Symbolic-Numeric Computation
D. Wang and L. Zhi, Eds.
Trends in Mathematics, 335–347
© 2007 Birkhäuser Verlag Basel/Switzerland

Symbolic Computation Sequences and Numerical Analytic Geometry Applied to Multibody Dynamical Systems

Wenqin Zhou, David J. Jeffrey and Greg J. Reid

Abstract. The symbolic-numeric computing described here consists of an extensive symbolic pre-processing of systems of differential-algebraic equations (DAE), followed by the numerical integration of the system obtained. The application area is multibody dynamics. We deal symbolically with a DAE system using differentiation and elimination methods to find all the hidden constraints, and produce a system that is leading linear (linear in its leading derivatives). Then we use LU symbolic decomposition with Large Expression Management to solve this leading linear system for its leading derivatives, thereby obtaining an explicit ODE system written in terms of computation sequences obtained from using the MAPLE package `LargeExpressions`. Subsequently the Maple command `dsolve` is applied to this explicit ODE to obtain its numeric solution. Advantages of this strategy in avoiding expression explosion are illustrated and discussed. We briefly discuss a new class of methods involving Numerical Algebraic and Analytic Geometry.

Mathematics Subject Classification (2000). Primary 70E55; Secondary 68U01; Tertiary 15A09.

Keywords. LU symbolic decomposition, large expression management, differential elimination, DAE, computation sequence, straight line programme, hidden constraint, computer algebra.

1. Introduction

In the study of multibody dynamics, both purely symbolic and purely numeric methods separately suffer drawbacks. The symbolic methods encounter large expressions and the numerical methods are very slow in formulating the system. Examples of the applications of multibody dynamics are robot arms, vehicle suspensions, and automatic barriers. A modern robot arm, such as the Space Shuttle's Remote Manipulator System (RMS, or Canadarm) consists of several rigid bodies (the arm segments and the end effectors) linked by joints. Therefore the systems have at least 7 degrees of freedom, and typically more. In order to simulate the behaviour of systems like these, a number of programs have been developed that automatically generate the equations of motion for the system from its engineering description [3]. Several computer-algebra based packages have been developed, for example DYNAFLEX and SYMOFROS, that will generate the equations of motion in symbolic form. At present, these equations are handed over to purely numerical systems for integration as soon as the symbolic system has generated the equations.

What is described in this paper is a closer coupling of the symbolic analysis of the mechanical system and the final numerical integrations used by a simulation. The coupling takes the form of a symbolic pre-processing of the equations that improves the efficiency of their numerical solution. This increased interaction between the symbolically based part of the computation and the numerical part deserves to be called a "symbolic–numeric" calculation, we feel, even though the calculation is quite different from the numerical polynomial algebra [4] that has so far been the subject covered by the description "symbolic–numeric computation".

The equations describing the dynamics of a multibody system take the general form

$$M(t,q,\dot{q})\ddot{q} + \Phi_q^T \lambda = F(t,q,\dot{q}) \ , \tag{1}$$

$$\Phi(t,q) = 0 \ . \tag{2}$$

Here, q is a vector of generalized co-ordinates, $M(t,q,\dot{q})$ is the mass matrix, Φ is a vector of the constraint equations and λ is a vector of Lagrange multipliers [1, 2]:

$$\Phi = \begin{bmatrix} \Phi^1 \\ \vdots \\ \Phi^m \end{bmatrix} , \qquad \Phi_q = \begin{bmatrix} \Phi^1_{q_1} & \cdots & \Phi^1_{q_n} \\ \vdots & \vdots & \vdots \\ \Phi^m_{q_1} & \cdots & \Phi^m_{q_n} \end{bmatrix} , \qquad \lambda = \begin{bmatrix} \lambda^1 \\ \vdots \\ \lambda^m \end{bmatrix} .$$

The Lagrange multipliers λ can be interpreted as the forces that the joints must exert between the elements in order to enforce the geometrical constraints they impose.

These symbolic models are too complicated to be solved symbolically, both because of their nonlinearity and because of the number of equations. However there is still the possibility of symbolically pre-processing them before attempting a numerical solution. It can be noted from the general form of the equation system (1)–(2) that it is differential-algebraic (DAE), with the numerical difficulties that that entails. Suitable objectives for the pre-processing include the simplification of the system and the determination of all constraints in order to facilitate the subsequent numerical simulation of the system. More specifically, we show below that we can convert the system to a purely differential one, and moreover separate the constraint forces λ from the simulation variables q. Our primary tool for this is the RIFSIMP package, which uses differentiation and elimination methods to simplify any over-determined polynomially nonlinear PDE or ODE system and to return a canonical differential form [7].

Direct application of the RifSimp package to multibody systems reveals that it has difficulty handling the large systems generated by Dynaflex. We use Implicit Reduced Involutive Form (IRIF) to assist in alleviating such large expression swell problems [10]. Implicit RIF form is converted to RIF form by symbolically solving the IRIF for its leading linear derivatives. In particular we use symbolic LU decomposition with large expression management to obtain RIF form, expressed in terms of computation sequences.

