SYMOFROS: A Flexible Dynamics Modeling Software

J.-C. Piedbœuf M. Doyon P. Langlois
R. L'Archevêque
Space Technologies, Canadian Space Agency,
6767 route de l'Aeroport, St-Hubert, Quebec, Canada
tel: (514) 926-4688, fax: (514) 926-4695 email: Jean-Claude.Piedboeuf@space.gc.ca

ABSTRACT

This paper describes the modeling of mechanisms in tree topology with closed kinematic loops and non-holonomic constraints. The dynamics equations are built using Jourdain's principle. The kinematics and the dynamics are developed recursively to optimize the model. The method is implemented in the modeling software SYMOFROS using the symbolic language Maple. The recursive procedures of Maple are used to obtain an efficient model generation. The model generated is totally symbolic. From that model, a C model compatible with Matlab/Simulink is generated. A graphical user interface has been developed to simplify the data input by the user. The objects are chosen from a library and the mechanism is build by linking the different objects together. Many system parameters can be fixed interactively.

1 INTRODUCTION

Canada is currently developing the Mobile Servicing System (MSS) that will be used to build and maintain the International Space Station (ISS). The MSS consists of a mobile base on which is mounted a large manipulator with seven actuators called the space station remote manipulator system (SSRMS). At the tip of the SSRMS, two smaller arms are attached on a rotating joint. This second assembly is called the special purpose dextrous manipulator (SPDM). Each arm has seven actuators. The SSRMS is 17 meters long and has flexible joints and links. The SPDM is 3.4 meter long and has flexible joints. A simplified model of the complete system includes 22 rigid degrees of freedom (dof) and more than 30 flexible ones.

In a typical maintenance task, one of the SPDM arm will grasp a stabilization point creating a closed kinematic loop. The other will be used to remove and replace a part on the station. Therefore, the contact dynamics of the system must be understood.

The MSS system is quite complex but this complexity is typical of many existing mechanisms. In order to improve a design, to develop control, or to simulate a system, dynamic models are required. These models can be obtained through a variety of methods: Newton-Euler, Lagrange, d'Alembert, Kane. These methods can be applied using purely numerical approaches or using symbolic computation. In the second case, the model is generated symbolically and can be used for simulation or control.

In the past twenty years, the Canadian Space Agency (CSA) has developed several modeling and simulation tools for off-line and real-time simulation of space manipulators. These modeling programs are based on a recursive Newton-Euler approach implemented numerically [1]. The real-time version is currently used for astronaut training for the future missions related to the ISS.

In the last few years, CSA has explored symbolic computation to model flexible manipulators. Symbolical programs such as Maple or Mathematica permit manipulation of symbols. Therefore, the dynamic model can be generated prior to the simulation and symbolic approaches should be more efficient for simulation. By contrast, in a purely numerical approach, the dynamic model must be re-created at each integration step[1].

In this regard, we have developed a general purpose program based on Maple: SYMOFROS [2]. The current version is able to model manipulators in tree topology with flexible links and joints and with closed kinematic loops. The model is developed using a recursive Jourdain approach and the foreshortening of the flexible link is included [3]. SYMOFROS has been used extensively to simulate and control experimental robots with flexible links and joints. It has also been used to develop simulation models of more industrial robotic applications. It is available on a multitude of platforms and is suited to real-time applications.

In this paper, we will go over SYMOFROS, starting with the graphical user interface, the model generation, the C implementation and the modeling done in the Matlab-Simulink environment. SYMOFROS (Fig.1) is a modeling and simulation tool based on Maple for the symbolic model genera-

[1] For real-time applications, parts of the model are assumed constant for a few steps.

