



Werner Schiehlen (Editor)

Multibody Systems Handbook

With 165 Figures

Springer-Verlag Berlin Heidelberg New York
London Paris Tokyo Hong Kong

Dr.-Ing. Werner Schiehlen

Full Professor of Mechanics

Institute B of Mechanics

University of Stuttgart

Pfaffenwaldring 9

7000 Stuttgart 80

FRG

ISBN 978-3-642-50997-1
DOI 10.1007/978-3-642-50995-7

ISBN 978-3-642-50995-7 (eBook)

Library of Congress Cataloging-in-Publication Data
Multibody systems handbook/Werner Schiehlen (editor).

ISBN 978-3-642-50997-1

1. Dynamics--Computer programs. I. Schiehlen, W. O. (Werner O.)

QA845.M86 1990

531'.11'0113--dc20 8926255

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, broadcasting, reproduction on microfilms or in other ways, and storage in data banks. Duplication of this publication or parts thereof is only permitted under the provisions of the German Copyright Law of September 9, 1965, in its version of June 24, 1985, and a copyright fee must always be paid. Violations fall under the prosecution act of the German Copyright Law.

© Springer-Verlag Berlin Heidelberg 1990

Softcover reprint of the hardcover 1st edition 1990

The use of registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Offsetprinting: Color-Druck Dorfi GmbH, Berlin; Bookbinding: Lüderitz & Bauer, Berlin
2161/3020 543210

Preface

Dynamics of multibody systems is of great importance in the fields of robotics, biomechanics, spacecraft control, road and rail vehicle design, and dynamics of machinery. Many research problems have been solved and a reasonable number of multibody formalisms has been developed in the last two decades. They were discussed especially during a IUTAM Symposium in 1977 in Munich and a IUTAM/IFTOMM Symposium in 1985 in Udine. A considerable number of computer codes based on these formalisms is now available. With the Multibody Systems Handbook it is intended to collect software systems for multibody system dynamics which are well established and have found acceptance in the users community. The Handbook will aid the reader in selecting the software system which is most appropriate to his needs.

Altogether 17 research groups contributed with their programs or software systems to the Handbook. A compact summary of important capabilities of these software systems is presented in tabular form.

It has to be mentioned that during the preparation of the Handbook all contributors agreed to deal with two typical test examples, a planar mechanism and a spatial robot. Thus, it is very easy to compare the results and to identify more clearly the advantages of one or the other formalism.

The Handbook is a really international collection of programs and software. Therefore, the editor is indebted to all contributors for the excellent cooperation overcoming all borders.

The editorial work of the Handbook was supported by the

Institute B of Mechanics, University of Stuttgart. In particular, I would like to thank Dipl.-Ing. Günter Leister for his valuable help and Mrs. Regine Weber for all the secretarial work. Furthermore, thanks are due to the Springer-Verlag for the encouraging and efficient cooperation.

Stuttgart, September 1989

Werner Schiehlen

Table of Contents

Introduction	1
Overview	3
General remarks	3
Tables of information	6
Test Examples	10
Descriptions of Codes	20
NUBEMM - Theory and application of the MBS program ..	21
SYM - Program package for computer-aided generation of optimal symbolic models of robot manipulators	37
CAMS - A graphical interactive system for computer simulation and design of multibody systems	61
AUTOLEV - A new approach to multibody dynamics	81
UCIN-DYNOCOMBS - Software for the dynamic analysis of constrained multibody systems	103
SPACAR - Computer program for dynamic analysis of flexible spatial mechanisms and manipulators	123
NBOD & DISCOS - Dynamic interaction simulation of controls and structure	145
DADS - Dynamic Analysis and Design System	161
NEWEUL - Software for the generation of symbolical equations of motion	181
MEDYNA - An interactive analysis and design program for geometrically linear and flexible multibody systems	203
AUTODYN & ROBOTRAN - Computer programmes	225

SIMPACK - A computer program for simulations of large-motion multibody systems	265
COMPAMM - A simple and efficient code for kinematic and dynamic numerical simulation of 3-D multibody systems with realistic graphics	285
DYMAC & DYSRIM - Programs for the dynamic analysis and simulation of planar mechanisms & multibody systems	305
MESA VERDE - A general-purpose program package for symbolical dynamics simulations of multibody systems	341
ADAMS - Multibody system analysis software	361
PLEXUS - Software for the numerical analysis of the dynamical behavior of rigid and flexible mechanisms	403
Contributors and Distributors	427
Index	431

Introduction

In the design of mechanical systems, such as vehicles, machinery or robots, it is now customary to use computer simulations to observe the dynamic response of the system being designed. This form of computer-aided design drastically reduces the need to construct and test prototypes. However, the equations of motion of the system must first be obtained.

Multibody systems are characterized by rigid and/or flexible bodies with inertia and springs, dampers and servomotors without inertia, interconnected by rigid bearings or supports. Furthermore, friction and contact forces may be included. The dynamics of multibody systems are primarily characterized by their linear and/or nonlinear equations of motions. In addition the equations of reaction are needed for strength considerations.

Various computational methods and computer formalisms for mathematical modeling and simulation of complex multibody systems have been developed over the past two decades. Since a number of these software systems is well established and has found acceptance with users we feel that it is time to collect them in a handbook for bridging the gap between system designers and a possibly wide user community.

As is true of handbooks in general, the authors can not treat their topics exhaustively. Each writer is in favour of his own view of the subject. Therefore, the overview in chapter 2 summarizes the most important features of the codes included in the Handbook. Of course, the reader has to proceed to the description chapter to get enough information for selecting the software which is most appropriate to his problems.

In particular, the test examples presented in chapter 3, offer the unique opportunity to compare not only formalisms and software systems in theory but also in practice. Thus, the test examples may be used as bench marks for formalisms and codes being improved or developed in the future, too.

Typically, the description given in chapter 4 provides many details of software systems as they appear to the user and should motivate the application of these systems in problem solving. Of course, the description will also provide a basis for users to compare different software and to give advice to select the software which fits his needs most appropriate. The contributions do not concentrate on details of the specific programming technique. They will include, however, a description of the various fields of application. Up-to-date information is given for scientists working in the field of multibody system dynamics.

For more detailed information the contributors should be contacted a list of whom is presented in chapter 5. Further, the distributors or marketing organizations, respectively, are given, too.

Overview

The overview summarizes the most important features of the codes or software systems, respectively, in writing and by tables. Altogether 20 codes are considered.

The descriptions of the formalisms and codes given in chapter 4 are to a certain extent inhomogeneous. Therefore, in an overview, the typical features of the codes are summarized in tabular form. This enables the reader to get quick and comparable information. The information is compiled from a standardized software description form prepared by the contributors to the Handbook.

For the overview the same order is used as in the list of contributors, chapter 5, according to the date of the receipt of the contributions by the editorial office. This order does not express any rating.

Topology of Multibody Systems

Chain and tree structured systems can be handled by all codes, and quite a number deals with closed kinematical loops. All but one codes are designed for planar and spatial multibody systems.

Elements Included in Software

Particles, rigid bodies, joints representing holonomic constraint, and applied forces without friction generated by passive elements like springs and dampers are found in all codes. Further, many codes consider position control or rheonomic constraints, respectively, and some allow nonholonomic constraints, too. Applied forces including friction, force control by active elements, and subsystems are partially available features.

Coordinates Used for Software

All kinds of coordinates are used in the codes: inertial, relative and mixed coordinates. Further, a moving reference frame may be offered. A minimal number of generalized coordinates corresponding to the number of degrees of freedom is widely applied. But a regular number of coordinates corresponding to the number of bodies only and a maximal number of coordinates corresponding to the number of bodies and constraints are also found.

Computation Methods Applied in Software

Most codes make use of numerical computation, some prefer symbolical computation or a combination of numerical and symbolical computation. Nevertheless, the symbolical formula manipulators do not require special symbolic languages, the codes are all written in standard computer languages.

Dynamical Methods Used in Software

The methods used include Lagrange's equation of the first kind, Newton-Euler-equations, D'Alembert's principle (principle of virtual work) and Jourdain's principle (principle of virtual

power). In the western hemisphere Jourdain's principle is also referred to as Kane's equation.

Equations Generated by Software

A broad variety of equations is found: nonlinear equations of motion (differential equations), nonlinear equations of reaction (algebraical equations), mixed nonlinear equations of constraints and motion (differential-algebraical equations), linear equations of motion, partially linear equations of motion and mixed linear equations of constraints and motion (differential-algebraical equations).

Simulation Methods Used by Software

Usually, the numerical integration requires methods for nonstiff differential equations only, but some codes use methods for stiff differential equations and differential-algebraic equations solver.

Signal Analysis Used by Software

The signal analysis is typically not included in the multibody systems codes.

Preprocessing

The alphanumerical input includes the batch mode and the interactive mode via an editor. Graphical input is not really developed.

Postprocessing

The standard postprocessing feature is the time history plot. Frequency response and root locus plot are sometimes offered. Single pictures and moving pictures are available with some codes.

Software Engineering

The most important computer language is FORTRAN, some codes are available in PASCAL and C, one code uses TURBO PASCAL. The operating systems include VMS, MS-DOS, UNIX, etc.. The computers used for the codes are personal computers, workstations and mainframe computers.

Distribution of Software

Most of the codes are commercially available and the maximum number of installations reported for one code is 120. The marketing organizations are listed in chapter 5 together with the contributors.

Software Applications

The applications are typical for the multibody systems method: Rail and road vehicles, airplanes and satellites, manipulators and robots, machines and mechanisms, human body and sports.

From the information given by the authors in a standardized software description form, three tables were compiled printed at the following pages. The tables are useful for quick reference.

	NUBEMM	SYM	CAMS	AUTOLEV	DYNOCOMBS	SPACAR	DISCOS	NBOD	DADS	NEWEUL	MEDYNA	AUTODYN	ROBOTTRAN	SIMPACK	COMFAMM	DYMAC	DYSPLAM	MESA VERDE	ADAMS	PLEXUS
TOPOLOGY																				
Tree Structure	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
Closed Loops	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
Plane motion	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
Spatial motion	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
ELEMENTS																				
Rigid bodies	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
Elastic bodies								●	●	●										
Joints	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
Position control	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
Nonholonomic constr.			●	●																
Applied forces	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
Friction forces								●	●	●										
Force control	●			●																
Subsystems		●						●	●	●							●			
COORDINATES																				
Inertial coordinates	●	●	●	●	●			●	●	●	●					●	●	●	●	●
Relative coordinates	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
Mixed coordinates	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
Moving reference	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
Minimal number	●		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
Regular number			●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
Maximal number			●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
COMPUTATION																				
Numerical	●			●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
Symbolical		●																		
Num./Symb.	●		●	●	●	●	●	●	●	●	●		●		●	●	●	●	●	●

Test Examples

This chapter presents two mechanical systems which are appropriate as test examples for multibody systems. The plane motion of the mechanism with one degree of freedom was firstly used by Giles [1] and Manning [2] for testing. Later, Homolka [3] and Leister [4] analysed the mechanism using the programs NEWEUL and ADAMS, respectively. The topology of the mechanism is characterized by closed kinematic loops. The second example is the spatial motion of a robot with five degrees of freedom. The topology of such a mechanical system shows chain configuration.

Since every multibody formalism presents the time history of system variables in different format, the results of a simulation of motion are often not comparable. In contrary, the visualisation of a simulation by computer graphics animation is a general mean for the system analyst to identify the results easily. For this purpose only an animation of both the examples will be presented. A detailed collection of corresponding simulations is available in Ref. [5].

Description of the Mechanism

The mechanism shown in Fig. 1 consists of 7 bodies interconnected by joints without friction. The cartesian coordinates of the points A, B, C, and all the other parameters were adopted from Giles [1].

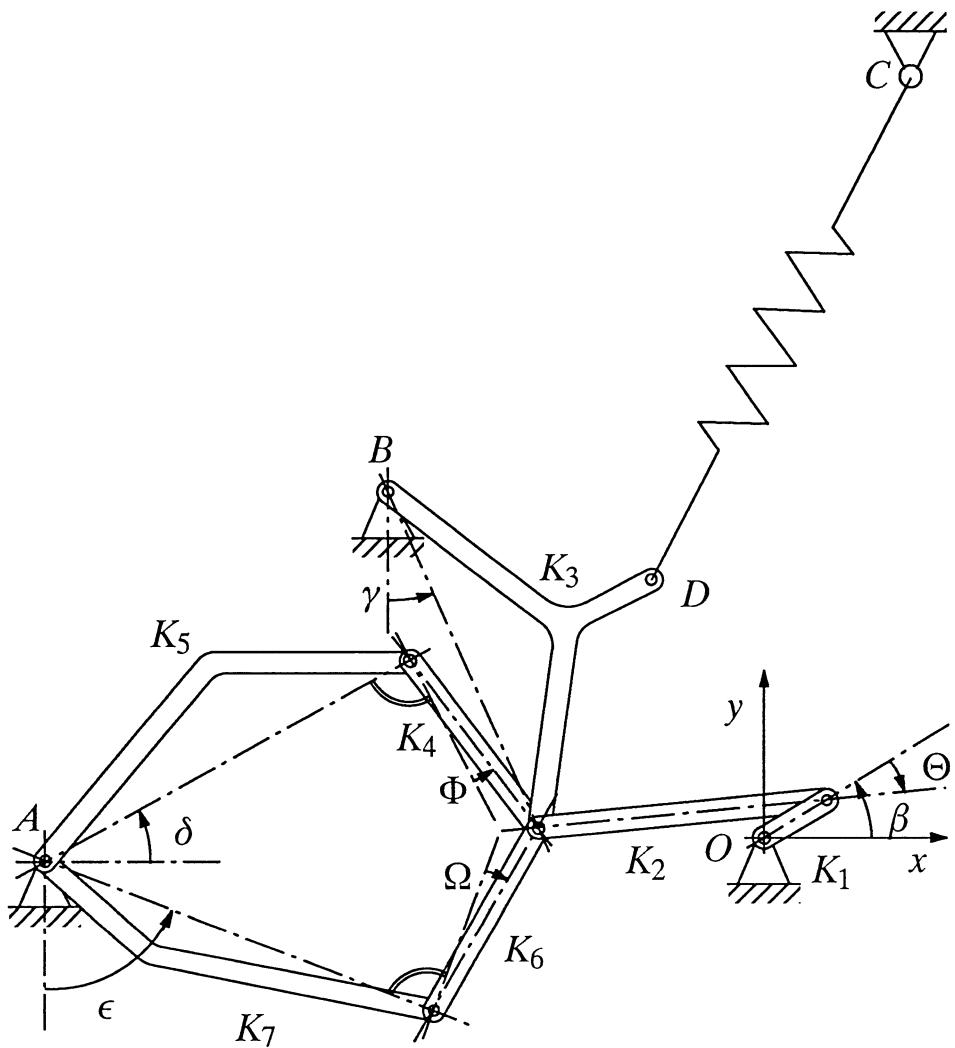


Fig. 1. Seven body mechanism

The coordinates of the points A, B, C read as

$$x_A = -0,06934 \text{ m}, \quad y_A = -0,00227 \text{ m},$$

$$x_B = -0,03635 \text{ m}, \quad y_B = 0,03273 \text{ m},$$

$$x_C = 0,01400 \text{ m}, \quad y_C = 0,07200 \text{ m}.$$

The geometrical and inertia parameters of all bodies are given in Fig. 2 and Table 1. Further, for the spring yields:

$$\text{spring coefficient } c_0 = 4530 \text{ N/m},$$

$$\text{unstretched length } l_0 = 0,07785 \text{ m}.$$

The body-fixed frames x_i, y_i , $i=1(1)7$, are located in the center of mass of each body. The mechanism is driven by a motor located in point 0. The constant drive torque is given by $M_{OM} = 0,033 \text{ Nm}$.

The motion of the mechanism is to be simulated for one revolution. At time zero all the velocities of the mechanism are vanishing and the coordinates satisfying the constraints are given as

$$\beta_0 = -0,0620 \text{ rad}, \quad \theta_0 = 0,0000 \text{ rad},$$

$$\gamma_0 = 0,4552 \text{ rad}, \quad \Phi_0 = 0,2227 \text{ rad},$$

$$\delta_0 = 0,4873 \text{ rad}, \quad \Omega_0 = 0,2227 \text{ rad}.$$

$$\epsilon_0 = 1,2305 \text{ rad},$$

Each of these coordinates can be chosen as generalized coordinate. An adequate choice is the angle β of the first body.

Fig. 3 presents the animation of motion of the mechanism for an intervall of 15 ms with constant steps of 1 ms. Within the intervall 0 to 15 ms link 1 completes just one revolution.

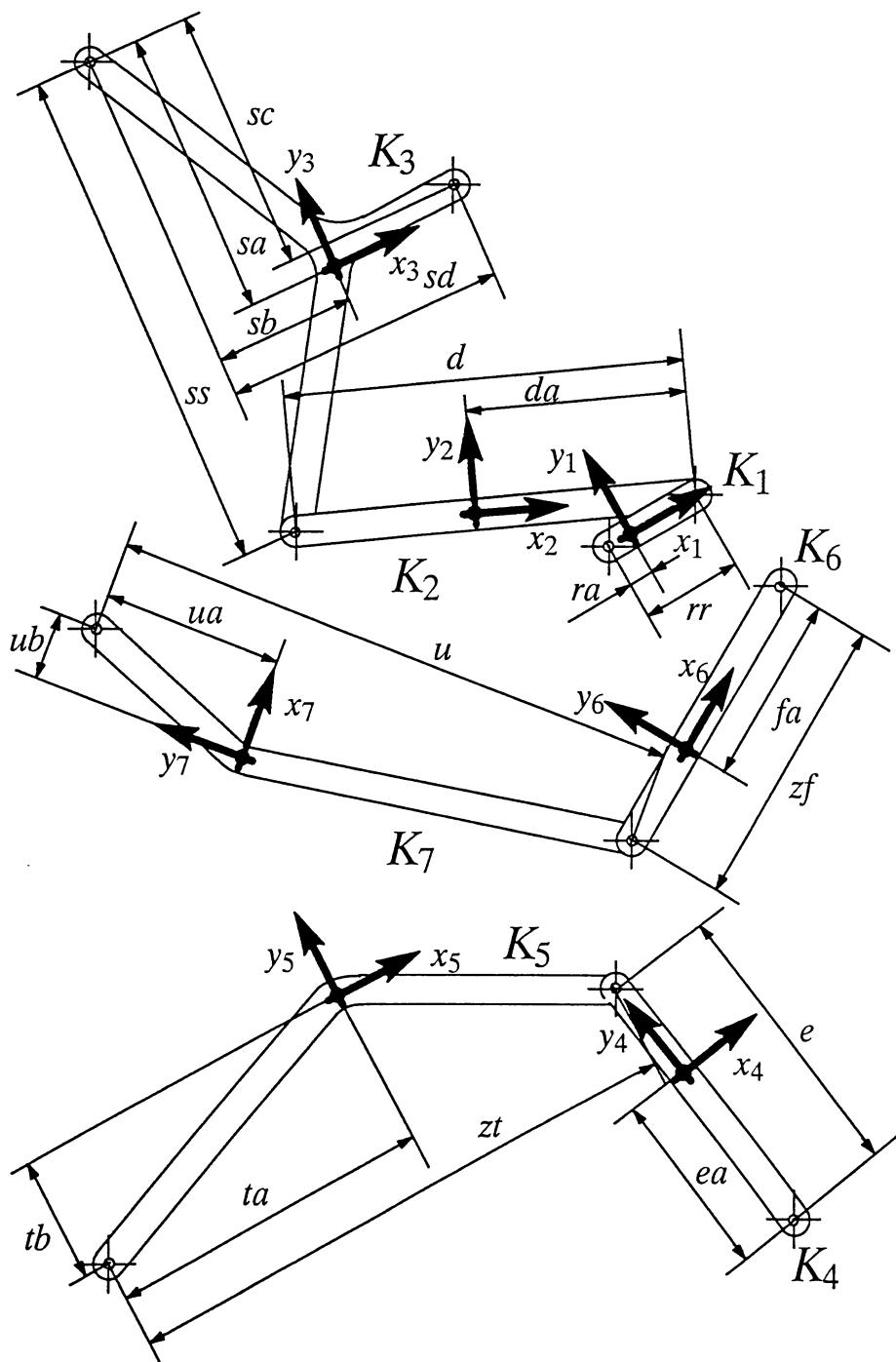


Fig. 2 Geometrical design

Table 1 Geometrical and inertia parameters

d = 0,028	m	sb = 0,01043	m
da = 0,0115	m	sc = 0,018	m
e = 0,02	m	sd = 0,02	m
ea = 0,01421	m	zt = 0,04	m
zf = 0,02	m	ta = 0,02308	m
fa = 0,01421	m	tb = 0,00916	m
rr = 0,007	m	u = 0,04	m
ra = 0,00092	m	ua = 0,01228	m
ss = 0,035	m	ub = 0,00449	m
sa = 0,01874	m		

Link	Mass M1 to M7 [kg]	Rotational inertia about centers of mass	
		I1 to I7 [kg·m ²]	
1	0,04325	2,194 · 10 ⁻⁶	
2	0,00365	4,410 · 10 ⁻⁷	
3	0,02373	5,255 · 10 ⁻⁶	
4	0,00706	5,667 · 10 ⁻⁷	
5	0,07050	1,169 · 10 ⁻⁵	
6	0,00706	5,667 · 10 ⁻⁷	
7	0,05498	1,912 · 10 ⁻⁵	

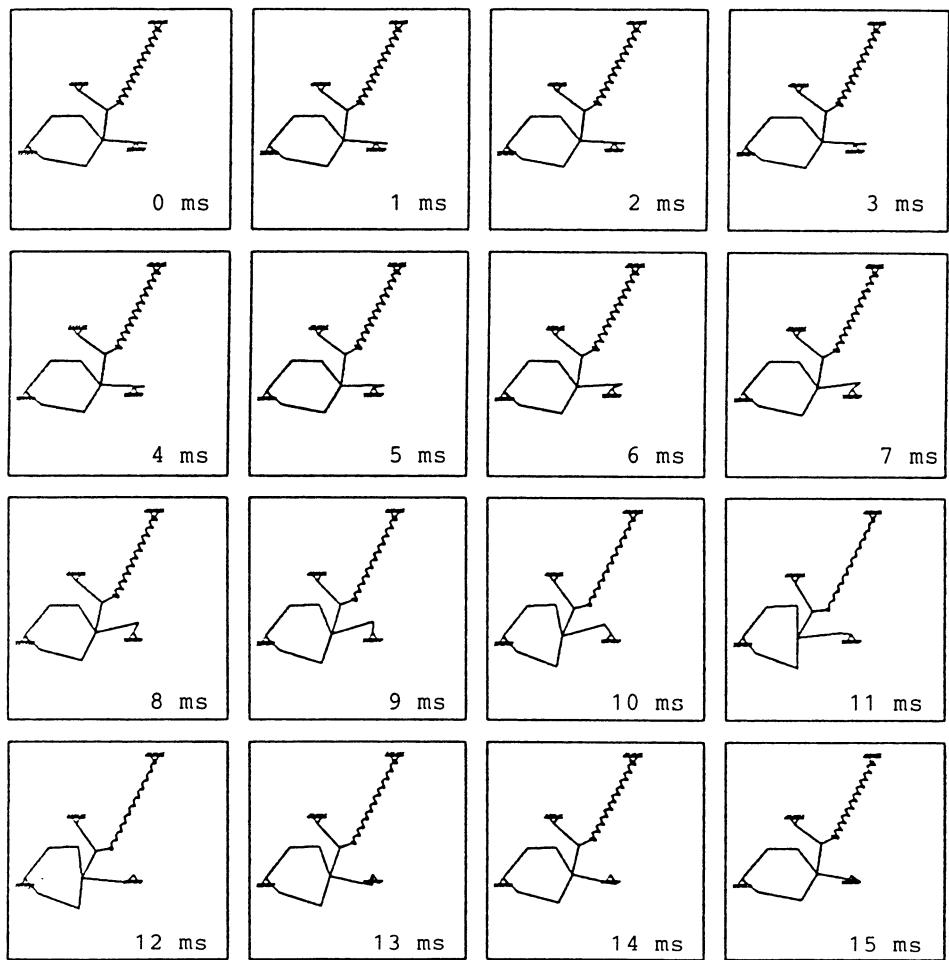


Fig. 3 Animation of motion

Description of the Robot

The robot shown in Fig. 4 consists of 3 bodies interconnected by two rotational-translational joints and one rotational joint. The position of the robot hand can be described by the generalized coordinates Z_1 , $GA1$, Y_2 , $BE2$ and $AL3$.

The inertia parameters are given in Table 2. The geometrical parameters C and L read as

$$C = 0,05 \text{ m} ,$$

and

$$L = 0,50 \text{ m} .$$

The body-fixed frames are located in the center of mass, C_i , of each body. For each joint, the force and torque of the drive actuator has to be defined.

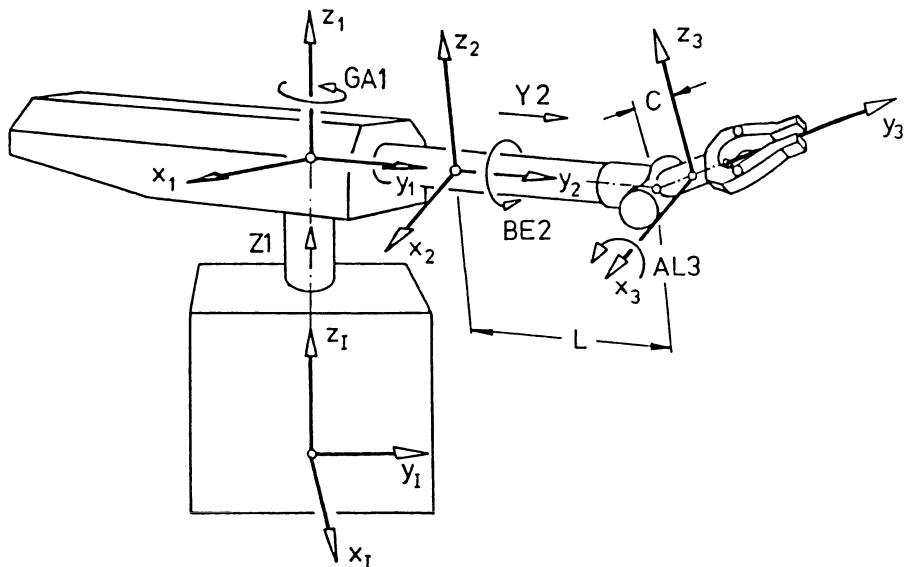


Fig. 4 Robot with location of body-fixed frames

Table 2 Inertia parameters

	Body		
	1	2	3
Mass [kg]	250	150	100
Rotational inertia about body-fixed frames [kg·m ²]	I _x (90) I _y (10) I _z 90	13 0,75 13	4 1 4,3

The motion of the robot hand is to be simulated for a straight line between an initial point and a defined end point. The movement is performed with a trapezoidal velocity profile, i.e., the robot hand starts with a constant acceleration, then continues with constant velocity, and finally it moves with a constant deceleration. At its initial and final position, the velocity of each body is vanishing. The initial position is given by the generalized coordinates as

$$\begin{aligned} z_{10} &= 2,25 \text{ m}, \\ g_{a10} &= -0,5236 \text{ rad}, \\ y_{20} &= 0,75 \text{ m}, \\ b_{e20} &= 0, \\ a_{l30} &= 0. \end{aligned}$$

The drive functions are listed in Table 3.

An animation is presented in Fig. 5. Obviously, the joint of the robot hand is moving on the prescribed path.

Table 3 Drive functions

Time of simulation t [s]	Drive function [N] , [Nm]
0,0 to 0,5	$F1Z = 6348$ $F2Y = 36 \cdot t + 986$ $L1Z = 673 \cdot t - 508$ $L3X = 63,5$
0,5 to 1,5	$F1Z = 4905$ $F2Y = - 2$ $L1Z = 148 \cdot \exp(-5,5 \cdot (t-0,5)) + 8$ $L3X = 49,05$
1,5 to 2,0	$F1Z = 3462$ $F2Y = - 1019$ $L1Z = 240$ $L3X = 34,6$

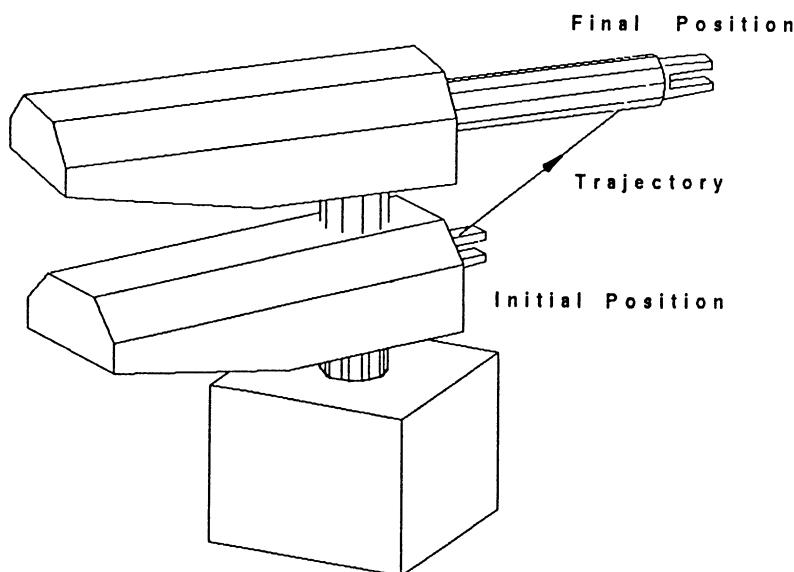


Fig. 5 Animation of motion

References

- [1] Giles, D.R.A.: A comparision of three problem-oriented simulation programs for dynamic mechanical systems. A thesis presented to the University of Waterloo. Waterloo, Ontario, 1978.
- [2] Manning, D.W.: A computer technique for simulating dynamic multibody systems based on dynamic formalism. A thesis presented to the University of Waterloo. Waterloo, Ontario, 1981.
- [3] Homolka, S.: Untersuchung eines ebenen Gelenkmechanismus mit dem Programmsystem NEWEUL. Studienarbeit STUD - 41. Stuttgart: Institut B für Mechanik, 1987.
- [4] Leister, G.: Untersuchung eines ebenen Gelenkmechanismus mit dem Programmsystem ADAMS. Zwischenbericht ZB - 36. Stuttgart: Institut B für Mechanik, 1988.
- [5] Daberkow, A.; Eismann, W.; Schiehlen, W.: Test examples for multibody systems. Stuttgart: Universität Stuttgart, Institut B für Mechanik, Institutsbericht IB - 13, 1989.

Descriptions of Codes

In this chapter altogether 17 research groups present a detailed description of multibody system formalisms and the corresponding computer codes.

NUBEMM	E. Pankiewicz, FRG
SYM	M. Vukobratovic; N. Kircanski, A. Timcenko; M. Kircanski, YUGOSLAVIA
CAMS	L. Lilov; B. Bekjarov; M. Lorer, BULGARIA
AUTOLEV	D.A. Levinson; T.R. Kane, USA
DYNOCOMBS	R.L. Huston; T.P. King; J.W. Kamman, USA
SPACAR	J.B. Jonker; J.P. Meijaard, THE NETHERLANDS
NBOD & DISCOS	H. Frisch, USA
DADS	R.L. Smith; E.J. Haug, USA
NEWEUL	E. Kreuzer; W. Schiehlen, FRG
MEDYNA	O. Wallrapp; C. Führer, FRG
AUTODYN & ROBOTRAN	P. Maes; J.C. Samin; P.Y. Willems, BELGIUM
SIMPACK	W. Rulka, FRG
COMPAMM	J.M. Jimenez; A. Avello; A. Garcia- Alonso; J. Garcia de Jalon, SPAIN
DYMAC & DYSPAM	B. Paul; R. Schaffa, USA
MESA VERDE	J. Wittenburg; U. Wolz; A. Schmidt, FRG
ADAMS	R.R. Ryan, USA
PLEXUS	A. Barraco, B. Cuny, et.al., FRANCE

NUBEMM - Theory and Application of the MBS Program

by E. Pankiewicz, Federal Republic of Germany

Abstract

In this paper the theoretical background as well as the application of the program NUBEMM is presented. Handling of the program will be demonstrated by using an example.

1. Introduction

In order to enable the vehicle development engineering branch to report on vehicle handling as well as on the forces arising during operation when still in the design phase, digital time simulation is utilized.

Such simulation, however, requires a mechanical substitute that describes the real vehicle and its behavior in an idealized and task-related manner.

The basic substitute of such a physical system is illustrated in Fig. 1. It is composed of

rigid bodies and/or mass points,
massless springs and dampers,
servomotors,
rigid massless constraining elements (bearings, guides, joints).

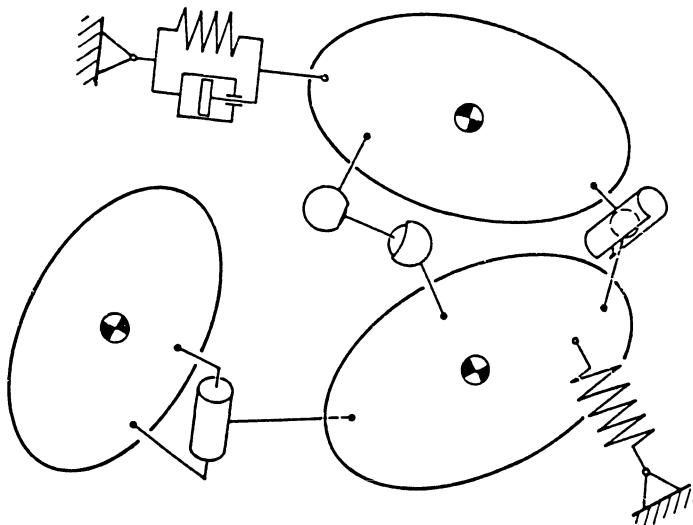


Fig. 1 Multibody system

The constraining elements shall be holonomic, without elasticity and friction; they constrain the motion of the bodies and create constraint forces and torques in the joints.

Elastic components that generate impressed forces can be modelled with the spring and damper elements.

The servomotors are controlled and/or regulated active elements that can act on the joints as well as between the bodies.

2. NUBEMM description of a "linear" MBS formalism [1]

The NUBEMM program is primarily designed and applied for dynamic simulation and the study of vehicle handling.

In order to also depict very complex vehicle models "directly from the design", however, a compromise was made between the costly complete nonlinear calculations and the partially linearized calculation in NUBEMM.

This simplification of the partial linearization is absolutely justified for vehicle dynamic simulation because while the vehicle body carries out large (non-linear) motions, the motions of the wheels as well as of the axle systems are small relative to the vehicle body and can be linearized.

This aspect has also been confirmed by comparison between simulation computations and measurements [1], [2].

Moreover, the equations of motion of NUBEMM are to be coupled each vehicle subsystem via forces and torques (see Fig. 2), i.e. an open symbolic interface with the following variables has been generated here:

the generalized coordinates of position, velocity and acceleration,
the spring and damper forces,
the external forces and torques, e.g. tire forces,
symbols for path excitation and forces from servomotors,
constraint forces and torques (for modelling of frictional forces).

All these symbolic variables are available to the user. In the case of any geometrical, mass and inertia model change, on the other hand, the equations of motion must be generated anew.

The derivation of the partially linearized equations of motion now follows.

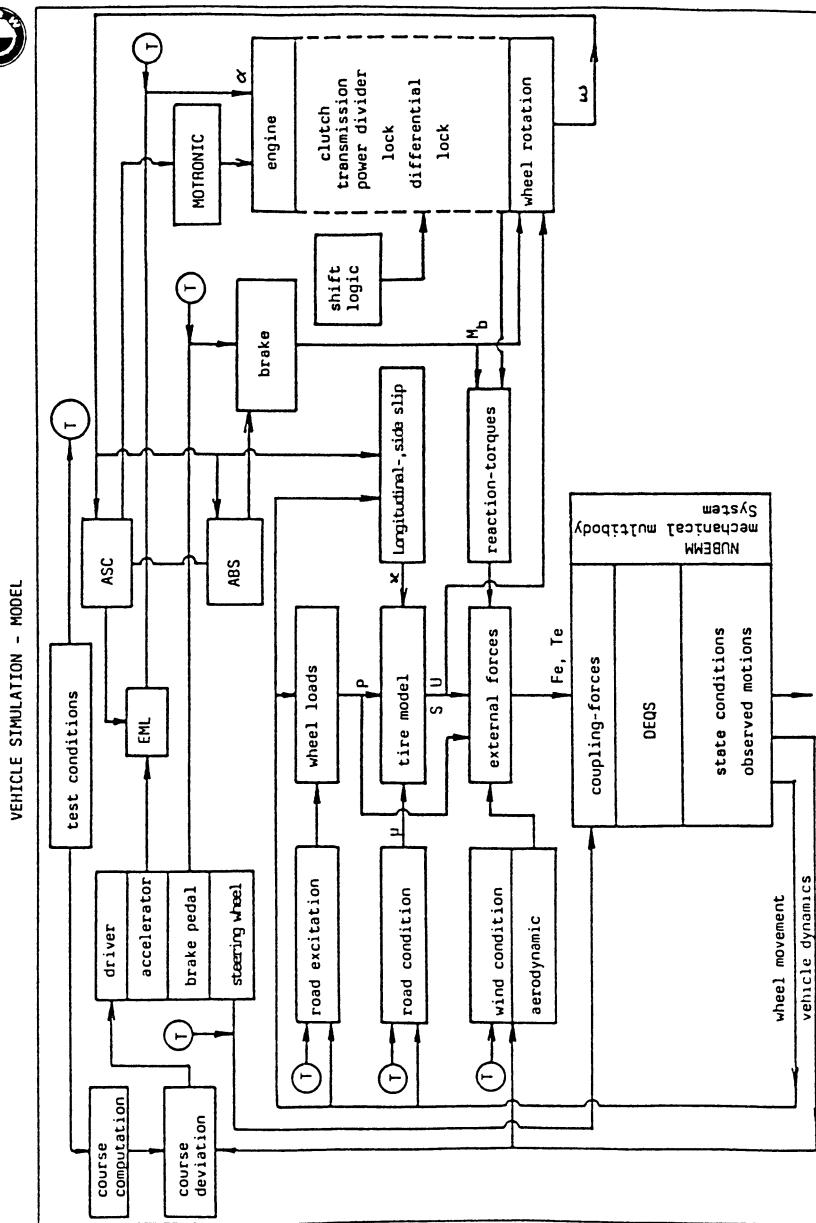


Fig. 2 Vehicle MBS model with subsystems

Kinematic relationships in an accelerated moving reference frame

The kinematic equations of a body of the MBS are shown relative to the moving reference frame (see Fig. 3).

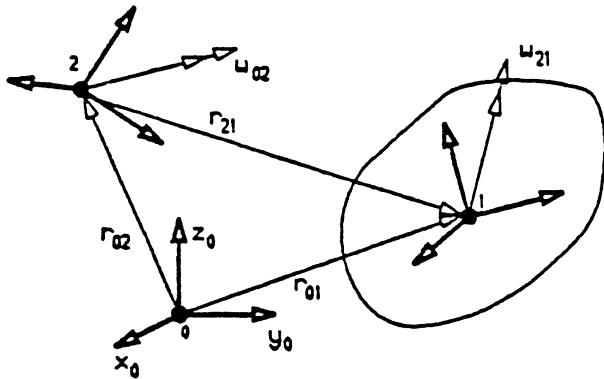


Fig. 3 MBS body in a moving reference frame

In order to linearize the equations of motion later, the positional vectors and the rotational matrices of all bodies are split up as follows:

$$\mathbf{r}_{01} = \mathbf{r}_{02} + [{}^0\mathbf{S}] \mathbf{r}_{21,2}, \quad (1.a)$$

$$[{}^1\mathbf{S}] = [{}^0\mathbf{S}] [{}^2\mathbf{S}]. \quad (1.b)$$

Differentiation of equation (1), leads to translational and rotational velocities relative to the inertial frame. These equations are transformed into coordinates of the moving reference frame with the help of rotational matrix $[{}^2\mathbf{S}]$

$$\begin{bmatrix} {}^6\mathbf{x}_1 \\ {}^6\mathbf{v}_{01,2} \\ {}^6\omega_{01,2} \end{bmatrix} = \begin{bmatrix} {}^6\mathbf{x}_1 \\ {}^6\mathbf{v}_{21,2} \\ {}^6\omega_{21,2} \end{bmatrix} + \begin{bmatrix} \mathbf{E} & -[\tilde{\mathbf{r}}_{21,2}] \\ 0 & \mathbf{E} \end{bmatrix} \begin{bmatrix} {}^6\mathbf{x}_2 \\ {}^6\mathbf{v}_{02,2} \\ {}^6\omega_{02,2} \end{bmatrix}. \quad (2)$$

In this case, $({})^*$ means differentiation in the inertial frame and $({})'$ differentiation in the moving reference frame.

Analogously, the acceleration equations are calculated in coordinates of the moving reference frame,

$$\begin{bmatrix} \overset{*}{r}_{01,2} \\ \overset{*}{\omega}_{01,2} \end{bmatrix} = \begin{bmatrix} \overset{*}{r}_{21,2} \\ \overset{*}{\omega}_{21,2} \end{bmatrix} + \begin{bmatrix} E & -[\overset{*}{r}_{21,2}] \\ 0 & E \end{bmatrix} \begin{bmatrix} \overset{*}{r}_{02,2} \\ \overset{*}{\omega}_{02,2} \end{bmatrix} + \begin{bmatrix} [\overset{*}{\omega}_{02,2}]^2 r_{21,2} + 2[\overset{*}{\omega}_{02,2}] \dot{r}_{21,2} \\ [\overset{*}{\omega}_{02,2}] \omega_{21,2} \end{bmatrix}. \quad (3)$$

Equations (2) and (3) can be shown in compact written form with corresponding distribution matrices as follows:

$$\overset{*}{p}_i = \dot{p}_i + [Q(p_i)] \overset{*}{z}, \quad (4a)$$

$$\overset{*}{\ddot{p}}_i = \ddot{p}_i + [Q(p_i)] \overset{*}{\ddot{z}} + a(p_i, \dot{p}_i, z), \quad (4b)$$

$$i = 1(1)n. \quad (4c)$$

Ideal constraining elements; derivation of the constraint equations

If the individual bodies are connected to one another via joints, the relative motions of the bodies are restricted. In a constraining link as many generalized constraint forces occur as there are restrictions of motion. These constraint forces appear in the equations of motion as unknown and must be eliminated since they cannot be replaced by any law of forces.

Constraining coordinate system "3" is inserted in such a way that it describes the free and locked motions of the joint. This coordinate system is fixed to body "1".

If one examines any connecting element between two bodies (Fig. 4), the translational and rotational "kinematic loop" complies with the following equations,

$$r_2 + [{}^0S]u_{2,2} - r_1 - [{}^1S]u_{1,1} = [{}^0S][{}^1S]r_{3,3} \equiv r_3, \quad (5a)$$

$$[{}^2S][{}^1S][{}^0S] = E. \quad (5b)$$

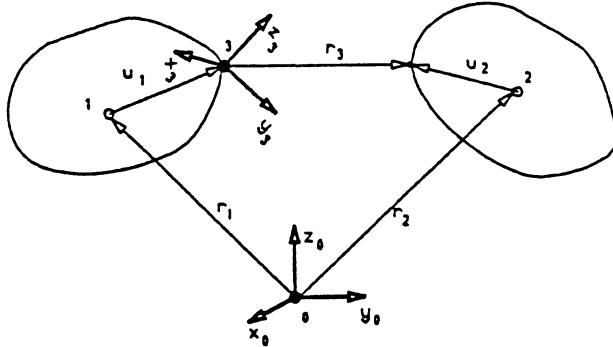


Fig. 4 Kinematics of a rigid link joint

The differentiation of equation (5) leads to

$$\dot{r}_2 + [\tilde{\omega}_2][^0S]u_{2,2} - \dot{r}_1 - [\tilde{\omega}_1][^1S]u_{1,1} = [\tilde{\omega}_1][^0S][^1S]r_{3,3} + [^0S][^1S]\dot{r}_{3,3}, \quad (6a)$$

$$\omega_2 - \omega_1 = \omega_{12} = [^0S][^1S]u_{3,3}. \quad (6b)$$

Equation (6) can now be transformed with the help of the projection matrices of the translation $[^3S_{ot}] [^1S]$ and the rotation $[^3S_{or}] [^0S]$ in the direction of the locked joint modes.

Thereby the $nx3$ matrix $[^3S_{ot}]$ represents the locked translational movements and the $norx3$ matrix $[^3S_{or}]$ represents the locked rotational modes.

The results of this multiplication are summarized in matrix form below:

$$\begin{bmatrix} \dot{r}_{03,3} \\ \omega_{03,3} \end{bmatrix} = \begin{bmatrix} [^3S_{ot}] [^1S] & -[^3S_{ot}] [^1S] [\tilde{\omega}_2] \\ [0] & [^3S_{or}] [^0S] \end{bmatrix} \begin{bmatrix} \dot{r}_2 \\ \omega_2 \end{bmatrix} - \begin{bmatrix} [^3S_{ot}] [^1S] & -[^3S_{ot}] [^1S] [\tilde{\omega}_1 + \tilde{r}_3] \\ [0] & [^3S_{or}] [^0S] \end{bmatrix} \begin{bmatrix} \dot{r}_1 \\ \omega_1 \end{bmatrix}. \quad (7)$$

The following abbreviations are introduced here:

$$[\tilde{\omega}_1] = [^0S][\tilde{u}_{1,1}][^0S] \quad 3x3 - \text{skew-symmetric matrix},$$

$$[\tilde{\omega}_2] = [^0S][\tilde{u}_{2,2}][^0S] \quad 3x3 - \text{skew-symmetric matrix}.$$

In (7) there are $nb=not + nor$ equation which correspond to the number of constraint forces and torques of the constraining element.

Reduction to generalized coordinates

According to equation (7), one can combine all nzb holonomic constraint conditions into a linear equation system in matrix form:

$$[C_b(p)]\dot{p} = 0. \quad (8)$$

If $f=6n-nzb$ (n number of bodies) coordinates of \dot{p} are used as generalized coordinates of the MBS, equation (8) can be split as follows:

$$[C_{ba}]\dot{p}_a + [C_{bu}]\dot{y} = 0. \quad (9)$$

Thereby,

\dot{p}_a is the $nzb \times 1$ vector of the dependent coordinates and
 \dot{y} is the $f \times 1$ vector of the independent (generalized)
coordinates.

After solving equation (9) towards the dependent coordinates and restoration of the original order of the velocity coordinates \dot{p} with the help of the permutation matrix $[Q_p]$, the following $6 \times f$ JACOBI matrix is obtained:

$$[J_y] = [Q_p] \begin{bmatrix} -[C_{ba}]^{-1}[C_{bu}] \\ [E] \end{bmatrix}. \quad (10)$$

and thus the following applies:

$$\dot{p} = [J_y]\dot{y}. \quad (11)$$

The nzb generalized constraint forces (vector f_{zb}), which are produced by the constraining elements, act in the direction of the locked modes of the constraining systems. They are distributed to the center of mass of the bodies with the help of the constraining matrix to a 6×1 force vector f_z ,

$$f_z = [C_b^T]f_{zb}. \quad (12)$$

In order to now eliminate the unknown constraint forces f_z from the equations of motion, it is sufficient to multiply equation (12) from the left with the transposed JACOBI matrix,

$$[J_y^T][C_b^T] = [0]. \quad (13)$$

Linearization of the kinematic equations

In NUBEMM the motions of the bodies relative to the moving reference frame are regarded as small. Linearization of the positional vectors and the rotational matrices between the center of mass of the bodies and the moving reference frame is carried out relative to a known nominal position. Here it is important that the given design position represents a geometrical nominal position.

The positional vectors and the rotational matrices are linearized as follows:

$$\mathbf{r}(t) = \mathbf{r}(0) + \Delta\mathbf{r}(t) + \dots 0, \quad (14a)$$

$$[\mathbf{S}(t)] = [\mathbf{E} + \tilde{\Delta\varphi}(t) + \dots 0][\mathbf{S}(0)], \quad (14b)$$

$$\dot{\mathbf{r}}(t) = \Delta\dot{\mathbf{r}}(t), \quad (14c)$$

$$\omega(t) = \Delta\dot{\varphi}(t). \quad (14d)$$

The equations of motion in a moving reference frame

The equations of motion of the MBS are obtained with the help of the NEWTON-EULER equations. Thus a total of n subproblems are to be solved for a system of n rigid bodies.

The NEWTON and EULER equations are written up in coordinates of the moving reference frame by using the kinematic equations (4):

$$[\mathbf{M}_i] \{ \ddot{\mathbf{p}}_i + [\mathbf{Q}_i(\mathbf{p}_i, \mathbf{p}_i(0))]^* \ddot{\mathbf{z}} + \mathbf{a}_i(\mathbf{p}_i, \mathbf{p}_i(0), \mathbf{z}) \} = \mathbf{f}_i^e + \mathbf{f}_i^k + \mathbf{f}_i^z, \quad (15a)$$

$$i = 1, (1), n. \quad (15b)$$

Here there is:

$$[\mathbf{M}_i] = \begin{bmatrix} m_i & 0 \\ 0 & [\mathbf{I}_i] \end{bmatrix} \hat{=} \text{ mass and inertia matrix,}$$

$$\mathbf{f}_i^e = \begin{bmatrix} f_i^e \\ l_i^e \end{bmatrix} \hat{=} \text{ vector of applied forces,}$$

$$\mathbf{f}_i^k = \begin{bmatrix} 0 \\ -[\tilde{\mathbf{p}}_i + \tilde{\mathbf{Q}}_i \tilde{\mathbf{z}}][\tilde{\mathbf{I}}_i][\dot{\mathbf{p}}_i + \dot{\mathbf{Q}}_i \dot{\mathbf{z}}] \end{bmatrix} \hat{=} \text{ vector of gyroscopic forces,}$$

$$\mathbf{f}_i^z = \begin{bmatrix} f_i^z \\ l_i^z \end{bmatrix} \hat{=} \text{ vector of constraint forces.}$$

$$\mathbf{p}_i(t) = [\Delta x_i, \Delta y_i, \Delta z_i, \Delta \alpha_i, \Delta \beta_i, \Delta \gamma_i]^T \hat{=} \text{ vector of position variables,}$$

$$\mathbf{p}_i(0) = [x_{0i}, y_{0i}, z_{0i}, 0, 0, 0]^T \hat{=} \text{ nominal position vector,}$$

$$\mathbf{z}^*(t) = [\mathbf{x}_f^*, \mathbf{y}_f^*, \mathbf{z}_f^*, \dot{\omega}_x, \dot{\omega}_y, \dot{\omega}_z]^T \hat{=} \text{ reference system.}$$

After elimination of the constraint forces and torques in accordance with equation (13) and combination of all $6n$ scalar equations (15), the equations of motion of the MBS can be given in form of a $fx1$ vector differential equation.

$$J_y^T M J_y \ddot{y} + J_y^T M Q(y)^{**} \ddot{z} = J_y^T [f_y^e + f_y^k - Ma(y, \dot{y}, \ddot{z})]. \quad (16)$$

Here there is:

$$M = \text{diag}[M_1^T, M_2^T, \dots, M_n^T]^T \hat{=} \text{mass and inertia matrix},$$

$$Q = [Q_1^T, Q_2^T, \dots, Q_n^T]^T \hat{=} \text{global distribution matrix},$$

$$a = [a_1^T, a_2^T, \dots, a_n^T]^T \hat{=} \text{global acceleration vector},$$

$$f_y^e = [f_1^{eT}, f_2^{eT}, \dots, f_n^{eT}]^T \hat{=} \text{vector of applied forces},$$

$$f_y^k = [f_1^{kT}, f_2^{kT}, \dots, f_n^{kT}]^T \hat{=} \text{vector of gyroscopic forces.}$$

In equation (16) there are f equations for $f+6$ unknowns, i.e. \ddot{y} and \ddot{z} .

Assuming that the moving reference frame has to be connected to one body of the MBS, the motion of which are contained in the vector of the generalized coordinates \ddot{y} , 6 additional trivial equations

$$\ddot{y} = [\ddot{y}_1^T, \ddot{y}_2^T, \dots, \ddot{o}_k^T, \dots, \ddot{y}_{q-1}^T, \ddot{y}_q^T]^T$$

are obtained in order to solve equation (16) unambiguously [3].

One ensures, moreover, that vector \ddot{z}^{**} represents inertial acceleration given in coordinates of the moving reference frame. Thus the motions of the moving reference frame cannot be obtained directly via integration of the acceleration values \ddot{z} .

3. Structure of the NUBEMM data set [4]

After the mechanical substitute model is available as MBS, the bodies, constraining elements, coupling elements, applied forces as well as the observing elements are numbered arbitrarily. These specified numbers having up to three digits are used for identification in addition to the symbolic interface generated by NUBEMM when the scalar equations of motion are output.

A detailed input description of NUBEMM can be found in [4]. The structure of the data set as well as the essential features of the individual input blocks are explained below.

The data set contains

- key words,
- data lines and
- comment lines.

Comment lines may appear anywhere in the data set; they are identified as "C" in column 1. The key words are designated as

STEUER	(input/output control)
FUEHRUNG	(description of moving reference frame)
KOERPER	
MASSEN	(mass and inertia block)
BINDUNG	(constraining element)
KOPPEL	(applied spring/damper forces)
KRAEFTE	(forces and torques)
EOBACHTER	(observer)

The input blocks do not all have to be available and can be input in any order.

The data lines contain real numbers and integers as well as symbolic abbreviations.

All geometric input data can either be input in the reference frame or in the body-fixed coordinate system. One coordinate system is specified per body and point of constraint.

The individual input blocks will now be explained briefly:

STEUER (CONTROL)

The "STEUER" data field contains control variables to select certain program variants, particularly for determination of the output format of the differential equations, generation of plots, output of certain system matrices and tests.

FUEHRUNG (REFERENCE FRAME)

The motions of the moving reference frame and the body number with which the moving reference frame is connected are defined here.

KOERPER (BODY)

The nominal configuration of the bodies is specified here by input of the position of the center of mass (x_0, y_0, z_0) and position of the body-fixed coordinate system relative to the inertial frame. Up to six possible motion (3 translations, 3 rotations) are permitted for each body relative to the inertial frame. These are specified by a 6-digit integer, the first 3 digits of which are reserved for translations and the last 3 for rotations.

These are:

- 0 = The motion is not permitted relative to the inertial frame.
- 1 = The motion is permitted.
- 2 = The motion is wanted as a generalized coordinate.

Reduction of the motion of all bodies to the generalized coordinates is carried by the program with the help of the constraint input block.

MASSE (MASS)

This block contains the input of mass and inertial tensor in the body-fixed coordinate system.

BINDUNG (CONSTRAINING ELEMENT)

The kinematic constraining elements restrict the motion of the bodies. A constrained coordinate system (the coordinate system is fixed with body "I") is applied for each constraining element (joints, guide). The relative motions between two bodies are specified in it (the inertial frame is body "0"). The static position of the constraining system is defined relative to the inertial frame by three cardan angles or by modelling one axis of the constraining system. A 6-digit integer specifies which relative motions are now free and which are locked.

These are:

- 0 = the relative motion between body I and J is locked;
- 1 = the relative motion between body I and J is free;
- 2 = the relative motion between body I and J is controlled.

KOPPEL (COUPLING ELEMENTS)

An input is made here to compute the applied spring and damper forces. The elastic connections are modelled in the MBS by spring and damper elements. These elements can be translational (TRA) or rotational (RTA) components. In the case of rotational elements, the directional vector must additionally be given in coordinates of the inertial frame.

KRAEFTE (FORCES)

Input of external applied forces and torques (e.g. tires and aerodynamic forces) is carried out in this block. In addition, a rotor with constant rotational speed can be modelled in this block.

BEOBACHTER (OBSERVER)

An observer is a 'measuring' element which computes the position, velocity and/or acceleration.

These are required when comparing simulation calculations and measurements but also, for example, for computation of the slip angle to calculate the tire forces.

4. Example

The setup of a complete NUBEMM data set is to be shown using the mechanism as an example (see appendix in this book).

Modelling of the mechanism with NUBEMM first of all requires knowledge of the nominal position; this can be easily determined from the given initial condition.

The assumption, on which NUBEMM is based (all bodies shall carry out small motions relative to a moving reference frame) and which is used for calculation of the equations of motion, is not given for this example. Therefore, the equations of motion are generated relative to the inertial frame. Since the equations of motion are linear in this case, a simulation calculation is dispensed with. Attention must be paid to the partially generated symbolic interface of the equations of motion.

NUBEMM data-set of the mechanism**GRAVITATION**

```
C!----- !----- !
C! gravity constant GX or GY or GZ
C!----- !----- !
!GY !+0000.0! ! ! ! ! ! ! ! ! ! !----- !
C!----- !----- !
```

STEUER

```
C!----- !----- !
C! control statement
C! LOG(4) = systemeigenvalue
C! LOG(10) = constraint forces and torque
C! LOG(71) = spring preloadforces
C! OUTPUT = ACSL CSMP FORT
C!----- !----- !
C!out! ! !LOG(??) T=yes F=no
C!put ! ! !1----+---1----+---2----+---3----+---4----+---5!
C!----- !----- !1----+---6----+---7----+---8----+---9----+--- !
!ACSL! ! !
!PLOT! ! !
C!----- !----- !----- !
```

FUEHRUNG

```
C!----- !
C! description of the referenz frame
C!----- !
C! !DOF of ! !
C!par!ref. !INDEX ! DYNAM or KINEM or INERTIAL
C!nr !frame !
C!----- !----- !----- !----- !----- !----- !----- !
! 0 !000000 !INERTIAL ! ! ! ! ! ! !----- !
C!----- !----- !----- !----- !----- !----- !----- !
```

KOERPER

```
C!----- !
C! central of mass and part coordinate system
C!----- !
C! ! D O F ! central of mass ! orientation of the !
C!par!of the ! coordinate ! parts coor. system !
C!nr! part ! x0 ! y0 ! z0 ! alfa ! beta ! game !
C!----- !----- !----- !----- !----- !----- !----- !
!1 !110002 !.92192 !-.05722! ! ! ! ! !-3.5520!
!2 !110001 !-4.4915!.279010! ! ! ! ! !-3.5528!
!3 !110001 !-18.743!20.4872! ! ! ! ! !26.0948!
!4 !110001 !-30.227!12.0728! ! ! ! ! !40.7096!
!5 !110001 !-53.245!16.6312! ! ! ! ! !27.9327!
!6 !110001 !-28.542!-10.715! ! ! ! ! !57.7470!
!7 !110001 !-59.264!-10.605! ! ! ! ! !70.5242!
!80 !000000 !14.0000!72.0000! ! ! ! ! !----- !
C!----- !----- !----- !----- !----- !----- !----- !
```

MASSE

```
C!-----
C!      mass and inertia
C!-----
C!par! mass   !    inertia
C!nr ! m     ! ixx  ! iyy  ! izz  ! ixy  ! ixz  ! iyz  !
C!----!-----!-----!-----!-----!-----!-----!-----!-----!
!1  !4.32E-5!      !      !12.19E-3!
!2  !3.65E-6!      !      !14.41E-4!
!3  !2.37E-5!      !      !15.25E-3!
!4  !7.06E-6!      !      !15.66E-4!
!5  !7.05E-5!      !      !1.17E-2!
!6  !7.06E-6!      !      !15.66E-4!
!7  !5.49E-5!      !      !1.91E-2!
!80 !          !      !      !      !
C!----!-----!-----!-----!-----!-----!-----!-----!-----!
```

KOPPEL

```
C!-----
C!      spring and damper forces and torque
C!-----
C!ele!between! point on part i      ! point on part j      ! ! !
C!nr !part   !
C!  ! i ! j ! xi ! yi ! zi ! xj ! yj ! zj !
C!----!-----!-----!-----!-----!-----!-----!-----!
C!  !index  !direction vektor pj-pi ! spring!damper ! exita- !
C!  !TA !RA ! x0 ! y0 ! z0 !           !           ! tion !
C!----!-----!-----!-----!-----!-----!-----!-----!
!1  !3  !80 !9.57000! .745 !      !      !      !KK!RR!
!...!TA !      !      !      !CS !      !      !....!
C!----!-----!-----!-----!-----!-----!-----!-----!-----!
```

KRAEFTE

```
C!-----
C!      external forces and torques T-torque F-force
C!-----
C!  ! at    ! point at part i      ! direction vektor      !
C!ele!part  ! coordinate        ! coordinate        !
C!nr !nr !ind! xi ! yi ! zi ! x0 ! y0 ! zo !
C!----!-----!-----!-----!-----!-----!-----!-----!-----!
!1  !1  !T !      !      !      !      ! 1. !K !R !
C!----!-----!-----!-----!-----!-----!-----!-----!-----!
```

BINDUNG

```

C!-----
C!      constraint elements (two line input)
C!-----
C! constr. ! point on part i   ! point on part j   ! ! !
C!between part
C! ! i ! j ! xi ! yi ! zi ! xj ! yj ! zj ! ! !
C!-----!
C!con!joint ! joint system !
C!nr !DOF ! alpha ! beta ! gama !
C!-----!
!10 !1 ! 0 !-0.9237! 0.0000! 0.0000! ! ! !K !R !
!...!000001 ! ! ! !.....!.....!.. ....!.. !
!30 !3 ! 0 !-10.430!18.7450! 0.0000! ! ! !K !R !
!...!000001 ! ! ! !.....!.....!.. ....!.. !
!50 !5 ! 0 !-23.080!-9.1630! 0.0000! ! ! !K !R !
!...!000001 ! ! ! !.....!.....!.. ....!.. !
!70 !7 ! 0 ! 4.4930!12.2800! 0.0000! ! ! !K !R !
!...!000001 ! ! ! !.....!.....!.. ....!.. !
!12 !1 ! 2 ! 6.0763! 0.0000! 0.0000!11.5000! 0.0000! 0.0000!KK!RR!
!...!000001 ! ! ! ! ! ! ! ! ! ! ! ! ! !
!54 !5 ! 4 !16.9200!-9.1630! 0.0000!-0.0000! 5.7900! 0.0000!KK!RR!
!...!000001 ! ! ! ! ! ! ! ! ! ! ! ! ! !
!76 !7 ! 6 ! 4.4930!-27.720! 0.0000!-5.7900! 0.0000! 0.0000!KK!RR!
!...!000001 ! ! ! ! ! ! ! ! ! ! ! ! ! !
!23 !2 ! 3 !-16.500! 0.0000! 0.0000!-10.430!-16.255! 0.0000!KK!RR!
!...!000001 ! ! ! ! ! ! ! ! ! ! ! ! ! !
!43 !4 ! 3 ! 0.0000!-14.210! 0.0000!-10.430!-16.255! 0.0000!KK!RR!
!...!000001 ! ! ! ! ! ..!.....!.....!.. ....!.. !
!63 !6 ! 3 !14.2100!-0.0000! 0.0000!-10.430!-16.255! 0.0000!KK!RR!
!...!000001 ! ! ! ! ! ..!.....!.....!.. ....!.. !
C!-----!

```

BEOBACHTER

```

C!-----
C!      mess and observers elements (two line input)
C!-----
C! !between! point on part i   ! point on part j   ! ! !
C! ! part ! coordinate       ! coordinate       ! ! !
C! ! i ! j ! xi ! yi ! zi ! xj ! yj ! zj ! ! !
C!-----!
C!obs! index !dirction from pj==>pi !
C!nr !tra!rot! x0 ! y0 ! z0 !
C!-----!
!3 !3 ! 0 !
!...!TP ! ! 1. !
!3 !3 ! 0 !
!...! !RP ! 1. !
C!-----!

```

NUBEMM generated equations of the mechanism

```

' ..numeric symbolic equation of motion ..... .
' ..input member ..... QE37027.NUBEMM.DATA (BUCH7) .. .

' ..spring deflection in spring direction..... .

SL1 = 8.4630679E-05*PK1

' ..generalized spring forces .. .

FS1 = 8.4630679E-05*FSS1

' ..external forces and torges .. .

QFP3 = TE1

' ..sum of all generalized external forces .. .

QFQ1 = QFP3

' ..sum of all generalized forces .. .
' ..external and spring damper forces .. .

QFX1 =-FS1+QFQ1

' ..generalized accelerations .. .

AK1 = 4.3168043E+02*QFX1

' ..mess and observers equations .. .

BPX3 =-3.9350438E-05*PK1-1.8743000E+01
BPY3 =-5.6590502E-05*PK1+2.0487200E+01
CONSTANT BPZ3 =0.0
CONSTANT BPI3 =0.0
CONSTANT BPPJ3 =0.0

' ..ACSL integrations statements .. .
VK1 = INTEG(AK1 , 0. )
PK1 = INTEG(VK1 , 0. )

```

5. Literature

- /1/ Pankiewicz,E.: Anwendung rechnergestützter Verfahren zur Generierung der Bewegungsgleichungen im Kraftfahrzeugbau. Fortschr.-ber VDI reihe 12 NR. 69. Düsseldorf: VDI-Verlag 1986.
- /2/ Thomson,B.; Rathgeber, H.:Automated Systems Used for Rapid and Flexible Generation of Vehicle Simulation Models Exemplified by a Verified Passenger Car and a Motorcycle Model. Proc. 8th iAVSD ,Volume 12 1983.
- /3/ Schmid,T.W.: Ein Simulationsverfahren für nichtlineare gekoppelte Fahrzeuggespanne. Dissertation TH Karlsruhe, 1988.
- /4/ BMW-interne-Bericht, NUBEMM-Benutzeranleitung. BMW AG, München 1987.

SYM - Program Package for Computer-aided Generation of Optimal Symbolic Models of Robot Manipulators

Vukobratović Miomir, Kirćanski Nenad, Timčenko Aleksandar, Kirćanski Manja
Laboratory for Robotics and Flexible Automation
Mihailo Pupin Institute, 11000 Beograd, Yugoslavia

Abstract

This contribution presents a new program package for the generation of efficient manipulator kinematic and dynamic equations in symbolic form. Since the computational efficiency of the generated symbolic models is extremely high, the real-time implementation of rather complicated mechanical equations become possible even on low-cost microcomputers.

The program package SYM is concerned with serial-link manipulators with stiff or elastic joints. The basic algorithm belongs to the class of customized algorithms that reduce the computational burden by taking into account the specific characteristics of the manipulator to be modeled. The high-level program code for computing various kinematic and dynamic variables (elements of homogeneous transformation matrices, Jacobian matrices, driving torques, elements of dynamic model matrices etc.) is generated. The application of recursive symbolic relations yields high, but not the minimal numerical complexity. Thus, we also apply the second-step optimization of the generated symbolic code. This step is based on the extraction of the expressions which appear more than once in the output code.

1. Introduction

In order to compensate for nonlinear dynamic effects during the manipulator motion, various control laws employ robot dynamic equations, either in the direct or in the feedback branch of the control structure. Beside the dynamic model, the control structure includes regulators, parameter estimators and other subsystems. Thus the computation of the dynamic model must be performed in a minimal possible time.

The development of the computer-oriented methods for the generation of dynamic robot models can roughly be divided into three stages: 1. Numeric methods [1 - 8], 2. Numeric-symbolic

methods [9 - 11] and 3. Symbolic methods [12 - 17]. This is also the chronological order of development of the modeling techniques. The later the model appeared, the less computational burden it offered. Naturally, the model generation algorithms were more and more complex.

The computational inefficiency of the numerical procedures prevented their on-line application in robot controllers. For example, the most efficient numeric method based on Newton-Euler's (NE) equations requires 804 multiplications and 666 additions for a 6 joint manipulator [1].

In numeric-symbolic models the parameters are treated numerically, while the joint coordinates, velocities and accelerations are treated symbolically. This method reduces the computational burden for the robots with less than 6 joints drastically. However, for robots with more than 6 joints this method does not yield much better results than numeric methods.

The latest - symbolic modeling techniques treat both parameters and joint coordinates as symbolic variables. They are the most complicated techniques since they involve algebraic operations between symbolic expressions and the computer-aided generation of new symbolic expressions. These techniques yield [12 - 17] very low computational complexity. For example, the symbolic model of Stanford arm involves 155 multiplications and 135 additions. For a general 6 link robot both the number of multiplications and the number of additions does not exceed 300. The method for generating symbolic expressions within the program package SYM is described in Sec. 2. Although it is several times more efficient than the numeric model described in Ref. [1], it is still hard to compute joint torques in several hundreds of micro seconds by the use of low-cost microcomputers.

Several papers dealing with parallel processing of robot dynamics have appeared since 1982 [18 - 20]. In most of these papers parallel processing of the numeric NE equations by means of several processors is considered. The computation time was reduced from 2.5 to 4.6 times. But, the communication between

processors via a common memory region usually takes a significant percentage of the computing time. Thus, even more than 6 processors might be necessary to compute the model in several hundreds of μ s. The increased number of processors might make the overall system unreliable for industrial applications.

Now the question is how can the computational time be further reduced? There are two ways to achieve this. The first is to employ more advanced methods for robot dynamics modeling which offer lower computational burden, and the second is to implement them on fast computer systems. The program package SYM is designed to accommodate to both of these requirements. The sophisticated optimization based on the extraction of the expressions which appear more than once in the output code is introduced (Sec. 3). The numerical efficiency of the generated code is thus improved from 1.2 to 3 times.

The generated programs with the SYM are examined on several real-time control computers. Experiments indicate that the typical computing time on a standard 16-bit micro-computer is 5 - 20 ms, while the typical computing time on an array processor is 50-100 μ s.

2. Symbolic recursive relations

In this section some of the recursive relations which are used by the SYM are described. The recursive relations describe the kinematic and dynamic behavior of an arbitrary serial-link manipulator. The kinematic equations are based on Denavit - Hartenbergs (D-H) parameters, while the dynamic equations are based on Newton-Euler's formalism.

The program package SYM generates the following types of models:

- Homogeneous transformation matrix between the hand and base coordinate frame
- Jacobian matrix relating joint and world coordinate derivatives

Inverse dynamic model for computing driving torques
 Matrix of inertia
 Matrix of Christoffel symbols, etc.

Since SYM generates various types of robot models, we will present one of them: Newton - Euler's formalism for obtaining inverse dynamic robot model.

The vector form of Newton-Euler's formalism for an arbitrary serial link manipulator with revolute joints is summarized in Appendix. This form is not suitable for symbolic processing because there are several types of indexed variables (vectors, matrices, tensors), etc. Much more suitable is the scalar form of variables and expressions. The scalar expressions are formed strictly following the equations in Table 1. For example, the expression for computing the angular accelerations

$$\dot{\omega}_i^{i-1} = \dot{\omega}_{i-1}^{i-1} + (\ddot{q}_i \vec{\omega}^{i-1} \vec{z}_{i-1} + \dot{\omega}_{i-1}^{i-1} \times \dot{q}_i^{i-1} \vec{z}_{i-1})$$

is translated into the scalar form

$$\begin{aligned} OMDi(i-1)1 &= OMD(i-1)(i-1)1 + QDi * OM(i-1)(i-1)2 \\ OMDi(i-1)2 &= OMD(i-1)(i-1)2 - QDi * OM(i-1)(i-1) \\ OMDi(i-1)3 &= OMD(i-1)(i-1)3 + QDDi \end{aligned} \quad (1)$$

Here, $OMDijk$ corresponds to k-th component of the vector $\dot{\omega}_i^j$, while QDi and $QDDi$ correspond to \dot{q}_i and \ddot{q}_i , respectively. Such symbolic relations are general, i.e. they hold for any manipulator.

The scalar expressions are built in SYM in the form of a data file. User can also introduce his own model. For example, the equations determining the position of end-effector with respect to a given point along the robot arm, can be added easily.

The user creates the list of D-H parameters as well as the list of dynamic parameters (the mass and inertia of each link) for any particular robot. Thus, the user defines the type of each joint (revolute or sliding), number of joints, D-H parameters

which are zero etc. The user also defines which elements of inertia tensors are zero. The numerical values of parameters are not supplied except to that parameters which values are exactly zero or one.

3. Symbolic customizer and optimizer

The Symbolic Customizer/Optimizer (SyCO) represents the program which generates the customized and optimized model in the form of a high-level language program. The customized model means the symbolic model relevant to a particular robot structure. The optimized model means the customized model which is additionally optimized with respect to the number of floating-point operations.

The input of the SyCO represents both the file composed of the symbolic expressions and a data file relevant to a particular robot structure described in the forgoing section. The SyCO includes customization and optimization of the general symbolic expressions and generates the set of computationally optimal expressions, given a particular robot.

The symbolic model generator (Fig. 1) consists of two modules: The first module (Symbolic Customizer - SyC) customizes the general symbolic expressions by eliminating unnecessary computations (multiplication by 1, adding zero etc.). The customization improves the efficiency of the obtained models several times comparing to the most efficient numerical modeling techniques [17]. The second module (Symbolic Optimizer - SyO) is concerned with the mathematical optimization of the code obtained by the application of SyC. This module additionally reduces the number of floating point operations by 20-70 %.

3.1 Symbolic customizer

Each parameter or variable A that appear in the input file is assigned three magnitudes: code, value, and name. Code may take value 1, 2, and 3, where 1 indicates that A is constant, 2 - that A is a simple variable equal to another already evaluated,

and 3 - that A is a complex variable that requires at least one additional multiplication or addition for its evaluation. Value contains numerical value of A if it is a constant, and name contains the symbolic name of A. A can also be an array of variables/parameters.

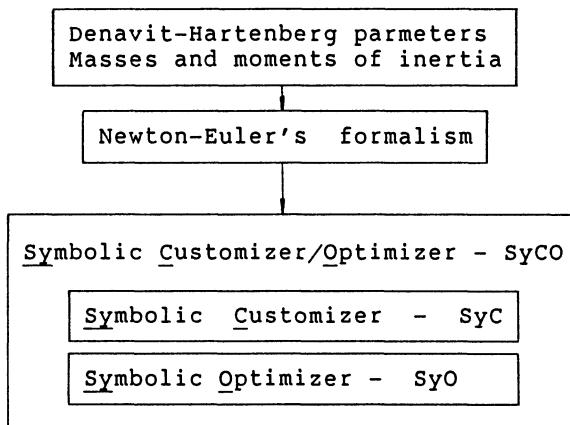


Fig. 1. Symbolic model generator

The input file of SyC may include different types of expressions. The most usual form is

$$z = \pm x_1 [* y_1] \pm \dots \pm x_k [* y_k] \quad (2)$$

where \pm denotes + or - sign, and [.] denotes that the expression in parentheses is optional. Here x_i , y_i , $i = 1, \dots, k$, are parameters or variables that have already been evaluated in the recursive algorithm, so that they all have already determined codes, values and names. The parameters and variables on the right-hand side of (47) are replaced by their names while generating the program line to evaluate z, except if they are equal to 0, ± 1 , or if the variables are simple variables (code = 2). In that case they are replaced by their real values. Thus the multiplications by 0, ± 1 are eliminated as well as unnecessary loading from one variable to another. If the expression for z is reduced to the form with at least one

multiplication or addition, z becomes a complex variable and the corresponding line is written down into the output program. The SyC automatically eliminates the lines which are not used for computing the output variables.

3.2 Symbolic optimizer

The output program code of the SyC represents the input for the Symbolic optimizer SyO. The first step of the optimization is to represent the variables of the customized models in a particular polynomial form. Thus, SyO proceeds to the analysis of data-dependency among recursive expressions generated by SyC. After this analysis each variable including the output variables is expressed in the form

$$z = \sum_{l=1}^L c_l \prod_{i=1}^n (\cos q_i)^{e_{li}^c} (\sin q_i)^{e_{li}^s} q_i^{e_{li}^q} \dot{q}_i^{e_{li}^{\dot{q}}} \ddot{q}_i^{e_{li}^{\ddot{q}}} \quad (2)$$

where c_l are constants which depend on robot parameters, e_{li}^x ($x \in \{c, s, q, \dot{q}, \ddot{q}\}$) are constant exponents which satisfy the condition: $e_{li}^x \in \{0, 1, 2\}$, and n is the number joints [9 - 11]. For example, the driving torque of the second joint of a two-link manipulator can be presented as

$$P_2 = c_1 (\sin q_2) \dot{q}_1^2 + c_2 \ddot{q}_1 + c_3 \ddot{q}_2 + c_4 (\cos q_2) \ddot{q}_1 \quad (3)$$

where $c_1 = A_1 * A_2 * (AM_2 + AMK_2)$, ..., $c_4 = A_1 * A_2 * (AM_2 + AMK_2)$ are constants.

The polynomial (2) can be presented as

$$z = \sum_{l=1}^L c_l(z) \prod_{k=1}^K p_k(z)^{e_{lk}} \quad (4)$$

The constants c_l , polynomial arguments p_k and the exponents e_{lk} depend on particular variable z , thus $c_l = c_l(z)$, $p_k = p_k(z)$ and $e_{lk} = e_{lk}(z)$. Note that the polynomial (4) is more general

than the polynomial (2) since the arguments p_k may present any variable or constant. For example, $p_k = \dot{q}_2^2 \sin(q_1)$ or $p_k = A_1*A_2*(AM_2+AMK_2)$ may be used as polynomial arguments. The trigonometric polynomial (4) can be presented by the use of the the matrices

$$\begin{aligned} S_c &= [c_1 \dots c_L]^T && \text{matrix of the constants;} \\ S_p &= [p_1 \dots p_K] && \text{matrix of the variables;} \\ S_e &= \begin{bmatrix} e_{11} \dots e_{1K} \\ e_{L1} \dots e_{LK} \end{bmatrix} && \text{matrix of the exponents.} \end{aligned} \quad (5)$$

The ordered triple $S = (S_c, S_p, S_e)$ will be referenced as polynomial matrix in the text to follow. For example, the polynomial (3) can be presented in matrix form as

$$S_c = [c_1 \ c_2 \ c_3 \ c_4]^T \quad (L = 4)$$

$$S_p = [\cos q_2 \ \sin q_2 \ q_1 \ \dot{q}_1 \ \ddot{q}_2] \quad (K = 4)$$

$$S_e = \begin{bmatrix} 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Now let us denote the set of the polynomial matrices by S . It is clear that each symbolic model variable can be represented with an infinite number of polynomial matrices. The matrices that represent the same variable are called equivalent polynomial matrices. For example, the polynomial P_2 can be presented as

$$P_2 = c_1 (\sin q_2) \dot{q}_1^2 + c_2 \ddot{q}_1 + c_3 \ddot{q}_2 + c_4 (\cos q_2) (\cos q_1^2) \ddot{q}_1 + c_5 (\cos q_2) (\sin q_1^2) \ddot{q}_1$$

where $c_4 = c_5$. The corresponding polynomial matrix has 5 rows in contrast to the polynomial matrix of the same polynomial (3) which has 4 rows.

SyO has the possibility to minimize the number of rows in a polynomial matrix. It is equivalent to minimizing the number of additions of the corresponding polynomial. Furthermore, SyO incorporates a number of routines for algebraic operations among polynomial matrices: addition/subtraction, multiplication etc. The algebraic formalism is based on the algebraic operations between the polynomials of the form (4) which are simply extended to the matrix operations.

The algebraic routines are necessary for obtaining the polynomial matrices of the left-hand variables of the customized expressions (output of SyC). Following the recursive symbolic expressions the polynomial matrices of the output variables are formed.

The next step is to reconstruct the output variable's polynomial matrices back into the program file in the way that minimizes its numerical complexity. More precisely, the matrix reconstruction algorithm is the second part of SyO and it minimizes the number of multiplications required for computation of the output variable's polynomial matrices.

First, we slightly modify the polynomial matrix of each variable in order to eliminate the equal constants from the matrix of constants S_c . These constants are replaced by 1 or -1 depending on the sign. The matrix of exponents is expanded by one colon corresponding to the argument p_{k+1} equal to this constant.

Let us now designate the set of the output variable's polynomial matrices by SMPM (from Symbolic Model Polynomial Matrix). After the repetitive constant extraction, the SMPM is ready for the reduction by monomial extraction algorithm. The main idea of that algorithm will be explained now. SMPM defines the set of expressions of the form (4). Each of these expressions includes a number of multiplications which can be easily determined. Each of these expressions can be presented as

$$z = \mu z_I + z_{II}$$

where:

$$\mu = p_{j_1}^{e_{j_1}} p_{j_2}^{e_{j_2}} \dots p_{j_m}^{e_{j_m}} \quad (8)$$

is the monomial of degree $d(\mu) = e_{j_1} + e_{j_2} + \dots + e_{j_m}$.

It is clear now that the extraction of the monomial μ reduces the number of multiplications comparing to the initial set of expressions. The number of multiplications is more decreased if there are more rows in SMPM which contain μ and if $d(\mu)$ is larger.

If we calculate the monomial μ in advance, $d(\mu)$ multiplications in each monomial which contains μ will be saved. If we denote the number of such monomials by $n(\mu)$, we will save $d(\mu)*n(\mu)$ multiplications. This number has to be decreased by the number of expressions where μ appears, because we have to multiply the sum of monomials divided by μ with μ itself, which requires one additional multiplication for each expression. Finally, the computation of μ itself takes $d(\mu) - 1$ multiplications. Thus, we can easily compute the decrement of the number of multiplications given a monomial μ . Now the optimization algorithm is evident: monomial μ which maximizes the decrement of the number of multiplications is determined in each step.

When the monomial μ is determined from the set of expressions $\{z\}$, the new set $\{z_i, z_{ii}\}$ is generated. If the degree of the extracted monomial is greater than 1, that monomial is written into the output program file. The monomial extraction procedure continues until there is no more monomials which can reduce the number of multiplications. When the SMPM is reduced completely, it degenerates to the sum of simple expressions. Then the optimization procedure terminates and determined monomials are written down into a program file.

It is clear that the model optimization algorithm described above is suboptimal since each optimization step does not depend on the previous steps, and does not reflect in the steps to follow.

4. Fields of Application of the Software Package SYM

The software package SYM is designed to set up computationally optimized robot models suitable for real-time application. The generated models are coded in "C" and/or FORTRAN language. SYM runs under VAX/VMS operating system (till 12 d.o.f.), MS-DOS (till 6 d.o.f.). The uVAX/ULTRIX operating system version is currently under development. The fields of application are

- robot dynamics and simulation
- robot control (kinematics and dynamic control on monoprocessor, multi-processor and array processor architecture)
- biomechanics (control and simulation)
- serial-link satellites and machines (control and simulation)

5. Program package features - short description

In this chapter we will briefly describe the structure of SYM. It is organized as a menu-oriented dialogue with self-explanatory commands and an on-line HELP facility. Also, there is a screen-oriented input of the Denavit-Hartenberg manipulator parameters coupled with a 3D graphical presentation of the kinematic scheme. It makes SYM user-friendly in a sense that it gives the user opportunity to work with it without a manual or any introductory instructions.

SYM generates following types of manipulator models:

- driving torques
- Coriolis + centrifugal force vector
- inertia matrix
- gravitational force vector
- Coriolis + centrifugal + gravitational force vector
- transformation matrix between base and end-effector coordinate frames
- Jacobian matrix which relates joint and world

- coordinate's derivatives
- joint position vectors

Also, there is a set of various transformations of the generated models:

- model rearrangement for the computation on an array processor with several parallel multipliers and adders
- additional optimization method that decreases the numerical burden up to two times
- model differentiation by an arbitrary parameter or variable, i.e. generation of the sensitivity and linearized models
- time differentiation
- simultaneous optimization of several models

Also, a part of SYM is a preprocessor which converts input equations into an internal form which enables symbolic model generation for a specified manipulator. It means that user has to define the equations which relates the input variables and parameters with desired output, and to pose them in the way appropriate for SYM's preprocessor. Then, all the job is done automatically.

Now we can summarize the complete algorithm of symbolic model generation:

1. Input of the model equations
2. Input equations transformation by SYM preprocessor
3. Input of the manipulator parameters
4. Model generation by SyC algorithm
5. (optionally) Model transformations: differentiation by parameter, variable or time; model combination according to the rules of model algebra
6. (optionally) Model optimization by SyO algorithm
7. (optionally) Model rearrangement due to the computation on an array-processor

6. Example

For the second test example ("robot") defined in this handbook, the following models are generated: inverse dynamics (computation of joint torques given joint coordinates, velocities and accelerations), inertia matrix, Jacobian and matrices of transformation between the joint and world coordinates. The kinematic and dynamic parameters of the "robot" are given in Table 1. The computational efficiency (number of multiplications and additions / subtractions) is given in Table 2. We see that the computation of driving torques/forces requires only 50 multiplications and 25 additions/subtractions. According to the present literature in computational robot dynamics this seems to be the most efficient code obtained until now. The output of the Symbolic Optimizer (SyO) of the inverse dynamic model is listed in Table 3. The generated inertia matrix program code, i.e. the output of Symbolic Customizer and Optimizer is listed in Tables 4 and 5. The numerical efficiency for several other well-known 6-link robots is illustrated in Table 6.

By the use of the generated inertia matrix model and the model for computing joint torques/forces that originate from Coriolis, centrifugal and gravity forces, we obtained the simulation results for the second test example defined in this handbook. The robot joint coordinates evolution (Z_1 - vertical sliding, GA_1 - rotation about vertical axis, and Y_2 - horizontal sliding) during 2 seconds are presented in Fig. 2 to 4.

The software package SYM is supported by 3D graphics (Fig. 5). Given the joint coordinates and a view angle, we obtain the graphical presentation of the robot.

Table 1. Denavit-Hartenberg's kinematic parameters (a , α , θ , d), position vectors of links center of mass in local coordinates (R_x , R_y , R_z), and the elements of inertia tensors (J_{xx} , ..., J_{zz}) of the "robot"

Parameter	Trans.	Rot.	Trans.	Rot.	Rot.
a [m]	0	0	0	0	0.05
α [rad]	0	$-\pi/2$	0	$\pi/2$	0
θ [rad]	$\pi/2$	q_2	0	q_4	q_5
d [m]	q_1	0	q_3	0	0
m [kg]	0	250	150	0	100
R_x [m]	0	0	0	0	0
R_y [m]	0	0	0	0	0
R_z [m]	0	0	-0.5	0	0
J_{xx} [kg m ²]	0	90	13	0	1
J_{xy} [kg m ²]	0	0	0	0	0
J_{xz} [kg m ²]	0	0	0	0	0
J_{yy} [kg m ²]	0	90	13	0	4.3
J_{yz} [kg m ²]	0	0	0	0	0
J_{zz} [kg m ²]	0	10	0.75	0	4

Table 2. Number of multiplications (*) and additions/subtractions (+/-) in computer-generated code of various types of robot models (SyC - Symbolic model Customizer, SyO - Symbolic model Optimizer)

Model	SyC		SyO	
	*	+/-	*	+/-
Inverse dynamics	71	52	50	25
Inertia matrix	61	28	23	11
Jacobian matrix	19	5	17	4
Transformation matrix	19	7	18	9

Table 3. Inverse dynamic model of the "robot" obtained as the output of the Symbolic Optimizer (SyO)

	cntd.
ROBOT_DU_OPT: OFF-LINE program	CONS3 = AM5+AM3+AM2
SUBROUTINE ROBOT_DU_OPT\$OFF	CONS4 = AM5*G0+AM3*G0+AM2*G0
INCLUDE 'M\$SYM:ROBOT_DUOPT.CMM'	CONS5 = AJYY5+DJ52-AJZZ5-A5*AM5*A5
RRR41= RR42*RR42	CONS6 = -UNITY-DJ51
BJXX4= AJXX4+AM4*RRR41	CONS7 = -UNITY+AJYY5+A5*AM5*A5
RRR43= RR42*RR42	CONS8 = -UNITY-DJ51+AJYY5-DJ52+A5*AM5*2*A5
BJZZ4= AJZZ4+AM4*RRR43	CONS9 = AJYY5+A5*AM5*A5
UNITY= 1.	CONS10 = -AJYY5+DJ52-A5*AM5*2*A5
DJ41= BJZZ4-AJYY4	CONS11 = AJZZ5+A5*AM5*A5
DJ43= AJYY4-BJXX4	CONS12 = BJYY3+AJYY2
DJ51= AJZZ5-UNITY	CONS13 = AM5+AM3
DJ52= AJXX5-AJZZ5	CONS14 = AMS33+AMS33
DJ53= UNITY-AJXX5	CONS15 = -AMS33
AMS42= AM4*RR42	CONS16 = DJ51
CONS1 = AM5*2*A5	CONS17 = -DJ52+A5*AM5*A5
CONS2 = AM5*A5	CONS18 = -A5*AM5*G0
	END

	cntd.
ROBOT_DU_OPT: ON-LINE program	VAR35 = -CONS11+VAR21
SUBROUTINE ROBOT_DU_OPT\$ON	VAR36 = VAR20+VAR22
INCLUDE 'M\$SYM:ROBOT_DUOPT.CMM'	VAR37 = VAR3+VAR4
C4=COS(Q4)	VAR38 = -VAR8+CONS15
S4=SIN(Q4)	VAR39 = S4*QD2
C5=COS(Q5)	VAR40 = C5*VAR39
S5=SIN(Q5)	VAR41 = VAR36*VAR40
VAR1 = -QD5*CONS10	VAR42 = VAR27*QD2
VAR2 = QDD4*CONS9	VAR43 = VAR28*QDD2
VAR3 = Q3*CONS2	VAR44 = VAR43*C4
VAR4 = CONS7*S5	VAR45 = VAR30*S4
VAR5 = S5*C4	VAR46 = VAR45*S5
VAR6 = CONS17*C4	VAR47 = VAR31*Q3
VAR7 = S5*QD2	VAR48 = VAR32*C4
VAR8 = CONS13*Q3	VAR49 = VAR33*C5
VAR9 = QDD1*CONS2	VAR50 = VAR34*QD2
VAR10 = QD3*CONS1	VAR51 = VAR50*QD3
VAR11 = VAR5*VAR10	VAR52 = VAR35*VAR39
VAR12 = S5*VAR3	VAR53 = VAR52*QD4
VAR13 = QDD1*CONS3	VAR54 = VAR41*CONS7
VAR14 = CONS2*C5	VAR55 = VAR37*S4
VAR15 = VAR14*QDD3	VAR56 = VAR55*QDD2
VAR16 = VAR6*QD2	VAR57 = VAR38*QD2
VAR17 = -C5*QD2	VAR58 = VAR57*QD2
VAR18 = VAR17*VAR3	VAR59 = VAR24*VAR39
VAR19 = Q3*CONS3	VAR60 = VAR42+VAR44
VAR20 = -S4*VAR7	VAR61 = VAR60+VAR46
VAR21 = VAR4*S5	VAR62 = VAR61+VAR15
VAR22 = -QD4*C5	VAR63 = VAR62+VAR53
VAR23 = CONS16*VAR5	VAR64 = VAR63+VAR54
VAR24 = VAR23*VAR7	VAR65 = VAR25+VAR48
VAR25 = VAR2*C5	VAR66 = VAR65+VAR56
VAR26 = CONS13*QDD3	VAR67 = VAR47+CONS12
VAR27 = VAR11+VAR18	VAR68 = VAR26+VAR58

Table 3. Continued.

