Using Gröbner bases to generate efficient kinematic solutions for the dynamic simulation of multi-loop mechanisms

Thomas Uchida *, John McPhee

Department of Systems Design Engineering, University of Waterloo, 200 University Avenue West, Waterloo, Ontario, Canada N2L 3G1

A R T I C L E   I N F O

Article history:
Received 1 September 2011
Received in revised form 15 January 2012
Accepted 21 January 2012
Available online 23 February 2012

Keywords:
Closed kinematic chain
Computational efficiency
Differential-algebraic equation
Gröbner basis
Kinematic loop
Symbolic computation

A B S T R A C T

Many mechanical systems of practical interest contain closed kinematic chains, and are most conveniently modeled using a set of redundant generalized coordinates. The governing dynamic equations for systems with more coordinates than degrees-of-freedom are differential-algebraic, and can be difficult to solve efficiently yet accurately. In this work, the embedding technique is used to eliminate the Lagrange multipliers from the dynamic equations and obtain one ordinary differential equation for each independent acceleration. Gröbner bases are then generated 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 solution can be obtained that is still more efficient than using existing techniques. The proposed approach is first applied to the Gough–Stewart platform, whose fully triangular solution motivates the block-triangular solution strategy for a five-link suspension system. Finally, a fully triangular solution is obtained for an aircraft landing gear mechanism.

© 2012 Elsevier Ltd. All rights reserved.

1. Introduction

The design and testing of multibody systems and their controllers often involve performing a large number of numerical simulations. Design optimization, sensitivity analysis, parameter identification, and controller tuning tasks can require hundreds or thousands of simulations, so even a moderate improvement in performance can result in significant time and cost savings. High-performance and parallel computing can be used to alleviate some of the computational burden associated with these tasks, but such facilities may not be available. Furthermore, there are scenarios in which highly efficient simulation code is essential, such as hardware- and operator-in-the-loop testing, model-predictive controller design, and other real-time applications.

The motion of open-loop or branched-topology systems can be described by pure ordinary differential equations (ODEs) derived in terms of a set of independent generalized coordinates, such as joint coordinates. Many systems of practical interest, however, such as parallel robots and vehicle suspensions, contain closed kinematic chains or kinematic loops, and are most readily modeled using a set of redundant coordinates. Dependent generalized coordinates add nonlinear algebraic constraint equations to the ODEs of motion, thereby producing a set of differential-algebraic equations (DAEs) that can be difficult to solve in an efficient yet accurate manner. Several approaches have been developed for simulating closed-kinematic-chain systems efficiently, some of which involve approximating the system equations [1,2], modifying the system model [3,4], or using a constraint stabilization technique [5,6]. The characteristic pair of joints approach [7] generates triangular, or recursively solvable, systems of kinematic equations that can often be solved without the use of iteration. Unlike many other approaches, the characteristic pair of joints technique enforces the constraints exactly, but can only be applied to systems modeled with joint coordinates.

* Corresponding author. Tel.: +1 519 888 4567 x33825; fax: +1 519 746 4791.
E-mail addresses: tkuchida@uwaterloo.ca (T. Uchida), mcphee@real.uwaterloo.ca (J. McPhee).

0094-114X/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mechmachtheory.2012.01.015
In this work, the embedding technique is used to eliminate the Lagrange multipliers from the dynamic equations and obtain one ODE for each independent acceleration. Gröbner bases [8] are then generated to triangularize the kinematic constraint equations, thereby producing recursively solvable systems for calculating the dependent generalized coordinates given values of the independent coordinates. A detailed description of the Gröbner basis approach and four relatively simple, fully triangularizable example systems have been presented by Uchida and McPhee [9]. The focus of the present paper is the analysis of three classes of multi-loop mechanisms:

1. Those whose kinematics can be fully triangularized with relative ease, such as the Gough–Stewart platform presented in Section 4.1;
2. Those whose kinematics can be reduced to a block-triangular form, such as the five-link suspension presented in Section 4.2; and
3. Those whose kinematics cannot be easily solved manually, such as the aircraft landing gear mechanism presented in Section 4.3.

When coupled with a graph-theoretic formulation procedure [10] in a symbolic computational environment, the proposed approach can be fully automated and used to generate highly efficient simulation code for systems modeled with an arbitrary set of coordinates. Although some additional effort is required at the formulation stage, this one-time investment pays continual dividends, as all ensuing simulations are faster. This strategy is particularly attractive in situations involving numerous simulations of the same multibody system, such as design optimization, parameter identification, and controller development applications, and for the vehicle models implemented in driving simulators and on-board dynamic controllers.

A brief review of some existing techniques for reducing DAEs to ODEs is provided in Section 2. The theory of Gröbner bases and its application to triangularizing constraint equations are discussed in Section 3. The Gough–Stewart platform, five-link suspension, and aircraft landing gear mechanism examples are presented in Section 4. Finally, conclusions and directions for future work are outlined in Section 5.

