AC 2008-87: TEACHING MULTIBODY DYNAMICS IN AN UNDERGRADUATE CURRICULUM – AN INTUITIVE AND EXPLICIT FORMALISM BASED ON PARASITIC ELEMENTS

Geoff Rideout, Memorial University of Newfoundland

Geoff Rideout received his B.Eng. (Mechanical) from Memorial University in 1993, his M.A.Sc. (Eng.) from Queen's University in 1998, and his Ph.D. from the University of Michigan in 2004. He is currently an assistant professor of engineering at Memorial University, teaching mechanics and design courses. He is conducting research in the area of automated generation of computer simulation models for dynamic system design.

Teaching Multi-Body Dynamics in an Undergraduate Curriculum: An Intuitive and Explicit Formalism Based on Parasitic Elements

Abstract

Typical undergraduate mechanical engineering curricula in North America do not include a course in multi-body dynamics. A rigid body dynamics course covering single-body kinetics is usually completed in early semesters, and often the material is not revisited before graduation. Students typically graduate without a sense of how to simulate the forward dynamics of even simple multi-body systems such as slider-crank or four-bar mechanisms. Engineers should have some increased depth of understanding in this area even if they will exclusively use specialized commercial software after graduation.

A physically intuitive, explicit multi-body formalism is presented that will allow senior students to review and refresh their knowledge of dynamics, understand how to handle constraint forces, and write their own forward dynamics simulation code using software such as MATLAB. The formalism is based on the use of parasitic (stiff) springs to allow a small but finite relaxation of ideal joint constraints. Stiff springs break dependencies among the generalized coordinates of connected bodies and thereby allow derivation of a set of explicit first-order ordinary differential equations. Joint forces are found from parasitic spring deflections. Moreover, a consistent set of initial conditions can be generated without resorting to nonlinear equation solution.

The formulation and its advantages are demonstrated using a double pendulum and a slider-crank mechanism case study. The formulation gives the student a practical tool for dynamic analysis of mechanisms and prepares him or her for more advanced study of topics such as Lagrange's equations and Lagrange multipliers.

Introduction

The absence of multi-body dynamics in the typical North American mechanical engineering curriculum is a gap that has received surprisingly little attention. The typical graduating mechanical engineer in North America cannot write a coherent set of governing dynamic equations for a slider-crank or four-bar mechanism, and then explain how these equations might be solved. These are simple multi-body systems and fundamental building blocks of mechanical systems, and contain multiple rigid bodies. In the rigid body dynamics course that is usually completed in early semesters, single bodies are the focus. A sampling of five undergraduate curricula in North America shows a pattern of only introductory exposure to kinematics and kinetics, and only early on in the program[1,2,3,4,5]. Typical dynamics-related course offerings include:

- rigid body dynamics (kinematics and kinetics of particles and single bodies)
- theory of mechanisms and machines (graphical and analytical methods for kinematics of planar mechanisms; instantaneous determination of static and dynamic loads resulting from prescribed motion)
- vibrations (free-body diagrams and equations of motion for simple mass-spring-damper systems)

- numerical methods (numerical solution of systems of nonlinear equations, possibly numerical integration of coupled first-order ordinary differential equations).

The "inverse dynamics" problem (predicting the forces and torques required to create a desired motion) may or may not be addressed in a mechanism theory or robotics course. MATLAB/Simulink-based treatments of kinematics and inverse dynamics of mechanisms have been presented[6], but without treatment of the "forward dynamics" problem (predicting the motion resulting from applied forces and torques). Other studies[7] affirm that engineering curricula often do not include mandatory courses where multi-body dynamics is taught. Students should actually be given the opportunity to use the equations of motion determined using concepts from introductory mechanics. Students with some knowledge of numerical methods should be shown the connection between equations of motion and time integration, so that they feel they are receiving a greater "return on investment" for their efforts in generating equations[8] in dynamics courses.

The typical new engineer who must perform dynamic simulation of rigid body systems may avail him or herself of commercial software without having a fundamental sense of how connected bodies and the constraint forces between them can be modeled and then simulated. Large-scale commercial software packages have been used to illustrate mechanical system behaviour in junior courses, followed by advanced multi-body systems courses covering Jacobian matrices, redundant constraints, singular positions, and Lagrange multipliers[9]. Such topics would be quite demanding for North American curricula that are not dynamics-intensive, and a simpler approach in which student programming skills are exercised is presented here. Given that students in some studies have found software programming and debugging to be time-consuming and challenging[7,8], one could argue that further exposure to computer programming in later semesters is useful, especially if the programming environment has broad application and is user-friendly. MATLAB can be used in many courses, is widely used in industry, and is seen as being more versatile and "user friendly" than languages such as C++ and Fortran by mechanical engineering students.

A senior elective on machine dynamics, recently initiated by the author, reviews relevant concepts from rigid body kinematics and kinetics, and introduces the multi-body dynamics formalism described herein. The governing equations are implemented by the students in MATLAB, an application to which students have been exposed in previous courses. The formalism can be easily implemented in other programming languages such as Fortran or C++, as long as a numerical integration routine is provided or generated by the student.

