DynaFlexPro for Maple

RAJANKUMAR M. BHATT and VENKAT N. KROVI

DynaFlexPro (DFP), released in the middle of 2005 by MotionPro, Inc., and currently in version 2.2.8, is marketed by MapleSoft, Inc., of Waterloo, Ontario [1]. DFP is a Maple toolbox that allows control engineers to symbolically create mathematical models for analyzing the dynamic behavior of large articulated multibody systems. DFP combines graph-theoretic modeling techniques with Maple's computer-algebra manipulation capabilities to automatically create the symbolic equations of motion (EOMs).

BACKGROUND

Machines and mechanisms typically possess an underlying articulated skeleton composed of multiple rigid or flexible bodies interconnected by joints. Examples range from piston cranks in internal combustion engines, automobile steering and suspension mechanisms, aircraft landing gear and control-surface mechanisms, to CD disk changers, reclining chair mechanisms, and windup toy mechanisms, to automated processing, packaging, and handling machines in industry. The spatial layout of bodies and joints determines the permissible and restricted motions and forces and thereby governs the transmission, modulation, and redirection of energy from the source into a usable mechanical form.

Designers have created, refined, and tailored *articulated multibody systems*, with various types and numbers of bodies and joints, to match the diversity of end applications. Early efforts relied on the intuition of the designer to understand the interactions among bodies and thus the overall system behavior. The term *system behavior* encompasses both the motions and forces experienced within the articulated multibody system as well as with respect to the environment. Even as late as the 1950s, the design and control of multibody systems was the realm of master-designers and their apprentices due to a lack of mathematical modeling and analysis methods. More importantly, it was faster and easier to build physical prototypes and test controllers in situ than to analyze the devices using mathematical techniques.

Modern control techniques use plant models to analyze system behaviors, which for multibody systems can be created from the governing EOMs. Linearized plant models are desirable from the linear-superposition-based analysis perspective but can adequately capture system behavior only for a limited range of inputs and operating conditions. Consequently, controlling systems such as robots and disk drives requires full nonlinear dynamic models to achieve enhanced performance over a wide range of operating conditions.

COMPLEXITY OF MULTIBODY DYNAMICS

Textbooks such as [2]–[6] discuss the challenges encountered in modeling and analyzing the governing EOMs of articulated multibody systems. In their simplest form, the governing EOMs are a system of constant-coefficient linear ordinary differential equations (ODEs). For example, linear ODEs can model the dynamics of one-dimensional (1D) translational systems such as strings of springs, masses, and dampers. However, rotations in two and three dimensions introduce nonlinearities in the form of trigonometric terms with multivalued inverses, which make the mathematical handling difficult. These difficulties are compounded when transitioning from planar [two-dimensional (2D)] systems to spatial [three-dimensional (3D)] systems.

Second, many multibody systems such as active car suspensions and flight simulators possess one or more closed kinematic loops to enhance stiffness and payload capacity. In effect, these loops reduce actuation requirements by creating algebraic equality constraints that must be satisfied by the system's degrees of freedom (DOFs). The algebraic constraints combine with the ODEs to comprise a system of differential algebraic equations (DAEs). Hence, the modeling and performance analysis of such multibody systems requires the ability to mathematically and computationally handle DAE systems.

Finally, products such as disk-drive read heads, composite-fiber bicycle frames, and microelectromechanical-systems (MEMS) devices are designed to be lightweight and fast, in which case the effects of body flexibility become increasingly important. These effects can be modeled by using systems of partial differential equations (PDEs) in place of rigid-body ODEs, so that a flexible multibody system with closed loops entails a system of PDE-based DAEs. The spatially dependent PDE-based system is more difficult to handle, both mathematically and computationally, when compared to the original ODE-based system.

SOFTWARE PACKAGES FOR MULTIBODY DYNAMICS

Over the past three decades, numerous multibody computer-aided engineering (MCAE) software packages have been

DFP is a Maple toolbox that allows control engineers to symbolically create mathematical models for analyzing the dynamic behavior of large articulated multibody systems.

developed to automate and simplify the modeling and analysis of multibody system performance; well-known examples include DADS, ADAMS, Working Model, visualNASTRAN, and SD/FAST [7]–[10]. These tools allow the user to piece together multibody systems by specifying the spatial layout of the components and interconnections within a 3D graphical user interface. More importantly, the formulation and solution engines allow the user to simulate and analyze the multibody system under a variety of initial conditions and inputs. However, these software packages do not give the user direct access to the underlying EOMs. Hence, control methods that depend on explicit equations would require either an independent derivation or a technique for fitting models by system identification.

Several noncommercial (and typically unsupported) packages have been created for the automatic symbolic generation of the EOMs of multibody systems, leveraging general-purpose symbolic manipulation engines such as Maple and Mathematica. For example, MBsymba [11], [12] is a Maple-based GNU freeware licensed package for symbolically modeling multibody systems that implements both Lagrangian and Newton-Euler formulations. However, the constraints imposed by symbolic manipulation of large matrices, the required symbolic differentiations, or the unavailability of appropriate recursive formulations for closed-loop multibody systems [13] limit these packages to medium-sized multibody systems.

DYNAFLEXPRO FEATURES

DFP offers unique capabilities that help overcome some of the limitations noted above. Specifically, DFP uses a graph-theoretic-modeling approach to create kinematic and dynamic EOMs within a systematic and automated symbolic implementation [14]–[17]. While a symbolic or numeric study of the system is possible using Maple's built-in ODE solver dsolve, DFP also offers the capability to export the EOMs to other platforms (C, Fortran, and Matlab) using code-generation tools. Although SimMechanics/RealTimeWorkshop [18] also offers code-generation capabilities for real-time multibody simulations, DFP has a distinct advantage since it uses the symbolic EOMs as the basis for code generation. The resulting optimized and thus efficient code is better-suited for real-time simulation and real-time control implementations.

