Mathematical and Computer Modelling of Dynamical Systems
2004, Vol. 10, No. 1, pp. 1–23

Dynamic Modelling of Mechatronic Multibody Systems With Symbolic Computing and Linear Graph Theory

JOHN McPHEE[1], CHAD SCHMITKE[2] AND SCOTT REDMOND[3]

ABSTRACT

The application of linear graph theory to the modelling of flexible multibody systems is described. When combined with symbolic computing methods, linear graph theory leads to efficient dynamic models that facilitate real-time simulation of systems of rigid bodies and flexible beams. The natural extension of linear graphs to the modelling of mechatronic multibody systems is presented, along with a recently-developed theory for building complex system models from models of individual subsystems.

Keywords: Mechatronic multibody systems, dynamic modelling, symbolic computing, graph theory, subsystems.

1. INTRODUCTION

Linear graph theory, invented by Leonhard Euler in 1736, was applied to the modelling and analysis of one-dimensional physical systems in the 1950s and 1960s [1]. The basic theory of this modelling approach was established during this period, and subsystem analysis methods were developed to facilitate the modelling of very large electrical networks [2]. Linear graph theory was extended to the modelling of multi-dimensional particle mass systems in the 1970s [3], and to spatial multibody systems of rigid bodies in the 1980s and 1990s [4, 5].

Quite recently, graph-theoretic methods for modelling flexible multibody systems were developed [6]; an overview is presented in the following sections. Graph theory

[1]Address correspondence to: John McPhee, Systems Design Engineering, University of Waterloo, Ont., Canada N2L 3G1. E-mail: mcphee@real.uwaterloo.ca
[2]Systems Design Engineering, University of Waterloo, Ont., Canada N2L 3G1.
[3]MD Robotics, 9445 Airport Road, Brampton, Ont., Canada L6S 4J3.

can be used to generate equations in terms of user-selected coordinates, including absolute or joint or indirect coordinates, or some combination of the three. By selecting coordinates that are well-suited to a given problem, one can reduce the number and complexity of the governing equations.

Given that linear graphs were originally applied to electrical networks, the extension of graph-theoretic methods to the modelling of mechatronic multibody systems is natural and straightforward. From a single linear graph of the entire system, the coupled equations for the electrical and mechanical domain are systematically derived. Symbolic programming was used to implement this graph-theoretic formulation; the resulting Maple program (DynaFlex) for modelling flexible multibody mechatronic systems is briefly described.

The modelling of complex systems is greatly facilitated by the use of subsystem models; a newly-developed graph-theoretic generalization of the Norton and Thevenin theorems from electrical network theory is used to generate models of multibody mechatronic subsystems. This subsystem modelling approach is demonstrated on a PD-controlled parallel robot.

2. FLEXIBLE MULTIBODY SYSTEMS: LINEAR GRAPH REPRESENTATION AND COORDINATE SELECTION

Consider the planar slider-crank mechanism shown in Figure 1, and its linear graph representation in Figure 2. Consistent with the terminology of Kesavan et al. [1] and Roe [2], a linear graph G is a collection of e edges that intersect only at v nodes (or vertices). A circuit is defined as a subgraph of G such that there are exactly two distinct paths between every pair of nodes of the subgraph. Essentially, a circuit is a closed loop of edges. A tree is defined as any subgraph that contains all the nodes of G and has no circuits; edges in the tree are called branches, while the remaining edges (in the cotree) are chords. Finally, a outset is a subset of edges that, when

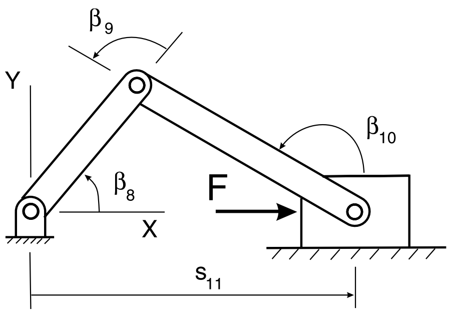

Fig. 1. Slider-crank mechanism.

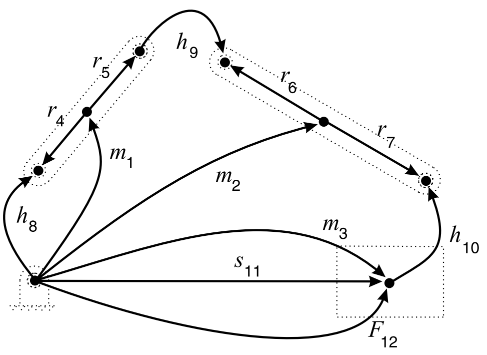

Fig. 2. Linear graph of slider-crank mechanism.

removed, divides the graph into two parts, and no subset of this cutset has this property.

Nodes in the multibody system graph in Figure 2 represent body-fixed reference frames, while edges represent kinematic or dynamic transformations associated with physical components. Note that the graph is directed; the arrowhead on each edge establishes a positive direction for measuring transformations between the frames at the tip and tail nodes. Edges $m_1 - m_3$ represent the three bodies, which may be rigid or flexible. Newton's Laws require that the edges originate at an inertial frame (node) and terminate at a body-fixed frame (node) – the center of mass for a rigid body, and at one end of a flexible beam element. For clarity, the physical components are superimposed on the linear graph with dotted lines.

Edges $h_8 - h_{10}$ are the three revolute joints connecting the bodies, while s_{11} is the prismatic joint between the ground and the slider. The "arm elements" $r_4 - r_7$ represent transformations from the body frame to the body-fixed joint frames. These transformations are functions of the body rotation and, for flexible bodies, the deformation coordinates. Finally, edge F_{12} is the external force acting on the slider. Additional physical effects, for example, motor torques, springs, and weights, are easily added to the system graph.

Associated with each edge are through (force, torque) and across (translation, rotation) variables that satisfy the fundamental cutset and circuit equations [1]. For mechanical systems, the cutset equations provide dynamic equilibrium equations for any combination of components, while the circuit equations provide closure conditions around any loop. These cutset and circuit equations are systematically generated from an incidence matrix representation of the linear graph. Thus, the kinematic and dynamic equations can be automatically generated for a system with any topology, as demonstrated in the subsequent sections.

If a tree is selected for the graph, then the circuit equations can be used to express all kinematic variables as linear combinations of the coordinates associated with the tree edges ("branches") [5]. Thus, *by selecting a spanning tree, the branch coordinates* $\mathbf{q}$ *that appear in the kinematic and dynamic equations are defined.* This is an important feature of a graph-theoretic approach, since most other multibody formulations are restricted to a pre-defined set of coordinates, usually absolute [7] or joint [8].

As an example, consider the slider-crank mechanism from Figure 1. Rigid arm elements (i.e. $r_4 - r_7$) are always selected into the tree of a multibody system graph because they represent constant kinematic transformations; since there are no unknown coordinates associated with an arm element, they don't introduce any variables into $\mathbf{q}$. By also selecting h_8, h_{10}, and s_{11} into the tree, one will obtain equations in the corresponding branch coordinates $\mathbf{q} = [\beta_8, \beta_{10}, s_{11}]^T$. All cotree variables can be expressed in terms of $\mathbf{q}$ using the circuit equations, for example, for the rotation of cotree rigid body m_2:

$$\underline{\theta}_2 = \underline{\theta}_{11} + \underline{\theta}_{10} - \underline{\theta}_7 \tag{1}$$

$$\underline{\omega}_2 = \underline{\omega}_{11} + \underline{\omega}_{10} - \underline{\omega}_7 \tag{2}$$

where $\underline{\theta}_i$ and $\underline{\omega}_i$ are the orientation and angular velocity for element i, i.e., the orientation and angular velocity of the tip node relative to the tail node. These across variables are defined by the constitutive equations for the corresponding element type, for example, $\underline{\theta}_i = \beta_i \hat{k}$ for revolute joints 8, 9, and 10, and $\underline{\omega}_{11} = 0$ for the prismatic joint. In this manner, all kinematic variables are systematically expressed in terms of $\mathbf{q}$.

