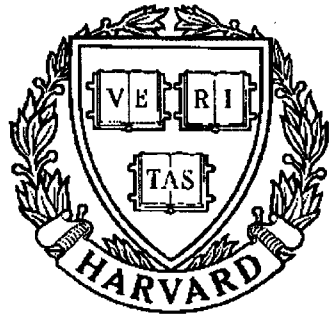
TECHNICAL RESEARCH REPORT

SYSTEMS RESEARCH CENTER



Supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), the University of Maryland, Harvard University, and Industry

Multibody Simulation in an Object Oriented Programming Environment

by N. Sreenath, and P.S. Krishnaprasad

TR 88-35

Cover Sheet

Title : **Multibody Simulation in an Object Oriented Programming Environment**

Author 1 : **N. Sreenath,**

Assistant Professor, Systems Engineering Department, Case Western Reserve University, Cleveland, OH 44106, *Tel* : (216) 368-6129.

Author 2 : **P.S. Krishnaprasad,**

Professor, Electrical Engineering Department, University of Maryland, College Park, MD 20742, krishna@eneevax.umd.edu, *Tel* : (301) 454-6866.

Abstract : A multibody system simulation architecture capable of generating the dynamical equations of the multibody system symbolically, automatically create the computer code to simulate these equations numerically, run the simulation and graphically display the results is discussed. The power of object oriented programming is used systematically to manipulate the symbolic, numeric and graphic modules and produce an effective tool for understanding the complicated motions of multibody systems. The architecture has been implemented for planar two and three body systems in **OOPSS** (Object Oriented Planar System Simulator) a software package written in *Zeta-Lisp*. The package is supported by a nice *user interface* which has the capability to interactively modify system parameters, change runtime initial conditions and introduce feedback control. Plans are underway to implement the architecture for complex multibody systems.

Number of Words : 4100 (approx).

Key Words : *multibody, simulation, object oriented programming, symbolic manipulation, automatic equation generation.*

This work was supported in part by the AFOSR University Research Initiative Program under grant AFOSR-87-0073 and by the National Science Foundation's Engineering Research Centers Program: NSFD CDR 8803012.

in "Advance in Symbolic Computation, SIAM Publ., Philadelphia
ed. R. Grossman, 1988.

Multibody Simulation in an Object Oriented Programming Environment

N.Sreenath[1] **P.S. Krishnaprasad**[2]

Abstract : A multibody system simulation architecture capable of generating the dynamical equations of the multibody system symbolically, automatically create the computer code to simulate these equations numerically, run the simulation and graphically display the results is discussed. The power of object oriented programming is used systematically to manipulate the symbolic, numeric and graphic modules and produce an effective tool for understanding the complicated motions of multibody systems. The architecture has been implemented for planar two and three body systems in **OOPSS** (Object Oriented Planar System Simulator) a software package written in *Zeta-Lisp*. The package is supported by a nice *user interface* which has the capability to interactively modify system parameters, change runtime initial conditions and introduce feedback control. Plans are underway to implement the architecture for complex multibody systems.

1 Introduction

A multibody system is simply a collection of bodies, rigid or non-rigid, interconnected by means of joints with certain specific kinematic properties [25]. A large class of mechanical systems can be classified as multibody systems. We list here a few examples : spacecraft, robot manipulators, land vehicles (automobiles etc.), the human body, molecular interconnection of atoms, cables modeled as series of rigid bodies etc., [26]. A *planar multibody system* is a multibody system with motions of the system restricted to a plane. To study the motions of multibody systems the knowledge of the dynamics of the system is essential. The dynamics of a multibody system cannot

[1]Assistant Professor, Systems Engr. Dept., Case Western Reserve Univ., Cleveland, OH 44121

[2]Professor, Dept. of Electrical Engr., Univ. of Maryland, College Park MD 20742.

be adequately approximated by linear differential equations, since large motions are characteristic of such systems. Further complications arise with the introduction of control variables, and, external and internal disturbances.

Simulation has been one of the important tools for understanding the motions of a multibody system. The steps involved in simulating the motions of a complex multibody system are as follows. First, one generates a dynamical model of the multibody system in the form of differential equations. Next a simulation program for numerically integrating these differential equations is created. Then the simulation code is run for a relevant set of parameters and a specified initial state of the system to compute the trajectory of the system.

Thus the primary step in understanding the motions of a multibody system is the formulation of the dynamical equations. For the case when the system is in the form of an open kinematic chain (Figure 1) , i.e., the path from one body in the system to any other body in this system is unique, and each body in the system can be modeled as a rigid body, we get a set of coupled, highly nonlinear first or second order ordinary differential equations.

Multibody dynamical formalism has been the subject of a lot of research [4], [7], [8], [12],[14], [17], [19], [21], [28], [24], [25]. The formulation of dynamical equations by hand is a tedious process and often prone to errors. Many researchers have considered the possibility of computer-aided methods to generate these equations. General purpose multibody computer programs capable of generating the dynamical equations as well as simulating them are available for quite sometime. See [18], [21] for references.

These computer-oriented methods may be classified as numerical [5], and symbolic programs [1], [3], [13], [15], [18], [20], [23], [25]. Numerical programs are characterized by numerical digital computation whereas symbolic programs generate equations and accompanying expressions in symbolic (or alpha-numeric) form on the basis of alpha-numeric data. Symbolic programs in general are more efficient in terms of running time in comparison with numerical programs [21], [27].

In this paper we present a general purpose software system architecture designed

to generate the dynamical equations of a multibody system *symbolically* [3] , automatically generate the computer code to simulate these equations *numerically*, run the simulation and display the results graphically. This architecture is implemented for planar two and three body systems in the package called **OOPSS** - Object Oriented Planar System Simulator. A nice *user interface* is part of **OOPSS**.

The paper is organized as follows. In Section 2 we discuss the system features, followed by a brief exposition on multibody dynamics in Section 3. The Object Oriented Programming methodology and the software architecture of **OOPSS** is discussed in Section 4. Section 5 discusses the implementation of **OOPSS** architecture for planar two and three body systems followed by the conclusion in Section 6.

2 System Features

OOPSS uses Object Oriented Programming along with *symbolic manipulation* to formulate and simulate the dynamics of a planar multibody system automatically. A mathematical model describing the motion of a planar multibody system (dynamic model) is generated by **OOPSS** symbolically. The symbolic manipulation has been implemented in MACSYMA[4]. A program to numerically simulate these differential equations is generated. **OOPSS** animates the multibody system by exploiting the high resolution graphics and windowing facilities of a LISP machine and has been implemented in Zeta-Lisp on a Symbolics 3600 [5] series machine. A nice *user interface* is provided for interacting with the symbolic, numeric, and graphic elements of **OOPSS**.

Users can interactively :

(i) choose any kinematic or physical parameters for the system,

[3]Only multibody systems connected in the form of a tree (Figure 1) with pin joints are considered here.

[4]MACSYMA is a trademark of Symbolics Inc., Mass.

[5]manufactured by Symbolics Inc., Mass.

(ii) change any runtime initial condition - system energy, system angular momentum, time step, maximum problem time, initial values of state and other variables (angles, conjugate momentum variables),

(iii) select display parameters for the graphs,

(iv) choose feedback control torque laws and gains.

One of the significant features of **OOPSS** is that various control schemes can be easily implemented and evaluated. Currently *proportional, proportional-derivative, sinusoidal-spring*, and, *sinusoidal-spring with bias*, feedback control schemes have been implemented. These control laws can be interactively selected, evaluated and the control gains tuned. A model-dependent control scheme, for example exact linearization [9], [10], could be easily implemented since the associated feedback control law could be formulated using symbolic computation.

OOPSS can be used a design tool to design the multibody system parameters. It can be used as an experimental work-bench to study certain problems of mechanics. To enhance our knowledge about the phase space of a multibody system, we could simulate the equilibria of the multibody system and explore the stability of such equilibria.

