A Symbolic Approach to Compute a Null-Space Basis in the Projection Method

Mark Giesbrecht and Nam Pham

Abstract We present a hybrid symbolic-numeric approach for the so-called projection method for solving the parameterized differential-algebraic constraint equations associated with multibody mechanical systems. A primary problem in this approach is computing a null-space basis of a matrix of multivariate rational functions, the Jacobian of the symbolic constraint matrix. A purely symbolic approach is untenable in terms of the sheer size of the output, whereas a purely numerical approach does not offer the flexibility of leaving some or all parameters unspecified. Instead we propose a hybrid approach, which does a symbolic preconditioning, followed by representing the null-space basis by straight-line C code, i.e., a black-box null-space basis. We do this in a numerically sensitive way, and show that our black box is numerically robust at almost all parameter settings. This is verified by experimental results on inputs from typical multibody models.

1 Introduction

In recent years, a large amount of work has been devoted to developing symbolic-computation software for modelling and simulating physical systems; for example, MapleSim [15]. The primary motivation has been potential gains over a purely numerical approach that such software could provide engineers in the model-based development process. In this paper, we aim to use and optimize a symbolic approach in solving a constrained mechanical system using the so-called *projection method* [3].

M. Giesbrecht (✉) · N. Pham (✉)
Cheriton School of Computer Science,
University of Waterloo, N2L 3G1, Waterloo, ON, Canada
e-mail: mwg@uwaterloo.ca

N. Pham
e-mail: npham@uwaterloo.ca

R. Feng et al. (eds.), *Computer Mathematics*, DOI 10.1007/978-3-662-43799-5_19

Consider a constrained mechanical system represented by a set of m nonlinear algebraic equations:

$$\Phi(q, t) = 0, \tag{1}$$

where q is a set of n generalized coordinates, $q = (x_1, \ldots, x_n)$, and each constraint equation is a function of the coordinates q and the time t. The Jacobian is the $m \times n$ matrix of first partial derivatives of the constraint equations (1):

$$\Phi_q = \left[\frac{\partial(\Phi_1, \Phi_2, \cdots, \Phi_m)}{\partial(x_1, x_2, \cdots, x_n)}\right] = \begin{bmatrix} \frac{\partial \Phi_1}{\partial x_1} & \frac{\partial \Phi_1}{\partial x_2} & \cdots & \frac{\partial \Phi_1}{\partial x_n} \\ \frac{\partial \Phi_2}{\partial x_1} & \frac{\partial \Phi_2}{\partial x_2} & \cdots & \frac{\partial \Phi_2}{\partial x_n} \\ \cdots & \cdots & \ddots & \cdots \\ \frac{\partial \Phi_m}{\partial x_1} & \frac{\partial \Phi_m}{\partial x_2} & \cdots & \frac{\partial \Phi_m}{\partial x_n} \end{bmatrix}. \tag{2}$$

The equations describing the system dynamics in augmented form can then be obtained as

$$M\ddot{x} + \Phi_q^{\mathrm{T}}\lambda = F, \tag{3}$$

where M is an $n \times n$ symmetric generalized mass matrix, Φ_q is the $m \times n$ Jacobian of the constraint matrix Φ with respect to the generalized coordinates, and λ is the $(m \times 1)$ Lagrange multiplier. We will assume that the Jacobian matrix Φ_q above has rank r, which may not be full. This in itself is atypical—full rank is assumed in the standard situation [8]—but is important in the case of automatically generated equations where redundancies may lead to rank deficiencies.

We could solve the $n+m$ differential and algebraic equations formed by (1) and (3) for the coordinate q and the Lagrange multiplier λ by using the appropriate numerical methods. However, solving those nonlinear DAEs is computationally expensive, making it unsuitable for real-time simulation. Hence, it is desirable to express the system equations in a set of purely ordinary differential equations (ODEs).

Reference [2] proposed a new method, called the *projection method*, to eliminate algebraic equations from (3). Specifically, in this method, we find a null-space basis D for Φ_q, an $n \times r$ matrix such that

$$\Phi_q D = 0 \text{ or } D^{\mathrm{T}}\Phi_q^{\mathrm{T}} = 0. \tag{4}$$

By multiplying both sides of (3) by D^T, the resulting equation can be written in an embedded form:

$$D^{\mathrm{T}}M\ddot{x} = D^{\mathrm{T}}F. \tag{5}$$

The tangent velocity u in D is defined as

$$\dot{x} = Du. \tag{6}$$

The determination of the null-space D and its tangent velocity u allows us to analyze the constrained system as two separate problems. First, by substituting (6) into (5), we can derive the dynamic equations of the constrained system in the subspace defined by D:

$$D^{\mathrm{T}} M D\dot{u} = D^{\mathrm{T}}(F - M\dot{D}u). \tag{7}$$

Equations (1) and (7) form a system of $n+r$ purely differential equations in x and u, which can be easily solved using a standard differential equation solver. Given x_0 and v_0, we can compute the required initial tangent speeds for simulation. Secondly, the constraint reactions can be determined at any instant of the simulation by substituting (6) into (3):

$$\lambda = (\Phi_q M^{-1} \Phi_q^{\mathrm{T}})^{-1} \Phi_q (M^{-1} F - \dot{D}u). \tag{8}$$

Thus, the projection method requires an effective algorithm capable of finding a null-space basis D of a large and complicated matrix Φ_q. In the traditional setting, the matrices M and Φ_q are purely numerical, real valued matrices. However, in many applications, these equations depend upon unknown parameters, which would have to be instantiated before the projection method could be applied. A more interesting symbolic approach has been pursued by MapleSim [11, 15], where unknown parameters are simply left as parameters and the equations are handled symbolically, i.e., over the function field defined by the parameters. This has the advantage of completely reflecting knowledge (or lack thereof) of the underlying system, but of course can lead to massive expression swell. The approach we will take here is a hybrid one: from the symbolic system we will generate straight-line code to evaluate solutions to the symbolic system at *any* specific settings of the parameters. This can potentially be used in further manipulations, to reconstruct a symbolic solution (say via sparse interpolation), or as a plugin to a numerical solver. The code produced will (heuristically) be numerically robust in that, assuming reasonable properties of the symbolic system, the straight-line code will have good relative error at most specific parameters.