Using graph theory, one can generate equations in absolute coordinates (select all bodies into the tree), joint coordinates (select joints into the tree), or some combination of the two. One can even select two different trees – one to generate rotational equations and one to generate translational equations. By doing this, one can reduce the total number of equations to be solved, and reduce their complexity at the same time [5]. By selecting coordinates that are tailored to the problem at hand, one can obtain equations that are well-suited to real-time simulation, in either a virtual reality environment or an operator-in-the-loop simulation.

3. KINEMATICS OF MULTIBODY SYSTEMS

For dependent branch coordinates, which is always the case for systems with closed kinematic chains, the kinematic constraint equations are generated by *projecting the circuit equations for cotree joints onto the reaction space for that joint.* For holonomic constraints, one obtains m nonlinear algebraic equations in terms of the n branch coordinates $\mathbf{q}$:

$$\mathbf{\Phi}(\mathbf{q}, t) = \mathbf{0} \tag{3}$$

where $n - m = f$, the degrees of freedom (DOF) of the system.

To demonstrate, consider again the slider-crank with the dependent branch coordinates $\mathbf{q} = [\beta_8, \beta_{10}, s_{11}]^T$, i.e., with h_8, h_{10}, and s_{11} selected into the tree. The reaction space for a joint is defined by the unit vectors that span the space of reaction forces and moments arising in the joint. Thus, the reaction space for cotree joint h_9 is spanned by unit vectors $\hat{\imath}$ and $\hat{\jmath}$ (the directions of the joint reaction forces), onto which the circuit equation for h_9 is projected:

$$\mathbf{\Phi} = \underline{r}_9 \cdot \hat{\imath}, \hat{\jmath} \tag{4}$$

$$= (\underline{r}_6 - \underline{r}_7 + \underline{r}_{10} + \underline{r}_{11} - \underline{r}_8 + \underline{r}_4 - \underline{r}_5) \cdot \hat{\imath}, \hat{\jmath} \tag{5}$$

where $\underline{r}_i$ is the translational displacement vector for element i. Substituting the elemental constitutive equations, for example, $\underline{r}_{10} = 0$, and evaluating,

$$\mathbf{\Phi} = \begin{Bmatrix} L_{67}\cos\beta_{10} + s_{11} - L_{45}\cos\beta_8 \\ L_{67}\sin\beta_{10} - L_{45}\sin\beta_8 \end{Bmatrix} = \mathbf{0} \tag{6}$$

where $L_{45} = L_4 + L_5$, etc., and $L_i = |\underline{r}_i|$. Thus, one obtains $m = 2$ constraint equations in terms of the $n = 3$ branch coordinates for this 1-DOF system.

To get a smaller set of coordinates and equations for the same example, one can select h_8, h_9, and h_{10} into the tree for translational equations, and h_8, h_{10}, and s_{11} into a second tree for generating rotational equations. This results in the $n = 2$ branch coordinates $\mathbf{q} = [\beta_8, \beta_{10}]^T$. The only non-null constraint equation is obtained from the circuit equation for cotree joint s_{11}, projected onto the unit vector $\hat{\jmath}$ normal to its axis of sliding (direction of normal reaction force) to obtain the single kinematic constraint equation:

$$\mathbf{\Phi} = L_{45}\sin\beta_8 - L_{67}\sin\beta_{10} = \mathbf{0} \tag{7}$$

This procedure for generating kinematic equations is valid for any system topology, and for any selected set of branch coordinates. The corresponding set of velocity or acceleration constraints are obtained by projecting the circuit equations at the velocity or acceleration level, respectively, onto the same joint reaction space. Alternatively, one can use symbolic computing to time-differentiate the position-level Constraint Equations (3).

4. DYNAMICS OF MULTIBODY SYSTEMS

Once the kinematic equations are obtained, the dynamic equations can be systematically generated by *projecting the cutset equations for branch components onto the motion space for that component* [5]:

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{\Phi}_{\mathbf{q}}^T \boldsymbol{\lambda} = \mathbf{F} \tag{8}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{\Phi_q}$ is the Jacobian matrix of the constraint equations, the Lagrange multipliers $\boldsymbol{\lambda}$ correspond to reactions in cotree joints, and $\mathbf{F}$ contains external forces and quadratic velocity terms.

Alternatively, the principle of virtual work may be employed, especially when flexible bodies are included in the system model [6], to obtain the Dynamic Equations (8). To facilitate this approach, the first variation of the constraint equations is partitioned into f independent coordinates $\mathbf{q_i}$ and m dependent coordinates $\mathbf{q_d}$:

$$\delta\mathbf{\Phi} = \mathbf{\Phi_{q_d}}\delta\mathbf{q_d} + \mathbf{\Phi_{q_i}}\delta\mathbf{q_i} = \mathbf{0} \tag{9}$$

where $\mathbf{\Phi_{q_i}} = \partial\mathbf{\Phi}/\partial\mathbf{q_i}$ and $\mathbf{\Phi_{q_d}} = \partial\mathbf{\Phi}/\partial\mathbf{q_d}$ is non-singular as long as the given physical constraints are not redundant. One can then solve these Linear Equations (9) for the transformation from dependent to independent variations:

$$\delta\mathbf{q_d} = -\mathbf{\Phi_{q_d}^{-1}}\mathbf{\Phi_{q_i}}\delta\mathbf{q_i} = \mathbf{J}\delta\mathbf{q_i} \tag{10}$$

and for the transformation from all variations to independent variations:

$$\delta\mathbf{q} = \begin{Bmatrix} \delta\mathbf{q_d} \\ \delta\mathbf{q_i} \end{Bmatrix} = \begin{bmatrix} \mathbf{J} \\ \mathbf{1} \end{bmatrix}\delta\mathbf{q_i} = \mathbf{G}\delta\mathbf{q_i} \tag{11}$$

where $\mathbf{G}$ is easily shown to be an orthogonal complement to the Jacobian matrix $\mathbf{\Phi_q}$, that is, the matrix product $\mathbf{G\Phi_q} = \mathbf{0}$.

By summing up the contributions of all working components, the system virtual work δW is obtained. If there are n_T moments $\underline{T}_i$, n_F forces $\underline{F}_j$ (e.g., springs, dampers, weights, applied forces), n_R rigid bodies, and n_L flexible bodies,

$$\begin{aligned}\delta W = {} & \sum_{i=1}^{n_T}\underline{T}_i\cdot\delta\underline{\theta}_i + \sum_{j=1}^{n_F}\underline{F}_j\cdot\delta\underline{r}_j \\ & - \sum_{k=1}^{n_R}[m_k\underline{a}_k\cdot\delta\underline{r}_k + (\underline{\underline{I}}_k\cdot\underline{\dot{\omega}}_k + \underline{\omega}_k\times\underline{\underline{I}}_k\cdot\underline{\omega}_k)\cdot\delta\underline{\theta}_k] + \sum_{l=1}^{n_L}\delta W_{f_l}\end{aligned} \tag{12}$$

where m_k is the mass of body k, $\underline{\underline{I}}_k$ is its inertia dyadic about its center of mass, $\underline{a}_k$ is the acceleration of its mass center, and $\underline{\omega}_k$ is its angular velocity. The terms $\delta\underline{r}$ and $\delta\underline{\theta}$ are translational and rotational virtual displacements. They represent a small displacement taking place at a given time, which is compatible with the system constraints. The virtual work of a flexible body δW_f is defined by:

$$\delta W_f = -\int_V \delta\epsilon^T\boldsymbol{\sigma}dV + \int_V \delta\underline{r}\cdot(\underline{f}_b - \gamma\underline{\ddot{r}})dV \tag{13}$$

where $\delta\epsilon$ is the column matrix of varied strain components, $\boldsymbol{\sigma}$ is the matrix of corresponding stress components, $\underline{f}_b$ is the body force, including gravity, and $-\gamma\underline{\ddot{r}}$ is the inertial force per unit volume. The first volume integral in Equation (13) represents the virtual strain energy, while the second integral is the virtual work of body forces, including gravity, external forces and inertial forces. By making assumptions about the material properties and deformation field, one can express δW_f in terms of the branch coordinates $\mathbf{q}$, which include the deformation coordinates for the flexible body. For a Rayleigh beam with linear elastic material properties, the details of this derivation are presented in [9].