3 Multibody Motion

Before embarking on the description of the **OOPSS** system it is necessary to introduce the various vectors, parameters and variables associated with the motion of a multibody system. Associated with every body in the system are physical parameters like such as mass, inertia and various kinematic parameters. The vectors connecting the a joint and the center of mass of the body, and, the same joint and a corresponding joints on an adjacent body, provide information on the kinematic description of the multibody system. Since the individual bodies are considered rigid, at any time instant, the orientation of the body and the location of its center of mass provides sufficient information to determine the configuration of the body. Other quantities such

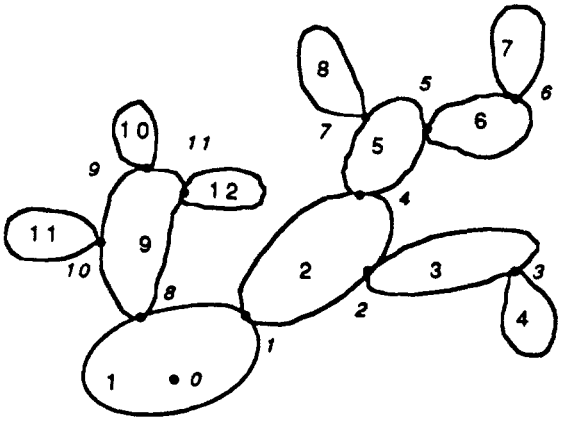


Figure 1: Multibody system connected in the form of a tree

as angular velocity and angular acceleration are associated with each body, and quantities such as kinetic energy, angular momentum and linear momentum are associated with the multibody system itself [6].

The motions of the multibody system are defined with respect to an inertial coordinate system. For a planar multibody system, a local coordinate system is defined for every body in the system and an angle this coordinate system makes with respect to the inertial coordinate system is the inertial angle associated with the body. The angle between the local coordinate systems of two bodies is the relative angle between the two bodies.

The dynamics of a planar multibody system connected in the form of a tree can be described by a set of first order differential equations in terms of the Hamiltonian of the system $\mathbf{H}$ [7] (which is also the kinetic energy of the system) [21] , the relative

[6]To define the multibody system mathematically it is necessary to label every body and every joint in the system uniquely in increasing order of consecutive integers. Also for simplicity of computation it is convenient to label the bodies such that the body labels are of increasing magnitude along any topological path starting at body 1; the joint connecting a body i to the body with a lesser body-label is labelled as joint *(i-1)*. The joint *(i-1)* is also known as the *previous joint* of body i.

[7]See Appendix 1 for a planar two body system dynamics example.

angles $\theta_{i,j}$ between the bodies, and the conjugate momenta μ_i.

4 System Description

OOPSS is implemented using the *Object Oriented Programming* (OOP) technique. We start with a brief introduction to the OOP methodology and discuss the important **flavors, methods,** and **functions** used in the program. The system architecture is discussed next.

4.1 Flavors, Methods and Functions

Objects are entities that combine the properties of procedures and data, since they perform computation and save local state [22]. Also, objects could be linked to real world things. A program could be built using a set of *objects*. In OOP we have uniform usage of objects whereas conventional programming uses separate procedures and data. Sending *messages* between objects causes action in OOP. Message sending is a form of indirect procedure call and supports *data abstraction*. The *inheritance* property in OOP enables the transmission of changes at a higher level to be broadcast throughout the lower levels. *Functionality encapsulation* and the *inheritance* properties enable the designer to create *reusable software* components termed as the *Software-IC*'s in OOP [2].

We have used *Zeta-Lisp* a programming language capable of object oriented programming. General descriptions of objects in *Zeta-Lisp* are in the form of **flavors**. **Flavors** are abstract type of object class. In Zeta-Lisp a conceptual class of objects and their operations are realized by the **Flavor System**, where part of its implementation is simply a convention in procedure calling style; part is a powerful language feature, called **Flavors**, for defining classes of abstract objects. Flavors have *inheritance* property; thus if we build a flavor using other flavors then all the properties of the latter are inherited by the former. Any particular object is an *instance* of a flavor. The variables associated with a generic object are known as *instance variables*.

Specific operations could be associated with the objects using *methods*. One can create a method to define a specific operation on any instance of a flavor and attribute special properties to it. For instance, one can define a method for a body which is the leaf of a tree and so has only one joint (i.e., only one body attached to it - contiguous to and inboard) whereas a generic body has two or more bodies attached to it. Functions are used to send messages to instances of flavors through the already defined methods.

The OOP methodology is conducive to building *iconic user* interfaces since all except the necessary information can be hidden so as to limit clutter and enhance clarity of the issues under consideration. *Methods* provide the basic abstraction of a class of the objects via the *inheritance* property of the OOP.

In the planar multibody setting the primary flavor used to describe a *generic body* in the system is :

general-planar-body

A *generic body* in the planar multibody system can be defined as an object with the following instance variables : a vector connecting the previous joint to the center of mass of body, vector(s) connecting the previous joint and other joint(s) on the body, and, the angle made by the body frame with respect to the inertial coordinate system (also called orientation of the body).

The center of mass of a *generic body* is defined by the use of the method **:center-of-mass**, this ensures that at any time instance knowing the location of the previous joint and the orientation of the body, the body center of mass could be calculated. The information about the orientation of the body is calculated in the **Numerical Simulator** using the dynamical equations. The position of the next joint is defined by using the method **:next-joint**. Note here that the next joint of one body will be the previous joint of another body.

Similarly the **:draw-body** method is used to draw the *generic body* for animation on the screen. This method utilizes the center of mass, the next joint information, and the orientation of the body to create the animation of the multibody motion.

Another flavor

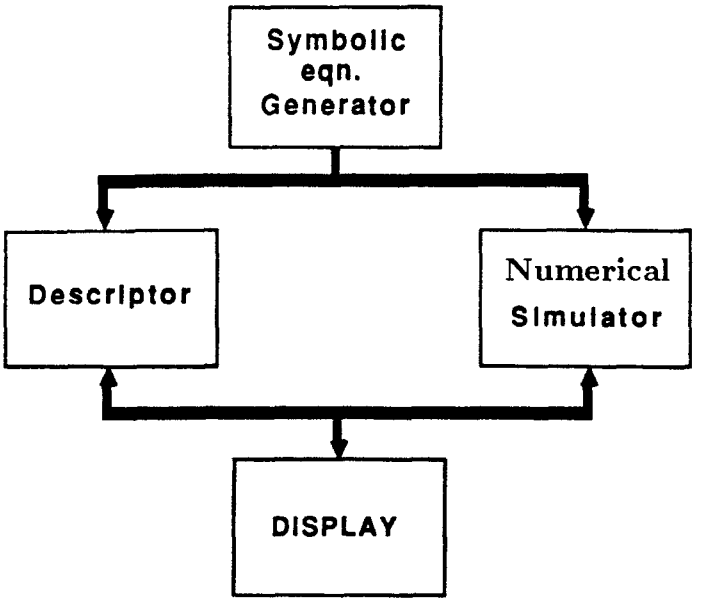


Figure 2: Block diagram representation of **OOPSS**

graphics-window-frame

is implemented to create the *graphics window* and the various *panes* associated with it, for animation and data display purposes. This flavor also implements the *menu* and mouse selectable capabilities for the OOPSS system. A number of *functions* are implemented to send messages to the flavors; a few important functions are listed and their functions are self explanatory : *set-parameters, set-window-frame, on-line-info, show-herald*, etc.

4.2 System Architecture

The **OOPSS** system can be represented by : **Symbolic Equation Generator**, **Numerical Simulator**, **Descriptor**, and, **Display** modules which are interconnected as shown in Figure 2. A detailed description of each module is given in the following paragraphs.

4.2.1 Symbolic Equation Generator

The symbolic equation generator generates the dynamics of a planar multibody system connected in the form of a tree structure in the Hamiltonian setting (see

Sreenath [21] Chapter 3, for the formulation and notation). This module is implemented in MACSYMA. The input data for this module consists of information describing the way the bodies are interconnected, kinematic and physical parameters like lengths, mass, inertia etc., any control (actuating) or disturbance torques acting on the multibody system. For a two body system example one form of the output equations from this module are shown in Appendix 1.