2. Existing techniques

The equations of motion for closed-kinematic-chain systems are most readily derived using a set of redundant generalized coordinates. If an augmented formulation [11] 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):

\[
\begin{align*}
\mathbf{M}\ddot{\mathbf{q}} + \Phi_\mathbf{q}'\mathbf{\lambda} &= \mathbf{F} \\
\Phi(\mathbf{q}) &= 0
\end{align*}
\]

where \( \mathbf{M} \) is the mass matrix, \( \mathbf{q} \) is the vector of generalized coordinates, \( \mathbf{\lambda} \) is the vector of Lagrange multipliers, \( \mathbf{F} \) is a vector containing the external loads and gyroscopic terms, and \( \Phi \) contains the constraint equations, which are functions of \( \mathbf{q} \). 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 a DAE as much as possible [12], ideally to zero. Note, however, that differentiating Eq. (2) can lead to significant position-level constraint violations. As such, several techniques have been developed for reducing the index of DAE systems while maintaining the integrity of the numerical solution. We shall briefly discuss three common approaches for reducing DAEs to ODEs; a more detailed discussion of index reduction can be found in the work of Laulusa and Bauchau [13,14].

2.1. Modifying the system model

The system model can be modified by adjusting the equations of motion directly [2,15], or by replacing a closed-loop multibody system with a similar open-loop system and superimposing the effects of the neglected components using tabulated data. An example of the latter can be found in the work of Sayers [16], where a simplified branched-topology vehicle model consisting of a sprung mass, an unsprung mass, and four linear tires is used for predicting the handling characteristics of passenger vehicles. This model was extended to include additional dynamic effects, and has evolved into the commercially-available CarSim software package [3]. Although CarSim supports most vehicle topologies and is capable of real-time performance, 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.

Force coupling is another modification technique that converts DAEs into ODEs. In this case, one joint in each kinematic loop is replaced with a virtual spring that approximately satisfies the constraints that were enforced by the removed joint. The stiffness of the virtual spring must be high enough to maintain an acceptably small distance between the disjointed frames. As the spring 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 [17].
2.2. Constraint stabilization

The objective of constraint stabilization techniques is to integrate derivatives of the kinematic constraints along with the ODEs of motion in a way that avoids the accumulation of position-level constraint violations. Baumgarte stabilization [5] replaces Eq. (2) with a linear combination of the position-, velocity-, and acceleration-level constraints:

$$\dot{\Phi} + 2\alpha_\Phi \Phi + \beta_\Phi^2 \Phi = 0.$$  \(\text{(3)}\)

Although it is relatively straightforward to implement, the effectiveness of this technique relies heavily on the judicious selection of stabilization parameters \(\alpha_\Phi\) and \(\beta_\Phi\). Parameters \(\alpha_\Phi = 1/h\) and \(\beta_\Phi = 1/h^2\), where \(h\) is the integration step size, are theoretically optimal but impractically large, as they introduce artificial stiffness into the system [18]. In practice, \(\alpha_\Phi\) and \(\beta_\Phi\) are usually chosen to be equal, thereby achieving critical damping and providing the fastest error reduction [14], with values typically between 1 and 20 [19]. Note that twice differentiating Eq. (2) 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. A system of ODEs can be obtained either by eliminating the Lagrange multipliers symbolically, or by solving them explicitly [20]:

$$\lambda = \left(\Phi_q M^{-1} \Phi_q^T\right)^{-1}\left(\Phi_q M^{-1} F - \gamma\right)$$ \(\text{(4)}\)

$$\dot{q} = M^{-1}\left(F - \Phi_q^T \lambda\right)$$ \(\text{(5)}\)

where \(\gamma\) is the right-hand side of the Baumgarte-stabilized constraints:

$$\dot{\Phi}_q \dot{q} = \gamma - 2\alpha_\Phi \Phi - \beta_\Phi^2 \Phi = \dot{\gamma}.$$ \(\text{(6)}\)

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 [6]:

$$M \dot{q} + \Phi_q^T \rho \left(\Phi_q \dot{q} + \Phi \dot{\Phi} + 2\omega \xi \Phi + \omega^2 \Phi\right) = F$$ \(\text{(7)}\)

where \(\rho, \omega,\) and \(\xi\) are diagonal matrices containing, respectively, the penalty numbers, natural frequencies, and damping ratios associated with each penalty system. 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 specified tolerance regardless of the size of the penalty values [19]:

$$\dot{q}^{(0)} = M^{-1} F$$ \(\text{(8)}\)

$$\dot{q}^{(k)} = \left(M + \Phi_q^T \rho \Phi_q\right)^{-1}\left(M \dot{q}^{(k-1)} - \Phi_q^T \rho \left(\Phi_q \dot{q} + \Phi \dot{\Phi} + 2\omega \xi \Phi + \omega^2 \Phi\right)\right), \quad k \geq 1$$ \(\text{(9)}\)

where the parenthetic superscript indicates the iteration number.

2.3. Solving the constraints separately

The final approach we shall consider involves solving the kinematic constraint equations separately from the projected dynamic equations. The embedding technique [11] is one form of projection, and can be described as a symbolic analogue of the null space formulation [13]. Once the generalized coordinates have been partitioned into independent and dependent sets (\(q_i\) and \(q_d\), respectively), the Lagrange multipliers and dependent accelerations can be eliminated from Eq. (1) to obtain a set of \(f\) second-order ODEs, one for each independent coordinate [21]:

$$q_i = \left(B^T MB\right)^{-1}\left(B^T F - B^T M \left(B q_i + C\right)\right),$$ \(\text{(10)}\)

