An Algebraic Method for Analyzing Open-Loop Dynamic Systems

W. Zhou, D.J. Jeffrey, and G.J. Reid

Department of Applied Mathematics, The University of Western Ontario,
London, Ontario, Canada N6A 5B7

Abstract. This paper reports on the results of combining the MAPLE packages DYNAFLEX and RIFSIMP. The DYNAFLEX package has been developed to generate the governing dynamical equations for mechanical systems; the RIFSIMP package has been developed for the symbolic analysis of differential equations. We show that the output equations from DYNAFLEX can be converted into a form which can be analyzed by RIFSIMP. Of particular interest is the ability of RIFSIMP to split a set of differential equations into different cases; each case corresponds to a different set of assumptions, and under some sets of assumptions there are significant simplifications. In order to allow RIFSIMP to conduct its analysis, the governing equations must be converted from trigonometric form into a polynomial form. After this is done, RIFSIMP can analyze the system and present its results either graphically, or in list form. The mechanical systems considered are restricted to open-loop systems, because at present, closed-loop systems require too much computation by RIFSIMP to permit analysis.

Keywords: DYNAFLEX, RIFSIMP, Case Splitting, Symbolic Simplification, Graph Theory, Computer Algebra.

1 Introduction

A principal goal of multibody dynamics is the automatic generation of the equations of motion for complex mechanical systems [1]. Numerically based programs exist, for example ADAMS, JAMES and WORKING MODEL, that can generate the governing equations [2]; they are commercial programs and are in widespread use in the automotive, aerospace and robotics industries [3]. However, being numerically based programs they have several drawbacks.

– It is difficult to check the equations of motion, because they are represented by large volumes of numerical data.
– One cannot develop any physical insight.
– Closed-form solutions for the numerical equations are not possible.
– When used for simulations, they are inefficient because the equations are effectively re-assembled at each time step, and this may include many multiplications by 0 and 1.

An alternative approach to equation generation uses symbolic processing. DYNAFLEX (`http://real.uwaterloo.ca/~dynaflex/`) is a Maple package that generates the equations of motion for a mechanical system from an input description based on a graph-theoretic method [4]. The equations are in a completely analytical form, and this offers several advantages [5].

– The structure of the equations is easily obtained and manipulated, giving the user a physical insight into the system.
– The equations can be easily exchanged with other groups of engineers or design groups.
– Real-time simulations are facilitated.

The symbolic models generated by DYNAFLEX are usually too complex to be solved symbolically, but there is still the possibility of symbolically preprocessing the output of DYNAFLEX before attempting a solution. The purpose of this paper is to apply the package RIFSIMP to this task. The RIFSIMP package analyzes ODE- and PDE-systems and returns canonical differential forms. The basic form is Reduced Involutive Form (RIF), and the package has the following features [6]:

– Computation with polynomial nonlinearities.
– Advanced case splitting capabilities for the discovery of particular solution branches with desired properties.
– A visualization tool for the examination of the binary tree that results from multiple cases.
– Algorithms for working with formal power series solutions of the system.

When RIFSIMP is applied to an equation system, it identifies special cases during the reduction. The analysis of the system then splits according to the special cases. In a full case analysis, some cases can be very complicated while others are simple enough to be analytically solvable. The canonical form generated by RIFSIMP is of low (0 or 1) differential index, which is suitable for the application of numerical solvers (whereas the output of DYNAFLEX may not have been suitable). An important option with RIFSIMP is the possibility of excluding special cases that are known to be not of interest. Thus if RIFSIMP detects a special case, say $m = 0$, but we know that this is of no physical interest, then we can pass this information to RIFSIMP as an inequation $m \neq 0$ appended to the input system.

1.1 Open-Loop and Closed-Loop Systems

The problems addressed by DYNAFLEX can be separated into two classes: open-loop systems and closed-loop systems. The terminology comes from the graph-theoretic method which is the basis of DYNAFLEX. The example in the next section will show the basic graph-theory method. For the present, we can note that an open-loop system corresponds to a system such as a robot arm, in which several components, such as joints and arm segments, are joined together in a configuration that terminates at a free end. This is in contrast to a framework, such as a four-bar mechanism, in which the components connect back to the structure being considered.

When a structure forms a closed loop, then Dynaflex will generate constraint equations that describe the drop in the degrees of freedom that accompanies the closing of a loop. From the point of view of this paper, we have discovered that the RifSimp package takes a great deal of time and memory to analyze closed-loop systems, but can make good progress with open-loop ones. This is what is reported here.