4.2.2 Numerical Simulator

The **Numerical Simulator** which simulates the dynamical equations generated by the symbolic equation generator, has been implemented using FORTRAN-77 running on the LISP machine. The **Numerical Simulator** needs numerical values of all parameters to be in an data file. This input data file is generated by the **DESCRIPTOR**. The file contains the numerical values of all kinematic and physical parameters, the system angular momentum and system energy values, problem time, time step etc., associated with the particular example.

The state and related variables (for example, angular velocities) at any instance of time could be passed onto the **DISPLAY** module by means of *lispfunctions* to be used for animation and display purposes. The *lispfunction – displaybody* passes the relevant variables like orientation, angular velocities of the bodies etc., from the FORTRAN program to the **DISPLAY** module for animating the motion of the multibody system. The function '*displaybody*' is implemented in Zeta-Lisp in the **DISPLAY** package for the actual animation.

The initial condition for initiating the simulation is chosen such that the the various physical laws governing the conservation of select quantities (like system angular momentum, system energy) are satisfied.

4.2.3 DESCRIPTOR

The **DESCRIPTOR** consists of descriptions of various flavors to implement the display and the user interface. It also contains flavors to define a generic body

in the multibody system. Using *methods* we can attribute special properties to the instances of these flavors. This module also functions as an intermediary between the user interface and the **Numerical Simulator** module by generating the input data file for the FORTRAN program before a numeric simulation run is started, based on input from the user.

4.2.4 DISPLAY

The **DISPLAY** module is the implementation of various functions to drive the instances of flavors by sending messages to them. **DISPLAY** keeps track of sending proper messages to the relevant panes as and when it receives data from the **Numerical Simulator** module. **DISPLAY** is characterized by a '*tv:alu-xor*' option which helps in erasing the display at time t and creating a new display at time $t + \Delta t$. '*displaybody*' and '*cleanbody*' are functions to display the system and clean the displayed picture off the relevant pane respectively.

The *user interface* consists of a window with many panes (Figure 3). Three frames of references in the corresponding window panes: inertial frame of reference and two other selectable frames (from various joint and/or body frames), have been implemented. The animation of the multibody system in the selected frames of reference are displayed in their respective panes. A part of runtime data from the FORTRAN program is displayed in the *message pane*. Graphs of state and/or other variables are drawn as functions of time in the *simulation pane*. The *menu pane* which is in the lower right hand corner is self descriptive. Every item in this pane is *mouse sensitive* (mouse selectable). We will say more on this in the following sections as we deal with particular examples of planar two-body and planar three-body systems. A brief online *HELP* facility exists and information can be got by clicking left using the mouse when it is highlighted.

5 Implementation

Presently the **OOPSS** system architecture has been implemented for the planar

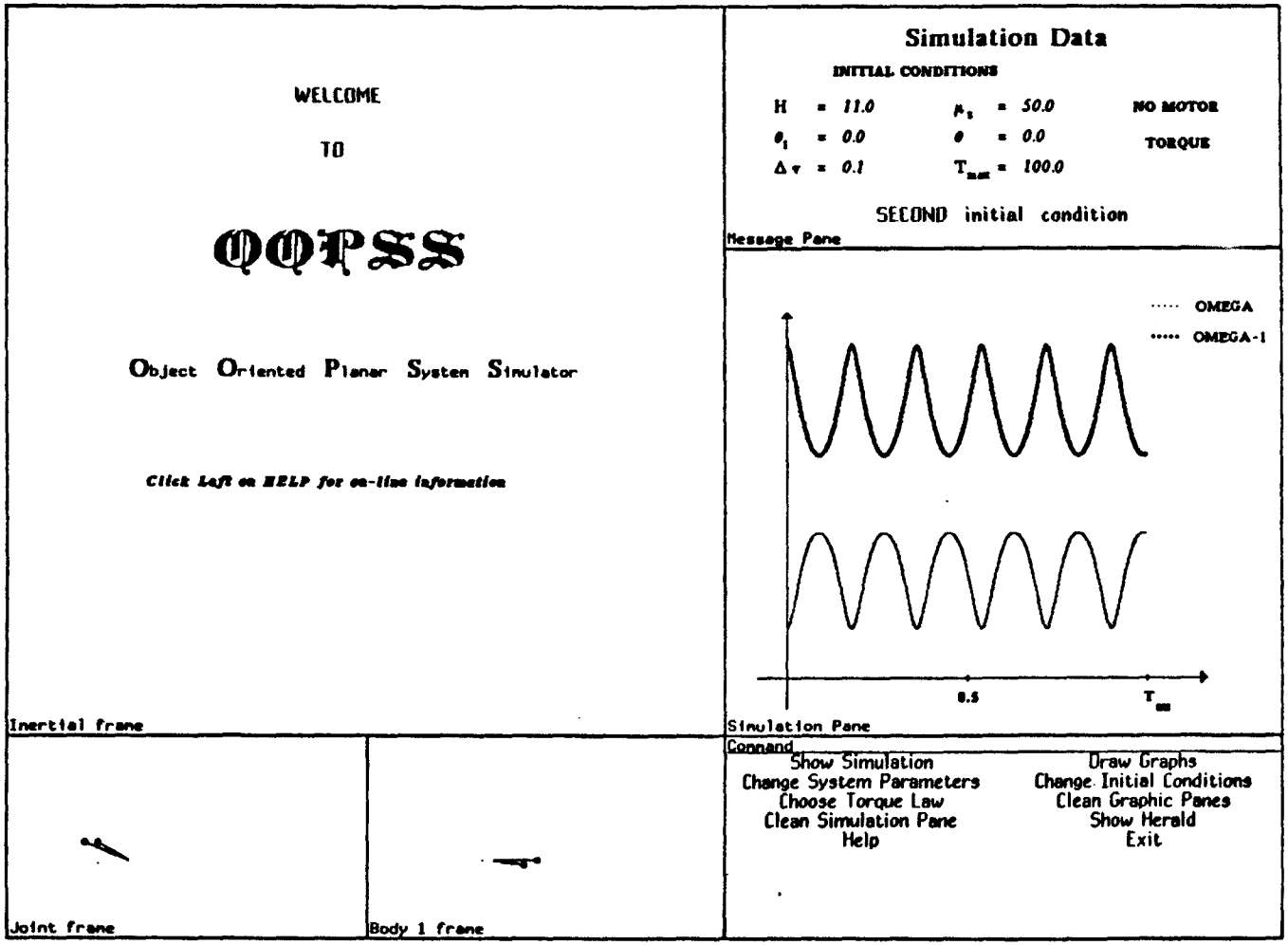


Figure 3: **OOPSS** window

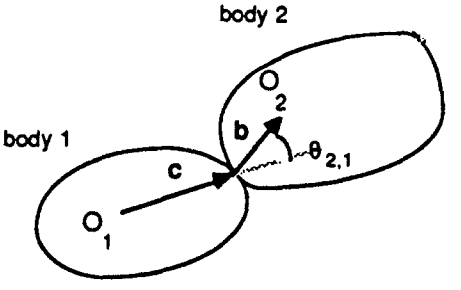


Figure 4: Planar two body system

two-body and the planar three-body systems. The following examples illustrate the capabilities of the **OOPSS** system.

5.1 Two-Body System

A system of two bodies in space connected together by a one degree of freedom joint occurs in many contexts (Figure 4). One of the bodies, for example, may be a large space based sensor (say a space telescope). Rapid re-orientation of the sensor from one stationary position to another may be desirable. Such a "step-stare" maneuver would require a through knowledge of the dynamics and thus a good mathematical model [8] to formulate the necessary control [16].

Figure 5.a shows the menu pane for this example. Clicking left on the mouse when *system parameters* is highlighted gives Figure 5.b. Clicking left on *Torque Law* results in Figure 5.c. Further clicking on *Proportional-Derivative* (P-D) torque law implements a joint torque law (internal torque) - a proportional sinusoidal biased spring plus derivative controller, i.e., $T_{\text{joint}} = (K_p \sin(\theta_{2,1} - \theta_{\text{bias}}) + K_d \dot{\theta}_{2,1})$. Gains K_p (proportional gain) and K_d (derivative gain), $\theta_{2,1}$ is the relative angle between body

[8] Refer to Sreenath [21] Chapter 4 for a detailed description of the problem.

