits. The minimization of the arm across variables in these equations is less critical than their presence in Step III, since the arm across variables found in the basic kinematic equation will only be found once in the final kinematic equations. In the case of Step III, the arm across variables found in the branch transformation equations could be substituted multiple times in the system's equations. This means that the Flexible Body Heuristic takes precedence to any observation that can be made for the kinematic equations. This being said, a tree selected with the Flexible Body Heuristic will add arm across variables only to the kinematic equations generated by chords attached to the arm's end node. Since the Flexible Body Heuristic forces each edge connected to the arm's end node to be chords, the only way one can influence the arm across variables added in Step II is to intelligently define the flexible body's orientation when creating the mechanism's graph. This is done by placing the flexible arm's end nodes where the fewest number of joints, force/torque and SDAs connect to the flexible beam.

The last step influencing the number of across variables in the final system equations

happens when the dot products are evaluated in Step IV of the equation formulation procedure. As mentioned numerous times in this report, predicting the complexity added by these dot products is very difficult. However, it is possible to state that the complexity these dot products will bring is influenced by the primary variables found in the rotational tree. When the Flexible Body Heuristic is applied, it assures that each body is related to the ground with the fewest number of arm across variables possible. In most cases, this will result in none of the bodies containing arm across variables to relate its position to the ground (in this case, the complex equations describing the flexibility of the flexible bodies will be enforced only by the kinematic equations). Because of this, if the ground frame is used as the basis for all the dot products, the number of arm across variables added by the evaluation of these dot products should be minimized by the Flexible Body Heuristic.

An example of the validity of the Flexible Body Heuristic can be seen in Appendix A.5, where a spatial slider-crank containing a flexible beam is modelled using various tree combinations. It can be noted that a number of other benchmark problems were used to establish the validity of the Flexible Body Heuristic. However, these supplementary benchmarks were not added to the Appendices in order to keep this report sufficiently concise.

4.1.6 Tree Similarity

From Section 3.1.1 through Section 4.1.5, heuristics were established for tree selection. These heuristics are mostly based on equations formulation analysis and topology. However, the final equation complexity of the system will also be affected by geometry, equation simplification, and equation code optimization, which are extremely difficult, if not impossible to predict using these methods. Because of this, there can still exist multiple tree sets producing different model simulation efficiency that can satisfy the heuristics developed thus far. In order to try and predict geometry, equation simplification, and equation code optimization, a different approach is taken to develop further heuristics.

At this point in time, we will once again look directly at simulation results of various benchmark problems allowing multiple tree combinations were modelled and tested for simulation efficiency. Some of these benchmarks included: a 4-Bar Mechanism, a 3-RRR

planar parallel manipulator, and a Peaucellier-Lipkin mechanism. These benchmark problems are presented in Appendix A.1, A.7, and A.9 respectively. The result of this study showed that the more similar the translational and rotational trees are to each other, the more efficient the model's simulations were. These results are reminiscent of those used to develop the Tree Similarity Heuristic of the Variable-Based Tree Selection Method in Section 3.1.5. This is why the *Tree Similarity Heuristic* developed in that section shall also be used in the Formulation-Based Tree Selection Method.

When looking at the optimal trees of each of these three mechanisms, one can see that they always obey the heuristics established in the preceding sections, thus establishing that the Tree Similarity Heuristic does not take precedence over these heuristics. It can also be interesting to note that in some cases, such as with the 3-RRR planar parallel manipulator, the added efficiency provided by the Tree Similarity Heuristic can be quite substantial.

In Section 3.1.5, the Tree Similarity Heuristic was developed only on the basis of benchmark observations. However, at this time, we have the added option of analysing the equation formulation procedure in Section 2.3 to better explain the underlying reasons for these benchmark observations. To do this, the system's symbolic equations for each of the aforementioned benchmark mechanisms for each of the possible tree combinations were formulated by hand, using the four steps used in graph theory described in Section 2.3. From these, it was possible to note that when the mechanical system's trees differed, most of the system's equations were functions of a larger variety of variables. This does not mean that the number of modelling variables present in the equations is necessarily bigger. It simply indicates that each equation will be dependent on a wider range of variables. This wider range of variables will hinder the ability of the equation simplification and equation code optimization algorithms to transform these equations into a more compact form.

Appendix E presents the symbolic equation formulation of the four-bar mechanism, first described in Section 3.1.5. From this model, the dependency of its equations on a wide range of modelling variable when modelled with differing trees as well as the increased coupling of its equations is clearly shown.

One of the Variable-Based Tree Selection shortcomings presented in Section 3.4 highlighted the difficulties of applying the Tree Similarity Heuristic. The application of this

heuristic is complicated by the fact that each tree must be selected separately. In the Variable-Based Tree Selection, this heuristic only ensured that the translational tree resembled the rotational tree and not vice versa. In order to avoid this limitation, a new Tree Similarity implementation method shall be used in the Formulation-Based Tree Selection.

As discussed is Section 4.1.3, the translational tree is highly affected by the rotational tree's properties. This makes it necessary for the rotational tree to be selected first in the Formulation-Based Tree Selection. In this case, once the rotational tree is selected; the translational tree can be selected while enforcing the Tree Similarity Heuristic. However, the Formulation-Based Tree Selection will go one step further by reselecting the rotational tree once both trees have already been selected. This second rotational tree selection will enforce the Tree Similarity Heuristic. If the same rotational tree is reselected, the tree selection is good. However, if this process results in the selection of a different rotational tree, the translational tree should be reselected as well. This tree re-selection process should continue until the trees selected are the same as the trees selected in the previous cycle. In order to avoid that too many loops are executed, thus greatly slowing down the algorithm, or to avoid the possibility off non-convergence, this loop is limited to four passes.

From experience it was found that this tree re-selection process generally finds the most similar trees on the first try. However, for some specific benchmarks, such as the 3-RRR planar parallel mechanism and the Peaucellier-Lipkin mechanism, the tree selection process must sometimes go through two complete tree selection cycles in order to converge to a solution. Particularly in the 3-RRR planar parallel mechanism, this extra step in the tree selection process can provide a substantial improvement in equation efficiency.

4.1.7 Procedure Summary

At this point all the heuristics necessary to find a set of trees have been described in the previous sections. This section will summarize these heuristics and will describe the order in which they must be applied.

A diagram of the procedure used to automate the selection of a mechanism's trees is presented in Figure 4.8. In this diagram, the sections representing each tree selection

contain a series of heuristics that are placed in order of importance. The most important heuristics are placed at the top and the heuristics that should have less influence in the tree selection are placed in the bottom. The method with which these heuristics are enforced during these tree selections will be established and demonstrated in Section 4.2.

Each tree selection process incorporates similar heuristics with a similar importance hierarchy. In each tree selection process, the Flexible Body Heuristic is of major importance. This heuristic states that the flexible arms should always be a leaf edges with its end nodes being the leaf node.

The next most important heuristics are the Absolute Branch and Chord Heuristics. These heuristics state that edges whose across variables are explicitly known functions must be placed in the tree and edges whose through variables are explicitly known functions must be placed in the cotree.

This heuristic is followed in importance by the Body Torque Heuristic for the rotational tree and the Body Force Heuristic for the translational tree. The Body Torque Heuristic states that the total number of body torque instances found in all the basic dynamic equations obtained from the rotational tree must be minimal, with a slight preference being made on minimizing chord body torques over branch body torques. The Body Force Heuristic states that one must minimize the number of chord rigid body force instances needed to formulate every rotational and translational dynamic equation of the system. Of similar importance to the Body Torque and Body Force Heuristics is the Kinematic Equation Heuristic that states that the number of kinematic equations in a mechanism should be minimized. As described in Section 4.1.4, in most circumstances, this heuristic will be of less importance than the Body Torque and Body Force Heuristics. However there exists an unavoidable relation between these heuristics for which reason they must be enforced simultaneously.

The last heuristic that must be considered during the tree section process is the Tree Similarity Heuristic that states that the translational and rotational trees should be as similar to each other as possible.

As mentioned in Section 4.1.3, the rotational tree is the first tree to be selected. Once this tree has been selected, the number of occurrences of each arm force in the basic rotational dynamic equation is calculated. The translational tree is then selected where

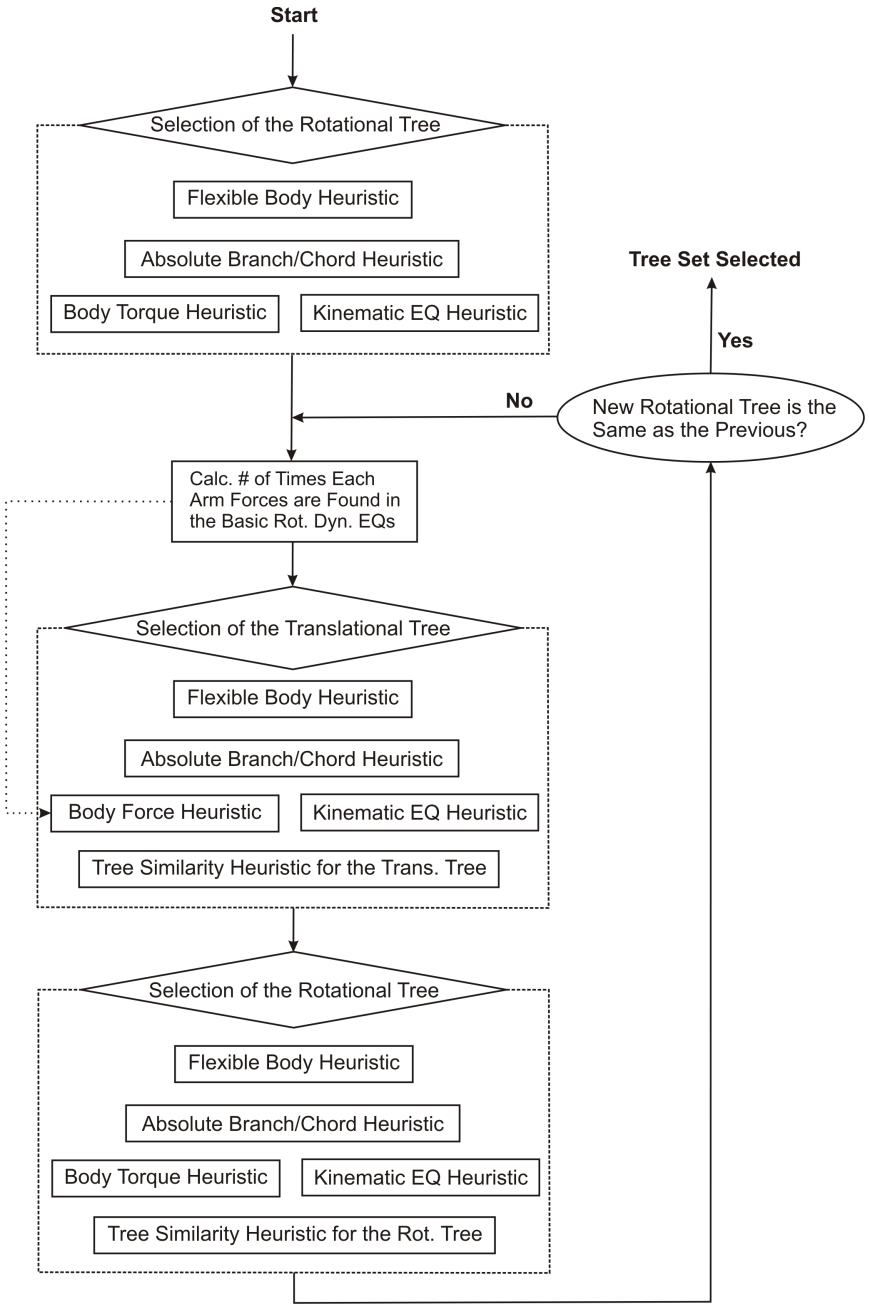


Figure 4.8: Procedure summary.

the Body Force Heuristic is enforced using the number of arm forces calculated previously. The selection of this translational tree also applies to the Tree Similarity Heuristic so that it selects the tree that resembles the rotational tree the most.

As mentioned in Section 4.1.6, it is necessary to assure that the rotational tree also takes the form that is as close to the translational tree as possible. In order to do this, the rotational tree is selected again using the same method as when it was first selected with the addition of the Tree Similarity Heuristic. If the new rotational tree is the same as the one selected previously, the present rotational and translational trees are used to model the system.

If the rotational tree selected using the Tree Similarity Heuristic is different than the rotational tree that was previously selected, the tree selection process enters in a loop where the translational and rotational trees are both selected using the Tree Similarity Heuristic until the results converge to a final solution.

4.2 Algorithm

This section will present the algorithm used to automate the selection trees for mechanical systems based on the series of heuristics presented in Section 4.1 and summarized in Section 4.1.7. This section will start by presenting and justifying the basic algorithm structure used to implement these heuristics. Once the basic algorithm structure is known, each of the following subsections will augment the basic algorithm with the structures necessary to enforce one or more heuristics, eventually providing a full picture of the algorithm and its inner workings.

4.2.1 Algorithm Structure

As described in Section 2.2, the typical method to find a graph's tree is to use a minimal spanning tree algorithm. However, the edge weight assignments necessary to enforce certain heuristics presented in Section 4.1 are not as straightforward as most tree selection circumstances. By looking at the importance and implementation difficulties associated with each heuristic presented in Section 4.1, one can establish a few important criteria to take under consideration when choosing a tree selection algorithm.

First of all, the Absolute Branch and Chord Heuristics, as well as the Tree Similarity Heuristic, were already implemented in the Variable-Based Tree Selection Algorithm and presented no particular challenge. Furthermore, the Flexible Body Heuristic can also easily be implemented in a minimal spanning tree algorithm. This will be shown in Section 4.2.2.

Enforcing the Kinematic Equation Heuristic requires the estimation of the number of kinematic equations required to model the system. As discussed in Section 4.1.4, this can be done by determining the system's DOF and subtract this number from the number of dynamic equations of the system. Since the DOF calculation does not require a tree in this case, the only criterion necessary is to be able to predict the number of dynamic equations of the system. As will be discussed further in Section 4.2.4, this can be accomplished reasonably easily in most minimal spanning tree algorithms.

When trying to implement the Body Torque and Body Force Heuristics, things quickly become much more complex. These heuristics are based on properties of f-cutset equations. Since these equations are only completely known when the tree is fully selected, their

properties will have to be estimated during the tree selection process. Since the estimated properties of the system's f-cutset equation will change each time a new edge is added to the tree, the edge weights used to describe these properties will have to be assigned dynamically during the tree selection.

In Section 4.1.2, it was observed that when a body's center of mass node is added to the tree, the f-cutset equations in which this body's edge's terminal equations will be found depends on the branch path relating this node to the ground. As will be discussed further in Section 4.2.3, this observation will be a key part of being able to predict an f-cutset's properties. In order for this observation to always be valid during the tree selection process, *the ground node must be the first node placed in the tree*, since the addition of all future body center of mass nodes will need to refer to it. This shall constitute the *first algorithm selection criterion*. Furthermore, *the edges added to the tree at each step of the tree selection process should be adjacent to one of the already existing tree edges*. This will ensure that each new branch added to the tree will have a branch path relating it to the ground node. This will constitute the *second algorithm selection criterion*.

Because of the special features required by the two algorithm selection criteria required to enforce the present tree selection heuristics, a new algorithm shall be developed especially for the selection of trees for a graph of mechanical systems. This new algorithm shall use Prim's Algorithm as its basis. The pseudo-code of Prim's algorithm is presented in Appendix D. Out of the three major minimal spanning tree algorithms discussed in Section 2.2, Prim's algorithm is the only one capable of satisfying both algorithm selection criteria presented above. Furthermore, this minimal spanning tree algorithm is also very simple and its structure provides a flexible framework in which it is easy to add modifications.

The pseudo-code of the modified Prim's algorithm is presented in Figure 4.9. In this pseudo-code, the modifications made to Prim's algorithm are shown in italics and will be explained in future sections of this report. The section of the algorithm presented in this figure will be referred to as the *Base Algorithm*, while the complete tree selection algorithm including the functions called by this Base Algorithm, will be called the *Formulation-Based Tree Selection Algorithm.*

Base Algorithm

- *Execute the* ***Initiation Function*** *(Figure 4.12)*
- Create a list of all the edges in the graph and their incident nodes
- Place the *ground node* of the graph in the tree
- Repeat the following steps until all the nodes are found in the tree:
  - Find all the potential edges (edges that connect to one node in the present tree and one node not in the present tree)
  - *Assign weights to each of the potential edges using the* ***Weighting Function*** *(Figure 4.10)*
  - From these potential edges, find the edge with the lowest weight
  - Add this edge and its new node to the tree
  - *Update the properties of the branches with the* ***Update Function*** *(Figure 4.15)*

Figure 4.9: The Base Algorithm.

4.2.2 Implementation of Static Weights

Since the weights of certain edges must be assigned dynamically, the edge weighting process will be done within the tree selection algorithm rather than prior to its execution. Each time the Base Algorithm finds a series of potential edges (edges that connect to one node in the present tree and one node not in the present tree), it subsequently assigns weights to these edges. This weighting process is done by calling the *Weighting Function*, whose pseudo-code is described in Figure 4.10. In this pseudo-code, the term $Edge.Weight$ refers to the weight associated with the given edge. This notation will be used throughout this algorithm's pseudo-code to represent certain properties associated to certain edges.

The Weighting Function will first give a large weight (100000) to edges connected to a flexible arm's end node. This will implement the Flexible Body Heuristic by guaranteeing that none of these edges are placed in the tree, thus assuring that the flexible arm's edge will be a leaf edge and its end node a leaf node.

Similar to the Variable-Based Tree Selection Algorithm, the Absolute Branch and Chord Heuristics will be implemented by assigning a very low weight (-2) to arm elements or motion drivers, and a high weight (90000) to force/torque drivers and SDAs. The weight of 90000 was chosen so that it is high, yet lower than the weight assigned to enforce the Flexible Body Heuristic, which is a heuristic of higher priority. Since the weight of zero and -1 will be given in future weighting processes, -2 was chosen as the lowest possible weight.

If the potential edges provided to the Weighting Function are not assigned a weight by the previously-described weighting process, they will be sent to the *Dynamic Weighting Function*, whose pseudo-code is presented in Figure 4.13. The exact content of this algorithm will be described further in Section 4.2.3. For now, all we need to know is that all of the edge weights assigned in the Weighting Function will have constant weights, while the weights assigned in the Dynamic Weighting Function will be dynamically assigned.

The last part of the Weighting Function implements the Tree Similarity Heuristic. This is done by multiplying the edge weights obtained thus far by ten and then determining if the given edges are found in the opposite tree than the one being selected. A unit weight is then added to the weight of the edges not found in the opposite tree. Once again, this

Weighting Function

- If the Edge is connected to a flexible arm's end node:
  - $Edge.Weight = 100000$
- Else if the Edge is an arm element or a motion driver:
  - $Edge.Weight = -2$
- Else if the Edge is forced in the tree by the user:
  - $Edge.Weight = -1$
- Else if the Edge is a force/torque driver or an SDA:
  - $Edge.Weight = 90000$
- Else:
  - Find the weight using the **Dynamic Weighting Function** (Figure 4.13)
- $Edge.Weight = Edge.Weight \times 10$
- If the Edge is not in the opposite tree:
  - $Edge.Weight = Edge.Weight + 1$

Figure 4.10: The Weighting Function.

is the same technique as the one used to implement the Tree Similarity Heuristic in the Variable-Based Tree Selection Algorithm.

It is also possible to note that the Weighting Function pseudo-code, presented in Figure 4.9, also calls other functions and has some other properties that are not discussed in this section. These parts of the Weighting Function are used for the implementation of other heuristics that will be presented in subsequent sections of this report.

4.2.3 Implementation of the Body Torque and Force Heuristics

Both the Body Torque and Body Force Heuristics rely on the presence of body edges in the f-cutset equations that are used to form the basic dynamic equations of the system. Since this value cannot be determined until a valid tree is selected, this value will have to be predicted during the tree selection process. Two important observations are used to elaborate a method of predicting the presence of body elements in cutset equations.

The first such observation, called the *branch path observation*, was originally described in Section 4.1.2, and states that *a branch's f-cutset equations will include the body edges of every center of mass body node whose branch path to the ground node passes through the given branch.*

One can make a second important observation if the tree selection takes the shape of the Formulation-Based Tree Selection Algorithm described thus far in Figures 4.9 and 4.10. Within this context, it is possible to determine that *the addition of any joint, body edge, driver or SDA to the tree results in the inevitable addition of a new center of mass body node to the tree. Furthermore, the branch path between this new center of mass body node and the ground will include this joint, body edge, driver or SDA.* This observation shall be called the *added body edge observation.*

This observation can be explained by the fact that each joint, body edge, driver or SDA is either connected directly to a CMBN (center of mass body node) or to an arm that connects to a CMBN. In the first case, the addition of the CMBN to the tree is obvious. In the second case, after adding the given joint, body edge, driver or SDA to the tree, the next edge to be added by the Formulation-Based Tree Selection Algorithm will automatically be the arm to which the given edge is connected into the tree. This is because the arms have

the lowest weight of any edge. Furthermore, the Formulation Tree Selection Algorithm, like Prim's Algorithm, effectively "grows" the tree one branch at a time starting from the ground node. Since trees contain no closed-loops, the joint, body edge, driver or SDA that introduces the CMBN to the tree must be part of its branch path to the ground node.

Combining these two observations, it is possible to establish the *Minimal Body Rule*, which states that *each time a new joint, body edge, driver or SDA is added to the tree, each basic dynamic equation this edge generates can be said to contain at least one body edge. Furthermore, one body edge can be added to the estimation of the minimum possible number of body edges of each dynamic equation generated by the branches in the branch path relating this new edge to the ground node.*

Using this concept, one can temporarily add a potential edge to the tree and calculate the number of body edges that must, as a minimum, be found in the final system's basic dynamic equations, given the edges that are already known to be in the tree. This number of body edges shall be called the *minimum body edge number* in this report. It can be used to place the potential edge, whose resulting tree has the lowest minimum body edge number, in the tree. This will enable the implementation of the Body Torque and Body Force Heuristics within the Formulation-Based Tree Selection Algorithm.

To clarify this process, we can look at the properties of the serial mechanism depicted in Figure 4.11. In this mechanism the body m_1 is connected to the ground with the universal joint u_{10} and the bodies m_1 and m_2 are connected together by the revolute joint h_{11}. The bodies m_2 and m_3 are connected together by the revolute joint h_{12}. The edges r_4-r_9 represent the location where each joint is connected to the bodies, relative to each body's center of mass. Finally, the tree selection for this mechanism is in progress and the branches established thus far in the tree selection process are depicted in bold. For each branch joint, the f-cutset that is known at this point in time in the tree selection, referred to as the *known cutset*, is depicted in dotted lines. These known cutsets "cut" the chords that *must* be present, as a minimum, in the given branch's f-cutset equation, no matter which edges are selected as branches later in the tree selection process.

In order to find the minimum body edge number associated with a given edge, it is necessary to keep track of certain edge characteristics. These characteristics are listed below:

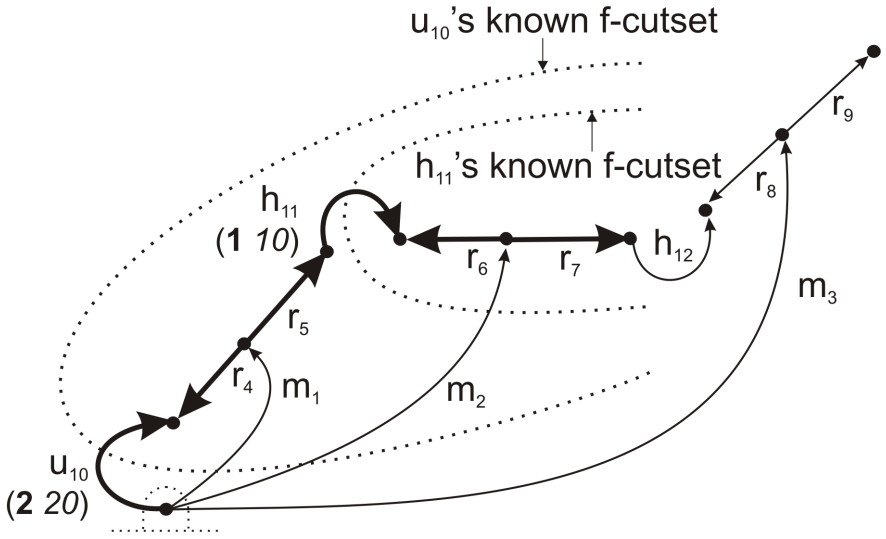


Figure 4.11: Serial mechanism whose rotational tree is in the process of being selected.

NumDynEQs: This characteristic will provide the number of dynamic equations generated by the edge in each tree type. Consequently, it also represents the number of across variables the edge has in the given tree type. Since this characteristic stays constant during the tree selection, its value can be assigned prior to the execution of the tree selection algorithm. In the example presented in Figure 4.11, the universal joint u_{10} has a rotational $NumDynEQs$ value of two, the revolute joints will have a rotational $NumDynEQs$ value of one. All three of these joints will have a translational $NumDynEQs$ value of zero and the arm elements will have a rotational and translational $NumDynEQs$ value of zero.

NumEQs: This characteristic will provide the number of times the f-cutset equations of a given edge will be used within the elaboration of the dynamic equations of the system. In the rotational domain, this shall result in the $NumEQs$ of an edge being equal to its $NumDynEQs$ value, since each edge's f-cutset will only be used in the elaboration of the rotational dynamic equations. However, as discussed in Section 4.1.3, the f-cutset equations generated by the translational tree will be crucial to the formulation of both the translational and rotational dynamic equations, resulting in the value of $NumEQs$ being sometimes higher than the number of translational

equations generated by the given edge. This concept shall be further clarified later in this section. In the example depicted in Figure 4.11, the $NumEQs$ value of the edges u_{10} and h_{11} are depicted in bold under the edge names.

$CutsetWeight$: This characteristic will keep track of the fewest number of body elements that must be found in the given edge's own f-cutset equation. This value will be different for rotation and translation and will constantly be modified during the tree selection process in order to take advantage of the added information provided by each additional branch to the tree.

Its value will be proportional, but not directly proportional, to the number of bodies in the f-cutset equation. When selecting the rotational tree, each chord body element found in the edge's f-cutset equation will add a weight of 10 to this characteristic, while branch bodies will add a weight of 9. For the translational tree, each chord body will add a weight of ten to this characteristic, while branch bodies won't add any weight. The weight differences between chord and branch bodies is due to the different treatment given to them in the Body Torque and Body Force Heuristics.

Let us look at the example depicted in Figure 4.11 where the tree being selected is the rotational tree. In this case, the edge u_{10} will have a $CutsetWeight$ (indicated in italics under the edge's name) of 20 since its known cutset indicates that its f-cutset must at least cut through the chord body edges m_1 and m_2. On the other hand, the edge h_{11}, must only, as a minimum, have the chord body edge m_2 in its cutset, giving it a $CutsetWeight$ of 10. Finally, if the edge m_3 was chosen as a branch in the future, its $CutsetWeight$ would be valued at 9 since in these circumstances the only body edge its cutset would cut is the edge m_3 itself, which would be a branch edge.

$NumInRotEQs$: This characteristic will keep track of the number of times the forces of each arm and each joint allowing translation will be present in the dynamic equations generated by the rotational tree. This characteristic will vary during the rotational tree selection process and will have no translational tree equivalent.

For example, if we consider the partially selected rotational tree depicted in Figure 4.11, the arms r_4 and r_5 will have a $NumInRotEQs$ of two each. This is because

the body m_1 (the body to whom these arms are attached) must only, as a minimum, be present in the edge u_{10}'s cutset (it is only "cut" by u_{10}'s cutset). Since this cutset will be used to generate two rotation dynamic equations (since u_{10} has a $NumEQs$ of two), m_1's two arms will each be present twice, one in each of the two dynamic equations. Similarly, the arm edges r_6 and r_7 will have a $NumInRotEQs$ of three since the body m_2 will appear in a minimum of three dynamic equations (two generated by u_{10}'s cutset and one generated by h_{11}'s cutset).

Now that these edge characteristics have been defined, the pseudo-code enabling the implementation of the Body Torque and Body Force Heuristics, namely the Initiation, Dynamic Weighting and Update Functions, can be presented. However, it is important to note that each of the edge characteristics presented above, with the exception of the $NumInRotEQs$, have different values for the rotational and translational domain. Because of this, though the pseudo-codes will be presented in their entirety (all of the code necessary for both rotation and translation), the selection of the rotational tree and translational tree will be discussed separately in the explanation that follows. This should help avoid confusion and simplify the understanding of the tree selection process.

Before being able to use the edge characteristics within the tree selection, initial values must be given to these characteristics for each edge in the graph. These initial values are assigned by the *Initiation Function*, whose pseudo-code is presented in Figure 4.12. This function is the first thing to be executed within the Formulation-Based Tree Selection Algorithm, since it is called in the first line of the Base Algorithm, presented in Figure 4.9.