The remainder of the paper is organized as follows. Section 2 develops an example of the type of problem that would frustrate the typical undergraduate, and proposes that a means of handling constraint forces between connected bodies is the missing link. Section 3 describes a formalism based on stiff springs and dampers in place of ideally rigid joints, and illustrates the approach using a slider-crank mechanism. Section 4 discusses implementation issues and student difficulties. Section 5 contains a summary and conclusions.

The Problem

The freshly minted engineer, if asked to predict the motion of a double pendulum released from rest (Figure 1), would likely embrace the challenge optimistically and enthusiastically. The ability to predict the link motions and pin forces of the double pendulum is more relevant than it might first appear. An early course in mechanism theory will impress upon most students that slider crank and four-bar mechanism have broad application and are building blocks for more complex mechanisms. The slider crank is a double pendulum with the endpoint constrained to move in a straight line. The four-bar is a triple pendulum with the endpoint fixed. The influence of dynamic effects on pin forces is essential to sizing the pins using the concepts learned in machine component design. Unfortunately, problems will likely be encountered by a student in attempting dynamic analysis of the double pendulum.

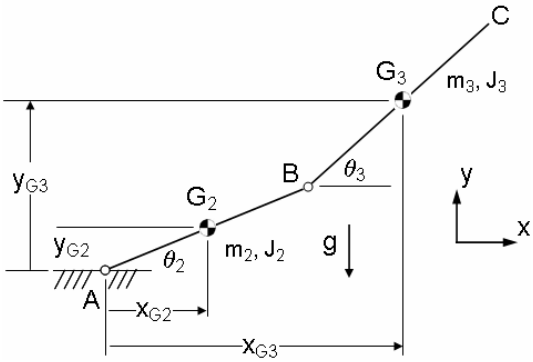

Figure 1 – Double pendulum

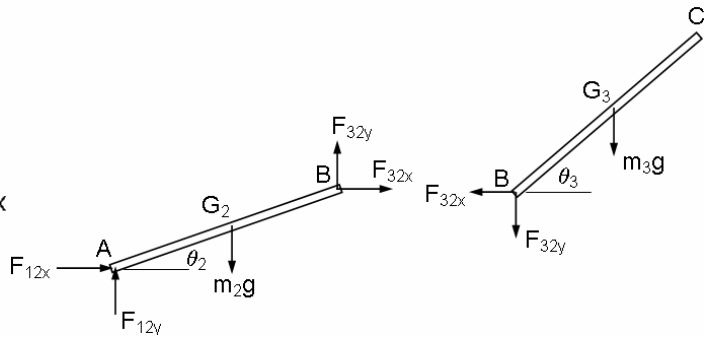

Figure 2 – Free body diagrams

The best students might correctly draw free body diagrams of each body as in Figure 2, and then write and assemble the equations using Newton's Laws. In Figure 2 and following figures, F_{ij} denotes the force of body i on body j.

Newton's Laws, Body 2 (force summations in x and y directions; moment summation about centre of gravity G_2):

$$F_{12x} + F_{32x} = m_2 \ddot{x}_{G2} \tag{1}$$

$$F_{12y} + F_{32y} - m_2 g = m_2 \ddot{y}_{G2} \tag{2}$$

$$F_{12x} AG_2 \sin\theta_2 + F_{32y} G_2 B \cos\theta_2 - F_{12y} AG_2 \cos\theta_2 + F_{32x} G_2 B \sin\theta_2 = J_2 \ddot{\theta}_2 \tag{3}$$

where AG_2, G_2B, etc. are distances between points.

Newton's Laws, Body 3 (recognizing that the constraint forces, while unknown, are equal and opposite on the connected bodies):

$$-F_{32x} = m_3 \ddot{x}_{G3} \tag{4}$$

$$-F_{32y} - m_3 g = m_3 \ddot{y}_{G3} \tag{5}$$

$$F_{32y} BG_3 \cos\theta_3 - F_{32x} BG_3 \sin\theta_3 = J_3 \ddot{\theta}_3 \tag{6}$$

In informal discussions between the author and students, the presence and treatment of the constraint forces at this point is problematic. Some students recognize the constraints among x_{G2}, y_{G2}, and θ_2, and among x_{G3}, y_{G3}, and θ_3:

$$x_{G2} = AG_2 \cos\theta_2 \tag{7}$$

$$y_{G2} = AG_2 \sin\theta_2 \tag{8}$$

$$x_{G3} = AB\cos\theta_2 + BG_3 \cos\theta_3 \tag{9}$$

$$y_{G3} = AB\sin\theta_2 + BG_3 \sin\theta_3 \tag{10}$$

Straightforward application of Newton's Laws and geometry thus yields 10 equations and 10 unknowns. The student may realize that in theory an equal number of equations and unknowns means that a solution is feasible; however, the computer implementation will likely pose a challenge. An undergraduate numerical methods or system dynamics course may introduce systems of first-order differential equations of the form

$$\dot{\vec{x}} = \vec{f}\left(\vec{x}, \vec{u}\right) \tag{11}$$

where the right hand sides of the state equations are strictly functions of the state variables and known inputs. The presence of unknown constraint forces, which may be thought of as Lagrange multipliers associated with the constraint equations[10], in the right-hand sides creates a set of differential-algebraic equations (DAE's). Numerical tools for direction entry and solution of DAE's are not commonly available to the typical undergraduate, and the use of familiar tools such as MATLAB and Maple for numerical solution of the double pendulum requires either

- elimination of the constraint forces from the equations
- index reduction of the DAE's through manual differentiation[11], and/or numerical solution of nonlinear equations with a Newton-Raphson routine at each time step
- a Lagrangian formulation with a minimal set of generalized coordinates
- expressing constraint forces in terms of state variables.