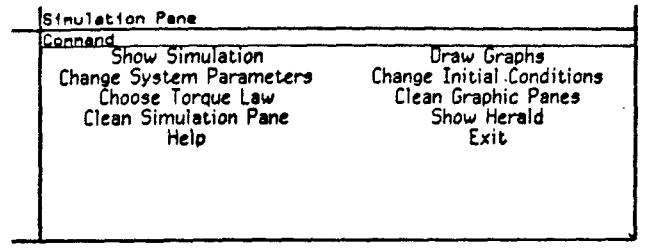
1 and body 2, and, the *bias* angle θ_{bias} could be chosen (see Figure 5.d) interactively by the user.

The evolution of motion of a two body system with a fixed value of system energy and angular momentum is given in Figure 6. The largest *pane* of the display window is the *Inertial frame* where the motion of the system could be observed in the inertial coordinate frame. Each body is represented by a stick figure with a big *filled-in-circle* at one end (representing the center of mass of the body) and a smaller circle at the other end representing the joint connecting the other body. The center of mass of the multibody system is defined by the point in the center of the *inertial* frame. Since no external force is present on the system the center of mass of the system is a fixed point in inertial frame. Two other coordinate frames where the motion of the system could be observed - the *joint frame* and the *Body-1 frame*[9] are also displayed below the *inertial frame.*

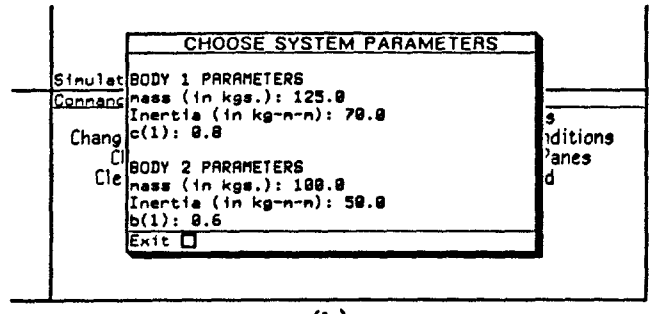
The two-body system has two relative equilibria [10] when the bodies are in a extended position (stable) and when the bodies are in folded position (unstable). There is a *homoclinic orbit* associated with the unstable equilibrium point. The *stable equilibrium* is displayed in Figure 7. The trace shown in the inertial frame is the *trace* left by the joint as it moves in space. Simple calculation shows that this trace is indeed a circle when the system is in stable equilibrium position. Figure 8 displays a trajectory when the system is at a point very near the *unstable equilibrium* point. If a (P-D) torque is introduced in the system then the resulting trajectory is as shown in Figure 9 (no bias) and Figure 10 (bias). Notice that with K_p equal to zero and K_d positive (Figures 9-10) the system always goes to the stable equilibrium and confirms the 'stabilization theorem' - Theorem 6.3.1 in Sreenath [21], i.e., introduction of a feedback internal torque proportional to the rate of the relative angle stabilizes the system. One could also interpret this result as follows: by introducing this torque

[9]The *joint-frame* is a frame parallel to the inertial coordinate frame with the origin at the joint; the *Body-1 frame* is the local coordinate frame of body 1 located at the body center of mass.

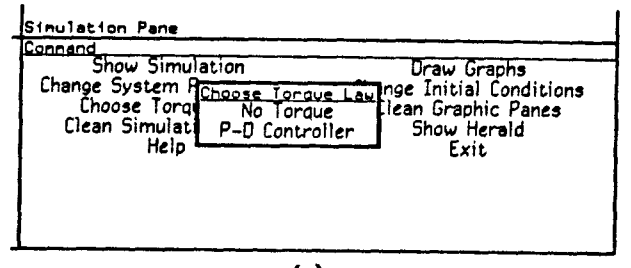
[10]When all the bodies in the system are rotating with constant inertial angular velocity and no relative motion we have relative equilibrium



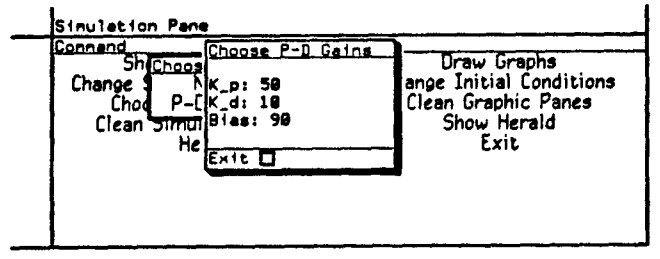
(a)



(b)



(c)



(d)