ILLUSTRATIVE CASE-STUDY

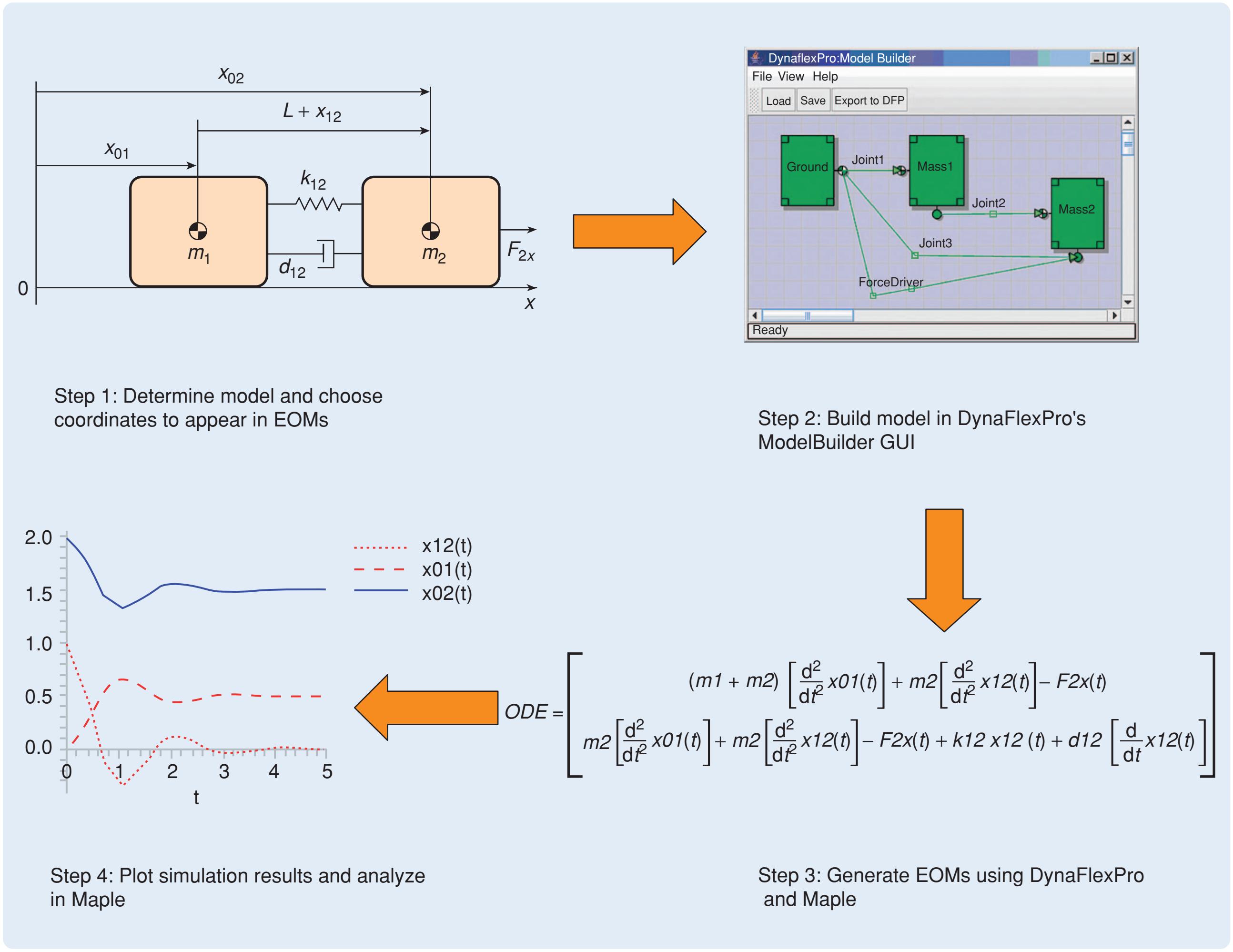
We now illustrate the various features of DFP using a case-study of a system of two masses connected by a linear spring and damper. All of the results given below are generated using DynaFlexPro 2.2.8 with Maple 10.02 on a 1.7 GHz Pentium-M PC with 256 MB RAM running Windows XP.

Figure 1 illustrates the four stages of creating and simulating the EOMs. The first stage involves the construction of frames of reference, interconnections between bodies, and application of motion and force drivers. In the second stage, a Java-based GUI ModelBuilder (MB) aids the user in translating the modeling decisions of the first stage into a form suitable for further processing by the set of Maple routines. The automated generation of symbolic EOMs from the ASCII based *.dfp file forms the third stage, culminating in the creation of a Maple module providing multiple access methods to the EOM data. The fourth stage is the simulation of the system dynamic equations that can either use Maple's numeric and symbolic capabilities or can be implemented on an external simulation platform.

Stage 1: Preliminary Modeling

Consider the system of two masses coupled by a spring and damper shown in Figure 2. The two masses are assumed to be at rest separated by a distance L when a spring having zero rest length and a translational damper are attached. A time-varying external force $F_{2x}(t)$ is assumed to be acting on the second mass. We assume that the absolute translational motions of the two masses are constrained along the global x axis.

The selection of generalized coordinates to describe the system's DOFs can create significantly different formulations. Absolute generalized coordinates can be created in terms of the displacements $x_{0i}(t)$ of the centers of mass m_i for each of the two bodies $(i = 1, 2)$ from a preferred inertial frame. Alternatively, the relative displacements $x_{01}(t)$ and $x_{12}(t)$ of various bodies can serve as relative generalized coordinates, $x_{12}(t)$ where denotes the relative displacement m_2 of with respect to m_1 from the equilibrium configuration, that is, $x_{12}(t) = x_{02}(t) - L - x_{01}(t)$. Alternative coordinate descriptions are possible, for example, the motion of the system's center of mass (rigid mode) as well as the relative motions of various bodies (flexible modes) can be used in a vibration analysis setting. We assume that the initial conditions in terms of absolute coordinates are $x_{01}(0) = 0$ and $x_{02}(0) = L + \delta$, where δ is the initial displacement of the mass m_2 from its equilibrium position. Correspondingly, in terms of relative coordinates we assume $x_{01}(0) = 0$ and $x_{12}(0) = \delta$.



x_{02}
$L + x_{12}$
x_{01}
k_{12}
m_1
d_{12}
m_2
F_{2x}
0
x
DynaflexPro:Model Builder
File View Help
Load Save Export to DFP
Ground
Joint1
Mass1
Joint2
Mass2
Joint3
ForceDriver
Ready
Step 1: Determine model and choose coordinates to appear in EOMs
Step 2: Build model in DynaFlexPro's ModelBuilder GUI
2.0
1.5
1.0
0.5
0.0
0 1 2 3 4 5
t
x12(t)
x01(t)
x02(t)
$ODE = \left[\begin{array}{c} (m1+m2)\left[\frac{d^2}{dt^2}x01(t)\right] + m2\left[\frac{d^2}{dt^2}x12(t)\right] - F2x(t) \\ m2\left[\frac{d^2}{dt^2}x01(t)\right] + m2\left[\frac{d^2}{dt^2}x12(t)\right] - F2x(t) + k12\,x12\,(t) + d12\left[\frac{d}{dt}x12(t)\right] \end{array}\right]$
Step 4: Plot simulation results and analyze in Maple
Step 3: Generate EOMs using DynaFlexPro and Maple

FIGURE 1 The four stages of simulating a system with DynaFlexPro. The user selects model topology and variables of interest in Stage 1 and thus retains influence over the form of the resulting EOMs. In Stage 2, a GUI helps translate this user intent into an appropriate DFP model. Stage 3 then facilitates the use of high-level DFP commands for the automated generation of symbolic EOMs that can be parametrically simulated in Stage 4.

Stage 2: ModelBuilder and DFP File Creation

DFP supports a library of modeling components, including rigid bodies, flexible beams, forces, torques, springs, dampers, and a variety of joints. System models are generated by piecing together individual components available from the library and exporting the resulting model as an ASCII DFP input file (*.dfp file). A Java-based MB graphical user interface (GUI) considerably simplifies the process of building the system model in block-diagram form, visualization of various interconnections, setting of various parameters, and exporting the final ASCII *.dfp file. However, an experienced user can quickly create or modify a system *.dfp file by using only an ASCII text editor.

Two mechanical body elements are added to the default ground body in an MB window to create the model for a linear spring-mass-damper system. The primary frame of reference for each rigid body is a center of mass (COM) frame, represented by a crossed circle symbol. Alternative body-fixed frames can be defined relative to this frame at appropriate points-of-interest, such as locations of joints or the application of external forces.

