Real-time Dynamic Simulation of Constrained Multibody Systems using Symbolic Computation

by

Thomas Kenji Uchida

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2011

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.

Abstract

The main objective of this research is the development of a framework for the automatic generation of systems of kinematic and dynamic equations that are suitable for real-time applications. In particular, the efficient simulation of constrained multibody systems is addressed. When modelled with ideal joints, many mechanical systems of practical interest contain closed kinematic chains, or kinematic loops, and are most conveniently modelled using a set of generalized coordinates of cardinality exceeding the degrees-of-freedom of the system. Dependent generalized coordinates add nonlinear algebraic constraint equations to the ordinary differential equations of motion, thereby producing a set of differential-algebraic equations that may be difficult to solve in an efficient yet precise manner. Several methods have been proposed for simulating such systems in real time, including index reduction, model simplification, and constraint stabilization techniques.

In this work, the equations of motion are formulated symbolically using linear graph theory. The embedding technique is applied to eliminate the Lagrange multipliers from the dynamic equations and obtain one ordinary differential equation for each independent acceleration. The theory of Gröbner bases is then used to triangularize the kinematic constraint equations, thereby producing recursively solvable systems for calculating the dependent generalized coordinates given values of the independent coordinates. For systems that can be fully triangularized, the kinematic constraints are always satisfied exactly and in a fixed amount of time. Where full triangularization is not possible, a block-triangular form can be obtained that still results in more efficient simulations than existing iterative and constraint stabilization techniques.

The proposed approach is applied to the kinematic and dynamic simulation of several mechanical systems, including six-bar mechanisms, parallel robots, and two vehicle suspensions: a five-link and a double-wishbone. The efficient kinematic solution generated for the latter is used in the real-time simulation of a vehicle with double-wishbone suspensions on both axles, which is implemented in a hardware- and operator-in-the-loop driving simulator. The Gröbner basis approach is particularly suitable for situations requiring very efficient simulations of multibody systems whose parameters are constant, such as the plant models in model-predictive control strategies and the vehicle models in driving simulators.

Acknowledgements

Thank you to ...

... my supervisor, John McPhee, for guiding and encouraging another little fish;

... the alumni and current members of the Motion Research Group,
for helping me navigate the occasionally turbulent waters;

... my numerous collaborators from all corners of the world,
for your reflections from diverse schools of thought;

... the University of Waterloo and the Governments of Ontario and Canada,
for the scholarships that have helped me stay afloat;

... and, of course, my parents, brother, and family, for their unwavering support,
and for teaching me the value of swimming against the current.

Dedication

To my parents who, in cultivating the fruits of my labours,
have provided copious amounts of sunlight. And fertilizer.

Foreword

Entombed herein ideas slumber,
Buried under text and number,
Caged by bars of Times New Roman
Marching 'cross these pages woven.

Go henceforth, revive once more
These visions that my mind once bore;
  And if thou art the curious kind,
  Perhaps some interest thou shalt find
  Amid the seeds and pith and rind,
As that's the purpose of this lore.

$\mathcal{T}.\ \mathcal{K}.\ \mathcal{U}.$

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Multibody dynamics is the branch of physics concerned with the motion of interconnected rigid or flexible bodies and the forces that are responsible for this motion. One of the fundamental objectives in this field is the automatic generation of the equations governing the motion of a multibody system, given a description of its components and the interconnections between them, or its topology. The form of the governing dynamic equations can vary greatly, depending on the characteristics of the system under investigation and the procedure used to formulate the equations of motion. Though mathematically equivalent, the various forms of these equations may be suitable for different simulation environments and practical applications. The main objective of this research is the development of a framework for the automatic generation of systems of equations that are suitable for real-time applications.

1.1 Background

Real-time hardware-in-the-loop (HIL) simulation can be used to test electronic and mechanical components of a multibody mechatronic system in isolation, and its use in industry has been firmly established for many years. A real-time model is characterized by its ability to calculate the trajectory of a physical system faster than it would evolve in reality. HIL simulation refers to the replacement of one or more components of a software model with the analogous hardware components, which communicate with the remaining software elements throughout the simulation. Operator-in-the-loop simulation is similar, except that

a human user provides some of the system inputs and observes or otherwise senses some of the system outputs during the simulation. Since the transmission of signals between the software model and the hardware components or human user is not instantaneous, it is essential that the model operates faster than real time.

The governing dynamic equations for the software model in a real-time simulation can be generated numerically or symbolically. In general, numerical formulation techniques produce matrices that are only valid for a given instant of time, so the equations must be reformulated at every time step of a simulation. Although numerical formulations are used by many popular commercial simulation packages, the relatively slow process of repeatedly reformulating the system equations may prohibit their use in real-time applications. Symbolic formulation techniques, on the other hand, produce sets of equations that are eternally valid, so must only be generated once. Prior to simulation, the symbolic equations can also be greatly simplified in numerous ways, which can result in models that simulate many times faster than those modelled using a purely numerical approach [110]. Despite their advantages, symbolic formulation techniques produce sets of equations that can become prohibitively large for complex systems. An efficient symbolic computation package and an intelligent formulation procedure can alleviate this difficulty [36].

1.2 Motivation

Multibody systems of practical interest are generally governed by differential and algebraic equations that would be tedious to derive manually. Automated formulation procedures are employed to ensure these equations are generated both quickly and correctly. Furthermore, by automating this procedure, the analyst need not be concerned with the underlying computational algorithms, and can instead focus on the design of the mechanical system [81]. In this sense, an automated formulation procedure serves the same purpose as a traction control system in a vehicle, which allows the driver to focus on high-level tasks such as navigating and obeying traffic signals rather than low-level tasks such as compensating for a brief reduction in traction. An algorithmic approach to generating the equations of motion also makes the modelling and simulation of multibody systems accessible to a wider audience. An engineer developing a complex control strategy for a dynamic system may

not be familiar with the theory of Gröbner bases, but could benefit from the use of such concepts when generating a model-predictive control strategy or HIL test platform.

There are several compelling reasons for using HIL simulation. In contrast to the use of full physical prototyping, HIL simulation naturally isolates physical subsystems, which can help identify interactions among the various components in a complex mechatronic system. Simulations can also be performed relatively quickly and easily, even before the remaining system components have been manufactured. Furthermore, the need to perform expensive field tests in inconvenient or potentially hazardous environments can often be reduced or eliminated altogether. In some situations, it may be logistically impractical to perform experiments in the environment or under the conditions for which a device was designed, as is often the case in aerospace and automotive safety applications. In addition, since the testing is being performed in a virtual environment, extreme or unusual conditions can be replicated at will, enabling the repeated simulation of cold-start engine tests in the summer, for example [73]. Finally, HIL simulation can greatly reduce the cost of system testing, as failure of the component under scrutiny will not endanger other components in the physical system, and tests can be conducted without human interaction [114]. Thus, HIL simulation isolates hardware components both physically and analytically, and allows the engineer to perform repeatable simulations quickly, easily, safely, and with minimal cost.

1.3 Challenges

The motion of a multibody system can be described using a set of pure ordinary differential equations (ODEs), provided the modelling coordinates are independent. In such situations, the number of ODEs is equal to the number of degrees-of-freedom (DOF) of the system, or the number of independent ways in which the mechanism can move. A closed kinematic chain, or kinematic loop, is a chain of interconnected bodies where the last body is connected back to the first body, as shown in Figure 1.1. Closed-kinematic-chain systems are most readily described in terms of a set of generalized coordinates of cardinality exceeding the DOF of the system. Dependent generalized coordinates add nonlinear algebraic constraint equations to the ODEs of motion, thereby producing a set of

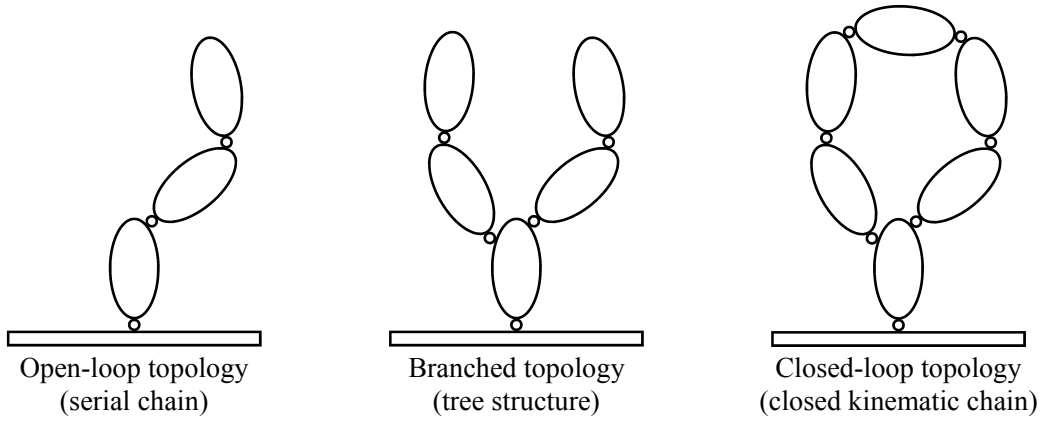


Figure 1.1: Topological nomenclature

differential-algebraic equations (DAEs) that may be difficult to solve in real time. Algebraic constraints also appear in the equations governing the motion of open-loop systems if the generalized coordinates are not independent.

The difference between the modelling of open-loop and constrained mechanisms is illustrated here with a simple example. Consider the unconstrained double-pendulum shown in Figure 1.2. When modelled using joint coordinates $\mathbf{q} = \{\theta_1, \theta_2\}$, the motion of this system is described mathematically by two second-order ODEs:

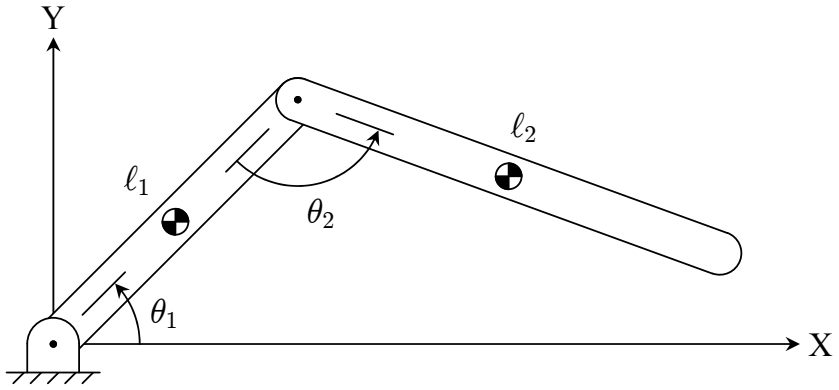


Figure 1.2: Unconstrained double-pendulum

$$\begin{bmatrix} M_{11}(\mathbf{q}) & M_{12}(\mathbf{q}) \\ M_{21}(\mathbf{q}) & M_{22}(\mathbf{q}) \end{bmatrix} \begin{Bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{Bmatrix} = \begin{Bmatrix} F_1(\mathbf{q}, \dot{\mathbf{q}}, t) \\ F_2(\mathbf{q}, \dot{\mathbf{q}}, t) \end{Bmatrix} \tag{1.1}$$

where the entries of the mass matrix (M_{ij}) are functions of the position-level coordinates, and the entries of the force vector (F_i) are functions of the positions and velocities; the

simulation time t may also appear explicitly in the force vector. If the end of the second pendulum is constrained to slide in a horizontal track, as illustrated in Figure 1.3, and the same modelling coordinates are used, we instead obtain a system of two ODEs coupled with one algebraic constraint equation:

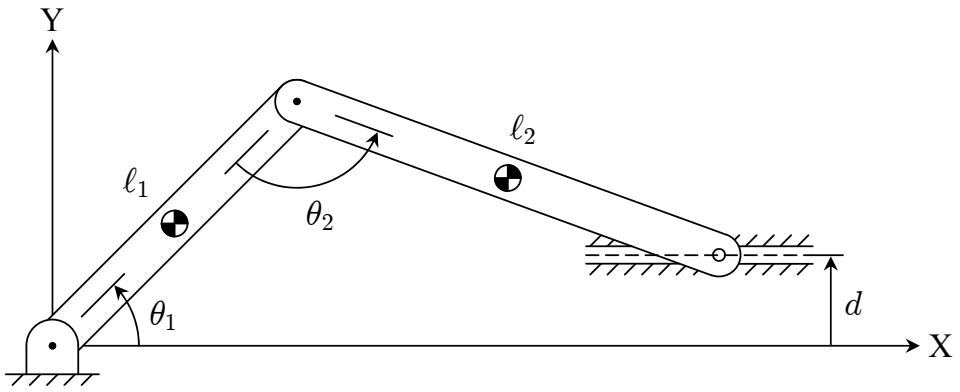


Figure 1.3: Constrained double-pendulum

$$\begin{bmatrix} M_{11}(\mathbf{q}) & M_{12}(\mathbf{q}) \\ M_{21}(\mathbf{q}) & M_{22}(\mathbf{q}) \end{bmatrix} \begin{Bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{Bmatrix} + \begin{Bmatrix} \ell_1 \cos(\theta_1) - \ell_2 \cos(\theta_1 + \theta_2) \\ -\ell_2 \cos(\theta_1 + \theta_2) \end{Bmatrix} \lambda = \begin{Bmatrix} F_1(\mathbf{q}, \dot{\mathbf{q}}, t) \\ F_2(\mathbf{q}, \dot{\mathbf{q}}, t) \end{Bmatrix} \tag{1.2}$$

$$\ell_1 \sin(\theta_1) - \ell_2 \sin(\theta_1 + \theta_2) - d = 0 \tag{1.3}$$

where λ is referred to as a *Lagrange multiplier*, and corresponds to the vertical reaction force at the end of the second pendulum. Although the same modelling coordinates have been used in both systems, the numerical integration of the latter is more challenging than that of the former: whereas the unconstrained double-pendulum has 2 DOF, the constrained double-pendulum has only 1 DOF. Thus, the numerical integration of θ_1 and θ_2 must proceed while ensuring that the constraint equation (1.3) is satisfied.

1.4 Applications

This work is particularly applicable to situations in which one requires very efficient simulations of multibody systems whose parameters are constant. One application is computing the inverse kinematics of parallel robots, such as the Gough–Stewart platform commonly used in flight and driving simulators. Motion controllers for such systems must continually determine, in real time, the leg lengths and joint angles that result in the desired

position and orientation of the platform. A related application is the solution of the constraint equations in the dynamic simulation of closed-kinematic-chain systems. Many systems of practical interest contain closed kinematic chains, including vehicle suspensions and deployable structures; hardware- and operator-in-the-loop simulations of such systems require computationally efficient simulation code to maintain real-time performance. Although branched-topology vehicle models are often used in real-time scenarios, it may be desirable to evaluate the performance of a vehicle and its controllers using a more detailed closed-kinematic-chain model. A third application is the development of plant models for model-predictive controllers, where the demand for very efficient simulation code is substantial—particularly if the control strategy is to be implemented on an embedded processor. Improving the efficiency of a predictive vehicle model, for example, permits the use of less powerful—and, therefore, less expensive—electronic control units. Alternatively, more sophisticated models could be implemented on the existing hardware, potentially resulting in more effective control strategies and safer vehicles. Further potential applications include design optimization, sensitivity analysis, parameter identification, and controller tuning tasks, which can require hundreds or thousands of simulations.

1.5 Thesis organization

A brief description of the contents of this thesis is provided here. The first half of Chapter 2 focuses on the reduction of DAEs to ODEs, which are more amenable to numerical integration; the second half introduces several techniques for solving systems of nonlinear equations, such as the kinematic constraints encountered in the modelling of closed-kinematic-chain systems. In Chapter 3, we propose the use of Gröbner bases for generating efficient systems of kinematic equations, using a simple example to demonstrate each step in the formulation procedure. The three subsequent chapters further explore the Gröbner basis approach through the kinematic and dynamic simulation of a series of mechanisms of increasing complexity. We begin Chapter 4 by studying simple single-loop and cascading-loop mechanisms, then proceed to six-bar mechanisms and parallel robots. Efficient kinematic solutions for the five-link and double-wishbone vehicle suspensions are generated in Chapter 5, the latter of which is used for the real-time simulation of a vehi-

cle with double-wishbone suspensions on both axles. This vehicle is implemented in the hardware- and operator-in-the-loop driving simulator discussed in Chapter 6. The significant contributions of this work are highlighted in Chapter 7, where recommendations for future research are also provided.

Chapter 2

Literature Review

The equations of motion for multibody systems containing closed kinematic chains are most readily derived using a set of redundant generalized coordinates. If an augmented formulation [111] is used, a system of n second-order ODEs (one for each generalized coordinate) and m nonlinear algebraic constraint equations is obtained for a multibody system with $f = n - m$ degrees-of-freedom (DOF):

$$\mathbf{M}\,\ddot{\mathbf{q}} + \mathbf{\Phi}_{\mathbf{q}}^{\mathrm{T}}\,\boldsymbol{\lambda} = \mathbf{F} \tag{2.1}$$

$$\mathbf{\Phi}(\mathbf{q}) = \mathbf{0} \tag{2.2}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{q}$ is the vector of generalized coordinates, $\boldsymbol{\lambda}$ is the vector of Lagrange multipliers, $\mathbf{F}$ is a vector containing the external loads and gyroscopic terms, $\mathbf{\Phi}$ contains the constraint equations, and $\mathbf{\Phi}_{\mathbf{q}} = \partial\mathbf{\Phi}/\partial\mathbf{q}$. This system of $n + m$ DAEs is of index 3 since, in addition to algebraic manipulations, three time differentiations would be required to express it as a system of $(2n+m)$ first-order ODEs. To improve the tractability of numerical integration, it is desirable to reduce the index of DAE systems as much as possible [13]. Note, however, that differentiating (2.2) can lead to the accumulation of significant position-level constraint violations. Several techniques have been developed for reducing the index of DAE systems while maintaining the integrity of the numerical solution. A review of these approaches in the context of real-time simulation is provided in Section 2.1. As described below, an efficient yet accurate simulation strategy can be employed if the position-level constraint equations can be solved symbolically. Techniques for solving systems of nonlinear equations are described in Section 2.2.

2.1 Differential-algebraic equations of motion

The equations of motion are rarely integrated using the index-3 differential-algebraic form shown in (2.1) and (2.2). In this section, some existing solution approaches involving index reduction, constraint stabilization, and system modification are presented. Two existing strategies for solving the constraint equations separately from the dynamic equations are also discussed.

2.1.1 Numerical integration of index-3 DAEs

The motivation for index reduction can be illustrated by considering the numerical solution of an index-3 DAE system using the implicit Euler backward differentiation formula (BDF), which transforms a set of nonlinear differential-algebraic equations into a set of nonlinear algebraic difference equations using the following relationship [13]:

$$\dot{\mathbf{x}}(t) = \frac{\mathbf{x}(t) - \mathbf{x}(t-h)}{h} =: \frac{\Delta \mathbf{x}}{h} \tag{2.3}$$

where $\mathbf{x}(t)$ are the unknown system states at time t, $\mathbf{x}(t-h)$ are the states from the previous time step, and h is the integration step size. The dynamic equations (2.1) are first converted into first-order form by introducing velocity variables $\mathbf{p} = \dot{\mathbf{q}}$, which results in the following system:

$$\mathbf{M}(\mathbf{q})\,\dot{\mathbf{p}} + \boldsymbol{\Phi}(\mathbf{q})_{\mathbf{q}}^{\mathrm{T}}\,\boldsymbol{\lambda} - \mathbf{F}(\mathbf{q},\mathbf{p}) = \mathbf{0} \tag{2.4}$$

$$\mathbf{p} - \dot{\mathbf{q}} = \mathbf{0} \tag{2.5}$$

$$\boldsymbol{\Phi}(\mathbf{q}) = \mathbf{0} \tag{2.6}$$

where the dependencies have been shown explicitly. This system can be solved by applying difference relation (2.3) and using Newton's method to determine the values of $\{\mathbf{q}, \mathbf{p}, \boldsymbol{\lambda}\}$, which requires the following Jacobian matrix [89]:

$$\mathbf{J} = \begin{bmatrix} \frac{1}{h}\,\mathbf{M}_{\mathbf{q}}\,\Delta\mathbf{p} + \boldsymbol{\Phi}_{\mathbf{qq}}^{\mathrm{T}}\,\boldsymbol{\lambda} - \mathbf{F}_{\mathbf{q}} & \frac{1}{h}\,\mathbf{M} - \mathbf{F}_{\mathbf{p}} & \boldsymbol{\Phi}_{\mathbf{q}}^{\mathrm{T}} \\ -\frac{1}{h}\,\mathbf{I} & \mathbf{I} & \mathbf{0} \\ \boldsymbol{\Phi}_{\mathbf{q}} & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{2.7}$$

where $\mathbf{I}$ is an identity matrix of dimension n. Similar Jacobians are obtained for higher-order BDF methods. Two important observations can be made. First, note that Jacobian

$\mathbf{J}$ becomes ill-conditioned as the integration step size h is decreased, which can cause instabilities in adaptive-step-size solvers. Also note that the integration error cannot be monitored for the velocities $\mathbf{p}$ or Lagrange multipliers $\boldsymbol{\lambda}$ [89]. It is for these reasons that index-2 and index-1 formulations are preferred.

2.1.2 Reduction to index-2 or index-1 DAEs

Replacing (2.2) with its time derivative results in an index-2 system of DAEs, which can be integrated using a sufficiently sophisticated DAE solver, but the solution will drift away from the constraint manifold linearly over time [3]. The stabilized index-2 (SI2) formulation first proposed by Gear et al. [41] uses additional Lagrange multipliers $\boldsymbol{\mu}$ in the kinematic differential equations:

$$\mathbf{M}\,\dot{\mathbf{p}} + \boldsymbol{\Phi}_{\mathbf{q}}^{\mathrm{T}}\,\boldsymbol{\lambda} = \mathbf{F} \tag{2.8}$$

$$\mathbf{p} - \dot{\mathbf{q}} + \boldsymbol{\Phi}_{\mathbf{q}}^{\mathrm{T}}\,\boldsymbol{\mu} = \mathbf{0} \tag{2.9}$$

$$\dot{\boldsymbol{\Phi}}(\mathbf{p}, \mathbf{q}) = \mathbf{0} \tag{2.10}$$

$$\boldsymbol{\Phi}(\mathbf{q}) = \mathbf{0} \tag{2.11}$$

The position- and velocity-level constraint violations remain bounded provided Lagrange multipliers $\boldsymbol{\mu}$ remain small. In this case, the Jacobian $\mathbf{J}$ does not become ill-conditioned as the integration step size is decreased, and the integration error can be monitored for both the positions $\mathbf{q}$ and velocities $\mathbf{p}$. Even more tractable for numerical integration, however, are approaches that result in index-1 DAEs. One form of the stabilized index-1 (SI1) formulation replaces (2.2) with its second time derivative, but a less computationally expensive alternative involves introducing new variables $\dot{\boldsymbol{\chi}} = \boldsymbol{\lambda}$ and $\dot{\boldsymbol{\nu}} = \boldsymbol{\mu}$ [89]:

$$\mathbf{M}\,\dot{\mathbf{p}} + \boldsymbol{\Phi}_{\mathbf{q}}^{\mathrm{T}}\,\dot{\boldsymbol{\chi}} = \mathbf{F} \tag{2.12}$$

$$\mathbf{p} - \dot{\mathbf{q}} + \boldsymbol{\Phi}_{\mathbf{q}}^{\mathrm{T}}\,\dot{\boldsymbol{\nu}} = \mathbf{0} \tag{2.13}$$

$$\dot{\boldsymbol{\Phi}}(\mathbf{p}, \mathbf{q}) = \mathbf{0} \tag{2.14}$$

$$\boldsymbol{\Phi}(\mathbf{q}) = \mathbf{0} \tag{2.15}$$

This system is of index 1 since only one differentiation of $\dot{\boldsymbol{\Phi}}$ is required to obtain explicit expressions for $\dot{\boldsymbol{\chi}}$ and $\dot{\boldsymbol{\nu}}$. Since the integration error can be monitored for all states, the SI1 formulation can provide more accurate results than the SI2 formulation.

Although the index-2 and index-1 formulations address the issue of constraint violation, a DAE or stiff ODE solver is required to numerically integrate the resulting equations. In general, DAE solvers assemble implicit systems of nonlinear equations at each time step, and use an iterative technique such as Newton's method to calculate the new states. DASSL is one such solver, and uses a variable-step-size, variable-order form of the BDF methods [13], which have been proven to converge for index-2 problems [41]; RADAU is another, and is based on the implicit Runge–Kutta methods [50]. Such numerical integration schemes pose two fundamental difficulties in the context of real-time simulation. First, real-time applications—particularly those involving hardware- or operator-in-the-loop components—typically require fixed sampling time intervals, which precludes the use of variable-step-size solvers. Furthermore, it cannot be guaranteed that an iterative technique will maintain its real-time performance unless the number of iterations is limited, in which case it cannot be guaranteed that the integration tolerances will be satisfied. The real-time-capable linearly-implicit Euler method, for example, uses a single Newton step for solving a system of implicit equations and projecting the resulting solution onto the constraint manifold at each time step, but the constraint equations are never satisfied exactly [19]. Consequently, the DAEs governing the motion of constrained multibody systems are often replaced with a system of pure ODEs that approximately describes the motion of the original system. A low-order, fixed-step-size, non-stiff ODE solver, such as the simple one-step explicit Euler scheme, is then often employed [3].

2.1.3 Reduction to ODEs using stabilized constraints

The objective of constraint stabilization techniques is to integrate the derivatives of the kinematic constraints along with the ODEs of motion in a way that avoids the accumulation of position-level constraint violations. Replacing (2.2) with its second time derivative lowers the DAE index to 1, but the position-level constraint violations will grow quadratically over time [3]. Baumgarte stabilization [7] prevents the accumulation of significant constraint violations by replacing (2.2) with a linear combination of the position-, velocity-, and acceleration-level constraints:

$$\ddot{\mathbf{\Phi}} + 2\alpha_B \dot{\mathbf{\Phi}} + \beta_B^2 \mathbf{\Phi} = \mathbf{0} \tag{2.16}$$

Although it is relatively straightforward to implement, the effectiveness of Baumgarte's method relies heavily on the judicious selection of stabilization parameters α_B and β_B. Parameters $\alpha_B = 1/h$ and $\beta_B = 1/h^2$, where h is the integration step size, are theoretically optimal but impractically large, as they introduce artificial stiffness into the system [31]. In practice, α_B and β_B are usually chosen to be equal, thereby achieving critical damping and providing the fastest error reduction [6], with values typically between 1 and 20 [40]. On its own, twice differentiating (2.2) and applying Baumgarte stabilization only reduces the index of the system equations by two, and index-1 DAEs are not suitable for the non-stiff ODE solvers that are typically used in real-time applications. In order to obtain a system of ODEs, Baumgarte's method can be applied once the Lagrange multipliers have been eliminated using, for example, the null space formulation [74]:

$$\begin{bmatrix} \mathbf{\Gamma}^{\mathrm{T}}\,\mathbf{M} \\ \mathbf{\Phi_q} \end{bmatrix} \ddot{\mathbf{q}} = \begin{bmatrix} \mathbf{\Gamma}^{\mathrm{T}}\,\mathbf{F} \\ \hat{\boldsymbol{\gamma}} \end{bmatrix} \tag{2.17}$$

where $\mathbf{\Gamma}$ is an orthogonal complement of the Jacobian $\mathbf{\Phi_q}$, and $\hat{\boldsymbol{\gamma}}$ is the right-hand side of the Baumgarte-stabilized constraints:

$$\mathbf{\Phi_q}\,\ddot{\mathbf{q}} = \boldsymbol{\gamma} - 2\alpha_B\dot{\mathbf{\Phi}} - \beta_B^2\mathbf{\Phi} = \hat{\boldsymbol{\gamma}} \tag{2.18}$$

Alternatively, the Lagrange multipliers can be solved explicitly [141]:

$$\boldsymbol{\lambda} = \left(\mathbf{\Phi_q}\,\mathbf{M}^{-1}\,\mathbf{\Phi}_{\mathbf{q}}^{\mathrm{T}}\right)^{-1}\left(\mathbf{\Phi_q}\,\mathbf{M}^{-1}\,\mathbf{F} - \hat{\boldsymbol{\gamma}}\right) \tag{2.19}$$

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}\left(\mathbf{F} - \mathbf{\Phi}_{\mathbf{q}}^{\mathrm{T}}\,\boldsymbol{\lambda}\right) \tag{2.20}$$

Another constraint stabilization technique is the penalty formulation, which treats the constraint equations as ideal mass-spring-damper systems that are penalized by large values, and are used in place of the Lagrange multipliers [9]:

$$\mathbf{M}\,\ddot{\mathbf{q}} + \mathbf{\Phi}_{\mathbf{q}}^{\mathrm{T}}\,\boldsymbol{\rho}\left(\ddot{\mathbf{\Phi}} + 2\boldsymbol{\omega}\boldsymbol{\zeta}\dot{\mathbf{\Phi}} + \boldsymbol{\omega}^2\mathbf{\Phi}\right) = \mathbf{F} \tag{2.21}$$

where $\boldsymbol{\rho}$, $\boldsymbol{\omega}$, and $\boldsymbol{\zeta}$ are diagonal matrices containing, respectively, the penalty numbers, natural frequencies, and damping ratios associated with each penalty system. Larger penalty numbers $\boldsymbol{\rho}$ enforce the constraints more strictly; García de Jalón and Bayo suggest using values 10^7 times greater than the largest element of the mass matrix, which was found to

produce acceptable results in real-time simulations [8]. This approach is conceptually similar to force coupling, and suffers from the same numerical ill-conditioning as the penalty values are increased. To overcome this issue, García de Jalón and Bayo use the following iterative process at each time step, which ensures that the constraints are satisfied to a desired tolerance regardless of the size of the penalty values [40]:

$$\ddot{\mathbf{q}}^{(0)} = \mathbf{M}^{-1}\,\mathbf{F} \tag{2.22}$$

$$\ddot{\mathbf{q}}^{(k)} = \left(\mathbf{M} + \boldsymbol{\Phi}_{\mathbf{q}}^{\mathrm{T}}\,\boldsymbol{\rho}\,\boldsymbol{\Phi}_{\mathbf{q}}\right)^{-1}\left\{\mathbf{M}\,\ddot{\mathbf{q}}^{(k-1)} - \boldsymbol{\Phi}_{\mathbf{q}}^{\mathrm{T}}\,\boldsymbol{\rho}\left(\dot{\boldsymbol{\Phi}}_{\mathbf{q}}\,\dot{\mathbf{q}} + \dot{\boldsymbol{\Phi}}_{t} + 2\boldsymbol{\omega}\boldsymbol{\zeta}\dot{\boldsymbol{\Phi}} + \boldsymbol{\omega}^{2}\boldsymbol{\Phi}\right)\right\}, \quad k \geq 1 \tag{2.23}$$

where the parenthetic superscript indicates the iteration number. Although this strategy iterates at each time step, using only a single iteration can considerably widen the range of acceptable penalty values [40].

2.1.4 Conversion to ODEs by modifying the system model

A common approach used for obtaining a real-time-capable system of equations that approximately describes the motion of a constrained system is to adjust the system model. Modifying the equations of motion directly (using a constant [114], linearized [19], or only periodically updated [5] Jacobian, for example) is a technique that can reduce the computational cost considerably. Perhaps the most common modification is replacing a closed-loop multibody system with a similar open-loop system and superimposing the effects of the neglected components using tabulated data. For example, Sayers [106] predicts the handling characteristics of passenger vehicles using a simplified branched-topology vehicle model consisting of three main components: a sprung mass, representing the vehicle body, drivetrain, and parts of the suspensions and front wheels; an unsprung mass, representing the rear wheels and the remaining parts of the suspensions and front wheels; and four tires, using a simple linear tire model that is valid only for the small deflections typical of highway driving. This model was extended to include the effects of the most significant components influencing the handling, braking, and acceleration performance of a vehicle—such as the relationship between vertical displacement and tire camber caused by the closed-loop suspension kinematics—and has evolved into the commercially-available CarSim software package [107]. When exported into the MATLAB/Simulink environment,

the simulation code generated by CarSim can be executed faster than real time on a 233-MHz Intel Pentium processor, and has been used in hardware-in-the-loop simulations [107]. ADAMS/Car RealTime uses a similar vehicle model. Although these software packages are capable of generating real-time simulation code with minimal user input, they are limited in three fundamental ways:

1. Since the dynamic equations have been hand-derived, only specific vehicle topologies and configurations can be simulated.

2. The amount of computation required during the simulation has been reduced using modelling approximations, which may not be suitable for detailed dynamic analyses.

3. The tabulated data used to describe the nonlinear suspension kinematics must be obtained either experimentally or through the simulation of a more detailed closed-kinematic-chain model.

To avoid the use of tabulated data, Eichberger and Rulka [27] developed the macro joint approach, which systematically reduces a detailed vehicle model containing flexible bodies and compliant joints to one suitable for real-time applications by neglecting the masses of suspension components. The macro joint approach retains the parameterization of the original suspension model, adding kinematic equations that describe the jounce, rebound, and steering behaviour to those that describe the quasi-static deformation behaviour of the eliminated components.

Force coupling is another modification technique used for converting DAEs into ODEs. In this case, one joint in each kinematic loop is replaced with a virtual spring, which acts on the detached bodies to approximately satisfy the constraints that were enforced by the removed joint. A damper can also be added to mitigate the oscillations induced by the spring. In addition to obtaining a system of ODEs, the use of force coupling results in a block-diagonal mass matrix, which facilitates the distribution of the simulation code over multiple processors [136]. The stiffness of the virtual spring must be high enough to maintain an acceptably small distance between the detached bodies. As the stiffness increases, however, high-frequency oscillations are introduced in the system, which necessitates the use of a small integration step size to produce an accurate simulation [109]. Thus, as is

the case with Baumgarte's method, the effectiveness of this technique relies on the judicious selection of system-dependent parameters. The use of a compliant joint model [112] accomplishes the same goal, and is hindered by the same practical difficulties.

2.1.5 Reduction to ODEs by solving the constraints separately

Rather than using a simplified model or attempting to solve a system of DAEs, the constraint equations can be solved separately from the dynamic equations; since only the latter must be integrated, a standard ODE solver can be used. To this end, Kecskeméthy et al. [68] proposed the characteristic pair of joints approach, a three-step strategy in which systems of algebraic equations are solved symbolically for a set of independent loops. The first step of this approach involves identifying in the topological system graph the cycle basis with the minimum total weight, where the edge weights are based on the number of coordinates associated with the corresponding joint. The minimal cycle basis can be found using standard graph-theoretic algorithms [58], and is similar to the notion of fundamental circuits. The objective of the second step is to generate a triangular, or recursively solvable, system—that is, a sequence of equations in which each successive equation contains exactly one more unknown than its predecessor—for each independent kinematic loop. These equations are obtained by projecting loop closure equations onto geometrically invariant elements, such as the squared distance between two points [66]. By varying the form of the loop closure equations and projecting onto the appropriate geometric elements, recursively solvable systems can often be found. Expressions for velocities and accelerations are obtained using loop closure equations at the velocity and acceleration levels. In the final step of this approach, the local solutions are assembled into a global solution and the optimal solution flow, or order in which the local solutions must be solved, is determined. MOBILE is an object-oriented C++ implementation of this technique that uses symbolic solutions generated in Mathematica [67].

Although the characteristic pair of joints technique has been used to generate efficient simulation code for many mechanisms, the approach has several limitations. First, minimizing the number of generalized coordinates does not always result in the fastest simulations [75]. Furthermore, a purely joint coordinate formulation, as required by the characteristic pair of joints approach, is often less efficient than a formulation using a

mixed set of joint and absolute coordinates. Finally, there may not be a sufficient supply of suitable geometrically invariant elements in order to form recursively solvable systems for all independent loops, in which case the algorithm resorts to an iterative procedure. The characteristic pair of joints approach is discussed further in Section 4.1, where it is used to generate recursively solvable kinematic solutions for two single-loop mechanisms.

Symbolic computation is also exploited in the embedding technique [111], which is a form of projection and can be described as a symbolic analogue of the null space formulation (2.17). First, virtual displacements $\delta\mathbf{q}$ are partitioned into independent and dependent sets ($\delta\mathbf{q}_i$ and $\delta\mathbf{q}_d$, respectively), and are transformed into the former as follows [83]:

$$\delta\mathbf{q} = \begin{bmatrix} -\mathbf{\Phi}_d^{-1}\,\mathbf{\Phi}_i \\ \mathbf{I} \end{bmatrix} \delta\mathbf{q}_i = \mathbf{B}\,\delta\mathbf{q}_i \tag{2.24}$$

where $\mathbf{\Phi}_d = \partial\mathbf{\Phi}/\partial\mathbf{q}_d$, $\mathbf{\Phi}_i = \partial\mathbf{\Phi}/\partial\mathbf{q}_i$, and $\mathbf{I}$ is an identity matrix of dimension f. Since transformation matrix $\mathbf{B}$ is an orthogonal complement of the Jacobian $\mathbf{\Phi}_\mathbf{q}$, pre-multiplying (2.1) by $\mathbf{B}^\mathrm{T}$ eliminates the Lagrange multipliers from the dynamic equations:

$$\mathbf{B}^\mathrm{T}\,\mathbf{M}\,\ddot{\mathbf{q}} = \mathbf{B}^\mathrm{T}\,\mathbf{F} \tag{2.25}$$

The transformation equation (2.24) can be written at the velocity level:

$$\dot{\mathbf{q}} = \begin{bmatrix} -\mathbf{\Phi}_d^{-1}\,\mathbf{\Phi}_i \\ \mathbf{I} \end{bmatrix} \dot{\mathbf{q}}_i + \begin{bmatrix} -\mathbf{\Phi}_d^{-1}\,\mathbf{\Phi}_t \\ \mathbf{0} \end{bmatrix} = \mathbf{B}\,\dot{\mathbf{q}}_i + \mathbf{C} \tag{2.26}$$

where $\mathbf{\Phi}_t = \partial\mathbf{\Phi}/\partial t$, and differentiated with respect to time:

$$\ddot{\mathbf{q}} = \mathbf{B}\,\ddot{\mathbf{q}}_i + \dot{\mathbf{B}}\,\dot{\mathbf{q}}_i + \dot{\mathbf{C}} \tag{2.27}$$

The dependent accelerations can then be eliminated from (2.25):

$$\mathbf{B}^\mathrm{T}\,\mathbf{M}\,\mathbf{B}\,\ddot{\mathbf{q}}_i = \mathbf{B}^\mathrm{T}\,\mathbf{F} - \mathbf{B}^\mathrm{T}\,\mathbf{M}\left(\dot{\mathbf{B}}\,\dot{\mathbf{q}}_i + \dot{\mathbf{C}}\right) \tag{2.28}$$

Thus, the embedding technique produces a set of f second-order ODEs, one for each independent coordinate. The generalized coordinate partitioning technique [103] is similar, in that one ODE is obtained for each DOF, but the reduction is performed numerically rather than symbolically. In either case, note that the resulting ODEs of motion (2.28) are functions of $\mathbf{q}$ and $\dot{\mathbf{q}}$. The dependent velocities $\dot{\mathbf{q}}_d$ can be calculated from $\dot{\mathbf{q}}_i$ at each time

step using the velocity transformation (2.26); the dependent positions $\mathbf{q}_d$ can be determined by integrating $\dot{\mathbf{q}}_d$ and using constraint stabilization [101], or by iterating over the position constraints using Newton's method [37]. Depending on the structure of the Jacobian $\mathbf{\Phi_q}$, it may be possible to iterate over subsets of $\mathbf{q}_d$; as will be demonstrated in Chapter 4, this approach is generally faster than solving for all dependent coordinates simultaneously. Simulation code that is even more efficient can be obtained if the position constraints can be converted into a triangular form, which is the motivation of the next section.

2.2 Solving systems of nonlinear equations

Closed kinematic chains generally add nonlinear algebraic constraint equations to the ODEs of motion, thereby producing a set of index-3 DAEs that may be difficult to solve in real time. Even if the embedding technique is used to reduce the system to a minimal set of ODEs, nonlinear equations relating the independent and dependent coordinates must be solved. If Newton's method is used and permitted to iterate until convergence is obtained, real-time performance of the simulation cannot be guaranteed. On the other hand, if only a fixed number of iterations is permitted, the precision of the simulation results cannot be guaranteed. This section describes some of the existing techniques for obtaining recursively solvable systems that can be used in place of the coupled nonlinear algebraic constraint equations. Recursively solvable systems provide exact solutions in a fixed amount of time, which is ideal for the real-time simulation of constrained multibody systems.

2.2.1 Goniometric equations

In general, the systems of equations obtained in the kinematic analysis of multibody systems are goniometric—that is, they involve trigonometric functions of the generalized coordinates. Since goniometric equations are more difficult to solve than polynomial equations, one of three methods is typically used to convert the former into the latter before any other simplification techniques are applied. The first, referred to as the tangent-half-angle substitution, uses the following familiar trigonometric identities:

$$\sin(\vartheta) = \frac{2\tan(\vartheta/2)}{1+\tan^2(\vartheta/2)} \qquad \cos(\vartheta) = \frac{1-\tan^2(\vartheta/2)}{1+\tan^2(\vartheta/2)} \tag{2.29}$$

Polynomial equations can then be obtained in terms of a new variable $x = \tan(\vartheta/2)$. Once the denominators have been cleared, however, extraneous roots $x = \pm\sqrt{-1}$ are generated by the $(1 + x^2)$ terms. The second technique, called Euler substitution, applies the following identities:

$$\sin(\vartheta) = \frac{1 - y^2}{2y}\, j \qquad \cos(\vartheta) = \frac{1 + y^2}{2y} \tag{2.30}$$

where $y = e^{j\vartheta}$ and $j = \sqrt{-1}$. Although extraneous factors are introduced in this case as well, they are of a simpler form since they originate from the linear factor y rather than the quadratic term $(1 + x^2)$ [72]. The third substitution method uses the following simple transformations:

$$\sin(\vartheta) = x \qquad \cos(\vartheta) = y \tag{2.31}$$

which requires the introduction of auxiliary equations of the form $x^2 + y^2 - 1 = 0$ [100]. The application of these substitution techniques to a practical problem will be explored in Section 3.2.1. For the remainder of this section, we shall assume that one of these three methods has been employed to obtain a system of polynomial equations.

2.2.2 Polynomial continuation

Polynomial continuation is a numerical procedure that can be used to find all the solutions to a system of polynomial equations. This technique is motivated by the fact that, in general, small changes made to the coefficients of a system of equations will only change the numerical values of the solutions by a small amount. Thus, if the roots of a suitable start system are known, the unknown roots of a target system can be determined by tracking the solution paths as the start system is gradually morphed into the target system [96]. Newton–Raphson iteration is typically used to determine the roots of each intermediate system, using those found for the previous system as initial guesses. Provided the start system and the transformation procedure (or homotopy) are chosen appropriately, polynomial continuation is guaranteed to find all the solutions of the target system, and has been applied to several problems in kinematics [86, 121]. In addition, since each solution path can be tracked independently, this approach can be readily parallelized [120]. The two shortcomings of polynomial continuation in the context of real-time simulation are its use of iteration at each step in the morphing process, and the fact that it finds numerical

solutions for particular instants of time, not symbolic solutions that remain valid for an entire simulation.

2.2.3 Dialytic elimination

The elimination methods were first proposed by Cayley in 1848 [20]. Dialytic elimination, the most popular of these methods, yields all the solutions to a system of polynomial equations, and consists of six steps [99]:

1. Rewrite the equations with one variable suppressed, or assumed known.
2. Define the remaining monomials, including "1", as independent unknowns.
3. Generate new linearly independent homogeneous equations until the same number of equations as unknowns is obtained.
4. Set the determinant of the coefficient matrix to zero to obtain a polynomial in the suppressed variable.
5. Determine the roots of this polynomial to solve for all possible values of the suppressed variable.
6. Substitute for the suppressed variable and solve for the remaining unknowns.