Numerically, the matrix D is often calculated via the Singular Value Decomposition (SVD) of the Jacobian matrix Φ_q [1]. However, from the symbolic modelling point of view, any numeric processes should be avoided since they require a repeated evaluation of the model. Moreover, a symbolic computation of the SVD (as a symbolic eigenvalue computation) will be both very large, and, when it is used for numerical evaluation, will not necessarily lead to a numerically robust algorithm.

On the other hand, classical symbolic methods (such as fraction-free Gaussian elimination; see, e.g., [4]) to compute the null-space basis will generally work only for small problem instances, or when the number of parameters is very small. The constraint matrix Φ_q in a multi-body system is often condensed with many multivariate functions. For any medium-sized model, symbolic manipulation easily leads to intermediate swell problems that exceed the capabilities of general purpose symbolic software like Maple and Mathematica [13]. To handle the intermediate swell problems, [16] proposed to use fraction-free factoring and implicit reduced involutive form to control the generation of large expression during computation [16, 17].

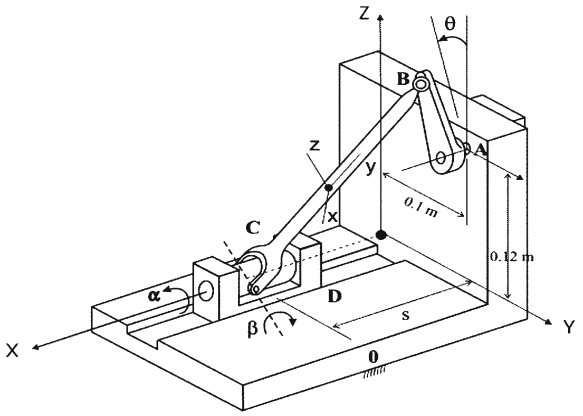

Fig. 1 Slider crank mechanism [7]

In this paper, we make preliminary steps towards a different approach. We aim to develop a hybrid symbolic-numeric computation technique to derive the orthogonal complement matrix D directly. First, we replace the SVD computation by Gaussian elimination with complete row and column pivoting. We then show how to efficiently find row and column permutations of Φ_q such that an LU decomposition of Φ_q proceeds without further pivoting (equivalently, we are determining an *a priori* pivoting strategy we hope is effective almost everywhere). From the permuted matrix, we can then generate straight-line C code to evaluate a clearly defined null-space basis at any settings of the parameters. The method has similarity to the so-called *static pivoting* approach of [10], where a good pivoting strategy is established in advance for the purposes of parallelization.

While we could, at this point, reconstruct a symbolic representation of the null-space (say with sparse interpolation—[5]), it is more likely that the black box is used directly in later numerical solvers. Our method has an important advantage that once the symbolic preconditioning is done, the generated code evaluates a single, unique basis for the null-space.

1.1 Example: Slider Crank Mechanism

In this section, we will introduce the simple slider crank mechanism, which will be used in the following sections to demonstrate the basic ideas of our approach.

As illustrated in Fig. 1 above, a slider-crank consists of three bodies: a crankshaft rotating around the fixed axis, a slider and a connecting rod. The system has one degree of freedom. It is modelled using four generalized coordinates $q = [\alpha, \beta, \theta, s]^T$, which are coupled by three algebraic constraints. Numerical values for system parameters are borrowed from [7].

The Jacobian of the constraint matrix is a 3×4 matrix. Each entry is a function of the coordinates. For example, Φ_q has second column

$$\Phi_q[*, 2] = \begin{bmatrix} -3/10 \cos(\beta) \\ -3/10 \sin(\beta)\cos(\alpha)\cos(\theta) - 3/10 \sin(\beta)\sin(\alpha)\sin(\theta) \\ 3/10 \sin(\beta)\cos(\alpha)\sin(\theta) - 3/10 \sin(\beta)\sin(\alpha)\cos(\theta) \end{bmatrix}.$$

It is convenient to convert all trigonometric functions (such as sine and cosine) into rational functions by using the tangent half-angle formula $\sin(\alpha) = \frac{2z}{1+z^2}$ and $\cos(\alpha) = \frac{1-z^2}{1+z^2}$ where $z = \tan(\frac{\alpha}{2})$, whence

$$\Phi_q[*; 2] = \begin{bmatrix} \frac{-3}{10} \cdot \frac{1-z_3^2}{1+z_3^2} \\ \frac{-3}{5} \cdot \frac{(1-z_2^2)z_3(1-z_1^2)}{(1+z_2^2)(1+z_3^2)(1+z_1^2)} - \frac{12}{5} \cdot \frac{z_2 z_1 z_3}{(1+z_2^2)(1+z_3^2)(1+z_1^2)} \\ \frac{6}{5} \cdot \frac{z_2 z_3(1-z_1^2)}{(1+z_2^2)(1+z_3^2)(1+z_1^2)} - \frac{6}{5} \cdot \frac{(1-z_2^2)z_3 z_1}{(1+z_2^2)(1+z_3^2)(1+z_1^2)} \end{bmatrix}.$$

2 Generating a Straight-Line Program to Compute a Null-Space Basis

Throughout the rest of this paper we will assume that we are working with a matrix A whose entries are rational functions with real (floating point) coefficients. That is, trigonometric and other functions have been replaced by independent indeterminates. This will be represented algebraically by the function field $\mathsf{F} = \mathbb{R}(z_1, z_2, \cdots, z_\ell)$.