In this paper, we use the simple example of a two-dimensional slider crank to illustrate our approach to the symbolic-numeric solving of this kind of DAE system. After receiving the DAE model of the slider crank from Dynaflex we first use implicit RifSimp to compute all the hidden constraints of this higher index DAE, by reducing it to an index one or zero DAE system. The details are given in Sects. 2 and 3. Then in Sect. 4, we use symbolic LU matrix factoring with large expression management to get the canonical form for the differential equations, which helps the numerical integration. Using these symbolic equations with computation sequences, we illustrate the numerical simulation in Sect. 5. Finally in Sect. 6, we discuss some new ideas for solving the constraints to obtain consistent initial values for the integration. It should be emphasized that each of the sub-tasks described here are fully algorithmic and automated computations. Their integration into a single piece of software is in principle straightforward.

2. Two-Dimensional Slider Crank

Space limitations prevent us from describing a realistic mechanical system, both because the system description would take up significant space, and also because the equations generated by Dynaflex would by too lengthy for the reader to follow. The two-dimensional slider crank is a simple example of a closed-loop system with $q^T = (\theta_1, \theta_2)^T$ where $\theta_1 = \theta_1(t)$ and $\theta_2 = \theta_2(t)$ are the angles shown in Fig. 1. The system is given by (1)–(2) where:

$$M = \begin{bmatrix} l_1^2(\frac{1}{4}m_1 + m_2 + m_3) + J_1 & -l_1 l_2 \cos(\theta_1+\theta_2)(\frac{1}{2}m_2 + m_3) \\ -l_1 l_2 \cos(\theta_1+\theta_2)(\frac{1}{2}m_2 + m_3) & l_2^2(\frac{1}{4}m_2 + m_3) + J_2 \end{bmatrix}, \quad (3)$$

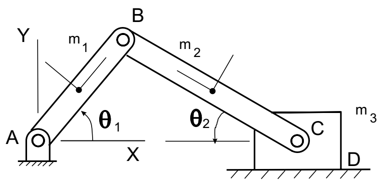

Figure 1. The two-dimensional slider crank. The arm of length l_1 and mass m_1 rotates while the mass m_3 attached to the end of the arm of length l_2 moves left and right. Each arm has mass m_i and moment of inertia J_i, for $i = 1, 2$.

$$F = \begin{bmatrix} -l_1 g(\frac{1}{2}m_1 + m_2 + m_3)\cos\theta_1 - l_1 l_2 \dot{\theta_2}^2 \sin(\theta_1+\theta_2)(\frac{1}{2}m_2 + m_3) \\ l_2 g(\frac{1}{2}m_2 + m_3)\cos\theta_2 - l_1 l_2 \dot{\theta_1}^2 \sin(\theta_1+\theta_2)(\frac{1}{2}m_2 + m_3) \end{bmatrix}, \quad (4)$$

and there is a single constraint equation between the angles:

$$\Phi = l_1 \sin\theta_1 - l_2 \sin\theta_2 = 0 \ . \quad (5)$$

Therefore, $\Phi_q^T \lambda$ in (1) is given by $\Phi_q^T\lambda = \begin{pmatrix} l_1\cos\theta_1 \\ -l_2\cos\theta_2 \end{pmatrix}\lambda$. Note that in this example, λ is a scalar. Thus, in addition to generating the constraint (5), DYNAFLEX automatically generated the constraint force $\lambda(t)$. For this example, the challenge now is to analyze the equations with computer algebraic methods such as RIFSIMP.

3. Implicit Reduced Involutive Form

We use implicit RIFSIMP to get all hidden constraints in the multibody dynamical general model (1)–(2).

Definition [Implicit Reduced Involutive Form]. Let $\prec$ be a ranking. Then a system $L = 0, N = 0$ is said to be in implicit reduced involutive form if there exist derivatives $r_1, \dots, r_k$ such that L is leading linear in $r_1, \dots, r_k$ with respect to $\prec$ (i.e. $L = A[r_1, \dots, r_k]^T - b = 0$) and

$$[r_1, \dots, r_k]^T = A^{-1}b \ , \quad N = 0 \ , \quad \det(A) \neq 0 \quad (6)$$

is in reduced involutive form.

This form is of interest because computing A^{-1} symbolically in practice can be very expensive. Sometimes implicit RIF-form can be obtained very cheaply, just by appropriate differentiation of the constraints.