To illustrate this technique, we shall compute all real solutions of the following system of polynomial equations:

$$f_1 = x^2 + y^2 + z^2 - 8x + 2y - 8 = 0 \tag{2.32}$$

$$f_2 = 3x^2 - 2xy - y^2 + 2yz + 2z - 4 = 0 \tag{2.33}$$

$$f_3 = 2x - 10y - 3z = 0 \tag{2.34}$$

which corresponds to finding the points of intersection of an ellipsoid, a hyperboloid, and a plane in $\mathbb{R}^3$. Solving first for z, the system is rewritten as follows:

$$(1)\,x^2 + (1)\,y^2 + (-8)\,x + (2)\,y + \left(z^2 - 8\right)1 = 0 \tag{2.35}$$

$$(3)\,x^2 + (-2)\,xy + (-1)\,y^2 + (2z)\,y + (2z - 4)\,1 = 0 \tag{2.36}$$

$$(2)\,x + (-10)\,y + (-3z)\,1 = 0 \tag{2.37}$$

In this case, the independent unknowns are $\{x^2, xy, y^2, x, y, 1\}$, which are sometimes replaced with new variables to emphasize their independence. Note that there are six unknowns but only three equations. Two more linearly independent equations can be obtained by multiplying (2.34) by x and y:

$$f_3 \times x \quad \rightarrow \quad (2)\, x^2 + (-10)\, xy + (-3z)\, x = 0 \tag{2.38}$$

$$f_3 \times y \quad \rightarrow \quad (2)\, xy + (-10)\, y^2 + (-3z)\, y = 0 \tag{2.39}$$

A sixth linearly independent equation cannot be generated in this fashion without introducing new independent unknowns. Multiplying (2.32) by x, for example, introduces x^3 and xy^2:

$$f_1 \times x \quad \rightarrow \quad (1)\, x^3 + (1)\, xy^2 + (-8)\, x^2 + (2)\, xy + \left(z^2 - 8\right) x = 0 \tag{2.40}$$

which necessitates the generation of two additional linearly independent equations. In this example, we ultimately obtain a system of 10 equations in 10 unknowns, consisting of (2.35) to (2.40) as well as the following:

$$f_1 \times y \quad \rightarrow \quad (1)\, x^2y + (1)\, y^3 + (-8)\, xy + (2)\, y^2 + \left(z^2 - 8\right) y = 0 \tag{2.41}$$

$$f_2 \times x \quad \rightarrow \quad (3)\, x^3 + (-2)\, x^2y + (-1)\, xy^2 + (2z)\, xy + (2z - 4)\, x = 0 \tag{2.42}$$

$$f_2 \times y \quad \rightarrow \quad (3)\, x^2y + (-2)\, xy^2 + (-1)\, y^3 + (2z)\, y^2 + (2z - 4)\, y = 0 \tag{2.43}$$

$$f_3 \times x^2 \quad \rightarrow \quad (2)\, x^3 + (-10)\, x^2y + (-3z)\, x^2 = 0 \tag{2.44}$$

Recall that a homogeneous system of linear equations has nontrivial solutions if, and only if, the determinant of its coefficient matrix is zero [2]. In this example, the determinant of

the coefficient matrix can be computed as follows:

$$
\begin{vmatrix}
0 & 0 & 0 & 0 & 1 & 0 & 1 & -8 & 2 & z^2-8 \\
0 & 0 & 0 & 0 & 3 & -2 & -1 & 0 & 2z & 2z-4 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & -10 & -3z \\
0 & 0 & 0 & 0 & 2 & -10 & 0 & -3z & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 2 & -10 & 0 & -3z & 0 \\
1 & 0 & 1 & 0 & -8 & 2 & 0 & z^2-8 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & -8 & 2 & 0 & z^2-8 & 0 \\
3 & -2 & -1 & 0 & 0 & 2z & 0 & 2z-4 & 0 & 0 \\
0 & 3 & -2 & -1 & 0 & 0 & 2z & 0 & 2z-4 & 0 \\
2 & -10 & 0 & 0 & -3z & 0 & 0 & 0 & 0 & 0
\end{vmatrix}
$$

$$
\begin{aligned}
&= -279624z^4 + 231680z^3 + 4067584z^2 - 3706880z + 6502400 \\
&= -8\,(z-4)\,(z+4)\left(34953z^2 - 28960z + 50800\right) \qquad (2.45)
\end{aligned}
$$

Upon equating (2.45) to zero, two nontrivial solutions are obtained for $z \in \mathbb{R}$: $z_1 = 4$ and $z_2 = -4$; we shall ignore the two complex solutions. Note that the trivial solution of all unknowns equal to zero is inadmissible, since "1" is among the unknowns. To determine the values of x and y corresponding to the two solutions for z, we first substitute z_1 into f_1, f_2, and f_3:

$$x^2 + y^2 - 8x + 2y + 8 = 0 \qquad (2.46)$$

$$3x^2 - 2xy - y^2 + 8y + 4 = 0 \qquad (2.47)$$

$$2x - 10y - 12 = 0 \qquad (2.48)$$

Suppressing y and equating the determinant of the resulting coefficient matrix to zero, we obtain two solutions for y; substituting back into the original system to solve for x, we find one solution in $\mathbb{R}^3$: $(x, y, z) = (1, -1, 4)$. Repeating the procedure for z_2, we find $(x, y, z) = (4, 2, -4)$. As suggested by this example, problems can become impractically large if new unknowns are introduced while generating additional linearly independent equations. Dialytic elimination has been used for the analysis of several specific mechanisms [134], but such procedures cannot be easily automated.

2.2.4 Resultant methods

Several methods of solving systems of polynomial equations are based on calculating the resultant of two polynomials [64], and have been applied to a variety of problems in kinematics [139]. The motivation for this approach is to reduce a system of nonlinear equations into an equivalent system for which the solutions can be easily obtained, and is roughly analogous to Gaussian elimination [43]. Consider two polynomials $A(x)$, $B(x) \in \mathbb{R}[x]$:

$$A(x) = \sum_{k=0}^{d_a} a_k x^k \qquad B(x) = \sum_{k=0}^{d_b} b_k x^k \tag{2.49}$$

The Sylvester matrix of polynomials $A(x)$ and $B(x)$ is a square matrix of size $d_a + d_b$, defined as follows (shown for $d_a = 4$, $d_b = 3$):

$$S = \begin{bmatrix} a_4 & a_3 & a_2 & a_1 & a_0 & 0 & 0 \\ 0 & a_4 & a_3 & a_2 & a_1 & a_0 & 0 \\ 0 & 0 & a_4 & a_3 & a_2 & a_1 & a_0 \\ b_3 & b_2 & b_1 & b_0 & 0 & 0 & 0 \\ 0 & b_3 & b_2 & b_1 & b_0 & 0 & 0 \\ 0 & 0 & b_3 & b_2 & b_1 & b_0 & 0 \\ 0 & 0 & 0 & b_3 & b_2 & b_1 & b_0 \end{bmatrix} \tag{2.50}$$

where the first d_b rows contain the coefficients of $A(x)$ and the last d_a rows contain those of $B(x)$. The resultant is simply the determinant of the Sylvester matrix. For example, the resultant, with respect to x, of (2.32) and (2.33) can be calculated as follows:

$$\begin{aligned} \mathrm{res}_x(f_1, f_2) &= \begin{vmatrix} 1 & -8 & y^2+z^2+2y-8 & 0 \\ 0 & 1 & -8 & y^2+z^2+2y-8 \\ 3 & -2y & -y^2+2yz+2z-4 & 0 \\ 0 & 3 & -2y & -y^2+2yz+2z-4 \end{vmatrix} \\ &= 20y^4 - 16y^3z + 24y^3 + 32y^2z^2 - 72y^2z - 444y^2 - 12yz^3 \\ &\quad - 4yz^2 + 408yz + 208y + 9z^4 - 12z^3 - 116z^2 + 464z - 368 \end{aligned} \tag{2.51}$$

Note that (2.51) is a polynomial in y and z—that is, x has been eliminated. A system of polynomial equations can be solved by eliminating variables in this manner, creating a new

set of polynomials following the elimination of each variable until a single univariate polynomial remains. Using the example from the previous section, we define $F_1 = \{f_1, f_2, f_3\}$ as the first set of polynomials, which is in terms of x, y, and z. Resultants can then be calculated pairwise to eliminate x:

$$\begin{aligned} f_{12} = \text{res}_x(f_1, f_2) = {} & 20y^4 - 16y^3z + 24y^3 + 32y^2z^2 - 72y^2z - 444y^2 - 12yz^3 \\ & - 4yz^2 + 408yz + 208y + 9z^4 - 12z^3 - 116z^2 + 464z - 368 \end{aligned} \tag{2.52}$$

$$f_{13} = \text{res}_x(f_1, f_3) = 104y^2 + 60yz - 152y + 13z^2 - 48z - 32 \tag{2.53}$$

The second set of polynomials, $F_2 = \{f_{12}, f_{13}\}$, is in terms of only y and z. Finally, F_3 consists of a single univariate polynomial in z:

$$\begin{aligned} \text{res}_y(f_{12}, f_{13}) = {} & 26746035600z^8 - 153807171200z^7 - 509155874048z^6 \\ & + 4842191929344z^5 - 3082535620608z^4 - 32918369501184z^3 \\ & + 67097262882816z^2 - 82913048657920z + 48247183769600 \end{aligned} \tag{2.54}$$

The system can now be solved by back-substituting either symbolic or numeric solutions from F_3 into F_2, and those from F_2 into F_1. The list of sets of polynomials $[F_1, F_2, F_3]$ is called a *reduced system*.

There are two practical limitations associated with this method. First, a reduced system can only be obtained if there exists a sufficient supply of nonzero resultants, but this conclusion cannot be established until all possible resultant pairs have been exhausted. The procedure may also fail for particular choices of resultants, requiring the selection of different pairings at a previous step. The second practical issue is the introduction of extraneous roots. Resultants identify the roots that are common to a pair of polynomials; however, the roots common to a pair are not necessarily common to a triple. In the above example, an eighth-degree univariate polynomial is obtained in (2.54), but by Bézout's Theorem, F_1 can have at most four solutions [23]. In addition to polluting the problem with inadmissible solutions, extraneous roots also increase both the number of terms and the size of the coefficients in the intermediate expressions.

2.2.5 Exact Gröbner bases

Let $\mathcal{F}$ be a field (such as the rational numbers $\mathbb{Q}$, the real numbers $\mathbb{R}$, or the finite field $\mathbb{Z}_p$, where p is prime) and $\mathcal{R} = \mathcal{F}[x_1, \ldots, x_v]$ be a polynomial ring in v variables over $\mathcal{F}$. Every finite set of polynomials $\{b_1, \ldots, b_k\} \subset \mathcal{R}$ forms a basis of an ideal $\mathcal{I}$:

$$\mathcal{I} = \langle b_1, \ldots, b_k \rangle = \left\{ \sum_{\sigma=1}^{k} r_\sigma b_\sigma : r_\sigma \in \mathcal{R} \right\} \tag{2.55}$$

The *variety* of $\mathcal{I}$ is the set of all solutions to the polynomial equations $b_k = 0$, which are also solutions to $b = 0$ for all polynomials $b \in \mathcal{I}$:

$$V(\mathcal{I}) = \{\mathbf{u} \in \mathcal{F}^v : b_1(\mathbf{u}) = \ldots = b_k(\mathbf{u}) = 0\} = \{\mathbf{u} \in \mathcal{F}^v : b(\mathbf{u}) = 0 \text{ for all } b \in \mathcal{I}\} \tag{2.56}$$

It is often of interest to determine the size of $V(\mathcal{I})$, whether $\mathcal{I} = \mathcal{R}$, or whether a particular $r \in \mathcal{R}$ belongs to $\mathcal{I}$ [133]. A Gröbner basis is a unique, canonical representation of $\mathcal{I}$ that can be useful in such pursuits.

In contrast to polynomial continuation, dialytic elimination, and the resultant methods, a Gröbner basis can be obtained algorithmically, and generates a system of equations with the same solutions as the original system. When generated in a particular way, a Gröbner basis introduces indeterminates one after the other—that is, it triangularizes the system in a way that is analogous to Gaussian elimination, but for nonlinear systems. The notion of a Gröbner basis was first proposed in 1964, under the appellation *standard basis*, as a canonical form for representing bases of ideals; however, it was the development of an algorithm by Buchberger [17] for transforming an arbitrary set of polynomials into such a form that made the concept practical. The utility of Gröbner bases was not immediately recognized, however, in part due to the disparity between the computational effort required by Buchberger's algorithm and the computational resources available at the time. Although established in the 1980s as a potentially useful tool for solving systems of algebraic equations [11], Gröbner bases were considered to be too computationally expensive to be of use in most practical problems, and retained this reputation well into the 1990s [96]. Algorithms based on Buchberger's work have now been incorporated into almost every modern computer algebra software package, including Maple and Mathematica, as well as several non-commercial packages designed specifically for the efficient generation and manipulation of Gröbner bases [54]. The theory of Gröbner bases will be described here

with the aid of a simple example; detailed mathematical introductions [23, 39] as well as more application-oriented presentations [16, 54] are widely available.

Consider the set $H = \{h_1, h_2\}$ comprised of the following bivariate polynomials:

$$h_1 = 4x^2y + 9xy + 4x - 7y - 4 \tag{2.57}$$

$$h_2 = xy - y + 3 \tag{2.58}$$

where we wish to solve $h_1 = 0$ and $h_2 = 0$ for x and y. First, a *term ordering* must be selected to determine the order in which power products are to be eliminated. A pure lexicographic ordering with $x \succ y$ orders power products involving indeterminates x and y as follows:

$$1 \prec y \prec y^2 \prec \ldots \prec x \prec xy \prec xy^2 \prec \ldots \prec x^2 \prec x^2y \prec x^2y^2 \prec \ldots \tag{2.59}$$

where order relations $\succ$ and $\prec$ are used to indicate the relative weight of each indeterminate or power product. The highest-weighted power product appearing in a polynomial is referred to as the *leading power product*, and is eliminated first. The reduction procedure begins by subtracting the smallest multiples of h_1 and h_2 such that their leading power products cancel:

$$\begin{aligned} s_{1,2} &= h_1 - 4xh_2 \\ &= 13xy - 8x - 7y - 4 \end{aligned} \tag{2.60}$$

where $s_{1,2}$ is referred to as the S-polynomial (subtraction polynomial) of h_1 and h_2. Multiples of h_1 and h_2 are now subtracted from $s_{1,2}$ so as to reduce the weight of its leading power product as much as possible:

$$\begin{aligned} s'_{1,2} &= s_{1,2} - 13h_2 \\ &= -8x + 6y - 43 \end{aligned} \tag{2.61}$$

Since the leading power product of $s'_{1,2}$ cannot be further reduced by subtracting a multiple of h_1 or h_2, $s'_{1,2}$ is said to be *reduced modulo* H. Set H is successively augmented with reduced, nonzero S-polynomials until all reduced S-polynomials are congruent to zero

modulo H:

$$h_3 = s'_{1,2} \tag{2.62}$$

$$H' = H \cup \{h_3\} = \{h_1, h_2, h_3\} \tag{2.63}$$

$$s_{1,3} = 2h_1 + xyh_3 = 6xy^2 - 25xy + 8x - 14y - 8 \tag{2.64}$$

$$s'_{1,3} = s_{1,3} - 6yh_2 = -25xy + 8x + 6y^2 - 32y - 8 \tag{2.65}$$

$$s''_{1,3} = s'_{1,3} + 25h_2 = 8x + 6y^2 - 57y + 67 \tag{2.66}$$

$$s'''_{1,3} = s''_{1,3} + h_3 = 6y^2 - 51y + 24 \tag{2.67}$$

$$h_4 = \tfrac{1}{3}s'''_{1,3} = 2y^2 - 17y + 8 \tag{2.68}$$

$$H'' = H' \cup \{h_4\} = \{h_1, h_2, h_3, h_4\} \tag{2.69}$$

Note that the greatest common divisor of the coefficients in $s'''_{1,3}$ is removed from h_4 in (2.68) to mitigate the growth of coefficients. All remaining S-polynomials are congruent to zero modulo H'':

$$s_{1,4} = yh_1 - 2x^2h_4 \equiv 0 \qquad (\text{mod } H'') \tag{2.70}$$

$$s_{2,3} = 8h_2 + yh_3 \equiv 0 \qquad (\text{mod } H'') \tag{2.71}$$

$$s_{2,4} = 2yh_2 - xh_4 \equiv 0 \qquad (\text{mod } H'') \tag{2.72}$$

$$s_{3,4} = y^2h_3 + 4xh_4 \equiv 0 \qquad (\text{mod } H'') \tag{2.73}$$

H'' is a Gröbner basis of H with respect to the selected term ordering (2.59), and can be simplified further by reducing all $h_k \in H''$ modulo $H'' - \{h_k\}$:

$$g_1 = h_1 - (4x + 13)\, h_2 - h_3 = 0 \tag{2.74}$$

$$g_2 = 8h_2 + yh_3 - 3h_4 = 0 \tag{2.75}$$

$$g_3 = h_3 = -8x + 6y - 43 \tag{2.76}$$

$$g_4 = h_4 = 2y^2 - 17y + 8 \tag{2.77}$$

$G = \{g_3, g_4\}$ is the *completely reduced* Gröbner basis, and has the same solutions as the original system H.

Several general observations can be made about the Gröbner basis reduction technique. First, note that completely reduced Gröbner basis G is comprised of polynomials that introduce the indeterminates one after the other. Consequently, rather than use $h_1(x, y)$

and $h_2(x, y)$ to solve for x and y simultaneously, $g_4(y)$ can be used to solve for y first, and the solution can then be substituted into $g_3(x, y)$ to solve for x. The pure lexicographic term ordering always triangularizes systems in this manner. Next, notice that Gröbner basis H'' contains twice as many polynomials as H, and the coefficients are generally larger. Intermediate expression swell and coefficient growth are well-known challenges associated with the Gröbner basis technique, and will be discussed in more detail below. Although the size of H doubled in this example, the number of intermediate polynomials can be substantially larger in practice. In fact, the problem of computing a Gröbner basis has a rather bleak complexity bound—exponential at best [133]—though, fortunately, the current algorithms typically demonstrate far superior performance. Finally, note that the algorithm presented above is the simplest form of the reduction procedure developed by Buchberger [15]. More advanced Gröbner basis algorithms carefully select the order in which S-polynomials are computed to avoid unnecessary reductions [14, 42, 46] and use linear algebra to reduce polynomials simultaneously [32,33,124]. Another popular strategy is to first generate a Gröbner basis using the graded reverse lexicographic ordering, which orders power products involving indeterminates x and y as follows:

$$1 \prec y \prec x \prec y^2 \prec xy \prec x^2 \prec xy^2 \prec x^2y \prec x^2y^2 \prec \dots \tag{2.78}$$

where $x \succ y$. Although it does not generally result in a triangular system, this term ordering typically results in the fastest Gröbner basis computation [22]. The resulting basis can then be converted into one relative to the desired term ordering—often the pure lexicographic ordering (2.59)—using, for example, the Faugère–Gianni–Lazard–Mora (FGLM) algorithm [34].

We shall illustrate the difference between the pure lexicographic and graded reverse lexicographic term orderings by revisiting the geometric example from Section 2.2.3. The following Gröbner basis for $\{f_1, f_2, f_3\}$ is generated by Maple using a pure lexicographic term ordering with $x \succ y \succ z$:

$$\begin{aligned} G_1 = \big[& 34953z^4 - 28960z^3 - 508448z^2 + 463360z - 812800, \\ & 272676096y + 803919z^3 - 11291792z^2 + 89390832z + 44330624, \\ & 272676096x + 4019595z^3 - 56458960z^2 + 37940016z + 221653120 \big] \end{aligned} \tag{2.79}$$

As in the dialytic elimination method, the fourth-degree polynomial in z can be factored:

$$\begin{aligned} &34953z^4 - 28960z^3 - 508448z^2 + 463360z - 812800 \\ &= (z-4)(z+4)\left(34953z^2 - 28960z + 50800\right) \end{aligned} \tag{2.80}$$

which has the same solutions as (2.45). Since G_1 contains polynomials that are linear in y and x, these variables can be solved immediately once z has been determined:

$$y = -\frac{267973}{90892032}z^3 + \frac{705737}{17042256}z^2 - \frac{1862309}{5680752}z - \frac{346333}{2130282} \tag{2.81}$$

$$x = -\frac{1339865}{90892032}z^3 + \frac{3528685}{17042256}z^2 - \frac{790417}{5680752}z - \frac{1731665}{2130282} \tag{2.82}$$

A Gröbner basis is now generated for the same system of polynomials, but using a graded reverse lexicographic term ordering with $x \succ y \succ z$:

$$\begin{aligned} G_2 = \big[\, & 2x - 10y - 3z, \quad 368yz - 65z^2 + 4864y + 1640z + 816, \\ & 736y^2 + 167z^2 - 6688y - 2232z - 1168, \\ & 803919z^3 - 11291792z^2 + 272676096y + 89390832z + 44330624 \,\big] \end{aligned} \tag{2.83}$$

Note that the coefficients in G_2 are generally smaller than those in G_1. In general, the coefficients of an S-polynomial will be larger than those of the original polynomials; in the worst case, they will be double the length. Since the graded reverse lexicographic ordering typically requires fewer S-polynomial computations than the pure lexicographic ordering [77], we expect smaller coefficients in the resulting basis. Note, however, that G_2 is not triangular; further processing would be necessary to obtain a recursively solvable system from this basis.

Gröbner bases have been applied to problems in a wide variety of areas, including statistics, coding theory, and automated geometric theorem proving [18]. In the kinematics community, Gröbner bases have typically been used for performing off-line analyses of specific mechanisms, including planar n-bar mechanisms [26], planar and spatial manipulators [63], and spatial parallel robots [35, 87, 92]. A unique application was proposed by Dhingra et al. [25], who generate Gröbner bases using a graded lexicographic term ordering, which can be computed efficiently, and use the resulting polynomials as new linearly independent equations for constructing the Sylvester matrix in the dialytic elimination method. A hitherto unexplored application, and the topic of Chapter 3, is the

use of Gröbner bases for the automated generation of real-time-capable dynamic simulation code, where the efficiency of triangular systems is particularly advantageous. We first review a topic of relatively recent development: the computation of Gröbner bases using floating-point arithmetic.

2.2.6 Approximate Gröbner bases

Until the mid-1990s, all Gröbner bases were exact—that is, they were computed from polynomials with exact coefficients using exact arithmetic. As illustrated in the previous section, however, the coefficients can grow substantially when exact arithmetic is used: though given a system of polynomials with coefficients no larger than 10, the maximum coefficient contained in Gröbner bases G_1 and G_2 is 272676096. Since the order in which power products are eliminated is entirely governed by the term ordering, Buchberger's algorithm is analogous to Gaussian elimination without pivoting, and is numerically unstable. One method of addressing this issue is to truncate the coefficients to a desired precision once an exact Gröbner basis has been generated. The primary concern, however, is the memory required for storing large coefficients, which is already burdened by a large number of intermediate polynomials. Floating-point Gröbner bases were first proposed in 1993 by Shirayanagi [115], who replaced each exact coefficient with a bracket coefficient consisting of a floating-point number and an approximation of its absolute error. In contrast to integers or rational numbers, floating-point numbers can represent very small and very large quantities using a fixed amount of memory, and are particularly suitable for problems in which the coefficients are only known approximately. Although Shirayanagi's method is proven to converge provided the computation is performed using a sufficient numerical precision, a method for determining a priori the precision required for a given problem does not yet exist [117]. Instead, a sequence of floating-point Gröbner bases is computed, incrementing the numerical precision until the resulting basis stabilizes. To explore this approach further, the author developed a Maple implementation of Buchberger's improved algorithm [15] using bracket coefficients, following the approach of Shirayanagi [116]; the implementation details are omitted here. Shown below are three examples that demonstrate the application of floating-point Gröbner bases to the solution of polynomial systems.

We first consider a perturbed version of set H from the previous section:

$$h_1 = 4x^2y + 9xy + 4x - 7y - 4 \tag{2.84}$$

$$\tilde{h}_2 = xy - y + 3.000001 \tag{2.85}$$

where we wish to solve $h_1 = 0$ and $\tilde{h}_2 = 0$ for x and y. The slight numerical perturbation in h_2 merely shifts the solutions a small amount. One method of generating a Gröbner basis for $\widetilde{H} = \{h_1, \tilde{h}_2\}$ is to convert all coefficients into integers or rational numbers and compute the Gröbner basis exactly. An exact representation of $\tilde{h}_2$ is obtained using the `convert/rational` function in Maple:

$$\hat{h}_2 = xy - y + \frac{3000001}{1000000} \tag{2.86}$$

which results in the following exact Gröbner basis, generated using a pure lexicographic term ordering with $x \succ y$:

$$\begin{aligned}\widehat{G} = \big[\, & 1500000000000y^2 - 12750004250000y + 6000005000001, \\ & 8000004x - 6000000y + 43000013 \,\big]\end{aligned} \tag{2.87}$$

Despite being mathematically correct, this solution is not entirely satisfying: since H and $\widetilde{H}$ are identical but for a single slightly perturbed coefficient, it would be preferable to obtain a basis whose coefficients are more similar to those of the original basis, namely, (2.76) and (2.77). Of course, we could simply divide each polynomial by the coefficient of its leading term, but it would be advantageous to avoid large coefficients altogether. Using our Maple implementation of Shirayanagi's method, we first generate the following sequence of floating-point Gröbner bases for the original system H:

$$\mathcal{G}^{(2)} = \big[\, y^2 - 9.3y + 4.3, \qquad x - 0.70y + 5.5 \,\big] \tag{2.88}$$

$$\mathcal{G}^{(3)} = \big[\, y^2 - 8.53y + 4.00, \qquad x - 0.750y + 5.40 \,\big] \tag{2.89}$$

$$\mathcal{G}^{(4)} = \big[\, y^2 - 8.500y + 4.000, \qquad x - 0.7500y + 5.375 \,\big] \tag{2.90}$$

$$\mathcal{G}^{(5)} = \big[\, y^2 - 8.5000y + 4.0000, \quad x - 0.75000y + 5.3750 \,\big] \tag{2.91}$$

where the parenthetic superscript denotes the number of digits of precision used. Since the progression from four digits of precision to five only serves to suffix each coefficient

with a zero, we can conclude—albeit, tentatively—that we need not proceed further. The following sequence of bases is obtained for the perturbed system $\widetilde{H}$:

$$\begin{array}{cc} \vdots & \vdots \end{array}$$

$$\widetilde{\mathcal{G}}^{(6)} = \left[\, y^2 - 8.50000y + 4.00000, \quad x - 0.750000y + 5.37500 \,\right] \tag{2.92}$$

$$\widetilde{\mathcal{G}}^{(7)} = \left[\, y^2 - 8.500001y + 4.000003, \quad x - 0.7499996y + 5.374997 \,\right] \tag{2.93}$$

$$\widetilde{\mathcal{G}}^{(8)} = \left[\, y^2 - 8.5000027y + 4.0000034, \quad x - 0.74999962y + 5.3749988 \,\right] \tag{2.94}$$

where bases $\mathcal{G}^{(k)}$ and $\widetilde{\mathcal{G}}^{(k)}$ are equal for $k \leq 6$, which is expected since the perturbation in $\tilde{h}_2$ occurs in the seventh significant digit. Note that the coefficients are not permitted to grow arbitrarily long, as they were in the exact Gröbner basis computation.

Now consider the following overdetermined system $P = \{p_1, p_2, p_3\}$:

$$p_1 = x^2 + y^2 - 6x - 2y + 5 \tag{2.95}$$

$$p_2 = x^2 - 6x - y + 7 \tag{2.96}$$

$$p_3 = x - 4y + 3 \tag{2.97}$$

which corresponds to finding the points of intersection of a circle, a parabola, and a line in $\mathbb{R}^2$. In its exact form, P has one solution at $(x, y) = (5, 2)$, which is identified by an exact Gröbner basis computation. A small numerical perturbation is now introduced in p_3:

$$\tilde{p}_3 = x - 4y + 3.000001 \tag{2.98}$$

In the previous example, the numerical perturbation merely shifted the solutions a small amount. In this case, since the original system P is overdetermined, an exact Gröbner basis algorithm will report that the system is inconsistent and no solution exists. Using Shirayanagi's method, the following sequence of floating-point Gröbner bases is obtained for the perturbed system $\widetilde{P} = \{p_1, p_2, \tilde{p}_3\}$, again using a pure lexicographic term ordering with $x \succ y$:

$$\widetilde{\mathcal{Q}}^{(2)} = \left[\, y - 2.0, \quad x - 5.0 \,\right] \tag{2.99}$$

$$\begin{array}{cc} \vdots & \vdots \end{array}$$

$$\widetilde{\mathcal{Q}}^{(6)} = \left[\, y - 2.00000, \quad x - 5.00000 \,\right] \tag{2.100}$$

$$\widetilde{\mathcal{Q}}^{(7)} = \left[\, y - 2.000000, \quad x - 4.999999 \,\right] \tag{2.101}$$

$$\widetilde{\mathcal{Q}}^{(8)} = \left[\, y - 1.9999999, \quad x - 4.9999986 \,\right] \tag{2.102}$$

which indicates that the perturbed system $\widetilde{P}$ has the same solution as the original system P when fewer than seven digits of precision are used, as expected.

For our final example, we consider a modified version of system P:

$$\check{p}_1 = x^2 + y^2 - 6x - \frac{3e\pi - \varphi}{12}\, y + \frac{10\sqrt{26} - 1}{10} \tag{2.103}$$

$$\check{p}_2 = x^2 - \frac{21\sqrt{3} - 3\sqrt{17}}{4}\, x - y + 7 \tag{2.104}$$

where φ is the golden ratio and the following expressions have been used in place of the original coefficients:

$$\frac{3e\pi - \varphi}{12} = 2.00009738\ldots \approx 2 \tag{2.105}$$

$$\frac{10\sqrt{26} - 1}{10} = 4.99901951\ldots \approx 5 \tag{2.106}$$

$$\frac{21\sqrt{3} - 3\sqrt{17}}{4} = 6.00093752\ldots \approx 6 \tag{2.107}$$

Note that p_3 has been removed, so system $\check{P} = \{\check{p}_1, \check{p}_2\}$ is not overdetermined. Although all coefficients in system $\check{P}$ are known exactly, an exact Gröbner basis cannot be computed without first eliminating all irrational numbers. Square roots can be eliminated by introducing new variables $z_k = \sqrt{k}$ and appending auxiliary equations of the form $z_k^2 - k = 0$; however, increasing the number of indeterminates can significantly increase the Gröbner basis computation time. We also note that transcendental constants e and π cannot be removed in this fashion. A common approach is to replace each coefficient with a rational approximation, though accurately approximating irrational numbers generally requires many digits which, again, has consequences in terms of both computation time and memory use. Furthermore, rational approximations can make overdetermined systems inconsistent. Using our Maple implementation of Shirayanagi's method, we obtain the following sequence of floating-point Gröbner bases, again using a pure lexicographic term

ordering with $x \succ y$:

$$\check{\mathcal{Q}}^{(3)} = \left[\, y^2 - 1.00y - 2.00, \quad x^2 - 6.00x - 1.00y + 7.00 \,\right] \tag{2.108}$$

$$\check{\mathcal{Q}}^{(4)} = \left[\, y^2 - 1.000y - 2.001, \quad x^2 - 6.000x - 1.000y + 7.000 \,\right] \tag{2.109}$$

$$\begin{aligned}\check{\mathcal{Q}}^{(5)} = \big[\, & y^4 - 2.0003y^3 - 2.9964y^2 + 3.9971y + 3.9933, \\ & x + 1111.1y^2 - 1111.2y - 2223.3 \,\big]\end{aligned} \tag{2.110}$$

$$\begin{aligned}\check{\mathcal{Q}}^{(6)} = \big[\, & y^4 - 2.00021y^3 - 2.99611y^2 + 3.99675y + 3.99264, \\ & x + 1063.83y^2 - 1063.94y - 2128.70 \,\big]\end{aligned} \tag{2.111}$$

Note that $\check{\mathcal{Q}}^{(4)}$ and $\check{\mathcal{Q}}^{(5)}$ differ in their coefficients as well as their power products, which are the two main indicators that an insufficient numerical precision has been used.

As demonstrated in the above examples, floating-point Gröbner bases inhibit the growth of coefficients, and are particularly useful for solving systems whose coefficients are irrational or only known approximately. A fundamental challenge for all approximate Gröbner basis approaches is determining, in a numerically stable way, whether a coefficient is zero. Incorrectly identifying zeros can lead to erroneous cancellations during the reduction process, in which case all ensuing computations, as well as the resulting Gröbner basis, will be incorrect. The zero criterion used by Shirayanagi [116] is simple: if the absolute value of the coefficient is less than the approximation of its error, then assume the coefficient is zero. A weakness of interval arithmetic, however, is that it often overestimates the effect of data inaccuracy, leading to erroneous cancellations. Sasaki and Kako [104] use bracket coefficients in conjunction with other stabilization strategies. Traverso and Zanoni [129] use two numerical precisions concurrently, defining zero coefficients to be those whose single- and double-precision floating-point representations are sufficiently dissimilar. A similar approach is used by Lichtblau [78], the developer of the `GroebnerBasis` function in Mathematica, who also demonstrates that Gröbner bases can be computed significantly faster when using approximate arithmetic that avoids large coefficients [79]. A related approach uses modular, single-precision, double-precision, and interval arithmetic simultaneously [10]. Stetter [123] observed that small perturbations in the coefficients of the original system can lead to Gröbner bases with entirely different structures, as demonstrated by $\check{\mathcal{Q}}^{(4)}$ (2.109) and $\check{\mathcal{Q}}^{(5)}$ (2.110) above. This discontinuous behaviour can be addressed by relaxing the term ordering in favour of preserving numerical stability. In particular,

polynomials whose leading power products have small coefficients should not be used for generating or reducing S-polynomials. Kondratyev et al. [71] avoid such situations by explicitly encoding the magnitude of the coefficients into new variables, which are included in the term ordering. This strategy is analogous to the use of pivoting in Gaussian elimination to preserve numerical stability. The stable computation of approximate Gröbner bases remains an active area of research.

2.3 Chapter summary

In this chapter, we have examined several techniques for solving the equations governing the motion of closed-kinematic-chain systems, including index reduction, constraint stabilization, and system modification approaches. A strategy of particular interest involves solving the constraint equations separately from the dynamic equations, a task for which Gröbner bases appear to be particularly suitable. Although approximate Gröbner bases exhibit several advantages over exact Gröbner bases, only the latter is considered in the remainder of this work for reasons of efficiency, simplicity, and convenience. First, whereas very efficient algorithms have been developed for the computation of exact Gröbner bases, the work on approximate Gröbner basis algorithms has focused primarily on maintaining numerical stability, often at the expense of computational efficiency. Secondly, the use of exact Gröbner bases avoids the need to determine a suitable numerical precision for each system, which can involve repeating lengthy computations several times. Finally, since the equations of motion are generated in this work using the Multibody library in MapleSim, it is convenient to employ the efficient exact Gröbner basis algorithms already available in Maple. As will be shown, the use of exact Gröbner bases is adequate for many practical problems.

Chapter 3

Gröbner Basis Approach

In this chapter, we explore the use of Gröbner bases for generating triangular, or recursively solvable, systems of kinematic equations. In contrast to other methods for solving systems of polynomial equations, as discussed in Section 2.2.5, a Gröbner basis can be obtained algorithmically, and generates a computationally efficient system of equations with the same solutions as the original system. For systems that can be fully triangularized, the kinematic constraints are always satisfied exactly and in a fixed amount of time, the latter of which is a particularly advantageous property in real-time applications. Where full triangularization is not possible, a block-triangular form can be obtained that still results in more efficient simulations than existing iterative and constraint stabilization techniques. Although traditionally used as a mathematical tool for the analysis of specific mechanisms, the objective here is to propose a strategy for incorporating Gröbner bases into an automated formulation procedure. A simple planar mechanism will be used throughout this chapter to demonstrate each step in the process, from formulating the equations of motion to generating optimized dynamic simulation code. The chapter concludes with a summary of the Gröbner basis approach.

3.1 Formulating equations of motion

In this section, we briefly describe the graph-theoretic formulation used throughout this work. Although the Gröbner basis approach can be applied to systems generated using any symbolic formulation procedure, familiarity with concepts from linear graph theory

will facilitate later discussion. We begin with a description of the system that will serve as an example throughout the chapter, and then generate the differential-algebraic equations governing its motion.

3.1.1 System description

The planar slider-crank mechanism, shown in Figure 3.1, consists of four bodies: a crank of mass m_1 and length ℓ_1, a connecting rod of mass m_2 and length ℓ_2, a piston of mass m_3, and the ground. Revolute joints attach the crank to the ground, the connecting rod to the crank, and the piston to the connecting rod; the piston is also attached to the ground with a prismatic joint, which completes the closed kinematic chain. Gravity acts in the $-$Y direction, and a time-varying force $F(t)$ is applied to the piston, as shown. The number of degrees-of-freedom (DOF) can be determined using the Chebychev–Grübler–Kutzbach criterion [47]:

$$f = 3\,(n_b - 1) - 2n_j \tag{3.1}$$

where n_b is the number of bodies and n_j is the number of joints. Thus, this mechanism has $f = 1$ DOF—that is, its motion can be controlled using only one actuator, such as a motor mounted between the ground and the crank.

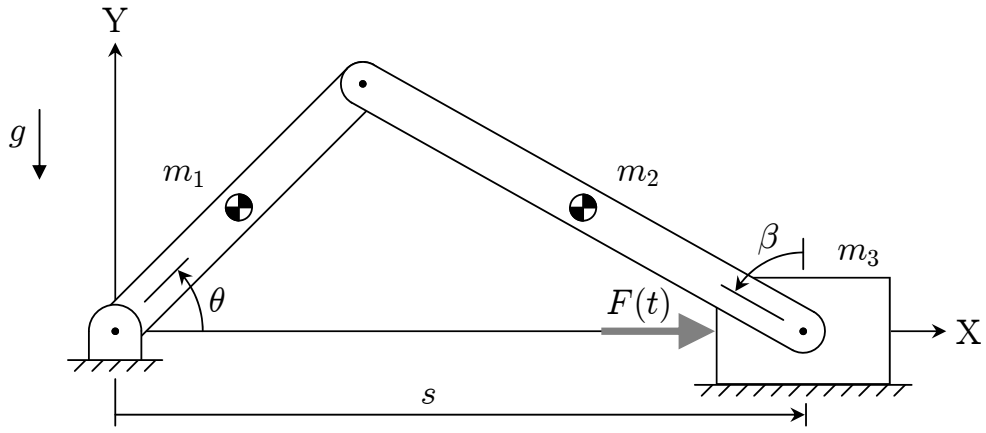


Figure 3.1: Planar slider-crank mechanism

3.1.2 Graph-theoretic formulation

In this work, the graph-theoretic formulation proposed by McPhee [81] is used to generate the equations of motion. The benefits of this approach include its ability to automatically generate very efficient symbolic equations in terms of any desired set of modelling coordinates, and its ability to model multi-domain systems seamlessly. In this section, the equations of motion for the planar slider-crank mechanism are derived manually to demonstrate the graph-theoretic formulation procedure. A Maple implementation of this approach [82] has been incorporated into the Multibody library in MapleSim, which is used in the remainder of this work to generate symbolic equations of motion automatically.

The first step in the procedure involves representing the topology of the system as a linear graph, where the nodes represent body-fixed reference frames and the edges represent the *through* and *across* variables associated with physical components such as rigid bodies, joints, and external forces. Through variables include forces $\underline{F}$ and torques $\underline{T}$, and represent quantities that flow through components; across variables include displacements, velocities, and accelerations. The linear graph corresponding to the planar slider-crank mechanism is shown in Figure 3.2. Edges $\vec{m}_1$, $\vec{m}_2$, and $\vec{m}_3$ begin at the ground node and terminate

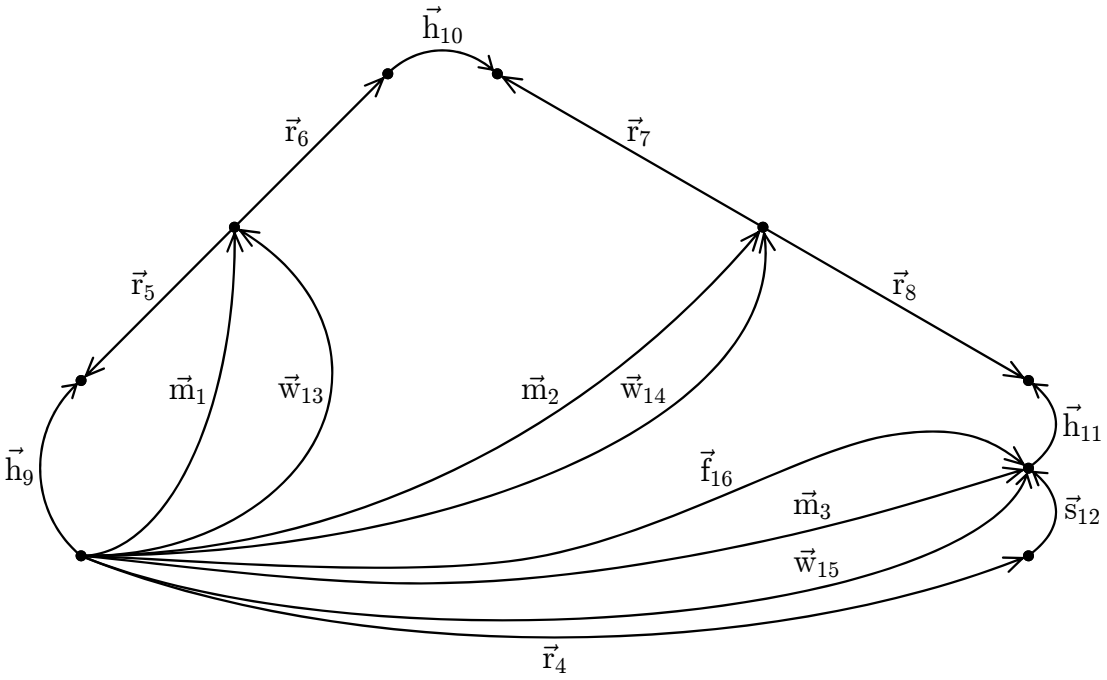


Figure 3.2: Linear graph representation of planar slider-crank mechanism

at nodes representing the mass centers of the other three bodies in the system. Rigid-arm elements $\vec{r}_4$ to $\vec{r}_8$ represent kinematic transformations between nodes on these bodies, and can involve both translations and rotations in general—though all rigid-arm elements are purely translational in this example. The revolute (hinge) and prismatic (slider) joints are represented by edges $\vec{h}_9$, $\vec{h}_{10}$, $\vec{h}_{11}$, and $\vec{s}_{12}$, where the displacement of the node at the head is measured relative to that at the tail. Finally, edges $\vec{w}_{13}$, $\vec{w}_{14}$, $\vec{w}_{15}$, and $\vec{f}_{16}$ represent external forces, the first three of which are gravitational.