Substituting the element constitutive equations into Equation (12), and using the Transformation Equation (11), one obtains:

$$\delta W = \mathbf{Q}^T \delta\mathbf{q} = \mathbf{Q}^T \mathbf{G} \delta\mathbf{q}_\mathbf{i} = \mathbf{Q}_\mathbf{i}^T \delta\mathbf{q}_\mathbf{i} = 0 \tag{14}$$

where $\mathbf{Q}$ and $\mathbf{Q}_\mathbf{i} = \mathbf{G}^T\mathbf{Q}$ are the generalized forces associated with $\delta\mathbf{q}$ and $\delta\mathbf{q}_\mathbf{i}$, respectively. Since $\delta\mathbf{q}$ are not independent quantities, $\mathbf{Q}$ can only be set to zero if it is augmented by the Lagrange multipliers $\boldsymbol{\lambda}$. The result is the set of Dynamic Equations (8), which are said to be written in "augmented" form.

Alternatively, the dynamic equations can be written in an "embedded" form, in which the constraint reactions $\boldsymbol{\lambda}$ do not appear. Since $\delta\mathbf{q}_\mathbf{i}$ are independent, each generalized force $\mathbf{Q}_\mathbf{i}$ can be set directly to zero to obtain one dynamic equation per degree of freedom:

$$\widetilde{\mathbf{M}}\ddot{\mathbf{q}} = \widetilde{\mathbf{F}} \tag{15}$$

where $\widetilde{\mathbf{M}} = \mathbf{G}^T\mathbf{M}$ is an unsymmetric $f \times n$ mass matrix and $\widetilde{\mathbf{F}} = \mathbf{G}^T\mathbf{F}$. Note that the embedded Dynamic Equations (15) can be directly obtained by pre-multiplying the augmented Dynamic Equations (8) by $\mathbf{G}^T$ to eliminate the Lagrange multipliers. In Section 7, these embedded equations will be used to advantage in an inverse dynamic analysis.

In a forward dynamic analysis, one has to solve a set of differential-algebraic equations (DAEs) to determine the time response $\mathbf{q}(t)$. Equations (3) and (8) comprise a set of $n + m$ index-3 DAEs that can be solved simultaneously for $\mathbf{q}(t)$ and $\boldsymbol{\lambda}(t)$. An attractive alternative is to solve the smaller set of n index-2 DAEs (3) and (15) for only the branch coordinates $\mathbf{q}(t)$. There are quite a number of different numerical methods available for solving the DAEs that govern the forward dynamics of multibody systems [10].

5. INDIRECT COORDINATES

It has already been shown that one can generate models in absolute or joint coordinates, by selecting a suitable tree for the linear graph representation. By defining and selecting "virtual joints" into the tree [11], it is also possible to generate equations in "indirect coordinates" corresponding to bodies that are not necessarily adjacent [12]. Consider the open-loop multibody system taken from [12], and its corresponding linear graph in Figure 3.

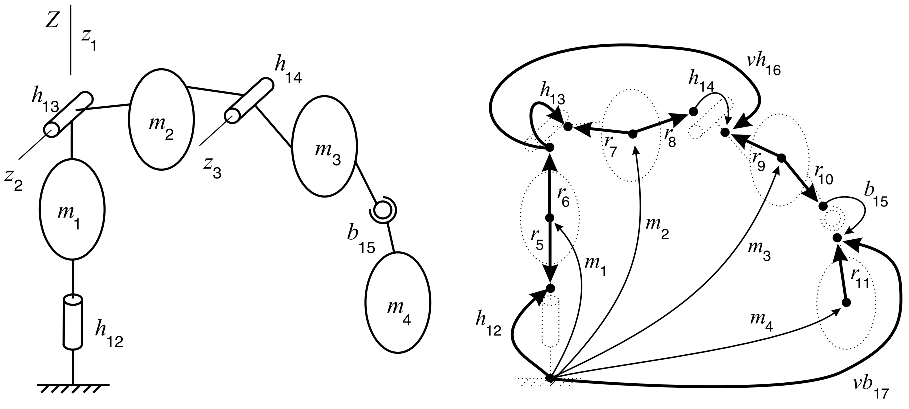

Fig. 3. Open-loop multibody system and linear graph.

Once again, bodies are represented by edges $m_1 - m_4$, body-fixed transformations by arm elements $r_5 - r_{11}$, revolute joints by edges $h_{12} - h_{14}$, and the spherical joint by b_{15}. In addition, two virtual joints are added: a virtual revolute joint vh_{16} representing the rotation of m_3 relative to m_1, and a virtual spherical joint vb_{17} to represent the rotation of m_4 relative to the ground. Fayet and Pfister [12] have shown that the corresponding indirect coordinates result in equations that are simpler in form than the more traditional joint coordinate equations. This is due to the fact that joint axes z_2 and z_3 are parallel, and absolute angular coordinates give simpler equations than relative angular coordinates [13]. With a graph-theoretic approach, one can generate the motion equations in indirect coordinates by simply selecting vh_{16} and vb_{17} into the tree in place of h_{14} and b_{15}.

To demonstrate, consider a serial manipulator comprised of the first three bodies $m_1 - m_3$ in Figure 3. Using this graph-theoretic approach, the Dynamic Equations (8) were generated in terms of the joint coordinates $\mathbf{q}_{\boldsymbol{\beta}} = [\beta_{12}, \beta_{13}, \beta_{14}]^T$ and the indirect coordinates $\mathbf{q}_{\mathbf{i}} = [\beta_{12}, \beta_{13}, \beta_{16}]^T$. The joint angle β_{12} measures rotation about a vertical Z axis, while joint angles β_{13} and β_{14} are about horizontal axes z_2 and z_3 fixed in m_2 and m_3, respectively. The indirect coordinate β_{16}, corresponding to virtual joint vh_{16}, measures the orientation of m_3 relative to the non-adjacent m_1.

Both sets of coordinates are independent for this serial manipulator, so there are no kinematic constraint equations and no constraint reactions $\boldsymbol{\lambda}$ appearing in the dynamic equations. However, some of the entries in these equations will differ for the two sets of coordinates, for example, the second diagonal entry in the two mass matrices:

$$\mathbf{M}_{\boldsymbol{\beta}}(2,2) = I_{2z} + m_2 L_7^2 + I_{3z} + m_3[L_{78}^2 + 2L_{78}L_9 \cos\beta_{14} + L_9^2]$$
$$\mathbf{M}_{\mathbf{i}}(2,2) = I_{2z} + m_2 L_7^2 + m_3 L_{78}^2$$

Table 1. Number of operations for dynamic equations of first three bodies ($m_1 - m_3$).

	Adds.	Mults.	Functs.
$\mathbf{M}_\beta$	27	53	12
$\mathbf{M}_\mathbf{i}$	20	43	11
$\mathbf{F}_\beta$	62	149	59
$\mathbf{F}_\mathbf{i}$	60	138	57

where $\mathbf{M}_\beta$ is in joint coordinates while $\mathbf{M}_\mathbf{i}$ is written in indirect coordinates. The entries in row 2, column 3 are:

$$\mathbf{M}_\beta(2,3) = I_{3z} + m_3 l_9^2 + m_3 l_{78} l_9 \cos\beta_{14}$$
$$\mathbf{M}_\mathbf{i}(2,3) = m_3 l_{78} l_9 \cos(\beta_{16} - \beta_{13})$$

The use of indirect coordinates results in a simplification of the mass matrix, for which the diagonal entries correspond to moments of inertia of "compound augmented bodies" [12]. In this case, $\mathbf{M}_\mathbf{i}(2,2)$ is the moment of inertia I_{2z} of m_2, plus the mass of m_3 concentrated at joint 14, about the z_2 axis. In contrast, the more complex $\mathbf{M}_\beta(2,2)$ is the combined moment of inertia of m_2 and m_3 about z_2.