Proc. Fifth International Symposium on Artificial Intelligence, Robotics and Automation in Space, 1-3 June 1999 (ESA SP-440)

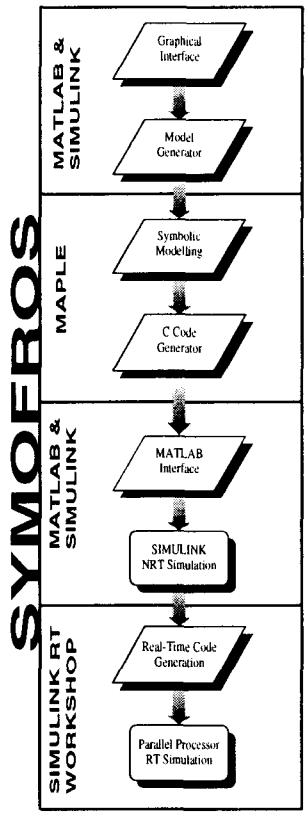

Figure 1: From a System to a Real-Time Simulation

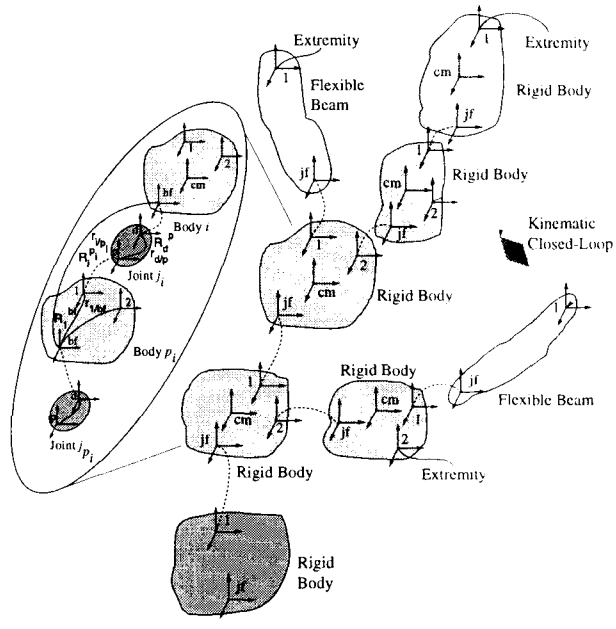

Figure 2: A Tree Structure with Closed-Loop

tion and on Matlab/Simulink for the graphical user interface (GUI), the simulation and the real-time implementation.

2 SYSTEM DESCRIPTION

One of the main difficulties in the development of a general purpose program for modeling is the representation of a system. This representation should be flexible enough to allow modeling of different systems easily. It should allow the addition or removal of an object without having to redefine all the structure. It should also allow the creation of a library of manipulators or parts of manipulators that can be re-used in creating new models. The development of a good GUI is closely linked to an adequate description of the system topology. The main difficulty is the processing of the different branches and the closed kinematic loops.

In SYMOFROS, we choose an object-oriented approach to describe a mechanism. The two main objects are: generalized body and closure. In Figure 2 a general system is described using these two objects. A generalized body is composed of a body (rigid or flexible) and a joint as illustrated.

2.1 Body: Rigid and Flexible

The geometric properties of a rigid body are defined by giving the relationships between the extremity frames and the body frame (Fig. 2). Any number of extremity frames can be defined. The required informations are the rotation matrices $\mathbf{R}_i^{bf}$ (from frame i to frame bf) and the position vectors $\mathbf{r}_{i/bf}$ (origin of frame i with respect to the origin of frame bf).

For a rigid body, the center of mass frame is added to define the inertia parameters. This frame is defined with respect to the body frame by a rotation matrix $\mathbf{R}_{cm}^{bf}$ and a position vector $\mathbf{r}_{cm/bf}$. The inertia parameters are the body mass and the inertia matrix. This matrix can be defined either in the body frame or in the center of mass frame.

For a rigid or a massless body, external forces and torques acting either on the body frame or on the center of mass frame (for a rigid body) can be specified. For the rigid body, it is also possible to specify a reduction ratio to represent the gyroscopic effect of the rotor of an electrical motor connected through a reducer.