The equations of motion are derived by combining linear topological equations with nonlinear constitutive equations, the latter of which define the physical behaviour of the corresponding component. For example, the translational and rotational motions of the crank are governed by the following constitutive equations [81]:

$$\underline{F}_1 = -m_1 \underline{a}_1 \tag{3.2}$$

$$\underline{T}_1 = -I_1 \underline{\alpha}_1 - \underline{r}_5 \times \underline{F}_5 - \underline{r}_6 \times \underline{F}_6 \tag{3.3}$$

where I_1 is the inertia of the crank about its center of mass, $\underline{a}_1$ and $\underline{\alpha}_1$ are its translational and rotational accelerations, and the vector cross products in (3.3) are the torques about the center of mass generated by the reaction forces applied to the ends of the crank. The constitutive equations for rigid-arm elements $\vec{r}_5$ and $\vec{r}_6$ are determined geometrically:

$$\underline{r}_5 = -\frac{\ell_1}{2}\cos(\theta_1)\,\hat{\imath} - \frac{\ell_1}{2}\sin(\theta_1)\,\hat{\jmath} \tag{3.4}$$

$$\underline{r}_6 = +\frac{\ell_1}{2}\cos(\theta_1)\,\hat{\imath} + \frac{\ell_1}{2}\sin(\theta_1)\,\hat{\jmath} \tag{3.5}$$

where $\hat{\imath}$ and $\hat{\jmath}$ are unit vectors along the X- and Y-axes. Ideal revolute joints permit the relative rotation of two bodies, but prohibit their relative translation:

$$\underline{T}_h \cdot \hat{k} = 0 \tag{3.6}$$

$$\underline{r}_h = 0 \tag{3.7}$$

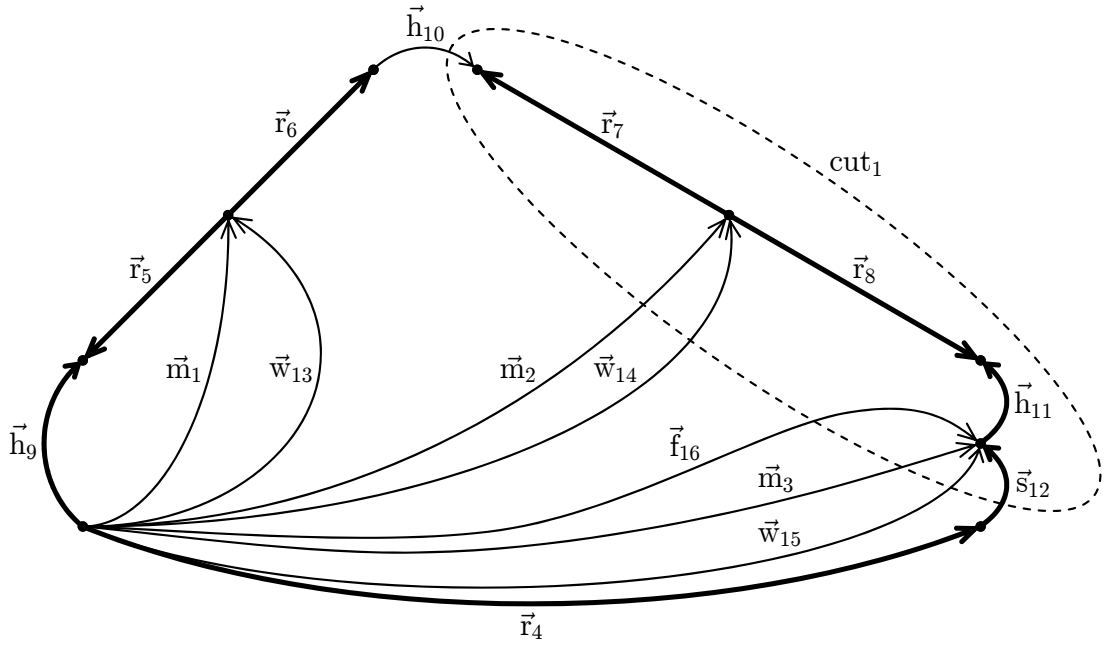
where $\hat{k}$ is the axis of rotation for all revolute joints in this planar system. Similarly, the ideal prismatic joint in this system allows the piston to slide along the X-axis, but prohibits all other motion:

$$\underline{F}_{12} \cdot \hat{\imath} = 0 \tag{3.8}$$

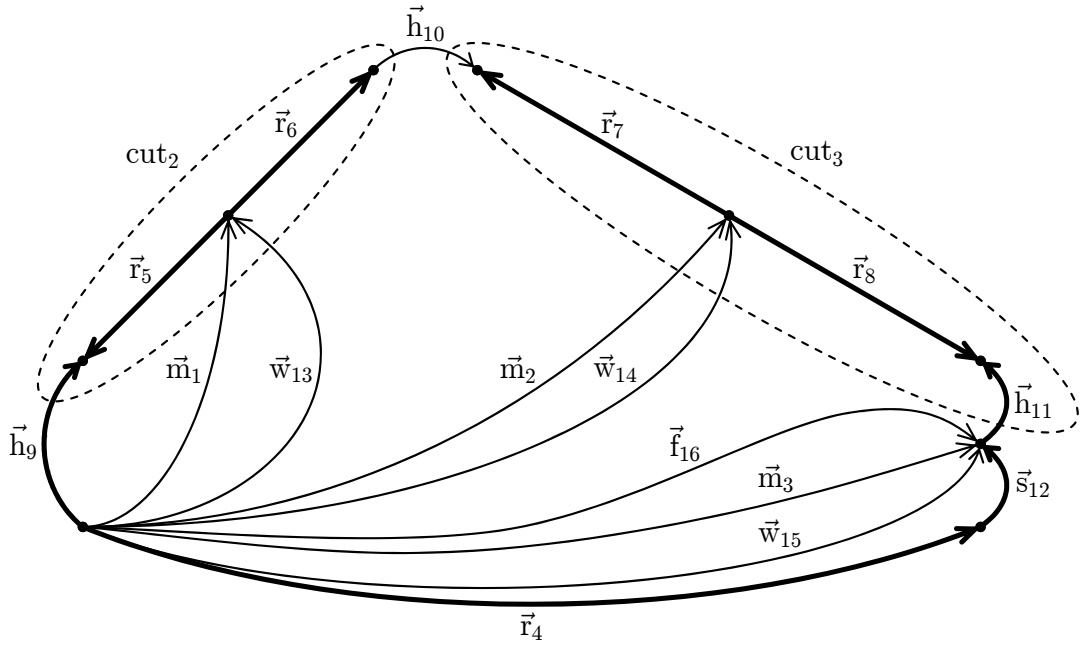
$$\underline{\omega}_{12} = 0 \tag{3.9}$$

In order to generate the topological equations, we must first select *spanning trees* for the translational and rotational domains. A spanning tree is a set of edges, referred to as *branches*, that connects all the nodes in a linear graph without creating closed loops; the remaining edges, called *chords*, form the complement of the tree, or the cotree [12]. The tree selection step is critical, as the spanning trees determine the generalized coordinates that will appear in the equations of motion. Guidelines for selecting spanning trees that will result in the most efficient system of dynamic equations are provided by Léger and McPhee [75]. In this example, we wish to obtain equations in terms of joint coordinates $\mathbf{q} = \{\theta, \beta, s\}$, which can be accomplished by selecting the spanning trees shown in Figure 3.3. Note that placing rigid body components in the trees results in an absolute coordinate formulation, where the configuration of each body is described by its position and orientation relative to the ground frame. Also shown in Figure 3.3 are the three fundamental cutsets (f-cutsets) that will be transformed into the equations of motion. Each f-cutset consists of exactly one branch, and is the smallest set of edges required to cut the graph into two parts [38]. The remaining f-cutsets are used to express the branch through variables as linear combinations of the chord through variables, which are called *chord transformations*. Once the spanning trees have been selected, we can also identify fundamental circuits (f-circuits), which are connected subgraphs consisting of exactly one chord and passing through each node in the graph no more than once [38]. The f-circuits are used to express the chord across variables in terms of the branch across variables, which are called *branch transformations*. The branch across variables and chord through variables are known as *primary variables*; the branch through variables and chord across variables are called *secondary variables*, and can be expressed in terms of the primary variables using the branch and chord transformations.

Once the system topology has been defined and the spanning trees have been selected, the equations of motion can be generated entirely automatically using the three-step procedure described by McPhee [81]. First, the f-cutset equation for each branch is projected onto the motion space associated with the corresponding component. Since the prismatic joint permits translation along the X-axis and the revolute joints permit rotation about the axis perpendicular to the plane, we project cut_1 onto $\hat{\imath}$ and the other two f-cutset equations



(a) Translational domain



(b) Rotational domain

Figure 3.3: Spanning trees, shown in bold, for joint coordinate formulation of planar slider-crank mechanism

onto $\hat{k}$:

$$\text{cut}_1 \quad \rightarrow \quad (\underline{F}_2 + \underline{F}_3 + \underline{F}_{10} + \underline{F}_{12} + \underline{F}_{14} + \underline{F}_{15} + \underline{F}_{16}) \cdot \hat{\imath} = 0 \tag{3.10}$$

$$\text{cut}_2 \quad \rightarrow \quad (\underline{T}_1 + \underline{T}_9 - \underline{T}_{10} + \underline{T}_{13}) \cdot \hat{k} = 0 \tag{3.11}$$

$$\text{cut}_3 \quad \rightarrow \quad (\underline{T}_2 + \underline{T}_{10} + \underline{T}_{11} + \underline{T}_{14}) \cdot \hat{k} = 0 \tag{3.12}$$

Note that the f-cutset equation associated with branch $\vec{\text{s}}_{12}$ in the rotational graph is projected onto $\hat{\imath}$ and $\hat{\jmath}$, which are both orthogonal to the rotational domain and, thus, result in trivial equations; the f-cutset equations associated with branches $\vec{\text{h}}_9$ and $\vec{\text{h}}_{11}$ in the translational graph are similarly null. Substituting the constitutive equations into (3.10), (3.11), and (3.12) results in the following three second-order ODEs, one for each generalized coordinate:

$$\text{cut}_1 \quad \rightarrow \quad -m_2\,(\underline{a}_2 \cdot \hat{\imath}) - m_3\,(\underline{a}_3 \cdot \hat{\imath}) + R_X + F(t) = 0 \tag{3.13}$$

$$\text{cut}_2 \quad \rightarrow \quad -I_1\big(\underline{\alpha}_1 \cdot \hat{k}\big) - (\underline{r}_5 \times \underline{F}_5) \cdot \hat{k} - (\underline{r}_6 \times \underline{F}_6) \cdot \hat{k} = 0 \tag{3.14}$$

$$\text{cut}_3 \quad \rightarrow \quad -I_2\big(\underline{\alpha}_2 \cdot \hat{k}\big) - (\underline{r}_7 \times \underline{F}_7) \cdot \hat{k} - (\underline{r}_8 \times \underline{F}_8) \cdot \hat{k} = 0 \tag{3.15}$$

where $\underline{F}_{16} = F(t)\hat{\imath}$ is the force applied to the piston, and $\underline{F}_{10} = R_X\hat{\imath} + R_Y\hat{\jmath}$ is the reaction force in the pin between the crank and the connecting rod.

Next, the f-circuit equations for all kinematic constraints are projected onto the reaction space associated with the corresponding component, which produces the kinematic constraint equations. The two f-circuits in this example traverse the perimeter of each graph shown in Figure 3.3. Since the reaction space of $\vec{h}_{10}$ is orthogonal to the rotational domain, only the f-circuit equation from the translational domain is nontrivial:

$$(\underline{r}_{10} - \underline{r}_7 + \underline{r}_8 - \underline{r}_{11} - \underline{r}_{12} - \underline{r}_4 + \underline{r}_9 - \underline{r}_5 + \underline{r}_6) \cdot \hat{\imath} = 0 \tag{3.16}$$

$$(\underline{r}_{10} - \underline{r}_7 + \underline{r}_8 - \underline{r}_{11} - \underline{r}_{12} - \underline{r}_4 + \underline{r}_9 - \underline{r}_5 + \underline{r}_6) \cdot \hat{\jmath} = 0 \tag{3.17}$$

where $\underline{r}_5$ and $\underline{r}_6$ are defined in (3.4) and (3.5), $\underline{r}_7$ and $\underline{r}_8$ are of an analogous form, $\underline{r}_4 = 0$ since the slider axis passes through the origin, and $\underline{r}_9 = \underline{r}_{10} = \underline{r}_{11} = 0$, as defined in (3.7). Upon substitution of these expressions, we obtain the following two kinematic constraint equations:

$$\ell_1 \cos(\theta) + \ell_2 \sin(\beta) - s = 0 \tag{3.18}$$

$$\ell_1 \sin(\theta) - \ell_2 \cos(\beta) = 0 \tag{3.19}$$

The constraints can be readily verified geometrically, as they simply ensure that the crank and connecting rod meet at their common pin joint.

In the third and final step of the procedure, the branch and chord transformations are used to eliminate secondary variables from (3.13), (3.14), and (3.15). First, the rigid-arm forces are eliminated using f-cutset equations from the translational graph:

$$\underline{F}_5 = \underline{F}_1 - \underline{F}_{10} + \underline{F}_{13} = -m_1\underline{a}_1 - (R_X\hat{\imath} + R_Y\hat{\jmath}) - m_1 g\,\hat{\jmath} \tag{3.20}$$

$$\underline{F}_6 = \underline{F}_{10} = R_X\hat{\imath} + R_Y\hat{\jmath} \tag{3.21}$$

$$\underline{F}_7 = -\underline{F}_{10} = -\left(R_X\hat{\imath} + R_Y\hat{\jmath}\right) \tag{3.22}$$

$$\underline{F}_8 = \underline{F}_2 + \underline{F}_{10} + \underline{F}_{14} = -m_2\underline{a}_2 + R_X\hat{\imath} + R_Y\hat{\jmath} - m_2 g\,\hat{\jmath} \tag{3.23}$$

The rigid bodies in the translational cotree are then eliminated using f-circuit equations from the translational graph:

$$\underline{a}_1 = -\underline{a}_5 + \underline{a}_9 = -\left(\underline{\alpha}_1 \times \underline{r}_5 + \omega_1^2\underline{r}_5\right) \tag{3.24}$$

$$\underline{a}_2 = -\underline{a}_8 + \underline{a}_{11} + \underline{a}_{12} + \underline{a}_4 = -\left(\underline{\alpha}_2 \times \underline{r}_8 + \omega_2^2\underline{r}_8\right) + \ddot{s}\,\hat{\imath} \tag{3.25}$$

$$\underline{a}_3 = \underline{a}_{12} + \underline{a}_4 = \ddot{s}\,\hat{\imath} \tag{3.26}$$

Finally, the rigid bodies in the rotational cotree are eliminated using f-circuit equations from the rotational graph:

$$\underline{\alpha}_1 = -\underline{\alpha}_5 + \underline{\alpha}_9 = \ddot{\theta}\hat{k} \tag{3.27}$$

$$\underline{\alpha}_2 = -\underline{\alpha}_8 + \underline{\alpha}_{11} + \underline{\alpha}_{12} + \underline{\alpha}_4 = \ddot{\beta}\hat{k} \tag{3.28}$$

Upon eliminating all secondary variables from (3.13), (3.14), and (3.15), we obtain the following ODEs in terms of the generalized coordinates $\mathbf{q} = \{\theta, \beta, s\}$ and system parameters:

$$\text{cut}_1 \quad \rightarrow \quad -\left(m_2 + m_3\right)\ddot{s} + \tfrac{1}{2}m_2\ell_2\mathrm{c}_\beta\ddot{\beta} + R_X - \tfrac{1}{2}m_2\ell_2\mathrm{s}_\beta\dot{\beta}^2 + F(t) = 0 \tag{3.29}$$

$$\text{cut}_2 \quad \rightarrow \quad -\left(I_1 + \tfrac{1}{4}m_1\ell_1^2\right)\ddot{\theta} + \ell_1\mathrm{s}_\theta R_X - \ell_1\mathrm{c}_\theta R_Y - \tfrac{1}{2}m_1\ell_1 g\mathrm{c}_\theta = 0 \tag{3.30}$$

$$\text{cut}_3 \quad \rightarrow \quad -\left(I_2 + \tfrac{1}{4}m_2\ell_2^2\right)\ddot{\beta} + \tfrac{1}{2}m_2\ell_2\mathrm{c}_\beta\ddot{s} - \ell_2\mathrm{c}_\beta R_X - \ell_2\mathrm{s}_\beta R_Y + \tfrac{1}{2}m_2\ell_2 g\mathrm{s}_\beta = 0 \tag{3.31}$$

where $\sin(\varphi)$ and $\cos(\varphi)$ have been abbreviated s_φ and c_φ. Assembling these ODEs with

constraint equations (3.18) and (3.19), we obtain the following system of index-3 DAEs:

$$\begin{bmatrix} I_1 + \frac{1}{4}m_1\ell_1^2 & 0 & 0 \\ 0 & I_2 + \frac{1}{4}m_2\ell_2^2 & -\frac{1}{2}m_2\ell_2\mathrm{c}_\beta \\ 0 & -\frac{1}{2}m_2\ell_2\mathrm{c}_\beta & m_2 + m_3 \end{bmatrix} \begin{Bmatrix} \ddot{\theta} \\ \ddot{\beta} \\ \ddot{s} \end{Bmatrix} + \begin{bmatrix} -\ell_1\mathrm{s}_\theta & \ell_1\mathrm{c}_\theta \\ \ell_2\mathrm{c}_\beta & \ell_2\mathrm{s}_\beta \\ -1 & 0 \end{bmatrix} \begin{Bmatrix} R_X \\ R_Y \end{Bmatrix} = \begin{Bmatrix} -\frac{1}{2}m_1\ell_1 g\mathrm{c}_\theta \\ \frac{1}{2}m_2\ell_2 g\mathrm{s}_\beta \\ -\frac{1}{2}m_2\ell_2\mathrm{s}_\beta\dot{\beta}^2 + F(t) \end{Bmatrix} \tag{3.32}$$

$$\begin{Bmatrix} \ell_1\mathrm{c}_\theta + \ell_2\mathrm{s}_\beta - s \\ \ell_1\mathrm{s}_\theta - \ell_2\mathrm{c}_\beta \end{Bmatrix} = \mathbf{0} \tag{3.33}$$

which are of the general form shown in (2.1) and (2.2). These equations govern the motion of the planar slider-crank mechanism, and serve as the starting-point for the Gröbner basis approach. The next two sections focus on triangularizing the constraint equations (3.33); the further processing of the dynamic equations (3.32) is discussed in Section 3.4.1.

3.2 Preparing constraints for triangularization

In this section, we prepare the equations for which a Gröbner basis will be generated. In particular, the goniometric kinematic constraint equations must be converted into polynomial equations, and the presence of floating-point coefficients must be addressed.

3.2.1 Goniometric equations

As discussed in Section 2.2.1, goniometric equations are more difficult to solve than polynomial equations. As such, one of three methods is typically employed to convert the former into the latter. The resulting system of polynomial equations can then be solved using a variety of techniques. In this section, we explore the application of each of the three aforementioned substitution techniques to the kinematic constraint equations for the planar slider-crank mechanism.

We first consider the tangent-half-angle substitution:

$$x_1 = \tan(\theta/2) \qquad \rightarrow \qquad \sin(\theta) = \frac{2x_1}{1+x_1^2}, \quad \cos(\theta) = \frac{1-x_1^2}{1+x_1^2} \tag{3.34}$$

$$x_2 = \tan(\beta/2) \qquad \rightarrow \qquad \sin(\beta) = \frac{2x_2}{1+x_2^2}, \quad \cos(\beta) = \frac{1-x_2^2}{1+x_2^2} \tag{3.35}$$

Substituting into the constraint equations (3.33) and multiplying by $(1+x_1^2)\,(1+x_2^2)$ to clear the denominators, we obtain a system of degree 5 in $\{s, x_1, x_2\}$:

$$sx_1^2x_2^2 + \ell_1x_1^2x_2^2 + sx_1^2 + sx_2^2 - 2\ell_2x_1^2x_2 + \ell_1x_1^2 - \ell_1x_2^2 + s - 2\ell_2x_2 - \ell_1 = 0 \tag{3.36}$$

$$\ell_2x_1^2x_2^2 + 2\ell_1x_1x_2^2 - \ell_2x_1^2 + \ell_2x_2^2 + 2\ell_1x_1 - \ell_2 = 0 \tag{3.37}$$

Using geometric parameters $\{\ell_1 = 3/10, \ell_2 = 2/5\}$ and a pure lexicographic term ordering with $s \succ x_1 \succ x_2$, we obtain the following exact Gröbner basis:

$$\begin{aligned}
&\big[\, 2x_1^2x_2^2 + 3x_1x_2^2 - 2x_1^2 + 2x_2^2 + 3x_1 - 2, \\
&\;\; 10sx_2^4 - 4x_1x_2^4 - 3x_2^4 + 20sx_2^2 - 8x_2^3 - 6x_2^2 + 10s + 4x_1 - 8x_2 - 3, \\
&\;\; 30sx_1x_2^2 - 40sx_1^2 + 16x_1^2x_2 + 9x_1x_2^2 + 30sx_1 - 12x_1^2 + 12x_2^2 - 40s + 9x_1 + 16x_2 \,\big]
\end{aligned} \tag{3.38}$$

Note that (3.38) is triangular, since the first polynomial can be solved for x_1 as a function of x_2, and the second and third polynomials can be solved for s as functions of x_1 and x_2:

$$\begin{aligned}
\Delta_1 = \Bigg[& x_1 = \frac{-3x_2^2 - 3 \pm \sqrt{-7x_2^4 + 50x_2^2 - 7}}{4\,(x_2^2 - 1)}, \quad s = \frac{4x_1x_2^2 + 3x_2^2 - 4x_1 + 8x_2 + 3}{10\,(x_2^2 + 1)}, \\
& s = \frac{16x_1^2x_2 + 9x_1x_2^2 - 12x_1^2 + 12x_2^2 + 9x_1 + 16x_2}{10\,(-3x_1x_2^2 + 4x_1^2 - 3x_1 + 4)} \Bigg]
\end{aligned} \tag{3.39}$$

Thus, given a numerical value for x_2, the first expression in Δ_1 can be used to determine the corresponding value of x_1, whereupon either of the other two expressions can be used to calculate the slider displacement s.

Now consider the Euler substitution method, the application of which requires the following definitions:

$$y_1 = e^{j\theta} \qquad \rightarrow \qquad \sin(\theta) = \frac{1-y_1^2}{2y_1}\,j, \quad \cos(\theta) = \frac{1+y_1^2}{2y_1} \tag{3.40}$$

$$y_2 = e^{j\beta} \qquad \rightarrow \qquad \sin(\beta) = \frac{1-y_2^2}{2y_2}\,j, \quad \cos(\beta) = \frac{1+y_2^2}{2y_2} \tag{3.41}$$

where $j = \sqrt{-1}$. Again substituting into the constraint equations (3.33) and clearing the denominators, we obtain a system of degree 4 in $\{s, y_1, y_2, j\}$:

$$\ell_2 y_1 y_2^2 j + 2s y_1 y_2 - \ell_1 y_1^2 y_2 - \ell_2 y_1 j - \ell_1 y_2 = 0 \tag{3.42}$$

$$\ell_1 y_1^2 y_2 j + \ell_2 y_1 y_2^2 - \ell_1 y_2 j + \ell_2 y_1 = 0 \tag{3.43}$$

$$j^2 + 1 = 0 \tag{3.44}$$

Note that j has been appended to the list of indeterminates, since all variables are treated as such when generating a Gröbner basis, and a new equation has been added to the system. Using the same parameters as before and a pure lexicographic ordering with $s \succ y_1 \succ y_2 \succ j$, we obtain the following Gröbner basis:

$$\begin{aligned}\big[\, & j^2 + 1, \quad -4y_1 y_2^2 j + 3y_1^2 y_2 - 4y_1 j - 3y_2, \quad 4y_2^3 j + 10 s y_2^2 - 3y_1 y_2^2, \\ & -30 s y_2 j + 25 y_1 y_2 j + 40 s y_1 - 12 y_1^2 + 12 y_2^2 \,\big]\end{aligned} \tag{3.45}$$

from which triangular system Δ_2 can be extracted:

$$\begin{aligned}\Delta_2 = \Bigg[& y_1 = \frac{2y_2^2 j + 2j \pm \sqrt{-4y_2^4 + y_2^2 - 4}}{3y_2}, \\ & s = -\frac{2}{5} y_2 j + \frac{3}{10} y_1, \quad s = \frac{25 y_1 y_2 j - 12 y_1^2 + 12 y_2^2}{10\,(3y_2 j - 4y_1)} \Bigg]\end{aligned} \tag{3.46}$$

Although simpler equations are obtained in this case, they involve complex numbers and, thus, may be more time-consuming to solve.

The third and final substitution method described in Section 2.2.1 employs the following simple definitions:

$$\sin(\theta) = \mathrm{s}_\theta, \quad \cos(\theta) = \mathrm{c}_\theta \tag{3.47}$$

$$\sin(\beta) = \mathrm{s}_\beta, \quad \cos(\beta) = \mathrm{c}_\beta \tag{3.48}$$

Since these substitutions do not introduce rational functions, we need not multiply the resulting equations by polynomials to clear denominators. As such, the system we obtain in this case is only of degree 2:

$$\ell_1 \mathrm{c}_\theta + \ell_2 \mathrm{s}_\beta - s = 0 \tag{3.49}$$

$$\ell_1 \mathrm{s}_\theta - \ell_2 \mathrm{c}_\beta = 0 \tag{3.50}$$

$$\mathrm{s}_\theta^2 + \mathrm{c}_\theta^2 - 1 = 0 \tag{3.51}$$

$$\mathrm{s}_\beta^2 + \mathrm{c}_\beta^2 - 1 = 0 \tag{3.52}$$

though there are now five indeterminates. Nevertheless, the resulting Gröbner basis is far simpler than the others:

$$\left[\mathrm{s}_\beta^2 + \mathrm{c}_\beta^2 - 1, \quad 3\mathrm{s}_\theta - 4\mathrm{c}_\beta, \quad 9\mathrm{c}_\theta^2 - 16\mathrm{s}_\beta^2 + 7, \quad 10s - 3\mathrm{c}_\theta - 4\mathrm{s}_\beta \right] \tag{3.53}$$

where we have used a pure lexicographic ordering with $s \succ \mathrm{c}_\theta \succ \mathrm{s}_\theta \succ \mathrm{c}_\beta \succ \mathrm{s}_\beta$. The recursively solvable system Δ_3 obtained in this case is clearly more efficient than those obtained above, and agrees with the hand-derived geometric solution $\measuredangle_3$:

$$\Delta_3 = \left[\mathrm{s}_\theta = \frac{4}{3}\mathrm{c}_\beta, \quad \mathrm{c}_\theta = \pm\frac{1}{3}\sqrt{16\mathrm{s}_\beta^2 - 7}, \qquad s = \frac{3}{10}\mathrm{c}_\theta + \frac{2}{5}\mathrm{s}_\beta \right] \tag{3.54}$$

$$\measuredangle_3 = \left[\mathrm{s}_\theta = \frac{\ell_2}{\ell_1}\mathrm{c}_\beta, \quad \mathrm{c}_\theta = \pm\frac{1}{\ell_1}\sqrt{\ell_2^2\mathrm{s}_\beta^2 + \ell_1^2 - \ell_2^2}, \quad s = \ell_1\mathrm{c}_\theta + \ell_2\mathrm{s}_\beta \right] \tag{3.55}$$

The results of this section are summarized in Table 3.1, and suggest that increasing the number of indeterminates may be preferable to increasing the degree of the input polynomials. A rough complexity bound for the computation of a Gröbner basis for a zero-dimensional system—that is, a system with a finite number of solutions—is d^n for a system of maximal total degree d in n indeterminates [51]. Note that the bases being sought in this work are positive-dimensional, for which the complexity bound is even less optimistic [80]. Fortunately, these rather bleak upper bounds are not generally experienced in practice; however, they do indicate the importance of restraining the number of indeterminates as well as the total degree of the system. As such, only the third substitution method is used in the remainder of this work.

Table 3.1: Comparison of triangular systems for planar slider-crank mechanism

Substitution method	Complexity of original system		Cost of Gröbner basis	
	Maximum degree	Indeterminates	Multiplications	Additions
Tangent-half-angle	5	3	59	23
Euler	4	4	33	10
Sine-and-cosine	2	5	11	7

3.2.2 Floating-point coefficients

Since the ultimate objective is to develop an approach that can be applied to realistic engineering systems, the appearance of floating-point quantities is inevitable. Indeed, all physical measurements are intrinsically approximate, and even some physical constants—most notably, the gravitational constant [45]—are not known precisely. As such, many practical engineering applications are content with three significant figures [56]; thus, it does not appear to be necessary to compute Gröbner bases exactly in this context. The numerical perturbation of 0.000001 in (2.85), for example, would represent a measurement error of 1 [µm], which can be safely ignored in all applications considered in this work. Perhaps the most obvious way of proceeding is to use an algorithm that is capable of computing Gröbner bases with inexact coefficients, such as those described in Section 2.2.6. Such an algorithm appears to be a logical choice, as the coefficients appearing in the constraint equations are generally computed from approximate physical measurements. Furthermore, unlike integers or rational numbers, all floating-point numbers occupy the same amount of space, which alleviates the practical memory issues associated with coefficient growth during the elimination procedure. The computation of approximate Gröbner bases is still an active area of research, largely due to the stability issues inherent in floating-point arithmetic, which adds a level of uncertainty to the computed results. As such, several other approaches have been proposed.

An interesting strategy, called the Gröbner trace [128], involves performing an exact Gröbner basis computation over the finite field $\mathbb{Z}_p$—that is, using modular arithmetic. Since all coefficients remain smaller than the prime modulus p, the Gröbner basis computation is unhampered by the storage of large integers. By recording the sequence of S-polynomials that leads to the final basis, unnecessary reductions can be avoided when the computation is ultimately repeated using floating-point arithmetic with the original coefficients. The trace computation can be repeated with prime moduli of increasing size until the basis appears to have stabilized, an approach that is similar to that used by Shirayanagi [116] in the computation of floating-point Gröbner bases. A related idea involves replacing floating-point coefficients with small random integers [117], again with the intention of avoiding unnecessary reductions when computing the actual basis. Although intriguing, Gröbner trace algorithms are probabilistic in nature, and rely on the fortuitous

selection of random numbers. Another alternative is to replace each floating-point coefficient with a new variable. These variables can then be treated either as indeterminates, appending them to the end of the term ordering, or as part of the coefficients. In the latter case, the coefficients become rational functions in terms of these new variables, which results in a slower Gröbner basis computation but generally provides a simpler basis, since it is devoid of polynomials relating only the symbolic coefficients [76].

The typical approach adopted by the kinematics community is to convert all coefficients into rational numbers [26, 87], using either exact rational representations or rational approximations. Since the kinematic constraint equations for mechanical systems involve geometric parameters, such as lengths ℓ_1 and ℓ_2 in (3.33), the use of rational approximations can be interpreted as using exact measurements from a physical system of very similar dimensions. Provided a sufficiently precise approximation is used, the resulting Gröbner basis will be valid for systems manufactured within a particular tolerance. The `convert/rational` function in Maple allows the user to control the precision of a rational approximation, which is employed in this work to determine sufficiently precise approximations that use as few digits as possible.

3.3 Triangularizing systems

In this section, we examine the generation of triangular systems using Gröbner bases, assuming that the kinematic constraint equations have been converted into a system of polynomials with rational coefficients. We first discuss whether a given system can be triangularized, which is dependent on the topology of the mechanism, the selection of generalized coordinates, and the complexity of the kinematic equations. The computation of a Gröbner basis has been discussed at length in Section 2.2.5; however, the resulting triangular system has been given little attention. As such, the second half of this section focuses on the unique characteristics intrinsic to triangular systems generated from systems of kinematic equations.

3.3.1 Triangularizability

As discussed in Section 2.1.5, the embedding technique can be used to eliminate the dependent accelerations from the symbolic dynamic equations, but the resulting second-order ODEs (2.28) are functions of the independent and dependent positions ($\mathbf{q}_i$ and $\mathbf{q}_d$) as well as the independent and dependent velocities ($\dot{\mathbf{q}}_i$ and $\dot{\mathbf{q}}_d$). The dependent velocities $\dot{\mathbf{q}}_d$ can be calculated from $\dot{\mathbf{q}}_i$ at each time step using the linear velocity transformation equations (2.26); however, the dependent positions $\mathbf{q}_d$ are related to $\mathbf{q}_i$ by the nonlinear kinematic constraints, and are typically determined by iterating over these equations using Newton's method [37]. Although Newton–Raphson iteration converges quadratically, it is not guaranteed to converge to the solution that is closest to the initial guess. Consequently, the computed trajectory may jump between adjacent solution branches, which can affect the reliability of the simulation [94]. In many cases, the kinematic constraint equations can be either fully or at least partially triangularized using Gröbner bases, thereby reducing the use of iteration. Where full triangularization is possible, the constraints are always satisfied to machine precision, not simply to within a specified iteration tolerance—provided, of course, the resulting equations are of degree no greater than four [140], which is generally the case in this work. Furthermore, such fully triangular systems can be solved in a fixed amount of time, which is particularly advantageous in real-time applications. Where full triangularization is not possible, a block-triangular form can be obtained that still results in more efficient simulations than existing iterative and constraint stabilization techniques. Thus, the objective at this stage of the procedure is to obtain a maximally triangular system of kinematic equations that can be used to determine $\mathbf{q}_d$ given $\mathbf{q}_i$.

Whether a particular mechanism yields a fully triangular solution depends on a number of factors, some of which may be beyond the control of the analyst. In the planar slider-crank example, a pure lexicographic term ordering with $s \succ \mathrm{c}_\theta \succ \mathrm{s}_\theta \succ \mathrm{c}_\beta \succ \mathrm{s}_\beta$ resulted in system Δ_3 (3.54), which computes $\mathbf{q}_d = \{\theta, s\}$ given $\mathbf{q}_i = \{\beta\}$. If the ordering $s \succ \mathrm{s}_\beta \succ \mathrm{c}_\beta \succ \mathrm{s}_\theta \succ \mathrm{c}_\theta$ is used instead, we obtain the following triangular system:

$$\Delta_4 = \left[\mathrm{c}_\beta = \frac{3}{4}\mathrm{s}_\theta, \quad \mathrm{s}_\beta = \pm\frac{1}{4}\sqrt{9\mathrm{c}_\theta^2 + 7}, \qquad s = \frac{3}{10}\mathrm{c}_\theta + \frac{2}{5}\mathrm{s}_\beta \right] \tag{3.56}$$

$$\text{✍}_4 = \left[\mathrm{c}_\beta = \frac{\ell_1}{\ell_2}\mathrm{s}_\theta, \quad \mathrm{s}_\beta = \pm\frac{1}{\ell_2}\sqrt{\ell_1^2\mathrm{c}_\theta^2 - \ell_1^2 + \ell_2^2}, \quad s = \ell_1\mathrm{c}_\theta + \ell_2\mathrm{s}_\beta \right] \tag{3.57}$$

where $\mathbf{q}'_d = \{\beta, s\}$ is computed given $\mathbf{q}'_i = \{\theta\}$. In this 1-DOF single-loop case, the choice of whether to drive the system with β or θ does not affect its triangularizability. Now consider the 1-DOF planar multi-loop mechanism shown in Figure 3.4 [57]. This system consists of

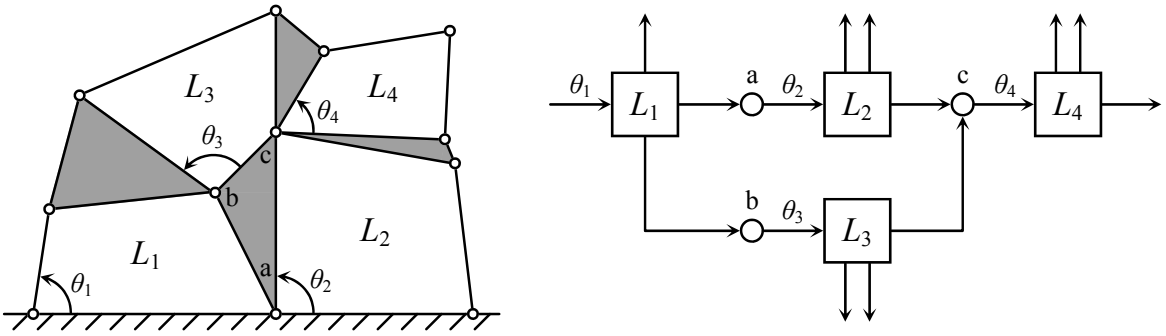


Figure 3.4: Planar multi-loop mechanism and corresponding solution flow

four independent loops, each of which is a planar four-bar mechanism that can be solved by specifying any one of its four interior angles. As shown in the adjacent solution flow block diagram, specifying θ_1 allows us to solve loop L_1 immediately. Angles θ_2 and θ_3 can then be determined and used to solve loops L_2 and L_3. Only once these two loops have been solved can θ_4 be computed, which drives the solution for the last loop. If θ_4 were specified instead of θ_1, loop L_4 could be solved immediately, but iteration would be required to determine the other two angles at joint c. Thus, we can conclude that *triangularizability is dependent on the system topology as well as the solution flow, which is directly related to the choice of kinematic inputs or, equivalently, independent coordinates.* In a kinematic simulation, the inputs to a system may be determined by factors other than simulation efficiency, such as the location of actuators throughout a mechatronic system, in which case an iterative solution may be unavoidable. In the context of forward dynamic simulation, however, no limitation of this nature exists, so the most convenient set of independent coordinates can be used.

Even if the choice of independent coordinates is unconstrained, only f such coordinates can be selected, one for each degree-of-freedom. A procedure for determining the kinematic solution flow for a fully triangularizable f-DOF mechanism can be summarized as follows:

1. Identify a loop L_1 with f_1 local DOF, where $f_1 \leq f$.

2. Assign f_1 generalized coordinates from L_1 to independent coordinate vector $\mathbf{q}_i$.

3. Since it is known that the local solution for L_1 can be obtained given $\mathbf{q}_i$, all coordinates $\mathbf{q}_1$ in loop L_1 are now treated as known.

4. Identify a loop L_2 with f_2 local DOF, where $|\{\mathbf{q}_1\} \cap \{\mathbf{q}_2\}| \geq f_1 + f_2 - f$ or, stated more simply, no more than $f - f_1$ coordinates must be appended to $\mathbf{q}_i$ in order to solve L_2.

5. Assign the minimum number of coordinates from L_2 to $\mathbf{q}_i$ such that L_2 can be solved given $\mathbf{q}_i$ and $\mathbf{q}_1$.

6. Proceed in this fashion until all independent loops have been solved.

Thus, in order to obtain a solution flow that is devoid of iteration, there must exist at least one loop with as many knowns as DOF at each step in the solution. Note that there is still some freedom in the choice of independent coordinates as well as the sequence in which independent loops are solved. In general, it is advisable to choose a solution flow that involves forming local solutions given n topologically adjacent coordinates, since the corresponding Gröbner basis may be simpler and, therefore, take less time to compute. One simple strategy used in this work is to begin at the ground node and traverse the topological graph, placing the first f nodes in the independent coordinate vector. This strategy is employed in the example of Section 4.2.4. Note that the order in which nodes are encountered while traversing the topological graph can also be used to determine the order in which terms are to be eliminated when generating a Gröbner basis. As demonstrated above, a pure lexicographic term ordering with $\mathbf{q}_d \succ \mathbf{q}_i$ results in a triangular system in which the dependent coordinates can be solved given values of the independent coordinates. Thus, a suitable elimination order is the reverse of the order in which the graph is traversed to determine the independent coordinates. Unfortunately, since even a small change in the original system of polynomials can have a dramatic influence on the Gröbner basis computation time, determining the optimal elimination order is difficult. We also note that Buchberger's algorithm is nondeterministic—hence the development of various S-polynomial selection strategies—and, as such, the Gröbner basis computation time may vary even if the original system is unaltered. Finally, experience indicates that the decision of whether to order $\mathrm{s}_\vartheta \succ \mathrm{c}_\vartheta$ or $\mathrm{c}_\vartheta \succ \mathrm{s}_\vartheta$ is inconsequential in most cases.

Whereas some multi-loop mechanisms can be triangularized if their independent loops are solved in the correct order, some mechanisms are inherently untriangularizable. Block-triangular solutions can be found using a strategy similar to that outlined above, with modifications to ensure that the required iteration is performed over the minimum number of coordinates possible. Although such an algorithm could be designed, the topological graph is not generally complicated enough to warrant such an endeavour. In fact, particularly for complex mechanisms, even an exhaustive search of the topological graph would require considerably less computation time than would generating the corresponding Gröbner bases. Furthermore, a system that is only block-triangularizable when modelled using one set of generalized coordinates may be fully triangularizable using another—that is, *triangularizability is dependent on the choice of modelling coordinates themselves.* An effective strategy for parallel multi-loop mechanisms, or mechanisms whose independent loops share more than one joint, is to use absolute coordinates to describe the position and orientation of the end-effector. For mechanisms with full mobility, whose end-effectors can be positioned and oriented independently in all coordinate directions [44], this strategy can lead to a fully triangular solution, as will be demonstrated in Section 4.2. In the event that full triangularization is still not possible, the use of absolute coordinates on one body can result in an effective block-triangular form, as will be demonstrated by the examples of Chapter 5. Although using absolute coordinates results in more independent loops than if a purely joint coordinate formulation is used, the equations corresponding to each loop are simpler and, therefore, more suitable for generating Gröbner bases.

The use of absolute coordinates highlights another important strategy when triangularizing multi-loop mechanisms. Since the complexity of computing a Gröbner basis grows roughly exponentially with the number of indeterminates, triangularizing the equations associated with each loop separately can decrease the Gröbner basis generation time considerably. Kinematic equations are often decoupled in this manner to facilitate their solution [26, 65]. For complex mechanisms, use of a decoupling strategy is essential to ensure that the duration of the Gröbner basis computation is reasonable, and that it can be performed using the available computational resources. Thus, from a practical perspective, we can state that *triangularizability is dependent on the complexity of the equations as well as the time and equipment available for performing the computation.* As will be

discussed in Chapter 5, the Gröbner bases generated in this work approach the limits of the computational resources available for this research. Although generating a Gröbner basis can be computationally expensive, the resulting system of equations is both efficient and eternally valid, requiring generation only once for a given set of system parameters.

3.3.2 Triangular systems in kinematics

Once a solution flow has been determined and all Gröbner bases have been generated, the only remaining task at the position level is to solve the resulting polynomials for the dependent coordinates in the correct order. In fact, this task is quite simple, since the order in which the indeterminates must be solved is determined by the Gröbner basis elimination order. In some cases, there may be more than one basis polynomial that could be used to determine a particular indeterminate, as in Δ_1 (3.39) and Δ_2 (3.46) above. Except possibly in some rare situations, discussed below, the most computationally efficient equation can be selected. We also note that it is generally advisable to avoid equations of the form $\mathrm{s}_\vartheta^2 + \mathrm{c}_\vartheta^2 - 1 = 0$ to ensure that the corresponding angle ϑ is always resolved into the correct quadrant. Two characteristics of triangular systems derived from kinematic equations deserve further discussion: mechanism configurations and the *specialization* of Gröbner bases.

We shall begin by returning to the planar slider-crank example. Using a pure lexicographic term ordering with $s \succ \mathrm{s}_\beta \succ \mathrm{c}_\beta \succ \mathrm{s}_\theta \succ \mathrm{c}_\theta$, we obtain triangular system Δ_4 (3.56). Given a numerical value for θ, we can compute the corresponding values for β and s as precisely as we wish, and can do so in a fixed amount of time. Note that each value of θ corresponds to two values of s_β, which represent the two possible mechanism configurations for the same crank angle, as shown for $\theta = \pi/4$ [rad] in Figure 3.5. Recall that one of the attractive properties of a Gröbner basis is that its solutions are the same as those of the original system. Thus, all mechanism configurations are retained in the triangular solution and, as is typically the case, the analyst must select the desired configuration. We now generate a Gröbner basis for the planar slider-crank mechanism using a pure lexicographic

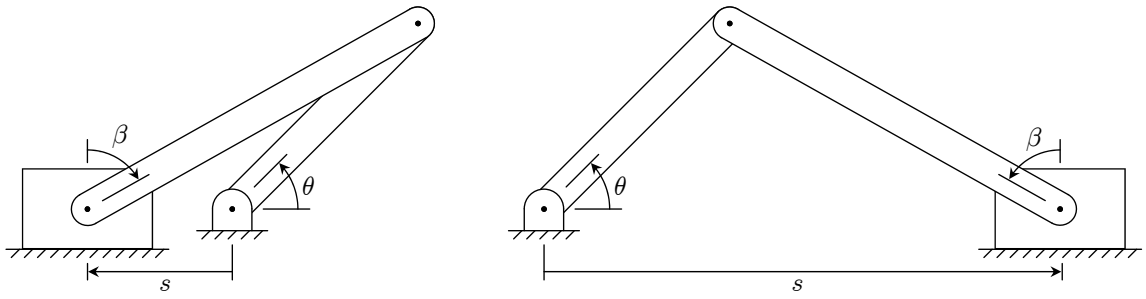


Figure 3.5: Two configurations of planar slider-crank mechanism for $\theta = \pi/4$ [rad]

term ordering with $\mathrm{s}_\beta \succ \mathrm{c}_\beta \succ \mathrm{s}_\theta \succ \mathrm{c}_\theta \succ s$, which results in the following triangular system:

$$\Delta_5 = \left[\mathrm{c}_\theta = \frac{100s^2 - 7}{60s}, \quad \mathrm{s}_\theta = \pm\sqrt{-\mathrm{c}_\theta^2 + 1}, \quad \mathrm{c}_\beta = \frac{3}{4}\mathrm{s}_\theta, \quad \mathrm{s}_\beta = -\frac{3}{4}\mathrm{c}_\theta + \frac{5}{2}s \right] \tag{3.58}$$

$$\text{✍}_5 = \left[\mathrm{c}_\theta = \frac{s^2 + \ell_1^2 - \ell_2^2}{2\ell_1 s}, \quad \mathrm{s}_\theta = \pm\sqrt{-\mathrm{c}_\theta^2 + 1}, \quad \mathrm{c}_\beta = \frac{\ell_1}{\ell_2}\mathrm{s}_\theta, \quad \mathrm{s}_\beta = \frac{-\ell_1\mathrm{c}_\theta + s}{\ell_2} \right] \tag{3.59}$$

In this case, each value of s corresponds to two values of s_θ, which correspond to the *elbow-up* and *elbow-down* configurations shown in Figure 3.6. System Δ_5 raises an important issue: solving a Gröbner basis for the dependent coordinates can introduce denominators that vanish during a simulation. The underlying issue is related to the behaviour of a Gröbner basis under *specialization*, or upon substitution of numerical values for the variables [23]. Another example is shown in the hand-derived solution ✍$_5$ (3.59), which can also be obtained by treating ℓ_1 and ℓ_2 as symbolic parameters and appending them to the term ordering list. Clearly, $\ell_1 = 0$, $\ell_2 = 0$, and $s = 0$ all result in vanishing denominators, and all are non-physical. In this case, the geometry of the mechanism restricts s to the ranges $-\ell_1 \pm \ell_2 \leq s \leq \ell_1 \pm \ell_2$, so the denominator of the expression for c_θ never becomes

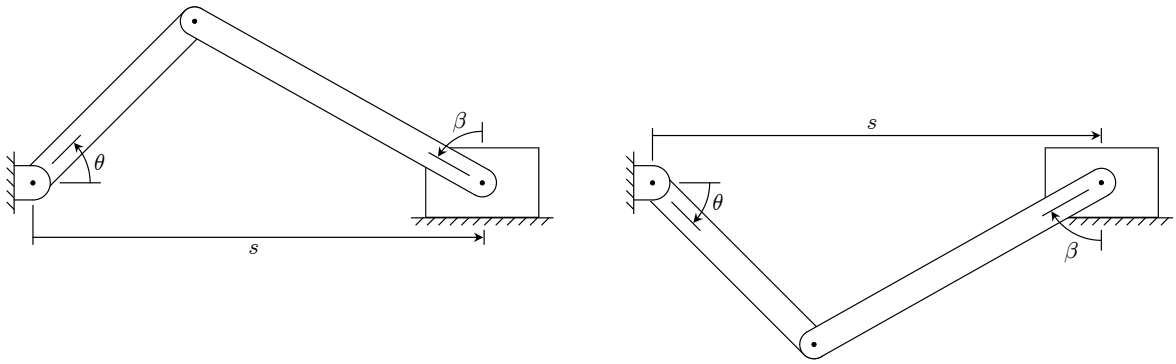


Figure 3.6: Two configurations of planar slider-crank mechanism for $s = 0.551$ [m]

zero. In theory, we could avoid such situations by computing a comprehensive Gröbner basis [85, 138], which retains the properties of a Gröbner basis under all specializations; however, experience has shown that vanishing denominators are almost never encountered and, when they are, they can be eliminated using symbolic simplifications. The same observation holds for the values of trigonometric variables: provided the solutions remain physically valid, the values of s_ϑ and c_ϑ remain in the range $[-1, 1]$. As a final note, although polynomials of high degree can be obtained for arbitrary systems, the triangular systems generated in this work typically result in no more than two solutions for each variable. The only exception that has been discovered is discussed in Section 4.2.3, where a Gröbner basis must be generated for two independent loops simultaneously to avoid an iterative kinematic solution.