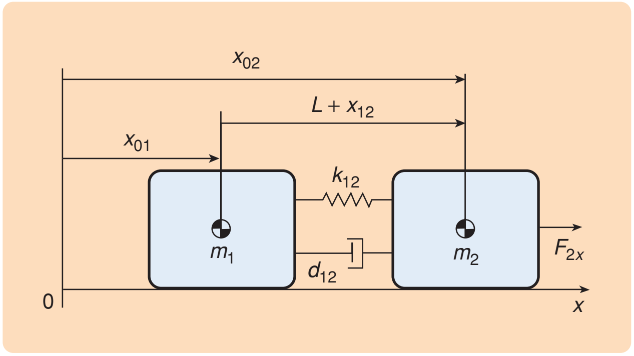
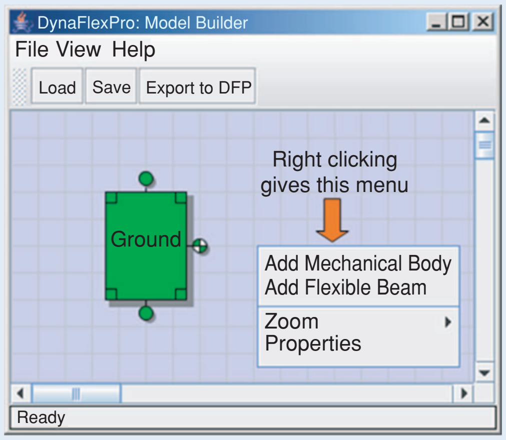


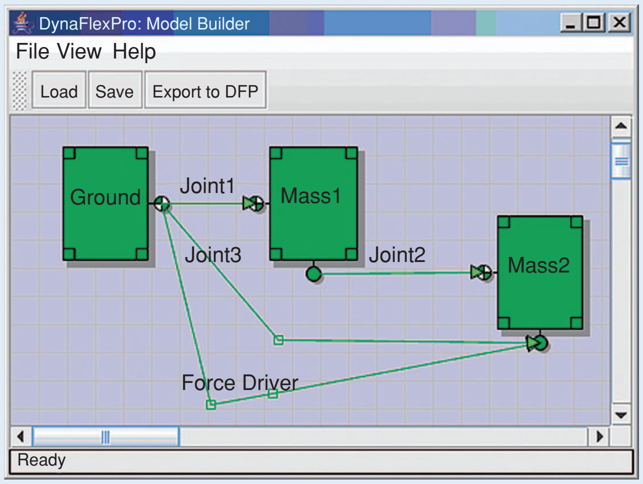
FIGURE 2 A translational two-mass-spring-damper system. The case-study illustrates the capabilities of DFP for formulating EOMs in relative or absolute coordinates, solving them in numeric and analytic form, performing forward- and inverse-dynamics analyses, incorporating nonlinear elements, performing sensitivity analysis, and exporting to external packages.

On Mass 1, Frame 1 is located at a distance L along the local (positive) x axis, while Frame 2 on Mass 2 is colocated with the COM frame. The lumped mass at the COM is denoted by $m_1 (i = 1, 2)$. The inertia defined in the local COM reference frame can be any positive-definite matrix whose eigenvalues satisfy the triangle property as required by realistic bodies. However, if the COM frame is aligned with the principal mass distribution axes, then the inertia is a diagonal matrix. Properties for the two rigid-bodies are assigned in symbolic form, as shown in Figure 3(d).

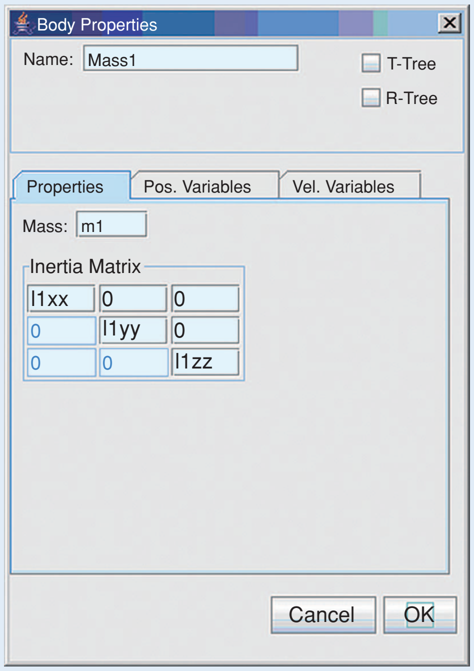
Joint elements are created by dragging a directed arrow to connect appropriate frames as shown in Figure 3(b). The joint properties window shown in Figure 4(a) facilitates the selection of joint types, orientation of critical axes, naming of user variables in numeric or symbolic form, and assignment of motion- or force- drivers. Joints 1, 2, and 3 are selected to be prismatic joints acting along the positive x axis. In this example, Joint 3 is a redundant constraint added to facilitate later derivations in absolute coordinates. One of the unique features of DFP is that redundant constraints can be detected and



(a)



(b)



Name: Mass1

Mass: m1

Inertia Matrix

I1xx	0	0
0	I1yy	0
0	0	I1zz

(c)

Property	Body Element 1	Body Element 2
Name	Mass1	Mass2
Mass	M1	M2
Inertia	$\begin{bmatrix} I1xx & 0 & 0 \\ 0 & I1yy & 0 \\ 0 & 0 & I1zz \end{bmatrix}$	$\begin{bmatrix} I2xx & 0 & 0 \\ 0 & I2yy & 0 \\ 0 & 0 & I2zz \end{bmatrix}$
T-tree	Unchecked	Unchecked
R-tree	Unchecked	Unchecked

(d)

FIGURE 3 The ModelBuilder GUI helps create the DFP input file. (a) A new model window; (b) two-mass-spring-damper model; (c) a typical rigid-body element-properties window; and (d) assigned rigid-body properties in the two-mass-spring-damper model. The point-and-click interface simplifies adding bodies, connecting them with articulations, and setting various properties in numeric and symbolic form, as shown for the two-mass-spring-damper case.

automatically eliminated from the EOMs. While springs and dampers can be individually modeled, parallel stiffness and damping can be directly included with the prismatic-joint definition as done for Joint 2. The assigned properties for each of the three joints are given in Figure 4(b).

Careful selection of a body, joint, or force element into the T-tree or R-tree allows the user to specify the variables for the final formulation. Such a selection must also ensure that all frames are connected without the formation of closed loops, as discussed below.

Stage 3: Generation of EOMs from the *.dfp File

The process of generating the EOMs is accomplished using the four lines of code (see Figure 5) once the DFP input file "DoubleMSD.dfp" is available.

Stage 4: Simulating the System

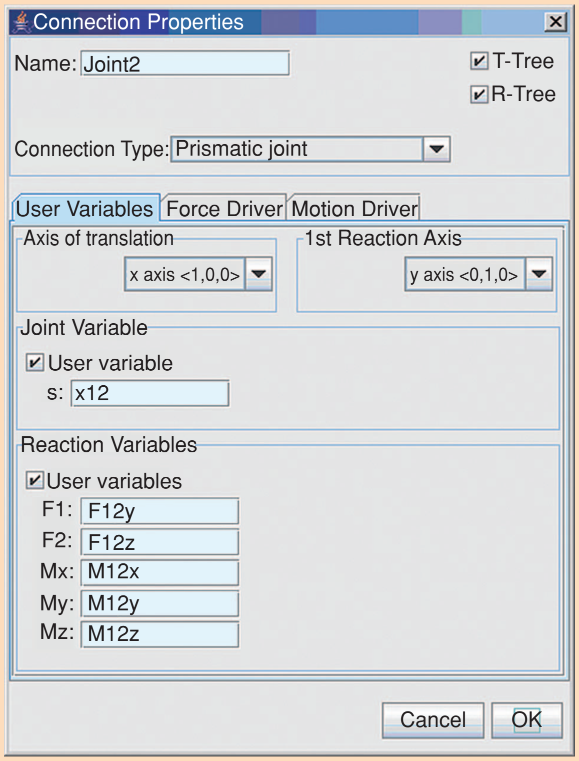
Forward Dynamics

The user has the ability to generate solutions for the EOMs in any desired combination of symbolic and numeric form, taking advantage of Maple's unique capabilities. We assigned numeric values for each of the masses ($m_i = 1\,\text{kg}, i = 1, 2$), damping coefficient ($d_{12} = 1\,\text{N-s/m}$), and spring constant ($k_{12} = 5\,\text{N/m}$). Setting $F_{2x}(t) = 0$ and $\delta = 0.5$ m allows us to compute an analytical expression for the initial-condition, zero-input system response UnforcedSoln, which is then plotted in Figure 6(b).