VAR28 = VAR12+CONS11	VAR69 = VAR10+VAR49
VAR29 = VAR13+CONS4	VAR70 = VAR66*C5
VAR30 = -CONS18+VAR9	VAR71 = VAR67*QDD2
VAR31 = CONS14+VAR8	VAR72 = VAR69*VAR40
VAR32 = CONS18-VAR9	VAR73 = VAR59+VAR70
VAR33 = VAR16+VAR1	VAR74 = VAR73+VAR72
VAR34 = CONS14+VAR19	VAR75 = VAR51+VAR71
	END

Table 4. Generated inertia matrix model after symbolic customization (SyC)

<u>cntd.</u>	
SANJA_HU : OFF-LINE program	AFC312= AFA312-AMS33
SUBROUTINE SANJA_HU\$OFF	AFA512= AM5*VD5512
INCLUDE 'M\$SYM:SANJA_HU.CMM'	AFA522= AM5*VD5522
RRR32= RR33*RR33	AFA532= AM5*VD5532
BJYY3= AJYY3+AM3*RRR32	ANC522= OMD5522*AJYY5
AMS33= AM3*RR33	ANC532= C4*AJZZ5
AFA525= AM5*A5	AFF5412= C5*AFA512-S5*AFA522
RF535= A5*AFA525	AFF5422= S5*AFA512+C5*AFA522
ANN5535= AJZZ5+RF535	AFF4312= C4*AFF5412+S4*AFA532
H43= 0.0	AFF3312= AFF4312+AFC312
H34= 0.0	AMSV322=-AMS33*Q3
H54= 0.0	RF322= Q3*AFF3312
H45= 0.0	RF522=-A5*AFA532
H55=ANN5535	RF532= A5*AFA522
END	ANN5522= ANC522+RF522
SANJA_HU : ON-LINE program	ANN5532= ANC532+RF532
SUBROUTINE SANJA_HU\$ON	ANN5412= C5*OMD5512-S5*ANN5522
INCLUDE 'M\$SYM:SANJA_HU.CMM'	ANN5422= S5*OMD5512+C5*ANN5522
IF (F\$_1) THEN	ANN4322= S4*ANN5412-C4*ANN5532
	ANN3322= ANN4322-BJYY3+RF322+AMSV322
Min frequency	ANN2222= ANN3322-AJYY2
C4=COS(Q4)	AFA513= AM5*S5
S4=SIN(Q4)	AFA523= AM5*C5
C5=COS(Q5)	AFF5423= S5*AFA513+C5*AFA523
S5=SIN(Q5)	AFF3333= AFF5423+AM3
VD5511=-C5*S4	RF533= A5*AFA523
VD5521= S5*S4	UR534=-C5*A5
AFA511= AM5*VD5511	AFA534= AM5*UR534
AFA521= AM5*VD5521	ANC524= C5*AJYY5
AFA531= AM5*C4	RF524=-A5*AFA534
AFF5411= C5*AFA511-S5*AFA521	ANN5524= ANC524+RF524
AFF5421= S5*AFA511+C5*AFA521	ANN5424= S5*S5+C5*ANN5524
AFF4311= C4*AFF5411+S4*AFA531	H11=-AFF2221
AFF4321= S4*AFF5411-C4*AFA531	H21=-ANN3321
AFF3321= AFF4321-AM3	H31= AFF5421
AFF2221= AFF3321-AM2	H41= ANN5421
RF321= Q3*AFF4311	H51= RF531
RF521=-A5*AFA531	H12=-ANN3321
RF531= A5*AFA521	H22=-ANN2222
ANN5411=-S5*RF521	H32= AFF5422
ANN5421= C5*RF521	H42= ANN5422
ANN4321= S4*ANN5411-C4*RF531	H52= ANN5532
	H13= AFF5421

Table 4. Continued.

ANN3321= ANN4321+RF321	H23= AFF5422
OMD5512=-C5*S4	H33= AFF3333
OMD5522= S5*S4	H53= RF533
VD4412=-C4*Q3	H14= ANN5421
VD4432=-S4*Q3	H24= ANN5422
UR522= C4*A5	H44= ANN5424
UR532=-OMD5522*A5	H15= RF531
VD5512= C5*VD4412	H25= ANN5532
VD5522= UR522-S5*VD4412	H35= RF533
VD5532= UR532+VD4432	END IF
AFA312=-AM3*Q3	END

Table 5. Generated inertia matrix model of the "robot" after symbolic optimization (SyO)

ROBOT_HU_OPT : OFF-LINE program	cndt.
SUBROUTINE ROBOT_HU_OPT\$OFF	H45= 0.0
INCLUDE 'M\$SYM:ROBOT_HUOPT.CMM'	H55=ANN5535
RRR32= RR33*RR33	CONS1 = AM5+AM3+AM2
BJYY3= AJYY3+AM3*RRR32	CONS2 = -A5*AM5
AMS33= AM3*RR33	CONS3 = AJYY5+A5*AM5*A5
AFA525= AM5*A5	CONS4 = A5*AM5+AM5*A5
RF535= A5*AFA525	CONS5 = AJZZ5+A5*AM5*A5
ANN5535= AJZZ5+RF535	CONS6 = BJYY3+AJYY2
H43= 0.0	CONS7 = AM5+AM3
H34= 0.0	CONS8 = AMS33+AMS33
H54= 0.0	CONS9 = -1+AJYY5+A5*AM5*A5
	END
ROBOT_HU_OPT : ON-LINE program	cndt.
SUBROUTINE ROBOT_HU_OPT\$ON	VAR19 = VAR13+CONS6
INCLUDE 'M\$SYM:ROBOT_HUOPT.CMM'	VAR20 = -VAR11+CONS5
C4=COS(Q4)	VAR21 = VAR7-VAR10
S4=SIN(Q4)	VAR22 = CONS5-VAR11
C5=COS(Q5)	VAR23 = VAR7-VAR10
S5=SIN(Q5)	VAR24 = VAR15+VAR6
VAR1 = C5*CONS2	VAR25 = S4*C5
VAR2 = C4*VAR1	VAR26 = VAR16*S4
VAR3 = C4*C4	VAR27 = VAR26*S4
VAR4 = S5*CONS2	VAR28 = VAR18*Q3
VAR5 = C5*C5	VAR29 = VAR20*C4
VAR6 = S5*S5	VAR30 = VAR21*VAR25
VAR7 = S5*CONS9	VAR31 = VAR22*C4
VAR8 = CONS4*S5	VAR32 = VAR23*VAR25
VAR9 = S4*VAR4	VAR33 = VAR27+VAR28
VAR10 = Q3*CONS2	VAR34 = VAR33+VAR19
VAR11 = Q3*VAR4	VAR35 = -VAR9
VAR12 = CONS3*VAR6	VAR36 = -VAR2
VAR13 = VAR3*CONS5	VAR37 = -VAR2
VAR14 = Q3*CONS7	VAR38 = -VAR1
VAR15 = VAR5*CONS3	VAR39 = VAR2
VAR16 = VAR5+VAR12	VAR40 = -VAR9
VAR17 = VAR8+VAR14	VAR41 = -VAR1
VAR18 = VAR17+CONS8	END

Table 6. Computational complexity of the generated program codes for several 6-link manipulators (SyC - Customized model, SyO - Optimized model)

Robot	Model	Inverse dynamics		Inertia matrix	
		SyC	SyO	SyC	SyO
SCARA	*	121	97	169	88
	+	98	61	110	77
STANFORD	*	178	136	230	89
	+	142	112	147	71
PUMA	*	192	152	273	71
	+	150	108	175	49

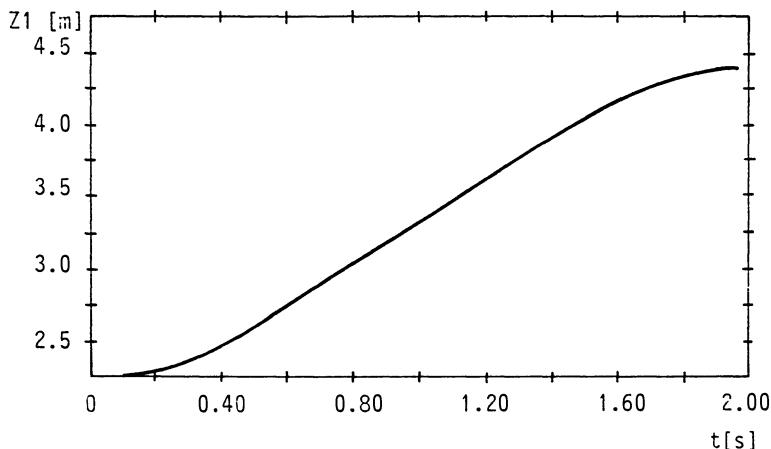


Fig. 2. Vertical sliding motion

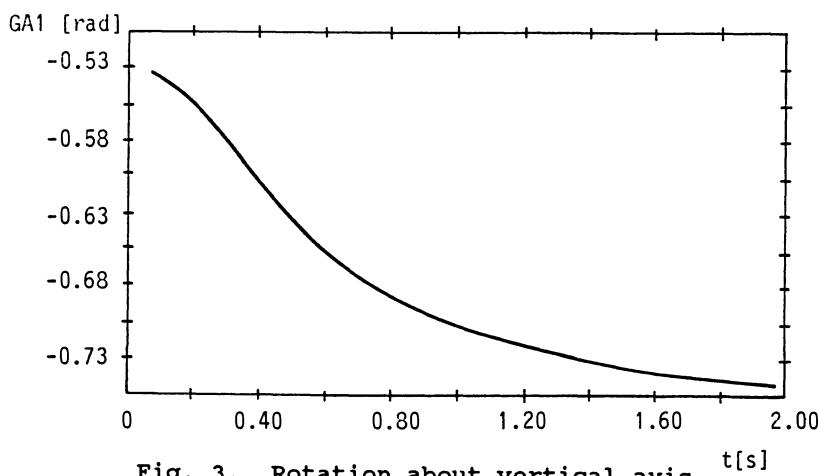


Fig. 3. Rotation about vertical axis

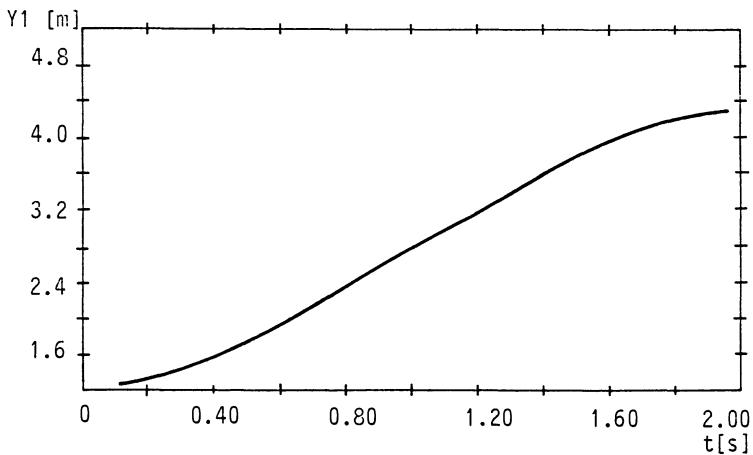


Fig. 4. Horizontal sliding

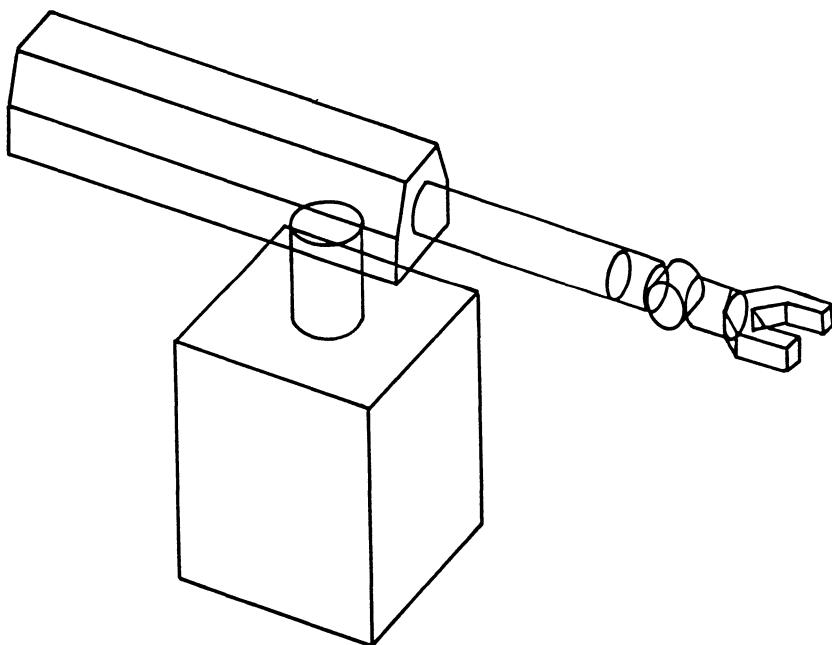


Fig. 5. Graphical presentation of the example "robot" in a configuration during motion

References

- [1] Luh J.Y.S.; Walker M.W.; Paul R.P.C.: On-line Computational Scheme for Mechanical Manipulators", ASME J. of Dyn. Sys. Meas. and Cont., 102, 69-76, 1980.
- [2] Vukobratovic M.; Stepanenko Y.: Mathematical Models of General Anthropomorphic Systems, Math. Biosci., 17, 191-242, 1973.
- [3] Stepanenko Y.: Dynamics of Spatial Mechanisms (in Russian), Mathematical Institute, Belgrade, 1974.
- [4] Stepanenko Y.; Vukobratovic M.: Dynamics of Articulated Open-Chain Active Mechanisms, Math. Biosci., 28, No. 1/2, 1976.
- [5] Paul P.R.: Robot Manipulators: Mathematics, Programming, and Control (MIT Press, Cambridge Mass., 1981).
- [6] Vukobratovic M.; Potkonjak V.: Contribution to Computer Construction of Active Chain Models via Lagrangian Form, Trans. ASME J. Appl. Mechan., 46, No. 1, 1979.
- [7] Hollerbach M.J.: A Recursive Lagrangian Formulation of Manipulator Dynamics and a comparative Study of Dynamics formulation Complexity", IEEE Trans. on SMC, SMC-10, No. 11, 730-740, 1980.
- [8] Kane T.R.; Levison D.L.: The Use of Kane's Dynamic Equations in Robotics, Int. J. Robotics Research 2, No. 3, 3-21, 1983.
- [9] Kircanski N.; Vukobratovic M.: Computer Aided Procedure of forming Robot Motion Equations in Analytical Form, Proc. VI IFTOMM Congress, New Delhi, 1983.
- [10] Vukobratovic M.; Kircanski N.: Computer Assisted Generation of Robot Dynamic Models in Analytical Form, Acta Appl. Math., Int. J. Appl. Math. and Math. Appl., 2, No. 2, 1984.
- [11] Vukobratovic M.; Kircanski N.: Real-Time Dynamics of Manipulation Robots, Series: Scientific Fundamentals of Robotics, 4, Springer-Verlag, Berlin, 1985.
- [12] Aldon M.J.; Legois A.: Computational Aspects in Robot

Dynamics Modeling, Proc. of Advanced Software in Robotics (Elsevier Science Publishers B.V., Liege, Belgium), 3-14, 1983.

- [13] Neuman P.Ch.; Murray J.J.: Computational Robot Dynamics: Foundations and Applications, J. Robotic Systems 2, No. 4, 425-452, 1985.
- [14] Renaud M.: An Efficient Iterative Analytical Procedure for Obtaining a Robot Manipulator Dynamic Model, Proc. 1st Int. Symp. of Robotics Research, Bretton Woods, New Hampshire, USA, 1983.
- [15] Khosla P.K.; Neuman P.Ch.: Computational Requirements of Customized Newton-Euler Algorithms, J. Robotic Syst. 2(3), 309-327, 1985.
- [16] Khalil W.; Kleinfinger J.F.; Gautier M.: Reducing the Computational Burden of the Dynamic Models of Robots, Proc. IEEE Int. Conf. on Robotics and Automation, San Francisco, 525-532, 1986.
- [17] Kircanski M.; Vukobratovic M.; Kircanski N.; Timcenko A.: A New Program Package for the Generation of Efficient Manipulator Kinematic and Dynamic Equations in Symbolic Form, Robotica, Vol. 6, 311-318, 1988.
- [18] Luh J.Y.S.; Lin C.S.: Scheduling of Parallel Computation for a Computer-Controlled Mechanical Manipulator, IEEE Trans. on Systems, Man, and Cyber., SMC-12, No. 2, 1982.
- [19] Kasahara H.; Narita S.: Practical Multiprocessor Scheduling Algorithms for Efficient Parallel Processing, IEEE Trans. on Computers, C-33, No. 11, 1984.
- [20] Kircanski N.; Timcenko A.; Jovanovic Z.; Vukobratovic M., Kircanski M.; Milunov R.: Computation of Customized Symbolic Robot Models on Peripheral Array Processors, Proc. IEEE Int. Conf. on Robotics and Autom., Scottsdale, Arizona, 1988.

Appendix: Newton-Euler's formalism

Notation:

I link inertia tensor about its center of mass
 m link mass
 s position vector of the center of mass in local coordinates
 z unit vector of a joint axes
 r position vector of a joint with respect to
 the previous one
 q joint coordinate
 a, a, d Denavit-Hartenberg's parameters
 w/v angular/linear velocity of a link
 F inertial and gravitational force of a link acting
 at the center of mass
 f force exerted on a link by the previous link
 N inertial moment of a link acting at the center of mass
 n moment exerted on a link by the previous one
 P driving torque in a joint

Parameters initialization:

$$\begin{aligned}
 {}^i J_{xx} &= {}^i I_{xx} + m_i ({}^i s_{iy}^2 + {}^i s_{iz}^2), & {}^i J_{xy} &= {}^i I_{xy} - m_i {}^i s_{ix} {}^i s_{iy} \\
 {}^i J_{yy} &= {}^i I_{yy} + m_i ({}^i s_{ix}^2 + {}^i s_{iz}^2), & {}^i J_{xz} &= {}^i I_{xz} - m_i {}^i s_{ix} {}^i s_{iz} \\
 {}^i J_{zz} &= {}^i I_{zz} + m_i ({}^i s_{ix}^2 + {}^i s_{iy}^2), & {}^i J_{yz} &= {}^i I_{yz} - m_i {}^i s_{iy} {}^i s_{iz} \\
 {}^i J_1 &= {}^i J_{zz} - {}^i J_{yy}, & {}^i J_1 &= {}^i J_{zz} - {}^i J_{yy}, & {}^i J_1 &= {}^i J_{zz} - {}^i J_{yy} \\
 {}^i \vec{z}_i &= [0 \ 0 \ 1]^T, & {}^i \vec{r}_i &= [a_i \ d_i \sin\alpha_i \ d_i \cos\alpha_i]^T, & {}^i \vec{s}_i &= \text{const.}
 \end{aligned}$$

Recursive Newton-Euler's algorithm:

For $i = 1, \dots, n$:

$$\begin{aligned}
 {}^{i-1} A_i &= \begin{bmatrix} \cos\alpha_i & -\sin\alpha_i \cos\theta_i & \sin\alpha_i \cos\theta_i & {}^i r_i \\ \sin\alpha_i & \cos\alpha_i \cos\theta_i & -\cos\alpha_i \sin\theta_i & 0 \\ 0 & \sin\theta_i & \cos\theta_i & 0 \end{bmatrix} \\
 {}^{i-1} \dot{\vec{w}}_i &= {}^{i-1} \dot{\vec{w}}_{i-1} + \dot{q}_i {}^{i-1} \vec{z}_{i-1}, & {}^i \vec{w}_i &= {}^{i-1} A_i^T {}^{i-1} \dot{\vec{w}}_i \\
 {}^{i-1} \dot{\vec{w}}_i &= {}^{i-1} \dot{\vec{w}}_{i-1} + (\ddot{q}_i {}^{i-1} \vec{z}_{i-1} + {}^{i-1} \vec{w}_{i-1} \times \dot{q}_i {}^{i-1} \vec{z}_{i-1}), & {}^i \dot{\vec{w}}_i &= {}^{i-1} A_i^T {}^{i-1} \dot{\vec{w}}_i \\
 U_i &= \begin{bmatrix} -({}^i w_{i3}^2 + {}^i w_{i2}^2) & {}^i w_{i1} {}^i w_{i2} - {}^i \dot{w}_{i3} & {}^i w_{i1} {}^i w_{i3} + {}^i w_{i2} \\ {}^i w_{i1} {}^i w_{i2} + {}^i \dot{w}_{i3} & -({}^i w_{i3}^2 + {}^i w_{i1}^2) & {}^i w_{i2} {}^i w_{i3} - {}^i \dot{w}_{i1} \\ {}^i w_{i1} {}^i w_{i3} - {}^i \dot{w}_{i2} & {}^i w_{i2} {}^i w_{i3} + {}^i \dot{w}_{i1} & -({}^i w_{i2}^2 + {}^i w_{i1}^2) \end{bmatrix} \\
 {}^i \dot{\vec{v}}_i &= {}^{i-1} A_i^T {}^{i-1} \dot{\vec{v}}_{i-1} + U_i {}^i \vec{f}_i
 \end{aligned}$$

$${}^i \vec{F}_i = m_i {}^i \dot{\vec{v}}_i + U_i m_i {}^i \vec{s}_i$$

For $i = n, n-1, \dots, 1$

$${}^i \vec{f}_i = {}^i \vec{f}_{i+1} + {}^i \vec{F}_i, \quad {}^{i-1} \vec{f}_i = {}^{i-1} A_i {}^i \vec{f}_i$$

$${}^i \vec{N}_{s,i} = \begin{bmatrix} {}^i w_{i1} {}^i J_{xx} + {}^i w_{i2} {}^i w_{i3} & {}^i J_{11} + {}^i u_{21} & {}^i J_{xz} - {}^i u_{31} & {}^i J_{xy} + ({}^i w_{i2}^2 - {}^i w_{i3}^2) {}^i J_{yz} \\ {}^i w_{i2} {}^i J_{yy} + {}^i w_{i3} {}^i w_{i1} & {}^i J_{22} + {}^i u_{32} & {}^i J_{xy} - {}^i u_{12} & {}^i J_{yz} + ({}^i w_{i3}^2 - {}^i w_{i1}^2) {}^i J_{xz} \\ {}^i w_{i3} {}^i J_{zz} + {}^i w_{i1} {}^i w_{i2} & {}^i J_{33} + {}^i u_{13} & {}^i J_{yz} - {}^i u_{23} & {}^i J_{xz} + ({}^i w_{i1}^2 - {}^i w_{i2}^2) {}^i J_{xy} \end{bmatrix}$$

$${}^i \vec{n}_i = {}^i \vec{n}_{i+1} + {}^i \vec{N}_{s,i} + {}^i \vec{r}_i \times {}^i \vec{f}_i + m_i {}^i \vec{s}_i \times {}^i \dot{\vec{v}}_i, \quad {}^{i-1} \vec{n}_i = {}^{i-1} A_i {}^i \vec{n}_i$$

$$P_i = {}^{i-1} \vec{n}_i {}^{i-1} \vec{z}_{i-1}$$

CAMS - A Graphical Interactive System for Computer Simulation and Design of Multibody Systems

L.Lilov, B.Bekjarov, M.Lorer, University of Sofia, Bulgaria

Abstract

CAMS is an integrated graphical interactive software system for symbolical modelling, numerical evaluation and design of technical devices which can be considered as multibody systems. No constraints are imposed on the type of the hinges or on the topological structure of the system, i.e. the system may contain an arbitrary number of bodies interconnected by hinges in a kinematic chain with an arbitrary number of closed loops. A 3D geometric modelling system DESCARTES and a system MOVIES for Movement simulation and animation are incorporated in the CAMS software system . CAMS could be applied both - to the analysis of the already existing technical systems for full utilization of their resources in a specific technological process, and on the stage of the preliminary design - for synthesis of systems with optimal characteristics. CAMS is created in several versions for IBM PC compatible, COMPAQ, PDP 11-34, PERQ II etc. Here the theoretical foundations of CAMS software are described and a brief specification of input data, main functions, output data and post processing possibilities are presented.

1. Structure of multibody systems

Any finite number of $(n+1)$ rigid bodies, interconnected in some fashion by m hinges ($m \geq n$), could be considered as a multibody system. The most typical examples of multibody systems in mechanical engineering and biomechanics are industrial robots, manipulation systems, mechanisms, vehicles, satellites, measurement devices, artificial limbs, exoskeletons etc. The same symbolism as in papers [1-3] will be used to describe the system structure. The bodies and the hinges are numbered from 0 to n and from 1 to m , respectively. Usually one body of the system plays a special role and its motion is prescribed or known in advance. This body always gets number 0. The remaining numeration is arbitrary [1]. The considered mechanical system can be represented by a graph Γ whose vertices s_i ($i=0,1,\dots,n$) and edges u_a ($a=1,\dots,m$) symbolize the bodies and the hinges respectively. This modelling comprises a description and analysis of both the interactions between the separate bodies and the structure of the kinematic constraints. If for every two contiguous bodies a reference body for the motion of the neighbouring body is chosen, then a pair of integer functions $i^+(a)$ and

$i^-(a)$ is defined. These functions give the indices of the two bodies coupled by the hinge number a , with $i^+(a)$ denoting the reference body. The graph Γ can be oriented assigning as a direction of the edge number a the direction from the vertex $i^+(a)$ to the vertex $i^-(a)$. The oriented edges are further called arcs. As it follows from the graph theory, every cyclic graph ($m > n$) can be transformed into a graph Γ with a tree structure by removing just $\hat{n} = m-n$ appropriately selected edges from the graph. We assume without loss of generality that the removed edges are u_{n+1}, \dots, u_m . The structure of Γ can be represented by the incidence matrix $\underline{I} = (S_{ia})$ where $S_{ia} = 1$ if $i = i^+(a)$, $S_{ia} = -1$ if $i = i^-(a)$ and $S_{ia} = 0$ otherwise ($i=0, 1, \dots, n$; $a=1, \dots, m$). The matrix \underline{I} can be written in the form $\underline{I} = (\underline{S}_0^T, \underline{S}^T)^T$, where $\underline{S}_0 = (S_{0a})$ ($a=1, \dots, m$), $\underline{S} = (S_{ia})$ ($i=1, \dots, n$; $a=1, \dots, m$).

A sequence of arcs u_{a_1}, \dots, u_{a_k} is said to be a path between two vertices s_i and s_j when it is possible to proceed from the vertex s_i to the vertex s_j along the sequence, so that no arc is passed twice. The paths connecting the vertex s_0 with the remaining vertices play an important role. They are used for determining the transformation matrix from the coordinate system attached to the given body to the coordinate system in the body 0 [3]. The choice of the path from s_0 to s_j further on denoted by (s_0, s_j) and called effective path is left entirely to the user. The effective path (s_0, s_i) must be chosen in such a way, so that it has to be the best path in some sense, for instance to pass the least number of arcs between s_0 and s_i or the number of the generalized coordinates in the hinges belonging to the path to be the minimal and so on. The matrix of the effective paths $\underline{\Psi} = (\Psi_{ai})$ ($a=1, \dots, m$; $i=1, \dots, n$) is introduced with elements $\Psi_{ai} = 1$ if $u_a \in (s_0, s_i)$ and is directed toward s_0 , $\Psi_{ai} = -1$ if $u_a \in (s_0, s_i)$ and is directed from s_0 and $\Psi_{ai} = 0$ otherwise [3]. The following important relationship exists between the incidence matrix \underline{I} , the effective paths matrix $\underline{\Psi}$ and the well known in the graph theory fundamental loops matrix (cyclomatic matrix) $\underline{\Phi}$:

$$\underline{\Psi}^T \underline{I}^T = (-\underline{1}_n, \underline{E}_n), \quad \underline{I} \underline{\Phi}^T = \underline{0}_{(n+1) \times \hat{n}} \quad (1), (2)$$

Here and further on \underline{E}_n will be the unit n by n matrix, $\underline{1}_n$ will be a column matrix with all elements 1 and $\underline{0}_{k \times j}$ will be a scalar k by j matrix, all elements being zero.

2. Kinematics of multibody systems

The set of all positions (linear and angular) of a given rigid free body with respect to a fixed coordinate system can be considered as a six-dimensional manifold \mathfrak{M} , defined as a Cartesian product $R^3 \times \mathfrak{X}$ of the three-dimensional Euclidean space R^3 and the manifold \mathfrak{X} of all 3×3 orthogonal matrices. In a similar way a set of all possible rigid body velocity distributions can be considered as a six-dimensional Euclidean space $\mathfrak{M}_1 = R^3 \times R^3$. If the imposed in hinge number a ($a=1, \dots, m$) constraints (scleronomic or rheonomic) are holonomic, then the manifold of all possible positions of the body $i^-(a)$ with respect to the body $i^+(a)$ is a submanifold \mathfrak{M}_a of \mathfrak{M} . This submanifold determines fully the submanifold of velocity distributions in \mathfrak{M}_1 . In the case when some of the constraints are nonholonomic, the manifold of the body $i^-(a)$ states is a submanifold of $\mathfrak{M} \times \mathfrak{M}_1$, the component in \mathfrak{M}_1 being determined by the constraints imposed on the velocities following both from the holonomic and nonholonomic constraints. Let $\underline{q}_a = (q_{a1}, \dots, q_{an})^T$ be a certain parametrization of \mathfrak{M}_a . Suitable reference frames $c_{ia} \vec{e}^{(i(a))}$, $i=i^-(a)$, are attached to the contiguous bodies for determining

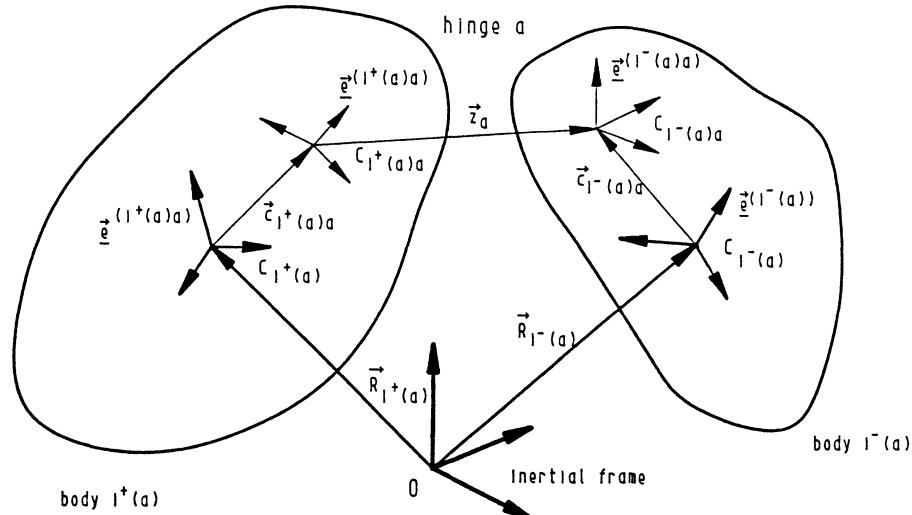


Fig.1. Vectors describing the kinematics of contiguous bodies.

the relative motion in a hinge number a (Fig.1). For instance, in the case of a spherical or revolute hinge the choice $C_i^+(a)a = C_i^-(a)a$ is preferable. The attitude of an arbitrary body i relative to the world space $O\xi\eta\xi$ is located with the radius vector \vec{R}_i of a point C_i , and with an orthonormal triad $\underline{\underline{e}}^{(i)}$ ($i=0,1,\dots,n$). The reference frames $C_i\underline{\underline{e}}^{(i)}$ and $C_{ia}\underline{\underline{e}}^{(ia)}$, $i=i^+(a)$ are denoted by $[i]$ and $[i,a]$ respectively. The angular position of $[i,a]$ relative to $[i]$ is determined by the constant matrix G_{ia} . The points C_{ia} are called hinge points, their radius vectors in $[i]$ are denoted by \vec{C}_{ia} , $i=i^+(a)$. The relative position of the contiguous bodies in the hinge number a is determined both by the transformation matrix $G_a = \underline{\underline{e}}^{(ja)} \cdot \underline{\underline{e}}^{(ka)T}$, $j=i^+(a)$, $k=i^-(a)$ between the frames $[i^+(a),a]$ and $[i^-(a),a]$ and by the hinge vector $\vec{z}_a = \vec{C}_{ja} - \vec{C}_{ka}$ (Fig.1). The quantities \vec{z}_a and G_a are functions of t and \underline{q}_a . Further on the formulas $\vec{z}_a = \vec{t}_a^T \underline{x}_a + \vec{t}_{a0}$, $\vec{\Omega}_a = \vec{p}_a^T \underline{x}_a + \vec{p}_{a0}$ take place, where \vec{z}_a and $\vec{\Omega}_a$ are respectively the relative translation and angular velocities of the frame $[i^-(a),a]$ relative to the frame $[i^+(a),a]$ and \underline{x}_a are independent nonholonomic variables (quasivelocities). In particular \underline{x}_a might be \underline{q}_a when only holonomic constraints are imposed. In the case of nonholonomic constraints the inequality $\dim \underline{x}_a < \dim \underline{q}_a$ holds. In any case the relationships $\dot{\underline{q}}_a = \dot{\underline{q}}_a(t, \underline{q}_a, \underline{x}_a)$ are known (kinematic equations).

All calculations are executed in the coordinate system [0]. The transformation matrix $A^{(i)} = \underline{\underline{e}}^{(0)} \cdot \underline{\underline{e}}^{(i)T}$ ($i=1,\dots,n$) giving the orientation of the frame $[i]$ relative to the frame [0] is specified by means of both the transformation matrix $B_a = G_i^T(a)a G_a G_i^-(a)$ between the contiguous frames $[i^-(a)]$ and $[i^+(a)]$ and the matrix of the effective paths $\underline{\Psi}$. The matrices $A^{(i)}$ have optimal for the user structure because of the extremal properties of the effective paths.

The quantities $\underline{q} = (\underline{q}_1^T, \dots, \underline{q}_m^T)^T$, $\underline{x} = (\underline{x}_1^T, \dots, \underline{x}_m^T)^T$ describing the kinematics of the relative motions are connected with the quantities describing the kinematics of the bodies in the world space $O\xi\eta\xi$ such as $\vec{R}_i, \vec{R}_j, A^{(i)}, \vec{\omega}_i$ (the absolute angular velocity of the body i) and so on by means of the structure matrices I and $\underline{\Psi}$, using (1). In particular for $\vec{R} = (\vec{R}_1, \dots, \vec{R}_n)^T$, $\vec{\omega} = (\vec{\omega}_1, \dots, \vec{\omega}_n)^T$ the formulas

$$\begin{aligned}\vec{\underline{R}} &= -\vec{\underline{\Psi}}^T \left[(\text{diag } \vec{\underline{t}} - \text{diag } \vec{\underline{p}} \vec{\underline{\Psi}} \times (\vec{\underline{c}} + \vec{\underline{c}}^*)^T) \right] \underline{x} + \dots \\ \vec{\underline{\omega}} &= -\vec{\underline{\Psi}}^T \text{diag } \vec{\underline{p}}^T \underline{x} + \dots\end{aligned}\quad (3)$$

take place, where $\vec{\underline{c}} = (s_{ia} \vec{\underline{c}}_{ia})$, $\vec{\underline{c}}^* = (s_{ia}^+ \vec{\underline{z}}_a)$ ($i = 1, \dots, n$; $a = 1, \dots, m$) ($s_{ia}^+ = 1$ if $i = i^+(a)$ and $s_{ia}^+ = 0$ otherwise). The quasi-diagonal matrix along whose principal diagonal the elements A_1, \dots, A_n of the quantity A are located, is denoted by the symbol $\text{diag } \underline{A}$. These elements can be scalars, vectors, matrices, tensors, etc. In relations (3) only terms depending on \underline{x} are given. In a similar way the quantities $\vec{\underline{R}}$, $\vec{\underline{\omega}}$ are being represented.

In the case when the multibody system includes closed loops the constraint equations are found out by means of the fundamental loops matrix $\vec{\underline{\Phi}}$ using (2). For instance the once differentiated constraint equations have the form

$$\begin{aligned}\vec{\underline{\Phi}} \left[\text{diag } \vec{\underline{t}} - \text{diag } \vec{\underline{p}} \vec{\underline{\Psi}} \times (\vec{\underline{c}} + \vec{\underline{c}}^*) \right]^T \underline{x} + \vec{\underline{\Phi}} \vec{\underline{g}} &= \vec{\underline{0}}_{n \times 1}, \\ \vec{\underline{\Phi}} \text{diag } \vec{\underline{p}}^T \underline{x} + \vec{\underline{\Phi}} \vec{\underline{p}}_0 &= \vec{\underline{0}}_{n \times 1}\end{aligned}\quad (4)$$

where $\vec{\underline{g}}$ and $\vec{\underline{p}}_0$ are certain expressions not depending on \underline{x} .

3. Geometric and kinematic characteristics of the functional capabilities of multibody systems

Let (s_0, s_i) be an arbitrary effective path. It will be considered as a single branch system and for simplicity let \underline{q} in this section be the column matrix of all generalized coordinates in the hinges belonging to the path. The body i is called end-effector. The constraints imposed on the variations of \underline{q} and $\dot{\underline{q}}$ are reduced to the conditions $\underline{q} \in Q$, $\dot{\underline{q}} \in Q_1$. In the case when the multibody system is a single loop system, as it is usually with industrial robots, then Q and Q_1 are parallelepipeds.

Let \underline{m} be an arbitrary element of the manifold \mathfrak{M} and \underline{m}_1 - an arbitrary element of the manifold \mathfrak{M}_1 . We will denote with $\alpha = (\underline{m}, \underline{m}_1)$ an arbitrary element of the manifold $\mathfrak{M} \times \mathfrak{M}_1$ of the end effector's states. The element α can be considered as an 18-dimensional column matrix the components of which are determined by the coordinates R_i of C_i in the reference frame [0], by the elements of the transformation matrix $A^{(i)}$ and by the linear \underline{v}

and angular ω velocities of the end effector relative to [0]. These quantities are expressed as functions of \underline{q} and $\dot{\underline{q}}$, i.e. a mapping $g : (Q \times Q, \rightarrow \mathbb{M} \times \mathbb{M})$, $\alpha = g(\underline{q}, \dot{\underline{q}})$ is defined.

Let $\underline{\delta}$ be an 18-dimensional column matrix with elements having values zero or unity only. The matrix $\underline{\delta}$ is denoted by the symbol $\underline{\delta} = \delta_{i_1, \dots, i_k}^{i_1, \dots, i_k}$ ($i_1 < i_2 < \dots < i_k$) as well, where i_j are the numbers of the nonzero elements of $\underline{\delta}$. Let us introduce the matrix $\Delta(\underline{\delta}) = \text{diag } \underline{\delta}$. Then in the manifold $\mathbb{M} \times \mathbb{M}$, an equivalence relation is introduced in the following way. The element $\alpha_1 \in \mathbb{M} \times \mathbb{M}$, is assumed to be equivalent to the element $\alpha_2 \in \mathbb{M} \times \mathbb{M}$, and it is denoted by $\alpha_1 \sim \alpha_2$ then and only then when $\Delta(\underline{\delta})\alpha_1 = \Delta(\underline{\delta})\alpha_2$ applies [4]. For an arbitrary subset D of the manifold $\mathbb{M} \times \mathbb{M}$, the quotient set obtained after the factorization of D with respect to the equivalence relation E will be denoted by $D/E_{\underline{\delta}}$. In this terms the *working space* of the end effector consisting of that and only that points of the reference space [0] which are reachable by the origin C_i of [i] when varying arbitrarily $\underline{q} \in Q$ can be represented in the form $Z_R = g(Q \times Q,)/E_{\underline{\delta}}^{1, 2, 3}$. In a similar way the sets $Z_A = g(Q \times Q,)/E_{\underline{\delta}}^{4, \dots, 12}$, $Z_V = g(Q \times Q,)/E_{\underline{\delta}}^{13, 14, 15}$ and $Z_{\omega} = g(Q \times Q,)/E_{\underline{\delta}}^{16, 17, 18}$ can be called *angular working space*, *space of the reachable translation velocities* and *space of the reachable angular velocities* of the end effector respectively.

One of best known qualitative characteristics of multibody systems is the so called *service coefficient* introduced in [5]. Let the point C_i be fixed in a certain point H of the working space and let all admissible positions of the unit vector $\vec{e}^{(i)}$ by the every possible $\underline{q} \in Q$ which retain $\overrightarrow{C_0 C_i}$ be considered. The end of $\vec{e}^{(i)}$ sweeps a certain region on the unit sphere. The ratio of the area of this region to the area of the unit sphere is namely the service coefficient in the point H. It characterizes the capability of the multibody system for positioning an axis, fixed at the end effector in a certain point of the working space with different orientations.

It turns out that the service coefficient can be considered as a rather particular case of the following general approach. Let B be an arbitrary subset of the phase space $(\underline{q}, \dot{\underline{q}})$ and D - an arbitrary limited subset of the manifold $\mathbb{M} \times \mathbb{M}$. The set $S(D, \underline{\delta}, B) = \{g[B \cap (Q \times Q,)] \cap D\}/E_{\underline{\delta}}$ is used as a gene-

rator of multibody system functional capacities. We introduce the dimensionless coefficient $\lambda(D, \underline{\delta}, B) = \mu_{dim(D/E_0)}^{S(D, \underline{\delta}, B)} / \mu_{dim(D/E_{\underline{\delta}})}^{(D/E_{\underline{\delta}})}$, where μ_k is a functional specifying the volume measure in R^k . The service coefficient can be derived now by setting in λ : $\underline{\delta} = \underline{\delta}_{4, 7, 10}$, $B = R^{2dimQ}$, $D = Rx\Pi_A \times \Pi_V \times \Pi_{\omega}$, where R is the representation of the vector C_H determining the fixed point H , Π_R , Π_A , Π_{ω} are arbitrarily chosen limited sets such that $\Pi_R \supset Z_R$, $\Pi_A \supset Z_A$, $\Pi_{\omega} \supset Z_{\omega}$. Another coefficient characterizing the abilities of the system to realize different translational velocities of the end effector can be introduced in the following way. Let for certain configuration $\underline{q} = \underline{q}_0$ the radius-vector of the point C_i is R_0 . Now if $D = R_0 \times \Pi_A \times \Pi_V \times \Pi_{\omega}$, $\underline{\delta} = \underline{\delta}_{13, 14, 15}$ and B is the linear manifold $(\underline{q}_0, \underline{0}) + (\underline{0} \times R^n)$, then the quantity λ is called "mobility coefficient" [5].

A second type characteristics for functional capacities of multibody systems can be introduced in the phase space $(\underline{q}, \dot{\underline{q}})$, namely $\chi(\mathfrak{M}, B) = \mu_{dimB} [B \cap \mathfrak{M}^{-1}(\mathfrak{M})]$ where \mathfrak{M} is an arbitrary subset of the manifold $\mathfrak{M} \times \mathfrak{M}$, and B is an arbitrary subset of the phase space. Substituting in χ , for instance $\mathfrak{M} = \Lambda_R \times \Pi_A \times \Pi_V \times \Pi_{\omega}$, $B = R^{2dimQ}$, where Λ_R is a subset of Z_R , the characteristic "approach coefficient for Λ_R " is obtained, which characterizes the multibody system capability for positioning the origin C_i of the end effector with different configurations \underline{q} in the points of the set Λ_R .

The suggested method for introducing quantitative estimations not only makes clear the genesis of the characteristics used at present but can be used as a basis of generation of new estimations, each one representing a specific aspect of multibody system operation, as well. This gives the possibility of selecting optimal structure of the designed multibody system on the basis of properly chosen characteristics or to determine areas in the world space where a given multibody system has the largest kinematical functional capacities.

4. Accuracy of multibody systems

In the investigation of the properties of every real multibody system an ideal model called an ideal system and denote by M is used. In the technical realization of the ideal multibody system errors inevitably appear in the producing, assembly and performance, i.e. each specific realization M^* of the ideal multibody system M is characterized by its own set of errors.

The sources of errors in an arbitrary multibody system can be divided in three basic groups. The first group of errors is due to the inaccurate performance of the control and supervise system and can be treated as errors in the generalized coordinates execution. The second group of errors appears from the inexact realization of the elements of the bodies. As a result of the inexact geometry the reference frames $[i]$ and $[i,a]$ attached to the bodies of the ideal system and determining their positions in the world space, take new positions in the real bodies and respectively new positions in the world space. The third group of errors arises from the errors in the realization of the elements of the kinematic hinges, reduced to clearances.

A mathematical apparatus is developed [6, 7] for investigation of the errors influence on the bodies linear and angular positions in the world space. For lack of space here it will be given an idea only how the influence of the clearances in the hinges can be considered. In the ideal system the vector \underline{q}_a of the generalized coordinates in hinge number a determines the relative position of the contiguous bodies $i^+(a)$ and $i^-(a)$ and the element m_a of the submanifold \mathfrak{M}_a of the manifold \mathfrak{M} is a function of \underline{q}_a , $m_a = m_a(\underline{q}_a)$. In the real system, additional degrees of freedom appear, because of the clearances, and even though \underline{q}_a is fixed the bodies $i^+(a)$ and $i^-(a)$ can still move relatively to each other. In this case the element m_a sweeps a neighbourhood $O_c(\underline{q}_a)$ defined by the boundaries of the clearances and this neighbourhood is six-dimensional. Thus, the existence of clearances yields not only to a variation $\Delta_{C,a}$ of the generalized coordinates but to additional $(6-n_a)$ degrees of freedom, as well. Using differential geometry notions a mathematical apparatus is developed for determining and computing these quantities when given clearance boundaries.

Let $\Delta \vec{R}_i$ and $\Delta \vec{\pi}_i$ be the vectors defining the linear and angular displacements of the body i in the absolute frame due to clearances in the hinges. Furthermore $\Delta \vec{R}_i$, $\Delta \vec{\pi}_i$ and $\Delta \vec{R}_i$, $\Delta \vec{\pi}_i$ are the vectors defining the displacements of the i -th body due to inexact realization of the geometry and of the generalized coordinates. Due to the independance of the three basic groups of primary errors the positioning error vector $\vec{\Delta R}_i$ and the orientation error vector $\vec{\Delta \pi}_i$ can be written taking into account only the linear terms in the form $\vec{\Delta R}_i = \vec{\Delta R}_i + \vec{\Delta R}_i + \vec{\Delta R}_i$, $\vec{\Delta \pi}_i = \vec{\Delta \pi}_i + \vec{\Delta \pi}_i + \vec{\Delta \pi}_i$. The positioning error vector determines the linear displacement of

the point C_i , as a result of an inaccurate realization of the ideal system. By analogy the vector $\Delta\vec{\pi}_i$, defines the angular displacement of the reference frame [i] in the [0] frame. The vectors $\Delta\vec{R}_i$ and $\Delta\vec{\pi}_i$ are used as main accuracy characteristics of the real multibody systems.

5. Dynamics of multibody systems

The motion equations for the multibody systems are derived from the variational principles of mechanics, while the emphasis is on the principle of the virtual power (the principle of Jourdain).

$$\sum_{i=1}^n \int (\vec{r}_i dm - \vec{dF}) \cdot \dot{\delta r}_i \equiv \delta \vec{R}^T \cdot (\text{diag } \underline{m} \cdot \vec{R} - \vec{F} - \underline{s} \vec{X}) + \delta \underline{\omega}^T \cdot \left[\text{diag } \underline{I} \cdot \vec{\omega} + \text{diag } \underline{\omega} \times \text{diag } \underline{I} \cdot \vec{\omega} - \vec{M} - \underline{s} \vec{Y} - (\vec{C} + \vec{C}^*) \times \vec{X} \right] = 0$$

for $\delta \vec{r} = \vec{0}$ ($\delta \vec{R}_i = \vec{0}$, $\delta \underline{A}^{(i)} = \underline{0}_{3 \times 3}$ $i=1, \dots, n$)

and the principle of Gauss

$$\sum_{i=1}^n \int (\vec{rdm} - \vec{dF}) \cdot \ddot{\delta r} \equiv \delta Z_2 = 0$$

with $Z_2 = 1/2 \left[\vec{R}^T \cdot \text{diag } \underline{m} \cdot \vec{R} + \vec{\omega}^T \cdot \text{diag } \underline{I} \cdot \vec{\omega} + 2\vec{\omega}^T \cdot \text{diag } \underline{\omega} \times \text{diag } \underline{I} \cdot \vec{\omega} - 2\vec{R}^T \cdot (\vec{F} + \underline{s} \vec{X}) - 2\vec{\omega}^T \cdot [\vec{M} + \underline{s} \vec{Y} + (\vec{C} + \vec{C}^*) \times \vec{X}] \right]$

for $\delta \vec{r} = \dot{\delta r} = \vec{0}$ ($\vec{R}_i = \vec{0}$, $\delta \underline{A}^{(i)} = \underline{0}_{3 \times 3}$, $\dot{\delta R}_i = \delta \vec{\omega}_i = \vec{0}$, $i=1, \dots, n$).

Here \vec{r} is the radius vector of the differential element of mass in the world (inertial) space $O\xi\eta\zeta$, m_i and \underline{I}_i are the total mass and the central inertia tensor of body i respectively, $\underline{F} = (\vec{F}_1, \dots, \vec{F}_n)^T$, $\vec{M} = (\vec{M}_1, \dots, \vec{M}_n)^T$, \vec{F}_i and \vec{M}_i are the vector and the moment of the external forces acting on body i and applied in the mass center C_i , $\vec{X} = (\vec{x}_1, \dots, \vec{x}_m)^T$, $\vec{Y} = (\vec{y}_1, \dots, \vec{y}_m)^T$, \vec{x}_a and \vec{y}_a are respectively the resultant force and moment applied on body $i^+(a)$ by body $i^-(a)$ and acting at the hinge point $C_{i^-(a)a}$. According to (3) $\dot{\delta R} = -\underline{\Psi}^T [(\text{diag } \underline{t} - \text{diag } \underline{p}\Psi \times (\vec{C} + \vec{C}^*))^T \delta \underline{x}$, $\delta \vec{\omega} = -\underline{\Psi}^T \text{diag } \underline{p}^T \delta \underline{x}$. Substituting all quantities in the variational principles (5), (6) and in the constraint equations (4) by their expressions as functions of t , \underline{q} , \underline{x} we can represent them in the form

$$\delta \dot{\underline{x}} \cdot [\underline{A}(\underline{t}, \underline{q}) \dot{\underline{x}} - \underline{b}(\underline{t}, \underline{q}, \dot{\underline{x}})] = 0 \quad \underline{K}(\underline{t}, \underline{q}) \delta \dot{\underline{x}} = \underline{0} \quad \underline{Kx} = \underline{k}(\underline{t}, \underline{q}) \quad (7)$$

for the principle of virtual power and

$$\frac{1}{2} \dot{\underline{x}}^T \underline{A} \dot{\underline{x}} - \dot{\underline{x}}^T \underline{b} \longrightarrow \min \quad \text{for } \dot{\underline{x}} = \underline{k}_1(\underline{t}, \underline{q}, \dot{\underline{x}}) \quad (8)$$

for the principle of Gauss [8]. The motion equations can be found from (7) by determining that submatrix of the matrix \underline{K} which is a carrier of its rank. These equations provide with the kinematical equation $\dot{\underline{q}} = \underline{q}(\underline{t}, \underline{q}, \dot{\underline{x}})$ a complete system for the variables \underline{q} and $\dot{\underline{x}}$. In the numerical integration of the motion equations it is preferable to use the principle of Gauss which reduce the determining of the accelerations to a minimization of a square functional under linear constraints. As a functional can serve the form (8) but also the form (6). The last form is used in the package CAMS parallel with the first form for multibody systems with one degree of freedom hinges such as industrial robots and manipulators. In this case instead of constraint equations (4) conditions for $\dot{\underline{R}}_i$ and $\dot{\underline{\omega}}_i$, for each hinge are formulated [8]. After fulfilment of the minimization procedure and the integration of $\dot{\underline{R}}_i$ and $\dot{\underline{\omega}}_i$ we determine the multibody system motion direct in the world space. When necessary the generalized coordinates \underline{q} can be found from the constraint equations in the hinges.

6. General description of the CAMS software system

CAMS is an integrated graphical interactive software system for symbolical modelling, numerical evaluation and design of :

- industrial robots and manipulators;
- mechanisms and machines;
- vehicle systems;
- measurement devices;
- satellites

and other technical systems which can be considered as multibody systems. No constraints are imposed on the type of the hinges or on the topological structure of the system, i.e. the system may contain an arbitrary number of bodies interconnected by hinges in a kinematic chain with an arbitrary number of closed loops.

The geometry of the bodies is described by a 3D geometric modelling system DESCARTES incorporated in CAMS software system. The movements in the 3D

space of the considered technical systems are simulated using the software system MOVIES (MOVements Investigation Estimation and Simulation).

CAMS could be applied both - in the analysis of already existing technical systems for full utilization of their resources in a specific technological process, or on the stage of preliminary design - for synthesis of systems with optimal characteristics. CAMS is created in several versions for IBM PC compatible, COMPAQ, PDP 11-34, PERQ II etc.

CAMS structure

CAMS is designed on the module principle with four-level hierarchical structure. The general functional scheme of the CAMS system is shown on Fig.2.

First level modules - data input

The first level modules organize the dialogue with the user. CAMS requires a minimal set of input data, which identifies the concrete mechanical system , such as structure, kinematical scheme, geometrical parameters characterizing the bodies, forces etc. It is possible to enter data in an interactive mode or using an arbitrary text-editor. Interaction with CAMS is organized with the help of DESCARTES 3D modelling system. In terms of DESCARTES the user describes the geometry of the bodies and the quantities attached to them using:

- arbitrary 2D curves;
- splines and Bezier curves;
- surfaces - planar, cylindrical, conical, ruled and lofting;
- 3D solids - polyhedrals, spheres, ellipsoids, cilinders etc.
- primitives of the type "frame";
- different views;
- data manipulation options - copy, transform, drag, delete etc.

After that all mass and inertia characteristics of the bodies are calculated automatically.

CAMS offers powerful techniques for describing the external loading, internal forces and torques e.g. it is possible to enter them graphically or to use analytical expressions including standart computer available functions - sin, cos, exp, atan etc. CAMS also supportes some databases with standart most commonly used hinges, bodies, kinematical loops, mechanisms.

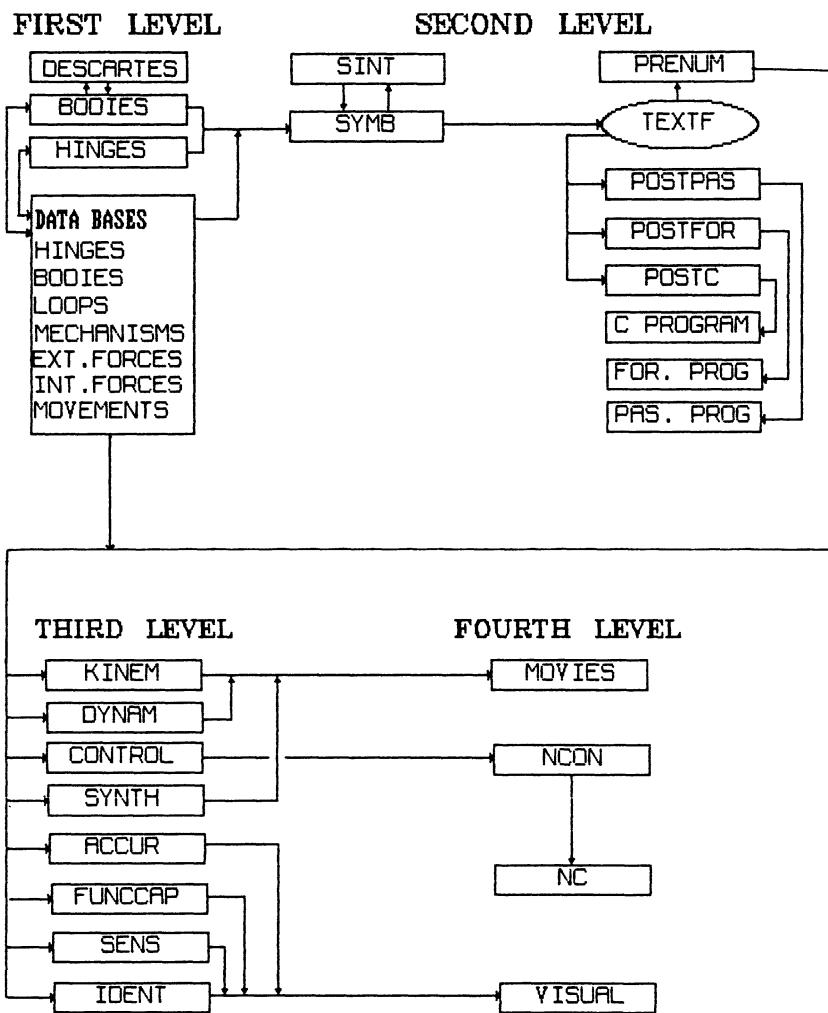


Fig.2. Principle scheme of the CAMS software system

Second level modules - symbolical work

The system of programme modules SYMB for symbolical or alpha-numerical modelling of multibody system kinematics,dynamics and accuracy operates on the second level. All modules are based on a specialy created language for analytical computations SINT(Symbolical Interpretation). SINT is a virtual language with syntax and functional abilities similar to those of the REDUCE language. This gives the possibilty to run all modules written in SINT in the specific REDUCE enviroment and vice versa.The main advantage of SINT is the powerfull memory managment, based on overlay structures, virtual disk and buffer data transfer. Thus, analytical expressions with different complexity could be processed by personal computers like IBM -PC/AT.

All results obtained by the system SYMB are stored in a text file TEXTF. The symbolical information stored in such file is ready for direct use by the third level modules for numerical analysis and design. Neither compilation nor linking is needed after mathematical modell is symbolically created.

Furthermore the postprocessors POSTFOR, POSTPAS and POSTC give the possibility for an output code generation of a programme in FORTRAN, PASCAL or C languages.

Third level modules - numerical calculations

The third level modules realize the numerical modelling, investigation and design of a multibody system. These modules solve the following fundamental problems,which appear in the process of evaluation and design of technical systems:

- determination of the degrees of freedom and automatic selection of the generalised coordinates (module KINEM);
- kinematical analysis - determination of position, linear and angular velocity and acceleration of an arbitrary body (module KINEM);
- inverse kinematical problem - determination of the generalized coordinates, velocities and accelerations, realizing a prescribed motion of a given body (module KINEM);
- movement planning with obstacle avoidance (module CONTROL)
- direct dynamical problem - computation of the generalized coordinates, velocities and accelerations, hinge reaction forces and torques in confirmity with given external loading, internal forces and torques (module DYNAM);

- inverse dynamical problem - computation of the actuators forces and torques in confirmity with external loading and prescribed motion of a given body (module DYNAM);
- actuators test - estimation of the actuators capabilities to realize the motion of a given body with prescribed linear velocity (module DYNAM);
- computation and visualization of different geometrical, kinematical and dynamical characteristics , for instance approach and mobility coefficient, service coefficient, general inertia coefficient etc (module FUNCCAP);
- probabilistic as well as deterministic analysis of multibody systems accuracy. All important for the practice error sources are taken into account: (i) errors in the values of generalized coordinates, (ii) errors due to clearances in joints, (iii) errors resulting from inaccurate realizations of parameters specifying the geometry of bodies (module ACCUR);
- probabilistic as well as deterministic error-sensitivity analysis (module SENS);
- identification and compensation of error sources (module IDENT);
- optimal design - determination of optimal system parameters with respect to a given goal function reflecting the kinematics, dynamics or accuracy of the system (module SYNTH);

Fourth level modules - output data (movement simulation , postprocessing)

The results of the modules KINEM, DYNAM, CONTROL and SYNT can be illustrated by the software system MOVIES (MOVements Investigation Estimation and Simulation). MOVIES offers the user an 3D animation of arbitrary movement of the considered technical system with elements of realistic graphics :

- orthogonal, isometric and perspective views;
- hidden lines and surfaces removal;
- lighting;
- colour shading etc.

The results of the modules KINEM, DYNAM, FUNCCAP, ACCUR, IDENT, SENS and SYNT can be illustrated using the graphical program VISUAL for visualization of 2D scalar fields, isolines, graphics of function, histogramme, 3D axonometric view of a scalar field behaviour etc. For

instance by means of this module the distribution of geometrical, kinematical and other scalar characteristics in a given intersection of the working space with a fixed plane in 3D space could be illustrated. The visualization is performed comparing a fixed black&white shade to any value of the characteristic in the nodes of a square set covering the considered intersection. The discrete color change from white to black corresponds to the increments of the scalar characteristic values. In this way an adequate black&white image of the scalar field is obtained.

It is possible by means of DESCARTES 3D geometric modelling system to prepare the technical documentation for considered multibody systems.

The NCCON module allows the user to connect the software system CAMS with different devices for numerical control of robots, machines etc.. Thus CAMS could be used as part of an intelligent station for off-line control.

7. Examples

CAMS software system as well as other systems mentioned above - DESCARTES, MOVIES, SYMB, SINT, KINEM, ACCUR, DYNAM, VISUAL etc. are used for investigation and design of industrial robots and manipulation systems, vehicles, electric- and motor cars and at the study of human abilities in some sports like weight lifting and basketball. Several results of the module VISUAL work are presented on Fig. 3 , Fig.4 and Fig. 5.

For the test example of the seven body planar mechanism defined in this handbook the equations of motion are generated symbolically by CAMS software system. The diagonal elements of the matrix A (Eqs.7) are :

```

A[1,1]:=MAS[1]*(GM[1,1][1]*GM[1,1][1]+GM[1,1][2]*GM[1,1][2])+  

    MAS[2]*(GM[1,2][1]*GM[1,2][1]+GM[1,2][2]*GM[1,2][2])+  

    IM[1,3,3]+IM[2,3,3];  

A[2,2]:=MAS[2]*(GM[2,2][1]*GM[2,2][1]+GM[2,2][2]*GM[2,2][2])+  

    IM[2,3,3];  

A[3,3]:=MAS[3]*(GM[3,3][1]*GM[3,3][1]+GM[3,3][2]*GM[3,3][2])+  

    IM[3,3,3];  

A[4,4]:=MAS[4]*(GM[4,4][1]*GM[4,4][1]+GM[4,4][2]*GM[4,4][2])+  

    MAS[5]*(GM[4,5][1]*GM[4,5][1]+GM[4,5][2]*GM[4,5][2])+  

    IM[4,3,3]+IM[5,3,3];  

A[5,5]:=MAS[5]*(GM[5,5][1]*GM[5,5][1]+GM[5,5][2]*GM[5,5][2])+  

    IM[5,3,3];  

A[6,6]:=MAS[6]*(GM[6,6][1]*GM[6,6][1]+GM[6,6][2]*GM[6,6][2])+  

    MAS[5]*(GM[6,7][1]*GM[6,7][1]+GM[6,7][2]*GM[6,7][2])+  

    IM[6,3,3]+IM[7,3,3];  

A[7,7]:=MAS[7]*(GM[7,7][1]*GM[7,7][1]+GM[7,7][2]*GM[7,7][2])+  

    IM[7,3,3];

```

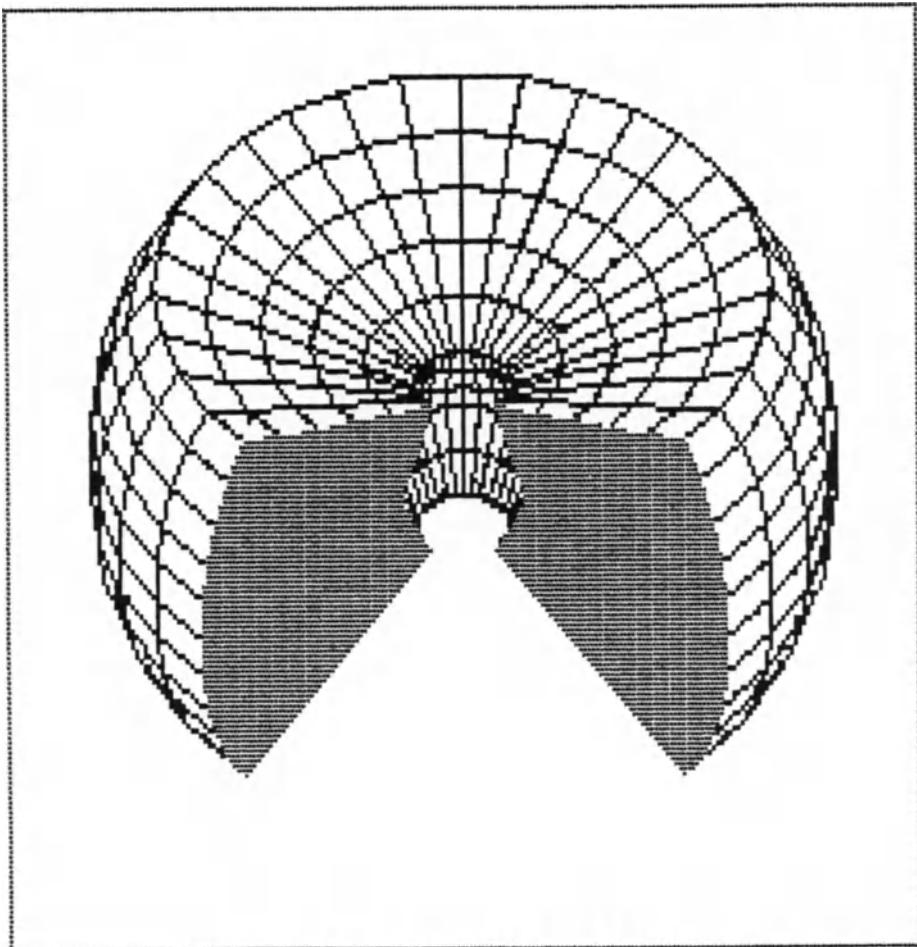


Fig.3. Working space for the robot UNIMATE 2030.

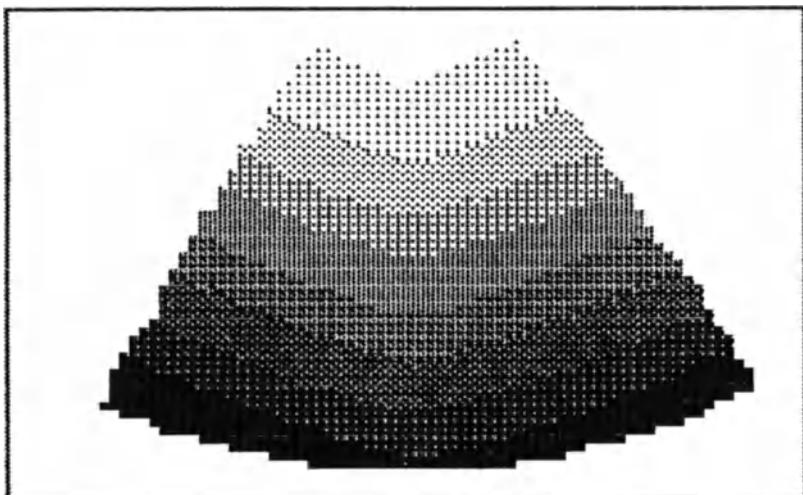


Fig.4 Mobility coefficient in a given intersection of a vertical plane with the working space of UNIMATE 2030.

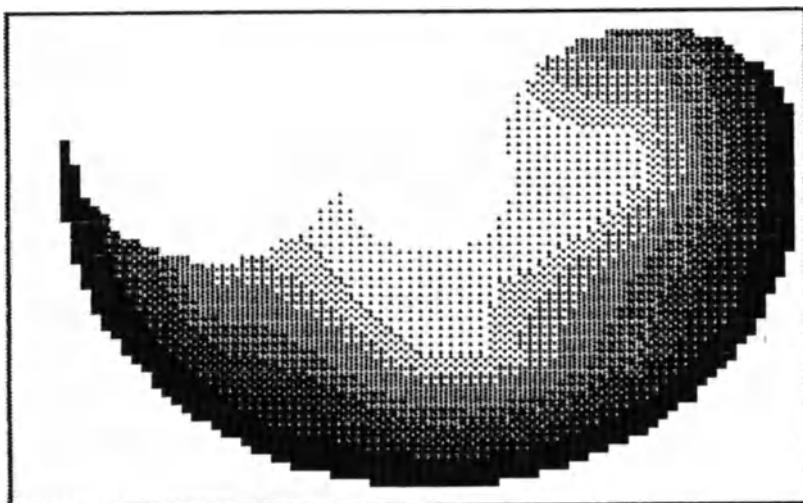


Fig.5. Visualization of the maximal positioning error distribution in the working space of the robot SCARA.

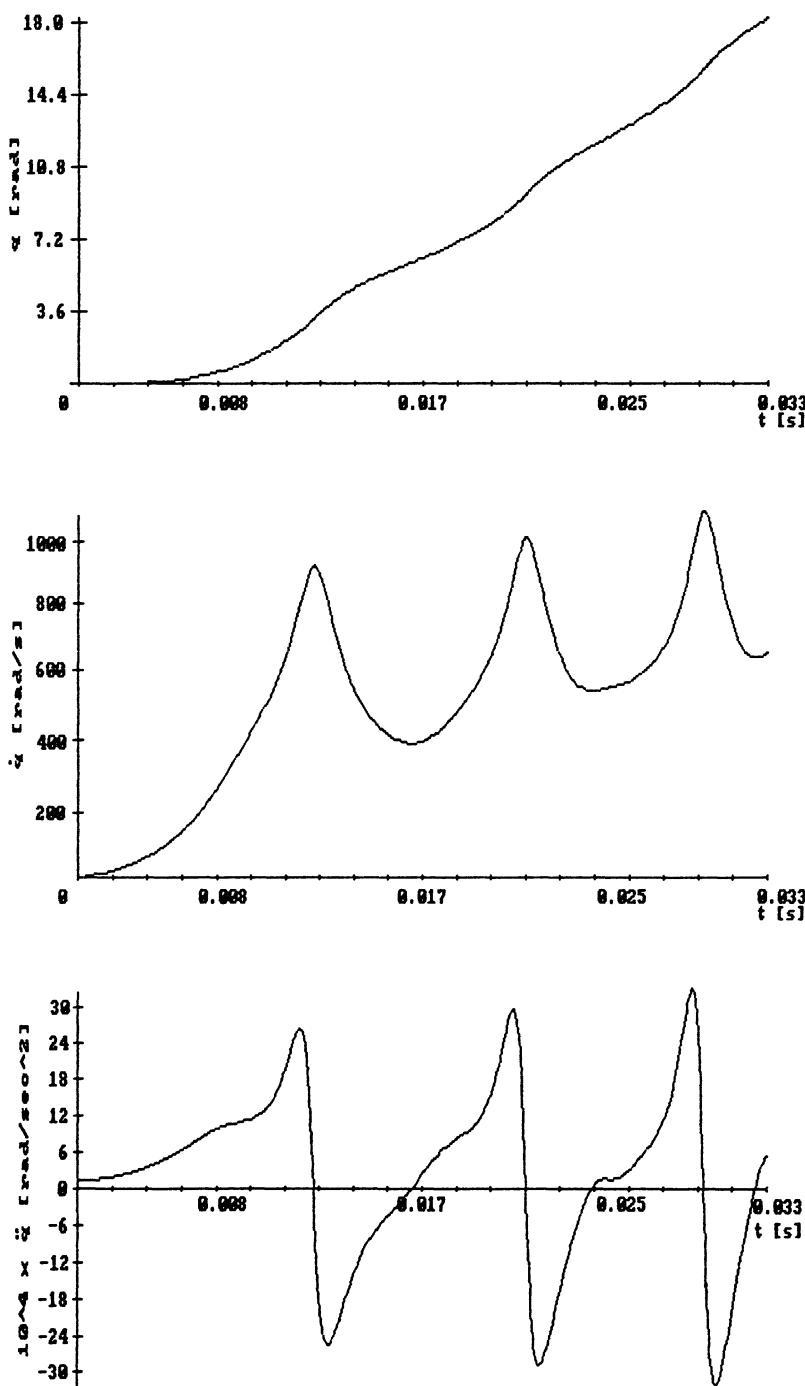


Fig.6. Time history of the seven body mechanism motion

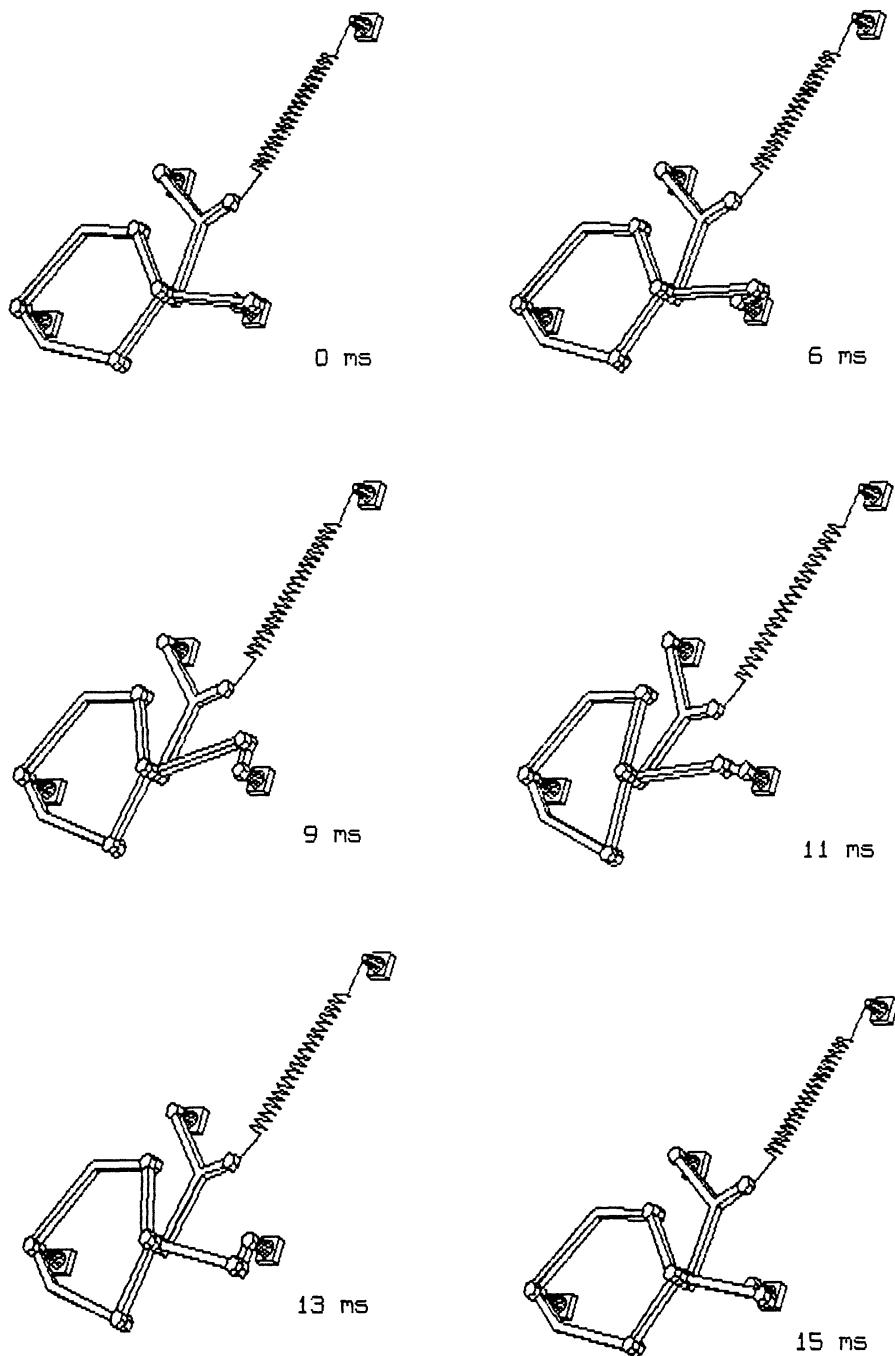


Fig.7. Motion animation of the seven body mechanism

The symbolical equations are solved by numerical integration and the results are graphically illustrated on Fig.6. The animation of the corresponding mechanism's motion is shown on Fig.7.

REFERENCES

- [1] *Roberson, R.; 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 3-rd IFAC Congress, London, 1966. London: Butterworth and Co., Ltd., 1967.
- [2] *Lilov, L.; Wittenburg, J.*: Equations of Motion for Systems of Rigid Bodies with arbitrary Hinges (in german). Z. f Angew. Math und Mech. (ZAMM), 57 (1977), H.3.
- [3] *Lilov, L.*: Structure, Kinematics and Dynamics of Systems of Rigid Bodies (in russian). Advances in Mechanics, 6 (1983), №.1/2.
- [4] *Lilov, L.; Bekjarov, B.*: Geometrical and Kinematical Qualitative Characteristics for Functional Capacities of Manipulation Systems. Proceedings of the 5-th CISM-IFToMM Symposium ROMANSY. Kogan Page, London, Hermes Publications.
- [5] *Kobrinskii A.*: On Mechanical Qualities of Manipulation Systems (in russian). DAN - USSR, 241 (1978), №. 4.
- [6] *Lilov, L.; Bekjarov, B.*: Accuracy of Multibody Systems with Tree-like Structure and Arbitrary Joints (in russian). Theoretical and Applied Mechanics, 14 (1983), №. 1.
- [7] *Lilov, L.; Bekjarov, B.*: Accuracy of Multibody Systems. Mechanics of Structure and Machines, 17 (1989), №.2.
- [8] *Lilov, L.; Lorer, M.*: Dynamic Analysis of Multirigid-Body System Based on the Gauss Principle. ZAMM, 62 (1982), H.10.

AUTOLEV - A New Approach to Multibody Dynamics

David A. Levinson
Lockheed Palo Alto Research Laboratory
92-30/250, 3251 Hanover Street
Palo Alto, California 94304, USA

and

Thomas R. Kane
Department of Mechanical Engineering
Stanford University
Stanford, California 94305, USA

Introduction

AUTOLEV, an interactive symbolic dynamics program based on the method set forth in [1] for formulating equations of motion, differs fundamentally from other multibody dynamics programs in that it gives the user step-by-step control of the equation formulation process. Its unstructured format places virtually no restrictions on the types of dynamical systems that it accommodates, so that one can deal equally easily with one, two, and three dimensional holonomic and nonholonomic systems, closed loops, moving constraints, etc. Moreover, the process of formulating equations of motion is unencumbered by the computer memory limitations and slow run times associated with classical methods of mechanics, and the program thus can be used on a desktop computer.

AUTOLEV produces complete, fully formatted, ready-to-compile-and-run FORTRAN simulation programs in which, to minimize computation time, repeated strings of symbols have been replaced automatically with new, individual symbols. In addition, each program can incorporate provisions for checking the correctness of the equations of motion under consideration.

To explain how one works with AUTOLEV to produce multibody simulations, we shall describe in detail the process employed to generate response curves for the linkage system shown in Figure 1. Readers interested in learning more about AUTOLEV are referred to [2].

Illustrative Examples

We begin by typing

```
DOF(1,7)
```

This tells AUTOLEV that the system to be analyzed possess one degree of freedom, and that its motion is characterized by seven generalized speeds (see [1], p. 40), which AUTOLEV names internally as U1, . . . , U7. Next we tell AUTOLEV the names of all the reference frames (rigid bodies) making up the linkage (see Figure 1):

```
FRAMES(K1,K2,K3,K4,K5,K6,K7)
```

For each frame listed in the FRAMES command, AUTOLEV assigns names to various items of interest associated with the frame. For example, a dextral set of mutually perpendicular unit vectors K11, K12, and K13 is designated as fixed in K1, and the mass center of K1 is denoted K1STAR. Similar names are automatically introduced in connection with K2, . . . , K7. A Newtonian reference frame N is defined internally when AUTOLEV is activated, and a dextral set of mutually perpendicular unit vectors N1, N2, and N3 is fixed in N.

The points of interest on the linkage are listed in a POINTS command:

```
POINTS(0,A,B,C,P1,P22,P23,P24,P26,P3,P4,P5)
```

Points 0, A, B, C, P1, P3, P4, and P5 can be found in Figure 1, and P2i denotes that point of Ki located at P2 (i = 2, 3, 4, 6) (see Figure 1 for P2). Since these points are identified for purely kinematical purposes, but do not correspond to particles, we tell this to AUTOLEV by typing

```
MASSLESS(0,A,B,C,P1,P22,P23,P24,P26,P3,P4,P5)
```

The given angles β , γ , δ , and ϵ (see Figure 1) are designated as variables, as are θ_2 , θ_4 , and θ_6 , angles characterizing the orientations of K2, K4, and K6, respectively, in N:

```
VAR(BETA,GAMMA,DELTA,EPSILON,THETA2,THETA4,THETA6)
```

Similarly, given constants are designated in CONST commands. The distances XA and YA are not listed in a CONST command, because these quantities are not needed in formulating equations of motion. SIGMA denotes the spring constant, L0 the natural length of the spring, and TORQK1 is the (constant) value of the K13 measure number of the torque of the driving couple. We type

```
CONST(D,DA,E,EA,ZF,FA,R,RA,S,SA,SB,SC,SD,ZT,TA,TB,UX,UA,UB)
CONST(XB,YB,XC,YC,SIGMA,LO,TORQK1)
```

The MASS command is used to assign names to the masses of K1, . . . , K7. Here, the mass of Ki is denoted as Mi (i = 1, . . . , 7):

```
MASS(K1,M1,K2,M2,K3,M3,K4,M4,K5,M5,K6,M6,K7,M7)
```

With the INERTIA command we assign the name I1 to the moment of inertia of K1 about a line passing through K1STAR and parallel to K13:

```
INERTIA(K1,0,0,I1,0,0,0)
```

and similar commands are used to assign the name Ij to the moment of inertia of Kj about a line passing through KjSTAR and parallel to Kj3 (j = 2, . . . , 7).

Moving on to kinematical considerations, we define U1 by expressing WK1N, the angular velocity of K1 in N, as

```
WK1N=U1*K13
```

Similarly, we set

```
WK2N=U2*K23
```

and so forth.

Kinematical equations relating time-derivatives of linkage orientation angles to generalized speeds are written as

```
BETA'=U1
GAMMA'=U3
DELTA'=U5
EPSILON'=U7
THETA2'=U2
THETA4'=U4
THETA6'=U6
```

At this juncture, we invoke the SIMPROT command to obtain the direction cosines associated with the simple rotation, in N, of K1 in the K13 (also N3) direction through the angle β :

```
SIMPROT(N,K1,3,BETA)
```

In response to this, rather than prompting us to input another line, AUTOLEV returns

-> (33) DIRCOS(N,K1,COS(BETA),-SIN(BETA),0,SIN(BETA),COS(BETA),0,0,0,1)

The arrow to the left of the line number indicates that this is an AUTOLEV response, rather than a user input. The nine quantities following K1 in the output are the direction cosines N1 · K11, N1 · K12, . . . , N3 · K13. After proceeding similarly in connection with K2, . . . , K7, we form the direction cosines relating K11, K12, and K13 to K21, K22, and K23 by employing a DIRCOS command:

DIRCOS(K1,K2)

to which AUTOLEV responds with

-> (47) DIRCOS(K1,K2,COS(BETA-THETA2),SIN(BETA-THETA2),0,-SIN(BETA-THETA2),COS(BETA-THETA2),0,0,0,1)

The direction cosines relating K41, K42, and K43 to K51, K52, and K53, as well as those relating K61, K62, and K63 to K71, K72, and K73 are determined similarly.

Next, referring to the drawing provided by Professor Schiehlen showing the geometric parameters, we form POK1STAR, the position vector from 0 to K1STAR as

POK1STAR=RA*K11

and POP1, the position vector from 0 to P1 as

POP1=R*K11

After informing AUTOLEV that VON, the velocity of 0 in N, is equal to zero,

VON=0

we invoke the V2PTS command, which employs the theorem

$$\underline{F_v}^P = \underline{F_v}^Q + \underline{F_\omega}^D \times \underline{p}^{QP}$$

where F is a reference frame, D is a rigid body, P and Q are points fixed in D, $\underline{F_v}^P$ is the velocity of P in F, $\underline{F_v}^Q$ is the velocity of Q in F, $\underline{F_\omega}^D$ is the angular velocity of D in F, and \underline{p}^{QP} is the position vector from Q to P. The associated AUTOLEV command is written V2PTS(F,D,Q,P). For the case at hand, we type

V2PTS(N,K1,0,K1STAR)

which yields VK1STAR, the velocity of K1STAR in N, as

-> (56) $VK1STAR = RA * U1 * K12$

and

$V2PTS(N, K1, 0, P1)$

which gives $VP1N$, the velocity of $P1$ in N , as

-> (58) $VP1N = R * U1 * K12$

After forming $PP1K2STAR$ and $PP1P22$, the position vectors from $P1$ to $K2STAR$ and $P22$, respectively, as

$PP1K2STAR = -DA * K21$

$PP1P22 = -D * K21$

in preparation for obtaining expressions for $VK2STAR$ and $VP22N$, the velocities of $K2STAR$ and $P22$ in N , we once again employ the $V2PTS$ command

$V2PTS(N, K2, P1, K2STAR)$

which gives

-> (62) $VK2STAR = R * U1 * K12 - DA * U2 * K22$

Before invoking $V2PTS$ again for the purpose of obtaining $VP22N$, we command AUTOLEV to perform some simplifications by expressing $VK2STAR$ solely in the $K21$, $K22$, $K23$ basis:

$EXPRESS(VK2STAR, K2)$

Here AUTOLEV defines two new quantities, $Z1$ and $Z2$,

-> (64) $Z1 = R * SIN(BETA - THETA2)$

-> (65) $Z2 = COS(BETA - THETA2) * R$

which permits it to write

-> (66) $VK2STAR = -U1 * Z1 * K21 + (-DA * U2 + U1 * Z2) * K22$

AUTOLEV uses the Z 's as a means for simplifying expressions. Each Z stands for a collection of symbols, and the Z is used in subsequent computations in place of the collection. This process eliminates unnecessary repetition of operations. In a similar fashion, we obtain $VP22N$, as well as the velocities in N of all of the remaining link mass centers and points of links meeting at $P2$.

The velocities in N of all link points meeting at $P2$ are equal to each other. Since four links meet at $P2$, there are three independent vector equations that must be satisfied to ensure that the equal velocity

conditions are met, and these equations can be expressed as $VP22N - VP23N = VP22N - VP24N = VP22N - VP26N = 0$. To enable AUTOLEV to make use of this fact, we form three vectors, MATCH23, MATCH24, and MATCH26, as

```
MATCH23=ADD(VP22N,-VP23N)
```

which leads to

```
-> (118) MATCH23=-U1*Z1*K21+(-D*U2+U1*Z2)*K22-S*U3*K31
```

and so forth. From these three vectors, one can form six scalar constraint equations relating U_2, \dots, U_7 to U_1 . In preparation for this we form three sets of direction cosines by typing

```
DIRCOS(K2,K3)
```

```
-> (124) DIRCOS(K2,K3,COS(GAMMA-THETA2),-SIN(GAMMA-THETA2),0,SIN  
(GAMMA-THETA2),COS(GAMMA-THETA2),0,0,0,1)
```

and similarly for K_2 and K_4 , and K_4 and K_6 . Then we form the scalars CONSTRAINT1 and CONSTRAINT2 by dot-multiplying MATCH23 with K21 and K22:

```
CONSTRAINT1=DOT(MATCH23,K21)
```

```
-> (130) Z9=COS(GAMMA-THETA2)*S
```

```
-> (131) CONSTRAINT1=-Z1*U1-Z9*U3
```

```
CONSTRAINT2=DOT(MATCH23,K22)
```

```
-> (133) Z10=S*SIN(GAMMA-THETA2)
```

```
-> (134) CONSTRAINT2=Z2*U1-D*U2-Z10*U3
```

Similarly, we obtain CONSTRAINT3, . . . , CONSTRAINT6 by dot-multiplying MATCH24 and MATCH26 with K21 and K22. Now we invoke the CONSTRAIN command, which sets each of CONSTRAINT i ($i = 1, \dots, 6$) equal to zero and then solves the six resulting equations for U_2, \dots, U_7 as functions of U_1 ; that is, we type

```
CONSTRAIN
```

which leads to the creation of Z 's and the desired equations,

```
-> (171) U2=Z29*U1
```

```
.
```

```
.
```

```
.
```

```
-> (176) U7=Z37*U1
```

The CONSTRAIN command continues to work, substituting for U_2, \dots, U_7

from lines (171) - (176) into all equations containing these U's. This gives

-> (177) $WK2N=U1*Z29*K23$

and so on.

Now we proceed with the formulation of angular accelerations by employing the DERIV command. ALFK1N, the angular acceleration of K1 in N, is obtained by differentiating W1KN, the angular velocity of K1 in N, with respect to the time T, in reference frame N:

$ALFK1N=DERIV(WK1N,T,N)$

-> (211) $ALFK1N=U1'*K13$

The remaining six angular accelerations are obtained similarly.