3.4 Generating dynamic simulation code

If we were only interested in performing kinematic simulations, we could simply apply motion drivers to the triangular systems generated in the previous section, and evaluate the equations numerically at each time step of a desired interval. In order to perform dynamic simulations, we use the embedding technique to eliminate the Lagrange multipliers from the dynamic equations and obtain one ordinary differential equation for each independent acceleration. Once written in first-order form, the resulting equations can be numerically integrated from one time step to the next, thereby providing the independent positions that drive the kinematic solution. As will be shown, this strategy results in very efficient dynamic simulations that are suitable for real-time applications.

3.4.1 Projecting dynamic equations

As described in Section 2.1.5, the embedding technique produces a set of f second-order ODEs for an f-DOF mechanism. In Section 3.1.2, we used a joint coordinate formulation to derive the equations of motion for the planar slider-crank mechanism and obtained three ODEs that describe its dynamic behaviour, which will now be reduced to a single second-order ODE. Note that the choice of independent coordinate $\mathbf{q}_i$ is arbitrary in this case, since the kinematics of this mechanism can be solved recursively given β (3.55), θ (3.57),

or s (3.59). Selecting $\mathbf{q}_i = \{\theta\}$ and $\mathbf{q}_d = \{\beta, s\}$, we first rearrange (3.32) to agree with the form shown in Section 2.1.5:

$$\begin{bmatrix} I_2 + \frac{1}{4}m_2\ell_2^2 & -\frac{1}{2}m_2\ell_2\mathrm{c}_\beta & 0 \\ -\frac{1}{2}m_2\ell_2\mathrm{c}_\beta & m_2 + m_3 & 0 \\ 0 & 0 & I_1 + \frac{1}{4}m_1\ell_1^2 \end{bmatrix} \begin{Bmatrix} \ddot{\beta} \\ \ddot{s} \\ \ddot{\theta} \end{Bmatrix}$$

$$+ \begin{bmatrix} \ell_2\mathrm{c}_\beta & \ell_2\mathrm{s}_\beta \\ -1 & 0 \\ -\ell_1\mathrm{s}_\theta & \ell_1\mathrm{c}_\theta \end{bmatrix} \begin{Bmatrix} \lambda_1 \\ \lambda_2 \end{Bmatrix} = \begin{Bmatrix} \frac{1}{2}m_2\ell_2 g\mathrm{s}_\beta \\ -\frac{1}{2}m_2\ell_2\mathrm{s}_\beta\dot{\beta}^2 + F(t) \\ -\frac{1}{2}m_1\ell_1 g\mathrm{c}_\theta \end{Bmatrix} \tag{3.60}$$

$$\mathbf{\Phi} = \begin{Bmatrix} \ell_1\mathrm{c}_\theta + \ell_2\mathrm{s}_\beta - s \\ \ell_1\mathrm{s}_\theta - \ell_2\mathrm{c}_\beta \end{Bmatrix} = \mathbf{0} \tag{3.61}$$

where $\lambda_1 = R_X$ and $\lambda_2 = R_Y$. Transformation matrix $\mathbf{B}$ is then computed as follows:

$$\mathbf{B} = \begin{bmatrix} -\mathbf{\Phi}_d^{-1}\,\mathbf{\Phi}_i \\ 1 \end{bmatrix} = \begin{bmatrix} -\frac{\ell_1\mathrm{c}_\theta}{\ell_2\mathrm{s}_\beta} \\ -\ell_1\mathrm{s}_\theta - \frac{\ell_1\mathrm{c}_\theta\mathrm{c}_\beta}{\mathrm{s}_\beta} \\ 1 \end{bmatrix} \tag{3.62}$$

which is an orthogonal complement of the Jacobian—that is, $\mathbf{B}^{\mathrm{T}}\,\mathbf{\Phi}_{\mathbf{q}}^{\mathrm{T}} = \mathbf{0}$. Thus, pre-multiplying (3.60) by $\mathbf{B}^{\mathrm{T}}$ eliminates the Lagrange multipliers $\boldsymbol{\lambda}$ and results in the form shown in (2.25). Upon substitution of the acceleration-level transformation equation (2.27) and simplifying, we obtain one second-order ODE for the independent acceleration:

$$\begin{aligned} \ddot{\theta} = \Big[& \ell_1^2\big(4m_2\ell_2^2\mathrm{s}_\theta\mathrm{c}_\theta\mathrm{s}_\beta\mathrm{c}_\beta^2 + 8m_3\ell_2^2\mathrm{s}_\theta\mathrm{c}_\theta\mathrm{s}_\beta\mathrm{c}_\beta^2 + 4m_2\ell_2^2\mathrm{c}_\theta^2\mathrm{c}_\beta^3 + 8m_3\ell_2^2\mathrm{c}_\theta^2\mathrm{c}_\beta^3 - 3m_2\ell_2^2\mathrm{s}_\theta\mathrm{c}_\theta\mathrm{s}_\beta \\ & - 4m_3\ell_2^2\mathrm{s}_\theta\mathrm{c}_\theta\mathrm{s}_\beta + 4I_2\mathrm{s}_\theta\mathrm{c}_\theta\mathrm{s}_\beta - 4m_2\ell_2^2\mathrm{c}_\theta^2\mathrm{c}_\beta - 8m_3\ell_2^2\mathrm{c}_\theta^2\mathrm{c}_\beta - 2m_2\ell_2^2\mathrm{c}_\beta^3 - 4m_3\ell_2^2\mathrm{c}_\beta^3 \\ & + 2m_2\ell_2^2\mathrm{c}_\beta + 4m_3\ell_2^2\mathrm{c}_\beta\big)\dot{\theta}^2 + \ell_1^2\mathrm{c}_\theta\big(-2m_2\ell_2^2\mathrm{s}_\theta\mathrm{s}_\beta\mathrm{c}_\beta^2 - 2m_2\ell_2^2\mathrm{c}_\theta\mathrm{c}_\beta^3 + 4m_2\ell_2^2\mathrm{s}_\theta\mathrm{s}_\beta \\ & + 4m_3\ell_2^2\mathrm{s}_\theta\mathrm{s}_\beta + 3m_2\ell_2^2\mathrm{c}_\theta\mathrm{c}_\beta + 4m_3\ell_2^2\mathrm{c}_\theta\mathrm{c}_\beta + 4I_2\mathrm{c}_\theta\mathrm{c}_\beta\big)\dot{\theta}\dot{\beta} + 2m_2\ell_1\ell_2^3\left(\mathrm{c}_\beta^2 - 1\right)\left(\mathrm{s}_\theta\mathrm{c}_\beta^2\right. \\ & \left. - \mathrm{c}_\theta\mathrm{s}_\beta\mathrm{c}_\beta - \mathrm{s}_\theta\right)\dot{\beta}^2 + 2\ell_1\ell_2^2\left(\mathrm{c}_\beta^2 - 1\right)\left(2F\mathrm{s}_\theta\mathrm{s}_\beta + 2F\mathrm{c}_\theta\mathrm{c}_\beta + m_1 g\mathrm{c}_\theta\mathrm{s}_\beta + m_2 g\mathrm{c}_\theta\mathrm{s}_\beta\right)\Big] \Big/ \\ & \Big[4\ell_1^2\ell_2^2\mathrm{s}_\theta\mathrm{c}_\theta\big(m_2 + 2m_3\big)\mathrm{s}_\beta^2\mathrm{c}_\beta + \ell_2^2\big(4m_2\ell_1^2\mathrm{c}_\theta^2 + 8m_3\ell_1^2\mathrm{c}_\theta^2 - m_1\ell_1^2 - 4m_2\ell_1^2 - 4m_3\ell_1^2 \\ & - 4I_1\big)\mathrm{s}_\beta\mathrm{c}_\beta^2 + \big(-3m_2\ell_1^2\ell_2^2\mathrm{c}_\theta^2 - 4m_3\ell_1^2\ell_2^2\mathrm{c}_\theta^2 + 4I_2\ell_1^2\mathrm{c}_\theta^2 + m_1\ell_1^2\ell_2^2 + 4m_2\ell_1^2\ell_2^2 \\ & + 4m_3\ell_1^2\ell_2^2 + 4I_1\ell_2^2\big)\mathrm{s}_\beta\Big] \end{aligned} \tag{3.63}$$

where the dependent positions $\mathbf{q}_d$ can be calculated from $\mathbf{q}_i$ at each time step using triangular system $\measuredangle_4$ (3.57), and the dependent velocities $\dot{\mathbf{q}}_d$ can be calculated from $\dot{\mathbf{q}}_i$ using

the velocity transformation (2.26):

$$\dot{\beta} = -\frac{\ell_1 \mathrm{c}_\theta}{\ell_2 \mathrm{s}_\beta}\,\dot{\theta} \tag{3.64}$$

$$\dot{s} = \left(-\ell_1 \mathrm{s}_\theta - \frac{\ell_1 \mathrm{c}_\theta \mathrm{c}_\beta}{\mathrm{s}_\beta}\right)\dot{\theta} \tag{3.65}$$

Note that the numerical singularity at $\mathrm{s}_\beta = 0$ is never encountered when $\ell_1 < \ell_2$, as in this example. The simulation process has been summarized in Figure 3.7. Note that only $\dot{\mathbf{q}}_i$ and $\ddot{\mathbf{q}}_i$ are numerically integrated. In real-time applications, the maximum integration step size is bounded by the sampling rate, which is generally small: automotive electronic control units, for example, typically operate with a 10-millisecond cycle time [29]. Consequently, the relatively large numerical effort associated with high-order integration methods cannot be compensated by taking large steps in time. Low-order, fixed-step-size, non-stiff ODE solvers, such as the explicit Euler scheme, are often used in real-time and hardware-in-the-loop applications [3].

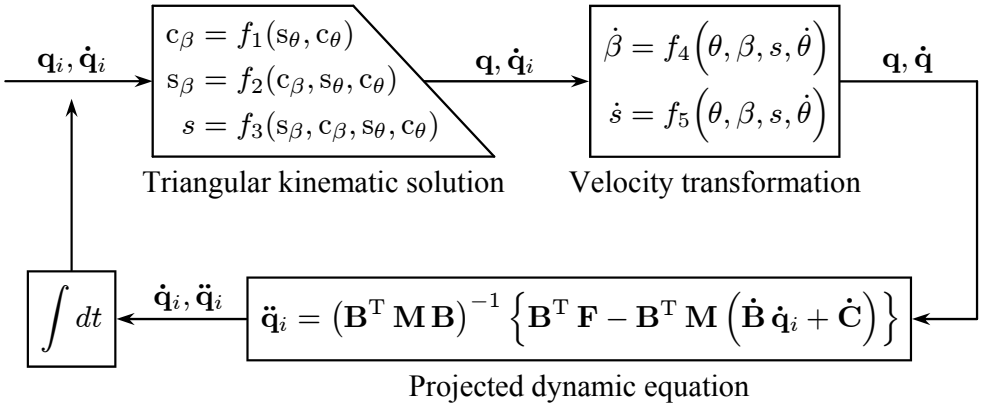


Figure 3.7: Solution flow for dynamic simulation of planar slider-crank mechanism

3.4.2 Simplification and optimization

We conclude this chapter with a brief discussion of computational efficiency. In the previous section, we found that the projected dynamic equation (3.63) was considerably more complex than the original system of ODEs (3.60). In fact, a numerical formulation would typically involve far more arithmetic operations, and would require reformulating the equations of motion at every time step of a simulation. Although numerical formulations are

used by many popular commercial simulation packages, such as MSC.ADAMS, the relatively slow process of repeatedly reformulating the system equations may prohibit their use in real-time applications. Symbolic formulation techniques, on the other hand, produce sets of equations that are eternally valid, so must only be generated once. Prior to simulation, the symbolic equations can also be greatly simplified in numerous ways, such as eliminating multiplications by 0 or 1, simplifying trigonometric expressions, and identifying and removing repeated calculations of sub-expressions [142]. Such simplifications can result in models that simulate between five and ten times faster than those modelled using a purely numerical approach [103]. As shown in Table 3.2, the code optimization facilities available in Maple can further alleviate the computational burden associated with lengthy symbolic expressions. Note that the `dsolve/numeric/optimize` routine generally outperforms `codegen/optimize`, though the latter is currently more versatile than the former. Also note that the computation sequences generated by `dsolve/numeric/optimize` recycle temporary variables when they are no longer needed, which can reduce the memory requirements of the resulting simulation code considerably.

Table 3.2: Cost of evaluating dynamic equation (3.63) for planar slider-crank mechanism

Metric	Original expression	`codegen/optimize`	`dsolve/optimize`
Multiplications	383	195	91
Additions	46	46	37
Functions	97	4	4
Temporary variables	0	47	18

3.5 Chapter summary

The approach described in this chapter employs existing symbolic computing and graph-theoretic concepts to generate efficient kinematic and dynamic simulation code for multibody systems containing closed kinematic chains. The formulation procedure can be summarized as follows:

1. Develop a model of the physical system of interest.

2. Determine the optimal kinematic solution flow and the corresponding set of modelling coordinates.

3. Formulate the index-3 differential-algebraic equations of motion.

4. Triangularize the kinematic constraint equations.

5. Project the dynamic equations to obtain one ordinary differential equation for each independent acceleration.

6. Export the optimized dynamic simulation code to the target simulation language.

Although the governing dynamic equations for the example presented above were generated using a graph-theoretic formulation, any symbolic formulation procedure can be used. Furthermore, despite being convenient at present, the use of exact Gröbner bases for triangularizing the kinematic constraint equations represents only one approach of many. Nevertheless, as will be demonstrated by the examples presented in the next three chapters, the approach presented herein can be used to generate real-time-capable dynamic simulation code for relatively complex multibody systems.

Chapter 4

Fully Triangular Systems

In this chapter, we use the Gröbner basis approach described in Chapter 3 to generate computationally efficient simulation code for several mechanisms, all of whose kinematic equations can be fully triangularized. When full triangularization is possible and the resulting equations are of degree no greater than four, a recursively solvable system is obtained that can be evaluated exactly and in a fixed amount of time. Such a scenario is ideal for real-time applications that demand efficient yet precise simulation code. Two single-loop mechanisms—one planar and one spatial—are studied in Section 4.1, where we also generate recursively solvable systems using the characteristic pair of joints approach. In Section 4.2, we focus on multi-loop mechanisms, which can be described as the union of two or more single-loop mechanisms. Among the mechanisms studied therein are a Stephenson-III six-bar, a planar parallel robot, and a Gough–Stewart platform, all of which have practical applications. Further discussion of block-triangularizable systems is deferred until Chapter 5.

4.1 Single-loop mechanisms

The most readily triangularized systems of kinematic equations originate from constrained mechanisms consisting of only one kinematic loop. This section contains the analysis of two such mechanisms: a planar slider-crank and a spatial four-bar. We also compare the triangular solutions generated using the Gröbner basis approach to those generated using the characteristic pair of joints approach described in Section 2.1.5. As will be shown, the

ability to vary the elimination order when generating Gröbner bases can result in more efficient systems of equations than those obtained using characteristic pairs of joints.

4.1.1 Planar slider-crank mechanism

We begin with a simple example to explain the characteristic pair of joints approach in detail. Consider the 1-DOF planar slider-crank mechanism studied throughout Chapter 3, the geometry of which is shown in Figure 4.1. Clearly, $\underline{r}_1$ is the vector sum of $\underline{r}_3$ and $\underline{r}_2$.

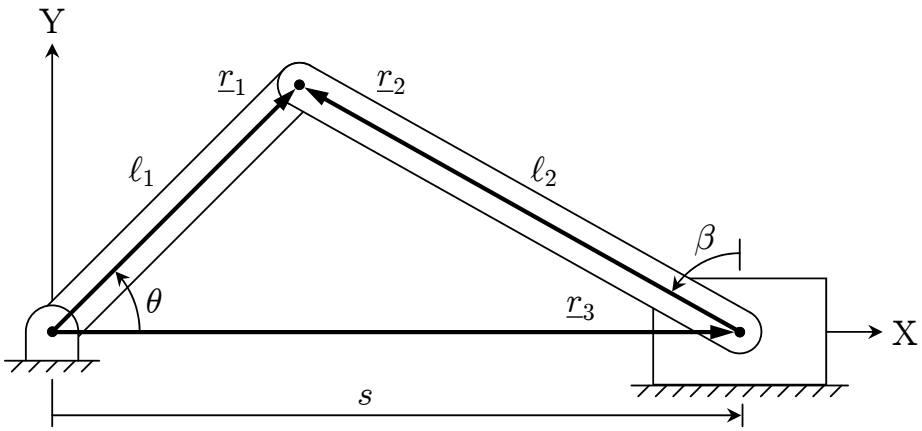


Figure 4.1: Geometry of planar slider-crank mechanism

Since the length of the crank is constant, so is the length of vector $\underline{r}_1$:

$$\|\underline{r}_1\| = \|\underline{r}_3 + \underline{r}_2\| \tag{4.1}$$

$$\|\ell_1 \cos(\theta)\,\hat{\imath} + \ell_1 \sin(\theta)\,\hat{\jmath}\| = \|(s - \ell_2 \sin(\beta))\,\hat{\imath} + \ell_2 \cos(\beta)\,\hat{\jmath}\| \tag{4.2}$$

$$\ell_1^2 = s^2 - 2\ell_2 \sin(\beta) s + \ell_2^2 \tag{4.3}$$

which can be verified geometrically using the law of cosines. Thus, the invariance of the distance between the revolute joints on the crank—the characteristic pair of joints—is used to obtain a solution for s as a function of β:

$$s = \ell_2 \sin(\beta) \pm \sqrt{-\ell_2^2 \cos^2(\beta) + \ell_1^2} \tag{4.4}$$

Alternatively, the vector sum could be expressed as $\underline{r}_2 = -\underline{r}_3 + \underline{r}_1$, whereupon a solution for s would be obtained as a function of θ. In either case, the remaining joint coordinates are determined geometrically [57] which, although trivial for this mechanism, may be time-consuming in general.

More formally, the loop-closure condition can be expressed as the product of six homogeneous transformation matrices [122]:

$$ {}^{0}\mathbf{H}_{1}\,{}^{1}\mathbf{H}_{2}\,{}^{2}\mathbf{H}_{3}\,{}^{3}\mathbf{H}_{4}\,{}^{4}\mathbf{H}_{5}\,{}^{5}\mathbf{H}_{0} = \mathbf{I} \tag{4.5} $$

where ${}^{i}\mathbf{H}_{k}$ represents the transformation from frame $\mathcal{B}_i$ to frame $\mathcal{B}_k$, as shown in Figure 4.2. The loop-closure condition simply states that the start and end frames are coincident upon traversal of the entire kinematic loop. Since we wish to exploit the invariance of the crank length, we write (4.5) in the following equivalent form:

$$ {}^{0}\mathbf{H}_{1}\,{}^{1}\mathbf{H}_{2} = {}^{0}\mathbf{H}_{5}\,{}^{5}\mathbf{H}_{4}\,{}^{4}\mathbf{H}_{3}\,{}^{3}\mathbf{H}_{2} \tag{4.6} $$

$$ \begin{bmatrix} \mathrm{c}_\theta & -\mathrm{s}_\theta & 0 \\ \mathrm{s}_\theta & \mathrm{c}_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & \ell_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & s \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{s}_\beta & \mathrm{c}_\beta & 0 \\ -\mathrm{c}_\beta & \mathrm{s}_\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -\ell_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -\mathrm{s}_\sigma & -\mathrm{c}_\sigma & 0 \\ \mathrm{c}_\sigma & -\mathrm{s}_\sigma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.7} $$

$$ \begin{bmatrix} \mathrm{c}_\theta & -\mathrm{s}_\theta & \ell_1\mathrm{c}_\theta \\ \mathrm{s}_\theta & \mathrm{c}_\theta & \ell_1\mathrm{s}_\theta \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -\mathrm{s}_\beta\mathrm{s}_\sigma + \mathrm{c}_\beta\mathrm{c}_\sigma & -\mathrm{s}_\beta\mathrm{c}_\sigma - \mathrm{c}_\beta\mathrm{s}_\sigma & s - \ell_2\mathrm{s}_\beta \\ \mathrm{s}_\beta\mathrm{c}_\sigma + \mathrm{c}_\beta\mathrm{s}_\sigma & -\mathrm{s}_\beta\mathrm{s}_\sigma + \mathrm{c}_\beta\mathrm{c}_\sigma & \ell_2\mathrm{c}_\beta \\ 0 & 0 & 1 \end{bmatrix} \tag{4.8} $$

$$ = \begin{bmatrix} \mathrm{c}_\theta & -\mathrm{s}_\theta & s - \ell_2\mathrm{s}_\beta \\ \mathrm{s}_\theta & \mathrm{c}_\theta & \ell_2\mathrm{c}_\beta \\ 0 & 0 & 1 \end{bmatrix} \tag{4.9} $$

where $\mathrm{s}_\varphi = \sin(\varphi)$, $\mathrm{c}_\varphi = \cos(\varphi)$, and $\sigma = \theta - \beta$. We now compute the position vector emanating from the origin of frame $\mathcal{B}_0$ and terminating at the origin of frame $\mathcal{B}_2$ using

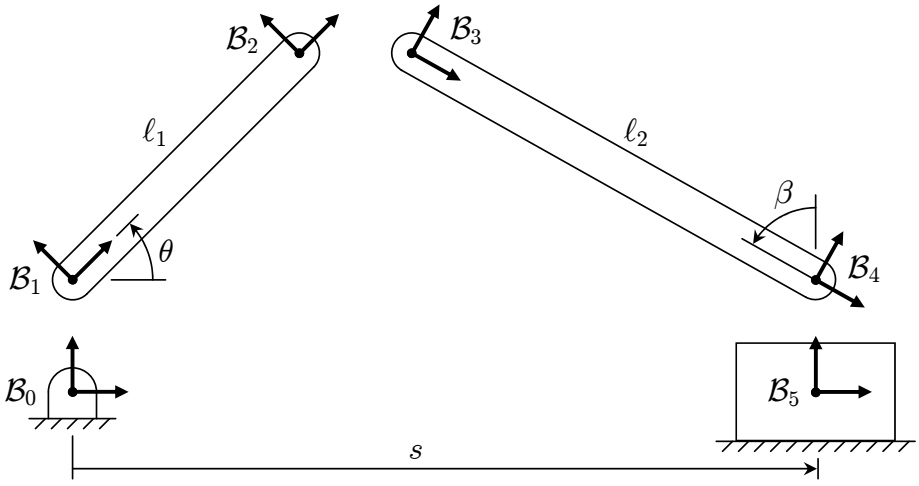


Figure 4.2: Body-fixed reference frames for planar slider-crank mechanism

each of these transformation matrices [66]:

$$\begin{bmatrix} \mathrm{c}_\theta & -\mathrm{s}_\theta & \ell_1\mathrm{c}_\theta \\ \mathrm{s}_\theta & \mathrm{c}_\theta & \ell_1\mathrm{s}_\theta \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} 0 \\ 0 \\ 1 \end{Bmatrix} = \begin{bmatrix} \mathrm{c}_\theta & -\mathrm{s}_\theta & s - \ell_2\mathrm{s}_\beta \\ \mathrm{s}_\theta & \mathrm{c}_\theta & \ell_2\mathrm{c}_\beta \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} 0 \\ 0 \\ 1 \end{Bmatrix} \tag{4.10}$$

$$\begin{Bmatrix} \ell_1\mathrm{c}_\theta \\ \ell_1\mathrm{s}_\theta \\ 1 \end{Bmatrix} = \begin{Bmatrix} s - \ell_2\mathrm{s}_\beta \\ \ell_2\mathrm{c}_\beta \\ 1 \end{Bmatrix} \tag{4.11}$$

Equating the magnitudes of these vectors results in the solution for s as a function of β found previously (4.3). Note that the same solution can be obtained automatically by generating a Gröbner basis using a pure lexicographic term ordering with $\mathrm{s}_\theta \succ \mathrm{c}_\theta \succ s \succ \mathrm{s}_\beta \succ \mathrm{c}_\beta$. Unlike the characteristic pair of joints approach, however, a Gröbner basis can be generated using any desired elimination order, and does not demand the use of joint coordinates. As will be shown in the next example, varying the order in which coordinates are solved can result in triangular systems that are more computationally efficient than those obtained using the characteristic pair of joints technique.

4.1.2 Spatial four-bar mechanism

We now consider the 1-DOF spatial four-bar mechanism shown in Figure 4.3. This system is modelled using 4 joint coordinates: β_1, the angle of the crank as it is driven about the

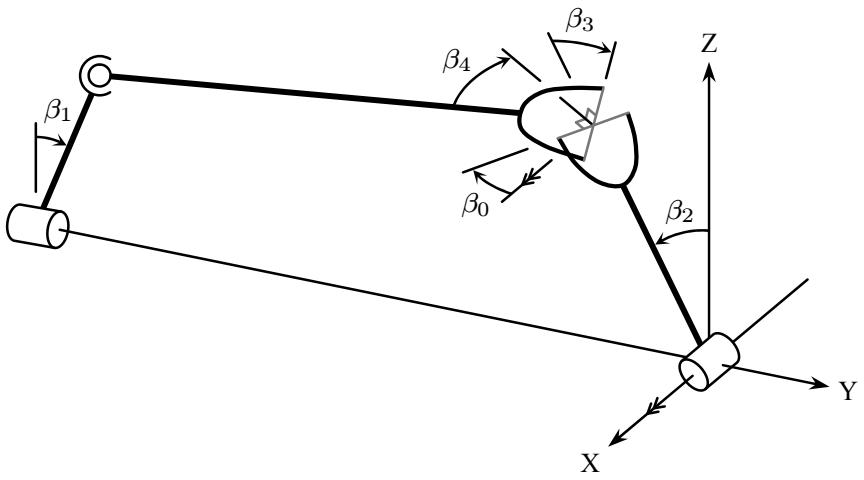


Figure 4.3: Spatial four-bar mechanism

$-$Y-axis; β_2, the angle of the follower as it rotates about the X-axis; and β_3 and β_4, the angles associated with the universal joint between the connecting rod and the follower. Parameter β_0 is used to define the amount of twist in the follower. Note that the rotation axes corresponding to angles β_2 and β_3 are parallel only when $\beta_0 = k\pi$, $k \in \mathbb{Z}$. The $m = n - f = 3$ constraint equations can be expressed as follows:

$$L_c \sin(\beta_1) - L_r\Big(\sin(\beta_0)\cos(\beta_3)\cos(\beta_4) + \cos(\beta_0)\sin(\beta_4)\Big) = 0 \tag{4.12}$$

$$\begin{aligned} &L_c \cos(\beta_1) - L_f \cos(\beta_2) - L_r \sin(\beta_0)\sin(\beta_2)\sin(\beta_4) \\ &\quad + L_r \cos(\beta_4)\Big(\cos(\beta_0)\sin(\beta_2)\cos(\beta_3) - \cos(\beta_2)\sin(\beta_3)\Big) = 0 \end{aligned} \tag{4.13}$$

$$\begin{aligned} &L_g - L_f \sin(\beta_2) + L_r \sin(\beta_0)\cos(\beta_2)\sin(\beta_4) \\ &\quad - L_r \cos(\beta_4)\Big(\cos(\beta_0)\cos(\beta_2)\cos(\beta_3) + \sin(\beta_2)\sin(\beta_3)\Big) = 0 \end{aligned} \tag{4.14}$$

where L_c, L_r, L_f, and L_g are the lengths of the crank, connecting rod, follower, and ground link, respectively. The characteristic pair of joints technique [68] can be used to generate the following triangular, or recursively solvable, system of equations when $\beta_0 = 0$:

$$\begin{aligned} &z_1 = 6\mathrm{c}_1, \quad z_2 = z_1^2 + 400, \quad z_3 = \pm\sqrt{z_2 - \frac{3721}{16}}, \quad \mathrm{c}_2 = \frac{61z_1 + 80z_3}{4z_2}, \\ &\mathrm{s}_2 = \frac{305 - z_1 z_3}{z_2}, \quad z_1 = \pm\sqrt{\frac{9}{4}\mathrm{c}_1^2 - 20\mathrm{s}_2 - z_1\mathrm{c}_2 + 29}, \quad z_2 = \frac{3}{2}\mathrm{c}_1, \\ &\mathrm{c}_3 = \frac{5\mathrm{c}_2 - z_2\mathrm{s}_2}{z_1}, \quad \mathrm{s}_3 = \frac{5\mathrm{s}_2 + z_2\mathrm{c}_2 - 2}{z_1}, \quad \mathrm{c}_4 = \frac{1}{4}z_1, \quad \mathrm{s}_4 = \frac{3}{8}\mathrm{s}_1 \end{aligned} \tag{4.15}$$

where z_k are temporary variables introduced by the `dsolve/numeric/optimize` routine in Maple to avoid repeating calculations; $\mathrm{s}_k = \sin(\beta_k)$, $\mathrm{c}_k = \cos(\beta_k)$, and the system parameters $L_c = 1.5$, $L_r = 4$, $L_f = 2$, and $L_g = 5$ have been substituted. The positive and negative roots for z_3 represent the two possible configurations of the mechanism: if the former is chosen, the mechanism will be in the upright configuration shown in Figure 4.3; if the latter is chosen, the follower will be below the XY-plane.

The first step in the Gröbner basis approach involves converting the constraint equations into polynomials using the substitution technique discussed previously. The system parameters are then substituted and converted into rational numbers, which results in the

following system of polynomials when $\beta_0 = 0$:

$$\tfrac{3}{2}s_1 - 4s_4, \quad \tfrac{3}{2}c_1 - 2c_2 + 4s_2c_3c_4 - 4c_2s_3c_4, \quad 5 - 2s_2 - 4c_2c_3c_4 - 4s_2s_3c_4,$$
$$s_1^2 + c_1^2 - 1, \quad s_2^2 + c_2^2 - 1, \quad s_3^2 + c_3^2 - 1, \quad s_4^2 + c_4^2 - 1 \tag{4.16}$$

A Gröbner basis is then generated. Since the crank is being driven in this example, β_1 is a known function of time and s_1 and c_1 are placed at the end of the term ordering list; the unknown variables can be computed in any order. To illustrate the effect of the solution order on the resulting kinematic equations, two Gröbner bases are generated using different term orderings. We first use a pure lexicographic ordering with $s_4 \succ c_4 \succ s_3 \succ c_3 \succ s_2 \succ c_2 \succ s_1 \succ c_1$ to solve for the unknowns in the same order as in the characteristic pair of joints solution, starting with c_2 and working backwards through the list. The Maple `Groebner:-Basis` command produces the following triangular system:

$$s_1^2 + c_1^2 - 1, \quad 576c_1^2\underline{c_2^2} - 2928c_1\underline{c_2} + 6400\underline{c_2^2} - 2679, \quad 24c_1c_2 + 80\underline{s_2} - 61,$$
$$\left\{2928c_1c_2\underline{c_3^2} - 2928c_1c_2 - 2880c_2^2\underline{c_3^2} + 3721c_2^2 + 2679\underline{c_3^2} - 2679\right\},$$
$$\left\{576c_1^2\underline{c_3^2} - 576c_1^2 + 3520\underline{c_3^2} - 2679\right\}, \quad \left[696c_1c_2c_3 - 1920c_1\underline{s_3} + 4880c_2\underline{s_3} - 1769c_3\right],$$
$$\left[2088c_1^2c_2c_3 - 5760c_1^2\underline{s_3} - 5307c_1c_3 + 23200c_2c_3 - 26790\underline{s_3}\right],$$
$$\left[1464c_1c_3^2 - 1464c_1 - 2880c_2c_3^2 + 3721c_2 - 2320\underline{s_3}c_3\right],$$
$$72c_1^2c_2c_3 - 183c_1c_3 + 800c_2c_3 + 290s_3 - 640\underline{c_4}, \quad 3s_1 - 8\underline{s_4} \tag{4.17}$$

where the unknown variable at each stage in the solution process has been underlined. Note that either of the two braced expressions can be used to solve for c_3, and any one of those enclosed in brackets can be used to solve for s_3; the expression involving the fewest arithmetic operations is selected in each case. The following equations have been extracted from the above Gröbner basis and optimized using the `dsolve/numeric/optimize` routine:

$$z_1 = 9c_1^2 + 100, \quad z_2 = \frac{183}{8}c_1, \quad z_3 = z_1 - \frac{3721}{64}, \quad c_2 = \frac{z_2 \pm 10\sqrt{z_3}}{z_1},$$
$$s_2 = \frac{61}{80} - \frac{3}{10}c_1c_2, \quad c_3 = \pm\frac{1}{8}\sqrt{64 - \frac{841}{z_1 - 45}}, \quad z_2 = \frac{1}{80}c_3(z_1c_2 - z_2),$$
$$s_3 = \frac{29z_2}{z_3}, \quad c_4 = z_2 + \frac{29}{64}s_3, \quad s_4 = \frac{3}{8}s_1 \tag{4.18}$$

In this case, the positive and negative roots for c_2 and c_3 represent the two possible configurations of the mechanism: if the same sign is used for both roots, the mechanism will be

in the upright configuration; if the roots have opposite signs, the follower will be below the XY-plane. A second Gröbner basis is now generated for the same system of polynomials, but using a pure lexicographic ordering with $s_2 \succ c_2 \succ s_3 \succ c_3 \succ s_4 \succ c_4 \succ s_1 \succ c_1$, which again results in a triangular system:

$$\begin{aligned}
&s_1^2 + c_1^2 - 1, \quad 9c_1^2 - 64\underline{c_4^2} + 55, \quad 3s_1 - 8\underline{s_4}, \quad 576c_1^2\underline{c_3^2} - 576c_1^2 + 3520\underline{c_3^2} - 2679, \\
&29\underline{s_3} + 64c_3^2c_4 - 64c_4, \quad \left\{1464c_1\underline{c_2}c_4 - 1830c_1c_3 + 3600\underline{c_2}c_3 - 6400c_3^2c_4 + 2679c_4\right\}, \\
&\left\{72c_1^2\underline{c_2} - 183c_1 + 800\underline{c_2} - 640c_3c_4\right\}, \quad 24c_1c_2 + 80\underline{s_2} - 61
\end{aligned} \tag{4.19}$$

where either of the two braced expressions can be used to solve for c_2. In this case, we obtain a recursively solvable system of equations in which c_4 is solved first:

$$\begin{aligned}
&z_1 = 9c_1^2 + 55, \quad c_4 = \pm\frac{1}{8}\sqrt{z_1}, \quad s_4 = \frac{3}{8}s_1, \quad c_3 = \pm\frac{1}{8}\sqrt{64 - \frac{841}{z_1}}, \\
&s_3 = \frac{64}{29}c_4\left(1 - c_3^2\right), \quad c_2 = \frac{183c_1 + 640c_3c_4}{8\left(z_1 + 45\right)}, \quad s_2 = \frac{61}{80} - \frac{3}{10}c_1c_2
\end{aligned} \tag{4.20}$$

Note that triangular systems (4.18) and (4.20), both obtained using Gröbner bases, are more computationally efficient than that obtained using the characteristic pair of joints approach (4.15), as indicated in Table 4.1.

Similar results are obtained when the rotation axes corresponding to angles β_2 and β_3 are not assumed to be parallel. The following triangular system is obtained using the

Table 4.1: Cost of triangular systems for spatial four-bar mechanism when $\beta_0 = 0$

Approach	Computational cost of resulting triangular system			
	Multiplications	Additions	Square roots	Temporary variables
Characteristic pair of joints (4.15)	21	10	2	3
Gröbner basis, β_2 solved first (4.18)	16	8	2	3
Gröbner basis, β_4 solved first (4.20)	16	6	2	1

characteristic pair of joints technique when β_0 is treated as a symbolic parameter:

$$z_1 = 6\mathrm{c}_1, \quad z_2 = z_1^2 + 400, \quad z_3 = \pm\sqrt{z_2 - \frac{3721}{16}}, \quad z_4 = \mathrm{s}_0\mathrm{s}_1, \quad \mathrm{c}_2 = \frac{61z_1 + 80z_3}{4z_2},$$
$$\mathrm{s}_2 = \frac{305 - z_1z_3}{z_2}, \quad z_1 = \mathrm{c}_0^2, \quad z_2 = 1 - z_1, \quad z_3 = z_2\mathrm{c}_2^2, \quad z_5 = \mathrm{c}_1\mathrm{s}_2, \quad z_6 = \mathrm{c}_1\mathrm{c}_2,$$
$$z_7 = \mathrm{c}_0(10\mathrm{c}_2 - 3z_5), \quad z_1 = \frac{9}{4}\Big(\mathrm{c}_1^2\left(2z_1 + z_3 - 1\right) - z_1\Big) + \frac{3}{2}\Big(z_4z_7 + z_6\left(10z_2\mathrm{s}_2 - 4\right)\Big),$$
$$z_3 = \pm 2\sqrt{z_1 - 20\mathrm{s}_2 - 25z_3 + \frac{125}{4}}, \quad \mathrm{c}_3 = \frac{3z_4 + z_7}{z_3}, \quad \mathrm{s}_3 = \frac{10\mathrm{s}_2 + 3z_6 - 4}{z_3},$$
$$\mathrm{c}_4 = \frac{1}{8}z_3, \quad \mathrm{s}_4 = \frac{3}{8}\left(\mathrm{c}_0\mathrm{s}_1 + z_5\mathrm{s}_0\right) - \frac{5}{4}\mathrm{s}_0\mathrm{c}_2 \tag{4.21}$$

Note that system (4.21) reduces to system (4.15) upon substitution of $\beta_0 = 0$. The Gröbner basis approach proceeds as before. We first convert the constraint equations into polynomials and substitute the link lengths to obtain the following system:

$$\begin{aligned}
&\tfrac{3}{2}\mathrm{s}_1 - 4\mathrm{s}_0\mathrm{c}_3\mathrm{c}_4 - 4\mathrm{c}_0\mathrm{s}_4, \quad \tfrac{3}{2}\mathrm{c}_1 - 2\mathrm{c}_2 - 4\mathrm{s}_0\mathrm{s}_2\mathrm{s}_4 + 4\mathrm{c}_0\mathrm{s}_2\mathrm{c}_3\mathrm{c}_4 - 4\mathrm{c}_2\mathrm{s}_3\mathrm{c}_4,\\
&5 - 2\mathrm{s}_2 + 4\mathrm{s}_0\mathrm{c}_2\mathrm{s}_4 - 4\mathrm{c}_0\mathrm{c}_2\mathrm{c}_3\mathrm{c}_4 - 4\mathrm{s}_2\mathrm{s}_3\mathrm{c}_4, \quad \mathrm{s}_0^2 + \mathrm{c}_0^2 - 1, \quad \mathrm{s}_1^2 + \mathrm{c}_1^2 - 1,\\
&\mathrm{s}_2^2 + \mathrm{c}_2^2 - 1, \quad \mathrm{s}_3^2 + \mathrm{c}_3^2 - 1, \quad \mathrm{s}_4^2 + \mathrm{c}_4^2 - 1
\end{aligned} \tag{4.22}$$

Two Gröbner bases are generated using the same term orderings as before, but with s_0 and c_0 appended to the end of the lists. Using a pure lexicographic term ordering with $\mathrm{s}_4 \succ \mathrm{c}_4 \succ \mathrm{s}_3 \succ \mathrm{c}_3 \succ \mathrm{s}_2 \succ \mathrm{c}_2 \succ \mathrm{s}_1 \succ \mathrm{c}_1 \succ \mathrm{s}_0 \succ \mathrm{c}_0$ results in the following recursively solvable system:

$$z_1 = 64\left(9\mathrm{c}_1^2 + 100\right), \quad z_2 = 183\mathrm{c}_1, \quad \mathrm{c}_2 = \frac{8\left(z_2 \pm 10\sqrt{z_1 - 3721}\right)}{z_1},$$
$$\mathrm{s}_2 = \frac{61}{80} - \frac{3}{10}\mathrm{c}_1\mathrm{c}_2, \quad z_3 = \mathrm{c}_0\left(\frac{1}{8}z_1\mathrm{c}_2 - z_2\right), \quad z_2 = z_1 - 7817,$$
$$z_4 = 3255\mathrm{c}_0^2, \quad z_5 = z_4 - z_2, \quad z_6 = 3072\mathrm{c}_0\mathrm{s}_1, \quad z_5 = z_5^2 - z_6^2, \quad z_6 = \mathrm{s}_0\mathrm{s}_1,$$
$$\mathrm{c}_3 = \pm\sqrt{\frac{1}{z_5}\left(841z_2 - \frac{1682}{3255}z_4\left(z_1 - 5348.5\right) + 4036.8z_3z_6 + z_5\right)},$$
$$\mathrm{s}_3 = \frac{29\mathrm{c}_3\left(z_3 - 240z_6\right)}{10\left(z_1 + z_4 - 6976\right)}, \quad z_1 = \frac{1}{5120}z_1\mathrm{c}_2 - \frac{183}{640}\mathrm{c}_1, \quad z_2 = \frac{3}{8}\mathrm{s}_1,$$
$$\mathrm{c}_4 = \mathrm{c}_3\left(z_1\mathrm{c}_0 + z_2\mathrm{s}_0\right) + \frac{29}{64}\mathrm{s}_3, \quad \mathrm{s}_4 = z_2\mathrm{c}_0 - z_1\mathrm{s}_0 \tag{4.23}$$

A Gröbner basis is now generated for the same system of polynomials, but using a pure lexicographic ordering with $s_2 \succ c_2 \succ s_3 \succ c_3 \succ s_4 \succ c_4 \succ s_1 \succ c_1 \succ s_0 \succ c_0$, which leads to the following system of kinematic equations:

$$
\begin{aligned}
&z_1 = c_0 s_1, \quad z_2 = 9 s_1^2, \quad z_3 = 64 z_2, \quad s_4 = \frac{3}{8} z_1 \pm \frac{1}{64} s_0 \sqrt{3255 - z_3}, \\
&z_1 = 4096 - 3255 s_0^2 - 3072 z_1 s_4 + z_3, \quad c_4 = \pm \frac{1}{64} \sqrt{z_1}, \quad s_3 = \frac{1856 c_4}{z_1}, \\
&c_3 = \frac{8 s_3 \left(3 s_1 - 8 c_0 s_4\right)}{29 s_0}, \quad z_1 = s_0 s_4 - c_0 c_3 c_4, \quad s_2 = \frac{24 z_1 c_1 + 76.25}{109 - z_2}, \\
&c_2 = \frac{3}{10} c_1 s_2 - \frac{4}{5} z_1
\end{aligned}
\tag{4.24}
$$

Note that triangular systems (4.21) and (4.23) are of similar computational complexity, while system (4.24) is significantly more efficient than the other two, as shown in Table 4.2.

Although the characteristic pair of joints and Gröbner basis approaches both generate recursively solvable systems, three important differences should be noted. First, the Gröbner basis approach is capable of generating equations that involve fewer arithmetic operations and require fewer temporary variables for storing intermediate calculations, which can be attributed to its flexibility in the solution order. Secondly, the characteristic pair of joints approach requires the analyst to use joint coordinates, while the Gröbner basis approach can be used with joint coordinates, absolute coordinates, or any combination thereof. Finally, note that the characteristic pair of joints technique can accommodate

Table 4.2: Cost of triangular systems for spatial four-bar mechanism when $\beta_0 \neq 0$

Approach	Computational cost of resulting triangular system			
	Multiplications	Additions	Square roots	Temporary variables
Characteristic pair of joints (4.21)	40	20	2	7
Gröbner basis, β_2 solved first (4.23)	40	19	2	6
Gröbner basis, β_4 solved first (4.24)	30	10	2	3

symbolic system parameters (e.g., link lengths L_k). Although β_0 was treated as a symbolic parameter in the example above, a large number of indeterminates can make generating a Gröbner basis computationally impractical, so numeric values must typically be substituted for the symbolic parameters before generating the basis. As a consequence, the Gröbner basis must be recomputed if the system parameters are modified.

Kinematic simulations are performed in Maple to compare the computational efficiency of the following four solution approaches when $\beta_0 = 0$:

1. Solving the original constraint equations iteratively using the `fsolve` command.
2. Solving the original constraint equations iteratively using the `LinearSolve` command and Newton's method, with optimized procedures for evaluating the Jacobian and constraint equations at each iteration, where the converged solution from the previous time step is used as the initial guess for the current time step.
3. Solving triangular system (4.15) obtained using the characteristic pair of joints technique.
4. Solving triangular system (4.20) extracted from Gröbner basis (4.19).

The mechanism is simulated for 10 seconds using 1-millisecond time steps on a 3.00-GHz processor, where the crank angle is defined as $\beta_1(t) = 2\pi t$ [rad]. The `Digits` environment variable is set to 5 for all approaches, and Newton's method iterates to a tolerance of 10^{-3}; the simulation results are summarized in Table 4.3. Not only is the Gröbner basis approach faster than the other approaches, but it can be used to generate arbitrarily precise solutions without increasing the required number of arithmetic operations.