A flexible body is defined as a flexible beam. Only one extremity frame can be defined because an ideal beam is slender. The only geometric information required is the beam length. The rotation matrix and the position vector between the body frame and the extremity frame are computed by the program. The beam foreshortening is taken into account so beam stiffening is included in the model.

A flexible beam can have deformations in bend-

ing in two perpendicular directions, and in torsion around the longitudinal axis. The beam deformations are represented using an extended assumed mode method [4]. The user needs only to supply the number of modes that are used for each direction. Zero modes in a given direction is equivalent to assume a rigid beam in that direction. The default shape functions for the assumed modes are spline functions but any other assumed modes can be chosen from a library of functions. The beam's internal damping is represented using a Voigt-Kelvin model.

External forces and torques can also be specified for a flexible beam. The external forces are applied on the centroid of the beam sections. They are integrated along the beam axis during the construction of the equations of motion.

The rigid bodies do not have any internal degrees of freedom (dof). Flexible beams have internal dofs to represent the beam's flexibility. The relative motion between bodies is represented through the joints. They contain all the rigid body motion dof.

2.2 Joint

A joint is characterized by the relationship between the proximal frame and the distal frame (Fig. 2). The rotation matrix $\mathbf{R}_d^p$ and the position vector $\mathbf{r}_{d/p}$ must be provided by the user. A joint can have from zero (constant rotation matrix and position vector) to six dof. This implies that all the different joint types can be represented.

The internal forces and torques between the bodies are specified in the joints. These forces can represent the motor torque, the elastic torque of an elastic joint or the damping force. The internal forces are represented by giving their work function. From that work, the program can compute the generalized forces associated with each generalized coordinates.

2.3 Closed Kinematic Loops

If a closed kinematic loop exists, the closure conditions must be specified. The closures are applied by connecting two extremity frames. One of the two frames is chosen as the reference frame to specify the closure conditions. The user indicates the directions in translation and in rotation along which the motion is not permitted. Closure equations are generated by SYMOFROS, along with the constraints' jacobian matrix and the non-linear terms of the constraints' second time derivative. This allows easy implementation of kinematic constraints through Lagrange multiplyers. Since this approach imposes the constraints at the acceleration level, Baumgarte stabilisation is possible for enhanced stability at the position and velocity level.

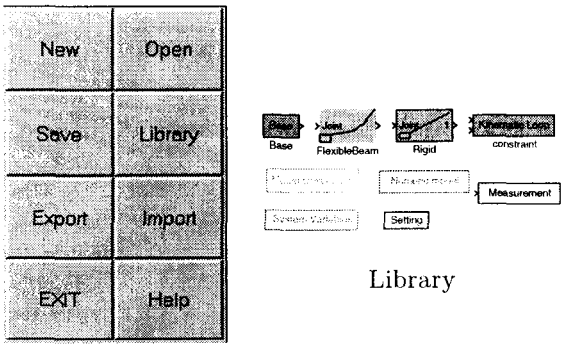

Figure 3: SYMOFROS Interface

2.4 System Parameters

In addition to the body, joint and closure descriptions, the system parameters, the generalized coordinates and the input variables must be specified. The generalized coordinates are specified only for rigid body motions, i.e., the joint variables. The coordinates associated to the beam flexibilities are defined by the program using the number of modes fixed by the user.

The user has also the possibility to give some general flags to determine how the model will be evaluated. For example, it is possible to linearize the model around a point or generate the equations required to compute the energy. It is also possible to specify non-holonomic constraints.

The gravity can be specified for the complete system by defining it on the base body. Only one base body can be specified for a system. The body frame of the base body is equivalent to the inertial reference frame.

3 THE GRAPHICAL USER INTERFACE

SYMOFROS is based both on Maple for the code generation and on Matlab-Simulink for the simulation. The code generation requires four text files that contain the complete information describing the robotic system. This information includes the model's topology, the bodies' symbolic description, the joints' symbolic description, the numerical values and the different paths. These files, being read by a program, must have a specific format and must be flawless. Although some expert SYMOFROS users are able to type in the information on their own, most beginning users are not and rely on the graphical user interface (GUI).