Our goal in this section is to demonstrate how to take a matrix $A \in \mathsf{F}^{m \times n}$ and generate efficient code which, at "almost all" settings of the parameters $z_1, \ldots, z_\ell$ from $\mathbb{R}$, produces an evaluation of a specific basis of the null-space of A at those values. The two primary difficulties are that (i) there is no unique null-space basis for A, and (ii) we need to ensure that the output is numerically "good" for any setting of the parameters. A typical symbolic computation of the null-space will require us to make arbitrary decisions on pivots. Under any specialization of the values of $z_1, \ldots, z_\ell$ we must be sure to make the same choices, and hence evaluate the same null-space. We will assume for the remainder of this paper that A has rank r and that entries in A are rational functions whose numerators and denominators have degree less than d.

Our approach is to do a randomized symbolic preconditioning, after which a canonical basis of the null-space is completely determined, even under "almost all" specializations. The idea is similar to the static pivoting strategy of [10], though for entirely different purposes. Moreover, we heuristically proceed in a manner that our choice of pivots is numerically good at "most" numerical specializations.

2.1 Algebraically Effective Static Pivoting

We start by building a preconditioner for A so that all leading $k \times k$ minors are non-singular, for $1 \leq k \leq r$. Such a matrix will have a unique, *strict* LU-decomposition, with no need for pivoting choices (i.e., always use the diagonal element as a pivot). An efficient algorithm is provided to do the preconditioning, which involves no expression swell. This is equivalent to recording the pivots in a Gaussian elimination with complete pivoting and applying them every time (but for the efficiency of the generated code we will actually reorder the input matrix). Note that if we only computed the null-space *after* specializing the variables $(z_1, \ldots, z_\ell) \leftarrow (\alpha_1, \ldots, \alpha_\ell) \in \mathbb{R}^\ell$, it would be difficult to determine which basis to choose. In particular, it is important that we are evaluating a *specific* (symbolic) null-space basis at different points $(\alpha_1, \ldots, \alpha_\ell)$, and not simply evaluating a different null-space basis depending upon the parameters (for example, we could not interpolate a symbolic representation from the latter approach).

We start by observing some standard properties of the symbolic matrix A over the function field $\mathsf{F} = \mathbb{R}(z_1, \ldots, z_\ell)$.

Theorem 1 *Let $A \in \mathsf{F}^{m \times n}$ have $m \leq n$ and rank r, and be such that all the leading $k \times k$ minors of A are nonzero, for $1 \leq k \leq r$. Then there exists a* unique *lower triangular matrix $L \in \mathsf{F}^{m \times m}$, a* unique *upper triangular matrix $U \in \mathsf{F}^{n \times n}$, each with only ones on the diagonal, such that $LAU = D$, where $D = \mathrm{diag}(d_1, \ldots, d_r, 0, \ldots, 0) \in \mathsf{F}^{m \times n}$ is a (truncated) diagonal matrix. A basis for the null-space of A is formed by the last $n - r$ columns of U.*

Proof This follows from standard Gaussian elimination with complete (row and column) pivoting. See, for example [12], Sect. 3.10. □

The null-space basis given by the last $n - r$ columns of U in the above theorem is uniquely defined, even up to order; we will refer to it as the *canonical null-space* of A: $w_1, \ldots, w_{n-r} \in \mathsf{F}^{n \times 1}$.

Our first goal then is to find and fix permutation matrices P and Q such that the conditions of the theorem are met by PAQ. Our approach is a very simple randomization technique.

2.1.1 Algebraic Static Pivot Selection

(1) Choose "random" values $\alpha_1, \ldots, \alpha_\ell$ of parameters $z_1, \ldots, z_\ell$ from a finite subset $\mathscr{S} \subseteq \mathbb{C}$;
(2) Return P, Q such that $P \cdot A(\alpha_1, \ldots, \alpha_\ell) \cdot Q$ has an LU-decomposition (without pivoting), using Gaussian elimination with complete pivoting. This is equivalent to recording the choice of pivots used to do the Gaussian elimination with complete pivoting.

Since $P \cdot A(\alpha_1, \ldots, \alpha_\ell) \cdot Q$ has an LU-decomposition, all its leading minors are nonsingular, and hence this must be the case that PAQ is as well. The only thing that could go wrong is that the rank of $A(\alpha_1, \ldots, \alpha_\ell)$ drops below $\operatorname{rank} A$ at this evaluation point. The next theorem shows this happens rarely.

Theorem 2 *Suppose $A \in \mathbb{R}(z_1, \ldots, z_\ell)^{m \times n}$, with $m \leq n$, has rank r and the degrees of all numerators and denominators of entries are less than d. Let $\delta = \mathrm{nm}^2 d \cdot \xi$, for some user-chosen parameter ξ, and $\mathscr{S}$ a subset of $\mathbb{R}$ with at least δ numbers. If $\alpha_1, \ldots, \alpha_\ell$ are chosen uniformly and randomly from $\mathscr{S}$, then* $\operatorname{rank} A(\alpha_1, \ldots, \alpha_\ell) = \operatorname{rank} A$ *with probability at least* $1 - 1/\xi$.

Proof The product of all denominators in A has degree at most nmd, so we may assume that $A \cdot h \in \mathbb{R}[z_1, \ldots, z_\ell]^{m \times n}$ for some polynomial h of degree at most nmd, and $A \cdot h$ has total degree at most $nm(d+1)$. There exists a nonsingular $r \times r$ minor $\Delta \in \mathbb{R}[z_1, \ldots, z_\ell]$ of $A \cdot h$ which has degree at most $rnm(d+1)$. Thus, if we choose $\alpha_1, \ldots, \alpha_\ell$ such that $h(\alpha_1, \ldots, \alpha_\ell) \neq 0$ and $\Delta(\alpha_1, \ldots, \alpha_\ell) \neq 0$, $\operatorname{rank} A(\alpha_1, \ldots, \alpha_\ell) = \operatorname{rank} A$. It is easy to see that $\deg h + \deg \Delta \leq 3\mathrm{nm}^2 d$, so by the Schwarz–Zippel Lemma [14, 18], the probability that both $(h \cdot \Delta)(\alpha_1, \ldots, \alpha_\ell) \neq 0$ is at least $1 - (nmd + rnm(d+1))/\#S \geq 1 - 1/\xi$. □

A better probability estimate might be obtained by looking more carefully at the degrees of the entries of A and of the common denominator of its entries.