Intensive algebraic manipulations may eliminate the constraint forces, leaving two highly coupled equations in θ_2, θ_3 and their derivatives. This manipulation becomes very cumbersome for mechanisms more complex than a single pendulum. Lagrange's equations are outside the scope of the vast majority of curricula, and will not readily surrender the constraint forces. MathCAD contains an Index 3 DAE solver that allows direct entry of the equations in the form of Eq'ns (1)-(10), but requires initial guesses of constraint forces and gives the user little control over simulation parameters such as integration tolerances. To the best of the author's knowledge, MATLAB is much more commonly employed in undergraduate curricula and industry. The formulation in the following sections has more potential value as a teaching tool in that it gives greater insight into system dynamics and numerical simulation issues.

Parasitic Element Formulation and Example

In a multi-body system, DAE's result due to constraints such as Eq'ns (7)-(10)[12]. To create an explicit set of first-order ordinary differential equations (ODE's), some violation of the position constraints must be allowed so that the primitive position coordinates can vary independently.

This is possible if compliance is introduced in the joints. The constraint forces arising from joints can then be expressed in terms of state variables and stiffness parameters.

"Parasitic" Springs and Dampers

Consider the slider-crank shown below in Figure 3. For rigid, ideal joints the x-component of the position of point C (x_C) would be equal to the slider position x_D, while $y_C = y_D = 0$. If we envision springs at the revolute joint, and between the slider and ground, then the centres of gravity of the connecting rod and slider can move independently in both coordinate directions. The right portion of the figure shows an exaggerated view of a joint where the positions of point *C* and *D* can differ due to joint compliance. As the spring stiffness approaches infinity, the joint approaches an ideal rigid joint. For sufficiently high stiffness, the joint will perform to all appearances like an ideal joint and the constraint violation will be low.

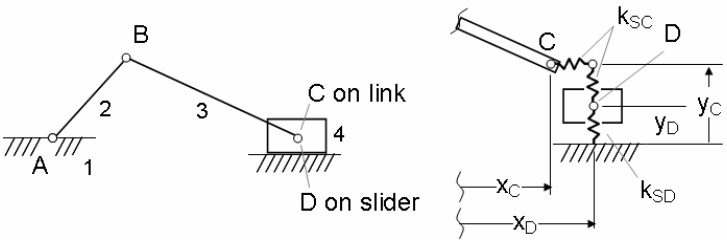

Figure 3 – Slider crank with nonrigid joints

Such springs have been termed "parasitic" elements[13]. Parasitic elements have been proposed for creating explicit differential equations for ease of simulation, but they have not been advanced as a teaching tool. Parasitic elements have a physical basis in that any real joint will have finite stiffness. The stiffness parameter can be estimated based on material properties and the element then treated as a higher-fidelity model component, or it can simply be given a sufficiently high value and used as a numerical integration tool. In order to remove high-frequency transients, a parasitic damper is typically placed in parallel with each parasitic stiffness. The issue of parameter tuning is further treated in the Discussion section.

In the interests of brevity, a general description of the formulation is eschewed in favour of an illustrative example which illustrates the ease with which the parasitic formulation can be applied to closed kinematic chains to write constraint forces in terms of state variables. The approach taken in the example can be generalized to other single or multiple loop mechanisms.

Slider-Crank Equations

Figure 4 shows a slider-crank mechanism. A torque is applied to the crank at *A*, and viscous friction is included in the pin at *C* and between the slider and base. Figure 5 shows the locations and orientations of parasitic elements, for which parameters are chosen to be the same in the *x* and *y* directions at each point. The links are drawn, where possible, so that all quantities (including angles) have positive values. In Figure 4, angle θ_3 is instantaneously negative. Drawing link 3 as in Figure 5 makes it clear that $y_{B3} = y_{G3} - A_3 G_3 \sin\theta_3$. A common student error would be to examine Figure 4 and state that $y_{B3} = y_{G3} + A_3 G_3 \sin\theta_3$. The positive

orientation of Figure 5 also facilitates consistent definition of spring displacements as positive in tension, and assignment of the correct algebraic sign to resulting constraint forces. Figure 6 shows individual free body diagrams.