The GUI is based on Matlab-Simulink and uses the Simulink block diagram approach to describe the system's topology. Blocks from a library are dragged

and dropped, then linked together using the Simulink arrows. These blocks represent either a rigid body, a flexible beam, a kinematic loop, or model parameters. Each block can be double-clicked to display a window that contains it's relevant information. The information is mainly entered as variables which are later assigned a value. This enables Maple to generate the model in a symbolic form. Different numerical applications of the same model can then easily be produced. The numerical values are assigned either as constants to be hard coded in a C program or as parameters that will be given as an input to the model at run-time.

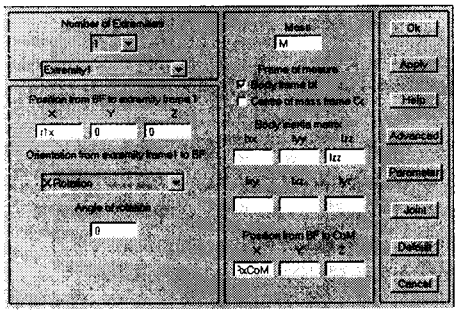

Figure 4: Description Window of a Rigid Block.

The two main building blocks for robots are rigid bodies and flexible beams. The kinematic and dynamic properties of both blocks are entered through a window-type interface, by typing in the names of the variables representing the parameters that are to be considered by the model. Beam flexibility is modeled using assumed modes. The user can enter the number of modes and rigidity to be used to describe the vibrations in the XY-plane, the XZ-plane and torsion around the local X axis.

While bodies are mostly static entities, joints describe rotations and/or translations between bodies. To keep the graphical model concise, all joints are included in the bodies and are always preceding them. Torques, damping, elasticity can all be entered into the model using a virtual power formulation. The virtual power is an expression of the type :

$$\mathbf{F}(\mathbf{q}, \dot{\mathbf{q}}) \cdot \delta \left(\frac{d\mathbf{q}}{\mathrm{dt}} \right) \qquad (1)$$

with which many are unfamilliar but the interface provides a more convivial way to define it for simple joints.

SYMOFROS, along with the interface, allows a user to start a medium-sized model from scratch and have a compiled executable within the hour. This executable can then be interrogated to give the mass matrix and non-linear vector, or any of the matrices that are functions of the model's states and inputs.

4 MODELING

SYMOFROS obtains the symbolic model of a system using Jourdain's principle[5], which is a variation of the generalized d'Alembert's principle. Since it is a variational method, the constraint forces are eliminated. A more complete description of the methodology can be found in Piedbœuf[3].

The kinematics are obtained recursively using Maple. A system of temporary variables is used to avoid an exponential increase in memory requirement for an increasing system complexity[2].

The flexible beams are modeled using the Euler-Bernoulli approximations. The foreshortening is included by considering second-order strain-displacement relationships. Using a consistant elimination of higher order terms, the resulting equations of motion are exact to the first order in terms of the flexible coordinates.

A symbolic linearization of the model is done and a number of C functions required for the simulation and control are generated. The C code is optimized using the Maple optimization. The generated code is ready to be compiled, then used with Matlab/Simulink and the Simulink Real-Time Workshop (RTW).

5 SIMULINK INTERFACE

Once Maple has processed the information related to the model's description, it is able to generate C code. The generated C model is to be used at the Matlab Prompt, in a Simulink simulation or with RTW. In addition, SYMOFROS supports the use of multiple models at the same time. The C model is used in the Matlab/Simulink Environment (essentially using mxArray [2] data structures for storage) but can very easily be adapted to pure C. Finally, SYMOFROS is a multi platform supported package (NT, Win95, SunOS, QNX) giving more flexibility and robustness for the user.