Figure 5: Two-body problem menu-pane

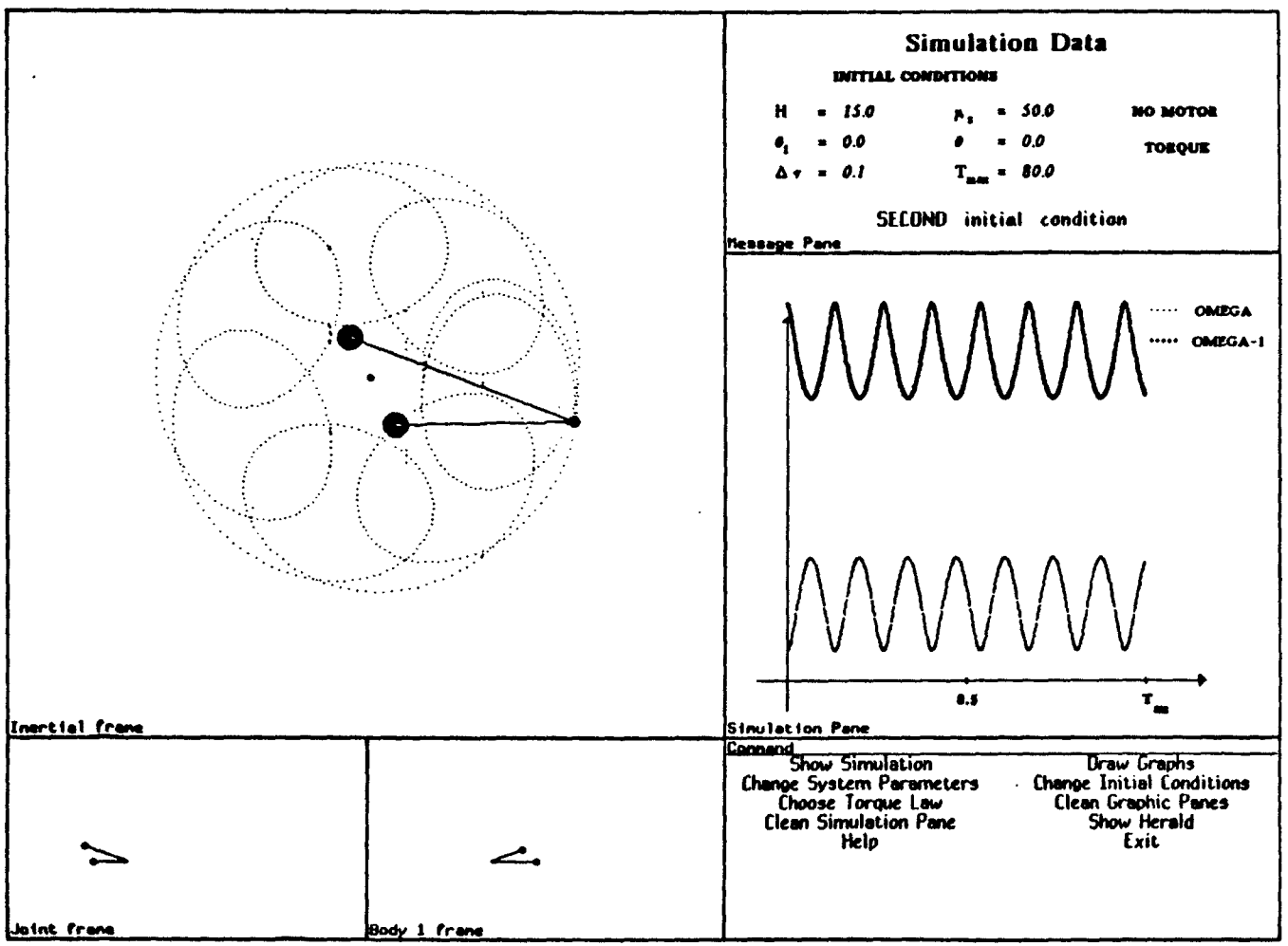


Figure 6: Two-body problem : $H = 15$

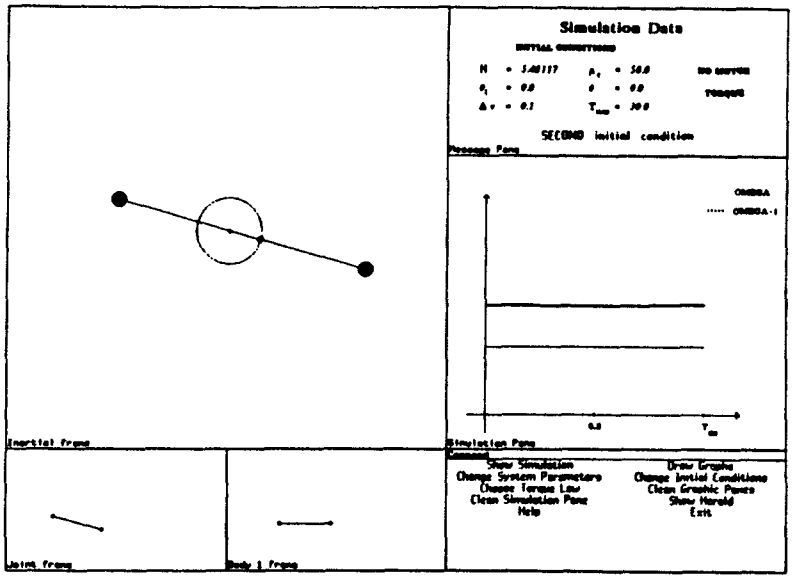


Figure 7: Two-body problem : stable equilibrium

law the energy in the system is dissipated till the system goes to a minimum energy state which is the stable (stretched out) relative equilibrium.

5.2 Three-Body System

A general three-body example (Figure 11) has also been implemented on the basis of the **OOPSS** architecture. Figure 12 shows a general three-body system wherein the center of mass of the middle body is not along the line joining the two joints. The *filled-in circles* represents the center of mass of each body (the first body represented by a big circle, second with a smaller circle and the third with the smallest circle). Display frames could be chosen by clicking left on the 'Choose display frames' using the mouse. Figure 13 displays special kinematic case where the center of mass of the middle body is along the line joining the two joints. Joint torques of the proportional sinusoidal bias spring plus derivative type has be introduced at the joints.

5.3 Complex Multibody Examples

Plans are underway to implement complex multibody system examples. As the complexity of the examples grow one may find oneself limited by the processing capabilities of the currently available LISP machines. One could take advantage of the