Matrices \(B\) and \(C\) are defined as shown in the following velocity-level transformation equation:

$$q = \begin{bmatrix} -\Phi_d^{-1} \Phi_i & 1 \\ 0 & -\Phi_d^{-1} \Phi_i \end{bmatrix} q_i + \begin{bmatrix} -\Phi_d^{-1} \Phi_i \\ 0 \end{bmatrix} = B q_i + C$$ \(\text{(11)}\)

where \(\Phi_d = \partial \Phi / \partial q_d, \Phi_i = \partial \Phi / \partial q_i, \Phi_i = \partial \Phi / \partial t,\) and \(1\) is an identity matrix of dimension \(f\). The generalized coordinate partitioning technique [11] is similar, in that one ODE is obtained for each DOF, but the reduction is performed numerically rather than symbolically. Note that Eq. (10) is a function of \(q_i\) and \(q_d\). The dependent velocities \(q_d\) can be readily computed from \(q_d\) at each time step using Eq. (11); the dependent positions \(q_d\) can be determined from Eq. (2) using Newton’s method [22].
structure of the Jacobian $\Phi$, it may be possible to iterate over subsets of $\mathbf{q}$; as will be demonstrated in Section 4, this approach is generally faster than solving for all dependent coordinates simultaneously.

3. Gröbner basis triangularization

As discussed in Section 2, the embedding technique can be used to reduce a system of $n + m$ DAEs to a system of $f$ ODEs; however, nonlinear equations relating the independent and dependent positions must be solved since $\mathbf{q}$ appears implicitly in Eq. (10). Although Newton–Raphson iteration converges quadratically, explicit expressions for $\mathbf{q}$ as functions of $\mathbf{q}$ can result in more efficient simulation strategies that satisfy the constraints exactly, not simply to within a specified tolerance. Also note that Newton–Raphson iteration 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 [23]. Finally, note that real-time applications often demand the execution of a fixed number of arithmetic operations at each time step. Limiting the number of iterations can affect the accuracy of the simulation, while allowing the iteration to converge can violate the real-time constraint. It is, therefore, desirable to avoid iterative solution strategies whenever possible.

In contrast to other techniques for solving nonlinear systems, such as dialytic elimination, resultant methods, and polynomial continuation [24], a Gröbner basis [8] can be obtained algorithmically, and generates a system of equations with the same solutions as the original system. When generated in a particular manner, 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 as a canonical form for representing bases of ideals; however, it was the development of an algorithm by Buchberger [25] 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 [26], 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 [24]. 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 [27]. Detailed mathematical introductions to the theory of Gröbner bases [28,29] as well as more application-oriented presentations [8,27] are widely available.

Gröbner bases have been applied to problems in many diverse areas, including statistics, coding theory, and automated geometric theorem proving [30]. In the kinematics community, Gröbner bases have typically been used for performing off-line analyses of specific mechanisms, including planar $n$-bar mechanisms [31], planar and spatial manipulators [32], and spatial parallel robots [33–35]. A unique application was proposed by Dhingra et al. [36], 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. In this work, we focus on 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.

In order to generate a Gröbner basis, it is first necessary to convert the kinematic constraint equations involving trigonometric functions into polynomial equations. The tangent-half-angle [37] and Euler [38] substitution methods generally increase both the number and the total degree of terms in an expression, which is undesirable for generating a Gröbner basis. In this work, transformations of the form $s_\theta = \sin(\theta)$ and $c_\theta = \cos(\theta)$ are used, which necessitates the introduction of auxiliary equations $s_\theta^2 + c_\theta^2 - 1 = 0$. Although additional variables and equations are introduced, the resulting systems are of lower degree and have been found to be suitable for multibody applications [9].

Floating-point coefficients must also be addressed before generating a Gröbner basis. Since the coefficients appearing in the constraint equations are often computed from physical parameters that are only known approximately, an obvious approach is to use an algorithm that is capable of computing Gröbner bases with inexact coefficients. Unfortunately, there does not yet exist a generally accepted floating-point Gröbner algorithm that is capable of avoiding the stability issues inherent in floating-point arithmetic. Since physical parameters are typically only known to a limited precision, the approach adopted herein is to convert all coefficients into rational numbers using the convert/rational function in Maple (for example, 1.618 becomes 809/500).

A fundamental difference between Gaussian elimination and Gröbner bases is that the former typically uses pivoting to determine the elimination order, whereas the latter requires the specification of a term ordering to guide the reduction process. Using a pure lexicographic ordering with $x > y$, for example, power products in $x$ and $y$ are ordered as follows:

$$1 < y < y^2 < \ldots < x < xy < xy^2 < \ldots < x^2 < x^2y < x^2y^2 < \ldots$$

where order relations $>$ and $<$ are used to indicate the relative importance of eliminating each indeterminate or power product. To illustrate the effect of the term ordering, consider the set $F = \{f_1, f_2\}$ composed of the following bivariate polynomials:

$$f_1 = 4x^2y + 9xy + 4x - 7y - 4$$
$$f_2 = xy - y + 3$$
where we wish to solve $f_1 = 0$ and $f_2 = 0$ for $x$ and $y$. The resulting Gröbner basis $G$, generated relative to the term ordering given in Eq. (12), is as follows:

\[
\begin{align*}
g_1 &= 8x - 6y + 43 \\
g_2 &= 2y^2 - 17y + 8.
\end{align*}
\]

Rather than use $f_1(x, y)$ and $f_2(x, y)$ to solve for $x$ and $y$ simultaneously, $g_2(y)$ can be used to solve for $y$ first, and the solution can then be substituted into $g_1(x, y)$ to solve for $x$. A pure lexicographic ordering with $x < y$ directs the algorithm to favor the elimination of terms involving $y$ and, consequently, results in a triangular system in which $x$ is solved first.