5.1 Accessing the model

The approach taken is similar to the idea exploited by Matlab with the *SimStruct* where the *SimStruct* contains all the information related to an S-Function.

In each MODEL, a C structure (ModelStruct) contains all the relevant information related to it. The ModelStruct contains a set of pointers to functions (see table 1) that, once initialized, point to all the available SYMOFROS functions; the ModelStruct contains pointers to the data storage arrays for the calculation results and a set of informative structures representing the dimensions and configuration parameters. The important issue here is that

[2] A Simulink data structure

each generated MODEL has its own static ModelStruct variable and static functions (initialized in the ModelStruct pointer-to-function section) that can be accessed externally via an initialization function.

Type of Functions
Model Dynamics
Model Kinematics
Holonomic Constraints
Non Holonomic Constraints
Energy

Table 1: SYMOFROS Functions

5.2 Interface with Matlab

The generated model is interfaced with Matlab, Simulink and RTW. It is always very useful to be able to examine the response of a MODEL in the Simulink/RTW environment and at the Matlab command prompt as well. The mechanisms involved in Matlab and Simulink/RTW are different (mexFunction vs SFunction). Since in the end, the same C code is used for the model, a simple interface file has been designed to support both, and properly allocate and free the memory. In Simulink, each MODEL is called once at initialization using *mexCallMATLAB* to get the pointer to the model. Once the pointer to the desired model is obtained, it is accessed directly instead of using *mexCallMATLAB*. The *mexCallMATLAB* is only used in the Simulink environment and not in the RTW. To overcome the problem, a compilation flag is used to determine if Simulink or RTW is to be used. In the later, a direct call to the initialization function is made.

5.3 Generic and Reusable

As will be described in the next section, a set of operators on the model has been developed. These operators are generic enough that the same Simulink diagram can be reused with multiple different SYMOFROS generated models. For example, one could perform a first serie of tests with a rigid model and then study another model with flexible or elastic parameters always using the same Simulink diagram (probably with a different initialisation file).

6 SYMOFROS LIBRARY

By accessing the models' functions with Simulink, it's possible to create a complete real-time simulation within a short development time. SYMOFROS provides the "symoSFunction" block, a Simulink block used to query the model in real-time. By using multiple instances of "symoSFunction", it's possible to build a complete simulation that interacts with one or many models. As shown in table 2, this SYMOFROS query block has two parameters.

Parameters Name	Description
ModelName	Name of the model used to perform the query
ModelFunction	Model function executed

Table 2: Parameters of the block "symoSFunction"

6.1 Library Description

SYMOFROS provides a set of Simulink blocks that allow the execution of model queries, and some standard operations used in robotics. These standard operations are divided in 8 categories.

Initialisation The two blocks defined in this category are used to setup a simulation environment and the model parameters. For multi-model simulations, each model must have an associated "Model Initialisation block".

Dynamics blocks provide the functionality to apply commands to the model. Through these blocks, one can apply torques and trajectories to the different model's joints. It also allows the application of perturbations (external forces and torques) to the model. As results, we obtain the updated states, the joints' accelerations, the joints' dynamic friction forces and constraint forces. Moreover, an inverse dynamics block computes the joints torque from the joints' trajectory (position, velocity and acceleration).

Kinematics are implemented by numerical methods. The direct kinematics outputs the positions, velocities and accelerations of the model extremities. An inverse kinematics block computes the joints trajectory from a cartesian trajectory of a model's extremity. The translational and rotational jacobians and their time derivatives can also be accessed.

Inputs blocks specify predetermined joint and cartesian trajectories. It also allows to predefine a sequence of joint torque, trajectory and perturbation to the model. Generaly, the inputs are specified in an independent file.

Controller blocks provide friction compensation, cartesian linear control and cartesian feedback control.

Graphical blocks deal with plotting the simulation results. Data such as the model states, model joints acceleration and extremity behavior are stored during the simulation. Some blocks are

then used to display the results in graph format. Moreover some development is beeing made to SYMOFROS to get a 3D visual feedback in real-time.