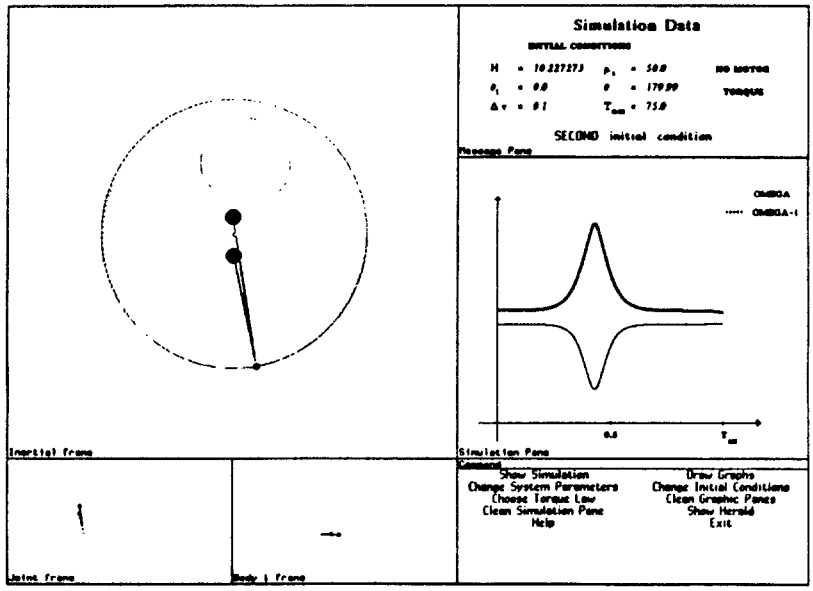


Figure 8: Two-body problem : unstable equilibrium

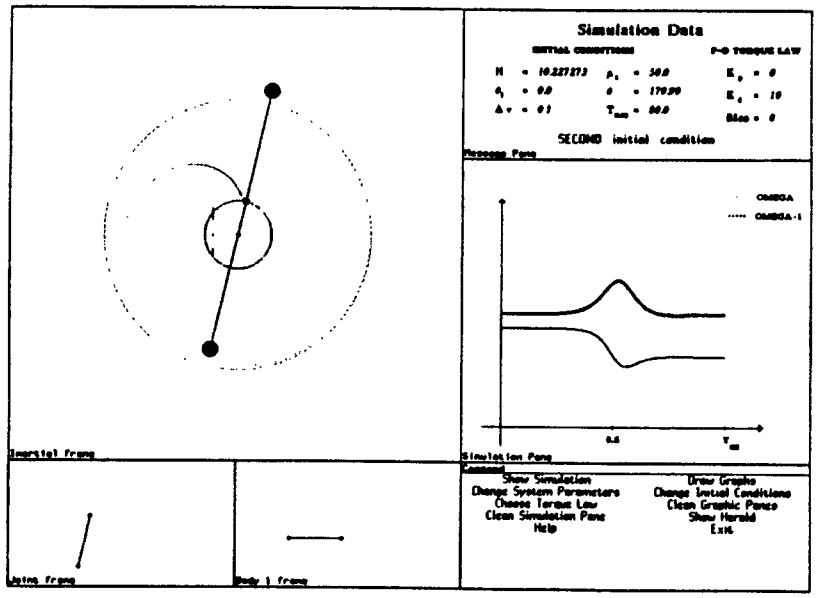


Figure 9: Two-body problem : $K_d = 10$

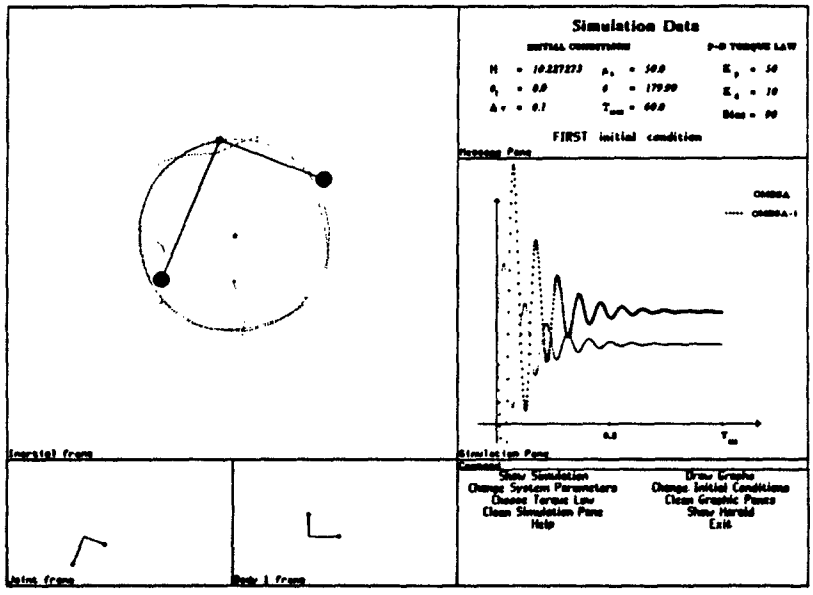


Figure 10: Two-body problem : $K_p = 50$, $K_d = 10$, bias=90

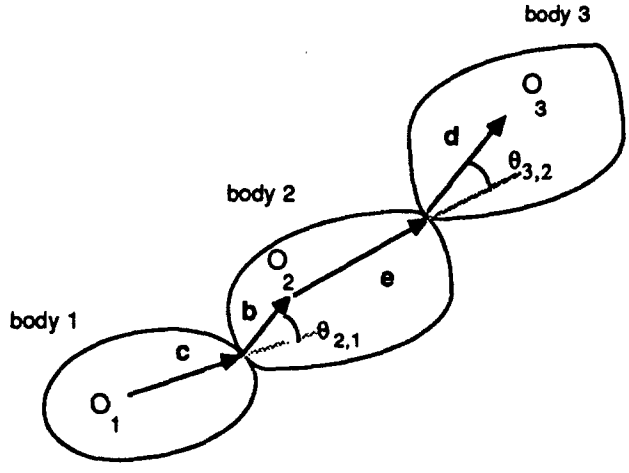


Figure 11: Planar three body system

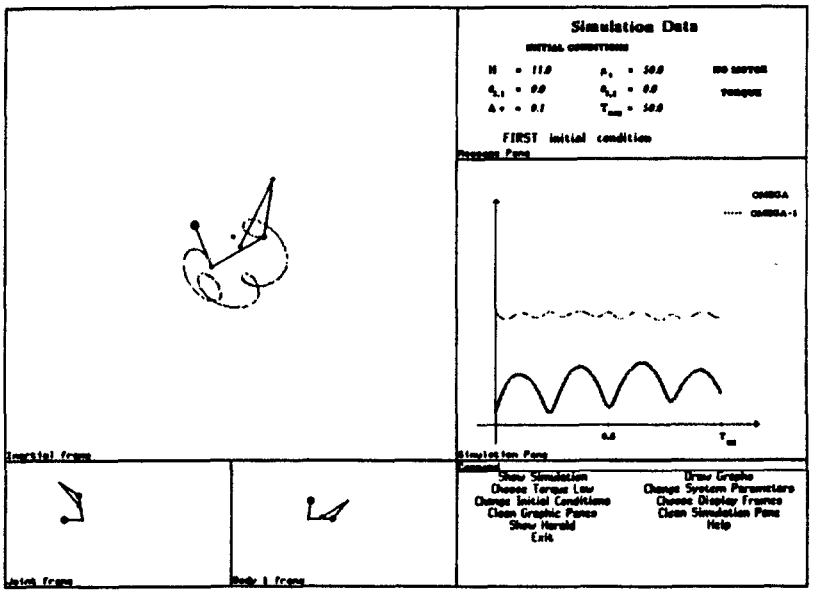


Figure 12: Three-body problem : general case

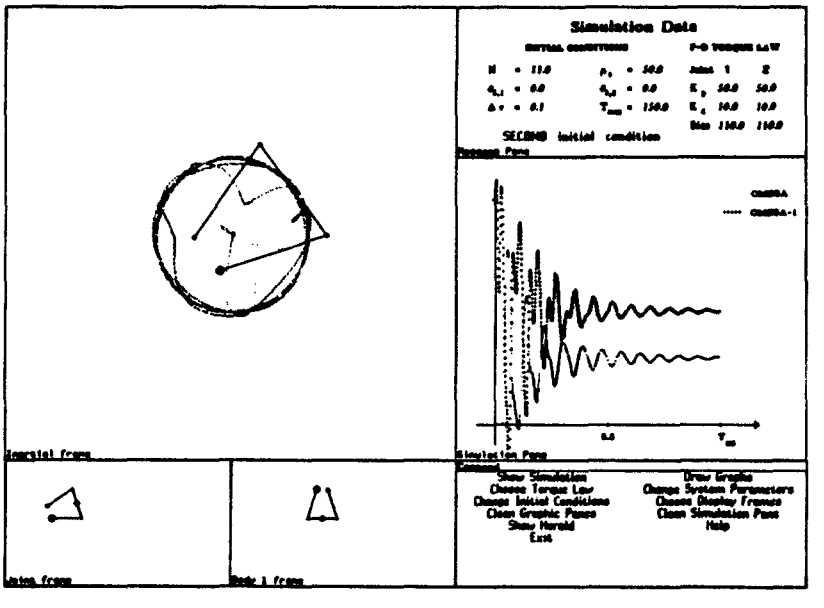


Figure 13: Three-body problem : special kinematic case with joint torques

existing parallelism in these problems by utilizing the processing power of parallel LISP processors. A *Connection Machine* [11] may just serve the purpose. Thus dynamics of complex multibody systems may be generated automatically, simulated and animated.

6 Conclusion

A prototype object oriented software architecture for multibody system simulation is discussed. **OOPSS** can generate the dynamic model of a planar multibody system symbolically, generate the computer code to simulate it numerically, run the simulation and display the result by means of animation and graphs. **OOPSS** could be used as a test-bed to evaluate control algorithms, select control gains, and design the system parameters for the multibody system, interactively. The system has been successfully implemented for planar two and three body systems on a Symbolics 3600 series machine in Zeta-LISP, FORTRAN-77 and MACSYMA. Plans are underway to implement the **OOPSS** architecture for more complex multibody systems.

[11] manufactured by Thinking Machines Inc, Cambridge, Mass.

Appendix

We develop the dynamical equations of a planar multibody system in space in terms of the Poisson bracket. The symmetries (translational and rotational) acting on the system are taken into account to reduce the dynamics appropriately. We refer the interested reader to Sreenath[23] and Sreenath, Oh, Krishnaprasad and Marsden [25] for details and proof.

A tree connected multibody system in space (see Figure 1), with the total system angular momentum conserved, is considered. The configuration space Q for such a system is

$$Q \cong \underbrace{(S^1 \times \cdots \times S^1)}_{N \text{ times}} \times \mathbb{R}^2,$$

where N is the number of bodies in the system. One way of coordinatizing the system on the tangent bundle TQ is by (θ_i, ω_i), $i = 1, \ldots, N$. The Lagrangian can then be written in these coordinates as

$$L = \frac{1}{2}\underline{\omega}^T \mathbf{J} \underline{\omega} + \frac{\| \mathbf{p} \|^2}{2m}.$$

where $\underline{\omega}$ is the vector of angular velocities, $\mathbf{p}$ is the linear momentum of the center of mass of the system, and $\mathbf{J}$ is the pseudo-inertia matrix associated with the system and is a function of relative angles between the bodies [12].

The Hamiltonian is simply the kinetic energy of the system and can be constructed using the Legendre transformation as

$$H = \frac{1}{2}\underline{\mu}^T \mathbf{J}^{-1} \underline{\mu} + \frac{\| \mathbf{p} \|^2}{2\mathbf{m}}.$$

where $\underline{\mu}$ is the conjugate momentum vector and is related to $\underline{\omega}$ by

$$\underline{\mu} = \mathbf{J}\underline{\omega}$$

We now recognize the symmetries in the system and reduce the dynamics accordingly. The reduction technique we use is originally due to Arnold [1] and developed further by Marsden and Weinstein [17]. In the general setting one starts

[12] For example, for the case of planar two-body system $\mathbf{J} = \mathbf{J}(\theta_{2,1})$.

with a Poisson manifold P and a Lie group G acting on P by canonical transformations. The reduced phase space P/G (provided it has no singularities) has a natural Poisson structure whose symplectic leaves are the Marsden-Weinstein-Meyer spaces $J^{-1}(\mu)/G_\mu \approx J^{-1}(\mathcal{O})/G$ where $\mu \in \mathrm{g}^*$, the dual of the Lie algebra of G, $J : P \to \mathrm{g}^*$ is an equivariant momentum map for the action of G on P, G_μ is the isotropy group of μ (relative to the coadjoint action) i.e., $G_\mu = \left\{g \in G : Ad^*_{g^{-1}}\mu = \mu\right\}$, and $\mathcal{O}$ is the coadjoint orbit through μ. The coadjoint orbit $\mathcal{O}$, is even dimensional.