Turning to the formulation of accelerations, we obtain AK1STARN, the acceleration of K1STAR in N, by differentiating VK1STARN, the velocity of K1STAR in N, with respect to T in N:

$AK1STARN=DERIV(VK1STARN,T,N)$

-> (272) $Z85=RA*U1$

.

-> (275) $AK1STARN=-Z86*K11+RA*U1'*K12$

The remaining six mass center accelerations are determined similarly.

We are now in a position to formulate the generalized inertia force (see [1], pp. 124-125) for the linkage. To do this, we type

FRSTAR

which, after generating still more Z's, yields

-> (362) $F1STAR=Z139*U1'-Z140$

We proceed to the determination of the generalized active force (see [1], pp. 99, 106) for the linkage, starting by considering the forces applied to the linkage by the driving motor. We assume that the system of forces in question is equivalent to a couple applied to K1, and that the torque of this couple is given by

$TORQUE(K1)=TORQ*K13$

The scalar quantity TORQ is listed in a CONTROLS command,

CONTROLS(TORQ)

which will insert TORQ into a subroutine when the FORTRAN program is generated, where TORQ will be defined as

```
TORQ=TORQK1
```

with TORQK1 a constant. Having defined TORQ in this manner, we can have the best of two worlds in connection with the FORTRAN simulation program. For the problem at hand, the constant value of TORQK1 will be read from a data file and assigned to TORQK1, and then, in turn, to TORQ. When TORQ is not a constant, we can edit the aforementioned subroutine, replacing TORQK1, on the right-hand side of this equation, with any desired function.

To take into account the spring force, we must first determine the position vector PP5C from P5 to C (see Figure 1). To this end, we input the position vectors POB and POC, from O to B and C, respectively,

```
POB=XB*N1+YB*N2
```

```
POC=XC*N1+YC*N2
```

and then use the ADD command to form the position vector PBC from B to C:

```
PBC=ADD(POC,-POB)
```

```
-> (369) PBC=(-XB+XC)*N1+(-YB+YC)*N2
```

while the position vector PBP5 from B to P5 is given by

```
PBP5=SD*K31-SC*K32
```

so that the ADD command can be used to obtain PP5C, the desired result, as

```
PP5C=ADD(PBC,-PBP5)
```

```
-> (372) PP5C=(-XB+XC)*N1+(-YB+YC)*N2-SD*K31+SC*K32
```

The determination of the magnitude of PP5C is accomplished by dot-multiplying PP5C with itself, which leads to the square of the magnitude of PP5C,

```
MAGPP5C2=DOT(PP5C,PP5C)
```

```
-> (374) MAGPP5C2=(-XB+XC)*(-XB+XC)-2*(-XB+XC)*COS(GAMMA)*SD-2*(-  
-XB+XC)*SC*SIN(GAMMA)+(-YB+YC)*(-YB+YC)+2*(-YB+YC)*COS(GAMMA)  
*SC-2*(-YB+YC)*SD*SIN(GAMMA)+SC*SC+SD*SD
```

whereupon the RIGHT command is used to raise the right-hand side of the equation in line (374) to the .5 power, yielding the magnitude of PP5C, to which we give the name MAGPP5C:

```
MAGPP5C=RIGHT(MAGPP5C2)^.5
```

```
-> (376) MAGPP5C=((-XB+XC)*(-XB+XC)-2*(-XB+XC)*COS(GAMMA)*SD-2*(-XB+XC)*SC*SIN(GAMMA)+(-YB+YC)*(-YB+YC)+2*(-YB+YC)*COS(GAMMA)*SC-2*(-YB+YC)*SD*SIN(GAMMA)+SC*SC+SD*SD)^.5
```

Next, after forming another Z to represent the right hand side of the equation in line (376), we use the RIGHT command in the construction of a unit vector UP5C pointing from P5 to C:

```
UP5C=RIGHT(PP5C)/RIGHT(MAGPP5C)
```

```
-> (385) UP5C=(-XB+XC)*N1/Z141+(-YB+YC)*N2/Z141-SD*K31/Z141+SC*K32/Z141
```

With Z146 defined as

```
-> (395) Z146=L0-Z141
```

we determine the extension of the spring, called STRETCH and given by

```
-> (398) STRETCH=-Z146
```

and then use the RIGHT command to express the force exerted by C on P5, by means of the spring, as (recall that SIGMA is the spring constant)

```
FORCE(C/P5)=SIGMA*RIGHT(STRETCH)*RIGHT(UP5C)
```

```
-> (400) FORCE(C/P5)=-SIGMA*Z142*Z146*N1-SIGMA*Z143*Z146*N2+SIGMA*Z144*Z146*K31-SIGMA*Z145*Z146*K32
```

One more task must be performed before the generalized active force can be computed, namely, the determination of the velocity VP5N of P5 in N, and the velocity VCN of C in N, these being the inertial velocities of the two endpoints of the spring, needed for the determination of partial velocities (see [1], pp. 45-46). The first of these can be found by means of a V2PTS command:

```
V2PTS(N,K3,B,P5)
```

```
-> (402) Z147=SC*Z30
```

```
-> (403) Z148=SD*Z30
```

```
-> (404) VP5N=-U1*Z147*K31-U1*Z148*K32
```

and the second is simply given by

```
VCN=0
```

The FR command then generates the generalized active force:

```
FR
```

-> (407) F1=COS(GAMMA)*SIGMA*Z142*Z146*Z147+COS(GAMMA)*SIGMA*Z14
 3*Z146*Z148-SIGMA*SIN(GAMMA)*Z142*Z146*Z148+SIGMA*SIN(GAMMA)*
 Z143*Z146*Z147-SIGMA*Z144*Z146*Z147+SIGMA*Z145*Z146*Z148+TORQ

and the dynamical equation of motion (see [1], pp. 158-159) of the linkage system is formed via the command:

KANE

which produces

-> (409) R = 1: Z139*U1'-Z140+COS(GAMMA)*SIGMA*Z142*Z146*Z147+
 COS(GAMMA)*SIGMA*Z143*Z146*Z148-SIGMA*SIN(GAMMA)*Z142*Z146*Z1
 48+SIGMA*SIN(GAMMA)*Z143*Z146*Z147-SIGMA*Z144*Z146*Z147+SIGMA
 *Z145*Z146*Z148+TORQ = 0

As an aid in checking the FORTRAN simulation program to be generated by AUTOLEV, we determine the kinetic energy of the linkage by means of the KE command:

KE

-> (411) ZKE1=I1*U1*U1+M1*Z85*Z85

.

-> (417) ZKE7=(Z121*Z121+Z122*Z122)*M7+I7*Z68*Z68

-> (418) KE=.5*(ZKE1+ZKE2+ZKE3+ZKE4+ZKE5+ZKE6+ZKE7)

Note that KE generates ZKE's in lines (411)-(417) for simplification purposes. The potential energy PE is input as

(419) PE=.5*SIGMA*RIGHT(STRETCH)^2

which gives

-> (420) PE=.5*(-Z146)^2*SIGMA

To cause AUTOLEV to write a computer program for the simulation of motions of the linkage, we issue the command

(428) CODE(LINKAGE,ENERGY,CONTROLS)

This yields a FORTRAN program called LINKAGE.FOR, containing a subroutine ENERGY for computing the total energy of the linkage, as well as a subroutine CNTRL in which TORQ is computed. A portion of the FORTRAN program is shown in Figure 2. Here one can see that AUTOLEV has produced a

ready-to-compile-and-run program, complete with DIMENSION statements, COMMON blocks, READ and WRITE statements, and so forth.

When the initial values of β , γ , δ , and ϵ (see Figure 1) are given, those of θ_2 , θ_4 , and θ_6 can be determined easily. In general, however, any one, but no more than one, of these seven initial values can be specified independently, and the remaining six must be found by solving a system of six coupled transcendental equations. Indeed, the determination of initial values of angles is a major obstacle that must be overcome whenever one is dealing with closed loops of bodies, and it is not handled easily by conventional multibody programs. To show how AUTOLEV can be used to come to grips effectively with this dilemma, we determine expressions for the sine and cosine of each of θ_2 , θ_4 , and θ_6 , completely in terms of known quantities. Use of these expressions in conjunction with the FORTRAN function DATAN2 permits one to obtain initial values of the three angles. Moreover, these six relations are precisely the ones needed to determine the values of any six of the linkage angles when the value of the seventh is available. The strategy we employ to find the relationships of interest is to obtain closed loop conditions for each of three such loops in the linkage. By picking these loops in such a way that each involves at most one of K2, K4, and K6, we can obtain a separate expression for each of θ_2 , θ_4 , and θ_6 . We make use of the fact that (see Figure 1) the position vector from P1 to P2 is equal to the sum of the position vectors from P1 to O, from O to B, and from B to P2; the position vector from P3 to P2 is equal to the sum of the position vectors from P3 to A, from A to B, and from B to P2; and the position vector from P4 to P2 is equal to the sum of the position vectors from P4 to A, from A to B, and from B to P2. We first input the position vector POA from O to A, in terms of the given quantities XA and YA, so that we can form the position vector PBA from B to A by means of the ADD command, in conjunction with the position vector POB from O to B, which is already in the workspace:

POA=XA*N1+YA*N2

PBA=ADD(POA,-POB)

-> (435) PBA=(XA-XB)*N1+(YA-YB)*N2

Now all of the position vectors needed for the computations at hand are in the workspace. We construct the position vector from P1 to P2 by means of the ADD command, arbitrarily calling this vector VEC2 so as to distinguish it from PP1P22, the direct vector along K2 from P1 to P2, which is already in the workspace.

```

    VEC2=ADD(-POP1,POB,PBP23)
-> (437) VEC2=XB*N1+YB*N2-R*K11-S*K32

```

In the same manner, we form VEC4 as the position vector from P3 to P2, and VEC6 as the position vector from P4 to P2. Thereafter, dot-multiplication of PP1P22 and VEC2 each with N1 gives, respectively, the left hand side and right hand side of an equation for determining the cosine of θ_2 :

```

    DOT21=DOT(PP1P22,N1)
-> (443) DOT21=-COS(THETA2)*D
    DOT21=DOT(VEC2,N1)
-> (445) DOT21=-COS(BETA)*R+S*SIN(GAMMA)+XB

```

while dot-multiplying the same vectors with N2 yields the two sides of an equation for computing the sine of θ_2 :

```

    DOT22=DOT(PP1P22,N2)
-> (447) DOT22=-D*SIN(THETA2)
    DOT22=DOT(VEC2,N2)
-> (449) DOT22=-COS(GAMMA)*S-R*SIN(BETA)+YB

```

Equations for the sine and cosine of θ_4 and θ_6 are obtained similarly. Substitution for DOT21 from the equation in line (443) into the one in line (445) gives an equation which can be solved for COS(THETA2). Similarly, lines (447) and (449) lead to an expression for SIN(THETA2), and THETA2 is then given by DATAN2(SIN(THETA2),COS(THETA2)). THETA4 and THETA6 can be found in similar fashion. For the given parameter values and initial conditions, these equations reveal that THETA2 = -0.0620105 rad, THETA4 = 0.710540 rad, and THETA6 = 1.00785 rad. The complete AUTOLEV input command file, used to generate LINKAGE.FOR, as well as the formulas for determining initial values of θ_2 , θ_4 , and θ_6 , is shown in Figure 3.

Compiling and executing the FORTRAN program, and confining attention to the first 0.03 seconds of the motion, we obtain the output shown in Table 1, which appears to be in complete agreement with Professor Schiehlen's results. The program also can be used to make runs incorporating more realistic input torques and simulation times, as one can see by setting TORQ equal to $-0.5\sin(2\pi t)$, and running the program for ten seconds. Now one obtains, for example, the time-histories of U1 and β shown in Figure 4.

In practice, one rarely has at one's disposal plots obtained independently which can be used to check the results of one's simulation

program. Accordingly, the issue of checking one's output is a major one. In the present case, we can validate a significant portion of our FORTRAN program by setting TORQK1 equal to zero, and repeating the run leading to Table 1. For all runs made with LINKAGE.FOR, the total mechanical energy of the linkage is automatically recorded on an output data file, and, for the run in which TORQK1 = 0, the energy remained constant.

AUTOLEV's capabilities for dealing with three dimensional problems come to light in connection with the robot arm problem proposed by Professor Schiehlen. The complete AUTOLEV command file for this problem is shown in Figure 5, and numerical output obtained with the aid of the associated AUTOLEV generated FORTRAN program appears in Table 2. Once more, the results appear to be in complete agreement with those of Professor Schiehlen.

Conclusion

As the examples discussed above show, AUTOLEV is intended for use by dynamicists, that is, individuals who possess a thorough knowledge of the discipline of dynamics. In the hands of such individuals, AUTOLEV is a powerful tool that enables them to exercise creativity without being burdened by elementary chores.

The preparation of a new version of AUTOLEV, written in the language C, is in progress. This version will have a number of additional features and will make it possible to execute AUTOLEV not only on personal computers of many types, but also on most mainframe computers.

References

- [1] Kane, Thomas R.; Levinson, David A.: Dynamics: Theory and Applications. New York: McGraw-Hill Book Company, 1985.
- [2] Schaechter, David B.; Levinson, David A.: Interactive Computerized Symbolic Dynamics for the Dynamicist. The Journal of the Astronautical Sciences, Vol. 36, No. 4, October-December 1988, pp. 365-388.

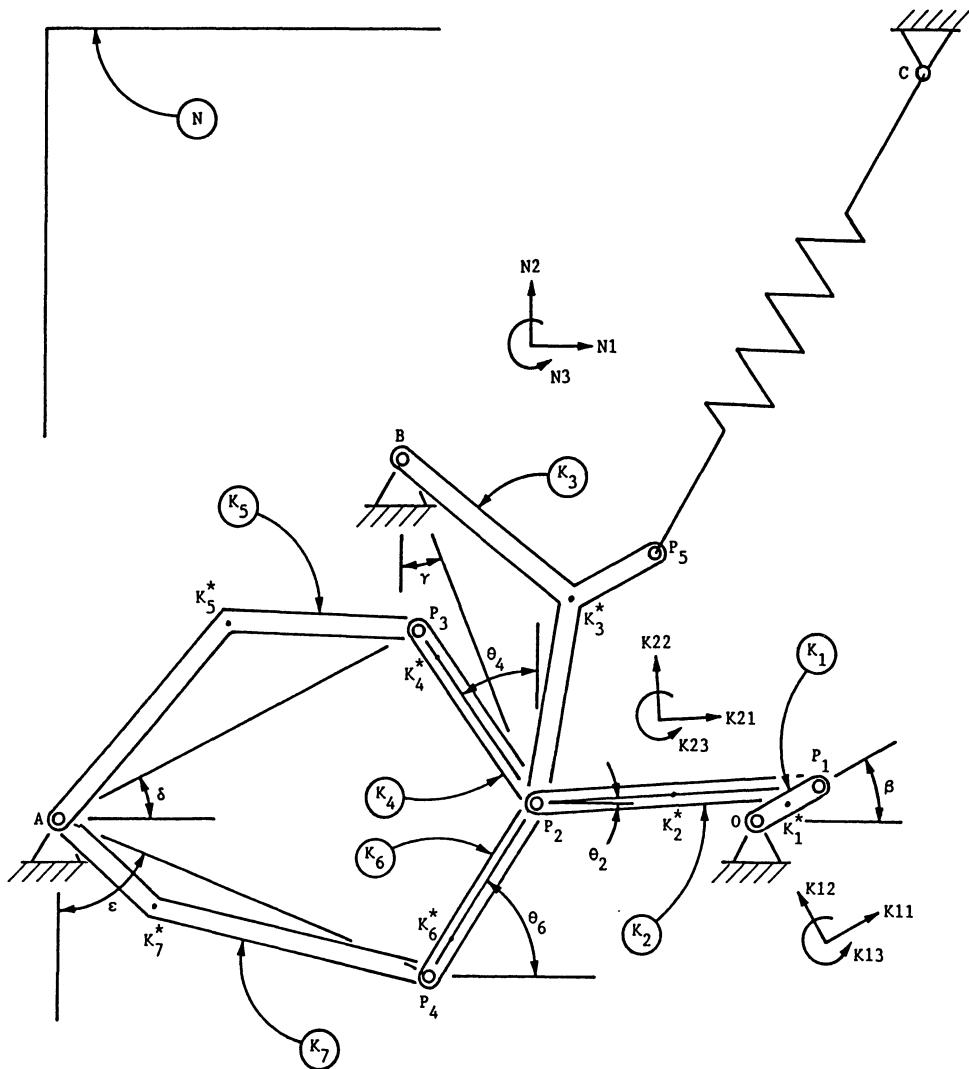


Figure 1. Linkage.

```

C THE NAME OF THIS PROGRAM IS LINKAGE.FOR
C CREATED BY AUTOLEV ON 02-13-1989 AT 12:58:35
C
C IMPLICIT DOUBLE PRECISION (A-Z)
INTEGER JLOOP,NSTEPS,NCUTS,NEQS,ILOOP,COUNTER,NPSTEP
LOGICAL STPSZ
EXTERNAL EQNS
CHARACTER MSG(75)
DIMENSION U(8),UU(14),Z(148)
COMMON/CPAR/I1,I2,I3,I4,I5,I6,I7,M1,M2,M3,M4,M5,M6,M7,PI,D,DA,E,EA
& ,ZF,FA,R,RA,S,SA,SB,SC,SD,ZT,TA,TB,UU,UA,UB,XB,YB,XC,YC,SIGMA,LO
& ,TORQK1
COMMON/CONT/TORQ
COMMON/DFQLST/T,STEP,RELERR,ABSERR,NCUTS,NEQS,STPSZ
C
OPEN(UNIT=11,FILE='LINKAGE.IN ',STATUS='UNKNOWN')
OPEN(UNIT=12,FILE='LINKAGE.OU1',STATUS='UNKNOWN')
OPEN(UNIT=13,FILE='LINKAGE.OU2',STATUS='UNKNOWN')
OPEN(UNIT=14,FILE='LINKAGE.OU3',STATUS='UNKNOWN')
OPEN(UNIT=20,FILE='LINKAGE.NRG',STATUS='UNKNOWN')
OPEN(UNIT=31,FILE='LINKAGE.CO1',STATUS='UNKNOWN')
PI = 4.0*DATAN(1.0D0)
DEGTORAD = PI/180.0D0
RADTODEG = 1.0D0/DEGTORAD
WRITE(*,601)
C
READ(11,*) D,DA,E,EA,ZF,FA,R,RA,S,SA,SB,SC,SD,ZT,TA,TB,UU,UA,UB,XB
& ,YB,XC,YC,SIGMA,LO,TORQK1
READ(11,*) M1,M2,M3,M4,M5,M6,M7
READ(11,*) I1
READ(11,*) I2
READ(11,*) I3
READ(11,*) I4
READ(11,*) I5
READ(11,*) I6
READ(11,*) I7
READ(11,*) (U(ILOOP),ILOOP = 1,1)
READ(11,*) BETA,GAMMA,DELTA,EPSILON,THETA2,THETA4,THETA6
C
WRITE(* ,602)
READ(*,*) TMAX,PSTEP,STEPO
NPSTEP = IDINT((PSTEP-1.D-8)/STEPO + 1)
STEP = PSTEP/NPSTEP
WRITE(* ,603)
READ( * ,604) (MSG(ILOOP),ILOOP = 1,75)

.
.
.

```

Figure 2. Portion of FORTRAN program generated by AUTOLEV.

```

! LINKAGE
!
DOF(1,7)
FRAMES(K1,K2,K3,K4,K5,K6,K7)
POINTS(0,A,B,C,P1,P22,P23,P24,P26,P3,P4,P5)
MASSLESS(0,A,B,C,P1,P22,P23,P24,P26,P3,P4,P5)
VAR(BETA,GAMMA,DELTA,EPSILON,THETA2,THETA4,THETA6)
CONST(D,DA,E,EA,ZF,FA,R,RA,S,SA,SB,SC,SD,ZT,TA,TB,UX,UA,UB)
CONST(XB,YB,XC,YC,SIGMA,LO,TOROK1)
MASS(K1,M1,K2,M2,K3,M3,K4,M4,K5,M5,K6,M7,M7)
INERTIA(K1,0,0,I1,0,0,0)
INERTIA(K2,0,0,I2,0,0,0)
INERTIA(K3,0,0,I3,0,0,0)
INERTIA(K4,0,0,I4,0,0,0)
INERTIA(K5,0,0,I5,0,0,0)
INERTIA(K6,0,0,I6,0,0,0)
INERTIA(K7,0,0,I7,0,0,0)
WK1N=U1*K13
WK2N=U2*K23
WK3N=U3*K33
WK4N=U4*K43
WK5N=U5*K53
WK6N=U6*K63
WK7N=U7*K73
BETA'=U1
GAMMA'=U3
DELTA'=U5
EPSILON'=U7
THETA2'=U2
THETA4'=U4
THETA6'=U6
SIMPROT(N,K1,3,BETA)
SIMPROT(N,K3,3,GAMMA)
SIMPROT(N,K5,3,DELTA)
SIMPROT(N,K7,3,EPSILON)
SIMPROT(N,K2,3,THETA2)
SIMPROT(N,K4,3,THETA4)
SIMPROT(N,K6,3,THETA6)
DIRCOS(K1,K2)
DIRCOS(K4,K5)
DIRCOS(K6,K7)
POK1STAR=RA*K11
POP1=R*K11
VON=0
V2PTS(N,K1,0,K1STAR)
V2PTS(N,K1,0,P1)
PP1K2STAR=-DA*K21
PP1P22=-D*K21
V2PTS(N,K2,P1,K2STAR)
EXPRESS(VK2STARN,K2)
V2PTS(N,K2,P1,P22)
EXPRESS(VP22N,K2)
VBN=0

```

Figure 3. AUTOLEV command file for linkage problem.
(continued on next page)

```

PBK3STAR=SB*K31-SA*K32
V2PTS(N,K3,B,K3STAR)
PBP23=-S*K32
V2PTS(N,K3,B,P23)
PAK5STAR=TA*K51+TB*K52
VAN=0
V2PTS(N,K5,A,K5STAR)
PAP3=ZT*K51
V2PTS(N,K5,A,P3)
PP3K4STAR=(EA-E)*K42
V2PTS(N,K4,P3,K4STAR)
EXPRESS(VK4STARN,K4)
PP3P24=-E*K42
V2PTS(N,K4,P3,P24)
EXPRESS(VP24N,K4)
PAK7STAR=-UB*K71-UA*K72
V2PTS(N,K7,A,K7STAR)
PAP4=-UX*K72
V2PTS(N,K7,A,P4)
PP4K6STAR=(ZF-FA)*K61
V2PTS(N,K6,P4,K6STAR)
EXPRESS(VK6STARN,K6)
PP4P26=ZF*K61
V2PTS(N,K6,P4,P26)
EXPRESS(VP26N,K6)
MATCH23=ADD(VP22N,-VP23N)

MATCH24=ADD(VP22N,-VP24N)
MATCH26=ADD(VP22N,-VP26N)
DIRCOS(K2,K3)
DIRCOS(K2,K4)
DIRCOS(K2,K6)
CONSTRAINT1=DOT(MATCH23,K21)
CONSTRAINT2=DOT(MATCH23,K22)
CONSTRAINT3=DOT(MATCH24,K21)
CONSTRAINT4=DOT(MATCH24,K22)
CONSTRAINT5=DOT(MATCH26,K21)
CONSTRAINT6=DOT(MATCH26,K22)
CONSTRAIN
ALFK1N=DERIV(WK1N,T,N)
ALFK2N=DERIV(WK2N,T,N)
ALFK3N=DERIV(WK3N,T,N)
ALFK4N=DERIV(WK4N,T,N)
ALFK5N=DERIV(WK5N,T,N)
ALFK6N=DERIV(WK6N,T,N)
ALFK7N=DERIV(WK7N,T,N)
AK1STARN=DERIV(VK1STARN,T,N)
AK2STARN=DERIV(VK2STARN,T,N)
AK3STARN=DERIV(VK3STARN,T,N)
AK4STARN=DERIV(VK4STARN,T,N)
AK5STARN=DERIV(VK5STARN,T,N)
AK6STARN=DERIV(VK6STARN,T,N)
AK7STARN=DERIV(VK7STARN,T,N)

```

Figure 3. AUTOLEV command file for linkage problem.
(continued on next page)

```

FRSTAR
TORQUE(K1)=TORQ*K13
CONTROLS(TORQ)
TORQ=TORQK1
POB=XB*N1+YB*N2
POC=XC*N1+YC*N2
PBC=ADD(POC,-POB)
PBP5=SD*K31-SC*K32
PP5C=ADD(PBC,-PBP5)
MAGPP5C2=DOT(PP5C,PP5C)
MAGPP5C=RIGHT(MAGPP5C2)^.5
MAGVEC=RIGHT(MAGPP5C)*N3
ZEE(MAGVEC)
MAGPP5C=DOT(MAGVEC,N3)
UP5C=RIGHT(PP5C)/RIGHT(MAGPP5C)
ZEE(UP5C)
STRETCHVEC=(RIGHT(MAGPP5C)-LO)*N3
ZEE(STRETCHVEC)
STRETCH=DOT(STRETCHVEC,N3)
FORCE(C/P5)=SIGMA*RIGHT(STRETCH)*RIGHT(UP5C)
V2PTS(N,K3,B,P5)
VCN=0
FR
KANE
KE
PE=.5*SIGMA*RIGHT(STRETCH)^2
UNITS(D,M,DA,M,E,M,EA,M,ZF,M,FA,M,R,M,RA,M,S,M,SA,M,SB,M,SC,M)
UNITS(SD,M,ZT,M,TA,M,TB,M,UU,M,UA,M,UB,M,LO,M,XB,M,YB,M,XC,M,YC,M)
UNITS(SIGMA,N/M,MASS,KG,INERTIA,KG*M^2,TORQ,N*M,TORQK1,N*M,ENERGY,N*M)
UNITS(BETA,RAD,GAMMA,RAD,DELTA,RAD,EPSILON,RAD)
UNITS(THETA2,RAD,THETA4,RAD,THETA6,RAD)
UNITS(U1,RAD/S,U2,RAD/S,U3,RAD/S,U3,RAD/S,U4,RAD/S,U5,RAD/S)
UNITS(U6,RAD/S,U7,RAD/S)
CODE(LINKAGE,ENERGY,CONTROLS)
!
! ADDITIONAL COMPUTATIONS FOR INITIAL VALUES OF THETA2, THETA4, THETA6
!
AUTOZ(OFF)
POA=XA*N1+YA*N2
PBA=ADD(POA,-POB)
VEC2=ADD(PBP23,-POP1,POB)
VEC4=ADD(PBP23,-PAP3,-PBA)
VEC6=ADD(PBP23,-PAP4,-PBA)
DOT21=DOT(PP1P22,N1)
DOT21=DOT(VEC2,N1)
DOT22=DOT(PP1P22,N2)
DOT22=DOT(VEC2,N2)
DOT41=DOT(PP3P24,N1)
DOT41=DOT(VEC4,N1)
DOT42=DOT(PP3P24,N2)
DOT42=DOT(VEC4,N2)
DOT61=DOT(PP4P26,N1)
DOT61=DOT(VEC6,N1)
DOT62=DOT(PP4P26,N2)
DOT62=DOT(VEC6,N2)

```

Figure 3, continued. AUTOLEV command file for linkage problem.

Table 1. Output from AUTOLEV-generated linkage program.

T (S)	U1 (RAD/S)	U2 (RAD/S)	U3 (RAD/S)	U4 (RAD/S)	U5 (RAD/S)
0.000D+00	0.00000D+00	0.00000D+00	0.00000D+00	0.00000D+00	0.00000D+00
5.000D-03	1.51063D+02	4.05619D+01	-6.68871D+00	-1.19603D+01	1.48195D+00
1.000D-02	7.12113D+02	-7.49052D+01	-1.36449D+02	-2.35497D+02	4.82635D+00
1.500D-02	4.41630D+02	6.63431D+01	4.66062D+01	8.24863D+01	-9.51138D+00
2.000D-02	7.26688D+02	-2.21120D+01	-1.45760D+02	-2.49726D+02	9.82700D+00
2.500D-02	5.87417D+02	1.09914D+02	4.30339D+01	7.66602D+01	-9.26604D+00
3.000D-02	1.13997D+03	-2.84467D+02	1.11423D+01	2.00597D+01	5.74991D-01

T (S)	U6 (RAD/S)	U7 (RAD/S)	BETA (RAD)	GAMMA (RAD)	DELTA (RAD)
0.000D+00	0.00000D+00	0.00000D+00	-6.19941D-02	4.55440D-01	4.87519D-01
5.000D-03	8.39177D+00	-5.05046D+00	2.10881D-01	4.49270D-01	4.88901D-01
1.000D-02	1.99198D+02	-4.11251D+01	2.16040D+00	1.58559D-01	5.25402D-01
1.500D-02	-5.83187D+01	3.36789D+01	5.65556D+00	4.26089D-01	4.93832D-01
2.000D-02	2.01803D+02	-5.77498D+01	8.18514D+00	2.09585D-01	5.22772D-01
2.500D-02	-5.39224D+01	3.20038D+01	1.21073D+01	4.41160D-01	4.90672D-01
3.000D-02	-1.91577D+01	3.27049D-01	1.58110D+01	4.09515D-02	5.24695D-01

T (S)	EPSILON (RAD)	THETA2 (RAD)	THETA4 (RAD)	THETA6 (RAD)	
0.000D+00	1.23087D+00	-6.20105D-02	7.10540D-01	1.00785D+00	
5.000D-03	1.22619D+00	9.21964D-03	6.99490D-01	1.01560D+00	
1.000D-02	1.06869D+00	2.76555D-01	1.96830D-01	1.39726D+00	
1.500D-02	1.20906D+00	-1.78583D-01	6.58257D-01	1.04464D+00	
2.000D-02	1.08653D+00	2.94223D-01	2.84529D-01	1.32477D+00	
2.500D-02	1.22011D+00	-1.50095D-01	6.85017D-01	1.02577D+00	
3.000D-02	1.04836D+00	5.41888D-02	-9.82046D-03	1.58288D+00	

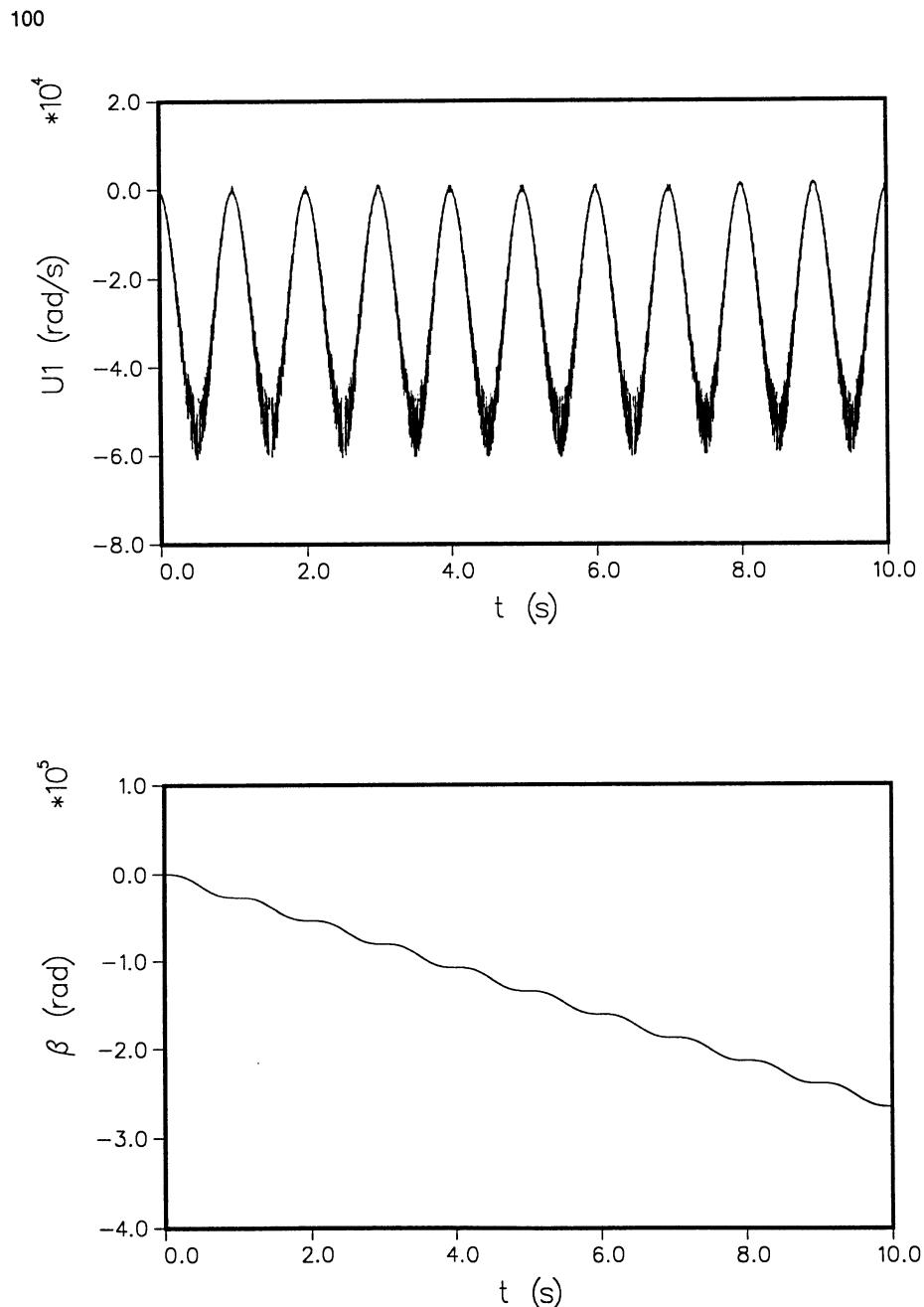


Figure 4. Time histories for 10 second run.

```

! ARM
!
DOF(5)
FRAMES(B1,B2,B3)
PRINCIPAL(B1,B2,B3)
CONST(L,C,G)
WB1N=U1*B13
VB1STAR=N=U2*B13
WB2B1=U3*B22
VB2STARB1=U4*B22
VB3B2=U5*B31
GA1'=U1
ZZ1'=U2
BE2'=U3
YY2'=U4
AL3'=U5
VB2N=ADD(WB1N,WB2B1)
WB3N=ADD(WB2N,WB3B2)
SIMPROT(N,B1,3,GA1)
SIMPROT(B1,B2,2,BE2)
SIMPROT(B2,B3,1,AL3)
DIRCOS(B2,N)
DIRCOS(B3,N)
EXPRESS(WB2N,B2)
EXPRESS(WB3N,B3)
POINTS(B1BARB2STAR,P)
MASSLESS(B1BARB2STAR,P)
PB1STAR1BARB2STAR=YY2*B12
V2PTS(N,B1,B1STAR,B1BARB2STAR)
VB2STAR=N=ADD(VB1BARB2STAR,N,VB2STARB1)
EXPRESS(VB2STAR,N,B2)
PB2STARP=L*B22
V2PTS(N,B2,B2STAR,P)
PPB3STAR=C*B32
V2PTS(N,B3,P,B3STAR)
EXPRESS(VB3STAR,N,B3)
ALFB1N=DERIV(WB1N,T,N)
ALFB2N=DERIV(WB2N,T,N)
ALFB3N=DERIV(WB3N,T,N)
AB1STAR=DERIV(VB1STAR,N,T)
AB2STAR=DERIV(VB2STAR,N,T)
AB3STAR=DERIV(VB3STAR,N,T)
FRSTAR
TORQUE(B1)=L1Z*B13
FORCE(B1STAR)=-MASSB1*G*N3
POINTS(NSTAR)
MASSLESS(NSTAR)
VNSTAR=0
FORCE(NSTAR/B1STAR)=F1Z*B13
TORQUE(B1/B2)=L2Y*B22
FORCE(B1STAR/B2STAR)=F2Y*B22
TORQUE(B2/B3)=L3X*B31
FORCE(B2STAR)=-MASSB2*G*N3
FORCE(B3STAR)=-MASSB3*G*N3
FR
KANE
CONTROLS(F1Z,F2Y,L1Z,L3X,L2Y)
PNSTARB1STAR=ZZ1*B13
PB1STARB2STAR=RIGHT(PB1STARB1BARB2STAR)
PNSTARB2STAR=ADD(PNSTARB1STAR,PB1STARB2STAR)
PB2STARB3STAR=ADD(PB2STARP,PPB3STAR)
PNSTARB3STAR=ADD(PNSTARB2STAR,PB2STARB3STAR)
ANGMOM(NSTAR,B1)
UNITS(U1,RAD/S,U2,M/S,U3,RAD/S,U4,M/S,U5,RAD/S)
UNITS(ZZ1,M,GA1,RAD,YY2,M,BE2,RAD,AL3,RAD,C,M,L,M)
UNITS(ANGMOM,N*M*S,MASS,KG,INERTIA,KG*M^2)
UNITS(F1Z,N,F2Y,N,L1Z,N*M,L3X,N*M,L2Y,N*M,G,M/S^2)
CODE(ARM,ANGMOM,CONTROLS)

```

Figure 5. AUTOLEV command file for robot arm problem.

Table 2. Output from AUTOLEV-generated robot arm program.

T (S)	U1 (RAD/S)	U2 (M/S)	U3 (RAD/S)	U4 (M/S)	U5 (RAD/S)
0.000D+00	0.00000D+00	0.00000D+00	0.00000D+00	0.00000D+00	0.00000D+00
2.500D-01	-2.49508D-01	7.21488D-01	-5.49465D-05	9.97339D-01	1.24991D-03
5.000D-01	-2.54942D-01	1.44297D+00	-1.84886D-04	2.01965D+00	2.87874D-03
7.500D-01	-1.33939D-01	1.44297D+00	2.37645D-04	2.03230D+00	2.86076D-03
1.000D+00	-8.41940D-02	1.44297D+00	5.12784D-04	2.03658D+00	2.85453D-03
1.250D+00	-5.79981D-02	1.44297D+00	7.13914D-04	2.03792D+00	2.85708D-03
1.500D+00	-4.22493D-02	1.44297D+00	8.78649D-04	2.03792D+00	2.86726D-03
1.750D+00	-3.40515D-02	7.21527D-01	9.85913D-04	1.02022D+00	-2.72497D-03
2.000D+00	-3.14650D-02	7.51099D-05	1.01902D-03	2.26674D-03	-7.51688D-03

T (S)	GA1 (RAD)	Z1 (M)	BE2 (RAD)	Y2 (M)	AL3 (RAD)
0.000D+00	-5.23600D-01	2.25000D+00	0.00000D+00	7.50000D-01	0.00000D+00
2.500D-01	-5.59781D-01	2.34019D+00	-3.83511D-06	8.74096D-01	1.52503D-04
5.000D-01	-6.26999D-01	2.61074D+00	-3.54518D-05	1.25089D+00	6.56220D-04
7.500D-01	-6.73020D-01	2.97149D+00	-2.49571D-05	1.75768D+00	1.37341D-03
1.000D+00	-6.99538D-01	3.33223D+00	7.10158D-05	2.26638D+00	2.08761D-03
1.250D+00	-7.17009D-01	3.69297D+00	2.25389D-04	2.77573D+00	2.80139D-03
1.500D+00	-7.29389D-01	4.05371D+00	4.24989D-04	3.28523D+00	3.51678D-03
1.750D+00	-7.38771D-01	4.32428D+00	6.59798D-04	3.66751D+00	3.53505D-03
2.000D+00	-7.46871D-01	4.41448D+00	9.11753D-04	3.79532D+00	2.22165D-03

T (S)	F1Z (N)	F2Y (N)	L1Z (N*M)	L3X (N*M)	L2Y (N*M)
0.000D+00	6.34800D+03	9.86000D+02	-5.08000D+02	6.35000D+01	0.00000D+00
2.500D-01	6.34800D+03	9.95000D+02	-3.39750D+02	6.35000D+01	0.00000D+00
5.000D-01	4.90500D+03	-2.00000D+00	1.56000D+02	4.90500D+01	0.00000D+00
7.500D-01	4.90500D+03	-2.00000D+00	4.54203D+01	4.90500D+01	0.00000D+00
1.000D+00	4.90500D+03	-2.00000D+00	1.74613D+01	4.90500D+01	0.00000D+00
1.250D+00	4.90500D+03	-2.00000D+00	1.03922D+01	4.90500D+01	0.00000D+00
1.500D+00	3.46200D+03	-1.01900D+03	8.60484D+00	3.46000D+01	0.00000D+00
1.750D+00	3.46200D+03	-1.01900D+03	8.60484D+00	3.46000D+01	0.00000D+00
2.000D+00	3.46200D+03	-1.01900D+03	8.60484D+00	3.46000D+01	0.00000D+00

UCIN-DYNOCOMBS - Software for the Dynamic Analysis of Constrained Multibody Systems

R. L. Huston

T. P. King

University of Cincinnati

Cincinnati, OH 45221-0072, USA

J. W. Kamman

Naval Coastal Systems Center

Panama City, FL 32407, USA

Summary

DYNOCOMBS simulates the dynamics of constrained multibody systems [1]. The systems simulated may have general (three-dimensional) motion including translation between adjoining bodies and closed loops. Arbitrary force systems may be applied to the bodies and between the bodies.

DYNOCOMBS uses Kane's equations to simulate the dynamics [2]. Partial velocity and partial angular velocity vectors form fundamental arrays in the development of the governing equations. DYNOCOMBS also uses Euler parameters to avoid geometrical singularities.

Constraint equations arising from the geometric and kinematic constraints (closed loops or specified motion) are appended to the dynamics equation. The resulting system of equations is reduced to a consistent set by using orthogonal complement arrays.

The governing differential equations are solved using a fourth-order Runge-Kutta procedure - although other integration techniques may be employed.

As input, the user supplies a body connection array, geometrical and physical properties of the bodies, constraint conditions, applied forces, and initial conditions. As output, DYNOCOMBS provides a description of the system configuration and movement at equal time intervals. Forces exerted on the bodies of the system are also provided.

The language of DYNOCOMBS is FORTRAN.

The following paragraphs provide an outline of the theoretical basis of DYNOCOMBS. Example simulations are also discussed.

Preliminary Considerations

1. Multibody Systems

Figure 1 depicts a typical multibody system as might be studied

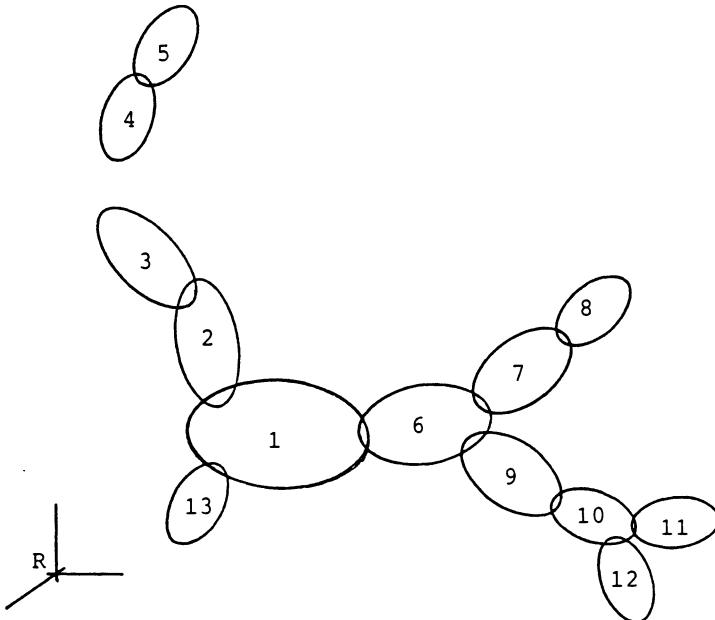


Figure 1. A Typical Multibody System

using DYNOCOMBS. The system consists of a series of connected (or nearly connected) bodies forming an "open-tree" or "open-chain". The bodies may be connected to each other by free, spherical, revolute (pin), prismatic (sliding), or fixed joints. The bodies of the system are labelled and identified by numbers. An arbitrary body of the system is selected as the "reference body" and labelled as "body 1". The remaining bodies are then numbered or labelled in ascending progression through the branches of the tree system.

In the particular system depicted in Figure 1 if bodies 8 and

11 are connected, the system will contain a closed loop. The system is then no longer an open-chain system. However, it may be studied as though it were an open-chain, if there is a geometric constraint equation included in the analysis to account for the loop.

Finally, in Figure 1 let R be an inertia reference frame in which the system moves.

Such multibody systems may be used to model a wide range of physical systems including chains, cables, biosystems, robots, mechanisms, vehicles and structures.

2. Body Connection Arrays

When the system is labelled and numbered as described above and as illustrated in Figure 1 each body has a unique adjoining lower numbered body. (The lower numbered body of body 1, is the inertia frame R and is assigned the number 0.) The connection configuration of the system can then be described by an integer array of labels of the adjoining lower numbered bodies, called "the lower body array". For example, for the system shown in Figure 1 the body array $L^0(K)$ and the lower body array $L(K)$ are

$$L^0(K) = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13)$$

and

$$L(K) = (0, 1, 2, 3, 4, 1, 6, 7, 6, 9, 10, 10, 1)$$

Observe that once the system is labelled and numbered, the lower body array $L(K)$ is uniquely determined. Alternatively, $L(K)$ may be viewed as defining the connection configuration. That is, by knowing $L(K)$ an algorithm can be written to numerically describe the system and its configuration.

$L(K)$ may be considered to be an operator on the body number K. In this context, we can form a lower array to $L(K)$ -- that is, $L(L(K))$ [or $L^2(K)$], called the "second lower-body array". For the system shown in Figure 1, $L^2(K)$ is

$$L^2(K) = (0, 0, 1, 2, 3, 0, 1, 6, 1, 6, 9, 9, 0) \quad (2)$$

where $L(0)$ is 0.

By continuing this process we can form lower arrays until all entries are zero. For the system of Figure 1 the complete set of lower arrays are listed in Table 1.

	Body Number												
Array	1	2	3	4	5	6	7	8	9	10	11	12	13
$L(K)$	0	1	2	3	4	1	6	7	6	9	10	10	1
$L^2(K)$	0	0	1	2	3	0	1	6	1	6	9	9	0
$L^3(K)$	0	0	0	1	2	0	0	1	0	1	6	6	0
$L^4(K)$	0	0	0	0	1	0	0	0	0	0	1	1	0
	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1. Set of Lower Body Arrays for the System of Figure 1

Observe that once $L(K)$ is known the entire set of lower body arrays, as in Table 1, can be automatically generated.

Observe further that in addition to defining the configuration of the system, the $L(K)$ array may be used to automatically develop the kinematics of the system. For example, for the system of Figure 1 the angular velocity of body 11, in R, may be expressed as (angular velocity addition theorem):

$$\dot{\omega}_{11} = \hat{\omega}_{10} + \hat{\omega}_9 + \hat{\omega}_6 + \hat{\omega}_1 \quad (3)$$

where the "hat" designates the angular velocity relative to the adjacent lower-numbered body. Observe that the indices of Equation (3) correspond to the entries in the column of body 11 in Table 1.

3. Transformation Matrices

Consider a typical pair of adjoining bodies B_j and B_k as in Figure 2. Let n_{ji} and n_{ki} ($i=1,2,3$) be orthogonal, dextral unit vector sets fixed in B_j and B_k . The relative orientation of the bodies can then be defined in terms of the relative inclination of these unit vector sets. Let SJK be a transformation matrix whose elements are the projections of the unit vectors from one set to the other (direction cosines). That is,

$$SJK_{mn} = n_{jm} \cdot n_{kn} \quad (4)$$

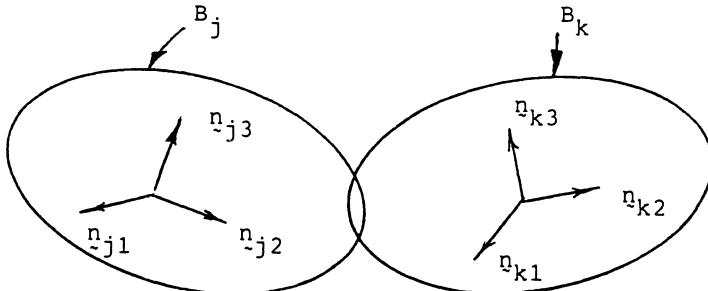


Figure 2. Two Typical Adjoining Bodies

The unit vectors are then related to each other by the expression

$$\underline{n}_{jm} = SJK_{mn} \underline{n}_k \quad (5)$$

[Regarding notation, the J and K in SJK and the first subscripts on the unit vectors refer to the bodies B_j and B_k , and repeated indices, such as the n in Equation (5), designate a sum over the range of the index.]

The transformation matrices are seen to be orthogonal: that is, the inverse is equal to the transpose. Also, they obey the transitive relation

$$SJL = SJK SKL \quad (6)$$

By repeated use of Equation (6) we can obtain a transformation matrix between the unit vectors of any given body and the inertia frame R. For example, for body 8 of the system of Figure 1 we have

$$S08 = S01 S16 S67 S78 \quad (7)$$

Observe that the indices of Equation (7) correspond to the column entries of B_8 in Table 1.

The transformation matrices are used to obtain inertia frame components of vectors originally referred to body-fixed unit vectors. That is, if \underline{v} is a vector expressed in the forms:

$$\underline{v} = V_{km} \underline{n}_{km} = V_{on} \underline{n}_{on} \quad (\text{no sum on k or o}) \quad (8)$$

where \underline{n}_{oj} are unit vectors fixed in R, then the components V_{ki} and V_{oj} are related by the expression:

$$V_{on} = SOK_{nm} V_{km} \quad (9)$$

4. Differentiation Algorithm

Since DYNOCOMBS uses Kane's equations, a major aspect of the analysis is the differentiation of vector quantities (as opposed to Lagrangian methods which use scalar energy functions). It happens that many vector quantities of interest are fixed in bodies of the system. For these vectors, their derivatives may be calculated by vector multiplication. For example, if \underline{r} is a vector fixed in a body B_k , the derivative of \underline{r} in inertia frame R is:

$$\frac{d\underline{r}}{dt} = \omega_k \times \underline{r} \quad (10)$$

where ω_k is the angular velocity of B_k in R.

Equation (10) forms the basis of a series of useful and efficient algorithms used in DYNOCOMBS. The equation shows that derivatives may be calculated by multiplication. For example, for the transformation matrices SOK, Equation (10) leads to the differentiation algorithm

$$\frac{d(SOK)}{dt} = WOK \cdot SOK \quad (11)$$

where WOK is the 3×3 array with elements:

$$WOK_{im} = -e_{imn}\omega_{kn} \quad (12)$$

where e_{imn} is the standard permutation symbol and the ω_{kn} ($n=1, 2, 3$) are the n -on components of ω_k .

5. Euler Parameters, Generalized Speeds

DYNOCOMBS uses Euler parameters as its orientation parameters. The use of Euler parameters to describe orientation of rigid bodies has recently become popular among many analysts. Euler parameters are a set of four parameters ϵ_{ki} ($i=1, \dots, 4$) defined as follows: Let B_k be brought into a general orientation relative to the adjacent lower-numbered body B_j by a rotation through an angle θ_k about a line fixed in both B_j and B_k . If λ_k is a unit vector parallel to the line, then the Euler parameters are [3]:

$$\epsilon_{ki} = \lambda_{ki} \sin(\theta_k/2) \quad (i=1, 2, 3) \quad (\text{no sum on } k)$$

$$\epsilon_{k4} = \cos(\theta_k/2) \quad (13)$$

where the λ_{ki} are the n_{ki} components of λ_k .

Euler parameters present an alternative to orientation angles. However, only three orientation angles are required to define the orientation of a body. Hence, Euler parameters are redundant. Indeed, from Equation (13) we see that the Euler parameters are dependent. That is,

$$\epsilon_{k1}^2 + \epsilon_{k2}^2 + \epsilon_{k3}^2 + \epsilon_{k4}^2 = 1 \quad (14)$$

With Euler parameters the kinematic equations take a linear form. With orientation angles they are nonlinear. Hence, the use of Euler parameters avoids singularities which can cause numerical difficulties. Specifically, the Euler parameters are related to the relative angular velocity components by the equations:

$$\begin{aligned}\dot{\epsilon}_{k1} &= (1/2)(\epsilon_{k4}\hat{\omega}_{k1} + \epsilon_{k3}\hat{\omega}_{k2} - \epsilon_{k2}\hat{\omega}_{k3}) \\ \dot{\epsilon}_{k2} &= (1/2)(-\epsilon_{k3}\hat{\omega}_{k1} + \epsilon_{k4}\hat{\omega}_{k2} + \epsilon_{k1}\hat{\omega}_{k3}) \\ \dot{\epsilon}_{k3} &= (1/2)(\epsilon_{k2}\hat{\omega}_{k1} - \epsilon_{k1}\hat{\omega}_{k2} + \epsilon_{k4}\hat{\omega}_{k3}) \\ \dot{\epsilon}_{k4} &= (1/2)(-\epsilon_{k1}\hat{\omega}_{k1} - \epsilon_{k2}\hat{\omega}_{k2} - \epsilon_{k3}\hat{\omega}_{k3})\end{aligned} \quad (15)$$

Also, the transformation matrix SJK may be expressed in terms of the Euler parameters as:

$$SJK = \begin{bmatrix} (\epsilon_{k1}^2 - \epsilon_{k2}^2 - \epsilon_{k3}^2 + \epsilon_{k4}^2) & 2(\epsilon_{k1}\epsilon_{k2} - \epsilon_{k3}\epsilon_{k4}) & 2(\epsilon_{k1}\epsilon_{k3} + \epsilon_{k2}\epsilon_{k4}) \\ 2(\epsilon_{k1}\epsilon_{k2} + \epsilon_{k3}\epsilon_{k4}) & (-\epsilon_{k1}^2 + \epsilon_{k2}^2 - \epsilon_{k3}^2 + \epsilon_{k4}^2) & 2(\epsilon_{k2}\epsilon_{k3} - \epsilon_{k1}\epsilon_{k4}) \\ 2(\epsilon_{k1}\epsilon_{k3} - \epsilon_{k2}\epsilon_{k4}) & 2(\epsilon_{k2}\epsilon_{k3} + \epsilon_{k1}\epsilon_{k4}) & (-\epsilon_{k1}^2 - \epsilon_{k2}^2 + \epsilon_{k3}^2 + \epsilon_{k4}^2) \end{bmatrix} \quad (16)$$

Finally, in formulating the governing equations it is convenient to use linear combinations of generalized coordinate derivatives as the fundamental dependent variables of the system. These variables, called "generalized speeds", cannot generally be expressed as derivatives of individual coordinate variables. Hence, with generalized speeds the analysis is said to be

be formulated in terms of "quasi-coordinates". In DYNOCOMBS, relative angular velocity components are used as generalized speeds with Euler parameters serving as the orientation parameters. Equations (15) together with the dynamics equations (see the following section) thus form the system of governing first-order ordinary differential equations.

Dynamics

1. Kinematics

If a multibody system has N bodies, the system can have as many as $6N$ degrees of freedom. DYNOCOMBS initially treats each system as being general and thus having the full $6N$ degrees of freedom. DYNOCOMBS then reduces the number of degrees of freedom by using specialized joints and by accomodating the external constraints on the system.

Let x_ℓ ($\ell=1,\dots,6N$) be parameters describing the degrees of freedom and let y_ℓ ($\ell=1,\dots,6N$) (generalized speeds) be linear combinations of the derivatives of the x_ℓ . Then DYNOCOMBS formulates the system kinematics in terms of the y_ℓ and Euler parameters as described above. Specifically, for the translation degrees of freedom, DYNOCOMBS uses the relative translation between adjoining bodies for the y_ℓ and for the rotational degrees of freedom, DYNOCOMBS uses the components of the relative angular velocity between adjoining bodies for the y_ℓ . Thus DYNOCOMBS uses relative coordinates.

Using these parameters it is readily seen that the angular velocity of a typical body B_k of the system relative to the inertia frame R may be expressed in the form [3]

$$\omega_k = \omega_{klm} y_l n_{om} \quad (17)$$

where the ω_{klm} are n_{om} components of the "partial angular velocity" of B_k in R, as used by Kane, et. al. [2,3]. The ω_{klm} may be expressed in terms of the transformation matrices. Hence, by using the body connection array, algorithms are readily developed for automatic computation of the ω_{klm} array.

The angular acceleration of B_k in R may be obtained by dif-

ferentiation in Equation (17):

$$\ddot{a}_k = (\omega_{klm} \dot{y}_l + \dot{\omega}_{klm} y_l) n_{om} \quad (18)$$

where the $\dot{\omega}_{klm}$ are obtained from algorithms based upon Equation (11).

Similarly, if G_k is the mass center of B_k , the velocity and acceleration of G_k in R may be expressed in the forms

$$\dot{v}_k = v_{klm} y_l n_{om} \quad \text{and} \quad \ddot{v}_k = (v_{klm} \dot{y}_l + \dot{v}_{klm} y_l) n_{om} \quad (19)$$

where the v_{klm} are the n_{om} components of the "partial velocity" of G_k in R [2,3]. As with the ω_{klm} and $\dot{\omega}_{klm}$, simple algorithms have been developed for the computation of the v_{klm} and \dot{v}_{klm} .

By knowing the connection configuration of the system and the geometric parameters of the bodies, DYNOCOMBS automatically develops the ω_{klm} , $\dot{\omega}_{klm}$, v_{klm} , and \dot{v}_{klm} arrays. These are the principal arrays needed for the development of the governing dynamical equations.

2. Kinetics

Let the system be subjected to an externally applied force field. On a typical body B_k let this force field be represented by a single force F_k passing through G_k together with a couple with torque M_k . Then the generalized applied ("active") force on B_k , corresponding to the generalized speed y_l , may be expressed as:

$$F_l = F_{km} v_{klm} + M_{km} \omega_{klm} \quad (20)$$

where F_{km} and M_{km} are the n_{om} components of F_k and M_k .

Similarly let the inertia forces on typical body B_k be replaced by a single force F_k^* passing through G_k together with a couple with torque M_k^* . Then F_k^* and M_k^* may be expressed as [2]

$$F_k^* = -m_k a_k \quad \text{and} \quad M_k^* = -I_k \cdot \ddot{a}_k - \omega_k \times (I_k \cdot \omega_k) \quad (\text{no sum}) \quad (21)$$

where m_k is the mass of B_k and I_k is the central inertia dyadic of B_k .

The generalized inertia ("passive") force on B_k , corresponding to the generalized speed y may then be expressed as:

$$F_{\ell}^* = F_{km}^* V_{k\ell m} + M_{km}^* \omega_{k\ell m} \quad (22)$$

where F_{km}^* and M_{km}^* are the n_{om} components of F_k^* and M_k^* .

Finally, if the system is subjected to constraints there will be constraining forces exerted on the bodies. Let these constraining forces be represented on B_k by a force F'_k passing through G_k together with a couple with moment M'_k . Then the generalized constraint force, corresponding to y_{ℓ} , may be expressed as

$$F'_{\ell} = F'_{km} V_{k\ell m} + M'_{km} \omega_{k\ell m} \quad (23)$$

where F'_{km} and M'_{km} are the n_{om} components of F'_k and M'_k .

3. Dynamical Equations

The dynamical equations for the system are obtained by using Kane's equation which may be expressed as [2,3,4]:

$$F_{\ell} + F_{\ell}^* + F'_{\ell} = 0 \quad \ell=1, \dots, 6N \quad (24)$$

By substituting from Equations (17) to (23) into Equation (24), the dynamical equations may be written as

$$a_{\ell p} \dot{y}_p = f_{\ell} + F'_{\ell} \quad (25)$$

where the coefficients $a_{\ell p}$ and f_{ℓ} are:

$$a_{\ell p} = m_k V_{k\ell m} V_{kpm} + I_{kmn} \omega_{k\ell m} \omega_{kpn} \quad (26)$$

and

$$\begin{aligned} f_{\ell} = & F_{\ell} - (m_k V_{k\ell m} \dot{V}_{kpm} y_p + I_{kmn} \omega_{k\ell m} \dot{\omega}_{kpn} y_p \\ & + e_{rsm} I_{ksn} \omega_{k\ell m} \omega_{kqr} \omega_{kpn} y_p y_q) \end{aligned} \quad (27)$$

where I_{kmn} are the n_{om} , n_{on} components of I_k .

Observe the central role of the $V_{k\ell m}$ and $\omega_{k\ell m}$ arrays in Equations (26) and (27).

4. Constraint Equations

If the system is subject to constraints either by the presence of closed loops or by specified motion of points or bodies of the system, there occur constraint equations which may be expressed in the form

$$b_{q\ell} y_\ell = g_q \quad (q=1, \dots, m) \quad (m < 6N) \quad (28)$$

It is shown in [4] that the $b_{q\ell}$ are related to the generalized constraint force F'_ℓ by the expression

$$F'_\ell = b_{q\ell} \lambda_q \quad (29)$$

where λ_q ($q=1, \dots, m$) are components of the constraint forces and moments.

5. Governing Equations

Equations (25) and (28) form a set of $n+m$ equations for the n y_ℓ and the m λ_q . Using Equations (29), Equations (25) and (28) may be written in matrix form as:

$$\dot{Ay} = f + B^T \lambda \quad \text{and} \quad By = g \quad (30)$$

where the elements of the arrays are identified by comparison with Equations (25) and (28).

Equations (30) are coupled differential/algebraic equations. As such they are not readily adaptable to differential equation solvers. However, they may be converted into a "solvable" system by eliminating the λ array. To this end, let C be an orthogonal complement of B -- that is, an $n \times (n-m)$ array C such that

$$BC = 0 \quad \text{or} \quad C^T B^T = 0 \quad (31)$$

Then by premultiplying the first of Equations (30) by C^T , the governing equations become:

$$C^T A \dot{y} = C^T f \quad \text{and} \quad By = g \quad (32)$$

DYNOCOMBS uses a zero-eigenvalue procedure developed by Walton

and Steeves [5] to obtain an orthogonal complement array. Singular value decomposition may also be used [6].

Numerical Solution Methods

Equations (26) and (27) show that the coefficients of the governing equations are explicit functions of the v_{klm} , \dot{v}_{klm} , w_{klm} , and \dot{w}_{klm} arrays and the physical and geometrical parameters of the system. Hence, algorithms for the development and solution of the governing equations might proceed as follows: After the connection array L(K) and the physical and geometrical parameters are recorded, initial values of the dependent variables are determined from the initial configuration of the system. This initial configuration also defines initial values of the Euler parameters which in turn can be used to find initial values of the four kinematic arrays: v_{klm} , \dot{v}_{klm} , w_{klm} , and \dot{w}_{klm} . Then by knowing the applied forces and the constraints, initial values of a_{lp} , f_l , b_{ql} , and g_q are obtained. The governing equations can then be solved to find incremental values in the dependent variables and the Euler parameters. The process may then be repeated until the time history of the dynamics of the system is determined.

DYNOCOMBS uses a fourth order Runge-Kutta integrator to numerically solve the governing differential equations.

Example Problems

To illustrate the use of DYNOCOMBS two problems are considered: The first is the spatial motion of a robot - one of the designated example problems of the handbook. Figure 3 depicts the robot system and an animation of the motion. The physical and geometrical data as well as the specified kinematical data are identical to that defined by the handbook editors for this example. Figures 4, 5, and 6 show the computed tower lift, tower rotation, and arm pullout.

The second example is a simulation of a towing chain as depicted in Figure 7. The chain consists of 10 identical pin-connected cylindrical links each with mass 1.0 slug (14.6kg), length 1.0 ft (0.3048m), and diameter 0.5 in (0.0127m). The ends of

the chain are initially separated horizontally by 3.0 ft (0.9144m) and also vertically by 3.0 ft (0.9144m) as shown in Figure 7. Figure 7 also depicts the static equilibrium configuration of the system obtained through DYNOCOMBS by damping gravity induced motion from an arbitrary initial configuration.

From the initial configuration of Figure 7, the lower end of the chain is accelerated at a uniform rate of 3.0 ft/sec² (0.9144m/s²). The resulting movement of the chain during the subsequent 2.0 seconds is shown in Figure 8.

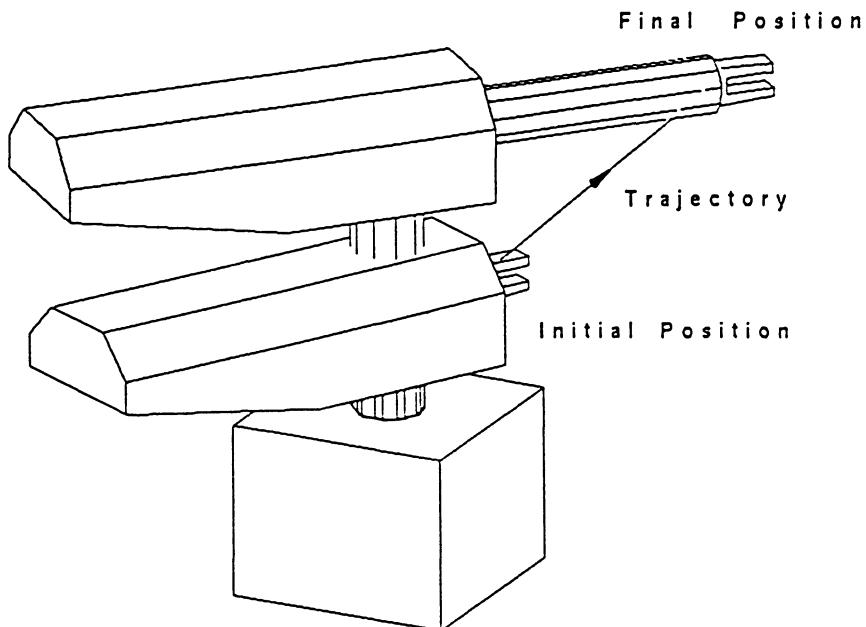


Figure 3. Robot System and Motion Animation

**Example Problem -- Spatial Robot
Lift of Tower (Z1)**

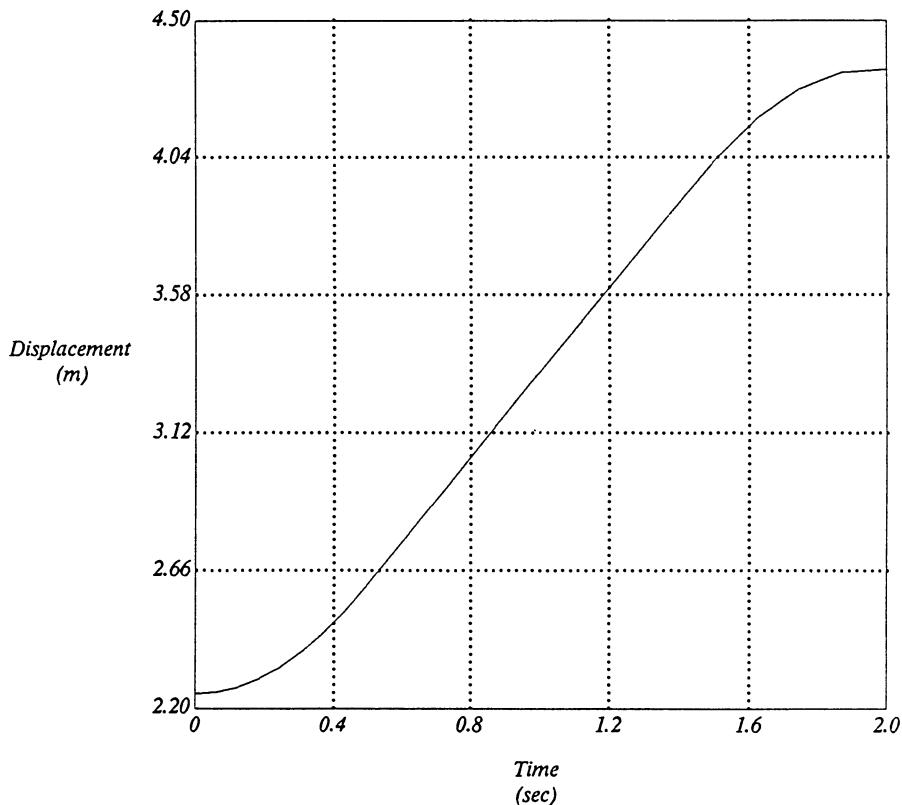


Figure 4. Computed Robot Motion: Tower Lift

**Example Problem -- Spatial Robot
Rotation of Tower (GA1)**

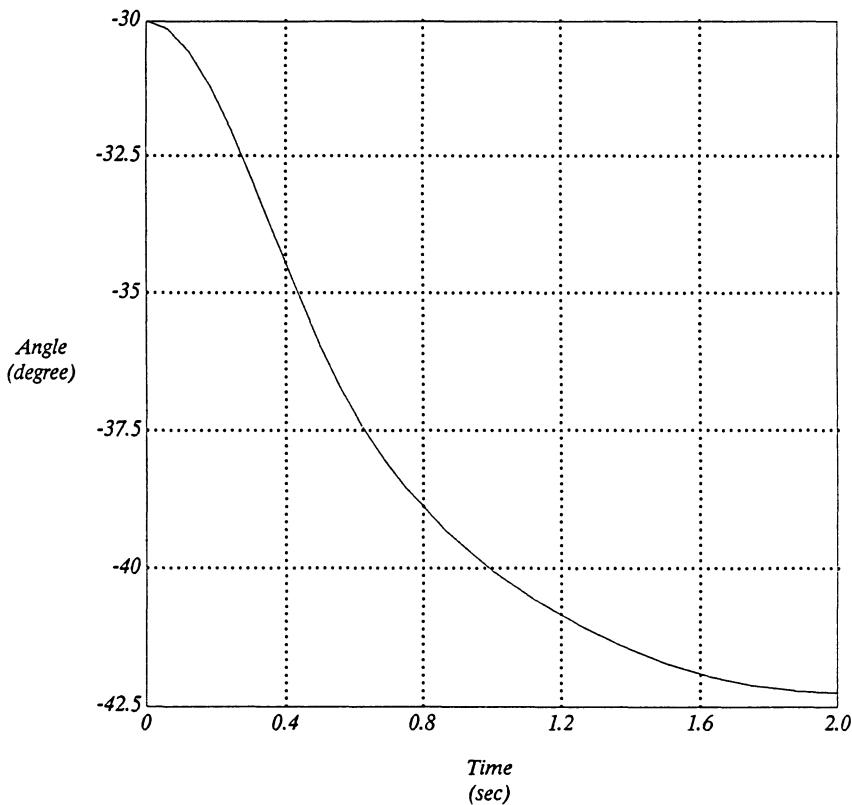


Figure 5. Computed Robot Motion: Tower Rotation

**Example Problem -- Spatial Robot
Pull out of Arm (Y2)**

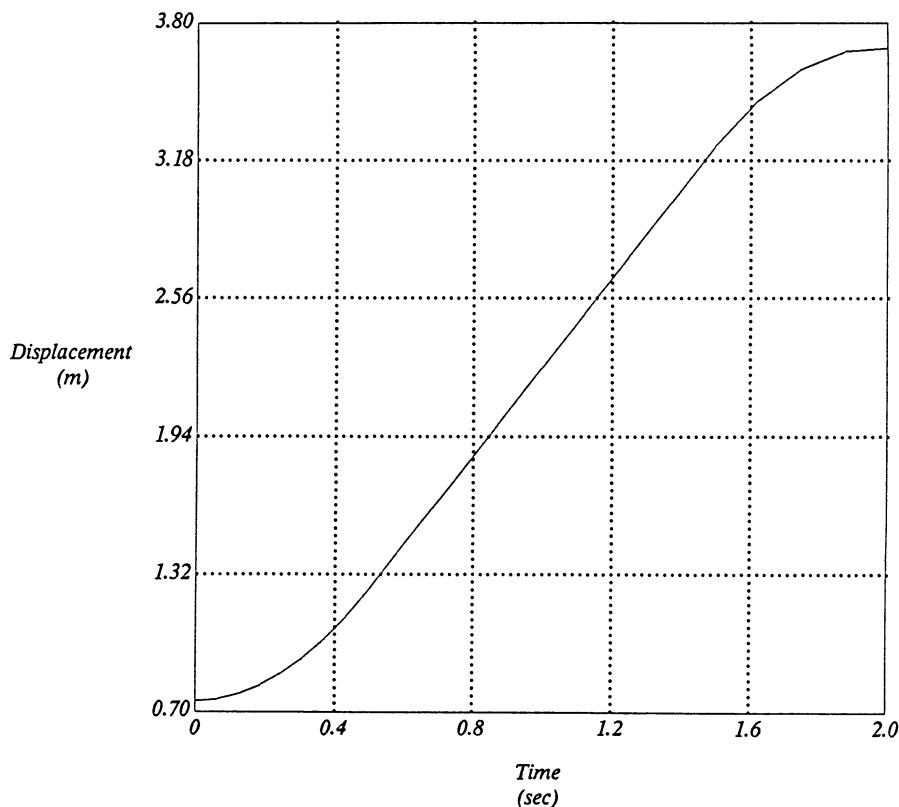


Figure 6. Computed Robot Motion: Arm Pullout

**Example Problem -- Towing Chain
(Static Equilibrium Position)**

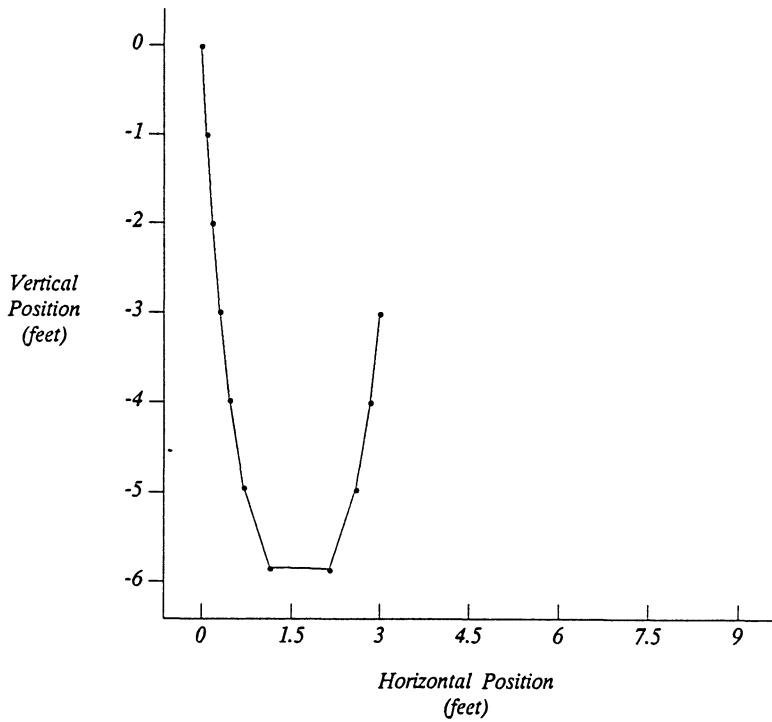


Figure 7. Towing Chain: Initial Configuration

**Example Problem -- Towing Chain
($\Delta t = 0.25$ sec)**

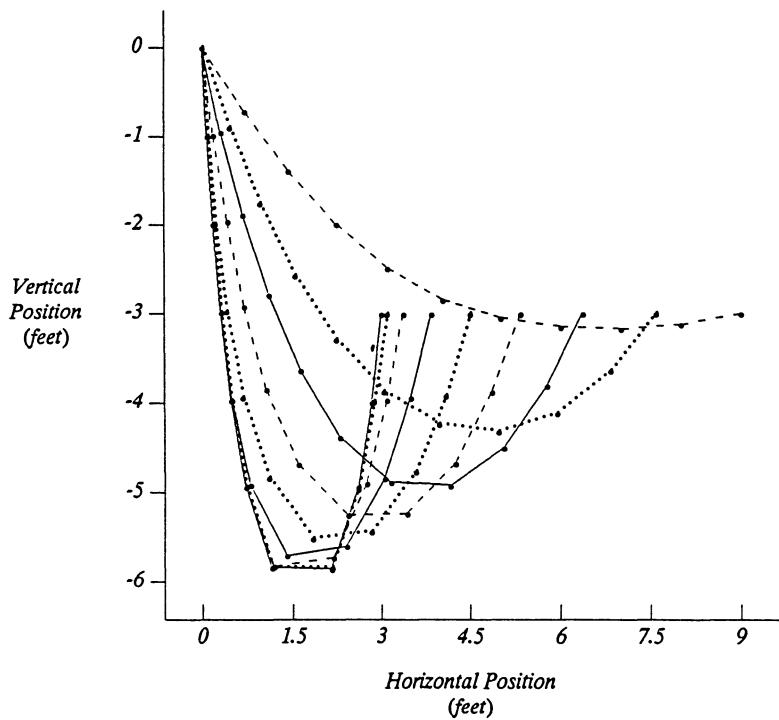


Figure 8. Towing Chain Configuration During 0.25 Second Time Intervals

Acknowledgement

Partial support for research leading to the development of DYNOCOMBS has been provided by the National Science Foundation under Grant MEA-8404414 to the University of Notre Dame and Grant MSM-8612970 to the University of Cincinnati.

References

- [1] Kamman, J. W. and Huston, R. L.: User's Manual for UCIN DYNOCOMBS-II, National Technical Information Services, Springfield, VA, Report No. PB87-216594/A05, 1987.
- [2] Kane, T. R. and Levinson, D. A.: Dynamics: Theory and Applications, McGraw Hill, 1985.
- [3] Huston, R. L.; Passerello, C. E.; Harlow, M. W.: Dynamics of Multirigid-Body Systems, Journal of Applied Mechanics, Vol. 45(4), 1978, pp. 889-893.
- [4] Wang, J. T. and Huston, R. L.: Kane's Equations with Undetermined Application with Constrained Multibody Systems, Journal of Applied Mechanics, 54(2), 1987, pp. 424-429.
- [5] Walton, W. C., Jr. and Steeves, E. C.: A New Matrix Theorem and Its Application for Establishing Independent Coordinates for Complex Dynamical Systems with Constraints, NASA Technical Report TR-326, 1969.
- [6] Lawson, C. L. and Hanson, R. J.: Solving Least Squares Problems, Prentice Hall, 1974.

SPACAR - Computer Program for Dynamic Analysis of Flexible Spatial Mechanisms and Manipulators

by J.B. Jonker, Twente University of Technology,
and J.P. Meijaard, Delft University of Technology,
The Netherlands.

Abstract

In this contribution the theoretical background of the computer program SPACAR is presented. The program is based on the finite element formulation for multi-degree of freedom mechanisms. A brief survey of the program and its major modules is given. Three examples, the mechanism and the robot in this handbook and a cantilever beam vibrating with large amplitude, are included to illustrate the capabilities of the program.

1. Introduction

The key point in the finite element theory on which the program SPACAR is based is the specification of deformation modes in the description of strain, stress and associated stiffness of the elements. This way of finite element discretization forms the algebraic analogue of the continuous field concept for the description of deformations and stresses of deformable bodies (Besseling [2]). The definition of the deformation modes, termed by Argyris [1] "natural modes", includes the specification of rigid body displacements as displacements for which the deformations are zero. If the deformation modes are defined by nonlinear functions of the nodal coordinates, valid for arbitrarily large displacements and rotations, then also mechanisms can be analysed. Characteristic for the present finite element approach to mechanism analysis is that both links and joints are considered as specific finite elements. The kinematic constraints of the elements are obtained by imposing conditions on the deformation modes of the elements. In case of flexible elements, constitutive equations for the stresses have to be supplied.

The finite element formulation for mechanisms has been originated by Van der Werff [10] and, in the 1980's, has progressed with the development of the description for spatial mechanisms with flexible links [4,11]. In these publications an algorithm is presented for the numerical determination of the geometric transfer functions of multi-degree of freedom mechanisms. With the aid of the geometric transfer functions, the equations of motion are directly derived in terms of degrees of freedom. This approach is based on the principle of virtual power (Jourdain's principle [8]) and leads to a minimal system of ordinary differential equations that can be solved numerically in an effective way. In addition, the geometric transfer function formalism provides a systematic approach for generating locally linearized mechanism models.

2. Finite element representation of mechanisms

One of the basic steps in any finite element dynamic analysis of a mechanism is creating its representation as an assembly of finite elements of simple geometry. The links may be modelled by one or more beam elements that may be rigid or deformable, depending on whether the flexibility is expected to play a role in the dynamic analysis.

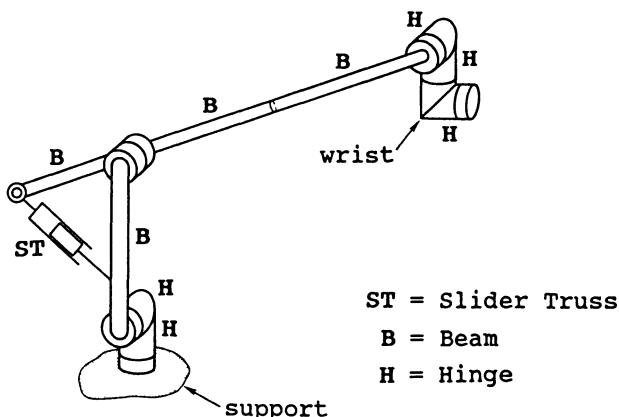


Fig. 1. Manipulator with six degrees of freedom

Fig. 1 shows a manipulator mechanism with six degrees of freedom which is modelled by three different types of elements. The hydraulic cylinder is modelled as an active slider-truss element. The manipulator arms are modelled by beam elements and furthermore we have six cylindrical hinge elements, some of which are actuated by torque servos. Hinge elements are also used to model the wrist of the manipulator.

The location of each element is specified by a set of nodal coordinates $(x_i^k) \in X^k$, some of which may be Cartesian coordinates (x_i^k) of the end nodes, while others describe the orientation of orthogonal triads, rigidly attached at the element nodes, which will be denoted by (λ_i^k) . The superscript k is added to show that a specific element k is considered. We call X^k the configuration space of the element k . The instantaneous values of the nodal coordinates determine a fixed number of deformation modes for the element, which is equal to the number of nodal coordinates minus the number of degrees of freedom of the element as a rigid body. In this way we can define for each element a map

$$D^k : X^k \rightarrow E^k, \quad \text{or} \quad e^k = D^k(x^k), \quad (2.1)$$

where e^k is the vector of deformation mode coordinates, $(e_i^k) \in E^k$, some of which are associated with large relative displacements and rotations (e_i^k) between the element nodes, while others describe small elastic deformations of the element and will be denoted by (ε_i^k) . We call E^k the deformation space of the element k . The deformation mode coordinates must be invariant with respect to rigid body motions of the element. This necessarily implies geometrically nonlinear expressions for the deformation mode coordinates as functions of the nodal coordinates. The deformation functions are chosen in such a way that they have a clear physical meaning. This facilitates the description of strength and stiffness. The map D^k and its derivative maps

$$D_i D^k \equiv \frac{\partial D^k}{\partial x_i^k}, \quad \text{and} \quad D_{ij} D^k \equiv \frac{\partial^2 D^k}{\partial x_i^k \partial x_j^k},$$

play a major role in the derivation of the theory. In the next sections explicit expressions are presented for the deformation functions D^k of the spatial beam element. This element has been selected here to be used wherever the theory will be illustrated at element level.

Fig. 2 shows a spatial beam element in an x, y, z inertial coordinate system. The configuration of the element is determined by the position vectors x^P and x^q of the end nodes and the angular orientation of orthogonal triads (n_x^-, n_y^-, n_z^-) rigidly attached to each end point. In the undeflected state the triads coincide with the axis pq and the principal axes of its cross section. The rotational part of the motion of the (flexible) beam is described by the rotations of the triads (n_x^-, n_y^-, n_z^-) , which are determined by rotation matrices R^P and R^q . If the beam is rigid, then the rotation matrices are identical and in the initial undeflected state they are equal to the identity matrix.

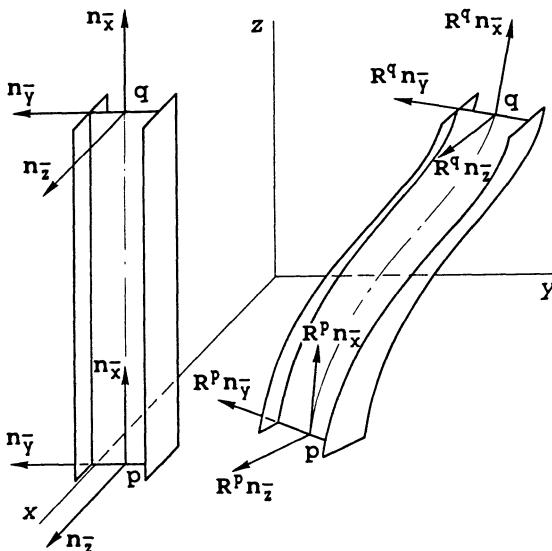


Fig. 2. Beam element, initial and deformed state

The components of the rotation matrices are expressed in terms of Euler parameters [12]. In this case the rotation matrix R can be

written in terms of four Euler parameters $\lambda_0, \lambda_1, \lambda_2, \lambda_3$ as

$$R = \begin{bmatrix} \lambda_0^2 + \lambda_1^2 - \lambda_2^2 - \lambda_3^2 & 2(\lambda_1\lambda_2 - \lambda_0\lambda_3) & 2(\lambda_1\lambda_3 + \lambda_0\lambda_2) \\ 2(\lambda_1\lambda_2 + \lambda_0\lambda_3) & \lambda_0^2 - \lambda_1^2 + \lambda_2^2 - \lambda_3^2 & 2(\lambda_2\lambda_3 - \lambda_0\lambda_1) \\ 2(\lambda_1\lambda_3 - \lambda_0\lambda_2) & 2(\lambda_2\lambda_3 + \lambda_0\lambda_1) & \lambda_0^2 - \lambda_1^2 - \lambda_2^2 + \lambda_3^2 \end{bmatrix}. \quad (2.2)$$

By definition the Euler parameters satisfy the constraint equation

$$\lambda_0^2 + \lambda_1^2 + \lambda_2^2 + \lambda_3^2 = 1, \quad \text{or} \quad \lambda^T \lambda = 1. \quad (2.3)$$

The use of Euler parameters avoids singularity problems associated with Euler angles and has the additional advantage that the components of the rotation matrices are quadratic algebraic expressions in terms of these parameters instead of complicated goniometric functions, as is the case when for instance Euler angles are used; algebraic expressions are computationally more efficient than goniometric functions.

With the vector $l^k = x^q - x^p$, the deformation functions of the beam element can now be written as follows [3]:

$$\begin{aligned} \text{elongation: } \varepsilon_1^k &= D_1^k = \|l^k\| - \ell_0^k + \\ &+ \frac{1}{30\ell_0^k} [2(\varepsilon_3^k)^2 + \varepsilon_3^k \varepsilon_4^k + 2(\varepsilon_4^k)^2 + 2(\varepsilon_5^k)^2 + \varepsilon_5^k \varepsilon_6^k + 2(\varepsilon_6^k)^2], \end{aligned} \quad (2.4^a)$$

$$\text{torsion: } \varepsilon_2^k = D_2^k = [(R^p n_{\bar{z}}, R^q n_{\bar{y}}) - (R^p n_{\bar{y}}, R^q n_{\bar{z}})] \ell_0^k / 2, \quad (2.4^b)$$

$$\text{bending: } \varepsilon_3^k = D_3^k = -(R^p n_{\bar{z}}, l^k), \quad (2.4^c)$$

$$\varepsilon_4^k = D_4^k = (R^q n_{\bar{z}}, l^k), \quad (2.4^d)$$

$$\varepsilon_5^k = D_5^k = (R^p n_{\bar{y}}, l^k), \quad (2.4^e)$$

$$\varepsilon_6^k = D_6^k = -(R^q n_{\bar{y}}, l^k). \quad (2.4^f)$$

Here, $\|l^k\|$ and ℓ_0^k represent the actual length and the reference length of the element; $(,)$ stands for the inner product of two vectors. The terms in the expression for the elongation ε_i^k that are quadratic in the bending deformations represent the longitudinal deformation associated with the deflection of the beam element. The deformation mode coordinates in Eqs. (2.4) possess the proper invariance with respect to rigid body motion of the beam element. Since the expressions for the bending deformations are defined with respect to orthogonal triads oriented according to the element axis and the principal axes of its cross section, they have a clear physical meaning. If the deformations (ε_i^k) remain sufficiently small ($\varepsilon_i^k/\ell^k \ll 1$), then in the elastic range they are linearly related to known beam quantities as normal force σ_1^k , twisting moment σ_2^k and bending moments σ_3^k , σ_4^k , σ_5^k , σ_6^k by the beam constitutive equations [3]

$$\sigma^k = S^k \varepsilon^k, \quad (2.5)$$

where S^k is a symmetric matrix containing the elastic constants.

In addition to the physical deformations we must consider the condition (2.3) to be imposed on the Euler parameters. For each orthogonal triad we introduce a deformation function of the so-called lambda element

$$\varepsilon^k = D^k(\lambda^k) = \lambda^{kT} \lambda^k - 1. \quad (2.7)$$

Then the constraint equation for the Euler parameters can be treated as an undeforability condition ($\varepsilon^k=0$) for the lambda element.