Table 4.3: Performance of spatial four-bar mechanism simulations in Maple

Approach	Simulation time	Performance relative to `fsolve`
`fsolve`	79.06 s	—
Newton's method	44.91 s	1.8× faster
Characteristic pair of joints	5.46 s	14.5× faster
Gröbner basis	5.12 s	15.4× faster

4.2 Multi-loop mechanisms

In this section, we present several multi-loop mechanisms for which fully triangular solutions can be obtained. The simplest such mechanisms are those consisting of a cascade of single-degree-of-freedom loops, where the output of each loop drives the motion of the subsequent loop. Two examples of this type are discussed in Section 4.2.1. In Section 4.2.2, a Stephenson-III six-bar mechanism is used to illustrate the situations in which fully triangular and block-triangular solutions can be expected; an exception is discussed in Section 4.2.3, where a Gröbner basis is generated for two independent loops simultaneously to avoid a block-triangular solution. A planar parallel robot is presented in Section 4.2.4 to demonstrate the effectiveness of the proposed technique in the context of dynamic simulation. Finally, the Gough–Stewart platform is used to demonstrate the application of the Gröbner basis approach to a complex spatial mechanism in Section 4.2.5.

4.2.1 Cascade of single-degree-of-freedom mechanisms

We begin our analysis of multi-loop mechanisms with the hydraulic excavator shown in Figure 4.4. The boom is pin-connected to the cab, and is raised and lowered by extending and retracting a hydraulic piston; the stick is similarly actuated. A third piston rotates the bucket indirectly through a pair of links, which provides a large range of motion while maintaining a small workspace. This system can be described as a cascade of three planar slider-crank mechanisms, all slider-actuated, followed by a planar four-bar mechanism. For mechanisms of this type, we can employ a simple solution strategy: solve each loop in series, starting from the ground frame.

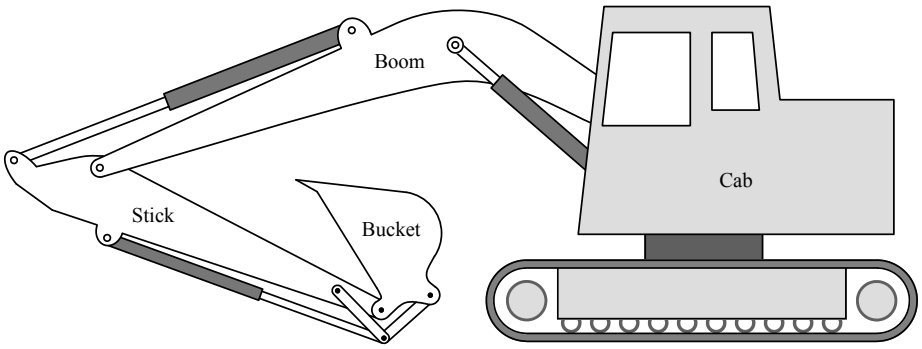


Figure 4.4: Hydraulic excavator

A similar strategy can be used to simulate the deployment of a synthetic aperture radar satellite antenna that was launched on the European research satellite ERS-1 [57]. Each side of the folding truss holds two panels and is deployed in two stages, as shown in Figure 4.5 for the right half of the satellite. Although this is a 2-DOF mechanism, only one joint is actuated during each phase of deployment. In the first stage, the upper panel is extended by a passive spring in joint A, which locks once straightened. A motor in joint B is then activated to swing the panels into their final positions. This system can be described as a cascade of two planar four-bar mechanisms during each phase of deployment; thus, we can employ the same strategy as above, solving each loop in series. In the remainder of this section, we consider mechanisms whose loops are not arranged in a strictly cascading manner.

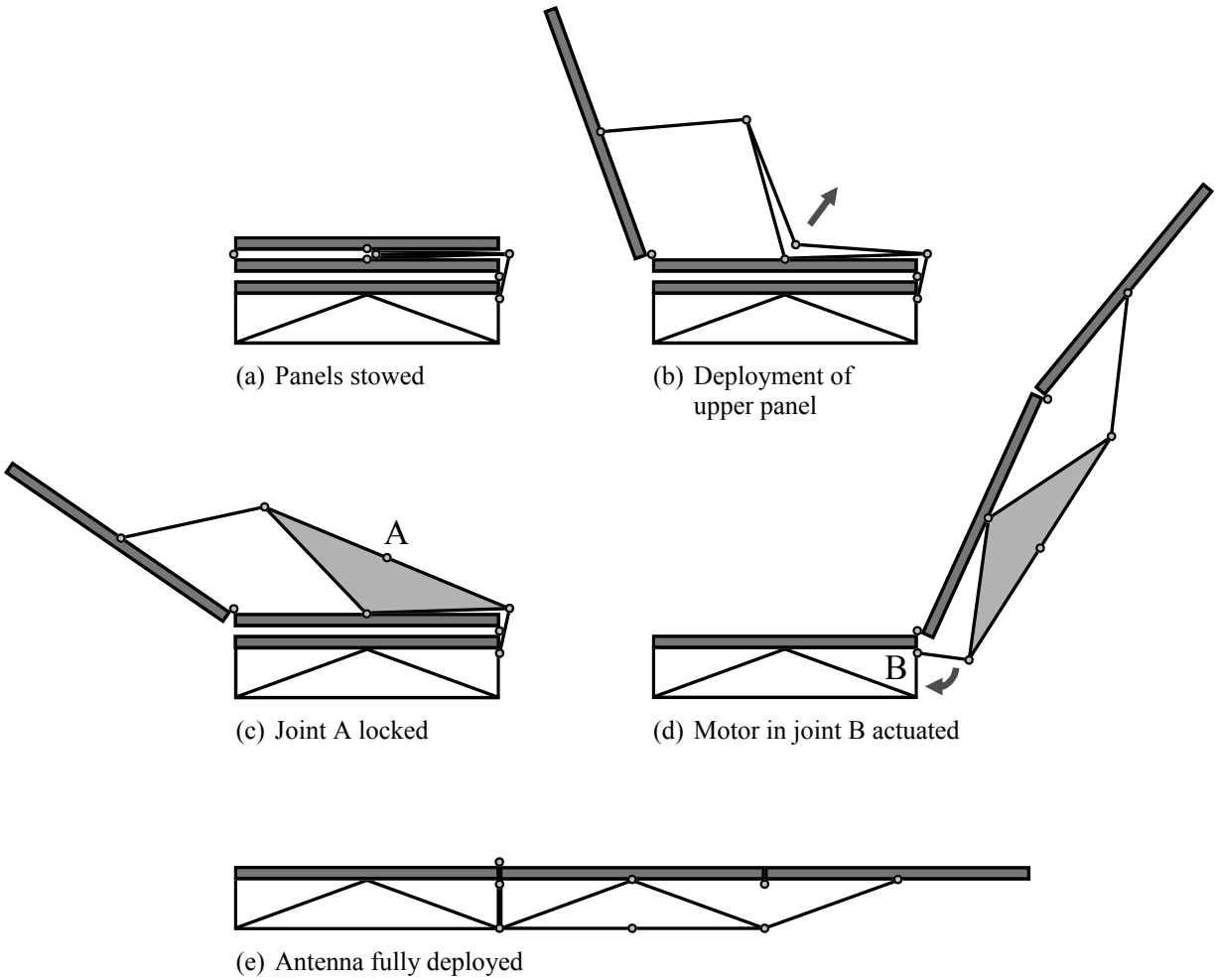


Figure 4.5: Deployment of synthetic aperture radar satellite antenna

4.2.2 Stephenson-III six-bar mechanism

We now consider the 1-DOF Stephenson-III six-bar mechanism shown in Figure 4.6. If joint coordinates $\mathbf{q} = \{\beta_1, \beta_2, \beta_3, \beta_4, \theta_1\}$ are used to model this system, $m = n - f = 4$ constraint equations are generated:

$$\begin{aligned} \phi_1(\beta_1, \beta_2, \beta_3, \beta_4) &= 0 \qquad & \phi_3(\theta_1, \theta_2, \theta_3) &= 0 \\ \phi_2(\beta_1, \beta_2, \beta_3, \beta_4) &= 0 \qquad & \phi_4(\theta_1, \theta_2, \theta_3) &= 0 \end{aligned} \tag{4.25}$$

where constants relate θ_2 and θ_3 to β_2 and β_3, respectively. Note that ϕ_1 and ϕ_2 represent the loop-closure constraints for loop L_β, and ϕ_3 and ϕ_4 represent those for the second independent loop, L_θ. Local solutions are generated by transforming the constraint equations associated with each independent loop into Gröbner bases. Although it is often possible to generate a single Gröbner basis for an entire system, processing the constraint equations for each loop separately can reduce the total computation time considerably. In the present example, we shall focus on determining the sequence in which the local solutions are to be computed.

An important consideration in the Gröbner basis approach is the choice of independent coordinates $\mathbf{q}_i$. In the context of kinematic simulation, the inputs to a system are specified by the analyst. Since each step in the global solution flow must involve computing unknown coordinates from known coordinates, the choice of inputs in a kinematic simulation is directly related to the triangularizability of the system. If we were to use $\beta_1 = f(t)$ as the

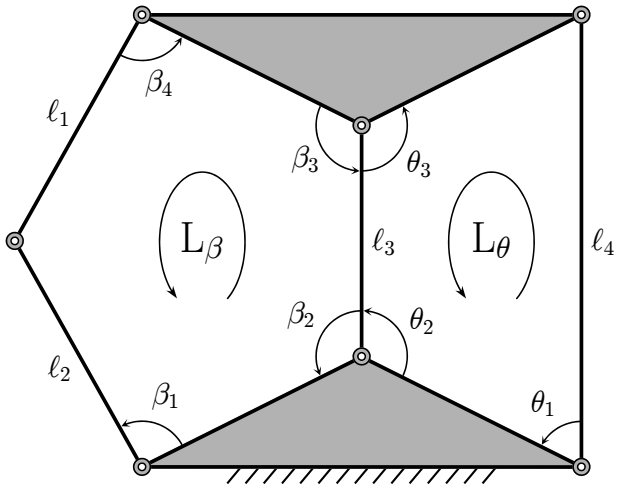


Figure 4.6: Stephenson-III six-bar mechanism

driven angle in this mechanism, it is only possible to obtain a block-triangular solution in general. This limitation can be explained kinematically by noting that loop L_β is a closed chain of five links and, therefore, has 2 local DOF; knowledge of β_1 alone is insufficient to solve this loop. Mathematically, note that knowledge of β_1 leaves three unknowns in ϕ_1 and ϕ_2 (β_2, β_3, and β_4), and three unknowns in ϕ_3 and ϕ_4 (θ_1, θ_2, and θ_3), so neither of these pairs of equations can be solved immediately. Thus, if β_1 is chosen as the driven angle, the global solution will generally involve iteration over at least one coordinate, as shown for β_2 in Figure 4.7. Although iteration is used to determine β_2, the remaining

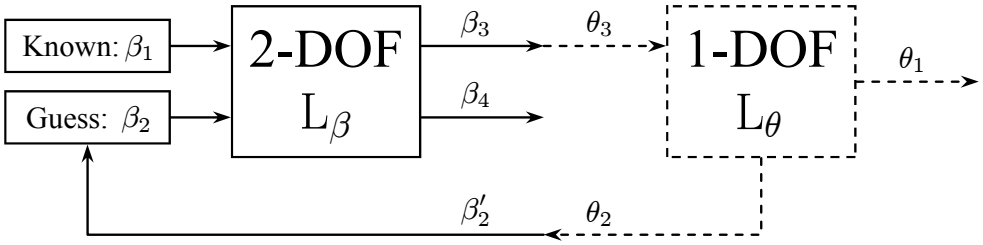


Figure 4.7: Kinematic solution flow for Stephenson-III six-bar when β_1 is driven

angles can be solved recursively once β_2 has been found, so this global solution strategy is said to have a block-triangular form. Note that the iterative procedure only involves repeated computations of β_3 given β_2, and θ_2 given θ_3, so the remaining angles need not be computed until convergence has been reached. By reducing the amount of computation required for each iteration and computing the remaining angles recursively, the block-triangular solution strategy shown in Figure 4.7 yields more efficient simulations than a fully iterative approach. Of course, a fully triangular solution is even more efficient and, in this case, can be obtained if $\theta_1 = f(t)$ is used as the driven angle, as shown in Figure 4.8.

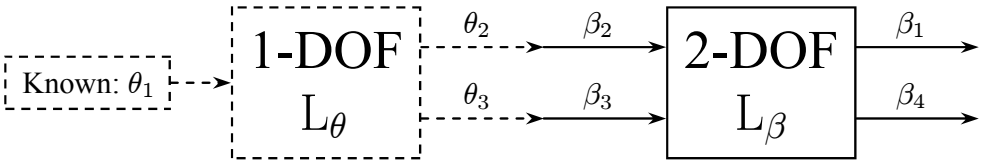


Figure 4.8: Kinematic solution flow for Stephenson-III six-bar when θ_1 is driven

To demonstrate the computational advantage associated with triangularizing the constraint equations, kinematic simulations are performed using the systems shown in Figures 4.7 and 4.8, and the simulation times are compared to those obtained using a typical

Newton–Raphson iterative approach. The initial configuration of the system is shown in Figure 4.6. The leg lengths are chosen to be $\ell_1 = \ell_2 = 0.295$ [m], $\ell_3 = 0.26$ [m], and $\ell_4 = 0.51$ [m], and the two shaded triangles are isosceles with base 0.5 [m] and height 0.125 [m]. All coefficients are converted into rational numbers using the `convert/rational` function in Maple. The motion driver $\beta_1(t) = 2.09 + \sin(\pi t)$ [rad] is used to drive the block-triangular solution; tabulated data for $\theta_1(t)$ from the simulation of the block-triangular solution is used to drive the fully triangular solution through the same trajectory. Simulations are performed using optimized procedures generated with the `codegen/optimize` routine in Maple, which are exported to the C programming language. Rather than compute the Jacobian required for Newton–Raphson iteration, we simply use the secant method to determine β_2 in the block-triangular solution; Gaussian elimination with partial pivoting is used to solve the linear system of equations in the Newton–Raphson approach. The resulting simulation code is compiled using the default Lcc compiler in MATLAB R14 (Lcc) as well as the Microsoft Visual C/C++ compiler (MS). In each case, a 1-second simulation is performed on a 3.00-GHz processor using 1-millisecond time steps; the average simulation times are shown in Tables 4.4 and 4.5. Note that, while the block-triangular solution outperforms the iterative approach, the fully triangular solution provides exact results in even less time.

Table 4.4: Performance of Stephenson-III six-bar simulations when β_1 is driven

Compiler	Gröbner basis · tolerance 10^{-3}	Gröbner basis · tolerance 10^{-6}	Newton–Raphson · tolerance 10^{-3}	Newton–Raphson · tolerance 10^{-6}
Lcc	3.74 ms	5.96 ms	6.41 ms	9.27 ms
MS	1.50 ms	2.50 ms	2.71 ms	3.96 ms

Table 4.5: Performance of Stephenson-III six-bar simulations when θ_1 is driven

Compiler	Gröbner basis · exact	Newton–Raphson · tolerance 10^{-3}	Newton–Raphson · tolerance 10^{-6}
Lcc	2.01 ms	6.61 ms	9.32 ms
MS	0.85 ms	2.71 ms	3.70 ms

4.2.3 Aircraft landing gear mechanism

In the previous section, we studied a 1-DOF mechanism with independent loops of 1 and 2 local DOF, and found that applying a kinematic input to the 2-DOF loop resulted in a block-triangular solution. This observation is generally applicable to systems whose loops are triangularized separately; however, it may be possible to obtain a fully triangular solution in such situations by triangularizing all loops simultaneously. Consider the mechanism shown in Figure 4.9, which was designed for deploying and retracting landing gear at the nose of an aircraft [52]; the topology of the system is shown in Figure 4.10. The main strut is pin-connected to the fuselage at point A, the piston at point B, and the lower link at point F. Revolute joints at points C, D, and E attach the drag strut to the base of the hydraulic cylinder, the fuselage, and the lower link, respectively. When the piston is extended, the main strut pivots about point A in a counterclockwise direction, while the drag strut rotates about point D in a clockwise direction due to its attachment to the base of the hydraulic cylinder.

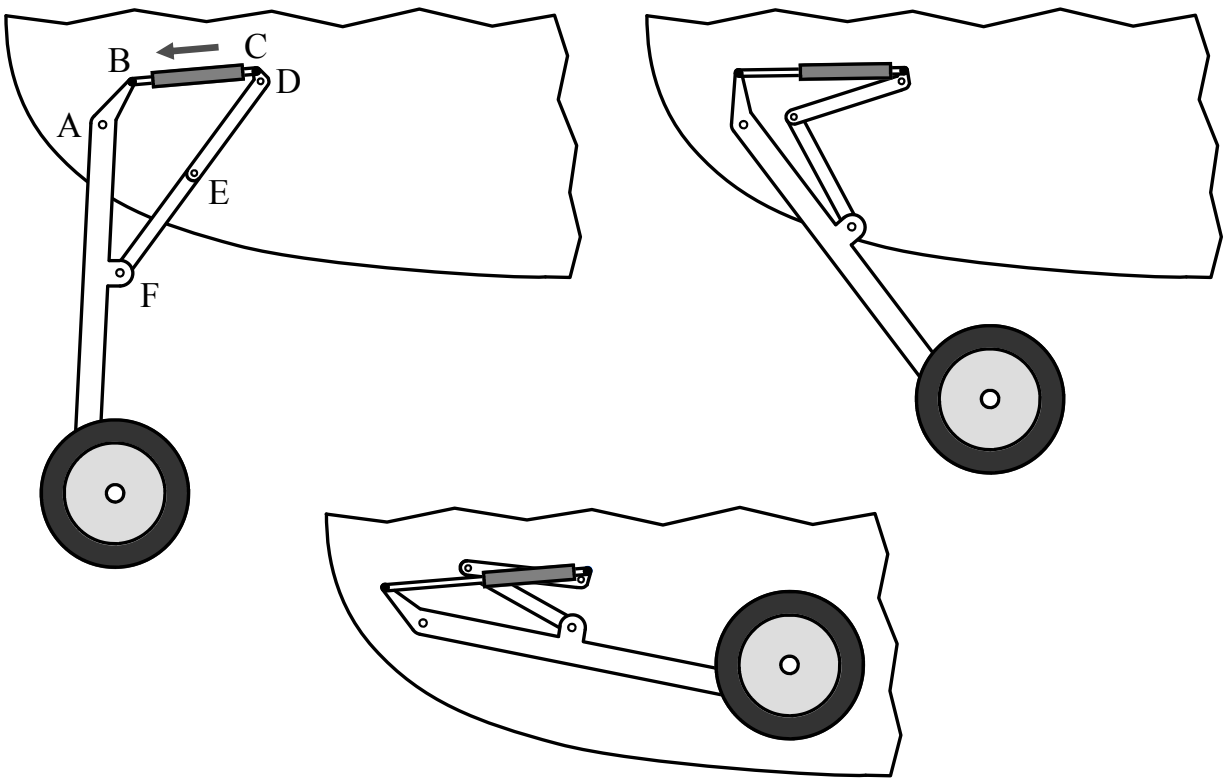


Figure 4.9: Aircraft landing gear mechanism

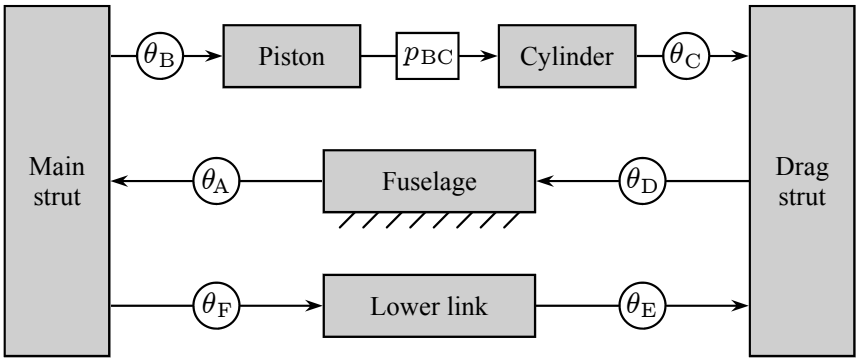


Figure 4.10: Topology of aircraft landing gear mechanism

The 1-DOF aircraft landing gear mechanism consists of two independent loops, and has nearly the same topology as the Stephenson-III mechanism studied in the previous section. The lower loop (ADEFA) is a 1-DOF planar four-bar mechanism, which could be solved first if one of its angles were driven. In this case, the actuation is applied to the upper loop (ABCDA), which is a 2-DOF five-bar mechanism; knowledge of the length BC at a particular instant of time does not allow us to solve the upper loop in isolation, since it effectively becomes a 1-DOF planar four-bar. Thus, we are faced with the same situation as when β_1 was used to drive the Stephenson-III mechanism; however, rather than resort to a block-triangular solution, a single Gröbner basis can be generated for the entire system of constraint equations. Although a triangular solution is obtained in this case, the complexity of the Gröbner basis computation precludes the use of this strategy in general. Furthermore, as will be shown, we obtain a sextic equation for one variable, which is not generally solvable using a finite number of arithmetic operations.

Using joint coordinates $\mathbf{q} = \{\theta_A, \theta_B, \theta_D, \theta_F, p_{BC}\}$ and substituting the geometric parameters given in Appendix A.1, we obtain the following four constraint equations:

$$\mathrm{s_A s_B} p_{BC} - \mathrm{c_A c_B} p_{BC} + 62\mathrm{s_A} - 42\mathrm{c_A} + \tfrac{1697}{100}\mathrm{s_D} + \tfrac{152}{25}\mathrm{c_D} + 231 = 0 \tag{4.26}$$

$$\mathrm{s_A c_B} p_{BC} + \mathrm{c_A s_B} p_{BC} + 42\mathrm{s_A} + 62\mathrm{c_A} + \tfrac{152}{25}\mathrm{s_D} - \tfrac{1697}{100}\mathrm{c_D} - 62 = 0 \tag{4.27}$$

$$\tfrac{892}{5}\mathrm{s_A s_F} - \tfrac{892}{5}\mathrm{c_A c_F} - 212\mathrm{s_A} - 25\mathrm{c_A} - \tfrac{822}{5}\mathrm{c_D} + 231 = 0 \tag{4.28}$$

$$\tfrac{892}{5}\mathrm{s_A c_F} + \tfrac{892}{5}\mathrm{c_A s_F} + 25\mathrm{s_A} - 212\mathrm{c_A} - \tfrac{822}{5}\mathrm{s_D} - 62 = 0 \tag{4.29}$$

where $\mathrm{s}_k = \sin(\theta_k)$ and $\mathrm{c}_k = \cos(\theta_k)$. Note that (4.26) and (4.27) represent the loop-closure constraints for the upper loop, and (4.28) and (4.29) represent those for the lower loop.

Since knowledge of the piston length p_{BC} leaves three unknowns in the upper loop (θ_A, θ_B, and θ_D) and three unknowns in the lower loop (θ_A, θ_D, and θ_F), neither of these pairs of equations can be solved immediately in isolation. Generating a Gröbner basis using a pure lexicographic term ordering with $c_F \succ s_F \succ c_A \succ s_A \succ c_D \succ s_D \succ c_B \succ s_B \succ p_{BC}$ results in the triangular system described in Table 4.6; similar results are obtained for other elimination orders. Although a triangular system has been obtained, its numerical evaluation still requires iteration to solve the sextic equation $g_1(s_B, p_{BC}) = 0$ for s_B. Note that four of the six solution branches are complex, the fifth describes the motion shown in Figure 4.9, and the sixth describes a similar motion of the main strut, but where the drag strut rotates counterclockwise. Finally, whereas the coefficients in the original constraint equations are no more than four digits in length, those in the triangular system are substantially longer. This observation indicates both the complexity of the Gröbner basis computation in this case, as well as the danger of adopting such a strategy in general.

Table 4.6: Triangular system obtained for aircraft landing gear mechanism

Polynomial	Degree	Number of terms	Longest coefficient
$g_1(s_B, p_{BC})$	6 in s_B	28	62 digits
$g_2(c_B, s_B, p_{BC})$	1 in c_B	40	132 digits
$g_3(s_D, c_B, s_B, p_{BC})$	1 in s_D	37	245 digits
$g_4(c_D, s_D, c_B, s_B, p_{BC})$	1 in c_D	6	9 digits
$g_5(s_A, c_D, s_D, c_B, s_B, p_{BC})$	1 in s_A	37	264 digits
$g_6(c_A, s_A, c_D, s_D, c_B, s_B, p_{BC})$	1 in c_A	37	264 digits
$g_7(s_F, c_A, s_A, c_D, s_D, c_B, s_B, p_{BC})$	1 in s_F	37	252 digits
$g_8(c_F, s_F, c_A, s_A, c_D, s_D, c_B, s_B, p_{BC})$	1 in c_F	37	254 digits

4.2.4 Planar parallel robot

The last two example systems presented in this chapter are parallel robots with full mobility. A parallel robot is a multi-loop mechanism whose end-effector is connected to the ground by two or more chains; robots whose end-effectors can be positioned and oriented arbitrarily in all coordinate directions are said to have full mobility. Although a

parallel robot typically has a smaller workspace than a serial robot of similar size, the closed-kinematic-chain topology of the former makes it more rigid and, therefore, more accurate [49]. Furthermore, if all the actuators can be placed on the base, the resulting robot will be both lighter and faster [84]. In addition to their many useful industrial applications, full-mobility parallel robots also provide a convenient solution strategy for multi-loop mechanisms with limited mobility, as will be shown in Chapter 5.

Consider the 3-DOF planar parallel robot shown in Figure 4.11, the design of which was studied by Gosselin and Angeles [48]. If modelled using joint coordinates $\mathbf{q} = \{\theta_1, \ldots, \theta_7\}$, the following $m = n - f = 4$ constraint equations are obtained:

$$\begin{aligned} \phi_1(\theta_1, \theta_2, \theta_4, \theta_5, \theta_7) &= 0 \qquad & \phi_3(\theta_1, \theta_3, \theta_4, \theta_6, \theta_7) &= 0 \\ \phi_2(\theta_1, \theta_2, \theta_4, \theta_5, \theta_7) &= 0 \qquad & \phi_4(\theta_1, \theta_3, \theta_4, \theta_6, \theta_7) &= 0 \end{aligned} \tag{4.30}$$

where ϕ_1 and ϕ_2 are the loop-closure constraints for the first loop, herein referred to as loop L_1, and ϕ_3 and ϕ_4 are those for the second loop, L_2. Loops L_1 and L_2 are both closed chains of six links and, therefore, each has 3 local DOF. Clearly, choosing $\mathbf{q}_i = \{\theta_1, \theta_2, \theta_3\}$ leads to an iterative solution, since each pair of constraint equations would involve three unknowns. In a kinematic simulation, where θ_1, θ_2, and θ_3 might be desired motor angles as

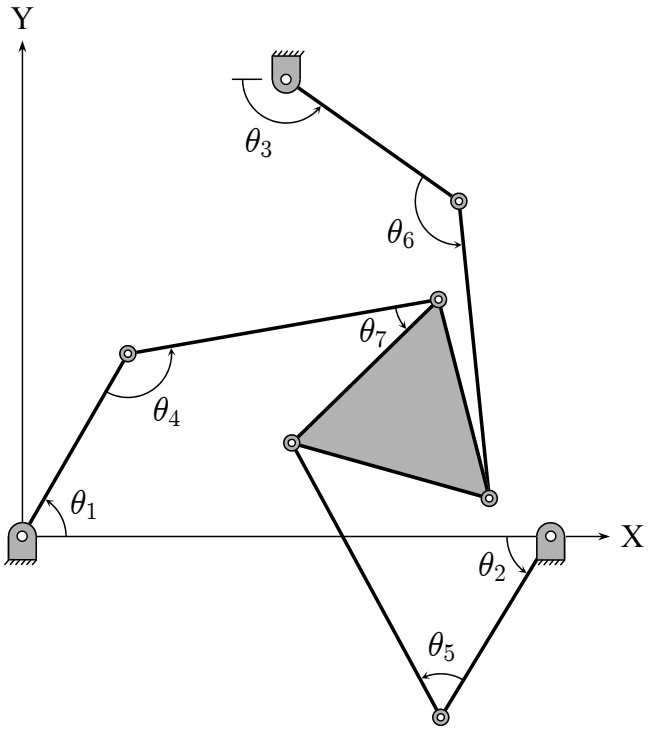


Figure 4.11: Planar parallel robot

functions of time, we would be forced to use this iterative solution. In a forward dynamic simulation, however, a more convenient set of independent coordinates can be chosen. Selecting $\mathbf{q}_i = \{\theta_1, \theta_2, \theta_4\}$, for example, results in a triangular system, since ϕ_1 and ϕ_2 can be used to determine θ_5 and θ_7 recursively, and θ_1, θ_4, and θ_7 can then be used as the inputs to loop L_2. Selecting $\mathbf{q}_i = \{\theta_1, \theta_4, \theta_7\}$ also triangularizes the system, but in this case we have the ability to solve loops L_1 and L_2 in parallel, distributing the computation over two processors.

As mentioned above, one of the benefits of the Gröbner basis approach is its indifference to the generalized coordinates used to model a system. This flexibility is exploited here by using absolute coordinates to describe the motion of the end-effector and joint coordinates elsewhere. Using generalized coordinate vector $\mathbf{q}' = \{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, x_{ee}, y_{ee}, \theta_{ee}\}$, we obtain the following $m = n - f = 6$ constraint equations:

$$\begin{aligned} &\phi'_1(\theta_1, \theta_4, x_{ee}, \theta_{ee}) = 0 \qquad \phi'_3(\theta_2, \theta_5, x_{ee}, \theta_{ee}) = 0 \qquad \phi'_5(\theta_3, \theta_6, x_{ee}, \theta_{ee}) = 0 \\ &\phi'_2(\theta_1, \theta_4, y_{ee}, \theta_{ee}) = 0 \qquad \phi'_4(\theta_2, \theta_5, y_{ee}, \theta_{ee}) = 0 \qquad \phi'_6(\theta_3, \theta_6, y_{ee}, \theta_{ee}) = 0 \end{aligned} \tag{4.31}$$

Note that this choice of modelling coordinates results in three independent kinematic loops and six constraint equations, but these equations are simpler than those obtained using purely joint coordinates. Now, choosing as independent coordinates $\mathbf{q}'_i = \{x_{ee}, y_{ee}, \theta_{ee}\}$, a parallelizable triangular system is obtained that can be distributed over three processors. Since the constraint equations are simpler in this case, the Gröbner bases are generated much faster than before, as indicated in Table 4.7. Note that the projected dynamic equations, described in Section 2.1.5, are also more efficient in this case.

We conclude this example with a comparison between the performance of the two triangular systems described above and that of a typical approach using Newton–Raphson iteration. The initial configuration of the mechanism is shown in Figure 4.11, which is assumed to lie in a horizontal plane; the system parameters are given in Appendix A.2. The approximation $\sqrt{3} \approx 362/209$ provided by the `convert/rational` function is used to convert the coefficients in the constraint equations into rational numbers. Simulations are performed using procedures generated with the `codegen/optimize` routine in Maple, which are exported to the C programming language; Gaussian elimination with partial pivoting is used to solve the linear system of equations in the Newton–Raphson approach. The resulting simulation code is compiled using the default Lcc compiler in MATLAB R14

Table 4.7: Comparison of triangular systems obtained for planar parallel robot

Metric	$\mathbf{q}_i = \{\theta_1, \theta_4, \theta_7\}$	$\mathbf{q}'_i = \{x_{ee}, y_{ee}, \theta_{ee}\}$
Computational cost of original kinematic constraint equations	112 multiplications 47 additions	40 multiplications 31 additions
Time to compute all Gröbner bases on a 3.00-GHz processor	11 minutes, 27 seconds	4.3 seconds
Computational cost of optimized triangular system extracted from Gröbner basis	531 multiplications 522 additions 102 temporary variables	293 multiplications 286 additions 52 temporary variables
Computational cost of optimized dynamic equations after applying projection	1195 multiplications 926 additions 598 temporary variables	642 multiplications 268 additions 232 temporary variables

(Lcc) as well as the Microsoft Visual C/C++ compiler (MS). In each case, a 3-second simulation is performed on a single 3.00-GHz processor using 1-millisecond time steps and a first-order Euler integration scheme; the average simulation times are shown in Tables 4.8 and 4.9. Tabulated data for the independent coordinates from each dynamic simulation is used to drive the corresponding kinematic simulation through the same trajectory; thus, the "Kinematic" results shown in Tables 4.8 and 4.9 represent the amount of time required to process the kinematic portion of each dynamic simulation. The Gröbner basis approach clearly outperforms the iterative approach in all cases, providing dynamic simulation code that executes up to 300 times faster than real time. Note that generalized coordinate vector $\mathbf{q}'$ provides a faster Gröbner basis solution than vector $\mathbf{q}$, as expected. This observation confirms that the reduced computational cost associated with the $\mathbf{q}'$ solution, as shown in Table 4.7, leads to a noticeable increase in performance. Also note that the Newton–Raphson approach is slower when using $\mathbf{q}'$, which is to be expected since a larger linear system is being solved at each iteration. Finally, while parallel processing has not been employed here, the parallelism inherent in the Gröbner basis solutions presented above can be exploited to obtain further increases in performance.

Table 4.8: Performance of planar parallel robot simulations with $\mathbf{q}_i = \{\theta_1, \theta_4, \theta_7\}$

Compiler	Simulation	Gröbner basis · exact	Newton–Raphson · tolerance 10^{-3}	Newton–Raphson · tolerance 10^{-6}
Lcc	Kinematic	8.75 ms	13.81 ms	25.21 ms
	Dynamic	30.67 ms	34.89 ms	46.24 ms
MS	Kinematic	4.58 ms	5.62 ms	10.26 ms
	Dynamic	15.00 ms	16.76 ms	21.35 ms

Table 4.9: Performance of planar parallel robot simulations with $\mathbf{q}'_i = \{x_{ee}, y_{ee}, \theta_{ee}\}$

Compiler	Simulation	Gröbner basis · exact	Newton–Raphson · tolerance 10^{-3}	Newton–Raphson · tolerance 10^{-6}
Lcc	Kinematic	7.48 ms	21.41 ms	40.00 ms
	Dynamic	24.05 ms	38.13 ms	56.78 ms
MS	Kinematic	3.25 ms	9.22 ms	16.72 ms
	Dynamic	9.70 ms	17.08 ms	23.44 ms

4.2.5 Gough–Stewart platform

The final fully triangular system we shall consider is the 6-DOF Gough–Stewart platform shown in Figure 4.12, where the upper and lower leg segments are connected with prismatic joints (P), and are connected to the platform and base with spherical (S) and universal (U) joints, respectively. This spatial parallel robot was originally designed in 1947 for

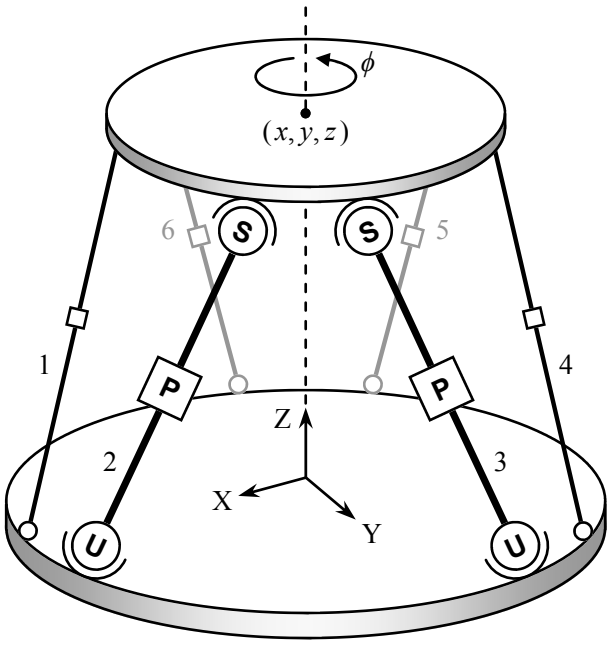


Figure 4.12: Gough–Stewart platform

testing the wear of vehicle tires [84], and has since become known for its use as a motion platform in flight and driving simulators [1, 113]. As such, the dynamic performance of the Gough–Stewart platform has been a popular topic of study [135], as has the direct kinematics problem [59]—that is, determining the position and orientation of the platform given the leg lengths. As in the previous example, we use absolute coordinates to describe the motion of the full-mobility end-effector (platform) and joint coordinates elsewhere, thereby focusing on the inverse kinematics problem: given the position and orientation of the platform, determine the corresponding leg lengths [35]. Generalized coordinates $\{x, y, z\}$ represent the position of the center of mass of the platform in the global reference frame, and $\{\phi, \theta, \psi\}$ represent the corresponding 3-2-1 Euler angles. The configuration of universal joint k is specified by its angles of rotation about the global X-axis (α_k) and the rotated Y′-axis (β_k), making each local Z″-axis coincident with its respective leg; the

prismatic joint displacements are given by s_k. In the dynamic simulations described below, forces $F_1 = F_3 = F_5 = -0.1\sin(2\pi t/3)$ [N] and $F_2 = F_4 = F_6 = 0.1\sin(2\pi t/3)$ [N] are applied at the prismatic joints (linear actuators) to rotate the platform about its vertical axis, starting from the configuration shown in Figure 4.12.

Using a purely joint coordinate formulation with $\mathbf{q}_{\text{joint}} = \{\alpha_{1\ldots6}, \beta_{1\ldots6}, s_{1\ldots6}, \zeta, \eta, \xi\}$, where $\{\zeta, \eta, \xi\}$ are the angles associated with one spherical joint, we obtain $m = n - f = 15$ constraint equations, three for each independent loop. If we instead use $\mathbf{q}_{\text{mixed}} = \{x, y, z, \phi, \theta, \psi, \alpha_{1\ldots6}, \beta_{1\ldots6}, s_{1\ldots6}\}$—that is, absolute coordinates to describe the motion of the platform and joint coordinates elsewhere, as suggested by Geike and McPhee [44]—we obtain $m = n - f = 18$ constraint equations, three for each leg k:

$$\mathbf{\Phi}_k = \left\{ \begin{array}{c} x - B_{x,k} - L_k \sin(\beta_k) + f_{x,k} \\ y - B_{y,k} + L_k \sin(\alpha_k)\cos(\beta_k) + f_{y,k} \\ z - L_k \cos(\alpha_k)\cos(\beta_k) + f_{z,k} \end{array} \right\} = \mathbf{0}, \quad k = 1, \ldots, 6 \tag{4.32}$$

where L_k is the total length of the k^{th} leg, $B_k = (B_{x,k}, B_{y,k}, 0)$ is the position of the corresponding universal joint on the base, and $f_{x,k}$, $f_{y,k}$, and $f_{z,k}$ are functions of ϕ, θ, ψ, and P_k, the position of the k^{th} spherical joint on the platform. Although the use of $\mathbf{q}_{\text{mixed}}$ results in more constraint equations and more independent loops than using $\mathbf{q}_{\text{joint}}$, the constraints are simpler [75] and, therefore, more suitable for generating a Gröbner basis. Choosing as independent coordinates $\mathbf{q}_i = \{x, y, z, \phi, \theta, \psi\}$, the three constraint equations associated with leg k involve three dependent coordinates: α_k, β_k, and s_k. Thus, the embedding technique is applied to eliminate the Lagrange multipliers from the dynamic equations and obtain six second-order ODEs for $\ddot{\mathbf{q}}_i$, which are expressed in first-order form. These equations can be integrated forward in time to determine the values of the independent coordinates; given $\mathbf{q}_i$, the constraint equations can be used to determine $\mathbf{q}_d = \{\alpha_{1\ldots6}, \beta_{1\ldots6}, s_{1\ldots6}\}$.

Since the constraint equations associated with leg k can be solved independently of the others, each independent loop is triangularized separately. The system parameters are given in Appendix A.3 and are obtained from the work of Tsai [130], who provides geometric quantities to three decimal places. As such, the floating-point coefficients in (4.32) can be converted into rational numbers exactly using the `convert/rational` function in Maple. We again use transformations of the form $\mathrm{s}_\vartheta = \sin(\vartheta)$ and $\mathrm{c}_\vartheta = \cos(\vartheta)$ to eliminate the

trigonometric functions, and introduce auxiliary equations $\mathrm{s}_\vartheta^2 + \mathrm{c}_\vartheta^2 - 1 = 0$ to preserve the relationship between these variables. Once these substitutions have been performed, we obtain the following five polynomial equations in five unknowns for the first leg:

$$x + \tfrac{17}{100}\mathrm{c}_\phi\mathrm{c}_\theta + \tfrac{119}{200}\left(\mathrm{c}_\phi\mathrm{s}_\theta\mathrm{s}_\psi - \mathrm{s}_\phi\mathrm{c}_\psi\right) - \tfrac{2}{5}\left(\mathrm{c}_\phi\mathrm{s}_\theta\mathrm{c}_\psi - \mathrm{s}_\phi\mathrm{s}_\psi\right) - (2+s_1)\,\mathrm{s}_{\beta_1} + \tfrac{53}{25} = 0 \quad (4.33)$$

$$y + \tfrac{17}{100}\mathrm{s}_\phi\mathrm{c}_\theta + \tfrac{119}{200}\left(\mathrm{s}_\phi\mathrm{s}_\theta\mathrm{s}_\psi + \mathrm{c}_\phi\mathrm{c}_\psi\right) - \tfrac{2}{5}\left(\mathrm{s}_\phi\mathrm{s}_\theta\mathrm{c}_\psi - \mathrm{c}_\phi\mathrm{s}_\psi\right) + (2+s_1)\,\mathrm{s}_{\alpha_1}\mathrm{c}_{\beta_1} - \tfrac{687}{500} = 0 \quad (4.34)$$

$$z - \tfrac{17}{100}\mathrm{s}_\theta + \tfrac{119}{200}\mathrm{c}_\theta\mathrm{s}_\psi - \tfrac{2}{5}\mathrm{c}_\theta\mathrm{c}_\psi - (2+s_1)\,\mathrm{c}_{\alpha_1}\mathrm{c}_{\beta_1} = 0 \quad (4.35)$$

$$\mathrm{s}_{\alpha_1}^2 + \mathrm{c}_{\alpha_1}^2 - 1 = 0 \quad (4.36)$$

$$\mathrm{s}_{\beta_1}^2 + \mathrm{c}_{\beta_1}^2 - 1 = 0 \quad (4.37)$$

where the values of independent coordinates $\mathbf{q}_i = \{x, y, z, \phi, \theta, \psi\}$ are known, and the values of dependent coordinates $\mathbf{q}_{d_1} = \{\mathrm{s}_{\alpha_1}, \mathrm{c}_{\alpha_1}, \mathrm{s}_{\beta_1}, \mathrm{c}_{\beta_1}, s_1\}$ are unknown. Systems of the same form are obtained for the other five legs.

The final preparation before generating a Gröbner basis is determining a suitable term ordering with respect to which the basis is to be generated. By traversing the topological graph of the system, we obtain the following term ordering for leg k:

$$\mathrm{s}_{\alpha_k} \succ \mathrm{c}_{\alpha_k} \succ \mathrm{s}_{\beta_k} \succ \mathrm{c}_{\beta_k} \succ s_k \succ \mathrm{s}_\psi \succ \mathrm{c}_\psi \succ \mathrm{s}_\theta \succ \mathrm{c}_\theta \succ \mathrm{s}_\phi \succ \mathrm{c}_\phi \succ z \succ y \succ x$$

where $\mathbf{q}_d \succ \mathbf{q}_i$, thereby resulting in a triangular system in which the dependent coordinates can be solved recursively, given values of the independent coordinates. Note that any term ordering of the form $\mathbf{q}_d \succ \mathbf{q}_i$ can be used, most of which result in systems of comparable complexity. Each Gröbner basis is generated on one core of a 3.00-GHz Intel Xeon E5472 processor in about 2 hours using the 64-bit version of Maple 14. A recursively solvable system of the following form is extracted from the k^{th} Gröbner basis:

$$s_k = g_{1,k}(\mathrm{s}_\psi, \mathrm{c}_\psi, \mathrm{s}_\theta, \mathrm{c}_\theta, \mathrm{s}_\phi, \mathrm{c}_\phi, z, y, x) \quad (4.38)$$

$$\mathrm{c}_{\beta_k} = g_{2,k}(s_k, \mathrm{s}_\psi, \mathrm{c}_\psi, \mathrm{s}_\theta, \mathrm{c}_\theta, \mathrm{s}_\phi, \mathrm{c}_\phi, z, y, x) \quad (4.39)$$

$$\mathrm{s}_{\beta_k} = g_{3,k}(\mathrm{c}_{\beta_k}, s_k, \mathrm{s}_\psi, \mathrm{c}_\psi, \mathrm{s}_\theta, \mathrm{c}_\theta, \mathrm{s}_\phi, \mathrm{c}_\phi, z, y, x) \quad (4.40)$$

$$\mathrm{c}_{\alpha_k} = g_{4,k}(\mathrm{s}_{\beta_k}, \mathrm{c}_{\beta_k}, s_k, \mathrm{s}_\psi, \mathrm{c}_\psi, \mathrm{s}_\theta, \mathrm{c}_\theta, \mathrm{s}_\phi, \mathrm{c}_\phi, z, y, x) \quad (4.41)$$

$$\mathrm{s}_{\alpha_k} = g_{5,k}(\mathrm{c}_{\alpha_k}, \mathrm{s}_{\beta_k}, \mathrm{c}_{\beta_k}, s_k, \mathrm{s}_\psi, \mathrm{c}_\psi, \mathrm{s}_\theta, \mathrm{c}_\theta, \mathrm{s}_\phi, \mathrm{c}_\phi, z, y, x) \quad (4.42)$$

Note that $g_{1,k}$ and $g_{2,k}$ contain square roots; the analyst must determine whether the positive or negative branches correspond to the desired configuration. In this case, the

two solutions for s_k correspond to the configuration shown in Figure 4.12, where the lower and upper leg segments are directed towards each other, and a non-physical configuration in which the leg segments are directed away from each other; the two solutions for c_{β_k} correspond to the two possible universal joint solutions that result in the same mechanism configuration.