Network & Communication Some blocks have been developed for the support of communication links. Mainly, SYMOFROS has a transmitter and a receiver that transfer data through an internet protocol socket.

Finally, SYMOFROS provides some blocks for generic tasks such as frame transformations, orientation type conversion, orientation error computation, etc.

6.2 Simulation Block Use

To use a particular SYMOFROS block, the user only needs to drag and drop the desired block from the SYMOFROS blockset to his simulink sheet. By double-clicking on the block, a menu appears with the block's settings (figure 5). A short block description and a help button are accessible from that menu.

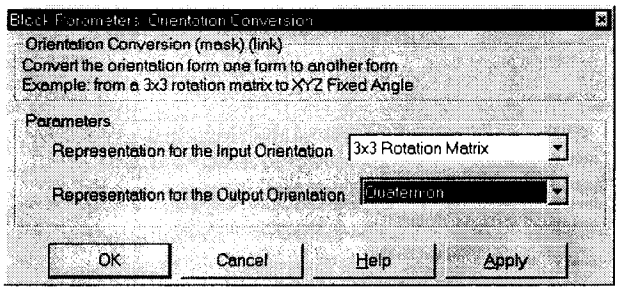

Figure 5: Block Settings

6.3 Simulation Block Implementation

The SYMOFROS blockset is implemented in 3 ways:

Simulink + Toolboxes (.mdl) Use of the standard Simulink blockset and of the Toolboxes blockset (like Digital Signal Processing). Most of the SYMOFROS Blockset has been developed with pre-built Simulink blocks. This method accelerates the development and maintenance processes. Since the simulink diagrams are portable, it is easy to generate real-time code with this method.

SFunctions (.c) C Source code embeded in a Simulink C source file template (through the use of specific macros). This method is used for the functionalities that are not supported by the Simulink blocksets. For example, the SYMOFROS network blocks are implemented with this method in order to support the socket communication. To obtain efficient code, the Target Language Compiler (TLC) is used to define the rules for the code generation.

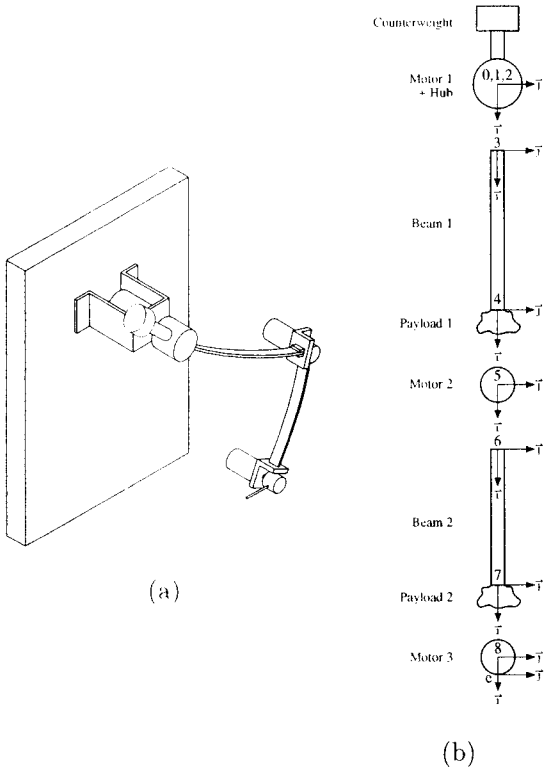

Figure 6: Planar Flexible Robot

Matlab Scripts (.m) This non real-time method is generally used for the initialization processes (simulation + models). Thus, the definition of trajectories and inputs is developed with Matlab scripts.