2.2 Numerically Effective Static Pivoting

Since we are concerned also about the numerical properties of our algorithm, it is not reasonable to assume that the choice of points will be effective at all settings of the parameters. To remedy this, we will be more numerically judicious in our choice of static pivots.

First, we recall the following easy theorem about the pivots, assuming that they are all on the diagonal.

Fact 3 Let $A \in \mathbb{R}(z_1, \ldots, z_\ell)^{m \times n}$ be such that all leading $i \times i$ minors are nonsingular, for $1 \leq i \leq m$. Suppose $A = LU$, where $L \in \mathbb{R}(z_1, \ldots, z_\ell)^{m \times m}$ is lower triangular with ones on the diagonal, and $U \in \mathbb{R}(z_1, \ldots, z_\ell)^{m \times n}$ is upper triangular. Then

$$U_{ii} = \det A \begin{bmatrix} 1 \ldots i \\ 1 \ldots i \end{bmatrix} \Big/ \det A \begin{bmatrix} 1 \ldots i-1 \\ 1 \ldots i-1 \end{bmatrix}.$$

In other words the pivots are quotients of minors of A. Whereas from a symbolic point of view, any nonzero pivot is sufficient for effective Gaussian elimination, numerically we want to choose the largest pivot (see [6], Sect. 3.4.8). Our challenge is to estimate the size of the minors/pivots *a priori*. Thus, we will attempt to estimate the "size" of the symbolic pivots via random evaluations. For this, we offer two

heuristics: evaluating at all but one variable for each variable in turn (to get an idea of the degree of the pivot), and evaluating at all variables to get an idea of coefficient size. All evaluations are on the unit circle which both supports our analysis and heuristically at least provides some numerical robustness.

2.2.1 Estimating the Degrees of the Chosen Pivots

We first show how to estimate the degree of an unknown, but evaluable, rational function through partial evaluation. This will applied to the pivots in our application. First, the degree of a *univariate* rational function is defined as the degree of the numerator minus the degree of the denominator. Next, the *maximum degree* and *maximum specialized degree* of a multivariate rational function at an evaluation point, respectively, are as defined as follows.

$$\operatorname{maxdeg} f(z_1, \ldots, z_\ell) = \max_i \deg_{z_i} f,$$

$$\operatorname{maxdeg}_{\alpha_1,\ldots,\alpha_\ell} f(z_1, \ldots, z_\ell) = \max \begin{cases} \deg_{z_1} f(z_1, \alpha_2, \ldots, \alpha_\ell), \\ \deg_{z_2} f(\alpha_1, z_2, \ldots, \alpha_\ell), \\ \quad \vdots \\ \deg_{z_\ell} f(\alpha_1, \alpha_2, \ldots, z_\ell). \end{cases}$$

We will regard the maximum degree as a good estimate of the overall size of f, and estimate it by showing that the maximum degree is equal to the maximum specialized degree at most specializations.

Theorem 3 *Let $f \in \mathbb{R}(z_1, \ldots, z_\ell)$ with* total *(of numerator or denominator) degree at most d. Choose a prime $p > 2\mu d\ell$, where $\mu > 1$ is a "confidence" parameter. Let $\omega \in \mathbb{C}$ be a primitive pth root of unity. Then, for uniformly and randomly chosen $e_j \in \{0, \ldots, p-1\}$ we have*

$$\deg_{z_i} f = \deg_{z_i} f(\omega^{e_1}, \ldots, \omega^{e_{i-1}}, z_i, \omega^{e_{i+1}}, \ldots, \omega^{e_\ell}),$$

for all $1 \le i \le \ell$, *with probability at least* $1 - 1/\mu$.

Proof For each i, let $g_i(z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_\ell)$ and $h_i(z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_\ell)$ be the leading terms of the numerator and denominator f, respectively, as a polynomials in z_i. Then the Schwartz–Zippel Lemma [14, 18] implies that the probability that $g_i(\omega^{e_1}, \ldots, \omega^{e_{i-1}}, \omega^{e_1}, \ldots, \omega^{e_{i-1}}) \neq 0$ and $h_i(\omega^{e_1}, \ldots, \omega^{e_{i-1}}, \omega^{e_1}, \ldots, \omega^{e_{i-1}}) \neq 0$ is at least $1 - d/(\mu d\ell)$. The probability that this is true for at each variable simultaneously is then at least $(1 - 1/(\mu\ell))^\ell > 1 - 1/\mu$. □

2.2.2 Estimating Coefficient Size

For an estimate of the coefficient size of an unknown (but evaluable) multivariate polynomial, the following is useful. First, for

$$f(z_1, \ldots, z_\ell) = \sum_i f_i z_1^{e_{i1}} z_2^{e_{i2}} \cdots z_\ell^{e_{i\ell}},$$

let $\|f\|_2 = \sqrt{\sum_i |f_i|^2}$, the coefficient 2-norm. The next theorem says the expected value of f at an appropriately random point on the unit circle closely approximates $\|f\|_2$ (see also Lemma 3.1 of [9]).