3. Kinematical analysis

A kinematic mechanism model can be build up with finite elements by letting them have nodal points in common. In this way, the configuration spaces of the individual elements can be regarded as subspaces of the mechanism configuration space X , that is

$$X = \sum_k X^k. \quad (3.1)$$

In the same way the element deformation spaces can be regarded as subspaces of the space E of deformation mode coordinates for the entire mechanism. Since deformation mode coordinates (e_i^k) are only related to the element k , E is the direct sum of the spaces E^k , that is

$$E = \bigoplus_k E^k. \quad (3.2)$$

The deformation functions of the individual elements can be taken together in a continuity map for the entire mechanism; we write symbolically

$$D = \sum_k D^k : X \rightarrow E, \quad \text{or } e = D(x). \quad (3.3)$$

The continuity map in (3.3) constitutes the basic equations for the kinematic analysis. The spaces X and E can now be decomposed into subspaces in accordance with the constraint conditions and the choice of the generalized coordinates. We have

$$X = X^0 \oplus X^c \oplus X^m, \quad \text{and} \quad E = E^0 \oplus E^m \oplus E^c, \quad (3.4)$$

where the superscripts $0, c$ and m denote the space of invariant, dependent and independent (generalized) coordinates respectively. The problem now formulated for the kinematic analysis is the determination of the nodal coordinates and deformation mode coordinates for given values of the generalized coordinates (x_i^m, e_i^m) . Hence determine the maps

$$F^x: X^m \times E^m \rightarrow X, \quad \text{or } x = F^x(x^m, e^m), \quad (3.5)$$

$$F^e: X^m \times E^m \rightarrow E, \quad \text{or } e = F^e(x^m, e^m). \quad (3.6)$$

The maps F^x and F^e are called the geometric transfer functions of the mechanism; they express the configuration and deformation state as explicit functions of the set of generalized coordinates. The velocity vectors \dot{x} and \dot{e} at (x, e) can be calculated from Eqs. (3.5) and (3.6) as

$$\dot{\mathbf{x}} = \frac{\partial F^x}{\partial x^m} \dot{\mathbf{x}}^m + \frac{\partial F^x}{\partial e^m} \dot{\mathbf{e}}^m, \quad \text{or} \quad \dot{\mathbf{x}} = DF^x \cdot (\dot{\mathbf{x}}^m, \dot{\mathbf{e}}^m), \quad (3.7)$$

$$\dot{\mathbf{e}} = \frac{\partial F^e}{\partial x^m} \dot{\mathbf{x}}^m + \frac{\partial F^e}{\partial e^m} \dot{\mathbf{e}}^m, \quad \text{or} \quad \dot{\mathbf{e}} = DF^e \cdot (\dot{\mathbf{x}}^m, \dot{\mathbf{e}}^m), \quad (3.8)$$

where (\cdot) denotes differentiation with respect to time. The derivative maps DF^x and DF^e are called the first-order geometric transfer functions. Again differentiating with respect to time yields the accelerations

$$\ddot{\mathbf{x}} = (D^2F^x \cdot (\dot{\mathbf{x}}^m, \dot{\mathbf{e}}^m)) \cdot (\dot{\mathbf{x}}^m, \dot{\mathbf{e}}^m) + DF^x \cdot (\ddot{\mathbf{x}}^m, \ddot{\mathbf{e}}^m), \quad (3.9)$$

$$\ddot{\mathbf{e}} = (D^2F^e \cdot (\dot{\mathbf{x}}^m, \dot{\mathbf{e}}^m)) \cdot (\dot{\mathbf{x}}^m, \dot{\mathbf{e}}^m) + DF^e \cdot (\ddot{\mathbf{x}}^m, \ddot{\mathbf{e}}^m), \quad (3.10)$$

where D^2F^x and D^2F^e are the second-order geometric transfer functions.

We consider now how the geometric transfer functions correspond with the continuity equation (3.3). Substituting Eqs. (3.5) and (3.6) into Eq. (3.3) yields the algebraic relation

$$F^e = D \circ F^x, \quad \text{for all } (x^m, e^m). \quad (3.11)$$

Due to the nonlinear character of the continuity equations the unknown geometric transfer functions F^x and F^e cannot be calculated directly from the equations in (3.11). It will be shown first that expressions for the first-order geometric transfer functions can be obtained from the derivative map DD . Differentiation of Eq. (3.9) yields with the chain rule

$$DF^e = DD(\mathbf{x}) \cdot DF^x, \quad (3.12)$$

The derivative map $DD(\mathbf{x})$ is composed from the derivative maps DD^k of the individual elements. For expressions of the derivative maps DD^k , the reader is referred to [4]. In accordance with the decomposition in Eq. (3.4), the system in (3.12) can be partitioned as

$$\begin{bmatrix} DF^{e^0} \\ DF^{e^m} \\ \hline DF^{e^c} \end{bmatrix} = \begin{bmatrix} D^0 D^0 & D^c D^0 & D^m D^0 \\ D^0 D^m & D^c D^m & D^m D^m \\ \hline D^0 D^c & D^c D^c & D^m D^c \end{bmatrix} \begin{bmatrix} DF^{x^0} \\ DF^{x^c} \\ DF^{x^m} \end{bmatrix} \quad (3.13)$$

The superscripts 0 , c and m combined with the operator D represent partial differentiation with respect to the corresponding nodal coordinates (x_i^0) , (x_i^c) and (x_i^m) . The only unknowns in this equations are the partial maps DF^{x^c} and DF^{e^c} . For the other partial maps we have

$$DF^{e^0} = \begin{bmatrix} \frac{\partial e^0}{\partial x^m} & \frac{\partial e^0}{\partial e^m} \end{bmatrix} = [0, 0], \quad DF^{e^m} = \begin{bmatrix} \frac{\partial e^m}{\partial x^m} & \frac{\partial e^m}{\partial e^m} \end{bmatrix} = [0, I], \quad (3.14)$$

$$DF^{x^0} = \begin{bmatrix} \frac{\partial x^0}{\partial x^m} & \frac{\partial x^0}{\partial e^m} \end{bmatrix} = [0, 0], \quad DF^{x^m} = \begin{bmatrix} \frac{\partial x^m}{\partial x^m} & \frac{\partial x^m}{\partial e^m} \end{bmatrix} = [I, 0], \quad (3.15)$$

where O and I are zero and identity matrices. If the mechanism is not in a singular configuration, then the unknown first-order geometric transfer function DF^{x^c} can be calculated from Eq. (3.13) by

$$DF^{x^c} = \begin{bmatrix} D^c D^0 \\ D^c D^m \end{bmatrix}^{-1} \begin{bmatrix} -D^m D^0 & O \\ -D^m D^m & I \end{bmatrix}. \quad (3.16)$$

For the first-order geometric transfer function DF^{e^c} , we obtain from Eq. (3.13)

$$DF^{e^c} = D^c D^c \cdot DF^{x^c} + D^m D^c \cdot DF^{x^m}. \quad (3.17)$$

Expressions for the second-order geometric transfer functions can be obtained from the derivative map DD in a similar way. By differentiating Eq. (3.12) and using the decomposition in Eq. (3.4) one obtains for the unknown second-order geometric transfer functions $D^2 F^{x^c}$ and $D^2 F^{e^c}$

$$D_{ij}F^{xc} = - \begin{bmatrix} D^c D^0 \\ D^c D^m \end{bmatrix}^{-1} \begin{bmatrix} (D^2 D^0 \cdot D_i F^x) \cdot D_j F^x \\ (D^2 D^m \cdot D_i F^x) \cdot D_j F^x \end{bmatrix} \quad (3.18)$$

and

$$D_{ij}F^{ec} = (D^2 D^c \cdot D_i F^x) \cdot D_j F^x + D^c D^c \cdot D_{ij} F^{xc}. \quad (3.19)$$

With the first-order and second-order geometric transfer functions a new configuration \mathbf{x} with corresponding deformations \mathbf{e} can be approximated from a previous configuration \mathbf{x}_0 and deformations \mathbf{e}_0 by

$$\mathbf{x}_1 = \mathbf{x}_0 + DF_0^X \cdot (\Delta x^m, \Delta e^m) + \frac{1}{2}(D^2 F_0^X \cdot (\Delta x^m, \Delta e^m)) \cdot (\Delta x^m, \Delta e^m), \quad (3.20)$$

$$\mathbf{e}_1 = \mathbf{e}_0 + DF_0^e \cdot (\Delta x^m, \Delta e^m) + \frac{1}{2}(D^2 F_0^e \cdot (\Delta x^m, \Delta e^m)) \cdot (\Delta x^m, \Delta e^m), \quad (3.21)$$

where

$$\Delta x^m = \mathbf{x}_1^m - \mathbf{x}_0^m \quad \text{and} \quad \Delta e^m = \mathbf{e}_1^m - \mathbf{e}_0^m.$$

A Newton-Raphson based iteration process is applied in order to guarantee that ultimately

$$D^m(\mathbf{x}_1) = \mathbf{e}_1^m \quad \text{and} \quad D^0(\mathbf{x}_1) = 0. \quad (3.22)$$

4. Dynamical analysis

4.1. Consistent mass formulation for the flexible beam element

The inertia properties of the distributed mass of the flexible beam element are described by using consistent mass matrices. The derivation of the consistent mass formulation for the flexible beam element is based on the elastic line concept [7]. In order to account for the distributed inertia forces of the element, it is necessary to specify the elastic line configuration of the element in terms of the nodal coordinates (x_i^k, λ_i^k) and the flexible deformation mode coordinates (ϵ_i^k) so that the global position of every point on the elastic line is uniquely determined by these coordinates. At the boundary between the elements, the position and orientation should be

completely determined by the position and orientation of the corresponding node in order to guarantee structural continuity. The global position vector x^s of an arbitrary point s on the beam element in the undeflected state is specified in terms of the Cartesian nodal coordinates (x_i^k) as

$$x^s = \Phi x^k, \quad (4.1)$$

where the coefficient matrix Φ is defined by

$$\Phi = \begin{bmatrix} 1-\xi & 0 & 0 & | & \xi & 0 & 0 \\ 0 & 1-\xi & 0 & | & 0 & \xi & 0 \\ 0 & 0 & 1-\xi & | & 0 & 0 & \xi \end{bmatrix}, \quad (4.2)$$

with ξ being a normalized coordinate ($0 \leq \xi \leq 1$) along the element axis measured from point p . Let $(\varepsilon_i^k, i=3,4,5,6)$ be the flexible deformation mode coordinates representing the bending modes of the element k , defined in Eqs. (2.4). Their geometric meaning is visualized in Fig. 3.

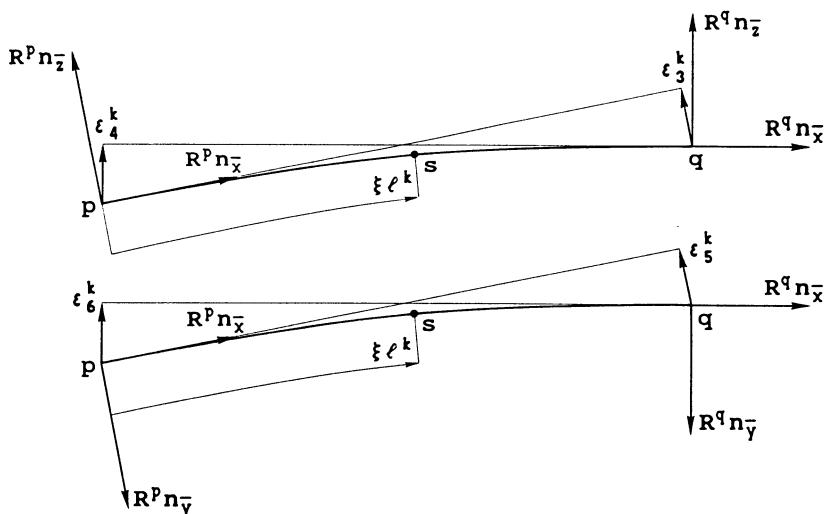


Fig. 3. Visualization of deformation modes representing the deflection of the spatial beam element

Using cubic polynomial interpolations for the bending deformations, the global position vector r^s of point s in the deflected state of the element can be expressed as a vectorial sum of the vector x^s of the point s in the undeflected state and the elastic bending deformation with a displacement field expressed in terms of the flexible deformation mode coordinates (ε_i^k) as

$$\begin{aligned} r^s = x^s + \varepsilon_3^k R^p n_{\bar{z}} (\xi^3 - 2\xi^2 + \xi) + \varepsilon_4^k R^q n_{\bar{z}} (-\xi^3 + \xi^2) + \\ - \varepsilon_5^k R^p n_{\bar{y}} (\xi^3 - 2\xi^2 + \xi) - \varepsilon_6^k R^q n_{\bar{y}} (-\xi^3 + \xi^2), \end{aligned} \quad (4.3)$$

where

$$n_{\bar{z}} = C^k n_z \quad \text{and} \quad n_{\bar{y}} = C^k n_y. \quad (4.4)$$

Here, C^k is an orthogonal matrix relating the principal element axes in the initial undeflected state to the global (x, y, z) coordinate system. The polynomial equation (4.3) shows that the displacement field associated with the bending deformation of the beam element is described with respect to local coordinate systems that coincide with the orthogonal triads $(n_{\bar{x}}, n_{\bar{y}}, n_{\bar{z}})$ at the element nodes p and q. The interpolation functions satisfy differentiability requirements and with expression (4.3) the completeness and continuity requirements for the element are fulfilled.

In order to apply Eq. (4.3) as a basis for the consistent mass formalism of the element we turn to the use of the principle of virtual power. The general consistent mass formulation is found by equating the virtual power of the distributed inertia forces and the virtual power of the equivalent forces at the element nodes. The virtual power of the distributed inertia forces, associated with the elastic line configuration of the beam element, can be expressed by the scalar product

$$-m^k \ell^k \int_0^1 \langle \dot{r}^s, \ddot{r}^s \rangle d\xi, \quad (4.5)$$

where m^k is the mass density of the element per unit of length and ℓ^k is the length of the element. The virtual velocity vector \dot{r}^s and the acceleration vector \ddot{r}^s are obtained by differentiating Eq. (4.3) twice with respect to time. Evaluation of Eq.(4.5) yields

$$\begin{aligned} -m^k \ell^k \int_0^1 & \langle \dot{r}^s, \ddot{r}^s \rangle d\xi = \\ & = -[\dot{x}^{kT}, \dot{\lambda}^{kT}, \dot{\epsilon}^{kT}] \left[\begin{bmatrix} (M^{xx})_c^k & (M^{x\lambda})_c^k & (M^{x\epsilon})_c^k \\ (M^{\lambda x})_c^k & 0 & (M^{\lambda\epsilon})_c^k \\ (M^{\epsilon x})_c^k & (M^{\epsilon\lambda})_c^k & (M^{\epsilon\epsilon})_c^k \end{bmatrix} \begin{bmatrix} \ddot{x}^k \\ \ddot{\lambda}^k \\ \ddot{\epsilon}^k \end{bmatrix} + \begin{bmatrix} (f_{in}^x)_c^k \\ 0 \\ (\sigma_{in})_c^k \end{bmatrix} \right], \end{aligned} \quad (4.6)$$

where

$$\begin{bmatrix} (f_{in}^x)_c^k \\ (\sigma_{in})_c^k \end{bmatrix} = \begin{bmatrix} ((J^x)_c^k \cdot \dot{\lambda}^k) \cdot \dot{\lambda}^k \\ ((J^\epsilon)_c^k \cdot \dot{\lambda}^k) \cdot \dot{\lambda}^k \end{bmatrix} + \begin{bmatrix} ((L^x)_c^k \cdot \dot{\epsilon}^k) \cdot \dot{\lambda}^k \\ ((L^\epsilon)_c^k \cdot \dot{\epsilon}^k) \cdot \dot{\lambda}^k \end{bmatrix}. \quad (4.7)$$

Here, M_c^k is the consistent mass matrix of the element k and $(J_c^k \cdot \dot{\lambda}^k) \cdot \dot{\lambda}^k$ and $(L_c^k \cdot \dot{\epsilon}^k) \cdot \dot{\lambda}^k$ are quadratic velocity vectors representing the rate dependent inertia forces associated with the vectors $\dot{\epsilon}^k$ and $\dot{\lambda}^k$. In [4], expressions are presented for the partitioned mass matrices and for the rate dependent inertia vectors in Eqs. (4.6) and (4.7). It is shown in [4] that the matrices $(M^{x\lambda})_c^k$, $(M^{\epsilon\lambda})_c^k$ and the components of $(J^x)_c^k$ and $(J^\epsilon)_c^k$ in Eq.(4.7) depend linearly on the flexible deformation mode coordinates (ϵ_i^k) . This implies that the dynamics of the spatial beam element, undergoing only a deformation along the length of the element, is completely determined by the translational mass matrix $(M^{xx})_c^k$, defined by

$$(M^{xx})_c^k = m^k \ell^k \int_0^1 [\Phi^T \Phi] d\xi. \quad (4.8)$$

Substituting Eq. (4.2) in Eq. (4.8) yields after evaluation of the integrals

$$(M^{xx})_c^k = \frac{m^k \ell^k}{6} \begin{bmatrix} 2 & 0 & 0 & 1 & 0 & 0 \\ & 2 & 0 & 0 & 1 & 0 \\ & & 2 & 0 & 0 & 1 \\ & & & 2 & 0 & 0 \\ & & & & 2 & 0 \\ & & & & & 2 \\ & & & & & & 2 \end{bmatrix} \quad (4.9)$$

symm.

This is the same mass matrix as occurs in linear finite element analysis representing the consistent mass matrix for a pin-joint bar element [7]. The matrices $(M^{x\epsilon})_c^k$ and $(M^{\epsilon x})_c^k$ represent the principal dynamic coupling between the gross motion and the elastic deformation of the element.

4.2. Equations of motion

By means of the first-order and second-order geometric transfer functions, the equations of motion are formulated in terms of the degrees of freedom, thereby eliminating the constraint forces associated with the rigid link motion. With the notation $DF^T = [DF^{xt}, DF^{et}]$ the equations of motion can be cast in the matrix form

$$[DF^T MDF] \begin{bmatrix} \ddot{x}^m \\ \ddot{e}^m \end{bmatrix} = DF^T \begin{bmatrix} f - f_{in} \\ -\sigma - \sigma_{in} \end{bmatrix} - DF^T M \cdot (D^2 F \cdot (\dot{x}^m, \dot{e}^m)) \cdot (\dot{x}^m, \dot{e}^m). \quad (4.10)$$

Here, $[DF^T MDF]$ denotes the system mass matrix, f the vector of externally applied nodal forces and σ the stress vector which describes the loading state of the elements constituting the mechanism. The vectors f_{in} and σ_{in} represent the rate dependent inertia forces of flexible elements, defined in Eq. (4.7).

The stresses of the flexible elements are characterized by Hooke's law as defined in Eq. (2.5). In case of generalized coordinates (e_i^m) associated with large relative displacements and rotations, constitutive equations describing the behaviour of built-in driving actuators can be added to the system of equations. In this way it is possible to study the dynamics of

active spatial mechanisms such as manipulators and robot mechanisms [4]. The equations of motion form a non-linear system of ordinary differential equations of second order and describe the general case of coupled rigid link motion and small elastic deformation.

4.3. Linearized equations of motion

Using a Taylor series expansion of Eq. (4.10) about a nominal trajectory $(\mathbf{x}_0^m, \mathbf{e}_0^m)$ and disregarding second and higher order terms yields the linearized equations of motion

$$[\mathbf{M}_0] \begin{bmatrix} \delta \ddot{\mathbf{x}}^m \\ \delta \ddot{\mathbf{e}}^m \end{bmatrix} + [\mathbf{C}_0 + \mathbf{D}_0] \begin{bmatrix} \delta \dot{\mathbf{x}}^m \\ \delta \dot{\mathbf{e}}^m \end{bmatrix} + [\mathbf{K}_0 + \mathbf{G}_0^F + \mathbf{G}_0^K] \begin{bmatrix} \delta \mathbf{x}^m \\ \delta \mathbf{e}^m \end{bmatrix} = [\mathbf{DF}^{\mathbf{x}^T}, \mathbf{DF}^{\mathbf{e}^T}] \begin{bmatrix} \delta \mathbf{f} \\ -\delta \sigma \end{bmatrix}. \quad (4.11)$$

The coefficient matrices may be divided into the symmetric matrices \mathbf{M}_0 , being the system mass matrix, \mathbf{D}_0 , the damping matrix, \mathbf{K}_0 , the system stiffness matrix and \mathbf{G}_0^F , the geometric stiffness matrix, and the non-symmetric matrices \mathbf{C}_0 and \mathbf{G}_0^K , being the geometric damping matrix and the dynamic stiffness matrix respectively. The coefficients subscribed with a \circ are evaluated along the nominal trajectory whereas the perturbation of a coordinate is denoted by δ . The nominal trajectory determines the position, velocity and acceleration of the mechanism with the restriction that all flexible deformations of the links are suppressed. The calculation of the coefficient matrices in Eq. (4.10) fits naturally in the framework of the geometric transfer function formalism [4]. The linearized equations are of interest from both analysis as well as control point of view. For analysis, they enable us to study the stability of highly complex dynamic systems. From the point of view of manipulator control the linearized equations provide a basis for the development of reduced order linearized models suitable for control system design.

5. The program SPACAR

The computer program SPACAR is based on the finite element theory for multi-degree of freedom mechanisms as presented above. The program which is written in FORTRAN-77 language is capable of analysing the dynamics of spatial mechanisms and manipulators with flexible links. The program system contains six modules, which obtain their input from format free user supplied data. For every module appropriate key words have been defined. In the following a short description of every module will be given. The functional connections between the modules are illustrated in Fig. 4.

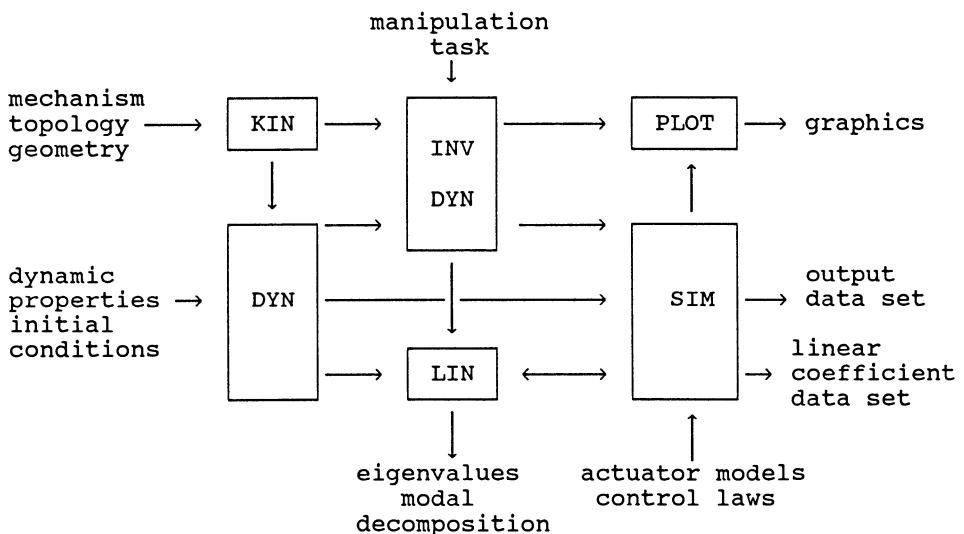


Fig. 4. Functional relations between modules in SPACAR

The kinematics module, KIN, analyses the geometry of motion of the mechanism. The kinematic properties of the motion are specified by the geometric transfer functions. The following steps are provided by the kinematics module:

- o Definition of the mechanism (topology and geometry).
- o System preparation.
- o Calculation of the geometric transfer function values.

The dynamics module, DYN, generates the equations of motion in a form suitable for numerical integration. Furthermore, it generates and solves the equations for the kinetostatic analysis. The inertia and stiffness properties and the initial conditions should be available from data.

The inverse dynamics module, INVDYN, performs the inverse kinematics and dynamics and generates the setpoints for the simulation of manipulator dynamics. The desired motion of the manipulator is described by means of manipulation tasks provided by the user.

The linearization module, LIN, utilizes the outputs from the dynamics module and the inverse dynamics module for the determination of the linearized equations of motion. The linearization module provides for:

- o The evaluation of the matrix coefficients in the linearized equations of motion.
- o The calculation of the eigenvalues and eigenvectors of the linearized equations.
- o Time-domain solution of the linearized equations that describe the perturbation response of the mechanism about the predetermined nominal motion.

The simulation module, SIM, utilizes the outputs of the previously introduced modules for the purpose of dynamic simulation. Several integration routines are embedded in the simulation module to calculate the time-domain solution of the nonlinear differential equations of motion. For the simulation of the dynamics of active mechanisms and manipulators, actuator models and control laws can be provided through user supplied subroutines. All output data is stored in separate datasets and can be plotted afterwards by the PLOT program. For a description of the SPACAR modules, execution programs, keywords and standard tests, the reader is referred to the SPACAR documentation [5].

6. Examples

6.1. Mechanism

The bodies of the first test example are modelled by plane beam elements, one element for each limb, and the spring is modelled by a plane truss element. The masses are replaced by equivalent nodal masses. The equation of motion for the generalized coordinate $\beta(t) - \beta(0)$ is integrated by the routine of [9]. In Fig. 5 the angle β is given and Fig. 6 shows the mean normal force in the coupler K2.

6.2. Robot

Because of the discontinuities in the drive functions, the equations of motion for the second test example are integrated by the classical fourth-order Runge-Kutta method with a stepsize of 0.05 s. Fig. 7 shows the lift of the tower z_1 , the pull-out of the arm y_2 and the yaw angle γ_1 .

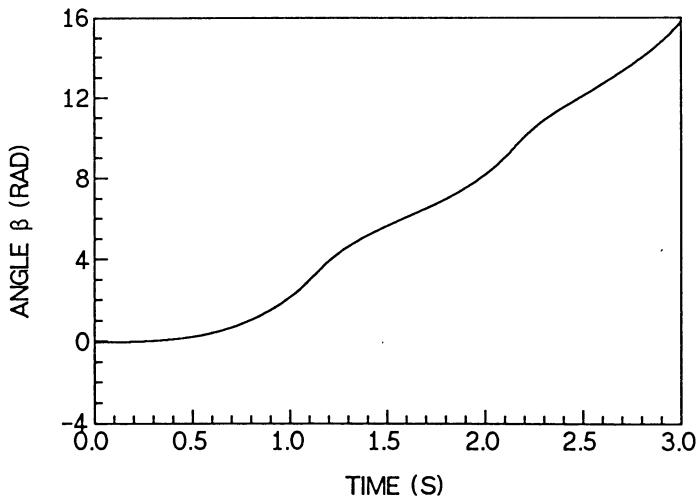


Fig. 5. Angle β

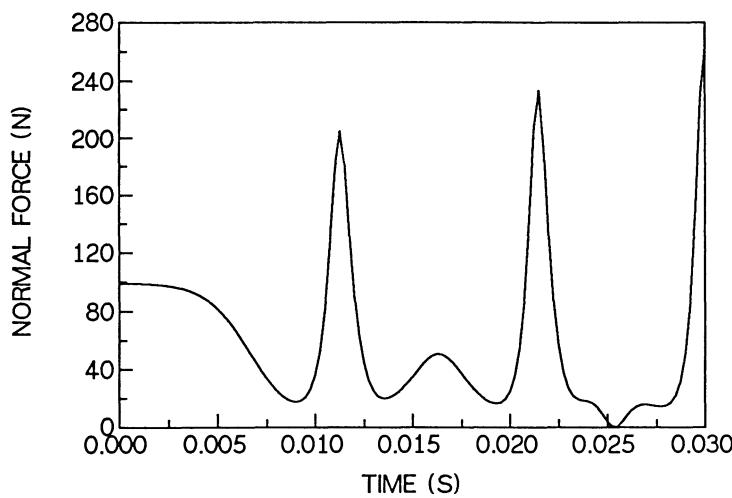


Fig. 6. Normal force in coupler K2

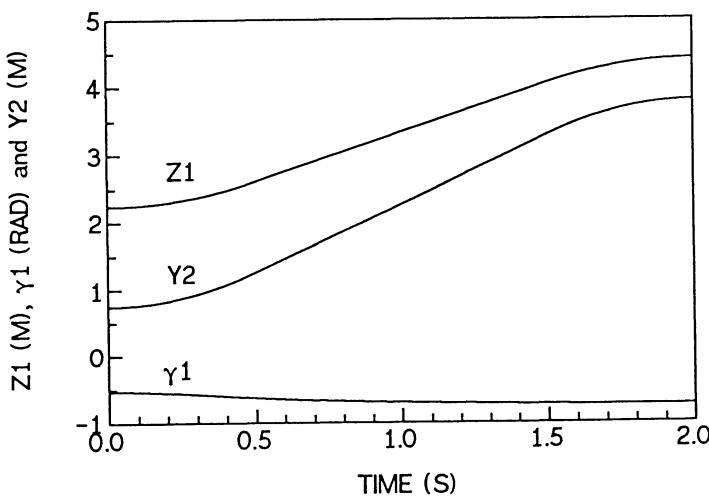


Fig. 7. Configuration of the robot

6.3. Cantilever beam

We illustrate the capabilities of modelling flexible bodies by an additional example. A clamped-free beam is bent to a semicircle by an end moment and then released. The length of the beam is 3 m, its flexural rigidity 1 Nm^2 , its normal stiffness infinite, its mass per unit of length 1 kgm^{-1} , and no damping is modelled. The beam is subdivided in six plane beam elements with a length of 0.5 m. The motion is simulated over 20 seconds; Fig. 8 shows the tip position: the x -direction is perpendicular to the undeflected beam axis and the y -direction is along this axis. Note that high frequency components are present. More accurate results can be obtained by subdividing the beam in more elements.

Other examples can be found in [4] and [6].

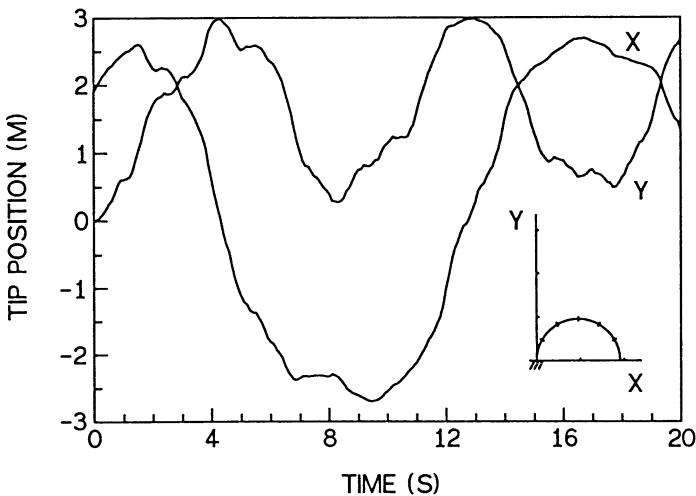


Fig. 8. Tip position of the cantilever beam

References

- [1] Argyris, J.H., et al., Finite Element Method - The Natural Approach, Comp. Meths. Appl. Mech. Eng. 17/18 (1979), pp.1-106.
- [2] Besseling, J.F., The Complete Analogy between the Matrix Equations and the Continuous Field Equations of Structural Analysis, International Symposium on Analogue and Digital Techniques Applied to Aeronautics, Liège, 1963, pp.223-242.
- [3] Besseling, J.F., Nonlinear Theory for Elastic Beams and Rods and its Finite Element Representation, Comp. Meths. Appl. Mech. Eng. 31 (1982), pp.205-220.
- [4] Jonker, J.B., A Finite Element Dynamic Analysis of Flexible Spatial Mechanisms and Manipulators, Doctor's Thesis, Delft University of Technology, 1988.
- [5] Jonker, J.B., et al., SPACAR User Manual and System Description, Delft University of Technology, Laboratory for Engineering Mechanics, 1988, report TM872.
- [6] Jonker, J.B., A Finite Element Dynamic Analysis of Spatial Mechanisms with Flexible Links, to appear in Comp. Meths. Appl. Mech. Eng., 1989.
- [7] Przemieniecki, J.S., Theory of Matrix Structural Analysis, McGraw-Hill, New York, 1968.
- [8] Schiehlen, W., Technische Dynamik, Teubner, Stuttgart, 1985.
- [9] Shampine, L.F., and Gordon, M.K., Computer Solution of Ordinary Differential Equations, The Initial Value Problem, W.J. Freeman, San Francisco, CA, 1975.
- [10] Werff, K. van der, Kinematic and Dynamic Analysis of Mechanisms, A Finite Element Approach, Doctor's Thesis, Delft University of Technology, 1977.
- [11] Werff, K. van der, and Jonker, J.B., Dynamics of Flexible Mechanisms. In: Computer Aided Analysis and Optimization of Mechanical System Dynamics, Ed. E.J. Haug, Springer-Verlag, 1984, pp.381-400.
- [12] Wittenburg, J., Dynamics of Systems of Rigid Bodies, Teubner, Stuttgart, 1977.

NBOD & DISCOS – Dynamic Interaction Simulation of Controls and Structure

by Harold P. Frisch
NASA Goddard Space Flight Center
Greenbelt, Maryland U.S.A.

ABSTRACT

The general purpose multibody dynamics program NBOD and its follow-on companion DISCOS were developed with the expressed purpose of supporting spacecraft attitude control system design and analysis needs. During the past 2 decades, the interplay of NASA requirements with a desire to maximize software application generality has guided their development. The intent of this chapter is to chronologically outline the major technological advancements that enabled the development of these codes. Underlying theory is presented at the highest level with an emphasis on providing engineering insight needed for basic understanding and to judge program suitability for particular applications.

Introduction

During the early 1960s, NASA spacecraft designers were conceiving and building spacecraft with innovative controllers faster than project support engineers could derive, code, and debug system stability and performance analysis programs. In a desperate attempt to reduce analysis cost, several efforts were initiated within the aerospace community to create a general purpose design and analysis capability. Researchers were convinced that it was feasible; project support engineers were skeptical; they felt that program complexity, validation, development, and computational cost problems could never be overcome. Implementation and need realities soon tempered the enthusiasm and skepticism of both groups. Program developers had great difficulty meeting delivery schedules,

securing enthusiastic user groups and, hence, finding sustained sources of support. Project engineers had no recourse for complex multibody dynamics problems; they were forced to use the new capabilities, and only then did they slowly begin to believe in their predictive capabilities. Despite this adverse situation, several groups developed in-house propriety codes with impressive capability, and the NASA/GSFC brought both NBOD & DISCOS into the public domain. These two multibody codes are still in active use within the aerospace project support community. User groups within other communities, such as biodynamics, are developing.

NBOD's Origin

From the perspective of NBOD development, the program SADII by Velman and Haupt, at Hughes Aircraft, and the theoretical work of Hooker, at Lockheed, were foundational. In 1967 NASA/GSFC and Hughes Aircraft were collaborating to develop a dual-spin spacecraft attitude control system design capability. At the same time research groups at Hughes Aircraft, UCLA, Aerospace Corp, NASA/JPL, TRW, Lockheed, et al. were introducing multibody theory to the aerospace community. The program SADII applied evolving general purpose multibody theory to the dual-spin research activities at Hughes. An outline of its theoretical basis can be found in Reference 1; unfortunately, all detailed equation development work exists in several 1967 limited distribution interdepartmental correspondences by John Velman and Marty Haupt.

The theoretical foundation of SADII and its GSFC follow-on successor NBOD is Velman's nested body approach. Rather than viewing a multibody system as N individual bodies, Velman recognized that more computationally efficient recursive relations would result if the system was viewed as N nests of bodies. The i-th nest is defined as all bodies outboard of the i-th hinge with hinge 1 defined as body 1's connection to inertial space.

Classical Newton-Euler methods are used to define equations of motion for each nest. These are obtained by equating time rate of change of momentum for each nest to the sum of all external loads acting on the nest. The external load set includes all constraint loads acting on the nest at the hinge point which connects it to the inboard portion of the multibody system. Constraint and other loads internal to the nest cancel because they appear in equal and opposite pairs. The major limitation imposed in SADII and NBOD is that relative motion between rigid bodies is restricted to be rotational; there was no need then to allow a general translation capability. Furthermore, from our perspective, multibody theory had not yet advanced to that stage. The rotation only limitation permitted the analytic elimination of all forces of constraint within the system. The state

vector for resultant vector-dyadic equations of motion contained an inertial angular velocity vector for each nest and the inertial linear velocity vector for the composite system. Velman recognized that more computationally efficient recursive relations would result if each inertial angular velocity vector was rewritten as a sum of relative velocity vectors. The equations in final matrix vector-dyadic format are

$$\mathbf{I} \dot{\boldsymbol{\omega}} = \boldsymbol{\eta} + \mathbf{P}_c + \mathbf{P}_e + \mathbf{P}_h,$$

where \mathbf{I} is a symmetric inertia matrix of dyadic expressions that can each be computed via recursive relations. $\boldsymbol{\omega}$, a column matrix of relative velocity vectors, $\boldsymbol{\eta}$, a column matrix of gyroscopic cross coupling loads, \mathbf{P}_c , \mathbf{P}_e , and \mathbf{P}_h are respectively the column matrices of hinge constraint loads, external loads, and all active control and passive loads acting at nest hinge points.

Determination of the hinge constraint load vector \mathbf{P}_c was a major computational problem for SADII. The solution method used a Lagrange multiplier technique which culminated with a matrix inversion step. Once this was accomplished and the result included in the motion equations, a second matrix inversion was required to set equations up in a format compatible with numerical integration. Two matrix inversions per integration cycle were intolerable. This effectively limited the application life of SADII at NASA/GSFC.

The constraint torque computation problem was resolved in Reference 2 by Hooker in 1970. There Hooker defines a vector space in which motion occurs and an orthogonal vector space in which motion cannot occur. By the simple use of unit vectors defined along each Euler rotation axis at each hinge, the vector space in which motion is allowed is defined. All constraint load vectors are orthogonal to this vector space. The resultant set of scalar equations used for numerical integration is obtained by a sequence of vector dot products that project vector-dyadic equations of motion into the space in which motion is allowed. Since all constraint loads are orthogonal to this vector space, their projections are identically equal to zero and need not be computed. Topology dependent patterns can be used to optimize computational speed at this step. It was immediately clear that the elimination of one matrix inversion per step would significantly enhance computational speed. This provided the justification to develop a new generation multibody code, namely NBOD.

NBOD makes use of Hooker's concept to obtain the scalar equations of motions needed for numerical integration. To satisfy evolving spacecraft control analysis needs, Velman's original equations were redeveloped to accept momentum wheels within rigid bodies;

all rigid bodies may be gyrostats, to accept point masses at limb ends and to accept body flexibility. Gyrostats were essential for spacecraft control system modeling, point masses were needed to model nutation dampers and flexibility was needed to model solar arrays, antennas, and other flexible appendages. There was never any need or thought given to including a topological loop capability. Provision was also made for arbitrary user-defined nongyroscopic loads, such as those associated with an active attitude control system. These can be defined by any user-coded set of nonlinear algebraic and differential equations. NBOD provides a clean interface between the equations of motion solution routines and the user specified inputs.

NBOD Lessons

The ability to define a gyrostat with one or more high-speed momentum wheels was an absolute necessity for spacecraft controls analysis. It is not practical to model symmetric momentum wheels as distinct bodies; determination of associated kinematics is computationally intensive and not needed for overall system state determination. The inclusion of a point mass capability has proven to be useful in a variety of situations. The inclusion of a flexible body capability was determined to be feasible; however, it became apparent while coding the relevant equations that computational speed would be slow and that a better theoretical foundation was needed. Older versions of NBOD still retain a terminal flexible body capability; the newest versions have had that capability removed. The original numerical integration method was fixed step fourth order Runge Kutta; other methods can and have been easily substituted. An ability exists to input engineering judgement commands. Many physically realizable inertia and gyrodynamics loading effects are identified; users may inhibit recomputation of those which are judged to be near constant or zero. In application, the balance between computational speed improvement and associated numerical error accumulation is poor. The capability is now rarely used. A default output data subroutine exists. Users are expected to edit it to suit problem unique needs and to then pass data to a data plotting facility. Users are expected to write a subroutine to define all nongyroscopic loads such as nonlinear springs, dampers, motors, and external loads. Through many years of application experience, this user demand has proven to be most desirable. An analogous demand has therefore been placed upon DISCOS users. For diagnostic purposes, users may request an annotated listing of all equations of motion in a vector-dyadic format as numerical integration is proceeding. References 3, 4, and 5 are still the primary references for NBOD; new versions differ only in cosmetics. Those in need of a flexible body capability are referred to DISCOS. As a point of reference, NBOD is based upon an Order N^2 recur-

sive algorithm, with less generality but still very similar to what is currently being rediscovered in the 1987+ literature. It has been shown through the years that, when applicable, NBOD's recursive algorithm will consistently run several times faster than DISCOS's Order N³ algorithm.

Table 1 is provided to illustrate the pitfalls associated with any attempt to make definitive statements relative to computational speed without providing associated problem detail. The problem solved is known as the Wittenburg chain. It consists of a folded chain of 6 rigid bodies with one degree of freedom (dof) at each hinge. The gimbal axes at either end of each body are orthogonal to each other. The root body is fixed, while the rest of the chain drops in a gravity field. Extremely high accelerations are encountered during the resultant whipping action of the chain; various levels of damping are used to moderate associated accelerations. CPU seconds for a 10-second simulation are provided. In Table 1, different integration methods are explored for three different levels of hinge viscous damping (strong, moderate, and weak).

Table 1. DISCOS/NBOD RELATIVE SPEED
CPU seconds for 10 seconds real time

Viscous Damping Strong		Viscous Damping Moderate		Viscous Damping Weak	
NBOD FxRK	1023	NBOD FxRK	1009	NBOD FxRK	1005
NBOD RKF	1106	NBOD RKF	644	NBOD RKF	995
NBOD ABM	695	NBOD ABM	183	NBOD ABM	385
NBOD BDF	1065	NBOD BDF	1893	NBOD BDF	4481
DISCOS FxRK	3726	DISCOS FxRK	3696	DISCOS FxRK	2472

FxRK — Fixed Step Runge Kutta
RKF — Runge Kutta Fehlberg

ABM — Adams Bashford Moulton
BDF — Backward Differentiation

DISCOS's Origin

In 1972 NASA made the decision to initiate a major effort to develop a new state of the art multibody dynamics capability. In 1973 proposals were received and evaluated. A contract was awarded to Martin Marietta, Denver, for the development of DISCOS. As with NBOD, evolving needs dictated program contents. Body flexibility, 6 degrees of freedom between contiguous bodies, linearization of equations of motion, and single input-single output transfer function frequency domain analysis were major capability goals. By 1977 all initial objectives were met, and DISCOS was released into the public domain. From 1977 to 1979 additional capability was identified, developed, and added to produce the current version of DISCOS. It was released in 1979. It should be remarked

that the DISCOS development team's primary focus was on providing a simulation capability for very complex problems; it was assumed that special purpose programs would be developed for simple problems. As a consequence, the team did not feel compelled to create extensive internal coding logic to recognize and solve simple textbook type problems with an efficiency comparable to special purpose codes.

During the 1979 to 1987 period, DISCOS capability satisfied NASA's needs. Furthermore, there were no major analytic advancements that could be used as justification for the development of a DISCOS upgrade. In 1988 NASA's commitment to the development of a space station flight telerobotic servicer and its ongoing large flexible spacecraft analysis needs, coupled with major analytic advances in multibody equation formulation and in the availability of low-cost parallel processor work stations, has provided the justification needed to initiate a major upgrade to DISCOS; namely, Order N DISCOS.

DISCOS Flexible Body Characterization Theory

The theoretical foundation of DISCOS rests on the work of DISCOS coauthors Bodley and Park, provided in Reference 6. In that work a general formulation is used to determine the influence of structural flexibility on the dynamic response of spinning spacecraft. The formulation uses a general Lagrange approach to determine the equations of motion for the j-th body; these are each of the form

$$\dot{U}_j = M_j^{-1} [G_j + B_j^T * \lambda]$$

where U_j contains all body j velocity state variables, M_j is its mass matrix, G_j all state and time dependent forces acting on body j, and $B_j^T * \lambda$ all body j interconnection constraint forces. λ is a column matrix of to-be-determined Lagrange multipliers. Unlike NBOD and many other multibody codes, DISCOS does not invert a large order system mass matrix. The order of the symmetric body j mass matrix that it must invert is $6 +$ the number of flexible body modes + the number of embedded wheels; this is not a major computation problem. The computational burden in DISCOS is associated with inverting the matrix needed to determine the Lagrange multipliers of column matrix λ .

The incorporation of body flexibility is based upon the following modal expansions for body inertia and momentum:

$$\text{Body Mass} \quad M = M_0$$

$$\text{Moment of Inertia} \quad J = J_0 + J_{1m} * Q_m + J_{2m,n} * Q_m * Q_n$$

$$\text{Static Mass Moment} \quad S = S_0 + S_{1m} * Q_m$$

$$\text{Linear Momentum} \quad L = L_0 + L_{1m} * \dot{Q}_m$$

$$\text{Angular Momentum} \quad H = H_0 + H_{1m} * \dot{Q}_m + H_{2m,n} * Q_m * \dot{Q}_n$$

where repeated index implies summation, Q_m are modal coordinates. Coefficients $[M_0, J_0, S_0, L_0, H_0]$ are mode independent (rigid body), $[J_1, S_1, L_1, H_1]$ linear mode dependent, and $[J_2, H_2]$ quadratic mode dependent. All linear and quadratic terms are retained within DISCOS computation. Each coefficient is defined as a mode dependent volume integral over the j-th flexible body. All mode dependent gyrodynamic terms within the equations of motion are functionally defined in terms of these coefficients. Either orthogonal modes from conventional eigenanalysis or non-orthogonal modes from any other modal synthesis procedure can be used. There are no assumptions within DISCOS that place restrictions on modal synthesis procedure method. The need for not placing modal synthesis procedure restrictions stems from lessons learned during the development of Reference 7. Through the years this freedom of choice has proven to be essential for the characterization of flexible bodies interior to the multibody system.

Disparaging comments have recently appeared in the literature relative to neglected terms in DISCOS and all other multiflexible body codes, Reference 8. In response, users of multiflexible body software codes MUST take problem specific demands into account when defining a simulation model. The modeling of the dynamics of a cantilever beam on a moving base should have been performed with a chain of spring-connected rigid or flexible bodies. This problem is an excellent example of how important gyrodynamic effects will be lost if due consideration is not given to problem specifics and the modeling limitations imposed by the underlying methods used in multiflexible body theory to characterize body flexibility.

Constraints and Kinematics of Hinge and Sensor Points

A hinge point defines where a body is to be connected to a contiguous body; sensor points are all other points at which kinematic information is needed. For example, momentum wheels, spacecraft attitude sensors, and control actuators are located at sensor points. During NBOD application it was recognized that a lumped stiffness model of a complex antenna support mechanism could not be modeled by a simple rotational hinge. DISCOS

permits 6 degrees of relative freedom at every hinge point. Hinge-point reference frames are defined on each body. Users then define kinematic constraint conditions between contiguous pairs of hinge-point reference frames. Each of the six relative degrees of freedom may be defined as either kinematically constrained (fixed), free, or rheonomic (defined by an a-priori function of time and state.) To support antenna deployment studies through lockup and studies of antenna slewing between hard stops, it was also found necessary to include a capability to allow dof's to switch between free and fixed with switching logic provided via user-defined code. Constraints associated with rolling and sliding along curved arcs and surfaces are not easily implemented; this is one of the subject areas under consideration for Order N DISCOS.

Several major technology advances were required to successfully implement the time dependent constraint capability. Within the formulation for setting up the matrix equation which defines the Lagrange multiplier vector λ , provision had to be made for user-defined constraint functions. These functions provide the ability to prescribe dof motion by an a-priori function of time and state. Provision also had to be made to allow dof's to switch back and forth between fixed and free. The dof switching capability required two innovations. To properly implement switching logic, the software system had to know if switching would occur before the end of the integration step. This problem was resolved by the creation of a new fourth order Runge Kutta integration algorithm. It provided a second order estimate of the answer on the second cycle through. This information was used to set up switching logic. The next problem was numerical noise that reflected itself through the computation as poorly satisfied kinematic constraint conditions. It stemmed from the fact that dof's were not switched at exactly the right instant. The problem was resolved by a new constraint stabilization procedure. At the end of every integration step, the degree to which all kinetic constraints are satisfied is monitored. Deviations are viewed as the result of an extraneous impulsive load to the system. If the deviation is small, then the fictitious impulsive load can be computed and applied back to the system in an equal and opposite manner. This not only helped solve the dof switching problem but it also decreased numerical error as measured by various conservation of laws.

In order to solve for kinematic state variables and for modal displacement, velocity transformation equations are developed that relate absolute velocity state variables to relative velocity. The details of these transformations are quite complex. In addition to the computation of transformation matrices between hinge frames, body frames, and the inertial frame, transformation matrices are also developed for all sensor point reference frames. Users are free to use these in their development of all nongyroscopic loads. As with NBOD, DISCOS requires a user-supplied subroutine to define nongyroscopic loads relative to kinematically free dof's. There are no user restrictions.

Topological Loops

For topological trees, the solution of the kinematic transformations from absolute to relative coordinates is deterministic; for topological loops, it is not. DISCOS does not allow topological loops of rigid bodies. If a topological loop exists, at least one of the bodies in the loop must be flexible, modes must be such that the transformation matrix between absolute and relative coordinates is nonsingular. On a few occasions this capability has been used. The capability works BUT. Mode selection is nontrivial and high frequencies usually enter in such a manner that numerical integration is costly. The topological loop capability was not part of initial DISCOS development plans. It was included to the degree possible without causing major programmatic changes. In essence, the flexible body requirement brings added degrees of freedom into the state velocity vector. If modes are properly chosen, the velocity transformation matrix becomes nonsingular and the determination of relative velocity components is deterministic. Users needing a topological loop capability are referred to other multibody codes. Current plans are to include a general topological loop capability within Order N DISCOS. It will retain the ability for users to define functional relations that kinematically constrain and release dof's anywhere within the system. NASA's current interest in flight telerobotics and man/machine interaction dynamics are demanding this expansion of capability.

DISCOS Linearized Equations of Motion

A control system design and frequency domain analysis capability for DISCOS was an essential part of the DISCOS development effort. This effort required major innovations to transform desire into reality. From the beginning, it was known that simply linearizing equations of motion and setting up transfer functions between generalized input and output coordinates was not wanted. Users had to be given the ability to define physically realizable "sensor signals" to characterize the plant output state. It was also necessary for users to define physically realizable "torque signals" to characterize the plant input state. Associated sensor and actuator dynamics are considered to be part of the controller and, hence, part of the user-supplied component of DISCOS simulation software. The need was to create open-loop, closed-loop and quasi-open-loop transfer functions between torque signal input and sensor signal output.

To achieve these objectives, users must define both sensor and torque signal equations. By definition, sensor signals are functions of plant state variables, and torque signals are functions of control system state variables. The linearization procedure linearizes both plant and control state equations, along with all sensor and torque signal equations

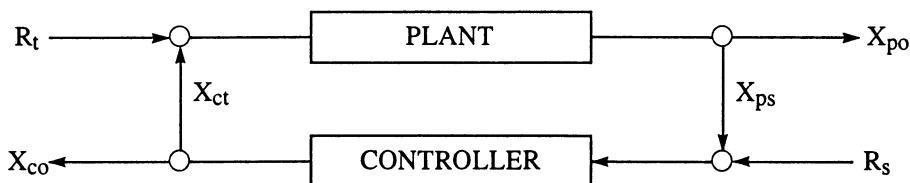
about a user-definable equilibrium state. The linearization procedure is a specially designed quadratic interpolation algorithm. By intent it is designed to fail if an ill-conditioned computing problem is being created. Users may alter failure criteria. However, they are advised to give more thought toward the interpretation of diagnostic information to locate the source of the ill-conditioning problem. They should then upgrade their original simulation model.

DISCOS Multivariable Transfer Function Matrix

To create transfer functions between sensor and torque signals, a similarity transformation had to be developed. This was a major accomplishment. In essence the algorithm compares plant state variables with sensor signal variables, looks for those most similar, and performs the variable substitution. The same is done with control state variables and torque signals. The net result is a set of linearized system equations expressed in terms of sensor signals X_{ps} , other plant state variables X_{po} , torque signals X_{ct} , and other control state variables X_{co} . The matrix form of this equation is

$$\begin{Bmatrix} \dot{X}_{po} \\ \dot{X}_{ps} \\ \dot{X}_{co} \\ \dot{X}_{ct} \end{Bmatrix} = \begin{bmatrix} * & * & 0 & * \\ * & * & 0 & * \\ 0 & * & * & * \\ 0 & * & * & * \end{bmatrix} \begin{Bmatrix} X_{po} \\ X_{ps} \\ X_{co} \\ X_{ct} \end{Bmatrix} + \begin{bmatrix} * \\ * \\ 0 \\ 0 \end{bmatrix} \{R_t\} + \begin{bmatrix} 0 \\ 0 \\ * \\ * \end{bmatrix} \{R_s\}$$

where * implies nonzero matrix partition and R_t and R_s are reference torque and sensor signals. The associated block diagram of this matrix equation is



By selectively setting feedback coefficients equal to zero, the user can open as many feedback loops as desired to create fully open loop or quasi-open-loop transfer functions. The system state equations for any type of transfer function can be set up, between any reference sensor signal input R_s to the control system, and any reference torque signal input R_t to the plant. DISCOS allows users to request poles, zeros, and associated gain for single input-single output transfer functions. By looping within

DISCOS, a full multivariable matrix of transfer functions can be created. Computation of poles is done by straightforward eigenanalysis. Computation of zeroes via methods in common use in mid-1970 consistently failed in practical application; this required development of a new algorithm. A full frequency domain analysis capability, Bode, Nichols, Nyquist, and Root Locus for single input-single output transfer functions is provided. A plotting capability existed, but associated display hardware was discarded many years ago. The preferred mode of operation today is to generate transfer function polynomial ratios within DISCOS and then process them in a follow-on controls analysis program. For example, DISCOS can be used to obtain linearized equations and associated open-loop transfer functions between all plant torque signal inputs and plant sensor signal outputs. These are then passed to the program INCA for interactive control system design and linear time and frequency domain analysis with modern interactive graphics capabilities, see Reference 9.

Other DISCOS Capabilities

In addition to the previously defined capabilities, a few other special needs had to be addressed during the DISCOS development years. Computation relative to an accelerating frame of reference was required to study attitude dynamics of rockets during burn over short periods of time. Mass expulsion effects are not within DISCOS's capability. Momentum wheels had to be defined as either variable or constant rate. Wheels being a prime control mechanism for spacecraft, this capability was essential. A means for circumventing the need to supply user routines was developed. In hindsight, the rules for applying this capability are more difficult to understand than the rules for simply adding a few lines of FORTRAN code into the default versions of the appropriate subroutines. Gravity gradient was needed for large spacecraft. The need for an earth-based gravity field never existed, and, hence, it must be user-supplied. It's not a difficult problem, and it provides a good learning exercise for new users. Alternative methods for defining body flexibility exist. Lumped and consistent mass methods are acceptable. Alternative numerical integration methods are not easily substituted. The ability for dof's to switch between free and fixed relies upon a special integration method; alternative methods would destroy this capability. Furthermore, the constraint stabilization capability performs operations that violate user rules associated with conventional numerical integration subroutines. DISCOS contains no plot capability. Users are, however, given access to the output data subroutine, and they can format output data in a format compatible with in-house plotting capabilities. References 10 and 11 and comment cards within the DISCOS source code still provide primary reference material, while reference 12 provides a brief outline of theory, with equations, and some application discussion. Through

the years, DISCOS has supported application-oriented groups who have minimal incentive to publish beyond the project reporting level. As a consequence there is very little published in the open literature on DISCOS application experience.

Order N DISCOS

The development of Order N DISCOS is a new initiative. The intention is to cut the equation development heart from standard DISCOS and to replace it with an Order N heart. The method will not be burdened by a major matrix inversion within every integration cycle. Mass matrix inversion will be done body by body as is currently done within DISCOS. The degree of the matrix inversion problem associated with Lagrange multiplier determination is reduced to what is needed to implement the topological loop capability. As with NBOD, projection to the vector space in which motion is allowed eliminates the need to determine most of the Lagrange multipliers and associated constraint loads. In addition, several new capabilities are planned. Our first goal is to include a topological loop capability for rigid and/or flexible bodies. Other new capabilities required to support DISCOS's diverse user group are now in the planning stage. At a minimum inverse dynamics, special hinge and special load modeling needs associated with robotics, large flexible spacecraft, biomechanics and biochemistry are being planned. The development of Order N DISCOS is to be carried out as a cooperative research project between Harold P. Frisch at the NASA/GSFC, Jim Turner and Hon Chun at the Cambridge Research Corporation, and Sung-Soo Kim and Ed Haug at The Industry/University Cooperative Research Center at the University of Iowa.

Examples

The program NBOD does not have the modeling capability to solve either of the two test examples defined in this handbook. It does not have the topological loop capability required for the mechanism problem, and it does not have the translational joint capability required for the robot problem.

The program DISCOS does not have the topological loop of rigid bodies capability to solve the mechanism problem as defined. However, if the problem required the inclusion of link flexibility and of ground support compliance, then DISCOS could be used. Computational efficiency would totally depend upon the frequency content of the multiflexible body model. If links or supports are very stiff, use of another multibody code that permits rigid body loops and fixed supports should be considered. Further-

more, DISCOS does not have a planar motion restriction capability. Associated forces and torques of constraint must be computed at every integration step to enforce planar motion. This is a waste of computation time for planar problems; for linear or planar problems, use of another multibody code should be considered.

The robot problem is well within the capabilities of DISCOS. Associated with the DISCOS program is a tutorial input data preparation program. As a point of reference, the required input data file was casually created in less than 1 hour. This data file contains job control information and the complete definition of the physical system.

DISCOS users are required to insert problem unique FORTRAN code within certain subroutines to define all nongyroscopic loads acting on the system. The robot problem required code insertions into two subroutines.

In subroutine KHINGE all forces and torques acting at hinges are defined. DISCOS requires that the HNGT array be user-defined. Each column of length 6 in HNGT contains a torque/force vector defined relative to the associated joint degrees of freedom. The following lines of code were required to define the drive functions associated with the defined robot problem:

```

IF(T.GE.0.0 .AND. T.LT.0.5) THEN
  F1Z = 6348.0
  F2Y = 36.0*T + 986
  L1Z = 673.0*T - 508
  L3X = 63.5
ELSE IF(T.GE.0.5 .AND. T.LT.1.5) THEN
  F1Z = 4905.0
  F2Y = -2.0
  L1Z = 148.0*DEXP(-5.5*(T-0.5)) + 8.0
  L3X = 49.05
ELSE IF(T.GE.1.5 .AND. T.LE.2.0) THEN
  F1Z = 3462.0
  F2Y = -1019.0
  L1Z = 240.0
  L3X = 34.6
ELSE
  STOP
ENDIF
HNGT(3,1)=L1Z

```

HNGT(6,1)=F1Z
 HNGT(5,2)=F2Y
 HNGT(3,3)=L3X

This code implies that arm torque L2Y equals zero. If arm rotation is allowed, then significant rotational motion results due to gyrodynamic effects. If arm rotation is to be kinematically constrained, then it must be so defined in the input data file. If arm rotation is restrained by nonlinear viscoelastic effects, users must provide code to define the resultant viscoelastic torque L2Y. If arm rotation is restrained by an active control system with dynamic components, associated differential equations can be entered into the code, solved, and resultant arm torque L2Y computed. L2Y must then be equated to the (2,2) element of the HNGT array to effect DISCOS communication.

At NASA/GSFC there has never been a pressing need to include the effects of a gravity. If there were, a few lines of DISCOS software code would have been added to provide the capability via input command. A gravity gradient capability command exists; however, it is not needed for earth-based system analysis. Relative to DISCOS gravity is an external force acting at the center of gravity of each body. DISCOS requires that users apply all external forces and torques to sensor points. Each sensor point is located and an associated reference frame defined and oriented relative to the body fixed reference frame in the input data file. External force effects are defined to DISCOS in subroutine EXTOR. Two arrays must be defined. Each column of the array TEX contains a 6 element torque/force vector defined relative to the reference frame defined at the sensor point to which it will be applied. The array ISPON correlates columns of TEX with the sensor point numbering system defined by the user. For the robot problem, the following lines of code were required to include the effect of gravity:

```

NTEX = 3
ISPON(1) = 1
ISPON(2) = 2
ISPON(3) = 3
DO 230 N=1,NTEX
C DEFINE GRAVITY FORCE FOR BODY "N"
  XMM = 9.8*AM(21,N)
C USE TRANSFORMATION MATRIX R0L TO CREATE GRAVITY FORCE
C VECTOR IN SENSOR COORDINATES,
C GRAVITY ALONG INERTIAL "3" AXIS
  
```

```

DO 220 I=1,3
  TEX(3+I,N) = -R0L(3,I,N)*XMN
220 CONTINUE
230 CONTINUE

```

The robot problem, as defined, allows unrestricted rotational motion of the hand and, hence, the physically unrealistic possibility of it passing through the arm in response to gyrodynamic effects. DISCOS users may restrict motion by nonlinear rotational springs that would be added to the hand torque term L3X in subroutine KHINGE. Users are also provided with the option to define functional relations that define physically realizable stops. When hand motion reaches a stop, the associated dof becomes fixed, i.e., kinematically constrained. System motion proceeds until the associated dof constraint load changes sign or exceeds a stiction value. When this user-specified functional relationship is satisfied, the kinematic constraint is released and motion proceeds until the next stop condition is encountered. Users are provided with a clean interface to this capability so that associated coding requirements are minimal.

Summary

The general purpose multibody dynamics codes NBOD and DISCOS have been serving a growing user group for over a decade. They are for the project support analyst who needs maximum capability and flexibility in creating unique problem specific simulation models. Users are expected to have a surface understanding of underlying theory and the associated software code. They are expected to enter FORTRAN code into appropriate subroutines to define nongyroscopic loading effects and to output data in a format compatible with reporting and plotting desires. This user demand has proven to be most desirable. It has allowed the programs to reach far beyond their original beginnings into research communities that did not even exist when the codes were written. On-going efforts to produce a new Order N DISCOS will retain the current user interface philosophy; it will also contain capabilities required to better serve its expanding user group.

References:

1. Velman, J. R., "Simulation Results for a Dual-Spin Spacecraft," *Proceedings of the Symposium on Attitude Stabilization and Control of Dual-Spin Spacecraft*, 1-2 August 1967, Aerospace Corporation, El Segundo, California, November 1967, Aerospace Report No. TR-0158(3307-01)-16, Air Force Report No. SAMSO-TR-68-191.

2. Hooker, W.W., "A Set of r Dynamical Attitude Equations for an Arbitrary n-Body Satellite Having r Rotational Degrees of Freedom," *AIAA J.*, Vol. 8, No. 7, July 1970, pp.1205-1207.
3. Frisch, H. P., *A Vector-Dyadic Development of the Equations of Motion for N-Coupled Rigid Bodies and Point Masses*, NASA TN D-7767, October 1974.
4. Frisch, H. P., *A Vector-Dyadic Development of the Equations of Motion for N-Coupled Flexible Bodies and Point Masses*, NASA TN D-8047, August 1975.
5. Frisch, H.P., *The N-BOD2 User's and Programmer's Manual*, NASA Technical Paper 1145, February 1978.
6. Bodley, C. S., and A. C. Park, "The Influence of Structural Flexibility on the Dynamic Response of Spinning Spacecraft," AIAA Paper 72-348, San Antonio, Texas, 1972.
7. Benfield, W. A., C. S. Bodley, and G. Morosow, "Modal Synthesis Methods," *Symposium on Substructure Testing and Synthesis*, NASA Marshall Space Flight Center, 1972.
8. Kane, T.R., R. R. Ryan, and A. K. Banerjee, "Dynamics of a Cantilever Beam Attached to a Moving Base," *Journal of Guidance, Control, and Dynamics*, Vol. 10, No. 2, March-April 1987.
9. Bauer, F. H., and J. P. Downing, "Control System Design and Analysis Using the Interactive Controls Analysis (INCA) Program," AIAA Paper No. 87-2517, Presented at 1987 AIAA Guidance and Control Conference, Monterey, California.
10. Bodley, C. S., A. D. Devers, A. C. Park, and H. P. Frisch, *A Digital Computer Program for the Dynamic Interaction Simulation of Controls and Structure (DISCOS)*, Volumes I,II, NASA Technical Paper 1219, May 1978.
11. Frisch, H. P., *A Digital Computer Program for the Dynamic Interaction Simulation of Controls and Structure (DISCOS)*, Volumes IV, Supplementary Documentation, NASA Technical Memorandum 80546, August 1979.
12. Bodley, C.S., A. C. Park, A. D. Devers, and H. P. Frisch, "Dynamic Response and Stability Analysis of Flexible, Multibody Systems," contained in *Dynamics and Control of Large Flexible Spacecraft*, L. Meirovitch, ed., *Proceedings of a Symposium held in Blacksburg, Virginia*, June 13-15, 1977.

DADS - Dynamic Analysis and Design System

R.C. Smith and E.J. Haug
Computer Aided Design Software, Inc.
Oakdale, Iowa

1.1 Purpose and Use of DADS

The Dynamic Analysis and Design System software is a set of general purpose computer programs that can be used to model and predict the motion of a variety of real world mechanical systems. Using a set of data that describes the machine to be modeled, DADS builds a mathematical model of the real system that calculates positions, velocities, and accelerations of the various parts of the machine, as well as resultant forces that act in the system. By using such a computer program to analyze a machine, the designer can simulate the behavior of a wide range of alternate designs prior to building and testing prototypes. The DADS program is often used for analysis of existing mechanical systems also.

1.2 Types of Machines That Can Be Modeled

DADS contains a large library of mechanical elements that can be used to build a model. These include rigid and flexible bodies, joints and other constraints, force- and torque-producing elements, as well as control and hydraulic elements. Models can be created in two or three dimensions. Any machine whose motion is entirely planar can be modeled more easily in 2D and can be analyzed much more quickly by the analysis program, because of the smaller number of variables required. The 3D version is used to model any machine whose motion is not contained within a plane, allowing complete generality. If desired, planar systems can also be modeled in 3D.

The control elements can be used in any rigid or flexible body model that is analyzed using the dynamic option, or can be used independently to

create a control system model without any bodies in it. The control elements can be used to apply forces or torques to the rigid or flexible body system. The control elements define a system of first order differential equations that are solved along with the set of second order differential equations associated with the rigid or flexible body equations of motion.

The ability to model the flexibility effect allows much more detailed mechanical models to be built. Data for flexible bodies are generated by using the DADS Intermediate Processor to process data from a finite element program. Any elements in the finite element model can be used, not just simple beam elements between nodes. DADS uses modal coordinates, which are synthesized from the finite element data, to represent the flexibility effects of all flexible bodies in the model. This approach allows the most generality for modeling flexible bodies.

1.3 Degrees of Freedom in a Model

In a mathematical model, each body in a machine is described by its position and orientation. In a two dimensional model, three coordinates are used: the body's x- and y-coordinates in the plane of motion, and its angular orientation coordinate, phi. Thus, a body in a plane has three degrees of freedom of motion, two translational and one rotational. In 3D space, a body's position is described by its spatial x-, y-, and z-coordinates, and its orientation is described by four angular coordinates. Three different systems for initially defining a body's orientation are recognized by DADS: Bryant angles, Euler angles, and Euler parameters. The use of each is described in detail under the 3D Body Element description. Thus, each 3D body has seven generalized coordinates, three translational and four rotational, but since there is one constraint among the four Euler parameters there are only six total degrees of freedom.

As joints or other constraints are added to the model to connect the various bodies, some degrees of freedom of the connected bodies are removed, limiting the scope of their relative motion as prescribed by the particular joint or constraint. For example, fixing a body to ground

removes all its degrees of freedom; i.e., all three in 2D and all six in 3D. As another example, connecting two bodies in 2D by a revolute joint constrains them to remain in contact at that particular point, but allows them to rotate about it. Thus, the two translational degrees of freedom for one of the bodies are removed.

When a model of a mechanism has been completely defined, the total number of generalized coordinates of the bodies minus the number of independent constraint equations for joints and constraints yields the number of degrees of freedom present in the machine. Depending on the type of analysis to be performed, these degrees of freedom can remain in the model or they may be removed by specifying the motion of a particular body or bodies as a function of time (i.e., in kinematic or inverse dynamic analysis).

$$\text{NDOF} = \text{NGC} - \text{NC}$$

where

NDOF = number of degrees of freedom

NGC = number of generalized coordinates

NC = number of constraints

Table of Constraints and Generalized Coordinates

2D Element Type	DOF removed	DOF added
body	0	3
revolute	2	
translational	2	
point	2	
position, X, Y, Phi	1	
revolute-revolute	1	
revolute-translational	1	
distance	1	
bracket	3	
difference, X, Y Phi	1	
gear	1	
point follower	2	1
ground	3	
driver, X, Y, Phi	1	
distance, difference	1	
cam flat follower	2	1
cam convex follower	2	1
belt	1	
rack and pinion	2	
rack and convex pinion	2	

3D Element Type	DOF removed	DOF added
body	1	7
revolute	5	
translational	5	
cylindrical	4	
spherical	3	
point	3	
position X, Y, Z,	1	
orientation	3	
revolute-revolute, intersecting or perpendicular	4	
revolute-translational	4	
revolute-cylindrical	3	
revolute-spherical	2	
spherical-spherical	1	
distance	1	
bracket	6	
difference, X, Y, Z	1	
ground	6	
universal	4	
screw	6	1
driver, X, Y, Z	1	
distance,	1	
difference	1	
relative angle	2	1
planar	3	
lock out rotation	4	
gear	3	2
angle (nonholonomic)	1	

1.4 The Model Analysis Process

Once a model has been defined from the library of either 2D or 3D elements, the data set is processed by the DADS analysis program and the model is mathematically assembled. The equations of motion for the various bodies in the model are automatically generated and numerically solved. The process of creating a mathematical model of a real world system often introduces mathematical redundancies, which are not intuitively evident in the real system. These "redundant constraints" would normally halt any computer analysis of such a model. However, DADS can recognize such redundancies and remove them automatically before analysis, allowing the user to describe the system as precisely as possible and still be able to simulate it numerically.

Results of the simulation are the positions, velocities, and accelerations of all bodies in the machine. Also included are various data on any force elements in the model and, for some types of analysis, the internal reaction forces due to any joints or constraints in the model. The user can specify the print interval of output as well as the total length of time for the simulation. Having a good mathematical representation of the physical system, it is a simple process to change various specifications of either the model or the method of analysis. Thus, DADS can be used as an iterative design tool for creating new products, as well as an analysis tool for examining existing products.

1.4.1 Reference Frames

DADS gives the user a choice of several reference frames in which to work. During the process of model creation, the user can enter all joint data relative to either a global reference frame, a local body-fixed reference frame at the center of gravity of each body, or at a non-centroidal body-fixed reference frame.

If the user requests reaction force data, the forces and torques may be reported in one of three frames. The first frame is the global reference frame. The second is the body-fixed reference frame on each body. The third is a joint or constraint reference frame defined relative to each body at the point where the joint or constraint acts. In a few of

the joints, this third reference frame is actually a moving frame, due to the nature of the joint. In such cases, this is mentioned explicitly in the description of the joint.

1.5 Methods of Analysis

A machine can be analyzed by the DADS analysis program using a number of different methods. These are described below and a more detailed explanation can be found in the DADS Theoretical Manual.

1.5.1 Kinematic Analysis

Kinematic analysis is used to calculate the motion of the various bodies in the mechanism, disregarding both their mass properties and any forces in the system. Mechanisms analyzed kinematically must have all degrees of freedom eliminated from the mathematical model. This is accomplished by specifying a number of "drivers". After the user has translated the physical mechanism into a preliminary model data set, this mathematical model may have one or more degrees of freedom. For each degree of freedom that remains, a driver must be added to the data that defines the motion of a particular coordinate or relationship between coordinates, as a function of time. Each driver removes a degree of freedom from the model. The results of kinematic analysis are the positions, velocities, and accelerations of all the bodies in the model, for each time step in the analysis.

1.5.2 Dynamic Analysis

In dynamic analysis, the model may have any number of degrees of freedom, from one up to the total possible degrees of freedom in the machine. The motion of the bodies is calculated from the forces acting upon them and the mass properties of the bodies. These forces include gravity and any external forces specified by the user. The equations of motion are defined in terms of the masses and forces. The resulting second order differential equations are then integrated numerically using

a variable step and order algorithm. One important requirement of dynamic analysis is that the initial conditions of the bodies, i.e., initial position and velocity, must be specified at the start of analysis. Rather than specifying initial conditions for every coordinate in the model, it is necessary to specify initial conditions for only the independent coordinates, the number of which is equal to the number of degrees of freedom in the model. Since a cursory examination of the model may not readily identify the independent coordinates, it is often necessary to make an initial dynamic analysis run without initial conditions in order to identify these independent coordinates. The results of such an analysis run identify the set of independent coordinates, as selected by DADS. The appropriate initial conditions can then be added to the model and the complete dynamic analysis run performed. The results of dynamic analysis are the positions, velocities, and accelerations of all the bodies in the model for each time step. Internal reaction forces in the joints and constraints in the model can also be reported, if requested by the user.

1.5.3 Static Analysis

Static analysis uses the mass properties of the bodies and any forces acting upon them to calculate a model configuration that minimizes the potential energy of the system. The resultant position of each body is reported, along with the potential energy of the model at the beginning and end of the analysis run. Static analysis is valid only for a conservative system. If there are any nonconservative forces, such as rotational or translational damping elements or any other nonconservative external forces, then static analysis will not find a configuration with minimized potential energy. To simulate static analysis for a model that contains nonconservative forces, the user can run the model under dynamic analysis instead, making sure to include one damping element for each degree of freedom in the system. When velocities of all bodies approach zero, the model will be in an equilibrium configuration.

1.5.4 Inverse Dynamic Analysis

Inverse dynamic analysis uses features of both kinematic and dynamic analysis. As in kinematic analysis, drivers must be added to the model to

eliminate all degrees of freedom. However, mass properties of all the bodies are also required, as in dynamic analysis. From the motion of the bodies specified by the drivers, forces required to produce the motion are calculated. The results are body positions, velocities, and accelerations and the reaction forces in the various joints and constraints in the model. These reaction forces can be interpreted as the forces necessary to generate the prescribed motion.

1.5.5 Assembly Analysis

Assembly analysis is used to assemble all bodies in the model into a configuration that satisfies all the joint and constraint connections between the bodies. The assembly process allows for the possibility of slight errors in body positions specified by the model data set. These initial data need not be exact, since the assembly algorithm minimizes the error in all the joints and constraints, to determine a set of body positions that best satisfy these constraints. Assembly analysis is performed prior to all other types of analysis, or it can be performed by itself. If any degrees of freedom are present in the model, the solution will represent one of a number of possible solutions. As in dynamic analysis, initial conditions must be added to the model in order to specify the state of the system at the beginning. Results of assembly are the positions of all the bodies in the model.

1.6 Analysis Output

Three standard output files are generated by the DADS analysis program, for all types of analysis. These are an information file, an ASCII (readable character data) output file, and a binary output file. When a DADS analysis is initiated, the program will request the name of the input file, as well as names of the three output files. The input file name can be entered in full, or the user can choose to enter the file name without the four-character file extension and have DADS supply the default file extension. These file extensions are supplied by the DADS

Preprocessor whenever it is used to create 2D or 3D input files for the DADS analysis program. After entering the input file name, the user may choose to enter his or her own names for the three output files, or, by merely entering one or more carriage returns, may request DADS to create a default file name for any or all of the three output files. These default names are created by appending a file extension character string to the input file name. The file extensions used are as follows:

<u>File type</u>	<u>File Extension</u>
2D input file	.FM2
3D input file	.FM3
ASCII output file	.OUT
Binary output file	.BIN
Information file	.INF

1.6.1 ASCII Output File

The ASCII output file contains the results of analysis for each body in the model. Body positions, velocities, and accelerations are printed for each time step performed in the analysis. Depending on the type of analysis, joint and constraint reaction forces are also printed at each time step, along with information on the state of any force element in the model. If control elements are present, the values of the state variables and their time derivatives are printed out for each time step. This file is written as standard ASCII character data and can be examined using a text editor or can be printed out on a system printer.

1.6.2 Binary Output File

The binary output file contains exactly the same information as the ASCII output file, but is formatted differently for compactness and ease of access by other computer programs. The binary file also contains general information on the results of the analysis, as well as a copy of the data input file from which the results were obtained. The binary file

is used by the DADS Postprocessor for displaying analysis results in tabular or graphical form.

1.6.3 Information File

The information file contains any error, warning, or informational messages generated by the analysis run, which are not actual simulation results. The results of the assembly process are printed here. These results include a report on the success of the assembly process and a listing of the calculated positions of all the bodies in the model. In some types of analysis, the set of independent coordinates is also reported here. Like the ASCII output file, the information file can be examined using a text editor or can be printed on a system printer.

1.7 DADS Pre and Post Processing Programs

The DADS software package includes Preprocessor and Postprocessor programs as well as an animation program and a program for translating finite element data. The total software package contains the following programs:

Preprocessor
DADS2D Analysis
DADS3D Analysis
Postprocessor
Geomake
Intermediate Processor
Inertia Relief Load Vector Calculator

1.7.1 DADS Preprocessor

The DADS Preprocessor is a command-driven system for creating and editing models for analysis by DADS. It can be used to define the elements in a mechanical model, save them in a file, and alter the model at a later time. The command-driven nature of the Preprocessor allows it

to be easy to learn, while not slowing down the experienced user. The elements in a model are defined and manipulated using several different sets of available commands. Numerous menus and thousands of lines of help text are always available for reference, so that the novice user can have many of his or her questions answered, while continuing to run the program.

The Preprocessor is organized in a tree structure, with functions separated into different levels. Throughout the tree structure of the Preprocessor, moving from one level to another is accomplished in exactly the same manner. To move down to a sublevel, the user merely types the name of the lower level. To move back up, the command "UP" will always suffice.

Each level of the Preprocessor serves to delineate and implement a specific function or functions. Thus, each level has its own set of available commands. These are identified in the menu for that level and are described in the various help texts for that level. Some commands are "global" in nature, being available at all levels, others are available at several different levels, while others are specific to a single level.

1.7.2 DADS Postprocessor

The DADS Postprocessor provides a means of examining the results of a DADS analysis run. Any variable in the model can be graphically plotted or displayed in tabular form. These variables include all body coordinate positions, velocities, and accelerations at each time step, data on any force elements in the model, total and potential energy of the model at each time step, and, if requested, all internal reaction forces in the joints and constraints in the model. Data from two or more different analysis runs can be plotted on the same graph for comparison. Use of the DADS Postprocessor is described further in Chapter 3. The Postprocessor currently supports Tektronix 40xx and 41xx series terminals, as well as other graphics terminals that emulate the Tektronix standards, and several engineering workstations.

1.7.3 DADS Geomake program

This program provides a way to create geometry and associate it with the bodies in the DADS model for animation purposes. The resulting geometry can be animated on a variety of graphics devices but the best results are generally on an engineering workstation. The geometry is built up from simple primitive shapes which are rotated and translated into the proper positions. The primitives are grouped together according to which body they are attached to. Results from the DADS analysis run are extracted from the binary file using the Postprocessor and are used by Geomake to define the location of each body at each output time step.

1.7.4 DADS Intermediate Processor

The Intermediate Processor is used to extract "modal" data from the results of a static or vibration analysis done using one of the supported finite element programs. The mass and stiffness matrices, Eigen vector, and node locations are read from the finite element analysis results. The data is processed and saved in a file that can be loaded in the Preprocessor and merged with the DADS model data.

1.7.5 DADS Inertia Relief Load Vector Calculator

This program is used to calculate inertia relief loads for static analysis results that have rigid body modes present. This program is not needed if vibration analysis or constrained static analysis is used to model the flexible structure.

1.8 Summary of DADS Software and Documentation

The documentation for the DADS package is divided into three manuals: the Users Manual, the Examples Manual, and the Theoretical Manual. There are also several technical reports and published papers that are available from CADSI. The users manual describes the commands and options found in each program. The examples manual contains several thorough examples of mechanical systems that have been modeled using

DADS. Each example contains a problem description, diagrams, model input data, tabular results, graphical plots of results, and sample animation frames. The examples attempt to cover all major features available in DADS. The theoretical manual contains a complete discussion of the algorithms used and modeling techniques.

1.9 DADS Analysis of Spatial Robot

The following example shows the results of a DADS analysis of a spatial robot model. The system consists of four bodies (one of which is fixed to ground), two cylindrical joints, and one revolute joint. During the two seconds of simulation time, forces and torques are applied to the joints to produce a desired trajectory of the robot hand. The following data summarizes the physical characteristics of the bodies in the model.

Body 1 (tower): C.G. = (0.0, 0.0, 2.25)

Mass = 250 kg

$I_{xx} = 90 \text{ kg}\cdot\text{m}^2$

$I_{yy} = 10 \text{ kg}\cdot\text{m}^2$

$I_{zz} = 90 \text{ kg}\cdot\text{m}^2$

Body 2 (arm): C.G. = (0.3767531, 0.6485037, 2.25)

Mass = 150 kg

$I_{xx} = 13 \text{ kg}\cdot\text{m}^2$

$I_{yy} = 0.75 \text{ kg}\cdot\text{m}^2$

$I_{zz} = 13 \text{ kg}\cdot\text{m}^2$

Body 3 (hand): C.G. = (0.6530388, 1.1240731, 2.25)

Mass = 100 kg

$I_{xx} = 4 \text{ kg}\cdot\text{m}^2$

$I_{yy} = 1 \text{ kg}\cdot\text{m}^2$

$I_{zz} = 4.3 \text{ kg}\cdot\text{m}^2$

All three of the primary bodies have an initial rotation about the Z-axis of -0.5263 radians.

The forces are applied through the DADS TSDA element as an actuator force. The torques are applied through the RSDA element as an actuator torque. The forces and torques are defined by the following equations.

<u>Simulation time t (seconds)</u>	<u>Force or torque expression</u>
$0.0 \leq t < 0.5$	$F_{1Z} = 6348$ (Tower force, N) $F_{2Y} = 36t + 986$ (Arm force, N) $L_{1Z} = 673t - 508$ (Tower torque, N-m) $L_{3X} = 63.5$ (Hand torque, N-m)
$0.5 \leq t < 1.5$	$F_{1Z} = 4905$ $F_{2Y} = -2$ $L_{1Z} = 148 e^{(-5.5(t-0.5))} + 8$ $L_{3X} = 49.05$
$1.5 \leq t \leq 2.0$	$F_{1Z} = 3462$ $F_{2Y} = -1019$ $L_{1Z} = 240$ $L_{3X} = 34.6$

The following figures show some results of the analysis of this model. Figures 1 through 4 show plots of the tower force, arm force, tower torque, and hand torque functions, respectively. Figures 5, 6, and 7 show the tower vertical displacement, tower rotation, and arm displacement, respectively. Figure 8 shows two frames from the animation of the model. The two frames are the first and last time steps of the analysis. Note how the tower rises and rotates, and how the arm extends to provide the desired trajectory of the hand.

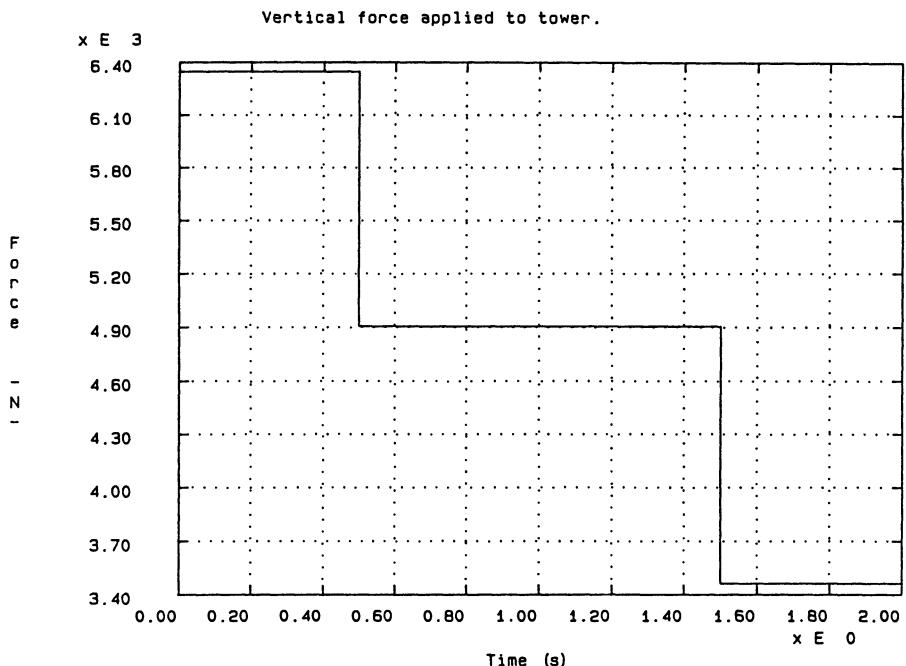


Figure 1.

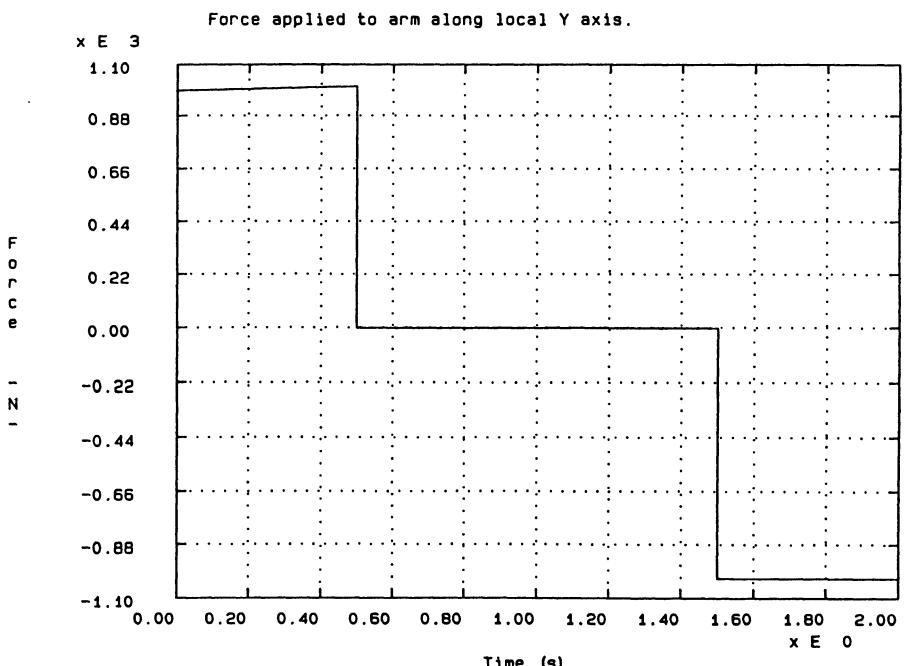


Figure 2.

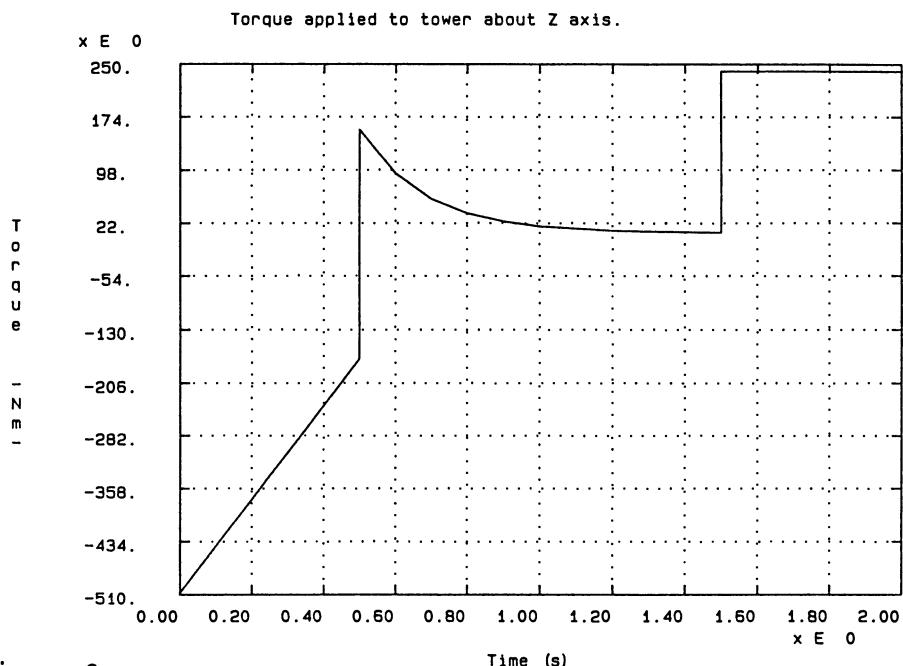


Figure 3.

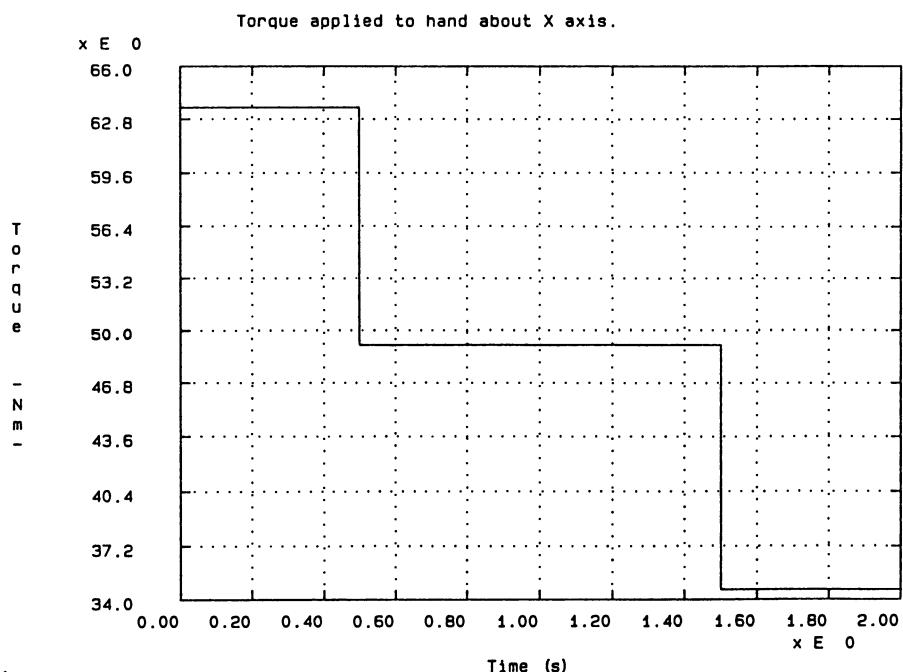


Figure 4.