2 Example System Analysed Using Dynaflex

In order to keep the examples within printable limits, we shall use a simple spinning top as an example. A top is an axisymmetric body that spins about its body-fixed symmetry axis. It can precess about a vertical (Z) axis, and nutate about the rotated X axis. Figure 1 shows gravity acting in the $-Z$ direction. The center of mass is located at C, and the spinning top is assumed to rotate without slipping on the ground; this connection is modelled by a spherical (ball-and-socket) joint at O. The joint coordinates at O are represented by Euler angles (ζ, η, ξ), in the form of 3-1-3 Euler angles, meaning that they correspond to precession, nutation, and spin, respectively.

2.1 The System Graph

The system graph for the top is shown in Figure 2. The graph consists of nodes and edges. The nodes correspond to centres of coordinates, while the edges describe the system. Thus in the figure, we see nodes labelled 1,2,3 which are connected by edges e1, e2, e3,e4.

Node (1) can be thought of as the datum, or ground, on which the system is resting. Nodes (2) and (3) denote the top. The geometric fact that the top is connected to the ground is described by the edge e2, connecting node 1 to node 2. The fact that the top is free to spin, but not free to slide is described by e2 being specified as a spherical joint (one can think of a ball in a socket). The fact

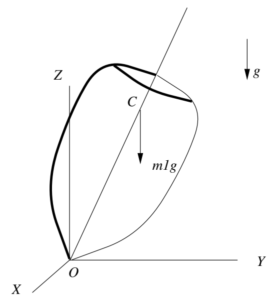

Fig. 1. The three-dimensional top. The centre of mass is at C and OC= l. The mass is here denoted $m1$, giving a gravitational force equal to $m1g$ acting in the $-Z$ direction

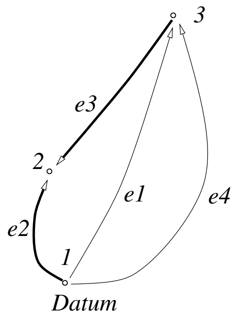

Fig. 2. The graph-theoretic method for the top

that the body is a top is described by edge e1 being specified as a rigid body with a diagonal moment of inertia. The position of the centre of mass is specified by e3, which formally is a rigid-arm element. Finally, gravity is defined by e4, being a force element.

It is important to note that the graph consisting of {e2, e3} is denoted by heavy lines, whereas elements e1 and e4 are drawn in light lines. The elements {e2,e3} correspond to the open loop referred to above.

2.2 Maple Input File

The Maple input file corresponding to the above system is as follows.

```
NOofedges:=4; NOofnodes:=3; Datum:=1;
#Note that Datum stands for the ground node.

edge[1]:=table([(1)=N, (2)=[1,3], (3)=BE_R,
     (4)=table([inert=[[x,0,0], [0,y,0], [0,0,z]], mass=c ])]);
edge[2]:=table([(1)=Y, (2)=[1,2], (3)=JE, (4)=SPH ]);
edge[3]:=table([(1)=Y, (2)=[3,2], (3)=AE_R,
     (4)=table([coords=[0,0,c] ])]);
edge[4]:=table([(1)=N, (2)=[1,3], (3)=FDE,
     (4)=table([type=PD, fz=-m1*g, force=gl ])]);
Iedge:=[];
```

2.3 The Symbolic Equations

The equations produced by Dynaflex from the above input file are presented below. By default, Dynaflex assigns its own notation for quantities such as Euler angles, moments of inertia, etc. Although Dynaflex notation is convenient of its internal programming, it results in equations which are difficult, and even ugly, to read when printed out for human use. Therefore we have edited the raw Dynaflex output format to simplify the notation to bring it in line with what human readers are used to seeing. The centre of mass is a distance l from the point of contact, the mass is m, the moment of inertia about the symmetry axis is C, and about a perpendicular axis is A. The Euler angles are given above.

$$C\ddot{\zeta}(t)\cos^2\eta(t) + C\ddot{\xi}(t)\cos\eta(t) - A\ddot{\zeta}(t)\cos^2\eta(t) - C\dot{\eta}(t)\dot{\xi}(t)\sin\eta(t)$$
$$-2C\dot{\eta}(t)\dot{\zeta}(t)\sin\eta(t)\cos\eta(t) + 2A\dot{\eta}(t)\dot{\zeta}(t)\sin\eta(t)\cos\eta(t) + A\ddot{\zeta}(t) = 0\ , \quad (1)$$
$$-mg\sin\eta(t) - A\dot{\zeta}^2(t)\sin\eta(t)\cos\eta(t) + A\ddot{\eta}(t) + C\dot{\xi}(t)\dot{\zeta}(t)\sin\eta(t)$$
$$+C\dot{\zeta}^2\,(t)\sin\eta(t)\cos\eta(t) = 0\ , \quad (2)$$
$$-C(-\ddot{\xi}(t) - \ddot{\zeta}(t)\cos\eta(t) + \dot{\eta}(t)\dot{\zeta}(t)\sin\eta(t)) = 0\ . \quad (3)$$