DFP supports a library of modeling components, including rigid bodies, flexible beams, forces, torques, springs, dampers, and a variety of joints.

In cases where such an analytic solution might not be feasible, dsolve/numeric can be used directly on the EOMs. However, invoking BuildSimCode converts the EOMs into a Maple function or Maple procedure as shown in Figure 7. Faster optimized simulations result when this code is numerically integrated within Maple, in which case the results are identical to those shown in Figure 6(b).

Alternatively, exporting the EOMs to other platforms such C, Fortran, or Matlab is possible using the code generation module of Maple. Figure 8 depicts the optimized



Name: Joint2
☑ T-Tree
☑ R-Tree
Connection Type: Prismatic joint
Axis of translation: x axis <1,0,0>
1st Reaction Axis: y axis <0,1,0>
Joint Variable
☑ User variable
s: x12
Reaction Variables
☑ User variables
F1: F12y
F2: F12z
Mx: M12x
My: M12y
Mz: M12z

(a)

Property	Joint 1	Joint 2	Joint 3
Name	Joint1	Joint2	Joint3
T-tree	Checked	Checked	Unchecked
R-tree	Checked	Checked	Unchecked
Connection Type	Prismatic Joint	Prismatic Joint	Prismatic Joint
Axis of Translation	x axis<1,0,0>	x axis<1,0,0>	x axis<1,0,0>
First Reaction Axis	y axis<0,1,0>	y axis<0,1,0>	y axis<0,1,0>
Variables	x01	x12	x02
Force Driver			
Spring Constant [K]	-	k12	-
Damping Coefficient	-	d12	-

(b)

FIGURE 4 Selecting interconnection properties. (a) Typical joint-element properties window and (b) assigned symbolic values for the three joints of the two-mass-spring-damper case. In addition to simplifying the selection of properties, the interface permits the user to specify which variables are to appear in the EOMs in either numeric or symbolic form.

Matlab code for the pXdot procedure generated by the BuildSimCode module.

Sensitivity Analysis

Dynamic parameters from the symbolic EOMs are accessible for parametric analysis. In this example, the EOMs can be solved in closed form while retaining one of the masses as a parameter m_1 and a closed-form expression obtained for the resulting sensitivity surface. Maple's function plot3d is used to visualize the effect of this parameter on the solution $x_{01}(t)$ in Figure 9.

Inverse Dynamics

The governing equations generated by DFP can also be used for inverse dynamics calculations. In a prescribed motion formulation, we specify a sinusoidal motion $x_{01} = \sin(t)$ with unit amplitude for mass m_1. The resulting equations can be solved to determine the required force $F_{2x}(t)$ and the relative displacement of the two masses $x_{12}(t)$ as shown in Figure 10.

The flexibility and power of DFP are evident once a basic model for a system is ready. By changing only a few parameters within the MB interface or DFP command line instructions, the dynamic equations for systems can be

```
with(DynaFlexPro):
dfpFile := cat(GetDFPdir( ), "/DoubleMSD.dfp"):
Model := BuildEQs(dfpFile,["DynSimpType", "Simplify"]):
with(Model): ODE := GetDynEQs( )
```

```
"Analyzing system..."
"Performing constraint analysis..."
"The system has 2 degree(s) of freedom. It is modeled using 2 generalized coordinate(s) coupled by 0 algebraic constraint(s)."
"Peforming a dynamic analysis on the constraint free system - system variables shown below:"
```

$$\text{"vQ"}, \begin{bmatrix} x01(t) \\ x12(t) \end{bmatrix}, \text{"vP"}, \begin{bmatrix} \frac{d}{dt}\, x01(t) \\ \frac{d}{dt}\, x12(t) \end{bmatrix}, \text{"vPdot"}, \begin{bmatrix} \frac{d^2}{dt^2}\, x01(t) \\ \frac{d^2}{dt^2}\, x12(t) \end{bmatrix}$$

```
"Dynamic analysis complete."
```

$$ODE := \begin{bmatrix} (m1+m2)\left(\frac{d^2}{dt^2}\, x01(t)\right) + m2\left(\frac{d^2}{dt^2}\, x12(t)\right) - F2x(t) \\ m2\left(\frac{d^2}{dt^2}\, x01(t)\right) + m2\left(\frac{d^2}{dt^2}\, x12(t)\right) - F2x(t) + k12x12(t) + d12\left(\frac{d}{dt}\, x12(t)\right) \end{bmatrix}$$

FIGURE 5 Generating EOMs in relative coordinates $x_{01}(t), x_{12}(t)$. These symbolic EOMs can be generated using only four lines of DFP code operating on the DFP input file. However, the user retains significant control over the types of analyses as well as the form of the EOMs through the command options.

```
ParaSubs := [m1 = 1, m2 = 1, d12 = 1, k12 = 5, F2x(t) = 0, L = 1]
ics := x01(0) = 0, x12(0) = 0.5, D(x01) (0) = 0, D(x12) (0) = 0
SysODE := eval(ODE, ParaSubs)
```

$$SysODE := \begin{bmatrix} 2\left(\frac{d^2}{d^2}\, x01(t)\right) + \frac{d^2}{dt^2}\, x12(t) \\ \frac{d^2}{dt^2}\, x01(t) + \frac{d^2}{dt^2}\, x12(t) + 5x12(t) + \frac{d}{dt}\, x12(t) \end{bmatrix}$$

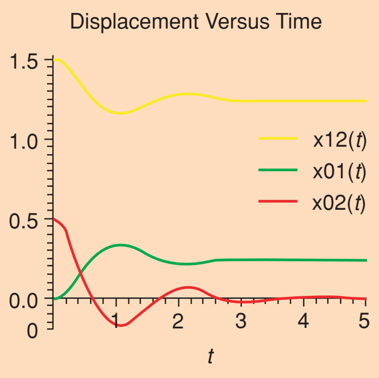
```
mySys := SysODE[1], SysODE[2]
```

$$mySys := 2\left(\frac{d^2}{dt^2}\, x01(t)\right) + \frac{d^2}{dt^2}\, x12(t),\ \frac{d^2}{dt^2}\, x01(t) + \frac{d^2}{dt^2}\, x12(t) + 5x12(t) + \frac{d}{dt}\, x12(t)$$

```
UnforcedSoln := dsolve( [mySys, ics] )
```