Reduction by translations (planar multibody problems) :

We reduce the dynamics by the action of the translation group $\mathbb{R}^2$. This group acts on the original configuration space Q by

$$\mathbf{v} \cdot ((R(\theta_1), \mathbf{r}_1), \ldots, (R(\theta_N), \mathbf{r}_N)) = ((R(\theta_1), \mathbf{r}_1 + \mathbf{v}), \ldots, (R(\theta_N), \mathbf{r}_N + \mathbf{v}))$$

where $\mathbf{r}_i$, $i = 1, \ldots, N$ is the vector from the origin of the frame of reference to the center of mass of body i; θ_i is the angle made by the body i with respect to the frame of reference. $R(\theta_i)$ is the (2×2) rotation matrix associated with body i ,

$$R(\theta_i) = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{bmatrix}, \quad i = 1, \ldots, N.$$

The induced momentum map on TQ is calculated by the standard formula

$$J_\xi = \frac{\partial L}{\partial \dot{q}_i} \xi^i_Q(q),$$

or on T^*Q by

$$J_\xi = p_i \xi^i_Q(q),$$

where ξ^i is the infinitesimal generator of the action on Q. (see Abraham and Marsden [2]). Coordinatizing Q by θ_i, $i = 1, \ldots, N$ we determine,

$$J_\xi = \langle \mathbf{p}, \xi \rangle, \ \xi \in \mathbb{R}^2.$$

Thus $J = \mathbf{p}$ is conserved since H is cyclic in $\mathbf{r}$ and so is translation invariant. The corresponding reduced space is obtained by fixing $\mathbf{p} = \mathbf{p}_0$ and letting

$$P_{\mathrm{p}_0} = J^{-1}(\mathbf{p}_0)/\mathbb{R}^2,$$

(see Chapter 4, Abraham and Marsden, [2],). But P_{p_0} is clearly isomorphic to $T^*(\underbrace{S^1 \times \cdots \times S^1}_{N \text{ times}})$ i.e. to the space of $\theta_1, \ldots, \theta_N$ and their conjugate momenta $(\mu_1, \ldots, \mu_N)$. The reduced Hamiltonian is simply the Hamiltonian as before with $\mathbf{p}$ regarded as a constant.

We can adjust H by a constant and thus assume $\mathbf{p} = 0$; this obviously does not affect the equations of motion. Thus,

$$H = \frac{1}{2}\underline{\mu}^T \mathbf{J}^{-1} \underline{\mu}.$$

Furthermore, the configuration space Q after reduction by translations i.e., reduction to the center of mass frame is

$$Q \cong (\underbrace{S^1 \times \cdots \times S^1}_{N \text{ times}}).$$

Reduction by Rotations :

The rotational symmetry group S^1 acts on the cotangent space as below :

$$\begin{aligned} \theta \cdot ((\theta_1, \mu_1), \cdots, (\theta_N, \mu_N)) &= ((\theta_1 + \theta, \mu_1), \cdots, (\theta_N + \theta, \mu_N)) \\ &= ((\theta_1, \mu_1), \cdots, (\theta_N, \mu_N)). \end{aligned}$$

Since S^1 is diffeomorphic to $SO(2)$ (the special orthogonal group of (2×2) matrices) and the Lie algebra of $SO(2)$ is $so(2)$ (skew-symmetric matrices with determinant not equal to zero), the momentum map can be viewed as a map

$$J : T^*Q \to so^*(2)$$

where $so^*(2)$ is the dual of the Lie algebra of $SO(2)$

Let $\xi \in so(2)$, then $\exp(t\xi) \in O(2)$. The *infinitesimal generator* $\xi_Q(q)$ can now be calculated as follows :

$$\begin{aligned} \xi_Q(q) &= \frac{d}{dt}\Phi(\exp(t\xi), q) \mid_{t-0} \\ &= \frac{d}{dt}(\theta_1 + t, \ldots, \theta_N + t) \mid_{t=0} \\ &= (1, \ldots, 1). \end{aligned}$$

The *momentum map* is given by

$$J : T^*Q \ \rightarrow \ so^*(2),$$

with the *momentum* $\mathbf{P} : TQ \rightarrow \mathbb{R}$

$$\begin{aligned} P(v_q) &= \hat{J}(\xi)(v_q) \\ &= FL(v_q) \cdot \xi_Q(q) \\ &= \langle v_Q, \xi_Q(q) \rangle \quad \text{on } TQ. \end{aligned}$$

where $FL: \ TQ \rightarrow T^*Q$ is the *fiber derivative.*

The metric here is the Riemannian metric associated to kinetic energy and the inner product '$\langle , \rangle$' is given by $\langle \underline{x}, \underline{y} \rangle = \underline{x}^T \mathbf{J} \underline{y}$.

$$\begin{aligned} \mathbf{P}(v_q) &= \langle (\omega_1, \ldots, \omega_N) \ , \ (1, \cdots, 1) \rangle \\ &= [1, \ldots, 1] \mathbf{J} \underline{\omega} \ \text{ on TQ}, \end{aligned}$$

or on T^*Q, for $\alpha_q \in T^*Q$ we have

$$\mathbf{P}(\alpha_q) \ = \ \mu_1 \ + \ \cdots \ + \ \mu_N.$$

i.e.,

$$J((\theta_1, \mu_1), \ldots, (\theta_N, \mu_N)) \ = \ \mu_1 \ + \cdots + \ \mu_N.$$

We now form the Poisson reduced space

$$P \ := \ T^*(\underbrace{S^1 \times \cdots \times S^1}_{N \ times})/S^1$$

whose symplectic leaves are the reduced symplectic manifolds

$$P_\mu \ = \ J^{-1}(\mu)/S^1 \subset P$$

We coordinatize P by $\theta_{k,J(k)} \ = \ \theta_k \ - \ \theta_{J(k)}$, $k = 2, \ldots, N$, and μ_j, $j = 1, \ldots N$, where $J(k)$ is the body label of the body connected to body k and $J(k) < k$ via the *previous joint* $(i-1)$ (in Figure 1, $J(5) = 2$, , $J(3) = 2$ and $J(2) = 1$, etc.; also see footnote on page 5).

Topologically,

$$P = \underbrace{S^1 \times \cdots \times S^1}_{(N-1)\ times} \times \mathbb{R}^N.$$

The Poisson structure on P is computed in the standard way: take two functions $F(\theta_{k,J(k)}, k = 1, \ldots, N, \mu_1, \ldots, \mu_N)$ and $H(\theta_{k,J(k)}, k = 2, \ldots, N, \mu_1, \ldots, \mu_N)$. Regard them as functions of $\theta_1, \ldots, \theta_N, \mu_1, \ldots, \mu_N$ by substituting $\theta_{k,J(k)} = \theta_k - \theta_{J(k)}$ and compute the canonical bracket.

We can now state our main theorem on the equations of motion of planar multibody systems connected in the form of a tree in terms of the non-canonical bracket.

Theorem A.1 : The dynamics of a multibody system, evolves over a reduced Poisson space P coordinatized by $\theta_{k,J(k)}$, $k = 2, \ldots, N$ and μ_k, $k = 1, \ldots N$. Topologically P is $\underbrace{S^1 \times \cdots \times S^1}_{N-1\ times} \times \mathbb{R}^N$. The system is Hamiltonian in the Poisson structure of P with the non-canonical bracket given by :

$$\{f, g\} = \sum_{k=2}^{N} \left[\left(\frac{\partial f}{\partial \mu_{J(k)}} - \frac{\partial f}{\partial \mu_k} \right) \frac{\partial g}{\partial \theta_{k,J(k)}} - \left(\frac{\partial g}{\partial \mu_{J(k)}} - \frac{\partial g}{\partial \mu_k} \right) \frac{\partial f}{\partial \theta_{k,J(k)}} \right],$$

where $f, g : \mathbb{R}^{2N-1} \to \mathbb{R}$.