Theorem 4 *Let $f \in \mathbb{R}[z_1, \ldots, z_\ell]$ with $\deg_{z_i} f = d_i$ for $1 \le i \le \ell$. For each i fix a prime $p_i > d_i$, with $p_i \ne p_j$ for $i \ne j$, and let $\omega_i \in \mathbb{C}$ be a primitive p_ith root of unity. Then*

$$\|f\|_2^2 = E\Big[\, |f(\omega_1^{e_1}, \omega_2^{e_2}, \ldots, \omega_\ell^{e_\ell})|^2 : \text{for randomly chosen } 0 \le e_i < p_i \Big].$$

Proof We start by doing a Kronecker-like substitution to convert this to a univariate problem. Let $Q_1, \ldots, Q_\ell$ be the multipliers from the Chinese remainder theorem for $p_1, \ldots, p_\ell$, i.e., $Q_i \equiv 1 \mod p_i$ and $Q_i \equiv 0 \mod p_j$ for $j \ne i$. Let $P = p_1 p_2 \cdots p_\ell$; clearly $Q_i < P$ for $1 \le i \le \ell$. Now let $F(y) = f(y^{Q_1}, y^{Q_2}, \ldots, y^{Q_\ell})$. It is easy to see that there is a 1–1 mapping between nonzero terms in f to nonzero terms in F. Now let ζ be a Pth root of unity, so ζ^{P/p_i} is a p_ith primitive root of unity. Then

$$\|f\|_2^2 = \|F\|_2^2 = \frac{1}{P} \sum_{0 \le i < P} |F(\zeta^i)|^2 = \frac{1}{P} \sum_{0 \le i_1 < p_1} \cdots \sum_{0 \le i_\ell < p_\ell} |f(\omega_1^{i_1}, \cdots \omega_\ell^{i_\ell})|^2,$$

where the univariate equality follows from the fact that the Fourier matrix on powers of ζ is orthogonal. □

This theorem is somewhat weaker than we would like: the expected value being indicative of the size does not prove that a random value is indicative. But nonetheless, heuristically we can take this as a good estimate, especially if we make more than one random choice.

2.2.3 Numerical Static Pivot Selection

Suppose $A \in \mathbb{R}(z_1, \ldots, z_\ell)^{m \times n}$, where numerators and denominators have degree at most d_i in variable z_i, and total degree at most d. As above we may assume that we can clear denominators through a common denominator $h \in \mathbb{R}[z_1, \ldots, z_\ell]$ of degree at most nmd_i in each variable z_i, or nmd in total degree, though we will certainly

not compute this. Thus, a crude bound on the degree of all pivots is $nm^2(d_i + 1)$ in each variable, or $nm^2(d + 1)$ in total. Choose a prime $p_0 > 200\,nm^2(d + 1)$ and $\omega_0 \in \mathbb{C}$ a p_0th root of unity. Similarly, choose distinct primes $p_i > 2\,nm^2(d_i + 1)$, and let $\omega_i \in \mathbb{C}$ be an p_ith root of unity, for $1 \le i \le \ell$.

(1) Choose random evaluation points:

(1.1) For $1 \le j \le \ell$ choose a random $c_j \in \{0, \ldots, p_0 - 1\}$, and let $\alpha_j^{(0)} = \omega_0^{c_j}$;
(1.2) For $1 \le i \le 4$ and $1 \le j \le \ell$ choose a random $e_j^{(i)} \in \{0, \ldots, p_j - 1\}$, and let $\alpha_j^{(i)} = \omega_j^{e_j^{(i)}}$.

(2) Perform Gaussian elimination "simultaneously" with full pivoting on each of the following $(\ell + 4)$ matrices:

(2.1) $A(z_1, \alpha_2^{(0)}, \ldots, \alpha_\ell^{(0)})$, $A(\alpha_1^{(0)}, z_2, \ldots, \alpha_\ell^{(0)})$, ..., $A(\alpha_1^{(0)}, \alpha_2^{(0)}, \ldots, z_\ell)$;
(2.2) $A(\alpha_1^{(i)}, \alpha_2^{(i)}, \ldots, \alpha_\ell^{(i)})$, for $i \in \{1, \ldots, 4\}$.

(3) At each stage choose *the same pivot* for all the eliminations, using a pivot that has the highest maxdeg. In case of tie, we choose the one with the largest average evaluation in (2.2). If any choices are zero, or "too small", restart the entire process with different random choices.
(4) Record all the pivot choices and construct permutation matrices $P \in \mathbb{Z}^{m \times m}$, $Q \in \mathbb{Z}^{n \times n}$ such that PAQ has the same elimination with no pivoting.

The idea behind this heuristic strategy is that in each case we estimate the largest pivot as a rational function, first by its degree on each variable, then by its evaluation random points. In particular, by Theorem 3 we expect to get the degree correct with probability at least 99/100. In a tie for highest degree, we expect to find the "largest" pivot by Theorem 4.

In practice we might consider many practical refinements to this approach, especially taking into account the source of the parameters and their meaning in the problem space. For example, some parameters may be known to be large, or near zero. Some parameters may be more important or more numerically sensitive, and hence crucial to get correct. We can also increase the probability of success by considering more choices of random evaluations for our simultaneous Gaussian elimination. We hope to explore these further with application-driven examples, better probabilistic analyses, and better heuristics.

2.3 Example with Numerical Static Pivoting

Continuing the slider crank example from the previous section, we note that the maximum degree in many variable is 2, so we should use 3 distinct primes greater than 216 and proceed to evaluate at random roots of unity of those orders to get evaluations in (1.1) and (1.2).

For the sake of a simple example, we will instead choose $p_0 = 3$, $p_1 = 3$, $p_2 = 5$, and $p_3 = 7$, so $\omega_0 = \exp(2\pi i/3)$, $\omega_1 = \exp(2\pi i/3)$, $\omega_2 = \exp(2\pi i/5)$, and $\omega_3 = \exp(2\pi i/7)$. We then choose $c_1 = 2$, $c_2 = 1$, and $c_3 = 3$ to create three univariate matrices $\Phi_q(z_1, \omega_2^1, \omega_3^3)$, $\Phi_q(\omega_1^2, z_2, \omega_3^3))$, and $\Phi_q(\omega_1^2, \omega_2^1, z_3)$. These matrices are messy to write down, but we note their "degree matrices" as follows:

$$\begin{bmatrix} 0\ 2\ 0\ 2 \\ 2\ 2\ 2\ 0 \\ 2\ 2\ 2\ 0 \end{bmatrix}, \begin{bmatrix} 0\ 2\ 0\ 2 \\ 2\ 2\ 2\ 0 \\ 2\ 2\ 2\ 0 \end{bmatrix}, \begin{bmatrix} 0\ 2\ 0\ 2 \\ 2\ 1\ 2\ 0 \\ 2\ 1\ 2\ 0 \end{bmatrix}.$$