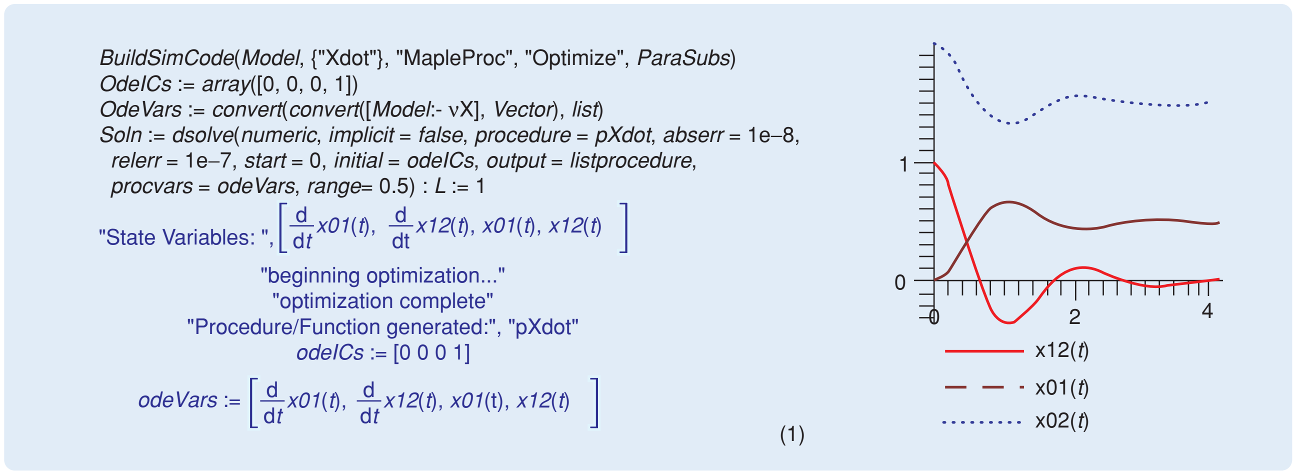
$$UnforcedSoln := \left\{x12(t) = e^{-t}\left(\frac{1}{6}\sin(3t) + \frac{1}{2}\cos(3t)\right),\ x01(t) = -\frac{1}{12}\, e^{-t}\sin(3t) - \frac{1}{4}\, e^{-t}\cos(3t)\right\}$$

(a)



(b)

FIGURE 6 Analytical forward dynamics. (a) Code for obtaining closed-form initial-condition response and (b) parametric plots of the resulting solution. DFP takes advantage of Maple's symbolic manipulation capabilities to obtain an analytical expression for system responses that can subsequently be plotted.



```
BuildSimCode(Model, {"Xdot"}, "MapleProc", "Optimize", ParaSubs)
OdeICs := array([0, 0, 0, 1])
OdeVars := convert(convert([Model:- vX], Vector), list)
Soln := dsolve(numeric, implicit = false, procedure = pXdot, abserr = 1e–8,
  relerr = 1e–7, start = 0, initial = odeICs, output = listprocedure,
  procvars = odeVars, range= 0.5) : L := 1
```

"State Variables: ", $\left[\frac{d}{dt}x01(t), \frac{d}{dt}x12(t), x01(t), x12(t)\right]$

"beginning optimization..."
"optimization complete"
"Procedure/Function generated:", "pXdot"
odeICs := [0 0 0 1]

$$odeVars := \left[\frac{d}{dt}x01(t), \frac{d}{dt}x12(t), x01(t), x12(t)\right] \qquad (1)$$

FIGURE 7 Numerical forward-dynamic solutions. When an analytic solution is not feasible or desirable, the EOMs can either be solved numerically using dsolve or converted into Maple functions and procedures. The latter approach facilitates efficient and stable numerical solution particularly for systems with differential-algebraic EOMs.

created in a variety of coordinate frames or formulations, as discussed below.

Modeling Nonlinear Components

DFP simplifies modeling of the nonlinear constitutive behavior of components. For instance, changing the spring constant and damping coefficient properties of Joint 2 to 'k12*x12(t)^2' and 'd12/sqrt(diff(x12(t),t)),' respectively, allows us to model the nonlinear force-displacement behavior $F_k = k_{12}x_{12}^3$ of the spring or force-velocity $F_d = d_{12}\sqrt{\dot{x}_{12}}$ behavior of the damper element. The resulting EOMs generated for such a system are shown in Figure 11.

Absolute Coordinates

In some cases, it might be desirable to have EOMs in absolute coordinates. Choosing appropriate body and joint elements for the spanning tree allows the user to select the variables that appear in the final EOMs. In this case, selecting Joint 3 in T-tree and R-Tree (in place of Joint 2) and reevaluating the same worksheet generates the EOMs in absolute coordinates as shown in Figure 12. Again, the third redundant constraint, arising from Joint 2, is automatically detected and eliminated.

Augmented Lagrangian Constrained Formulation

Introducing constraints between DOFs of a multibody system has traditionally complicated the formulation and solution of the governing equations. However, DFP can handle such constraints and automatically regenerate the EOMs. For example, a position-level constraint can be introduced between $x_{01}(t)$ and $x_{02}(t)$ by changing Joint 2 from a spring-damper to a weld joint within the MB interface. Note that we retain Joint 3 in T-tree and R-tree as in the previous subcase. Rerunning the original Maple worksheet with the modified DFP input file now creates the system EOMs in terms of an augmented Lagrangian formulation, as shown in Figure 13.

Minimal-Coordinate Constrained Formulation

Alternatively the same model can be used to create the minimal coordinate constrained formulation by simply changing the elements in the spanning tree. Restoring Joint 2 in the tree (that is, by checking the T-tree and R-tree properties) in place of Joint 3 yields a constrained formulation in terms of a minimal set of generalized coordinates, as shown in Figure 14.

Planar Models for the Two-Mass-Spring-Damper

In all of the examples considered thus far, the system translates along the x axis. An extension to 2D planar motions can be accomplished by changing the properties of Joint1 and Joint3 to "planar." The trigonometric nonlinearities, which are tedious to work with manually, are automatically handled using DFP. The two resulting formulations, namely, minimal unconstrained ODE versus

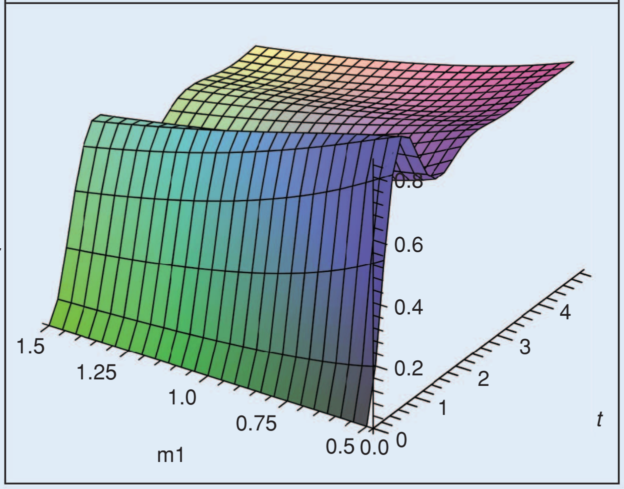
```
with(CodeGeneration) : Matlab(pXdot);
function pXdotreturn = pXdot (N, t, X_in, ODE)
    zz (2)  =  0.1e1;
    zz (1)  =  X_in (2) ;
    zz (6)  =  0.5000000000e0;
    zz (3)  =  0.1e1 *  zz (6) ;
    zz (4)  =  0.0e0;
    zz (5)  =  (–0.5e1 *  X_in (4) – 0.1e1 * zz (1) – 0.1e1 *
                zz (4) * zz (2) )  /  (0.1e1 – 0.1e1 * zz (2) * zz (3) ) ;
    ODE (1)  =  zz (4)  –  0.1e1  *  zz (5)  *  zz (3) ;
    ODE (2)  =  zz (5) ;
    ODE (3)  =  X_in (1) ;
    ODE (4)  =  zz (1) ;
    pXdotreturn  = ODE (4) ;
```

FIGURE 8 Export of EOMs to Matlab in state-space form. Additionally DFP can export the EOMs to C and Fortran taking advantage of Maple's code-generation tools. In all cases, using the symbolic EOMs as the basis for code generation creates optimized code that is well suited for real-time simulations or real-time control.

```
ParaSubs := [m2 = 1, d12 = 1, k12 = 5, F2x(t) = 0, L = 1];
SysODE := eval(ODE, ParaSubs);
mySys := SysODE[1], SysODE[2];
ics := x01(0) = 0, x12 (0) = 1, D(x01) (0) = 0, D(x12) (0) = 0;
UnforcedSoln := dsolve( [mySys, ics] );
assign(UnforcedSoln); L := 1;
```

(a)

```
plot3d(x01(t), t = 0..5, m1 = 0.5..1.5, axes = normal);
```



(b)

FIGURE 9 Parametric study of changes in mass m_1 from 0.5 to 1.5 kg with (a) DFP/Maple Code and (b) $x_{01}(t)$ versus m_1 and t. The user can take advantage of Maple's symbolic and numeric capabilities for simulation-based performance-variability studies using parametric sweeps.

augmented Lagrangian, can be created by selecting or eliminating Joints 2 and 3 within the spanning tree. The selection parameters are shown in Tables 1 and 2, while the corresponding EOMs are shown in Figure 15.