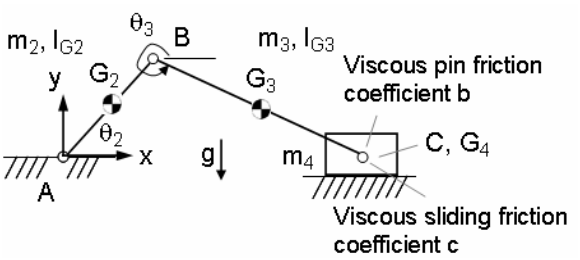

Figure 4 – Slider-crank schematic

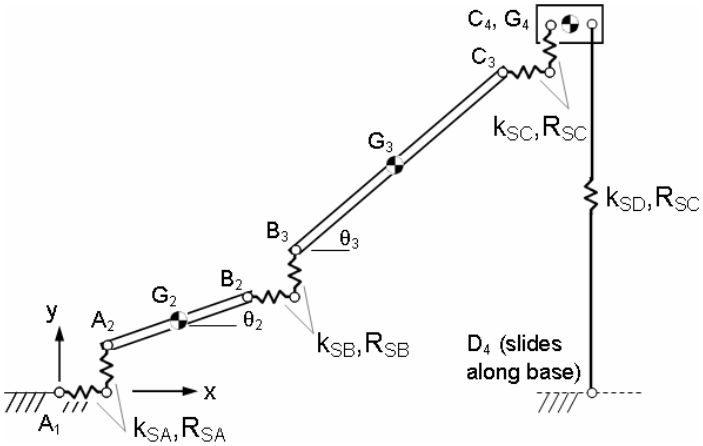

Figure 5 – Parasitic elements

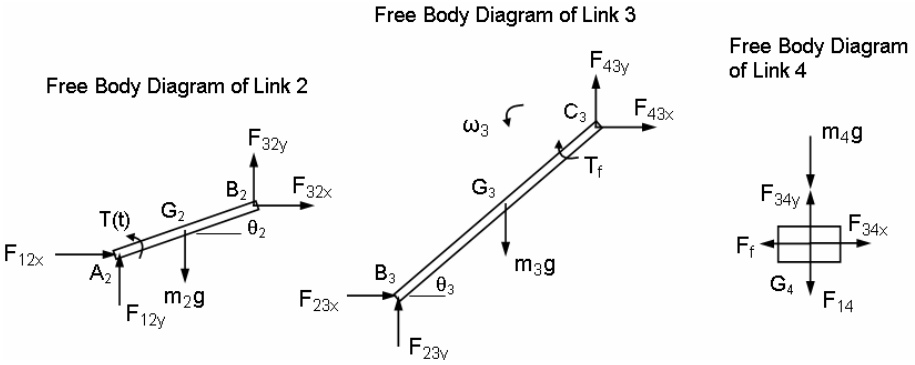

Figure 6 – Free body diagrams

Using Figure 6, the state equations can be written in first-order form using Newton's Laws:

$$\dot{x}_{G2} = v_{G2x}, \quad \dot{v}_{G2x} = {}^{1}\!/_{m_2}\left(F_{12x} + F_{32x}\right)$$

$$\dot{y}_{G2} = v_{G2y}, \quad \dot{v}_{G2y} = {}^{1}\!/_{m_2}\left(F_{12y} + F_{32y}\right) - g$$

$$\dot{\theta}_2 = \omega_2$$

$$\dot{\omega}_2 = {}^{1}\!/_{I_{G2}}\left(T(t) - F_{12y}A_2G_2\cos\theta_2 + F_{12x}A_2G_2\sin\theta_2 + F_{32y}G_2B_2\cos\theta_2 - F_{32x}G_2B_2\sin\theta_2\right)$$

(12)

$$\dot{x}_{G3} = v_{G3x}, \quad \dot{v}_{G3x} = {}^{1}\!/_{m_3}\left(F_{23x} + F_{43x}\right)$$
$$\dot{y}_{G3} = v_{G3y}, \quad \dot{v}_{G3y} = {}^{1}\!/_{m_3}\left(F_{23y} + F_{43y}\right) - g$$
$$\dot{\theta}_3 = \omega_3$$
$$\dot{\omega}_3 = {}^{1}\!/_{I_{G3}}\left(-T_f - F_{23y}B_3G_3\cos\theta_3 + F_{23x}B_3G_3\sin\theta_3 + F_{43y}G_3C_3\cos\theta_3 - F_{43x}G_3C_3\sin\theta_3\right) \tag{13}$$

$$\begin{aligned} \dot{x}_{G4} &= v_{G4x}, \quad \dot{v}_{G4x} = {}^{1}\!/_{m_4}\left(F_{34x} - F_f\right) \\ \dot{y}_{G4} &= v_{G4y}, \quad \dot{v}_{G4y} = {}^{1}\!/_{m_4}\left(F_{34y} - F_{14}\right) - g \end{aligned} \tag{14}$$

The viscous pin damping at C creates an equal and opposite torque T_f on bodies 3 and 4; however, the torque on body 4 is immaterial given that the slider may be treated as a particle. To determine the correct direction of a positive T_f as shown on the free body diagram, the damping constitutive law must be defined.

$$T_f = b\left(\omega_3 - \omega_4\right) = b\omega_3 \tag{15}$$

For viscous sliding friction the constitutive law is simply

$$F_f = cv_{G4x} \tag{16}$$

opposing the positive motion of the slider.