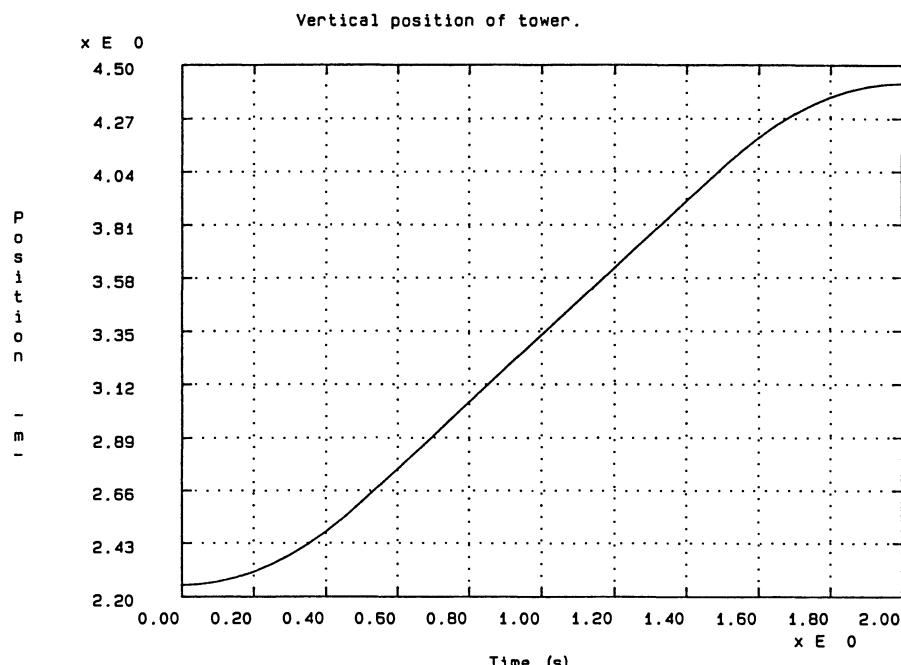


Figure 5.

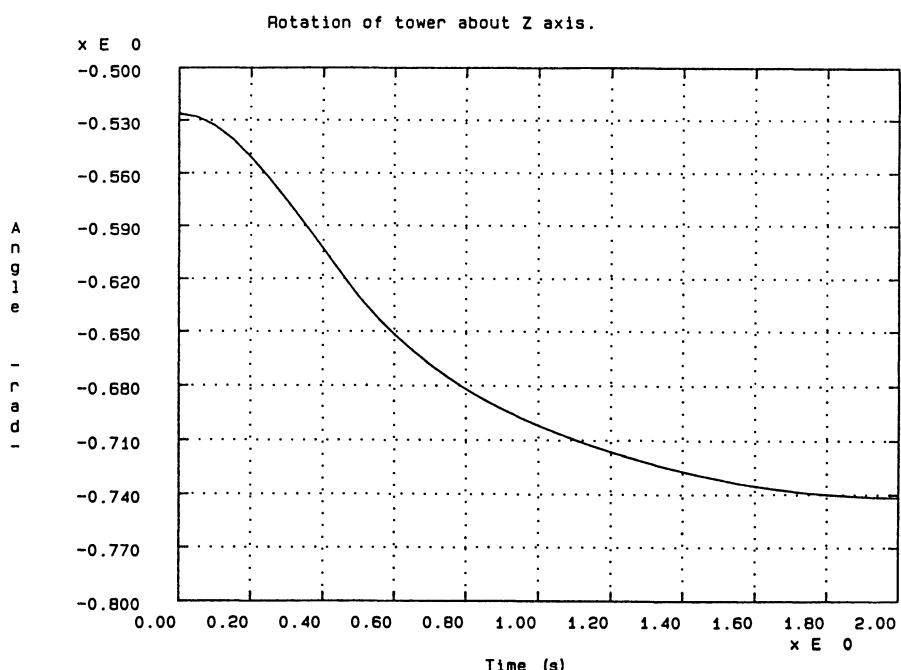


Figure 6.

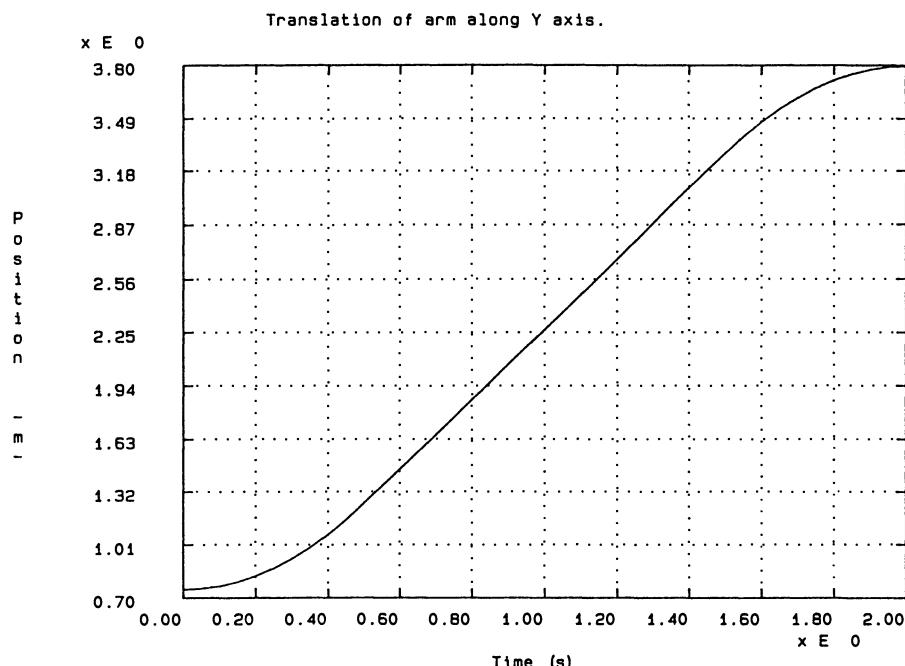


Figure 7.

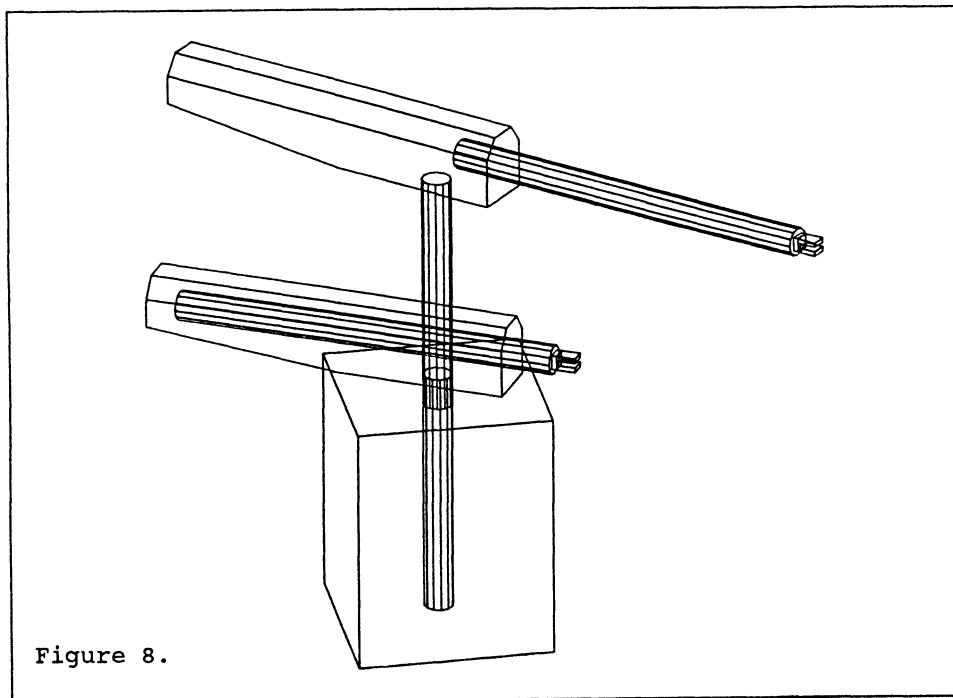


Figure 8.

NEWEUL - Software for the Generation of Symbolical Equations of Motion

E. KREUZER and W. SCHIEHLEN

Department of Ocean Engineering - Structural Dynamics
Technical University Hamburg-Harburg, Hamburg, FRG

and

Institute B of Mechanics
University of Stuttgart, Stuttgart, FRG

Summary

In this contribution the theoretical background of the software NEWEUL is presented, the necessary input information is given and the output generated by the program is discussed. Two examples, a mechanism and a robot, are included for demonstration of the program. NEWEUL is written in FORTRAN 77 resulting in an excellent portability to all kinds of computers from personal computers to main frames.

Introduction

Modeling is the first step for the mathematical analysis of the dynamics of a mechanical system, i.e. the real system must be transformed into a model with idealized elements. The method of multibody systems offers the following elements:

- rigid bodies with translational and rotational inertia,
- masspoints with translational inertia only,
- constraining elements without elasticity and friction (such as supports, bearings, Hooke's joints and rolling wheels),
- coupling elements without inertia acting at nodal points (such as springs, dampers, servomotors and tires).

A multibody system composed of the mentioned elements is shown in Fig. 1. The bodies are numbered from 1 to n.

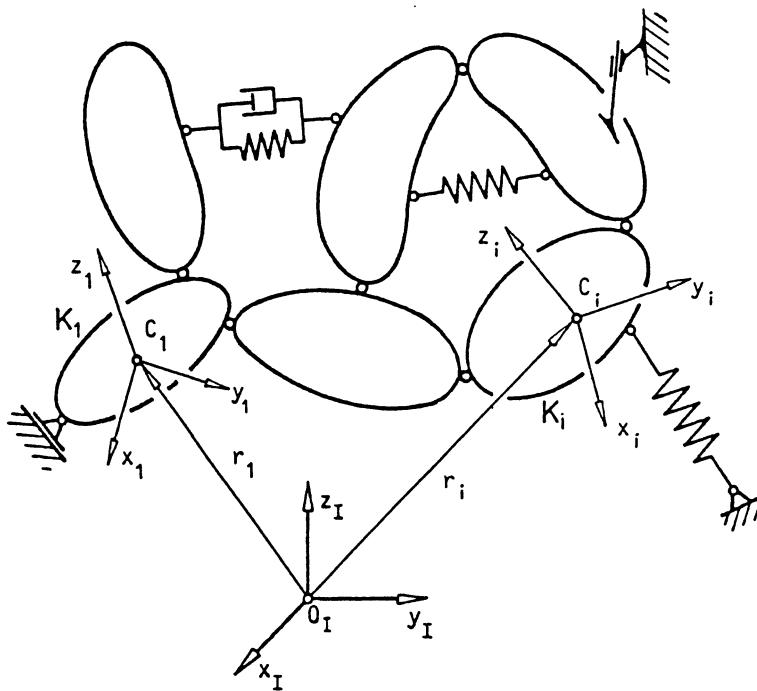


Fig. 1. Multibody system

Multibody Systems with Holonomic Constraints

NEWton's and EULer's equations are the basis of the software package NEWEUL. All vectors and matrices in this section are formulated relative to an inertial frame (unless the contrary is explicitly noted).

Let us consider a multibody system subject to q holonomic rheonomic constraints due to the constraining elements. Then there remain

$$f = 6n - q \quad (1)$$

positional degrees of freedom.

The position of the system is described by the $fx1$ -vector of

generalized coordinates y , i.e. its dimension is equal to the number of degrees of freedom. The Newton-Euler formalism now calls for a couple of variables for each body, Fig. 1.

The position of body K_i is given by the translation vector r_i from the origin of the inertial frame to the center of mass C_i and by the rotation matrix S_i describing the rotation of a body-fixed frame relative to the inertial frame,

$$r_i = r_i(y, t), \quad i=1, \dots, n, \quad (2)$$

$$S_i = S_i(y, t), \quad i=1, \dots, n. \quad (3)$$

The applied forces and torques due to coupling elements acting on the body are transformed to 3×1 -vectors f_i^e and l_i^e with the applied torques l_i^e referring to the center of mass C_i .

As inertia variables the mass m_i and the inertia tensor I_i attached to the center of mass C_i are required for each body.

The variables S_i , l_i^e , I_i are not regarded for masspoints.

Note as a rule that all above mentioned vectors and matrices have to be given with respect to the inertial frame. For convenient input, however, the software package NEWEUL offers also a description of rigid bodies, masspoints and joints/supports relative to other rigid bodies or joints/supports, respectively, already entered. Then, the relative quantities are automatically converted into absolute quantities.

Differentiation leads to the velocity and acceleration vectors of translation for each body K_i ,

$$\dot{v}_i = J_{Ti}(y, t) \dot{y} + \bar{v}_i(y, t), \quad (4)$$

$$\ddot{a}_i = J_{Ti}(y, t) \ddot{y} + \bar{a}_i(y, y, t), \quad (5)$$

and those of rotation,

$$\omega_i = J_{Ri}(y, t) \dot{y} + \bar{\omega}_i(y, t) , \quad (6)$$

$$\alpha_i = J_{Ri}(y, t) \ddot{y} + \bar{\alpha}_i(y, \dot{y}, t) , \quad i=1, \dots, n , \quad (7)$$

where the $3 \times f$ -Jacobians J_{Ti} and J_{Ri} , the local velocities \bar{v}_i and $\bar{\omega}_i$, and the acceleration components \bar{a}_i and $\bar{\alpha}_i$ independent of y are introduced.

Now the momentum principle (Newton's equation) of a free body K_i

$$m_i \bar{a}_i = f_i^e + f_i^r , \quad i=1, \dots, n , \quad (8)$$

and the angular momentum principle (Euler's equation) relative to the center of mass

$$I_i \bar{\alpha}_i + \omega_i \times (I_i \omega_i) = l_i^e + l_i^r , \quad i=1, \dots, n , \quad (9)$$

are written down for each body. Summarizing (8) and (9) for all bodies to one vector differential equation and applying (4) through (7) yields the Newton-Euler equations with respect to the inertial frame

$$\begin{aligned} \bar{M}(y, t) \ddot{y} + \bar{J}(y, t) \dot{y} + \bar{k}(y, \dot{y}, t) = \\ \bar{q}^e(y, \dot{y}, t) + \bar{q}^r(y, \dot{y}, t) . \end{aligned} \quad (10)$$

Here

$$\bar{M} = \text{diag}\{ m_1 E , \dots , m_n E , I_1 , \dots , I_n \} \quad (11)$$

is the $6n \times 6n$ -block diagonal matrix of masses and inertia tensors,

$$\bar{J} = [J_{T1}^T | \dots | J_{Tn}^T | J_{R1}^T | \dots | J_{Rn}^T]^T \quad (12)$$

is the global $6 \times f$ -Jacobian, \bar{k} is the 6×1 -vector of gyroscopic, centrifugal and Coriolis forces,

$$\bar{q}^e = [f_1^{eT} | \dots | f_n^{eT} | l_1^{eT} | \dots | l_n^{eT}]^T \quad (13)$$

is the 6×1 -vector of applied forces and torques and \bar{q}^r is the 6×1 -vector of constraint forces and torques, see e.g. [1] and [2].

Furthermore, the $6 \times f$ -inertia matrix

$$\bar{M} = \bar{M} \bar{J} \quad (14)$$

is introduced for abbreviation in the Newton-Euler equations (10).

Eliminating the constraint forces \bar{q}^r by multiplying (10) from the left by the transposed $f \times 6n$ -Jacobian \bar{J}^T one obtains according to D'Alembert's principle the equations of motion as

$$M(y, t) \ddot{y} + k(y, \dot{y}, t) = q(y, \dot{y}, t), \quad (15)$$

where M is the symmetric $f \times f$ -inertia matrix, k the $f \times 1$ -vector of the generalized gyroscopic forces and q the $f \times 1$ -vector of generalized applied forces.

Remark : Consequently the vector \bar{q}^r of constraint forces need not be regarded in calculating the equations of motion. \square

For completely linear systems, equations (15) are decomposed as

$$M(t) \ddot{y} + P(t) \dot{y} + Q(t) y = h(t), \quad (16)$$

and, if the matrices P and Q are time-invariant, even into

$$\ddot{M(t)} \dot{\mathbf{y}} + (\mathbf{D} + \mathbf{G}) \dot{\mathbf{y}} + (\mathbf{K} + \mathbf{N}) \mathbf{y} = \mathbf{h}(t). \quad (17)$$

Here P denotes the matrix of velocity dependent forces, D the damping matrix, G the matrix of gyroscopic forces, Q the matrix of position dependent forces, K the stiffness matrix and N the matrix of nonconservative forces. The matrices D and K represent the symmetric, the matrices G and N the skew-symmetric parts of the matrices P and Q . The right hand side of equations (16) and (17) is formed by the time-dependent fx1-excitation vector $\mathbf{h}(t)$.

Multibody Systems with Nonholonomic Constraints

Nonholonomic constraints restrict the velocity state of the system additionally by r nonintegrable kinematical constraints due to nonelastic wheels. Hence, a system composed of n rigid bodies with q holonomic and r nonholonomic constraints has

$$g = 6n - q - r \quad (18)$$

motional degrees of freedom.

Thus, the velocity state of nonholonomic systems can be described uniquely by g scalar generalized velocity coordinates summarized in the $gx1$ -vector $\mathbf{z}(t)$.

In holonomic systems $g = f$ holds true and, as a rule, it is the vector $\dot{\mathbf{y}}$ which serves to describe the velocity state. However, it may prove expedient even in holonomic systems to introduce a $fx1$ -velocity vector \mathbf{z} . In any case, the vector of the first derivative of generalized coordinates $\dot{\mathbf{y}}$ has to be given dependent on \mathbf{z} , \mathbf{y} and t :

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}, \mathbf{z}, t) . \quad (19)$$

In mechanical systems, this relation generally is linear in the velocity variables,

$$\dot{\mathbf{y}} = \mathbf{F}(\mathbf{y}, t) \mathbf{z} + \mathbf{H}(\mathbf{y}, t) . \quad (20)$$

Introducing now the Jacobians of generalized velocities

$$\mathbf{L}_{Ti} = \mathbf{J}_{Ti} \mathbf{F} \quad \text{and} \quad \mathbf{L}_{Ri} = \mathbf{J}_{Ri} \mathbf{F} , \quad (21)$$

the velocity and acceleration vectors (4) through (7) can be rewritten as analogous relations.

Using the matrices \mathbf{L}_{Ti} and \mathbf{L}_{Ri} the global Jacobian of generalized velocities of the system is given by

$$\bar{\mathbf{L}} = [\mathbf{L}_{T1}^T | \dots | \mathbf{L}_{Tn}^T | \mathbf{L}_{R1}^T | \dots | \mathbf{L}_{Rn}^T]^T . \quad (22)$$

Equations (8), (9) yield the Newton-Euler equations with generalized velocities :

$$\begin{aligned} \bar{\mathbf{M}}(\mathbf{y}, \mathbf{z}, t) \dot{\bar{\mathbf{L}}}(\mathbf{y}, \mathbf{z}, t) \mathbf{z} + \bar{\mathbf{k}}(\mathbf{y}, \mathbf{z}, t) = \\ \bar{\mathbf{q}}^e(\mathbf{y}, \mathbf{z}, t) + \bar{\mathbf{q}}^r(\mathbf{y}, \mathbf{z}, t) . \end{aligned} \quad (23)$$

Multiplying (23) from the left by the transposed of the global matrix $\bar{\mathbf{L}}$ one obtains according to Jourdain's principle the equations of motion of first order with generalized velocities

$$\mathbf{M}(\mathbf{y}, \mathbf{z}, t) \dot{\mathbf{z}} + \mathbf{k}(\mathbf{y}, \mathbf{z}, t) = \mathbf{q}(\mathbf{y}, \mathbf{z}, t) . \quad (24)$$

In the American literature, equations (24) are often called "Kane's equations".

Remark : Equations (24) are not sufficient for a complete

analysis of the system's motion, the relations (19) or (20) must be added. []

In case of complete linear systems the equations of motion (24) are decomposed once more according to (16) :

$$\dot{M(t)} \cdot z + P(t) \cdot z + Q(t) \cdot y = h(t) . \quad (25)$$

Further splitting as in (17) is not possible in case of nonholonomic systems, though.

Moving Reference Frames and Compression

For many problems it has proved conducive to set up the momentum principle (8) and the angular momentum principle (9) not with respect to an inertial frame, but with respect to a suitable moving reference frame. Different frames can be chosen for the momentum and the angular momentum principle and even for each body of the multibody system. Then, the kinematical velocity and acceleration relations (4) through (7) remain valid, though the vectors and matrices occurring have to be given with respect to the reference frame selected. The vector differential equation (15), too, is composed of vectors and matrices with respect to the reference frame selected. The additional work necessary with this method, namely the conversion of the kinematics, need not to worry the user of the software package NEWEUL, since all differentiations and transformations are done automatically by the program. For more details see reference [3].

The user's only commitment is to ensure that all vectors and tensors (translation, force and torque, inertia) are formulated, by means of so-called filemarks, relative to the reference frame selected.

Remark: Though a cleverly chosen reference frame can simplify input and calculation to a great extent, it does not influence

format or length of the equations of motion. Their parameters merely depend on the generalized coordinates and generalized velocities chosen. □

If multibody systems with many degrees of freedom and complex nonlinear kinematics are completely represented by symbolic equations of motion, storage and computation time at disposal are often unduly exceeded. Furthermore, the extension of the generated symbolical expressions encumbers numerical evaluation and physical interpretation of the equations.

For this reason NEWEUL optionally offers the calculation of equations of motion in compressed form. This means that after symbolic calculation of the kinematical relations the resulting expressions as well as applied forces and torques and inertia variables are abbreviated and calculation is continued with these abbreviations. In general such "compressed" equations of motion are many times less extensive than not compressed ones.

Different stages for compression are possible. Even a recursive procedure can be chosen. It should be noted that expressions are condensed only at definite points within the program and that the introduced abbreviations can be easily physically interpreted.

The following vectors and matrices are optionally abbreviated to simple variable names:

- the 3x3-inertia tensors with respect to the reference frame and the center of mass I_i , $i = 1, \dots, n$,
- the 3xg-Jacobians L_{Ti} of translation and the 3xg-Jacobians L_{Ri} of rotation, $i = 1, \dots, n$,
- the 3x1-vectors of applied forces and torques f_i^e and l_i^e , $i = 1, \dots, n$,
- the 3x1-vectors of angular velocities ω_i , $i = 1, \dots, n$,

- o the 3×1 -vectors of components \bar{a}_i and $\bar{\alpha}_i$ of the accelerations a_i and α_i , $i = 1, \dots, n$.

The non-vanishing elements of these vectors and matrices are given expressive abbreviations and these are used to set up the compressed equations of motion in the end.

Remark: The mentioned vectors depend on the reference frames selected. Therefore by carefully selecting the reference frames the symbolic expressions due for subsequent numerical evaluation and physical interpretation can be considerably reduced. \square

Equations of Constraint Forces

For the calculation of the constraint forces the Newton-Euler equations (10) or (23), respectively, provided by the program NEWEUL, have to be used.

Then, the equations of constraint forces can be established in the form:

$$\bar{Q} g = \bar{q}^r = \bar{M}(y, t) \ddot{y} + \bar{k}(y, \dot{y}, t) - \bar{q}^e(y, \dot{y}, t) \quad (26)$$

see e.g. SCHRAMM [4], where \bar{Q} stands for the $6n \times (q+r)$ -distribution matrix and g for the $(q+r) \times 1$ -vector of generalized constraint forces. These equations are numerically calculated and solved by the subpackage ZKSUB using the geometrical variables supplied by the software package NEWEUL in symbolic form.

Remark: For calculation of constraint forces all joints/supports must be entered correctly. It is not possible to enter only a few joints/supports and others not. \square

The calculation of the generalized constraint forces according to (26) requires the integration of the equations of motion (15), too. Therefore, it is recommended to use parallel computation.

Simulation of Motion

The symbolical equations of motion (15) or (19, 24), respectively, and the equations of reaction (26) have to be solved by numerical methods. In principle, the symbolical equations of motion can be solved by any integration code at hand. The program interfaces, however, have to be prepared by the user.

If the benefits of a free choice of the integration code are not desired, the integrated software system NEWSIM recently revised by Leister [5] can be used. Then, all the interface problems are automatically handled. The software NEWSIM applies the integration code of Shampine and Gordon [6] most reliable for the dynamical analysis of multibody systems.

The simulation data may then be displayed by computer graphics resulting in single or moving pictures. Future developments will include not only data postprocessing but also data preprocessing by combining CAD-software with multibody software like NEWEUL, see Ref. [7].

Fields of Application of the Software Package NEWEUL

The software package NEWEUL is designed to set up

- linear or
- nonlinear

equations of motion for multibody systems with

- holonomic and/or
- nonholonomic

constraints. Furthermore, symbolical expressions may be generated which enable the subpackage ZKSUB to numerically calculate the constraint forces occurring in joints/supports. The constraints may be scleronomic or rheonomic. The multibody system may have an open kinematic loop structure, a tree structure or a closed kinematic loop structure. General force laws may be included, too. Even if input and calculation are done thoroughly symbolically, constants may be automatically replaced by numerical values.

Systems with the above features are found e.g. in the following fields:

- vehicle dynamics,
- machine dynamics,
- robot dynamics,
- dynamics of mechanisms,
- satellite dynamics,
- biomechanics.

For further numerical analysis of the symbolical equations, e.g. calculation of eigenvalues or simulation, the results are always written in a format compatible directly to any FORTRAN program.

Program Structure

The software package NEWEUL consists of many subroutines which are included in the main program NEWEUL. This program works as a BATCH-program, i.e. it reads input data from a file called TAPE08 and writes results on three different files, OUTPUT, TAPE37 and TAPE38, Fig. 2.

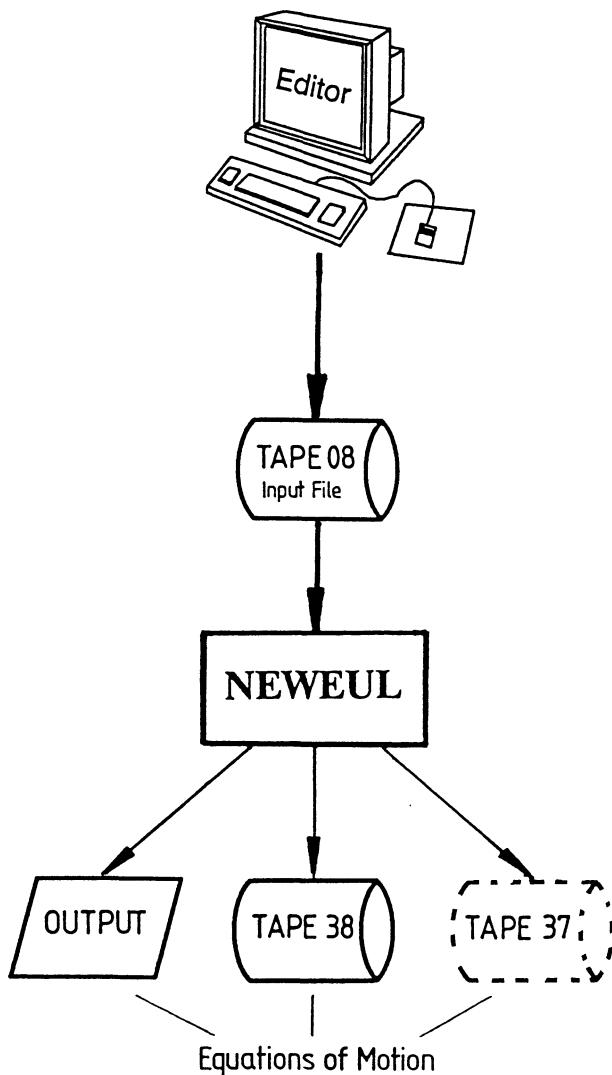


Fig. 2. Flow chart of program NEWEUL

The file OUTPUT is displayed on the screen and contains all results produced in an easy-to-read format, and for checking purposes also the complete set of input data. TAPE38 is similar to OUTPUT but in FORTRAN-compatible form. File TAPE37 contains the results in FORTRAN-compatible form only.

The input file is prepared within an editor by means of file-masks. On the basis of these tools it is easy to summarize and formulate the complete set of input data needed.

Logically the input file can be subdivided into six different sets which occur none, once or several times. These sets have to be ordered in a definite manner as symbolically shown in Fig. 3.

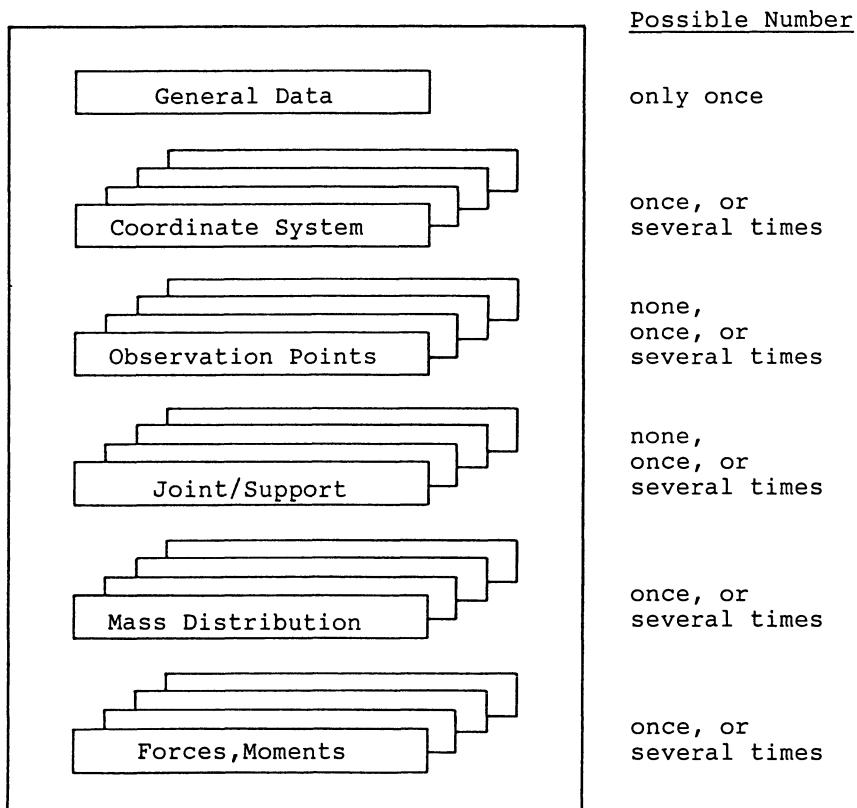


Fig. 3. Structure of input file

Special Features of the Software Package

In the software package NEWEUL the following elements are available for modeling:

- rigid bodies,
- masspoints,
- joints/supports,
- nodal points,
- observation points.

Note that joints/supports and masspoints exclude each other and cannot be used together. Nodal points are always situated on a rigid body.

Reference Frames

The position of each element can be described with respect to an arbitrary reference frame. Then, the position of a rigid body K_i , $i=1,\dots,n$, is represented by the vector r_i from the origin O_R of the reference frame to its center of mass c_i and the rotation matrix S_i describing the rotation of the body-fixed frame with respect to the reference frame and index R indicating the current number of reference frame.

The reference frame is allowed to be in arbitrary motion with respect to the inertial frame given by the translation vector r_R from the origin of the inertial frame to the origin of the reference frame and by the rotation matrix S_R describing the rotation of the reference frame relative to the inertial frame.

Relative Input

The position of rigid bodies, masspoints and joints/supports can be described either with respect to a certain frame (whether inertial or reference) or relative to another element.

The reference element must be a rigid body or a joint/support. Its translation vector and rotation matrix have been stored by the program, i.e. the element must have been designated explicitly as reference element. This is the reason why the elements should be properly numbered (especially rigid bodies and joints/supports). Relative input is always related to the origin of the frame of the reference element.

Nodal Points

Nodal points serve to describe the action of forces on bodies. A nodal point has to be assigned to a definite body. The situation depicted in Fig. 4 illustrates the application of nodal points. Force F acts on body K_i in nodal point A described by vector r from the body's center of mass C_i .

Using the nodal point approach applied torques, resulting from applied forces not acting on the center of mass of the body considered, are automatically included. Moreover, applied torques may be entered via nodal points as well.

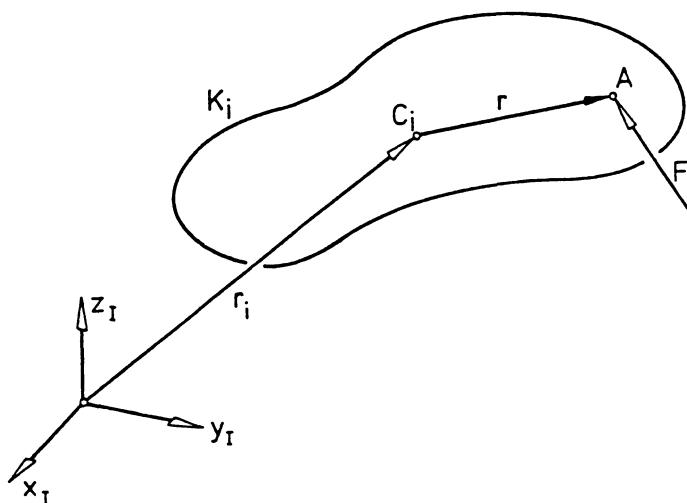


Fig. 4. Example for application of nodal points

Observation Points

Observation points allow to determine translational and rotational position, velocity and/or acceleration of arbitrary points of the multibody system with respect to any other point in an arbitrary reference frame. Observation points are very useful for the evaluation of applied forces, e.g. generated by tires.

NEWEUL Syntax Rules

The syntax for input is widely determined by the basic formula manipulation algorithm of index coding.

Allowed Characters

Allowed are capital letters, digits, the arithmetic operators plus mark, minus mark and multiplication mark and some special characters.

Variable Names

A name is a string of 1-6 alphanumeric characters beginning with a letter. As further restrictions must be observed:

- If time is used explicitly it must be denoted by the letter T, a variable always identified with time and therefore not allowed to identify other variables.
- Occasionally the program generates names by itself, namely in case of automatic derivation of a position vector, differentiation of auxiliary variables and calculation of compressed equations of motion. Names must be chosen so as to prevent ambiguities with these automatically generated names.

Constants

Constants can be integers or reals in the range of 10^{-10} through 10^{+10} . However, the program is able to deal with even lower and higher constants. The exact range depends on the computer in use. Constants overstepping the above range should not be used, though.

Operations, Brackets

Allowed operations are: + , - , * , ** , i.e. addition, subtraction, multiplication and exponentiation. Long expressions are reduced according to the usual arithmetic and the usual bracket rules. Exponents must be positive integers.

Functions

Allowed functions are sine and cosine. Further functions can be represented by means of auxiliary variables and simplification equations. The argument may be a simple variable or an expression.

Error Codes

If an error occurs during the program execution and the program is able to detect it, NEWEUL indicates it by writing

* NEWEUL Fxx .

Here xx stands for the error code, the meaning of which is explained in the NEWEUL - Error Listing. After this message calculation of the equations of motion by the program NEWEUL is aborted at once.

Examples

For the two test examples defined in this handbook, the equations of motion are generated symbolically by the program NEWEUL. These equations are simply solved by numerical integration for simulation of motion, using e.g. the simulation package NEWSIM. The detailed results are given in Ref. [8].

Mechanism

The compressed equations of motion for the mechanism read as follows:

$$\begin{aligned}
 M &= M1 * RA^{**2} & + LT2(1,1)^{**2} * M2 & + \\
 & LT2(2,1)^{**2} * M2 & + LT3(1,1)^{**2} * M3 & + \\
 & LT3(2,1)^{**2} * M3 & + LT4(1,1)^{**2} * M5 & + \\
 & LT4(2,1)^{**2} * M5 & + LT5(1,1)^{**2} * M4 & + \\
 & LT5(2,1)^{**2} * M4 & + LT6(1,1)^{**2} * M7 & + \\
 & LT6(2,1)^{**2} * M7 & + LT7(1,1)^{**2} * M6 & + \\
 & LT7(2,1)^{**2} * M6 & + I1 & + \\
 & LR2(3,1)^{**2} * I2 & + I3 * GA01^{**2} & + \\
 & I5 * DE01^{**2} & + LR5(3,1)^{**2} * I4 & + \\
 & I7 * EP01^{**2} & + LR7(3,1)^{**2} * I6 & \\
 \\
 K &= AQT2(1) * LT2(1,1) * M2 + AQT2(2) * LT2(2,1) * M2 + \\
 & AQT3(1) * LT3(1,1) * M3 + AQT3(2) * LT3(2,1) * M3 + \\
 & AQT4(1) * LT4(1,1) * M5 + AQT4(2) * LT4(2,1) * M5 + \\
 & AQT5(1) * LT5(1,1) * M4 + AQT5(2) * LT5(2,1) * M4 + \\
 & AQT6(1) * LT6(1,1) * M7 + AQT6(2) * LT6(2,1) * M7 + \\
 & AQT7(1) * LT7(1,1) * M6 + AQT7(2) * LT7(2,1) * M6 + \\
 & AQR2(3) * LR2(3,1) * I2 - AQR3(3) * I3 * GA01 - \\
 & AQR4(3) * I5 * DE01 + AQR5(3) * LR5(3,1) * I4 - \\
 & AQR6(3) * I7 * EP01 + AQR7(3) * LR7(3,1) * I6 \\
 \\
 QE &= FE3(1) * LT3(1,1) & + FE3(2) * LT3(2,1) & + \\
 & KM & - LE3(3) * GA01
 \end{aligned}$$

The generalized variable is the angle β , auxiliary variables are, e.g., $\gamma(\beta)$, $\delta(\beta)$, $\epsilon(\beta)$. In addition to the mechanical parameters, the elements of the Jacobian matrices LT (...), LR (...), the acceleration components AQT (...), AQR (...) and the first partial derivatives DE01, EP01, GA01 are used. The simulation result is shown in Fig. 5.

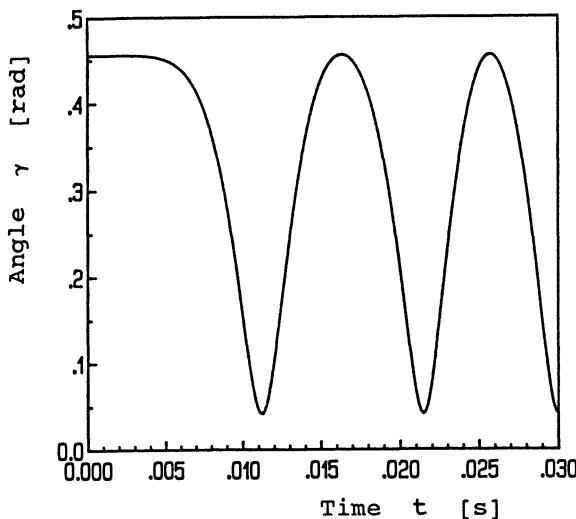


Fig. 5. Motion of mechanism without friction

Moreover, the program NEWEUL offers the possibility for the computation of the constraint or normal forces, respectively, with the package ZKSUB necessary for the inclusion of friction. For comparison, the simulation result is shown in Fig. 6.

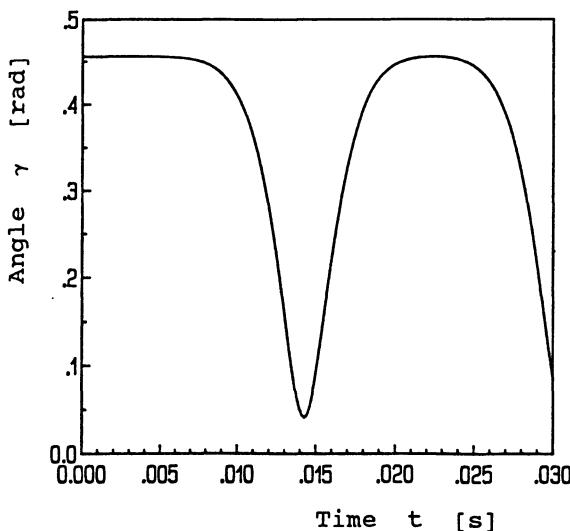


Fig. 6. Motion of mechanism with friction

Robot

Some characteristic elements of the symbolical equations of motion for the robot read as

$$\begin{aligned} M(1,1) &= M_1 + \\ &M_2 + \\ &M_3 \end{aligned}$$

$$M(2,1) = 0.$$

⋮

$$\begin{aligned} K(1) &= -2.*M_3*C*AL3P*BE2P*SIN(BE2)*COS(AL3) - \\ &M_3*C*AL3P**2*SIN(AL3)*COS(BE2) - \\ &M_3*C*BE2P**2*SIN(AL3)*COS(BE2) \end{aligned}$$

⋮

$$\begin{aligned} Q_E(5) &= -GE*M_3*C*COS(AL3)*COS(BE2) + \\ &L3X \end{aligned}$$

Furthermore, the animation of motion can be obtained from the simulation as presented in Fig. 7.

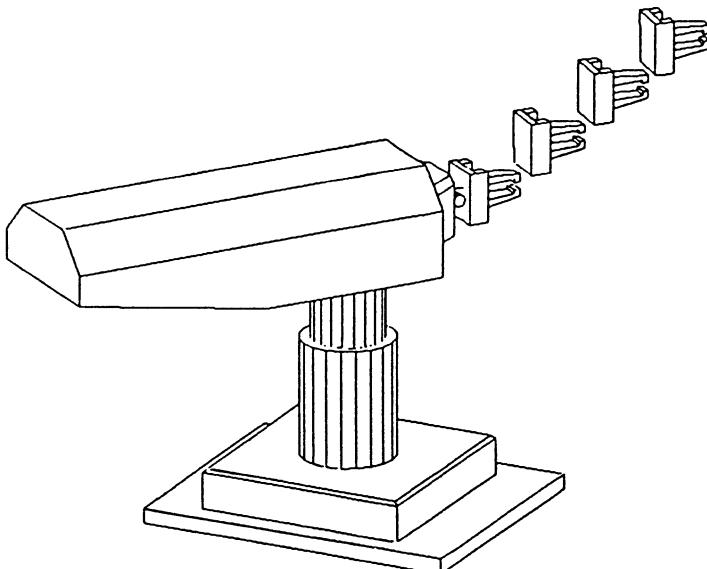


Fig. 7 Animation of robot motion

Acknowledgements

The development of the software NEWEUL started at the University of Stuttgart in 1977. Major contributions to the version NEWEUL '88 are due to Matthias Krieg, Klaus-Peter Schmoll, Dieter Schramm, Markus Ast, Bernd Keßler, Günter Leister.

References

- [1] Kreuzer, E.: Symbolische Berechnung der Bewegungsgleichungen von Mehrkörpersystemen. Düsseldorf: VDI-Verlag, Fortschr.-Ber. VDI-Z., Reihe 11, Nr. 32, 1979.
- [2] Schiehlen, W.: Technische Dynamik. Stuttgart: Teubner 1985.
- [3] Schmoll, K.-P.: Modularer Aufbau von Mehrkörpersystemen unter Verwendung der Relativkinematik. Düsseldorf: VDI-Verlag, Fortschr.-Ber. VDI-Z., Reihe 18, Nr. 57, 1988.
- [4] Schramm, D.: Ein Beitrag zur Dynamik reibungsbehafteter Mehrkörpersysteme. Düsseldorf: VDI-Verlag, Fortschr.-Ber. VDI-Z., Reihe 18, Nr. 32, 1986.
- [5] Leister, G.: Programmpaket NEWSIM. Stuttgart: Universität Stuttgart, Institut B für Mechanik, Anleitung AN-22, 1989.
- [6] Shampine, L.F.; Gordon, M.K.: Computer Solution of Ordinary Differential Equations. San Francisco: Freeman 1975.
- [7] Schiehlen, W.: Modeling, Simulation and Animation of Multibody Systems. In: Future Trends in Applied Mechanics. To appear.
- [8] Daberkow, A.; Eismann, W.; Schiehlen, W.: Test Examples for Multibody Systems. Stuttgart: Universität Stuttgart, Institut B für Mechanik, Institutsbericht IB-13, 1989.

MEDYNA – An Interactive Analysis and Design Program for Geometrically Linear and Flexible Multibody Systems

Oskar Wallrapp and Claus Führer

Institute for Flight Systems Dynamics
German Aerospace Research Establishment (DLR)
D-8031 Wessling (FRG)

1 Introduction

Simulation and computer-aided analysis of complex mechanical systems has become a task of increasing importance in computational mechanics. It requires software tools combining *modeling support, efficient generation of the equations of motion* as well as *modern numerical solution and system analysis techniques*.

MEDYNA¹ is an integrated approach - based on the method of multibody systems (MBS) - for simulating the dynamics of a large variety of mechanical systems such as *rail-guided and road vehicles, machines etc..*

The development of MEDYNA was initiated to assist the German high-speed railway research by the “Deutsche Eisenbahn Consulting” (DEC) together with some major companies of the railway vehicle industry (e.g. MAN Technology, Krauss-Maffei, MBB, Thyssen-Henschel). Therefore MEDYNA has been equipped with special features for modeling, analysing and simulating rail-guided vehicles as railroad cars and magnetically levitated vehicles.

We will present here the key features of MEDYNA together with its basic modeling assumptions. Furthermore a detailed description of the components used to construct a model in MEDYNA will be given and current applications will be summarized.

A final section considers the solution of the mechanism example of this handbook.

2 Multibody System Components and Model Assumptions of MEDYNA

For describing a complex mechanical system using a multibody formalism the following elements are used:

¹MEDYNA stands for the German expression MEhrkörperDYNAmik (= multibody dynamics)

- frames,
- bodies
- interconnections, and
- external excitations.

For increasing the variety of systems MEDYNA provides also special

- substructures

for coupling subsystems to the multibody model. An example of a multibody model composed by these components is presented in Fig. 1.

Coordinate Systems

The motion of the MBS is described with respect to a *reference coordinate system*², E^0 , which may be fixed in the inertial space (the *inertial coordinate system* is E^I) or which may perform a prescribed motion, see Fig.1.

Each body of the MBS has its own *body-fixed coordinate system*, E^i . It is used for describing data related to the individual bodies as attachment points of interconnections, external forces acting on the body etc.

Bodies

MEDYNA handles *rigid bodies* with nonzero masses and moments of inertia, reduced models like *planar bodies* and *point masses* as well as *flexible bodies*.

The following modeling assumption is made in MEDYNA:

Assumption 2.1 *The motion of the bodies and their deformation are assumed to be small with respect to a given nominal state. The nominal state is defined by the nominal configuration, by the nominal motion and by nominal forces and torques acting on the system. Thus, the kinematics of the MBS are linearized with respect to these quantities (geometric linearity).*

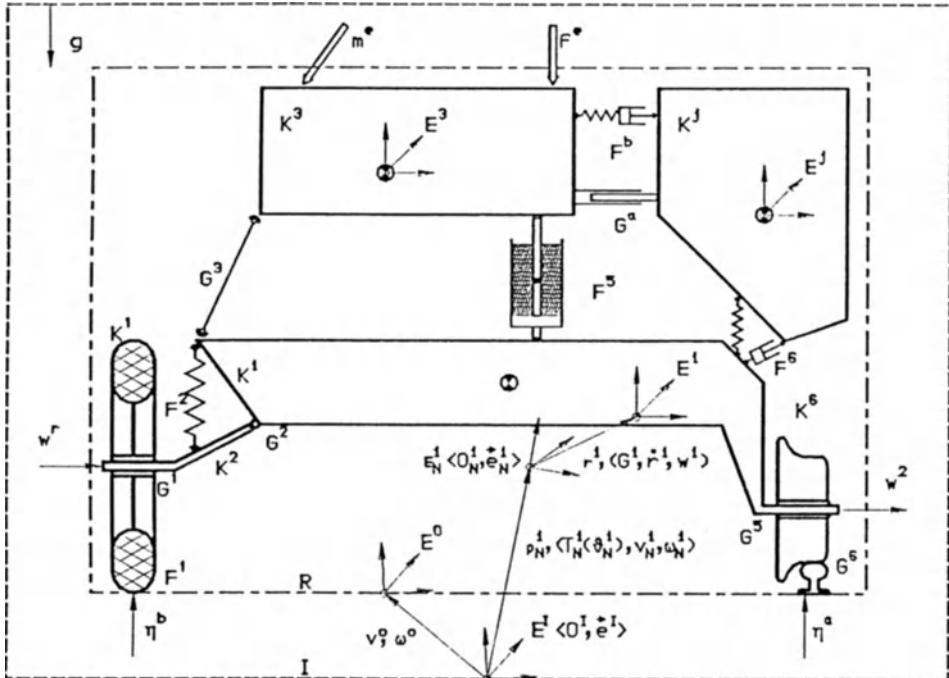
Nominal Motions

Nominal motions are motions of the reference system E^0 with respect to the inertial system E^I . They depend also on the nominal rotation of gyrostats with angular velocity w^r (see Fig. 1). They are causing external forces and torques acting on the MBS.

Rigid Body Data

The rigid body data are described with respect to the body-fixed coordinate system E^i (i being the index of the individual body) or, alternatively, with respect to a user defined

²MEDYNA allows also the use of several different reference systems to model larger structures e.g. railway trains



I: Inertial space (inertial system E^I)

R: Reference space (reference system E^0)

K^i : Rigid or elastic bodies

F^a : Interconnections: joints

w^r : Angular velocity of a gyrostat

F^b : Interconnections: force elements

f^e, m^e : External forces and torques

g : Acceleration of gravity

η^a : Kinematical excitations

v^0, ω^0 : Motion of the reference system E^0

η^a : Kinematical excitations

Figure 1: Components of the Multibody System

auxiliary coordinate system. The data consist of the *center of mass*, the *mass* and the *moments of inertia*, and the location of all *attachment points*. Attachment points are those points where the interconnections are attached to the body or points where the results of a simulation are of special interest (e.g. measurement points). In order to define a reduced body model, the restrictions on the free motions of the individual bodies with respect to E^0 are also considered as body data (e.g. planar model).

Flexible Body Data

The position r^p of special material points (nodes) of a body i , ($i = 1, \dots, n_K$), is described with respect to the body-fixed frame E^i by the reference position vector c^i and the displacement function u^i , which is expressed by a finite dimensional approximation:

$$r^p(c^i, t) = c^i + u^i(c^i, t) \quad \text{with} \quad u^i(c^i, t) = \Phi^i(c^i)q^i(t). \quad (2.1)$$

Herein Φ^i is a $3 \times n_q^i$ - matrix of space dependent assumed modes (*shape functions*) and q^i is a $n_q^i \times 1$ - vector of time dependent *deformation coordinates*.

This formulation requires a discrete model of the flexible body, which can be obtained applying the *finite element method* (FEM) or the *continuum approach*, resulting in the ordinary differential equation (ODE) [2]:

$$M_q^i \ddot{q}^i + D_q^i \dot{q}^i + (K_q^i + N_q^i)q^i = 0. \quad (2.2)$$

Again, Assumption 2.1 requires $q^i(t)$ being small. The input data for flexible MBS consist of the *shape functions* Φ^i , the *generalized mass matrix* M_q^i , the *damping matrix* D_q^i and the *stiffness matrix* K_q^i in addition to the rigid body data. In the case of nonzero nominal stresses a *generalized geometric stiffness matrix* N_q^i must be given, too. The preprocessor ASKESE provides MEDYNA with these data obtained from FEM - programs, see Section 5. Also, these data can be taken from measurements directly.

Interconnections

The interaction between two coupled bodies or between a body and the environment is modeled by *interconnections*. These are considered to be massless elements of the MBS causing interaction forces and torques. They are modeled by combining elementary *coupling elements*. These coupling elements can be divided in three classes:

- joints,
- force elements and
- friction elements.

Joints are restricting the relative motion between interconnected bodies and are causing corresponding *constraint forces and torques*. The restrictions are described in terms of kinematical relations in velocity and position coordinates, the so-called *system constraints*. MEDYNA treats MBS with linear holonomic constraints.

Force elements introduce applied forces and torques, which depend on the relative motion of the contiguous bodies and in some cases on further quantities. The interaction laws can be rather general: linear, nonlinear, statical, dynamical, active, and passive [23].

Friction elements result in applied forces and torques being themselves functions of the constraint forces and torques.

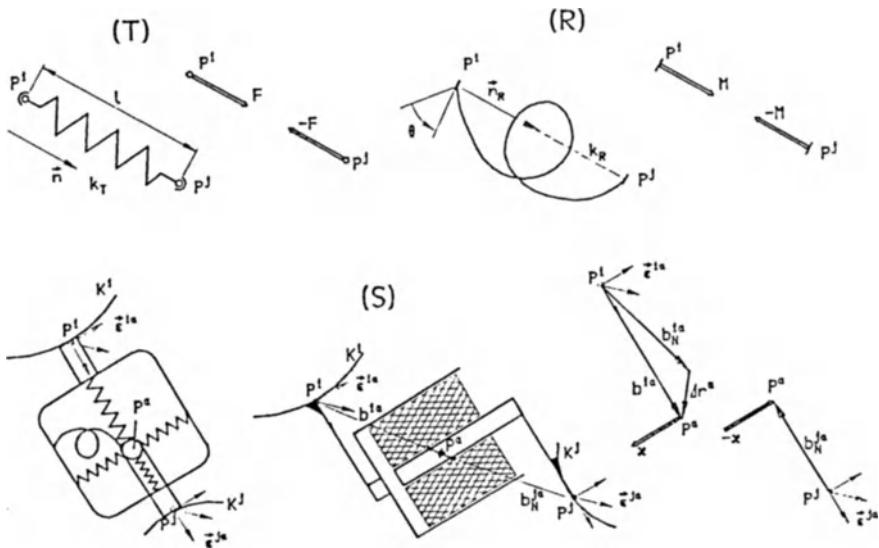
A broad class of coupling elements is predefined in a MEDYNA *coupling element library*, which is presented in Table 1.

A special coupling element is provided to describe the interaction across a wheel-rail interface. It is based on a linearized *wheel-rail contact geometry* and linearized force laws, by applying the techniques of harmonic linearization [15]. This *wheel-rail coupling element* is a combination of a joint, a force element, and a friction element³.

The force elements themselves can be subdivided into three subclasses (ref. Fig. 2):

³Further simulation possibilities for wheel-rail systems are the substructures as described below.

- *axial, translatory elements (T)* described by their interaction forces F^a , lengths l^a and their directions n^a ,
- *axial, rotatory elements (R)* described by their interaction torques M^a , their angles θ^a and their body-fixed directions n_R^a ,
- *spatial elements (S)* described by their interaction forces and torques $\kappa^a = \{F_1^a, F_2^a, F_3^a, M_1^a, M_2^a, M_3^a\}$ and relative coordinates ξ^a defined with respect to a special coordinate frame related to the particular coupling element, the *coupling element auxiliary frame*.



T: axial, translatory, R: axial, rotatory, S: spatial

Figure 2: Subclasses of Standard Force Elements in the MEDYNA - Library

General interconnections can be described by superimposing catalogued interaction laws. For special purposes the user can define additional interaction laws, which may depend on relative quantities, state variables, time, excitations and output variables.

Topology of the Multibody System

The topology of the MBS is defined in MEDYNA by describing the nominal position of the individual bodies with respect to the reference frame. The interconnections then are assigned to the bodies in an order given by the user. It is not necessary to distinguish between tree structured or closed loop systems.

For the definition of the topological structure of the MBS the following assumption is made:

Table 1: Library of Interconnection Elements in MEDYNA

Linear Elements	Parameters	Symbol
Linear spring	k_T, F_N	
Torsional spring	k_R, M_N	
Linear damper	d_T	
Torsional damper	d_R	
Linear spring and damper in series	k_T, d_T	
Torsional spring and damper in series	k_R, d_R	
Spherical element	K_S, D_S, κ_N	
Static or dynamic active element depending on the linearized state vector ζ_t , the control input u , and the measurement vector ξ_M	Matrices relating F, M, ζ_t, u, ξ_M	
Linkage	l_0	
General constraint element	Z, ζ_0	
Linear wheel-rail element	ORE-parameters generated by RS GEO	
Nonlinear Elements	Parameters	Symbol
General, static or dynamic element	to be specified in prepared user routines or by given model structures	
Nonlinear wheel-rail element	Profile data, material constants	

Assumption 2.2 An interaction always acts on two attachment points belonging in case of rigid bodies to different bodies. The reference system is treated in this context as a rigid body which is identified by the body index $i = 0$.

External Actions and Excitations

MEDYNA allows to model excitations of the systems in terms of prescribed motion, forces or torques. In particular these are (see Fig. 1):

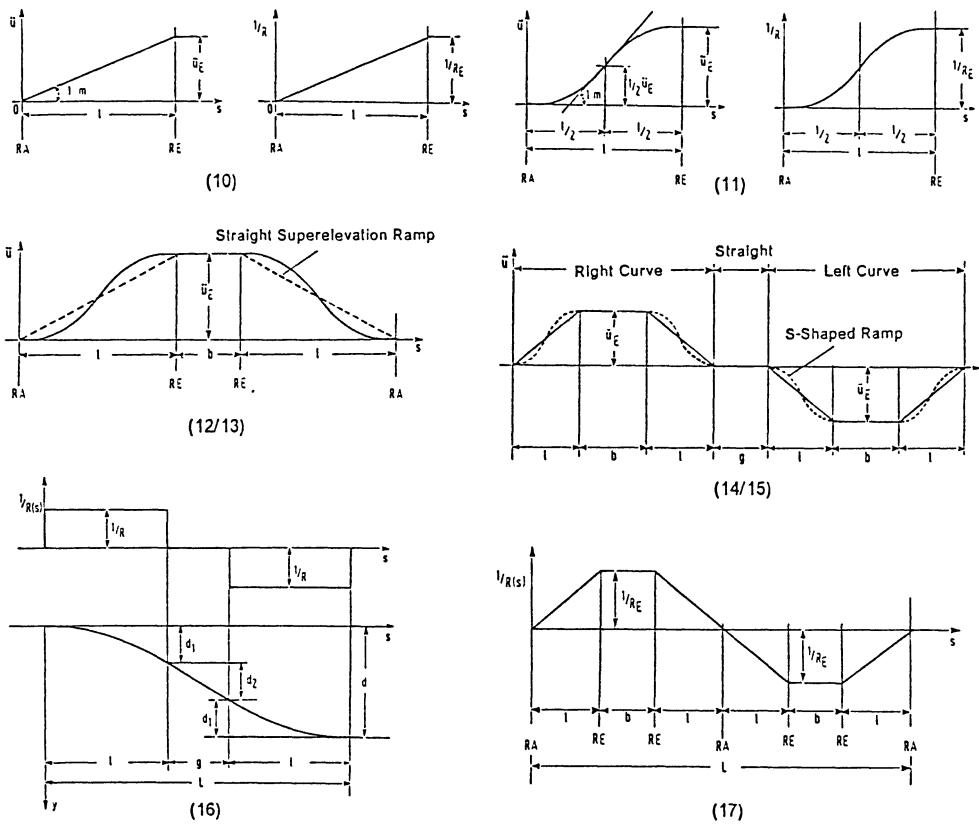
- the constant gravitational acceleration,
- the global motion of the reference frame,
- the nominal angular velocity of gyrostats,
- external applied forces f^e and torques m^e like preloads, wind forces, and pressure forces acting at attachment points of the bodies,
- kinematical excitations $\eta(t)$ defined at the attachment points of interconnections linked to the reference space (body 0) like prescribed displacements, rotations, velocities and accelerations. These are tools to describe rail disturbances and road irregularities. They result in interaction forces in the interconnections.

The global motion of the reference system is specified in MEDYNA by the following models:

- motion with constant speed along a prescribed curve being a
 - straight line (nominal motion)
 - circle with radius R^0 resulting in a constant angular velocity ω^0 (nominal motion)
 - curve defined by a time function for the angular velocity, $\omega^0 = \Delta\omega^0(t)$, which is catalogued in a special MEDYNA curving library, see Table 2.

The *external forces* and torques are described by their constant nominal values and small deviations. For *kinematical excitations* the nominal value is always assumed to be zero. The deviations in either case can be space-, time-, or frequency - dependent functions which can be taken from the MEDYNA library of excitation functions or defined by the user. In addition there is the feature in MEDYNA to handle also excitation models as stochastic processes, which is often required for modeling road or rail irregularities. These processes are given in terms of their power spectral density functions or (in time domain) by an appropriate shape filter.

Table 2: Library of Track Models (Type 10 -17)

**Notation:**

- 10 Transition track with straight superelevation ramp
 - 11 Transition track with S-shaped swung superelevation ramp
 - 12/13 Superelevated track with no sign change in track curvature
 - 14/15 Superelevated track with a sign change in track curvature and a point symmetric shape
 - 16 Cross over modeled by two circular tracks and a straight section (without superelevation)
 - 17 like model 16 but with superelevation and a transition track
- \bar{u} = Superelevation level, $1/R$ = Curvature, s = Track distance

Substructures

Beside typical multibody components like bodies, interconnections and external actions MEDYNA also makes use of special substructure elements.

Generally, *substructures* are defined as complex mechanical, electrical or hydraulical subsystems, for which a multibody formalism is not applicable.

A typical example for such a substructure is the complex, nonlinear model of the wheelset

motion on a railway track, see Fig. 3. In MEDYNA different wheelset substructure models are provided, [4]. Substructure elements are coupled to the MBS by force elements. The corresponding interactions depend on the motion of the MBS at their attachment points and on the states of the substructure.

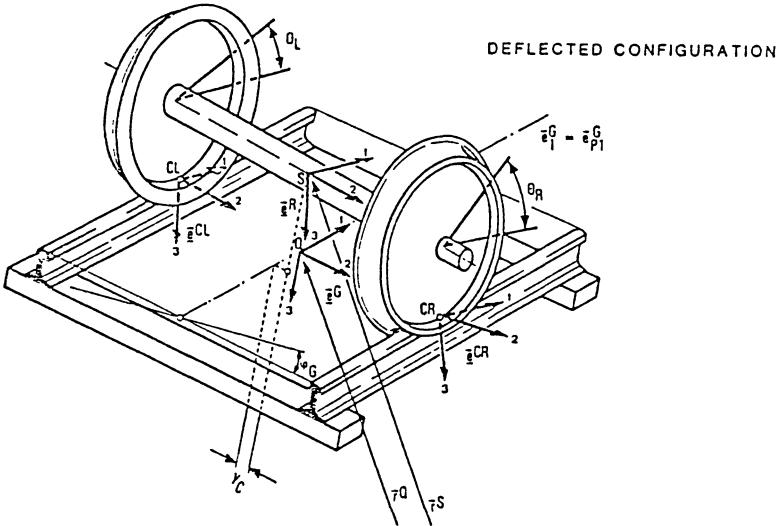


Figure 3: Model of a Wheel-Rail Substructure

3 Multibody Formalism and System Equations

3.1 Generalized Position and Velocity Coordinates

In order to describe small motions with respect to a given nominal state the components $r^i = \{r_\alpha^i\}$, $\alpha = 1, 2, 3$ of the vector $\vec{r}^i = e_N^T r^i$ and the Cardan angles $\gamma^i = \{\gamma_\alpha^i\}$, $\alpha = 1, 2, 3$ of the direction cosine matrix G^i are used in MEDYNA as rigid body coordinates (see Fig. 1). If flexible bodies are incorporated in the MBS this set of coordinates is supplemented by the deformation coordinates q^i , (2.1). For all bodies ($i = 1, \dots, n_K$) these coordinates are assembled to form the vector

$$p := \left\{ \begin{pmatrix} r^i \\ \gamma^i \\ q^i \end{pmatrix} \right\} = \left\{ p^i \right\}_{i=1, \dots, n_K} \quad \text{with} \quad p^i = \begin{pmatrix} r^i \\ \gamma^i \\ q^i \end{pmatrix} \quad (3.1)$$

and the dimension $n_p = \sum_{i=1}^{n_K} n_s^i + n_q^i$.

The number of rigid body position coordinates varies between zero and six corresponding to the complexity of the given model (planar, spatial).

In the same way as p the position vectors π_N and π of the nominal and total motion are defined by:

$$\pi_N(t) := \left\{ \begin{Bmatrix} \rho_N^i \\ \vartheta_N^i \\ q_N^i \end{Bmatrix} \right\}, \quad \pi(t) := \left\{ \begin{Bmatrix} \rho^i \\ \vartheta^i \\ q^i \end{Bmatrix} \right\} = \pi_N + p. \quad (3.2)$$

(Herein and in the sequel the notation using the two braces has the same meaning as defined in eq. (3.1)).

As generalized rigid body *velocity coordinates* the derivative with respect to time of r^i and q^i are combined with the components w^i of the angular velocity $\vec{\omega}$ defined in the basis \vec{e}_N^i :

$$s := \left\{ \begin{Bmatrix} \dot{r}^i \\ w^i \\ \dot{q}^i \end{Bmatrix} \right\}. \quad (3.3)$$

Again, the velocity vector s has the dimension n_p .

The components w^i of the angular velocity depend linearly on $\dot{\gamma}^i$ and nonlinearly on the angles γ^i . In MEDYNA the linearized kinematical equations are used:

$$s = \dot{p}. \quad (3.4)$$

3.2 Relative Quantities for Interconnections

As pointed out in Chapter 2 MEDYNA provides three classes of coupling elements: axial translatory, axial rotatory and spatial elements (subscript T, R, S). The corresponding linearized kinematic relationships between the relative coordinates of the interconnections (superscript a) and the generalized coordinates as defined above are

$$\begin{aligned} l^a &= l_N^a(\pi_N) + \Delta l^a(p, t), & \Delta l^a &= S_T^a p + V_T^a \eta_T(t), & \Delta \dot{l}^a &= S_T^a \dot{p} + V_T^a \dot{\eta}_T(t), \\ \vartheta^a &= \vartheta_N^a(\pi_N) + \Delta \vartheta^a(p, t), & \Delta \vartheta^a &= S_R^a p + V_R^a \eta_R(t), & \Delta \dot{\vartheta}^a &= S_R^a \dot{p} + V_R^a \dot{\eta}_R(t), \\ \xi^a &= \xi_N^a(\pi_N) + \Delta \xi^a(p, t), & \Delta \xi^a &= S_S^a p + V_S^a \eta_S(t), & \Delta \dot{\xi}^a &= S_S^a \dot{p} + V_S^a \dot{\eta}_S(t). \end{aligned} \quad (3.5)$$

The time dependent function $\eta = (\eta_T, \eta_R, \eta_S, \dot{\eta}_T, \dot{\eta}_R, \dot{\eta}_S)_{a=1,\dots}$ describes the *kinematical excitations*. The matrices $S_{T,R,S}^a$ and $V_{T,R,S}^a$ are constant geometry and excitation matrices.

For axial coupling elements the linearized dependency of the direction of action on the generalized coordinates is given by

$$n^a = n_N^a + \Delta n^a(p). \quad (3.6)$$

In MEDYNA the following assumption is made:

Assumption 3.1 *The nominal relative quantities l_N^a , ϑ_N^a and ξ_N^a are assumed to be time invariant, i.e. $\dot{l}_N^a = 0$, $\dot{\vartheta}_N^a = 0$, and $\dot{\xi}_N^a = 0$.*

3.3 Constraints

The joints in an MBS like revolute joints, prismatic joints, connecting rods (see Fig. 2) constrain the free motion of the bodies. In practice all joints can be described by *holonomic constraints*.

The holonomic constraints of a joint are defined in terms of the relative coordinates ξ^a by the condition $\varphi^a(\xi^a) = 0$ with φ being a n_z^a -vector valued function. If these contributions of all joints of the MBS are assembled together and transformed into generalized coordinates the $n_z := \sum_a n_z^a$ linearized constraint equations and the corresponding nominal conditions are obtained:

$$C_z^T p - V_\eta \eta(t) = 0 \quad \text{and} \quad \varphi(\pi_N) = 0. \quad (3.7)$$

Assumption 3.2 The $n_p \times n_z$ constraint matrix C_z and the $n_z \times n_\eta$ excitation matrix V_η are assumed to be time invariant.

Using (3.4) the velocity coordinates are then constrained by

$$C_z^T s - V_{\dot{\eta}} \eta(t) = 0. \quad (3.8)$$

Table 3: Types of Actions Induced by MBS-Components in MEDYNA

MBS component	Physical Quantity	Type	Generalized applied forces		
			$g = g_N - Np + \Delta g$	g_N	N
Joints	$f_z^a, m_z^a(\lambda^a)$	O, z, i	g_{zN}	N_z	$\Delta g_z = -C_z \Delta \lambda$
Force elements	$f_f^a, m_f^a(p, s, t)$	O, p, i	g_{fN}	N_f	$\Delta g_f(p, s, t)$
Friction elements	$f_{fz}^a, m_{fz}^a(\lambda^a)$	O, p, i	g_{fzN}	N_{zf}	$\Delta g_{fz} = -C_{fz} \Delta \lambda$
Kinem. excitation	$\eta^a(t)$	O, p, a			
External forces and torques	$f^e(t), m^e(t)$	O, p, a	g_{eN}	N_e	$\Delta g_e(t) = R_e \Delta \kappa_e(t)$
Gravitational acc.	g	V, p, a	g_{gN}	N_g	$\Delta g_g(t)$
Reference motion	$v^0, \omega^0(t)$	V, p, a	g_{cN}	N_c	$\Delta g_c(t) - K_c p - D_c s$
Gyrostatic effects	w^r	V, p, a	g_{rN}	N_r	$\Delta g_r(t) - D_r s$

Notation:

Type: O = surface action, V = volume action, z = constraint action,

p = applied action, i = internal interaction, a = external action,

Indices: N = nominal quantity, Δ = incremental quantity

Matrices: N = geometric stiffness matrix, D = damping matrix, K = stiffness matrix,

C_{fz} = friction matrix

3.4 Jourdain's Principle and Generalized Forces

In order to apply Jourdain's principle virtual velocities δs are introduced. These must satisfy the implicit constraint equations (3.8), which can be achieved by considering additional Lagrange multipliers λ . Again, these are partitioned into a nominal part and a small deviation: $\lambda = \lambda_N + \Delta\lambda$. Then, for a general MBS, Jourdain's principle results in the following equation:

$$\delta s^T \{ M \dot{s} + D_q s + K_q p + C_z \Delta\lambda + N_z p - g_p(\pi_N, \lambda_N, p, s, \Delta\lambda, t) \} = 0. \quad (3.9)$$

The first order term in a Taylor series expansion of φ contributes together with λ_N to the geometric stiffness matrix N_z ⁴. The other terms in eq. (3.9) are

$M = \text{diag}[M^i]_{i=1,\dots,n_K}$ $n_p \times n_p$ - total mass matrix

$D_q = \text{diag}[\text{diag}[0, 0, D_q^i]]_{i=1,\dots,n_K}$ $n_p \times n_p$ - total modal damping matrix

$K_q = \text{diag}[\text{diag}[0, 0, K_q^i + N_q^i]]_{i=1,\dots,n_K}$ $n_p \times n_p$ - generalized stiffness matrix of flexible bodies

$g_p(\pi_N, \lambda_N, p, s, \Delta\lambda, t)$ $n_p \times 1$ - vector of the generalized applied forces resulting from motions of the reference systems, gravitation, external forces and torques, actions of force and friction elements.

The linear form of the generalized applied forces consists of three parts:

$$g_p(\pi_N, \lambda_N, p, s, \Delta\lambda, t) = g_{pN}(\pi_N, \lambda_N) + \Delta g_p(p, s, \Delta\lambda, t) - N_p p \quad (3.10)$$

with N_p being again a geometric stiffness matrix, which depends on the nominal actions (forces and torques).

The expansion of the individual components is shown in Table 3.

The following fundamental equations are derived from eqs. (3.9) and (3.10) and are given here for the mere purpose for indicating their structure:

1. Nominal conditions of kinematics and statics:

$$\begin{aligned} \varphi(\pi_N) &= -z_N \stackrel{!}{=} 0 & \text{and} & \quad g_{pN}(\pi_N, \lambda_N) = g_N \stackrel{!}{=} 0 & \text{with} \\ g_N &= -[C_z + C_{fz}] \lambda_N + R_f \kappa_{fN} + g_{aN} \end{aligned} \quad (3.11)$$

2. Linearized Lagrange equations of the first kind

$$\begin{aligned} M \dot{s} + D s + K p + [C_z + C_{fz}] \Delta\lambda &= R_\zeta \zeta_l + R_n \Delta \kappa_n + R_\eta \eta + R_e \Delta \kappa_e \\ &\quad R_s \Delta \kappa_s + \Delta g_t \end{aligned} \quad (3.12)$$

$$\dot{\zeta}_l = A_{\zeta l} \zeta_l + A_{\zeta p} p + A_{\zeta s} s + A_{\eta} \eta \quad (3.13)$$

$$\dot{\zeta}_n = \dot{\zeta}_n(p, s, \eta, t) \quad (3.14)$$

$$\dot{\zeta}_s = \dot{\zeta}_s(\zeta_s, \Delta \kappa_s, t) \quad (3.15)$$

with the following coupling relations and notations:

⁴A detailed discussion of geometric stiffness matrices in different MBS-approaches is given in [23].

$D := D_q + D_f + D_c + D_r$ being the damping matrix,

$K := K_q + K_f + K_c + N$ the stiffness matrix,

$\Delta g_t(t) := \Delta g_g(t) + \Delta g_c(t) + \Delta g_r(t)$ the time dependent generalized forces,

ζ_l the linear dynamical actions,

$\Delta \kappa_n = \Delta \kappa_n(p, s, \eta, \zeta_n, t)$ nonlinear actions and the corresponding states ζ_n , and

$\Delta \kappa_s = \Delta \kappa_s(p, s, \eta, \zeta_s, t)$ nonlinear actions and the corresponding states ζ_s of the wheel-rail substructure.

The equations of motion (3.4), (3.12) - (3.15) together with the constraint equations (3.7) define a system of differential-algebraic equations (DAEs).

4 Solution Methods in MEDYNA

An important advantage of MEDYNA is the combination of equation generation techniques with efficient numerical solution and analyzing methods. Special care has been taken to adapt the numerical methods to the demands of computational mechanics and especially to computational vehicle dynamics. This is reflected in the choice of numerical algorithms ranging from nonlinear equation solvers adapted for the investigation of quasistatic curving behaviour of railroad vehicles and system analysis methods for linear deterministic and stochastic systems to modern codes for numerical integration of stiff and nonstiff ODEs.

4.1 Computation of a Static Equilibrium

The linearized Lagrange equations (3.12) - (3.15) require the nominal conditions of kinematics (3.10) and statics (3.11) to be satisfied. If a given MBS is not in a static (or for moving reference frames: quasistatic) equilibrium configuration MEDYNA provides two different approaches to fulfil the equilibrium requirements.

The first approach involves a computation that "adjusts" the interaction forces, κ_N, λ_N , and "freezes" the position of the bodies. The interaction forces corresponding to a system in an equilibrium state are numerically evaluated by solving a particular linear system depending on given values for $g_a N$, (3.11). Details are given in [23, ch.7.2].

The second approach "adjusts" the position of the bodies, the constraint forces , and consequently the applied interactions, so that for given external actions the system is transformed into a static equilibrium position. This method is iterative as a nonlinear system of algebraic equations must be solved. In MEDYNA this is done by an incremental continuation method based on a sequence of Newton-Raphson iterations, [23, ch.7.3], [17].

4.2 Transformation to State Space Form by Eliminating the Constraint Forces

The Lagrange equations (3.4), (3.12) - (3.15), (3.7) consist of a system of differential-algebraic equations. Whereas in the general nonlinear case special techniques are necessary

for handling this type of equations [1], the model class considered by MEDYNA allows to solve the constraint equations by some nullspace technique globally [8]. By using this method

- the number n_y of degrees of freedom is determined,
- an appropriate set of state variables y is selected together with the Jacobians [19] J_y, J_η satisfying $p = J_y y + J_\eta \eta$, which is the explicit form of the constraint equation (3.7),
- the system is reduced to its state space form, which consists of a lower order unconstrained system of ordinary differential equations with the constraint forces $\Delta\lambda$ being eliminated.

4.3 Linear System Analysis Methods

For linear system analysis purposes MEDYNA establishes the matrices of the state space form, which are the *system matrix* defined by mass, stiffness and damping terms, the *input matrix* corresponding to excitations and external actions and the *output matrices* belonging to specified output quantities (see Sec. 4.5). MEDYNA includes the following analysis methods [6]:

- *Modalanalysis* for investigating the stability of the linearized system in terms of its eigenvalues and eigenvectors,
- *Frequency response* and Bode diagrams,
- *Power spectral density function* of specified output quantities and *ride comfort criteria*, for analysing the performance of vehicles running over randomly disturbed roads or tracks,
- *Covariance analysis* for random disturbances given as an output of a linear dynamic filter [16].

4.4 Numerical Integration Methods

The nonlinear state equations are solved with *numerical integration* methods in the time domain. Numerical integration methods available in MEDYNA include a Runge-Kutta-Bettis code with error control and variable stepsize, and two multistep codes. Multistep codes are best suited for problems with dense output or “right-hand-side functions” which are expensive to evaluate. MEDYNA includes the multistep code, DE, for nonstiff problems, which is based on variable order Adams-Basforth-Moulton formulas [22], the other code, LSODA, is adaptive for systems with varying stiffness properties [18]. It automatically switches to ADAMS-formulas and fast iteration schemes designed for nonstiff problems and uses BDF-formulas (“Gear’s method”) together with the more expensive Newton-iteration only when required by stability demands in case of stiff mechanical systems. This code, also, has variable stepsizes and variable order, which drops to a simple

backward Euler formula if the problem is not sufficiently smooth. Stiff systems occur frequently in models with friction elements, road vehicles with special tire models and low speed railroad vehicles, [7].

4.5 Output Relations

In order to analyse the MBS sometimes not only the state variables are required, but also other output quantities like

- the relative- and absolute motion at given observation points,
- interaction force and torques in the interconnections,
- a linear combination of the generalized position and velocity coordinates,
- a linear combination of x_n , \dot{x}_n , u and $\Delta\kappa_n$,
- any nonlinear relation of the coordinates (defined by a user-given function).

5 Software Aspects and Handling

MEDYNA is an integrated interactive program containing the aforementioned modeling options and computational methods.

Handling

Referring to Fig. 4 the user is offered a menu of main routines from which he chooses the desired input or computational routine. All procedures, such as system definition, equation generation and system analysis, are unified into *one* program. Therefore the user can complete a design task and perform a system analysis in one session with uniform handling. Most of undesired program interrupts are avoided by a numerous set of built-in validity checks. Both short and detailed help-texts are available and self explanatory.

A special data organization concept [21] allows termination of the session at any input point, without loss of any of the previous session data or computational results (*restart capability*). All model data and results are stored in a single data file. By copying such a data file and subsequent modification, new models are easily built-up. As output medium, the user's terminal is used to display relevant remarks (program warnings or errors, as well as status information) and brief results. With a printer or graphics terminal, the user can print or display a detailed set of the results. The program also allows the use of user-specified routines to extend the functions of nonlinear interactions, system inputs or system outputs in prepared user-specified routines.

For large models or computer time expensive sessions the program is working in batch mode, too. MEDYNA is equipped with an automatic batch data generating facility using interactively defined input data from previous program runs.

It is worth noting, that the program itself has no limitations for the model size.

MEDYNA <i>MULTIBODY DYNAMICS</i>				
I. SYSTEM DEFINITION	II. GENERATION OF EQUATIONS	III. ANALYSIS OF STATICS	IV. ANALYSIS OF DYNAMICS	V. PRE- AND POSTPROCESSING
<ul style="list-style-type: none"> * Model Identifier * Multibody Configuration * Interconnections * Gyrostats * Motion of Reference Frame * Excitations * System Outputs 	<ul style="list-style-type: none"> * System Matrices * Time Dependent and Nonlinear Terms * Output Matrices 	<ul style="list-style-type: none"> * Nominal Interaction Forces * Static Equilibrium 	<ul style="list-style-type: none"> * Eigenvalues Eigenvectors * Frequency Response (Graphics) * Power Spectral Density (Graphics) * Covariance Analysis * Numerical Integration * Evaluation of Time Histories (Graphics) 	<ul style="list-style-type: none"> * Wheel-Rail Contact Geometry — RSGEO — * Transformation of FEM-Data — ASKESE —

Figure 4: Menu with the Options of the Program MEDYNA

Pre- and Postprocessing

ASKESE The preprocessor ASKESE has been developed for describing flexible bodies with data from finite element models. This processor presently transforms the system matrices from an FEM program into the whole set of flexible body data as required by MEDYNA. ASKESE is coupled with the finite element programs PERMAS, [20]. Other finite element programs can be used to develop the model, as long as the user converts the data to a form treatable by the preprocessor ASKESE.

RSGEO The preprocessor RSGEO generates geometry data for the wheel-rail modeling facilities in MEDYNA. The data for standard wheel and rail profile curvatures may be established, as well as the calculation of contact geometry functions dependent on the lateral wheelset displacement. These functions are necessary for the wheel-rail coupling elements, and the wheel-rail substructure. A special user routine makes it possible to input measured profile data.

Plotting The plot file provides the user with access to MEDYNA results in the time or frequency domain. Displaying the results can be done with the user's own hardware dependent graphics software.

6 Applications of the Program MEDYNA

An essential feature of MEDYNA was its motivation by the German research and development program for advanced high speed ground transportation systems. Main specifications for the modeling capabilities and tests for applicability of the program have resulted from

these efforts and have laid the basis for MEDYNA's distribution in railroad companies and organizations within Europe and abroad.

Many applications have been reported, especially in the *Journal of Vehicle System Dynamics*⁵; thus, the discussion can be kept short in the present article.

Rail Vehicles

A main field of application of MEDYNA is the design and analysis of wheel-rail vehicles. For simulation and system analysis purposes the contact of a wheel(set) on a rail can be described very effectively by the linear and nonlinear whee-rail coupling elements in the coupling element library of MEDYNA or, alternatively, the wheel-rail substructure element [4].

The modeling assumptions of MEDYNA, specially the guided reference frame are specially designed for this range of applications.

MEDYNA has been used to discuss the stability of several prototype vehicles of the german high speed train ICE, when running through a straight track. Also the dynamical and quasistatic curving behaviour of different wheel-rail vehicles have been investigated with MEDYNA by several authors [10,12].

For the numerical methods in MEDYNA the shunting of low speed freight cars is a challenging application, as with decreasing speed of the vehicle the stiffness of the differential equation (3.12) increases [7]. For a shunting train also a detailed model of the buffers including nonlinear friction elements is necessary. This application is presented in [9].

Magnetically Levitated Vehicles

As for wheel-rail systems the motion of the vehicles are in general small with respect to the guideway; thus, the essential modeling assumptions of MEDYNA apply to magnetically levitated vehicles (MAGLEV vehicles) as well. A summary of the applications to the MAGLEV system is reported in [14].

Due to the dynamics of the special coupling elements the system dimension of these systems is normally very high. In [13] the design of MAGLEV vehicles under track irregularities and external excitations due to wind gusts has been investigated using MEDYNA.

Road Vehicles

Many applications for analyzing road vehicle dynamics and vibrations can be performed by using a geometrically linear multibody program like MEDYNA. Especially the dynamical behaviour of road vehicles running over deterministic or stochastic road irregularities can be analyzed with the time- and frequency - domain methods of MEDYNA, [3]. Also, by using the static modules in MEDYNA the kinematical nonlinearity of a car axle can be investigated under load or when steered, [5].

⁵Special issues for the Proceedings of the IAVSD-Symposia on the Dynamics of Vehicles on Roads and on Tracks

Machinery

MEDYNA has also been applied for investigating vibration effects in machinery [11].

7 Mechanism Example

The dynamic problem of the mechanism given in this handbook requires a program, which provides full nonlinear kinematics. This can be seen, when considering the steering angle $\beta(t)$, which is an independent coordinate of the MBS with values larger than $\pm 0.2\text{rad}$.

The modeling assumptions in MEDYNA allow the computation of static equilibrium positions and kinematic compatible forces for different input angles β .

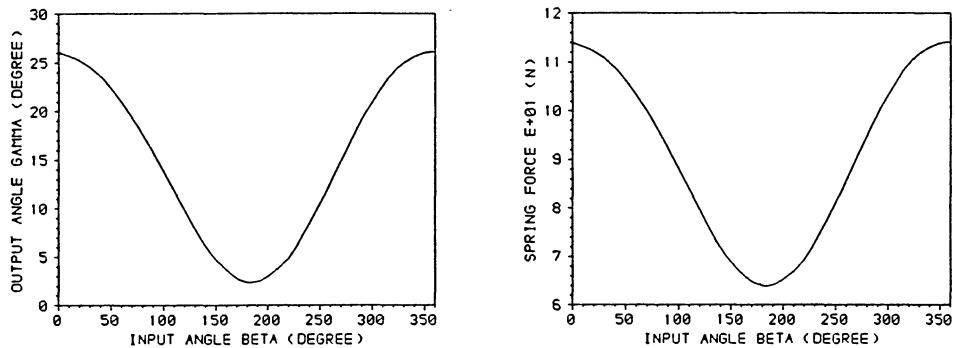


Figure 5: Kinematic Relation $\gamma(\beta)$ and corresponding Spring Force $F(\beta)$

Here the results of the output angles and the corresponding interaction force of the spring $F(\beta)$ are shown in their dependency on β , see Fig. 5.

In addition to that analysis, the dynamic part of MEDYNA allows a kinematic excited drive with a known function of the angle $\beta(t)$. The displacements of the hinge point between the rod and the drive are $\eta_x(t) = r \sin(\beta(t))$, and $\eta_y(t) = r \cos(\beta(t))$. Under these conditions the mechanisms has no degrees of freedom.

Considering $\beta(t)$ as a given kinematical excitation the angle $\gamma(t)$ has been computed with MEDYNA by using a constant Jacobian J_y , evaluated for the values $\eta_x = \eta_y = 0$. The result is shown in Fig. 6 in comparison with the angle obtained by considering nonlinear kinematics (NEWEUL). For this computation the input values $\beta(t)$ were taken from a previous simulation of this mechanism using NEWEUL.

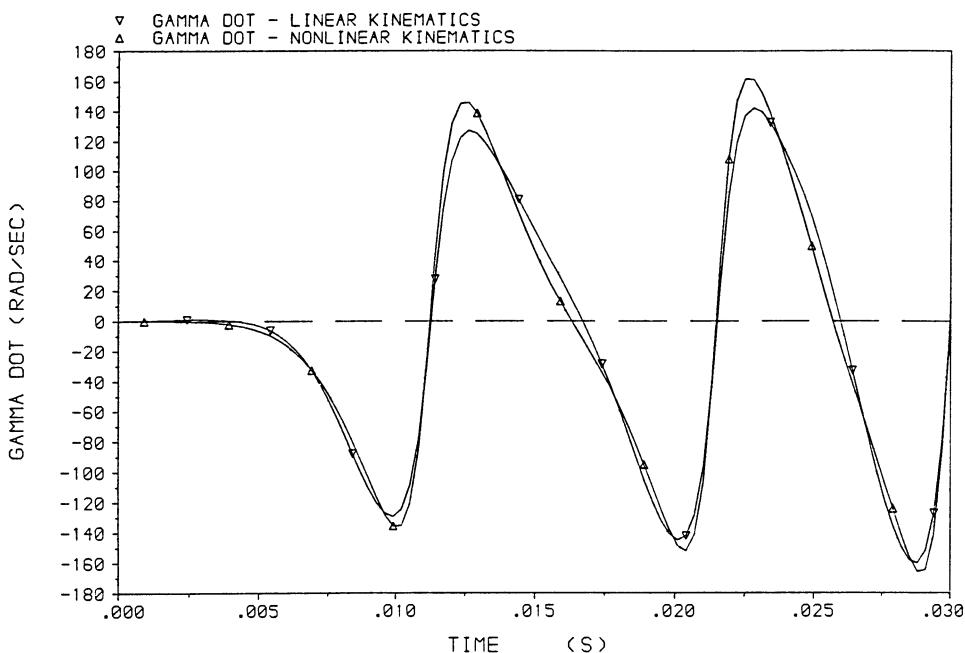


Figure 6: Angle $\gamma(t)$ resulting from linear and nonlinear kinematics

Acknowledgements

MEDYNA was generated mainly by the Multibody Dynamics Group at the Institute for Flight System Dynamics of DLR (=German Aerospace Research Establishment) under the direction of Dr. W. Kortüm. The developing group consisted of W. Duffek, W. Schuster, and W. Schwartz, together with the authors of this article; contributions were made by Dr. Mauer (MAN Technology) and Dr. Kik (Technical University Berlin).

The development of MEDYNA was financially supported by the German Ministry of Research and Technology (BMFT) and coordinated by the German Railroad Consulting (DEC). All contributions by individuals and institutions are highly appreciated.

The authors want also to thank Dr. Kortüm for many helpful comments on this article.