We then perform Gaussian elimination with complete row-column pivoting simultaneously on the three univariate matrices and four random evaluations of $\Phi_q(z_1, z_2, z_3)$:

$$\Phi_q(\omega_1^2, \omega_2^2, \omega_3^2) = \begin{bmatrix} 0.0 & 7.7405\text{e-}12 - 1.4447\text{e-}1i & 0.0 & 1.0 \\ -5.1923\text{e-}1 + 3.7140\text{e-}10i & 1.2421 - 8.6191\text{e-}10i & 3.9562\text{e-}1 - 8.7185\text{e-}2 & 0.0 \\ 3.5456\text{e-}10 + 5.3896\text{e-}1i & -8.5540\text{e-}10 - 1.19671i & -1.4832\text{e-}1 - 4.6630\text{e-}1i & 0.0 \end{bmatrix},$$

$$\Phi_q(\omega_1^1, \omega_2^3, \omega_3^6) = \begin{bmatrix} 0.0 & 4.8246\text{e-}11 - 1.3143i & 0.0 & 1.0 \\ 4.7239 + 1.7945\text{e-}9i & 5.0294 + 2.4527\text{e-}9i & -4.8475 + 8.7185\text{e-}2i & 0.0 \\ -1.7148\text{e-}9 + 4.9033\text{i} & -2.9437 + 4.8454\text{i} & -1.4832\text{e-}1 - 4.9760\text{i} & 0.0 \end{bmatrix},$$

$$\Phi_q(\omega_1^2, \omega_2^1, \omega_3^4) = \begin{bmatrix} 0.0 & 5.2880\text{e-}10 - 0.3761i & 0.0 & 1.0 \\ 4.4241 - 3.2567\text{e-}9i & 5.6790 - 4.7758\text{e-}9i & -4.1005 + 0.3693i & 0.0 \\ -3.3540\text{e-}9 + 4.4400i & -4.8105\text{e-}9 - 5.6586i & 0.3883 - 4.1323i & 0.0 \end{bmatrix},$$

$$\Phi_q(\omega_1^1, \omega_2^3, \omega_3^2) = \begin{bmatrix} 0.0 & 7.7405\text{e-}12 - 0.1444i & 0.0 & 1.0 \\ 0.5192 + 2.6403\text{e-}10i & 1.2421 + 4.1261\text{e-}10i & -0.6428 + 0.0871i & 0.0 \\ -1.9690\text{e-}10 + 0.5389i & -3.8619\text{e-}10 + 1.1967i & -0.1483 - 0.6116i & 0.0 \end{bmatrix}.$$

From this, we ultimately choose permutation matrices P, Q to maximize overall pivot size, in term of their *maxdeg* and then their coefficient sizes, as follows:

$$P = \begin{bmatrix} 0\ 1\ 0 \\ 0\ 0\ 1 \\ 1\ 0\ 0 \end{bmatrix}, \quad Q = \begin{bmatrix} 1\ 0\ 0\ 0 \\ 0\ 1\ 0\ 0 \\ 0\ 0\ 0\ 1 \\ 0\ 0\ 1\ 0 \end{bmatrix}.$$

2.4 Generating Straight-Line Code

Once we have identified permutation matrices P and Q such that PAQ has all leading minors nonsingular, and hence has a unique LU-decomposition as above, the computation of the canonical null-space $w_1, \ldots, w_{n-r} \in \mathsf{F}^{n\times 1}$ is completely determined. In particular, we can then generate straight-line code (i.e., without any conditional statements) which take parameters $\alpha_1, \ldots, \alpha_\ell \in \mathbb{R}$, and will produce $w_i(\alpha_1, \ldots, \alpha_\ell) \in \mathbb{R}^n$ for $1 \le i \le n - r$.

Theorem 5 *Let $A \in m \times n$, with $m \leq n$ and rank r. Suppose that we have determined permutation matrices $P \in \mathbb{Z}^{m\times m}$ and $Q \in \mathbb{Z}^{n\times n}$ such that all leading $k \times k$ leading minors of PAQ are nonsingular, so that the canonical null-space of PAQ is $w_1, \ldots, w_{n-r} \in \mathsf{F}^n$ Then we can generate straight-line (C) code which, given $\alpha_1, \ldots, \alpha_\ell \in \mathbb{R}$, produces*

$$w_1(\alpha_1, \ldots, \alpha_\ell) \in \mathbb{R}^n, \ldots, w_{n-r}(\alpha_1, \ldots, \alpha_\ell) \in \mathbb{R}^n.$$

Proof Assume the input in $\alpha_1, \ldots, \alpha_\ell \in \mathsf{R}$. The code to be generated is simply the code for LU-decomposition without pivoting on the matrix $(PAQ)(\alpha_1, \ldots, \alpha_\ell)$. The null-space basis is precisely the last $n - r$ columns of the computed U, evaluated at $\alpha_1, \ldots, \alpha_\ell$. □

Of course, the generated code is not foolproof. Aside from the (controllably small) probability that the rank of A is incorrectly computed through our randomized specialization (as in Theorem 2), there is also a probability that the instantiation of the parameters lies on a zero or pole of one of the pivots. This will be exceedingly rare, and will immediately cause a floating point exception, which is easy to catch. As in the proof of Theorem 2, any such point must be the root of a specific polynomial (defined by the input system) of degree $O(r^2nmd)$. Hence, the straight-line program will be defined almost everywhere.

Of perhaps greater concern is the numerical robustness of the evaluations provided by the straight-line program. For this our static pivot selection attempted to find pivots which were largest "most" of the time, by estimating the "size" of the pivot functions. Our method is only an effective heuristic, though we are working on stronger probabilistic guarantees and heuristic, as well as numerically robust "fallback" procedures (i.e., provide black boxes for two distinct bases, one of which is guaranteed to be valid and numerically robust everywhere the input is). Another possibility, explored by [10] is to do some number of steps of iterative refinement on the solution.