Although pre-generating Gröbner bases can be computationally expensive, the resulting simulation code outperforms existing iterative and constraint stabilization techniques. In particular, we compare the computation time required to perform the same dynamic simulation using five solution approaches. The first three approaches involve integrating the projected dynamic equations obtained using the embedding technique, described in Section 2.1.5, and solving the kinematics using the following methods:

1. Recursively solving k triangular systems—one for each leg—extracted from the Gröbner bases described above.

2. Solving for all dependent coordinates in a single Newton–Raphson iterative procedure.

3. Solving for the dependent coordinates associated with each leg in a separate Newton–Raphson iterative procedure.

The dynamic solution flow for the Gröbner basis approach is shown in Figure 4.13. Note that the Gröbner basis approach provides exact solutions for the dependent coordinates; tolerances of 10^{-3} and 10^{-6} are used for the iterative approaches. The performance of the Baumgarte stabilization and penalty formulation techniques, described in Section 2.1.3, are also evaluated. We use Baumgarte stabilization parameters $\alpha_B = 1$ and $\beta_B = 18$, which result in a peak constraint violation of 9.5×10^{-6}, the minimum that was found for $\{\alpha_B, \beta_B \in \mathbb{N} : 1 \leq \alpha_B, \beta_B \leq 20\}$. In the penalty formulation approach, we use one iteration with $\boldsymbol{\rho} = 10^3\,\mathbf{I}$, $\boldsymbol{\zeta} = \mathbf{I}$, and $\boldsymbol{\omega} = 18\,\mathbf{I}$, which results in a peak constraint violation of 1.3×10^{-5}. Gaussian elimination with partial pivoting is used to solve the linear systems of equations in the Newton–Raphson approaches to maintain numerical stability, and in the two constraint stabilization approaches due to impractically large symbolic solutions.

The expressions for all solution approaches are generated and simplified symbolically in Maple. Optimized simulation code is then generated using the `dsolve/numeric/optimize`

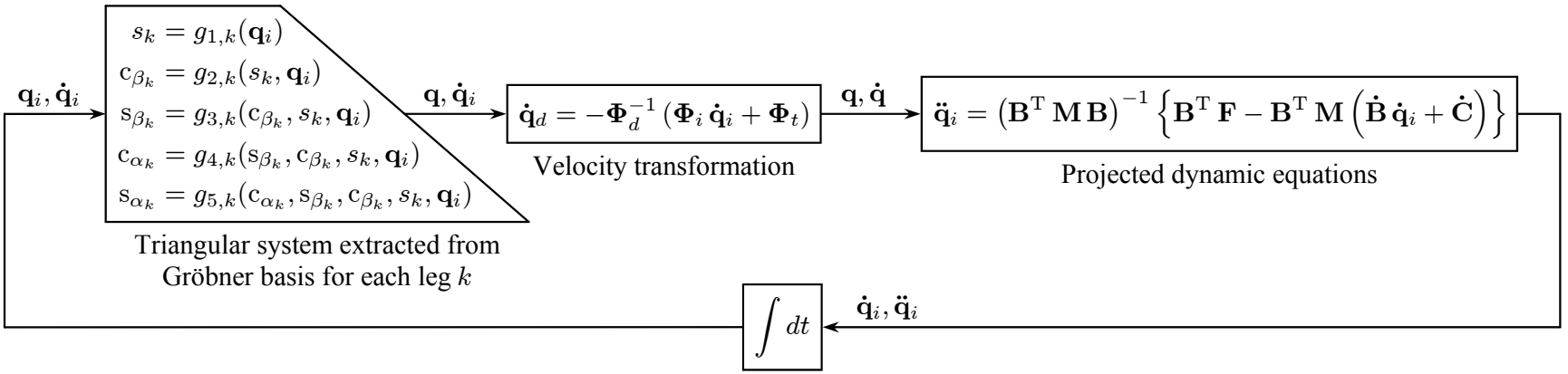


Figure 4.13: Solution flow for dynamic simulation of Gough–Stewart platform using Gröbner basis approach

Table 4.10: Performance of solution approaches for dynamic simulation of Gough–Stewart platform

Compiler	Simulation	Gröbner basis	Separate Newton procedures		Single Newton procedure		Baumgarte stabilization	Penalty formulation
		· exact	· tol. 10^{-3}	· tol. 10^{-6}	· tol. 10^{-3}	· tol. 10^{-6}		
Lcc	Kinematic	6.8 ms/s	11.9 ms/s	21.3 ms/s	91.7 ms/s	181.9 ms/s	79.2 ms/s	164.7 ms/s
	Dynamic	19.4 ms/s	24.8 ms/s	34.0 ms/s	104.7 ms/s	193.8 ms/s		
MS	Kinematic	2.7 ms/s	5.6 ms/s	9.2 ms/s	37.7 ms/s	73.8 ms/s	43.3 ms/s	85.7 ms/s
	Dynamic	10.3 ms/s	13.3 ms/s	16.6 ms/s	45.3 ms/s	80.6 ms/s		

routine (which generally outperforms `codegen/optimize`) and exported to the C programming language. The resulting simulation code is compiled using the default Lcc compiler in MATLAB R14 (Lcc) as well as the Microsoft Visual C/C++ compiler (MS). In each case, a 3-second dynamic simulation is performed on a single 3.00-GHz Intel Pentium 4 processor using 1-millisecond time steps and a first-order explicit Euler integration scheme. Low-order, fixed-step-size, non-stiff solvers, such as the explicit Euler integrator, are often used in real-time and hardware-in-the-loop applications [3]. The average computation time required for each simulated second is shown in Table 4.10. Where applicable, the amount of time required to process the kinematic portion of each dynamic simulation is also reported. The Gröbner basis approach clearly outperforms the other techniques, providing dynamic simulation code that satisfies the constraints exactly and executes nearly 100 times faster than real time—over 20% faster than the most efficient iterative approach. Note that a substantial improvement can be obtained using Newton–Raphson iteration simply by calculating the dependent coordinates for each leg separately. Also note that the penalty formulation is significantly slower than Baumgarte stabilization, as the former involves solving a linear system of size $n = 24$ at each time step, while the size of the linear system being solved in the latter case is $m = 18$. Finally, while parallel processing has not been employed here, the calculation of $\mathbf{q}_d$ could be distributed over six processors in the Gröbner basis approach, if so desired.

4.3 Chapter summary

The focus of this chapter was the analysis of mechanisms whose kinematic equations can be fully triangularized. As has been shown, the recursively solvable nature of fully triangular solutions results in highly efficient simulation code, and outperforms many existing solution techniques. Although it also produces triangular systems, the characteristic pair of joints approach demands the use of joint coordinates. The Gröbner basis approach, on the other hand, can accommodate the use of any desired generalized coordinates, which was shown to be particularly useful in the analysis of the planar parallel robot and Gough–Stewart platform. Furthermore, since a Gröbner basis can be generated relative to any desired elimination order, it is capable of identifying more efficient triangular systems than

the characteristic pair of joints technique. Although additional computational effort is required at the formulation stage, the triangular systems extracted from Gröbner bases provide exact results in a fixed amount of time, which may be preferred over the simpler, but generally slower, Newton–Raphson and constraint stabilization approaches.

The examples presented in this chapter demonstrate the applicability of the Gröbner basis approach to the real-time simulation of real-world multibody systems, including machines, deployable structures, and parallel robots. In particular, the following closed-kinematic-chain mechanisms were examined:

- Single-loop: planar slider-crank and spatial four-bar mechanisms.
- Cascading-loop: hydraulic excavator and synthetic aperture radar satellite antenna.
- Six-bar: Stephenson-III and aircraft landing gear mechanisms.
- Parallel: planar parallel robot and Gough–Stewart platform.

As shown in the last two examples, choosing as independent coordinates the position and orientation of a full-mobility end-effector leads to a fully triangular solution. Such a selection of independent coordinates requires the solution of the inverse kinematics problem rather than the direct kinematics problem which, for constrained mechanisms, is the more challenging of the two. As will be shown in the next chapter, the situation becomes somewhat more complicated—though not hopeless—when the end-effector has limited mobility.

Chapter 5

Block-triangular Systems

In the previous chapter, we studied several mechanisms whose kinematic equations can be fully triangularized, and obtained highly efficient simulation code that outperforms many existing solution techniques. Choosing as independent coordinates the position and orientation of a full-mobility end-effector was found to be an effective strategy for parallel robots. In this chapter, two popular vehicle suspension systems are studied: the five-link and the double-wishbone. Although the wheel carrier has limited mobility in both cases—one degree-of-freedom once the steering input has been specified—we use a strategy similar to that employed in the analysis of the Gough–Stewart platform. As will be demonstrated, the resulting block-triangular solutions are still more efficient than using existing solution techniques. The efficient kinematic solution developed in Section 5.2 is used for the real-time simulation of a vehicle with double-wishbone suspensions on both axles, which is discussed in Chapter 6.

5.1 Five-link suspension

We first consider the 1-DOF five-link suspension system, a schematic of which is shown in Figure 5.1. The relatively large number of design parameters enables the five-link suspension to meet complex kinematic and dynamic performance requirements [118]. Each of the five rigid links is connected to the ground (chassis) with a universal joint, and to the end-effector (wheel carrier) with a spherical joint. The configuration of universal joint k is specified by its angles of rotation about the global X-axis (α_k) and the rotated Y′-

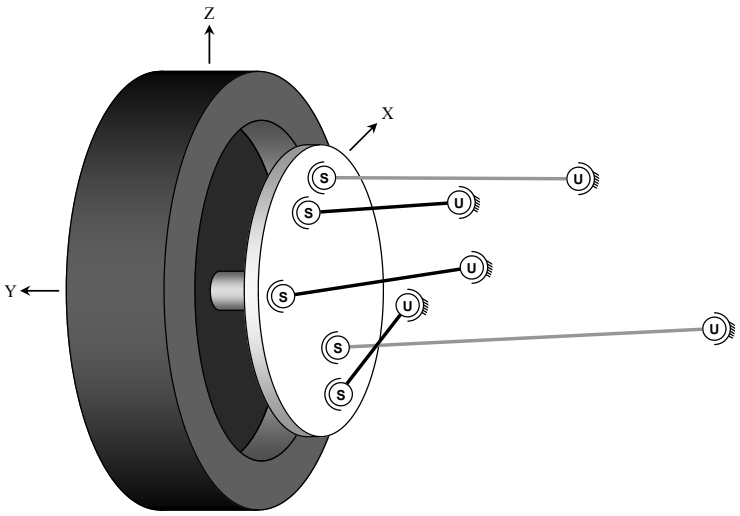


Figure 5.1: Five-link suspension

axis (β_k); each local X″-axis is coincident with its respective link. Generalized coordinates $\{x, y, z\}$ represent the position of the center of mass of the wheel carrier in the global reference frame, and $\{\phi, \theta, \psi\}$ represent the corresponding 3-2-1 Euler angles. This system is topologically similar to the Gough–Stewart platform presented in Section 4.2.5, which motivates the strategy for solving its kinematics. If placed on the front axle of a vehicle, a steering rack could be included in the model to allow the wheel to steer. Since the rack displacement would be an input to such a model, neither the degrees-of-freedom nor the kinematic solution strategy presented below would change. All geometric parameters are given in Appendix A.4 and describe a suspension used on the rear axle of Mercedes–Benz vehicles in the 1980s. The longitudinal displacement (x), lateral displacement (y), and toe (δ), camber (γ), and caster (τ) angles of the wheel carrier are shown as functions of its vertical displacement (z) in Figure 5.2. These results were obtained simply by iterating over the constraint equations described below, and agree with the results reported by Knapczyk [70]. Since we are primarily interested in the kinematics of this mechanism, the spring-damper and tire components are not included in the model. Dynamic parameters are obtained by assuming the links are cylindrical with radii 0.015 [m], the wheel carrier is cylindrical with radius 0.15 [m] and thickness 0.02 [m], and all components are of density 8.0×10^3 $\left[\text{kg/m}^3\right]$.

Many existing strategies for simulating multi-link suspensions employ techniques that modify the system model. Elmqvist et al. [28] discuss four common approaches:

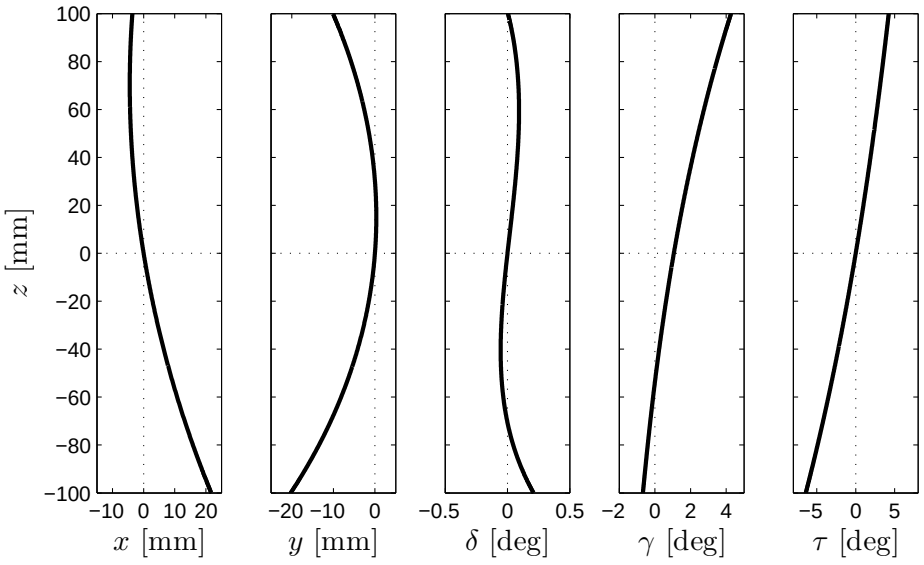


Figure 5.2: Kinematic response of five-link suspension

1. Replace the suspension with a single vertically-oriented prismatic joint, and use polynomials to describe the camber and toe angles as functions of wheel vertical displacement.

2. Replace the multi-link suspension with a suspension of similar performance for which a kinematic solution can be more easily obtained.

3. Neglect the mass and inertia of small suspension components.

4. Replace all ideal joints with flexible bushings.

While an ideal-joint, rigid-link model is only an approximation of the actual physical system, the first three of these approaches deviate even further from reality. The fourth approach can provide more realistic simulations of passenger vehicles, whose suspensions generally contain bushings, but involves solving a system of stiff ODEs and, therefore, demands the use of an implicit integrator. Since low-order, fixed-step-size, non-stiff ODE solvers are typically used in real-time applications [3], the use of bushings can present difficulties in such contexts. A general complication with these techniques is the need to maintain two separate models: a detailed model with stiff terms and constraints for accurate off-line simulations, and a simplified real-time-capable model. When a closed-kinematic-chain model is used, kinematic analyses are generally performed using iterative

techniques, such as those based on interval analysis [94], trust region methods [69], or Newton–Raphson iteration.

Following the Gough–Stewart platform example, we use absolute coordinates to describe the motion of the wheel carrier and joint coordinates elsewhere—that is, $\mathbf{q} = \{x, y, z, \phi, \theta, \psi, \alpha_{1\ldots5}, \beta_{1\ldots5}\}$. In this case, $m = n - f = 15$ constraint equations are obtained, three for each link k:

$$\begin{Bmatrix} x + L_k \mathrm{c}_{\beta_k} - C_{x,k} \\ y + L_k \mathrm{s}_{\alpha_k} \mathrm{s}_{\beta_k} - C_{y,k} \\ z - L_k \mathrm{c}_{\alpha_k} \mathrm{s}_{\beta_k} - C_{z,k} \end{Bmatrix} + \begin{bmatrix} \mathrm{c}_\phi \mathrm{c}_\theta & \mathrm{c}_\phi \mathrm{s}_\theta \mathrm{s}_\psi - \mathrm{s}_\phi \mathrm{c}_\psi & \mathrm{c}_\phi \mathrm{s}_\theta \mathrm{c}_\psi + \mathrm{s}_\phi \mathrm{s}_\psi \\ \mathrm{s}_\phi \mathrm{c}_\theta & \mathrm{s}_\phi \mathrm{s}_\theta \mathrm{s}_\psi + \mathrm{c}_\phi \mathrm{c}_\psi & \mathrm{s}_\phi \mathrm{s}_\theta \mathrm{c}_\psi - \mathrm{c}_\phi \mathrm{s}_\psi \\ -\mathrm{s}_\theta & \mathrm{c}_\theta \mathrm{s}_\psi & \mathrm{c}_\theta \mathrm{c}_\psi \end{bmatrix} \begin{Bmatrix} W_{x,k} \\ W_{y,k} \\ W_{z,k} \end{Bmatrix} = \mathbf{0} \quad (5.1)$$

where L_k is the length of the k^{th} link, $C_k = (C_{x,k}, C_{y,k}, C_{z,k})$ is the position of the corresponding universal joint on the chassis, and W_k is the position of the k^{th} spherical joint on the wheel carrier. Note the similarity between (5.1) and the position constraints obtained for the Gough–Stewart platform (4.32). A fundamental difference between these models, however, is the number of degrees-of-freedom they possess or, equivalently, the number of independent generalized coordinates we may choose. Since the end-effector of the Gough–Stewart platform has full mobility, we were able to select six independent coordinates, thereby obtaining a system of three constraint equations involving three dependent coordinates for each kinematic loop. For the 1-DOF five-link suspension system, we select $\mathbf{q}_i = \{z\}$ since each value of z corresponds to a unique value of x, y, ϕ, θ, and ψ. The embedding technique is applied to obtain one second-order ODE for $\ddot{z}$, which is expressed in first-order form. We integrate forward in time to determine z at each time step, then solve a system of five equations to determine $\mathbf{q}_{d1} = \{x, y, \phi, \theta, \psi\}$. Finally, $\mathbf{q}_{d2} = \{\alpha_{1\ldots5}, \beta_{1\ldots5}\}$ is computed recursively using the equations extracted from the Gröbner bases described below. The five equations used for calculating $\mathbf{q}_{d1}$ are obtained by isolating the terms involving α_k and β_k on the left-hand side of (5.1) and squaring; the sum of these equations is a constant-distance constraint [66] of the following form:

$$L_k^2 = f_k(x, y, z, \phi, \theta, \psi), \qquad k = 1, \ldots, 5 \quad (5.2)$$

Since the complexity of the constant-distance constraints (5.2) precludes the generation of a Gröbner basis, Newton–Raphson iteration is used to calculate $\mathbf{q}_{d1}$ given $\mathbf{q}_i$. Nevertheless, the triangularization of $\mathbf{q}_{d2}$ results in a significant improvement in simulation time.

We triangularize the three constraint equations associated with each loop separately. The `convert/rational` function in Maple is used to convert the floating-point coefficients in (5.1) into rational numbers. In this case, an exact conversion of coefficients L_k would result in rational numbers with many digits, which is impractical for Gröbner basis generation. Thus, the minimum number of digits is used in each rational approximation such that the error associated with the conversion is less than 10^{-6}. Since all coefficients are geometric parameters, this approach represents a measurement error of less than 1 [µm]. The trigonometric functions are eliminated using the same transformations as before. Note that auxiliary equations $\mathrm{s}_\vartheta^2 + \mathrm{c}_\vartheta^2 - 1 = 0$ need not be introduced for the known quantities ϕ, θ, and ψ. We use the following pure lexicographic term ordering for loop k:

$$\mathrm{s}_{\alpha_k} \succ \mathrm{c}_{\alpha_k} \succ \mathrm{s}_{\beta_k} \succ \mathrm{c}_{\beta_k} \succ z \succ y \succ x \succ \mathrm{s}_\psi \succ \mathrm{c}_\psi \succ \mathrm{s}_\theta \succ \mathrm{c}_\theta \succ \mathrm{s}_\phi \succ \mathrm{c}_\phi \tag{5.3}$$

We again note that any term ordering of the form $\mathbf{q}_d \succ \mathbf{q}_i$ can be used. Each Gröbner basis is generated on one core of a 3.00-GHz Intel Xeon E5472 processor in about 45 minutes using the 64-bit version of Maple 14. A recursively solvable system of the following form is extracted from the k^{th} Gröbner basis:

$$\mathrm{c}_{\beta_k} = g_{1,k}(z, y, x, \mathrm{s}_\psi, \mathrm{c}_\psi, \mathrm{s}_\theta, \mathrm{c}_\theta, \mathrm{s}_\phi, \mathrm{c}_\phi) \tag{5.4}$$

$$\mathrm{s}_{\beta_k} = g_{2,k}(\mathrm{c}_{\beta_k}, z, y, x, \mathrm{s}_\psi, \mathrm{c}_\psi, \mathrm{s}_\theta, \mathrm{c}_\theta, \mathrm{s}_\phi, \mathrm{c}_\phi) \tag{5.5}$$

$$\mathrm{c}_{\alpha_k} = g_{3,k}(\mathrm{s}_{\beta_k}, \mathrm{c}_{\beta_k}, z, y, x, \mathrm{s}_\psi, \mathrm{c}_\psi, \mathrm{s}_\theta, \mathrm{c}_\theta, \mathrm{s}_\phi, \mathrm{c}_\phi) \tag{5.6}$$

$$\mathrm{s}_{\alpha_k} = g_{4,k}(\mathrm{c}_{\alpha_k}, \mathrm{s}_{\beta_k}, \mathrm{c}_{\beta_k}, z, y, x, \mathrm{s}_\psi, \mathrm{c}_\psi, \mathrm{s}_\theta, \mathrm{c}_\theta, \mathrm{s}_\phi, \mathrm{c}_\phi) \tag{5.7}$$

In this case, $g_{2,k}$ contains a square root, which corresponds to the two possible universal joint solutions that result in the same mechanism configuration. As shown in Table 5.1 for $k = 1$, the coefficients in the Gröbner basis polynomials are of a reasonable length, despite enforcing a rational approximation error of less than 10^{-6}.

We compare the computational efficiency of six solution approaches: one involving the use of Gröbner bases, three purely iterative approaches, and two constraint stabilization techniques. The initial configuration of the mechanism is shown in Figure 5.1, where $z = 0$. Kinematic simulations are performed by driving the vertical displacement of the wheel carrier through the sinusoidal trajectory shown in Figure 5.3(a); forward dynamic simulations are driven by applying to the wheel carrier the following C^0-continuous, time-

Table 5.1: Triangular system obtained for first link of five-link suspension

Polynomial	Degree	Number of terms	Longest coefficient
$g_{1,1}(\mathrm{c}_{\beta_1}, \mathbf{q}_i, \mathbf{q}_{d1})$	1 in c_{β_1}	8	15 digits
$g_{2,1}(\mathrm{s}_{\beta_1}, \mathrm{c}_{\beta_1}, \mathbf{q}_i, \mathbf{q}_{d1})$	2 in s_{β_1}	28	29 digits
$g_{3,1}(\mathrm{c}_{\alpha_1}, \mathrm{s}_{\beta_1}, \mathrm{c}_{\beta_1}, \mathbf{q}_i, \mathbf{q}_{d1})$	1 in c_{α_1}	6	15 digits
$g_{4,1}(\mathrm{s}_{\alpha_1}, \mathrm{c}_{\alpha_1}, \mathrm{s}_{\beta_1}, \mathrm{c}_{\beta_1}, \mathbf{q}_i, \mathbf{q}_{d1})$	1 in s_{α_1}	8	15 digits

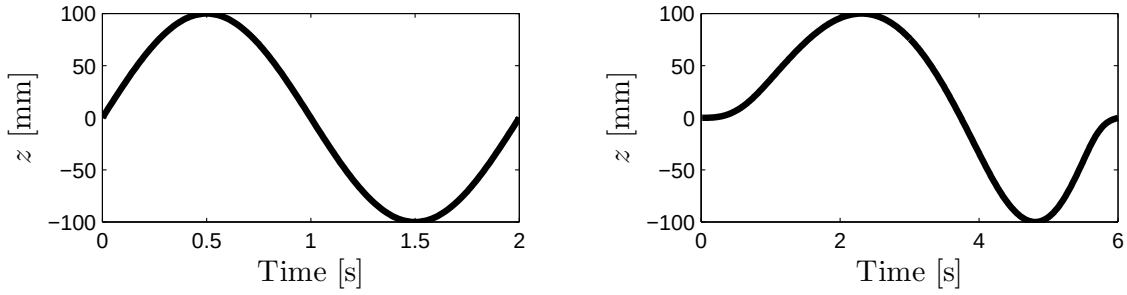
varying vertical force (in Newtons):

$$F_z(t) = 183.6 + \begin{cases} 1.84\sin(\pi t), & \text{if } t < 1 \\ -2.38\sin(\pi(t-1)/3), & \text{if } 1 \le t < 4 \\ 4.62\sin(\pi t/2), & \text{if } 4 \le t < 5 \\ -4.62\cos(\pi t), & \text{if } 5 \le t < 5.5 \\ 6.50\sin(2\pi t), & \text{if } t \ge 5.5 \end{cases} \tag{5.8}$$

which results in a similar trajectory, as shown in Figure 5.3(b). The first four solution approaches involve integrating the projected dynamic equations obtained using the embedding technique, described in Section 2.1.5, and solving the kinematics using the following methods:

1. Solving for all dependent coordinates in a single Newton-Raphson iterative procedure.
2. Solving for $\mathbf{q}_{d1}$ iteratively using (5.2), then solving for $\mathbf{q}_{d2}$ using one of three methods:
   (a) Recursively solving triangular systems extracted from the Gröbner bases described above;
   (b) Solving for the dependent coordinates associated with each loop in a separate iterative procedure; or
   (c) Solving for all remaining dependent coordinates in a single iterative procedure.

The dynamic solution flow for the Gröbner basis approach is shown in Figure 5.4. The performance of the Baumgarte stabilization and penalty formulation approaches are also evaluated. Baumgarte stabilization parameters $\alpha_B = 7$ and $\beta_B = 17$ are used, which



(a) Kinematic simulation

(b) Dynamic simulation

Figure 5.3: Vertical displacement of wheel carrier in five-link simulations

result in a peak constraint violation of 3.1×10^{-6}, the minimum that was found for $\{\alpha_B, \beta_B \in \mathbb{N} : 1 \leq \alpha_B, \beta_B \leq 20\}$. In the penalty formulation approach, we use one iteration with $\boldsymbol{\rho} = 10^3\,\mathbf{I}$, $\boldsymbol{\zeta} = 7\,\mathbf{I}$, and $\boldsymbol{\omega} = 17\,\mathbf{I}$, which results in a peak constraint violation of 8.0×10^{-5}. Tolerances of 10^{-6} are used for the iterative approaches, thereby providing a similar level of precision as the constraint stabilization techniques. Gaussian elimination with partial pivoting is used to solve the linear systems of equations in all iterative procedures to maintain numerical stability, and in the constraint stabilization approaches due to impractically large symbolic solutions.

The expressions for all solution approaches are generated and simplified symbolically in Maple, and optimized simulation code is obtained using the `dsolve/numeric/optimize` routine. Since the Microsoft Visual C/C++ compiler consistently outperforms the default Lcc compiler in MATLAB R14, the simulation code is only compiled with the former. All simulations are performed on a single 3.00-GHz Intel Pentium 4 processor using 1-millisecond time steps. The dynamic simulations use a first-order explicit Euler integration scheme, which is often used in real-time and hardware-in-the-loop applications [3]. The average computation time required for each simulated second is shown in Table 5.2 for both kinematic and dynamic simulations. The kinematic simulation times provide an approximate measure of the amount of time required to process the kinematic portion of each dynamic simulation. Once again, we find that the Gröbner basis approach outperforms the other techniques, providing dynamic simulation code that executes about 65 times faster than real time, and over 15% faster than the most efficient iterative approach. Note that a substantial improvement can be obtained using Newton–Raphson iteration simply by calculating the dependent coordinates for each independent loop separately. In this

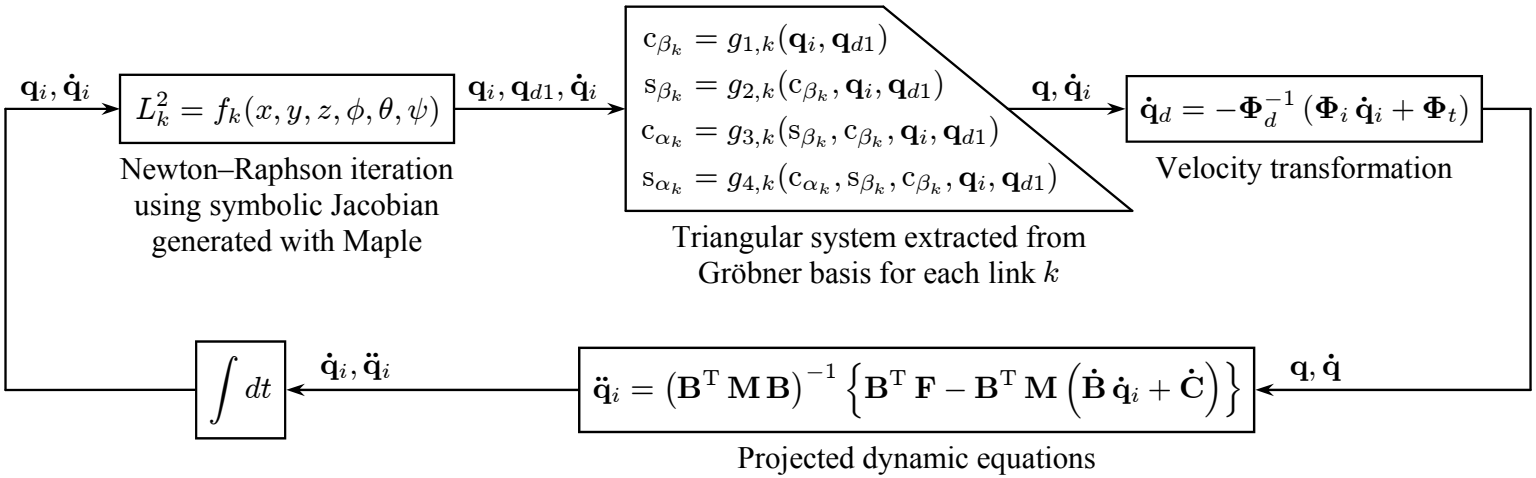


Figure 5.4: Solution flow for dynamic simulation of five-link suspension using Gröbner basis approach

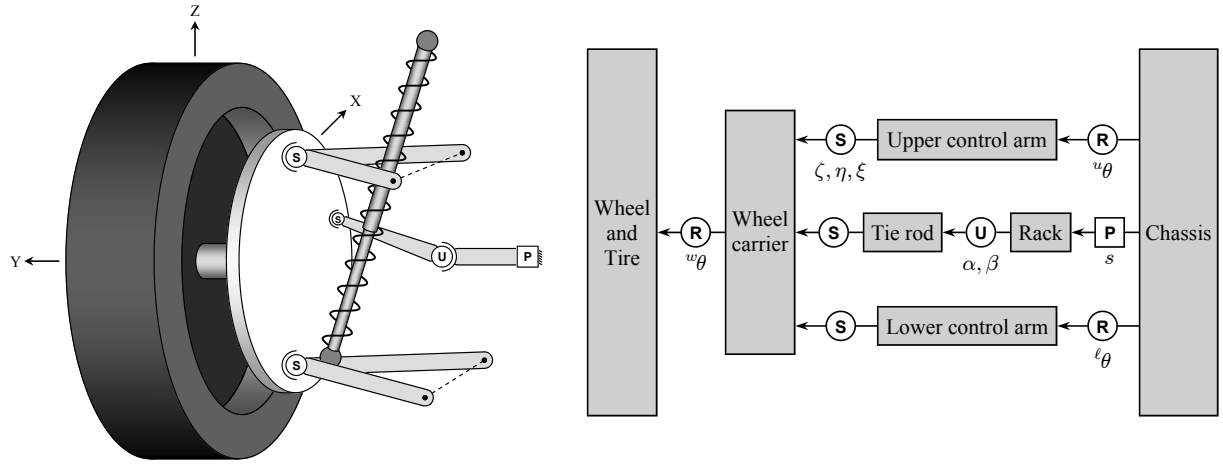
Table 5.2: Performance of solution approaches for kinematic and dynamic simulations of five-link suspension

Compiler	Simulation	Gröbner basis	Separate iterations for each loop in $\mathbf{q}_{d2}$	Single iteration for solving $\mathbf{q}_{d2}$	Single iteration for solving $\mathbf{q}_d$	Baumgarte stabilization	Penalty formulation
MS	Kinematic	5.5 ms/s	9.9 ms/s	28.1 ms/s	47.8 ms/s	28.5 ms/s	29.9 ms/s
	Dynamic	15.5 ms/s	18.3 ms/s	32.6 ms/s	55.7 ms/s		

case, the penalty formulation is only marginally slower than Baumgarte stabilization, since the former involves solving a linear system of size $n = 16$ at each time step, while the size of the linear system in the latter case is $m = 15$. We again note that, while parallel processing has not been employed here, the calculation of $\mathbf{q}_{d2}$ could be distributed over five processors.

5.2 Double-wishbone suspension

The final system we consider is the double-wishbone suspension, a schematic of which is shown in Figure 5.5(a); the corresponding topological graph is shown in Figure 5.5(b). The ability to adjust many aspects of its kinematics makes the double-wishbone suspension popular on high-performance vehicles [97]; its load-handling capabilities also make it suitable for use on the front axle of medium- and heavy-duty vehicles [21]. The upper and lower control arms are connected to the wheel carrier with spherical joints (S), and to the chassis with revolute joints (R). One end of the tie rod is connected to the wheel carrier with a spherical joint; a universal joint (U) at the other end connects it to either the rack (on the front axle) or the chassis (on the rear axle). All geometric suspension parameters are provided in Appendix A.5. The system is modelled using joint coordinates $\mathbf{q} = \left\{{}^{w}\theta, \zeta, \eta, \xi, {}^{u}\theta, {}^{\ell}\theta, \alpha, \beta, s\right\}$, as labelled in Figure 5.5(b). The configuration of the universal joint is specified by its angles of rotation about the global Z-axis (α) and the rotated



(a) Schematic (b) Topological graph

Figure 5.5: Double-wishbone suspension (front-left corner)

X′-axis (β); $\{\zeta, \eta, \xi\}$ represent the 3-2-1 Euler angles associated with the spherical joint between the upper control arm and the wheel carrier. Neglecting the motion of the chassis, this system has three degrees-of-freedom: the rotation of the wheel (${}^{w}\theta$), the displacement of the rack (s), and the vertical displacement of the wheel carrier (z). Note that rack displacement s is an input to the model, which reduces the number of degrees-of-freedom to two.

As in the previous example, the simulation of double-wishbone suspensions has been approached in a number of ways, all of which have been discussed in Section 2.1. Perhaps the simplest method is to satisfy the constraint equations using Newton–Raphson iteration [55], which is not generally suitable for real-time applications. Constraint stabilization techniques [101] and compliant joint models [21] have also been used, and require the tuning of system-specific parameters. Another familiar approach involves replacing the double-wishbone with a suspension of similar performance that has an open-loop topology [62], but this strategy provides only an approximation of the true suspension kinematics; the use of massless links [119] amounts to approximating the dynamic performance. The approach adopted herein is, again, to triangularize the kinematic constraint equations; however, the use of Gröbner bases in this case is beyond the capabilities of the computational resources available for this research. As shown in Table 5.3, the constraint equations for the double-wishbone suspension are significantly more complex than those obtained for the five-link which, themselves, required nearly 4 hours of computation time. Furthermore, unlike the

Table 5.3: Complexity of constraint equations for five-link and double-wishbone

Suspension	Loop	Maximum degree	Number of indeterminates	Maximum number of terms
Five-link	$L_{1\ldots5}$	3	13	8
Double-wishbone	L_1	4	10	26
(joint coordinates)	L_2	4	13	29
Double-wishbone	L_1	4	20	22
(absolute on carrier,	L_2	4	20	22
joint elsewhere)	L_3	5	23	35

previous example, the double-wishbone suspension is fixed to a moving chassis rather than the ground; thus, describing the motion of the wheel carrier using absolute coordinates introduces the coordinates of the chassis as well. We, therefore, triangularize the constraints manually using Maple, as described below. Although labour-intensive, this approach results in an entirely parametric solution since, unlike a Gröbner basis solution, there is no requirement to use purely numeric coefficients.

We proceed with a purely joint coordinate formulation to minimize the number of modelling coordinates. The Multibody library in MapleSim is used to generate the following six constraint equations, three for each independent kinematic loop:

$$\phi_{1,2,3}\big(\zeta, \eta, \xi, {}^{u}\theta, {}^{\ell}\theta\big) = 0 \qquad \phi_{4,5,6}\big(\zeta, \eta, \xi, {}^{u}\theta, \alpha, \beta, s\big) = 0 \tag{5.9}$$

Drawing from the results of the preceding examples, we employ the following strategy:

1. Assign one of $\{\zeta, \eta, \xi\}$ to the independent coordinate vector $\mathbf{q}_i$.
2. Compute the two remaining spherical joint angles iteratively using two of the six constraint equations (5.9).
3. Compute the four remaining generalized coordinates $\big\{{}^{u}\theta, {}^{\ell}\theta, \alpha, \beta\big\}$ recursively by triangularizing the remaining constraint equations.

We choose as independent coordinates $\mathbf{q}_i = \{{}^{w}\theta, \xi\}$—the former, out of necessity; the latter, for its one-to-one correspondence with the vertical displacement of the wheel carrier (z), as shown in Figure 5.6. Note that η cannot be used to uniquely determine the configuration of the wheel carrier; the use of ζ may present difficulties around $z = -100$ [mm], since a small error in the value of ζ could result in a large error in the vertical displacement of the wheel carrier.

The first objective is to derive a triangular system to solve for $\mathbf{q}_{d1} = \big\{{}^{u}\theta, {}^{\ell}\theta, \alpha, \beta\big\}$ recursively, given values of $\{\xi, \zeta, \eta\}$. We begin by expressing ϕ_1 in the following form:

$$A_1 \sin\big({}^{\ell}\theta\big) + B_1 \cos\big({}^{\ell}\theta\big) = C_1 \tag{5.10}$$

where coefficients A_1, B_1, and C_1 are functions of $\{\xi, \zeta, \eta, {}^{u}\theta\}$. The following solutions can

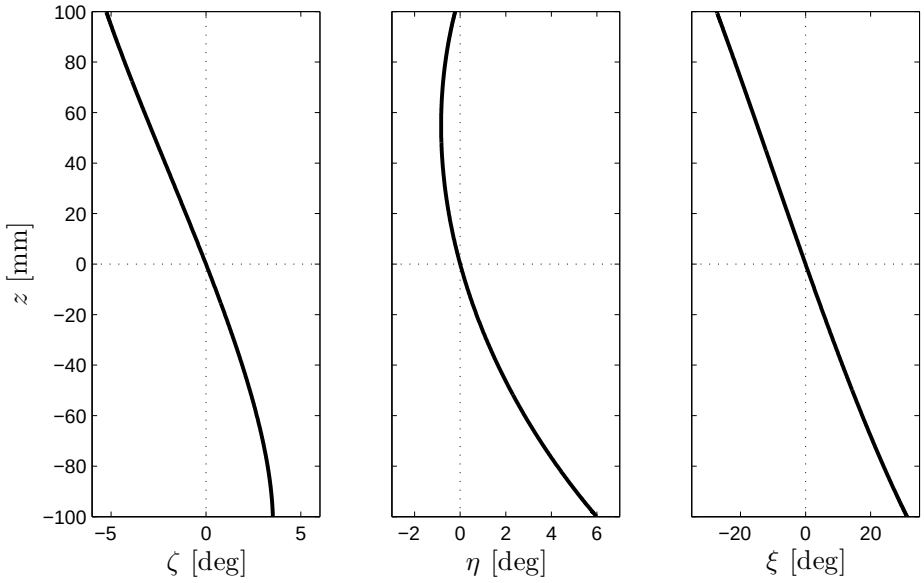


Figure 5.6: Kinematic response of double-wishbone suspension

then be obtained for $\sin\left({}^{\ell}\theta\right)$ and $\cos\left({}^{\ell}\theta\right)$:

$$\sin\left({}^{\ell}\theta\right) = \frac{A_1 C_1 \pm B_1\sqrt{A_1^2 + B_1^2 - C_1^2}}{A_1^2 + B_1^2} \tag{5.11}$$

$$\cos\left({}^{\ell}\theta\right) = \frac{B_1 C_1 \mp A_1\sqrt{A_1^2 + B_1^2 - C_1^2}}{A_1^2 + B_1^2} \tag{5.12}$$

Note that these solutions are functions of $\{\xi, \zeta, \eta, {}^{u}\theta\}$, only the first three of which are assumed to be known at this stage of the solution. We obtain an expression for ${}^{u}\theta$ as a function of $\{\xi, \zeta, \eta\}$ by substituting (5.11) and (5.12) into ϕ_2 which, once simplified, can be expressed in the following form:

$$A_2 \sin\left({}^{u}\theta\right) + B_2 \cos\left({}^{u}\theta\right) + D_2 \sin\left({}^{u}\theta\right)\cos\left({}^{u}\theta\right) + E_2 \cos^2\left({}^{u}\theta\right) = C_2 \tag{5.13}$$

where all coefficients are functions of $\{\xi, \zeta, \eta\}$. Note that the second-order trigonometric terms appear upon eliminating the square roots introduced by (5.11) and (5.12). Although (5.13) can be solved for $\sin\left({}^{u}\theta\right)$ and $\cos\left({}^{u}\theta\right)$, the solution is quartic and would require iteration to evaluate numerically. Instead, we note that the coefficients in D_2 and E_2 are significantly smaller than those appearing elsewhere, so expect these terms to have negligible effect on the numerical solution. Thus, we use the approximations $D_2 \approx 0$ and $E_2 \approx 0$ to obtain an equation analogous to (5.10), which results in solutions for ${}^{u}\theta$ of the form shown in (5.11) and (5.12). Evaluating numerically over the full range of

motion of the wheel carrier, we find that this approximation introduces an error of less than 5.7×10^{-4} [rad] (0.1%) in the solution for ${}^{u}\theta$, which is deemed to be an acceptable compromise given our real-time intentions.

A similar strategy can be employed to triangularize ϕ_4 and ϕ_5. In particular, ϕ_4 can be expressed as follows:

$$A_4 \sin(\beta) + B_4 \cos(\beta) = C_4 \tag{5.14}$$

where coefficients A_4, B_4, and C_4 are functions of $\{\xi, \zeta, \eta, {}^{u}\theta, \alpha\}$. Solutions for $\sin(\beta)$ and $\cos(\beta)$ can then be obtained as before. Upon substitution of these solutions into ϕ_5, we obtain an equation of the following form:

$$A_5 \sin(\alpha) + B_5 \cos(\alpha) + D_5 \sin(\alpha)\cos(\alpha) + E_5 \cos^2(\alpha) = C_5 \tag{5.15}$$

where all coefficients are functions of $\{\xi, \zeta, \eta, {}^{u}\theta\}$. In this case, the coefficients appearing in D_5 and E_5 are not of negligible magnitude, so cannot be eliminated outright; however, we note that α remains small over the full range of motion of the wheel carrier. Thus, we apply the approximation $\cos(\alpha) \approx 1$ to the third and fourth terms of (5.15) to arrive at the following form:

$$(A_5 + D_5) \sin(\alpha) + (B_5 + E_5) \cos(\alpha) = C_5 \tag{5.16}$$

whereupon solutions of the form shown in (5.11) and (5.12) are obtained. As will be verified below, the error introduced by this approximation—less than 4.5×10^{-4} [rad] (1.5%) in the solution for α—does not significantly degrade the quality of the results. In summary, we have generated the following triangular system to solve for $\mathbf{q}_{d1} = \left\{{}^{u}\theta, {}^{\ell}\theta, \alpha, \beta\right\}$ recursively, given values of $\{\xi, \zeta, \eta\}$:

$${}^{u}\theta = f_1\big(\xi, \zeta, \eta\big) \tag{5.17}$$

$${}^{\ell}\theta = f_2\big(\xi, \zeta, \eta, {}^{u}\theta\big) \tag{5.18}$$

$$\alpha = f_3\big(\xi, \zeta, \eta, {}^{u}\theta\big) \tag{5.19}$$

$$\beta = f_4\big(\xi, \zeta, \eta, {}^{u}\theta, \alpha\big) \tag{5.20}$$

The only remaining task at the position level is to construct two equations to solve for $\mathbf{q}_{d2} = \{\zeta, \eta\}$ given $\mathbf{q}_i = \{\xi\}$. Unfortunately, the double-wishbone constraint equations (5.9) do not yield a form analogous to that found for the five-link suspension (5.2).