Selection of the Rotational Tree

Looking at the Initiation Function from the point of view of selecting the rotational tree, we see that the $NumDynEQs$ characteristic of each edge is assigned a value equivalent to the number of dynamic equations that this edge will produce if it is placed in the rotational tree. These values are found in Appendix C. The $NumEQs$ characteristic is then assigned the same value. This is because, as mentioned earlier, in the case of the rotational tree, the f-cutsets, which are used to provide a torque equilibrium equation, are only used during the formulation of basic rotational dynamic equations. This also coincides with the Body

Initiation Function

- For each Edge in the graph:
  - $Edge.NumDynEQs$ = Value found in Table C.1
  - $Edge.NumEQs = Edge.NumDynEQs$
  - If the translational tree is being selected and the Edge is not an arm:
    - $Edge.NumEQs = Edge.NumEQs + Edge.NumInRotEQs$
    - Find the arm edges whose end node is one of the nodes to which this edge is connected (they shall be named: CArms):
      - $Edge.NumEQs = Edge.NumEQs + \sum CArm.NumInRotEQs$
  - $Edge.CutsetWeight = 0$
  - $Edge.NumInRotEQs = 0$
  - $Edge.Weight = 0$
- $DOF = 6(nb - 1) - \sum nj_i$ (See Section 4.2.4 for more detail)
- Execute the **Kinematic Complexity Function** (Figure 4.17)

Figure 4.12: The Initiation Function.

Torque Heuristic that is only concerned with the body elements present in the f-cutset equations used to formulate the rotational basic dynamic equations.

The next stage of the Initiation Function sets the value of the $CutsetWeight$, $NumInRotEQs$ and $Weight$ characteristics to zero. The two final parts of the Initiation Function estimates the degrees of freedom, assigns them to the variable DOF, and executes the Kinematic Complexity Factor. These features will only be used when enforcing the Kinematic Equations Heuristic and will hence only be fully discussed in Section 4.2.4.

Once the Initiation Function is completed, the Base Algorithm (Figure 4.9) goes on to establish a series of potential edges that could be added to the present tree. Each of these potential edges are then sent to the Weighting Function (Figure 4.10), which first enforces the Flexible Body Heuristic as well as the Absolute Chord and Branch Heuristics. If the potential edge's weight is not assigned by these initial considerations, it is sent to the Dynamic Weighting Function, whose pseudo-code is presented in Figure 4.13. This function assigns a weight to the potential edges by enforcing the Body Torque and Kinematic Equation Heuristics.

The Dynamic Weighting Function will enforce the Body Torque Heuristic by using the Minimal Body Rule. This shall be accomplished by first temporarily placing the provided potential edge in the tree and updating the $CutsetWeight$ characteristic of each branch to keep track of the changes brought forth by this new branch. Then, the sum of the multiplications of each branch's $NumEQs$ value (the number of rotational dynamic equation generated by the edges) with its $CutsetWeight$ value (proportional to the minimal body edges number of the edge's own f-circuit equation) will provide a weight proportional to the minimal body edges number of the present tree. This value shall be assigned to the potential edge's weight characteristic.

Before looking at the details of how the Dynamic Weighting Function implements this process, let us look at an example by applying it to the serial manipulator depicted in Figure 4.11. At the stage of the tree selection process depicted in this figure, the two potential edges of this mechanism will be the edges m_3 and h_{12}. Let us first calculate the edge m_3's weight by temporarily placing it in the tree. Since the branch path to the ground of m_3's body node only consists of the edge m_3 itself, the $CutsetWeight$ of all the present branches are not affected by the addition of the edge m_3 to the tree (the known cutsets

Dynamic Weighting Function

- $Edge.Weight = Edge.NumEQ \times 10$
- If Edge is a body edge:
  - If the rotational tree is being selected:
    - $Edge.CutsetWeight = Edge.NumEQ \times 9$
  - Else:
    - $Edge.CutsetWeight = Edge.NumEQ \times 10$
- Find branch path relating the edge to the ground node
- For every branch in the graph:
  - If the Branch is part of the branch path to the ground node:
    - $Edge.Weight = Edge.Weight + (Branch.NumEQ \times (Branch.CutsetWeight + 10))$
  - Else:
    - $Edge.Weight = Edge.Weight + (Branch.NumEQ \times Branch.CutsetWeight)$
- If DOF >(TotNumDynEQs + Edge.NumDynEQs)
  - $Edge.Weight = Edge.Weight + (TotNumDynEQs + Edge.NumDynEQs - DOF) \times KinEQComplexity$

Figure 4.13: The Dynamic Weighting Function.

of all the other edges will not include m_3). At this point in time we know that m_3 has a $NumEQs$ of 3 and it is given a $CutsetWeight$ of 9 since its known cutset "cuts" itself (a branch body edge). The calculation of m_3's weight is then shown in equation (4.12), where each edge's $NumEQs$ characteristic value is depicted in bold and the edge's $CutsetBodies$ characteristic value is depicted in italics, and where the edge to which these values belong to is identified underneath these values. The arm edges are not included in this calculation since these edges add nothing to this weight calculation because their $NumEQs$'s value is always zero.

$$\text{Edge } m_3\text{'s Weight:} \quad \underbrace{\mathbf{2} \times \mathit{20}}_{u_{10}} + \underbrace{\mathbf{1} \times \mathit{10}}_{h_{11}} + \underbrace{\mathbf{3} \times \mathit{9}}_{m_3} = 77 \tag{4.12}$$

If we want to know the weight of the potential edge h_{12}, we place this edge in the tree and remove the edge m_3. The addition of h_{12} will inevitably add m_3's body node to the tree, since the arms r_8 and r_9 must be found in the tree (due to the Absolute Branch Heuristic). m_3's body node will have a branch path to the ground passing through all the present branches. This will increase the $CutsetWeight$ of each branch (including the new branch h_{12}) by 10, since their known cutset now "cuts" the chord body edge m_3. This is depicted in Figure 4.14. The calculation of h_{12}'s weight is then shown in equation (4.13) which uses the same conventions as the equation (4.12).

$$\text{Edge } h_{12}\text{'s Weight:} \quad \underbrace{\mathbf{2} \times \mathit{30}}_{u_{10}} + \underbrace{\mathbf{1} \times \mathit{20}}_{h_{11}} + \underbrace{\mathbf{1} \times \mathit{10}}_{h_{12}} = 90 \tag{4.13}$$

These results show that the addition of m_3 to the tree will result in fewer body edge torques in the system's resulting dynamic equations, thus satisfying the Body Torque Heuristic. This can be easily confirmed by deriving the rotational dynamic equations for the final tree containing the edge m_3 and the final tree containing the edge h_{12}.

Now that the method used to enforce the Body Torque Heuristic has been properly demonstrated, let us look at how this process is implemented in the Dynamic Weighting Function. First, the Added Body Edge Observation, presented earlier in this section, demonstrates that if the potential edge is added to the tree, the body edge of the potential edge's distal body must be found in this potential edge's f-cutset equation. This is the

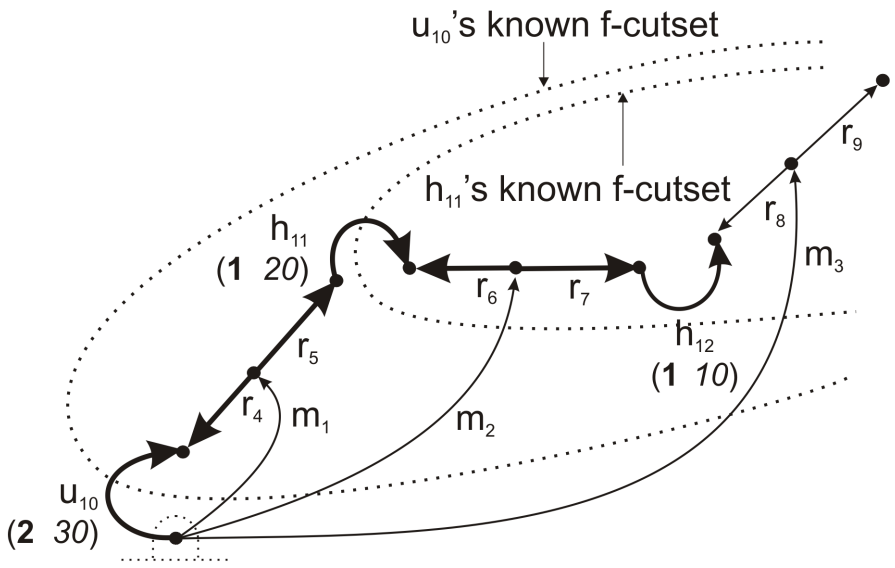


Figure 4.14: Serial mechanism whose rotational tree includes h_{12}.

only body edge whose presence in this potential edge's f-cutset equation is inevitable. Because of this, the weight this potential edge adds to its own total weight will be equal to this potential edge's $NumEQs$ characteristic multiplied by nine if the potential edge is a body edge, and multiplied by ten if the potential edge is of any other edge type. This is implemented in the Dynamic Weighting Function's pseudo-code's first six lines.

Now that the weight that the potential edge adds to itself has been taken into account, the weight associated to all the other branches must be added to the potential edge's weight. While doing this, it is important to keep track of the changes to each branch's $CutsetWeight$ characteristic brought forth by the addition of the potential edge and its distal body's arms to the tree. The nature of these changes are described in the *branch path observation*, which stipulates that the body edge associated to the new body added to the tree by the potential edge will be found in the f-cutset equations of every branch relating this body to the ground. This means that the $CutsetWeight$ characteristic of each of the branches in this potential edge's branch path to the ground must be increased by a value of ten. Note that branch body edges will only be found in its own branch path to the ground node, and as such we do not need to worry about multiplications by nine instead

of ten when modifying the branch path's $CutsetWeight$ characteristic.

However, it is important to note that the changes made to each branch's $CutsetWeight$ characteristic in the Dynamic Weighting Function are not made in a permanent fashion. This is done on purpose in order to ensure that the changes that must be made to these branch elements do not affect the weight calculation of other potential edges. However, once the potential edge with the lowest weight is found and placed in the tree, the changes to the branch's $CutsetWeight$ characteristic, brought forth by the permanent addition of this new branch to the tree must be made permanent. This shall be carried out by the Update Function, which is called at the final pseudo-code line of the Base Algorithm (Figure 4.9), and whose own pseudo-code is presented in Figure 4.15.

However, before turning our attention to the Update Function, it is important to take note that the two final pseudo-code lines of the Dynamic Weighting Function will implement the Kinematic Equation Heuristic, and as such will only be presented in the next subsection. The Kinematic Equation Heuristic must be implemented in parallel with the Body Torque and Body Force Heuristics, as described in Sections 4.1.2 and 4.1.3.

Returning to the Update Function's pseudo-code, the first nine lines of this pseudo-code are clearly dedicated to updating the branch's CutsetWeight values as previously described. The Update Function is performed on all new tree edges and not only on edges whose weights were established by the Dynamic Weighting Function. Because of this, one can note that the $CutsetWeight$ value given to arm elements does not realistically predict the actual number of bodies in the f-cutsets of these elements. However, since the $NumEQs$ value for arms is always zero, we can conclude that this erroneous $CutsetWeight$ value for arms will have no impact on the tree selection and therefore no special exception is established to correct this inaccuracy.

If the new branch is an arm or an edge that allows translation, the two next lines of the pseudo-code, following the $CutsetWeight$ updates, set the $NumInRotEQs$ value of the newly-added branch. This characteristic is found by determining how many dynamic equations contain the body edge of the body to which this new branch is attached. The *branch path observation* states that this body edge will be found in the dynamic equations of all the branches relating this new branch to the ground. Hence, the value of the new branch's $NumInRotEQs$ characteristic is found by simply summing the $NumEQs$ value

Update Function

- $Edge.CutsetWeight = 10$
- If Edge is a body edge:
  - If the rotational tree is being selected:
    - $Edge.CutsetWeight = 9$
  - Else:
    - $Edge.CutsetWeight = 10$
- Find branch path relating the Edge to the ground node
- For every branch in this path:
  - $Branch.CutsetWeight = Branch.CutsetWeight + 10$
  - If the rotational tree is being selected and the edge is an arm or an edge that allows translation:
    - $Edge.NumInRotEQs = Edge.NumInRotEQs + Branch.NumEQs$
- $TotNumDynEQs = TotNumDynEQs + Edge.NumDynEQs$

Figure 4.15: The Update Function.

of all the branch edges relating this new branch to the ground.

The final pseudo-code line found in the Update Function updates the $TotNumDynEQs$ by adding to its value the number of dynamic equations added by the new branch. The $TotNumDynEQs$ variable is a characteristic associated to the graph itself and keeps track of the fewest number of dynamic equations that must be used to model the system given the knowledge obtained by the edges already in the tree and in the opposite tree. Clearly, this graph characteristic will be used to implement the Kinematic Equation Heuristic and so its use shall only be discussed further in Section 4.2.4.

Selection of the Translational Tree

Now that we have gone through all the steps necessary to establish the rotational tree, we shall look more closely at the selection of the translational tree.

There are only three fundamental differences between the selection of the rotational and translational trees. The first is the fact that the process required to assign the $NumInRotEQs$ characteristic of the branches is not present during the selection of the translational tree. However, this distinction is rather obvious since the nature and use of the $NumInRotEQs$ allows no translational equivalent to exist.

The second variation is that, if the translational tree is being selected and the potential edge presented to the Dynamic Weighting Function is a body edge, the weight that this edge adds to its own total weight will be zero rather than nine. This is due to the fact that, unlike the Body Torque Heuristic, the Body Force Heuristic only minimizes the presence of chord body edges in the dynamic equations. This will also affect the Update Function, which will assign a $CutsetWeight$ value of zero to body edges in the translational tree.

Finally, the third variation in the selection of the translational tree lies in assigning the value of the $NumEQs$ edge characteristic that is found in the pseudo-code of the Initiation Function presented in Figure 4.12. Unlike the rotational tree's f-cutset equations, the translational tree's f-cutset equations are going to be used in the formulation of both the rotational and translational dynamic equations. This fact is also expressed in the Body Force Heuristic that takes into account the chord body forces present in the formulation of all the dynamic equations.

During the rotational tree selection, the $NumInRotEQs$ characteristic recorded the number of times each arm force and each force associated to edges allowing translation were present in the rotational basic dynamic equation. These forces will be substituted by their edge's translational f-cutset equations during the rotational dynamic equation formulation. In the case of the edges allowing translation, their $NumInRotEQs$ can simply be added to the number of dynamic equations they generate in translation (their $NumDynEQs$ characteristic) to establish the total number of times the translational f-cutset equations of these edges will be used in the dynamic equation formulation process (their $NumEQs$ characteristic). This is implemented in the fifth pseudo-code line of the Initiation Function (Figure 4.12).

However, in the case of the f-cutset equation of arm elements needed for the rotational dynamic equation formulation, things are not as simple. Since, according to the Absolute Branch Heuristic, the arm elements must be in the tree, these edges are assigned constant weights during the Weighting Function and are not sent to the Dynamic Weighting Function. The Dynamic Weighting Function is the only place where the number of body forces used to generate the dynamic equations of the system is calculated. Since the arms are not sent to this function, the information about the arm's translational f-cutset equation needed in the system's equation formulation must be acquired in another, more indirect, fashion.

In order to find such a solution, let us look at the general form that the arm f-cutsets take. Figure 4.16 represents a section of a graph at a certain time in the tree selection process. The edges already selected in the tree are shown in bold and the limit of this graph section is depicted by the edge r_7 which fades out in this figure but in reality could connect to a graph of any shape (it can be noted that the body edges of the rest of the graph are not depicted either). In this figure, the properties of the f-cutsets of each branch arm element that can be determined with certainty at this stage of the tree selection are depicted by doted lines crossing each branch arm element. These f-cutsets are not fully defined since the tree is not yet fully selected; however, the final f-cutsets will have to at least take the form shown and will have to at least cut through the body elements it already cuts through in this figure, no matter what edges are selected in the tree in the future.

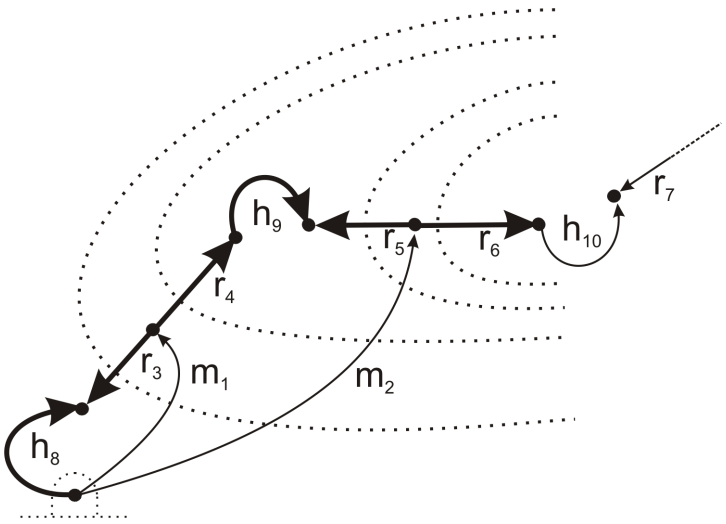


Figure 4.16: General form of the arm cutsets.

Looking at the body edges cut by each of the arm f-cutset in Figure 4.16, it is possible to establish that the arm cutsets of the arm edge immediately preceding and immediately succeeding a branch joint, driver or SDA will include the exact same body elements as the branch joint, driver or SDA's f-cutset includes. In the figure, this can be seen by the f-cutsets of arm r_4 and arm r_5 both at least cut through the body edge m_2 (in other words, their f-cutsets must at least contain the chord body edge m_2, no matter what new branches are added to the tree), which is the exact same body the f-cutset for the joint h_9 must at least cut through. In the case where an arm is connected to multiple joints, the arm element's f-cutset will include the chords cut by the f-cutset of all of the joints the arm is connected to. These observations lead to the conclusion that the number of times each arm's translational f-cutset appears in the rotational dynamic equations (Its $NumInRotEQs$ characteristic) can be transferred to the $NumEQs$ characteristic of the joint, driver or SDA to which this arm is connected. This concept can be seen in the sixth and seventh line of the Initiation Function (Figure 4.12), where each edge that is not an arm is assigned the $NumInRotEQs$ value of the arm element's whose end node is one of the two nodes to which the given edge connects. The arm end nodes are not considered in order to avoid body edges to be assigned its arm's $NumInRotEQs$ value since if the body

edge is a branch, this body's arm edges f-cutset do not include this body edge.

Now that the number of chord body forces found in the rotational dynamic equation formulation added by the arms in the rotational tree are considered within the properties other edges, the translational tree selection process can continue in the same fashion as the selection of the rotational tree.

4.2.4 Implementation of the Kinematic Equation Heuristic

Section 4.1.2 defines the Kinematic Equation Heuristic as trying to minimize the number of kinematic equations used to model the system. This same section also suggests that minimizing the complexity of these equations would also be advisable, yet proposes no easily applicable technique to implement this. Trying to estimate the number and complexity of a system's kinematic equations within the context of the Formulation-Based Tree Selection Algorithm presented thus far proves to be difficult.

Section 4.1.2, describing the Kinematic Equation Heuristic, provides a description of two possible methods of predicting the number of kinematic equations of the system. The first of these methods relies on the use of a technique capable of predicting a system's degree of freedom. Once this degree of freedom is known, the number of kinematic equations in the system can be determined by simply subtracting the number of DOF from the number of dynamic equations in the system ($m = n - DOF$). There exist many different techniques of estimating the DOF of a system and many of these techniques are relatively simple and could easily be included in the tree selection process. This is clearly seen by the presence of the DOF global variable in the Initiation Function's last step. The use of this variable, as well as the techniques used to find it, shall be presented later in this section. Similarly, the number of dynamic equations needed to model the system at each step of the tree selection process is also easily calculated. Its calculation is already included in the tree selection algorithm via the last line of the pseudo-code for the Update Function presented in Figure 4.15. However, this method does carry a few disadvantages. First, though most methods of estimating the degree of freedom of a mechanism can give relatively good estimates for most systems, there exists no method capable of accurately determining the degree of freedom of all possible mechanical systems [32]. Furthermore, this method of

calculating the number of kinematic equations does not necessarily provide a good basis for estimating the complexity of these equations since the f-circuits creating these equations are not revealed with this method.

The calculation of the number of dynamic equations and DOF are already present in the pseudo-codes presented thus far. As this would suggest, the method chosen for determining the number of kinematic equations was to estimate the system's DOF and subtract it from the number of dynamic equations. This is because, despite its shortcomings, this method's simplicity as well as its easy implementation and relative accuracy make it a good choice for the algorithm.

Section 4.1.2's second suggested method of finding the number of dynamic equations of a system involves counting the number of independent vectors in the basis of the reaction space of each of the system's chords who have active or passive constraints. Even though this technique would have the potential to also provide a good basis for predicting the kinematic equation's complexity, its implementation within the algorithm established thus far proves too complex and hence will not be used. This is because the present tree selection algorithm finds a graph's tree by adding branches to a tree one at a time until a tree is selected. Determining the presence of chord elements is merely a consequence of this process. Since the chord elements are not actively selected from the graph, it is difficult to predict their properties while the tree is being selected.

If we turn our attention back to the DOF technique, it is clear that the first step in implementing this technique is to find a method of calculating the degrees of freedom of the mechanism. To do this, Gruebler's equation, which is simple and widely used, shall be implemented to find the system's DOF. The Gruebler's equation that is tailored for spatial systems is:

$$DOF = 6(nb - 1) - \sum nj_i \tag{4.14}$$

where nb represents the number of bodies found in the system (including the ground) and nj_i represents the number of independent reactions (the number of vectors in the base of the joint's constraint space) of the ith joint in the system.

Since no tree needs to be selected in order to execute Gruebler's equation, the calcula-

tion of the system's predicted DOF will be performed in the beginning of the tree selection process and, as mentioned earlier, is found in the Initiation Function presented in Figure 4.12, which is in turn called in the first pseudo-code line in the Base Algorithm of Figure 4.9. If, for some reason, Gruebler's equation provides results that are obviously inaccurate, such as negative values, the DOF variable is simply set to zero. It is important to note that, unlike some edge characteristics, the global DOF variable represents the same variable in both rotation and translation.

Now that the DOF of the system is predicted, this information can be used to predict the number of kinematic equations that must be present at each step of the tree selection process. As mentioned a few times in Section 4.1, the implementation of the Kinematic Equation Heuristic must be done in conjunction with the Body Torque and Body Force Heuristics. This is why it is implemented at the two last lines of the Dynamic Weighting Function of Figure 4.13.

It gets executed at a time in the tree selection process where a series of potential edges has been found. These potential edges have been assigned weights that implement the Flexible Body Heuristic, the Absolute Branch/Chord Heuristics as well as the Body Force or Torque Heuristic. It is at this point that the Formulation-Based Tree Selection Algorithm will assign weights in order to enforce the Kinematic Equation Heuristic.

The number of dynamic equations that must, as a minimum, be generated by the tree containing the given potential edge, which we shall call the *minimum dynamic equation number*, can easily be found. This is done by adding the number of dynamic equations that will be generated by the given potential edge if placed in the tree with the number of dynamic equations that is required by the rest of the tree as well as the opposite tree (the tree, translational or rotational, not presently being selected by the algorithm). This value is found at the end of the Update Function (Figure 4.15) and is assigned to the global variable called $TotNumDynEQs$. As with the DOF variable, this global variable can be modified in both a rotation and translation context and does not contain separate values for both.

The Kinematic Equation Heuristic shall be enforced by first determining how many kinematic equations will be required to model the system if the provided potential edge is added to the present tree. This value will then be multiplied by a factor representing

the complexity of these kinematic equations, called $KinEQComplexity$, and the resulting value will be added to the weight of the edge. Again, the $KinEQComplexity$ variable will have different values for both rotation and translation. This process is clearly defined by the Kinematic Heuristic Function's pseudo-code for both rotation and translation.

Determining the $KinEQComplexity$ for the rotational tree is quite difficult. The equation complexity is highly dependent on the properties of each chord's f-circuit equation. However, it can be very difficult to determine the f-circuit equations before the tree is fully selected. Furthermore, even if these f-circuit equations were known, it would be extremely difficult to predict the complexity of these equations since they are highly affected by the evaluation of dot products, which require extensive knowledge of the system's geometry. For these reasons, a trial and error approach was adopted in order to find an appropriate weighting factor. Through this process it was established that *a $KinEQComplexity$ factor of approximately three provides the best results.* This is shown in the Kinematic Complexity Function, whose pseudo-code is presented in Figure 4.17, which is executed at the end of the Initiation Function. This would indicate that each kinematic equation generated by the rotational tree adds roughly one third of the complexity of a body torque found in the dynamic equation.

The $KinEQComplexity$ variable for translation is of greater importance than the one for rotation. This is because the weight used to implement the Body Force Heuristic assigns a weight of ten for every chord body force used to formulate the system's equations; however, the complexity added by one of these chord body forces is generally fairly low compared to the complexity added by a kinematic equation. This results in the fact that the weight due to the Kinematic Equation Heuristic that will be assigned to these edges will need to play a much bigger role in each edge's final weight than was the case for the rotational tree.

Thankfully, more knowledge of the system is available when trying to determine the $KinEQComplexity$ for the translational tree, since the rotational tree is already selected. The kinematic equations complexity is dependent on the dot products that, in turn, are dependent on the across variables found within the rotational tree. Since the exact f-circuits associated to each translational kinematic equation is not known, one single general $KinEQComplexity$ factor will have to be established to represent the complexity of the

Kinematic Complexity Function

- If selecting the rotational tree:
  - $KinEQComplexity = 3$
- Else:
  - Find the f-circuit equation edges (CirEdges) of each rotational chord
  - For each chord
    - $Rotations = \sum CirEdge.NumDynEQs[Rotation]$
  - $KinEQComplexity = max(Rotations) \times 10$

Figure 4.17: The Kinematic Complexity Function.

translational kinematic equations of the system.

The dot products found in the kinematic equations are dependent on many factors such as geometry and the amount of rotation between the reference frames used to model the system. Since most of the system's geometry is unknown in the pre-processing environment in which the tree selection is implemented, this factor shall not be used. Furthermore, each edge's terminal equation has the possibility to be expressed relative to different coordinate frames, which is also generally unknown in the pre-processing environment.

However, there is one important source of kinematic equation complexity that can be estimated. When edges that are chords in the rotational tree are found in the f-circuits that are used to generate the translational kinematic equation, the orientation of these edges, needed to determine the dot product, are found by the rotational branch transformation equations. These equations will be established by multiplying the rotation matrices of every edge in the rotational f-circuit of this chord. The resulting rotation matrix for this rotational chord generally contains a series of complex rotations that add much complexity to the system's kinematic equations when the dot products are performed.

In order to get a general approximation of the complexity added to the kinematic equations by the rotational chord edges, the number of rotations performed in every f-circuit of the rotational tree is found by adding the $NumDynEQs$ characteristic in rotation of each edge in these rotational f-circuits. Once the number of rotations found in every rotational f-circuit is determined, the number of rotations found in the chord having the maximum amount of rotation is multiplied by ten and used as the system's $KinEQComplexity$ variable in translation. The multiplication by ten is used to elevate the value of the $KinEQComplexity$ in order to better compete with the edge weight of ten given for every chord body in the system's dynamic equations. Unlike for the body torques, the complexity added to the system equations by a single body force is generally less than the complexity added by one kinematic equation.

Using this method of estimating the $KinEQComplexity$ for translation will ensure that more complex models will have higher $NumDynEQs$ values allowing the Kinematic Equation Heuristic to compete with the higher weight assigned by the Body Force Heuristic due to the mechanism's complexity. The Pseudo-code for this process is presented in the Kinematic Complexity Function of Figure 4.17.

Admittedly, this measure of the $KinEQComplexity$ for translation only has a limited connection to the real translational kinematic equations complexity. Other methods of estimating the complexity of the kinematic equations have been attempted. These were based on the number of edges in the graph, the number of bodies in the system, the degrees of freedom of the system, etc. However, no other method was capable of estimating the complexity of the kinematic equation of all the benchmarks tested as well as the method based on the rotational f-circuit rotations presented above.

Let us look at the example of the spatial slider-crank already discussed in Section 4.1.2, whose rotational graph is shown in Figure 4.3. In this figure the tree is depicted in bold. From this graph, we see that the f-cutset equation of the edge m_1, consisting of edges r_5, h_9 and r_4 will only contain one rotation provided by the edge h_9. After looking at every chord's f-circuit equation we find that edge b_{10}'s f-circuit equation contains the most rotations. It contains three rotations (two from edge h_9 and one from edge u_{11}). Because of this, this graph's $KinEQComplexity$ translational characteristic will have a value of thirty. This means that it is estimated that the kinematic equations generated for this

model should be approximately as complex as the complexity added by the presence of three chord body forces used to generate the system's dynamic equations.