3 Implementation and Experimental Results

In this section, we will present some experiment results to illustrate the efficiency of our approach in terms of time and memory management, and to compare its performance with the purely symbolic null-space solver provided by Maple. Note that this comparison is, in some sense, quite unfair: the symbolic solution is a complete solution over the function field, whereas we produce a *black box* which can evaluate that solution. Nonetheless, since the ultimate application is usually to evaluate the solution, we believe this is useful. Moreover, as we will see below, the complete symbolic solution is unattainable for even moderately sized problems.

All experiments are run on Intel Xeon E7330 4 Quad Core CPUS (16 CPUs in total) with 128 GB of RAM. Since the goal of our experiment is to evaluate the usefulness of our technique in real applications, we choose a test set of 4 typical

Table 1 Multibody models from MapleSim

Models	Number of coordinates	Number of constraints	Number of parameters
3D Rigid Slider Crank	4	3	3
Planar Seven Body Mechanism	7	6	7
Quadski Turning	19	11	16
Hydraulic Stewart Platform	24	18	41

Table 2 Running time (in seconds)

Models	Maple's nullspace	Our nullspace
3D Rigid Slider Crank	0.046	0.016
Planar Seven Body Mechanism	0.078	0.031
Quadski Turning	Timeout (>200 s)	0.56
Hydraulic Stewart Platform	Timeout (>200 s)	1.64

multibody models obtained from MapleSim. Refer to Table 1 for the description of each model.

3.1 Time Efficiency

We measured the running time of our algorithm and Maple's NullSpace function to give an idea of the overall performance. Since our algorithm is heuristic in choosing the pivots, its execution time was measured to the average of 5 rounds for each of the models. We also set a time limit of 200 seconds on every call. Table 2 presents the comparison of running time between our algorithm and Maple's NullSpace solver.

We also compare the running time of our algorithm with different numbers of parameters on a same model. Figure 2 shows its performance on the Hydraulic Stewart Platform model. Starting with 41 parameters from the original model, we reduce the number of parameters by substituting numerical values into each of them. When the number of symbolic parameters is reduced to zero, our algorithm is basically run on a purely numerical matrix. It's also worth noting that when the same experiment is performed using Maple's NullSpace, it fails to return an answer in a reasonable time for more than 3 parameters.

Of course, we are not really comparing similar quantities; on one hand Maple is generating a complete solution, whereas we are simply generating C code which

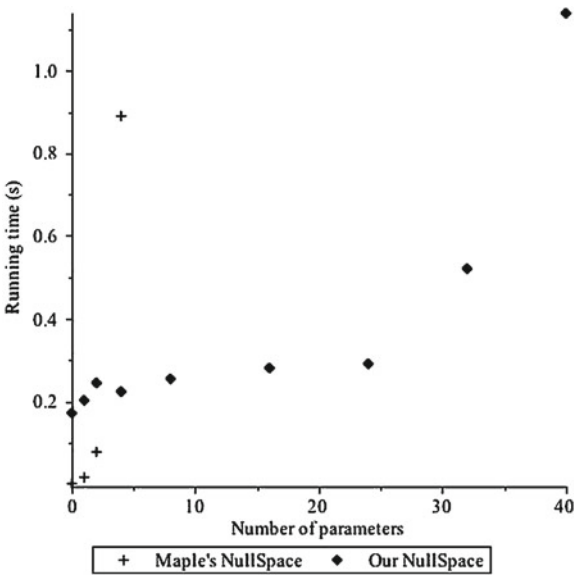

Fig. 2 Running time on hydraulic stewart platformwith different numbers of parameters

can evaluate one specialized solution (in fact the Maple code is remarkably fast in small instances). Nonetheless, the results clearly show that our generated C code is substantially faster than the NullSpace solver in Maple. Much more importantly, we are able to find a null-space basis far beyond when the computing the symbolic solution fails.

3.2 Memory Use

A primary difficulty with the symbolic computation is the need to produce a complete (and very large) presentation of the null-space, and also the need to work with these large entities during the Gaussian elimination. As the size of the intermediate result and the output can grow exponentially, it may well exhaust memory even with a relatively small input, and in particular one which would be quickly evaluable at any specialization of the parameters. We now examine the expected size of the null-space basis that our algorithm generates in term of the length of directed acyclic graph (DAG), which is Maple's internal representation for straight-line code:

Table 3 shows a weakness of our algorithm. Without doing any simplification, the size of the expression in our null-space basis will expand dramatically. However, there is at least some potential for identifying common subexpressions, computing them once, and then re-using these evaluations. While opportunities may be few in a truly generic matrix, in the potentially structured (or at least someone redundant)

Table 3 DAG size of the null-space of Φ_q

Models	Φ_q dimensions	DAG size of D
Planar Slider Crank	3×4	5671
Planar Seven Body Mechanism	6×7	75045
Quadski Turning	11×19	41706824
Hydraulic Stewart Platform	18×24	11849101

matrices from kinematic equations, there may well be many opportunities for such simplifications. Such simplification is, unfortunately, quite nontrivial in general, and a topic of current research.