In mechanical systems, the selection of a suitable term ordering can be facilitated by examining the topology of the system. Since a graph-theoretic formulation [10] is being used in this work, the topological graph is readily available. As will be demonstrated below, any pure lexicographic term ordering with $\mathbf{q}_1 \times \mathbf{q}_2$ will result in a triangular system in which the dependent generalized coordinates can be solved given values of the independent coordinates. Although it is not possible to predict which of these orderings will result in the most efficient system of equations for a particular mechanism, experience indicates that most orderings result in systems of comparable complexity. In this work, we simply arrange the terms in the same order as they are encountered when traversing the topological graph, starting at the independent coordinate closest to the ground node. The formulation procedure used herein can be summarized as follows:

1. Develop a model of the physical system of interest.
2. Determine an appropriate kinematic solution flow and the corresponding set of modeling coordinates [9].
3. Formulate the index-3 differential-algebraic equations of motion.
4. Select an appropriate set of independent coordinates.
5. Project the dynamic equations.
6. Triangularize the kinematic constraint equations:
   (a) Eliminate the floating-point coefficients and trigonometric terms.
   (b) Determine a suitable term ordering from the topological graph.
   (c) Generate a Gröbner basis using a pure lexicographic ordering.
   (d) Extract equations from the Gröbner basis to solve for the dependent coordinates recursively when given values of the independent coordinates.
7. Generate an optimized computation sequence in the target simulation language.

Since Gröbner bases can be generated algorithmically, this approach is suitable for use in automated formulation procedures and, as will be shown in the next section, can be used to generate real-time-capable dynamic simulation code.

4. Applications

This section first illustrates the application of the proposed approach to the kinematic and dynamic simulation of the Gough–Stewart platform and five-link suspension systems. The topological similarity of these mechanisms motivates their concurrent study. We also generate a fully triangular kinematic solution for an aircraft landing gear mechanism, which is not easily obtained manually. As will be demonstrated, the Gröbner basis approach is a promising technique for the automatic generation of simulation code that is more computationally efficient than that generated using existing iterative and constraint stabilization techniques. Note that the initial differential-algebraic equations of motion are generated using the Multibody library in MapleSim, which uses linear graph theory and symbolic computation to generate computationally efficient dynamic equations [39].

4.1. Kinematics and forward dynamics of a Gough–Stewart platform

We first consider the 6-DOF Gough–Stewart platform shown in Fig. 1, 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 testing the wear of vehicle tires [40], and has since become known for its spatial parallel robot was originally designed in 1947 for testing the wear of vehicle tires [40], and has since become known for its
z, \( \phi, \theta, \psi, \alpha_k, \beta_k, 1 \ldots 6 \) —that is, absolute coordinates to describe the motion of the platform and joint coordinates elsewhere, as suggested by Geike and McPhee [45]—we obtain \( m = n - f = 18\) constraint equations, three for each leg \( k \):

\[
\Phi_k = \begin{cases} 
    x - B_k \sin(\beta_k) + f_{x,k} & \\
    y - B_k \cos(\beta_k) + f_{y,k} & \\
    z - L_k \cos(\alpha_k) \sin(\beta_k) + f_{z,k} & 
\end{cases} = 0, \quad k = 1, \ldots, 6
\]  

(13)

where \( L_k \) is the total length of the \( k \)th leg, \( B_k = (B_{kx}, B_{ky}, 0) \) 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 \( \phi, \theta, \psi, \) and \( P_k \), the position of the \( k \)th spherical joint on the platform. Although the use of \( q_{\text{mixed}} \) results in more constraint equations and more independent loops than using \( q_{\text{joint}} \), the constraints are simpler [46] and, therefore, more suitable for generating a Gröbner basis. Choosing as independent coordinates \( q_k = (x, y, z, \phi, \theta, \psi) \), the three constraint equations associated with leg \( k \) involve three dependent coordinates: \( \alpha_k, \beta_k, \) 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 \( q_k \), which are expressed in first-order form. These equations can be integrated forward in time to determine the values of the independent coordinates; given \( q_k \), the constraint equations can be used to determine \( q_0 = (\alpha_1, \ldots, \beta_1, \ldots, s_1, \ldots, 6) \).

Since the constraint equations associated with leg \( k \) can be solved independently of the others, each independent loop is triangularized separately. In order to generate Gröbner bases, however, we must first transform the constraints into polynomial equations with rational coefficients. The system parameters are obtained from the work of Tsai [47], who provides geometric quantities to three decimal places. As such, the floating-point coefficients in Eq. (13) can be converted into rational numbers exactly using the convert/rational function in Maple. Next, transformations of the form \( s_k = \sin(\theta) \) and \( c_k = \cos(\theta) \) are used to eliminate the trigonometric functions; auxiliary equations \( s_0^2 + c_0^2 - 1 = 0 \) are introduced 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:

\[
\begin{align*}
    x = & \frac{17}{100} c_0 c_9 + \frac{119}{200} (c_9 c_6 - s_9 c_6) - \frac{2}{5} (c_9 c_6 - c_6 s_9) - (2 + s_1) s_{\nu_1} + \frac{53}{25} = 0, \\
    y = & \frac{17}{100} s_9 c_9 + \frac{119}{200} (s_9 c_6 + c_9 c_6) - \frac{2}{5} (s_9 c_6 - c_6 s_9) + (2 + s_1) s_{\nu_1} - \frac{687}{500} = 0, \\
    z = & \frac{17}{100} s_9 + \frac{119}{200} c_9 s_9 - \frac{2}{5} c_9 c_9 - (2 + s_1) s_{\nu_1} = 0, \\
    s_{\nu_1}^2 + c_{\nu_1}^2 - 1 = 0, \\
    s_{\nu_1}^2 + c_{\nu_1}^2 - 1 = 0, \\
\end{align*}
\]

where the values of independent coordinates \( q_k = (x, y, z, \phi, \theta, \psi) \) are known, and the values of dependent coordinates \( q_0 = (s_{\alpha_1}, c_{\alpha_1}, s_{\beta_1}, c_{\beta_1}, s_1) \) 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 \( k \):

\[
S_{\alpha_k} > C_{\alpha_k} > S_{\beta_k} > C_{\beta_k} > S_k > C_k > s_k > c_k > z > y > x
\]
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 h using the 64-bit version of Maple 14. A recursively solvable system of the following form is extracted from the 8th Gröbner basis:

\[
\begin{align*}
    s_k &= g_{1,k}(s_k, \mathbf{q}_i, \mathbf{q}_d, \mathbf{q}_i, \mathbf{q}_d) \\
    c_{k,b} &= g_{2,k}(s_k, \mathbf{q}_i, \mathbf{q}_d, \mathbf{q}_i, \mathbf{q}_d, z, y, x) \\
    s_{k,i} &= g_{3,k}(c_{k,i}, s_k, \mathbf{q}_i, \mathbf{q}_d, \mathbf{q}_i, \mathbf{q}_d, z, y, x) \\
    c_{k,b} &= g_{4,k}(s_{k,i}, c_{k,i}, s_k, \mathbf{q}_i, \mathbf{q}_d, \mathbf{q}_i, \mathbf{q}_d, z, y, x) \\
    s_{k,a} &= g_{5,k}(c_{k,a}, s_{k,i}, s_k, \mathbf{q}_i, \mathbf{q}_d, \mathbf{q}_i, \mathbf{q}_d, \mathbf{q}_d, \mathbf{q}_i, \mathbf{q}_d, z, y, x). 
\end{align*}
\]

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 Fig. 1, 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 $c_{k,b}$ 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 2.3, 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 Fig. 2. 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 2.2, are also evaluated. We use Baumgarte stabilization parameters $\alpha_0 = 1$ and $\beta_0 = 18$, which result in a peak constraint violation of $9.5 \times 10^{-6}$, the minimum that was found for $(\alpha_0, \beta_0 \in \mathbb{N}: 1 \leq \alpha_0, \beta_0 \leq 20)$. In the penalty formulation approach, we use one iteration with $p = 10^3 I$, $\xi = I$, and $w = 18 I$, which results in a peak constraint violation of $1.3 \times 10^{-5}$. 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` 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 [1]. The average computation time required for each simulated second is shown in Table 1. 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.
4.2. Kinematics and forward dynamics of a five-link suspension

We now consider the 1-DOF five-link suspension system, a schematic of which is shown in Fig. 3. The relatively large number of design parameters enables the five-link suspension to meet complex kinematic and dynamic performance requirements [48]. 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 ($\alpha_k$) and the rotated Y′-axis ($\beta_k$); 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 above, 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 obtained from the definition of suspension “S1” in the work of Knapczyk [49], 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 ($\delta$), camber ($\gamma$), and caster ($\tau$) angles of the wheel carrier are shown as functions of its vertical displacement ($z$) in Fig. 4. These results were obtained simply by iterating over the constraint equations described below, and agree with the results reported by Knapczyk [49]. 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 $8.0 \times 10^3$ [kg/m$^3$].

Many existing strategies for simulating multi-link suspensions employ techniques that modify the system model. Elmqvist et al. [50] discuss four common approaches:

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

\[ Table 1 \]

<table>
<thead>
<tr>
<th>Compiler</th>
<th>Simulation</th>
<th>Gröbner basis</th>
<th>Separate Newton iterations</th>
<th>Single Newton iteration</th>
<th>Baumgarte stabilization</th>
<th>Penalty formulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lcc</td>
<td>Kinematic</td>
<td>6.8 ms/s</td>
<td>11.9 ms/s</td>
<td>21.3 ms/s</td>
<td>91.7 ms/s</td>
<td>181.9 ms/s</td>
</tr>
<tr>
<td></td>
<td>Dynamic</td>
<td>19.4 ms/s</td>
<td>24.8 ms/s</td>
<td>34.0 ms/s</td>
<td>104.7 ms/s</td>
<td>193.8 ms/s</td>
</tr>
<tr>
<td>MS</td>
<td>Kinematic</td>
<td>2.7 ms/s</td>
<td>5.6 ms/s</td>
<td>9.2 ms/s</td>
<td>37.7 ms/s</td>
<td>73.8 ms/s</td>
</tr>
<tr>
<td></td>
<td>Dynamic</td>
<td>10.3 ms/s</td>
<td>13.3 ms/s</td>
<td>16.6 ms/s</td>
<td>45.3 ms/s</td>
<td>80.6 ms/s</td>
</tr>
</tbody>
</table>

Fig. 3. Five-link suspension.
Since low-order, fixed-step-size, non-stiff ODE solvers are typically used in real-time applications [1], 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 [23], trust region methods [51], or Newton–Raphson iteration.