3 Automatic Symbolic Simplification Using RifSimp

The package RifSimp can process systems of polynomially nonlinear PDEs with dependent variables $u_1, u_2, \ldots u_n$, which can be functions of several independent variables. For the present application the only independent variable is time. The variables u_i can obey differential equations of varying order. RifSimp takes as its input a system of differential equations and a ranking of dependent variables and derivatives. RifSimp orders the dependent variables lexicographically[1] and the derivatives primarily by total derivative order:

$$u_1 \prec u_2 \prec \ldots \prec u_n \prec u_1' \prec u_2' \prec \ldots \prec u_n' \prec u_1'' \prec \ldots \quad (4)$$

Then equations are classified as being either leading linear (i.e. linear in their highest derivative with respect to the ordering $\prec$) or leading nonlinear (i.e. nonlinear in their highest derivative).

RifSimp proceeds by solving the leading linear equations for their highest derivatives until it can no longer find any such equations. Leading nonlinear equations (the so-called constraints), are treated by methods involving a combination of Gröbner Bases and Triangular Decomposition. It differentiates the leading nonlinear equations and then reduce them with respect to the leading linear equations. If zero is obtained, it means the equation is included in the ideal generated by the leading linear equations. If not, it means that this equation is a new constraint to the system. This is repeated until no new constraints are found.

Theorem: (Output RifSimp form)
If v is a list of derivatives and w is a list of all derivatives (including dependent variables) lower in ranking than v, then the output of RifSimp has the structure

$$v = f(t, w) \quad (5)$$

subject to a list of constraint equations and inequations

$$g(t, w) = 0, \quad h(t, w) \neq 0 \quad (6)$$

Proof. See [6] and references therein.

[1] Readers who are not familiar with the ideas of term ordering in polynomial systems can read this as meaning that the variables are placed in alphabetical order.

Theorem: (Existence and Uniqueness)
Given an initial condition $g(t^0, w^0) = 0, h(t^0, w^0) \neq 0$, there is a local analytic solution with this initial condition.

Proof. See [6].

4 Application of RifSimp to Example Problem

In order to apply RifSimp to the example above, we proceed as follows.

1. Change coordinates using $\cos\eta = \frac{1-u(t)^2}{1+u(t)^2}, \sin\eta = \frac{2u(t)}{1+u(t)^2}$ to get a rational polynomial differential system instead of trigonometric nonlinear differential system. (RifSimp does not allow trigonometric functions.)
2. Give this polynomial differential equation system to RifSimp, specifying the case split option to get normalized differential equations.
3. Use RifSimp case tree to analyze the different cases.

With no assumptions on the parameters, we obtain 24 cases, each split corresponds to some quantity being zero or not. The Maple output lists, for each split, exactly what the quantity is.

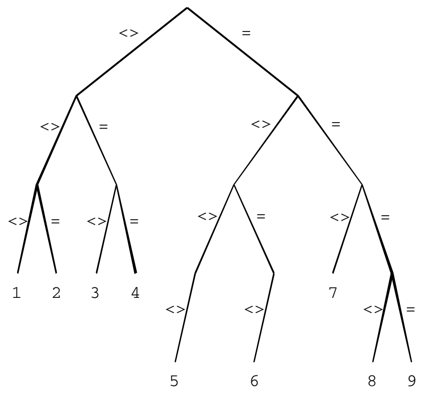

Fig. 3. The case-split tree after the exclusion of cases that are not of physical interest

An examination of the output case-split list shows that many of the cases can be removed on physical grounds. Thus RifSimp identifies that there is division by g, by m, and by l and therefore flags these cases as points of case splitting. Since we know in this application that these cases are not of physical interest, we can exclude them automatically from the case analysis by including a list of inequations in the RifSimp parameters. After including $g \neq 0$, $m \neq 0$, and $l \neq 0$, RifSimp returns a tree containing 9 cases, as shown in Figure 3.