The forces **F** are also simplified, albeit slightly, by the use of indirect coordinates. Table 1 provides the number of additions, multiplications, and function calls needed to evaluate **M** and **F** for both sets of coordinates. These are easily obtained using built-in routines from the Maple symbolic programming language.

6. MECHATRONIC MULTIBODY SYSTEMS

Graph-theoretic modelling is not restricted to mechanical systems; it has long been applied to electrical, pneumatic, and hydraulic systems [1, 2], and our graph-theoretic symbolic algorithms were easily extended to the modelling of mechatronic multibody systems [14, 15]. These systems are comprised of electrical networks, multibody systems, and transducer elements that convert mechanical energy into electrical energy and vice-versa. All components are represented within a single linear graph.

It has already been shown how the equations for the mechanical subsystem are generated. For the electrical domain, edges in the linear graph again correspond to physical components, while nodes represent points of interconnection. Through variables are currents while across variables are voltages. The circuit equations satisfy Kirchoff's voltage law around every fundamental circuit – one for each cotree element. The cutset equations are a generalization of Kirchoff's current law, and give a unique equation for each tree element. Once again, the tree selection determines the variables appearing in the final system equations.

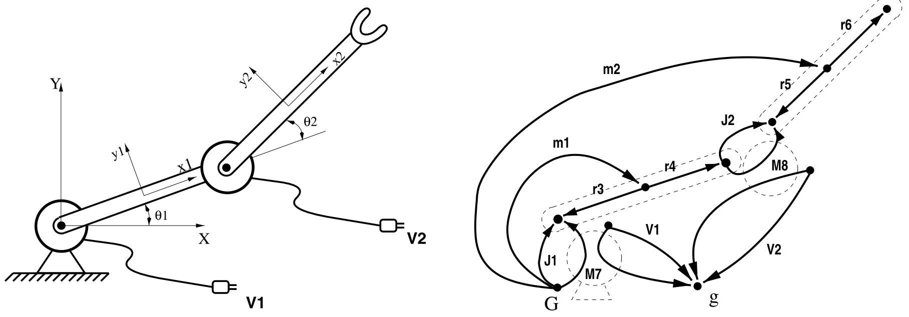

Fig. 4. DC motor-driven robot manipulator and linear graph.

To demonstrate, consider the two-link manipulator shown in Figure 4 with its linear graph representation. The system graph contains the standard mechanical components: m_1 and m_2 for the two rigid bodies, $r_3 - r_6$ for the body-fixed arm elements, and $J_1 - J_2$ for the two revolute joints. The corresponding joint angles θ_1 and θ_2 are controlled by two DC motors, M_7 and M_8, that are powered by voltage sources V_1 and V_2. Note that the graph consists of two parts, one for the mechanical domain and one for the electrical domain, and that the two domains are coupled by the motors, which have edges in each domain. This is always the case for a transducer element that converts energy between different domains.

By selecting the joints into the mechanical tree, and the voltage sources into the electrical tree, four differential equations are generated for the two-link manipulator – two for each of the two domains – in terms of θ_1, θ_2, and the two motor currents.

$$\begin{bmatrix} L_7 & 0 \\ 0 & L_8 \end{bmatrix} \begin{Bmatrix} \frac{di_7}{dt} \\ \frac{di_8}{dt} \end{Bmatrix} = \begin{Bmatrix} V_1(t) - R_7 i_7 - K_{v_7}\dot{\theta}_1 \\ V_2(t) - R_8 i_8 - K_{v_8}\dot{\theta}_2 \end{Bmatrix} \tag{16}$$

where i_j, R_j, L_j, and K_{v_j} are the current, armature resistance, inductance, and voltage constant for motor j. Note that a minimal number of system equations is automatically generated. Although the electrical sub-network in this example is trivial, networks of higher complexity can be efficiently modelled using linear graph theory.

For the mechanical domain, two second-order differential equations are generated:

$$[M(\theta_1, \theta_2)] \begin{Bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{Bmatrix} = \begin{Bmatrix} F_1(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2, i_7) \\ F_2(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2, i_8) \end{Bmatrix} \tag{17}$$

where the mass matrix **M** is a complex function of the joint coordinates, and the forces **F** are functions of the joint coordinates, joint velocities, and motor currents. Had the links been modelled as flexible beams, additional equations would be generated for the deformation coordinates, which would also appear in **M** and **F**.

Together, Equations (16)–(17) constitute the dynamic equations of the electro-mechanical system. These system equations can be used to find inverse solutions for the motor currents required to drive a particular trajectory, or as the basis of a forward dynamic simulation that tests out different controller strategies [14].

7. SYMBOLIC COMPUTER IMPLEMENTATION: DYNAFLEX

Symbolic programming provided an effective computer implementation of our graph-theoretic virtual work formulation [16], especially when real-time simulations of complex mechanical systems are required. Multiplications by 0 and 1 are eliminated in a symbolic program, and trigonometric simplifications are automated. The kinematic and dynamic equations can be visually examined for physical insight, or exported as optimized code (in which repeated expressions are identified and evaluated only once) for subsequent simulation.

We have used the Maple symbolic programming language to develop our DynaFlex software[4] for modelling flexible multibody systems. A variety of modelling components is available, including: three-dimensional rigid bodies or elastic beams; revolute, prismatic, spherical, planar, universal, and other joints; kinematic drivers; springs, dampers, actuators, and other force components. Symbolic kinematic and dynamic equations can be generated in absolute or joint coordinates or a combination of both. Dynamic equations can be generated in either the augmented form (8) or the embedded form (15). The DynaFlex program requires an input file that describes the system's topology and parameters; this file can be easily created using the graphical user interface, DynaGUI.

7.1. Mechanical Example

To demonstrate the modelling process using DynaGUI and DynaFlex, consider the spatial slider-crank mechanism [7] shown in Figure 5. A crank is driven at a constant rotational speed by the torque T. The crank is connected to the ground by a revolute joint at A, and to a connecting rod by a spherical joint at B. A sliding block is attached to the ground by a prismatic joint at D, and to the connecting rod by a universal joint at C. An inverse dynamic analysis is required to determine the driving torque T.

A screenshot of the DynaGUI model is shown in Figure 6. The three bodies are represented by rectangles; body-fixed frames are shown as attached circles, or as a square for the main body frame. The four joints are shown as lines connecting the three bodies and the ground. Each joint has a particular type and set of properties, and may be in the tree or cotree. In this example, the spherical joint is placed in the cotree; the branch coordinates corresponding to the remaining joints in the tree are $\mathbf{q} = [s, \theta, \eta, \beta]^T$, which are shown in Figure 5.

[4]http://real.uwaterloo.ca/~dynaflex

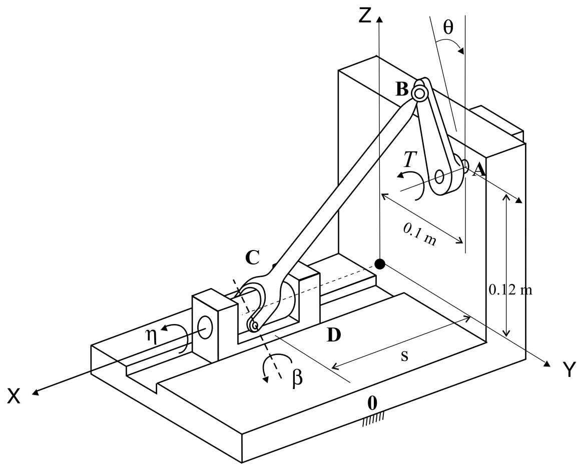

Fig. 5. Spatial slider-crank mechanism.