The corresponding dynamics in terms of the bracket are

$$\begin{aligned} \dot{\mu}_k &= \{\mu_k, H\} \qquad k = 1, \cdots, N, \\ \dot{\theta}_{k,J(k)} &= \left\{\theta_{k,J(k)}, H\right\} \qquad k = 2, \cdots, N. \end{aligned}$$

Proof : See Sreenath [23] Chapter 3.

Corollary A.1 : The sum of all the conjugate momentum variables μ_k, $k = 1, \ldots, N$ is equal to the angular momentum of the multibody system, in the center of mass frame.

Proof : See Sreenath [23] Chapter 3.

Planar Two-Body System

The dynamics of a planar two body system with a torque T_{joint} acting at the joint, is given by the following equations :

$$\begin{aligned}
\dot{\mu}_1 &= \frac{\partial H}{\partial \theta_{2,1}} + T_{\text{joint}}, \\
\dot{\mu}_2 &= -\frac{\partial H}{\partial \theta_{2,1}} - T_{\text{joint}}, \\
\dot{\theta}_{2,1} &= \frac{\partial H}{\partial \mu_2} - \frac{\partial H}{\partial \mu_1}.
\end{aligned}$$

The Hamiltonian (kinetic energy) of the planar two-body problem is given by

$$H = \frac{1}{2}\left[\mu_1, \mu_2\right] \mathbf{J}^{-1} \left[\mu_1, \mu_2\right]^T$$

where the $\mathbf{J}$ is a symmetric pseudo-inertia matrix dependent on the relative angle $\theta_{2,1}$ between the bodies. μ_1 and μ_2 are the conjugate momentum variables and are related to the angular velocities ω_1 and ω_2 of the bodies as below:

$$\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} = \mathbf{J} \begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix}.$$

Also,

$$\mu_1 + \mu_2 = \mu_s$$

where μ_s is the angular momentum of the system, a conserved quantity.

Bibliography

[1] Arnold, V. I., *Mathematical Methods of Classical Mechanics*, Springer Verlag, New York, 1978.

[2] Abraham, R., and Marsden, J. E., *Foundations of Mechanics*, Benjamin /Cummings, Reading, Mass., 1978.

[3] Ceasareo, G. , and Nicolo, F., "DYMIR: A code for generating dynamic models of robots", *in preprint*, 1984.

[4] Cox, B. J., *Object Oriented Programming - An Evolutionary Approach*, Addison-Wesley Publishing Co., 1986.

[5] Dillon, S. R., " Computer assisted equation generation in linkage dynamics",*Thesis*, Ohio State University, 1973.

[6] Fletcher, H. J., Rongved, L., and Yu, E. Y., "Dynamics analysis of a two-body gravitationally oriented satellite", *Bell Systems Tech. J.*, Vol 42, pp. 2239-2266, 1963.

[7] Frisch, H. P., "A vector-dyadic development of the equations of motion for N-coupled rigid bodies and point masses",*NASA TND-7767*,Oct., 1977.

[8] Frisch, H. P. "The NBOD2 User's and Programmer's Manual" in *NASA TP 1145* , Feb., 1978.

[9] Grossman, R., Krishnaprasad, P. S., and Marsden, J. E., "Dynamics of two coupled three dimensional rigid bodies", *Dynamical Systems approaches to*

Nonlinear Problems in Systems and Circuits, eds., F.M.A. Salam and M. Levi, SIAM Publ., 1987.

[10] Hooker, W. W., and Margulies, G., "The dynamical attitude equations for a n-Body Satellite",*J. Astronaut. Sci.*, Vol. 12, pp. 123-128, 1965.

[11] Hunt L. R., Su, R., and Meyer, G., "Global transformations of nonlinear systems".*IEEE Transactions on Automatic Control,* Vol. AC-29, pp. 24-31, Jan. 1983.

[12] Isidori, A., *Non-Linear Control Systems : An Introduction*, Lecture Notes in Control and Information Science, Vol 72, Springer-Verlag, New York, 1985.

[13] Khalil, W., *Modelization et commande par calculeteurs du manipulateurs*, Thesis, Universite' des Sciences et Technique du Languedoc, 1976.

[14] Krishnaprasad, P. S., "Lie-Poisson structures, dual-spin spacecraft and asymptotic stability", *Nonlinear Analysis, Theory, Methods and Applications*, Vol 9, N0. 10, pp. 1011-1035, 1985.

[15] Liegeois, A., Khalil, W., Dumas, J. M., and Renauld, M. "Mathematical and computer models of interconnected mechanical systems", *2nd International symposium on theory and practice of robots and manipulators*, pp. 5-17, Sept. 1976.

[16] Likins, P. W., "Analytical Dynamics and Nonrigid Spacecraft Simulation", *NASA Tech. Rept. 32-1593*, July 15, 1974.

[17] Marsden, J. E., and Weinstein, A., "Reduction of symplectic manifolds with symmetry", *Rep. Mathematical Physics*, 5, pp. 121-130, 1974.

[18] Murray, J. J., and Neuman, C. P., " ARM: An algebraic robot dynamic modeling program", IEEE 1984.

[19] Quartararo, R., "Two-body control for rapid attitude maneuvers", *Advances in Astronautical Sciences*, L.A. Morine ed., Vol 42, pp. 397-422, American Astronautical Society, San Diego (AAS 80-203), 1980.

[20] Roberson, R. E., and Wittenburg, J., "A dynamical formalism for an arbitrary number of interconnected rigid bodies with reference to the problem of satellite attitude control", *Proc. of the 3rd Intl. Congress of Automatic Control, London,1966*, pp. 46D.1-46D.8. Butterworth and Co., Ltd, London, England, 1967.

[21] Rosenthal, R. E., Sherman, M. A., "High Performance Multibody Simulations via Symbolic Equation Manipulation and Kane's Method", *J. of Astronautical Sciences*, Vol 34., No. 3, pp. 223-239, July-Sept. 1986.

[22] Schwertassek, R., and Roberson, R. E., "A State-Space Dynamical Representation for Multibody Mechanical Systems, Part I: Systems with Tree Configuration", *Acta Mechanica*, 50, pp. 141-161, 1984.

[23] Sreenath, N., *Modeling and Control of Multibody Systems*, Ph.D. Thesis, Univ. of Maryland, 1987.

[24] Sreenath, N., and Krishnaprasad, P. S. "DYNAMAN: A Tool for manipulator design and analysis", *IEEE Intl. Conf. on Robotics and Automation*, San Fransisco, Calif., Vol 2 , p836-842, April 7-11, 1986.

[25] Sreenath, N., Oh, Y. G., Krishnaprasad, P.S., and Marsden, J. E., "Dynamics of coupled planar rigid bodies Part I : reduction, equilibria and stability", J. of Dynamics and Control of Systems (to appear) 1988.

[26] Stefik, M., and Bobrow, D. G., "Object-Oriented Programming : themes and variations", *The AI magazine*, pp. 40-62, 1986.

[27] Sturges, R., " Teleoperators arm design program", *Report 3.2746* MIT Draper Lab. Cambridge Mass., 1973.

[28] Wertz, J., *Spacecraft Attitude Determination and Control*, Dordrecht, D. Reidel, 1978.

[29] Wittenburg , J. M., *Dynamics of Systems of Rigid Bodies*, B.G. Tuebner and Co., Stuttgart, West Germany, 1977.

[30] Wittenburg , J. M., "Dynamics of Multibody Systems", *Theoretical and Applied Mechanics*, Proc. XV IUTAM Conf. Toronto, pp. 197-207, 1980.

[31] Wittenburg, J. M., "MESA-VERDE: A symbolic program for nonlinear articulated-rigid-body dynamics",*ASME Design Engg. Division Conf.*, Cincinatti, Ohio, Sept., pp. 10-13, 1985.

[32] Velman, J. R., "Simulation results for a dual-spin spacecraft",*Proc. of the symposium on Attitude Stabilization and Control of Dual-Spin Spacecraft, El Segundo, Calif., 1967,* , Rept. SAMSO-TR-68-191, 1967.