At each time step, the state vector derivatives of Eq'ns (12-14) must be integrated, and the new state variable values made available to the subroutine or m-file in which state derivatives are calculated. Constraint forces must be written in terms of the state variables in Eq'ns (12)-(14). To do this, spring/damper endpoint positions and velocities are defined in terms of state variables, and then the constraint forces are made equal to the spring/damper forces.

Spring and damper endpoint motions are defined in Eq'n (17) as follows:

$$\begin{aligned}
x_{A2} &= x_{G2} - A_2G_2\cos\theta_2, & v_{A2x} &= v_{G2x} + A_2G_2\sin\theta_2\omega_2 \\
y_{A2} &= y_{G2} - A_2G_2\sin\theta_2, & v_{A2y} &= v_{G2y} - A_2G_2\cos\theta_2\omega_2 \\
x_{B2} &= x_{G2} + G_2B_2\cos\theta_2, & v_{B2x} &= v_{G2x} - G_2B_2\sin\theta_2\omega_2 \\
y_{B2} &= y_{G2} + G_2B_2\sin\theta_2, & v_{B2y} &= v_{G2y} + G_2B_2\cos\theta_2\omega_2 \\
x_{B3} &= x_{G3} - B_3G_3\cos\theta_3, & v_{B3x} &= v_{G3x} + B_3G_3\sin\theta_3\omega_3 \\
y_{B3} &= y_{G3} - B_3G_3\sin\theta_3, & v_{B3y} &= v_{G3y} - B_3G_3\cos\theta_3\omega_3 \\
x_{C3} &= x_{G3} + G_3B_3\cos\theta_3, & v_{C3x} &= v_{G3x} - G_3B_3\sin\theta_3\omega_3 \\
y_{C3} &= y_{G3} + G_3B_3\sin\theta_3, & v_{C3y} &= v_{G3y} + G_3B_3\cos\theta_3\omega_3
\end{aligned} \tag{17}$$

Spring deflections and damper velocities are defined as:

$$
\begin{aligned}
x_{SA} &= x_{A2} - x_{A1} = x_{A2} & v_{SAx} &= v_{A2x} - v_{A1x} = v_{A2x} \\
y_{SA} &= y_{A2} - y_{A1} = y_{A2} & v_{SAy} &= v_{A2y} - v_{A1y} = v_{A2y} \\
x_{SB} &= x_{B3} - x_{B2} & v_{SBx} &= v_{B3x} - v_{B2x} \\
y_{SB} &= y_{B3} - y_{B2} & v_{SBy} &= v_{B3y} - v_{B2y} \\
x_{SC} &= x_{G4} - x_{C3} & v_{SCx} &= v_{G4x} - v_{C3x} \\
y_{SC} &= y_{G4} - y_{C3} & v_{SCy} &= v_{G4y} - v_{C3y} \\
y_{SD} &= y_{G4} & v_{SDy} &= v_{G4y}
\end{aligned}
\tag{18}
$$

Constraint forces may then be written simply as:

$$
\begin{aligned}
F_{12x} &= -k_{SA} x_{SA} - R_{SA} v_{SAx} \\
F_{12y} &= -k_{SA} y_{SA} - R_{SA} v_{SAy} \\
F_{32x} &= k_{SB} x_{SB} + R_{SB} v_{SBx} \\
F_{32y} &= k_{SB} x_{SB} + R_{SB} v_{SBx} \\
F_{23x} &= -F_{32x} \\
F_{23y} &= -F_{32y} \\
F_{43x} &= k_{SC} x_{SC} + R_{SC} v_{SCx} \\
F_{43y} &= k_{SC} y_{SC} + R_{SC} v_{SCy} \\
F_{34x} &= -F_{43x} \\
F_{34y} &= -F_{43y} \\
F_{14} &= k_{SD} y_{SD} + R_{SD} v_{SDy}
\end{aligned}
\tag{19}
$$

where negative signs come from the fact that in Figure 5, springs and dampers are defined as being positive in extension. Eq'ns (19) reflect the fact that tension forces at point B, for example, will create negative forces F_{23x} and F_{23y} as drawn on the free body diagrams of Figure 6. In Figure 6 all constraint forces have been arbitrarily drawn in positive directions. One may instead choose to draw equal and opposite forces on the free body diagrams and remove the negative signs from Eq'ns (19). If only the equations for bodies 2 and 3 were considered, with force $\vec{F}_{32}$ omitted, then the resulting equations would be those for a double pendulum.

Assembling System Equations

While students typically did not have difficulty understanding the above equations, their lack of deep understanding of computer programming created problems when the equations had to be assembled for MATLAB coding. Students either did not write individual equations in proper input-output form, or had equations out of order with undeclared variables ending up on the right hand side. The above equations must be coded in the following order:

1. Eq'ns (17)-(18) define intermediate variables for spring and damper endpoint displacements and velocities, in terms of state variables.
2. Eq'ns (19) define spring-damper forces and constraint forces in terms of the intermediate variables in step 1.
3. Eq'ns (15)-(16) calculate friction forces and torques.
4. Eq'ns (12)-(14) calculate state derivatives in terms of state variables, constraint forces from step 2, known inputs, and other generalized forces and torques from step 3.