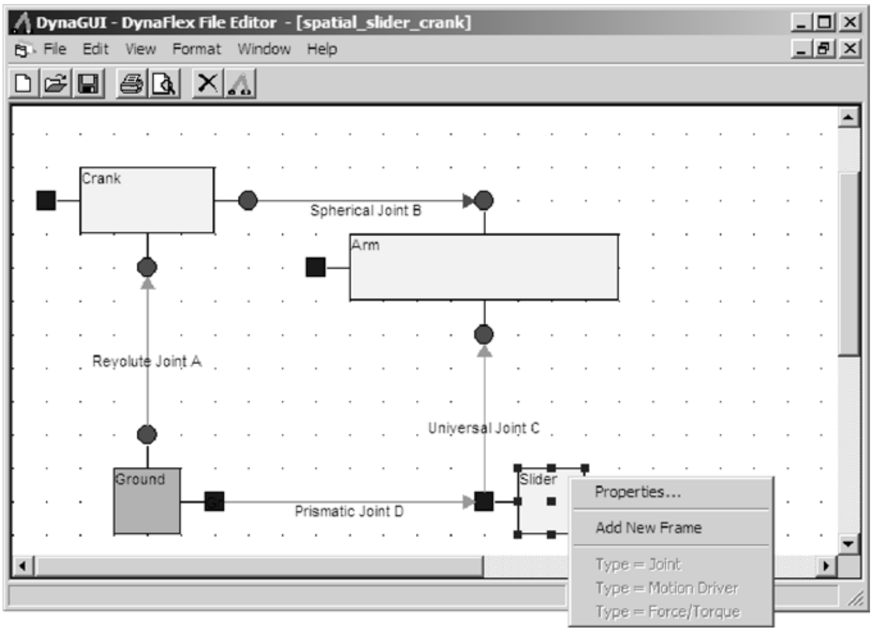

Fig. 6. DynaGUI model of spatial slider-crank.

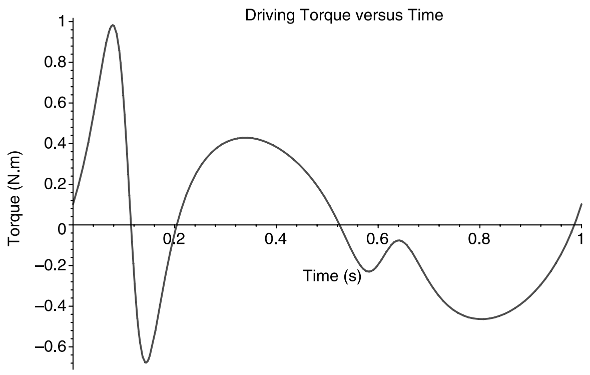

Fig. 7. Driving torque on crank of spatial slider-crank.

The motion for the crank is prescribed as $\theta = 2\pi t$, and the corresponding torque T is automatically included in the virtual work computation. Combining the prescribed motion with the projected circuit equations for the cotree spherical joint, and substituting the link lengths given by [7], DynaFlex generates the kinematic equations:

$$\mathbf{\Phi} = \left\{ \begin{array}{c} 2\sin\theta + 6\sin\eta\sin\beta + 25/10 \\ 2\cos\theta + 6\cos\eta\sin\beta + 3 \\ 25s - 6\cos\beta \\ \theta - 2\pi t \end{array} \right\} = \mathbf{0} \tag{18}$$

which are easily solved for $\mathbf{q}(t)$. By requesting the dynamic equations in the embedded form of Equation (15), DynaFlex produces a single dynamic equation for this 1-DOF system. The driving torque appears linearly in this equation, so a symbolic expression for T is easily obtained with Maple and plotted in Figure 7; the results are identical to those shown in [7], which were obtained by solving a large system of linear equations using numerical methods. Unlike the latter approach, our symbolic solution for T can be viewed, exported as optimized C or Fortran or Matlab code, and used to facilitate the real-time control of a physical prototype.

7.2. Electro-Mechanical Example

Shown in Figure 8 is a simple model of an electro-mechanical system, taken from [17]. The electrical circuit contains a resistor R_1, a moving-plate capacitor C_2 with plate separation s, an inductor L_3, and a voltage source E_4. The moving plate of the capacitor has a mass m_5, and a suspension stiffness and damping of k_6 and d_6, respectively. The model also includes the weight of the moving plate.

Shown in Figure 9 is the linear graph for the simple electro-mechanical system. Edges in the graph correspond to physical components on a one-to-one basis. An

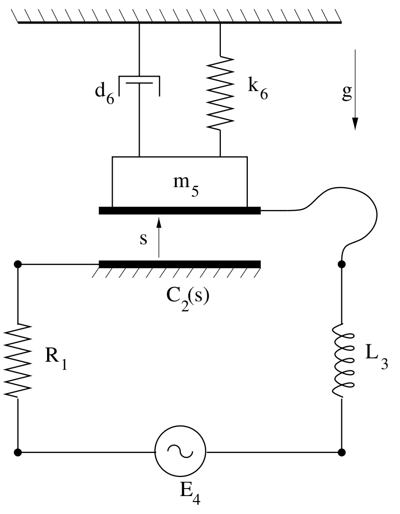

Fig. 8. A simple electro-mechanical system.

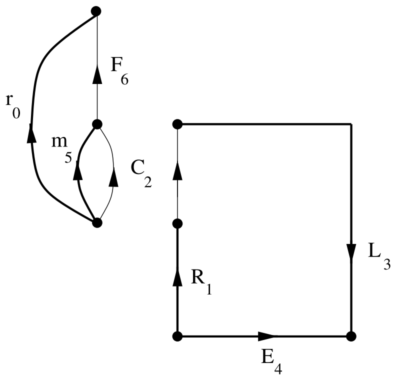

Fig. 9. Linear graph of simple electro-mechanical system.

additional component, r_0, specifies the location of the suspension support. Assuming that the spring is undeformed when $s = 0$, the constant distance r_0 equals the undeformed spring length. The tree selected for the graph is shown in bold; for this tree, the system equations will be formulated in terms of the plate separation s and capacitor current i_2.

Once again, the transducer element C_2 has an edge in both the mechanical and electrical domains. Due to the electrical attraction of the plates, a force arises that depends upon the voltage v_2. Conversely, the current through the capacitor depends upon the speed of separation $\dot{s}$. These phenomena are encapsulated in the two Constitutive Equations [18]:

$$F_2 = \frac{1}{2} C_2' v_2^2 \tag{19}$$

$$i_2 = C_2(s)\frac{dv_2}{dt} + C_2' \dot{s} v_2 \tag{20}$$

where the notation $C_2(s)$ emphasizes that the capacitance varies with plate separation, and $C_2' \equiv dC_2(s)/ds$.

Figure 10 shows the model that was assembled using the DynaGUI building blocks. From this DynaGUI model, a DynaFlex input file is automatically generated. This input file contains all the topological information embedded in the linear graph (so that it is not necessary to know how the graph is constructed), as well as the parameters for the various components. From this input file, DynaFlex generates two ordinary differential equations in terms of the chosen variables s and i_2:

$$C_2'\dot{s}\left(-R_1 i_2 - L_3\frac{di_2}{dt} + E_4(t)\right) + C_2(s)\left(-R_1\frac{di_2}{dt} - L_3\frac{d^2 i_2}{dt^2} + \frac{dE_4}{dt}\right) - i_2 = 0$$

$$m_5\ddot{s} + d_6\dot{s} + k_6 s + m_5 g - \frac{1}{2}C_2'\left(-R_1 i_2 - L_3\frac{di_2}{dt} + E_4(t)\right)^2 = 0$$

which are equivalent to those manually derived in [17].