4.2.5 Coordinate Type Selection

At this point in the elaboration of the Formulation-Based Tree Selection Algorithm, all the heuristics proposed in Section 4.1 have been implemented within the algorithm. This algorithm is capable of selecting trees, and hence coordinates, which provide models which produce low simulation times. Furthermore, if a user wants to know the value of other variables of the system not found in the automatically selected coordinates, he can obtain them by back-substitution as discussed in Section 2.3. However, no matter how accurate the tree selection algorithm is, there are still situations where the user would need a certain level of control over the coordinate selection. For example, the user could want to model a certain mechanism using a specific set of coordinates, or requires that certain variables be included in the coordinate set. The reason for these requirements could be, for example, the use of certain modelling variables to model the mechanical system for research purposes or for teaching purposes.

The first, and most important, user control feature to add to the tree selection algorithm, is to allow the user to indicate a series of edges whose across variables are required to be in the system's modelling coordinates. This can be implemented in a very simple fashion. When permanent weights are assigned to potential edges in the Weighting Function, a weight of -1 can be given to edges that the user requires in the tree. This can be seen in the Weighting Function's pseudo-code in Figure 4.10. The low weight of these edges will force them into the tree. The weight of -1 was chosen in order to be lower than any weight that can be assigned by the Dynamic Weighting Function, whose lowest weight is zero. It is also set higher than the -2 weight assigned to edges by the Absolute Branch Heuristic since this heuristic is necessary to avoid the limitation of McPhee's graph-theoretical equation formulation method [19] used in this report. This ensures that a valid tree is always selected. Once the edge is selected in the tree, the Update Function assures that this new branch's properties are taken under consideration when selecting the next edge to be placed in the tree.

This weighting system also ensures that the user does not force the selection of an invalid tree. Since the Formulation-Based Tree Selection Algorithm itself assures the selection of a valid tree, even if the user selects a series of edges having a closed-loop, the tree selection algorithm will automatically not select one of these edges in the tree despite their extremely low weights. If required, a warning can be given to the user when such a situation occurs with an explanation to the user on how to avoid this problem.

Furthermore, if the edges selected by the user do not form a complete tree by themselves, the most appropriate series of edges will be added to the tree by the Formulation-Based Tree Selection Algorithm. This will form a valid tree containing the modelling coordinates requested by the user as well as a set of extra coordinates that are necessary to form a valid tree. Once again, the users can be notified of the presence of these extra variables as well as the reason of their presence.

Another way the tree selection algorithm could be tailored to the user's preferences is to allow the user to model his system using the coordinate type of his choice. The present tree selection algorithm already selects Branch Coordinates by default since it is only limited by the selection of a valid tree from the graph.

To model Absolute Coordinates, the model does not even need to be sent to the tree selection algorithm since this coordinate type is acquired by simply selecting all the arms and body edges as the only edges in each tree. In the case of the Joint Coordinates, one simply needs to add a very big weight to all the body edges during the tree selection process. This will force the joints to be selected in the tree first and the use of the tree selection heuristics of Section 4.1 will select a set of Joint Coordinates. A combination of the two previous techniques can also be used to assure that the coordinates found by the tree selection algorithm are in the form of Absolute Angular Coordinates.

4.3 Example

Section 3.4 stated that using the minimal number of modelling variables in a mechanical system's coordinate set does not always produce the symbolic models having the most efficient simulation times. In most of the benchmarks where more than the minimal possible number of modelling variables resulted in faster simulation times, there were few extra variables and they were found in the translational tree. However, the 3-DOF spatial parallel manipulator, found in Appendix A.12, presented the most perplexing case of coordinate selection. This is because its most efficient model contained a number of modelling variables that far exceeded the minimal possible number. Furthermore, a large number of these extra variables were rotational variables, something not encountered in any other benchmark problem.

Because of the major difficulty associated with selecting the optimal coordinates of this system, it was chosen as the example problem for the Formulation-Based Tree Selection Algorithm. Since this tree selection algorithm's implementation can get very complex to demonstrate, this will be the only example developed in this section. The Formulation-Based Tree Selection Algorithm's tree selection results for other benchmark problems can be found in Appendix A.

The 3-DOF spatial parallel manipulator is presented in Figure 4.18, a detailed view of one of its legs is presented in Figure 4.19(a) and its graph is presented in Figure 4.19(b). This manipulator consists of a platform connected to the ground by three similar legs and has a tool placed on top of the platform. The legs are each connected to the ground by prismatic joints (s_{25}, s_{28}, and s_{31}), which originate from the global reference frame.

The first prismatic joint is oriented toward the global X axis while the two others are oriented 120 degrees and 240 degrees away from the global X axis respectively. The leg's shank body (m_1, m_3, and m_5) are connected to the leg's thigh body (m_2, m_4, and m_6) by a spherical joint (b_{26}, b_{29}, and b_{32}). The thigh bodies are then connected to the platform by revolute joints (h_{27}, h_{30}, and h_{33}) that are connected to the platform at 0, 120, and 240 degrees relative to the platform's local X axis. The orientation of the revolute joint axes of rotation are 90, 210, and 330 degrees relative to the platform's local X axis respectively. Finally, the geometry, inertia properties, and initial conditions of the mechanism are taken

from Simoni [28] and are given in Appendix A.12.

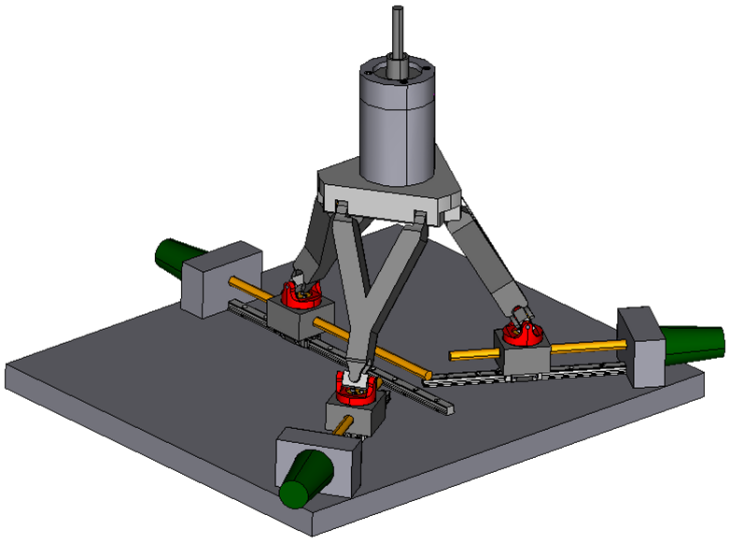
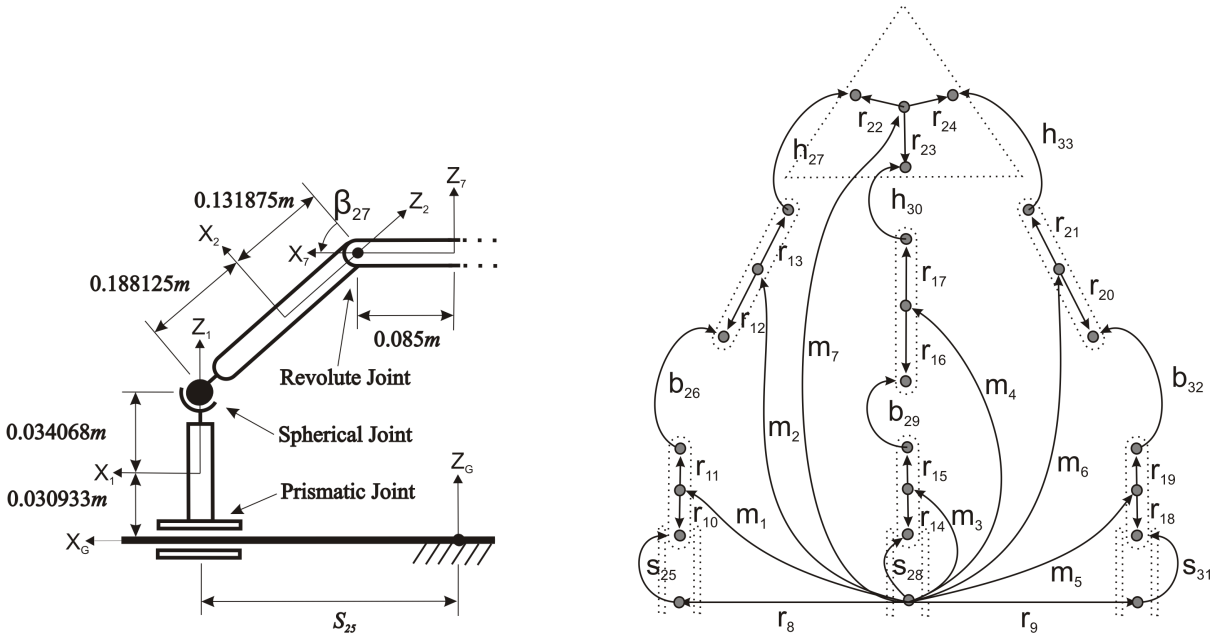


Figure 4.18: 3-DOF spatial parallel manipulator.

The first tree to be selected, as explained in Section 4.1.6, will be the rotational tree. The initial step of this tree selection will be to assign the value of the $NumEQs$ characteristic of each edge in the graph. The value of this characteristic will be equal to the number of rotational across variables of each edge, which can be found in Appendix C. The mechanism's graph in Figure 4.20(a) depicts the $NumEQs$ characteristic in the rotational domain of every edge.

It is also possible to determine the DOF in rotation of the mechanism using Gruebler's equation described in equation (4.14). In the present case, there are seven bodies, each of the three prismatic joints have five constraints, each of the three spherical joints have three constraints, and the three revolute joints each have five rotational constraints. When these values are entered in equation (4.14), it is predicted that the system has three DOF, which is the correct value for this system.

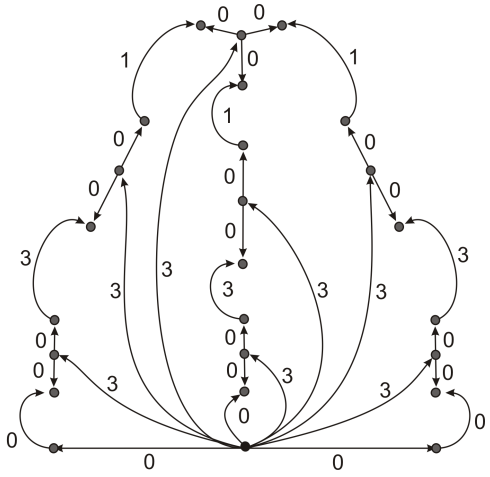
The next step in the tree selection process is to place the ground node in the tree and determine the potential tree edges. These are the edges that contain one node in the present tree and one node outside the present tree. The potential edges found at this time



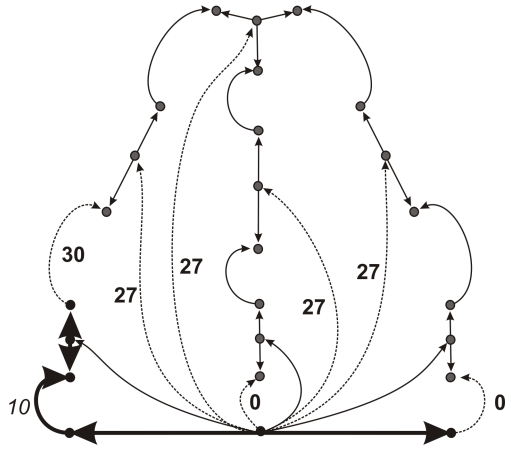
(a) Spatial parallel manipulator's legs

(b) Linear Graph

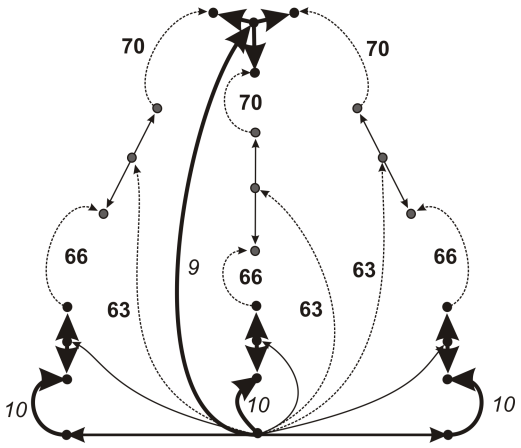
Figure 4.19: 3-DOF spatial parallel manipulator details and its graph.



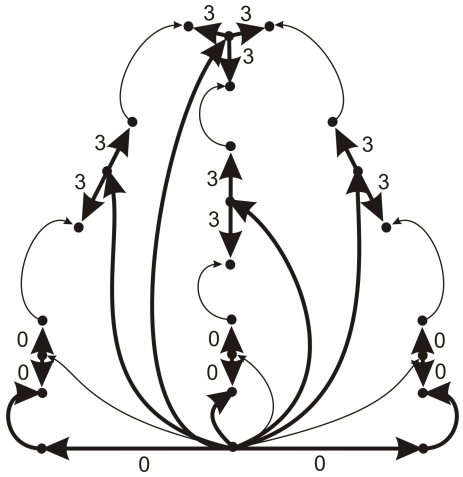
(a) $NumEQs$ characteristic values



(b) After five branches have been selected



(c) After 15 branches have been selected



(d) Rotational Tree $NumInRotEQs$

Figure 4.20: 3-DOF spatial parallel manipulator's rotational tree at various stages of the tree selection.

in the tree selection process are all the edges connected to the ground node (r_8, r_9, s_{28}, and m_1 to m_7). Since we know that arms are always given the lowest possible weight, it is obvious that the two arms r_8 and r_9 will be the first two edges placed in the tree.

Once these two arms are found in the tree, the potential edges become all three prismatic joints and all the body edges. In order to find the weight of these edges, the Dynamic Weighting Function, presented previously in Figure 4.13, is executed for each edge. In this case the prismatic joints will have a weight of zero since they have a zero value for their $NumEQs$ characteristic (shown in figure 4.20(a)) as well as all the other edges in the tree thus far (the two arms r_8 and r_9). Furthermore, no kinematic equation would be added to the system by the addition of one of the prismatic joints in the rotational tree. At this point in the tree selection process, the body edges would have a weight of 27 due to the presence of their own body torques, worth a weight of nine, in each of the three dynamic equations they would generate. As with the prismatic joints, the other edges already in the tree would not add to the body edge's weight and the three dynamic equations that the body edges would add to the system would not require the addition of kinematic equations since there are three "unused" DOF.

Since each of the three prismatic joints have the lowest weight, one is selected at random, and placed in the tree. In this case, let us suppose that the prismatic joint s_{25} was placed in the tree. Then, the next two edges to be added to the tree will automatically be the two arms of the prismatic joint's distal body, in this case r_{10} and r_{11}. The reason for this is that, since they both have the lowest possible weight, they are selected before any of the other potential edges against which they compete.

The tree selected at the present point in time is depicted in Figure 4.20(b), where the branches are depicted in bold, the nodes found in the tree are depicted in black, the potential edges are depicted as dashed edges, the bold numbers are the potential edges weights and the italic numbers are the branches' $CutsetBodies$ characteristic values (these are not shown for arms since arms always have a $NumEQs$ value of zero and, as such, their $CutsetBodies$ value are irrelevant).

Since the present branches do not generate any dynamic equations (all their $NumEQs$ value are zero) they will not affect the weights of the present potential edges. Because of this, the two remaining prismatic joints still have a weight of zero and the body edges still

have a weight of 27. The spherical joint b_{26}, which is a potential edge at this point in the tree selection, will have a weight of 30. This weight is due to the three body torques of the body m_2, worth a weight of ten, that would have to be present in each of the three dynamic equations generated by this joint if this joint was added to the tree. Also, similarly to the body edges, the three dynamic equations required to model the system if this spherical joint was placed in the tree would not generate any kinematic equations at the present time because there would not be any more dynamic equations needed than the three DOF of the system.

Since the two prismatic joints still have the lowest weight, one of these two will randomly be selected and placed in the tree. This will be followed by this prismatic joint's distal body's arm elements being the two next edges to be placed in the tree. This same process will ensure that the last prismatic joint as well as its distal body's arms are the next three edges placed in the tree.

At this point, the potential edges are the three spherical joints b_{26}, b_{29}, and b_{32}, as well as the body edges m_2, m_4, m_6, and m_7. Since the present tree still does not generate any dynamic equations, the weights of the spherical joints are again set at 30 and the weight of the body edges are also still set at 27. One of the body edges will then be selected at random and placed in the tree since they all have the lowest weight of the potential edges. Let us suppose that this will be the edge m_7 (if one of the three other body edges would be selected instead, the final results would be the same). As before, the next edges in the tree will be the edges r_{22} to r_{24} since they are arms, and arms have the lowest possible weight. The present tree is depicted in Figure 4.20(c). This figure uses the same conventions as used previously in 4.20(b).

Now, the weighting of the potential edges are not as simple, since some of the branches will affect these weights and all three DOF have now been accounted for by the tree dynamic equations of m_7. Let us start by looking at the weight of the potential edge m_2. The calculations made by the Dynamic Weighting Function to find the weight of this edge are summarized in equation (4.15). In this equation each edge's $NumEQs$ characteristic value is depicted in bold and the edge's $CutsetBodies$ characteristic value is depicted in italics where the edge to which these values belong is identified underneath these values. The term *kin* refers to the fact that the numbers placed above this term are due to the

Kinematic Equation Heuristic, with the first term being the number of kinematic equations and the second term being the $KinEQComplexity$ that is valued at 3 in the rotational domain. The value of the $CutsetBodies$ characteristic that is shown in this equation is the one obtained after the Dynamic Weighting Function makes the appropriate modification to chain branch edges relating the potential edge to the ground.

$$\text{Edge } m_2\text{'s Weight: } \Big(\underbrace{\mathbf{3} \times 9}_{m_2} + \underbrace{\mathbf{3} \times 9}_{m_7} \Big) + \underbrace{3 \times 3}_{kin} = 63 \tag{4.15}$$

$$\text{Edge } b_{26}\text{'s Weight: } \Big(\underbrace{\mathbf{3} \times 10}_{b_{26}} + \underbrace{\mathbf{3} \times 9}_{m_7} \Big) + \underbrace{3 \times 3}_{kin} = 66 \tag{4.16}$$

$$\text{Edge } h_{27}\text{'s Weight: } \Big(\underbrace{\mathbf{1} \times 10}_{h_{27}} + \underbrace{\mathbf{3} \times 19}_{m_7} \Big) + \underbrace{1 \times 3}_{kin} = 70 \tag{4.17}$$

Looking at this equation we see that the edge m_2 adds a value of 27 to its own weight due to the presence of its own body torque, having a weight of nine, in all three of its dynamic equations. Since the weight associated to each potential edge represents the total weight of the graph's tree, if the given potential edge was added to the tree, the branch m_7 also affects m_2's weight. This edge provides an extra weight of 27 due to the presence of its torque, also having a weight of nine, in its three dynamic equations. The other branches also participate in finding m_2's weight; however they are not shown here since they all have a $NumEQs$ value of zero. Finally, three kinematic equations would be used to model the system if m_2 was added to the tree, since the six dynamic equations it would require are superior to the three suspected rotational DOF. These three kinematic equations each add a weight of three, which gives a total weight of 63 to edge m_2. By applying the same logic that was presented for the edge m_2 to all the other body edges found in the potential edges, they can also be shown to each have a weight of 63.

If we look at the weight assigned to the potential spherical edge b_{26}, shown in equation (4.16), we can see that it adds 30 to its own weight. This is due to the unavoidable presence of the body m_2 in all three of potential edge b_{26}'s three dynamic equations. In this case, the edge m_2 is considered to have a weight of ten because if the edge b_{26} is added to the tree, the arms r_{12} and r_{13} will inevitably be added afterward, automatically making m_2 a

chord in this case. Similar to the weight assigned to the potential edge m_2, the branch m_7 adds a weight of 27 to the potential edge b_{26}'s weight and the three kinematic equations generated by the addition of this potential edge to the tree each add a weight of three for a grand total of 66. All other spherical joints will have the same weight.

Now we can look at the formulation of edge h_{27}'s weight whose summary can be seen in equation (4.17). In this case, the edge h_{27} will add a weight of ten to its own weight due to the unavoidable presence of m_2's torque, worth a weight of ten, in h_{27}'s dynamic equation. Additionally, since the branch path relating the potential edge h_{27} to the ground includes the branch m_7, m_2's torque will also be unavoidably found in all three of the m_7's dynamic equations, in addition to this branch's own torque, which is worth a weight of nine. This is why m_7's $CutsetBodies$ value is now worth 19 and why this branch will add a weight of 57 to the potential edge h_{27}'s weight. The addition of h_{27} to the tree would also require the formulation of one kinematic equation, which adds an additional weight of three to this potential edge's weight. This results in a weight of 70 being assigned to this, and all other, revolute joints in the potential edges.

This show's that the edges m_2, m_4, and m_6 all have the lowest potential edge weight. One of these will then be randomly picked and added to the tree. For the sake of this example let us suppose that the edge m_2 is the potential edge selected to be added to the tree at this time. As with all the other joints added to the tree, the arms r_{12} and r_{13} will be the two next edges to be added in the tree. At this point the edges m_4, m_6, b_{29}, b_{32}, h_{30}, and h_{33} will form the set of potential edges.

Continuing this tree selection process, the algorithm continues to add edges in the tree until the valid rotational tree presented in Figure 4.20(d) is found.

At this point in time, it is possible to find the complexity factor called $KinEQ-Complexity$ that shall be used to predict the relative complexity of the translational kinematic equations relative to the translational dynamic equations. According to Section 4.2.4, this factor is estimated using the number of possible rotations found in the most complex (the one with the most rotations) f-circuit of the rotational tree. In this case, all three revolute joint's f-circuit contains six rotations, three from the m_7 edge and three from one of the body edges. This results in the $KinEQComplexity$ variable to be equal

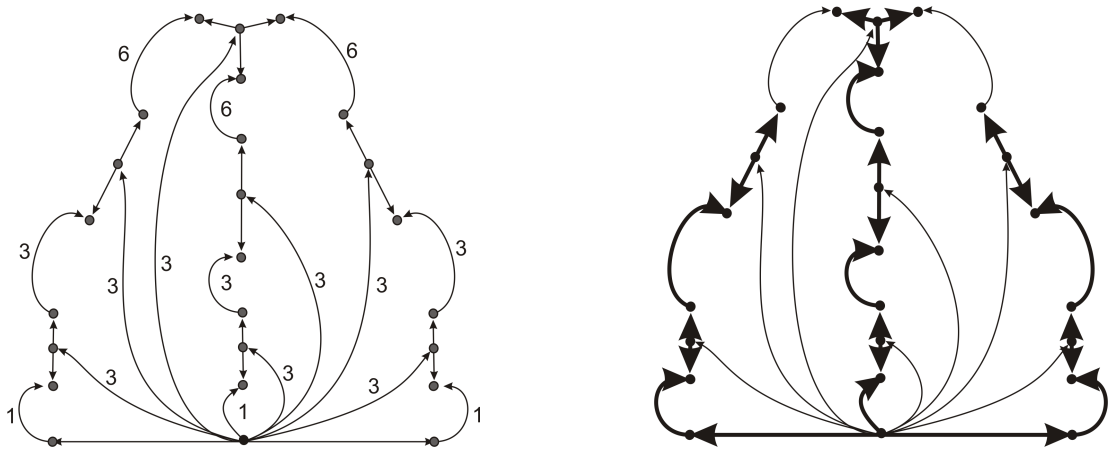
to 60.

Before starting the selection of the translational tree one must find the number of times the force of each arm and each joint allowing translation is found in the basic rotational dynamic equation. This value is calculated and assigned to their $NumInRotEQs$ characteristic via the Update Function of Figure 4.15. The basic concept is that the number of these appearances is going to be equal to the sum of the number of dynamic equations generated by the branches found in the branch path between the given arm or translational joint and the ground node. The result of this calculation is depicted in Figure 4.20(d) by the bold numbers found next to the arm edges. For example, the forces of the arms r_{12} and r_{13} are found in each of the three dynamic equations generated by the body edge m_2. This is why these arms have a $NumInRotEQs$ characteristic of three.

At this point in time the selection of the translational tree starts by assigning the $NumEQs$ characteristic of every edge, except the arm edges for which this characteristic is always zero. This is done by simply adding the number of translational across variables, found in Appendix C, with the $NumInRotEQs$ of each arm to which the given edge is connected. The only exception to this are branch body edges, whose own torque is never found in its own arm's f-cutset equations. This is represented in Figure 4.21(a) where the $NumEQs$ of arms, which must be zero, are not shown in order to improve the clarity of the figure.

This graph is then submitted to the Formulation-Based Tree Selection Algorithm that selects the translational tree using the same steps as the ones used to select the rotational tree. When selecting the translational tree, the Tree Similarity Heuristic can be enforced. Once this process is completed the translational tree obtained by the Formulation-Based Tree Selection Algorithm is shown in Figure 4.21(b).

Once the two trees have been selected, it is necessary to verify if the rotational tree also adequately enforces the Tree Similarity Heuristic. In order to verify this, the rotational tree is selected once again. In this case, the resulting new rotational tree is the same rotational tree as before. Since the loop ensuring the Tree Similarity Heuristic has already converged to a solution, the final trees of the mechanism are set to the rotational tree of figure 4.20(d) and the translational tree of figure 4.21(b).



(a) $NumEQs$ characteristic values (b) translational Tree

Figure 4.21: 3-DOF spatial parallel manipulator's $NumEQs$ in translation and final translational tree.

These trees will model the system using the coordinate set ζ_2, η_2, ξ_2, ζ_4, η_4, ξ_4,ζ_6, η_6, ξ_6, ζ_7, η_7, ξ_7, S_{25}, S_{28}, and S_{31}, which represent the rotations of the body edges m_2, m_4, m_6, and m_7, as well as the displacements of the prismatic joints s_{25}, s_{28}, and s_{31}, respectfully. This will generate a model with 15 dynamic equations and 12 kinematic equations.

Using ModelBuilder, this 3-DOF spatial parallel manipulator was modelled and the Formulation-Based Tree Selection Algorithm was executed finding the trees described previously in an average of 63 milliseconds. Once the system's equation were developed within DynaFlexPro, the forces depicted in Table 4.5 were placed on the prismatic joints. The system equations obtained formed a set of DAEs that were changed to a set of ODEs by twice differentiating the kinematic equations of the system. This set of ODEs was simplified and code optimization routines where performed within Maple. The system was then solved for a one second simulation using a Runge-Kutta Fehlberg method within the Maple 10 software. The solution times of these ODEs are based on a ten trial average and are presented in Table 4.6. A Pentium 4 of 1.80 GHz with 768 MB of RAM was used to perform these simulations.

The same system was also modelled using the exact procedure described above but using

Table 4.5: Forces at the 3-DOF Spatial Parallel Manipulator's Prismatic Joints

Prismatic Joints	s_{25}	s_{28}	s_{31}
Forces (N)	$65 + 5.0\cos(2\pi t)$	$65 + 5.0\sin(2\pi t)$	$65 + 3.0\sin(4\pi t)$

different tree sets. The solution time as well as the equation complexity factors obtained from the optimized code of the equations for each tree set are presented in Table 4.6, where the tree set selected by the Formulation-Based Tree Selection Algorithm is shown in bold and the tree set selected by the Variable-Based Tree Selection Algorithm is shown in italics. In the description of the various trees, the arm elements were not presented; however they are always found in both trees. The notation *Same Tree* for the translational tree indicates that the translational tree is composed of the same edges as the rotational tree. The term *Comp.* denotes the equation complexity that is described in Section 2.4. The notation "–" for the solution time and complexity of a mechanism model indicates that DynaFlexPro could not generate the equations of the system within 30 minutes when the system was modelled using the given set of trees. As discussed in the example presented in Section 3.3.2, the solution times presented in this table are slower than real-time due to the use of Maple as a numerical solver. However, this increased solution time does not affect the relative time difference between the simulations of models using various coordinate sets.

The results presented in Table 4.6 are very revealing. They confirm the link between the model's equation complexity and solution time. From these results, it is clear that the coordinate set selected by the Formulation-Based Tree Selection Algorithm provides the most efficient simulation of all those taken into consideration.