To convert a system of general form (1), (2) with non-trivial constraints to implicit reduced involutive form, one would have to at least differentiate the constraints twice [10]. Carrying this out we obtain

$$\begin{aligned} M\ddot{q} + \Phi_q^T\lambda &= F(t,q,\dot{q}), & (7)\\ D_t^2\Phi = \Phi_q\ddot{q} + H\dot{q} + 2\Phi_{tq}\dot{q} + \Phi_{tt} &= 0, & (8)\\ D_t\Phi = \Phi_q\dot{q} + \Phi_t &= 0, & (9)\\ \Phi(t,q) &= 0, & (10) \end{aligned}$$

where $\Phi_{tq} = \frac{\partial \Phi_q}{\partial t}$, $\Phi_{tt} = \frac{\partial^2\Phi}{\partial t^2}$ and

$$H = \begin{bmatrix} \sum_i \Phi^1_{q_1 q_i}\dot{q}_i & \cdots & \sum_i \Phi^1_{q_n q_i}\dot{q}_i \\ \vdots & \vdots & \vdots \\ \sum_i \Phi^m_{q_1 q_i}\dot{q}_i & \cdots & \sum_i \Phi^m_{q_n q_i}\dot{q}_i \end{bmatrix}.$$

We now show:

Theorem [10]. *Consider the ranking $\prec$ defined by $q \prec \dot q \prec \lambda \prec \ddot q \prec \dot\lambda \prec \dddot q \prec \cdots$ where the dependent variables q, λ are ordered lexicographically $q_1 \prec q_2 \prec \cdots$ and $\lambda_1 \prec \lambda_2 \prec \cdots$. The system* (7), (8), (9), (10) *is in implicit* RIF*-form with A, b, $[r_1, \ldots, r_k]^T$ in the definition above given by*

$$A = \begin{bmatrix} M & \Phi_q^T \\ \Phi_q & 0 \end{bmatrix}, \quad b = \begin{bmatrix} F(t,q,\dot q) \\ -H\dot q - 2\Phi_{tq}\dot q - \Phi_{tt} \end{bmatrix}, \quad [r_1, \ldots, r_k]^T = \begin{pmatrix} \ddot q \\ \lambda \end{pmatrix} \tag{11}$$

and $N = \{\Phi = 0,\ \Phi_q \dot q + \Phi_t = 0\}$, $\det(A) \neq 0$.

Applying this to the slider crank, with the ranking $\theta_1 \prec \theta_2 \prec \dot\theta_1 \prec \dot\theta_2 \prec \lambda \prec \ddot\theta_1 \prec \ddot\theta_2 \prec \dot\lambda \prec \cdots$, we obtain the implicit RIF-form for the system:

$$AX = b. \tag{12}$$

Here

$$A = \begin{bmatrix} l_1^2\left(\frac{1}{4}m_1 + m_2 + m_3\right) + J_1 & -l_1\, l_2 M\, \cos\theta_3 & l_1\, \cos\theta_1 \\ -l_1\, l_2 M\, \cos\theta_3 & {l_2}^2\left(\frac{1}{4}m_2 + m_3\right) + J_2 & -l_2\, \cos\theta_2 \\ l_1\, \cos\theta_1 & -l_2\, \cos\theta_2 & 0 \end{bmatrix};$$

$$b = \begin{bmatrix} -l_1\, g\left(\frac{1}{2}m_1 + m_2 + m_3\right)\cos\theta_1 - l_1\, l_2 M\, \dot\theta_2^{\,2}\sin\theta_3 \\ l_2 M\, g\cos\theta_2 - l_1\, l_2 M\, \dot\theta_1^{\,2}\sin\theta_3 l_1\, \sin\theta_1\dot\theta_1^{\,2} - l_2\, \sin\theta_2\dot\theta_2^{\,2} \end{bmatrix};$$

$$X^T = [r_1, r_2, r_3]^T = [\ddot\theta_1, \ddot\theta_2, \lambda]^T$$

where for the sake of a compact presentation, we have used $M = \frac{1}{2}m_2 + m_3$ and $\theta_3 = \theta_1 + \theta_2$. The constraints are

$$\Phi = l_1\, \sin\theta_1 - l_2\, \sin\theta_2 = 0, \tag{13}$$

$$D_t\Phi = l_1\, \cos\theta_1\dot\theta_1 - l_2\, \cos\theta_2\dot\theta_2 = 0. \tag{14}$$

We note that neither the matrix A, nor vector b include the dependent variable $\lambda(t)$, which was generated by DYNAFLEX. Because of this, the ode system with constraints (12), (13), (14), can be symbolically solved separately for the variables $\theta_1(t)$, $\theta_2(t)$.

4. Explicit Reduced Involutive Form with Large Expression Management

Above, we obtained the implicit RIF-form for the slider crank as (12), (13), (14). We could in principle symbolically invert the matrix A using Maple, and get the explicit RIF-form, but Maple will give a huge output for A^{-1} and require a lot of memory and computation time. Therefore, we have used symbolic inversion using large expression management (LEM). A full discussion of large expression management will be given elsewhere. Here we give a brief outline.

We have used the MAPLE package `LargeExpressions` which is based on tools first developed for perturbation calculations in fluid mechanics [13]. From the paper [13], we have the following definition for a hierarchy and the main idea for hierarchical representations.