Initial Conditions

Initial positions and velocities must be specified to the m-file that contains the *ode* command. For rigid joint formulations, consistent initial conditions are required. In the parasitic formulation, departures from consistent initial conditions are permissible. However, poor estimates of initial conditions will create excessive initial deflections of parasitic springs, giving high-frequency transients that can cause incorrect simulation results. Students are instructed to manually calculate initial conditions as accurately as possible using methods such as a Newton-Raphson routine for solving nonlinear position equations[6] or mechanism-specific closed-form solutions[14]. Nonetheless, the parasitic formulation allows the student to roughly calculate position initial conditions, and then refine them by simulating the mechanism with applied forces and torques removed. Initial parasitic transients will eventually die out and the mechanism will come to rest at a consistent configuration. This configuration can then be used as the initial conditions for the mechanism.

The initial conditions for the slider-crank problem can be easily calculated to many decimal places using simple trigonometry. However, consider the following set of initial conditions corresponding to an initial crank angle of 45^o. The initial angle of link 3 is erroneously stated as approximately 8^o (0.14 rad) instead of -8^o, violating the constraint that point C_3 must lie along the surface $y=0$.

$$[x_{G2}, v_{G2x}, y_{G2}, v_{G2y}, \theta_2, \omega_2, x_{G3}, v_{G3x}, y_{G3}, v_{G3y}, \theta_3, \omega_3, x_{G4}, v_{G4x}, y_{G4}, v_{G4y}]^T|_{t=0} =$$
$$[0.07, 0, 0.07, 0, 0.79, 0, 0.64, 0, 0.07, 0, \mathbf{0.14}, 0, 1.13, 0, 0, 0]^T$$

To demonstrate how the parasitic formulation can refine these initial conditions, the mechanism was simulated with the crank angle fixed at 45^o until a steady state was reached. Figure 7 shows rapid angular response of link 3 due to the initial condition inconsistency. Steady state was reached when the parasitic spring motions became damped out. The state vector at the end of 0.5 seconds,

$$[x_{G2}, v_{G2x}, y_{G2}, v_{G2y}, \theta_2, \omega_2, x_{G3}, v_{G3x}, y_{G3}, v_{G3y}, \theta_3, \omega_3, x_{G4}, v_{G4x}, y_{G4}, v_{G4y}]^T =$$
$$[0.0707, 0, 0.0707, 0, 0.785398, -0.588762, 0.636413, 0.000827, 0.070701, 0.000078,$$
$$\mathbf{-0.141878}, 0.000366, 1.131389, 0.0010533, 0, 0.000143]^T$$

was then used as the initial conditions. The simulation was re-run, with negligible initial transients as shown in the dashed line plot of Figure 7.

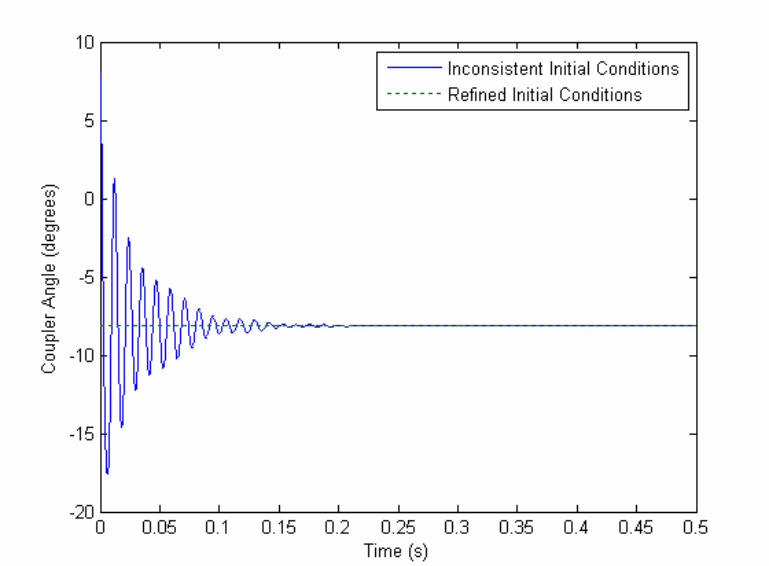

Figure 7 – Coupler angle transients due to inconsistent initial conditions

Simulation Results

Parameters for the slider crank mechanism are given in Appendix A. A constant torque of 10 N-m was applied to the crank, which began at an angle of 45°. The resulting coupler and slider motions are given in Figures 8 and 9. MATLAB's *ode15s* integrator was used, with absolute and relative integration tolerances of 10^{-6}. Figure 8 gives the crank angle time response. The vertical displacement of the slider, ideally zero, is shown to vary slightly from zero (on the order of fractions of a millimeter) as the parasitic spring k_{SD} deflects vertically. Figure 9 shows correct kinematic response of a slider crank based on the dimensions given in Appendix A.