7 EXAMPLE OF A MODEL

A planar robot with three harmonic drive motors and two flexible links as shown in Figure 6(a), is used to illustrate the capability of the GUI. This robot was built at École Polytechnique and is used to study the modeling and control of flexible robots. Figure 6(b) shows a simplified model using 9 bodies (the counterweight is combined with joint 1) with their associated reference frames. Figure 7 shows the GUI representation of the robot. As indicated on Figure 7, joint elasticity is taken into account in the modeling of the first motor (motor1) while the two others are supposed to be rigid. Beams 1 and 2 are flexible while all other bodies are assumed rigid.

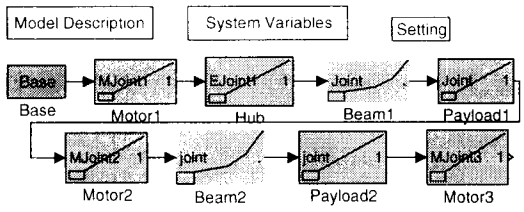

Figure 7: Graphical representation of the model

8 EXAMPLE OF SIMULATION

8.1 Simulation Diagram

Once the model has been entered through the interface, generated in C through Maple, and compiled and linked with Matlab libraries, it can be addressed by either Matlab or Simulink. Some examples of Matlab command lines are:

[SysDim, SysSet, FlexLink, Parameter, Frame, Info] =model(0);

for the initialization of the model's structures and

[Mnl, gnl]= model(1,X,U,Parameters);

to request the mass matrix and nonlinear vectors as a function of the model's states (X) and inputs (U). In the last expression, the first argument, 1, is a function flag indicating what is to be computed, and Parameters is an array containing the values of the model's parameters.

A better way to deal with the SYMOFROS model is to acces it through a Simulink S-Function, which will compute either the direct or inverse of the model's kinematics or dynamics. In fact, two models are generated, one for control and the other for the simulation. Here, the control model is simply the model of the equivalent rigid robot. It is easily obtained by assuming zero modes for each flexible link. An example of such a simulation diagram is shown on figure 8. For clarity of the diagram, blocks used for graphic purpose were removed.

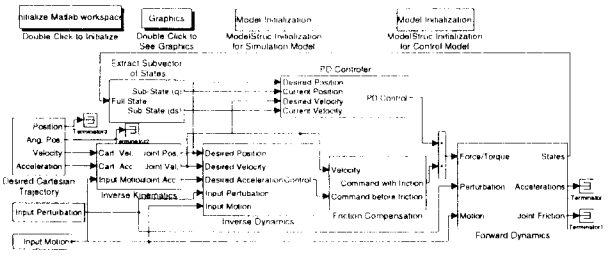

Figure 8: A simulation diagram of a closed-looped system using inverse kinematics.

The four top blocks are, in order, workspace initialization, the graphic display button, model initialization, and a comment block showing the simulation's title. The workspace initialization has to be double-clicked to be activated and calls a script that sets variables used in the simulation. A model's initialization block is required for each different model used in a simulation to set the models' parameters at simulation start-up. All the parameter values correspond to physical values measured on an experimental system that has been developed ad École Polytechnique de Montréal (Canada)??.

Motor	1	2	3
Min. vel. (rad/s)	0.001	0.001	0.001
Stat. fric. (N/m)	0.49	0.20	0.10
Dyn. fric. (N/m)	0.42	0.17	0.08

Table 3: Friction Parameters

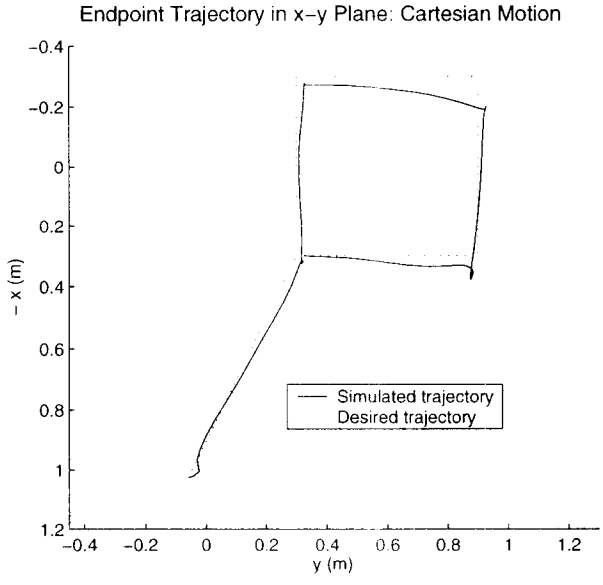

Figure 9: Endpoint Trajectory in the x-y plane.