As mentioned in previous chapters, many researchers [12, 16, 19, 31] have suggested that the model having the fewest number of equations would produce the most efficient simulations. The fewest number of modelling variables that can be used, given the limitation of having to select valid trees, are the coordinates ζ_7, η_7, ξ_7, θ_{27}, θ_{30}, θ_{33}, and S_{25}. Comparing this model to the Coordinates selected using the Formulation-Based Tree Selection Algorithm, we see that the latter are approximately 62% more efficient than the

Table 4.6: Efficiency of Various Tree Sets of the 3-DOF Spatial Parallel Manipulator

Rotational Tree	Translational Tree	Sol. Time (s)	Comp.	# Dyn. EQs	# Kin. EQs
$s_{25}, s_{28}, s_{31}, b_{26}, b_{29}, b_{32}, m_7$	$s_{25}, b_{26}, b_{29}, b_{32}, h_{27}, h_{30}, h_{33}$	3.98	48178	13	10
$s_{25}, s_{28}, s_{31}, b_{26}, b_{29}, b_{32}, m_7$	$s_{25}, s_{28}, s_{31}, b_{26}, b_{29}, b_{32}, h_{27}$	2.55	30013	15	12
$s_{25}, s_{28}, s_{31}, b_{26}, b_{29}, b_{32}, m_7$	Same Tree	2.54	31213	18	15
$s_{25}, s_{28}, s_{31}, b_{26}, b_{29}, b_{32}, m_7$	$s_{25}, b_{29}, b_{32}, h_{27}, h_{30}, h_{33}, m_7$	4.32	51346	16	13
$b_{26}, b_{29}, b_{32}, h_{27}, h_{30}, h_{33}, m_7$	$s_{25}, s_{28}, s_{31}, b_{26}, b_{29}, b_{32}, m_7$	–	–	21	18
$s_{25}, s_{28}, s_{31}, b_{26}, b_{29}, b_{32}, h_{27}$	$s_{25}, s_{28}, s_{31}, b_{26}, b_{29}, b_{32}, m_7$	3.73	41233	16	13
$s_{25}, s_{28}, s_{31}, h_{27}, h_{30}, h_{33}, m_7$	*$s_{25}, b_{26}, b_{29}, b_{32}, h_{27}, h_{30}, h_{33}$*	*6.14*	*66060*	*7*	*4*
$s_{25}, s_{28}, s_{31}, h_{27}, h_{30}, h_{33}, m_7$	$s_{25}, s_{28}, s_{31}, b_{26}, b_{29}, b_{32}, h_{27}$	4.11	43203	9	6
$s_{25}, s_{28}, s_{31}, h_{27}, h_{30}, h_{33}, m_7$	$s_{25}, s_{28}, s_{31}, b_{26}, b_{29}, b_{32}, m_7$	4.11	43356	12	9
$s_{25}, s_{28}, s_{31}, h_{27}, h_{30}, h_{33}, m_7$	Same Tree	6.02	68210	12	9
$s_{25}, s_{28}, s_{31}, h_{27}, m_2, m_4, m_6$	$s_{25}, s_{28}, s_{31}, m_2, m_4, m_6, m_7$	4.35	50181	22	19
$s_{25}, s_{28}, s_{31}, m_2, m_4, m_6, m_7$	**$s_{25}, s_{28}, s_{31}, b_{26}, b_{29}, b_{32}, m_7$**	**2.34**	**26595**	**15**	**12**

coordinates using the fewest number of equations.

The effects of the properties of the rotational tree on the selection of the translational tree is also quite apparent in this example. Had the translational tree only been selected according to the chord body forces present in its own dynamic equations, the translational tree could have selected only one of the prismatic joints in the tree rather than all three. As shown by the results in Table 4.6, this would have led to a tree selection that produces slower simulation time.

This example also shows the use of the slightly lower weight of branch body torques relative to chord body torques. Had these weights been the same, the spherical joints and body edges would have had the same weights throughout the rotational tree selection of this specific system. Because of this, the Tree Similarity Heuristic would have made sure the spherical joints were placed in the rotational tree rather than most of the body edges since these spherical joints are found in the translational tree. This would have resulted in a slightly less efficient coordinate set to be selected by the algorithm (this tree set is found in Table 4.6's third row and its simulation time is 9% slower than the optimal solution).

Finally, another interesting observation that one can make while looking at the results found in the Table 4.6, is that the coordinates found by the Formulation-Based Tree Selection Algorithm have very unintuitive characteristics. It uses 15 dynamic equations and 12 kinematic equations to model a 3-DOF system. These coordinates are not the standard coordinate sets in common use, such as Absolute Coordinates, Joint Coordinates or Absolute Angular Coordinates. It would have been extremely difficult, even for someone experienced in selecting appropriate modelling coordinate for mechanical systems, to choose to model the present systems with these coordinates.

This clearly shows the complexity of selecting an optimal coordinate set. This complexity is shown to be thoroughly taken into account by the Formulation-Based Tree Selection Algorithm that selects the most efficient set of coordinates for an extremely complex mechanical system. Furthermore, this example demonstrates the advantages of using an automated coordinate algorithm to select appropriate coordinates for a system.

Finally, further proof of this algorithm's validity is found in the benchmark problems depicted in the Appendix A. A brief summery of these results are presented in Table 4.7,where the column designated by *Optimal Tree Set* indicates whether or not the tree

set providing the best simulation times of all the trees tested in the context of this report was found for the given mechanism using the Formulation-Based Tree Selection Algorithm. The column designated by *% Longer Sim. Time* indicates, in percentage, how much longer the simulation time of the model using the tree set found by the Formulation-Based Tree Selection Algorithm is compared to the simulation time of the optimal tree set. When comparing these results to those obtained for the Variable-Based Tree Selection Algorithm (Table 3.3), one can note the added efficiency of the tree sets selected by the Formulation-Based Tree Selection Algorithm, especially for complex spatial mechanisms such as the Stewart-Gough Platform and the 3-DOF Spatial Parallel Manipulator.

Table 4.7: Results of the Formulation-Based Tree Selection for Various Mechanisms

Mechanism	Appendix	Optimal Tree Set	% Longer Sim. Time
Planar 4-Bar Mechanism	A.1	yes	–
Planar 5-Bar Mechanism	A.2	yes	–
Three Bodies Attached with Two Revolute Joints	A.3	no	145
Planar Slider-Crank	A.4	no	48
Planar Flexible Slider-Crank	A.5	no	38
Spatial Slider-Crank	A.6	yes	–
Planar 3-RRR Parallel Manipulator	A.7	yes	–
Planar 3-RPR Parallel Manipulator	A.8	no	44
Peaucellier-Lipkin Straight-Line Mechanism	A.9	yes	–
Spatial Serial Manipulator	A.10	no	18
Stewart-Gough Platform	A.11	no	28
3-DOF Spatial Parallel Manipulator	A.12	yes	–

4.4 Algorithm Shortcomings

The tree selection algorithm described in this chapter provides very good results in most circumstances. This can be clearly seen in the results obtained for the benchmark problems presented in Appendix A. Most importantly, it always finds a valid tree. However, it must be said that this algorithm is not perfect and in some circumstances does not select the optimal trees of the system. These cases are few in number and the trees selected by the algorithm are always valid trees that produce models that have simulation times that are generally fairly close to the fastest simulation times found with the tree sets tested in this report. The major reasons for the present algorithm's deficiencies are presented in this section.

Kinematic Equation Complexity Approximations

In both Section 4.1, where the Tree Selection Heuristics were established, as well as Section 4.2, where the tree selection algorithm was established, one shortcoming is clearly apparent in many occasions. This shortcoming is the lack of information concerning kinematic equations of the system. In the present tree selection process, the number of these equations is predicted using a DOF calculation. These methods are known to be inaccurate in certain situations. Most importantly, the prediction of the kinematic equation's complexity relative to the dynamic equations as well as the relative complexity between the kinematic equations themselves is not very precise. In the present algorithm, this complexity is based more on trial and error than on a direct observation of the composition as well as the formulation process of these equations. The reasons for these problems are mentioned repeatedly in both Section 4.1.4 and Section 4.2.4 and present a major challenge for the future improvement of the present algorithm.

In general, the methods used in the present tree selection algorithm seem to be able to predict the complexity of kinematic equations to an acceptable accuracy and it also seems to use this information effectively during the tree selection. However, it should be noted that the tree selection for planar mechanisms are most vulnerable to the problem associated with the imperfect prediction of kinematic equation complexity.

The bodies of planar mechanisms all rotate about one axis. Body rotation is a major

factor in a system's symbolic equation complexity. In addition, the model will contain much more symmetry and its translation will be limited to two dimensions. This will result in the use of a much simpler set of equations to model the system. This set of equations will be much easier to simplify and optimize. This added simplicity affects the dynamic and kinematic equations differently. However, the complexity of the kinematic equations relative to the dynamic equations (estimated by the $KinEQComplexity$ variable in the tree selection algorithm) was mostly estimated using trial and error and was intended for a spatial system. This relative complexity is often erroneous for more simple models such as planar systems. This proves especially crucial in the case of the translational tree, where the complexities of the dynamic and kinematic equations generated by this tree are of a much more similar nature. This means that their estimated relative complexity has much more impact on the tree selection process. Furthermore, the added symmetry in planar mechanical systems makes the DOF calculation of these mechanisms more difficult.

This problem in the tree selection algorithm is demonstrated in the benchmark problems presented in Appendix A.4 and A.8. Appendix A.4 shows the results of the slider-crank for various tree selections. In this case, the Formulation-Based Tree Selection Algorithm selects the prismatic joint in the translational tree when this should be avoided. This results in approximately a 72% increase in solution time relative to the known optimal tree.

Appendix A.8 depicts a planar 3-RPR mechanism. When using the Formulation-Based Tree Selection Algorithm for this mechanism, the three prismatic joints are not included in the modelling coordinates. Keeping the same coordinates and adding the three prismatic joints decreases the model's solution time by approximately 27%.

However, spatial mechanisms are not immune to this problem. For example, if we look at the spatial serial manipulator presented in Appendix A.10, we see that the body edge of the last body in the chain should be placed in the translational tree thus improving the solution time by 15%. This was not done by the Formulation-Based Tree Selection Algorithm. Upon closer inspection, it is possible to notice that, had the $KinEQComplexity$ variable been evaluated one unit lower, the optimal translational tree would have been found in this case.

Another problem also arises due to the method used to estimate the number of kine-

matic equations for the system at every stage of the tree's completion. Since the rotational tree is always selected first, all the dynamic equations that are considered as not adding any kinematic equations will have the tendency of being found mostly in the rotational tree. For example, if a system had five degrees of freedom, the first five dynamic equations generated by the rotational tree would not have any weight added by the Kinematic Equation Heuristic. In this case, all five of these DOFs could be "used" in the rotational tree and all the dynamic equations generated by the translational tree would have an extra weight added to them to compensate for the Kinematic Equation Heuristic.

This shows that it would be preferable to predict the number of DOF in rotation and translation separately. However, when attempts were made to modify Gruebler's equation for this purpose, the total number of DOF calculated was even more inaccurate than when the equation was used in its global form. For example, the rotational DOF of the spatial parallel manipulator is estimated at six using separate Grubler DOF calculations for rotation and translation. In reality, the mechanism only has a total of three DOF encompassing both rotation and translation. Because of this, the general form of Gruebler's equation was kept.

There are a few different methods one could consider to improve these problems. The first such method would be to devise a better method of estimating the number and complexity of the kinematic equations by either adding tree selection heuristics or devising a new tree selection algorithm all together.

When looking at the DOF estimation problem, another possible solution could be to simply implement another more accurate DOF calculation method in the tree selection algorithm. The other problem of finding a better approximation of the relative complexity between dynamic and kinematic equations could be partially solved by first asking the user if the model he provided is planar or not. Then one could apply a different, more appropriate, complexity approximations method in each case. This method could be further improved by creating an automated method of determining if a given mechanism is planar or not. However, this would require the extraction of extensive geometry information to the pre-processing environment.

Kinematic Equation Complexity Approximations

Another problem that the Formulation-Based Tree Selection Algorithm faced during its conception was the difficulty in predicting the results of simplification and code optimization procedures performed on the equations used to model the system. These procedures simplify the system's equations and reformulate the equations so that repeated variables and equation sections only need to be calculated once. The limited information available in the pre-processing environment in which the tree selection algorithm is elaborated makes it extremely difficult to predict the effect that these procedures will have on the system's equations. In fact, out of all the heuristics developed for tree selection, only the Tree Similarity Heuristic, which lowers the coupling between the equations, was elaborated specifically to take advantage of the effect of the simplification and code optimization procedures.

Unfortunately, the equation simplification and optimisation process have much greater effects on the final system model than can be predicted by simply using the Tree Similarity Heuristic. These effects seem to show themselves more in systems with many degrees of freedom. For example, Appendix A.3 depicts the modelling of three bodies connected together by two revolute joints and floating freely relative to the ground. In this case, the model using the trees selected by the Formulation-Based Tree Selection Algorithm is approximately 145% less efficient than the optimal tree set.

This clearly indicate that further efforts need to be made in order to better predict the effect of simplification and code optimization on the system model. For example, perhaps one could try taking into account the number of times each body's torque and forces are repeated in the formulation of the system equations. Certain factors could be used to take into account the code optimization effect on systems that use the same torques and forces multiple times in their equation formulation.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

A large percentage of the studies of optimal coordinate set selection have been done using a graph-theoretical approach. This approach gives insight on the mechanism's topology, which proves to be important when trying to determine efficient coordinate sets. When this graph-theoretical approach is used to select a model's coordinate set, valid trees from the system's graph must be selected. Furthermore, there is one coordinate selection criterion that is often used in the literature. It states that the coordinate set using the fewest possible number of modelling variables will produce the models having most efficient simulations. This criterion is based on the assumption that the more modelling variables are used to model the system, the greater the number of equations are necessary to model the system. Since all of these equations must be solved simultaneously to model the system, it is assumed that a lower number of equations will increase the model's simulation efficiency. Though, as will be discussed later, this assumption was later shown to not always be valid for every mechanical system.

A graph-theoretical approach was also used in this report to select coordinate sets. In addition to being a helpful tool for coordinate selection, graph theory can also be used to automatically formulate a system's symbolic equations. This technique was also used in this research to facilitate the analysis of various benchmark problems, and to compare the effect of multiple coordinate sets to model these benchmark problems. Many methods

of formulating a mechanical system's symbolic equations using graph theory have been developed. McPhee's method [19] was used in this report due to its capability of modelling a large number of coordinate sets such as Absolute Coordinates, Joint Coordinates, Absolute Angular Coordinates, Indirect Coordinates, as well as any combination of these coordinate sets.

The first tree selection algorithm developed in this report, called the Variable-Based Tree Selection, was based on modelling systems with coordinate sets containing a minimal number of variables. This was done by assigning weights to each graph edge and finding the minimal spanning trees, a technique that was first proposed as a coordinate selection method by Kim and Vanderploeg [12]. This method was augmented by the addition of extra tree selection heuristics that made it possible to properly select between multiple trees that would model mechanical systems with the same number of modelling variables. These heuristics included the necessity of using certain edge types as the tree edges or as cotree edges, the minimization of the number of modelling variables relating every edge to the ground, the minimization of the number of joints relating every edge to the ground, as well as selecting the rotational and translational trees so that they are as similar as possible to each other. However, when some of the benchmark problems were tested using various coordinate sets to validate the Variable-Based Tree Selection, it was discovered that using the minimal set of modelling coordinates does not always produce the most efficient results. Since this is the main heuristic used to select trees in the Variable-Based Tree Selection, its performance is not always optimal. When trying to modify this tree selection method to properly account for the new coordinate selection observations, it became clear that simple benchmark observations and model topology were insufficient to establish better tree selection heuristics.

A second tree selection algorithm, called Formulation-Based Tree Selection, was developed. This method's heuristics were mostly developed by carefully analysing McPhee's [19] symbolic equation formulation procedure. This provided great insight on how the mechanical system's symbolic equations were created and structured and provided explanation as to why certain systems could be simulated more efficiently when modelled using a higher than minimal set of modelling coordinates. Some of these heuristics included minimizing the number of body torques and chord body forces found in the system's cutsets equations

and, whenever possible, minimizing the number of kinematic equation used to model the system. The Formulation-Based Tree Selection also reused and improved some heuristics originally developed for the Variable-Based Tree Selection. These heuristics include the necessity of using certain edge types as the tree edges or as cotree edges and the necessity of selecting the translational and rotational trees to be as similar as possible. The Formulation-Based Tree Selection also included heuristics especially developed to take into account the effect of flexible bodies on the system's symbolic equations by placing the flexible arm edges as leaf edges in the mechanism's trees.

A Formulation-Based Tree Selection Algorithm was then developed to implement the heuristics described above. This algorithm was also based on graph theory and modified Prim's minimal spanning tree selection algorithm [24] so that it could assign edge weights dynamically during the tree selection process. This algorithm was tested with all of the benchmark problems. By comparing the trees obtained by the algorithm to other trees or to known optimal trees published in the literature, the algorithm's efficiency and robustness was shown. However, the algorithm was not without its shortcomings. For example, the number and complexity of the kinematic equations used to model a system proved to be quite difficult to predict within the Formulation-Based Tree Selection Algorithm. Furthermore, the equation simplification and code optimization algorithms, which can be applied to a system's symbolic equation after their formulation, are also difficult to predict. These problems result in the Formulation-Based Tree Selection Algorithm's selection of less than optimal trees for certain mechanisms, most notably for planar systems and systems with a high number of degrees of freedom.

Both tree selection algorithms developed in this report provide good tree selection results, especially the Formulation-Based Tree Selection Algorithm, which is capable of selecting optimal or nearly optimal trees for most mechanical systems. It was also shown that coordinate sets having a minimal number of modelling variables do not always provide the models with the most efficient simulations, which was a premise held by many researchers working on optimal coordinate selection. Furthermore, when formulating this report's two tree selection techniques, many coordinate selection heuristics were devised that provide a significant addition to the coordinate selection heuristics developed in the literature thus far.

5.2 Future Work

The literature exploring the topic of coordinate selection generally focuses on the comparison of only a few specific coordinate sets to one another. Because of this, few general coordinate selection criteria have been established. Though this work added significant improvements in this field by proposing two coordinate selection algorithms, there still exists a large number of possible enhancements to these algorithms as well as the field of optimal coordinate selection in general. This section will propose possible topics for future work in this field of research.

Heuristics Based on Other Symbolic Equation Formulation Procedures

The Formulation-Based Tree Selection Algorithm was developed based on the analysis of McPhee's [19] symbolic equation formulation procedure. This method provided much insight on the structure of a mechanical system's symbolic equations and demonstrated a few methods that could be used to reduce the complexity of these equations. Given the excellent results obtained when developing tree selection heuristics based on this method, it seems logical to look into the heuristics that could be derived from other formulation procedures. For example, Shi and McPhee [26] developed a formulation procedure based on analytical mechanics (i.e. virtual work). It would be interesting to see the heuristics that could be derived from this significantly different approach to symbolic equation formulation and see how these new heuristics could be combined with the ones already developed in this report.

Algorithm for Inverse Dynamics and Kinematics Problems

The tree selection algorithms developed in this report were based on lowering the simulation time for forward dynamic simulations. Forward dynamic simulations were chosen as the basis for this analysis since they are the most complex and time-consuming type of analysis that can be performed on mechanical systems. This analysis requires simultaneously solving all of the model's symbolic equations, which form a set of DAEs when kinematic equations are needed.

The present tree selection algorithms could be modified, or new algorithms could be

developed, to select trees that produce models that provide efficient simulations for other types of analyses such as inverse dynamic simulation, kinematic simulations, and static simulations. These models could require different approaches than those developed in this report. For example, a major part of the tree selection methods in this report focuses on reducing the complexity of dynamic equations and the reduction of the complexity of kinematic equations is considered secondary; however, kinematic simulation only requires the solution of the kinematic equations.

Geometry

During the elaboration of the Formulation-Based Tree Selection Algorithm, one of the major problems was the limited model information that was extracted from the pre-processing environment in which the tree selection algorithm was developed. For example, the system's geometry was not extracted from the model due to the limited usefulness of this information in the pre-processing environment that lacked the ability of performing symbolic manipulations. Because of this, the heuristics developed for tree selection were quite limited. Substantial improvements of these heuristics could hence be made if the system's geometry was extracted and analyzed using symbolic manipulations.

For example, when using only topology, one can know that a series of bodies are connected together by revolute joints; however, since the system's geometry is unknown, it is impossible to know if all the revolute joint's axis of rotation are parallel or not. This is very important since, if the axis are all parallel, one could determine if the mechanism is planar, in which case its properties would be quite different than if the mechanism were spatial in nature.

Automate the Addition of Extra Virtual Joints

The use of Indirect Coordinates to model a system has been shown to produce models that result in very efficient simulations in certain situations. This type of coordinate sets can model the motion of any frame in a system relative to any other frame in the system. Because of this, certain extra virtual joints must be added to the system in order to measure these motions. Both the algorithms presented in this report take full advantage of virtual joints if they are present in a system model. These algorithms are capable of using the

most appropriate virtual joints to find modelling coordinates. However, at the present time, these virtual joints must be manually added to the system's graph by the user.

For example, if we look at both examples presented in Section 3.3, a virtual planar joint was added between two revolute joints in the spatial serial manipulator of Section 3.3.1 and virtual planar joints were added between the ground node and every one of the system's bodies in the 3-RRR planar parallel manipulator presented in Section 3.3.2. In both of these cases, the added virtual joint substantially improved the models' simulation efficiency.

Fayet and Pfister [5], who first proposed this coordinate set, proposed a few rules as to where certain of these virtual joints can be added to a system in order to increase the model's simulation efficiency. These rules are limited to prismatic joints, revolute joints, and spherical joints. One could use these rules to automate the addition of virtual joints in a system's graph. This could even be taken one step further by using Fayet's rules as a basis to establish rules for other joint types such as universal joints and planar joints.

However, to properly enforce these rules, the mechanical system's geometry needs to be known. For example, Absolute Angular Coordinates generally produce models that produce very efficient simulations of planar systems. In order to model a system with these coordinates, virtual planar joints relating the ground node to every body must be added to the system, as shown in Section 3.3.2. Unfortunately, if a model's geometry is unknown, it is impossible to know with certainty that the system is planar and automatically add the needed virtual joints to the system.

Optimization Methods

As discussed in Section 2.4, fully formulating a system's symbolic equation within the DynaFlexPro Maple Package can sometimes take a large amount of time for systems using certain trees. Since each mechanical system has a large number of possible trees and that one of the tree selection criteria of this report is to develop quick and efficient algorithms, it was decided not to use optimization methods on fully-formulated models to find a system's optimal trees. Instead, the tree selection algorithms were developed in a pre-processing environment where the models did not need to be fully-formulated in order to estimate its performance.

Both of the algorithms presented in this report are capable of selecting relatively efficient trees in most circumstances, but none are capable of selecting a system's optimal trees for every possible mechanical system. These algorithm deficiencies are due in large part to the many factors affecting equation complexity that simply cannot be predicted within a pre-processing environment, such as equation simplification and code optimization.

Future work on optimal tree selection could try to combine the speed of selecting a tree within a pre-processing environment and the accuracy that could be obtained when using an optimization method to find the optimal fully-formulated model. For example, a tree selection algorithm developed in a pre-processing environment, similar to those developed in this report, could be used to narrow down a system's possible optimal trees to a small number that could then be sent to an optimization algorithm that formulates each model it evaluates. This process would surely be much slower than the algorithms presented in this report; however the improvement in the results could be advantageous to those for whom simulation efficiency is of extreme importance.

In-Depth Component Analysis and New Components

In this thesis, the addition of SDAs in the system's trees has been shown to have a negative effect on simulation time. However, none of the benchmarks tested included any SDAs. In order to verify that the theoretical conclusions reached in this report are valid, further benchmark problems containing SDAs should be tested in the future.

Furthermore, vehicles are often modelled using multibody dynamics software. When modelling vehicles, special attention must be taken to properly model its tires. Tire Components have recently been added to a symbolic equation formulation procedure of multibody systems using graph theory [21]. Tire components have also been added to the DynaFlex-Pro package.

The effect of tires on the symbolic equation complexity and solution times of vehicle models was not treated in this report. Further research could be made in this field in order to properly include this new modelling component in the present or future tree selection algorithms.

Different Basis for the Tree Selection Algorithms

Prim's algorithm was used as a basis for both tree selection algorithms presented in this report. Prim's algorithm was used due to its simplicity, modifiability and performance, as well as a few more specific characteristics discussed in Section 4.2.1. Many other tree selection algorithms exist. In the future, other tree selection algorithms could be explored. For example, certain tree selection algorithms are capable of predicting the effect that adding a certain branch can have on the selection of future branches. In other words, these algorithms are capable of looking many "layers" ahead. Also, other tree selection algorithms can be faster than Prim's algorithm.

Velocity and Acceleration Trees

The symbolic equation formulation procedure used in this report uses a system's position level trees to develop the position, velocity, and acceleration level equations of the system. Distinct velocity level trees and acceleration level trees could also be used to formulate a system's symbolic equations. Researchers have shown the increased efficiency that can be obtained by using velocity transformations when formulating a systems symbolic equations.

In the future, one could develop new heuristics or tree selection algorithms for the velocity and acceleration level trees. Since DynaFlexPro already provides the possibility of selecting separate velocity and acceleration level trees, this research could easily be done using this software package.

Electrical and Hydraulic Tree Selection

One of the major advantages of using graph theory is that it is capable of modelling multi-domain systems. For example, one could model a mechanical system with electrical and hydraulic components using one graph-theoretical model of the system. The present tree selection algorithms could be extended to include electrical and hydraulic components, thus making it capable of selecting optimal coordinates in a multi-domain system.

References

[1] G. Andrews and H.K. Kesavan. The vector-network model: A new approach to vector dynamics. *Mechanism and Machine Theory*, vol.10:pp.5775, 1975.

[2] J. Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Computer Methods in Applied Mechanics and Engineering*, vol.1:pp.1–16, 1972.

[3] J. García de Jalón and E. Bayo. *Kinematic and Dynamic Simulation of Multibody Systems: The Real-Time Challenge*. Springer-Verlag, 1994.

[4] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, vol.1:pp.269–271, 1959.

[5] M. Fayet and F. Pfister. Analysis of multibody systems with indirect coordinates and global inertia tensors. *European Journal of Mechanics, A/Solids*, vol.13(no.3):pp.431–457, 1994.

[6] T. Geike and J. McPhee. Inverse dynamic analysis of parallel robots with full mobility. *Mechanism and Machine Theory*, vol.38(no.6):pp.549–562, 2003.

[7] R.L. Graham and P. Hell. On the history of the minimum spanning tree problem. *Annals of the History of Computing*, vol.7(no.1):pp.43–57, 1985.

[8] E.J. Haug. *Computer-Aided Kinematics and Dynamics of Mechanical Systems*. Allyn and Bacon, 1989.

[9] R.L. Huston, Y.S Liu, and C. Liu. Use of absolute coordinates in computational multibody dynamics. *Computers and Structures*, vol.52:pp.17–25, 1994.

[10] B. Jonker. A new approach for the dynamic analysis of parallel manipulators. *Computer Methods in Applied Mechanics and Engineering*, vol.79:pp.17–40, 1989.

[11] V. Kevin and M. Whitney. Algorithm 422 minimal spanning tree [h]. *Communications of the ACM*, vol.15(no.4):pp.273–274, 1972.

[12] S.S. Kim and M.J. Vanderploeg. A general and efficient method for dynamic analysis of mechanical systems using velocity transformations. *Journal of Mechanisms, Transmissions, and Automation in Design*, vol.108:pp.176–182, 1986.

[13] H.E. Koenig, Y. Tokad, and H.K. Kesavan. *Analysis of Discrete Physical Systems*. McGraw-Hill, 1967.

[14] J.B. Kruskal. On the shortest spanning subtree and the traveling salesman problem. *Proceedings of the American Mathematical Society*, vol.7:pp.48–50, 1956.

[15] S.Y.T. Lang and H.K. Kesevan. Dynamics of planar mechanical systems: a graph theoretic approach. *Systems, Man, and Cybernetics, 1996., IEEE International Conference on*, vol.4:pp.3077–3082, 1996.

[16] T.W. Li and G.C. Andrews. Application of the vector-network method to constrained mechanical systems. *Journal of Mechanisms, Transmissions, and Automation in Design*, vol.108:pp.471–480, 1986.

[17] M.-J. Liu, C.-X. Li, and C.-N. Li. Dynamics analysis of the goughstewart platform manipulator. *IEEE Transactions on Robotics and Automation*, vol.16(no.1):pp.94–98, 2000.

[18] J. McPhee. Graph-theoretic modelling of multibody systems. SYDE 652 Course Notes, University of Waterloo.

[19] J. McPhee. Automatic generation of motion equations for planar mechanical systems using the new set of branch coordinates. *Mechanism and Machine Theory*, vol.33(no.6):pp.805–823, 1998.

[20] P. Mitiguy and T. Kane. Motion variables leading to efficient equations of motion. *International Journal of Robotics Research*, vol.15(no.5):pp.522–532, 1996.

[21] K. Morency, J. McPhee, and C. Schmitke. Using graph theory and symbolic computing to generate efficient models for vehicle dynamics. In *CSME Forum*, 2006.

[22] J. Nesetril, E. Milkova, and H. Nesetrilova. Otakar boruvka on minimum spanning tree problem: Translation of both the 1926 papers, comments, history. *Discrete Mathematics*, vol.233(no.1-3):pp.3–36, 2001.

[23] P.E. Nikravesh and H.A. Affifi. Construction of the equations of motion for multibody dynamics using point and joint coordinates. *Computer-Aided Analysis of Rigid and Flexible Mechanical Systems*, pages pp.31–60, 1994.

[24] R.C. Prim. Shortest connection networks and some generalisations. *Bell System Technical Journal*, vol.36:pp.1389–1401, 1957.