References

- [1] K. Brenan, S. Campbell, and L. Petzold. *The Numerical Solution of Initial Value Problems in Ordinary Differential-Algebraic Equations*. North Holland Publishing Co., 1989.
- [2] R. Clough and J. Penzien. *Dynamics of Structures*. Mc Graw Hill Kogakusha, Tokyo, 1975.
- [3] W. Duffek, C. Führer, W. Schwartz, and O. Wallrapp. Analysis and simulation of rail and road vehicles with the program MEDYNA. In O. Nordström, editor, *Proc. 9th IAVSD-Symposium on the Dynamics of Vehicles on Roads and on Tracks*, pages 71 – 85, Swets & Zeitlinger, B.V. Lisse, 1986.
- [4] W. Duffek and A. Jaschinski. Efficient implementation of wheel-rail contact mechanics in dynamic curving. In A. Wickens, editor, *Proc. 7th IAVSD-Symposium on the Dynamics of Vehicles on Roads and on Tracks*, pages 441 – 454, Swets & Zeitlinger, B.V. Lisse, 1982.
- [5] I. Faye. *Simulation of a Five-Point Suspension System for an Automobile Wheel with MEDYNA*. Technical Report DFVLR-IB 515-88-3, Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt (DFVLR), D-5000 Köln 90, 1988.
- [6] C. Führer. Algebraic methods in vehicle dynamics simulation. In A. de Pater and H. Pacejka, editors, *Proc. 3rd Seminar on Advanced Vehicle System Dynamics*, pages 329 – 346, Swets & Zeitlinger, B.V. Lisse, 1987.
- [7] C. Führer. Stiff differential equations in vehicle dynamics. In H. Neunzert, editor, *Proc. 2nd Workshop on Road Vehicles and Related Mathematics*, pages 131–141, B.G. Teubner, Stuttgart; Kluwer Academic Publishers, 1989.
- [8] C. Führer and O. Wallrapp. A computer-oriented method for reducing linearized multibody equations by incorporating constraints. *Comp. Meth. Apl. Mech. Eng.*, 46:169 – 175, 1984.
- [9] H. Horn, A. Jaschinski, and S. Sedelmair. Dynamic simulation of freight cars with nonlinear suspension and buffer models during curving. In M. Apetaur, editor, *Proc. 10th IAVSD-Symposium on the Dynamics of Vehicles on Roads and on Tracks*, pages 161 – 168, Swets & Zeitlinger, B.V. Lisse, 1988.
- [10] A. Jaschinski and W. Duffek. Evaluation of bogie models with respect to dynamic curving performance of rail vehicles. In K. Hedrick, editor, *Proc. 8th IAVSD-Symposium on the Dynamics of Vehicles on Roads and on Tracks*, pages 266 – 279, Swets & Zeitlinger, B.V. Lisse, 1984.
- [11] D. Karius and W. Kortüm. A contribution on calculation vibrations in coupled rigid and elastic multibody systems applied to a propulsion system - an application of the computer program MEDYNA. In M. Heller, editor, *Proc. 1th Intercont. Maritime Simul. Symp. and Math. Modelling Workshop*, Control Data GmbH, München, 1985.

- [12] W. Kik and H. Steinborn. Wheel/rail connexion-element for use in a multi-body-algorithm. In K. Hedrick, editor, *Proc. 8th IAVSD-Symposium on the Dynamics of Vehicles on Roads and on Tracks*, pages 303 – 316, Swets & Zeitlinger, B.V. Lisse, 1984.
- [13] W. Kortüm, W. Schwartz, and I. Faye. Dynamic modeling of high speed ground transportation vehicles for control design and performance evaluation. In G. Schweitzer and M. Mansour, editors, *IUTAM/IFAC Symposium Zurich/ Switzerland 1988: Dynamics of Controlled Mechanical Systems*, pages 335 –349, Springer, Berlin, 1988.
- [14] W. Kortüm, W. Schwartz, and I. Faye. Modeling, control design, and performance evaluation of high speed ground vehicle dynamics. *Mechanics of Structures and Machines*, to appear, 1989.
- [15] L. Mauer. *Die modulare Beschreibung des Rad/Schiene-Kontakts im linearen Mehrkörperformalismus*. Dissertation, Fachbereich 12 "Verkehrswesen", D83, Technische Universität Berlin, 1988.
- [16] P. Müller, K. Popp, and W. Schiehlen. Berechnungsverfahren für stochastische Fahrzeugschwingungen. *Ing. Archiv*, 49:235–254, 1980.
- [17] J. Ortega and W. Rheinboldt. *Iterative Solutions of Nonlinear Equations in Several Variables*. Academic Press, New York, 1970.
- [18] L. Petzold. *Automatic Selection of Methods for Solving Stiff and Nonstiff Systems of Ordinary Differential Equations*. Technical Report SAND80-8230, Sandia National Laboratories, Livermore, 1982.
- [19] W. Schiehlen. *Technische Dynamik*. B.G. Teubner, Stuttgart, 1986.
- [20] E. Schrem. *PERMAS Theory Manual*. INTES Publication, No. 302, Stuttgart, 1987.
- [21] W. Schuster and O. Wallrapp. DV-Konzept eines interaktiven integrierten Programmes zur Simulation mechanischer Systeme. In M. Goller, editor, *Proc. 1. Symposium Simulationstechnik, Informatik Fachberichte No. 56*, pages 465–474, Springer, Berlin, 1982.
- [22] L. Shampine and M. Gordon. *Computer Solution of Ordinary Differential Equations*. Freeman, San Francisco, 1975.
- [23] O. Wallrapp. *Entwicklung rechnergestützter Methoden der Mehrkörperdynamik in der Fahrzeugtechnik (Dissertation)*. Technical Report DLR-FB 89-17, Deutsche Forschungsanstalt für Luft- und Raumfahrt (DLR), D-5000 Köln 90, 1989.

AUTODYN & ROBOTRAN - Computer Programmes

P. Maes, J.Cl. Samin and P.Y. Willems
Université Catholique de Louvain
Louvain-la-Neuve, Belgium.

AUTODYN

Abstract

This contribution presents the main features of the AUTODYN programme. This programme based on d'Alembert Potential Power Principle, permits to derive the equations of motion of any mechanical system which can be represented by a set of interconnected rigid bodies. In particular, it has important applications in the fields of robotics and vehicle dynamics. The variables of the system are the generalized variables describing the relative motion of the various joints of the system. A joints' library including surface rolling interconnections (rail-wheel joint) is available. Constraints can be considered and in particular those resulting from loops of bodies are automatically generated. The Lagrange multipliers technique permits to derive the complete set of equations of motion; a system reduction via the elimination of these multipliers and a coordinate partitioning method is available. The obtained programme can be used as a sub-routine for any desired application such as numerical integration, stability analysis, control design, numerical linearization, eigenvalues determination.

Introduction

The mechanical system under consideration is assumed to be represented by a multibody system composed of interconnected rigid bodies. These bodies will be characterized by their mass distribution parameters and by geometrical quantities which permit to localize their various interconnections. These connections are defined as *joints* and may have up to six degrees of freedom.

Joints can be represented by mechanical devices such as springs, dampers, hinges, universal joints or have more sophisticated representations such as sliding or rolling between surfaces. Relative motion can take place in the joints and interactions are produced. The variables describing the relative motions in the joints will be used as generalized variables (even if other variables can be used as outputs of the programme). The joint forces and torques can either be unknown constraint quantities or can be described by dynamical relations (describing functions) between the various joint variables.

A library of predefined joints with their variables and describing functions is available; the user is only asked to specify the actual values of the corresponding parameters.

Graph Description

The kinematical description of the system is greatly simplified when the *graph* of this system is defined [1]. A tree structure is always used; for systems with loops of bodies, a cutting procedure is determined and an augmented tree structure is defined.

In order to simplify the introduction of the system data, conventions on the numbering of bodies and joints and on the localization of loop cuts are introduced. For this procedure, graph matrices have been abandoned in favour of the more readily usable notion of *filiation* [2][3]. This notion simply expresses that a given joint connects two neighbouring bodies.

First of all a *reference body* is defined and the index 0 is associated with this body which is assumed to be fixed with respect to inertial space and is the common ancestor of all the bodies of the system.

For chain and tree structures, the graph numbering is then very simple. All the bodies (considered as descendants) will be numbered according to a filiation order in the chain or in the tree; the joints are then given the number of the body they precede in this order. It is clear that for tree structures, this procedure favours certain branches of the system.

If the same procedure is applied to systems which include loops of bodies, it will soon appear that a given body is the direct ancestor of two (or more) bodies which have common descendants (see figure 1). This indicates that this body is at the *origin* of a loop. Conventionally, this body will then be separated in two parts, the original body which keeps all its physical properties and a fictitious loop *closure body* which only retains kinematical properties and permits to define an *augmented tree* structure. Once the cutting procedure is completed M_ℓ independent loops have been identified and M_ℓ closure bodies have been added as terminal bodies of the corresponding augmented tree.

It can be of interest to define a *main body* of the system. The index 1 is then as-

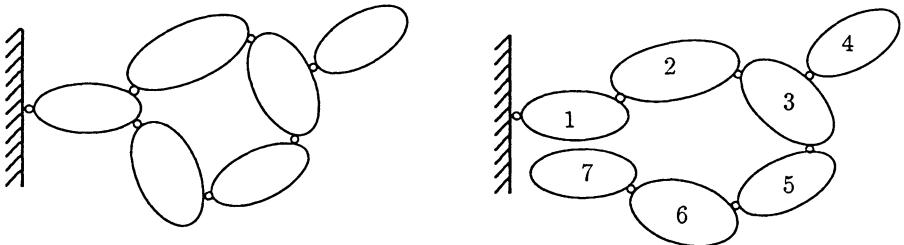


Figure 1: Loop and augmented tree

sociated to this body. The corresponding joint 1 (which connects the main body to the reference body) may have a fully or partially prescribed motion; from this simple consideration, the application of the programme to various situations is greatly facilitated and, further, any discussion on the use of fixed or moving reference frame becomes irrelevant.

The filiation and cutting procedure can be supplied to the main programme through the use of two matrix functions: *INBODY* and *INLOOP*.

The function *INBODY* is a vector which has as many elements as there are joints in the system or equivalently as there are descendants of the fixed reference body in the augmented tree. The j th element of this vector corresponds to the j th body (or equivalently to the j th joint) and includes the index of the direct parent of this j th body ($INBODY(j) = i$ thus indicates that body i is the direct parent of body j in the augmented tree). The recursive use of the function *INBODY* permits to obtain the indices of all the bodies which belong to the unique path between a given body and the reference body; the notation $i \leq j$ indicates that body i belongs to this path defined for body j , if further $j \neq i$, the notation $i < j$ is used.

The columns of the $(2 \times M_\ell)$ matrix *INLOOP* provides the information on the origin and the closure of the various loops. The α column of this matrix corresponds to the loop α and its two elements include the indices of the origin and of the closure bodies of this loop respectively. Further, the path in the loop α is provided by the recursive use of the function *INBODY* initiated at $j = INLOOP(2, \alpha)$ and interrupted when $INBODY(i) = INLOOP(1, \alpha)$.

Body Parameters

The equations of motion depend on the geometrical configuration and the mass distribution parameters of the various bodies.

The characteristic *geometrical lengths* are described by the vecteur $\underline{\ell}_{ij}$ defined as (see figure 2a) :

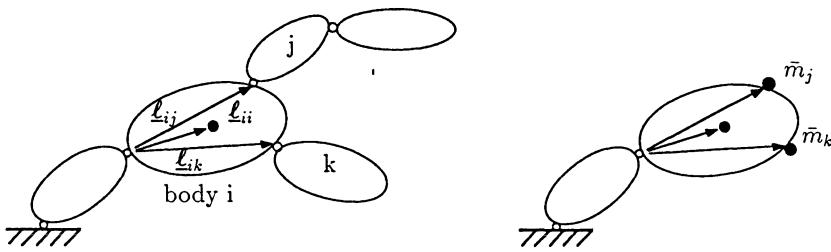


Figure 2: a) Geometrical parameters

b) Augmented body

if $i = INBODY(j)$, the vector $\underline{\ell}_{ij}$ is the vector connecting the reference points of joints i and j respectively;

if $i < j \leq k$, the vector $\underline{\ell}_{ik}$ is defined to be equal to the corresponding vector $\underline{\ell}_{ij}$;

if $j = i$, the corresponding vector $\underline{\ell}_{ii}$ is the position vector of the mass centre of body i with respect to joint i (for fictitious closure bodies this vector locates the point corresponding to the mass centre of the original body);

otherwise $\underline{\ell}_{ij} = 0$.

The bodies being supposed rigid, their mass distribution is fully described by the masses m_i , the above defined position vectors of the mass centres, $\underline{\ell}_{ii}$, and the tensors of inertia with respect to this mass centre, I_i . Further the vectors $\underline{\ell}_{ij}$, $\underline{\ell}_{ii}$ and the tensors I_i being defined for the body i have constant components in a base $\{\hat{x}_i\}$ attached to

this body. The bases attached to fictitious closure bodies are supposed to coincide with the bases of the corresponding origin bodies when the loops are closed; it is clear that these bodies have neither mass nor inertia tensor.

It can be shown – see [4] and [5], – that these body parameters combine in the expressions of the equations of motion which are given in terms of so-called barycentric parameters. The *barycentric tensors* appearing in the present formulation are given by the following expressions:

$$\bar{m}_i = \sum_{k:i \leq k} m_k \quad (1)$$

$$\bar{m}\underline{\mathbf{b}}_i = \sum_{k:i \leq k} m_k \underline{\ell}_{ik} \quad (2)$$

$$\underline{\mathbf{K}}_i = \underline{\mathbf{I}}_i - \sum_{k:i \leq k} m_k \tilde{\underline{\ell}}_{ik} \tilde{\underline{\ell}}_{ik} \quad (3)$$

where $\tilde{\underline{\mathbf{x}}}$ is the skew-symmetrical tensor associated with the vector $\underline{\mathbf{x}}$.

If one defines the *augmented body* i (figure 2b) as a body consisting of the body i and point masses equal to \bar{m}_j located at the reference point of the various joints j such that $INBODY(j) = i$, the above-defined barycentric parameters respectively represent the mass of the augmented body i and its first and second order tensors of inertia with respect to the reference point of joint i ; in particular $\underline{\mathbf{b}}_i$ represents the position vector of the mass centre of the augmented body with respect to this point.

Joint Description

As already seen, the joint j connects body j to body $i = INBODY(j)$. In both bodies, the corresponding joint reference or end points have already been defined and localized. Joint bases can also be associated with both ends of the joint; the base $\{\hat{\underline{\mathbf{x}}}^{\text{in}}_j\}$ is associated with body i and the base $\{\hat{\underline{\mathbf{x}}}^{\text{out}}_j\}$ with the body j . For practical reasons these bases may differ from the corresponding body bases but remain considered as body fixed; their relative orientation can be described by constant orthogonal matrices $[A_j^{\text{in}}]$ and $[A_j^{\text{out}}]$ with

$$\begin{aligned} [\hat{\underline{\mathbf{x}}}^{\text{in}}_j] &= [A_j^{\text{in}}] [\hat{\underline{\mathbf{x}}}_i] \\ \text{and} \quad [\hat{\underline{\mathbf{x}}}_j] &= [A_j^{\text{out}}] [\hat{\underline{\mathbf{x}}}^{\text{out}}_j] \end{aligned} \quad (4)$$

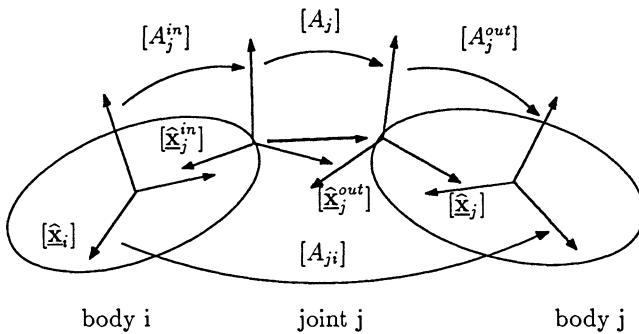


Figure 3: Joints

During the motion, the joint reference points may separate and the joint bases may take a relative orientation. This relative motion is described by the translation vector \underline{z}_j (oriented from body i to body j and expressed in the base $\{\hat{x}_j\}$ as $\underline{z}_j = [\hat{x}_j]^T [z_j]$) and the orientation matrix $[A_j^*]$ with

$$[\hat{x}_j^{out}] = [A_j^*] [\hat{x}_j^{in}] \quad (5)$$

The relative orientation of the base $\{\hat{x}_j\}$ with respect to the base $\{\hat{x}_i\}$ with $i = INBODY(j)$ is then given by matrix $[A_{ji}]$ with, from (4) and (5)

$$[A_{ji}] = [A_j^{out}] [A_j^*] [A_j^{in}]$$

The components of matrices $[z_j]$ and $[A_{ji}]$ need to be described in terms of the joint variables considered as the generalized coordinates $\{q_{j\alpha}\}$ and joint parameters with, in most cases, $[z_j] = 0$ and $[A_j^*] = [E]$ (where $[E]$ is the identity matrix) in the reference configuration. Further some constraints (holonomic or not) between these coordinates may be considered.

On the other hand, the joint mechanism produces interactions between the connected bodies. The corresponding forces and torques either can be constraint quantities (which usually can be ignored in the final equations) or can be described by dynamical describing functions.

In principle, in order to properly describe a joint, the user must provide the expressions of the matrices $[z_j]$ and $[A_{ji}]$, of the constraints (the first and second time derivatives of these expressions are also required as will be seen further) and of the interactions acting on the bodies.

This procedure is simplified by the use of a joint library. Six standard joints are currently included in the AUTODYN package; their parameters may be interactively adapted to the user application. These joints have been devised to cover a wide range of applications; they are given by the following sub-routines :

ROTRA : sequence of three Tait-Bryan rotations around (3,2,1)-axes followed by three translations;

TRARO : sequence of three translations followed by three rotations around (1,2,3)-axes;

AXE : rotation around one axis (1,2 or 3);

GLIS : translation along one axis (1,2 or 3);

ROUERAIL : motion of a surface of revolution on a cylinder;

RORACOUR : motion of a surface of revolution on a torus.

In the sub-routines ROTRA and TRARO, the variables corresponding to the degrees of freedom may be given as prescribed functions of time or constrained or suppressed.

Joints AXE and GLISS are simplified versions of the preceding ones. Together with the addition of intermediate massless bodies, they permit the composition of any sequence of degrees of freedom.

Joints ROUERAIL and RORACOUR were specifically developed for railway vehicles applications and in particular for system with loose wheels and articulated bogies [6].

In order to illustrate the joint concept, the ROUERAIL joint will now be described

An isolated wheel moving on its rail has five degrees of freedom, which can in principle be described by two horizontal displacements of its centre of mass, G, the rotation

angle around the axis of symmetry and the yaw and pitch angles. The rail and the wheel being profiled, the choice of these coordinates implies the solution of the non linear geometrical constraint corresponding to the contact with the rail.

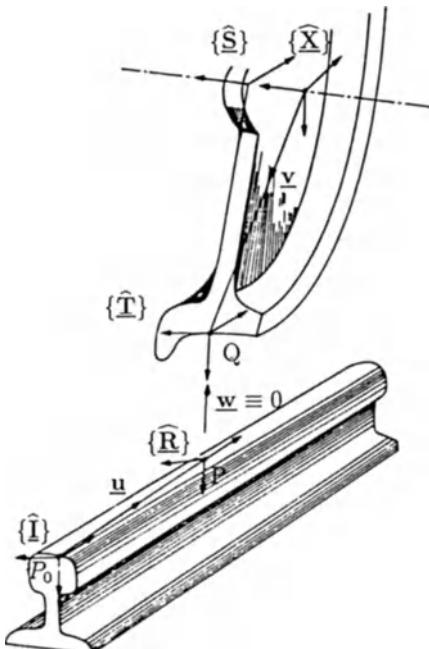


Figure 4: Wheel-rail joint

This relation is explicit under the form $u_3 = \mu(u_1)$. An auxiliary reference base $\{\hat{R}\}$, with the 3-axis aligned with the normal to the rail at the contact point, is associated with this contact point; its orientation with respect to the fixed base, $\{\hat{I}\}$, is obtained by a rotation α around the 2-axis with $\alpha = -\tan^{-1}\mu'(u_1)$ where μ' is the derivative of the profile function.

Similar functions are used to localize the contact point, Q , on the moving wheel surface. The base $\{\hat{X}\}$ is fixed to the wheel and the wheel reference point is its center of mass (supposed to belong to the axis of symmetry). The meridian plane to which Q pertains is chosen to be the (1,3) plane of the auxiliary base $\{\hat{S}\}$; the orientation of this base with respect to the wheel fixed base is described by the rotation angle θ (around the axis of symmetry). The position vector of the contact point Q with respect to the wheel mass center is given by the vector \underline{v} ; by definition the second component of this

In ROUERAIL, we introduced a set of coordinates which explicitly refers to the position of this contact point. The contact on the rail, P , is described by its position vector, \underline{u} , with respect to a rail fixed reference point, P_0 . The component u_2 of this vector expressed in rail fixed base $\{\hat{I}\}$ describes the longitudinal displacement along the track, the lateral, u_1 , and vertical, u_3 , components are related by a rail profile function which is described in the (1,3) plane (for perfectly linear tracks).

vector is equal to zero and, further, the other two components are related by the wheel profile function. Here too, this function is explicitated as $v_3 = \rho(v_1)$ and a wheel profile tangent base $\{\hat{\mathbf{T}}\}$ is obtained by a rotation β around the 2-axis, with $\beta = \tan^{-1} \rho'(v_1)$. The contact conditions imply that the points P and Q coincide as well as the normal to the two surfaces at these points; the 3-axes of the wheel and rail profile bases are then aligned – these bases are permitted to spin with a relative angle ϕ around this common axis.

The components of the displacement vector and the relative rotation matrix of the rail-wheel joint are easily expressed in terms of the generalized variables u_1, u_2, v_1, θ and ϕ . The auxiliary variables u_3, v_3, α and β are explicitly given and the geometry of the problem is thus completely described by the generalized variables.

The interaction describing functions used in *ROUERAIL* are based on the linear Kalker creep model [6],[7]; they require the knowledge of the contact normal force. In order to be able to estimate this interaction, we use an additional degree of freedom - the relative displacement of the contact points along the contact normal - and a posteriori constraint this displacement to vanish; the associated Lagrange multiplier has been shown to be equal to the normal force [8].

Kinematics : relative and absolute motion, constraints

The configuration of a rigid body is completely determined by the position of its mass centre and the orientation of its body-fixed base with respect to the reference body (with respect to inertial space).

The (absolute) position vector of body j, \underline{x}_j , can be expressed in terms of the vectors \underline{e}_{ij} and \underline{z}_i encountered along the path between the reference body and the body under consideration as :

$$\underline{x}_j = \sum_{k:k \leq j} (\underline{z}_k + \underline{e}_{kj})$$

The orientation of the base $\{\hat{\mathbf{x}}_j\}$ with respect to inertial space, given by the matrix $[A_j]$ – with $[\hat{\mathbf{x}}_j] = [A_j][\hat{\mathbf{I}}]$, is then obtained via the product of the relative orientation

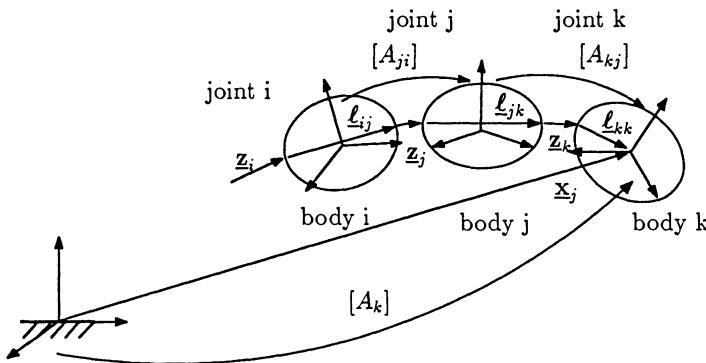


Figure 5: Kinematics

matrices encountered along the path between body j and the reference body 0, with $[\hat{x}_0] \equiv [\hat{I}]$.

The components (in the base $\{\hat{x}_j\}$) of relative angular velocity in the joint j , $\underline{\Omega}_j$ are obtained by the matrix relation $[\dot{A}_{ji}] = -[\widetilde{\Omega}][A_{ji}]$ and are linear functions of the time derivatives of the joint generalized coordinates. The absolute angular velocity of the base fixed to body j , $\underline{\omega}_j$, is then simply the (vector) sum of the relative angular velocities in the joints on the path from the reference body:

$$\underline{\omega}_j = \sum_{k:k \leq j} \underline{\Omega}_k$$

The expressions of the mass centre velocities and accelerations and of the body bases angular accelerations are then obtained in a straightforward manner.

The loop constraints are obtained by expressing the fact that for all the loops the origin and closure (fictitious) bodies coincide i.e. have the same mass centre and coinciding bases; these bases have the same angular velocities. We can write these relations as :

$$\begin{aligned} \underline{x}_\beta &= \underline{x}_\gamma \\ \dot{\underline{x}}_\beta &= \dot{\underline{x}}_\gamma \\ [A_\beta] &= [A_\gamma] \\ \underline{\omega}_\beta &= \underline{\omega}_\gamma \end{aligned} \tag{6}$$

$$\text{where } \begin{cases} \beta = \text{INLOOP}(1, \alpha) \\ \gamma = \text{INLOOP}(2, \alpha) \end{cases} \quad \alpha = 1, \dots, M_\ell.$$

The orientation constraints are not independant and we choose to retain their (1,2), (2,3) and (3,1) elements; the other relations must be checked a posteriori.

The first and second time derivatives of the position constraints are obtained in a straightforward manner. Instead of the time derivatives of the orientation constraints, we use the corresponding angular velocity constraints and their time derivatives. Further, the angular velocity constraints provide an appropriate pseudo-gradient for the retained orientation constraints [9]; the use of this pseudo-gradient also ensures the fulfilment of the complete set of orientation constraints — the third relation (6).

Once the M_{joint} joint constraints and M_{user} users' constraints have been added, we obtain a set of M constraints. If the constraints are holonomic, they are represented by M (independant) algebraic relations between the generalized variables of the problem q_α represented by the $(N \times 1)$ vector $[q]$. We write them under the form of a M vector relation

$$[S(q_\alpha, t)] = 0 \quad (7)$$

The time derivative of these constraints (or possibly equivalent pseudo-constraints which are linear combinations of these times derivatives) can be written in matrix form :

$$[A(q_\alpha, t)] [\ddot{q}] + [b(q_\alpha, t)] = 0 \quad (8)$$

where the $(M \times N)$ matrix $[A]$ is the gradient (or pseudo-gradient) matrix of the (holonomic) constraints. For loop constraints, this gradient is obtained from the second and forth relations in (6).

We assume that non-holomic constraints (arising from the joints — due to rolling without sliding for instance — or directly introduced by the user) are linear in the velocities (and then also in the generalized velocities). They then combine with the time derivatives of the remaining holonomic constraints to provide a system of the form (8); we also assume, possibly after elimination of redundant constraints, that the corresponding matrix $[A]$ is a full rank matrix.

The time derivative of this matrix equation (8) is also needed and can be written in the form

$$[A(q_\alpha, t)] [\ddot{q}] + [c(q_\alpha, \dot{q}_\alpha, t)] = 0 \quad (9)$$

Equations of motion

The equations of motion of the augmented tree will be obtained from the Potential Power Principle.

The local form of the equations of motion for any system (without local internal angular momentum distribution) is given by the relation

$$\ddot{\underline{x}} = \underline{f}$$

i.e. the acceleration of any point of the system $\ddot{\underline{x}}$ is equal everywhere (except possibly on set of points with zero mass measure) to the corresponding value of the local force density \underline{f} .

For any vector field defined by the vectors \underline{a} , we obtain the following scalar identity after integration over the system :

$$\int (\ddot{\underline{x}} - \underline{f}) \cdot \underline{a} dm = 0$$

We can choose the vectors \underline{a} to be a potential (possible) change of velocity compatible with the rigidity of the bodies.

For any point of the rigid body j , the velocity is given by

$$\dot{\underline{x}} = \dot{\underline{x}}_j + \underline{\omega}_j \times \underline{x}$$

The potential velocity (compatible with possible changes of the mass centre velocity, $\Delta\dot{\underline{x}}_j$, and angular velocity $\Delta\underline{\omega}_j$) is given by :

$$\underline{\Delta\dot{x}} = \underline{\Delta\dot{x}}_j + \underline{\Delta\omega}_j \times \underline{x}$$

and, after integration, the Potential Power Principle for a system of rigid bodies reads :

$$\sum_j [(\dot{\underline{N}}_j - \underline{F}_j) \cdot \underline{\Delta\dot{x}}_j + (\dot{\underline{H}}_j - \underline{L}_j) \cdot \underline{\Delta\omega}_j] = 0 \quad (10)$$

where

$\underline{N}_j = m_j \dot{\underline{x}}_j$ is the linear momentum vector of body j ;

$\underline{\mathbf{H}}_j = \underline{\mathbf{I}}_j \cdot \underline{\omega}_j$ is the angular momentum of body j with respect to its mass centre;

$\underline{\mathbf{F}}_j$ and $\underline{\mathbf{L}}_j$ are, respectively, the resultant of forces and torques (reduced to the mass centre) applied to the body j.

Alternatively, the relation (10) can be written :

$$\sum_j [\dot{\underline{\mathbf{N}}}_j \cdot \underline{\Delta \dot{\mathbf{x}}}_j + \dot{\underline{\mathbf{H}}}_j \cdot \underline{\Delta \omega}_j] = \Delta P \quad (11)$$

where ΔP is now the potential power of all the interactions acting on and in the system and can be decomposed into ΔP^{ext} , the potential power of the external interactions acting on the system and the sum of ΔP_j , the potential power of the interactions acting in the various joints i.e. :

$$\Delta P = \Delta P^{ext} + \sum_j \Delta P_j$$

The velocities of the mass centre and the angular velocities (and hence of all the material points of the system) being linear in the generalized velocities \dot{q}_α , all the potential velocities will be linear functions of the potential changes of these generalized velocities $\Delta \dot{q}_\alpha$. Indeed formally, for every point, the position vector is a function of the generalized coordinates, $\underline{\mathbf{x}} = \underline{\mathbf{x}}(q_\alpha, t)$. The corresponding velocities can be written in the form :

$$\dot{\underline{\mathbf{x}}} = \sum_\alpha \frac{\partial \underline{\mathbf{x}}}{\partial q_\alpha} \dot{q}_\alpha + \frac{\partial \underline{\mathbf{x}}}{\partial t} = \dot{\underline{\mathbf{x}}}(q_\alpha, \dot{q}_\alpha, t)$$

and the potential velocities compatible with the potential change of generalized coordinates, $\underline{\Delta \dot{\mathbf{x}}} \stackrel{\text{def}}{=} \dot{\underline{\mathbf{x}}}(q_\alpha, \dot{q}_\alpha + \Delta \dot{q}_\alpha, t) - \dot{\underline{\mathbf{x}}}(q_\alpha, \dot{q}_\alpha, t)$ is given by :

$$\underline{\Delta \dot{\mathbf{x}}} = \sum_\alpha \frac{\partial \underline{\mathbf{x}}}{\partial q_\alpha} \Delta \dot{q}_\alpha.$$

In particular this permits to write the left hand side of (11) as

$$\sum_j [\dot{\underline{\mathbf{N}}}_j \cdot \underline{\Delta \dot{\mathbf{x}}}_j + \dot{\underline{\mathbf{H}}}_j \cdot \underline{\Delta \omega}_j] = \sum_\alpha \Phi_\alpha \Delta \dot{q}_\alpha$$

this expression can be written in matrix form as $[\Phi]^T [\Delta \dot{q}]$ where $[\Phi]$ is the $(N \times 1)$ vector of the various $\Phi_\alpha(q_\alpha, \dot{q}_\alpha, \ddot{q}_\alpha, t)$.

It can easily be checked that these functions are linear in the second time derivative of the generalized coordinates and we can write :

$$[\Phi] = [M][\ddot{q}] + [F] \quad (12)$$

where $[M] = [M(q_\alpha, t)]$ is the generalized mass matrix and the vector $[F] = [F(q_\alpha, \dot{q}_\alpha, t)]$ corresponds to centrifugal and gyroscopic effects.

The potential power of a joint interaction can also be written as:

$$\Delta P_j = \sum_{\alpha} Q_{j\alpha} \Delta \dot{q}_{j\alpha}$$

where $Q_{j\alpha}$ is the generalized joint interaction corresponding to the joint variable $q_{j\alpha}$.

In most cases the constraint interactions in the joint do not contribute to the joint potential power when the corresponding constraints are potentially satisfied (this point has to be carefully checked). They could nevertheless appear in the expression of the above-defined generalized forces, but it will be seen this is generally not a problem.

The potential power of the complete set of interactions will be written as :

$$\Delta P = \sum_{\alpha} Q_{\alpha} \Delta \dot{q}_{\alpha}$$

or in matrix form

$$\Delta P = [Q]^T [\Delta \dot{q}] \quad (13)$$

where $[Q]$ is the $(N \times 1)$ vector of the generalized interactions, $Q_{\alpha}(q_{\alpha}, \dot{q}_{\alpha}, t)$

The Potential Power Principle can now be written in the form :

$$\{ [\Phi] - [Q] \}^T [\Delta \dot{q}] = 0 \quad (14)$$

for the variations of potential velocities which satisfy equation (8), i.e. which are such that

$$[A(q_{\alpha}, t)] [\Delta \dot{q}] = 0. \quad (15)$$

Using the classical *Lagrange multipliers technique*, the equations of motion are finally given by the N equations

$$\begin{aligned} [\Phi] - [A]^T [\lambda] &= [Q] \\ [M] [\ddot{q}] + [F] &= [Q] + [A]^T [\lambda] \end{aligned} \quad (16)$$

where the $(M \times 1)$ vector $[\lambda]$ is the corresponding vector of the *Lagrange Multipliers*.

This set of equations has still to be completed by the constraints in the form (7), $[S] = 0$, for holonomic constraints or in the form (8), $[A][\dot{q}] + [b] = 0$ for holonomic or non holonomic cases.

When the generalized interactions depend on constraint interactions, it proves to be useful to express those in terms of Lagrange multipliers (possibly associated with additional constraints between generalized coordinates. This procedure is used for the ROUERAIL joint.

System reduction

The system of equations can be reduced [10],[11], by the elimination of the Lagrange multipliers and of a set of variables (equal to the number of holonomic constraints).

Let us consider the set of M^* independant holonomic constraints, $[S^*] = 0$ and possibly after a reordering (optionaly imposed by the user or decided by the programme), the vector of generalized coordinates is partitioned as

$$q = \begin{bmatrix} u \\ \cdots \\ v \end{bmatrix}$$

where $[v]$ is the set of the M^* variables which will be eliminated. The corresponding gradient matrix $[A^*]$ will also be partitioned as

$$[A^*] = [A_u^* : A_v^*]$$

where the submatrix $[A_v^*]$ is a regular $(M^* \times M^*)$ matrix. This matrix being regular the coordinates v_α can in principle be (locally) expressed in terms of the others, u_α , but analytical expressions are generally impossible to obtain.

In AUTODYN the following recursive algorithm is used :

$$[A_v^*][v_{(k+1)} - v_{(k)}] + [S^*] = 0,$$

$$\text{where } \begin{cases} [S^*] &= [S^*(v_{(k)}, u^*)] \\ [A_v^*] &= [A_v^*(v^*, u^*)] \end{cases}$$

with $[u^*]$ the vector of the actual values of the vector $[u]$ or any prescribed value (such as a possible equilibrium value) and $[v^*]$, the recursive value of the vector $[v]$ i.e. $[v_{(k)}]$ or any value prescribed by the user; this permits to avoid the inversion of the matrix $[A_v^*]$ at each step of the recurrence (or even at each step of an application such as a numerical integration).

The time derivatives of $[v]$ in terms of the corresponding derivatives of the independent variables are obtained in the form :

$$[\dot{v}] = [B_{vu}][\dot{u}] + [b_v]$$

$$[\ddot{v}] = [B_{vu}][\ddot{u}] + [c_v]$$

$$\text{where } \begin{cases} [B_{vu}] &= -[A_v^*]^{-1}[A_u^*] \\ [b_v] &= [A_v^*]^{-1}[b_v^*] \\ [c_v] &= [A_v^*]^{-1}[c_v^*] \end{cases}$$

The matrices $[M]$ and $[F]$ are also partitioned as

$$M = \begin{bmatrix} M_{uu} & \vdots & M_{uv} \\ \cdots & \cdots & \cdots \\ M_{vu} & \vdots & M_{vv} \end{bmatrix} \text{ and } F = \begin{bmatrix} F_u \\ \cdots \\ F_v \end{bmatrix}$$

If one also defines the matrix $[B_{uv}]$ as $[B_{uv}] \stackrel{\text{def}}{=} [B_{vu}]^T$, the reduced equations of motion are given in the form :

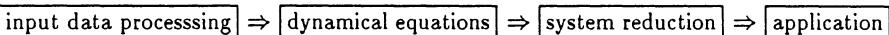
$$[\bar{M}_u][\ddot{u}] + [\bar{F}_u] = [\bar{Q}_u] \quad (17)$$

$$\text{where } \begin{cases} [\bar{M}_u] &= [M_{uu}] + [M_{uv}][B_{vu}] + [B_{uv}][M_{vv}] + [B_{uv}][M_{vv}][B_{vu}] \\ [\bar{F}_u] &= [F_u] + [B_{uv}][F_v] + \{ [M_{uv}] + [B_{uv}][M_{vv}] \}[c_v], \\ [\bar{Q}_u] &= [Q_u] + [B_{uv}][Q_v] \end{cases}$$

This set of equations must be completed by any non holonomic constraints.

Programme arrangement

An AUTODYN application is divided in four main step:



The input of the data preprocessor includes the graph information (the *INBODY* and *INLOOP* matrices), the geometrical (the \underline{L}_{ij} vectors) and mass distribution (the masses m_i and the \underline{I}_i tensors) parameters and the information relative to the joints (possibly provided by the joint library).

The output of the data preprocessor provides the barycentric parameters (1) - (3) and the generalized coordinates vector; the joint generalized interactions, $Q_{j\alpha}$, are also provided at this stage if the joint library is used.

The main programme computes the various terms appearing in the Potential Power Principle i.e. :

the matrix $[\Phi]$ under the form (12) : $[\Phi] = [M][\ddot{q}] + [F]$;

the generalized force matrix (13) : $[Q] = [Q(q_\alpha, \dot{q}_\alpha, (\lambda_\alpha), t)]$ which includes the effect of joint, extenal and control interactions;

the constraints (or equivalent pseudo constraints) and in particular (possibly after elimination of redondant constraints) the constraints gradient matrix $[A]$.

The outputs of the main programme are the dynamical equations written in the form (16) :

$$[M][\ddot{q}] + [F] - [A]^T[\lambda] = [Q]$$

and the constraints (and their time derivatives) under the form (7), (8) or (9) :

$$[S] = 0, \quad [A][\dot{q}] + [b] = 0, \text{ or } [A][\ddot{q}] + [c] = 0$$

the output consists of a system of $(N + M)$ (differential or mixed algebraic and differential) equations in the $(N + M)$ unknowns $[q]$ and $[\lambda]$.

This system can then be reduced, by the elimination of the Lagrange multipliers and of a set of variables (equal to the number of holonomic constraints). The obtained set of equations (17):

$$[\overline{M}_u][\ddot{u}] + [\overline{F}_u] = [\overline{Q}_u]$$

is completed by the non holonomic constraints to provide the final set of equations.

The organisation of the programme can be summarized as follows :

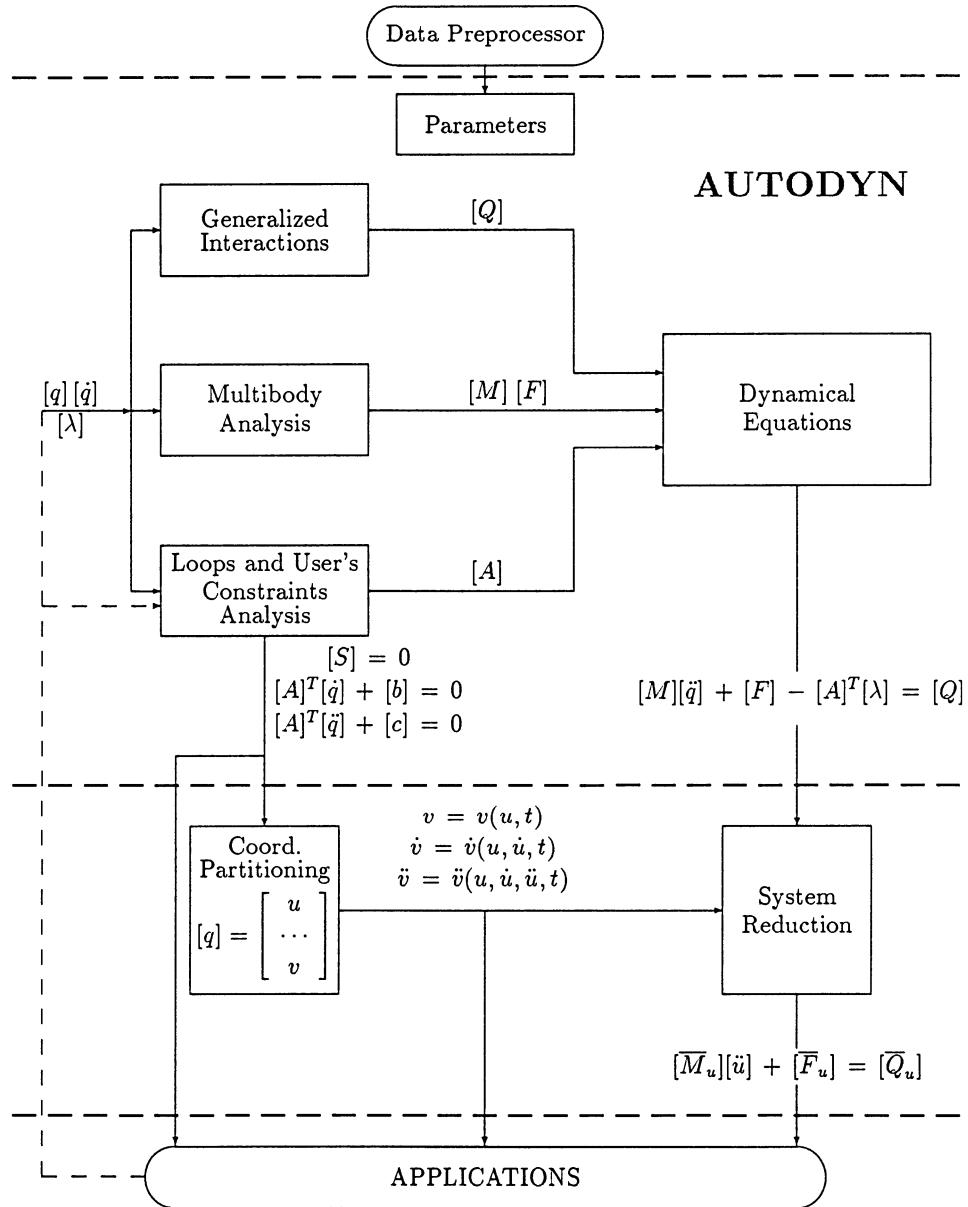


Figure 6: Programme AUTODYN

The user can now use the AUTODYN programme for his applications. The following applications are readily available: numerical integration, equilibrium investigation, numerical linearization, stability analysis, eigenvalue determination, control design.

Test example: Plane motion of a mechanism.

This system is a seven bodies mechanism. One fictitious body (body 8) has been added at the spring attachment point in order to simplify the description of the motion of this nodal point; three loop closure bodies (bodies 9,10 and 11) are also considered. Figure 7 represents the numbering used in the graph matrices. Figure 8 and 9 are typical figures obtained, from the output file, by using the MATLAB graphic software.

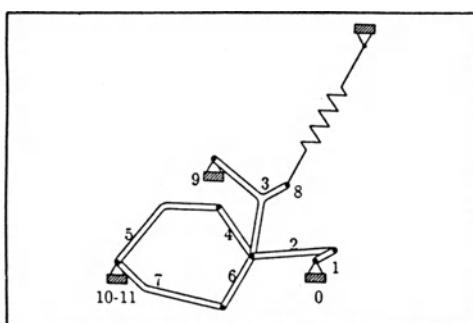


Figure 7: Body numbering.

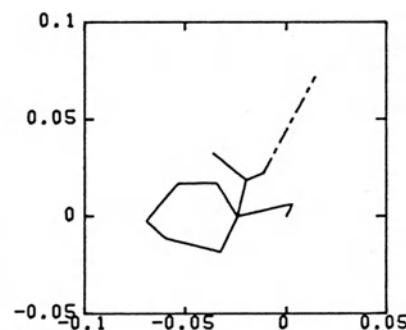


Figure 8: Configuration after 8 ms.

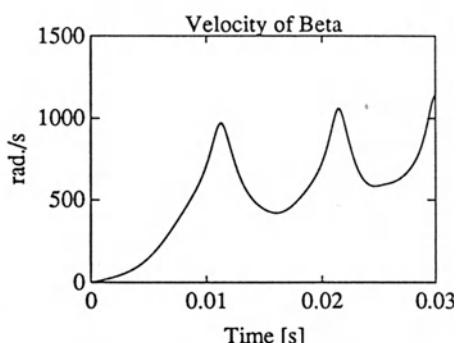
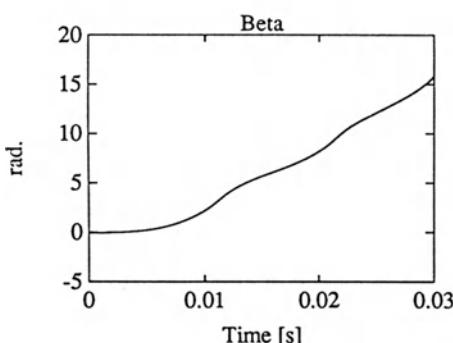


Figure 9: First body orientation angle, β and angular velocity, $\dot{\beta}$.

Figure 10 shows some features of the input data file and of the joint library.

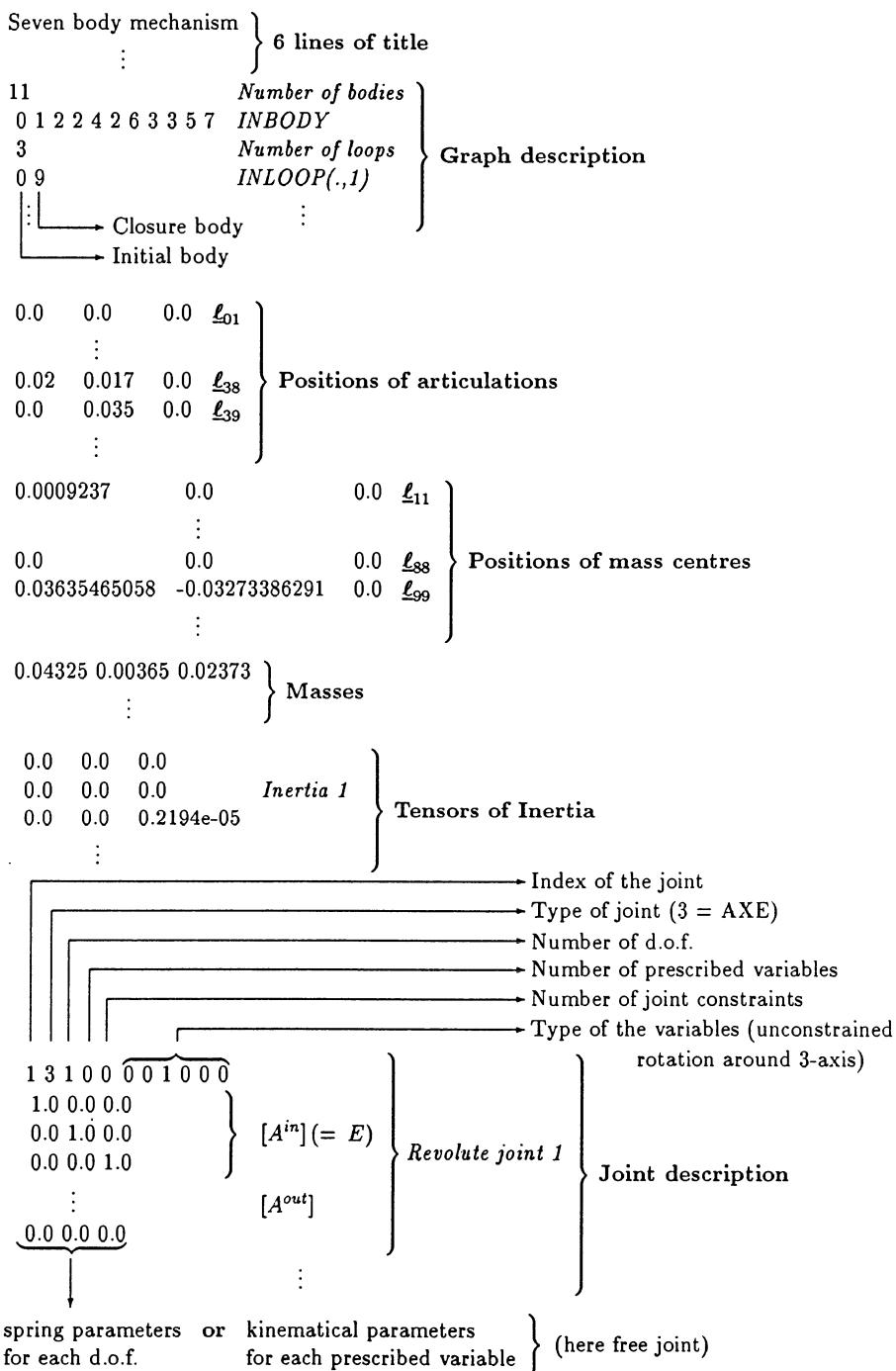


Figure 10: Input data of the "Seven bodies mechanism"

References

- [1] Roberson,R.E. and J.Wittenburg, A dynamical formalism for an arbitrary number of interconnected rigid bodies, with reference to the problem of satellite attitude control, Proceedings of the *Third International Congress of Automatic Control*, Butterworth and Co, London, 1966.
- [2] Wittenburg,J, *Dynamics of systems of rigid bodies*, B.G.Teubner, Stuttgart, 1977.
- [3] Samin,J.-Cl. and P.Y.Willems, Multibody formalism applied to non-conventional railway systems, *Dynamics of Multibody Systems*, G.Bianchi and W.Schiehlen(eds), Springer,Berlin, 1986.
- [4] Renaud,M., Quasi-minimal computation of the dynamic model of a robot manipulator utilizing the Newton-Euler formalism and the notion of augmented body, Proceeding of the *IEEE International Conference on Robotics and Automation*, Raleigh, North Carolina, 1987.
- [5] Maes,P., J.-Cl.Samin and P.Y.Willems, Linearity of multibody systems with respect to barycentric parameters. Dynamic and identification models obtained by symbolic generation, *Mechanics of Structures and machines*,1989,(to be published)
- [6] Chatelle,Ph., J.Duponcheel and J.-Cl.Samin, Investigations on non-conventional railway systems through a generalized multibody approach, Proceedings of the *Eighth IAVSD-IUTAM Symposium on the Dynamics of Vehicles on Road and Tracks*, Cambridge,Ma.,1983.
- [7] Kalker,J.J., Survey of wheel-rail rolling contact theory, *Journal of Vehicle System Dynamics*, vol 5, 1979.
- [8] Samin,J.-Cl., A multibody approach for dynamic investigation of rolling systems, *Ingenieur Archiv*, vol 54, 1984.
- [9] Samin,J.-Cl. and P.Y.Willems, Multibody systems with loops: a pseudo constraint approach, submitted for publication.
- [10] Boland,Ph., J-Cl Samin and P.Y.Willems, Stability analysis of interconnected deformable bodies with closed loop configuration, *AIAA Journal*, vol 13, 1975.
- [11] Wehage,R.A. and E.J. Haug, Generalized coordinate partitioning for dimension reduction in analysis of constrained dynamic systems, *Journal of Mechanical Structures*, vol 104, 1082.

ROBOTRAN

Abstract

This contribution presents the main features of the ROBOTRAN programme. This programme, based on d'Alembert Potential Power Principle, permits to derive, in symbolic form, the equations of motion of mechanical systems which can be represented by a set of rigid bodies, interconnected by one degree of freedom joints; it has important applications in the fields of robotics and vehicle dynamics. The variables of the system are the generalized variables of the various joints — linear displacement for prismatic joints and angular rotation for revolute joints. Constraints can be considered and in particular those resulting from loops of bodies can be generated by means of an auxiliary programme, CINEMA. The Lagrange multipliers technique permits to derive the complete set of equations of motion; a system reduction via the elimination of these multipliers and a coordinate partitioning method is possible. The obtained programme can be used as a sub-routine for any desired application such as numerical integration, stability analysis, control design, numerical linearization, eigenvalues determination.

Introduction¹

The mechanical system under consideration is assumed to be represented by a multibody system composed of interconnected rigid bodies. These bodies will be characterized by their mass distribution parameters and by geometric quantities which permit to localize their various interconnections. These connections are restricted to be one degree of freedom joints.

These *joints* can represent mechanical devices such as telescopic arms, guide rods, hinges, bearings. Relative motion can take place in the joints and interactions are produced. The variables describing the relative motions in the joints (linear or angular

¹The material which is common to this contribution and the joined contribution on the programme AUTODYN will not be repeated here; the lector is referred to this other contribution for more information.

displacements) are used as the generalized variables (even if other variables can in principle be used as outputs of the programme). The set of joint forces and torques consist of five constraint interactions (which normally do not appear in the final form of the equations) and one interaction associated with the degree of freedom and which can be expressed by an appropriate describing function.

Graph Description and System Parameters

The kinematical description of the system is greatly simplified when the *graph* of this system is defined. A tree structure is always used together with the notion of *filiation*. The notation $i \leq j$ means that body i belong to the direct path between the reference body and body j ; the notation $i < j$ means $i \leq j$ and $i \neq j$. For system with loops of bodies, a cutting procedure is determined and an augmented tree structure is defined. These procedures are equivalent to those used in *AUTODYN* (ref 1); the matrix *INBODY* is used in the main programme and, when there are closed loops the matrix *INLOOP* is defined in the auxiliary programme *CINEMA*.

The reference body of the system – body 0 – is assumed to be fixed with respect to inertial space. Nevertheless, for bedplate interaction evaluation, this body can be considered to be connected to inertial space by a fully constrained six degrees of freedom joint – this possibility is included in the so-called *identification* version of the programme.

The body parameters are the same as in *AUTODYN* but the joint library is restricted to the one degree of freedom *GLISS* and *AXE* joints.

In the *reference configuration*, all the body fixed bases $\{\hat{x}_j\}$ are supposed to coincide with the inertial reference base $\{\hat{I}\}$ and each joint axis, \hat{a}_j , is aligned with one of the axes of this inertial base.

For a *prismatic joint*, the displacement vector of body j with respect to the joint reference point, P_j (attached to the body which precedes j), \underline{z}_j , is given by $\underline{z}_j = z_j \hat{a}_j$ with $\hat{a}_j \equiv \hat{x}_{j\beta} \equiv \hat{x}_{i\beta} \quad \beta = 1, 2 \text{ or } 3$; z_j is the corresponding generalized variable. The relative orientation of the body bases is given by the identity matrix, $[A_{ji}] = [E]$, and the relative angular velocity is equal to zero, $\underline{\Omega}_j \equiv 0$.

For a *revolute joint*, the rotation around the joint axe $\hat{\mathbf{a}}_j \equiv \hat{\mathbf{x}}_{j\beta} \equiv \hat{\mathbf{x}}_{i\beta}$ $\beta = 1, 2$ or 3 is described by the rotation angle θ_j which is the corresponding generalized variable; the relative orientation matrix $[A_{ji}]$ is the corresponding elementary rotation matrix. The displacement vector is equal to zero, $\underline{\mathbf{z}}_j \equiv 0$ and the relative angular velocity vector is given by $\underline{\Omega}_j = \dot{\theta}_j \hat{\mathbf{a}}_j$. In this case, the joint reference point, P_j , is fixed with respect to the two bodies connected by the joint.

Kinematics and Constraints

The configuration of a rigid body j is completely determined by the position vector of its mass centre, $\underline{\mathbf{x}}_j$ and the orientation of its body-fixed base with respect to the reference body (with respect to inertial space), described by the matrix $[A_j]$.

The position vector of body j , $\underline{\mathbf{x}}_j$, can be expressed in terms of the vectors $\underline{\ell}_{ij}$ (which describe the relative position of the bodies for $i \neq j$ and the position of the mass centre for $i = j$ – see figure 2 in AUTODYN) and the vectors $\underline{\mathbf{z}}_i$ encountered along the path between the reference body and the body under consideration as:

$$\underline{\mathbf{x}}_j = \sum_{k:k \leq j} (\underline{\mathbf{z}}_k + \underline{\ell}_{kj}).$$

The position vector of the joint reference point is given by:

$$\underline{\mathbf{x}}_{P_j} = \sum_{k:k < j} (\underline{\mathbf{z}}_k + \underline{\ell}_{kj}).$$

The following barycentric parameters are defined from the masses, m_i , the geometric lengths (represented by the vectors $\underline{\ell}_{ij}$) and the inertia tensors, $\underline{\mathbf{I}}_i$, of the various bodies:

$$\overline{m}_i = \sum_{j:j \leq i} m_j,$$

$$\overline{m}_i \underline{\mathbf{b}}_i = \sum_{j:j \leq i} m_j \underline{\ell}_{ij},$$

$$\underline{\mathbf{K}}_i = \underline{\mathbf{I}}_i - \sum_{j:j \leq i} m_j \tilde{\underline{\ell}}_{ij} \tilde{\underline{\ell}}_{ij}.$$

where the notation $\tilde{\underline{\ell}}$ refers to the skew-symmetric tensor associated with the vector $\underline{\mathbf{x}}$.

The orientation of the base $\{\hat{x}_j\}$ with respect to inertial space is then obtained via the product of the relative orientation matrices encountered along the path between body j and the reference body. The absolute angular velocity of this base fixed to body j , $\underline{\omega}_j$, is simply the (vector) sum of the relative angular velocities in the joints along the path from the reference body:

$$\underline{\omega}_j = \sum_{k:k \leq j} \underline{\Omega}_k = \sum_{k:k \leq j} \dot{\theta}_k \hat{a}_k.$$

The velocities of the various mass centres are given by:

$$\dot{\underline{x}}_j = \sum_{k:k \leq j} [\dot{z}_k \hat{a}_k + \underline{\omega}_k \times (\underline{z}_k + \underline{\ell}_{kj})].$$

The expressions of the mass centre accelerations and of the body bases angular accelerations are then obtained by time derivation.

The loop constraints are obtained by expressing the fact that for the M_ℓ loops the origin and closure (fictitious) bodies (the bodies β_α and γ_α respectively) coincide i.e. have the same mass centre and coinciding bases; these bases have the same angular velocities. We can write these relations as:

$$\underline{x}_{\beta_\alpha} = \underline{x}_{\gamma_\alpha} \quad \dot{\underline{x}}_{\beta_\alpha} = \dot{\underline{x}}_{\gamma_\alpha} \quad [A_{\beta_\alpha}] = [A_{\gamma_\alpha}] \quad \text{and} \quad \underline{\omega}_{\beta_\alpha} = \underline{\omega}_{\gamma_\alpha} \quad \alpha = 1 \cdots M_\ell.$$

The orientation constraints are not independant and we choose to retain the elements (1,2), (2,3) and (3,1); the other relations must be checked a posteriori.

The first and second time derivatives of the position constraints are obtained in a straightforward manner.

Once the M_{joint} joint constraints and M_{user} users' constraints have been added, we obtain a set of M constraints. These constraints supposed to be holonomic are represented by M (independant) algebraic relations between the generalized variables of the problem q_α represented by the $(N \times 1)$ vector $[q]$. We write them under the form of a M vector relation

$$[S(q_\alpha, t)] = 0.$$

The time derivative of these constraints can be written in matrix form:

$$[A(q_\alpha, t)] [\dot{q}] + [b(q_\alpha, t)] = 0,$$

where the $(M \times N)$ matrix $[A]$ is the gradient matrix of the constraints. We assume that, possibly after elimination of redundant constraints, the corresponding matrix $[A]$ is a full rank matrix.

Equations of motion

The equations of motion of the augmented tree will be obtained from the Potential Power Principle.

The local form of the equations of motion for any system (without local internal angular momentum distribution) is given by the relation

$$\ddot{\underline{x}} = \underline{f},$$

i.e. the acceleration of any point of the system $\ddot{\underline{x}}$ is equal everywhere (except possibly on set of points with zero mass measure) to the corresponding value of the local force density \underline{f} .

For any vector field defined by the vectors \underline{a} , we obtain the following (scalar) identity after integration over the system:

$$\sum_j \int_{B_j} (\ddot{\underline{x}} - \underline{f}) \cdot \underline{a} dm = 0.$$

We can choose the vectors \underline{a} to be a potential (possible) change of velocity compatible with the choice of generalized coordinates, i.e;

$$\underline{\Delta \dot{x}} = \sum_k \underline{b}_{q_k} \Delta \dot{q}_k,$$

where here q_k is either z_k or θ_k . The vectors \underline{b}_{q_k} represent the field of potential velocities induced by a unit potential change of \dot{q}_k .

The Potential Power Principle can then be written

$$\sum_j \left[\int_{B_j} (\ddot{\underline{x}} - \ddot{\underline{f}}) \cdot \underline{b}_{q_k} dm \right] \Delta \dot{q}_k = 0.$$

With:

$$[\Phi_k] = \sum_j \int_{B_j} \ddot{\underline{x}} \cdot \underline{b}_{q_k} dm \quad \text{and} \quad [Q_k] = \sum_j \int_{B_j} \ddot{\underline{f}} \cdot \underline{b}_{q_k} dm,$$

the Potential Power Principle is then given by

$$\sum_k \{[\Phi_k] - [Q_k]\} \Delta \dot{q}_k = 0.$$

Let us consider the prismatic joint k. The corresponding vector \underline{b}_{z_k} will be given by:

$$\underline{b}_{z_k} = 0 \quad x \in B_j \quad j : j \leq k,$$

$$\underline{b}_{z_k} = \hat{\underline{a}}_k \quad x \in B_j \quad j : k \leq j.$$

For any point, x, in body j, we can write:

$$\underline{x} = \underline{x}_j + \underline{r}_j \quad \text{with} \quad \int_{B_j} \underline{r}_j dm = 0,$$

where \underline{r}_j is the position vector of x with respect to the centre of mass of the body j.

The term Φ_{z_k} will be given by:

$$\Phi_{z_k} = \sum_{j:k \leq j} \int_{B_j} \ddot{\underline{x}} \cdot \underline{a}_k dm = \sum_{j:k \leq j} m_j \ddot{\underline{x}}_j \cdot \underline{a}_k,$$

where $\ddot{\underline{x}}_j = \sum_{i:i \leq j} (\ddot{\underline{z}}_i + \ddot{\underline{\ell}}_{ij})$ with $\ddot{\underline{z}}_i = \ddot{z} \hat{\underline{a}}_i + \dot{\underline{\omega}}_i \times z_i \hat{\underline{a}}_i + 2\dot{\underline{\omega}}_i \times \dot{z}_i \hat{\underline{a}}_i + \dot{\underline{\omega}}_i \times \dot{\underline{\omega}}_i \times z_i \hat{\underline{a}}_i$

$$\text{and} \quad \ddot{\underline{\ell}}_{ij} = \dot{\underline{\omega}}_i \times \underline{\ell}_{ij} + \dot{\underline{\omega}}_i \times \dot{\underline{\omega}}_i \times \underline{\ell}_{ij}.$$

The $[\Phi_{z_k}]$ function can then be written:

$$[\Phi_{z_k}] = \left\{ \sum_{j:k \leq j} \sum_{i:i \leq j} m_j (\ddot{\underline{z}}_i + \ddot{\underline{\ell}}_{ij}) \right\} \cdot \hat{\underline{a}}_k \\ = \left\{ \sum_{j:k \leq j} \sum_{i:i < k} m_j (\ddot{\underline{z}}_i + \ddot{\underline{\ell}}_{ij}) + \sum_{j:k \leq j} \sum_{i:k \leq i \leq j} m_j (\ddot{\underline{z}}_i + \ddot{\underline{\ell}}_{ij}) \right\} \cdot \hat{\underline{a}}_k;$$

using the sum property $\sum_{j:k \leq j} \sum_{i:k \leq i \leq j} \equiv \sum_{i:k \leq i} \sum_{j:j \leq i}$ and the definition of the geometrical length vectors, $\underline{\ell}_{ij}$ $i \neq j$, and of the barycentric parameter \bar{m}_i and \underline{b}_i , this relation becomes

$$\Phi_{z_k} = \left\{ \bar{m}_k \sum_{i:i < k} (\ddot{\underline{z}}_i + \ddot{\underline{\ell}}_{ik}) + \sum_{i:k \leq i} \bar{m}_i (\ddot{\underline{z}}_i + \ddot{\underline{b}}_i) \right\} \cdot \underline{\hat{a}}_k.$$

Similarly, the corresponding generalized interaction is given by:

$$Q_{z_k} = \sum_{j:k \leq j} \int_{B_k} \underline{f} \cdot \underline{\hat{a}}_k dm = \sum_{j:k \leq j} \underline{F}_j^{ext} \cdot \underline{\hat{a}}_k + F_k,$$

where F_k is the force applied in the joint k (for instance a control force) and must be given by a describing function;

\underline{F}_j^{ext} is the resultant of external forces applied to body j (by the exterior of the system) and can be separated into applied force \underline{F}_j^{ext} and gravity forces $m_j \underline{g}$; for gravity forces, we can write:

$$\sum_{j:k \leq j} m_j \underline{g} = \bar{m}_k \underline{g}.$$

For a revolute joint k, the potential velocity field induced by a unit potential change of the potential angular velocity, $\Delta \dot{\theta}_k$, is described by:

$$\underline{b}_{z_k} = 0 \text{ and } \underline{\Delta \dot{\omega}} = 0 \quad x \in B_j \quad j:j \leq k,$$

$$\underline{b}_{z_k} = \underline{\hat{a}}_k \times \underline{p}_{kj} \text{ and } \underline{\Delta \dot{\omega}} = \Delta \dot{\theta}_k \underline{\hat{a}}_k \quad x \in B_j \quad j:k \leq j,$$

where \underline{p}_{kj} is the position vector of x (of body j) with respect to P_k .

The corresponding function Φ_{θ_k} is then equal to:

$$\Phi_{\theta_k} = \sum_{j:k \leq j} \int_{B_j} \ddot{\underline{x}} \cdot [\underline{\hat{a}}_k \times \underline{p}_{kj}] dm = \sum_{j:k \leq j} \int_{B_j} [\underline{p}_{kj} \times \ddot{\underline{x}}] \cdot \underline{\hat{a}}_k dm,$$

i.e. is the projection on $\underline{\hat{a}}_k$ of the moment of momentum with respect to the point P_k of the set of bodies B_j with $j:k \leq j$.

With

$$\underline{p}_{kj} = \sum_{i:k \leq i \leq j} (\underline{z}_i + \underline{\ell}_{ij}) + \underline{r}_j$$

and

$$\ddot{\underline{x}} = \sum_{\ell:\ell \leq j} (\ddot{\underline{z}}_\ell + \ddot{\underline{\ell}}_{\ell j}) + \ddot{\underline{r}}_j,$$

$$\Phi_{\theta_k} = \left\{ \sum_{i:k \leq i} \int_{B_i} \underline{r}_i \times \ddot{\underline{r}}_i dm + \sum_{j:k \leq j} \sum_{i:k \leq i \leq j} \sum_{\ell:\ell \leq j} m_j (\underline{z}_i + \underline{\ell}_{ij}) \times (\ddot{\underline{z}}_\ell + \ddot{\underline{\ell}}_{\ell j}) \right\} \cdot \hat{\underline{a}}_k.$$

The first term of this expression represents the time derivative of the moment of momentum of the body i with respect to its mass centre; the bodies being rigid, these terms can be expressed as:

$$\int_{B_i} \underline{r}_i \times \ddot{\underline{r}}_i dm = \underline{I}_i \cdot \dot{\underline{\omega}}_i + \underline{\omega}_i \times \underline{I}_i \cdot \underline{\omega}_i$$

The sum $\sum_{\ell:\ell \leq j}$ is divided into three parts $\sum_{\ell:\ell \leq j} \equiv \sum_{\ell:\ell < i} \vee \ell = i \vee \sum_{\ell:i < \ell \leq j}$ and the following sum properties are used:

$$\sum_{j:k \leq j} \sum_{i:k \leq i \leq j} \equiv \sum_{i:k \leq i} \sum_{j:i \leq j} \text{ and } \sum_{j:k \leq j} \sum_{i:k \leq i \leq j} \sum_{\ell:i < \ell \leq j} \equiv \sum_{i:k \leq i} \sum_{\ell:i < \ell} \sum_{j:\ell \leq j}.$$

Further, if the definitions of the barycentric parameters \underline{b}_i and \underline{K}_i are used, the function Φ_{θ_k} can be written:

$$\begin{aligned} \Phi_{\theta_k} = & \left\{ \sum_{i:k \leq i} \underline{z}_i \times \left[\bar{m}_i \sum_{\ell:\ell < i} (\ddot{\underline{z}}_\ell + \ddot{\underline{\ell}}_{\ell i}) + \sum_{\ell:i \leq \ell} \bar{m}_\ell (\ddot{\underline{z}}_\ell + \ddot{\underline{b}}_\ell) \right] \right. \\ & + \sum_{i:k \leq i} \left[\sum_{\ell:\ell < i} \bar{m}_i \underline{b}_i \times (\ddot{\underline{z}}_\ell + \ddot{\underline{\ell}}_{\ell i}) + \sum_{\ell:i < \ell} \ddot{\underline{\ell}}_{i\ell} \times \bar{m}_\ell (\ddot{\underline{z}}_\ell + \ddot{\underline{b}}_\ell) \right] \\ & \left. + \sum_{i:k \leq i} [\underline{K}_i \cdot \dot{\underline{\omega}}_i + \underline{\omega}_i \times \underline{K}_i \cdot \underline{\omega}_i + \bar{m}_i \underline{b}_i \times \ddot{\underline{b}}_i] \right\} \cdot \hat{\underline{a}}_k. \end{aligned}$$

Similarly, the corresponding generalized interaction is given by:

$$\begin{aligned} Q_{\theta_k} = & \sum_{j:k \leq j} \int_{B_k} \underline{f} \cdot [\hat{\underline{a}}_k \times \underline{p}_{kj}] dm = \sum_{j:k \leq j} \int_{B_k} [\underline{p}_{kj} \times \underline{f}] \cdot \hat{\underline{a}}_k dm, \\ = & \left\{ \sum_{j:k \leq j} \sum_{i:k \leq i < j} (\underline{z}_i + \underline{\ell}_{ij}) \times \underline{F}_j^{ext} + \sum_{j:k \leq j} \underline{L}_{P_j}^{ext} \right\} \cdot \hat{\underline{a}}_k + M_k, \end{aligned}$$

where $\underline{L}_{P_j}^{ext}$ is the resultant of external torques with respect to the point P_j of joint j (the sum of the moment of external forces with respect to this point and the sum of

pure external torques). This relation can also be written:

$$Q_{\theta_k} = \left\{ \sum_{i:k \leq i} \sum_{j:i < j} (\underline{\mathbf{z}}_i + \underline{\ell}_{ij}) \times \underline{\mathbf{F}}_j^{ext} + \underline{\mathbf{L}}_{P_j}^{ext} \right\} \cdot \hat{\underline{\mathbf{a}}}_k + M_k.$$

Here too, we can separate external forces into gravity forces and applied forces $\underline{\mathbf{F}}_i^{ext}$; for gravity, the corresponding generalized interaction is given by:

$$Q_{\theta_k}^g = \left\{ \sum_{i:k \leq i} \bar{m}_i (\underline{\mathbf{z}}_i + \underline{\mathbf{b}}_i) \times \underline{\mathbf{g}} \right\} \cdot \hat{\underline{\mathbf{a}}}_k.$$

For unconstrained systems, the various potential changes of generalized velocities are independant and their coefficients in the expression of the Potential Power Principle are equal to zero; this provides the equation of motion under the form:

$$F_k = \left\{ \bar{m}_k \sum_{i:i < k} (\ddot{\underline{\mathbf{z}}}_i + \ddot{\underline{\ell}}_{ik}) + \sum_{i:k \leq i} \bar{m}_i (\ddot{\underline{\mathbf{z}}}_i + \ddot{\underline{\mathbf{b}}}_i) - \bar{m}_k \underline{\mathbf{g}} - \sum_{i:k \leq i} \underline{\mathbf{F}}_i^{ext} \right\} \cdot \hat{\underline{\mathbf{a}}}_k$$

and

$$\begin{aligned} M_k = & \left\{ \sum_{i:k \leq i} \underline{\mathbf{z}}_i \times \left[\bar{m}_i \sum_{\ell:\ell < i} (\ddot{\underline{\mathbf{z}}}_\ell + \ddot{\underline{\ell}}_{\ell i}) + \sum_{\ell:i \leq \ell} \bar{m}_\ell (\ddot{\underline{\mathbf{z}}}_\ell + \ddot{\underline{\mathbf{b}}}_\ell) \right] \right. \\ & + \sum_{i:k \leq i} \left[\sum_{\ell:\ell < i} \bar{m}_i \underline{\mathbf{b}}_i \times (\ddot{\underline{\mathbf{z}}}_\ell + \ddot{\underline{\ell}}_{\ell i}) + \sum_{\ell:i < \ell} \ddot{\underline{\ell}}_{i\ell} \times \bar{m}_\ell (\ddot{\underline{\mathbf{z}}}_\ell + \ddot{\underline{\mathbf{b}}}_\ell) \right] \\ & + \sum_{i:k \leq i} [\underline{\mathbf{K}}_i \cdot \dot{\underline{\omega}}_i + \underline{\omega}_i \times \underline{\mathbf{K}}_i \cdot \underline{\omega}_i + \bar{m}_i \underline{\mathbf{b}}_i \times \ddot{\underline{\mathbf{b}}}_i] \\ & - \sum_{i:k \leq i} \bar{m}_i (\underline{\mathbf{z}}_i + \underline{\mathbf{b}}_i) \times \underline{\mathbf{g}} \\ & \left. - \sum_{i:k \leq i} \left[\sum_{j:j < i} (\underline{\mathbf{z}}_i + \underline{\ell}_{ij}) \times \underline{\mathbf{F}}_j^{ext} + \underline{\mathbf{L}}_{P_j}^{ext} \right] \right\} \cdot \hat{\underline{\mathbf{a}}}_k. \end{aligned}$$

One should note the recursive form of these relations. In particular if two successive revolute or prismatic joints have parallel axes, all the calculations carried out for the second one could be saved for the first one in the chain (such a procedure is implemented in Robotran which detects parallel axes)

It can also be of interest to note that all the parameters of these equations are bilinear functions of the above-defined barycentric parameters and the geometrical lengths – components of the vectors $\underline{\ell}_{ij}$ ($i \neq j$) -. This is particularly interesting for identification purposes [1].

For more complicated joints, as those considered in AUTODYN, the structure of the vector equations remains essentially the same, but the interpretation of joint generalized interactions is more delicate; the joint equations are then accordingly linear combinations of the components of the corresponding joint vector equations.

When constraints are considered, the potential power of the complete set of interactions will be written as:

$$\{ [\Phi] - [Q] \}^T [\Delta \dot{q}] = 0,$$

(where $[\Phi]$ and $[Q]$ are the $(N \times 1)$ vectors of the Φ_{q_k} and Q_{q_k} respectively) for the variations of potential velocities which satisfy the constraints, i.e. which are such that

$$[A(q_\alpha, t)] [\Delta \dot{q}] = 0.$$

Finally, the equations of motion are given by the N equations

$$[\Phi] - [A]^T [\lambda] = 0,$$

(where the $(M \times 1)$ vector $[\lambda]$ is the corresponding vector of *Lagrange Multipliers*) associated with the constraints in the form $[S] = 0$, for holonomic constraints or in the form $[A][\dot{q}] + [b] = 0$ for holonomic or non holonomic cases.

These equation can still be reduced, for instance by using a coordinate partitioning method or, when possible, by solving the constraints analytically and eliminating the Lagrange multipliers.

Programme arrangement

A ROBOTRAN application is divided in four main steps:

```

input data processsing
dynamical equations
system reduction
application

```

The input of the data preprocessor includes the graph information (the *INBODY* and *INLOOP* matrices), the geometrical (the ℓ_{ij} vectors) and mass distribution parameters (the masses m_i and the \mathbf{I}_i tensors) and the information relative to the joints. The output of the data preprocessor provides the barycentric parameters and the vector of generalized coordinates; the joint generalized interactions, F_k or M_k , are also provided at this stage.

The main programme computes the various terms appearing in the Potential Power Principle i.e.:

the matrix $[\Phi]$

the generalized interaction matrix $[Q^{ext}]$ which includes the effect of gravity and other external interactions.

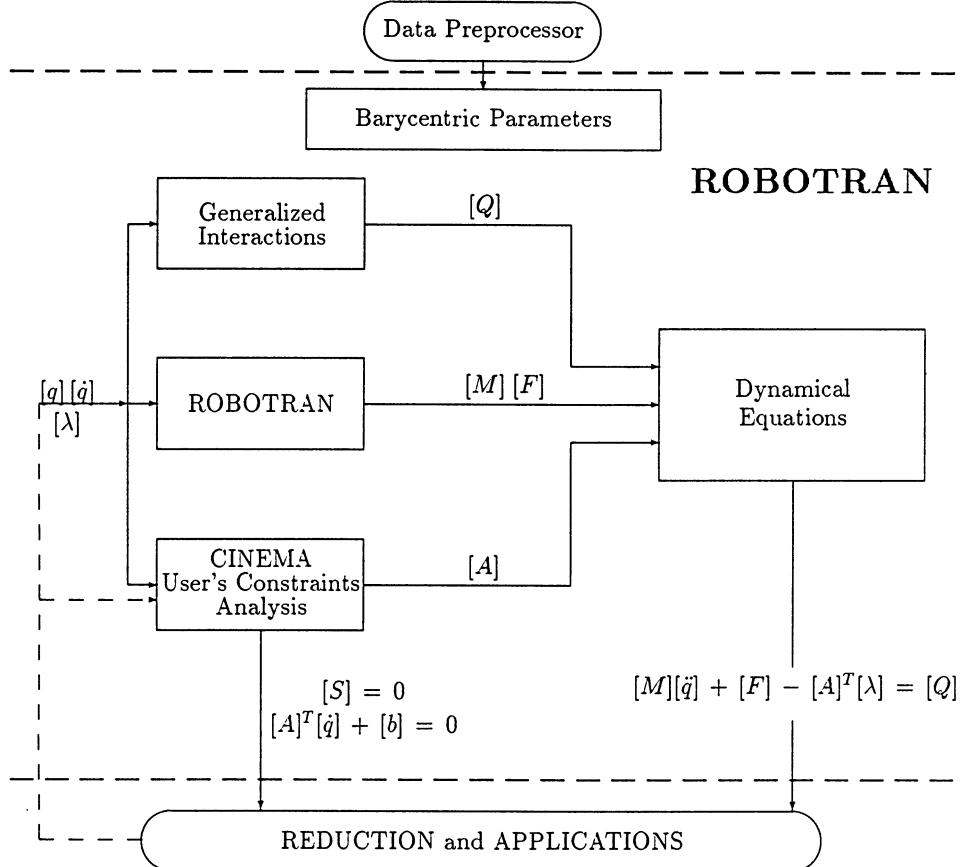
The loop constraints, their time derivative and their gradient matrix $[A]$ can be obtained from an auxiliary symbolic programme, CINEMA, which uses the some input information (except mass distribution and interaction parameters) as the main programme. The combined output consists of a system of $(N + M)$ (differential or mixed algebraic and differential) equations in the $(N + M)$ unknowns $[q]$ and $[\lambda]$.

This system can then be reduced, by the elimination of the Lagrange multipliers and of a set of variables (equal to the number of holonomic constraints). For simple constraints, this can be achieved by a classical symbolic manipulator, but for more complicated systems, this step has to be performed numerically; one can, for instance, procede as described in the joint paper on AUTODYN.

The following applications are readily available: numerical integration, equilibrium investigation, numerical linearization, stability analysis, eigenvalue determination, control design. ROBOTRAN is well suited for applications in robotic [1] and vehicle dynamics [2].

An extended version of ROBOTRAN, the *identification version*, permits to determine the constraint forces appearing in the various joints and in particular in the bed-plate – the joint which precedes the body 0.

The organisation of the programme can be summarized as follows:



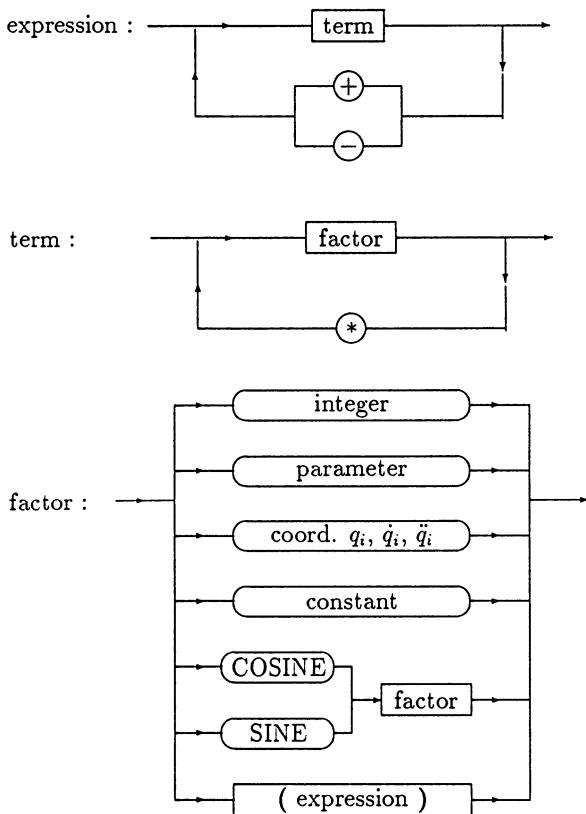
ROBOTRAN Syntax

We may divide the process which writes mathematical expressions in a symbolic form into several parts :

- the identification of an expression according to a predefined syntax;
- the translating of such an expression into a programming language;
- the manipulation (+, -, *) of the expressions;
- the reduction.

Each mathematical expression contains one or several terms linked by minus or plus signs, each term being the product of several factors. A factor can be an integer, a barycentric parameter, a generalized coordinate ($q_i, \dot{q}_i, \ddot{q}_i$), a geometrical constant, a trigonometrical function whose argument is a factor or a mathematical expression between brackets.

This can be schematically represented as follows [3]:

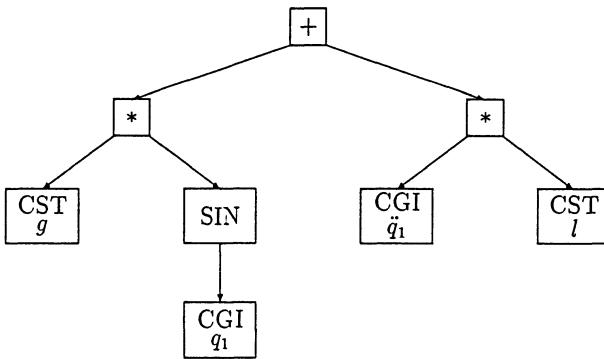


Thus defined, an expression can be considered as a tree whose nodes represent the *nature* of the expressions, i.e.

- integer whose value has to be given as an integer constant,
- barycentric parameter, generalized coordinate or geometrical constant which have to be identified by an appropriate string of characters,

- cosine or sine function whose argument (also an expression) must be given,
- plus, minus, times operators whose two operands must be given as expressions.

For instance, the expression $g * \sin(q_1) + \ddot{q}_1 * l$ could be represented by the following tree :



The increase in size of the expressions during the elaboration of the equations and the tree structure described above lead us to use dynamical variables (pointers) and an appropriate programming language for this kind of manipulations.

The operations that we have to perform in order to manipulate expressions are:

- creation of pointers and assignment of their nature,
- reading, initialisation or assignment of the value and/or identification for the integers, geometrical constants, barycentric parameters and generalized coordinates,
- assignment of the arguments and operands to the appropriate pointers for the trigonometrical functions and the arithmetical operations,
- deletion of redundant calculations (adding 0, multiplying by 0 or 1).

For instance, if we know the expressions “a” and “b”, the operation “a+b” consists in creating a new pointer whose nature is plus, whose left operand points to “a” and right one points to “b”.

In order to avoid expressions like :

$$x = a + b - a$$

we introduce a relation of order to classify the expressions. This order is given first by the position of the *natures* in the operators set and then by a lexicographical order on the strings of characters (for instance, $0 < \cos(q_1) < \sin(q_1) < \bar{m} * l < q_2 + q_3$). At each operation, we rewrite the resulting expression according to the prescribed order and then simplify consecutive equal terms with opposite signs. Since one of the goals of ROBOTRAN is to present the equations in a form amenable to localizing possible parameter combinations, each term is written on a single line according to the following syntax:

parameter × integer × generalized variables × constants × trigonometric expressions

We can thus ensure that the expression appearing in the obtained equations cannot be further reduced without using specific formulae of trigonometry. This kind of reductions can be performed by several programmes (MACSYMA, SMP, REDUCE, ...) which could be used subsequently.

Test example: Five degrees of freedom robot.

This robot consist of three bodies interconnected by two rotational-translational joints and one rotational joint. The two degrees of fredom joints have been divided into two successive joints, a revolute joint followed by a prismatic joint; these two joints are separated by a massless and dimensionless body which do not affect the form of the equations. In the next page, the input data file is presented.

The output of the data preprocessor provides the barycentric parameters. The complete set of equations is presented in the following pages (*LATEX* instructions have been used to provide this presentation of the output file). In these expressions some auxiliary variables have been used and some trivial trigonometric transformations have been performed. The parameters F2 and F4 are the control forces in the prismatic joints and M1,M2 and M5 are the corresponding torques in the revolute joints.

Spatial Motion of Robot		<i>Title</i>
5		<i>Number of bodies</i>
0 1 2 3 4		<i>INBODY</i>
0;		
m1;		
0;		<i>Masses</i>
m2;		
m3;		
0; 0; 0;		
0; 0;		
0;		
I111; 0; 0;		
I122; 0;		
I133;		
0; 0; 0;		
0; 0;		<i>Tensors of Inertia</i>
0;		
I211; 0; 0;		
I222; 0;		
I233;		
I311; 0; 0;		
I322; 0;		
I333;		
0; 0; z1;		
0; 0; 0;		
0; 0; 0;		<i>Positions of mass centres</i>
0; 0; 0;		
0; c; 0;		
0; 0; 0;		
0; 0; 0;		
0; 0; 0;		<i>Positions of articulation points</i>
0; 0; 0;		
0; 1; 0;		
R3 q1; qp1; qpp1;		<i>Type of joint</i>
T3 q2; qp2; qpp2;		R : revolute
R2 q3; qp3; qpp3;		T : translation
T2 q4; qp4; qpp4;		and corresponding variables (q, \dot{q}, \ddot{q})
R1 q5; qp5; qpp5;		
0; 0; 0;		<i>External forces</i>
⋮		
0; 0; 0;		<i>External torques</i>
⋮		
0; 0; -g;		<i>Gravity</i>

SPATIAL MOTION OF ROBOT - DYNAMICAL EQUATIONS

Barycentric Parameters of order 0

```
mb1      =  m1+m2+m3
mb3      =  m2+m3
mb5      =  m3
```

Barycentric Parameters of order 1

```
b42      =  mb5*l
b52      =  mb5*c
```

Barycentric Parameters of order 2

```
K211    =  I111
K222    =  I122
K233    =  I133
K411    =  I211+mb5*l*l
K422    =  I222
K433    =  I233+mb5*l*l
K511    =  I311+mb5*c*c
K522    =  I322
K533    =  I333+mb5*c*c
```

Trigonometric Functions

```
C1      =  cos(q1)
S1      =  sin(q1)
C3      =  cos(q3)
S3      =  sin(q3)
C5      =  cos(q5)
S5      =  sin(q5)
C55     =  C5*C5 - S5*S5
S55     =  S5*C5 + S5*C5
```

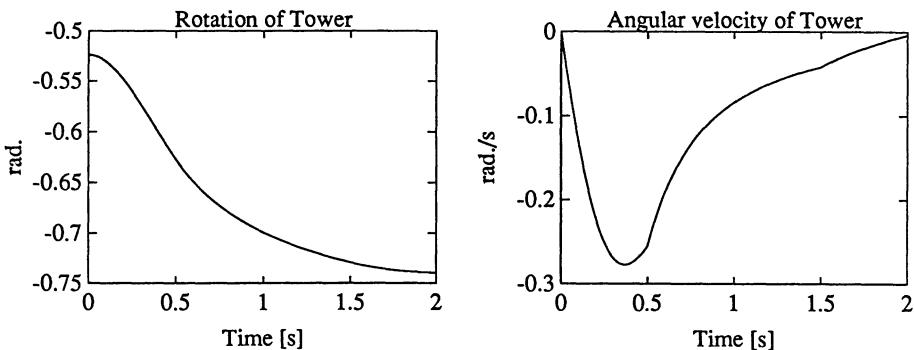
Auxiliary variables

```
bb42    =  b42+mb3*q4
Kb411   =  K411+b42*2*q4+mb3*q4*q4
Kb413   =  K413+b42*2*q4+mb3*q4*q4
```

Equations

M5 =	$-K_{511} * qpp1 * S_3$	F2 =	$mb1 * qpp2$
	$-b_{52} * qpp1 * C_5 * S_3 * (q4+l)$		$-b_{52} * qpp3 * S_3 * S_5$
	$+b_{52} * qpp2 * C_3 * C_5$		$+b_{52} * qpp5 * C_3 * C_5$
	$-b_{52} * qpp4 * S_5$		$-b_{52} * qpp3 * qpp3 * C_3 * S_5$
	$+K_{511} * qpp5$		$-b_{52} * 2 * qpp3 * qpp5 * C_5 * S_3$
	$-K_{522} * qpp1 * qpp1 * C_3 * C_3 * C_5 * S_5$		$-b_{52} * qpp5 * qpp5 * C_3 * S_5$
	$+b_{52} * qpp1 * qpp1 * S_5 * (q4+l)$		$+mb1 * g$
	$+K_{522} * qpp3 * qpp3 * C_5 * S_5$	M1 =	$K_{233} * qpp1$
	$-K_{533} * qpp3 * qpp3 * C_5 * S_5$		$+Kb_{411} * qpp1 * S_3 * S_3$
	$-K_{511} * qpp1 * qpp3 * C_3$		$+Kb_{433} * qpp1 * C_3 * C_3$
	$-K_{522} * qpp1 * qpp3 * C_3 * C_55$		$+K_{511} * qpp1 * S_3 * S_3$
	$+K_{533} * qpp1 * qpp3 * C_3 * C_55$		$+K_{522} * qpp1 * C_3 * C_3 * S_5 * S_5$
	$-b_{52} * 2 * qpp1 * qpp4 * C_5 * S_3$		$+K_{533} * qpp1 * C_3 * C_3 * C_5 * C_5$
	$+b_{52} * g * C_3 * C_5$		$+b_{52} * 2 * qpp1 * C_5 * (q4+l)$
F4 =	$b_{52} * qpp1 * S_3 * S_5$		$+K_{522} * qpp3 * C_3 * C_5 * S_5$
	$+mb3 * qpp4$		$-K_{533} * qpp3 * C_3 * C_5 * S_5$
	$-b_{52} * qpp5 * S_5$		$-b_{52} * qpp3 * C_3 * S_5 * (q4+l)$
	$-bb42 * qpp1 * qpp1$		$+b_{52} * qpp4 * S_3 * S_5$
	$-b_{52} * qpp1 * qpp1 * C_5$		$-K_{511} * qpp5 * S_3$
	$+b_{52} * 2 * qpp1 * qpp3 * C_3 * S_5$		$-b_{52} * qpp5 * C_5 * S_3 * (q4+l)$
	$+b_{52} * 2 * qpp1 * qpp5 * C_5 * S_3$		$-K_{522} * qpp3 * qpp3 * C_5 * S_3 * S_5$
	$-b_{52} * qpp5 * qpp5 * C_5$		$+K_{533} * qpp3 * qpp3 * C_5 * S_3 * S_5$
			$+b_{52} * qpp3 * qpp3 * S_3 * S_5 * (q4+l)$
M3 =	$K_{522} * qpp1 * C_3 * C_5 * S_5$		$+Kb_{411} * 2 * qpp1 * qpp3 * C_3 * S_3$
	$-K_{533} * qpp1 * C_3 * C_5 * S_5$		$-Kb_{433} * 2 * qpp1 * qpp3 * C_3 * S_3$
	$-b_{52} * qpp1 * C_3 * S_5 * (q4+l)$		$+K_{511} * 2 * qpp1 * qpp3 * C_3 * S_3$
	$-b_{52} * qpp2 * S_3 * S_5$		$-K_{522} * 2 * qpp1 * qpp3 * C_3 * S_3 * S_5 * S_5$
	$+K422 * qpp3$		$-K_{533} * 2 * qpp1 * qpp3 * C_3 * C_5 * C_5 * S_3$
	$+K_{522} * qpp3 * C_5 * C_5$		$+bb42 * 2 * qpp1 * qpp4$
	$+K_{533} * qpp3 * S_5 * S_5$		$+b_{52} * 2 * qpp1 * qpp4 * C_5$
	$-Kb_{411} * qpp1 * qpp1 * C_3 * S_3$		$+b_{52} * qpp5 * qpp5 * S_3 * S_5 * (q4+l)$
	$+Kb_{433} * qpp1 * qpp1 * C_3 * S_3$		$+K_{522} * 2 * qpp1 * qpp5 * C_3 * C_3 * C_5 * S_5$
	$-K_{511} * qpp1 * qpp1 * C_3 * S_3$		$-K_{533} * 2 * qpp1 * qpp5 * C_3 * C_3 * C_5 * S_5$
	$+K_{522} * qpp1 * qpp1 * C_3 * S_3 * S_5 * S_5$		$-b_{52} * 2 * qpp1 * qpp5 * (q4+l) * S_5$
	$+K_{533} * qpp1 * qpp1 * C_3 * C_5 * C_5 * S_3$		$-K_{511} * qpp3 * qpp5 * C_3$
	$-b_{52} * 2 * qpp1 * qpp4 * C_3 * S_5$		$+K_{522} * qpp3 * qpp5 * C_3 * C_55$
	$+K_{511} * qpp1 * qpp5 * C_3$		$-K_{533} * qpp3 * qpp5 * C_3 * C_55$
	$+K_{522} * qpp1 * qpp5 * C_3 * C_55$		$-b_{52} * 2 * qpp3 * qpp5 * C_3 * C_5 * (q4+l)$
	$-K_{533} * qpp1 * qpp5 * C_3 * C_55$		
	$-K_{522} * qpp3 * qpp5 * S_55$		
	$+K_{533} * qpp3 * qpp5 * S_55$		
	$-b_{52} * g * S_3 * S_5$		

Finally these equations have been integrated and typical outputs, the rotation angle and the angular velocity of the first body, are presented below.