Following the previous example, we use absolute coordinates to describe the motion of the wheel carrier and joint coordinates elsewhere—that is, \( \mathbf{q} = \{x, y, z, \phi, \theta, \psi, \alpha_1, ..., \alpha_n, \beta_1, ..., \beta_n\} \). In this case, \( m = n - f = 15 \) constraint equations are obtained, three for each link \( k \):

\[
\begin{align*}
L_k = f_k(x, y, z, \phi, \theta, \psi), & \quad k = 1, ..., 5. \\
\end{align*}
\]

where \( L_k \) is the length of the \( k \)th link, \( C_k = (C_{xk}, C_{yk}, C_{zk}) \) 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 Eq. (14) and the position constraints for the Gough–Stewart platform (Eq. (13)). 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-effectors of the Gough–Stewart platform have 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 \( \mathbf{q}_1 = \{z\} \) since each value of \( z \) corresponds to a unique value of \( x, y, \phi, \theta, \) and \( \psi \). The embedding technique is applied to obtain one second-order ODE for \( z \), 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 \( \mathbf{q}_{15} = \{x, y, \phi, \theta, \psi\} \). Finally, \( \mathbf{q}_{12} = \{\alpha_1, ..., \alpha_n, \beta_1, ..., \beta_n\} \) is computed recursively using the equations extracted from the Gröbner bases described below. The five equations used for calculating \( \mathbf{q}_{12} \) are obtained by isolating the terms involving \( \alpha_k \) and \( \beta_k \) on the left-hand side of Eq. (14) and squaring; the sum of these equations is a constant-distance constraint [52] of the following form:

\[
L_k^2 = f_k(x, y, z, \phi, \theta, \psi), \quad k = 1, ..., 5.
\]

Since the complexity of the constant-distance constraints precludes the generation of a Gröbner basis, Newton–Raphson iteration is used to calculate \( \mathbf{q}_{12} \) given \( \mathbf{q}_1 \). Nevertheless, the triangularization of \( \mathbf{q}_{12} \) 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 Eq. (14) 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 \( s_5^2 + c_5^2 = 1 = 0 \) need not be introduced for the known quantities \( \phi, \theta, \) and \( \psi \). We use the following pure lexicographic term ordering for loop \( k \):

\[
s_{\alpha_k} > c_{\alpha_k} > s_{\beta_k} > c_{\beta_k} > z > y > x > s_\theta > c_\theta > s_\phi > c_\phi > s_\psi > c_\psi.
\]
Table 2
Triangular system obtained for first link of five-link suspension.

<table>
<thead>
<tr>
<th>Polynomial</th>
<th>Degree</th>
<th>Number of terms</th>
<th>Longest coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>( g_{1,k}(c_i, q_i, q_{i1}) )</td>
<td>1 in ( c_{i1} )</td>
<td>8</td>
<td>15 digits</td>
</tr>
<tr>
<td>( g_{2,k}(s_i, c_i, q_i, q_{i1}) )</td>
<td>2 in ( s_{i1} )</td>
<td>28</td>
<td>29 digits</td>
</tr>
<tr>
<td>( g_{3,k}(c_i, s_i, c_i, q_i, q_{i1}) )</td>
<td>1 in ( c_{i1} )</td>
<td>6</td>
<td>15 digits</td>
</tr>
<tr>
<td>( g_{4,k}(s_i, c_i, s_i, c_i, q_i, q_{i1}) )</td>
<td>1 in ( s_{i1} )</td>
<td>8</td>
<td>15 digits</td>
</tr>
</tbody>
</table>

We again note that any term ordering of the form \( q_i q_j > q_i \) can be used. Each Gröbner basis is generated on one core of a 3.00-GHz Intel Xeon E5472 processor in about 45 min using the 64-bit version of Maple 14. A recursively solvable system of the following form is extracted from the \( k \)th Gröbner basis:

\[
\begin{align*}
    c_{ik} &= g_{1,k}\left(z, y, x, S_0, c_\beta, s_\gamma, c_\phi, s_\phi, c_\theta\right) \\
    s_{ik} &= g_{2,k}\left(c_{ik}, z, y, x, S_0, c_\beta, s_\gamma, c_\phi, s_\phi, c_\theta\right) \\
    c_{\alpha k} &= g_{3,k}\left(s_{ik}, c_{ik}, z, y, x, S_0, c_\beta, s_\gamma, c_\phi, s_\phi, c_\theta\right) \\
    s_{\alpha k} &= g_{4,k}\left(c_{\alpha k}, s_{ik}, c_{ik}, z, y, x, S_0, c_\beta, s_\gamma, c_\phi, s_\phi, c_\theta\right).
\end{align*}
\]

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 2 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 Fig. 3, where \( z = 0 \). Kinematic simulations are performed by driving the vertical displacement of the wheel carrier through the sinusoidal trajectory shown in Fig. 5(a); forward dynamic simulations are driven by applying to the wheel carrier the following \( C^0 \)-continuous, time-varying vertical force (in Newtons):

\[
F_z(t) = 183.6 + \begin{cases} 
1.84 \sin(\pi t), & \text{if } t < 1 \\
-2.38 \sin(\pi (t-1)/3), & \text{if } 1 \leq t < 4 \\
4.62 \sin(\pi t/2), & \text{if } 4 \leq t < 5 \\
-4.62 \cos(\pi t), & \text{if } 5 \leq t < 5.5 \\
6.50 \sin(2\pi t), & \text{if } t \geq 5.5
\end{cases}
\]

which results in a similar trajectory, as shown in Fig. 5(b). The first four solution approaches involve integrating the projected dynamic equations obtained using the embedding technique, described in Section 2.3, and solving the kinematics using the following methods:

2. Solving for \( q_{i1} \) iteratively using Eq. (15), then solving for \( q_{i2} \) 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;
   (c) Solving for all remaining independent coordinates in a single iterative procedure.