The minimal unconstrained ODE formulation is created in terms of four relative generalized coordinates corresponding to the three DOFs of Mass 1 in the plane coupled with the one DOF due to the prismatic joint. The ensuing system of four second-order differential equations in relative coordinates is shown in Figure 15(a). Figure 15(b) shows the EOMs developed using absolute coordinates of the two masses wherein six second-order ODEs coupled by two Lagrange multipliers are obtained. At first glance, it might appear that the absolute coordinate formulation gives a smaller and simpler set of equations as compared to the minimal ODE formulation. However, equations with unknown Lagrange multipliers cannot be solved independently of the two constraint equations arising from the prismatic joint, which is accessed using the procedure GetPosCons. Thus, a larger DAE system, involving the six ODEs and two nonlinear algebraic equations, must be solved to obtain the system response in contrast to the more direct solution process for the system of four ODEs of Figure 15(a).

```
x01 := t → sin(t);
m1 := 1 : m2 := 1 : k12 := 5 : b12 := 1;
ics := x12(0) = 0, D(x12) (0) = 0;
Force1 := solve( {ODE[1]}, F2x(t) );
Force2 := solve( {ODE[2]}, F2x(t) );
Force := subs(Force1, Force2);
dsolve( {ics, Force[1]}, {x12(t) } );
```

$$Force1 := \left\{ F2x(t) = -2\sin(t) + \frac{d^2}{dt^2}x12(t) \right\}$$

$$Force2 := \left\{ F2x(t) = -\sin(t) + \frac{d^2}{dt^2}x12(t) + 5x12(t) + \frac{d}{dt}x12(t) \right\}$$

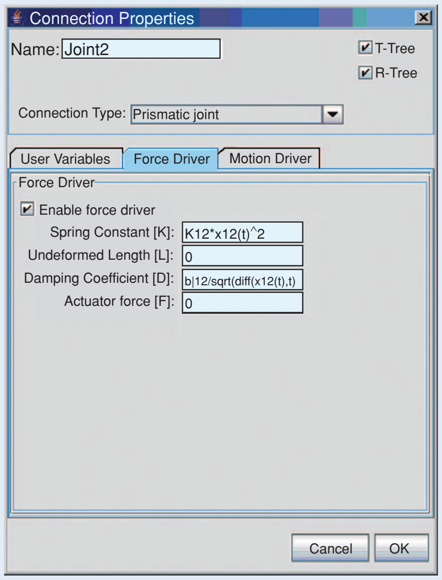
$$\text{Force} := \left\{ -2\sin(t) + \frac{d^2}{dt^2}x12(t) = -\sin(t) + \frac{d^2}{dt^2}x12(t) + 5x12(t) + \frac{d}{dt}x12(t) \right\}$$

$$x12(t) = \frac{1}{26}\cos(t) - \frac{5}{26}\sin(t) - \frac{5}{26}e^{-5t}$$

FIGURE 10 DFP/Maple code for an inverse-dynamics problem. The generated EOMs can be manipulated to solve various problems of interest. For example, by prescribing the motions of the system components, we can obtain symbolic expressions for the driving forces. This process is useful for selecting motors and controllers from a design perspective.

ADDITIONAL FEATURES OF DFP

Several features of DFP are not discussed in this review due to limited space. The DynaFlexPro Web portal [1] has many user-friendly tutorial examples and user-contributed examples in Maple E-book format that illustrate these features.



Name: Joint2

[✓] T-Tree

[✓] R-Tree

Connection Type: Prismatic joint

User Variables | Force Driver | Motion Driver

Force Driver

[✓] Enable force driver

Spring Constant [K]: K12*x12(t)^2

Undeformed Length [L]: 0

Damping Coefficient [D]: b|12/sqrt(diff(x12(t),t)

Actuator force [F]: 0

(a)

$$ODE := \begin{bmatrix} (m1+m2)\left(\frac{d^2}{dt^2}x01(t)\right) + m2\left(\frac{d^2}{dt^2}x12(t)\right) - F2x(t) \\ m2\left(\frac{d^2}{dt^2}x01(t)\right) + m2\left(\frac{d^2}{dt^2}x12(t)\right) - F2x(t) + k12x12(t)^3 + d12\sqrt{\frac{d}{dt}x12(t)} \end{bmatrix}$$

(b)

FIGURE 11 Nonlinear constitutive behavior. With (a) new elemental constitutive equations and (b) the resulting system EOMs. The graph-theoretic approach used by DFP uses element models, potentially with highly nonlinear constitutive laws, interconnected by a linear graph. Thus, initially unmodeled nonlinearities can be included by simply changing the corresponding elemental constitutive equation.

Introductory examples are organized around simple systems involving a single moving rigid body with one joint such as the simple pendulum, spring-mass-damper, two-dimensional particle motion, and the spinning top. Systems with more than one moving rigid body are introduced using a planar two-link, serial-chain manipulator example.

The planar four-bar, planar slider crank, and planar parallel manipulator examples extend the discussion to systems with one or more closed kinematic loops. The closed kinematic loops introduce algebraic constraints that interact with the underlying ODEs to create systems of DAEs. The suitable formulation and solution of such DAE systems has created many challenges in the past [19]–[21] but can be transparently handled by DFP as shown in the examples.

The presence of compliant structural members in a multibody system adds difficulty to symbolic EOMs generation. The planar spinup flexible beam and planar slider-crank with flexible-connecting-rod examples highlight the simplified automated processing possible using DFP. Additionally, these examples are structured to allow a novice to progress from rigid-body equivalents to their nonrigid counterparts.

The extension to spatial multibody systems is also illustrated using several examples. The spatial slider crank example is developed systematically as an extension of the planar slider-crank example by altering the

$$ODE := \begin{bmatrix} m1\left(\frac{d^2}{dt^2}x01(t)\right) - k12x12(t) - d12\left(\frac{d}{dt}x12(t)\right) \\ m2\left(\frac{d^2}{dt^2}x02(t)\right) - F2x(t) + k12x12(t) + d12\left(\frac{d}{dt}x12(t)\right) \end{bmatrix}$$

FIGURE 12 EOMs developed in terms of absolute coordinates $x_{01}(t)$ and $x_{02}(t)$. Selecting appropriate body and joint elements to include in the spanning tree allows the user to obtain the final EOMs in terms of absolute coordinates.

$$ODE := \begin{bmatrix} m1\left(\frac{d^2}{dt^2}x01(t)\right) - \lambda 1(t) \\ m2\left(\frac{d^2}{dt^2}x02(t)\right) + \lambda 1(t) - F2x(t) \end{bmatrix}$$

FIGURE 13 Augmented Lagrangian formulation. DFP provides the user with direct control over selecting the formulation and included variables. Thus, for constrained systems, the EOMs can be developed in extended generalized coordinates with added Lagrange multipliers. Solving such augmented systems permits the calculation of the internal constraint forces as well as the external system response.