Although substituting the above symbolic solutions into ϕ_3 and ϕ_6 would result in two equations in two unknowns, these expressions would be of considerable size and, therefore, time-consuming to evaluate numerically. As such, we adopt the kinematic solution flow shown in Figure 5.7, where the triangular system is evaluated each iteration. We obtain equation g_1 by solving ϕ_1 for $\sin\left({}^{\ell}\theta\right)$ and ϕ_2 for $\cos\left({}^{\ell}\theta\right)$, and substituting these expressions into ϕ_3. Similarly, g_2 is obtained by solving ϕ_4 for $\sin(\alpha)$ and ϕ_5 for $\cos(\alpha)$, and substituting the resulting expressions into ϕ_6. In so doing, we avoid the computation of $\mathrm{d}^{\ell}\theta/\,\mathrm{d}\zeta$, $\mathrm{d}^{\ell}\theta/\,\mathrm{d}\eta$, $\mathrm{d}\alpha/\,\mathrm{d}\zeta$, and $\mathrm{d}\alpha/\,\mathrm{d}\eta$. We have, thus, reduced the kinematic processing from solving six equations in six unknowns simultaneously, to solving two equations in two unknowns simultaneously, preceded by the recursive solution of four equations each iteration.

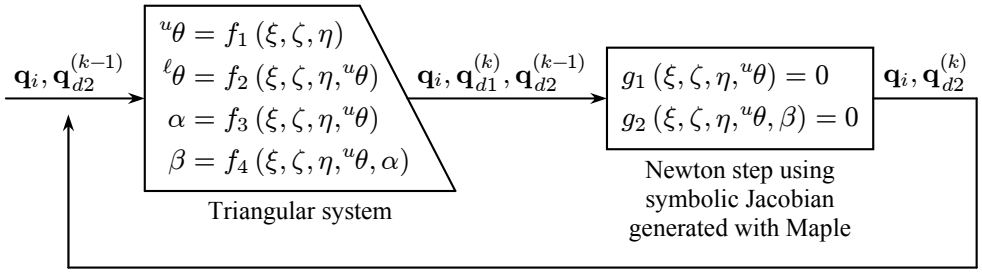


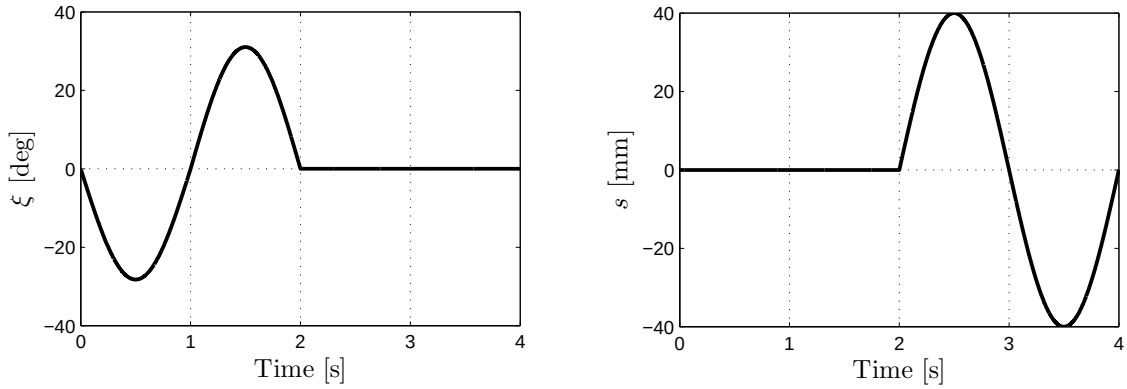
Figure 5.7: Kinematic solution flow for double-wishbone suspension (k^{th} iteration)

To conclude this example, kinematic simulations are performed by fixing the chassis to the ground and applying motion drivers to the independent coordinate ξ and rack displacement s, as shown in Figure 5.8. The input applied to spherical joint angle ξ drives the wheel carrier through its full range of vertical motion, as shown in Figure 5.9; the input applied to the rack corresponds to a steering wheel angle range of about $\pm 270°$, which is sufficient for maneuvers that evaluate the dynamic performance of this vehicle [61]. We compare the computational efficiency of three solution approaches:

1. Computing $\mathbf{q}_{d1}$ and $\mathbf{q}_{d2}$ using the block-triangular solution shown in Figure 5.7.

2. Computing $\mathbf{q}_d$ using Newton's method, iterating over the original constraint equations (5.9) and using Gaussian elimination with partial pivoting to invert the 6×6 Jacobian matrix numerically.

3. Computing $\mathbf{q}_d$ using Newton's method, iterating over the original constraint equations (5.9) and using a symbolic Jacobian inverse.



(a) Independent coordinate

(b) Rack displacement

Figure 5.8: Motion drivers for kinematic simulation of double-wishbone suspension

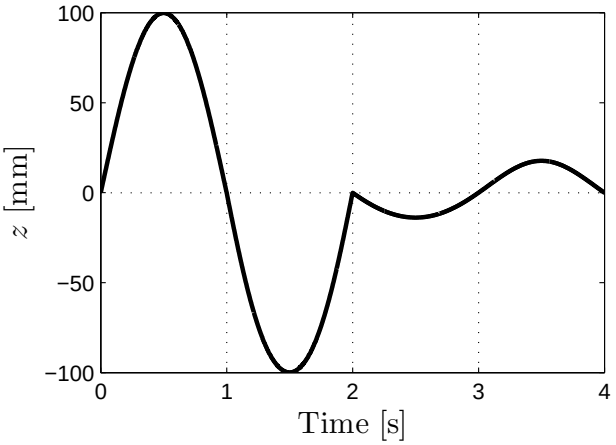


Figure 5.9: Vertical displacement of wheel carrier in double-wishbone simulations

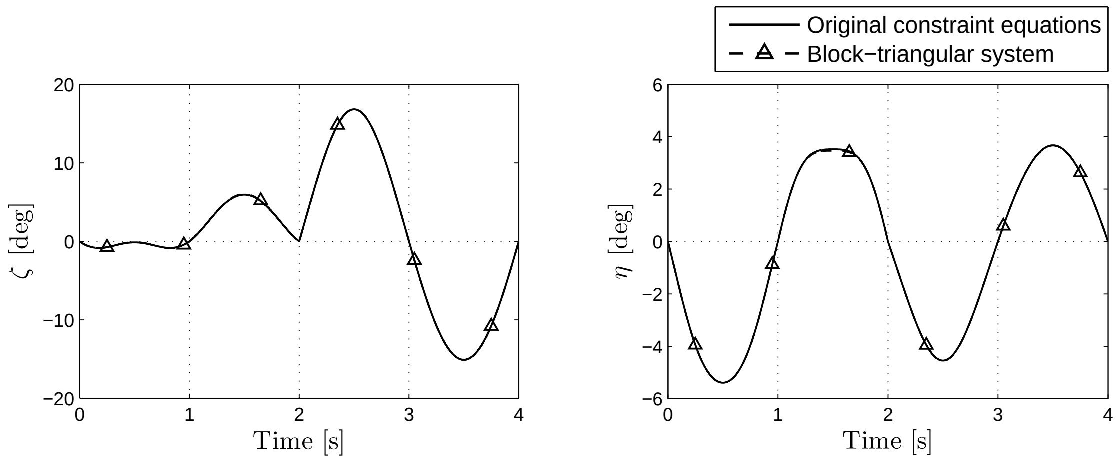
Iteration tolerances of 10^{-3} are used throughout. The expressions for all solution approaches are generated and simplified symbolically in Maple, and optimized simulation code is obtained using the `dsolve/numeric/optimize` routine. The resulting simulation code is compiled using the default Lcc compiler in MATLAB R14 (Lcc) as well as the Microsoft Visual C/C++ compiler (MS). In each case, a 4-second kinematic simulation is performed on a single 3.00-GHz Intel Pentium 4 processor using 1-millisecond time steps; the average computation time required for each simulated second is shown in Table 5.4. We again observe a computational advantage associated with triangularizing the kinematic constraints, and anticipate a 13% reduction in computation time when using the block-triangular solution in place of the fastest Newton–Raphson iterative approach. As will be shown in Section 6.2, this reduction in computation time can be significant in real-time applications. Approximating the equations used in the fully iterative approaches is also considered. In particular, we apply the approximations $\sin(\alpha) \approx \alpha$ and $\cos(\alpha) \approx 1$ to the original constraint equations (5.9), and eliminate the smallest terms from each equation until the error is comparable to that of the block-triangular solution. As shown in Table 5.5, the block-triangular approach remains computationally advantageous. Finally, we verify that the approximations used to obtain the block-triangular solution do not significantly degrade the quality of the results, as illustrated in Figure 5.10.

Table 5.4: Performance of double-wishbone suspension simulations

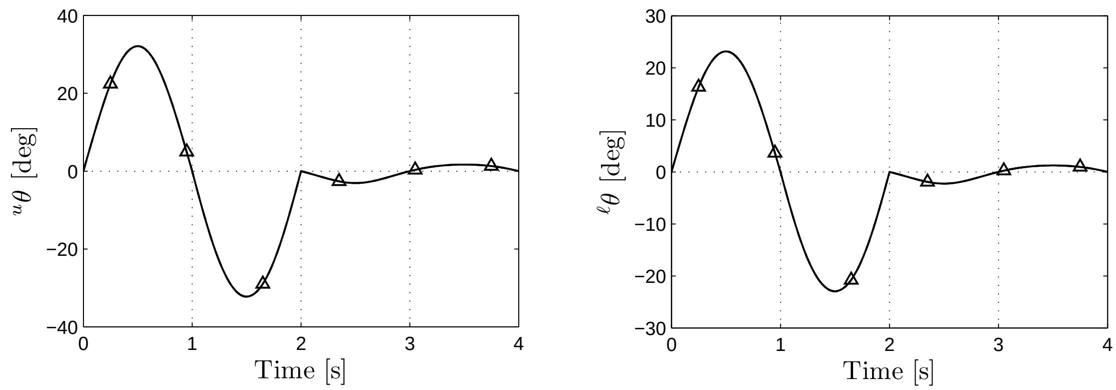
Compiler	Block-triangular solution	Iterating over original constraint equations (5.9)	
		using symbolic inverse	using numeric inverse
Lcc	16.6 ms/s	19.1 ms/s	23.7 ms/s
MS	13.8 ms/s	15.8 ms/s	17.3 ms/s

Table 5.5: Performance of approximated double-wishbone suspension simulations

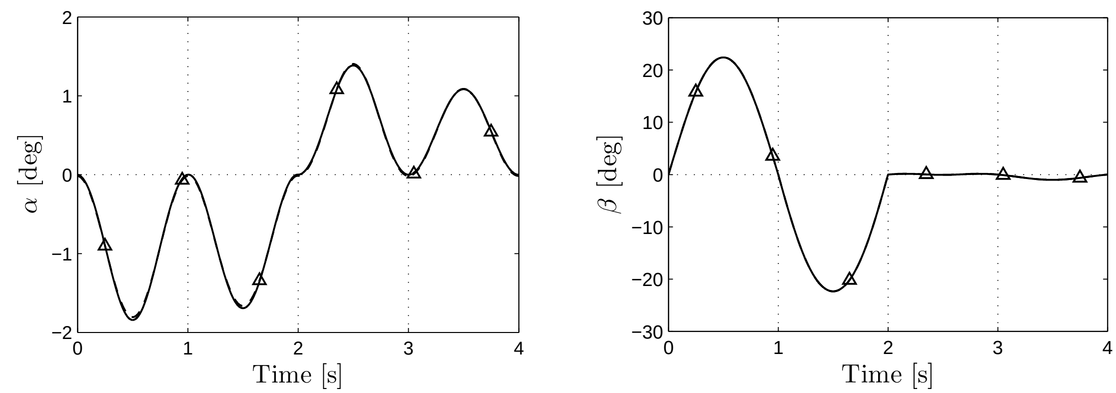
Compiler	Iterating over approximate constraint equations	
	using symbolic inverse	using numeric inverse
Lcc	18.4 ms/s	23.1 ms/s
MS	15.4 ms/s	17.2 ms/s



(a) Spherical joint angles



(b) Revolute joint angles



(c) Universal joint angles

Figure 5.10: Results from kinematic simulation of double-wishbone suspension

5.3 Chapter summary

In this chapter, the kinematics of the five-link and double-wishbone suspensions were studied, both of which have only one degree-of-freedom once the steering input has been specified. Consequently, only one independent coordinate can be selected in each case, which is insufficient for obtaining fully triangular solutions. Nevertheless, block-triangular solutions are obtained, where subsets of the dependent generalized coordinates are calculated recursively and the remaining coordinates are calculated iteratively. The solution flow used for the five-link suspension was inspired by the analysis of the topologically similar Gough–Stewart platform, which was the topic of Section 4.2.5. Whereas a purely joint coordinate formulation leads to only four independent kinematic loops, describing the motion of the wheel carrier using absolute coordinates increases the number of loops to five. The resulting constraint equations are suitable for generating Gröbner bases and, further, can be readily reduced to a form suitable for the iterative component of the solution flow. Although the complexity of the double-wishbone constraint equations prevented the use of Gröbner bases, a triangular solution was derived manually using approximations that do not introduce a significant amount of error in the computed results. Furthermore, since the resulting equations are both efficient and entirely parametric, they are suitable for use in real-time and on-line suspension tuning applications. Kinematic and dynamic simulation results indicate that the block-triangular solutions generated herein are more computationally efficient than existing Newton–Raphson and constraint stabilization techniques. The analysis of the double-wishbone suspension is resumed in Section 6.2, where its implementation in a hardware- and operator-in-the-loop driving simulator is discussed.

Chapter 6

Driving Simulator

The focus of this chapter is the hardware- and operator-in-the-loop driving simulator that has been developed by the author as a test platform for automotive research, and as a means of confirming the real-time capability of the simulation code generated in Chapter 5. In hardware- and operator-in-the-loop applications, hardware components and human users interact with a simulated system as they would with the analogous physical system. Consequently, the software components in such a system must be simulated faster than real time, which generally involves the synchronized operation of several pieces of specialized hardware. The equipment that has been used in the driving simulator is described in Section 6.1. In Section 6.2, we discuss the real-time simulation of a vehicle with double-wishbone suspensions on both axles, exploiting the efficiency of the kinematic solution developed in the previous chapter.

6.1 System description

As the complexity of automotive systems continues to increase, so too does the need for the testing of these systems early in the design stage—often before physical prototypes have been manufactured. The many benefits of hardware-in-the-loop simulation include its ability to replace preliminary field tests with safer, faster, and more rigorous automated tests, and its ability to replicate extreme or unusual conditions at will, enabling the repeated simulation of cold-start engine tests in the summer, for example [73]. Hardware-in-the-loop testing has become an essential phase of the controller design process, and is used in the

Figure 6.1: Hardware- and operator-in-the-loop driving simulator

development and validation of low-level engine controllers [137] as well as high-level vehicle stability controllers [61]. The simulator that has been developed in this work is shown in Figure 6.1. The vehicle model and several of its controllers are simulated on a Peripheral Component Interconnect (PCI) Extensions for Instrumentation (PXI) system from National Instruments. The PXI system contains a 2.26-GHz Intel Q9100 quad-core processor, and uses the Laboratory Virtual Instrument Engineering Workbench (LabVIEW) Real-Time operating system to maintain precise simulation timing. LabVIEW is a graphical programming language that expedites the development of multi-threaded applications, and facilitates communication with external devices. A supervisory, or *host*, computer running a Windows-based version of LabVIEW communicates with the PXI system over Ethernet throughout the simulation. In addition to rendering three-dimensional graphical feedback and updating plots in real time, the host processes universal serial bus interrupts generated by the steering wheel and pedals, and relays the relevant information to the PXI system for use as inputs to the vehicle model. An electronic control unit (ECU) from MotoTron is used as a platform for prototyping dynamic controllers, and communicates with

the PXI system over a Controller Area Network (CAN) bus, which is a standard message-based protocol used in vehicular networks [24]. The ECU contains an 80-MHz Motorola MPC5554 microprocessor, and has capabilities similar to the hardware that would be used for dynamic controllers in production vehicles. Programs for the ECU are developed on the host using the MotoHawk blockset in Simulink, compiled using the Green Hills MULTI compiler, and deployed to the ECU over the CAN bus. Since the Green Hills compiler only supports the most basic Simulink blocks, system-function (S-function) blocks must be either implemented in a Target Language Compiler (TLC) file, or translated into C++ code that is called by a TLC S-function wrapper [126].

The simultaneous use of hardware- and operator-in-the-loop functionalities enables the testing of controller prototypes in realistic computational and operational circumstances. The system shown in Figure 6.1 has been useful in validating the performance of traction control, electronic stability control (ESC), active front steering (AFS) [61], and advanced torque vectoring (ATV) [60] strategies. An ESC system automatically applies braking torques to individual wheels in critical driving situations, which can help maintain the stability and maneuverability of a vehicle [102]. ESC systems will be mandatory on all new passenger vehicles and light trucks operated in Canada [127] and the United States [90] by September, 2011. AFS and ATV controllers provide similar assistance by actively steering the front wheels or redistributing the driving torque, but have less effect on the vehicle speed [53]. Since the intervention of AFS and ATV controllers is generally imperceptible to the driver, they are suitable for use in all driving situations [98]. The ability to test drive a virtual vehicle equipped with such controllers can provide valuable insight into their operation and effectiveness, as perceived by a human driver.

The development of dynamic controllers generally involves a large number of simulations, particularly when parameter tuning is necessary. Consequently, a branched-topology vehicle model is often used in place of a closed-kinematic-chain model for the initial development of vehicle stability controllers. Such model simplification techniques are also common in real-time simulation, as was discussed in Section 2.1.4. The 14-DOF model shown in Figure 6.2 is recommended by Sayers and Han [108] for simulating the handling and braking behaviour of a vehicle. Four lumped masses, each representing one-quarter of all suspension components, are connected to the vehicle chassis by prismatic joints;

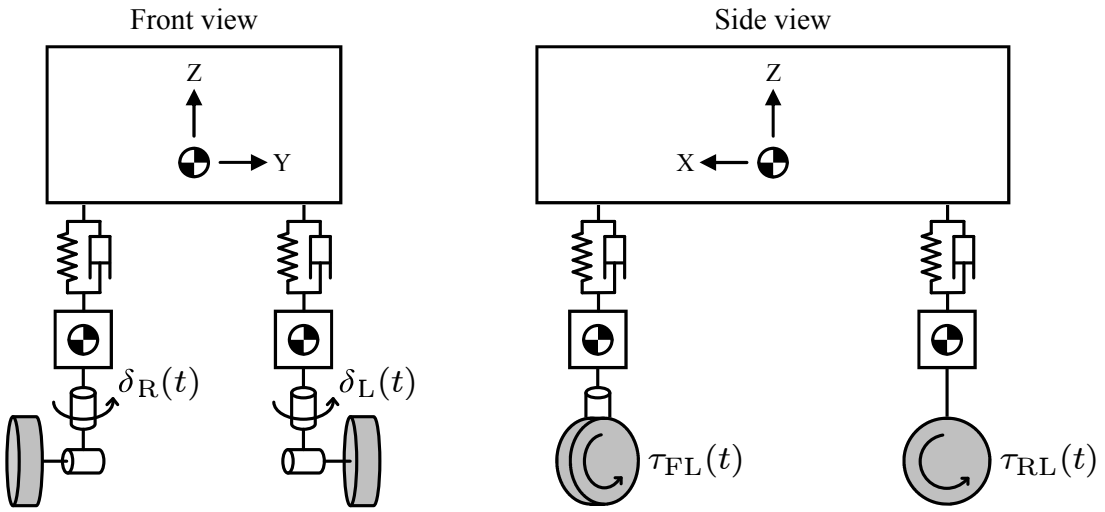


Figure 6.2: 14-degree-of-freedom branched-topology vehicle model

springs and dampers placed in parallel represent the suspension compliance. Each wheel is connected to its corresponding lumped mass with a revolute joint, which allows the wheel to spin. Vertically-oriented revolute joints on the front wheels ($\delta_{\mathrm{R}}(t)$ and $\delta_{\mathrm{L}}(t)$) are used to steer the vehicle through a trajectory that is either predetermined, calculated by a driver model, or prescribed by a human driver; thus, these revolute joints do not add any degrees-of-freedom to the system. In this work, the tire-road interaction is modelled using the Pacejka 2002 Magic Formula [93], and torques $\tau_{\mathrm{FL}}(t)$, $\tau_{\mathrm{FR}}(t)$, $\tau_{\mathrm{RL}}(t)$, and $\tau_{\mathrm{RR}}(t)$ are applied to the front-left, front-right, rear-left, and rear-right wheels by direct-drive electric in-wheel motors [132]. Due to the amount of computation involved, the vehicle model is implemented with a torque driver applied to each wheel, and is simulated on one central processing unit (CPU) core of the quad-core PXI system; the four electric in-wheel motor models are executed on a separate core. Cruise, traction, and torque vectoring controllers occupy the third CPU core, receiving sensor signals from the vehicle model and transmitting control signals (desired torques) to the low-level motor controllers at regular intervals; the fourth core is used for communicating with the host over Ethernet and the ECU over the CAN bus. Although a branched-topology model provides fast simulations, it is often desirable to evaluate the performance of a vehicle and its controllers using a more realistic closed-kinematic-chain model. The implementation of such a model in the driving simulator is the topic of the next section.

6.2 Double-wishbone vehicle model

Since nearly all vehicle suspension systems contain closed kinematic chains, the real-time simulation of high-fidelity vehicle models can be challenging. As discussed in Section 1.4, one of the applications of this research is the development of computationally efficient vehicle models that retain explicit representations of the actual kinematics and dynamics of the suspension. In this section, we discuss the real-time simulation of a vehicle with double-wishbone suspensions on both axles using the block-triangular kinematic solution generated in Section 5.2. Since the vehicle is bilaterally symmetric, the block-triangular solution generated earlier for the front-left (FL) corner can be readily applied to the front-right (FR) corner upon substitution of symmetric parameters. Although the suspension geometry on the rear axle differs from that on the front, the same strategy can be used to obtain block-triangular solutions for the rear-left (RL) and rear-right (RR) corners. Model parameters are provided in Appendix A.5.

Absolute coordinates are used to describe the position and orientation of the 6-DOF chassis: $\{x, y, z\}$ represent the position of the center of mass, and $\{\phi, \theta, \psi\}$ represent the corresponding 3-2-1 Euler angles which, in this context, are referred to as the yaw, pitch, and roll of the vehicle. Joint coordinates $\left\{{}^{w}\theta, \zeta, \eta, \xi, {}^{u}\theta, {}^{\ell}\theta, \alpha, \beta, s\right\}$ are used to model each of the four suspensions, as in Section 5.2. Since each suspension has a separate steering rack, Ackermann steering geometry and four-wheel steering [97] can be readily implemented. In this example, we simply apply rack displacements $s_{\mathrm{FL}}(t) = s_{\mathrm{FR}}(t) = s(t)$ to the two front suspensions, and fix $s_{\mathrm{RL}}(t) = s_{\mathrm{RR}}(t) = 0$ on the rear axle. We choose the following independent coordinates:

$$\mathbf{q}_i = \{x, y, z, \phi, \theta, \psi, {}^{w}\theta_{\mathrm{FL}}, {}^{w}\theta_{\mathrm{FR}}, {}^{w}\theta_{\mathrm{RL}}, {}^{w}\theta_{\mathrm{RR}}, \xi_{\mathrm{FL}}, \xi_{\mathrm{FR}}, \xi_{\mathrm{RL}}, \xi_{\mathrm{RR}}\} \tag{6.1}$$

and apply the embedding procedure as before. The kinematic solution flow shown in Figure 5.7 is used for each corner of the suspension; the dynamic solution flow is analogous to that shown in Figure 5.4. Although text-based programming scripts can be inserted into LabVIEW programs using Formula Node structures, a programming syntax is used that is similar to, but distinct from, the C programming language. As such, Microsoft Visual C++ is used to compile the optimized C code generated by Maple into a dynamic-link library (DLL), which can then be imported into LabVIEW directly [91].

A single-lane-change maneuver with prescribed rack displacement $s(t)$ is used to evaluate the performance of the same kinematic solution approaches that were considered in Section 5.2:

1. Computing $\mathbf{q}_{d1} = \left\{ {}^{u}\theta, {}^{\ell}\theta, \alpha, \beta \right\}$ and $\mathbf{q}_{d2} = \{\zeta, \eta\}$ for each corner using the block-triangular solution shown in Figure 5.7.

2. Computing $\mathbf{q}_d$ for each corner using Newton's method, iterating over the original constraint equations and using Gaussian elimination with partial pivoting to invert the Jacobian matrix numerically.

3. Computing $\mathbf{q}_d$ for each corner using Newton's method, iterating over the original constraint equations and using a symbolic Jacobian inverse.

All three approaches use the same set of projected dynamic equations. The vehicle is given an initial speed of 20 [m/s], and coasts through the maneuver using the predefined rack displacement shown in Figure 6.3. Although automotive ECUs typically operate with a 10-millisecond cycle time, the virtual vehicle in HIL applications is generally simulated using 1-millisecond time steps to reduce the effects of synchronization and communication delays [29]. Thus, we use a first-order Euler integration scheme with 1-millisecond time steps, and confirm that smaller time steps produce nearly identical results. The trajectory, roll, and pitch of the vehicle are shown in Figures 6.4 and 6.5(a); the upper control arm angles and lateral tire forces at the front-left and rear-left corners are shown in Figures 6.5(b) and 6.5(c). In all cases, an excellent agreement is observed between the block-triangular

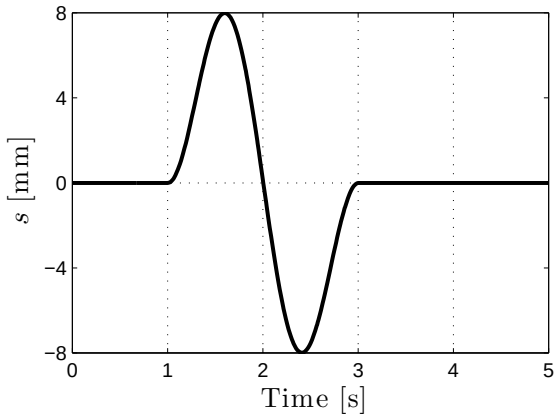


Figure 6.3: Rack displacement for dynamic simulation of double-wishbone vehicle

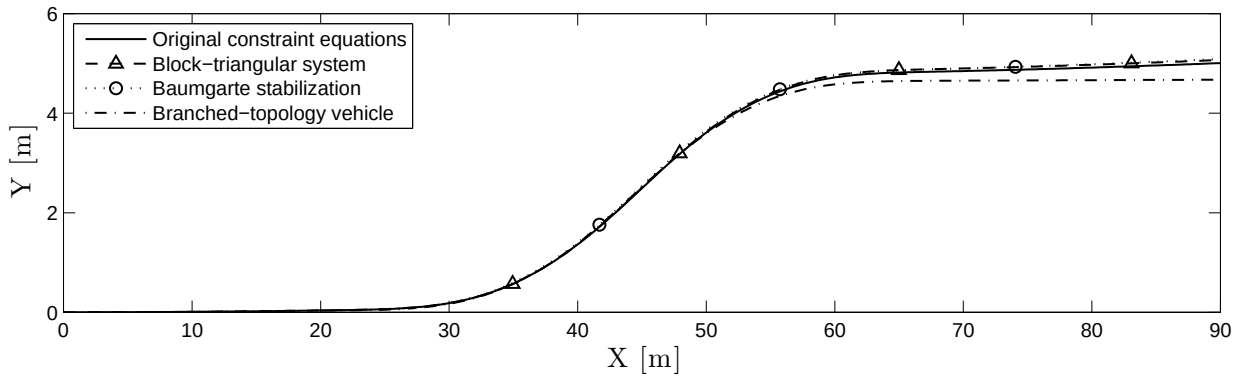


Figure 6.4: Trajectory of double-wishbone vehicle

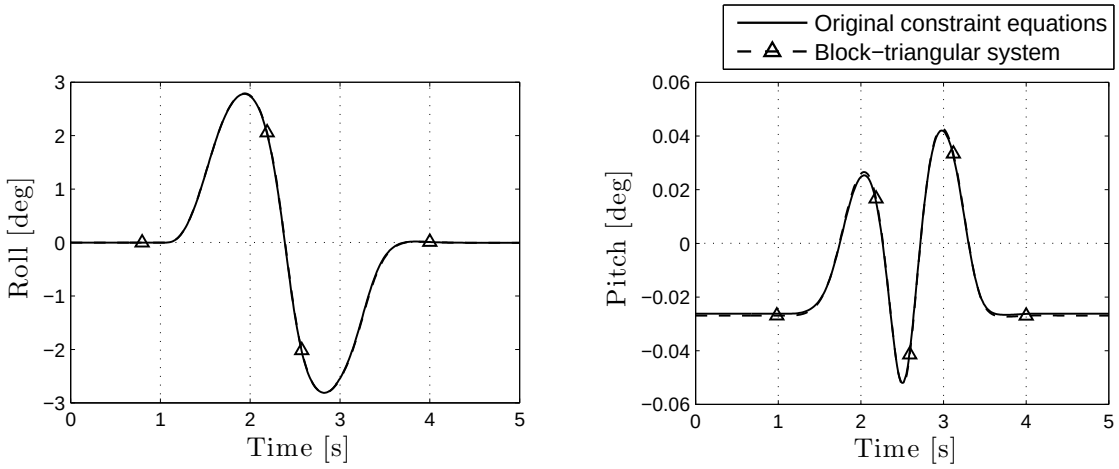
solution and the solutions involving iteration over the original constraint equations. The two additional trajectories shown in Figure 6.4 are obtained as follows:

1. Using Baumgarte's method with stabilization parameters $\alpha_B = \beta_B = 5$, and simulating in Maple using the Runge–Kutta–Fehlberg fourth-fifth-order integrator with error tolerances of 10^{-6}.

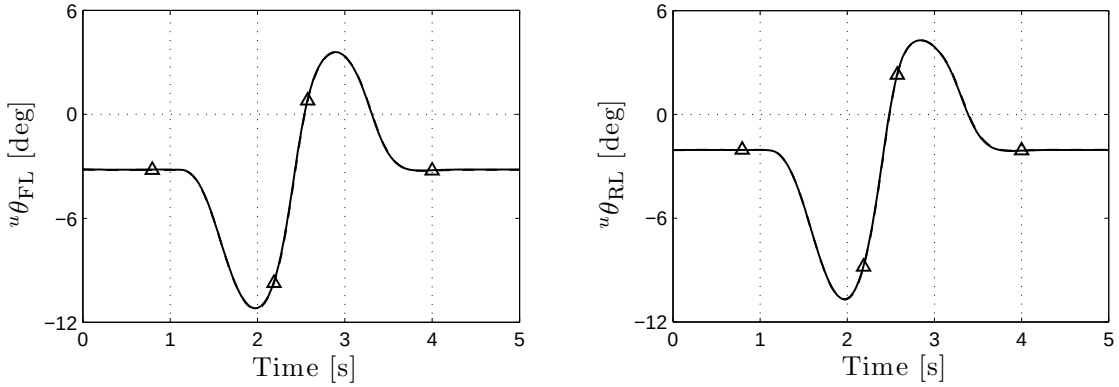
2. Using the 14-DOF branched-topology vehicle model illustrated in Figure 6.2 with system parameters that provide an approximation of the double-wishbone model [62], and simulating in LabVIEW using a first-order Euler integration scheme with 1-millisecond time steps.

Aside from that of the branched-topology vehicle, all trajectories differ by less than 6 [cm] in the lateral direction at the completion of the maneuver, or about 1% of the total lateral displacement, which is insignificant in most applications.

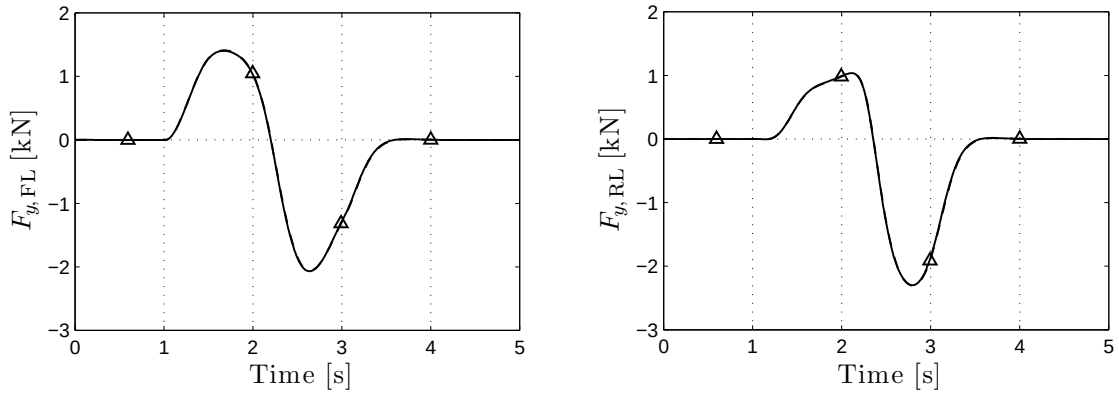
We now compare the computational efficiency associated with each solution strategy. Since this is a real-time application, where communication must occur at regular intervals, the maximum cycle time required for each approach must be considered; one relatively long cycle cannot be compensated by shorter subsequent cycles [3]. As such, the number of Newton steps is fixed in each case. The maximum cycle times required for dynamic simulations using the block-triangular and fully iterative solutions are shown in Table 6.1; the same metric for the branched-topology vehicle model is also provided for comparison. Note that two iterations of the block-triangular solution can be computed more rapidly than one iteration of the fastest Newton–Raphson approach. This observation may be



(a) Orientation of vehicle chassis



(b) Upper control arm angles at front-left and rear-left corners



(c) Lateral forces generated by front-left and rear-left tires

Figure 6.5: Results from dynamic simulation of double-wishbone vehicle

somewhat surprising; the kinematic simulation results obtained earlier, shown in Table 5.4, suggest that the block-triangular solution outperforms the Newton–Raphson approaches by a far narrower margin. We surmise that the overhead associated with executing DLLs in LabVIEW Real-Time—such as the allocation of memory for temporary variables—is significantly greater than the analogous overhead in MATLAB. Also note that the kinematic processing represents only a small portion of the total computation time, most of which is required for evaluating the dynamic equations.

As a final comparison, the maximum absolute error in the computation of $\mathbf{q}_d$ using each approach is shown in Table 6.2, where the computed values are compared to those obtained by iterating over the original constraint equations to a high tolerance. Note that Gaussian elimination with partial pivoting results in a more precise Jacobian inverse than that computed symbolically, thereby providing a more precise computation of $\mathbf{q}_d$. Since approximations have been used to generate the block-triangular solutions, it is not possible to obtain arbitrarily precise results by increasing the number of Newton steps. Nevertheless, the block-triangular solution is both more efficient and marginally more precise than the fastest Newton–Raphson approach when two iterations are used. Finally, although the vehicle model has been implemented on a single CPU core in this work, the calculation of $\mathbf{q}_d$ for each corner of the suspension could be performed on a separate core to obtain further increases in performance [95].

6.3 Chapter summary

The focus of this chapter was the hardware- and operator-in-the-loop driving simulator that has been developed as a test platform for automotive research. This system has been used to confirm the effectiveness of several dynamic control strategies, where its hardware- and operator-in-the-loop capabilities have been exploited to evaluate the performance of vehicle stability controllers in realistic computational and operational circumstances. Although an approximate branched-topology vehicle model provides faster cycle times, it is often desirable to use a closed-kinematic-chain model to obtain more accurate simulation results. The kinematic solution strategy presented in Section 5.2 has been used to simulate a vehicle with double-wishbone suspensions on both axles. The resulting real-

Table 6.1: Computational efficiency of double-wishbone vehicle simulations

Solution strategy	Temporary variables in kinematic solution	Number of Newton steps	Maximum cycle time
Branched-topology model	—	—	0.360 ms
Block-triangular solution	308	1	0.923 ms
		2	0.931 ms
		3	0.943 ms
Symbolic Jacobian inverse	564	1	0.941 ms
Numeric Jacobian inverse	208	1	1.233 ms

Table 6.2: Precision of double-wishbone vehicle simulations

Solution strategy	Newton steps	Maximum absolute error in $\mathbf{q}_d$ [rad]			
		Front-left	Front-right	Rear-left	Rear-right
Block-triangular solution	1	8.73 e−3	8.74 e−3	6.04 e−3	6.07 e−3
	2	3.56 e−4	3.58 e−4	2.81 e−4	2.85 e−4
	3	3.56 e−4	3.58 e−4	2.81 e−4	2.83 e−4
Symbolic Jacobian inverse	1	9.53 e−4	9.52 e−4	4.13 e−4	4.14 e−4
Numeric Jacobian inverse	1	3.37 e−4	3.36 e−4	1.66 e−4	1.66 e−4

time-capable block-triangular solution outperforms Newton–Raphson iterative approaches, providing more precise computations of the dependent coordinates in less time. Although the kinematic processing represents only a small portion of the total computation time, the results presented in this chapter confirm the utility of a block-triangular kinematic solution when highly efficient simulation code is desired, as is often the case in real-time hardware- and operator-in-the-loop applications.

Chapter 7

Conclusions

In this work, we have explored the use of triangular, or recursively solvable, systems of kinematic equations to facilitate the real-time simulation of constrained multibody systems. In the ideal case, an exact Gröbner basis can be used to obtain a fully triangular system. Such systems can be solved exactly and in a fixed amount of time, two properties that are highly desirable in many applications. Where full triangularization is not possible, a block-triangular form can be obtained that still results in more efficient simulations than existing techniques. When coupled with the embedding technique, the proposed approach results in computationally efficient dynamic simulation code that reduces the use of iteration. The significant contributions of this work are highlighted in Section 7.1; recommendations for future research are outlined in Section 7.2.

7.1 Contributions

The Gröbner basis approach yields simulation code with several advantageous properties.

Efficient In contrast to a fully iterative solution strategy, where more precise solutions require more iterations, fully triangular systems can be solved using a fixed number of arithmetic operations. As has been demonstrated, even partial triangularization can result in more efficient simulation code. Although considerable effort may be required at the formulation stage to generate the necessary Gröbner bases, this one-time investment pays con-

tinual dividends, as all ensuing simulations are faster. We also note that the Gröbner basis approach often yields a system of kinematic equations that can be readily parallelized.

Precise Provided all equations are of degree no greater than four, fully triangular solutions can be solved entirely recursively. In such situations, the corresponding kinematic constraint equations are satisfied to machine precision, providing arbitrarily precise solutions in a fixed amount of time. Although it cannot always be eliminated entirely, reducing the use of Newton–Raphson iteration may prevent unanticipated jumping between adjacent solution branches. Furthermore, since a Gröbner basis has the same solutions as the original system of polynomials, all mechanism configurations are preserved in the resulting kinematic equations, and no extraneous solutions are introduced.

Algorithmic Unlike many other methods for solving systems of polynomial equations, a Gröbner basis can be obtained algorithmically and, as such, is suitable for use in an automated formulation procedure. While an algorithmic method for determining the optimal term ordering remains elusive, traversal of the topological system graph—which is readily available when using a graph-theoretic formulation procedure—has been found to be an effective strategy.

Versatile In contrast to the characteristic pair of joints technique, the Gröbner basis approach can be applied to systems modelled with any set of generalized coordinates. Furthermore, there is often a great deal of latitude in the sequence in which the dependent generalized coordinates must be solved. Consequently, the Gröbner basis approach is capable of identifying more efficient systems of kinematic equations than the characteristic pair of joints technique. If only a subset of the dependent coordinates is of interest in a kinematic simulation, the term ordering can be selected so as to avoid the computation of unnecessary coordinates.

In addition to the development of the Gröbner basis approach, efficient kinematic solutions have been generated for several closed-kinematic-chain mechanisms:

- Single-loop: planar slider-crank and spatial four-bar mechanisms.
- Cascading-loop: hydraulic excavator and synthetic aperture radar satellite antenna.
- Six-bar: Stephenson-III and aircraft landing gear mechanisms.
- Parallel: planar parallel robot and Gough–Stewart platform.
- Vehicle suspensions: five-link and double-wishbone suspensions.

The Gröbner basis approach is particularly suitable for situations requiring very efficient simulations of multibody systems whose parameters are constant, such as the plant models in model-predictive control strategies and the vehicle models in driving simulators. In such situations, the requirement for efficient simulation code outweighs the preference for a rapid formulation procedure. Note that even small improvements in computational efficiency can be of significant importance. Improving the efficiency of a predictive vehicle model, for example, permits the use of less powerful—and, therefore, less expensive—electronic control units. Alternatively, more sophisticated models could be implemented on the existing hardware, potentially resulting in more effective control strategies and safer vehicles.

The final significant contribution of this work is the development of a hardware- and operator-in-the-loop driving simulator, whose modular design makes it suitable for a wide range of automotive research. The simultaneous use of hardware- and operator-in-the-loop functionalities has enabled the testing of several controller prototypes in realistic computational and operational circumstances. In particular, the performance of traction control, electronic stability control, active front steering, and advanced torque vectoring strategies have been evaluated. Furthermore, a real-time-capable model of a vehicle with double-wishbone suspensions on both axles has been implemented. As a final note, we add that a Maple implementation of Buchberger's improved algorithm [15] has been developed using bracket coefficients, following the approach of Shirayanagi [116].

7.2 Recommendations for future research

In this work, we have explored the use of Gröbner bases for triangularizing the nonlinear algebraic constraint equations that arise when modelling closed-kinematic-chain mechanisms, in the pursuit of efficient simulation code for real-time applications. Real-time simulation is not a new concept, and is widely used in the development and validation of dynamic controllers, in virtual reality simulators, and in physics-based video games. Thus, not surprisingly, many approaches to real-time simulation have already been developed, and have been successfully applied to practical applications in many fields. The focus of the present work has been the exploration of an approach using Gröbner bases. Despite its demonstrated advantages in the simulation of several practical mechanisms, three challenges remain that may prevent the widespread use of the proposed technique: full automation, reduction of computation time, and applicability to large systems.

7.2.1 Full automation

Ideally, an algorithm would exist to convert a set of differential-algebraic equations into a set of ordinary differential equations supported by a maximally triangular system of kinematic equations. A fundamental challenge preventing the full automation of the Gröbner basis approach is the interdependence of numerous factors on the triangularizability of a given system. In particular, one must determine the optimal set of modelling coordinates $\mathbf{q}$ and independent coordinates $\mathbf{q}_i$, as well as the optimal solution flow for the mechanism and elimination order for each independent loop, which are nontrivial tasks in the general case. In this work, we have presented some heuristics and several practical examples that may be used to guide the application of the proposed approach to additional systems. Nevertheless, it would be desirable to forge these elemental strategies into a fully automated approach.

Kecskeméthy et al. [68] select independent loops by identifying in the topological system graph the cycle basis with the minimum total weight, where the edge weights are based on the number of coordinates associated with the corresponding joint. The minimum cycle basis can be found using standard graph-theoretic algorithms [58]. Once the independent loops have been selected, the optimal solution flow can be readily determined. Note,

however, that purely joint coordinates are used in this approach, which greatly reduces the number of possible solution strategies. Since the use of absolute coordinates can be advantageous in some situations, as has been demonstrated, it may be interesting to pursue an extension of the ideas of Kecskeméthy et al. to arbitrary sets of modelling coordinates.

The Dymola modelling and simulation package processes models developed in the Modelica object-oriented language, and employs two techniques that may be useful in developing an automated triangularization algorithm. The first technique is Tarjan's algorithm [125], which partitions a set of algebraic equations into subsets that can be solved independently. If applied to the kinematic constraints for a complex multibody system, Tarjan's algorithm could be used to automatically isolate the equations associated with each leg in a parallel robot or each suspension in a vehicle, for example. The second potentially useful technique is called *tearing* [30], and is used for reducing the amount of iteration required to solve the many sparse equations typical of Modelica models. Dymola also uses tearing to generate the constraint equations in closed-kinematic-chain systems, which are then solved iteratively. It may be possible to apply similar ideas to obtain block-triangular forms of the constraints themselves.

Finally, we note that a prerequisite for generating each Gröbner basis is the determination of a suitable elimination order. Although the heuristics employed in this work have been largely successful, the relationship between the elimination order and the complexity of the resulting triangular system remains nebulous. Sawada [105] proposed some heuristics for general systems of polynomials based on experimental results, but assumes that the origin of the equations is unknown. Since each set of constraint equations in the present work is known to originate from a multibody system whose topological system graph is given, it is conceivable that a more effective strategy can be devised in this case.