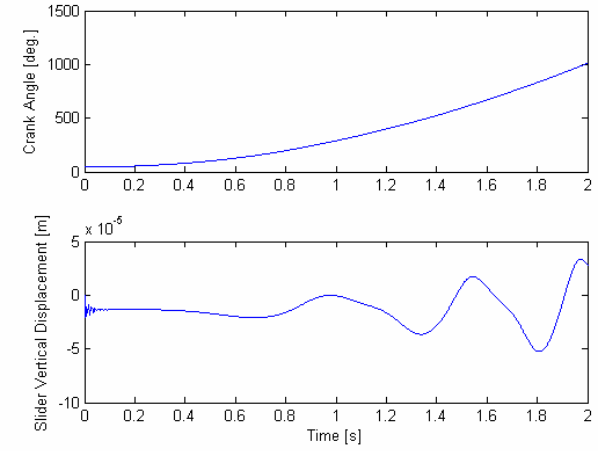

Figure 8 – Time response of slider-crank

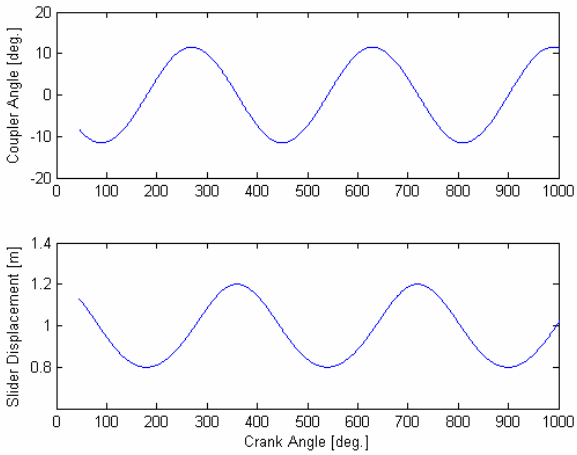

Figure 9 – Kinematic response

Implementation Issues and Discussion

Numerical Stiffness and Tuning Parasitic Parameters

The parasitic formulation is ideal for illustrating the problems of numerical stiffness (the presence of high-frequency dynamics that necessitate small time steps and thus elongated

simulation times). Direct simulation of DAE's can be more efficient, but as an educational tool the parasitic formulation allows the student to explore:

- suitability of different integration schemes, such as MATLAB's *ode45* (non-stiff systems) and *ode15s* (stiff systems such as the ones in question)
- increased simulation time as parasitic spring stiffnesses and damping coefficients increase
- decreased positional accuracy as parasitic spring stiffnesses and damping coefficients decrease
- effect of integration tolerances on simulation accuracy.

Numerical methods theory is not emphasized in the course. Instead, typical rules of thumb are applied in setting stiffnesses, damping constants and integration tolerances:

- increase stiffness until maximum spring deflection is below a certain threshold
- add damping so that initial parasitic transients die out in a few cycles
- start with loose integration tolerances, tighten until you see no change in the results.

Student Difficulties and Feedback

Typical student challenges were

- consistency in labeling equal and opposite constraint forces on free body diagrams
- writing equations in the correct order for MATLAB simulation
- understanding the difference between instantaneous and general position, leading to confusion in writing constraints and spring displacements in terms of positive angles
- program debugging and troubleshooting.

In spite of these difficulties, students showed ability to create and execute code in MATLAB even though at Memorial University the only programming courses are in C++. The comparative user-friendliness of MATLAB allowed students with minimal experience with the software to perform the course tasks after completing two one-hour self-guided tutorials.

Appendix B summarizes the results of a student questionnaire, administered after the end of the course (and after graduation). Six of 18 students responded. Overall, respondents agreed that the formalism as presented increased their ability to perform forward dynamics simulations, facilitated the handling of internal constraint forces, prepared them for simulation jobs in industry, and emphasized the importance of initial conditions and correct order of equation solution.

Pedagogical Overhead

Over a 12-week semester, approximately 2.5 weeks (7-8 50-minute lectures) are devoted to discussing

- problems in dealing with constraint forces when applying Newton's laws to multi-body systems
- numerical methods issues such as why computers prefer first-order equation forms; numerical stiffness, and implicit vs. explicit vs. differential-algebraic equations
- use of parasitic elements to express constraint forces in terms of state variables

- the general formalism and example applications

Forward dynamics simulation is incorporated into a student project in which the response of a spring-loaded catapult must be simulated. The catapult is then subjected to an inverse-dynamics analysis to determine the required motor torque to reset the mechanism.

Summary and Conclusions

A simple, physically-intuitive multi-body dynamics formulation has been presented for use in undergraduate engineering curricula where students have not been exposed to forward dynamics simulation of the types of mechanisms studied in mechanism kinematics courses. The formulation uses Newton's Laws for planar mechanisms, and expresses internal constraint forces in terms of mass centre position and velocity by replacing ideal rigid joints with stiff (parasitic) springs and dampers. Instead of differential-algebraic equations (DAE's), a set of explicit first-order ODE's results. These equations can be coded and simulated in MATLAB to predict mechanism motion resulting from applied forces and torques. The formulation allows students to hone programming skills, gain further expertise in MATLAB, gain an understanding of how to handle constraint forces within mechanisms, and explore the effects of numerical stiffness and initial conditions on simulation accuracy and efficiency. The method prepares graduates to use commercial multi-body software and interpret simulation results, or can be a starting point for more advanced study in areas such as Lagrangian dynamics. The results of a student survey suggest that the above benefits indeed accrued to the students. This simple approach to multi-body dynamics can fill a gap in many students' engineering educations in an area to which they may not have been exposed for several semesters.