Definition [Hierarchy]. A hierarchy is an ordered list $[S_0, S_1, \ldots]$ of symbols, together with an associated list $[D_0, D_1, \ldots]$ of definitions of the symbols. For each $s \in S_i$ with $i \geq 1$, there is a definition $d \in D_i$ of the form $s = f(\sigma_1, \sigma_2, \ldots, \sigma_k)$ where f is some well-understood function such as an elementary function and each σ_j is a symbol in $[S_0, S_1, \ldots, S_{i-1}]$ and is thus lower in the hierarchy than s.

A computation sequence c is recursively defined as an expression of the form $c = g(s_1, s_2, \ldots, s_k)$ containing symbols s_j from a known hierarchy, together with the computation sequences defined by the associated definitions $d_1, d_2, \ldots, d_k$ of the symbols appearing in c. Obviously a computation sequence defined in terms of symbols in S_0, the set of atoms of the system, is just an expression.

Intuitively, a hierarchy is a framework for constructing computation sequences, and a computation sequence is an expression defined in terms of simpler expressions. We used the package `LargeExpressions` to code our own LU symbolic decomposition routine and also the forward and backward substitutions for solving a linear system. The details will be given elsewhere [14]. After LU decomposition with pivoting and zero-recognition, we get matrices L and U with the pivoting P as follows:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ \frac{-{l_1}^2\left(\frac{1}{4}m_1+m_2+m_3\right)+J_1}{l_1\, l_2 M\, \cos\theta_3} & 1 & 0 \\ \frac{-\cos\theta_1}{l_2 M\, \cos\theta_3} & \frac{-4W_3}{W_1} & 1 \end{bmatrix}, \quad P = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{15}$$

$$U = \begin{bmatrix} -l_1\, l_2 M\, \cos\theta_3 & {l_2}^2\left(\frac{1}{4}m_2+m_3\right)+J_2 & -l_2\, \cos\theta_2 \\ 0 & -\frac{1}{8}W_1 & \frac{1}{2}W_2 \\ 0 & 0 & -2\,W_4 \end{bmatrix}, \tag{16}$$

where $PA = LU$. Now using forward and backward substitution in the usual way, we get X coded with the `LargeExpressions` package.

After that, we get Explicit RIF-form in the computation sequence form:

$$\ddot{\theta}_1 = -W_7, \tag{17}$$

$$\ddot{\theta}_2 = \frac{8}{W_1}[-l_1\, g\left(\frac{1}{2}m_1+m_2+m_3\right)\cos\theta_1 - l_1\, l_2\, \dot{\theta}_2^2 \sin(\theta_1+\theta_2)\left(\frac{1}{2}m_2+m_3\right) - \frac{1}{4}W_5 + \frac{1}{4}W_2 W_6], \tag{18}$$

$$\lambda = \frac{1}{2}W_6, \tag{19}$$

and with the constraints

$$\Phi = l_1 \sin\theta_1 - l_2 \sin\theta_2 = 0; \tag{20}$$

$$D_t\Phi = l_1 \cos\theta_1\dot{\theta_1} - l_2 \cos\theta_2\dot{\theta_2} = 0. \tag{21}$$

The complete symbolic expressions for the $W[i]$ are given in the appendix.

The equations (17), (18), (19), (20), (21) are also the explicit RIF form for the 2d slider crank. The symbolic preprocessing helps find all the hidden constraints in the general DAE system (1), (2). This preprocessing procedure makes it possible for us to integrate only interesting variables without computing all the independent variables. For example, to this two dimensional slider crank, as already mentioned, now we can symbolically separate (19) containing $\lambda(t)$ from the other (17), (18), (20), (21) and we can solve directly for $\theta_1(t)$ and $\theta_2(t)$.

5. Numerical Integration Using Computation Sequences

We now show how the two second-order (17), (18) for θ_1 and θ_2 can be integrated. In general it is not possible to unveil all the expressions in the array W. To do so would be to introduce the memory problems that were avoided by using the `Veil` command. Instead we set up computation sequences to pass to `dsolve`. We use MAPLE for the calculations. In particular we only use the MAPLE `LargeExpressions` command `Unveil` to a minimal depth 1, to express the computation sequence of substitution rules SW:

```
> SW := [seq(W[i] = Unveil[W](W[i],1), i = 1 .. LastUsed[W])];
```

Then in order to use `dsolve/numeric`, we make the variable substitutions as follows:

> YW := [subs$(\theta_1(t) = Y[1], \theta_2(t) = Y[3], \dot{\theta}_1(t) = Y[2], \dot{\theta}_2(t) = Y[4], SW)$];

Now we use the `codegen` package in Maple to make a procedure for the numeric integration.

```
> f := codegen[makeproc]( YW, YP[1] = Y[2],  YP[2] = rhs(odesys[1]),
YP[3] = Y[4],YP[4] =  rhs(odesys[2]], parameters = [N, t, Y, YP] );
```

In order to complete the demonstration, we integrate the system for sample numerical values. For the parameters, we choose the following values:

$l1 := 1; l2 := 2; m1 := 1; m2 := 1; m3 := 2; g := 9.8; J1 := 4.5; J2 := 5.5;$

For the initial conditions, we choose the following

$$\theta_1(0) = 0,\ \dot{\theta}_1(0) = 1,\ \theta_2(0) = 0,\ \dot{\theta}_2(0) = \frac{1}{2}$$

which are consistent with the constraint equations

$$\Phi = l_1 \sin\theta_1 - l_2 \sin\theta_2 = 0; \tag{22}$$

$$D_t\Phi = l_1 \cos\theta_1\dot{\theta_1} - l_2 \cos\theta_2\dot{\theta_2} = 0. \tag{23}$$

The Maple numeric dsolve routine can solve the system as follows.

```
> ics := array( [0,1,0,1/2] );
> dvars := [$\theta_1(t), \dot{\theta}_1(t), \theta_2(t), \dot{\theta}_2(t)$];
> dsol := dsolve(numeric, number=4, procedure =f, start =0,
    initial = ics, procvars = dvars);
```

Since the system is being integrated as an ordinary ODE system rather than a DAE one, we gain in speed but expect to lose accuracy. If we pick up some points for testing the numeric ODE solutions with respect to the algebraic constraints, we see a slow loss of precision. For example, when $t = 1$, we have the solutions *dsol* as $[t = 1., \theta_1(t) = .2629, \dot{\theta}_1(t) = -.4375, \theta_2(t) = .1303, \dot{\theta}_2(t) = -.2130]$, and errors in the constraints (22, 23) are $0.14e-7$ and $-0.69e-7$ respectively. When $t = 20$, the errors are $0.42e-5$ and $0.59e-6$ respectively.

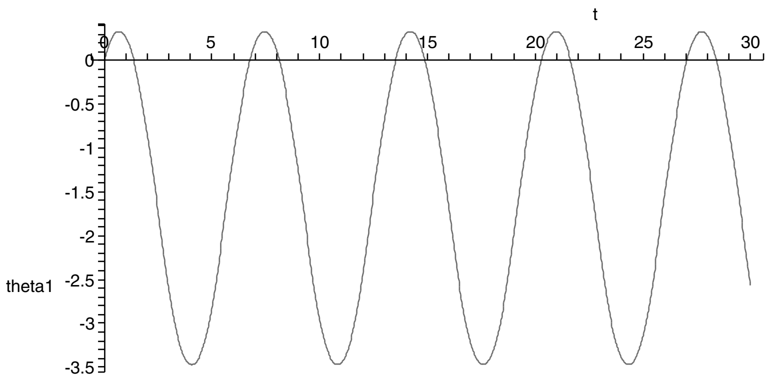

FIGURE 2. The angular θ_1 oscillating movement.

Plots of the numerical simulation of the two dimensional slider crank are shown below as Fig. 2 and Fig. 3. It tells us that the 2d slider crank is moving oscillation with the given initial conditions $\dot{\theta}_1(0) = 1$ and $\dot{\theta}_2(0) = 1/2$. If we increase the initial speeds of θ_1 and θ_2, for example let $\dot{\theta}_1(0) = 2$ and $\dot{\theta}_2(0) = 1$ which are consistent initial conditions, we get Fig. 4 and Fig. 5 which show us the monotonic mode of the angle θ_1.

6. Application of Numerical Algebraic and Analytic Geometry

In this section we discuss the use of some new techniques that are being applied to DAE such as those studied here. A significant problem in large systems, such as the target systems for this work, is the finding of consistent initial conditions. Again there is a need for combined symbolic-numeric methods. The first area that can be harnessed to DAE is that of Numerical Algebraic Geometry [5]. In that approach, components of a polynomial system are characterized by "witness" points, which

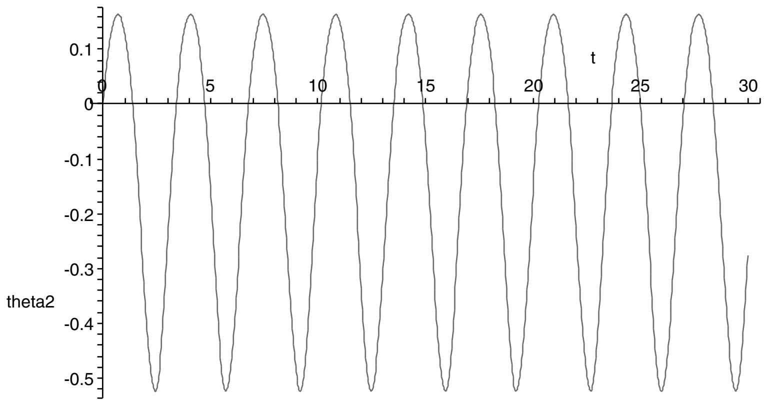

Figure 3. The angular θ_2 oscillating movement.