[25] S. Redmond and J. McPhee. Modelling multibody systems with indirect coordinates. *Computer Methods in Applied Mechanics and Engineering*, vol.195(no.50-51):pp.6942–6957, 2006.

[26] P. Shi and J. McPhee. Dynamics of flexible multibody systems using virtual work and linear graph theory. *Multibody System Dynamics*, vol.4:pp.355–381, 2000.

[27] P. Shi, J. McPhee, and G.R. Heppler. A deformation field for eulerbernoulli beams with applications to flexible multibody dynamics. *Multibody System Dynamics*, vol.5(no.1):pp.79–104, 2001.

[28] M. Simoni. Etude et conception d'un plateau orientable pour piloter les inclinaisons de broche. "Travail de diplme" (Diploma Thesis) from "L'Ecole d'ingénieurs et d'architectes de Fribourg".

[29] J. Unda, J. García de Jalón, F. Losantos, and R. Enparantza. A comparative study on some different formulations of the dynamic equations of constrained mechanical systems. *Journal of Mechanisms, Transmissions, and Automation in Design*, vol.109:pp.466–474, 1987.

[30] J. Wang and C. Gosselin. A new approach for the dynamic analysis of parallel manipulators. *Multibody System Dynamics*, vol.2(no.3):pp.317–334, 1998.

[31] J. Wittenburg. *Dynamics of Systems of Rigid Bodies*. B.G. Teubner, 1977.

[32] JS Zhao, K Zhou, and ZJ Feng. A theory of degrees of freedom for mechanisms. *Mechanism and Machine Theory*, vol.39(no.6):pp.621–643, 2004.

Appendix A

Benchmark Mechanisms

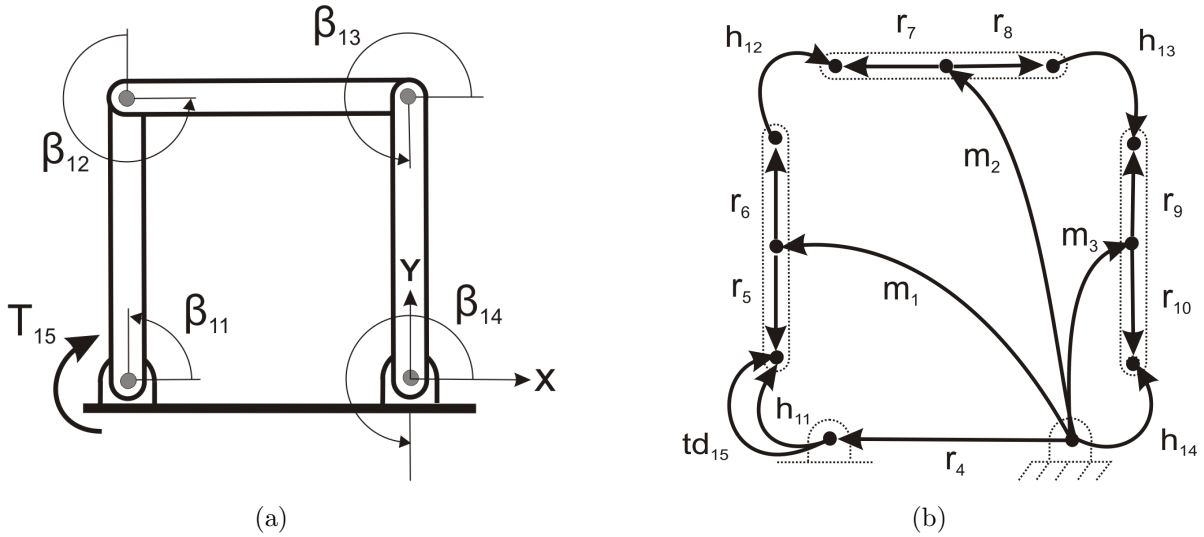
A.1 Planar 4-Bar Mechanism

The planar 4-bar mechanism is depicted in Figure A.1(a) and its graph is depicted in Figure A.1(b). This mechanism is composed of four bodies (m_1, m_2, and m_3, and the ground) connected by four revolute joints (h_{11} to h_{14}) that all rotate about the global Z axis. Each body's center of mass is connected to the ground by the edges m_1 to m_3 and the arms r_4 to r_{10} represent the location where each joint is connected, relative to each body's center of mass. A torque, depicted by td_{15}, is placed on the revolute joint h_{11}. The gravity is given as 9.81 m/s^2 and is directed in the global negative Y direction.

The geometry and inertial properties of the manipulator are shown in Table A.1, where each body's center of mass is found at the body's centroid and the moments of inertia of each body are expressed about this center of mass. The ground length is the length of the arm r_4. The mechanism starts at rest and its initial conditions are presented in Table A.2, where β_{11}, β_{12}, and β_{14} represents the rotation of the revolute joints h_{11}, h_{12}, and h_{14}, respectively. and the time-varying torque applied to the revolute joint h_{11} is given in equation (A.1). This mechanism has one degree of freedom (DOF).

$$T_{15} = 75\sin(2\pi t)\ Nm \tag{A.1}$$

The 4-bar mechanism was simulated using various tree sets. For every different tree set



(a) (b)

Figure A.1: The 4-bar mechanism and its graph.

Table A.1: Body Lengths and Inertia Properties of the 4-Bar Planar Mechanism

	Length (m)	Mass (kg)	Moments of Inertia ($kg \times m^2$)
Body 1	1.00	1.00	1.000
Body 2	2.00	1.00	1.000
Body 3	2.00	1.00	1.000
Ground	2.00	–	–

Table A.2: Initial Conditions of the 4-Bar Planar Mechanism

Coordinate	β_{11}	β_{12}	β_{14}
Initial Value (rad)	0.0	1.31812	1.82348

the kinematic equations were twice differentiated and simplification and code optimization routines were performed on the resulting system equations in Maple. The resulting sets of ODEs were solved for a 10 s simulation using a Runge-Kutta Fehlberg method with an absolute error of 10^{-6} and a relative error of 10^{-5}. The results, which are taken from an average of twenty simulations, are shown in Table A.3. A Pentium 4 of 1.80 GHz with 768 MB of RAM was used to perform these simulations. In the description of the various trees, the arm elements were not presented; however they are always found in both trees. The notation *Same Tree* for the translational tree indicates that this tree is composed of the same edges as the rotational tree. The equation complexity refers to the one described in Section 2.4.

Table A.3: Efficiency of Various Tree Sets of the 4-Bar Planar Mechanism

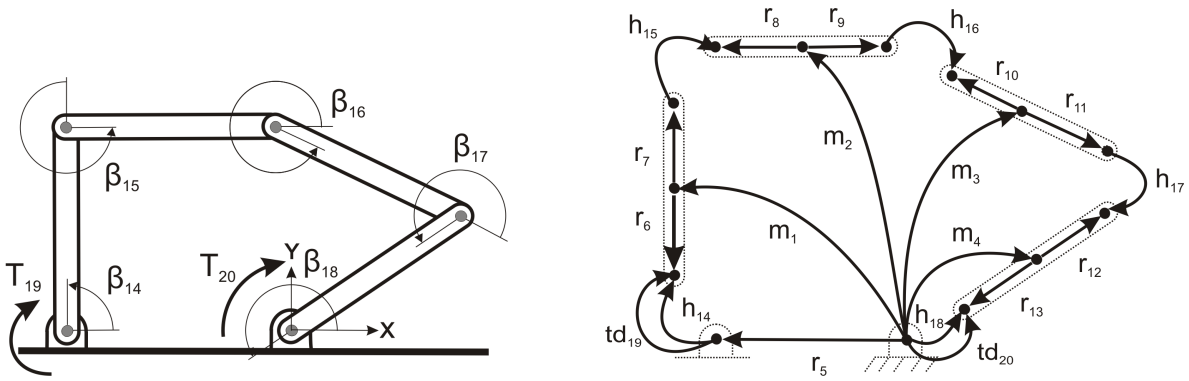
Rotational Tree	Translational Tree	Solution Time (s)	Complexity	# Dyn. EQs	# Kin. EQs
h_{11}, h_{12}, h_{13}	h_{11}, h_{12}, h_{14}	0.73	726	3	2
h_{11}, h_{12}, h_{13}	Same Tree	0.75	887	3	2
h_{11}, h_{12}, h_{14}	h_{11}, h_{12}, h_{13}	0.50	639	3	2
h_{11}, h_{12}, h_{14}	***Same Tree***	***0.48***	***531***	***3***	***2***
h_{11}, h_{13}, h_{14}	h_{11}, h_{12}, h_{14}	0.69	635	3	2
h_{11}, h_{13}, h_{14}	***Same Tree***	***0.51***	***526***	***3***	***2***

The tree sets selected by the Variable-Based Tree Selection Algorithm and the Formulation-Based Tree Selection Algorithm are the same and are depicted in bold italics in Table A.3. Two such tree sets are indicated since both algorithms will randomly select one of the two tree sets indicated. The Variable-Based Tree Selection Algorithm found its tree set in an average of 16 milliseconds while the Formulation-Based Tree Selection Algorithm took an average of 17 milliseconds to find its tree set.

A.2 Planar 5-Bar Mechanism

The planar 5-bar mechanism is depicted in Figure A.2(a) and its graph is depicted in Figure A.2(b). In this graph, the edges h_{14} to h_{18} represent the five revolute joints of the system, which all rotate about the global Z axis. The edges m_1 to m_4 represent the four bodies of the system and link the ground node to the center of mass of each body. Finally, the arm edges r_5 to r_{13} represent the location where each joint is connected, relative to each body's center of mass. Two torque drivers, depicted by td_{19} and td_{20}, are placed on the revolute joints h_{14} and h_{18} respectively. The gravity is given as 9.81 m/s^2 and is directed in the global negative Y direction.

All the bodies of the 5-bar mechanism are 2 m long. They have a mass of 1 kg and a moment of inertia of 1 $kg \times m^2$ about their center of mass at their centroid. The length of the arm r_5 is also 2 m. The mechanism starts at rest and its initial conditions are presented in Table A.4, where β_{14}, β_{15}, β_{17}, and β_{18} represents the rotation of the revolute joints h_{14}, h_{15}, h_{17}, and h_{18}, respectively. The two time-varying torques are given in equations (A.2) and (A.3). This mechanism has two DOF.



(a) (b)

Figure A.2: The 5-bar mechanism and its graph.

$$T_{19} = 30 \sin(2\pi t)\ Nm \tag{A.2}$$

Table A.4: Initial Conditions of the 5-Bar Planar Mechanism

Coordinate	β_{14}	β_{15}	β_{17}	β_{18}
Initial Value (rad)	1.57080	-1.57080	4.18879	2.61799

$$T_{20} = 20\sin(4\pi t)\ Nm \tag{A.3}$$

The 5-bar mechanism was simulated using various tree sets. For every different tree set the kinematic equations were twice differentiated and simplification and code optimization routines were performed on the resulting system equations in Maple. The resulting sets of ODEs were solved for a 10 s simulation using a Runge-Kutta Fehlberg method with an absolute error of 10^{-6} and a relative error of 10^{-5}. The results, which are taken from an average of twenty simulations, are shown in Table A.5. A Pentium 4 of 1.80 GHz with 768 MB of RAM was used to perform these simulations. In the description of the various trees, the arm elements were not presented; however they are always found in both trees. The notation *Same Tree* for the translational tree indicates that this tree is composed of the same edges as the rotational tree. The equation complexity refers to the one described in Section 2.4.

The tree set selected by both the Variable-Based Tree Selection Algorithm and the Formulation-Based Tree Selection Algorithm is depicted in bold italics in Table A.5. The Variable-Based Tree Selection Algorithm found its tree set in an average of 16 milliseconds while the Formulation-Based Tree Selection Algorithm took an average of 15 milliseconds to find its tree set.

A.3 Three Bodies Attached with Two Revolute Joints

This mechanism consists of three bodies attached with two revolute joints as represented in Figure A.3. The graph of this model is presented in Figure A.4, where m_1, m_2, and m_3 relates each body's center of mass to the ground, and h_{10} and h_{11} each represent a revolute

Table A.5: Efficiency of Various Tree Sets of the 5-Bar Planar Mechanism

Rotational Tree	Translational Tree	Solution Time (s)	Complexity	# Dyn. EQs	# Kin. EQs
$h_{14}, h_{15}, h_{16}, h_{17}$	$h_{15}, h_{16}, h_{17}, h_{18}$	1.04	1598	4	2
$h_{14}, h_{15}, h_{16}, h_{17}$	Same Tree	0.97	1598	4	2
$h_{14}, h_{15}, h_{16}, h_{18}$	$h_{14}, h_{16}, h_{17}, h_{18}$	0.69	1094	4	2
$h_{14}, h_{15}, h_{16}, h_{18}$	Same Tree	0.58	921	4	2
$h_{14}, h_{15}, h_{17}, h_{18}$	***Same Tree***	***0.47***	***706***	***4***	***2***
$h_{14}, h_{15}, h_{17}, h_{18}$	$m_2, h_{14}, h_{17}, h_{18}$	0.72	1106	6	4

joint. The location where each joint is connected, relative to each body's center of mass, is depicted by the edges r_4 to r_9.

All three of the mechanism's bodies are 2 m long. They have a mass of 2 kg, I_{zz} and I_{xx} values of 1 $kg \times m^2$, and a I_{yy} value of 0.1 $kg \times m^2$, which are all expressed relative to their center of mass at their centroid. The revolute joint designated by h_{10} rotates about m_2's local x axis, while the revolute joint h_{11} rotates about m_2's local z axis. This mechanism has eight DOF.

The mechanism starts at rest with β_{10} (the rotation of the revolute joint h_{10}) equal to $\pi/3$ rad and β_{11} (the rotation of the revolute joint h_{11}) equal to $\pi/4$ rad. The body m_2's local reference frame is initially positioned and oriented as the global reference frame. The mechanism then falls under the effect of gravity that is in the global negative Z direction.

This mechanism was simulated using various tree sets. For every different tree set the kinematic equations were twice differentiated and simplification and code optimization routines were performed on the resulting system equations in Maple. The resulting sets of ODEs were solved for a 5 s simulation using a Runge-Kutta Fehlberg method with an absolute error of 10^{-6} and a relative error of 10^{-5}. The results, which are taken from an average of twenty simulations, are shown in Table A.6. A Pentium 4 of 1.80 GHz with 768 MB of RAM was used to perform these simulations. In the description of the various trees, the arm elements were not presented; however they are always found in both trees.

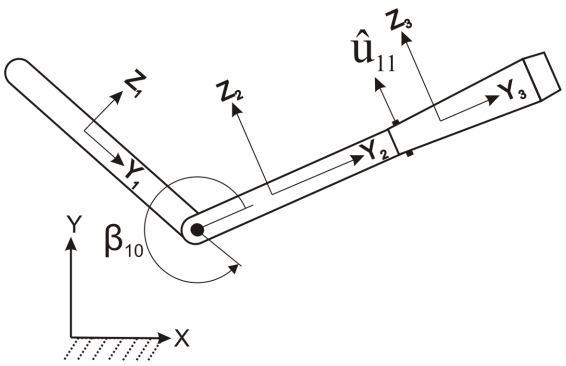


Figure A.3: Three bodies attached with two revolute joints.

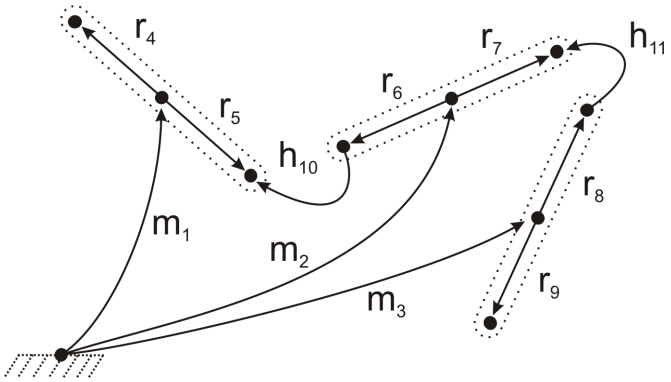


Figure A.4: Graph of three bodies attached with two revolute joints.

The notation *Same Tree* for the translational tree indicates that this tree is composed of the same edges as the rotational tree. The equation complexity refers to the one described in Section 2.4.

Table A.6: Efficiency of Various Tree Sets of the Three Bodies Attached with Two Revolute Joints

Rotational Tree	Translational Tree	Solution Time (s)	Complexity	# Dyn. EQs	# Kin. EQs
$\mathrm{m}_1, \mathrm{m}_2, \mathrm{m}_3$	Same Tree	5.62	8371	18	10
$\mathbf{m_1, m_2, m_3}$	$\mathbf{m_2, h_{10}, h_{11}}$	**7.07**	**10814**	**12**	**4**
m_1, h_{10}, h_{11}	*Same Tree*	*4.82*	*8975*	*8*	*0*
m_2, h_{10}, h_{11}	*Same Tree*	*2.88*	*4827*	*8*	*0*
$\mathrm{m}_1, \mathrm{m}_2, \mathrm{h}_{11}$	Same Tree	4.08	6359	13	5

The tree sets selected by the Variable-Based Tree Selection Algorithm are depicted in italics in Table A.6 and the tree set selected by the Formulation-Based Tree Selection Algorithm is depicted in bold in this table. Two tree sets are indicated for the Variable-Based Tree Selection Algorithm since it will randomly select one of the two tree sets indicated. The Variable-Based Tree Selection Algorithm found its tree set in an average of 9 milliseconds while the Formulation-Based Tree Selection Algorithm took an average of 10 milliseconds to find its tree set.

A.4 Planar Slider-Crank

The planar slider-crank mechanism is depicted in Figure A.5 and its graph is depicted in Figure A.6. In this graph, the edges m_1 to m_3 represent the three rigid bodies in the mechanism. The edges r_4 to r_7 are used to define the locations, relative to the body's center of mass frame, where the joints are connected to each of the bodies. The edges h_8-h_{10} characterize the three revolute joints that rotate about the global Z axis, while the

edge s_{11} represents the prismatic joint connecting the third body to the ground, which translates along the global X axis. The gravity is given as 9.81 m/s^2 and is directed in the global negative X direction.

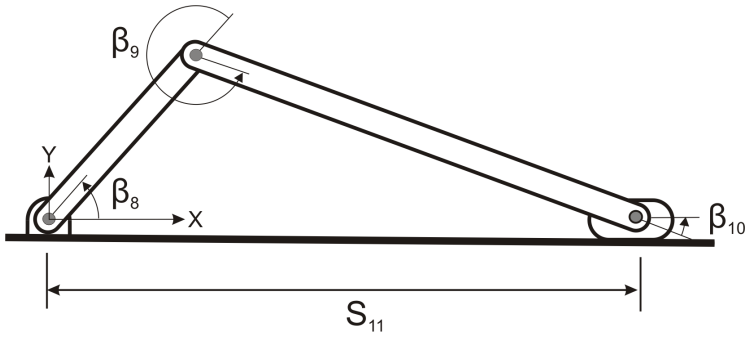


Figure A.5: The planar slider-crank.

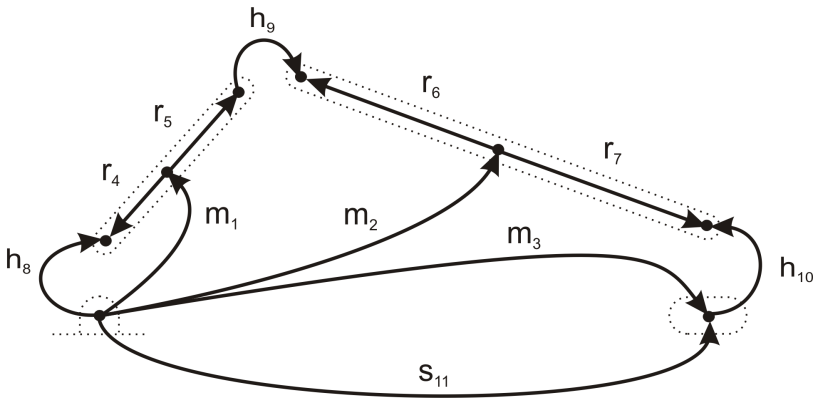


Figure A.6: The planar slider-crank's graph.

The geometry and inertial properties of the mechanism are shown in Table A.7, where the moments of inertia of each body are expressed about this center of mass at the centroid of each body. The numerical values for this system's parameters are based on those from the spatial slider-crank mechanism (Appendix A.6) from Haug [8] that were modified to become a planar mechanism. This mechanism has one DOF.

The mechanism starts at rest with β_8 (the rotation of the revolute joint h_8) equal to 1.00 *rad* and β_{10} (rotation of the revolute joint h_{10}) equal to -1.715380157 *rad*. It then

falls under the effect of gravity.

Table A.7: Body Lengths and Inertia Properties of the Planar Slider-Crank

	Length (m)	Mass (kg)	Moments of Inertia ($kg \times m^2$)
Body 1	0.08	0.12	0.0001
Body 2	0.30	0.50	0.004
Body 3	–	2.00	0.0001

This mechanism was simulated using various tree sets. For every different tree set the kinematic equations were twice differentiated and simplification and code optimization routines were performed on the resulting system equations in Maple. The resulting sets of ODEs were solved for a 10 s simulation using a Runge-Kutta Fehlberg method with an absolute error of 10^{-6} and a relative error of 10^{-5}. The results, which are taken from an average of twenty simulations, are shown in Table A.8. A Pentium 4 of 1.80 GHz with 768 MB of RAM was used to perform these simulations. In the description of the various trees, the arm elements were not presented; however they are always found in both trees. The notation *Same Tree* for the translational tree indicates that this tree is composed of the same edges as the rotational tree. The equation complexity refers to the one described in Section 2.4.

The tree set selected by the Variable-Based Tree Selection Algorithm is depicted in italics in Table A.8 and the tree set selected by the Formulation-Based Tree Selection Algorithm is depicted in bold in this table. The Variable-Based Tree Selection Algorithm found its tree set in an average of 15 milliseconds while the Formulation-Based Tree Selection Algorithm took an average of 16 milliseconds to find its tree set.

A.5 Planar Flexible Slider-Crank

The planar flexible slider-crank mechanism is depicted in Figure A.7 and its graph is depicted in Figure A.8. In this graph, the graph edges m_1 and m_3 represent the two rigid

Table A.8: Efficiency of Various Tree Sets of the Planar Slider-Crank

Rotational Tree	Translational Tree	Solution Time (s)	Complexity	# Dyn. EQs	# Kin. EQs
h_8, h_9, s_{11}	h_8, h_9, h_{10}	0.69	253	2	1
h_8, h_9, s_{11}	Same Tree	1.19	475	3	2
h_8, h_9, h_{10}	Same Tree	1.25	545	3	2
h_8, h_9, h_{10}	h_8, h_{10}, s_{11}	2.19	1016	4	3
h_8, h_{10}, s_{11}	*h_8, h_9, h_{10}*	*0.68*	*197*	*2*	*1*
h_8, h_{10}, s_{11}	**Same Tree**	**1.00**	**353**	**3**	**2**

bodies found in the mechanism, while the edge fm_2 represents the mechanism's flexible body. The edges r_4 and r_5 are used to define the location where the joints are connected on the body relative to the body m_1's center of mass frame. The edge fr_6 represents the length of the flexible body fm_2. The edges h_8 and h_9 characterize the two revolute joints that rotate about the global Z axis, while the edge s_{10} represents the prismatic joint connecting the third body to the ground, which translates along the global X axis. The edge md_7 is a motion driver that provides a constant rotation about the global Z axis. The gravity is given as 9.81 m/s^2 and is directed in the global negative Z direction.

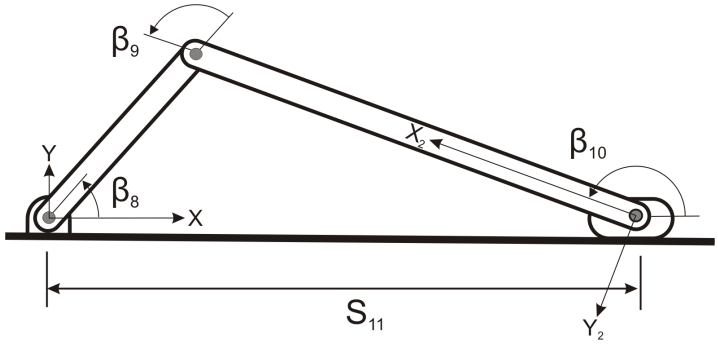


Figure A.7: The planar flexible slider-crank.

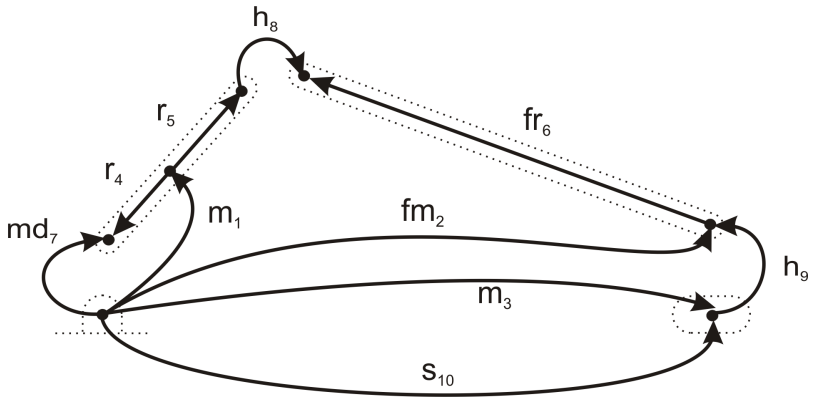


Figure A.8: The planar flexible slider-crank's graph.

This mechanism's properties are taken from Jonker [10], where the flexible body has a length of 0.3 m, a Young's modulus of elasticity (E) equal to 2×10^{11} N/m^2, and a mass density (ρ) of 7870 kg/m^3. The body's circular cross-sectional area has a diameter of 0.003 m. The body m_1 has a length of 0.15 m and the body m_3 has a mass that is half of the flexible body's mass. This mechanism has three DOF, one form the mechanism itself, and two from the flexible body. The beam's lateral deformations along the Y_2 axis, which is due to bending about the Z_2 axis, is considered. This deformation is discretized with three elastic coordinates that use Chebyshev polynomials in the Rayleigh-Ritz discretization.

The slider-crank starts at a horizontal position ($S_{10} = 0.45$ m) and the motion driver md_7 turns at a constant 150 rad/s about the global Z axis. The flexible beams starts with no deflection and no deflection velocity.

This mechanism was simulated using various tree sets. For every different tree set the kinematic equations were twice differentiated and Baumgarte stabilization [2] was applied to the resulting ODEs with the Baumgarte parameters α and β set to 10 and 10, respectively. The resulting set of equations where then sent to simplification and code optimization routines in Maple. The resulting sets of ODEs were solved for a 10 radians rotation of the crank (m_1). This simulation was done using a Runge-Kutta Fehlberg method with an absolute error of 10^{-6} and a relative error of 10^{-5}. The results, which are taken from an average of twenty simulations, are shown in Table A.9. A Pentium 4 of 1.80 GHz with 768 MB of RAM was used to perform these simulations. In the description of

the various trees, the arm elements were not presented; however they are always found in both trees. The notation *Same Tree* for the translational tree indicates that this tree is composed of the same edges as the rotational tree. The equation complexity refers to the one described in Section 2.4.

Table A.9: Efficiency of Various Tree Sets of the Planar Flexible Slider-Crank

Rotational Tree	Translational Tree	Solution Time (s)	Complexity	# Dyn. EQs	# Kin. EQs
h_7, h_8, s_{10}	*h_7, h_8, h_9*	*3.73*	*2182*	*4*	*1*
h_7, h_8, s_{10}	Same Tree	6.10	3290	5	2
h_7, h_8, h_9	Same Tree	5.35	2764	5	2
h_7, h_8, h_9	h_7, h_9, s_{10}	8.42	4859	6	3
h_7, h_9, s_{10}	*h_7, h_8, h_9*	*2.59*	*1411*	*4*	*1*
h_7, h_9, s_{10}	**Same Tree**	**3.57**	**2109**	**5**	**2**

The tree set selected by the Variable-Based Tree Selection Algorithm are depicted in italics in Table A.9 and the tree set selected by the Formulation-Based Tree Selection Algorithm is depicted in bold in this table. Two tree sets are indicated for the Variable-Based Tree Selection Algorithm since it will randomly select one of the two tree sets indicated. The Variable-Based Tree Selection Algorithm found its tree set in an average of 16 milliseconds while the Formulation-Based Tree Selection Algorithm took an average of 15 milliseconds to find its tree set.