Bibliography

- [1] Raucent B., G.Bastin, G.Campion and J.-Cl.Samin. Identification of barycentric parameters of robotic manipulators from external measurements. Proceedings of the *Eighth IFAC Symposium on Identification and System Parameter Estimation*, Beijing, China, 1988.
- [2] Bonivert L., P.Maes and J.-Cl.Samin, Simulation of the lateral dynamics of the GLT vehicle by means of ROBOTRAN; a model generator for robots, *Simulation in the Factory of the Future*, M.Muller and R. Reddy (eds), SCS publication, Belgium, 1988
- [3] Tremblay J.-P. and P.G. Sorenson. *An Introduction to Data Structures with Applications*. McGraw-Hill, New York, second edition, 1984.

The reader is invited to refer to the references of the joint paper on AUTODYN for additional bibliographical information.

SIMPACK – A Computer Program for Simulation of Large-motion Multibody Systems

Wolfgang Rulka, MAN-Technologie AG, 8000 München

1 Introduction

SIMPACK is a computer program for the simulation and calculation of motions and interacting forces of three-dimensional mechanical systems. The individual bodies can carry out geometrically large (non-linear) movements and may also exhibit elastic deformations.

The mechanical systems are defined modular in library elements such as joints, forces, etc., so that the user does not have to formulate the equations of motion. SIMPACK is an open program package which allows the user to intervene at many points in the program. For example, the fix implemented elements can be extended by self-programmed USER routines. By these routines SIMPACK can be adapted to the user's fields of application.

With the aid of the postprocessor which is an integral part of SIMPACK the numerical results can be displayed as x-y plots in an interpretable form. On special computers the results can also be shown as 3D animation to facilitate interpretation.

Fig. 1 shows typical applications for computer simulations using SIMPACK. The advantages of the formalism on which the SIMPACK program is based become especially evident in the case of systems having a large number of bodies. Its highest efficiency is shown with mechanical systems which with respect to kinematic constraints have a tree-like structure or which have relatively few kinematically closed loops in relation to the number of bodies in the system.

In the following sections the main characteristics of the SIMPACK program, its theoretical background and the method of working with the program are explained. Results of the two test examples are shown finally.

2 Keywords on theoretical Background

The main job of the formalism is the generation of non-linear differential equations of second order:

$$\ddot{z} = g_1(\dot{z}, z, t).$$

It gives the non-linear relationship between the second derivative \ddot{z} of the coordinates of position and the variable quantities \dot{z}, z and t used to describe the motions of the mechanical system. Moreover, the formalism also allows computation of the constraint forces

$$f^z = g_2(\dot{z}, z, t).$$

in the joints. Joints are massless connections which are constraining the relative motions of two bodies, such as translational rectilinear sliding pairs or a pendulum rod. The laws of force for the applied forces can be *passive* or *active* and process an eigendynamic characterized by a differential equation of first order:

$$\dot{x} = g_3(x, \dot{z}, z, t).$$

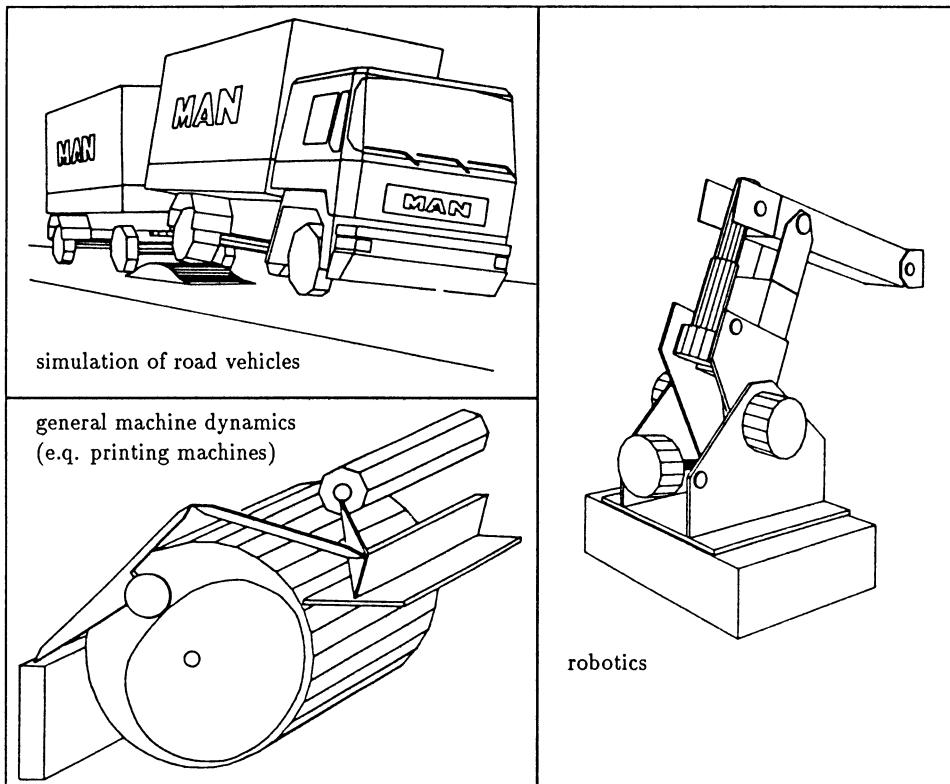


Figure 1: Typical applications

This enables, for example, complex controllers or the coupling of substructures to be taken into consideration.

The basic idea the formalism is built on is explained in section 6. Some of the keywords for the formalism on which the program is based are:

- Non-linear kinematics for the body motions. The bodies can have "*large*" relative movements to one another.
- The motions of the bodies are described by **relative coordinates**. These are e.g. the three rotation angles of a ball-and-socket joint. The choice of relative coordinates is of advantage in formulating the equations of motion for elastic bodies and also leads, in the case of systems which have a kinematically tree-like structure, direct to a simple differential equation in the minimal coordinates. As a special case the relative coordinates represent the absolute coordinates as free motion of a body relative to the inertial system.
- Restrictions in the bodies relative motion through joints can be expressed by *implicit* or *explicit* formulation of the constraints.

- Superimposed elastic deformations of the mass-affected bodies which are assumed to be *linearizably small* are permissible.
- Complete Taylor expansion to terms of second order for the elastic kinematics.
- The amount of work required to calculate the right side of the equations of motion grows only linear by the number of bodies in the system. Here the kinematic structure characteristics of the mechanical system are used (see Fig.2).
- The numerically computation instructions for kinematically branched systems can be parallelized.
- The memory requirement increases only linear with the number of bodies in a system (see Fig.2).
- Short and compact representation in theory and in realization by computer code through recursive formulation of kinematic and dynamic relationships.
- Simultaneous computation of constraint forces and accelerations.
- Kinematically closed chains are allowed. However, with an increasing number of kinematically closed chains and relatively few degrees of freedom the efficiency of the equations for time integration decreases.

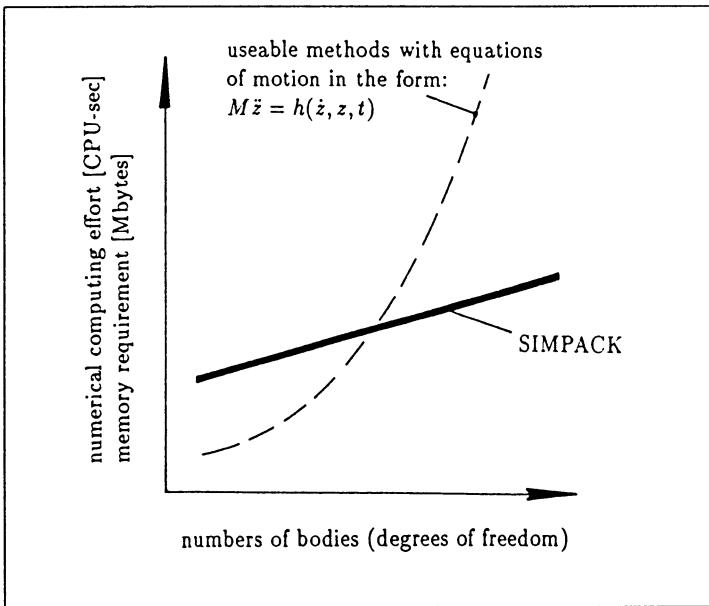


Figure 2: Increase in computation effort and memory requirement in relation to the number of bodies in a system

It should be emphasized that thanks to the short and compact description of all steps to calculate the equations of motion, constraint forces and accelerations, the choice of formalism alone leads to easy maintainability of the program.

3 Structure of the SIMPACK Program Package

Fig. 3 gives a schematic description of the program's structure. The core of the program is the SIMPACK formalism, which comprises the recursive computation instructions for formulation of the equations of motion. The individual matrix operations themselves are described in the block thereunder.

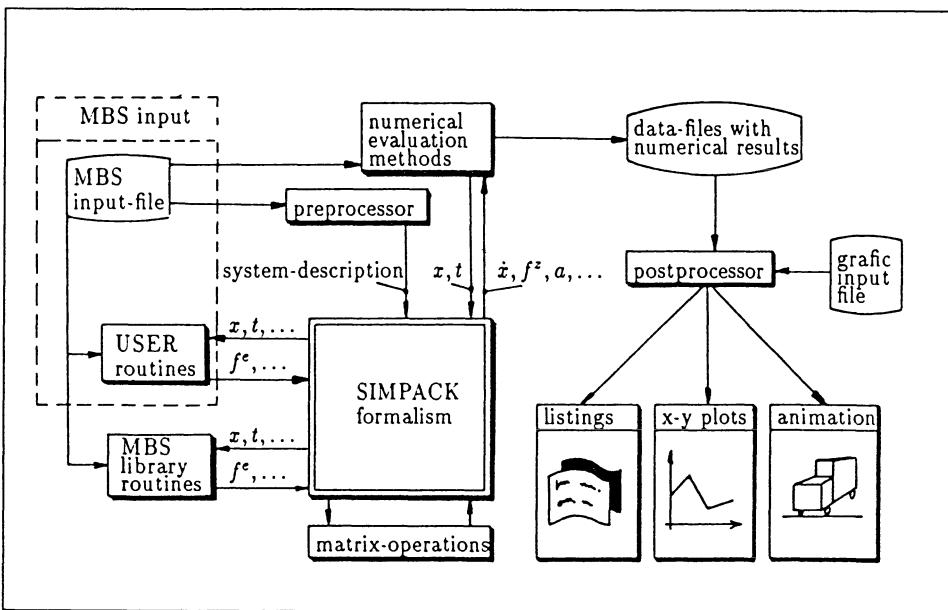


Figure 3: Schematic structure of SIMPACK program package

Input files

The formalism is served by input files and by subroutines (USER routines) which allows the user to increase the SIMPACK functionality. There are two kinds of input files: the multibody system input file (MBS input) and — for the experienced user — a file which contains functional histories in the form of value tables. These functional histories can later be used in USER routines, thereby allowing communication with program packages used independently of SIMPACK (e.g. path planning programmes for robots).

The MBS-input file essentially contains the description of the mechanical system, e.g. masses and inertia moments of individual bodies or information on connections between the bodies (e.g. rotational joints or springs). A simple *description language*, briefly touched on in section 5, has been created to define the mechanical system. Input of the mechanical system via a file has been chosen because a large number of data has to be defined. Experience has shown that this can be done much easier and with fewer mistakes by using the user surface of the computer (e.g. editor, multi-windowing) rather than through program-controlled dialogue. The *input language* allows the user great freedom in setting up the input file to suit his own taste and to use it for documentation of individual models.

Besides the description of the mechanical system, control-data for the numerical methods can also be fed into the MBS-input file, e.g. control-data for a time integration of the equations of motion.

Preprocessor

After the MBS-input data have been read in by the program they are prepared for the formalism by the preprocessor. Here the data are checked, as far as possible, for logical correctness. Input errors are disclosed to the user by an error code.

Library routines and modelling elements

The program contains a whole series of library and modelling elements, which form an integral part of the program. These library elements contain functional syntax such as the law of force for a linear compression-tension spring. Below is a brief survey of the main modelling elements.

- Joints:

- Fastening bolt as zero-freedom joint. This serve to calculate the interacting force at a given point on a rigid body.
- Single axis rectilinear sliding pairs and hinges (1 degree of freedom).
- Universal joints (2 degrees of freedom).
- Ball-and-socket joints with Eulerian angles or Cardan angles (3 degrees of freedom).
- Joints with two translations and a rotation for free-plane movements (3 degrees of freedom).
- Universal joint on which one rotation is rheonomous. Used for example for predetermined steering angle in motor vehicle simulations (1 degree of freedom).
- Free relative movement with three rotations (Cardan or Eulerian angles).
- General joint: the user determines the number and sequence of rotations and translations (0 — 6 degrees of freedom).

- Force elements:

- Articulated tension-compression springs with parallel damping elements. The force laws are linear.
- Articulated force elements with spring and damper in series.
- Spatial rubber elements with linear spring and damper characteristics.
- Force elements for speed-dependent friction with non-linear characteristic curve (force jump on adhesion)
- Electrical drive with control and downstream transmission. The control laws are of the type PIDT1.
- Tyre models (see Fig.4) after:
 - * H.B. Pacejca [1] with the Similarity Method.
 - * E. Bakker, L. Nyborg, H.B. Pacejca [2] with the *fit* of plotted curves.
- Routines for the description of road-surface unevenness in motor vehicle simulations.
- Downstream transmission.

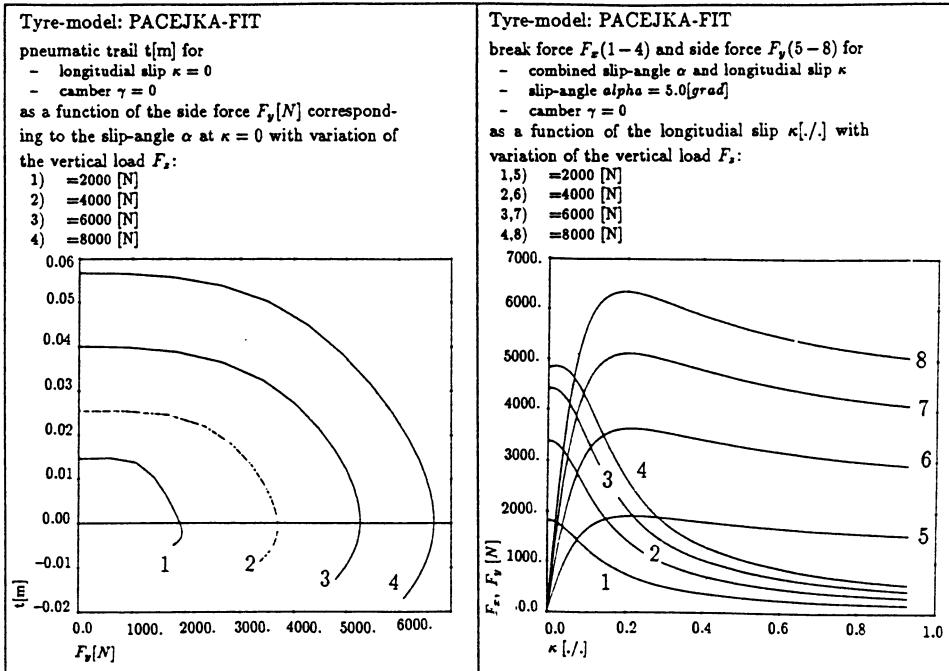


Figure 4: Some typical tyre characteristic curves

- Algebraic constraints as root-finding functions to characterize discontinuities of laws of force.
- Time functions for predetermined motion of moving reference frames:
 - Translation and rotation in and around an axis at constant speed.
- Routines to describe the relative motion of coupling points on the respective bodies as a function of state.

USER routines

The above library elements can be extended by USER routines. This possibility, besides the file input, is a particularly important program input form. Whereas through file input only constant parameters such as the mass of a body can be reported to the program, the USER routines extend the program's functionality. To demonstrate the great flexibility of the USER routines the interface definition for the USER routines for a law of force is explained here as an example.

Input through the parameter list is:

- Any number of kinematic measurements (sensors) between any points of a multibody system. These are defined in a *measuring field* in the input file.
- any number of time-constant parameters.
- any number of function histories available in form of tables of values defined in the FKT-input file. These tables also can contain the coefficients of a spline interpolation for the function histories.
- all state coordinates of the eigendynamic of this force element.
- all state coordinates of the multibody system: $t, z_G, \dot{z}_G, z_E, \dot{z}_E, z_F$

law of force

Output through the parameter list is:

- The applied forces f^e and/or applied torques l^e .
- First derivative of the coordinates \dot{z}_F describing the eigendynamic of the force element.
- Intermediate values obtained during calculation of the forces and which will be needed later for interpretation of the results (e.g. the slip angle of a road vehicle tyre during simulation).

Evaluation methods

One programming level above the formalism are the numerical evaluation methods located. These are at present:

- For a given state, i.e. an initial state of the system, all the results of the formalism are calculated, which are
 - the second derivative of the positional coordinates,
 - the constraining forces and torques of the joint,
 - the applied forces and torques,
 - intermediate values of the laws of force for the applied forces,
 - kinematic relative measurements (sensors for position, speed and acceleration).

This evaluation method serves especially as control of the model and self-written USER routines for input errors.

- The non-linear equations are linearised around a given state and/or a given initial position and initial speed. The **eigenvalues** and **eigenvectors** of the linear system matrix are calculated.
- For the linear system analysis a **parameter variation** with eigenvalue tracing can be carried out, e.g. for computation of the runup performance of rotors.

- The most important evaluation method is the numerical time-step integration.

The user can chose among the following mathematical integration methods at present:

- Euler method.
 - Runge-Kutta method.
 - Methods from the ODEPACK library [3]. These are methods with variable order and variable incrementation control. These methods choose, automatically or in accordance with a user input, a Gear or Adams method for the integration. The user either determines whether the differential equation to be integrated is *stiff* or *non-stiff* or the integrator chooses the correct method.
- Times at which discontinuities occur in the differential equation can be fed to the integrator, either explicitly or in the form of implicate algebraic root-finding functions. For SIMPACK library routines with discontinuities the root-finding functions are firmly implemented. For force elements written by the user they have to be given as USER routines.
- ODASSL [4] as an integration method for differential algebraic equation systems. Explicitly known discontinuities can be reported to this method. ODASSL alternatively permits an integration with:
 - * a standard-corrector,
 - * a nonlinear least-square method as corrector,
 - * a constrained nonlinear least-square method as corrector,
 - * an implizit state space method as corrector,
 - * or a Baumgarte-stabilisation.
 - With the aid of integration interface routines the above methods can be extended by user's own integration routines.

- For the integrated state-coordinates all the formalism's results are computed.
- If the mechanical model contains force elements for motor-vehicle tyres, tyre identification fields can be calculated using the parameters in the input file characterizing the law of force. This allows a check of the tyre parameters input for the actual model.

Output of computer results

Results of the numerical evaluation methods are written to data files under the model name. Various postprocessors have access to these data files and can reproduce the results as columns of figures, as x-y plots or as 3D animation. An integral feature of SIMPACK is a wide range of postprocessors which allow presentation of the results in various ways to facilitate interpretation.

x-y plots

The postprocessors for the x-y plots are **dialogue-controlled**. They have a momentary performance range as follows:

- Representation of the function histories which are calculated outside of SIMPACK, e.g. by a preliminary program, before being passed to SIMPACK in the form of value pairs. Such function histories are, e.g.:
 - predetermined trajectories for the servomotors of robots,

- road histories,
- non-linear spring characteristics.

to mention just a few possibilities.

- Representation of typical identification fields for motor-vehicle tyres, which are actually defined within a vehicle model.
- For the integration results time-history plots are generated for
 - state values,
 - statistical information of the integration method, such as:
 - * the actual step-size,
 - * the actual used order of the integration method,
 - * times of discontinuities taken into consideration of the integrator, ...
 - the kinematic results for position, speed and acceleration,
 - histories of constraining forces and torques,
 - histories of the applied forces and torques,
 - intermediate values for the laws of force elements
- To trace the eigenvalues of a parameter variation, eigenfrequencies and eigendamping can be shown over the varied parameter or as a root locus plot.

Animation

Besides the 2D graphics it also is absolutely essential for interpretation of the computer results to present simulated motion sequences in 3D animation. This gives a plausibility control of the results in a visual and easily interpretable form.

The postprocessor used for this purpose needs a further input file in which the spatial form of the individual bodies is described. To this end the postprocessor is fitted with modules for definition of basic bodies such as cubics, cylinders, truck cabs, roads, etc. With the aid of this modules relatively complex systems, such as a truck, can be built up in a short time with a minimal number of inputs.

The 3D postprocessor's functionality comprises:

- Representation of individual component groups. This is generally used as a control of the supplementary graphic inputs which are needed to give *abstract* bodies (defined by mass and inertia tensor) a three-dimensional form.
- The motion histories can be shown as on-line animation.
- Since efficient graphic work stations are needed for this, small systems can compute the pictures and store them as pixel data. The stored pictures can then be reproduced as animation.
- Various states of motion can be shown in one picture.

The various kinds of pictorial representation are:

- Wire models, wire models with hidden lines, or planar models with hidden surfaces and shading.

- Projection is perspective or axonometric, depending on whatever is needed for interpretation.

Fig. 1 shows printouts of a 3D postprocessor. Fig. 5 shows the buildup of truck component groups from basic bodies. The individual groups of components can move relatively to one another.

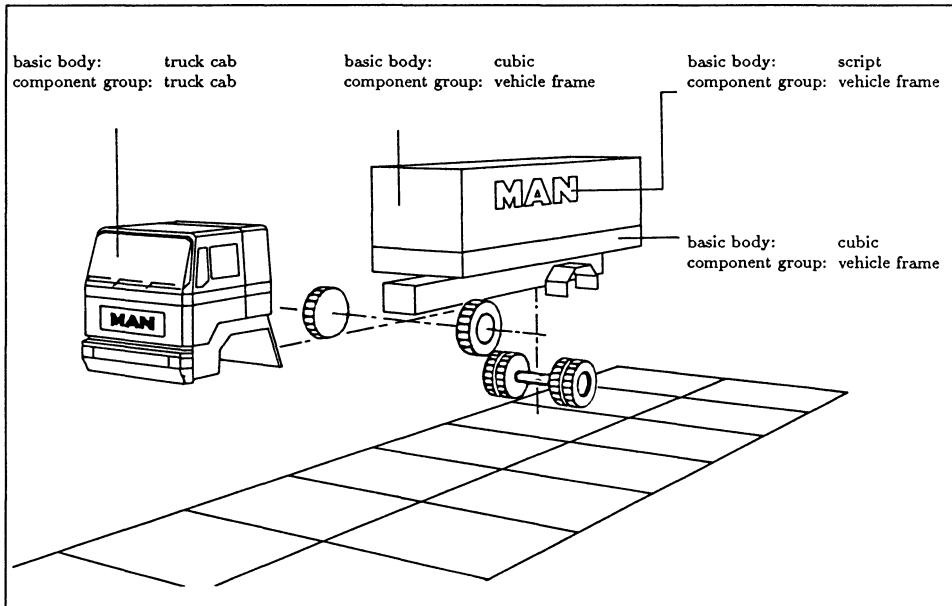


Figure 5: Buildup of the truck component groups

4 Use of SIMPACK for various Applications

Since the user is able to intervene at many points in the program, SIMPACK is very adaptable to various user environments and many applications. The various possibilities of intervention comprise the above-described **USER routines** for mechanical modelling elements by which SIMPACK's functionality is extended, connection of the user's own **integration methods**, and the possibility of exchanging the **input routines** for system description and for graphical definitions of the 3D animation. This latter extension allows the user to use the data already available in a user's own CAD program.

5 Example for Input of System Description

The form in which with the aid of *key words* the model is described is indicated briefly. Fig. 6 shows a sketch of two bodies coupled by a spring. The left-hand body has the number 17 and the right-hand one the number 1. On body 17 is the observation point 20, and on body 1 the

6 Theoretical Background to SIMPACK - the Formalism

6.1 The basic Idea

The basic idea the formalism is based on has already been documented [5],[6],[7],[8] and is extended to include systems of any structure type and with elastic bodies in RULKA [9]. To explain the formalism a system having a purely chain-like structure is considered first of all. In this system the bodies and joints of are consecutively numbered outwards with $i=1,NK$.

The kinematic of this system is described by *relative* coordinates. That is the kinematic relative vectors via a joint are expressed by the minimal coordinates z_j of the joint between the body i and the body $j=i+1$. E.g. the relative position vector:

$$\vec{r}_{ij} \equiv r_{ij}(z_j, t). \quad (1)$$

With this formula it is possible to build up recursively the kinematics of a system having a tree-like structure.

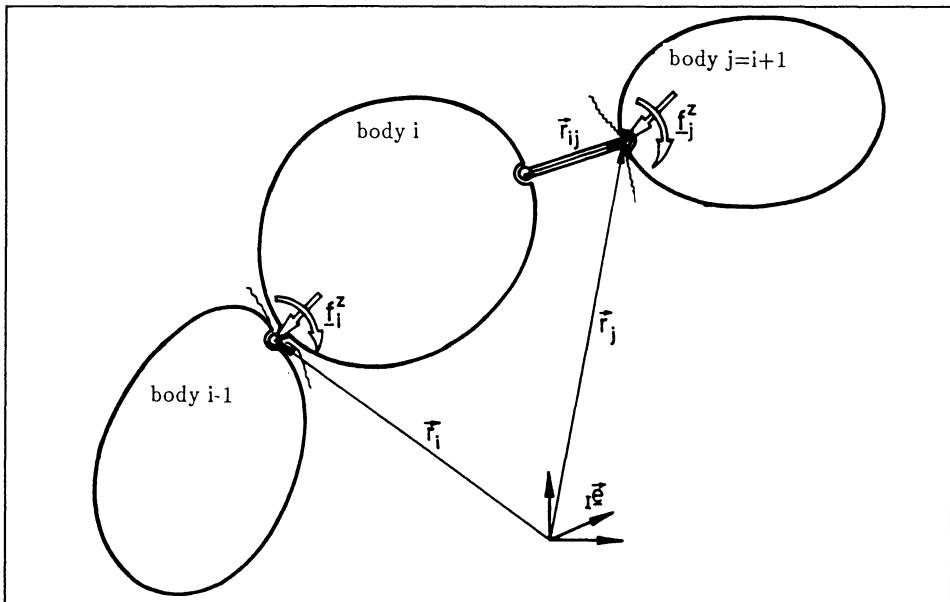


Figure 7: bodies i and $j=i+1$ in a chain

Recursively means, in this context, that the absolute values of the outer body j are built up as a function of the absolute values of the inner body i and the relative values of the joint between body i and body j . For example the 6×1 vector of the acceleration of body j is written recursively as follows:

$$a_j = C_j a_i + J_j(z_j, t) \ddot{z}_j + b_j(\dot{z}_j, z_j, t) \quad (2)$$

with the 6×6 geometric matrix C_j and the $6 \times NF_j$ joint-Jacobian matrix J_j . NF_j is the number of degrees of freedom of joint j .

observation point 19. The spring has the force element number 8 and is connected to the two bodies at observation points 20 and 19.

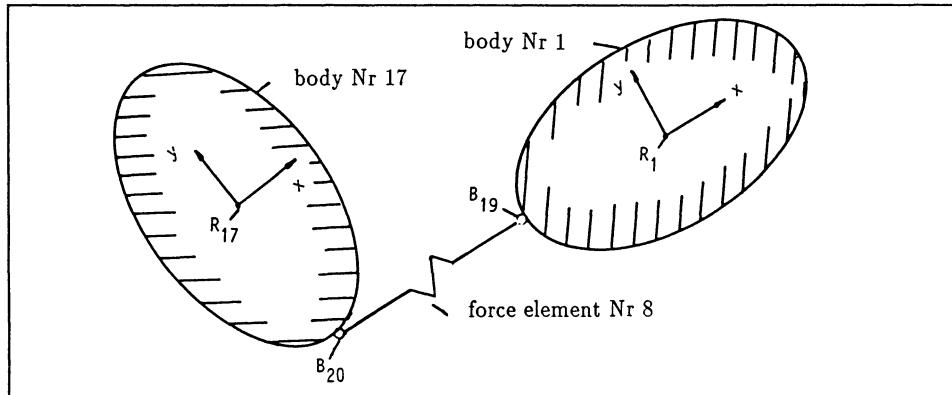


Figure 6: Sketch of a mechanical system

Description of the mechanical system is done by means of value assignment to *key words*. The following text is an excerpt from the input file for system description of the above-depicted mechanical model. The character "!" marks the beginning of a commentary.

```

! Definitions of the masses:
! -----
MASS(17) = 10. ! [kg] mass of body with the number 17
MASS( 1) = 7. ! [kg] mass of body with the number 1

! Definitions of observation points:
! -----
OBS_BODY(20) = 17           ! number of body on which
                            ! observation point 20 lies
OBS_RRO( 20) = 0., -1., 0. ! position vector of observation
                            ! point 20 on body 17
OBS_BODY(19) = 1           ! number of body on which
                            ! observation point 19 lies
OBS_RRO( 19) = -1., 0., 0. ! position vector of observation
                            ! point 19 on body 1

! Definitions of the force elements:
! -----
FEL_TYPE( 8 ) = 2          ! type of force element with the number
                            ! 8 is 2. This is a compression-tension
                            ! spring with linear characteristic curve
FEL_OBSI( 8 ) = 20         ! number of connection point i.
FEL_OBSJ( 8 ) = 19         ! number of connection point j.
FEL_PAR (1,8) = 10. ! [N/m] spring stiffness = 1st parameter
                    ! for force element 8.
FEL_PAR (2,8) = .5 ! [m] unstretched length of spring = 2nd
                    ! parameter for force element 8.

```

Under the influence of external forces the motion of a body is determined by the principles of linear and angular momentum. For a system having a chain-like structure these equations are written for the body i in matrix form as:

$$M_i(z, t) \mathbf{a}_i = h_i(\dot{z}, z, t) + f_i^z - C_j^T f_j^z. \quad (3)$$

In this equation M_i is the 6×6 inertia matrix of the body i , the 6×1 vector h_i contains the applied forces and the Coriolis forces acting on the body i , and f_i^z are the constraint forces and torques on body i (see Fig. 7).

In these equations the Cartesian coordinates of the constraint forces of a joint i can still be expressed as:

$$f_i^z = N_i \lambda_i \quad (4)$$

with the $6 \times N\lambda_i$ distribution matrix N_i and the $N\lambda_i = 6 - NF_i$ generalized constraint forces λ_i of the joint.

For a chain consisting of NK bodies equations (2) and (3) can be used as $12NK$ equations for the $12NK$ unknown quantities

$$\mathbf{a}_i, \ddot{\mathbf{z}}_i, \lambda_i \quad i = 1, NK,$$

if one bears in mind that at the beginning of a chain the acceleration a_0 of the moving reference frame is given and that at the end of a chain the constraint forces f_{NK+1}^z trivially are zero, since no further body follows. While the $12NK$ equations depend non-linear on state they are however linear in the unknown quantities.

The $12NK$ linear equations (2) and (3) can be resolved recursively by taking advantage of the theorem that the generalized constraint forces of a joint i are orthogonal to the free movements in the joint i (e.g. SCHWERTASSEK [10])

$$J_i^T N_i \equiv 0 \quad (5)$$

and by using the kinematical structural characteristics of the mechanical system. As ansatz is used

$$\ddot{\mathbf{z}}_i = P_i(z, t) \mathbf{a}_{i-1} + p_i(\dot{z}, z, t) \quad (6)$$

$$f_i^z = K_i(z, t) \mathbf{a}_{i-1} + k_i(\dot{z}, z, t) \quad (7)$$

$$\mathbf{a}_i = C_i \mathbf{a}_{i-1} + J_i(z_i, t) \ddot{\mathbf{z}}_i + b_i(\dot{z}_i, z_i, t) \quad (8)$$

in which the values $\ddot{\mathbf{z}}_i$, f_i^z , \mathbf{a}_i are expressed as linear functions of the absolute acceleration of body $i-1$.

By substituting the equations (6),(7),(8) and (2) in (3) one obtains the recursive calculation rules for the still unknown matrices P_i, p_i, K_i, k_i in (6) and (7). For details see [9].

The result of the above procedure is the differential equation for the mechanical system

$$\ddot{\mathbf{z}} = g_1(\dot{z}, z, t) \quad (9)$$

and the constraint forces and accelerations

$$f_i^z = g_2(\dot{z}, z, t) \quad (10)$$

$$\mathbf{a}_i = g_3(\dot{z}, z, t) \quad (11)$$

as functions of the input values \dot{z}, z, t . The relations can be evaluated by a forward, backward and a second forward recursion. They can be summarized as follows:

Input:

state variables: \dot{z}, z, t

Forward recursion:

do i=1,NK

$$\begin{aligned} C_i &= g_1(z, t) & \text{dim} &= 6 \times 6 \\ J_i &= g_2(z, t) & \text{dim} &= 6 \times NF_i \\ b_i &= g_3(\dot{z}, z, t) & \text{dim} &= 6 \times 1 \\ M_i &= const & \text{dim} &= 6 \times 6 \\ h_i &= g_4(\dot{z}, z, t) & \text{dim} &= 6 \times 1 \end{aligned}$$

end do

Backward recursion:

Starting values: $K_{NK+1} \equiv 0, k_{NK+1} \equiv 0, C_{NK+1} \equiv 0$

do i=NK,NK-1,...,1

$$\begin{aligned} \bar{M} &= M_i + C_{i+1}^T K_{i+1} & \text{dim} &= 6 \times 6 \\ \bar{h} &= h_i - C_{i+1}^T k_{i+1} & \text{dim} &= 6 \times 1 \\ \hat{M} &= J_i^T \bar{M} J_i & \text{dim} &= NF_i \times NF_i \\ P_i &= -\hat{M}^{-1} J_i^T \bar{M} C_i & \text{dim} &= NF_i \times 6 \\ p_i &= +\hat{M}^{-1} J_i^T (\bar{h} - \bar{M} b_i) & \text{dim} &= NF_i \times 1 \\ K_i &= \bar{M}(C_i + J_i P_i) & \text{dim} &= 6 \times 6 \\ k_i &= \bar{M}(b_i + J_i p_i) - \bar{h} & \text{dim} &= 6 \times 1 \end{aligned}$$

end do

Forward recursion:

Starting values: $a_0 \equiv 0$.

do i=1,NK

$$\begin{aligned} \ddot{z} &= P_i a_{i-1} + p_i & \text{dim} &= NF_i \times 1 \\ a_i &= C_i a_{i-1} + J_i \ddot{z}_i + b_i & \text{dim} &= 6 \times 1 \\ (f_i^z &= K_i a_{i-1} + k_i) & \text{dim} &= 6 \times 1 \end{aligned}$$

end do

6.2 Extension to general kinematic Structures

General tree configurated Systems

The equations obtained above can be extended without problems to a general tree configurated system (see Fig. 8) by analogous interpretation of the body indices i and j=i+1 numbered from inside outwards.

Here it should be particularly mentioned that in the case of a general tree configurated system the computation rules relating to bodies in branches lying parallel to one another are independant, and thus they can be processed parallel to one another.

Within the realized program, the necessary sequencing is arranged for a parallel processing.

Kinematically closed Chains

In the case that the mechanical system has kinematically closed chains the system is first converted into a kinematically tree-like structure by cutting NS joints. This basic system is treated exactly as described above, the additional values λ_S for the generalized constraint forces of the chain-closing joints are additionally appearing in the equations.

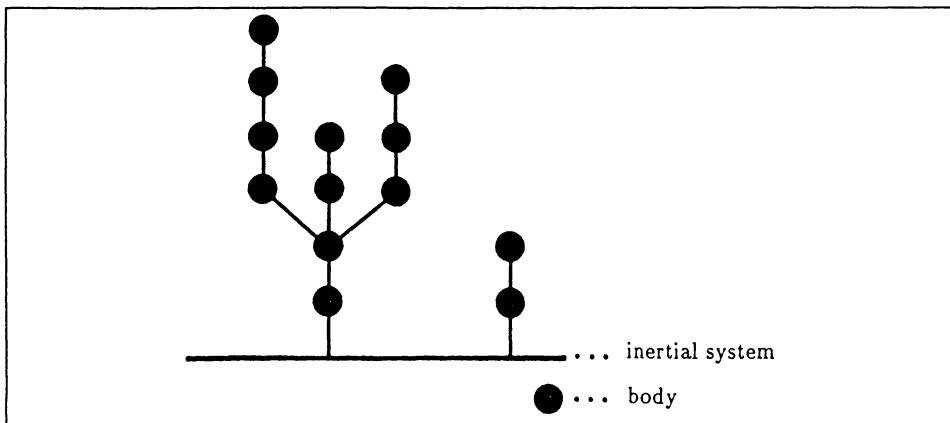


Figure 8: general tree configurated system

To determine these additional values the implicit formulation of the constraints of the chain-closing joints are used. To this end the kinematic relative vectors over a chain-closing joint are calculated, e.g. the position vector over this joint:

$$r_{lk} \equiv g_4(z, t)$$

with z being the minimal coordinates of the system with a tree-like structure.

The implicit constraints of the joint are $N\lambda_S$ algebraic equations as a function of the relative vectors over this joint, if this joint is restricting $N\lambda_S$ movements. The implicit constraints are formulated as algebraic equations for position, velocities and acceleration respectively for all chain-closing joints.

Thus to describe the dynamics one obtains a coupled differential algebraic system in the form:

$$\begin{aligned} \ddot{z} &= g_1(\dot{z}, z, t, \lambda) & ; \dim = NF \times 1 \\ g_2(z, t) &= 0 & ; \dim = N\lambda \times 1 \\ g_3(\dot{z}, z, t) &= 0 & ; \dim = N\lambda \times 1 \\ g_4(\ddot{z}, \dot{z}, z, t) &= 0 & ; \dim = N\lambda \times 1 \end{aligned}$$

where NF is the number of minimal coordinates in the system with a tree-like structure. $N\lambda$ is the number of chain-closing constraints in the system.

In the mathematical sense the above equation is a redundant differential algebraic system and can be correctly solved by using appropriate methods (FÜHRER [4]).

Besides chain-closing joints other joints can also be used in implicit form, however in this case the advantages of recursive formulation are less evident in respect of computing time.

6.3 Incorporation of elastic Bodies in the multibody System

Besides rigid bodies, which can make large movements, elastic deformations are also admissible for mass-affected bodies. Such deformations are assumed to be "*small*" in relation to their

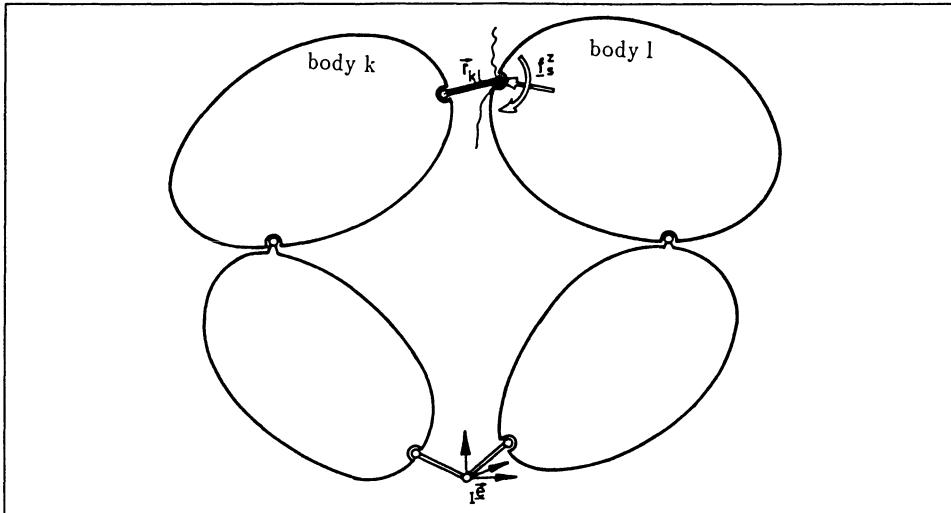


Figure 9: Constraining forces of a chain-closing joint

undeformed position. To describe elastic deformations the position vector r_{el} from the body-fixed reference frame to a mass point on this body is first formulated generally as a function of the elastic deformation coordinates v_{el} . This formulation contains the inner constraints on elastic deformation, such as the unbending cross section to neutral fibres on bending of a beam. In general it gives the non-linear syntax:

$$r_{el} \equiv r_{el}(v_{el}) \quad (12)$$

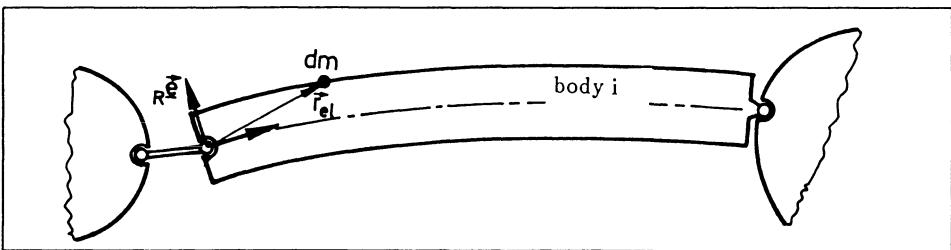


Figure 10: Elastically deformed body

In order to obtain a simple differential equation later a Ritz formulation is used for the deformation coordinates:

$$v_{el} = u^T(x) z_{el}(t) \quad (13)$$

with the given shape functions $u(x)$. Assuming small elastic deformations $z_{el} \ll 1$, the elastic movements are then linearized as

$$r_{el} \cong {}^0 r_{el} + {}^0 R z_{el} + {}^1 R(z_{el}) z_{el} \dots \quad (14)$$

It is stressed here that linearization of the elastic movements were carried out completely to terms of the second order, and consequently **complete** linear motion equations are available for the elastic motions.

Neglecting the terms of the second order, especially in connection with *great* inertia loads or *great* external preloads, would lead to physically wrong results completely contrary to actual practice.

The necessary functions $u(x)$ in (13) may:

- originate from measurements ,
- represent eigenforms from finite element calculations,
- or be mathematical functions (e.g. cubic spline functions).

By laying the body-fixed reference frame e_R of the elastic body in the joint connection point (see Fig.10) it is possible to assure the kinematically boundary conditions to every time step with a special choice of the functions $u(x)$. (This is a presupposition for the convergence of the Ritz formulation). In the sample, shown in Fig.10 the boundary conditions are

$$r_{el}(x = 0) \equiv 0 ; \quad r_{el}''(x = 0) \equiv 0$$

and are needing functions $u(x)$ in the Ritz formulation with the conditions:

$$u(x = 0) \equiv 0 ; \quad u''(x = 0) \equiv 0.$$

Equation (2) for the kinematic relationships in multibody systems, extended by the elastic deformation, reads:

$$a_j = C_j(z_{el,i}) a_i + J_{el,i}(z_{el,i}) \ddot{z}_{el,i} + J_j(z_j) \ddot{z}_j + b_j(\dot{z}, z, t) \quad (15)$$

and the kinematic equation (3) with the momentum principles:

$$\begin{bmatrix} M_S & H^T \\ H & M_{el} \end{bmatrix} \begin{pmatrix} a_i \\ \ddot{z}_{el,i} \end{pmatrix} = h_i + \begin{pmatrix} f_i^z \\ 0 \end{pmatrix} - \begin{bmatrix} C_i^T \\ J_{el,i}^T \end{bmatrix} f_j^z \quad (16)$$

The equations to define the matrices M_S , H , M_{el} and J_{el} are given in RULKA [9] or JOHANNY [11]. Using the same argumentation as for purely rigid body systems an extended recursive ansatz is employed to obtain the differential equation (9):

$$\ddot{z}_i = P_i a_{i-1} + P_{el,i} \ddot{z}_{el,i-1} + p_i \quad (17)$$

$$\ddot{z}_{el,i} = Q_i a_{i-1} + Q_{el,i} \ddot{z}_{el,i-1} + q_i \quad (18)$$

$$f_i^z = K_i a_{i-1} + K_{el,i} \ddot{z}_{el,i-1} + k_i \quad (19)$$

$$a_i = C_i a_{i-1} + J_{el,i-1} \ddot{z}_{el,i-1} + J_i \ddot{z}_i + b_i \quad (20)$$

Substitution of equations (17)–(20) and of (15) in (16) then gives the computation rules for recursive determination of matrices P, K, Q , etc. For details see [9].

Particularly when elastic bodies are present the advantages of recursive kinematics and recursive kinetics become apparent. Here too for evaluation of the right side they only lead to a linear increase in computation effort and memory requirement with increasing system size.

7 Further Developments of SIMPACK

- While the matrices for elastic deformation are being implemented within the recursion rules in SIMPACK, a preprocessing program is in development, which will compute the integral matrices for the shape-functions of the Ritz method. These matrices, needed as SIMPACK input, are e.g. the modal mass matrix M_{el} or the coupling-matrix H through the elastic deformations and the *large* accelerations a_i in (16) taking into account the terms of the 2nd order.
- Likewise in processing is the implementation for kinematic analysis of systems with kinematically closed chains, and the implementation of the inverse kinetic and the presentation of their results.
- Planned is realization of parallel processing on suitable computers such as transputers, the necessary relationships being already available in SIMPACK.
- SIMPACK with its various user routine applications is being continually extended. This comprises, for example, implementation of the kinematic relationships for special wheel suspension systems as standard joints.

8 Results of the Test Examples

Fig.11 shows the time history of the angle γ as simulation result of SIMPACK for the *7 body mechanism*. Fig.12 shows the time history of total of the constraint force in the revolute joint, connecting point B and body 3 of this mechanism.

The state of the mechanism at the times $t = 5, 45, 75$ and 105 [msec] is presented in Fig.13. For the robot example an animation is given in Fig.14.

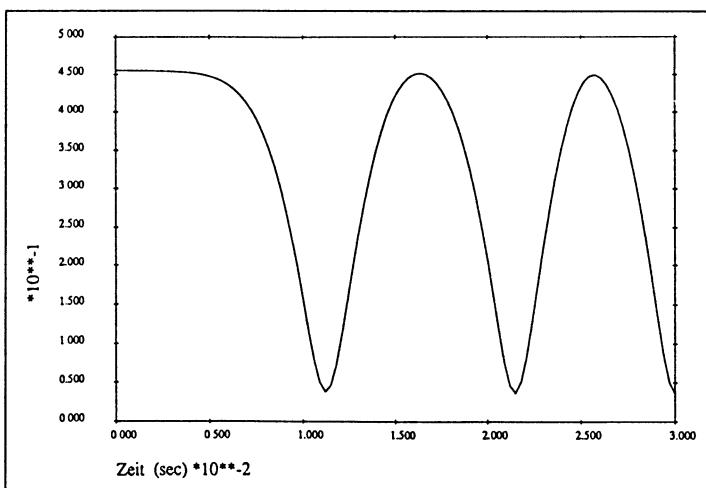


Figure 11: Time history of angle γ of the mechanism

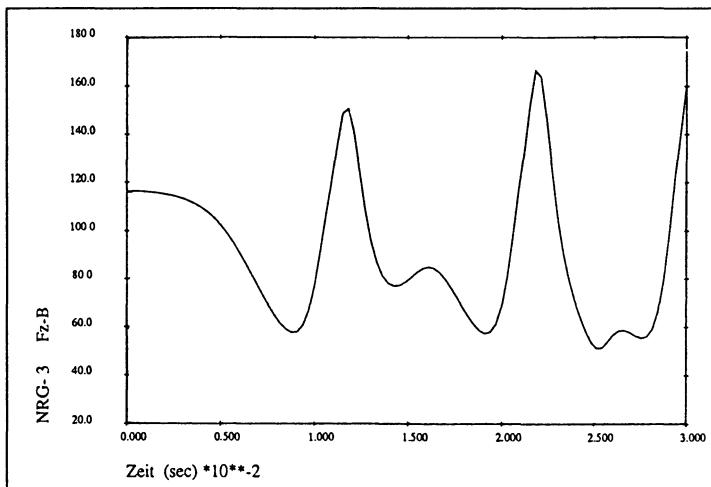


Figure 12: Time history of x-coordiante of the constrained force in the revolute joint, connecting point B and body 3

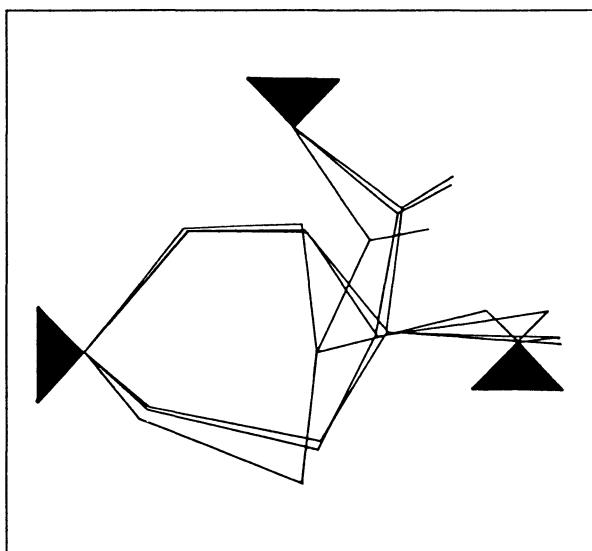


Figure 13: Animation of the mechanism motion

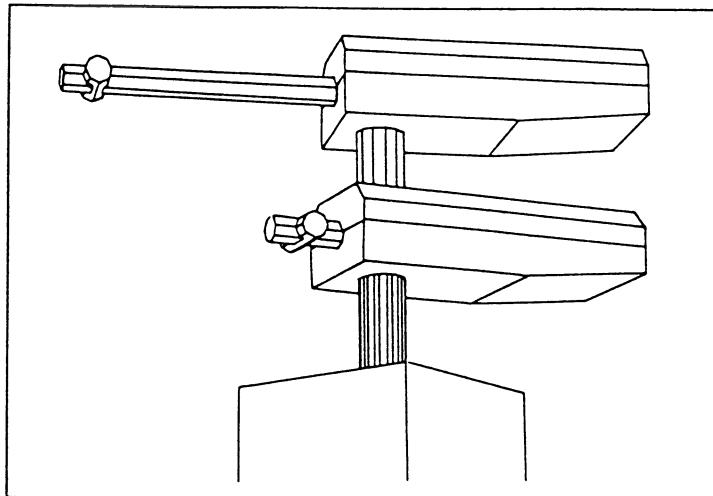


Figure 14: Animation of the robot motion

References

- [1] H.B. Pacejka: *Modelling of the Tyre as a Vehicle Component with Applications*. Technical Report, CARL-CRANZ-Gesellschaft, 1982. Course V2.01.
- [2] E. Bakker, L. Nyborg, and H.B. Pacejka: Tyre modelling for use in vehicle dynamic studies. *SAE Technical Paper Series 870421*, 1987.
- [3] A.C. Hindmarsh: ODEPACK a systematized collection of ODE solvers in scientific computing. *North Holland, Amsterdam, IMACS Transactions on Scientific Computation*, Vol. 1, pp 55-64, 1983.
- [4] C.Führer: *Differential-algebraische-Gleichungssysteme im mechanischen Mehrkörpersystem - Theorie, numerische Ansätze und Anwendungen*. PhD thesis, TU-München, 1988.
- [5] Vereshchagin: Computer simulation of the dynamics of complicated mechanisms of robot-manipulators. *Engineering Cybernetics*, No 6, pp 65-70, 1974.
- [6] W.W. Armstrong: Recursive solution to the equations of an n-link manipulator. *Theory of Machines and Mechanisms*, Vol 2, ASME, 1979.
- [7] R. Featherstone: The calculation of robot dynamics using articulated body inertias. *International Journal of Robotic Research*, Vol 2, No. 1, 1983.
- [8] H. Brandl, M. Otter, and R. Johanny: A very efficient algorithm for the simulation of robots and similar multibody systems without inversion of the mass matrix. *IFAC Wien*, 1986.
- [9] W. Rulka: *Bewegungsgleichungen für hybride mechanische Systeme mit freiheitsgradproportionalem Rechenaufwand*. Technical Report, DFVLR Oberpfaffenhofen, 1986. Interner Bericht IB 515-86/12.
- [10] R. Schwertassek and R. Roberson: A state-space dynamic representation for multibody mechanical systems, part 1 and part 2. *Acta Mechanica* 50/51, 1984.
- [11] R. Johanny: *Automatisches Aufstellen der Bewegungsgleichungen von baumstrukturierten Mehrkörpersystemen mit elastischen Körpern*. Master's thesis, Lst.B. für Mechanik TU-München, 1984.

COMPAMM - A Simple and Efficient Code for Kinematic and Dynamic Numerical Simulation of 3-D Multibody Systems with Realistic Graphics

J.M. Jiménez, A. Avello, A. García-Alonso and J. García de Jalón
University of Navarra and CEIT, San Sebastián, SPAIN

ABSTRACT

The theoretical foundations of a 3-D multibody program called COMPAMM (COMPuter Analysis of Machines and Mechanisms) are presented. Instead of using Euler angles or Euler parameters in order to define the spatial orientation of a rigid body, COMPAMM uses the cartesian coordinates of two or more points and the cartesian components of one or more unit vectors rigidly attached to the body. With this coordinates the constraint equations are quadratic and then the jacobian matrix is a linear function of them, needing for its evaluation far less arithmetic operations than with other methods. In addition to this, the mass matrix in the inertial reference frame is constant and Coriolis or centrifugal forces do not appear in the formulation. COMPAMM has also very advanced interactive and graphical capabilities that are very briefly described in this paper. Finally some examples are presented.

1. INTRODUCTION

The kinematic and dynamic analysis of 3-D complex multibody systems presents a great interest in many areas of mechanical engineering, such as robotics, ground vehicles, spacecraft, machinery, biomechanics, etc. In the last two decades a great deal of research has been done, most of them summarized in refs. [1] and [2]. As a result, some general purpose programs such as IMP, ADAMS, DADS, NEWEUL, ... have been used successfully in industrial projects.

In the last few years, a great emphasis has been put on the efficiency of the methods of analysis. Kinematic simulation interactivity is a very desirable capability of any program, and in the dynamic case large systems of nonlinear differential equations must be integrated as fast as possible, even in real time [3].

Looking for this improved efficiency, several authors [4], [5] have developed symbolical formalisms to obtain the motion differential equations in closed form. When applicable, these formalisms have better performance than the fully numerical formulations, but until now the latter remain the only approach to a truly general dynamic simulation, able to deal with mechanisms of changing topology and with special factors such as Coulomb friction, backlash, impacts, etc.

COMPAMM (COMPuter Analysis of Machines and Mechanisms) is a general purpose 3-D multi rigid body package based on particularly simple, general and efficient original idea [6]-[9].

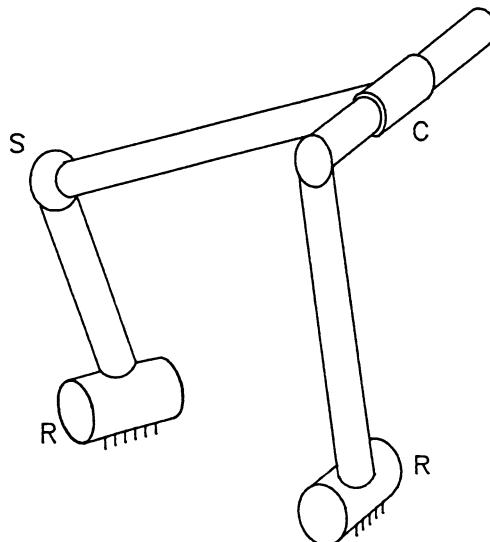


Figure 1 . RSCR 1 d.o.f Mechanism

In figure 1 a three-dimensional four bar (RSCR) mechanism with 1 degree of freedom is presented as an example. In order to analyze this mechanism it is necessary to choose a set of parameters (dependent coordinates) that unequivocally describe the position of the three moving links. This can be done in several ways, with different kinds of dependent coordinates. A first possibility is to use relative coordinates, that describe the position of a link relative to another link joined by a pair, using the coordinates of relative motion allowed by that pair. Another possibility is to describe the spatial position of each body with the cartesian coordinates of one of its points (usually the center of gravity) and with three of four parameters that describe the angular orientation (usually Euler angles or Euler parameters).

The dependent coordinates are related through the *constraint equations*. The partial derivatives of the constraint equations with respect to the dependent coordinates constitute the *jacobian matrix*. The jacobian matrix is very important because it determines in practice both the kinematic and the dynamic equations.

With relative coordinates the constraint equations arise from the closure condition of kinematic loops. With reference point coordinates the constraint equations arise from the limitations in relative motion between contiguous elements imposed by the pair that joins them.

In COMPAMM a new kind of dependent coordinates has been used: the position of a body is determined by the cartesian coordinates of at least two of its points and the cartesian components of at least one unit vector rigidly attached to this body. In figure 2 the modellization of the RSCR mechanism of figure 1 with these fully cartesian coordinates, that originally [6] were called *natural coordinates*, is shown .

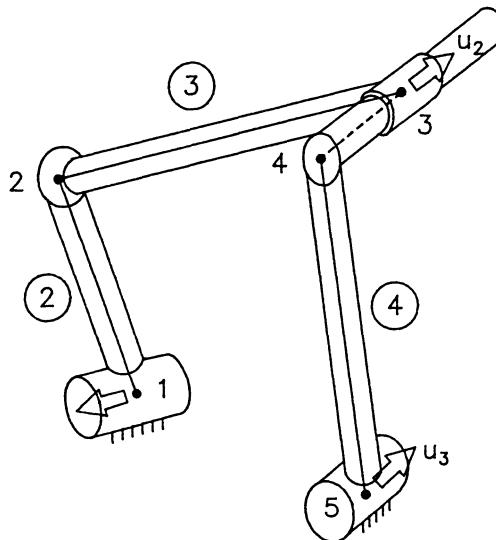


Figure 2. Mechanism RSCR modelled with natural coordinates

The example of figure 2 requires 4 relative coordinates, 18 (21) reference point coordinates with Euler angles (Euler parameters), and 12 natural coordinates. What first makes natural coordinates very attractive, in addition to their immediate geometrical

meaning, is the simplicity of the constraint equations that relate them, and, as a consequence, the simplicity of the jacobian matrix. For instance, for the example of Figure 2 the jacobian matrix with reference point coordinates and Euler parameters, evaluated according with reference [10], has 128 non-zero elements and requires nearly one thousand floating point operations; with natural coordinates the jacobian matrix has 57 non-zero elements and requires only twelve arithmetic operations, i.e. two orders of magnitud less.

2. CONSTRAINT EQUATIONS

The ultimate reason of the efficiency of natural coordinates can be found in the kind of constraint equations they determine. For instance, in the RSCR mechanism of figure 2 (and in most cases) the constraint equations arise in a double way. First, it is necessary that the points and vectors that belong to the same link move according to the rigid body condition. Second, it is necessary that points and vectors belonging to contiguous elements move according to the degrees of freedom of relative motion allowed by the pair that joins them.

Rigid body constraint equations can be introduced very easily. For instance, element 4 of the mechanism of figure 2 has two points (4 and 5) and two unit vectors (u_2 and u_3). There are 12 coordinates and 6 degrees of freedom, so 6 constraint equations must be found:

$$(x_4 - x_5)^2 + (y_4 - y_5)^2 + (z_4 - z_5)^2 - C_1 = 0 \quad (1)$$

$$(x_4 - x_5)u_{2x} + (y_4 - y_5)u_{2y} + (z_4 - z_5)u_{2z} - C_2 = 0 \quad (2)$$

$$(x_4 - x_5)u_{3x} + (y_4 - y_5)u_{3y} + (z_4 - z_5)u_{3z} - C_3 = 0 \quad (3)$$

$$u_{2x} u_{3x} + u_{2y} u_{3y} + u_{2z} u_{3z} - C_4 = 0 \quad (4)$$

$$u_{2x}^2 + u_{2y}^2 + u_{2z}^2 - 1 = 0 \quad (5)$$

$$u_{3x}^2 + u_{3y}^2 + u_{3z}^2 - 1 = 0 \quad (6)$$

Eq. (1) corresponds to a constant distance condition between points 4 and 5; eqs. (2), (3) and (4) correspond with constant angle conditions between segment 3-4 and vectors u_2 and u_3 ; eqs. (5) and (6) correspond to the unit module condition for vec-

tors u_2 and u_3 (in fact, eq. (6) is in this case unnecessary because u_3 is a fixed vector and so its components are not unknown).

A very interesting feature of natural coordinates is that points and unit vectors can be shared among two contiguous elements, contributing to the definition of the position of both and so keeping moderate the total number of coordinates. In addition to this, introduction of the constraint equations due to the pairs is strongly simplified, when not eliminated at all. For instance, the spherical pair in the mechanism of figure 2 does not introduce any equation because it is automatically introduced when links 2 and 3 share point 2. In the same way, as points 1 and 5 and vectors u_1 and u_3 are fixed and located on the axis of revolute pairs, both pairs neither introduce any constraint equation. On the other hand elements 3 and 4 share vector u_2 ; to introduce the constraint equations corresponding to pair C it is enough to impose the condition that points 3 and 4 (belonging respectively to links 3 and 4) remain permanently aligned with vector u_2 . Analytically this can be introduced through the vector product of vectors:

$$(u_2) \wedge ((r_3) - (r_4)) = (0) \quad (7)$$

(only two of these three equations are independent).

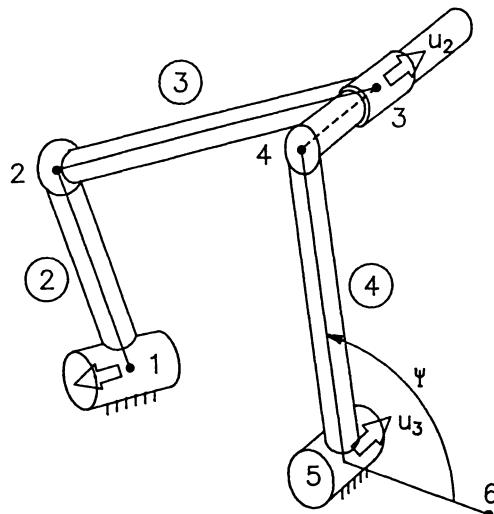


Figure 3. Mechanism RSCR with a driving angle

With natural coordinates angles are never necessary to describe the spatial position of a rigid body. However, sometimes the definition of an angle in a revolute joint can be very useful, for instance if the value of this angle need to be controlled by an actuator. In figure 3 the same RSCR mechanism of figure 2 is presented adding the angle ψ rotated by element 4. A new fixed point has been included. In order to include ψ among the dependent coordinates, the following new constraint equation, originated by the dot product of vectors, must be added,

$$(x_6 - x_5)(x_4 - x_5) + (y_6 - y_5)(y_4 - y_5) + (z_6 - z_5)(z_4 - z_5) - C_5 \cos\psi - C_6 = 0 \quad (8)$$

If the value of angle ψ is near ($n\pi$) radians ($n=0,1, \dots$) the dot product must be replaced by a component of the vector product of vectors.

With natural coordinates the constraint equations are always quadratic equations. As a consequence, the elements of the jacobian matrix are linear functions of natural coordinates; this is the reason of the extraordinary simplicity and efficiency of this formulation.

Inside COMPAMM constraint equations are formulated automatically from the mechanism position and topology.

3. KINEMATIC ANALYSIS

The constraint equations of a particular mechanism, both rheonomic and scleronomous, can be formulated symbolically as:

$$\{ \Phi(\{q\}, t) \} = (0) \quad (9)$$

where $\{q\}$ is the vector of natural coordinates.

The initial position (assembly) problem and the finite displacement problem, can be solved from the Newton-Raphson iterative scheme,

$$[\Phi_q (\{q\}_i, t)] [\{q\}_{i+1} - \{q\}_i] = -\{ \Phi (\{q\}_i, t) \} \quad (10)$$

from which a better approximation $\{q\}_{i+1}$ is obtained. $[\Phi_q]$ is the jacobian matrix.

For velocity and acceleration analysis it is enough to differentiate eq. (9). with respect to the time,

$$[\Phi_q((q), t)](\dot{q}) = - (\dot{\Phi}_t) \equiv (b) \quad (11)$$

$$[\Phi_q((q), t)](\ddot{q}) = - \{ \dot{\Phi}_t \} - \{ \dot{\Phi}_q \} (\dot{q}) \equiv (c) \quad (12)$$

In eqs. (10)-(12) the importance of the jacobian matrix for kinematic analysis is very clear.

4. DEPENDENT AND INDEPENDENT COORDINATES

For reasons that will be made evident later, it is very useful to formulate the dynamic equations in terms of a reduced set of F independent coordinates $\{z\}$ (F is the number of degrees of freedom of the mechanism). In this section the relationship between dependent $\{(q)\}$ and independent $\{z\}$ coordinates will be exposed. COMPAMM uses both types of coordinates.

A large family of methods [8], [11], [12] use as independent velocities $\{\dot{z}\}$ the projection of the dependent ones $\{\dot{q}\}$ on the rows of a constant ($F \times N$) matrix $[B]$.

$$[B] (\dot{q}) = (\dot{z}) \quad (13)$$

Putting together eqs. (11) and (13):

$$\begin{bmatrix} [\Phi_q] \\ [B] \end{bmatrix} \left\{ \dot{q} \right\} = \begin{Bmatrix} (b) \\ (\dot{z}) \end{Bmatrix} \quad (14)$$

If matrix $[B]$ is chosen in such a way that its rows are linearly independent of the rows of $[\Phi_q]$, it can be written:

$$(\dot{q}) = \begin{bmatrix} (\Phi_q) \\ [B] \end{bmatrix}^{-1} \begin{Bmatrix} (b) \\ (\dot{z}) \end{Bmatrix} \equiv [[S][R]] \begin{Bmatrix} (b) \\ (\dot{z}) \end{Bmatrix} = [S](b) + [R](\dot{z}) \quad (15)$$

This is a very important result and it can be taken as a definition of matrices $[S]$ and

[R]. Eq.(14) establishes that $\{\ddot{q}\}$ is the sum of a particular solution of eq. (11) plus the general solution of the homogeneous velocity equation. If the constraint equations are not time dependent, $\{b\} = \{0\}$, then only the second term of eq. (15) exists.

The matrix [R] defined in eq. (15) has a very great importance for the methods of dynamic analysis used in COMPAMM. It can be seen very easily that the column space of [R] is orthogonal to the row space of $[\Phi_q]$ and, as long as these columns are also linearly independent, they can be taken as a basis of the nullspace of $[\Phi_q]$, that is the space of allowable velocities with non time-dependent constraints. The independent velocities $\{\ddot{z}\}$ are the components of $\{\ddot{q}\}$ on the columns of [R], that are the basis vectors of the nullspace of $[\Phi_q]$.

For accelerations a similar result follows. Putting together eq. (12) and the derivative of eq. (13):

$$\begin{bmatrix} [\Phi_q] \\ [B] \end{bmatrix} \begin{Bmatrix} \ddot{q} \\ \ddot{z} \end{Bmatrix} = \begin{Bmatrix} (c) \\ (\ddot{z}) \end{Bmatrix} \quad (16)$$

inverting the matrix of this expression:

$$\{\ddot{q}\} = \begin{bmatrix} [\Phi_q] \\ [B] \end{bmatrix}^{-1} \begin{Bmatrix} (c) \\ (\ddot{z}) \end{Bmatrix} = [S](c) + [R](\ddot{z}) \quad (17)$$

This result is very useful in dynamic analysis. It still remains to describe the way in which the matrix [B] introduced in expression (13) can be chosen. According to eqs. (13) and (14) the only condition to be fulfilled by this constant matrix is that its rows be independent of the rows of the jacobian matrix $[\Phi_q]$. There are three main possibilities:

- a) Mani, Haug and Atkinson [11] use the Singular Value Decomposition (SVD) of $[\Phi_q]$ in a previous position so as to compute an orthogonal basis of the nullspace of $[\Phi_q]$ in this position. This basis is taken as matrix [B] (nullspace in a previous position) and it is expected that it remains independent of the row space of $[\Phi_q]$ for a large motion range.
- b) Kim and Vanderploug [12] make a similar approach to a) using the QR decomposition, that is cheaper than SVD. These authors compute matrices [S] and [R] by an updating technique that is more complicated and less efficient than the based in eq. (15).

c) García de Jalón and coworkers [7], [8] use for matrix [B] a boolean matrix that simply extracts some elements of $\{\dot{q}\}$ as independent velocities $\{\dot{z}\}$. The elements of $\{\dot{q}\}$ to be taken as independent are chosen after a Gaussian decomposition of $[\Phi_q]$ with total pivoting. This is the simplest method and it is also very efficient because with a boolean matrix [B] in eq. (15) the computation of matrices [S] and [R] is simpler. This is the standard method used in COMPAMM.

5. DYNAMIC ANALYSIS

In the context of natural coordinates virtual power is the most straightforward way to establish the motion differential equations. The virtual power of inertia forces of a link containing two points and two non coplanar unit vectors will be formulated next.

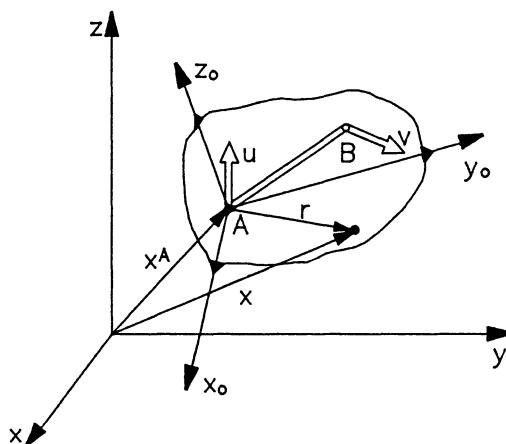


Figure 4. Inertia forces of a link with two points and two vectors

Consider the element of figure 4. The virtual power of this element can be expressed as:

$$W_{in} = \int_v \text{trace} \left[(\ddot{x}) (\dot{x})^T \right] dm \quad (18)$$

where $\{\ddot{x}\}$ is the acceleration vector and $\{\dot{x}\}$ is the vector of virtual velocities. Vector $\{x\}$ can be expressed as:

$$\{x\} = \{x^A\} + [T] \{r\} \quad (19)$$

being $[T]$ the rotation matrix. The inertial components of vectors u, v and segment AB can be expressed in the form:

$$\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} = [T] \begin{bmatrix} x_0 \\ \dot{x}_0 \\ \ddot{x}_0 \end{bmatrix} \quad (20)$$

where $[X]$ is a (3x3) matrix whose columns are:

$$\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} \equiv \begin{bmatrix} \{(x^B) - (x^A)\} & (u) & (v) \end{bmatrix} \quad (21)$$

According to eq. (19) the velocities and accelerations can be expressed as:

$$\{\dot{x}\} = \{\dot{x}^A\} + [\dot{T}] \{r\} \quad (22)$$

$$\{\ddot{x}\} = \{\ddot{x}^A\} + [\ddot{T}] \{r\} \quad (23)$$

matrices $[\dot{T}]$ and $[\ddot{T}]$ can be obtained differentiating eq. (20) and taking into account that matrix $[X_0]$ is constant (it contain the coordinates of AB, u and v in the moving frame),

$$\begin{bmatrix} \dot{T} \\ \ddot{T} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} \begin{bmatrix} x_0 \end{bmatrix}^{-1} \quad (24)$$

$$\begin{bmatrix} \dot{T} \\ \ddot{T} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} \begin{bmatrix} x_0 \end{bmatrix}^{-1} \quad (25)$$

Substituting eqs. (22)-(25) in eq. (18), after some algebraic manipulation, it is possible to arrive at the following result:

$$w_{in} = \{\dot{q}^e\}^T [M^e] \{\ddot{q}^e\} \quad (26)$$

where $\{\dot{q}^e\}$ and $\{\ddot{q}^e\}$ are respectively the natural virtual velocities and accelerations of the element. For instance, vector $\{\ddot{q}^e\}$ has the form:

$$\{\ddot{q}^e\}^T \equiv \left[(\ddot{x}^A)^T, (\ddot{x}^B)^T, (\ddot{u})^T, (\ddot{v})^T \right] \quad (27)$$

and matrix $[M^e]$ is the inertia matrix of the element. It is constant (independent of the element position) and has the following expression:

$$[M^e] = \begin{bmatrix} (z_{11} + 2s_1 + M)[I] & -(s_1 + z_{11})[I] & (s_2 + z_{12})[I] & (s_3 + z_{13})[I] \\ & z_{11}[I] & -z_{12}[I] & -z_{13}[I] \\ & & z_{22}[I] & z_{23}[I] \\ & & & z_{33}[I] \end{bmatrix} \quad (28)$$

In this expression $[I]$ is the (3×3) identity matrix and,

$$(S) = (s_1 \ s_2 \ s_3)^T = M [X_0]^{-1} (r^G) \quad (29)$$

$$[Z] = [X_0]^{-1} [J] [X_0]^{-T} \quad (30)$$

$$[J] = \int_V(r) \langle r \rangle^T dm \quad (31)$$

M is the mass of the element, $\{r^G\}$ the position vector of the center of gravity and $[J]$ the inertia tensor. Vector $\{s\}$ plays the rôle of a linear momentum and $[Z]$ is a transformed inertia tensor.

There are two very important points that need to be emphasized about these results: the first one is that the inertia forces are formulated in the inertial frame through a constant inertia matrix, allowing an increased efficiency in the solution of the dynamic equations; the second is that there aren't velocity dependent inertia forces, further improving the efficiency and simplifying the formulation.

The virtual power theorem applied to the whole mechanism leads to the expression:

$$\{\dot{q}\}^T \left[[M]\{\ddot{q}\} - \{Q(\{q\}, \{\dot{q}\})\} \right] = 0 \quad (32)$$

where $\{Q\}$ is the vector of external or internal forces that are position or velocity dependent (springs, dampers, friction, etc.) and $[M]$ is the constant mass matrix of the whole mechanism, obtained from the element matrices $[M^e]$ through an assembly process similar to the one used in the finite element method. The virtual velocity vector $\{\dot{q}'\}$ must satisfy the homogeneous constraint equation for velocities,

$$[\Phi_q] (\dot{q}') = (0) \quad (33)$$

Transposing this equation, multiplying it by a vector $\{\lambda\}$ of Lagrange Multipliers and adding it to eq. (32) the following equation is obtained:

$$(\ddot{q}')^T \left([M](\ddot{q}') + [\Phi_q]^T (\lambda) - (Q) \right) = 0 \quad (34)$$

now, for an arbitrary vector $\{\dot{q}'\}$ it is always possible to find a vector $\{\lambda\}$ that makes null the parenthesis of eq. (34). Putting this parenthesis together with eq. (12) the following equation is obtained:

$$\begin{bmatrix} [M] & [\Phi_q]^T \\ [\Phi_q] & [0] \end{bmatrix} \begin{Bmatrix} (\ddot{q}) \\ (\lambda) \end{Bmatrix} = \begin{Bmatrix} (Q) \\ (c) \end{Bmatrix} \quad (35)$$

This is perhaps the dynamic formulation most often found in the bibliography. Remember that natural coordinates make it specially appropriate because matrix $[M]$ is constant, matrix $[\Phi_q]$ is particularly easy to evaluate and there are not Coriolis or centrifugal inertia forces.

It is possible to derive another expression for the dynamic equations in which Lagrange multipliers do not appear. Using eq. (15) to introduce independent virtual velocities in eq.(32):

$$(\dot{z}')^T [R]^T \left((M)(\ddot{q}) - (Q) \right) = 0 \quad (36)$$

Now virtual velocities $\{\dot{z}'\}$ are independent and can be eliminated. Putting this equation together with eq. (12):

$$\begin{bmatrix} [\Phi_q] \\ [R]^T [M] \end{bmatrix} \begin{Bmatrix} \ddot{q} \\ \dot{z}' \end{Bmatrix} = \begin{Bmatrix} (c) \\ [R]^T (Q) \end{Bmatrix} \quad (37)$$

This is an alternative equation to eq. (35). Both equations can suffer a lack of stability in the fulfillment of constraint equations. This problem can be solved either using the Baumgarte stabilization method [13], integrating a mixed system of algebraic-differential equations, or formulating and integrating the motion differential equations in independent coordinates. This is the approach most currently used in COMPAMM.

Substituting in eq. (37) the expression for $\{\ddot{q}\}$ defined in eq. (16) the following equation is obtained:

$$[R]^T[M][R]\{\ddot{q}\} = [R]^T(Q) - [R]^T[M][S](c) \quad (38)$$

In order to apply this equation it is necessary to compute matrix $[R]$ from eq. (15) but not matrix $[S]$, because this matrix is never used separately but multiplied by vector $\{c\}$. From eq. (17) the term $([S]\{c\})$ can be computed as the vector of dependent accelerations $\{\ddot{q}\}$ when the independent accelerations $\{\ddot{z}\}$ are taken to be zero. Eq. (16) allows the computation of $\{\ddot{q}\}$ with this condition.

6. CURRENT ANALYSIS CAPABILITIES OF COMPAMM

The version 1.0 of COMPAMM is able to analyze open or closed loop mechanisms with the following joints: Spherical (S), Revolute (R), Cylindrical (C), Prismatic (P), Helical (H), Gears (G) and Universal (U). Other kinds of pairs can be introduced by the user through force constraints. In addition to this, the user can introduce externally driven absolute or relative motion of some links.

The multi-rigid-body problems that COMPAMM can currently solve are the following:

1. Kinematic problems:
 - . Initial position (assembly) problem
 - . Velocity and acceleration analysis
 - . Finite displacement problem (kinematic simulation)

2. Dynamic problems:
 - . Computation of driving forces and reactions (inverse dynamics)
 - . Computation of motion along the time (dynamic simulation)

COMPAMM can be executed interactively or in batch mode. When executed interactively it displays the motion of the mechanism with realistic rendering (see Section 7. and 8.) at the same time that the computations are done.

COMPAMM admits two kinds of external actions: external forces (only for dynamic simulation) and externally driven absolute or relative link motions (both for kinematic and dynamic simulation). External actions must be defined by the user through:

- a) ASCII files,
- b) user-written Fortran subroutines and
- c) interactive input devices such as knob boxes (currently, only for kinematic simulation).

The interaction between the user and COMPAMM can be established at two levels. In the first level the user rôle is limited to define the external actions previously referred to. In the second level the user is allowed to write a Fortran subroutine that have access to most of the COMPAMM internal information on the state of the mechanism; in this way the user is able to modify the number and the kind of joints (and so the number of degrees of freedom), and he can control other effects such as collisions, friction, backlash, non-linear springs and dampers, etc.

In the current version of COMPAMM, flexibility of the links is not allowed. Some insight in flexible behavior of mechanisms can be obtained substituting a flexible link by several rigid links joined by springs.

7. COMPAMM PREPROCESSORS

COMPAMM has been designed to be very easy to use and very powerful in its interactive and graphic capabilities. In order to get these features COMPAMM incorporates a language for the definition of the mechanism and a 3-D solid modeller that allows animation with realistic rendering.

The kinematic configuration of the mechanism and the dimensions and mass properties of its links are defined in an ASCII file with a free-format, very readable language of commands and alphanumeric data. The geometry of the links and their inertia properties are defined on a reference frame fixed to each link, so as not to have position dependent properties. The topological information that must be defined is limited to the points and vectors belonging to each link and for each joint the type of the joint and the elements that it relates to. From this information COMPAMM is able to formulate automatically the constraint and the motion equations.

In the referred ASCII file it is possible to provide, for each mechanism link, one or more file names in which a realistic geometric definition of the link is included. This in-

formation is composed of a list of vertex coordinates, a list of polygons and a list of normal vectors in the vertices of each polygon. This information can be generated by the 3-D solid modeller included in COMPAMM, or can be imported from external commercially available CAD systems, that provide output neutral files, just writing a simple program that changes the format of these files to COMPAMM format.

The solid modeller included in COMPAMM is based on Constructive Solid Geometry, and it is able to produce complicated faceted solid models from a large set of primitives (cylinders, prisms, spheres, torus, ...) and objects obtained by sweeping, using the boolean operations of union, intersection and difference.

8. COMPAMM POSTPROCESSORS

COMPAMM is a program specifically prepared to run on 3-D graphic workstations with very powerful solid rendering capabilities. These systems appeared on the market three or four years ago, and quite recently they have increased their capabilities very impressively (RISC processors, big graphic memories, graphic accelerators ...) at the same time the prices dropped dramatically, so making them affordable for a bigger user community. These 3-D workstations are the ideal computer platform for interactive mechanism simulation and for them COMPAMM has been designed and developed. Currently COMPAMM runs on Hewlett-Packard SRX and Silicon Graphics IRIS 4D workstations. Other hardware platforms will be available in the near future. Being interfaced with solid modellers, COMPAMM is able to do realistic animations whose speed depends on the current capabilities of the workstation. For simple simulations, low-end workstations (0,5 Mflops, 5000 shaded pol/sec) allow nearly real time interactive kinematic simulation (1 to 5 frames/sec). For top-end workstations (10 Mflops, 50000 shaded pol/sec) real time dynamic simulations are also at hand for many practical cases.

COMPAMM provide among others the following graphic capabilities:

1. Up to six point light sources plus ambient light. Light sources position, intensity and color, can be interactively controlled by the user through dials and buttons.
2. Switch between wireframe, wireframe with hidden lines removed, flat shading and Gouraud shading, just pushing a button. This can be done for each object separately.
3. Variable degree of transparency for each object, from opaque to invisible, interactively controlled.

4. Up to six simultaneous windows with different size. Up to six cameras fixed, or attached to moving elements. The camera position with respect to the fixed or the moving elements can be controlled by the user. Interactive pan, zoom and tilt for each camera.
5. Each window can show a different representation of the mechanism; wireframe, Gouraud shading, ... or its representation with natural coordinates (points and unit vectors).
6. With double buffering, COMPAMM offers flicker free images with pseudo color (4096 colors in each frame).

The postprocessor of COMPAMM can be used interactively, displaying the new position of the mechanism just after it has been computed by the kinematic or dynamic simulation module, or in play-back mode, displaying at a maximum frame rate (or at a lower rate chosen by the user) the results of a previous computation. In both modes COMPAMM offers the graphic capabilities previously referred to. In play-back mode COMPAMM also offers the possibility of examining the output file through X-Y plots on 2-D windows, in which time variation of the mechanisms variables (positions, velocities, accelerations, reaction or driving forces, etc.) are displayed.

COMPAMM also offers the possibility to record sequences in video tapes, both in real time or in frame by frame modes.

9. EXAMPLE 1: 3-D 5 d.of. robot

The modellization of 5 d.of. robot with natural coordinates has been carried out. There are 6 movable points and 3 movable unit vectors. The corresponding constraint equations are formulated automatically and so they are the motion differential equations.

In figure 5 a hardcopy of the motion of this robot under a system of external forces is presented. Several positions have been superposed on the screen so as to have a *static* motion representation. The representation of knob and button boxes appears on the screen.

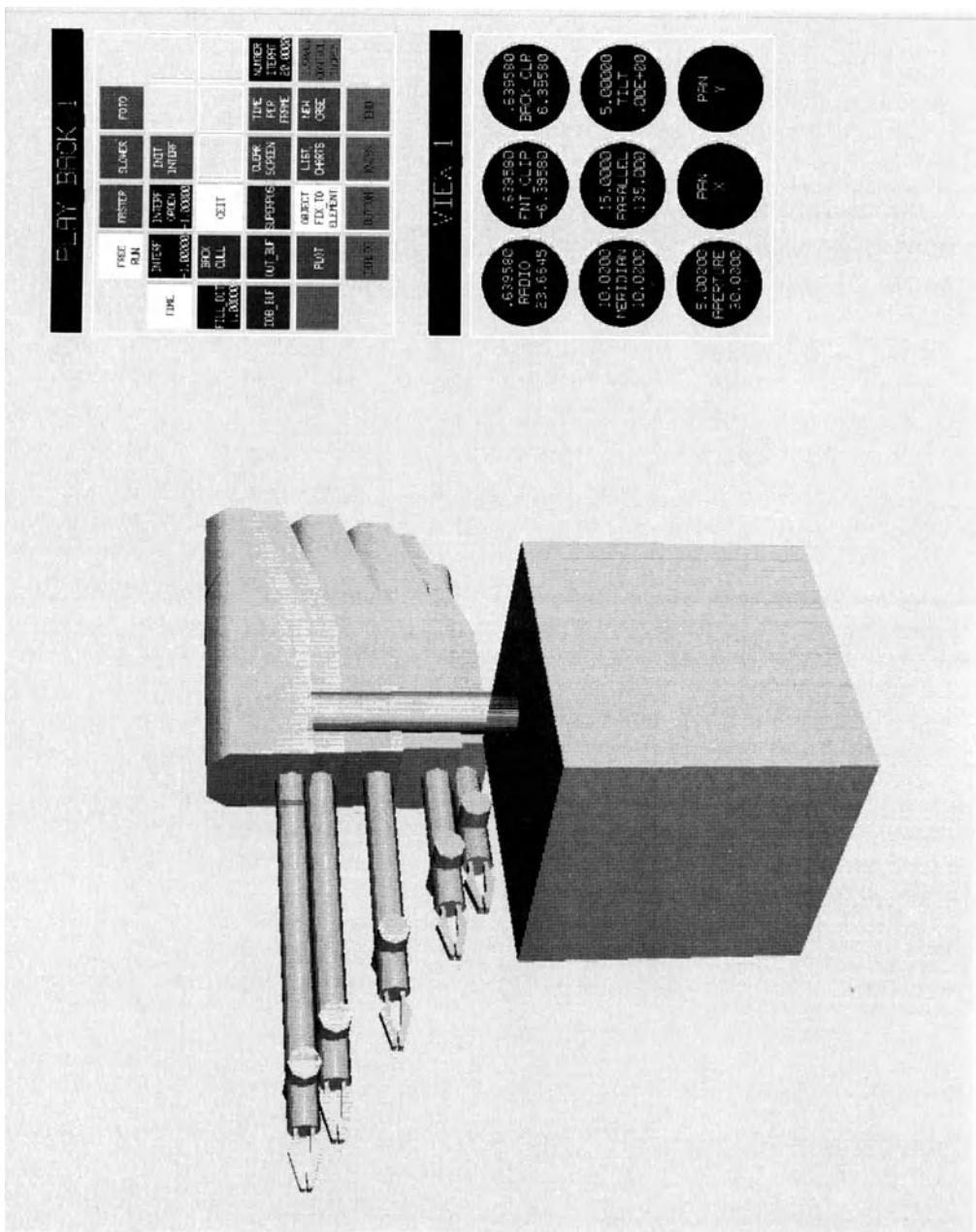


Figure 5

10. EXAMPLE 2. RSCR Mechanism

The RSCR 3-D mechanism previously referred to, has been analyzed dynamically, starting with initial velocity. The only external action is gravity. In figure 6 a black and white hardcopy of a possible configuration of the screen is shown. In this case there are two animated windows in which the mechanism motion can be seen from two different points of view. Up to figure 5 frames/sec. can be obtained in play-back mode with an HP 370 Turbo SRX workstation. There are also two overlay windows on which the time variation of velocities and accelerations of natural coordinates are shown.