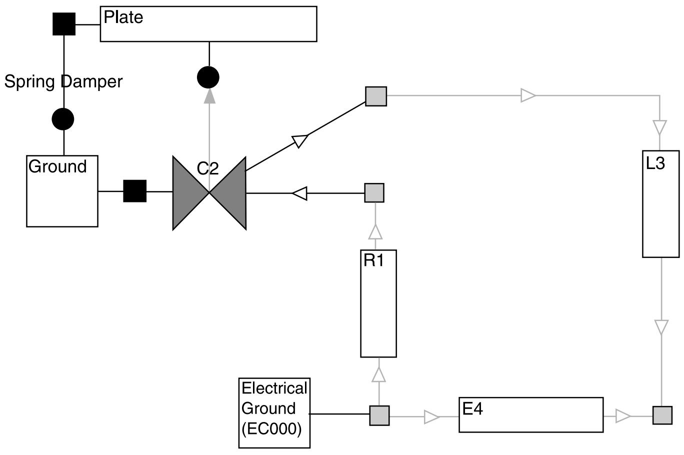

Fig. 10. DynaGUI model of simple electro-mechanical system.

8. EFFICIENT MODELLING VIA SUBSYSTEMS

The modelling of multibody systems is greatly facilitated by the use of subsystem models. Models of entire sub-assemblies can be created and stored (symbolically) for future use. Generating libraries in this fashion results in faster formulation times as only the interactions between library components need to be derived during formulation. Additionally, the use of pre-derived sub-assemblies simplifies the modelling of complex systems. In a time-varying topological situation, for example a virtual reality application in which the user is adding or removing parts to their model, the system equations can be reformulated quickly by focusing attention only on the modified subsystems.

Graph theory is naturally suited to the modelling of systems via subsystem models. Although the subsystem modelling of electrical circuits was achieved decades ago using graph theory [2], it has only recently been extended to the nonlinear kinematics and dynamics of multibody systems [19].

Subsystem models are particularly well-suited for the topologies that are typical of parallel manipulators: a number of kinematic chains (or "legs") in parallel from the ground to the end effector [20, 21]. Each leg can be modelled as a subsystem that is kinematically decoupled from the other leg subsystems. This allows the equations for kinematics and inverse dynamics to be generated and solved in parallel, either symbolically or numerically on a parallel computing platform.

In this section, we model one leg of a PD-controlled parallel manipulator as an abstracted subsystem using linear graph theory, which is then used to combine three of these subsystem models into a symbolic model of the parallel robot.

8.1. Subsystem Modelling of Parallel Manipulator Leg

Consider one leg of a planar parallel manipulator, shown in Figure 11. A revolute joint connects two planar links to one another, while an additional revolute joint connects

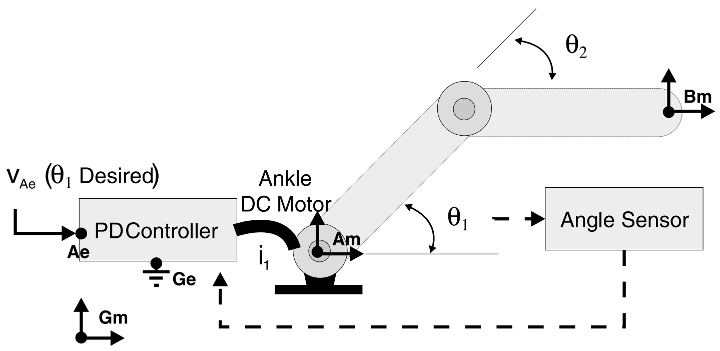

Fig. 11. PD controlled leg of parallel manipulator.

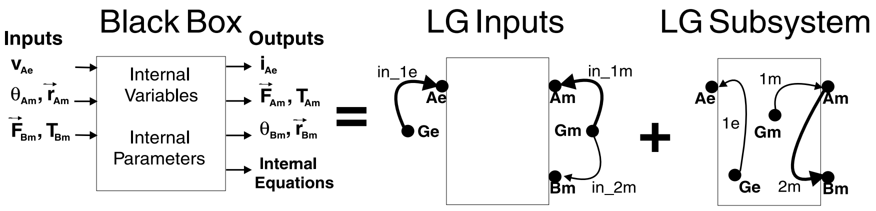

Fig. 12. Black box and linear graph representations of leg subsystem.

one of the links to the manipulator's base (which is free to translate and rotate). A DC motor actuates this joint, designated the "ankle", based on a voltage supplied by the proportional-derivative (PD) controller. The inputs to the controller consist of a user-defined voltage (related to the desired angle of the joint) and a voltage signal specifying the actual joint angle (supplied by an angle sensor). The output voltage of the PD-controller is a linear combination of the error and derivative of the error in the angle it is controlling.

The leg interacts with its mechanical environment at the moving base and tip (body-fixed reference frames *Am* and *Bm*). Electrical interaction occurs only at node *Ae*, where the voltage signal for the desired ankle angle is input. These interactions are depicted on the left-hand side of Figure 12, the black box model of the leg.

Although not shown on the figure, the derivatives for all of the input and output variables are also assumed. Note that when a voltage or position is expressed, it is assumed to be with respect to the electrical or mechanical ground, respectively.

The inputs (known variables from the environment) are shown on the left-hand side of the black box in Figure 12. For the leg subsystem, the inputs consist of the motion of the base frame ($\vec{r}_{Am}$, θ_{Am}), the forces acting at the leg's tip ($\vec{F}_{Bm}$, T_{Bm}), and the voltage to the controller (v_{Ae}). It should be noted that inputs are usually expressions: functions of time and other system variables external to the subsystem.

The subsystem's outputs appear on the right of the black box depiction, and can be expressed as two different sets of equations. The first set relates the input expressions to the internal variables used to model the dynamics of the subsystem. We call this set the subsystem's internal equations. Modelling the leg with the variables shown in Figure 11 (joint coordinates and motor currents) results in three internal variables: θ_1, θ_2, and i_1. As a result, three internal equations are generated.

The second set of output equations relate the inputs' complementary variables ($\vec{F}_{Am}$, T_{Am}, $\vec{r}_{Bm}$, θ_{Bm}, i_{Ae}) to the input expressions and internal variables. These equations are referred to as the subsystem's constitutive equations.

Although linear graph theory provides a systematic means of generating subsystem models [19] (including all of the advantages inherent with tree selection), hand-derived models, or models derived using a symbolic software package, are also easy to represent as linear graph subsystem models. The only requirements are the symbolic equations governing the subsystem and a clear understanding of the system's input, internal and output variables. In these cases, even if linear graph theory is not used to generate the subsystem model, it can still be effectively used to integrate the subsystem model into a larger system.

The right hand side of Figure 12 shows how the linear graph subsystem model is obtained from the black box representation of the subsystem. Notice that two graphs are required to represent the black box model. The first graph is a generic representation of the inputs that will be applied to the subsystem by the external system. For example, edge in_1m might be replaced by a full model for another robot that is controlling the motion of the base. The second graph represents the actual subsystem graph – the nodes and edges that make up the subsystem.

In the black-box representation of the subsystem, the direction of the arrow relative to the box's boundary indicates whether a variable represents an input or an output. In the linear graph representation of the subsystem, this is accomplished through the tree selection and the concept of primary variables. The details of this conversion can be found in [22].

8.2. Modelling of Parallel Manipulator Using Subsystems

A three degree of freedom (DOF) planar, parallel manipulator is shown in Figure 13. As can be seen, the manipulator consists of a single platform connected by revolute joints ($h10$, $h11$, $h12$) to three identical legs. The system's linear graph overlays a drawing of the manipulator in order to show the similarities between the physical system and linear graph. Once again, interconnections between components are shown as nodes, while edges correspond to components or subsystems. The system's tree is shown in bold. Nodes are denoted by a capital letter, while their corresponding domain appears as a lower case letter. Cartesian axes are attached to the mechanical nodes, to emphasize their interpretation as body-fixed reference frames. The linear graph representation of the leg subsystems is consistent with that shown in Figure 12. The only difference is the addition of the subsystem's prefix ($S1\_$, $S2\_$ and $S3\_$) to the edge identifiers.

The system's governing equations arise from equations internal to the system components and the interconnections between the system's components. The first group, the internal subsystem equations, can be directly obtained from the subsystem definitions. In the case of the parallel manipulator, each leg subsystem contains three internal dynamic equations (recall Section 8.1). Since none of the other components are subsystems, the rest of the governing equations arise from the interconnections between the system's components.