A.6 Spatial Slider-Crank

The spatial slider-crank mechanism is depicted in Figure A.9 and its graph is depicted in Figure A.10. The spatial slider-crank is formed by a revolute joint, depicted by the edge h_9, that connects a crank, depicted by the body m_1 and its two arms r_5 and r_6, to the ground. This revolute joint rotates about the global X axis. The location where

the revolute joint connects to the ground is depicted by the edge r_4. The crank is then connected to a connecting rod, represented by the edges m_2 and its two arms r_7 and r_8, by a spherical joint corresponding to the edge b_{10}. Finally, a sliding block (m_3) is connected to the connecting rod via a universal joint (u_{11}) and connected to the ground by a prismatic joint (s_{12}) that translates about the global X axis. This universal joint's angle α rotates about the global X axis, while this joint's β angle rotates about m_2's body-fixed z axis. Finally, gravity is given as 9.81 m/s^2 and is directed in the negative global Z direction.

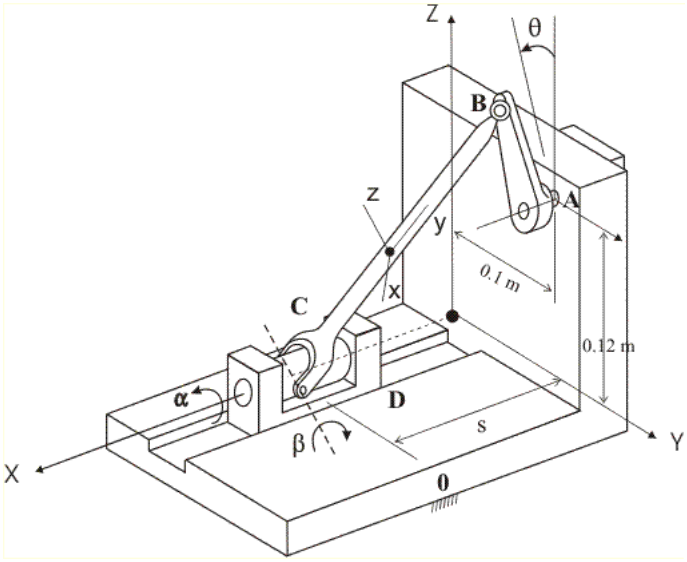


Figure A.9: The spatial slider-crank.

The geometry and inertial properties of the manipulator are shown in Table A.10, where each body's center of mass is found at the body's centroid and the moments of inertia of each body are expressed about this center of mass. The numerical values for all system parameters are taken from Haug [8]. The mechanism is simulated falling due to gravity and its initial conditions are given in Table A.11, where θ_9, α_{11}, and β_{11} represents the rotation of the revolute joints h_9, and the two rotation of the universal joint u_{11}, respectively. This mechanism has one DOF.

This mechanism was simulated using various tree sets. For every different tree set the kinematic equations were twice differentiated and simplification and code optimization

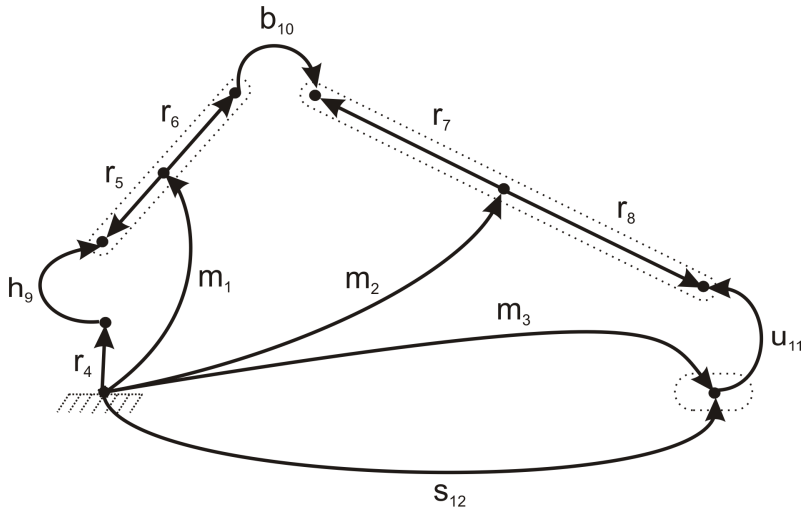


Figure A.10: The spatial slider-crank's graph.

Table A.10: Body Lengths and Inertia Properties of the Spatial Slider-Crank

	Length (m)	Mass (kg)	I_{xx} $(kg \times m^2)$	I_{yy} $(kg \times m^2)$	I_{zz} $(kg \times m^2)$
Body 1	0.08	0.12	0.00010	0.00001	0.00010
Body 2	0.30	0.50	0.00400	0.00400	0.00040
Body 3	–	2.00	0.00010	0.00010	0.00010

Table A.11: Initial Conditions of the Spatial Slider-Crank

Coordinate	$\theta_9\,(rad)$	$\alpha_{11}\,(rad)$	$\beta_{11}\,(rad)$	$\frac{d}{dt}\theta_9\,(rad/s)$	$\frac{d}{dt}\alpha_{11}\,(rad/s)$	$\frac{d}{dt}\beta_{11}\,(rad/s)$
Initial Value	0.00000	0.46365	-0.84107	-6.00000	-1.92000	1.07331

routines were performed on the resulting system equations in Maple. The resulting sets of ODEs were solved for a 10 s simulation using a Runge-Kutta Fehlberg method with an absolute error of 10^{-6} and a relative error of 10^{-5}. The results, which are taken from an average of twenty simulations, are shown in Table A.12. A Pentium 4 of 1.80 GHz with 768 MB of RAM was used to perform these simulations. In the description of the various trees, the arm elements were not presented; however they are always found in both trees. The notation *Same Tree* for the translational tree indicates that this tree is composed of the same edges as the rotational tree. The equation complexity refers to the one described in Section 2.4.

Table A.12: Efficiency of Various Tree Sets of the Spatial Slider-Crank

Rotational Tree	Translational Tree	Solution Time (s)	Complexity	# Dyn. EQs	# Kin. EQs
h_9, u_{11}, s_{12}	*h_9, b_{10}, u_{11}*	*1.87*	*926*	*3*	*2*
h_9, u_{11}, s_{12}	m_2, h_9, u_{11}	3.33	2070	6	5
h_9, u_{11}, s_{12}	**Same Tree**	**1.53**	**710**	**4**	**3**
h_9, u_{11}, s_{12}	m_2, h_9, s_{12}	2.88	1927	7	6
h_9, b_{10}, u_{11}	Same Tree	17.81	7521	6	5
h_9, b_{10}, u_{11}	h_9, u_{11}, s_{12}	21.58	8592	7	6
h_9, b_{10}, s_{12}	h_9, b_{10}, u_{11}	7.89	3390	4	3
h_9, b_{10}, s_{12}	h_9, u_{11}, s_{12}	8.66	3846	5	4

The tree set selected by the Variable-Based Tree Selection Algorithm is depicted in italics in Table A.12 and the tree set selected by the Formulation-Based Tree Selection Algorithm is depicted in bold in this table. The Variable-Based Tree Selection Algorithm found its tree set in an average of 16 milliseconds while the Formulation-Based Tree Selection Algorithm took an average of 15 milliseconds to find its tree set.

To verify if similar results could be obtained using other solvers, the system's equations were exported into Matlab S-functions. This Matlab S-function was then solved

in Matlab/Simulink using various ODE solvers and using the same parameters as those used in Maple. The results of these simulations are presented in Table A.13. Here, the term *ODE45* refers to the simulations solved using Matlab's ODE45, which is based on a Runge-Kutta formula called the Dormand-Prince pair. The term *ODE113* refers to the simulations solved using Matlab's ODE113, which is an Adams-Bashforth-Moulton PECE (predictor-evaluate-corrector-evaluate) solver. Finally, the term *ODE15s* refers to the simulations solved using Matlab's ODE15s, which is a stiff solver based on numerical differentiation formulas (NDFs).

Table A.13: Efficiency of Various Tree Sets of the Spatial Slider-Crank Using Other Solvers

Rotational Tree	Translational Tree	ODE45 (s)	ODE113 (s)	ODE15s (s)
h_9, u_{11}, s_{12}	h_9, b_{10}, u_{11}	*0.43*	*0.47*	*0.62*
$\mathbf{h_9, u_{11}, s_{12}}$	**Same Tree**	**0.30**	**0.35**	**0.47**

A.7 Planar 3-RRR Parallel Manipulator

The 3-RRR parallel manipulator is depicted in Figure A.11 and its graph is depicted in Figure A.12. This mechanism is comprised of three legs, each having three revolute joints. The joints connected to the ground (h_{25}, h_{28}, and h_{31}) are generally referred to as the ankle joints. The revolute joints connected to the platform (h_{27}, h_{30}, and h_{33}) are called the hip joints. Lastly, the revolute joints at the center of the legs (h_{26}, h_{29}, and h_{32}) are called knee joints. All of these revolute joints rotate about the global Z axis. Each body's center of mass is connected to the ground by the edges m_1 to m_7, and the location where each joint is connected, relative to each body's center of mass, is depicted by the edges r_8 to r_{24}.

The geometry and inertial properties of the manipulator are shown in Table A.14 and Table A.15, where the moments of inertia are about the center of gravity at the centroid of each section and where the platform is an equilateral triangle. The platform length

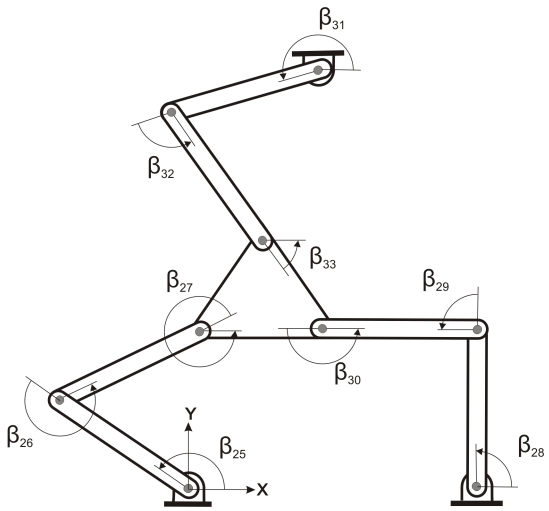


Figure A.11: The planar 3-RRR parallel manipulator.

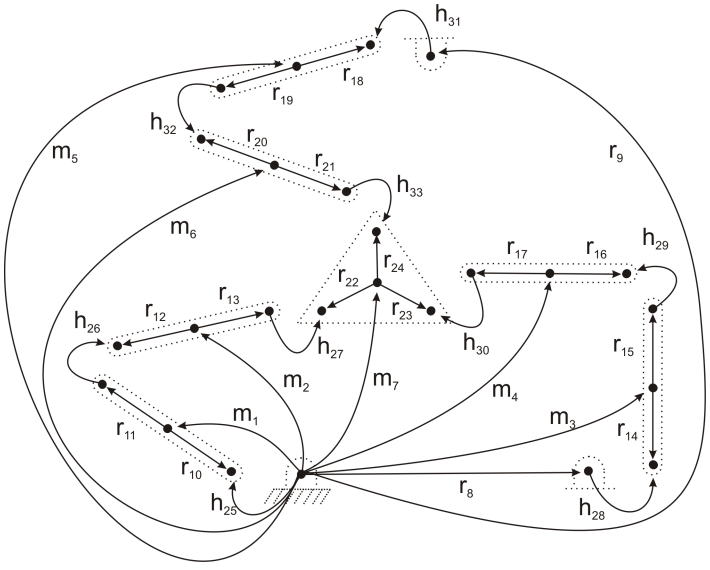


Figure A.12: The planar 3-RRR parallel manipulator's graph.

corresponds to the length of its sides and the term *Shank Bodies* refers to the bodies connected to the ground via a revolute joint while the term *Thigh Bodies* refer to the bodies connected to the platform via a revolute joint. The initial conditions used to model the system are presented in Table A.16, where β_{25}, β_{28}, and β_{31} represents the rotation of the revolute joints h_{25}, h_{28}, and h_{31}, respectively. The numerical values for all system parameters are taken from Geike and McPhee [6]. This mechanism has three DOF.

Table A.14: Body Lengths and Inertia Properties of the 3-RRR Planar Parallel Manipulator

	Length (m)	Mass (kg)	Moments of Inertia $(kg \times m^2)$
Shank Bodies	0.40	3.00	0.0400
Thigh Bodies	0.60	4.00	0.1200
Platform	0.40	8.00	0.0817

Table A.15: Position where the Legs are Connected to the Ground for the 3-RRR Planar Parallel Manipulator

Position	First Leg	Second Leg	Third Leg
x_G (m)	0.000	1.000	0.500
y_G (m)	0.000	0.000	$\sqrt{3}/2$

For every different tree set, the mechanism was simulated while starting at rest and falling for 3 s under the force of gravity, which is directed in the global negative Y direction. For every different tree set the kinematic equations were twice differentiated and Baumgarte stabilization [2] was applied to the resulting ODEs with the Baumgarte parameters α and β set to 10 and 5, respectively. The resulting set of equations where then sent to simplification and code optimization routines in Maple. The resulting sets of ODEs were solved using

Table A.16: Initial Conditions of the Manipulator of the 3-RRR Planar Parallel Manipulator

Coordinate	β_{25}	β_{28}	β_{31}
Initial Value (rad)	1.0472	4.1888	5.7596

a Runge-Kutta Fehlberg method with an absolute error of 10^{-6} and a relative error of 10^{-5}. The results, which are taken from an average of twenty simulations, are shown in Table A.17. A Pentium 4 of 1.80 GHz with 768 MB of RAM was used to perform these simulations. In the description of the various trees, the arm elements were not presented; however they are always found in both trees. The notation *Same Tree* for the translational tree indicates that this tree is composed of the same edges as the rotational tree. The equation complexity refers to the one described in Section 2.4.

Table A.17: Efficiency of Various Tree Sets of the 3-RRR Planar Parallel Manipulator

Rotational Tree	Translational Tree	Solution Time (s)	Complexity	# Dyn. EQs	# Kin. EQs
$\boldsymbol{h_{25} - h_{29}, h_{31}, h_{32}}$	***Same Tree***	***1.74***	***2718***	***7***	***4***
$h_{25} - h_{29}, h_{31}, h_{32}$	$h_{25} - h_{28}, h_{30}, h_{31}, h_{33}$	8.47	11948	7	4
$h_{25} - h_{28}, h_{30}, h_{31}, h_{33}$	$h_{25} - h_{29}, h_{31}, h_{32}$	5.88	7610	7	4
$h_{25}, h_{26}, h_{28} - h_{32}$	$h_{25} - h_{29}, h_{31}, h_{32}$	5.74	7798	7	4

The tree set selected by both the Variable-Based Tree Selection Algorithm and the Formulation-Based Tree Selection Algorithm is depicted in bold italics in Table A.17. The Variable-Based Tree Selection Algorithm found its tree set in an average of 16 milliseconds while the Formulation-Based Tree Selection Algorithm took an average of 79 milliseconds to find its tree set.

In order to be able to model the planar 3-RRR parallel manipulator using Absolute Angular Coordinates, virtual joints allowing rotations about only the global Z axis need to be added between the ground and every body in the system. In this case, the needed virtual joints can be simulated by adding planar joints between every body's center of mass and the ground node. The new system graph for one of the legs of the 3-RRR manipulator containing these extra planar joints (in bold) is illustrated in Figure A.13. The virtual planar joint added to the body m_i will be called vp_{i+33}.

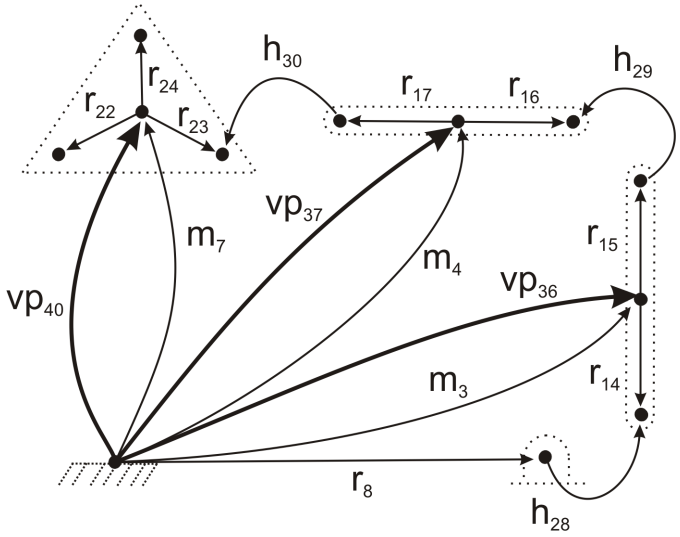


Figure A.13: The planar 3-RRR parallel manipulator's graph with added virtual planar joints.

The planar 3-RRR manipulator was once again simulated using various tree sets under the same conditions as the previous simulations. The results of this simulation are found in Table A.18, where the abbreviation *Compl.* denotes the equation complexity.

The tree set selected by both the Variable-Based Tree Selection Algorithm and the Formulation-Based Tree Selection Algorithm is depicted in bold italics in Table A.18. The Variable-Based Tree Selection Algorithm found its tree set in an average of 16 milliseconds while the Formulation-Based Tree Selection Algorithm took an average of 140 milliseconds to find its tree set.

Table A.18: Efficiency of Various Tree Sets of the 3-RRR Planar Parallel Manipulator with Virtual Joints

Rotational Tree	Translational Tree	Sol. Time (s)	Comp.	# Dyn. EQs	# Kin. EQs
$\boldsymbol{vp_{34} - vp_{40}}$	$\boldsymbol{h_{25} - h_{29}, h_{31}, h_{32}}$	***1.26***	***1983***	***7***	***4***
$\mathrm{vp}_{34} - \mathrm{vp}_{40}$	$\mathrm{h}_{25}, \mathrm{h}_{26}, \mathrm{h}_{28}, \mathrm{h}_{29}, \mathrm{h}_{30}, \mathrm{h}_{31}, \mathrm{vp}_{40}$	1.55	2080	9	6
$\mathrm{h}_{25}, \mathrm{h}_{27}, \mathrm{h}_{28}, \mathrm{h}_{30}, \mathrm{h}_{31}, \mathrm{h}_{33}, \mathrm{vp}_{40}$	$\mathrm{h}_{25} - \mathrm{h}_{28}, \mathrm{h}_{30}, \mathrm{h}_{31}, \mathrm{h}_{33}$	2.47	3672	7	4
$\mathrm{h}_{25}, \mathrm{h}_{27}, \mathrm{h}_{28}, \mathrm{h}_{30}, \mathrm{h}_{31}, \mathrm{h}_{33}, \mathrm{vp}_{40}$	Same Tree	2.90	4141	9	6
$\mathrm{h}_{25}, \mathrm{h}_{26}, \mathrm{h}_{28}, \mathrm{h}_{29}, \mathrm{h}_{31}, \mathrm{h}_{32}, \mathrm{vp}_{40}$	$\mathrm{h}_{25} - \mathrm{h}_{29}, \mathrm{h}_{31}, \mathrm{h}_{32}$	1.55	2502	7	4
$\mathrm{h}_{25}, \mathrm{h}_{26}, \mathrm{h}_{28}, \mathrm{h}_{29}, \mathrm{h}_{31}, \mathrm{h}_{32}, \mathrm{vp}_{40}$	Same Tree	1.51	2272	9	6

A.8 Planar 3-RPR Parallel Manipulator

The 3-RRR parallel manipulator is depicted in Figure A.14 and its graph is depicted in Figure A.15. This mechanism is comprised of three legs, each having two revolute joints and one prismatic joint. The legs are connected to the ground by revolute joints (h_{26}, h_{27}, and h_{28}). The legs also connect to the platform with revolute joints (h_{32}, h_{33}, and h_{34}). All of these revolute joints rotate about the global Z axis. Lastly, the two leg sections are connected together with prismatic joints (s_{29}, s_{30}, and s_{31}) that translate along the length of the legs. The bodies' centers of mass are connected to the ground by the edges m_1 to m_7 and the location where each joint is connected, relative to each body's center of mass, are depicted by the edges r_8 to r_{25}. The edges fd_{35} to fd_{37} represent force drivers placed on the prismatic joints of the mechanism. The virtual planar joint vp_{38} is added to track the planar movement of the platform. Finally, gravity is placed along the global Z axis, which is parallel to the axis of rotation of the revolute joints.

The geometry and inertial properties of the manipulator are shown in Table A.19 and Table A.20, where the moments of inertia are about the center of gravity at the centroid of each section. The platform is shaped as an Isosceles triangle and it's bottom length

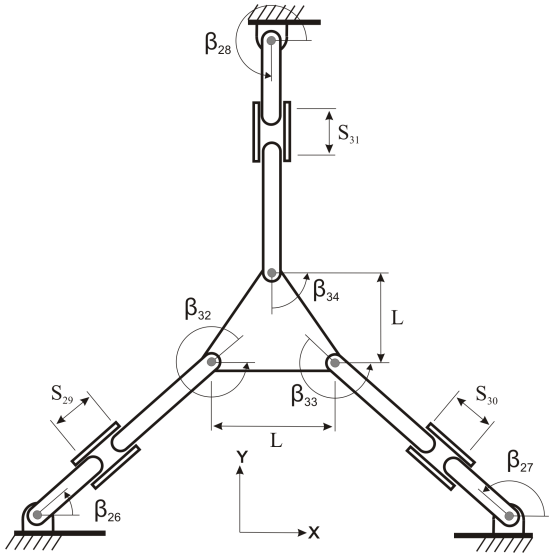


Figure A.14: The planar 3-RPR parallel manipulator.

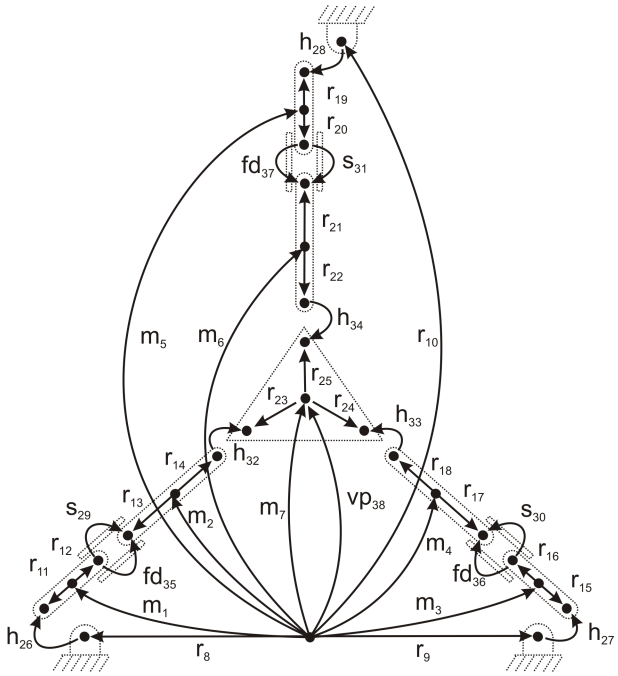


Figure A.15: The planar 3-RPR parallel manipulator's graph.

(the other two lengths being the two lengths of equal value), which is also it's height (as depicted in Figure A.14), is provided in Table A.19. The term *Shank Bodies* refers to the bodies connected to the ground via a revolute joint while the term *Thigh Bodies* refer to the bodies connected to the platform via a revolute joint. The initial conditions used to model the system are presented in Table A.21, where β_{38} represents the platform's orientation about the Z axis, and X_{38} and Y_{38} represents the platform's position along the ground reference frame's X and Y axis. Table A.22 presents the forces acting on each prismatic joint. This mechanism has three DOF.

Table A.19: Body Lengths and Inertia Properties of the 3-RPR Planar Parallel Manipulator

	Length (m)	Mass (kg)	Moments of Inertia $(kg \times m^2)$
Shank Bodies	1.00	3.00	0.2725
Thigh Bodies	1.50	1.50	0.2825
Platform	1.00	5.00	0.4167

Table A.20: Position where the Legs are Connected to the Ground for the 3-RPR Planar Parallel Manipulator

Position	First Leg	Second Leg	Third Leg
x_G (m)	-1.750	1.750	5.665
y_G (m)	0.000	0.000	0.000

For every different tree set, the mechanism was simulated for 10 s while starting at rest. For every different tree set the kinematic equations were twice differentiated and Baumgarte stabilization [2] was applied to the resulting ODEs with the Baumgarte parameters α and β set to 10 and 5, respectively. The resulting set of equations where then sent to simplification and code optimization routines in Maple. The resulting sets of ODEs were solved using

Table A.21: Initial Conditions of the Manipulator of the 3-RPR Planar Parallel Manipulator

Coordinate	$\beta_{38}\,(rad)$	$X_{38}\,(m)$	$Y_{38}\,(m)$
Initial Value	0.000	0.000	2.665

Table A.22: Forces Acting on the Prismatic Joints of the 3-RPR Planar Parallel Manipulator

Force	F_{35}	F_{36}	F_{37}
Value (N)	15	10	12

a Runge-Kutta Fehlberg method with an absolute error of 10^{-6} and a relative error of 10^{-5}. The results, which are taken from an average of twenty simulations, are shown in Table A.23. A Pentium 4 of 1.80 GHz with 768 MB of RAM was used to perform these simulations. In the description of the various trees, the arm elements were not presented; however they are always found in both trees. The notation *Same Tree* for the translational tree indicates that this tree is composed of the same edges as the rotational tree. The term *Comp.* denotes the equation complexity that is described in Section 2.4.

The tree set selected by the Variable-Based Tree Selection Algorithm is depicted in italics in Table A.23 and the tree set selected by the Formulation-Based Tree Selection Algorithm is depicted in bold in this table. The Variable-Based Tree Selection Algorithm found its tree set in an average of 16 milliseconds while the Formulation-Based Tree Selection Algorithm took an average of 62 milliseconds to find its tree set.

Table A.23: Efficiency of Various Tree Sets of the 3-RPR Planar Parallel Manipulator

Rotational Tree	Translational Tree	Sol. Time (s)	Comp.	# Dyn. EQs	# Kin. EQs
$\mathbf{h_{26} - h_{28}, s_{29} - s_{31}, vp_{38}}$	$\mathbf{h_{26} - h_{28}, h_{32} - h_{34}, vp_{38}}$	**1.57**	**2538**	**6**	**3**
$h_{26} - h_{28}, s_{29} - s_{31}, vp_{38}$	$h_{26} - h_{28}, s_{29}, h_{32} - h_{34}$	*1.32*	*2147*	*5*	*2*
$h_{26} - h_{28}, s_{29} - s_{31}, vp_{38}$	$h_{26} - h_{28}, s_{29} - s_{31}, h_{32}$	1.36	1939	7	4
$h_{26} - h_{28}, s_{29} - s_{31}, vp_{38}$	Same Tree	1.14	1827	9	6
$s_{29} - s_{31}, h_{32} - h_{34}, vp_{38}$	$h_{26} - h_{28}, s_{29}, h_{32} - h_{34}$	1.09	2731	5	2

A.9 Peaucellier-Lipkin Straight-Line Mechanism

The Peaucellier-Lipkin straight-line mechanism is depicted in Figure A.16 and its graph is depicted in Figure A.17. In this graph, the edges h_{23} to h_{32} represent the revolute joints of the system, which all rotate about the global Z axis. The edges m_1 to m_7 represent the bodies of the system and link the ground node to the center of mass of each body. The edges r_8 to r_{22} are the arm elements that define the location where each joint is connected, relative to each body's center of mass. A torque driver, depicted by td_{33} is placed on the revolute joint h_{31}. The gravity is directed in the global negative Z direction. Finally, a planar virtual joint (vp_{34}) is placed between the ground and the extremity of the body (m_6).

The geometry and inertial properties of the manipulator's three body types, each having a different length, (Depicted in Figure A.16) are shown in Table A.24, where each body's center of mass is found at the body's centroid and the moments of inertia of each body are expressed about this center of mass. The length of the arm r_8 is the same length as the length of the body type C. The mechanism starts at rest and its initial conditions are presented in Table A.25, where β_{23}, β_{24}, and β_{31} represents the rotation of the revolute joints h_{23}, h_{24}, and h_{31}, respectively.. The time-varying torque applied to the revolute joint h_{31} is given in equation (A.4). This mechanism has one DOF.

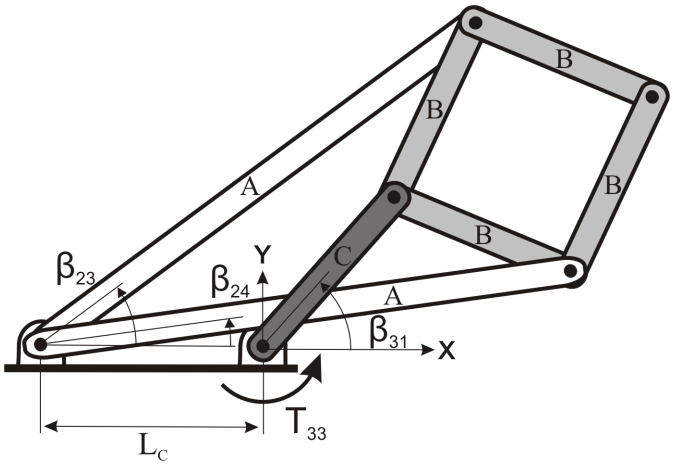


Figure A.16: The Peaucellier-Lipkin straight-line mechanism.