The diagram shows that the desired velocities and accelerations of the tip, as given by the user, are fed to an inverse kinematics block. The output of that block is the desired positions and velocities of the joints and is fed to a PD Controller, a friction compensation block and an inverse dynamics block that computes the necessary torques to input to the forward dynamics block. The control model is the equivalent rigid robot, easily obtained by assuming zero modes for each flexible links. The forward dynamics block calculates the acceleration of the joints as a function of the present state and the input forces and torques. It also takes into account Coulomb friction in the motor reducers (see table 3). There is no need to integrate the acceleration directly as this S-Function tells the simulation it's states' derivatives and the simulation's integrator takes care of the rest. All these blocks are existing SYMOFROS functions and the same simulation diagram could be called with a different model or different trajectory after

minor changes brought to the initialization file.

The example used for the simulation is the following. The robot starts with motor angles at -0.1, 0.1 and 0 radian for motors 1, 2 and 3 respectively. The goal is to go to point (0.3, 0.3) and from that point, trace a square with 0.6 m sides. the simulation diagram is shown in Figure 9. This shows that we can accurately simulate the behavior of a flexible robot, and the fact that the actual trajectory is not exactly the desired one only means that the controller used is not optimal.

In this example, the desired trajectory corresponded to a square. The figure below shows the superimposed desired and actual trajectories of the robot.

9 CONCLUSION

This paper described the development of a program to model flexible mechanisms in tree-topology with closed kinematic loops and non-holonomic constraints. The description of systems using an object oriented approach have been described. Three main objects are used: body, joint and closure. The dynamic equations are developed using Jourdain's principle with recursive kinematics. The flexible links are modeled as Euler-Bernoulli beams with the inclusion of the foreshortening effect. The method has been implemented in the program SYMOFROS. This program is based on the symbolic language Maple. The graphical user interface developed for SYMOFROS facilitates the input of a model, especially for user with little dynamics experience. The result of the symbolic modeling is a optimized C model. This model is fully compatible with Matlab/Simulink and can be run in non-real-time or in real-time on parallel processors. The SYMOFROS program is an appropriate tool for modeling and simulation of medium complexity mechanisms such as robots. It has been used to develop real-time control and hardware-in-the-loop simulation for robots. A copy of the program can be obtained from the first author.

References

[1] O. Ma, K. Buhariwala, N. Roger, J. MacLean, and R. Carr, "Mdsf- a generic development and simulation facility for flexible, complex robotic systems," *Robotica*, vol. 15, pp. 49–67, 1997.

[2] J.-C. Piedbœuf, "Modelling flexible robots with Maple," *MapleTech: The Maple Technical Newsletter*, vol. 3, no. 1, pp. 38–47, 1996.

[3] J.-C. Piedbœuf, "Recursive modelling of flexible manipulators," *The Journal of Astronautical Sciences*, vol. 46, January-March 1998.

[4] M. Saad, J.-C. Piedbœuf, O. Akhrif, and L. Saydy, "A comparison of different shape function in assumed-mode model of a flexible slewing beam," tech. rep., Canadian Space Agency, St-Hubert, QC, Canada, 1997.

[5] P. E. B. Jourdain, "Note on an analogue of Gauss' principle of least constraint," *The Quarterly Journal of Pure and Applied Mathematics*, vol. XL, pp. 153–157, 1909.