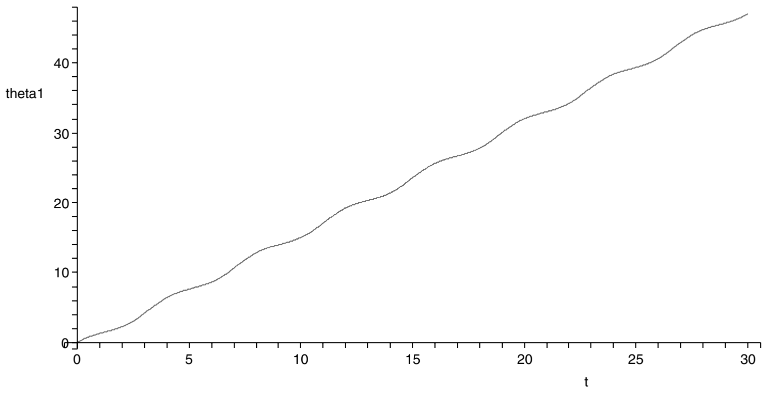

Figure 4. Monotonic mode for θ_1.

result from intersecting the component with random linear spaces of complementary dimension. For polynomial DAE, this gives a method of determining points on the constraints, for the consistent initialization of numerical integrators. Also the hybrid symbolic-numeric completion process described in [6] can be applied to polynomial DAE, using numerical algebraic geometry.

Another interesting possibility for analytic DAE is to extend the methods of Numerical Algebraic Geometry [5] as applied to polynomial PDE [6] to the case of analytic DAE. This requires using the substitution of approximate points on the components of the leading nonlinear systems to test ideal membership in the application of RIF. It is natural to generalize the techniques of algebraic geometry

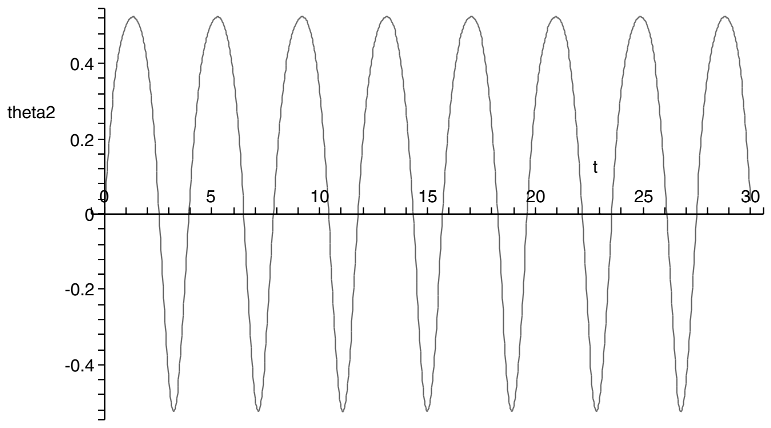

Figure 5. The angular motion θ_2 corresponding to monotonic mode of θ_1.

to the analytic case and to characterize irreducible components of analytic functions by points cut out by the intersection of the components with random linear spaces of complementary dimension. The systems are regularized by embedding in appropriate square systems, as in the polynomial case. Newton methods converge to the points on the components, locally. See [11, §5.2.3] for material on the range of non-constant entire functions. Global results like those in the polynomial case, using homotopy continuation, are much harder to obtain. Usually computations must be executed locally on some compact set.

For example, consider the determination of points on the components of an analytic function $f(z, w) = 0$ on some compact set $U \subset \mathbb{C}^2$. By intersecting the component with a random line $\alpha z + \beta w + \gamma = 0$, we obtain a univariate problem whose solutions can be found on $\mathbb{C}$. Powerful tools are already available to automate this process, for example the Maple command `Rootfinding[Analytic]`. This uses the Cauchy Integral Formula to determine the number of zeros in a given sub-domain of $\mathbb{C}$. This subdomain is then divided to isolate the roots. We note that α-theoretic methods can lead to guaranteed convergence, in the isolated zero case, provided certain local criteria are met [9, 8]. A forthcoming work will be devoted to such *Numerical Analytic Geometry* techniques.

7. Conclusion

This paper uses a simple multibody dynamic system to show how we can combine symbolic methods and numeric methods for simulating mechanical systems described by higher-index DAE systems. An advantage of this combination is not only that it helps the numeric method to find consistent initial conditions, but it also extend the potential for symbolic methods, such as RifSimp, to solve more complex symbolic multibody dynamic systems. Using computation sequences to invert a symbolic matrix allows the calculation to be completed in less memory and less time, more details being given in a upcoming paper [14].

We also discussed applying new methods from Numerical Algebraic Geometry and Numerical Analytic Geometry to DAE. Already in her analysis of analytic DAE, Ilie used computation sequences to establish polynomial cost methods. The method upon which she based her complexity analysis is Pryce's structural analysis of DAE [16, 17, 18]. Specifically Pryce's method can be regarded as an efficient way to obtain implicit RifSimp forms, in certain cases. Numerical analytic geometry naturally partners such methods.

Acknowledgements. We thank Silvana Ilie for discussions.