The dynamic solution flow for the Gröbner basis approach is shown in Fig. 6. 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 result in a peak constraint violation of \( 3.1 \times 10^{-6} \), 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 \( \rho = 10^3 I \), \( \xi = 7 I \), and \( \omega = 17 I \), which results in a peak constraint violation of \( 8.0 \times 10^{-5} \). Tolerances of \( 10^{-6} \) are used for the iterative \( \xi \) 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 \texttt{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 again use a first-order explicit Euler integration scheme. The average computation time required for each simulated second is shown in Table 3 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 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 $q_{d2}$ could be distributed over five processors.

4.3. Kinematics of an aircraft landing gear mechanism

Finally, we consider the mechanism shown in Fig. 7, which was designed for deploying and retracting landing gear at the nose of an aircraft [53]; the topology of the system is shown in Fig. 8, where the kinematics of the body-fixed reference frame at the head of each arrow are measured relative to that at its tail. The main strut (BAF) is pin-connected to the piston at point B, the fuselage at point A, and the lower link at point F. Revolute joints at points C, D, and E attach the drag strut (CDE) 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.

The 1-DOF aircraft landing gear mechanism consists of two independent loops, and has nearly the same topology as a Stephenson-III mechanism. The lower loop (ADEF) is a 1-DOF planar four-bar mechanism, whose kinematic equations could be solved first if one of its angles were driven. In this case, the actuation is applied to the upper loop (ABCD), 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 an unactuated 1-DOF planar four-bar.

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 may preclude the use of this strategy with large systems. Furthermore, as will be shown, we obtain a sextic equation for one variable, which is not generally solvable using a fixed number of arithmetic operations.

Table 3

<table>
<thead>
<tr>
<th>Compiler</th>
<th>Simulation</th>
<th>Gröbner basis</th>
<th>Separate iterations for each loop in $q_{d2}$</th>
<th>Single iteration for solving $q_{d2}$</th>
<th>Single iteration for solving $q_d$</th>
<th>Baumgarte stabilization</th>
<th>Penalty formulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>MS</td>
<td>Kinematic</td>
<td>5.5 ms/s</td>
<td>9.9 ms/s</td>
<td>28.1 ms/s</td>
<td>47.8 ms/s</td>
<td>28.5 ms/s</td>
<td>29.9 ms/s</td>
</tr>
<tr>
<td></td>
<td>Dynamic</td>
<td>15.5 ms/s</td>
<td>18.3 ms/s</td>
<td>32.6 ms/s</td>
<td>55.7 ms/s</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Using joint coordinates \( \mathbf{q} = [\theta_A, \theta_B, \theta_D, \theta_F, p_{BC}] \) and substituting the geometric parameters reported by Uchida [54], we obtain the following four constraint equations:

\[
\begin{align*}
\sin \theta_A \sin \theta_B p_{BC} - \cos \theta_A \cos \theta_B p_{BC} + 62 \sin \theta_A + 42 \cos \theta_A + \frac{1697}{100} \sin \theta_D + \frac{152}{25} \cos \theta_D + 231 &= 0 \\
\sin \theta_A \sin \theta_B p_{BC} + \cos \theta_A \cos \theta_B p_{BC} + 42 \sin \theta_A + 62 \cos \theta_A + \frac{152}{25} \sin \theta_D - \frac{1697}{100} \cos \theta_D - 62 &= 0 \\
\frac{892}{5} \sin \theta_F - \frac{892}{5} \cos \theta_F - 212 \sin \theta_A - 25 \cos \theta_A - \frac{822}{5} \sin \theta_D + 231 &= 0 \\
\frac{892}{5} \sin \theta_F + \frac{892}{5} \cos \theta_F + 25 \sin \theta_A - 212 \cos \theta_A - \frac{822}{5} \sin \theta_D - 62 &= 0
\end{align*}
\]

where \( s_k = \sin (\theta_k) \) and \( c_k = \cos (\theta_k) \). Note that Eqs. (16) and (17) represent the loop-closure constraints for the upper loop, and Eqs. (18) and (19) represent those for the lower loop. Since knowledge of the piston length \( p_{BC} \) leaves three unknowns in the upper loop (\( \theta_A, \theta_B, \) and \( \theta_D \)) and three unknowns in the lower loop (\( \theta_A, \theta_D, \) and \( \theta_F \)), neither of these pairs of equations can be solved in isolation. Generating a Gröbner basis using a pure lexicographic term ordering with \( c_F > s_F > c_A > s_A > c_D > s_D > c_B > s_B > p_{BC} \) results in the triangular system described in Table 4; 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 Fig. 7, 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 challenges associated with adopting such a strategy for larger systems.