5 Some Special Cases

The importance of the RifSimp analysis is the identification of special cases. The general ("generic") case cannot be simplified further. However, any analysis of a mechanical system should look for special cases, either to exploit these particularly simple cases or to avoid them. The example shows that RifSimp can be used to find these cases automatically.

5.1 First Group of Special Cases

We group two cases together that differ in that RifSimp identifies one case as being $A \neq 0$ and one as $A = 0$. However, the equation containing A is identically zero because of the condition $u = 0$, and hence the value of A is irrelevant. The equations simplify to

$$\ddot{v}(t) = -\ddot{w}(t), \quad u(t) = 0 \tag{7}$$

These equations can be integrated in closed form to

$$u(t) = 0, \quad v(t) = v_1 t + v_0, \quad w(t) = -v_1 t + w_0 \tag{8}$$

where v_1, v_0, w_0 are given by the initial conditions. From the definition of $\eta(t) = \arccos((1 - u(t)^2)/(1 + u(t)^2))$, we have $\eta = 0$ which means that the top is spinning vertically without any inclination with respect to axis Z.

5.2 Second Group of Special Cases

Again RifSimp identifies $A = 0$ and $A \neq 0$ separately. The common equations are

$$\ddot{w}(t) = 0, \quad \dot{u}(t) = 0, \quad \dot{v}(t) = \frac{gml}{C\dot{w}(t)} \tag{9}$$

and with constraint: $u(t)^2 - 1 = 0$. The analytic solution is:

$$u(t) = u_0, \quad v(t) = \frac{gmlt + v_0 w_1 C}{w_1 C}, \quad w(t) = w_1 t + w_0 \tag{10}$$

and with the constraints: $u_0^2 - 1 = 0$ and v_0, w_0, w_1 are given by the initial conditions. Also from the definition of $\eta(t) = arccos((1-u(t)^2)/(1+u(t)^2))$, we have $\eta = \pi/2$, which means the top is moving horizontally in the $x - y$ plane, i.e. it is precessing without nutation.

6 Conclusion and Future Work

Because of space limitations, we have restricted the example problem to being a simple one. Further experiments with other mechanical systems, such as double and triple pendulums, were made but not included here. However, from the example, we can identify successes and limitations of RifSimp. We first point to the success of RifSimp in identifying the special cases of top motion. In the simple example shown here, these might seem well known, but the important point is that these were identified *automatically* rather than by human inspection.

One of the limitations of the combination of RifSimp and Dynaflex is the fact that Dynaflex generates a plethora of parameters. Too many parameters can cause a serious degradation of the performance of a computer algebra system, as well as leading to a large number of special cases that can prevent a human user from seeing the patterns of interest in the output. This is reflected in the 24 special cases initially identified by RifSimp. Controlling the consequences of a large number of parameters will be vitally important to further applications.

This paper has concentrated on open loop systems, because RifSimp has been most successful in these cases. Closed loop systems generate further constraint equations that can cause RifSimp to exhaust computing resources before completing its analysis. One reason for this is the symbolic inversion of matrices in order to obtain RIF form. Computing techniques for handling large expressions have been developed in other contexts, see for example [7]. Combining these with RifSimp will increase the complexity of problems that can be solved.

Acknowledgments

We gratefully acknowledge John McPhee, Chad Schmidt and Pengfei Shi for their development of Dynaflex and for making their system available to us.

References

1. Schiehlen, W. *Multibody Systems Handbook*; Springer-Verlag: Berlin, 1990.
2. P. Rideau. *Computer Algegbra and Mechanics, The James Software*; Computer Algebra in Industry I, 1993 John Wiley & Sons Ltd.
3. Pengfei Shi, John McPhee. *Symbolic Programming of a Graph-Theoretic Approach to Flexible Multibody Dynamics*; Mechanics of Structures and Machines, 30(1), 123-154(2002).
4. P. Shi, J. McPhee. *Dynamics of flexible multibody systems using virtual work and linear graph theory*; Multibody System Dynamics, 4(4), 355-381(2000).
5. Christian Rudolf. *Road Vehicle Modeling Using Symbolic Multibody System Dynamics, Diploma Thesis*; University of Waterloo in cooperation with University of Karlsruhe(2003).
6. Reid, G.J., Wittkopf, A.D., and Boulton, A. *Reduction of Systems of Nonlinear Partial Differential Equations to Simplified Involutive Forms.* Eur. J. Appl. Math. **7** (1996): 604-635.
7. R. M. Corless, D. J. Jeffrey. *Two Perturbation Calculation in Fluid Mechanics Using Large-Expression Management*; J. Symbolic Computation(1996) 11, 1-17.