The "system-level" equations are obtained using one of the common graph-theoretic approaches to generating dynamic equations (Section 4). For the parallel manipulator, the inclusion of the platform mass in the tree results in three mechanical

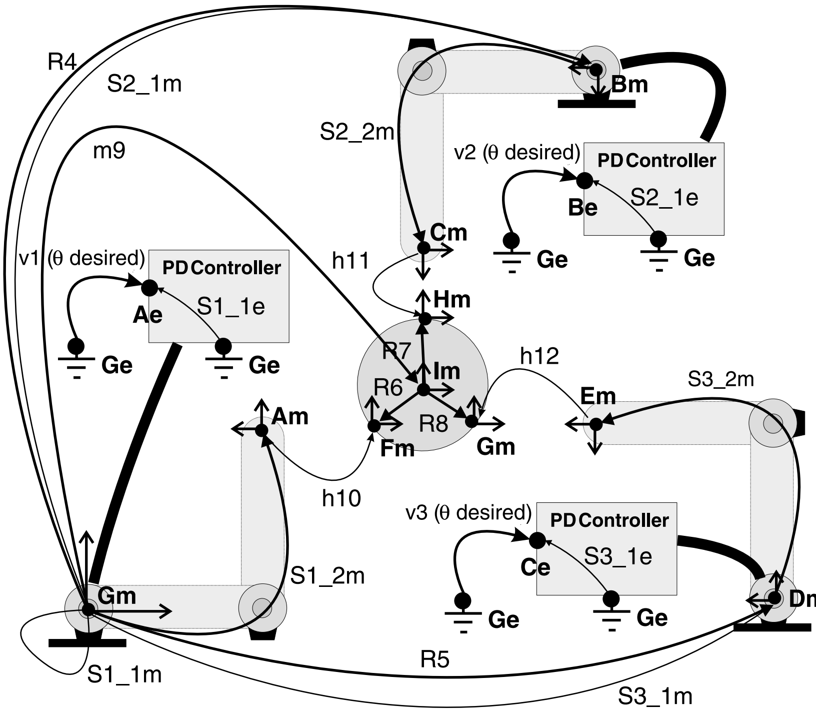

Fig. 13. 3-DOF parallel manipulator.

dynamic equations. These equations define the dynamic equilibrium of the platform, and are obtained from the cutsets for the mass edge *m*9.

$$\begin{aligned} Fx_{m9} + Fx_{h10} + Fx_{h11} + Fx_{h12} &= 0 \\ Fy_{m9} + Fy_{h10} + Fy_{h11} + Fy_{h12} &= 0 \\ Tz_{m9} + Tz_{h10} + Tz_{h11} + Tz_{h12} &= 0 \end{aligned} \tag{21}$$

where Fi_j and Ti_j represent the *i*th component of the force or torque, respectively, associated with edge *j*.

Including both of the internal and "system-level" equations results in $(3 \times 1) = 3$ electrical dynamic equations and $(3 \times 2 + 3) = 9$ mechanical dynamic equations. These equations are in terms of 12 modelling variables: the 3 motor currents, the 6 leg joint angles and 3 platform position/orientation variables. However, the manipulator has only 6 degrees of freedom: 3 mechanical and 3 electrical. Thus, we require 6 constraint equations, which couple the mechanical dynamic equations originating from the subsystems.

These 6 equations also arise from the interconnections between the system's components. From Section 3, they are obtained from the projected loop closure conditions imposed by the cotree revolute joints ($h10$, $h11$, and $h12$) that attach each leg to the platform:

$$\mathbf{\Phi} = \begin{bmatrix} x_{h10} - x_{R6} - x_{m9} + x_{S1\_2m} \\ y_{h10} - y_{R6} - y_{m9} + y_{S1\_2m} \\ x_{h11} - x_{R7} - x_{m9} + x_{R4} + x_{S2\_2m} \\ y_{h11} - y_{R7} - y_{m9} + y_{R4} + y_{S2\_2m} \\ x_{h12} - x_{R8} - x_{m9} + x_{R5} + x_{S3\_2m} \\ y_{h12} - y_{R8} - y_{m9} + y_{R5} + y_{S3\_2m} \end{bmatrix} = \mathbf{0} \tag{22}$$

where x_i and y_i represent the x or y displacement, respectively, of the ith edge.

Once the topological and constitutive equations have been systematically combined, the resulting equations are completely in terms of the system-level and internal subsystem primary variables. The modelling variables related to the subsystem edges (those starting with $S1\_$, $S2\_$ and $S3\_$, in the graph) have been eliminated using standard graph-theoretic transformations. As a result, the final equations are the same as those derived without using subsystem models. However, the equations are now topologically grouped. To see this more clearly, consider the mechanical dynamic equations for the parallel robot, written in the format given by Equation (8):

$$\mathbf{M}\begin{bmatrix} {}_1\ddot{\theta}_1 \\ {}_1\ddot{\theta}_2 \\ {}_2\ddot{\theta}_1 \\ {}_2\ddot{\theta}_2 \\ {}_3\ddot{\theta}_1 \\ {}_3\ddot{\theta}_2 \\ {}_0\ddot{x}_{m9} \\ {}_0\ddot{y}_{m9} \\ {}_0\ddot{\theta}_{m9} \end{bmatrix} + \begin{bmatrix} X & X & 0 & 0 & 0 & 0 \\ X & X & 0 & 0 & 0 & 0 \\ 0 & 0 & X & X & 0 & 0 \\ 0 & 0 & X & X & 0 & 0 \\ 0 & 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & 0 & X & X \\ X & 0 & X & 0 & X & 0 \\ 0 & X & 0 & X & 0 & X \\ X & X & X & X & X & X \end{bmatrix} \begin{bmatrix} {}_0Fx_{h10} \\ {}_0Fy_{h10} \\ {}_0Fx_{h11} \\ {}_0Fy_{h11} \\ {}_0Fx_{h12} \\ {}_0Fy_{h12} \end{bmatrix} = \mathbf{F} \tag{23}$$

In this equation, the leading subscript indicates the subsystem to which a variable belongs, while its trailing subscript refers to the component's edge identifier. System-level variables have "0" as a leading subscript. The "X" markers in the transpose of the constraint Jacobian matrix indicate a non-zero entry. It should also be noted that the mass matrix **M** is block diagonal.

By organising the mechanical dynamic equations in this manner it is clear how the equations can be decoupled. The two dynamic equations for each leg can be simultaneously solved in order to obtain the revolute reaction forces in terms of that subsystem's joint accelerations. These values can then be substituted into the expressions for the last three mechanical dynamic equations. The use of subsystems with linear graph theory clearly reveals this substitution process, which has been used in hand derivations of the dynamics of parallel robots [23]. The three electrical dynamic equations remain unchanged.

To validate our subsystems modelling theory, it was programmed into Maple. Using this software, the above equations were automatically generated using the aforementioned subsystem models. For simulation purposes, the decoupling strategy mentioned above was implemented on the mechanical equations. The three electrical equations, in terms of the currents running through the three DC-motors, were left unchanged. Finally, in order to generate a set of pure ordinary differential equations (ODEs), the six kinematic constraint equations were differentiated twice. No constraint stabilization method was used.

A forward dynamic analysis was then performed by providing an input voltage to drive each of the legs' ankle motors and solving the system ODEs. Once the forward analysis was completed, the values of the currents were used to determine the torque supplied by each motor during the simulation. These torques were then compared with an inverse dynamics solution of a purely mechanical version of the system obtained by Ma and Angeles [24], and confirmed by Geike and McPhee [25]. As expected, the motor torque for the PD-controlled system fluctuates about the inverse dynamic solution (Fig. 14).