Using joint coordinates \( \mathbf{q} = [\theta_A, \theta_B, \theta_D, \theta_F, p_{BC}] \) and substituting the geometric parameters reported by Uchida [54], we obtain the following four constraint equations:

\[
\begin{align*}
\sin \theta_A \sin \theta_B p_{BC} - \cos \theta_A \cos \theta_B p_{BC} + 62 \sin \theta_A + 42 \cos \theta_A + \frac{1697}{100} \sin \theta_D + \frac{152}{25} \cos \theta_D + 231 &= 0 \\
\sin \theta_A \sin \theta_B p_{BC} + \cos \theta_A \cos \theta_B p_{BC} + 42 \sin \theta_A + 62 \cos \theta_A + \frac{152}{25} \sin \theta_D - \frac{1697}{100} \cos \theta_D - 62 &= 0 \\
\frac{892}{5} \sin \theta_F - \frac{892}{5} \cos \theta_F - 212 \sin \theta_A - 25 \cos \theta_A - \frac{822}{5} \sin \theta_D + 231 &= 0 \\
\frac{892}{5} \sin \theta_F + \frac{892}{5} \cos \theta_F + 25 \sin \theta_A - 212 \cos \theta_A - \frac{822}{5} \sin \theta_D - 62 &= 0
\end{align*}
\]

where \( s_k = \sin (\theta_k) \) and \( c_k = \cos (\theta_k) \). Note that Eqs. (16) and (17) represent the loop-closure constraints for the upper loop, and Eqs. (18) and (19) represent those for the lower loop. Since knowledge of the piston length \( p_{BC} \) leaves three unknowns in the upper loop (\( \theta_A, \theta_B, \) and \( \theta_D \)) and three unknowns in the lower loop (\( \theta_A, \theta_D, \) and \( \theta_F \)), neither of these pairs of equations can be solved in isolation. Generating a Gröbner basis using a pure lexicographic term ordering with \( c_F > s_F > c_A > s_A > c_D > s_D > c_B > s_B > p_{BC} \) results in the triangular system described in Table 4; 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 Fig. 7, 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 challenges associated with adopting such a strategy for larger systems.
Table 4
Triangular system obtained for aircraft landing gear mechanism.

<table>
<thead>
<tr>
<th>Polynomial</th>
<th>Degree</th>
<th>Number of terms</th>
<th>Longest coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>(g_1(s_F,p_F))</td>
<td>6 in (s_B)</td>
<td>28</td>
<td>62 digits</td>
</tr>
<tr>
<td>(g_2(p_a,s_F,p_F))</td>
<td>1 in (p_B)</td>
<td>40</td>
<td>132 digits</td>
</tr>
<tr>
<td>(g_3(s_F,s_a,p_F,p_B))</td>
<td>1 in (s_D)</td>
<td>37</td>
<td>245 digits</td>
</tr>
<tr>
<td>(g_4(p_F,p_B,s_F,s_a,p_F))</td>
<td>1 in (s_B)</td>
<td>6</td>
<td>9 digits</td>
</tr>
<tr>
<td>(g_5(s_F,s_a,s_B,s_D,s_B,p_B))</td>
<td>1 in (s_A)</td>
<td>37</td>
<td>264 digits</td>
</tr>
<tr>
<td>(g_6(s_F,s_a,s_B,s_D,s_B,p_B))</td>
<td>1 in (s_C)</td>
<td>37</td>
<td>264 digits</td>
</tr>
<tr>
<td>(g_7(s_F,s_a,s_B,s_D,s_B,p_B))</td>
<td>1 in (s_F)</td>
<td>37</td>
<td>252 digits</td>
</tr>
<tr>
<td>(g_8(s_F,s_a,s_B,s_D,s_B,p_B))</td>
<td>1 in (c_F)</td>
<td>37</td>
<td>254 digits</td>
</tr>
</tbody>
</table>

5. Conclusions and future work

Gröbner bases have been used to generate recursively solvable systems from the constraint equations that arise in the modeling of closed-kinematic-chain systems. When coupled with the embedding technique, the Gröbner basis approach results in computationally efficient dynamic simulation code that avoids the use of iteration. In the case of the Gough–Stewart platform and aircraft landing gear mechanism, for which fully triangular solutions were obtained, the kinematic constraints can be satisfied exactly and in a fixed amount of time, two characteristics that are highly desirable in many applications. Although the five-link suspension system was not fully triangularized, a block-triangular form was obtained that is more efficient than existing iterative and constraint stabilization techniques.

The Gröbner basis approach is currently being applied to the simulation of a vehicle with double-wishbone suspensions on the front and rear axles. The resulting model will be implemented in a hardware- and operator-in-the-loop driving simulator for evaluating the effectiveness of various vehicle dynamics controllers, and for testing power management controllers for hybrid electric vehicles. Highly efficient code is required for simulating the vehicle dynamics in order to maintain real-time performance, which makes the Gröbner basis approach particularly attractive for this application. Finally, floating-point Gröbner basis algorithms are being investigated for alleviating the memory demands imposed by the growth of coefficients during the elimination procedure.

Acknowledgments

The authors wish to acknowledge the helpful suggestions of Dr. Chad Schmitke of Maplesoft, and the financial support of the Natural Sciences and Engineering Research Council of Canada.

References