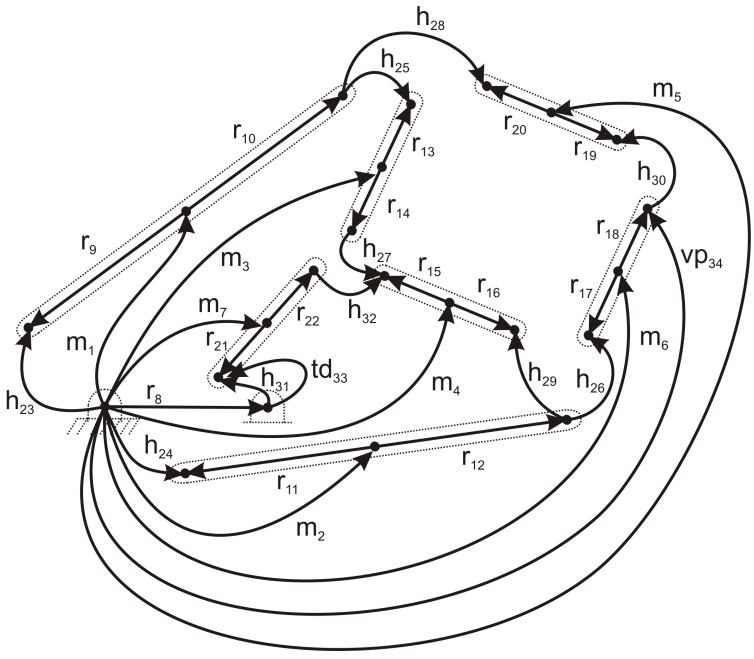


Figure A.17: The Peaucellier-Lipkin straight-line mechanism's graph.

Table A.24: Body Lengths and Inertia Properties of the Peaucellier-Lipkin Straight-Line Mechanism

	Length (m)	Mass (kg)	Moments of Inertia ($kg \times m^2$)
Body Type A	0.40	4.00	0.05333
Body Type B	0.20	2.00	0.00667
Body Type C	0.15	1.50	0.00281

Table A.25: Initial Conditions of the Peaucellier-Lipkin Straight-Line Mechanism

Coordinate	β_{23}	β_{24}	β_{31}
Initial Value (rad)	0.50536	-0.50536	0.00000

$$T_{33} = -0.1t\sin(\pi t) - 0.1t\sin(4\pi t)\ Nm \tag{A.4}$$

The Peaucellier-Lipkin mechanism is simulated using various tree sets. For every different tree set the kinematic equations were twice differentiated and simplification and code optimization routines were performed on the resulting system equations in Maple. The resulting sets of ODEs were solved for a 10 s simulation using a Runge-Kutta Fehlberg method with an absolute error of 10^{-6} and a relative error of 10^{-5}. The results, which are taken from an average of twenty simulations, are shown in Table A.26. A Pentium 4 of 1.80 GHz with 768 MB of RAM was used to perform these simulations. In the description of the various trees, the arm elements were not presented; however they are always found in both trees. The notation *Same Tree* for the translational tree indicates that this tree is composed of the same edges as the rotational tree. The term *Comp.* denotes the equation complexity that is described in Section 2.4.

The tree set selected by both the Variable-Based Tree Selection Algorithm and the Formulation-Based Tree Selection Algorithm is depicted in bold italics in Table A.26. The

Table A.26: Efficiency of Various Tree Sets of the Peaucellier-Lipkin Straight-Line Mechanism

Rotational Tree	Translational Tree	Sol. Time (s)	Comp.	# Dyn. EQs	# Kin. EQs
$h_{23} - h_{26}, h_{28}, h_{31}, h_{32}$	Same Tree	1.18	3308	7	6
$\boldsymbol{h_{23} - h_{26}, h_{31}, h_{32}, vp_{38}}$	$\boldsymbol{h_{23} - h_{26}, h_{28}, h_{31}, h_{32}}$	***1.17***	***3228***	***7***	***6***
$h_{23} - h_{26}, h_{31}, h_{32}, vp_{38}$	Same Tree	1.32	3715	9	8
$h_{25} - h_{29}, h_{31}, h_{32}$	Same Tree	3.53	9692	7	6

Variable-Based Tree Selection Algorithm found its tree set in an average of 16 milliseconds while the Formulation-Based Tree Selection Algorithm took an average of 78 milliseconds to find its tree set.

A.10 Spatial Serial Manipulator

A spatial serial manipulator is shown in Figure A.18(a). This manipulator was originally used by Fayet and Pfister [5] as an example of a system that is best modelled using Indirect Coordinates. Its graph, developed by McPhee and Redmond [25] is displayed in Figure A.18(b). The model consists of four bodies (m_1 to m_4), the first of which is connected to the ground with a revolute joint h_{12} that rotates about the global Z axis. The bodies m_1 and m_2 are connected together by a revolute joint (h_{13}) that rotates about the m_1's local x axis. Similarly, the bodies m_2 and m_3 are connected together by a revolute joint (h_{14}) that rotates about m_2's local x axis, which is parallel to h_{13}'s rotation axis. The bodies m_3 and m_4 are connected together with a spherical joint b_{15}. The location where each joint is connected, relative to each body's center of mass, is depicted by the edges r_5 to r_{11}.

According to Fayet and Pfister's rules for selecting Indirect Coordinates, described in Section 1.2, measuring the rotation of body m_3 relative to body m_1 and measuring the

rotation of body m_4 relative to the ground is the most efficient method of modelling the given system. This is why the bodies m_1 and m_3 are also connected together via a virtual planar joint vp_{16} that rotates about m_1's local x axis.

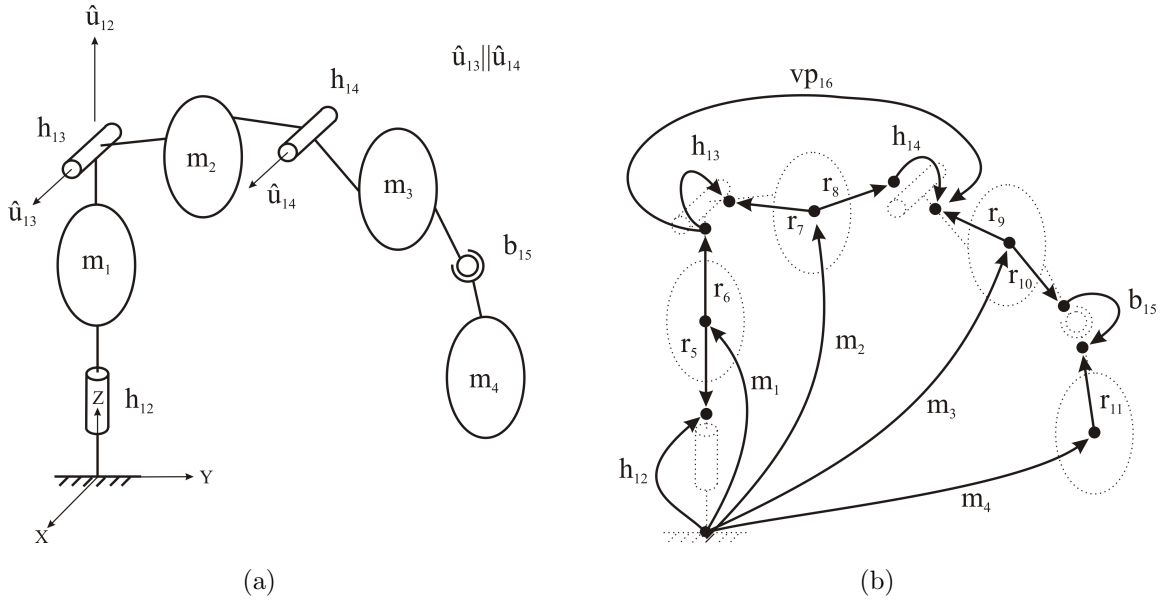


Figure A.18: The spatial serial manipulator and its graph.

The geometry and inertial properties of the manipulator bodies are taken from McPhee and Redmond [25] and are shown in Table A.27, where each body's center of mass is found at the body's centroid and the moments of inertia of each body are expressed about this center of mass. The mechanism starts at rest, its initial conditions presented in Table A.28, where β_{12}-β_{14} represents the rotation of the revolute joints h_{12}-h_{14}, respectively. The terms ζ_{15}, η_{15}, and ξ_{15} are the Euler 313 angles of rotation of the spherical joint b_15. Gravity is given the value of 9.81 m/s^2 and it is directed along the global negative Z axis. This mechanism has six DOF.

The spatial serial manipulator is simulated falling under the effect of gravity using various tree sets. For every different tree set the kinematic equations were twice differentiated and simplification and code optimization routines were performed on the resulting system equations in Maple. The resulting sets of ODEs were solved for a 4 s simulation using

Table A.27: Body Lengths and Inertia Properties of the Spatial Serial Manipulator

	Length (m)	Mass (kg)	$I_{xx}(kg \times m^2)$	$I_{yy}(kg \times m^2)$	$I_{zz}(kg \times m^2)$
Body 1	1	312.04	27.0435	27.0435	2.0803
Body 2	2	624.08	4.1605	210.1069	210.1069
Body 3	2	624.08	4.1605	210.1069	210.1069
Body 4	1	312.04	2.0803	27.0435	27.0435

Table A.28: Initial Conditions of the Spatial Serial Manipulator

Coordinate	β_{12}	β_{13}	β_{14}	ζ_{15}	η_{15}	ξ_{15}
Initial Value (rad)	0	$\pi/2$	0	$-\pi/2$	$\pi/2$	$\pi/2$

a Runge-Kutta Fehlberg method with an absolute error of 10^{-6} and a relative error of 10^{-5}. The results, which are taken from an average of twenty simulations, are shown in Table A.29. A Pentium 4 of 1.80 GHz with 768 MB of RAM was used to perform these simulations. In the description of the various trees, the arm elements were not presented; however they are always found in both trees. The notation *Same Tree* for the translational tree indicates that this tree is composed of the same edges as the rotational tree. The equation complexity refers to the one described in Section 2.4.

The tree set selected by both the Variable-Based Tree Selection Algorithm and the Formulation-Based Tree Selection Algorithm is depicted in bold italics in Table A.29. The Variable-Based Tree Selection Algorithm found its tree set in an average of 10 milliseconds while the Formulation-Based Tree Selection Algorithm took an average of 15 milliseconds to find its tree set.

Table A.29: Efficiency of Various Tree Sets of the Spatial Serial Manipulator

Rotational Tree	Translational Tree	Solution Time (s)	Complexity	# Dyn. EQs	# Kin. EQs
$\mathrm{h}_{12} - \mathrm{h}_{14}, \mathrm{m}_4$	$\mathrm{h}_{12} - \mathrm{h}_{14}, \mathrm{b}_{15}$	1.01	1835	6	0
$\mathrm{h}_{12} - \mathrm{h}_{14}, \mathrm{m}_4$	Same Tree	1.00	1479	9	3
$\boldsymbol{h_{12}, h_{13}, m_4, vp_{16}}$	$\boldsymbol{h_{12} - h_{14}, b_{15}}$	***0.92***	***1539***	***6***	***0***
$\mathrm{h}_{12}, \mathrm{h}_{13}, \mathrm{m}_4, \mathrm{vp}_{16}$	$\mathrm{h}_{12} - \mathrm{h}_{14}, \mathrm{m}_4$	0.78	1179	9	3
$\mathrm{h}_{12} - \mathrm{h}_{14}, \mathrm{b}_{15}$	Same Tree	4.09	5737	6	0

A.11 Stewart-Gough Platform

The Stewart-Gough Platform (SGP), illustrated in Figure A.19 (taken with permission from [30]), is a six degree of freedom (DOF) parallel manipulator consisting of six identical legs connecting a hexahedral-shaped platform to the ground. An illustration of a part of the graph model of the manipulator is shown in Figure A.20. In order for the illustration of the graph not to become too cluttered, only one of the manipulator's legs is shown. Each lower leg section (m_1) is attached to the ground with a universal joint (u_{10}), as shown in Figure A.21. The first universal joint angle α is about the global Z axis (Z_G), and measures the angle from Y_G to the leg's local Y axis (Y_{L1}), about which the second joint angle β is measured. Each leg's bottom and top sections (m_1 and m_2) are connected with a prismatic joint (s_{11}) with a force actuator (fd_{13}) directed along the axis Z_{L1}. Finally, each leg is connected to the platform with a spherical joint (b_{12}).

The Stewart-Gough platform is then modelled using the parameters found in Liu et al. [17]. The Cartesian coordinates of the points at which the legs attach to the ground and platform are described in Table A.30 and Table A.31, respectively:

The ground reference frame (G_{XYZ}) as well as the top ($L2_{XYZ}$) and bottom ($L1_{XYZ}$) local reference frames on each leg are shown in Figure A.22. The platform's local reference frame is found at its centroid.

Table A.32 shows the center of mass of each body relative to its local reference frame,

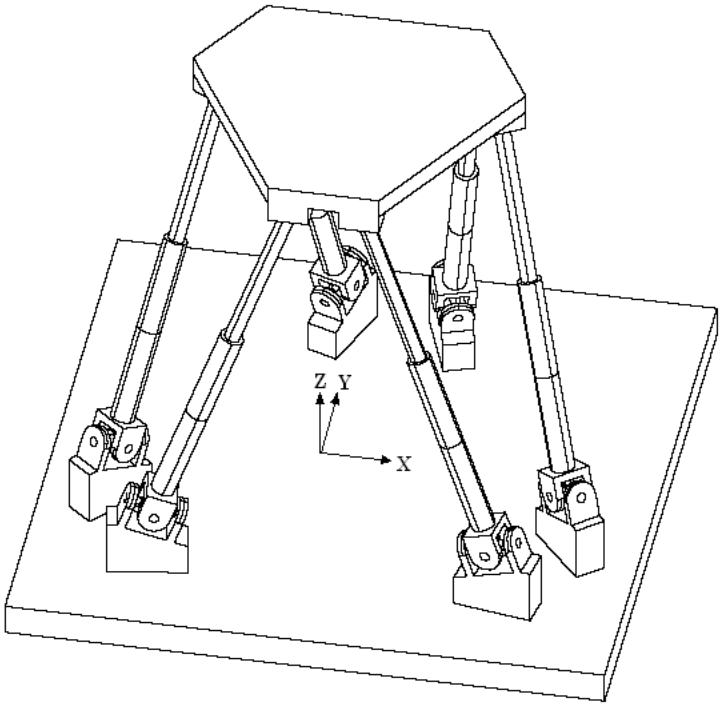


Figure A.19: The Stewart-Gough platform.

Table A.30: Points, Relative to the Ground Frame, Where Each of the Stewart-Gough Platform's Leg Attaches to the Ground

	X (m)	Y (m)	Z (m)
Leg 1	0.517638090	1.931851653	0.0
Leg 2	1.414213562	-1.414213562	0.0
Leg 3	-1.931851653	-0.517638090	0.0
Leg 4	-0.517638090	1.931851653	0.0
Leg 5	1.931851653	-0.517638090	0.0
Leg 6	-1.414213562	-1.414213562	0.0

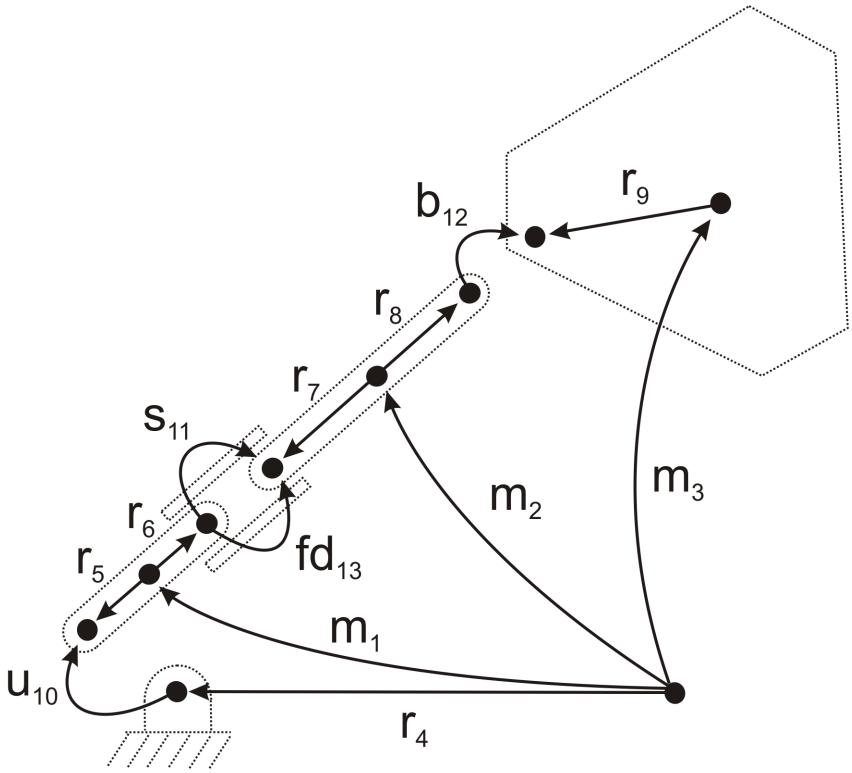


Figure A.20: The graph for one of the Stewart-Gough platform's legs.

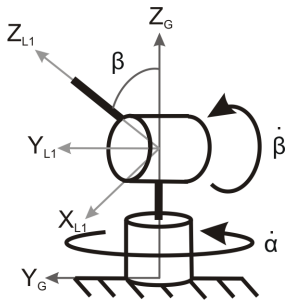


Figure A.21: The axes of rotation of each universal joint.

Table A.31: Points, Relative to the Platform's Center of Mass Frame, Where Each of the Stewart-Gough Platform's Leg Attaches to the Platform

	x (m)	y (m)	z (m)
Leg 1	0.707106781	0.707106781	0.0
Leg 2	0.258819045	-0.965925826	0.0
Leg 3	-0.965925826	0.258819045	0.0
Leg 4	-0.707106781	0.707106781	0.0
Leg 5	0.965925826	0.258819045	0.0
Leg 6	-0.258819045	-0.965925826	0.0

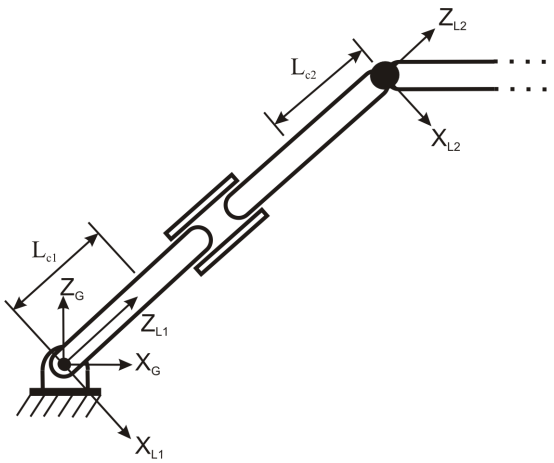


Figure A.22: Local reference frames on the legs of the SGP.

where the term *Shank Bodies* refers to the bodies connected to the ground via a universal joint while the term *Thigh Bodies* refer to the bodies connected to the platform via a spherical joint. It also gives the values of the mass and moment of inertia of each body.

Table A.32: Body Lengths and Inertia Properties of the Stewart-Gough Platform

	mass (kg)	Center of mass in local frame (m)			Moments of Inertia ($kg \times m^2$)		
		x	y	z	I_{xx}	I_{yy}	I_{zz}
Shank Bodies	2	0.0	0.0	0.4144	0.1145	0.1145	0.0
Thigh Bodies	1	0.0	0.0	-0.4144	0.0537	0.0537	0.0
Platform	10	0.0	0.0	0.0000	5.0000	5.0000	10.0

The manipulator is initially at rest with the platform in a horizontal and centered position at a height of 3 m. The orientation of the platform is measured with Euler 123 angles. The forces in the actuators are given in Table A.33, and gravity is acting in the global negative Z direction.

Table A.33: Forces Exerted by the Prismatic Actuators of the Stewart-Gough Platform

Leg	1	2	3	4	5	6
Force (N)	28	$28 + 3\sin(4\pi t)$	$28 + 3\sin(4\pi t)$	$28 + 5\sin(2\pi t)$	$28 + 5\sin(2\pi t)$	28

The Stewart-Gough Platform is simulated using various tree sets. For every different tree set the kinematic equations were twice differentiated and Baumgarte stabilization [2] was applied to the resulting ODEs with the Baumgarte parameters α and β set to 10 and 5, respectively. The resulting set of equations where then sent to simplification and code optimization routines in Maple. The resulting sets of ODEs were solved for a 2 s simulation using a Runge-Kutta Fehlberg method with an absolute error of 10^{-6} and a relative error

of 10^{-5}. The results, which are taken from an average of twenty simulations, are shown in Table A.34. A Pentium 4 of 1.80 GHz with 768 MB of RAM was used to perform these simulations.

In the description of the various trees, the arm elements were not presented; however they are always found in both trees. The notation u, b, s, and m_2 represents placing all the universal joints, spherical joints, prismatic joints, and top leg sections in the tree respectively. The notations $b(1)$ and $s(1)$ represent placing just one of the six spherical joints or prismatic joints in the tree respectively. The term m_3 represents placing the platform's body element in the tree. The notation *Same Tree* for the translational tree indicates that this tree is composed of the same edges as the rotational tree. The equation complexity refers to the one described in Section 2.4.

Table A.34: Efficiency of Various Tree Sets of the Stewart-Gough Platform

Rotational Tree	Translational Tree	Solution Time (s)	Complexity	# Dyn. EQs	# Kin. EQs
u, s, m_3	u, b, m_3	6.72	59428	18	12
u, s, m_3	u, m_2, m_3	2.86	24678	36	30
u, s, m_3	*$u, b, s(1)$*	*7.16*	*55202*	*16*	*10*
u, s, m_3	**$u, b(1), s$**	**3.24**	**25010**	**21**	**15**
u, s, m_3	Same Tree	2.52	21303	24	18
u, s, m_3	s, m_2, m_3	7.93	64147	42	36

The tree set selected by the Variable-Based Tree Selection Algorithm is depicted in italics in Table A.34 and the tree set selected by the Formulation-Based Tree Selection Algorithm is depicted in bold in this table. The Variable-Based Tree Selection Algorithm found its tree set in an average of 17 milliseconds while the Formulation-Based Tree Selection Algorithm took an average of 281 milliseconds to find its tree set.

To verify if similar results could be obtained using other solvers, the system's equations were exported into Matlab S-functions. This Matlab S-function was then solved in Mat-

lab/Simulink using various ODE solvers and using the same parameters as those used in Maple. The results of these simulations are presented in Table A.13. Here, the term *ODE45* refers to the simulations solved using Matlab's ODE45, which is based on a Runge-Kutta formula called the Dormand-Prince pair. The term *ODE113* refers to the simulations solved using Matlab's ODE113, which is an Adams-Bashforth-Moulton PECE (predictor-evaluate-corrector-evaluate) solver. Finally, the term *ODE15s* refers to the simulations solved using Matlab's ODE15s, which is a stiff solver based on numerical differentiation formulas (NDFs). The term "–" refers to the fact that the solver did not converge to a solution and hence was unable to find valid results with the given ODE solver.

Table A.35: Efficiency of Various Tree Sets of the Stewart-Gough Platform Using Other Solvers

Rotational Tree	Translational Tree	ODE45 (s)	ODE113 (s)	ODE15s (s)
$\mathrm{u, s, m_3}$	$\mathrm{u, m_2, m_3}$	0.15	0.15	0.35
u, s, m3	*u, b, s(1)*	*2.02*	*2.50*	–
u, s, m3	**u, b(1), s**	**0.15**	**0.14**	**0.30**
$\mathrm{u, s, m_3}$	Same Tree	0.14	0.13	0.28

A.12 3-DOF Spatial Parallel Manipulator

The 3-DOF spatial parallel manipulator is presented in Figure A.23, and its graph is presented in Figure A.24. It consists of a platform connected to the ground by three similar legs and has a tool placed on top of the platform. This Manipulator was developed by Simoni Michele for his diploma thesis at "L'Ecole d'ingénieurs et d'architectes de Fribourg" under the supervision of Professor Lui Jean-Piree Schmitt [28]. The geometry and inertia properties used in this report are based on Simoni Michele's preliminary configuration of his 3-DOF Spatial Parallel Manipulator.

The configuration of one of the mechanism's three legs is depicted in figure A.25, where

the reference frames of each body is placed at its center of mass. This figure also gives the dimensions of each of the leg's bodies. These dimensions are the same for all three of the mechanism's legs. The legs are each connected to the ground by prismatic joints (s_{25}, s_{28}, and s_{31}), which originate from the global reference frame. The first prismatic joint is oriented toward the global X axis while the two others are oriented 120° and 240° away about the global Z axis, respectively. The shank bodies (m_1, m_3, and m_5) are connected to the thigh bodies (m_2, m_4, and m_6) with spherical joints (b_{26}, b_{29}, and b_{32}). The thigh bodies are connected to the platform with two revolute joints. Since one of the two revolute joints is redundant, it is neglected when modeling the system. Because of this, only one revolute joint is considered for each leg in the system's graph (h_{27}, h_{30}, and h_{33}). The revolute joints are connected to the platform at 0°, 120°, and 240° about the platform's local Z axis. The orientation of the revolute joints' axes of rotation are 90°, 210°, and 330° away from the platform's local X axis, with these rotations being expressed about the platform's local Z axis, respectively.

The inertia properties of the mechanism's bodies are given in Table A.36,where the moments of inertia of each body are expressed about this center of mass at the centroid of each body. The manipulator's initial conditions are presented in Table A.37, where S_{25}, S_{28}, and S_{31} represent the translation of the prismatic joints s_{25}, s_{28}, and s_{31}, respectively. The terms X_7, Y_7, and Z_7 represent the position of the platform relative to the ground reference frame's X, Y and Z axes. The terms ζ_7, η_7, and ξ_7 are the Euler 123 angles of rotation of the platform relative to the ground reference frame.

The forces depicted in Table A.38 are placed on the prismatic joints of the system. The gravity is given as 9.81 m/s^2 and is directed in the global negative Z direction. This mechanism has three DOF.

The 3-DOF Spatial Parallel Manipulator is simulated using various tree sets. For every different tree set the kinematic equations were twice differentiated and Baumgarte stabilization [2] was applied to the resulting ODEs with the Baumgarte parameters α and β set to 10 and 5, respectively. The resulting set of equations where then sent to simplification and code optimization routines in Maple. The resulting sets of ODEs were solved for a 1 s simulation using a Runge-Kutta Fehlberg method with an absolute error of 10^{-6} and a relative error of 10^{-5}. The results, which are taken from an average of twenty

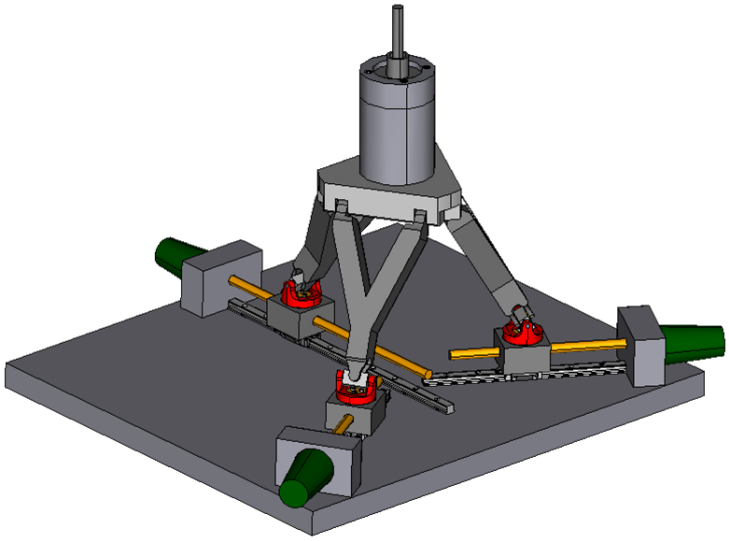


Figure A.23: 3-DOF spatial parallel manipulator.

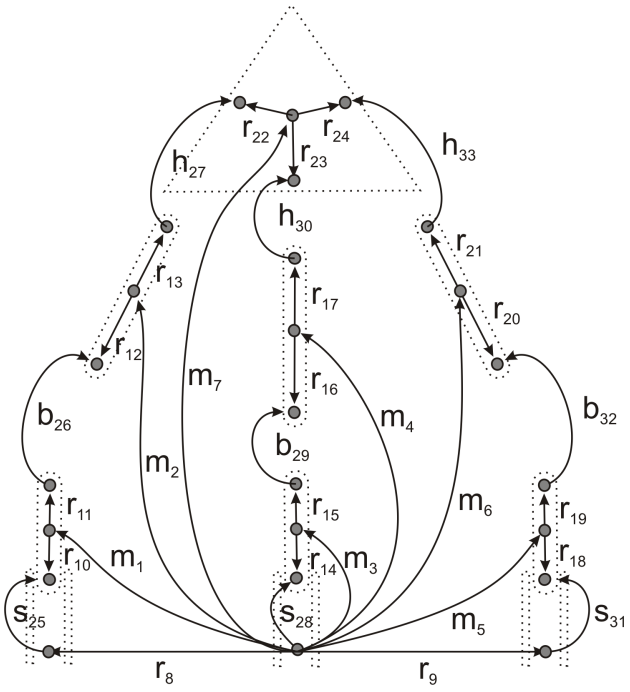


Figure A.24: 3-DOF spatial parallel manipulator's graph.