The governing equations generated by DFP can also be used for inverse-dynamics calculations.

joints. Finally, the development of symbolic, closed-form EOMs for a 6DOF Stewart-Gough platform serves as the capstone example.

LIMITATIONS AND SCOPE-OF-USE

No review would be complete without a discussion of the potential limitations and scope-of-use of such software. This discussion is especially pertinent in light of the numerous commercially available MCAE software packages.

$$ODE := \left[(m1 + m2)\left(\frac{d^2}{dt^2}\,x01(t)\right) - F2x(t)\right]$$

FIGURE 14 Minimal generalized-coordinate formulation for constrained systems. The EOMs can be automatically regenerated by altering the joint properties and rerunning the same Maple worksheet. This capability can be used to generate the EOMs in terms of a minimal set of generalized coordinates.

First, and foremost, there is an implicit assumption that a user has the capability to perform the initial modeling. Suitable selection of frames of reference, generalized coordinates, and other aspects can have a tremendous influence on the resulting form and size of the EOMs. Thus, while the Java GUI facilitates the specification of interconnections in block-diagram form, the user still needs to specify the sequence of interconnections. Second, although the current library of components is reasonably comprehensive, the modeling of certain large multibody systems could pose some challenges to a novice. Third, DFP still does not possess the conveniences that commercial-off-the-shelf packages offer in terms of model creation and simulation. Features such as 3D visualization capabilities, automated mass and inertial calculations (to name a few) are currently missing. Finally, while the automated processing to create the EOMs has been adequately shielded from the user, the user is expected to have a good grasp of Maple

TABLE 1 Planar joint properties for a relative-coordinate-based minimal unconstrained ODE formulation. The 1D translational system can be allowed to operate in the plane by converting joints 1 and 3 into planar joints. A minimal-coordinate formulation of the EOMs can be retained by preferentially including the relative coordinates, those of Joints 1 and 2, within the spanning tree.

Property	Joint 1	Joint 2	Joint 3
T-Tree	Checked	Checked	Unchecked
R-Tree	Checked	Checked	Unchecked
Connection type	Planar joint	Prismatic joint	Planar joint
First axis of translation	x axis <1,0,0>	–	x axis <1,0,0>
Second axis of translation	y axis <0,1,0>	–	y axis <0,1,0>
s1	x01	x12	x02
s2	y01	y12	y02
theta	theta1	–	theta2

TABLE 2 Planar joint properties for an absolute-coordinate-based augmented Lagrangian formulation. A planar two-mass-spring-damper results from modifying joints 1 and 3 to be planar joints. An augmented Lagrangian formulation of the EOMs, in terms of the absolute coordinates of the two masses and the Lagrange multipliers, results when joints 1 and 3 are included within the spanning tree.

Property	Joint 1	Joint 2	Joint 3
T-Tree	Checked	Unchecked	Checked
R-Tree	Checked	Unchecked	Checked
Connection type	Planar joint	Prismatic joint	Planar joint
First axis of translation	x axis <1,0,0>	–	x axis <1,0,0>
Second axis of translation	y axis <0,1,0>	–	y axis <0,1,0>
s1	x01	x12	x02
s2	y01	y12	y02
theta	theta1	–	theta2

programming concepts and data-storage constructs to effectively use the results.

Despite these limitations, DynaFlexPro represents the leading edge of rapid symbolic system model generation for high-fidelity performance analysis and control. As such, it is ideally suited for a user who is reasonably well versed with traditional analytical modeling, has encountered the limitations of traditional MCAE packages, and is

$$(m1 + m2)\left[\frac{d^2}{dt^2}x01(t)\right] + m2\cos(\theta 1(t))\left[\frac{d^2}{dt^2}x12(t)\right] - m2\sin(\theta 1(t))\,(L$$
$$+\, x12(t))\left[\frac{d^2}{dt^2}\theta 1(t)\right] - F2x(t) - m2\left[\frac{d}{dt}\theta 1(t)\right]^2\cos(\theta 1(t))\,L - m2\left[\frac{d}{dt}\theta 1(t)\right]^2\cos(\theta 1(t))\,x12(t)$$
$$-\,2\,m2\left[\frac{d}{dt}\theta 1(t)\right]\sin(\theta 1(t))\left[\frac{d}{dt}x12(t)\right]],\ [$$
$$(m1 + m2)\left[\frac{d^2}{dt^2}y01(t)\right] + m2\sin(\theta 1(t))\left[\frac{d^2}{dt^2}x12(t)\right] + m2\cos(\theta 1(t))\,(L$$
$$+\, x12(t))\left[\frac{d^2}{dt^2}\theta 1(t)\right] - F2y(t) - m2\left[\frac{d}{dt}\theta 1(t)\right]^2\sin(\theta 1(t))\,L - m2\left[\frac{d}{dt}\theta 1(t)\right]^2\sin(\theta 1(t))\,x12(t)$$
$$+\,2\,m2\left[\frac{d}{dt}\theta 1(t)\right]\cos(\theta 1(t))\left[\frac{d}{dt}x12(t)\right]],\ [$$
$$m2\cos(\theta 1(t))\left[\frac{d^2}{dt^2}x01(t)\right] + m2\sin(\theta 1(t))\left[\frac{d^2}{dt^2}y01(t)\right]$$
$$+\, m2\left[\frac{d^2}{dt^2}x12(t)\right] - \cos(\theta 1(t))\,F2x(t) - \sin(\theta 1(t))\,F2y(t) - m2\left[\frac{d}{dt}\theta 1(t)\right]^2 L - m2\left[\frac{d}{dt}\theta 1(t)\right]^2 x12(t) + k12x12(t)$$
$$+\, d12\left[\frac{d}{dt}x12(t)\right]],\ [$$
$$-m2\sin(\theta 1(t))\,(L + x12(t))\left[\frac{d^2}{dt^2}x\theta 1(t)\right] + m2\cos(\theta 1(t))\,(L + 12(t))\left[\frac{d^2}{dt^2}y01(t)\right] + (L^2\,m2 + 2\,L\,m2\,x12(t) + x12(t)^2\,m2 + I1zz$$
$$+\, I2zz)\left[\frac{d^2}{dt^2}\theta 1(t)\right] - \cos(\theta 1(t)\,L\,F\,2y(t) + \sin(\theta 1(t))\,L\,F2x + 2\,L\,m2\left[\frac{d}{dt}\theta 1(t)\right]\left[\frac{d}{dt}x12(t)\right] - \cos(\theta 1(t))x12(t)F2y(t)$$
$$+\,\sin(\theta 1(t))x12(t)\,F2x(t) + 2x12(t)\,m2\left[\frac{d}{dt}\theta 1(t)\right]\left[\frac{d}{dt}x12(t)\right]]$$