References

[1] P. Shi, J. McPhee. *Symbolic Programming of a Graph-Theoretic Approach to Flexible Multibody Dynamics.* Mechanics of Structures and Machines, **30**(1), 123–154 (2002).

[2] P. Shi, J. McPhee. *Dynamics of Flexible Multibody Systems Using Virtual Work and Linear Graph Theory.* Multibody System Dynamics, **4**(4), 355–381 (2000).

[3] W. Schiehlen. *Multibody Systems Handbook.* Springer-Verlag, Berlin, 1990.

[4] H. Stetter. *Numerical Polynomial Algebra.* SIAM, 2005.

[5] A.J. Sommese, C.W. Wampler. *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science.* World Scientific Press, Singapore, 2005.

[6] G. Reid, J. Verschelde, A.D. Wittkopf, W. Wu. *Symbolic-Numeric Completion of Differential Systems by Homotopy Continuation.* Proc. ISSAC 2005, 269–276. ACM Press, 2005.

[7] G. Reid, A. Wittkopf, A. Boulton. *Reduction of Systems of Nonlinear Partial Differential Equations to Simplified Involutive Forms.* Eur. J. Appl. Math., **7**, 604–635 (1996).

[8] M. Giusti, G. Lecerf, B. Salvy, J.-C. Yakoubsohn. *On Location and Approximation of Clusters of Zeros: Case of Embedding Dimension One.* To appear in Foundations of Computational Mathematics, 2006.

[9] M. Giusti, G. Lecerf, B. Salvy, J.-C. Yakoubsohn. *Location and Approximation of Clusters of Zeros of Analytic Functions.* Foundations of Computational Mathematics, **5**, 257–311 (2005).

[10] W. Zhou, D.J. Jeffrey, G.J. Reid, C. Schmitke, J. McPhee. *Implicit Reduced Involutive Forms and Their Application to Engineering Multibody Systems.* Proc. IWMM 2004, LNCS 3519, 31–43. Springer-Verlag, Berlin, 2005.

[11] T. Nishino. *Function Theory in Several Complex Variables.* Translations of Mathematical Monographs 193, AMS, 2001.

[12] R.M. Corless. *Essential Maple 7.* Springer-Verlag, Berlin, 2002.

[13] R.M. Corless, D.J. Jeffrey, M.B. Monagan, Pratibha. *Two Perturbation Calculation in Fluid Mechanics Using Large-Expression Management.* J. Symbolic Computation, **11**, 1–17 (1996).

[14] W. Zhou, D.J. Jeffrey. *LU Symbolic Decomposition with Large Expression Management Strategies.* In preparation.

[15] M.B. Monagan. *Gauss: a Parameterized Domain of Computation System with Support for Signature Functions.* Proc. DISCO '93, LNCS 722, 81–94. Springer-Verlag, Berlin, 1993.

[16] S. Ilie, R.M. Corless, G. Reid. *Numerical Solutions of Index-1 Differential Algebraic Equations Can Be Computed in Polynomial Time.* Numerical Algorithms, **41**, 161–171 (2006).

[17] S. Ilie. *Computational Complexity of Numerical Solutions of Initial Value Problems for Differential Algebriac Equations.* PhD thesis, University of Western Ontario, 2005.

[18] R.M. Corless, S. Ilie. *Polynomial Cost for Solving IVP for High-Index DAE.* Submitted.

[19] J.D. Pryce. *A Simple Structural Analysis Method for DAEs.* BIT **41**(2), 364–394 (2001).

[20] J.D. Pryce. *Solving High-index DAEs by Taylor Series.* Numer. Algorithms, **19**, 195–211 (1998).

Appendix

We append the source listing of the right-hand side of the differential equation, to show the result of Maple's automatic code generation.

$f := \mathrm{proc}(N, t, Y, YP)$
$W[1] := (-4l_1^2 l_2^2 \cos(Y[1]+Y[3])^2 m_2^2 - 16 l_1^2 l_2^2 \cos(Y[1]+Y[3])^2 m_2 m_3 - 16 l_1^2 l_2^2$
$\cos(Y[1]+Y[3])^2 m_3^2 + l_1^2 m_1 l_2^2 m_2 + 4 l_1^2 m_1 l_2^2 m_3 + 4 l_1^2 m_1 J_2 + 4 l_1^2 m_2^2 l_2^2 + 20 l_1^2 m_2$
$l_2^2 m_3 + 16 l_1^2 m_2 J_2 + 16 l_1^2 m_3^2 l_2^2 + 16 l_1^2 m_3 J_2 + 4 J_1 l_2^2 m_2 + 16 J_1 l_2^2 m_3 + 16 J_1 J_2)/(l_1$
$l_2 \cos(Y[1]+Y[3])(m_2 + 2m_3));$
$W[2] := (-2l_1^2 \cos(Y[1]) \cos(Y[1]+Y[3]) m_2 - 4 l_1^2 \cos(Y[1]) \cos(Y[1]+Y[3]) m_3$
$+ \cos(Y[3]) l_1^2 m_1 + 4\cos(Y[3]) l_1^2 m_2 + 4 \cos(Y[3]) l_1^2 m_3 + 4\cos(Y[3]) J_1)/(l_1$
$\cos(Y[1]+Y[3])(m_2+2m_3));$
$W[3] := (-2l_2^2 \cos(Y[3]) \cos(Y[1]+Y[3]) m_2 - 4 l_2^2 \cos(Y[3]) \cos(Y[1]+Y[3]) m_3$
$+ \cos(Y[1]) l_2^2 m_2 + 4 \cos(Y[1]) l_2^2 m_3 + 4\cos(Y[1]) J_2)/(l_2 \cos(Y[1]+Y[3])(m_2$
$+2m_3));$