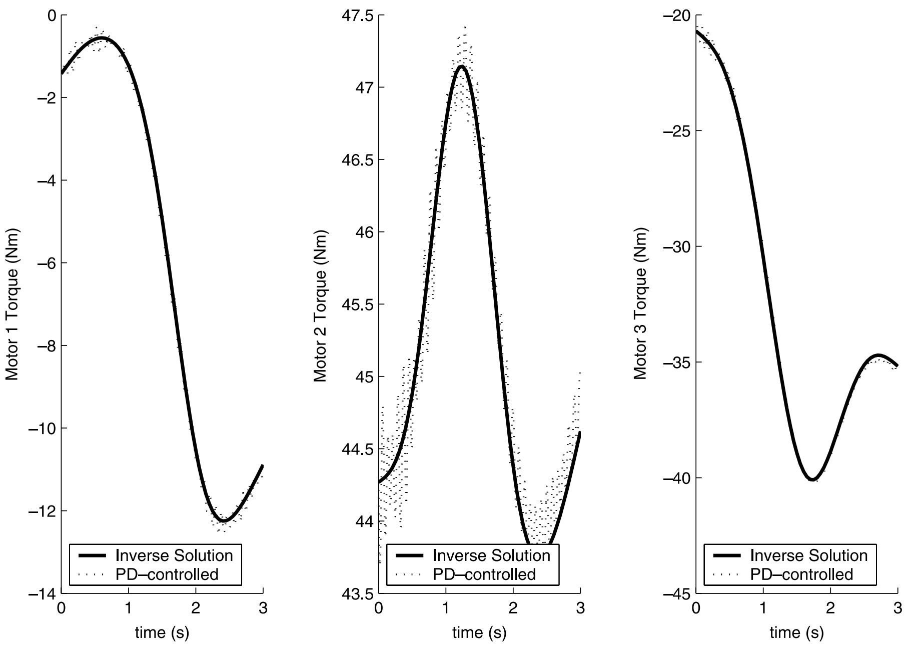

Fig. 14. Comparison of PD-controlled and inverse solution motor torques.

The parameters and initial conditions used for the mechanical system are identical to those used by Geike and McPhee. The electromechanical parameters for the DC-motors, angle sensors and PD-controllers are as follows: $K_t = K_v = K_s = L = R = 1, B = 0.01, K_p = 4000, K_d = 8000$. Note that the mass and inertia of the motors are ignored in order to maintain the same mechanical model used in [25].

The initial conditions for the currents running through DC-motors are ${}_1i_1 = -1.427A$, ${}_2i_1 = 44.268A$, ${}_3i_1 = -20.699A$. The input voltages to the leg subsystems were simply the angle drivers prescribed by Geike and McPhee multiplied by K_s. The large values for the currents are a result of the simple motor parameters and the lack of any gearing between the motors and the driven links.

9. CONCLUSIONS

When combined with symbolic programming, linear graph theory provides a very efficient approach to modelling flexible multibody mechatronic systems. By selecting a spanning tree for a linear graph of the system, one can define a set of coordinates that is well-suited to the problem at hand. From a single linear graph representation of a complex system, the coupled equations for the electrical and mechanical subsystems are systematically generated in symbolic form by our Maple implementation (DynaFlex) of the graph-theoretic formulation. Special topologies, for example for parallel manipulators, can be exploited, especially if a subsystem modelling approach is adopted. This newly-developed subsystems approach is a generalization of the Norton and Thevenin theorems for electrical networks, and greatly facilitates the modelling of mechatronic multibody systems.

ACKNOWLEDGEMENTS

Financial support for this research is provided by a Premier's Research Excellence Award and Waterloo Maple Inc., and by the Natural Sciences and Engineering Research Council of Canada.

REFERENCES

1. Koenig, H.E., Tokad, Y. and Kesavan, H.K.: *Analysis of Discrete Physical Systems*. McGraw-Hill, New York, 1967.
2. Roe, P.: *Networks and Systems*. Addison-Wesley, Reading, MA, 1966.
3. Andrews, G. and Kesavan, H.K.: The Vector-Network Model: A New Approach to Vector Dynamics. *Mech. Mach. Theory* 10 (1975), pp. 57–75.
4. Baciu, G. and Kesavan, H.K.: From Unidimensional to Multidimensional Physical Systems: Graph-Theoretic Modeling. *Syst. Cybernet.* 21 (1992), pp. 55–71.

5. McPhee, J.: Automatic Generation of Motion Equations for Planar Mechanical Systems Using the New Set of ‘Branch Coordinates’. *Mech. Mach. Theory* 33 (1998), pp. 805–823.
6. Shi, P. and McPhee, J.: Dynamics of Flexible Multibody Systems Using Virtual Work and Linear Graph Theory. *Multibody Syst. Dyn.* 4 (2000), pp. 355–381.
7. Haug, E.J.: *Computer-Aided Kinematics and Dynamics of Mechanical Systems*. Allyn and Bacon, Boston, 1989.
8. Wittenburg, J.: *Dynamics of Systems of Rigid Bodies*. B.G. Teubner, Stuttgart, 1977.
9. Shi, P., McPhee, J. and Heppler, G.: A Deformation Field for Euler-Bernoulli Beams With Applications to Flexible Multibody Dynamics. *Multibody Syst. Dyn.* 5 (2001), pp. 79–104.
10. Eich-Soellner, E. and Führer, C.: *Numerical Methods in Multibody Dynamics*. B.G. Teubner, Stuttgart, 1998.
11. Redmond, S. and McPhee, J.: Modelling Multibody Systems With Indirect Coordinates. In: *Proceedings of 11th World Congress in Mechanism and Machine Science*, Tianjin, China, 2004.
12. Fayet, M. and Pfister, F.: Analysis of Multibody Systems With Indirect Coordinates and Global Inertia Tensors. *Eur. J. Mech. A: Solids* 13 (1994), pp. 431–457.
13. Huston, R.L., Liu, Y.S. and Liu, C.: Use of Absolute Coordinates in Computational Multibody Dynamics. *Comput. Struct.* 52 (1994), pp. 17–25.
14. Scherrer, M. and McPhee, J.: Dynamic Modelling of Electromechanical Multibody Systems. *Multibody Syst. Dyn.* 9 (2003), pp. 87–115.
15. Sass, L., McPhee, J., Schmitke, C., Fisette, P. and Grenier, D.: A Comparison of Different Methods for Modelling Multibody Systems With Electrical Drives. *Multibody Syst. Dyn.* (2003), submitted.
16. Shi, P. and McPhee, J.: Symbolic Programming of a Graph-Theoretic Approach to Flexible Multibody Dynamics. *Mech. Struct. Mach.* 30 (2002), pp. 123–154.
17. Hadwich, V. and Pfeiffer, F.: The Principle of Virtual Work in Mechanical and Electromechanical Systems. *Arch. Appl. Mech.* 65 (1995), pp. 390–400.
18. Crandall, S., Karnopp, D., Kurtz, E. and Pridmore-Brown, D.: *Dynamics of Mechanical and Electromechanical Systems*. McGraw-Hill, 1968.
19. Schmitke, C. and McPhee, J.: Effective Use of Subsystem Models in Multibody and Mechatronic System Dynamics. *Int. J. Multiscale Comput. Eng.* 1 (2003), pp. 139–159.
20. Angeles, J.: *Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms*, 2nd edition. Springer, New York, 2003.
21. Merlet, J.-P.: *Parallel Robots*. Kluwer Academic, Dordrecht, 2000.
22. Schmitke, C. and McPhee, J.: Modelling Mechatronic Multibody Systems Using Symbolic Subsystem Models. *Multibody Syst. Dyn.* (2003), submitted.
23. Dasgupta, B. and Choudhury, P.: *Mech. Mach. Theory* 34 (1999), pp. 801–824.
24. Ma, O. and Angeles, J.: Direct Kinematics and Dynamics of a Planar 3-DOF Parallel Manipulator. *ASME Adv. Des. Autom.* 3 (1989), pp. 313–320.
25. Geike, T. and McPhee, J.: *Mech. Mach. Theory* 38 (2003), pp. 549–562.