(a)

$$ODE := [m1\left[\frac{d^2}{dt^2}x01(t)\right] + \sin(\theta 1(t))\,\lambda 1(t) - \cos(\theta 1(t))\left[k12x12(t) + d12\left[\frac{d}{dt}x12(t)\right]\right]],\ [$$
$$m1\left[\frac{d^2}{dt^2}y01(t)\right] - \cos(\theta 1(t))\,\lambda 1(t) - \sin(\theta 1(t))\left[k12x12(t) + d12\left[\frac{d}{dt}x12(t)\right]\right]],\ [$$
$$m2\left[\frac{d^2}{dt^2}x02(t)\right] - \sin(\theta 1(t))\,\lambda 1(t) - F2x(t) + \cos(\theta 1(t))\,k12x12(t) + \cos(\theta 1(t))\,d12\left[\frac{d}{dt}x12(t)\right]],\ [$$
$$m2\left[\frac{d^2}{dt^2}y02(t)\right] + \cos(\theta 1(t))\,\lambda 1(t) - F2y(t) + \sin(\theta 1(t))\,k12x12(t) + \sin(\theta 1(t))\,d12\left[\frac{d}{dt}x12(t)\right]],\ [$$
$$I1zz\left[\frac{d^2}{dt^2}\theta 1(t)\right] + (-(-y01(t) + y02(t)\sin(\theta 1(t)) + (x01(t) - x02(t))\cos(\theta 1(t))\,\lambda 1(t) - \lambda 2(t)$$
$$-\left[k12x12(t) + d12\left[\frac{d}{dt}x12(t)\right]\right](-\sin(\theta 1(t))\,x01(t) + \sin(\theta 1(t))\,x02(t) + \cos(\theta 1(t))y01(t) - \cos(\theta 1(t))y02(t))],\ [$$
$$I2zz\left[\frac{d^2}{dt^2}\theta 2(t)\right] + \lambda 2(t)]$$

(b)

FIGURE 15 EOMs for a planar two-mass-spring-damper in terms of (a) relative-coordinates and (b) absolute coordinates. A minimal set of four unconstrained second-order differential equations arises from the use of relative coordinates, whereas an augmented system of six second-order ODEs coupled by two Lagrange multipliers results from the use of absolute coordinates. DFP thus provides the flexibility to create either form with minimal added effort.

looking to exploit symbolic modeling in developing real-time engineering models of large multibody systems.

CONCLUSIONS

Although algorithmic and computational advances have helped create numerous MCAE tools for analyzing ever-more-complex multibody systems, these tools have limitations. First, the underlying general-purpose formulations can introduce numerical approximations and thus errors in subsequent numerically based solutions. Second, models created with most MCAE tools are unsuitable for real-time implementation. Further, models might be difficult to parameterize using dynamic system characteristics, such as masses or inertias as opposed to more traditional geometric characteristics such as lengths that are prevalent in MCAE systems. Finally, sensitivity analysis to parametric changes can at best be performed numerically where the computational overhead would restrict it to local analyses.

DFP is a new MCAE tool that exploits Maple's symbolic manipulation power for automatically creating the governing EOMs for large mechanical multibody systems in symbolic form. The parametric nature of the resulting symbolic model makes it well suited for the parametric system-performance studies, for use in design-refinement of such systems, or the development of suitable controllers. Furthermore, the code generation capability can be used to export the generated model to other platforms such as Matlab, C, or Fortran. Thus, engineers seeking to design, analyze, or control a mechanical multibody system can use DFP to efficiently and rapidly derive high-fidelity system-models that are suitable for use in real-time engineering simulators or hardware-in-the-loop testbeds.

AUTHOR INFORMATION

Rajankumar M. Bhatt received the B.Eng. degree in mechanical engineering from the M.S. University of Baroda, India in 2000 and the M.S. degree in mechanical engineering from State University of New York at Buffalo, in 2003, where he is currently working toward his Ph.D. His research interests include cooperative mobile robots, graph theory, and the dynamics of multibody systems.

Venkat N. Krovi (vkrovi@eng.buffalo.edu) received his M.S. and Ph.D. degrees in mechanical engineering from the University of Pennsylvania in 1995 and 1998, respectively. In 2001, he joined the Department of Mechanical and Aerospace Engineering at the State University of New York at Buffalo. His research interests include design, analysis and prototyping of articulated mechanical systems. He is the recipient of a 2003 National Science Foundation CAREER Award.

REFERENCES

[1] "DynaFlexPro portal," May 2006 [Online]. Available: http://www.maplesoft.com/dynaflexpro

[2] U. Ascher and L. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Philadelphia, PA: SIAM, 1998.

[3] J. García-de-Jalón and E. Bayo, *Kinematic and Dynamic Simulation of Multibody Systems: The Real-Time Challenge*. New York: Springer-Verlag: New York: 1994.

[4] E. Haug, *Computer Aided Kinematics and Dynamics of Mechanical Systems*. Boston, MA: Allyn and Bacon, 1989.

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

[6] A.A. Shabana, *Computational Dynamics*. New York: Wiley, 2001.

[7] A. Erdman and J.P. Sadler, *Modern Kinematics: Developments in the Last Forty Years*. New York: Wiley, 1992.

[8] R.B. Gillespie, "A survey of multibody dynamics for virtual environments," in *Proc. Int. Mechanical Engineering Conf. Exhibition*, Dallas, TX, 1997.

[9] W. Schiehlen, "Multibody system dynamics: Roots and perspectives," *Multibody Syst. Dynam.*, vol. 1, no. 2, pp. 149–188, 1997.

[10] "Multibody system dynamics: Research activities," May 2006 [Online]. Available: http://real.uwaterloo.ca/~mbody/

[11] R. Lot and M. Da Lio, "A symbolic approach for automatic generation of the equations of motion of multibody systems," *Multibody Syst. Dynam.*, vol. 12, no. 142–172, 2004.

[12] "MBsymba," May 2006 [Online]. Available: http://www.dim.unipd.it/lot/mbsymba.html.

[13] A. Durrbaum, W. Klier, and H. Hahn, "Comparison of automatic and symbolic differentiation in mathematical modeling and computer simulation of rigid body systems," *Multibody Syst. Dynam.*, vol. 7, no. 331–355, 2002.

[14] P. Shi and J. McPhee, "Dynamics of flexible multibody systems using virtual work and linear graph theory," *Multibody Syst. Dynam.*, vol. 4, no. 331–355, 2000.

[15] C. Schmitke and J. McPhee, "Effective use of Subsystem Models in Multibody and Mechatronic System Dynamics," *Int. J. Multiscale Computational Eng.*, vol. 1, no. 2, pp. 139–159, 2003.

[16] C. Schmitke, *Modelling Multibody, Multi-domain Systems using Subsystems and Linear Graph Theory.* Waterloo, ON, Canada: University of Waterloo, 2004.

[17] J. McPhee, C. Schmitke, and S. Redmond, "Dynamic Modelling of Mechatronic Multibody Systems with Symbolic Computing and Linear Graph Theory," *Math. Computer Modelling Dynam. Syst.*, vol. 10, no. 1, pp. 1–23, 2004.

[18] "MATLAB/Simulink," May 2006 [Online]. Available: http://www.mathworks.com/access/helpdesk/help/helpdesk.html

[19] W.A. Khan, V. Krovi, S.K. Saha, and J. Angeles, "Modular and recursive kinematics and dynamics for parallel manipulators," *Multibody Syst. Dynam.*, vol. 14, no. 4, pp. 419–455, 2005.

[20] W.A. Khan, V. Krovi, S.K. Saha, and J. Angeles, "Recursive kinematics and inverse dynamics for a planar 3R parallel manipulator," *ASME J. Dynam. Syst. Meas. Contr.*, vol. 127, no. 4, pp. 529–536, 2005.

[21] T. Geike and J. McPhee, "Inverse dynamics analysis of parallel manipulators with full mobility," *Mechanism Mach. Theory*, vol. 38, no. 6, pp. 549–562, 2003.