3.3 Code Generation

A strength of our approach is that the null-space is ultimately computed by straight-line C code. This lends itself to highly optimized compilation. As the null-space is used to computed to the mass matrix $\tilde{M}$ and the force $\tilde{D}$ during simulation and control, it is desirable to port its evaluation to C for numerical computation. The intermediate "symbolic" representation of the straight-line code also has the potential for some optimization. The Maple CodeGeneration package with the *"optimize"* option in Maple can identify at least some common subexpressions while generating the code [13]. For example, in the case of the slider crank mechanism, the null-space basis of Φ_q could be simplified and converted to the following C code for further computation:

```
CodeGeneration[C](NullSpaceD) = t1 = pow(x[2], 0.2e1);
t2 = 0.1e1 + t1; t3 = pow(x[0], 0.2e1); t4 = x[0] * t2;
t5 = 0.5e1 * t4 + 0.3e1 * (-0.1e1 + t3) * t1 + 0.3e1 * t3 - 0.3e1;
t6 = -0.1e1 + t1; t5 =  0.1e1 / t5; t7 = 0.1e1 / t6;
t8 = -0.3e1 + (0.5e1 + 0.3e1 * x[0]) * x[0]; t2 = 0.1e1 / t2;
t8 = 0.1e1 / t8;
cg17[0][0] = (-0.20e2 * t3 + 0.20e2 * t1 + 0.12e2 * t4
                + 0.10e2 * t1 * t3-0.10e2) * x[2] * t5* t7;
cg17[1][0] = 1; cg17[2][0] = (0.15e2 + 0.15e2 * t3) * x[2] * t8 * t2;
cg17[3][0] = -0.3e1 / 0.10e2 * t6 * t2;
```

A more thorough examination of the preconditioned LU-decomposition we employ, or use of some other equivalent method, will hopefully expose more optimization opportunities.

4 Conclusions and Future Work

In this paper, a new hybrid algorithm to compute the null-space basis of a multivariate matrix has been presented, and employed in the symbolic version of the projection method for solving a constrained mechanical system. The novelty of this algorithm lies in the combination of numeric and symbolic computation to generate fast "black box" code, which can be employed later for numerical or symbolic evaluation. Specifically, instead of computing the null-space directly using LU-decomposition, it is shown that by choosing the ordering of row and column interchanges "randomly", using numerical values from a constrained set, we can emit code to evaluate a symbolic null-space quickly. This is similar to so-called static pivoting schemes. This avoids many of the problems of intermediate expression swell encountered in purely symbolic approaches. Preliminary experiments have been presented to show that this approach is significantly faster than computing null-space symbolically, supporting the use of symbolic computation in engineering problems such as the projection method for multibody systems.

So far we have only done a pilot experiment with small to medium-sized multibody models. It is necessary to test our algorithm with larger models with tens or hundreds of parameters and constraints. There are also several avenues of algorithmic development. First, in the current implementation, we rely on Maple's CodeGeneration package to simplify the straight-line code at the the final step. A more careful analysis of the actual LU-decomposition or equivalent method should lead to identification of common subexpressions. Also, the ultimate "output" of our techniques is C code for a numerical evaluator at specialized values of the parameters. Currently, the numerical robustness of this code is relatively heuristic, estimating the static pivots through randomized evaluation. While we believe this is a good approach, a more rigorous treatment is warranted. Following [10], we will also investigate *post hoc* iterative refinement in cases when numerical stability is questionable.

Acknowledgments The authors thank Dr. Jürgen Gerhard, Maplesoft Inc., for the motivating problem and his many helpful remarks. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) and MITACS Canada.

References

1. Aghili, F., Piedbœuf, J.-C.: Simulation of motion of constrained multibody systems based on projection operator. Multibody Sys. Dyn. **9**(3), 283–309 (2003)
2. Arczewskia, K., Blajer, W.: A unified approach to the modelling of holonomic and nonholonomic mechanical systems. Math. Model. Syst. **2**(3), 157–174 (1996)
3. Blajer, W.: A projection method approach to constrained dynamic analysis. J. Appl. Mech. **59**(3), 643 (1992)
4. Geddes, K.O., Czapor, S.R., Labahn, G.: Algorithms For Computer Algebra. Kluwer Academic Publishers, Alphen aan den Rijn (1992)
5. Giesbrecht, M., Labahn, G., Lee, W-s: Symbolic-numeric sparse interpolation of multivariate polynomials. J. Symbolic Comput. **44**, 943–959 (2009). doi:10.1016/j.jsc.2008.11.003

6. Golub, G., Van Loan, C.: Matrix Computations. Johns Hopkins University Press, Baltimore (1996)
7. Haug, E.J.: Computer Aided Kinematics and Dynamics of Mechanical Systems. volume 1 of Allyn and Bacon series in engineering. Prentice Hall College Div, Upper Saddle River (1989)
8. Jalón, J.G., Bayo, E.: Kinematic and Dynamic Simulation of Multibody Systems: The Real-Time challenge. Springer, New York (1994)
9. Kaltofen, E., Yang, Z., Zhi, L.: On probabilistic analysis of randomization in hybrid symbolic-numeric algorithms. In: Proceedings of SNC'07, pp. 11–17 (2007)
10. Li, X.S., Demmel, J.W.: Making sparse gaussian elimination scalable by static pivoting. In: Proceedings of Supercomputing '98, pp. 1–17 (1998)
11. McPhee, J., Schmitke, C., Redmond, S.: Dynamic modelling of mechatronic multibody systems with symbolic computing and linear graph theory. Math. Comput. Model. Dyn. Syst. **101**(1), 1–23 (2004)
12. Meyer, C.D.: Matrix Analysis and Applied Linear Algebra. SIAM (2000)
13. Moore, B., Piedbœuf, J.-C., Bernardin, L.: Maple as an automatic code generator? Maple Summer Workshop, (2002)
14. Schwartz, J.T.: Fast probabilistic algorithms for verification of polynomial identities. J. Assoc. Comput. Mach. **27**, 701–717 (1980)
15. Waterloo Maple Inc., MapleSim User's Guide. (2011). http://www.maplesoft.com/view.aspx?SF=122742/387839/MapleSimUserGuid.pdf
16. Zhou, W., Jeffrey, D.J., Reid, G.J., Schmitke, C., McPhee, J.: Implicit reduced involutive forms and their application to engineering multibody systems. In IWMM/GIAE, pp. 31–43 (2004)
17. Zhou, W., Carette, J., Jeffrey, D.J., Monagan, M.B.: Hierarchical representations with signatures for large expression management. Proceedings of Artificial Intelligence and Symbolic Computation, Lecture Notes in Computer Science 4120, (2006)
18. Zippel, R.: Probabilistic algorithms for sparse polynomials. In: Proceedings of EUROSAM 79, pp. 216–226, Marseille (1979)