$W[4] := (-\cos(Y[1]) \cos(Y[3]) W[1] + W[3] W[2] \cos(Y[1]+Y[3]) m_2 + 2W[3]W[2]$
$\cos(Y[1]+Y[3]) m_3)/(\cos(Y[1]+Y[3])(m_2+2m_3)W[1]);$
$W[5] := (-g\cos(Y[3]) + l_1 Y[2]^2 \sin(Y[1]+Y[3]))(l_1^2 m_1 + 4 l_1^2 m_2 + 4 l_1^2 m_3 + 4 J_1)$
$/(\cos(Y[1]+Y[3]) l_1);$

$W[6] := (-l_1 \sin(Y[1])Y[2]^2 \cos(Y[1]+Y[3])W[1]+l_2 \sin(Y[3])Y[4]^2 \cos(Y[1]$
$+Y[3])W[1]-\cos(Y[1])W[1]g \cos(Y[3])+\cos(Y[1])W[1]l_1Y[2]^2 \sin(Y[1]+Y[3])$
$-2W[3] \cos(Y[1]+Y[3])l_1 g \cos(Y[1])m_1-4W[3] \cos(Y[1]+Y[3])l_1 g \cos(Y[1])m_2$
$-4W[3] \cos(Y[1]+Y[3])l_1 g \cos(Y[1])m_3-2W[3] \cos(Y[1]+Y[3])l_1 l_2 Y[4]^2 \sin(Y[1]$
$+Y[3])m_2-4W[3] \cos(Y[1]+Y[3])l_1 l_2 Y[4]^2 \sin(Y[1]+Y[3])m_3-W[3] \cos(Y[1]$
$+Y[3])W[5])/(W[4]W[1] \cos(Y[1] + Y[3]));$

$W[7] := (-l_2 g \cos(Y[3])W[1]m_2-2l_2 g \cos(Y[3])W[1]m_3+l_1 l_2 Y[2]^2 \sin(Y[1]$
$+Y[3])W[1]m_2+2l_1 l_2 Y[2]^2 \sin(Y[1]+Y[3])W[1]m_3-2l_2^2 m_2 l_1 g \cos(Y[1])m_1-4l_2^2$
$m_2^2 l_1 g \cos(Y[1])-20l_2^2 m_2 l_1 g \cos(Y[1])m_3-2l_2^3 m_2^2 l_1 Y[4]^2 \sin(Y[1]+Y[3])-12l_2^3$
$m_2 l_1 Y[4]^2 \sin(Y[1]+Y[3])m_3-l_2^2 m_2 W[5]-l_2^2 m_2 W[2]W[6]-8l_2^2 m_3 l_1 g \cos(Y[1])$
$m_1-16l_2^2 m_3^2 l_1 g \cos(Y[1])-16l_2^3 m_3^2 l_1 Y[4]^2 \sin(Y[1]+Y[3])-4l_2^2 m_3 W[5]-4l_2^2 m_3$
$W[2]W[6] - 8J_2 l_1 g \cos(Y[1])m_1 - 16J_2 l_1 g \cos(Y[1])m_2 - 16J_2 l_1 g \cos(Y[1])m_3$
$-8J_2 l_1 l_2 Y[4]^2 \sin(Y[1]+Y[3])m_2-16J_2 l_1 l_2 Y[4]^2 \sin(Y[1]+Y[3])m_3-4J_2 W[5]$
$-4J_2 W[2]W[6]+l_2 \cos(Y[3])W[6]W[1])/(W[1]l_1 l_2 \cos(Y[1]+Y[3])(m_2+2m_3));$

$YP[1] := Y[2]; YP[2] := W[7]; YP[3] := Y[4]; YP[4] := 8(-l_1 g(1/2m_1+m_2+m_3)$
$\cos(Y[1])-l_1 l_2 Y[4]^2 \sin(Y[1]+Y[3])(1/2m_2+m_3) - 1/4W[5]$
$-1/4W[2]W[6])/W[1];$

endproc

Wenqin Zhou, David J. Jeffrey and Greg J. Reid
Department of Applied Mathematics
The University of Western Ontario
London, Ontario, Canada
e-mail: `wzhou7@uwo.ca`
`djeffrey@uwo.ca`
`reid@uwo.ca`

!!! Please tell us your complete address, the streetname is missing