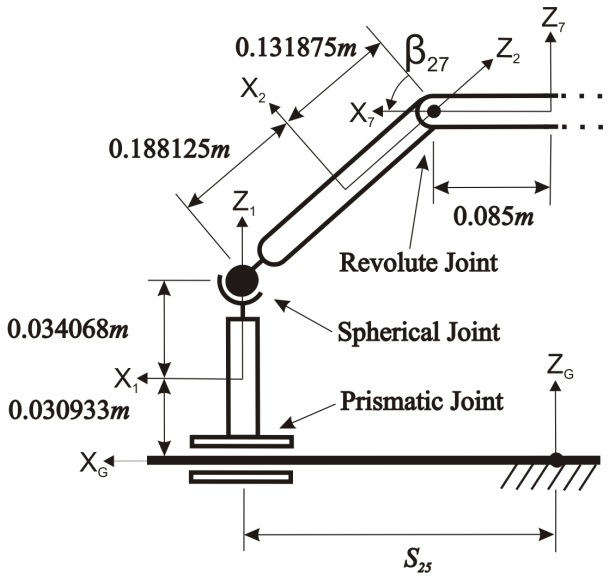


Figure A.25: The configuration of one of the mechanism's leg.

Table A.36: Inertia Properties of the 3-DOF Spatial Parallel Manipulator

	Mass	Moments of Inertia ($kg \times m^2$)					
	(kg)	I_{xx}	I_{yy}	I_{zz}	I_{xy}	I_{xz}	I_{yz}
Shank Bodies	3.06876	0.003138	0.002853	0.003819	0	0.000237	0
Thigh Bodies	7.35236	0.053655	0.040571	0.015956	0	0	0
Platform	21.8122	0.220751	0.220763	0.170223	0	0	0

Table A.37: Initial Conditions of the 3-DOF Spatial Parallel Manipulator

Coordinates	$S_{25}, S_{28}, S_{31}\,(m)$	$x_7\,(m)$	$y_7\,(m)$	$z_7\,(m)$	$\zeta_7, \eta_7, \xi_7\,(rad)$
Initial Value	0.245	0	0	0.4226	0

Table A.38: Forces at the 3-DOF Spatial Parallel Manipulator's Prismatic Joints

Prismatic Joints	s_{25}	s_{28}	s_{31}
Forces (N)	$65 + 5.0\cos(2\pi t)$	$65 + 5.0\sin(2\pi t)$	$65 + 3.0\sin(4\pi t)$

simulations, are shown in Table A.39. A Pentium 4 of 1.80 GHz with 768 MB of RAM was used to perform these simulations. In the description of the various trees, the arm elements were not presented; however they are always found in both trees. The notation *Same Tree* for the translational tree indicates that this tree is composed of the same edges as the rotational tree. The term *Comp.* denotes the equation complexity that is described in Section 2.4. The notation "–" for the simulation time and complexity of a mechanism model indicates that DynaFlexPro could not generate the equations of the system within 30 minutes when the system was modelled using the given set of trees.

The tree set selected by the Variable-Based Tree Selection Algorithm is depicted in italics in Table A.39 and the tree set selected by the Formulation-Based Tree Selection Algorithm is depicted in bold in this table. The Variable-Based Tree Selection Algorithm found its tree set in an average of 17 milliseconds while the Formulation-Based Tree Selection Algorithm took an average of 63 milliseconds to find its tree set.

To verify if similar results could be obtained using other solvers, the system's equations were exported into Matlab S-functions. This Matlab S-function was then solved in Matlab/Simulink using various ODE solvers and using the same parameters as those used in Maple. The results of these simulations are presented in Table A.13. Here, the term *ODE45* refers to the simulations solved using Matlab's ODE45, which is based on a Runge-Kutta formula called the Dormand-Prince pair. The term *ODE113* refers to the simulations solved using Matlab's ODE113, which is an Adams-Bashforth-Moulton PECE (predictor-evaluate-corrector-evaluate) solver. Finally, the term *ODE15s* refers to the simulations solved using Matlab's ODE15s, which is a stiff solver based on numerical differentiation formulas (NDFs).

Table A.39: Efficiency of Various Tree Sets of the 3-DOF Spatial Parallel Manipulator

Rotational Tree	Translational Tree	Sol. Time (s)	Comp.	# Dyn. EQs	# Kin. EQs
$s_{25}, s_{28}, s_{31}, b_{26}, b_{29}, b_{32}, m_7$	$s_{25}, b_{26}, b_{29}, b_{32}, h_{27}, h_{30}, h_{33}$	3.98	48178	13	10
$s_{25}, s_{28}, s_{31}, b_{26}, b_{29}, b_{32}, m_7$	$s_{25}, s_{28}, s_{31}, b_{26}, b_{29}, b_{32}, h_{27}$	2.55	30013	15	12
$s_{25}, s_{28}, s_{31}, b_{26}, b_{29}, b_{32}, m_7$	Same Tree	2.54	31213	18	15
$s_{25}, s_{28}, s_{31}, b_{26}, b_{29}, b_{32}, m_7$	$s_{25}, b_{29}, b_{32}, h_{27}, h_{30}, h_{33}, m_7$	4.32	51346	16	13
$b_{26}, b_{29}, b_{32}, h_{27}, h_{30}, h_{33}, m_7$	$s_{25}, s_{28}, s_{31}, b_{26}, b_{29}, b_{32}, m_7$	–	–	21	18
$s_{25}, s_{28}, s_{31}, b_{26}, b_{29}, b_{32}, h_{27}$	$s_{25}, s_{28}, s_{31}, b_{26}, b_{29}, b_{32}, m_7$	3.73	41233	16	13
$s_{25}, s_{28}, s_{31}, h_{27}, h_{30}, h_{33}, m_7$	*$s_{25}, b_{26}, b_{29}, b_{32}, h_{27}, h_{30}, h_{33}$*	*6.14*	*66060*	*7*	*4*
$s_{25}, s_{28}, s_{31}, h_{27}, h_{30}, h_{33}, m_7$	$s_{25}, s_{28}, s_{31}, b_{26}, b_{29}, b_{32}, h_{27}$	4.11	43203	9	6
$s_{25}, s_{28}, s_{31}, h_{27}, h_{30}, h_{33}, m_7$	$s_{25}, s_{28}, s_{31}, b_{26}, b_{29}, b_{32}, m_7$	4.11	43356	12	9
$s_{25}, s_{28}, s_{31}, h_{27}, h_{30}, h_{33}, m_7$	Same Tree	6.02	68210	12	9
$s_{25}, s_{28}, s_{31}, h_{27}, m_2, m_4, m_6$	$s_{25}, s_{28}, s_{31}, m_2, m_4, m_6, m_7$	4.35	50181	22	19
$s_{25}, s_{28}, s_{31}, m_2, m_4, m_6, m_7$	**$s_{25}, s_{28}, s_{31}, b_{26}, b_{29}, b_{32}, m_7$**	**2.34**	**26595**	**15**	**12**

Table A.40: Efficiency of Various Tree Sets of the 3-DOF Spatial Parallel Manipulator Using Other Solvers

Rotational Tree	Translational Tree	ODE45 (s)	ODE113 (s)	ODE15s (s)
$s_{25}, s_{28}, s_{31}, b_{26}, b_{29}, b_{32}, m_7$	$s_{25}, s_{28}, s_{31}, b_{26}, b_{29}, b_{32}, h_{27}$	0.53	0.31	0.89
$s_{25}, s_{28}, s_{31}, b_{26}, b_{29}, b_{32}, m_7$	Same Tree	0.41	0.27	0.71

Appendix B

Terminal Equations

The terminal equations associated with four of the most commonly-used components are presented in Tables B.1 - B.4. In these equations, the subscripts on unit vectors and rotation matrices labeled i indicate that they can be expressed in any reference frame. In the terminal equation for the body element, the term $\vec{r}$ represents the given edge's translational displacement vector with $\widehat{\imath}_i$, $\widehat{\jmath}_i$ and $\widehat{k}_i$ representing unit vectors about the X, Y and Z axes of an arbitrary coordinate system, respectively. The terms $\vec{v}$ and $\vec{a}$ represent the given edge's translational velocity vector and translational acceleration vector, respectively. The term m represents the body's mass and the term $[R]$ represents a rotation matrix with the three rotation values indicated by $\theta_1(t)$, $\theta_2(t)$, and $\theta_3(t)$. The terms $\vec{\omega}$ and $\vec{\alpha}$ refer to the rotational velocity vector and the rotational acceleration vector of the given edge, respectively. In the body element's torque, $\vec{\vec{J}}$ represents the inertia dyadic of the body. The summation $\sum \vec{r}_r \times \vec{F}_r$, where $\vec{r}_r$ and $\vec{F}_r$ refer to the displacement and force associated with a rigid arm, takes into account the torques generated by each arm attached to the body. In a similar fashion, the summation $\sum \vec{r}_t \times \vec{F}_t$ refers to the torques generated by any edge that allows translation and originates from the given body or one of its arms. The terms $\vec{\omega}_m$ and $\vec{\alpha}_m$ denotes the rotational velocity and acceleration of the body to which the component originates. Finally, the vectors $\widehat{u}$ and $\widehat{n}$ represent a component's joint axes and reaction axes, respectively.

Table B.1: Terminal Equations for a Rigid Body Element

	Translation	
$\vec{r}$	=	$r_x(t)\widehat{\imath}_i + r_y(t)\widehat{\jmath}_i + r_z(t)\widehat{k}_i$
$\vec{v}$	=	$\frac{d}{dt}\vec{r}$
$\vec{a}$	=	$\frac{d}{dt}\vec{v}$
$\vec{F}$	=	$-m\vec{a}$
	Rotation	
$[R]$	=	$[R(\ \theta_1(t),\ \theta_2(t),\ \theta_3(t)\ )]_i$
$\vec{\omega}$	=	$\omega_x\widehat{\imath}_i + \omega_y\widehat{\jmath}_i + \omega_z\widehat{k}_i$
$\vec{\alpha}$	=	$\frac{d}{dt}\vec{\omega}$
$\vec{T}$	=	$-\vec{\vec{J}}\cdot\vec{\alpha} - \vec{\omega}\times\vec{\vec{J}}\cdot\vec{\omega} - \sum\vec{r}_r\times\vec{F}_r - \sum\vec{r}_t\times\vec{F}_t$

Table B.2: Terminal Equations for an Rigid Arm Element

	Translation	
$\vec{r}$	=	$r_x\widehat{\imath}_i + r_y\widehat{\jmath}_i + r_z\widehat{k}_i$
$\vec{v}$	=	$\vec{\omega}_m\times\vec{r}$
$\vec{a}$	=	$\vec{\alpha}_m\times\vec{r} + \vec{\omega}_m\times(\vec{\omega}_m\times\vec{r})$
$\vec{F}$	=	$F_x(t)\widehat{\imath}_i + F_y(t)\widehat{\jmath}_i + F_z(t)\widehat{k}_i$
	Rotation	
$[R]$	=	$[R(\theta_1,\theta_2,\theta_3)]_i$
$\vec{\omega}$	=	$\vec{0}$
$\vec{\alpha}$	=	$\vec{0}$
$\vec{T}$	=	$T_x(t)\widehat{\imath}_i + T_y(t)\widehat{\jmath}_i + T_z(t)\widehat{k}_i$

Table B.3: Terminal Equations for a Revolute Joint

Translation		
$\vec{r}$	=	$\vec{0}$
$\vec{v}$	=	$\vec{0}$
$\vec{a}$	=	$\vec{0}$
$\vec{F}$	=	$F_x(t)\widehat{\imath}_i + F_y(t)\widehat{\jmath}_i + F_z(t)\widehat{k}_i$
Rotation		
$[R]$	=	$[R(\ \theta(t)\ , \widehat{u})]_i$
$\vec{\omega}$	=	$\omega(t)\widehat{u}$
$\vec{\alpha}$	=	$\frac{d}{dt}\vec{\omega}$
$\vec{T}$	=	$T_1(t)\widehat{n1}_i + T_2(t)\widehat{n2}_i$

Table B.4: Terminal Equations for a Prismatic Joint

Translation		
$\vec{r}$	=	$s\widehat{u}_i$
$\vec{v}$	=	$\dot{s}\widehat{u}_i + \vec{\omega}_m \times \vec{r}$
$\vec{a}$	=	$\ddot{s}\widehat{u}_i + \dot{\vec{\omega}}_m \times \vec{r} + 2(\vec{\omega}_m \times \dot{s}\widehat{u}_i) + \vec{\omega}_m \times (\vec{\omega}_m \times \vec{r})$
$\vec{F}$	=	$F_1(t)\widehat{n1}_i + F_2(t)\widehat{n2}_i$
Rotation		
$[R]$	=	$[I]$
$\vec{\omega}$	=	$\vec{0}$
$\vec{\alpha}$	=	$\vec{0}$
$\vec{T}$	=	$T_x(t)\widehat{\imath}_i + T_y(t)\widehat{\jmath}_i + T_z(t)\widehat{k}_i$

Appendix C

Number of Branch Coordinates

Table C.1: Number of Branch Coordinates Associated to Each Edge Type

Edge Type	Translation	Rotation
Revolute Joint	0	1
Prismatic Joint	1	0
Universal Joint	0	2
Spherical Joint	0	3
Planar Joint	2	1
Cylindrical Joint	1	1
Weld Joint (0-DOF)	0	0
Free Joint (6-DOF)	3	3
XYZ Translational Joint	3	0
Force Driver	3	3
Moment Driver	3	3
Motion Driver	0	0
Rigid Body	3	3
Arm Element	0	0
Flexible Body	3	3
Flexible Arm	0	0
Translational SDA	3	3
Rotational SDA	3	3

Appendix D

Prim's Algorithm and Dijkstra's Algorithm

Prim's Algorithm

- Create a list of all the edges in the graph and their incident nodes
- Place the origin node(in this case the ground node) of the graph in the tree
- Repeat the following steps until all the nodes are found in the tree:
  - Find all the potential edges (edges that connect to one node in the present tree and one node not in the present tree)
  - From these potential edges, find the edge with the lowest weight
  - Add this edge and its new node to the tree

Figure D.1: Pseudo-code of Prim's algorithm.

Dijkstra's Algorithm

- Create a list of all the edges in the graph and their incident nodes
- Create a table listing the SPW of each node and initiate it by giving a SPW of zero to the ground node and infinity to all other nodes
- Place the origin node(in this case the ground node) of the graph in the tree
- Repeat the following steps until all the nodes are found in the tree:
  - Find all the potential edges (edges that connect to one node in the present tree (T node) and one node not in the present tree (N node))
  - For each of the potential edges, find the SPW of its N node:
    N node SPW = Edge Weight + SPW of the edge's T node
  - From these potential edges, find the edge with the lowest N node SPW
  - Enter this edge's N node SPW to the SPW table
  - Add this edge and its new node to the tree

Figure D.2: Pseudo-code of Dijkstra's algorithm.

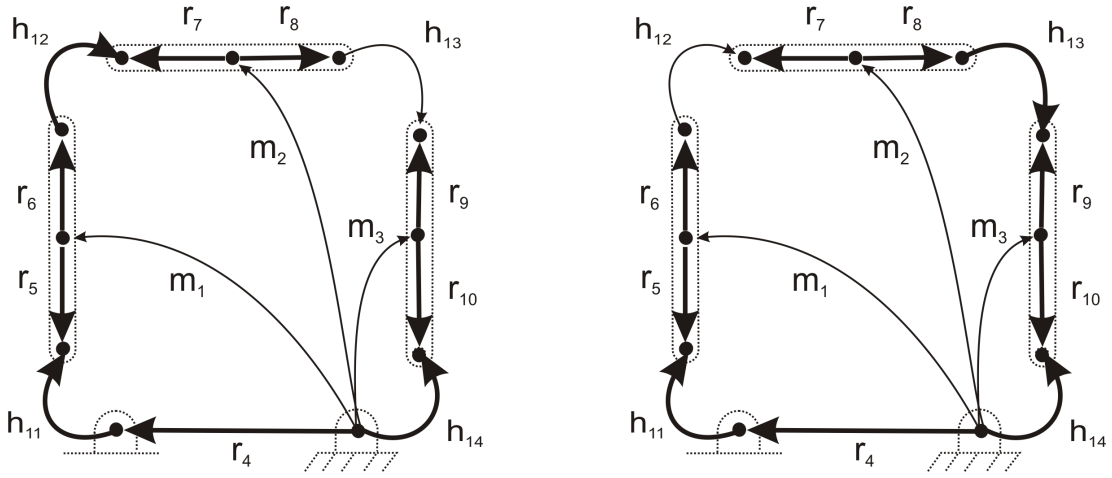
Appendix E

Tree Similarity Heuristic Demonstration

In order to better illustrate the Tree Similarity Heuristic of Section 4.1.6, let us look at the example of a planar four bar mechanism consisting of three bodies connected to each other and the ground with revolute joints. The graph of this four bar mechanism is presented in Figure E.1, where the angle of revolute joints h_{11}, h_{12}, and h_{13} are going to be represented by β_1, β_2, and β_3 respectively. Such a model will have two translational trees and two rotational trees satisfying all the heuristics of the Formulation-Based Tree Selection (except the Tree Similarity Heuristic). These two trees, called *Tree A* and *Tree B*, which are similar in rotation and translation, are presented in bold within the four-bar mechanism's graph in Figure E.1.

The equation formulation process for the case in which both the translational and rotational trees take the form of the *Tree A* is presented below.

First, step I of the equation formulation process provides the three basic dynamic equations. The terminal equations of each of the torques presented in these basic equations are then substituted in the equations and simplified knowing that the mechanism is planar, giving the equations (E.1) to (E.3). In order to further clarify the substitution process that will be executed during the equation formulation process, the secondary variables found in the equations are depicted in bold.



(a) *Tree A* (b) *Tree B*

Figure E.1: Two trees of the four-bar mechanism satisfying all rotational and translational tree heuristics.

$$(-Izz_1 \cdot \vec{\boldsymbol{\alpha}}_{\mathbf{1}} - l_5\widehat{\imath}_1 \times \vec{\boldsymbol{F}}_{\mathbf{5}} - l_6\widehat{\imath}_1 \times \vec{\boldsymbol{F}}_{\mathbf{6}} - Izz_2 \cdot \vec{\boldsymbol{\alpha}}_{\mathbf{2}} - l_7\widehat{\imath}_2 \times \vec{\boldsymbol{F}}_{\mathbf{7}} - l_8\widehat{\imath}_2 \times \vec{\boldsymbol{F}}_{\mathbf{8}} - \vec{T}_{13}) \cdot \widehat{k}_{11} = 0 \quad \text{(E.1)}$$

$$(-Izz_2 \cdot \vec{\boldsymbol{\alpha}}_{\mathbf{2}} - l_7\widehat{\imath}_2 \times \vec{\boldsymbol{F}}_{\mathbf{7}} - l_8\widehat{\imath}_2 \times \vec{\boldsymbol{F}}_{\mathbf{8}} - \vec{T}_{13}) \cdot \widehat{k}_{12} = 0 \quad \text{(E.2)}$$

$$(-Izz_3 \cdot \vec{\boldsymbol{\alpha}}_{\mathbf{3}} - l_9\widehat{\imath}_3 \times \vec{\boldsymbol{F}}_{\mathbf{9}} - l_{10}\widehat{\imath}_3 \times \vec{\boldsymbol{F}}_{\mathbf{10}} + \vec{T}_{13}) \cdot \widehat{k}_{14} = 0 \quad \text{(E.3)}$$

where Izz_i represents the moment of inertia about the line through the centroid, parallel to the Z axis of the body represented by the edge m_i, $\vec{\alpha}_i$ represents the rotational acceleration vector of the i^{th} edge. The term $\vec{T}_i$ represents the torque vector of the i^{th} edge. The terms l_i represent the length of the arm element r_i, while the term $\vec{F}_i$ represents the force vector associated to the i^{th} edge. The terms $\widehat{\imath}_i$ and $\widehat{k}_i$ represent the unit vector along the the i^{th} edge's end node's local X and Z axes, respectively.

The first part of Step III is then performed, giving the equations (E.4) to (E.9) that shall be used to replace the arm forces in the basic dynamic equations.

$$\vec{\boldsymbol{F}}_{\mathbf{5}} = -m_1\vec{\boldsymbol{a}}_{\mathbf{1}} - m_2\vec{\boldsymbol{a}}_{\mathbf{2}} - \vec{F}_{13} \quad \text{(E.4)}$$

$$\vec{\boldsymbol{F}}_{\mathbf{6}} = m_2\vec{\boldsymbol{a}}_{\mathbf{2}} + \vec{F}_{13} \quad \text{(E.5)}$$

$$\vec{\boldsymbol{F}}_{\mathbf{7}} = -m_2\vec{\boldsymbol{a}}_{\mathbf{2}} - \vec{F}_{13} \quad \text{(E.6)}$$

$$\vec{\boldsymbol{F}}_{\mathbf{8}} = \vec{F}_{13} \quad \text{(E.7)}$$

$$\vec{\boldsymbol{F}}_{\mathbf{9}} = -\vec{F}_{13} \quad \text{(E.8)}$$

$$\vec{\boldsymbol{F}}_{\mathbf{10}} = -m_3\vec{\boldsymbol{a}}_{\mathbf{3}} + \vec{F}_{13} \quad \text{(E.9)}$$

where m_i represent the mass of the body depicted by the edge m_i and $\vec{a}_i$ represents the translational acceleration vector of the i^{th} edge.

The second part of Step III provides the acceleration equations (E.10) to (E.12) that will be substituted in the present basic dynamic equations. Though not depicted, the velocity and position equations for the body edges m_1, m_2, and m_3 are also obtained in this substitution step by using the same methods applied to acceleration.

$$\vec{\boldsymbol{a}}_{\mathbf{1}} = -\vec{\boldsymbol{\alpha}}_{\mathbf{1}} \times l_5\widehat{\imath}_1 + \vec{\boldsymbol{\omega}}_{\mathbf{1}} \times (\vec{\boldsymbol{\omega}}_{\mathbf{1}} \times l_5\widehat{\imath}_1) - l_4\widehat{\imath}_G \quad \text{(E.10)}$$

$$\vec{\boldsymbol{a}}_{\mathbf{2}} = -\vec{\boldsymbol{\alpha}}_{\mathbf{1}} \times (l_5 - l_6)\widehat{\imath}_1 + \vec{\boldsymbol{\omega}}_{\mathbf{1}} \times (\vec{\boldsymbol{\omega}}_{\mathbf{1}} \times (l_5 - l_6)\widehat{\imath}_1) - \vec{\boldsymbol{\alpha}}_{\mathbf{2}} \times l_7\widehat{\imath}_2 + \vec{\boldsymbol{\omega}}_{\mathbf{2}} \times (\vec{\boldsymbol{\omega}}_{\mathbf{2}} \times l_7\widehat{\imath}_2) - l_4\widehat{\imath}_G \quad \text{(E.11)}$$

$$\vec{\boldsymbol{a}}_{\mathbf{3}} = -\vec{\boldsymbol{\alpha}}_{\mathbf{3}} \times l_{10}\widehat{\imath}_3 + \vec{\boldsymbol{\omega}}_{\mathbf{3}} \times (\vec{\boldsymbol{\omega}}_{\mathbf{3}} \times l_{10}\widehat{\imath}_3) \quad \text{(E.12)}$$

where $\vec{\omega}_i$ represent the rotational velocity vector of the i^{th} edge and the subscript G refers to the ground reference frame.

The third part of Step III provides the rotational velocity equations (E.13) to (E.15) that will also be substituted in the present basic dynamic equations. As with the previous step, the rotational velocities and rotational accelerations can be obtained using the same procedure.

$$\vec{\boldsymbol{\omega}}_{\mathbf{1}} = \frac{d}{dt}\beta_1(t)\widehat{k}_{11} \tag{E.13}$$

$$\vec{\boldsymbol{\omega}}_{\mathbf{2}} = \frac{d}{dt}\beta_1(t)\widehat{k}_{11} + \frac{d}{dt}\beta_2(t)\widehat{k}_{12} \tag{E.14}$$

$$\vec{\boldsymbol{\omega}}_{\mathbf{3}} = \frac{d}{dt}\beta_3(t)\widehat{k}_{G} \tag{E.15}$$

Step IV will evaluate all the dot products in the kinematic and dynamic equations. In order to keep this example as simple and short as possible, we will stop here, without writing out the final system's equations, and take a look at the results already obtained. If one follows through the substitution process, it becomes clear that the first dynamic equation will be a function of β_1 and β_2 as well as their derivatives. The second dynamic equation will be a function of the same variables, and the third dynamic equation will be a function of β_1 and its derivatives. Finally, though the kinematic equations were not included in the previous analysis, we can easily see that they will be a function of all three modelling variables.

Now let us consider what would happen if the translational tree would take the form of *Tree B* (presented in Figure E.1) while the rotational tree remained in the form of *Tree A*. In this case, since the rotational tree remains unchanged, the three dynamic equations obtained at Step I would remain the equations (E.1) to (E.3). The acceleration equations obtained by the first part of Step III however would change to the following equations:

$$\vec{\boldsymbol{F}}_{\mathbf{5}} = -m_1\vec{\boldsymbol{a}}_{\mathbf{1}} - \vec{F}_{12} \tag{E.16}$$

$$\vec{\boldsymbol{F}}_{\mathbf{6}} = \vec{F}_{12} \tag{E.17}$$

$$\vec{\boldsymbol{F}}_{\mathbf{7}} = -\vec{F}_{12} \tag{E.18}$$

$$\vec{\boldsymbol{F}}_{\mathbf{8}} = -m_2\vec{\boldsymbol{a}}_{\mathbf{2}} + \vec{F}_{12} \tag{E.19}$$

$$\vec{\boldsymbol{F}}_{\mathbf{9}} = m_2\vec{\boldsymbol{a}}_{\mathbf{2}} - \vec{F}_{12} \tag{E.20}$$

$$\vec{\boldsymbol{F}}_{\mathbf{10}} = -m_2\vec{\boldsymbol{a}}_{\mathbf{2}} - m_3\vec{\boldsymbol{a}}_{\mathbf{3}} + \vec{F}_{12} \tag{E.21}$$

The second part of Step III would become:

$$\vec{\boldsymbol{a}}_{\mathbf{1}} = -\vec{\boldsymbol{\alpha}}_{\mathbf{1}} \times l_5\widehat{\imath_1} + \vec{\boldsymbol{\omega}}_{\mathbf{1}} \times (\vec{\boldsymbol{\omega}}_{\mathbf{1}} \times l_5\widehat{\imath_1}) - l_4\widehat{\imath_G} \tag{E.22}$$

$$\vec{\boldsymbol{a}}_{\mathbf{2}} = -\vec{\boldsymbol{\alpha}}_{\mathbf{2}} \times l_8\widehat{\imath_2}\vec{\boldsymbol{\omega}}_{\mathbf{2}} \times (\vec{\boldsymbol{\omega}}_{\mathbf{2}} \times l_8\widehat{\imath_2}) - \vec{\boldsymbol{\alpha}}_{\mathbf{3}} \times (l_9 - l_{10})\widehat{\imath_3} + \vec{\boldsymbol{\omega}}_{\mathbf{3}} \times (\vec{\boldsymbol{\omega}}_{\mathbf{3}} \times (l_9 - l_{10})\widehat{\imath_3}) \tag{E.23}$$

$$\vec{\boldsymbol{a}}_{\mathbf{3}} = -\vec{\boldsymbol{\alpha}}_{\mathbf{3}} \times l_{10}\widehat{\imath_3} + \vec{\boldsymbol{\omega}}_{\mathbf{3}} \times (\vec{\boldsymbol{\omega}}_{\mathbf{3}} \times l_{10}\widehat{\imath_3}) \tag{E.24}$$

Again, since the rotational tree remains the same, the equations used in the last part of the substitution process of Step III also remain the same as those presented in equations (E.13) to (E.15).

Now, if one follows through the substitution and dot products of this new tree combination, it can be seen that the first dynamic equation is now expressed relative to all three of the modelling variables. The second and third dynamic equations are now expressed relative to β_2 and β_3 as well as their derivatives. Finally, the kinematic equations are still expressed relative to all three modelling variables.

It is clear that the first and third dynamic equations of the model that used different trees are each expressed relative to a wider range of variables than these same equations generated by the system modelled with similar trees. As mentioned in Section 4.1.6, the added diversity of modelling variables present in the system modelled with two different trees will result in the system's equations to be more coupled, which allows for less simplifications and code optimization to be performed on the system's equations, thus reducing the efficiency of the model's simulations. The full tree selection results for this benchmark problem can be found in Appendix A.1.