Bibliography

1. Memorial University Faculty of Engineering and Applied Science calendar, http://www.mun.ca/regoff/calendar.
2. University of Calgary Mechanical Engineering course descriptions, http://www.ucalgary.ca/pubs/calendar/current/What/Courses/ENME.htm.
3. University of Toronto Mechanical and Industrial Engineering course descriptions, http://www.mie.utoronto.ca/undergrad/mechut.htm.
4. University of Michigan Mechanical Engineering course descriptions, http://me.engin.umich.edu/current/course_information.shtml
5. University of California – San Diego Mechanical and Aerospace Engineering course descriptions, http://maeweb.ucsd.edu/undergrad/courses/courses.php
6. Gardner, J.F. (2001) *Simulations of Machines Using MATLAB and Simulink*. Wadsworth Group, Thomson Learning, Inc.
7. Cavacece, M. *et al.*, (2005) "Experiences in Teaching Multi-body Dynamics", *Multibody System Dynamics* **13**: 363-369.
8. Fisette, P. and Samin, J.C. (2005) "Teaching Multi-body Dynamics from Modeling to Animation", *Multibody System Dynamics* **13**: 339-351.
9. Fraczek, J. and Wojtyra, M. (2005) "Teaching Multi-body Dynamics at Warsaw University of Technology", *Multibody System Dynamics* **13**: 353-361.
10. Karnopp, D.C. (1997) "Understanding Multi-body Dynamics Using Bond Graph Representations." *Franklin Institute Journal*, 334B, No.4, pp.631-642.

11. Bendtsen, C., and Thomsen, P.G. (Eds.) (1999) *Numerical Solution of Differential Algebraic Equations*, IMM Department of Mathematical Modelling Technical Report IMM-REP-1999-8, Technical University of Denmark, Lyngby, Denmark.
12. Karnopp, D.C., Margolis, D.L., and Rosenberg, R.C. (2006) *System Dynamics: Modeling and Simulation of Mechatronic Systems (4th Ed.)*. John Wiley & Sons, Inc.
13. Karnopp, D.C. and Margolis, D.L. (1979) "Analysis and Simulation of Planar Mechanism Systems Using Bond Graphs", *Journal of Mechanical Design*, April 1979, pp. 187-191.
14. Waldron, K.J., and Kinzel, G.L. (2004) *Kinematics, Dynamics and Design of Machinery*. John Wiley & Sons, Inc.

Appendix A – Model Parameters

Parameter	Value	Units	Description
m_2	1	kg	crank mass
m_3	1	kg	coupler mass
m_4	1	kg	slider mass
I_{G2}	1	kg-m^2	crank moment of inertia
I_{G3}	1	kg-m^2	coupler moment of inertia
A_2B_2	0.2	m	crank length
B_3C_3	1	m	coupler length
k_{sA}, k_{sB}, k_{sC}, k_{sD}	10^6	N/m	parasitic stiffnesses, points A, B, C, D
R_{sA}, R_{sB}, R_{sC}, R_{sD}	200	N-s/m	parasitic damping, points A, B, C, D
T(t)	10	N-m	input torque
b	2	N-s/rad	pin C rotational damping constant
c	5	N-s/m	slider viscous damping constant

Appendix B – Student Questionnaire and Results

Questionnaire regarding ENGI7945 Machine Dynamics

Possible Answers

Strongly agree (5) Agree (4) Neutral (3) Disagree(2) Strongly disagree (1)

Questions and average scores (standard deviation in parentheses):

1. Prior to this course, I would not have been able to generate and numerically integrate differential equations for forward dynamics of simple multi-body systems

4.33 (0.52)

2. As a result of this course, I can predict the motion of simple mechanisms, given mass properties, geometry, and applied forces and torques

4.50 (0.55)

3. The multi-body formulation taught in the course, which uses stiff "parasitic" joint springs and dampers to generate equations:

a) makes it easier to write equations that can be numerically integrated using MATLAB

4.17 (0.75)

b) has increased my understanding of "numerical stiffness" and the effects of integration tolerances

3.50 (0.55)

c) provided insight into the importance of consistent initial conditions

4.33 (0.52)

d) makes it easier to deal with constraint forces when writing a system of equations for simulation

4.00 (0.63)

e) makes equation generation more complicated and makes troubleshooting of incorrect output more difficult

2.83 (0.41)

4. The use of MATLAB:

a) has increased my understanding of the importance of coding equations in the correct order

4.17 (0.41)

b) has increased my familiarity of different numerical integration methods

3.33 (0.82)

5. The multi-body dynamics portion of the course, and use of MATLAB, have better prepared me for jobs in industry requiring computer simulation.

4.00 (0.89)