7.2.2 Reduction of computation time

Although a fully automated algorithm for triangularizing kinematic constraints would be desirable, a concern of practical significance is the amount of time required for generating Gröbner bases. As has been demonstrated, generating a Gröbner basis for a relatively small number of polynomials using an appropriate elimination order can provide results in a reasonable amount of time; the use of floating-point coefficients can also alleviate the

computational burden associated with Gröbner basis generation. A heretofore unexplored avenue is the use of triangular sets or regular chains in place of Gröbner bases. In the present context, a triangular set can be interpreted as a finite set of triangular systems that describes all possible mechanism configurations for all choices of system parameters; a regular chain is one type of triangular set [4]. Consider, for example, the kinematic constraints for the planar slider-crank mechanism discussed in Chapter 3, shown here after applying the familiar trigonometric substitutions:

$$\left[\, \ell_1 \mathrm{c}_\theta + \ell_2 \mathrm{s}_\beta - s, \quad \ell_1 \mathrm{s}_\theta - \ell_2 \mathrm{c}_\beta, \quad \mathrm{s}_\theta^2 + \mathrm{c}_\theta^2 - 1, \quad \mathrm{s}_\beta^2 + \mathrm{c}_\beta^2 - 1 \,\right] \tag{7.1}$$

Using the `RegularChains:-Triangularize` command in Maple with the variable ordering $\mathrm{s}_\beta \succ \mathrm{c}_\beta \succ \mathrm{s}_\theta \succ \mathrm{c}_\theta \succ s \succ \ell_1 \succ \ell_2$, we obtain a set of four triangular systems:

$$\begin{aligned}
\Theta_1 = \big[\, & \ell_1 \mathrm{c}_\theta + \ell_2 \mathrm{s}_\beta - s, \;\; \ell_1 \mathrm{s}_\theta - \ell_2 \mathrm{c}_\beta, \;\; \mathrm{s}_\theta^2 + \mathrm{c}_\theta^2 - 1, \\
& s^2 - 2\ell_1 \mathrm{c}_\theta s + \ell_1^2 - \ell_2^2 \,\big] && \text{if } \ell_2 \neq 0 \text{ and } 2\ell_1 s \neq 0 \qquad (7.2) \\
\Theta_2 = \big[\, & \mathrm{s}_\beta^2 + \mathrm{c}_\beta^2 - 1, \;\; \mathrm{s}_\theta, \;\; \ell_1 \mathrm{c}_\theta - s, \;\; s^2 - \ell_1^2, \;\; \ell_2 \,\big] && \text{if } \ell_1 \neq 0 \qquad (7.3) \\
\Theta_3 = \big[\, & \ell_1 \mathrm{c}_\theta + \ell_2 \mathrm{s}_\beta, \;\; \ell_1 \mathrm{s}_\theta - \ell_2 \mathrm{c}_\beta, \;\; \mathrm{s}_\theta^2 + \mathrm{c}_\theta^2 - 1, \;\; s, \\
& \ell_1^2 - \ell_2^2 \,\big] && \text{if } \ell_2 \neq 0 \qquad (7.4) \\
\Theta_4 = \big[\, & \ell_2 \mathrm{s}_\beta - s, \;\; \mathrm{c}_\beta, \;\; \mathrm{s}_\theta^2 + \mathrm{c}_\theta^2 - 1, \;\; s^2 - \ell_2^2, \;\; \ell_1 \,\big] && \text{if } \ell_2 \neq 0 \qquad (7.5)
\end{aligned}$$

Note that Θ_1 results in the same recursively solvable system as that found using a pure lexicographic Gröbner basis with the same elimination order (3.59). Triangular system Θ_2 provides a solution for a planar slider-crank mechanism whose connecting rod is of zero length ($\ell_2 = 0$), Θ_3 assumes the crank and connecting rod are of equal length ($\ell_1^2 - \ell_2^2 = 0$) and the piston remains at the origin ($s = 0$), and Θ_4 applies when the length of the crank is zero ($\ell_1 = 0$). Clearly, Θ_1 is the only system of practical interest in this case; however, it may be possible to use regular chains to avoid vanishing denominators under specialization of a Gröbner basis. Furthermore, since the triangular system is partitioned into several simpler branches, it may be more computationally efficient to compute regular chains than Gröbner bases—though perhaps not with the algorithms currently available [32]. The theory of border bases [88], which relaxes the rigid definition of a term ordering used to generate Gröbner bases, may also be worth investigating.

Since triangularizing kinematic constraint equations is often time-consuming, it may be worthwhile to instead store a library of triangular systems that can be assembled based

on the topological system graph. Kecskeméthy et al. [68] use this strategy to generate efficient simulation code for mechanisms modelled with purely joint coordinates. Since using absolute coordinates can be advantageous in some situations, it would be desirable to retain this ability; however, the number of possible triangular systems dramatically increases when an arbitrary set of modelling coordinates is permitted. Furthermore, it would be necessary to compute all triangular systems with symbolic parameters, which can be computationally expensive. Nevertheless, it would only be necessary to generate such a library once, and the resulting systems could be evaluated efficiently using floating-point arithmetic. Once such facilities have been established, it may be more reasonable to pursue the full automation of the Gröbner basis approach. We also note that Gröbner bases were originally considered to be too computationally expensive to be of use in most practical problems, and retained this reputation well into the 1990s [96]; the computational resources of the future will be even better equipped to handle these computations.

7.2.3 Applicability to large systems

The applicability of the Gröbner basis approach to large systems is primarily dependent on reducing the computation time required to triangularize the constraints. Recall that the Gough–Stewart platform required about 2 hours of computation time; the five-link suspension, nearly 4 hours. Clearly, it may be difficult to apply the proposed approach to larger systems—such as the double-wishbone suspension, for which a Gröbner basis could not be generated. Also recall that the system parameters have generally been substituted prior to triangularizing in order to reduce the number of indeterminates. While this strategy facilitates the computation of Gröbner bases, retaining the original parameterization of the system would be desirable in many applications, such as those involving design optimization, sensitivity analysis, and parameter tuning tasks. Finally, although converting floating-point coefficients into rational numbers was suitable for the examples presented herein, the computation of approximate Gröbner bases may be an interesting avenue of exploration, particularly in situations where exact coefficients would be prohibitively lengthy.

7.2.4 Other topics

In this work, we have focused on triangularizing the kinematic constraint equations associated with closed-kinematic-chain mechanisms, relying on the embedding procedure to reduce the dynamic equations to a minimal set. Despite the many advantages of this approach, the dynamic equations are considerably more complex after projection than they are in their original form. It is conceivable that, in some situations, the benefits of a fully triangular kinematic solution would be outweighed by the computational cost of the projected dynamic equations. Thus, it may be more advantageous to seek an efficient set of dynamic equations first, and only then seek a maximally triangular kinematic solution. It would be interesting to investigate whether the heuristics proposed by Léger and McPhee [75] could be extended to account for the additional complexity introduced by the projection operation. Finally, it may be advantageous to apply the philosophy of triangularization to multibody systems containing flexible bodies, to the evaluation of systems of ordinary differential equations, and to systems of equations from electrical, hydraulic, thermal, and other physical domains.

So here I sit,
My thesis writ,
With one thing more to do:

To say my thanks
Throughout the ranks,
And pass my pen to you.

$\mathcal{T}.\,\mathcal{K}.\,\mathcal{U}.$

Permissions

Some of the content of this thesis, including part of Section 2.2.5 and much of Chapter 4, has been published in *Multibody System Dynamics* [131], and is reproduced with permission from Springer.

References

[1] J. Angeles. *Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms*. Mechanical Engineering Series. Springer Science+Business Media, New York, 3rd edition, 2007.

[2] H. Anton and C. Rorres. *Elementary Linear Algebra: Applications Version*. John Wiley & Sons, New York, 8th edition, 2000.

[3] M. Arnold, B. Burgermeister, and A. Eichberger. Linearly implicit time integration methods in real-time applications: DAEs and stiff ODEs. *Multibody System Dynamics*, 17(2–3):99–117, 2007.

[4] P. Aubry, D. Lazard, and M. Moreno Maza. On the theories of triangular sets. *Journal of Symbolic Computation*, 28(1–2):105–124, 1999.

[5] D. S. Bae, J. K. Lee, H. J. Cho, and H. Yae. An explicit integration method for realtime simulation of multibody vehicle models. *Computer Methods in Applied Mechanics and Engineering*, 187(1–2):337–350, 2000.

[6] O. A. Bauchau and A. Laulusa. Review of contemporary approaches for constraint enforcement in multibody systems. *ASME Journal of Computational and Nonlinear Dynamics*, 3(1):011005, 2008.

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

[8] E. Bayo, J. García de Jalón, A. Avello, and J. Cuadrado. An efficient computational method for real time multibody dynamic simulation in fully cartesian coordinates. *Computer Methods in Applied Mechanics and Engineering*, 92(3):377–395, 1991.

[9] E. Bayo, J. García de Jalón, and M. A. Serna. A modified Lagrangian formulation for the dynamic analysis of constrained mechanical systems. *Computer Methods in Applied Mechanics and Engineering*, 71(2):183–195, 1988.

[10] M. Bodrato and A. Zanoni. Numerical Gröbner bases and syzygies: an interval approach. In D. Petcu, V. Negru, D. Zaharie, and T. Jebelean, editors, *Proceedings of the 6^{th} International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pages 77–89, Timisoara, Romania, 26–30 September 2004.

[11] W. Boege, R. Gebauer, and H. Kredel. Some examples for solving systems of algebraic equations by calculating Groebner bases. *Journal of Symbolic Computation*, 2(1):83–98, 1986.

[12] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. Elsevier Science Publishing Co., New York, 1976.

[13] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics, Philadelphia, 1996.

[14] B. Buchberger. A criterion for detecting unnecessary reductions in the construction of Gröbner-bases. In E. Ng, editor, *Symbolic and Algebraic Computation*, volume 72 of *Lecture Notes in Computer Science*, pages 3–21. Springer Berlin, Heidelberg, 1979.

[15] B. Buchberger. Gröbner bases: an algorithmic method in polynomial ideal theory. In N. K. Bose, editor, *Multidimensional Systems Theory: Progress, Directions and Open Problems in Multidimensional Systems*, pages 184–232. D. Reidel Publishing Company, Dordrecht, 1985.

[16] B. Buchberger. Gröbner bases and systems theory. *Multidimensional Systems and Signal Processing*, 12(3–4):223–251, 2001.

[17] B. Buchberger. Bruno Buchberger's PhD thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal (translated by M. P. Abramson). *Journal of Symbolic Computation*, 41(3–4):475–511, 2006.

[18] B. Buchberger and F. Winkler, editors. *Gröbner Bases and Applications*. London Mathematical Society Lecture Note Series, volume 251. Cambridge University Press, Cambridge, 1998.

[19] B. Burgermeister, M. Arnold, and B. Esterl. DAE time integration for real-time applications in multi-body dynamics. *Journal of Applied Mathematics and Mechanics*, 86(10):759–771, 2006.

[20] A. Cayley. On the theory of elimination. *Cambridge and Dublin Mathematics Journal*, 3:116–120, 1848.

[21] V. Cherian, N. Jalili, and V. Ayglon. Modelling, simulation, and experimental verification of the kinematics and dynamics of a double wishbone suspension configuration. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 223(10):1239–1262, 2009.

[22] S. Collart, M. Kalkbrener, and D. Mall. Converting bases with the Gröbner walk. *Journal of Symbolic Computation*, 24(3–4):465–469, 1997.

[23] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer Science+Business Media, New York, 3rd edition, 2007.

[24] V. Denner, J. Maier, D. Kraft, and G. Spreitz. Data processing and communication networks in motor vehicles. In *Bosch Automotive Handbook*. Robert Bosch GmbH, Plochingen, 6th edition, 2004.

[25] A. K. Dhingra, A. N. Almadi, and D. Kohli. A Gröbner-Sylvester hybrid method for closed-form displacement analysis of mechanisms. *ASME Journal of Mechanical Design*, 122(4):431–438, 2000.

[26] A. K. Dhingra, A. N. Almadi, and D. Kohli. Closed-form displacement and coupler curve analysis of planar multi-loop mechanisms using Gröbner bases. *Mechanism and Machine Theory*, 36(2):273–298, 2001.

[27] A. Eichberger and W. Rulka. Process save reduction by macro joint approach: the key to real-time and efficient vehicle simulation. *Vehicle System Dynamics*, 41(5):401–413, 2004.

[28] H. Elmqvist, S. E. Mattsson, H. Olsson, J. Andreasson, M. Otter, C. Schweiger, and D. Brück. Real-time simulation of detailed automotive models. In P. Fritzson, editor, *Proceedings of the 3rd International Modelica Conference*, pages 29–38, Linköping, Sweden, 3–4 November 2003.

[29] H. Elmqvist, S. E. Mattsson, H. Olsson, J. Andreasson, M. Otter, C. Schweiger, and D. Brück. Realtime simulation of detailed vehicle and powertrain dynamics. Paper No. 2004-01-0768, Society of Automotive Engineers, 2004.

[30] H. Elmqvist and M. Otter. Methods for tearing systems of equations in object-oriented modeling. In A. Guasch and R. M. Huber, editors, *Proceedings of the 1994 European Simulation Multiconference*, pages 326–332, Barcelona, Spain, 1–3 June 1994.

[31] B. Esterl and T. Butz. Simulation of vehicle-trailer combinations by real-time capable DAE solvers. Paper No. 2006-01-0802, Society of Automotive Engineers, 2006.

[32] J.-C. Faugère. A new efficient algorithm for computing Gröbner bases (F_4). *Journal of Pure and Applied Algebra*, 139(1–3):61–88, 1999.

[33] J.-C. Faugère. A new efficient algorithm for computing gröbner bases without reduction to zero (F_5). In M. Giusti, editor, *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*, pages 75–83, Lille, France, 7–10 July 2002.

[34] J.-C. Faugère, P. Gianni, D. Lazard, and T. Mora. Efficient computation of zero-dimensional Gröbner bases by change of ordering. *Journal of Symbolic Computation*, 16(4):329–344, 1993.

[35] J.-C. Faugère, J.-P. Merlet, and F. Rouillier. On solving the direct kinematics problem for parallel robots. Technical Report 5923, Institut National de Recherche en Informatique et en Automatique, May 2006.

[36] P. Fisette, T. Postiau, L. Sass, and J.-C. Samin. Fully symbolic generation of complex multibody models. *Mechanics Based Design of Structures and Machines*, 30(1):31–82, 2002.

[37] P. Fisette and J.-C. Samin. Symbolic generation of a multibody formalism of order N—extension to closed-loop systems using the coordinate partitioning method. *International Journal for Numerical Methods in Engineering*, 39(23):4091–4112, 1996.

[38] L. R. Foulds. *Graph Theory Applications*. Springer–Verlag, New York, 1992.

[39] R. Fröberg. *An Introduction to Gröbner Bases*. John Wiley & Sons, Chichester, 1997.

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

[41] C. W. Gear, B. Leimkuhler, and G. K. Gupta. Automatic integration of Euler–Lagrange equations with constraints. *Journal of Computational and Applied Mathematics*, 12–13:77–90, 1985.

[42] R. Gebauer and H. M. Möller. On an installation of Buchberger's algorithm. *Journal of Symbolic Computation*, 6(2–3):275–286, 1988.

[43] K. O. Geddes, S. R. Czapor, and G. Labahn. *Algorithms for Computer Algebra*. Kluwer Academic Publishers, Dordrecht, 1992.

[44] T. Geike and J. McPhee. Inverse dynamic analysis of parallel manipulators with full mobility. *Mechanism and Machine Theory*, 38(6):549–562, 2003.

[45] G. T. Gillies. The Newtonian gravitational constant: recent measurements and related studies. *Reports on Progress in Physics*, 60(2):151–225, 1997.

[46] A. Giovini, T. Mora, G. Niesi, L. Robbiano, and C. Traverso. "One sugar cube, please" or Selection strategies in the Buchberger algorithm. In S. M. Watt, editor, *Proceedings of the 1991 International Symposium on Symbolic and Algebraic Computation*, pages 49–54, Bonn, Germany, 15–17 July 1991.

[47] G. Gogu. Chebychev–Grübler–Kutzbach's criterion for mobility calculation of multi-loop mechanisms revisited via theory of linear transformations. *European Journal of Mechanics A/Solids*, 24(3):427–441, 2005.

[48] C. M. Gosselin and J. Angeles. The optimum kinematic design of a planar three-degree-of-freedom parallel manipulator. *Journal of Mechanisms, Transmissions, and Automation in Design*, 110(1):35–41, 1988.

[49] C. M. Gosselin and J. Angeles. Singularity analysis of closed-loop kinematic chains. *IEEE Transactions on Robotics and Automation*, 6(3):281–290, 1990.

[50] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. Springer–Verlag, Berlin, 1991.

[51] A. Hashemi and D. Lazard. Sharper complexity bounds for zero-dimensional Gröbner bases and polynomial system solving. Technical Report 5491, Institut National de Recherche en Informatique et en Automatique, February 2005.

[52] W. M. Hawkins, Jr. Aircraft landing gear mechanism. United States Patent No. 2,690,888, 5 October 1954.

[53] J. He, D. A. Crolla, M. C. Levesley, and W. J. Manning. Integrated active steering and variable torque distribution control for improving vehicle handling and stability. Paper No. 2004-01-1071, Society of Automotive Engineers, 2004.

[54] A. Heck. *Introduction to Maple*. Springer–Verlag, New York, 3rd edition, 2003.

[55] S. Hegazy, H. Rahnejat, and K. Hussain. Multi-body dynamics in full-vehicle handling analysis under transient manoeuvre. *Vehicle System Dynamics*, 34(1):1–24, 2000.

[56] R. C. Hibbeler. *Engineering Mechanics: Statics & Dynamics*. Prentice–Hall, Upper Saddle River, 9th edition, 2001.

[57] M. Hiller, A. Kecskeméthy, and C. Woernle. A loop-based kinematical analysis of complex mechanisms. In *Proceedings of the ASME Design Engineering Technical Conference*, Paper No. 86-DET-184, Columbus, Ohio, 5–8 October 1986.

[58] J. D. Horton. A polynomial-time algorithm to find the shortest cycle basis of a graph. *SIAM Journal on Computing*, 16(2):358–366, 1987.

[59] M. L. Husty. An algorithm for solving the direct kinematics of general Stewart–Gough platforms. *Mechanism and Machine Theory*, 31(4):365–379, 1996.

[60] K. Jalali. *Stability Control of Electric Vehicles with In-wheel Motors*. PhD thesis, University of Waterloo, 2010.

[61] K. Jalali, T. Uchida, S. Lambert, and J. McPhee. Development of an advanced slip controller and an active steering system for an electric vehicle with in-wheel motors using soft computing techniques. Awaiting publication, 2011.

[62] K. Jalali, T. Uchida, J. McPhee, and S. Lambert. Integrated stability control system for electric vehicles with in-wheel motors using soft computing techniques. *SAE International Journal of Passenger Cars—Electronic and Electrical Systems*, 2(1):109–119, 2009.

[63] C. M. Kalker-Kalkman. An implementation of Buchberger's algorithm with applications to robotics. *Mechanism and Machine Theory*, 28(4):523–537, 1993.

[64] D. Kapur and T. Saxena. Comparison of various multivariate resultant formulations. In A. H. M. Levelt, editor, *Proceedings of the 1995 International Symposium on Symbolic and Algebraic Computation*, pages 187–194, Montreal, Québec, 10–12 July 1995.

[65] A. Kecskeméthy. On closed form solutions of multiple-loop mechanisms. In J. Angeles, G. Hommel, and P. Kovács, editors, *Computational Kinematics*, pages 263–274. Kluwer Academic Publishers, Dordrecht, 1993.

[66] A. Kecskeméthy and M. Hiller. Automatic closed-form kinematics-solutions for recursive single-loop chains. In G. Kinzel, C. Reinholtz, L. W. Tsai, G. R. Pennock, R. J. Cipra, and B. B. Thompson, editors, *Flexible Mechanisms, Dynamics, and Analysis: Proceedings of the 22nd Biennial ASME Mechanisms Conference*, pages 387–393, Scottsdale, Arizona, 13–16 September 1992.

[67] A. Kecskeméthy and M. Hiller. An object-oriented approach for an effective formulation of multibody dynamics. *Computer Methods in Applied Mechanics and Engineering*, 115(3–4):287–314, 1994.

[68] A. Kecskeméthy, T. Krupp, and M. Hiller. Symbolic processing of multiloop mechanism dynamics using closed-form kinematics solutions. *Multibody System Dynamics*, 1(1):23–45, 1997.

[69] J. Knapczyk and M. Maniowski. Elastokinematic modeling and study of five-rod suspension with subframe. *Mechanism and Machine Theory*, 41(9):1031–1047, 2006.

[70] J. Knapczyk and M. Maniowski. Stiffness synthesis of a five-rod suspension for given load-displacement characteristics. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 220(7):879–889, 2006.

[71] A. Kondratyev, H. J. Stetter, and F. Winkler. Numerical computation of Gröbner bases. In V. G. Ganzha, E. W. Mayr, and E. V. Vorozhtsov, editors, *Proceedings of the 7^{th} International Workshop on Computer Algebra in Scientific Computing*, pages 295–306, Saint Petersburg, Russia, 12–19 July 2004.

[72] P. Kovács and G. Hommel. On the tangent-half-angle substitution. In J. Angeles, G. Hommel, and P. Kovács, editors, *Computational Kinematics*, pages 27–39. Kluwer Academic Publishers, Dordrecht, 1993.

[73] H. Krisp, K. Lamberg, and R. Leinfellner. Automated real-time testing of electronic control units. Paper No. 2007-01-0504, Society of Automotive Engineers, 2007.

[74] A. Laulusa and O. A. Bauchau. Review of classical approaches for constraint enforcement in multibody systems. *ASME Journal of Computational and Nonlinear Dynamics*, 3(1):011004, 2008.

[75] M. Léger and J. McPhee. Selection of modeling coordinates for forward dynamic multibody simulations. *Multibody System Dynamics*, 18(2):277–297, 2007.

[76] D. Lichtblau. Gröbner bases in Mathematica 3.0. *The Mathematica Journal*, 6(4):81–88, 1996.

[77] D. Lichtblau. Solving finite algebraic systems using numeric Gröbner bases and eigenvalues. In *Proceedings of the 4th World Multiconference on Systemics, Cybernetics and Informatics*, Orlando, Florida, 23–26 July 2000.

[78] D. Lichtblau. Approximate Gröbner bases and overdetermined algebraic systems. In *Applications of Computer Algebra, Symbolic and Numeric Computation session*, Linz, Austria, 27–30 July 2008.

[79] D. Lichtblau. Exact computation using approximate Gröbner bases. In *Applications of Computer Algebra, Symbolic and Numeric Computation session*, Linz, Austria, 27–30 July 2008.

[80] E. W. Mayr and S. Ritscher. Degree bounds for Gröbner bases of low-dimensional polynomial ideals. In W. Koepf, editor, *Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation*, pages 21–27, Munich, Germany, 25–28 July 2010.

[81] J. McPhee. Automatic generation of motion equations for planar mechanical systems using the new set of "branch coordinates". *Mechanism and Machine Theory*, 33(6):805–823, 1998.

[82] J. McPhee, C. Schmitke, and S. Redmond. Dynamic modelling of mechatronic multibody systems with symbolic computing and linear graph theory. *Mathematical and Computer Modelling of Dynamical Systems*, 10(1):1–23, 2004.

[83] J. McPhee, P. Shi, and J.-C. Piedbœuf. Dynamics of multibody systems using virtual work and symbolic programming. *Mathematical and Computer Modelling of Dynamical Systems*, 8(2):137–155, 2002.

[84] J.-P. Merlet. *Parallel Robots*. Solid Mechanics and Its Applications, volume 128. Springer, Dordrecht, 2nd edition, 2010.

[85] A. Montes. A new algorithm for discussing Gröbner bases with parameters. *Journal of Symbolic Computation*, 33(2):183–208, 2002.

[86] A. Morgan and A. Sommese. Computing all solutions to polynomial systems using homotopy continuation. *Applied Mathematics and Computation*, 24(2):115–138, 1987.

[87] B. Mourrain. The 40 "generic" positions of a parallel robot. In M. Bronstein, editor, *Proceedings of the 1993 International Symposium on Symbolic and Algebraic Computation*, pages 173–182, Kiev, Ukraine, 6–8 July 1993.

[88] B. Mourrain. Pythagore's dilemma, symbolic-numeric computation, and the border basis method. In D. Wang and L. Zhi, editors, *Symbolic-Numeric Computation*, Trends in Mathematics, pages 223–243. Birkhäuser Verlag Basel, Switzerland, 2007.

[89] MSC.Software Corporation. *ADAMS/Solver Manual: Integrator*, 2005.

[90] National Highway Traffic Safety Administration. *Electronic Stability Control Systems*. Federal Motor Vehicle Safety Standard No. 126, 2007.

[91] National Instruments Corporation. *Using External Code in LabVIEW*. Part No. 370109B-01, 2003.

[92] J. Nielsen and B. Roth. On the kinematic analysis of robotic mechanisms. *The International Journal of Robotics Research*, 18(12):1147–1160, 1999.

[93] H. B. Pacejka. *Tire and Vehicle Dynamics*. SAE International, Warrendale, 2nd edition, 2002.

[94] Y. A. Papegay, J.-P. Merlet, and D. Daney. Exact kinematics analysis of a car's suspension mechanisms using symbolic computation and interval analysis. *Mechanism and Machine Theory*, 40(4):395–413, 2005.

[95] T. Postiau, L. Sass, P. Fisette, and J.-C. Samin. High-performance multibody models of road vehicles: fully symbolic implementation and parallel computation. *Vehicle System Dynamics*, 35(Suppl.):57–83, 2001.

[96] M. Raghavan and B. Roth. Solving polynomial systems for the kinematic analysis and synthesis of mechanisms and robot manipulators. *ASME Journal of Mechanical Design*, 117(B):71–79, 1995.

[97] J. Reimpell, H. Stoll, and J. W. Betzler. *The Automotive Chassis: Engineering Principles*. Butterworth–Heinemann, Oxford, 2nd edition, 2001.

[98] P. E. Rieth and R. Schwarz. ESC II—ESC with active steering intervention. Paper No. 2004-01-0260, Society of Automotive Engineers, 2004.

[99] B. Roth. Computations in kinematics. In J. Angeles, G. Hommel, and P. Kovács, editors, *Computational Kinematics*, pages 3–14. Kluwer Academic Publishers, Dordrecht, 1993.

[100] B. Roth. Computational advances in robot kinematics. In J. Lenarčič and B. Ravani, editors, *Advances in Robot Kinematics and Computational Geometry*, pages 7–16. Kluwer Academic Publishers, Dordrecht, 1994.

[101] W. Rulka and E. Pankiewicz. MBS approach to generate equations of motions for HiL-simulations in vehicle dynamics. *Multibody System Dynamics*, 14(3–4):367–386, 2005.

[102] M. K. Salaani, C. Schwarz, G. J. Heydinger, and P. A. Grygier. Parameter determination and vehicle dynamics modeling for the National Advanced Driving Simulator of the 2006 BMW 330i. Paper No. 2007-01-0818, Society of Automotive Engineers, 2007.

[103] J.-C. Samin and P. Fisette. *Symbolic Modeling of Multibody Systems*. Kluwer Academic Publishers, Dordrecht, 2003.

[104] T. Sasaki and F. Kako. Floating-point Gröbner basis computation with ill-conditionedness estimation. In D. Kapur, editor, *Computer Mathematics*, volume 5081 of *Lecture Notes in Computer Science*, pages 278–292. Springer Berlin, Heidelberg, 2008.

[105] H. Sawada. Automatic generation of ranking of variables for efficient computation of Gröbner bases in engineering applications. In V. G. Ganzha, E. W. Mayr, and E. V. Vorozhtsov, editors, *Proceedings of the 6th International Workshop on Computer Algebra in Scientific Computing*, pages 319–328, Passau, Germany, 20–26 September 2003.

[106] M. W. Sayers. *Symbolic Computer Methods to Automatically Formulate Vehicle Simulation Codes*. PhD thesis, University of Michigan, 1990.

[107] M. W. Sayers. Vehicle models for RTS applications. *Vehicle System Dynamics*, 32(4–5):421–438, 1999.

[108] M. W. Sayers and D. Han. A generic multibody vehicle model for simulating handling and braking. *Vehicle System Dynamics*, 25(Suppl.):599–613, 1996.

[109] W. Schiehlen, A. Rükgauer, and T. H. Schirle. Force coupling versus differential algebraic description of constrained multibody systems. *Multibody System Dynamics*, 4(4):317–340, 2000.

[110] C. Schmitke, K. Morency, and J. McPhee. Using graph theory and symbolic computing to generate efficient models for multi-body vehicle dynamics. *Proceedings of the Institution of Mechanical Engineers, Part K: Journal of Multi-body Dynamics*, 222(4):339–352, 2008.

[111] A. A. Shabana. *Dynamics of Multibody Systems*. Cambridge University Press, New York, 3rd edition, 2005.

[112] L. Shih, A. A. Frank, and B. Ravani. Dynamic simulation of legged machines using a compliant joint model. *The International Journal of Robotics Research*, 6(4):33–46, 1987.

[113] T. Shiiba and Y. Suda. Development of driving simulator with full vehicle model of multibody dynamics. *JSAE Review*, 23(2):223–230, 2002.

[114] T. Shiiba and Y. Suda. Evaluation of driver's behavior with multibody-based driving simulator. *Multibody System Dynamics*, 17(2–3):195–208, 2007.

[115] K. Shirayanagi. An algorithm to compute floating point Gröbner bases. In T. Lee, editor, *Mathematical Computation with Maple V: Ideas and Applications*, pages 95–106. Birkhäuser, Boston, 1993.

[116] K. Shirayanagi. Floating point Gröbner bases. *Mathematics and Computers in Simulation*, 42(4–6):509–528, 1996.

[117] K. Shirayanagi and H. Sekigawa. A new Gröbner basis conversion method based on stabilization techniques. *Theoretical Computer Science*, 409(2):311–317, 2008.

[118] P. A. Simionescu and D. Beale. Synthesis and analysis of the five-link rear suspension system used in automobiles. *Mechanism and Machine Theory*, 37(9):815–832, 2002.

[119] J.-H. Sohn, W.-S. Yoo, K.-S. Kim, and J.-N. Lee. Force element formulation of bushed massless links for numerical efficiency. *Mechanics Based Design of Structures and Machines*, 29(4):477–497, 2001.

[120] A. J. Sommese and I. Charles W. Wampler. *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science*. World Scientific Publishing Co., Singapore, 2005.

[121] A. J. Sommese, J. Verschelde, and C. W. Wampler. Advances in polynomial continuation for solving problems in kinematics. *ASME Journal of Mechanical Design*, 126(2):262–268, 2004.

[122] M. W. Spong and M. Vidyasagar. *Robot Dynamics and Control*. John Wiley & Sons, New York, 1989.

[123] H. J. Stetter. Stabilization of polynomial systems solving with Groebner bases. In W. W. Küchlin, editor, *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation*, pages 117–124, Maui, Hawaii, 21–23 July 1997.

[124] A. Suzuki. Computing Gröbner bases within linear algebra. In V. P. Gerdt, E. W. Mayr, and E. V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing*, volume 5743 of *Lecture Notes in Computer Science*, pages 310–321. Springer Berlin, Heidelberg, 2009.

[125] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[126] The MathWorks, Inc. *Matlab R2007a Help: Real-Time Workshop → Target Language Compiler → Getting Started → Inlining S-Functions*, 2007.

[127] Transport Canada. *Electronic Stability Control Systems*. Technical Standards Document No. 126, 2009.

[128] C. Traverso. Gröbner trace algorithms. In P. Gianni, editor, *Symbolic and Algebraic Computation*, volume 358 of *Lecture Notes in Computer Science*, pages 125–138. Springer Berlin, Heidelberg, 1989.

[129] C. Traverso and A. Zanoni. Numerical stability and stabilization of Groebner basis computation. In M. Giusti, editor, *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*, pages 262–269, Lille, France, 7–10 July 2002.

[130] L.-W. Tsai. Solving the inverse dynamics of a Stewart–Gough manipulator by the principle of virtual work. *ASME Journal of Mechanical Design*, 122(1):3–9, 2000.

[131] T. Uchida and J. McPhee. Triangularizing kinematic constraint equations using Gröbner bases for real-time dynamic simulation. *Multibody System Dynamics*, 25(3):335–356, 2011.

[132] H. S. Vogt, C. Schmitke, K. Jalali, and J. McPhee. Unified modelling and real-time simulation of an electric vehicle. *International Journal of Vehicle Autonomous Systems*, 6(3–4):288–307, 2008.

[133] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, Cambridge, 2$^{\text{nd}}$ edition, 2003.

[134] C. W. Wampler. Displacement analysis of spherical mechanisms having three or fewer loops. *ASME Journal of Mechanical Design*, 126(1):93–100, 2004.

[135] J. Wang and C. M. Gosselin. A new approach for the dynamic analysis of parallel manipulators. *Multibody System Dynamics*, 2(3):317–334, 1998.

[136] J. Wang, C. M. Gosselin, and L. Cheng. Modeling and simulation of robotic systems with closed kinematic chains using the virtual spring approach. *Multibody System Dynamics*, 7(2):145–170, 2002.

[137] A. Wanpal, M. G. Babu, N. Kankariya, K. Mundhra, S. A. Sundaresan, and A. S. Deshpande. ECU testing and verification using hardware-in-the-loop. Paper No. 2006-01-1444, Society of Automotive Engineers, 2006.

[138] V. Weispfenning. Comprehensive Gröbner bases. *Journal of Symbolic Computation*, 14(1):1–29, 1992.

[139] J. Weiss. Resultant methods for the inverse kinematics problem. In J. Angeles, G. Hommel, and P. Kovács, editors, *Computational Kinematics*, pages 41–52. Kluwer Academic Publishers, Dordrecht, 1993.

[140] D. Wells. *The Penguin Dictionary of Curious and Interesting Numbers*. Penguin Group, London, 1986.

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

[142] A. Wittkopf. Automatic code generation and optimization in Maple. *Journal of Numerical Analysis, Industrial and Applied Mathematics*, 3(1–2):167–180, 2008.

Appendix A

System Parameters

The parameters used for the modelling and simulation of the aircraft landing gear mechanism, planar parallel robot, Gough–Stewart platform, five-link suspension, and double-wishbone suspension and vehicle model are given below.

A.1 Aircraft landing gear mechanism

Shown in Figure A.1 are the geometric parameters used to model the aircraft landing gear mechanism presented in Section 4.2.3. The location of each point on a body is given relative to the body-fixed reference frame shown. No units are stated since the geometry for this mechanism was obtained by measuring an undimensioned figure drawn by the designer [52]. For the purposes of generating a Gröbner basis, the measurements shown in Figure A.1 are treated as exact quantities. The locations of all points relative to the fuselage frame are given in Table A.1.

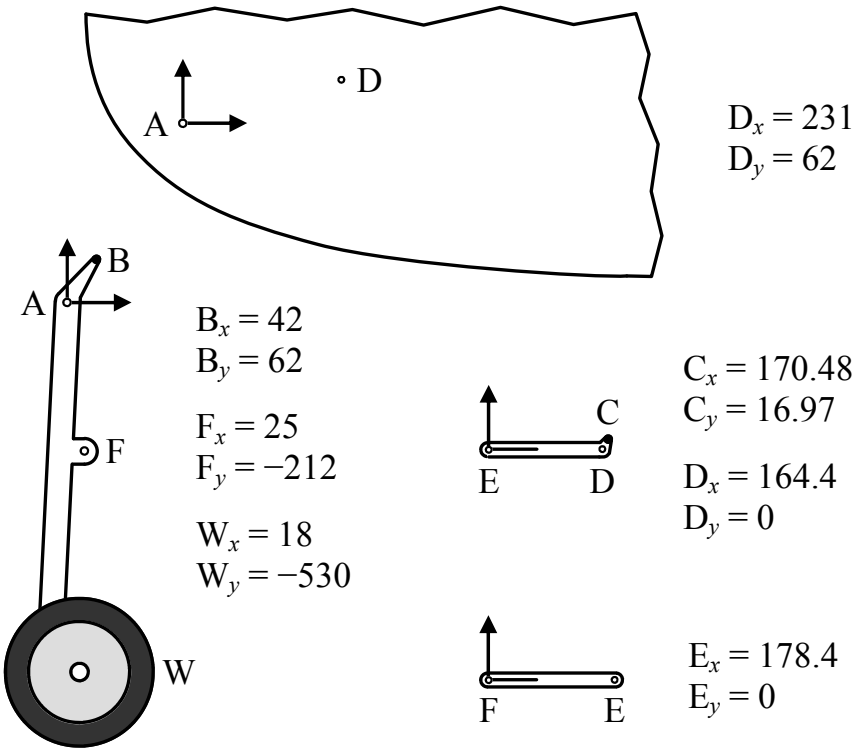


Figure A.1: Aircraft landing gear mechanism geometry

Table A.1: Point locations in aircraft landing gear mechanism relative to fuselage frame

Point	Landing gear deployed		Landing gear stowed	
	X coordinate	Y coordinate	X coordinate	Y coordinate
A	0.0	0.0	0.0	0.0
B	42.0	62.0	−46.4	58.8
C	221.2	77.1	237.5	78.8
D	231.0	62.0	231.0	62.0
E	131.2	−68.6	66.7	66.3
F	25.0	−212.0	209.6	−40.4
W	18.0	−530.0	510.6	−143.3

A.2 Planar parallel robot

The geometry of the planar parallel robot presented in Section 4.2.4 is shown in Figure A.2. Dynamic simulations are performed by applying torques $\tau_1(t) = 0.5\sin(t)$ [N·m], $\tau_2(t) = -0.25\cos(t)$ [N·m] and $\tau_3(t) = 0.5\sin(2t)$ [N·m] to joints θ_1, θ_2 and θ_3, respectively; all other system parameters are obtained from the work of Geike and McPhee [44], and are shown in Table A.2. Note that the end-effector is equilateral, as is the triangle defined by the three ground-fixed revolute joints, which are 1.0 [m] apart.

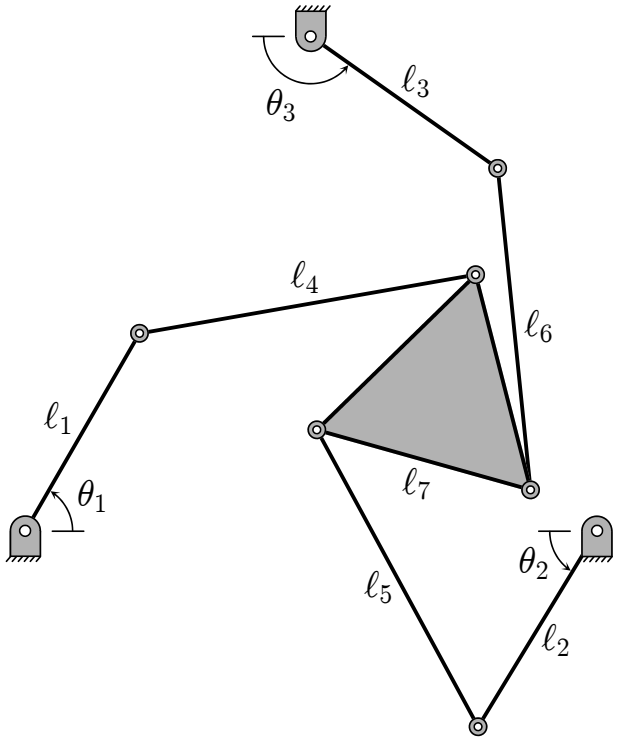


Figure A.2: Planar parallel robot geometry

Table A.2: Geometric and inertial parameters for planar parallel robot

Body	Length ℓ_k [m]	Mass [kg]	Inertia [kg·m^2]
1, 2, 3	0.4	3.0	0.04
4, 5, 6	0.6	4.0	0.12
7	0.4	8.0	0.0817

A.3 Gough–Stewart platform

The geometric and inertial parameters used to model the Gough–Stewart platform presented in Section 4.2.5 are obtained from the work of Tsai [130]. Each of the six legs consists of upper and lower segments of length 1.0 [m], which are attached to the base and platform at the points listed in Table A.3. All inertial parameters are given in Table A.4.

Table A.3: Joint locations for Gough–Stewart platform

Leg	Universal joint B_k (in base frame)			Spherical joint P_k (in platform frame)		
	X [m]	Y [m]	Z [m]	X [m]	Y [m]	Z [m]
1	−2.120	1.374	0	0.170	0.595	−0.4
2	−2.380	1.224	0	−0.600	0.150	−0.4
3	−2.380	−1.224	0	−0.600	−0.150	−0.4
4	−2.120	−1.374	0	0.170	−0.595	−0.4
5	0	−0.150	0	0.430	−0.445	−0.4
6	0	0.150	0	0.430	0.445	−0.4

Table A.4: Inertial parameters for Gough–Stewart platform

Body	Mass [kg]	Moments of inertia [kg·m^2]			Products of inertia [kg·m^2]		
		J_{xx}	J_{yy}	J_{zz}	J_{xy}	J_{xz}	J_{yz}
Platform	1.5	0.08	0.08	0.08	0	0	0
Leg segments	0.1	0.00625	0.00625	0	0	0	0

A.4 Five-link suspension

All geometric parameters are obtained from the description of suspension "S1" in the work of Knapczyk [70]. The links are attached to the chassis and wheel carrier at the points listed in Table A.5, which are provided relative to the wheel hub frame; link lengths L_k are computed from these values.

Table A.5: Joint locations for five-link suspension relative to wheel hub frame

Link	Universal joint C_k on chassis			Spherical joint W_k on wheel carrier		
	X [mm]	Y [mm]	Z [mm]	X [mm]	Y [mm]	Z [mm]
1	−103.2	−490.6	−65.6	−47.8	−50.1	−87.9
2	309.3	−204.6	−49.6	34.9	−77.3	−133.8
3	200.8	−283.6	−11.6	139.1	−47.4	−42.1
4	205.8	−269.6	95.4	76.9	−59.7	85.9
5	−1.7	−349.6	118.4	6.3	−52.2	119.9

A.5 Double-wishbone suspension and vehicle model

All system parameters for the double-wishbone vehicle model are obtained from the work of Jalali [60]. The suspension hardpoint locations, labelled in Figure A.3, are given for the front-left and rear-left corners in Tables A.6 and A.7, respectively; those for the front-right and rear-right corners can be obtained by noting the bilateral symmetry of the vehicle. Additional geometric and inertial parameters are given in Table A.8.

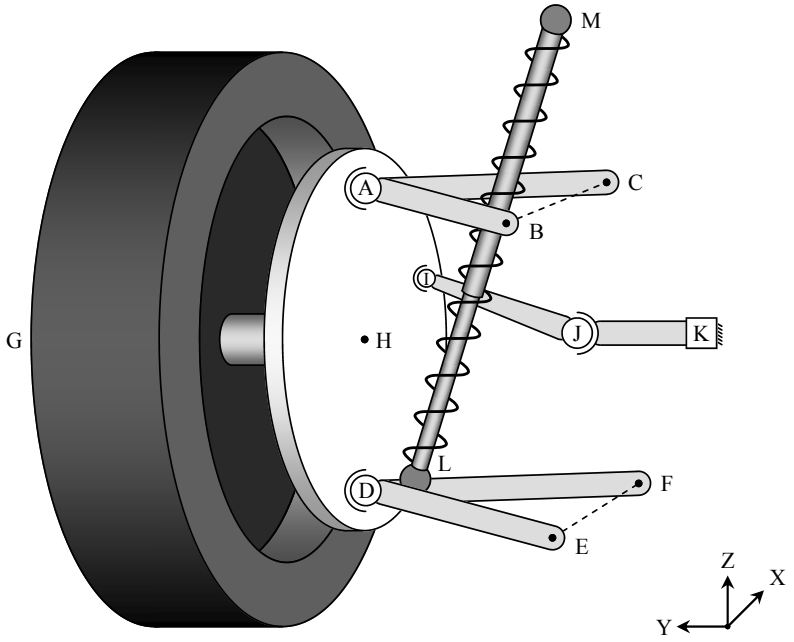


Figure A.3: Double-wishbone suspension hardpoints

Table A.6: Hardpoint locations for front-left double-wishbone suspension

Point	Description	X [mm]	Y [mm]	Z [mm]
A	Upper control arm on wheel carrier	−18.3	540.7	428.5
B	Upper control arm on chassis, aft	−86.9	360.0	412.6
C	Upper control arm on chassis, fore	52.2	360.0	428.5
D	Lower control arm on wheel carrier	17.3	615.5	159.5
E	Lower control arm on chassis, aft	−79.3	352.0	191.6
F	Lower control arm on chassis, fore	119.0	352.0	165.9
G	Wheel center	0.0	635.0	290.5
H	Drive shaft, inner	−0.1	579.0	290.0
I	Tie rod on wheel carrier	149.9	579.0	290.0
J	Tie rod on steering rack	40.0	303.3	298.5
K	Center of steering cylinder	40.0	0.0	298.6
L	Spring-damper on lower control arm	14.6	543.5	180.5
M	Spring-damper on chassis	−37.5	340.5	562.5

Table A.7: Hardpoint locations for rear-left double-wishbone suspension

Point	Description	X [mm]	Y [mm]	Z [mm]
A	Upper control arm on wheel carrier	0.0	578.2	428.5
B	Upper control arm on chassis, aft	−70.5	397.5	428.5
C	Upper control arm on chassis, fore	68.6	397.5	412.6
D	Lower control arm on wheel carrier	0.0	653.0	159.5
E	Lower control arm on chassis, aft	−101.7	389.5	165.9
F	Lower control arm on chassis, fore	96.7	389.5	191.6
G	Wheel center	0.0	672.5	290.5
H	Drive shaft, inner	−0.1	616.6	290.0
I	Tie rod on wheel carrier	−150.1	616.6	290.0
J	Tie rod on chassis	−40.0	334.8	297.3
L	Spring-damper on lower control arm	0.0	581.0	180.5
M	Spring-damper on chassis	0.0	378.0	562.5

Table A.8: Parameters for double-wishbone vehicle model

Parameter	Value	Unit
Wheelbase	1.8	m
Front track width	1.275	m
Rear track width	1.35	m
Chassis mass	882	kg
Front suspension stiffness	58.122	kN/m
Rear suspension stiffness	84.530	kN/m
Front suspension damping	9.250	kN·s/m
Rear suspension damping	11.801	kN·s/m
Tire mass	21.18	kg
Tire radius (unloaded)	0.2905	m
Tire width	0.175	m
Tire radial stiffness	132.724	kN/m
Tire radial damping	50	N·s/m