11. CONCLUSIONS

COMPAMM is a 3-D multi rigid body package that uses a new system of dependent coordinates, called *natural coordinates*, that have some advantages over conventional methods: the jacobian matrix is much easier to evaluate, the mass matrix is constant in the inertial reference frame and there are not Coriolis or centrifugal forces in the formulation. The program is able to analyze mechanisms with any kinematic configuration, formulating automatically the constraint and motion differential equations.

COMPAMM has advanced graphic capabilities: It incorporates a solid modeller and is able to import solid objects generated by commercial CAD packages. COMPAMM admits up to six light sources; six windows; fixed or movable cameras; wireframe, flat or Gouraud shading; transparency; high degree of interactivity and user controllable rendering conditions, etc.

REFERENCES

- [1] Paul, B. "Analytical Dynamics of Mechanisms - A Computer Oriented Overview", *Mechanism and Machine Theory*, vol. 10, pp. 481-507, 1975.
- [2] Haug, E.J. (ed.). "Computer Aided Analysis and Optimization of Mechanical System Dynamics", Springer-Verlag, 1984.
- [3] Bae, D.-S., Hwang, R.S. and Haug, E.J., "A Recursive Formulation for Real-Time Dynamic Simulation", *ASME Conference on Advances in Design Automation*, 1988.

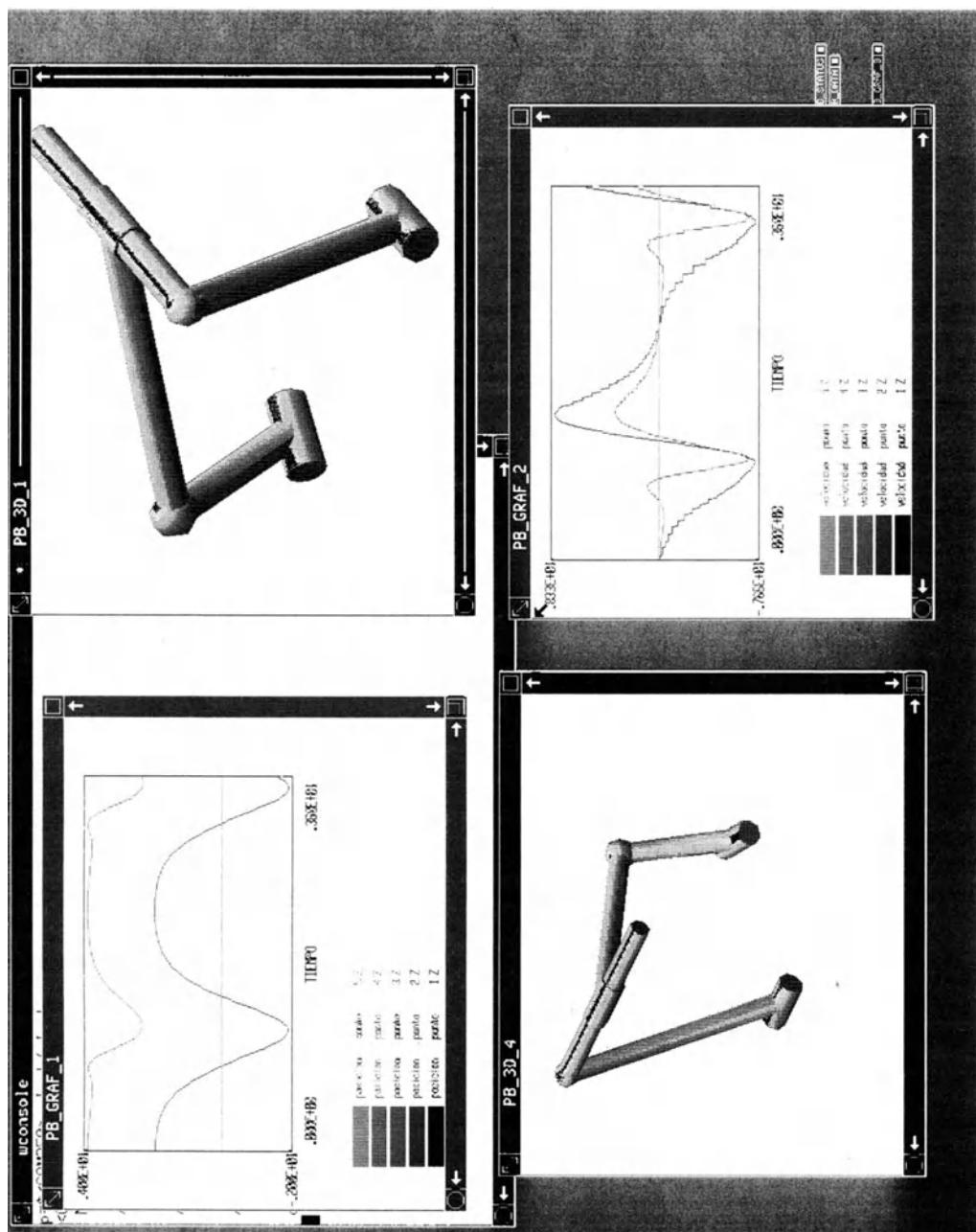


Figure 6

- [4] Schiehlen, W.O., "Dynamics of Complex Multibody Systems", SM Archives, vol. 9, pp.159-195, 1984.
- [5] Rosenthal, D.E. and Sherman, M.A., "High Performance Multibody Simulations Via Symbolic Equation Manipulation and Kane's Method", The Journal of the Astronautical Sciences, vol. 34, pp. 223-239, 1986.
- [6] Vilallonga, G., Unda, J. and García de Jalón, J., "Numerical Kinematic Analysis of Three-Dimensional Mechanism Using a Natural System of Dependent Coordinates", ASME Paper No. 84-DET-199, 1984.
- [7] García de Jalón, J., Unda, J. and Avello, A., "Natural Coordinates for the Computer Analysis of Three-Dimensional Multibody Systems", Computer Methods in Applied Mechanics and Engineering, vol. 56, pp. 309-327, 1986.
- [8] García de Jalón, J., Unda, J., Avello, A. and Jiménez, J.M., "Dynamic Analysis of Three-Dimensional Mechanisms in Natural Coordinates", ASME J. on Mechanisms, Transmissions and Automation in Design, vol. 109, pp. 460-465, 1987.
- [9] Unda, J., García de Jalón, J., Losantos, F. and Emperatza, R., "A Comparative Study on Some Different Formulations of the Dynamic Equations of Constrained Mechanical Systems", ASME J. of Mechanisms, Transmissions and Automation in Design, vol. 109, pp. 466-474, 1987.
- [10] Nikravesh, P.E., "Computer-Aided Analysis of Mechanical Systems", Prentice-Hall, 1988.
- [11] Mani, N.K., Haug, E.J. and Atkinson, K.E., "Application of Singular Value Decomposition for Analysis of Mechanical System Dynamics", ASME J. on Mechanisms, Transmissions and Automation in Design, vol. 107, pp. 82-87, 1985.
- [12] Kim, S.S. and Vanderploeg, M.J., "QR Decomposition for state Space Representation of Constrained Mechanical Dynamic Systems", ASME J. on Mechanisms, Transmissions and Automation in Design, vol. 108, pp. 176-182, 1986.
- [13] Baumgarte, J., "Stabilization of Constraints and Integrals of Motion in Dynamical Systems", Computer Methods in Applied Mechanics and Engineering, vol. 1, pp. 1-16, 1972.

DYMAC & DYSPAM - Programs for the Dynamic Analysis and Simulation of Planar Mechanisms and Multibody Systems

by B. Paul, University of Pennsylvania, Philadelphia, Pennsylvania
U.S.A.

DYMAC (DYnamics of MAChinery)

ABSTRACT

DYMAC (DYnamics of MAChinery) is an all FORTRAN computer program for the solution of problems in Dynamics of interconnected bodies which undergo planar motion. It is used to find displacements, velocities, accelerations, and joint reactions for multi-degree of freedom systems subjected to arbitrary, user supplied, forces. The mechanism may contain any number of closed or open loops, and be subjected to auxiliary constraints imposed by gears, cams, and user-supplied motion generators.

Descriptions of the program variables, input, output, modelling and preparation of input data, as well as solutions of sample problems, are given. A post-processing procedure is described which allows the user to provide for plotting, or otherwise processing any variables of interest to him, utilizing his local computer facilities.

1. INTRODUCTION

Program DYMAC enables users to do a complete dynamics analysis of multi-degree of freedom, arbitrary planar linkages under the influence of user-specified applied forces and moments which may be arbitrary functions of displacement and velocity. In addition, the user may specify a wide variety (virtually unlimited) of relations between the gross motions of various links. Such *control constraints* (or *motion generating constraints*) may be used, for example, to insure that a given link moves with a specified angular velocity, or to have points in a mechanism (e.g. wheel of a vehicle) move along user-specified paths (e.g. bumpy roadways). The control constraints may also be used to include the effects of gears and cams.

The mechanism may contain closed loops, (e.g. wheel suspension linkages) or open loops (e.g. multiple pendula). The user is required to provide the minimum information needed to describe the geometric properties and mass distribution of the system (e.g. link lengths, mass, moments of inertia, initial positions and velocities, etc.). The program internally formulates the appropriate dynamic equations of motion, automatically integrates them, and makes whatever internal checks are necessary to maintain accuracy.

The *principal results* calculated are the position, velocity, and acceleration associated with each link at user specified time intervals, as well as the forces and moments applied at the mass centers of all links. In addition, displacement, velocity and acceleration components of user-selected *points of interest* anywhere in the system are displayed at user specified time intervals.

The program is so organized that the user can easily request a wide variety of post-processing procedures, such as plotting, animation, file generation, or any other further calculations on output data that he considers desirable.

The program is based upon concepts of analytical mechanics described in Paul [1]-[5]. Computational procedures are described in the dissertations of Hud [6] and Amin [7]. Detailed procedures for using the program are given in the User's Manual [8]. The relationship of this work to other programs is discussed in [4] and [9].

In this paper, we intend to give a brief description of the program characteristics, method of analysis, modelling procedure, data format, and typical examples.

2. PROGRAM CHARACTERISTICS

AUTHORS: B. Paul, A. Amin, G. Hud

Department of Mechanical Engineering and Applied Mechanics

University of Pennsylvania

Philadelphia, Pennsylvania 19104

LANGUAGE: FORTRAN

MACHINES: Tested on various main frames (e.g. IBM, UNIVAC, DEC VAX),
and on personal computers compatible with IBM-AT.

PRECISION: Double Precision

STORAGE REQUIREMENTS: Approximately 192,000 Bytes (includes double precision accuracy)

CAPABILITIES: Handles up to 20 links, 30 edges, 10 degrees of freedom, 10 loops, 50 points of interest, 30 joints, 10 spatial constraints, 10 temporal constraints.

NUMBER OF LINES IN SOURCE PROGRAM:

Approximately 2,500, including 600 comment lines.

3. METHOD OF ANALYSIS

The equations of motion are generated using the d'Alembert-Lagrange (virtual

work) method, with equations of constraint based on loop closure, as described in [3], [4] and [7].

The position variables (angles and lengths which vary with time) are denoted by symbols¹ ($\psi_1, \psi_2, \dots, \psi_M$). In a system with F degrees of freedom, F of the quantities ψ_i are considered to be *primary* or *generalized* coordinates, and the remainder are considered to be *secondary* variables. The user initially selects the primary variables in any way he desires, and the program will, when necessary, automatically change the choice of primary variables in such a way as to minimize round-off error. Initial values of the primary variables and their time rates (angular or linear velocities) are supplied by the user, along with initial estimates of the secondary position variables. For example, the user may give the initial angle of one link of a four bar linkage along with estimates (possibly taken from a rough sketch) of the two unknown link angles. These initial guesses are automatically converted into accurate initial position data for all secondary variables, by means of a Newton-Raphson algorithm.

The equations of motion are generated internally, and subroutines are called upon to solve the equations of motion by a Runge-Kutta numerical integration algorithm. The updated position variables ψ_i , velocities $\dot{\psi}_i$, and accelerations $\ddot{\psi}_i$ are then passed to an output subroutine, OUTP, which prints ψ_i , $\dot{\psi}_i$, and $\ddot{\psi}_i$ at user-specified time intervals. OUTP also prints the Cartesian components of position, velocity, and acceleration for user-selected *points of interest*. The output of the program may be stored as needed, and may conveniently be retrieved for use in subroutine UPROC, a user-supplied post-processing routine for plotting, punching, or otherwise manipulating information.

4. MODELLING OF MECHANISM

In this Section, we list most of the steps needed to model the mechanism for DYMACH. A more complete description of the procedure is given in the DYMACH User's Manual [8].

(1). *Establish a fixed Cartesian global coordinate system (x,y), and sketch the kinematic network.* This is a collection of straight lines (*edges*) connecting each of the joints (*nodes*). The network corresponding to the example of Fig. 1a is shown in Fig. 1b. The nodes are labelled with the capital letters A,B, . . . etc.

¹The position variables (ψ_1, \dots, ψ_M) are called Lagrangian variables. All angles are measured positive counter-clockwise in radians.

(2). Identify and number the independent loops in the network.

A loop is classified as independent if it cannot be formed by merely superposing several of the other loops. Usually [5] it will suffice to select the interior loops of a mechanism as the independent loops, as illustrated in the example of Fig. 1. The positive sense of traversal along a loop is *counterclockwise*. This step is omitted if no closed loops are present in the system.

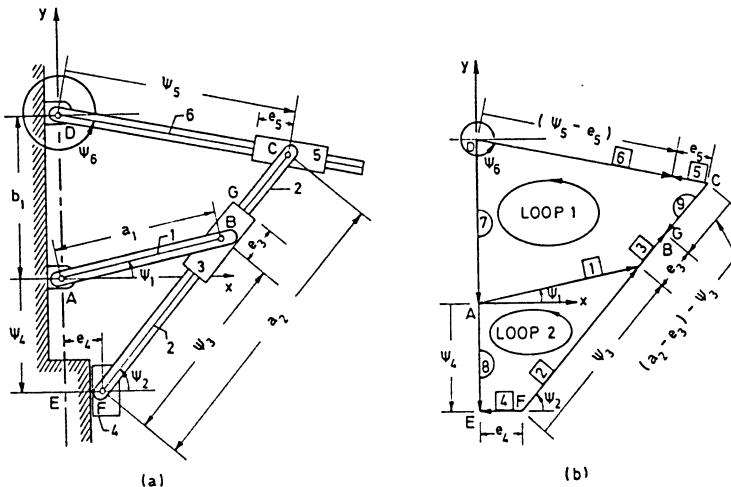


Fig. 1. (a) Example Mechanism; (b) Kinematic Network

(3). Determine the number of degrees of freedom, F .

For most cases F can be determined from the expression²

$$F = M - 2L - N_{uc} \quad (1)$$

where M is the number of Lagrangian variables, L is the number of independent loops, and N_{uc} is the total number of user-supplied constraints (see discussion in Sec. 6). For example, in Fig. 1, $F = 6 - (2)(2) - 0 = 2$.

(4). Orient edges and label the Lagrangian coordinates.

Choose a positive direction for each edge and indicate it by means of an arrowhead as illustrated in Fig. 1b. Label all edges with a number enclosed in a small box located on or near the edge. Label variable lengths and angles of the edges as Lagrangian variables $\psi_1, \psi_2, \dots, \psi_{NLAG}$. Measure angles positive counterclockwise from the direction of the global x -axis, as shown in Fig. 1b. Note that the numbering of bodies, edges, and Lagrangian variables are independent of one another.

²Exceptions to this rule are discussed by Paul [5]. Also see the discussion in Sec. 6 on motion generation constraints.

(5) ***Construct the Edge Data Table.***

Every edge i in a mechanism possesses a length ρ_i inclined at an angle α_i to the positive x -axis (measured positive counterclockwise). In turn, each ρ_i and α_i are composed of a constant part and a variable part. For example, edge 9 in Fig. 1b has

$$\rho_9 = (a_2 - e_3) - \psi_3 \quad \begin{matrix} \text{constant} \\ \text{variable} \end{matrix} \quad (2-a)$$

$$\alpha_9 = \pi + \psi_2 \quad \begin{matrix} \text{constant} \\ \text{variable} \end{matrix} \quad (2-b)$$

More generally, for edge i

$$\rho_i = CL_i \pm \psi_{i\ell} \quad (3-a)$$

$$\alpha_i = CA_i \pm \psi_{ia} \quad (3-b)$$

where CL_i and CA_i are the constant parts of ρ_i and α_i respectively, $i\ell$ is the index of the length variable and ia is the index of the angle variable.

The appropriate "edge data" is recorded in Table 1 as follows: Column 1 contains the edge numbers, I , in sequence. the index ($i\ell$) of the *length* variable (if any) is entered in column 2 with the *sign* (+ or -) that appears in front of $\psi_{i\ell}$ in equations such as (2-a). The constant part (CL_i) of the edge i is entered in column 3. The index (ia) of the angle variable ψ_{ia} associated with link I is entered in column 4, with the appropriate (+ or -) sign that appears in equations such as (2-b). The last column contains the constant angles CA_i associated with edge i . Table 1 corresponds to Fig. 1b.

Table 1. Edge Data

Edge (I)	Length Index	Length Constant	Angle Index	Angle Constant
1		a_1	1	
2	3	e_3	2	
3		e_4		
4		e_5		π
5		$-e_5$	6	π
6	5	b_1	6	
7				$3\pi/2$
8	4			$3\pi/2$
9	-3	$(a_2 - e_3)$	2	π

(6). ***Construct the Complete Loop Table.***

This table specifies which edges are contained in each loop and how they are oriented relative to the loop direction. Imagine that loop number 1 is being traversed counterclockwise, and make the following entry in Column J of line I:

+1 (-1) if edge J is traversed positively (negatively)

0 if edge J is not in the path.

As an example, Table 2 is the Complete Loop Table for the Mechanism of Fig.

- Only non-zero values need be entered in all tables.

Table 2. Complete Loop Data

Loop No (I)	Edge No. (J)								
	1	2	3	4	5	6	7	8	9
1	1	1	1	-1	1	-1	1	-1	
2	-1	1	-1					1	

(7) ***Construct the Points of Interest Table.***

The mass center of a link, the point of application of an external force, or any other point selected by the user is called a *point of interest*. The local polar coordinates (r_i, α_i) of all such points with respect to a user-selected reference edge in each link are specified in a Point of Interest Table.

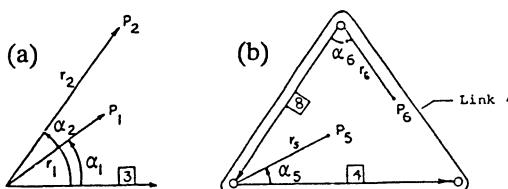


Fig. 2. Points of interest P_i , fixed relative to a reference edge.

For example, in Fig. 2a, the points of interest P_1 and P_2 are referred to the reference edge 3, which may or may not be attached to a physical link. Figure 2b shows different points of interest (P_5, P_6) referred to different reference edges (4 and 8) located in the same link (4). Table 3 shows how the data might look for a system containing link 4 as illustrated in Fig. 2b. If a point of interest is not attached to a physical body, the second column of the table has a blank entry for that point. If standard output (position, velocity, acceleration) is to be printed for a point of interest, the last column of the table contains a zero or blank, otherwise a 1 is entered to indicate that the

standard printout for that point is not required. Standard data is always available for post-processing by the user supplied subroutine UPROC, provided that at least one point is selected for printout, or if post-processing is required.

Table 3. Point of Interest Location Table

Point I	Link (I)	Reference Edge (I)	Angle ALFPI(I)	Radius RPI(I)	Print Option IOUT(I)
1		3	α_1	r_1	0
2		3	α_2	r_2	0
3	2	2	α_3	r_3	0
4	2	8	α_4	r_4	0
5	4	4	α_5	r_5	0
6	4	8	α_6	r_6	1
7	5	6	α_7	r_7	1

(8) Body Inertia Table

Body (or link) i has mass m_i , a center of mass located at a point of interest denoted by $IPIL(i)$, and a moment of inertia I_i about an axis, normal to the plane of motion, through the mass center. Each of these four items should be tabulated for each moving body.

(9) Given (Applied) Force Table

The given forces $FX(I)$, $FY(I)$, and the couple (torque) $TOR(I)$ are to be applied at a specified set of points (*force application points*) labelled $I = 1, 2, \dots, NFAP$, where I is called the *force index*. The point of application of the force with index I is the *point of interest* $IPIF(I)$ whose global coordinates have already been entered in the points of interest table. The data are to be entered in a Table of Given Forces, as illustrated in Table 4.

Table 4. Given Force Table

Force Index I	Point of Interest IPIF(I)	x-Comp. FX(I)	y-Comp. FY(I)	Torque TOR(I)
1	2	x_1	y_1	T_1
2	4	x_2	y_2	T_2
...
NFAP				

Only those components of force or torque which are constant (e.g. gravity force) need to be entered in this table. Variable components of force or torque must be described in a user-supplied subroutine FORCES (see Sec. 5); such variable forces will override any entry in Table 4.

(10) *Construct the Initial Conditions Table.*

Recall that in a mechanism with f degrees of freedom, f of the variables ψ_i are designated as primary variables (or generalized coordinates) and the remainder are said to be secondary variables. The physics of the problem requires that initial displacement and velocity corresponding to *primary* ψ_i and $\dot{\psi}_i$, must be prescribed precisely. However, merely approximate estimates are required for the initial values of secondary position variables ψ_i , because their values are refined internally by a Newton-Raphson iteration procedure. *Secondary* initial velocities $\dot{\psi}_i$, are *not* to be specified because they are implied by the primary data, and are calculated internally. The variables ψ_i and $\dot{\psi}_i$ are represented in the program by $\text{PSI}(I)$ and $\text{PSID}(I)$. Angles and angular velocities are expressed in radians and radians/sec., respectively. By assigning values to a *Type Index* [FORTRAN variable $\text{JQ}(I)$] in the manner shown below, the *user* identifies which ψ_i are *initially*³ to be considered as primary, which as secondary, and which as *controlled variables*. The concept of a controlled variable is explained in Sec. 6.

The type index $\text{JQ}(I)$ is set equal to 1, 0, or -1 accordingly as ψ_i is primary, secondary or controlled. As an example, the Initial Conditions Table for the mechanism of Fig. 1 might assume the form of Table 5, where ψ_1 and ψ_6 have been designated as primary. The initial values shown for ψ_1 , ψ_6 , $\dot{\psi}_1$, $\dot{\psi}_6$, are assumed to be given precisely.

Table 5. *Initial Conditions*

<u>Index</u> <u>(I)</u>	<u>Type</u> <u>JQ(I)</u>	<u>Pos'n</u> <u>PSI(I)</u>	<u>Vel.</u> <u>PSID(I)</u>
1	1	.26000	1.0
2		.87	
3		1.8	
4		1.2	
5		2.0	
6	1	6.02000	2.0

5. MODELLING OF ACTIVE FORCES, SPRINGS, AND DAMPERS

The applied forces and moments may be functions of time, displacements, and velocities. It should be noted that the weights of the links are treated as applied forces. To permit the solution of a wide range of problems, the user may write FORTRAN expressions for all nonzero values of active forces and torques and include them in an appropriate location in an otherwise standard subroutine called SUBROUTINE FORCES.

³ During subsequent calculations, the program chooses primary variables so as to minimize roundoff errors.

Specified external force components X_i , Y_i (and torques T_i) may act through specified *points of force application*, labelled 1,2, . . . , NFAP. The forces and torques may be functions of position, velocity and time. Internal forces, such as those due to dampers and springs, can be treated as pairs of external forces acting on the members to which they are attached.

For example, to model the forces applied to links 1 and 2 by the spring and damper shown in Fig. 3, the user could place the following FORTRAN expressions where indicated in a supplied dummy subroutine named FORCES.⁴

C***** USER-SUPPLIED EXPRESSIONS FOLLOW *****

```
F = -CD*PSID(5) - SK*(PSI(5) - PSI50)
FX(3) = -F*DCOS(PSI(4))
FY(3) = -F*DSIN(PSI(4))
FX(4) = -FX(1)
FY(4) = -FY(1)
```

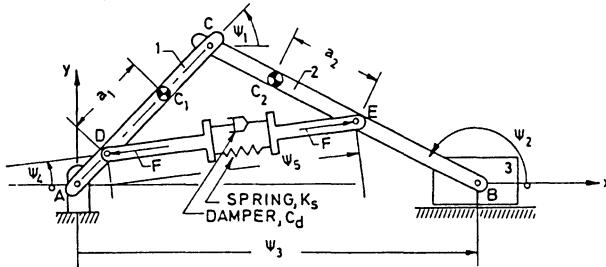


Fig. 3. Modelling Springs and Dampers

Recall that FORTRAN names for ψ_i and $\dot{\psi}_i$ are PSI(I) and PSID(I) respectively. The names for the forces and the torque applied at the force application point of index I are FX(I), FY(I), AND TOR(I). These are standard DYMAC conventions for all problems. The constants a_1 , a_2 , etc. were given given the FORTRAN names A1, A2, etc. peculiar to this problem. The numerical values of these constants may be passed through FORTRAN common statements or by other means supplied by DYMAC.

6. MODELLING MOTION GENERATOR CONSTRAINTS

In addition to permitting virtually arbitrary applied forces to act on the links, DYMAC gives the user the opportunity to specify relationships of his choosing between the variables ψ_i . These relationships are called *motion generator constraints* or *control constraints*. They are of the two general forms:

⁴ Details on the derivation of the FORTRAN expressions are given in [8].

$$h_i(\psi_1, \psi_2, \dots, \psi_M) = 0; \quad (i = 1, 2, \dots, N_s)$$

$$g_i(\psi_1, \psi_2, \dots, \psi_M, t) = 0 \quad (i = 1, 2, \dots, N_t)$$

where t is time. All relationships of this type are to be placed where indicated in a supplied dummy routine called MOTGEN. DYMACH requires that the user provide FORTRAN expressions in MOTGEN for the following partial and total derivatives of g_i . The FORTRAN names to be used for these derivatives are as follows:

g_i	$\partial g_i / \partial \psi_j$	$d/dt(\partial g_i / \partial \psi_j)$	$\partial g_i / \partial t$	$d/dt(\partial g_i / \partial t)$
$G(I)$	$DG(I,J)$	$DTDG(I,J)$	$DTG(I)$	$DTDTG(I)$

Similar names are used for h_i and its derivatives.

Each control constraint reduces the degree of freedom of the system by one. Thus the degree of freedom for a system with M Lagrangian variables and L independent loops is

$$F_r = M - 2L - N_s - N_t$$

For each control equation introduced, the user should designate, as a *controlled variable*, one variable (other than a primary variable) that appears in the equation, and indicate this by entering $JQ(I) = -1$ in the second column of table 5.

Example of constant velocity constraint:

Suppose, for example, that one wishes to specify that link 4 of a mechanism rotates with constant angular velocity $\dot{\psi}_4 = C$. In other terms

$$g_1 = \psi_4 - Ct - \psi_{4,0} = 0$$

where $\psi_{4,0}$ is the initial value of ψ_4 , at $t = 0$. The only nonvanishing derivatives of g_1 are:

$$\partial g_1 / \partial \psi_4 = 1, \quad \partial g_1 / \partial t = -C,$$

The corresponding FORTRAN statements needed in subroutine MOTGEN are:

$$G(1) = PSI(4) - C*T - P4O$$

$$DG(1,4) = 1.0$$

$$DTG(1) = -C$$

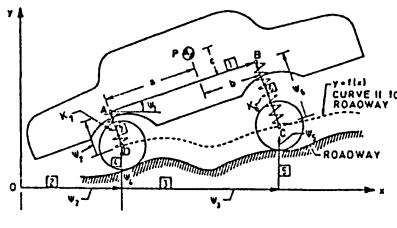
where C and $P4O$ are constants chosen by the user.

More sophisticated examples of motion generation constraints are given below.

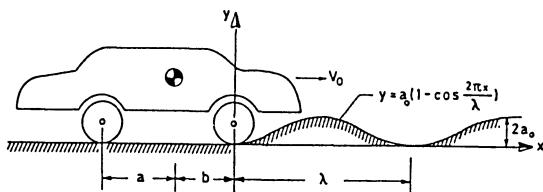
7. EXAMPLE 1. AUTOMOBILE ON A ROUGH ROAD

We consider the pitching and heaving motions of an automobile traversing a bumpy road, as shown in Fig. 4. We assume that the rear wheels move with constant horizontal velocity V_o while the wheel axles, C and D, move along a curve which is parallel to a user-specified road profile $y = f(x)$. An example of such a road profile

function is given by the sinusoidal roadway shown in Fig. 5. It is also assumed that the suspension springs, AD and BC, always remain normal to the fixed line AB in the rigid chassis. Full numerical data is given in [8] for all weights, inertias, stiffnesses, and geometry.



**Fig. 4. Vehicle on Rough Road.
Dampers (not shown) act in
Parallel With Springs.**



**Fig. 5. Initial State.
Uniform Horizontal
Initial Velocity.**

Upon introducing the seven Lagrangian coordinates (ψ_1, \dots, ψ_7) shown in Fig. 4, and noting the single independent loop ADEFBCA, one may assemble all input data in the form of tables (as in Sec. 4). Points of interest 1,2,3,4,5 correspond to the physical points P,A,B,D,C. Each of these points is also a point of force application. The only force acting at point 1 (P) is the weight of the chassis ($FY(1) = -2500$ lb) which is entered in the table of given forces. If the spring forces are denoted by F_6 and F_7 , the corresponding time-variable forces and torques are expressed, in subroutine FORCES, as follows:

```

F6 = -325.* (PSI(6) - 20.) - 10.*PSID(6)
F7 = -325.* (PSI(7) - 20.) - 10.*PSID(7)
FX(2) = -F7*DSIN(PSI(1))
FY(2) = F7*DCOS(PSI(1))
FX(3) = -F6*DSIN(PSI(1))
FY(3) = F6*DCOS(PSI(1))
FX(4) = -FX(2)
FY(4) = -FY(2)
FX(5) = -FX(3)
FY(5) = -FY(3)

```

The two "control constraint equations" which specify that the wheels maintain contact with the road profile, $y = f(x)$, are "spatial" constraints given by:

$$h_1 = \psi_4 - f(\psi_2) = 0$$

$$h_2 = \psi_5 - f(\psi_2 + \psi_3) = 0$$

The constraint equation which specifies the that the rear axle moves forward at

the uniform horizontal speed V_o is a "temporal" (time dependent) constraint given by:

$$g_1 = \psi_2 - V_o t + (a+b) = 0$$

where $-(a+b)$ is the value of ψ_2 at $t=0$.

The FORTRAN expressions corresponding to these constraint functions and their derivatives are placed in the provided subroutine MOTGEN [8]. The standard output of DYMAC includes the position, velocity, and acceleration of the mass centers of all moving rigid bodies (in this case there is only one such body), and the same for all specified points of interest. If one asks for output on the three points of interest A,B, and P (see Fig. 4), a sampling of the standard data output for those points would look as shown in Table 6.

Table 6. From Standard Output of DYSPAM, for Points of Interest

Time	Point	Position		Velocity		Acceleration	
		X	Y	XDOT	YDOT	XDDOT	YDDOT
0.0	1	-108	16.2	550	0	.3 E-10	.5 E-5
	2	54	16.2	550	0	.3 E-10	.5 E-5
	3	0	16.2	550	0	.3 E-10	.5 E-5
0.05	1	-80.5	16.1	549	-0.52	-27	-22.5
	2	-26.5	16.2	549	1.51	-27.1	68
	3	27.5	16.2	549	3.56	-27.2	158

By means of a simple user-supplied subroutine, UPROC, it is possible to capture selected output data and store them in a file for processing by a plotting routine. For example, the motion of the mass center of the automobile in this example was plotted by a printer plot program called by UPROC; the result is shown in Fig. 6.

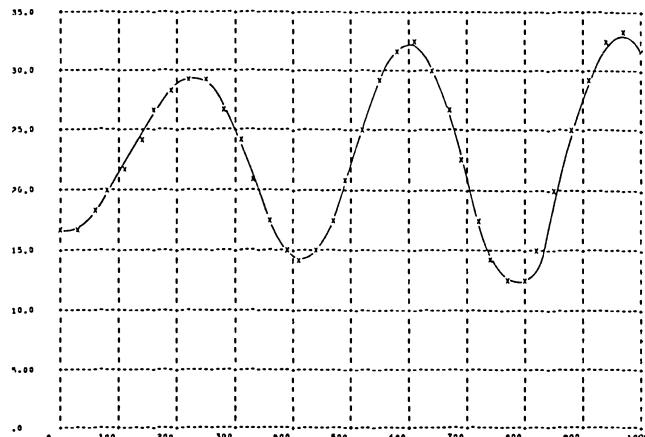
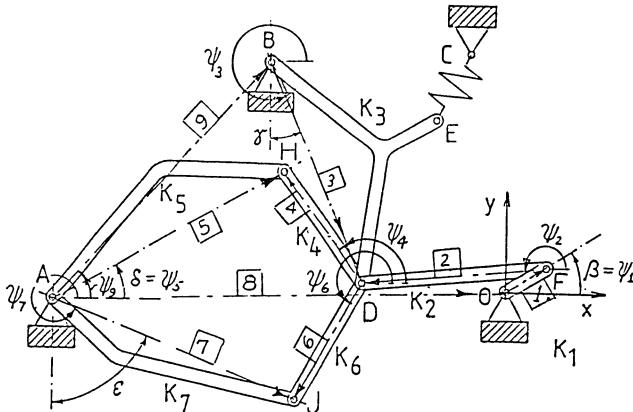


Fig. 6. Path of Mass Center. Speed $v_o = 31.25$ mi/hr (13.97 m/s); Road Profile: $y = 6(1 - \cos 2\pi x/360)$ in.

8. EXAMPLE 2. BENCHMARK PLANAR PROBLEM

The standard "comparison" or "benchmark" problem, used in this volume for planar multiloop mechanisms, is illustrated in Fig. 7. A suitable kinematic network (collection of directed edges and nodes) has been superposed on the figure. Treating point E as an unrestrained point loaded by a force (from the spring), the system may be seen to have three independent loops, which we take as OFDJA0, AJDHA, and AHDBA.



**Fig. 7. Multiloop Planar Mechanism.
Benchmark Problem**

The Lagrangian coordinates $(\psi_1, \dots, \psi_{10})$ shown all represent absolute angles, measured from the global x-axis. Note that an extensional spring attached to one of the links is fixed to the ground link at point C. The system starts from rest, under the influence of a constant torque motor driving the crank about the axis at O with a torque of 0.033 N-m. The geometric dimensions, initial angles, inertial characteristics of the system, and the stiffness and free length of the spring are given elsewhere in this volume; therefore they will not be listed here. However, in order to give readers a feeling for the amount of, and the form of, the input data required for such problems, we have shown, in Fig. 8, the complete input file required by DYSPAM.

Fig. 8. Input Data File for Example 2.

This input data is easily generated using any convenient program editor or word processor. However, it is significantly easier, and less error prone to use the interactive data preparation program DYMDATA, discussed below. DYSPAM was run with the data file shown, and produced the echo-print of the input data shown in Fig. 9. The specified external forces consist of a constant torque of 0.033 Nm acting on member 1, and the spring force acting at point E on member 3. The constant torque may be included as TOR(1) acting about the point of interest 1, as shown in the Given Force Table (see Fig. 9). However, position dependent spring forces must be described in the user-supplied subroutine FORCES, as follows:

```

C ***** USER-SUPPLIED EXPRESSIONS FOR FORCES FOLLOW *****
XB=-.03635465058
YB=.03273386291
XE=XB+.026907248*DCOS(PSI(3)+.837981225)
YE=YB+.026907248*DSIN(PSI(3)+.837981225)
D=SQRT((.014-XE)**2 + (.072-YE)**2)
F=4530*(D-.07785)
PSIC=DATAN2((.072-YE),(.014-XE))
FX(2)=F*DCOS(PSIC)
FY(2)=F*DSIN(PSIC)
      RETURN
      END

```

In interpreting the above code, note from the Points of Interest Table that point E is point of interest No. 8, and from the Given Force Table that Force set No. 2 acts through point 8 (point E). That is why the spring force components acting on point E are called FX(2), FY(2) in the subroutine Forces.

PROGRAM DYMCA EXAMPLE
PLANAR MECHANISM

NO. OF SPATIAL CONSTRAINTS (NS).....	0
NO. OF TEMPORAL CONSTRAINTS (NT).....	0
NO. OF LAGRANGIAN COORDINATES (NLAG).....	7
NO. OF MOVING LINKS (NLINK).....	7
NO. OF EDGES (NE).....	9
NO. OF JOINTS (NJNTD).....	0
NO. OF INDEPENDENT LOOPS (NLP).....	3
NO. OF POINTS OF INTEREST (NP1).....	8
NO. OF FORCE APPLICATION POINTS (NFAP).....	2
NO. OF UNKNOWN MOTION GENERATOR FORCES (NUF).....	0
OPEN LOOP INDEX (IOLP).....	0
PROCESSOR OPTION (IPROC).....	0
CHARACTERISTIC LENGTH (ELCHAR).....	.2000000-01
CHARACTERISTIC PERIOD (TCHAR).....	.1000000-01
CHARACTERISTIC DISPLACEMENT (OCHAR).....	.2000000+01

INITIAL TIME (PRMT(1)).....	.0000000-00
CHOSEN TIME INCREMENT (PRMT(3)).....	.1000000-02
PRINTOUT INTERVAL (PRMT(7)).....	.1000000-02
FINAL TIME (PRMT(2)).....	.3000000-01

EDGE DATA TABLE				
EDGE	LENGTH INDEX	LENGTH CONSTANT	ANGLE INDEX	ANGLE CONSTANT
1	IL(I)	CL(I)	IA(I)	CA(I)
1	0	.7000000-02	1	.0000000+00
2	0	.2800000-01	2	.0000000+00
3	0	.3500000-01	3	.0000000+00
4	0	.2000000-01	4	.0000000+00
5	0	.4000000-01	5	.0000000+00
6	0	.2000000-01	6	.0000000+00
7	0	.4000000-01	7	.0000000+00
8	0	.6938180-01	0	.3280570-01
9	0	.4810400-01	0	.8150930-01

EDGE ADJACENCY TABLE			
EDGE	EDGE END	ADJACENT	
I	IEDJAC(I,1)	IEDJAC(I,2)	EDGE END
1	-1	0	
2	2	3	
3	-3	9	
4	4	5	
5	-5	-7	
6	6	7	
7	-7	-8	
8	8	0	
9	-9	-5	

COMPLETE LOOP TABLE
EDGE NO.

LOOP NO.	1	2	3	4	5	6	7	8	9
1	1	1	0	0	0	1	-1	1	0
2	0	0	0	1	-1	-1	1	0	0
3	0	0	-1	-1	1	0	0	0	-1

POINTS OF INTEREST TABLE

POINT	LINK	EDGE	ANGLE	RADIUS	PRINT OPTION
1	LINK(I)	EDGE(I)	ALFPI(I)	RPI(I)	IOUT(I)
1	1	1	.0000000+00	.9237000-03	1
2	2	2	.0000000+00	.1150000-01	1
3	3	3	.5077550+00	.2145130-01	1
4	4	4	.0000000+00	.1421000-01	1
5	5	5	.3952830+00	.2483240-01	1
6	6	6	.0000000+00	.1421000-01	1
7	7	7	.3507500+00	.1307610-01	1
8	3	3	.8379800+00	.2690720-01	1

LINK INERTIA TABLE

LINK	REFERENCE POINT	MASS	M.O.I.
I	IPIL(I)	EM(I)	CMMOI(I)
1	1	.4325000-01	.2194000-05
2	2	.3650000-02	.4410000-06
3	3	.2373000-01	.5255000-05
4	4	.7060000-02	.5667000-06
5	5	.7050000-01	.1169000-04
6	6	.7060000-02	.5667000-06
7	7	.5498000-01	.1912000-04

GIVEN FORCE TABLE

FORCE INDEX	REFERENCE POINT	X-FORCE	Y-FORCE	MOMENT
I	IPIF(I)	FX(I)	FY(I)	TOR(I)
1	1	.0000000+00	.0000000+00	.3300000-01
2	8	.0000000+00	.0000000+00	.0000000+00

INITIAL CONDITIONS TABLE

(NOTE VARIABLE TYPES: 1-PRIMARY, 0-SECONDARY, -1-CONTROLLED)

INDEX	VARIABLE TYPE	INITIAL SET 1	INITIAL SET 2	DISPLACEMENT	INITIAL VELOCITY
I	JQ(I)	JOA(I)	PSI(I)	PSID(I)	
1	1	0	-.6199410-01	.0000000+00	
2	0	0	.3141590+01	.0000000+00	
3	0	1	.5167830+01	.0000000+00	
4	0	0	.2281100+01	.0000000+00	
5	0	0	.4875190+00	.0000000+00	
6	0	0	.4149500+01	.0000000+00	
7	0	0	.5943260+01	.0000000+00	

DEGREE OF FREEDOM= 1 (INCLUDES EFFECTS OF ALL CONSTRAINTS)

Fig. 9. Standard Form Echo-Print of Input Data

The standard output was intercepted by the user-supplied subroutine UPROC, mentioned above, and saved in a data file. This file was fed into the Lotus 123 plotting routine to produce the graphs shown in Figs. 10 — 13. These plots show no discernable difference from the published solutions for this problem.

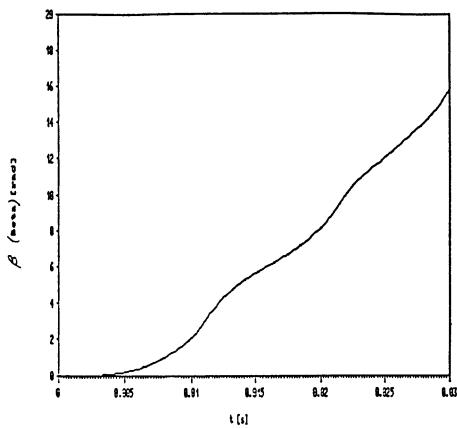


Fig. 10. $\psi_1=\beta$ Vs. time t

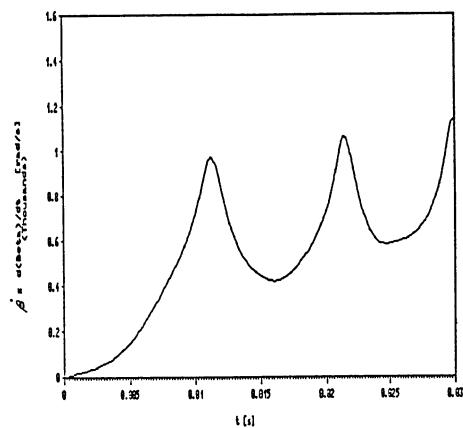


Fig. 11. $\dot{\psi}_1=\dot{\beta}$ Vs. time t

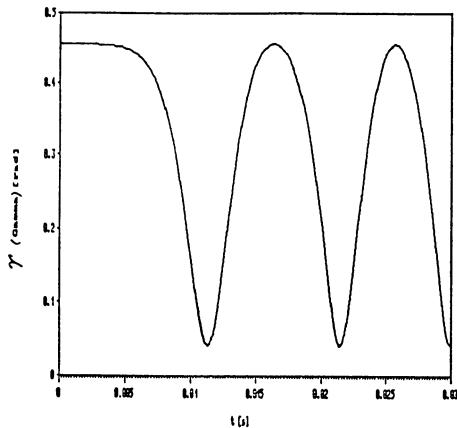


Fig. 12. $\psi_3 - 270^\circ = \gamma$ Vs. time t

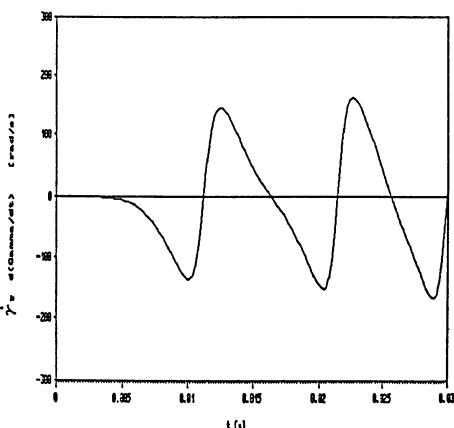


Fig. 13. $\dot{\psi}_3 = \dot{\gamma}$ Vs. time t

9. PRE- AND POST-PROCESSING OF DATA

DYMDATA is an interactive menu-oriented program for the generation of DYMACH input data files, such as that shown in Fig. 8. This program was written in the language True Basic, but does not require the True Basic compiler to be run. It can be run from an ".EXE" file on any computer (e.g. IBM PC compatibles) which utilizes the DOS operating system. It can also be run from True Basic on non-DOS machines (e.g. Macintosh) which support True Basic. The program has many useful error trapping features and other conveniences for storing and modifying data files.

DYMACH has convenient standard forms of printed output. However any computed result from DYMACH is easily stored for later use (e.g. plotting) by use of the user-supplied subroutine UPROC, as was illustrated in example No. 2 above.

10. CONCLUSIONS

DYMACH is a general purpose computer program for analysis and simulation of dynamics problems in planar mechanisms. It has been described in sufficient detail to suggest how a system is modelled, how input data is organized, and what level of effort is required to write user-supplied subroutines, which are only necessary when specifying motion constraints (other than those due to standard joints, and loop closure), or when specifying variable forces. The program finds positions, velocities, and accelerations for any points of interest to the user. It also finds internal joint forces (reactions), and permits the post-processing of output data in a variety of ways for graphical or numerical studies. An interactive menu oriented data preparation program is also available.

Acknowledgements:

The author appreciates the able assistance provided by Evan Strassberg and Kai Lo, both graduate students at the University of Pennsylvania, in the preparation of this paper. He also acknowledges the efforts of former senior student Michael Blaine, in the development of program DYMDATA.

REFERENCES

- [1] Paul, B.: A Unified Criterion for the Degree of Constraint of Plane Kinematic Chains, *Journal of Applied Mechanics*, vol. 27, *Trans ASME Ser E*, vol. 82, 196-200, discussion 751-752, 1960
- [2] Paul, B. and Krajcinovic, D.: Computer Analysis of Machines With Planar Motion—Part I: Kinematics, *Journal of Applied Mechanics*, *Trans. ASME, Series E*, vol. 37, 697–702, 1970
- [3] Paul, B. and Krajcinovic, D.: Computer Analysis of Machines With Planar Motion—Part II: Dynamics, *Journal of Applied Mechanics*, *Trans. ASME, Series E*, vol. 37, 703–712, 1970
- [4] Paul, B.: Analytical Dynamics of Mechanisms - A Computer-Oriented Overview, *Mechanisms and Machine Theory*, Vol. 10, 481-507, 1975
- [5] Paul, B.: *Kinematics and Dynamics of Planar Machinery*, Prentice-Hall, Englewood Cliffs, NJ, 1979
- [6] Hud, G.: *Dynamics of Inertia Variant Machinery*, Ph.D. Dissertation, University of Pennsylvania, Philadelphia, 1976
- [7] Amin, A.: *Automatic Formulation and Solution techniques in Dynamics of Machinery*, Ph.D. Dissertation, University of Pennsylvania, Philadelphia, 1979
- [8] Paul, B. and Amin, A.: User's Manual for Program DYMPC. Available through B. Paul.
- [9] Paul, B.: Machine Systems, in Chapter 12 of *Mechanical Design and Systems Handbook*, McGraw-Hill, N.Y.; Ed. by Rothbart, H.A., 1985

DYSPAM (DYnamics of SPAtial Mechanisms)

ABSTRACT

DYSPAM is a computer program, written in FORTRAN, for the analysis and simulation of the Kinematics, Statics, and Dynamics of spatial interconnected systems of bodies. The system may include both open kinematic chains, as typified by robot manipulators, and closed kinematic chains, as typified by automotive suspensions and general linkage machinery. DYSPAM automatically generates the differential equations of motion, using Lagrange's form of d'Alembert's Principle (Virtual Work). The program finds displacements, velocities, accelerations, joint reactions, motor torques, etc. for systems with multiple degrees of freedom which are subjected to user-supplied forces, and geometric constraints. A brief description is given of modelling techniques and of input/output procedures. The capability for post-processing (e.g. plotting) data produced by DYSPAM is provided so that users may utilize their own graphics hardware/software facilities. Interactive pre- and post-processors for DYSPAM, using personal computers, are available to simplify data input and to provide for graphics and animation.

1. INTRODUCTION

DYSPAM is designed to perform the following types of analyses for spatial multi-degree of freedom systems of interconnected bodies:

1. **KINEMATICS** — Time-varying position, velocity, and acceleration for all bodies, and for user-selected *points of interest*, are computed for specified driving links or for other motion-generating means specified by the user.
2. **STATICS** — Statics analysis is used to solve two kinds of problems. Either an *equilibrium configuration* is determined from user-supplied forces and constraints, or a set of *equilibrating forces* is determined from a user-specified configuration. For either case, joint reactions are computed, if desired.
3. **DYNAMICS** — Position, velocity, and acceleration, for all bodies of the system, and for selected points of interest are computed as a function of time by numerical integration of the differential equations of motion, which are generated automatically, subject to user-supplied forces and *motion generators* (imposed geometric constraints). When motion generators are used, the required driving forces, e.g. motor torques, can be determined. If desired, joint reaction forces and moments can also be computed.

The principal reference for DYSPAM is Schaffa [1]. The program evolved from earlier works, including Paul and Krajcinovic [2],[3], Paul [4], Paul [5], Paul, [6], and from the program DYMPC, which was restricted to planar motions (see Paul [7], and Amin [8]).

2. METHOD OF ANALYSIS

The configuration of a mechanism is determined by a set of time-dependent *Lagrangian coordinates*, denoted by the symbols $\psi_1, \psi_2, \dots, \psi_M$. These variables represent the *relative joint displacements*, which may be either angles (e.g. for revolute joints) or lengths (e.g. for prismatic joints).

The Cartesian coordinates of a typical point, referred to a *global* system of axes, can be expressed as :

$$x = f_1(\psi_1, \dots, \psi_M), \quad y = f_2(\psi_1, \dots, \psi_M), \quad z = f_3(\psi_1, \dots, \psi_M)$$

To permit the imposition of a wide variety of geometric constraints, users may specify *holonomic* relationships among the Lagrangian coordinates, and time t , of form:

$$g_i(\psi_1, \dots, \psi_M, t) = 0, \quad (i=1, \dots, N_{ac})$$

where N_{ac} is the number of such relationships — or in the *nonholonomic* form:

$$a_{11} \dot{\psi}_1 + a_{12} \dot{\psi}_2 + \dots + a_{1M} \dot{\psi}_M + a_{10} t = 0, \quad (i = 1, \dots, N_{ac})$$

where a_{ij} are user supplied functions of position and time.

Such relationships are specified in a FORTRAN standard form within a user-supplied subroutine called MOTGEN (MOTion GENeration constraint). MOTGEN is also used to solve the so-called *inverse kinematics* problem, which is of great interest in robotics. In these problems, it is required to find the joint angles and displacements needed to place a given body in a given position and orientation. The equations of constraint associated with the closure of kinematic loops are generated automatically.

Of the M Lagrangian coordinates, F will be designated as *generalized coordinates* (or independent coordinates). The remaining coordinates are called *secondary coordinates*, and they are computed internally from the generalized coordinates. The subset of Lagrangian coordinates which serve as generalized coordinates may be chosen automatically by DYSPAM, or they may be chosen by the user. The number and choice of generalized coordinates may vary with the state of the system (e.g. at singular configurations).

The method of solution used for each of the different categories of problems is briefly described here:

1. **KINEMATICS** — The user specifies the initial values of the generalized coordinates of his choice. If a generalized velocity is to remain constant (e.g. if a driving crank has constant angular velocity), it is only necessary to specify its initial (and time invariant) value. If the generalized velocity varies with time, this time dependence must be explicitly defined in the subroutine MOTGEN. The secondary variables are computed internally from the generalized coordinates by solving the nonlinear algebraic constraint equations via the Newton-Raphson algorithm.
2. **STATICS** — For the unknown Equilibrium configuration of a mechanism subjected to geometric constraints, and a given set of forces, the solution must satisfy the *system equilibrium equations* (vanishing generalized forces), and the geometric constraint equations. These two sets of nonlinear algebraic equations are solved simultaneously by a Newton-Raphson algorithm. The user may specify arbitrary applied forces (possibly dependent on time, position, and velocity) via a user-supplied subroutine FORCES.
3. **DYNAMICS** — The differential equations of motion are generated automatically using Lagrange's form of d'Alembert's Principle (virtual work). When geometric constraints exist, DYSPAM (unlike most competing programs) does not utilize Lagrange multipliers. The equations of motion are formulated according to the method of excess differential equations, as described in Paul [5,6], using a fourth order Runge-Kutta algorithm with automatic step control. If the number of differential equations generated equals the number of independent driving variables, DYSPAM recognizes that the motion is determined solely by kinematics, and the solution is obtained as explained under the heading of **KINEMATICS** above. The user may specify arbitrary applied forces (possibly dependent on time, position, and velocity) via the user-supplied subroutine FORCES.

HOMOGENEOUS TRANSFORMATIONS

We utilize the notation of the *homogeneous transformation* as used by Denavit and Hartenberg [9] and Paul [10]. Let $\mathbf{x}^A \equiv \{x,y,z\}^A$ represents the column vector of cartesian coordinates in a coordinate frame A, for a given point P. Similarly, let \mathbf{x}^B represent the coordinates of the same point in a frame B. Let us also define by $\mathbf{d}^A \equiv$

$\{d_x, d_y, d_z\}^A$ the vector, resolved in frame A, from the origin of frame A to the origin of frame B, and let the unit basis vectors of the two frames be denoted by e_i^A and e_i^B ($i=1,2,3$). Then the position vectors of point P, as observed from the two frames are related by the following transformation:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^A = \begin{bmatrix} x^A \\ y^A \\ z^A \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & d_x \\ r_{21} & r_{22} & r_{23} & d_y \\ r_{31} & r_{32} & r_{33} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{AB} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^B = \mathbf{T}^{AB} \mathbf{X}^B$$

where

$$r_{ij} \equiv e_i \cdot e_j$$

The orthonormal matrix $\mathbf{R} = [r_{ij}]$ represents the rotation of frame B relative to frame A, and we call the four dimensional vector $\mathbf{X} = [x \ 1]^t$ the *extended position vector*. Note that the matrix \mathbf{T}^{AB} contains all the information necessary to fully specify both the rotation and the translation of frame A relative to frame B. The matrix transformation described above is said to be a *homogeneous transformation*.

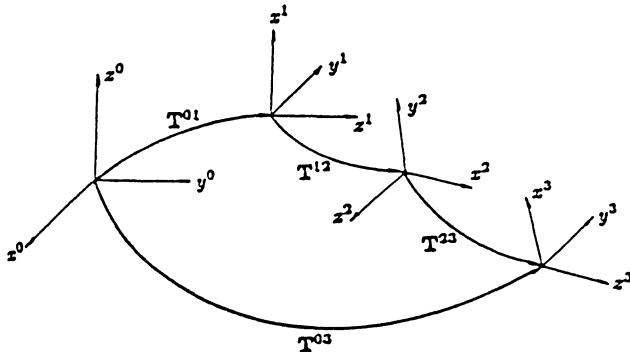


Fig. 1. Directed Transform Graph

Figure 1 depicts a set of distinct coordinate frames with axes $\{x,y,z\}^i$, ($i=0,1,2,3$). We shall think of frame 0 as the *global frame*. If a point has coordinates $(x,y,z)^3$ in frame 3, it is readily shown that its coordinates in the global frame are expressed by the homogeneous transformation

$$\mathbf{X}^0 = \mathbf{T}^{01} \mathbf{T}^{12} \mathbf{T}^{23} \mathbf{X}^3 = \mathbf{T}^{03} \mathbf{X}^3$$

where

$$\mathbf{T}^{03} = \mathbf{T}^{01} \mathbf{T}^{12} \mathbf{T}^{23}$$

The chain of transformations represented by this product of \mathbf{T} matrices is depicted in Fig. 1 as a *directed transform graph* (see R. P. Paul [10]).

3. MODELLING OF MECHANISM

In this section we describe the modelling procedures required by DYSPAM. Note that we do *not* use the *modelling* technique of Denavit and Hartenberg [9].

3.1 Body and Joint Triads

Each body of the system has a *Body Reference Triad*¹ (abbreviated as *BRT*) embedded in it. This is used to define features of the body (e.g. points of interest, or other embedded triads) relative to the body. It is also used to describe the position and orientation of the body relative to a *global (fixed) frame*. In particular, for each joint on a body there is a *joint triad* embedded in the body; there will be a corresponding joint triad embedded in the body connected to the first body by the joint in question. The relative motion of the two bodies is described by a *joint transformation matrix* which relates these two triads, as described in Section 3.2. A body may have several joints attached to it. If N is the identification number of the body, and joints 3,7,9 are attached to that body, the transformation matrices which locate the joint triads relative to the BRT are designated by

$$\mathbf{T}^{N3}, \mathbf{T}^{N7}, \mathbf{T}^{N9}$$

as illustrated in Fig. 2, for bodies A and B, and joint i. Note that Fig. 2 illustrates a notational convention for the inverse of a transformation matrix, *viz.*

$$\mathbf{T}^{iB} = (\mathbf{T}^{Bi})^{-1}$$

3.2 Modelling of Joints

The relative motion of the two bodies which are joined at joint i is described by a *joint transformation matrix* \mathbf{J}^i , as illustrated schematically in Fig. 2. Figure 2 is a directed transform graph which illustrates that the configuration of the BRT of body B relative to that of body A is given by the matrix product

$$\mathbf{T}^{AB} = \mathbf{T}^{Ai} \mathbf{J}^i \mathbf{T}^{iB}$$

¹ The word *triad* denotes a system of orthogonal Cartesian axes together with an associated set of unit base vectors.

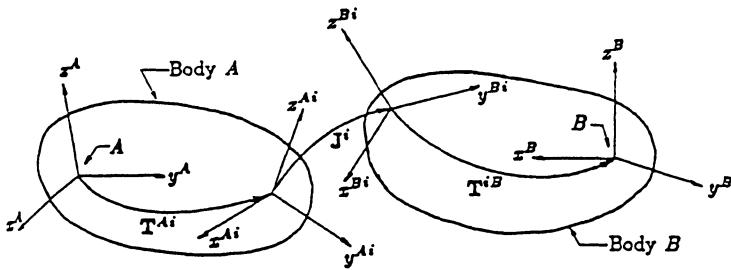


Fig. 2. Body Reference Triads for Bodies A and B; Joint Triad for Joint j.

To illustrate the nature of the joint matrices J^i , we give two examples. For the simple revolute joint illustrated in Fig. 3, the joint matrix is

$$J^i = \begin{bmatrix} \cos \theta_{i,1} & -\sin \theta_{i,1} & 0 & 0 \\ \sin \theta_{i,1} & \cos \theta_{i,1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

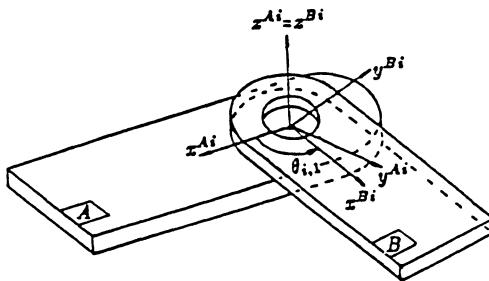


Fig. 3. Revolute Joint

Similarly, for the universal joint shown in Fig. 4, the joint transformation matrix is

$$J^i = \begin{bmatrix} \cos \theta_1 & \cos \theta_2 & -\cos \theta_1 & \sin \theta_2 & \sin \theta_1 & 0 \\ \sin \theta_1 & \cos \theta_2 & -\sin \theta_1 & \sin \theta_2 & -\cos \theta_1 & 0 \\ \sin \theta_2 & 0 & \cos \theta_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

where θ_1 and θ_2 represent the angles $\theta_{i,1}$ and $\theta_{i,2}$ shown in Fig. 4.

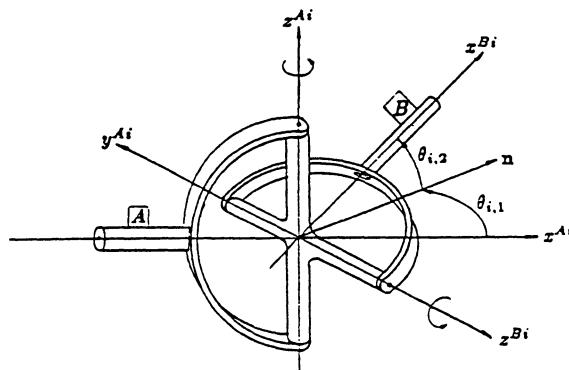


Fig. 4. Universal Joint

The various types of joints built in to DYSPAM are shown in Fig. 5; other types of joints may be simulated by the superposition of these joint types.

joint type	schematic representation	number of relative d.o.f.	joint type	schematic representation	number of relative d.o.f.
Revolute		1	Universal		2
Prismatic		1	Spheric		3
X-Prismatic		1	Planar		3
Screw		1	Gear		1
Cylindric		2			

Fig. 5. Standard Joint Types

3.3 Modelling The Kinematic Network

Bodies are numbered sequentially from zero (for the ground link). The topology of the system is summarized in a user-supplied *joint description table*. Each line of this

table corresponds to one simple joint. The order in which the two bodies connected by that joint are entered determines the positive sense of the joint variable, which is measured from the first to the second of the joint triads for that joint (see Figs. 3 and 4 for examples). The *type* of the joint (e.g. see Fig. 5) is also entered into the joint description table.

3.4 Modelling Body Geometry and Inertia

The term *geometric entity* includes points of interest, mass centers, coordinate frames (triads), etc. In addition to the joint triads, described above, the user must specify an *inertia triad* located at the mass center of each body. In defining this triad, the user is free to take advantage of inertial symmetries (principal axes of inertia), or to utilize any special feature of the geometry that is convenient. Any geometric entity attached to a body is specified with respect to the BRT (body reference triad). The user also creates a *points of interest table* which specifies the local position of all mass centers, points where forces are applied, and other points of interest to the user.

Inertia properties are specified, for each body, by the mass and by the inertia tensor (matrix) specified relative to the inertia triad. The units of mass and length are arbitrary, but must be consistent. The inertial characteristics for each body are entered into a *body inertia table*.

3.5 Modelling User-Supplied Forces and Moments

Forces² can be constants (e.g. body weight) or functions of time, position, and velocity. Forces may be applied to any body of the system, but the components of the forces must be specified in the global coordinate frame. A user-supplied *force table* contains the relevant data for each point of interest where one or more *constant* force components are applied. Any force component which is variable in magnitude or direction must be defined by the user in a standard subroutine called FORCES.

3.6 Auxiliary Constraints

DYSPAM gives users the opportunity to specify arbitrary relationships among the Lagrangian coordinates at all times. These relationships are called *auxiliary constraint equations*. They are of the general form

$$g_i(\psi_1, \psi_2, \dots, \psi_M, t) = 0, \quad (i=1, n_{ac})$$

² When not otherwise indicated, the word "forces" stands for "forces and moments."

where t is time, and n_{ac} is the number of such relationships to be specified.

DYSPAM requires the user to provide FORTRAN expressions for the functions g_i , and for all its nonzero derivatives, in a standard subroutine MOTGEN. The FORTRAN names to be used for g_i and its derivatives are:

$G(I)$	for g_i
$DG(I,J)$	for $\partial g_i / \partial \psi_j$
$DTDG(I,J)$	for $d(\partial g_i / \partial \psi_j) / dt$
$DTG(I)$	for $\partial g_i / \partial t$
$DTDTG(I)$	for $d(\partial g_i / \partial t) / dt$

To illustrate the use of auxiliary constraints, let us consider the case where the acceleration corresponding to variable 5 is to be held constant; i.e.:

$$\begin{aligned}\ddot{\psi}_5 &= a \\ \dot{\psi}_5 &= at + v_0 \\ \psi_5 &= at^2/2 + v_0 t + s_0\end{aligned}$$

where a is the desired constant acceleration, v_0 is the initial velocity, and s_0 is the initial displacement. The desired form for the constraint equation is therefore:

$$g_1 = \psi_5 - at^2/2 - v_0 t - s_0 = 0$$

The nonzero derivatives of g_1 are given by

$$\begin{aligned}\partial g_1 / \partial \psi_5 &= 1 \\ \partial g_1 / \partial t &= -at - v_0 \\ d(\partial g_1 / \partial t) / dt &= -a\end{aligned}$$

The user must place the following FORTRAN statement where indicated in the supplied dummy version of subroutine MOTGEN:

```
C$$$$...EXPRESSIONS FOR USER SUPPLIED CONSTRAINTS FOLLOW...$$$  
G(I)=PSI(5)-A*TIME**2/2.-VO*TIME-SO  
DG(1,5)=1  
DTG(5)=-A*TIME-VO  
DTDTG=-A
```

Subroutine MOTGEN is also used to specify *nonholonomic* constraints; i.e. when known algebraic relationships exist between derivatives of the Lagrangian coordinates but no corresponding relationship of the form $g(\psi_1, \dots, \psi_M, t) = 0$ exists. This approach is applied to the classic example of the rolling coin in Schaffa [1].

3.7 Inverse Kinematics

Users may specify the motion of selected points in the system. For example, one might specify the *path* described by a robot's wrist, and the orientation of the robot's hand. DYSPAM will then find the necessary values of the joint angles necessary to accomplish the task. This is an illustration of the so-called problem of *inverse kinematics*. More generally, the user provides an *inverse kinematics table*. In this table, up to ten points of interest are selected, and the user indicates by a simple code whether he is going to specify one, two or three of the global Cartesian coordinates for each such point. An example of inverse kinematics is given in Schaffa [1], where the wrist of a Stanford manipulator is made to move along a straight line while the end effector (hand) maintains a fixed orientation.

3.8 Initial Conditions

In a statics analysis, the user must provide initial values for all independent Lagrangian coordinates. When auxiliary geometric constraints are present, the user must also provide estimates of the initial values for all the dependent (secondary) coordinates. These approximate values are iteratively adjusted by DYSPAM until all constraint relations are satisfied.

For a kinematics or dynamics analysis, the physics of the problem requires that the initial values of the generalized coordinates, and of the generalized velocities, be given. It is also necessary that initial estimates be given of all secondary coordinates. These estimates will be refined internally by DYSPAM.

3.9 Springs and Dampers

An extensional spring may act between any two points of interest, and a rotational spring may act about any revolute or spherical joint. For linear elasticity and damping, the parameters for the springs are included in a *spring/damper* table. Nonlinear springs may be accounted for by means of simple FORTRAN statements in the user-supplied subroutine FORCES.

4. PRE- AND POST-PROCESSING OF DATA

Input may be generated in a variety of ways. The original, and still valid method is to use a text editor to create a text file with the input data arranged in lines and columns, as explained in the User's Manual. More recently, a Pre-Processor program

(called DYSIN, for DYSpam INput) was written for DYSPAM (see Strassberg [11]). This preprocessor is a menu driven interactive program written in the True Basic language. After the user has entered, edited, and checked the data to his satisfaction, the program produces a data input file in the format that DYSPAM will accept. The pre-processing program will run on any computer (e.g. IBM PC, XT, and AT compatibles) that supports the DOS operating system, without the need to run True Basic. It will also run from True Basic on any computer which supports True Basic [e.g. IBM (PC, XT, or AT) compatibles, Apple Macintosh, Amiga, Commodore, and others].

The *standard* (numerical) output from DYSPAM goes to the computer screen, and to a variety of output files. Each time a solution state is found (i.e the set of all Lagrangian coordinates, at a specified time), the user has the opportunity to process that information immediately, or store it for later processing, in a user supplied subroutine called UPROC. This permits users to customize the output to their own hardware/software/graphics facilities.

An alternative post-processing method for DYSPAM, with strong graphics and animation capabilities has recently been developed by Strassberg [11]. This post-processing PROGRAM, called ANIMATE, will run on any computer (e.g. IBM PC, XT, and AT compatibles) that supports the DOS operating system, without the need to run True Basic. For graphics output, and animation, the computer should have a graphics video board (e.g. EGA, VGA, Hercules) and a compatible video monitor. Like the preprocessor, the postprocessor will also run from True Basic on any computer which supports that language [e.g. IBM (PC, XT, AT) compatibles, Apple Macintosh, Amiga, Commodore, etc.].

5. EXAMPLES

We consider the "benchmark" examples defined in detail elsewhere in this volume.

5.1 Example 1. Spatial Motion of Robot.

Figure 6 shows the example robot , which consists of 3 bodies, interconnected so as to produce a total of five degrees of freedom. We have chosen to model the joints as shown in the figure. Figure 7^A is a representation of the robot, generated by the post-processing program ANIMATE.³

³ A superscript A next to a Figure No. citation indicates that the figure was generated by the post-processing program ANIMATE, and was reproduced directly from the computer screen.

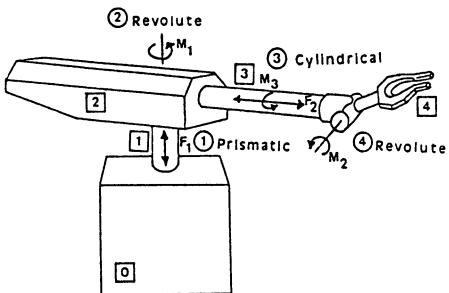


Fig. 6 Robot of Example 1

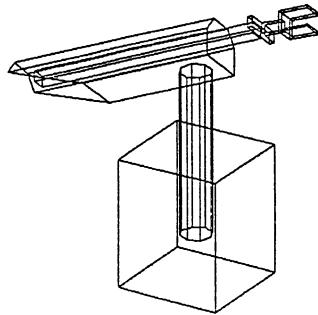
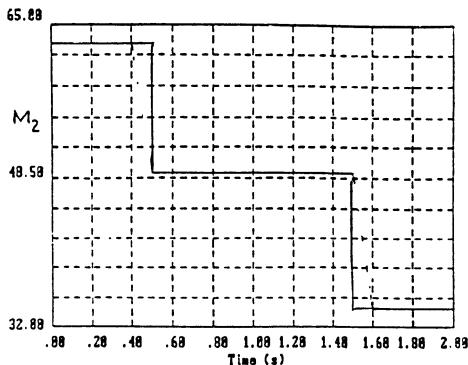
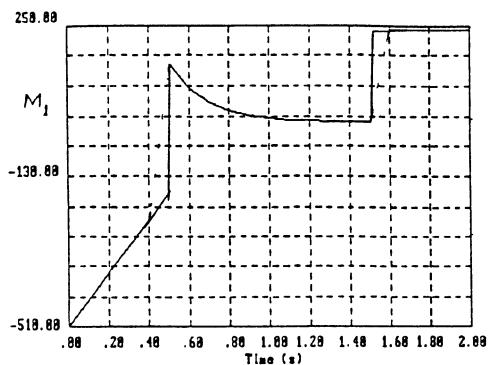


Fig. 7 Typical Animation Frame

Figures 8^A and 9^A illustrate the discontinuous nature of the applied moments (which are programmed by simple FORTRAN statement in the user-supplied subroutine FORCES).

Fig. 8 Hand Torque M_2 [N-m]
Vs. time t [s]Fig. 9 Tower Torque M_1 [N-m]
Vs. time t [s]

If you wish to prepare input for DYSPAM with the aid of the interactive program DYSIN, you will be greeted by a sequence of screen menus with detailed instructions on how to proceed. Two typical menus are indicated in Figs. 10 and 11. Figure 11 corresponds to an example in which 6 forces (NFOR=1,2, . . . 6) are to be specified by the user, who merely fills in the highlighted blank spaces on the screen.

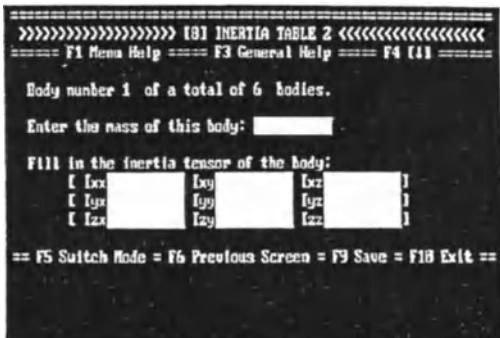


Fig. 10 Mass and Inertia Tensor Menu

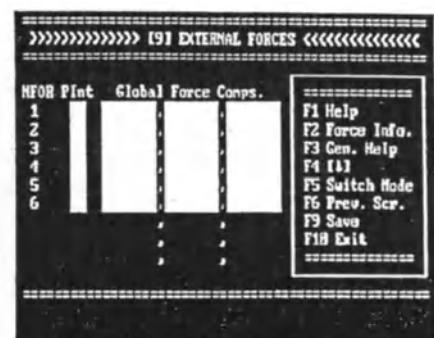


Fig. 11 External Forces Menu

As explained in Sec. 3.2, for each joint, a joint triad is fixed in each adjoining body, with their z-axes collinear. Figure 12 shows all the joint triads for this example. Note the following pairs of collinear axes, and Lagrangian coordinates:

Joint 1 (prismatic):	(z_{01}, z_{11})	ψ_1
Joint 2 (revolute):	(z_{12}, z_{22})	ψ_2
Joint 3 (cylindrical):	(z_{23}, z_{33})	ψ_3, ψ_4
Joint 4 (Revolute):	(z_{34}, z_{44})	ψ_5

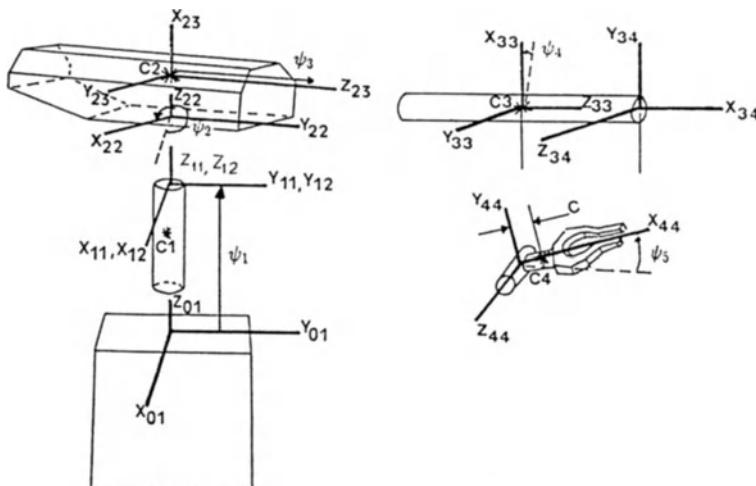


Fig. 12. Joint Triads and Lagrangian Coordinates

As mentioned above, the output from DYSPAM is easily captured and processed in a user-supplied subroutine UPROC, so that the numeric data may be processed by the user in accordance with whatever plotting utilities he works with. In this fashion, we recorded all the Lagrangian coordinates (ψ_1, \dots, ψ_5) at each time step, and used our local facilities to plot them versus time. All of these plots agreed without discernable error with the published solutions to this sample problem. Typical results are shown in Figures 13 and 14.

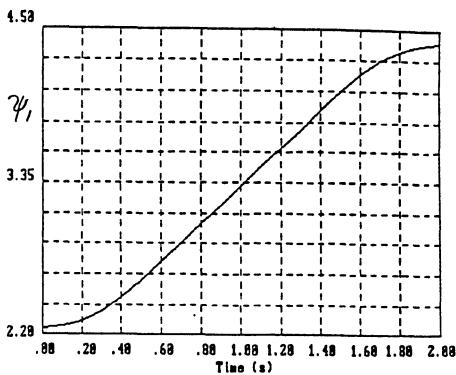


Fig. 13 Lift of tower ($\psi_1=z_1$) [m]
Vs. time (t) [s]

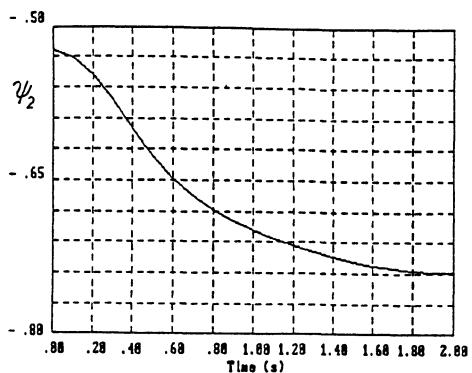


Fig. 14 Rotation of tower ($\psi_2=GA1$) [rad]
Vs. time (t) [s]

5.2 Example 2. Planar mechanism.⁴

This example illustrates DYSPAM's ability to handle closed loop and planar problems. The standard "benchmark" problem for planar multiloop mechanisms is shown in Fig. 15. In that figure, the Lagrangian coordinates (ψ_1, \dots, ψ_{10}) shown all represent relative angles at revolute joints. Note that ψ_3, ψ_4 , and ψ_5 represent the angular displacements for the three simple revolute joints which have a common axis of rotation. Also note that an extensional spring attached to one of the links is fixed to the ground link at point K. The system starts from rest, under the influence of a constant torque motor driving the crank about the axis at O with a torque of 0.033 N/m.

⁴ From the point of view of execution speed and ease of data input, a program such as DYMACH (5,7), which was designed expressly for planar problems, is preferable to a more general program, such as DYSPAM which is capable of treating both spatial and planar problems.

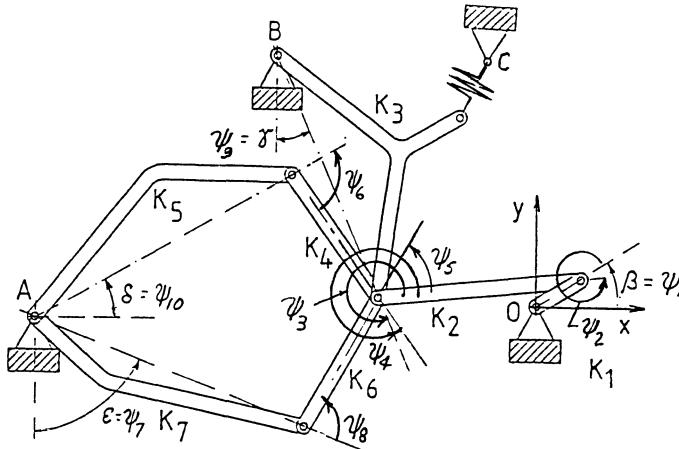


Fig. 15. Seven body, multiloop, planar mechanism

The geometric dimensions, initial angles, and inertial characteristics of the system are given elsewhere, and will not be repeated here. However, in order to give readers a feeling for the amount and form of the input data required for such problems, we have shown, in Fig. 16, the complete input file required by DYSPAM, for this example.

```

Planar Problem
10,7,0,0,9,0,1,1,1
0,0,1,0,0,0,1,/           Joint-Body Table
0,1,1
1,2,1
2,3,1
2,4,1
2,6,1
4,5,1
0,7,1
7,6,1
0,3,1
0,5,1
0,1,/                     Joint Triads Total of 14
0,7,-.06934442016,-.00227570607,0,-.06934442016,-1,0,
-.06934442016,-.00227570607,1
0,10,-.06934442016,-.00227570607,0/
0,9,-.03635465058,.03273386291,0,-.03635465058,-1,0
-.03635465058,.03273386291,1
1,1/
1,2,.007/
2,2/
2,3,-.028/
2,4,-.028/
2,5,-.028/
3,3,.035/
3,9/
4,4/
4,6,-.02/
5,10/
5,6,.04/
6,5/
6,8,-.02/
7,7/
7,8,.04/
1,.0009237/               inertias
.04325,.8*0.,2.194d-6
2,-.0115/
.00365,.8*0.,4.41d-7
3,.018745,.01043/
.02373,.8*0.,5.255d-6
4,-.01421/
.00706,.8*0.,5.667d-7
5,.02308,.009163/
.07050,.8*0.,1.169d-5
6,-.01421/
.00706,.8*0.,5.667d-7
7,.01228,-.004493/
.05498,.8*0.,1.912d-5
1,0,.0,.033             moment
-.061994103/            initial conditions
0,/
5.24/
5.48/
1.07/
1.34/
1.2308/
1.36/
.45544/
.487518/
0.,.001,.03             time
28.1d-6,.1d-5,10
1,.0009237/              points of interest
2,-.0115/
3,.018745,.01043/
4,-.01421/
5,.02308,.009163/
6,-.01421/
7,.01228,-.004493/
3,.018,.02/
0,.014,.072/
1
.01,.1,.1
8,9,.07785,4530./

```

Fig. 16. Input Data File for Example 2.

The amount of input data required is quite moderate, considering the complexity of the system. It is therefore not too onerous a task to generate the input file, using any convenient program editor or word processor. Nevertheless, it is significantly easier, and less error prone to use the interactive data preparation program DYSIN, discussed above. DYSPAM was run with the data file shown, and the standard output was intercepted by the user-supplied subroutine UPROC, as described above, which utilized an available plotting package to produce the results shown in Figs. 17—20. These plots show no discernable difference from the published solutions for this problem.

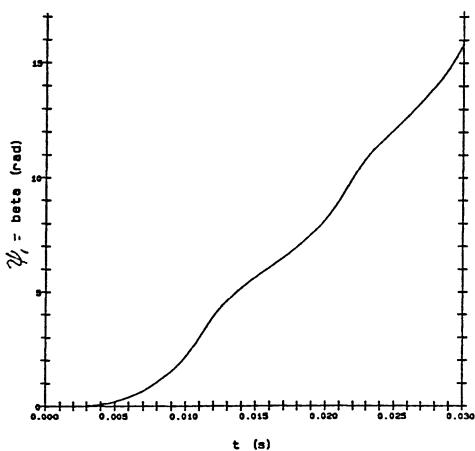


Fig. 17. $\psi_1 = \beta$ Vs. time t

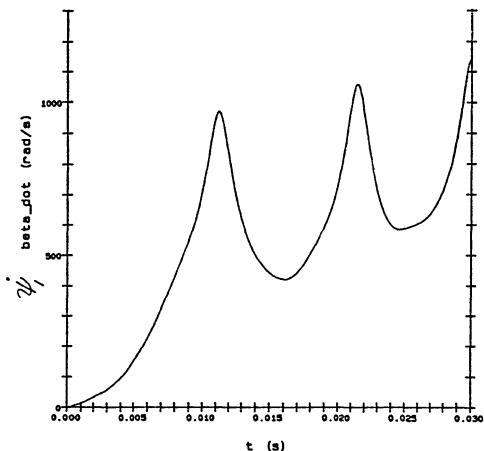


Fig. 18. $\dot{\psi}_1 = \dot{\beta}$ Vs. time t

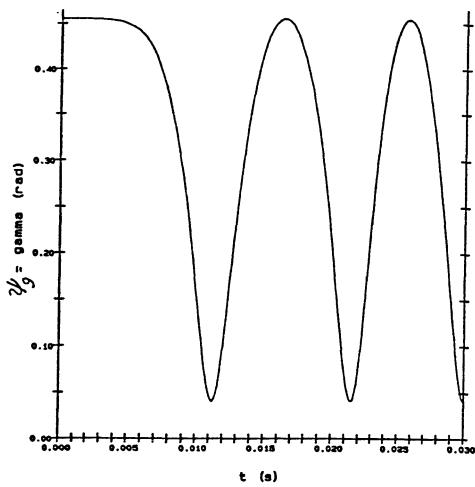


Fig. 19. $\psi_9 = \gamma$ Vs. time t

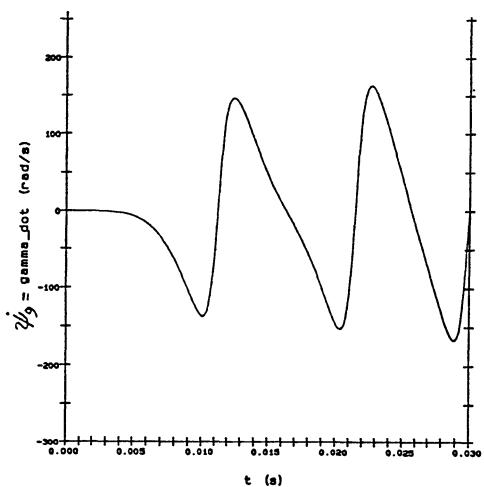


Fig. 20. $\dot{\psi}_9 = \dot{\gamma}$ Vs. time t

Acknowledgements:

We greatly appreciate the help of Evan Strassberg, who worked out Example 1 and the associated graphics with the aid of the pre- and post processors (DYSIN and ANIMATE) described in Strassberg [11]. We also appreciate the assistance of Ernest Otani and of Philip Lee who have given us the benefit of their experience using DYSPAM as a component of a computer graphics facility.

REFERENCES

- [1] Schaffa, R.B.: *Dynamic Analysis of Spatial Mechanisms*, Ph.D. Dissertation, University of Pennsylvania, Dept. of Mechanical Engineering and Applied Mechanics, Philadelphia, PA, 19104, 1984
- [2] Paul, B. and Krajcinovic, D.: Computer Analysis of Machines With Planar Motion—Part I: Kinematics, *Journal of Applied Mechanics, Trans. ASME, Series E*, vol. 37, 697–702, 1970
- [3] Paul, B. and Krajcinovic, D.: Computer Analysis of Machines With Planar Motion—Part II: Dynamics, *Journal of Applied Mechanics, Trans. ASME, Series E*, vol. 37, 703–712, 1970
- [4] Paul, B.: Analytical Dynamics of Mechanisms - A Computer-Oriented Overview, *Mechanisms and Machine Theory*, Vol. 10, 481-507, 1975
- [5] Paul, B.: *Kinematics and Dynamics of Planar Machinery*, Prentice-Hall, Englewood Cliffs, NJ, 1979
- [6] Paul, B.: Computer Oriented Analytical Dynamics of Machinery, in *Computer Aided Analysis and Optimization of Mechanical Systems Dynamics, (NATO ASI Series F, Vol. 9)*, Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, Ed. by E. Haug, 1984.
- [7] Paul, B.: Dynamics Analysis of Machinery Via Program DYMPC, Society of Automotive Engineers, Warrendale, Pa. Paper No. 770049 (1-11), 1978.

- [8] Amin, A.U.: *Automatic Formulation and Solution Techniques in Dynamics of Machinery*, Ph.D. Dissertation, University of Pennsylvania, Department of Mechanical Engineering and Applied Mechanics, Philadelphia, PA, 19104, 1979
- [9] Denavit, J. and R. S. Hartenberg: "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices," *J. of Applied Mechanics*, Vol. 22, Trans. ASME, Vol. 77, 215-221, 1955
- [10] Paul, R. P.: *Robot Manipulators: Mathematics, Programming and Control*, MIT Press, Cambridge, Mass., and London, 1981.
- [11] Strassberg, E.: *A Pre- and Post-Processor for Dynamic Simulation Using Program DYSPAM*, M.S Thesis, University of Pennsylvania, Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, Pa 19104, 1989

MESA VERDE - A General-purpose Program Package for Symbolical Dynamics Simulations of Multibody Systems

J. Wittenburg¹⁾, U. Wolz²⁾, A. Schmidt³⁾

Abstract

The purpose of this paper is to present the program package MESA VERDE for dynamics simulations of general multibody systems. The program name stands for "MEchanism, SAtellite, VEhicle and Robot Dynamics Equations". MESA VERDE generates in symbolical form a minimal set of nonlinear differential equations for large motions as well as expressions for kinematical quantities and for constraint forces in joints. The program package is used by university institutions as well as by engineering consulting companies and by engineers in industry. It is commercially available at the third author's address.

1. Introduction

Multibody systems in the sense of this paper are found in almost all branches of industry. Some typical examples are ground vehicles or vehicle components (wheel suspension systems for cars, motorcycles, truck-trailer units etc), industrial robots, biomechanical systems (the human body), spacecraft and many kinds of mechanisms ranging from mechanical switches to plotters and to aircraft landing gear mechanisms. All of these multi-body systems require dynamics simulations either in the design phase or in the operating phase.

In multibody systems the individual bodies are in large motion

1) Prof., Institute of Technical Mechanics, University of Karlsruhe, D-7500 Karlsruhe

2) Dr.-Ing., Zahnradfabrik Friedrichshafen,
D-7990 Friedrichshafen

3) Dr.-Ing., Ingenieurgemeinschaft Prof. Dr.-Ing. R. Gnadler,
Kaiserallee 111, D-7500 Karlsruhe

relative to one another under the combined action of inertia forces, external forces, internal forces (caused by springs, dampers and actuators) and constraint forces (caused by rigid-body contacts in joints of the system). MESA VERDE simulates the dynamics of such systems. Upon specification of a standard set of input data it generates in symbolical form

- a minimal set of nonlinear differential equations of motion,
- expressions for kinematical quantities (positions, velocities and accelerations of user-specified points, angular orientations, angular velocities etc) and
- expressions for constraint forces and torques in joints.

The generated expressions are stored as ready-to-compile subroutines for use in subsequent numerical programs. Two basic tasks for numerical programs are

- integration of equations of motion in cases when all forces are prescribed,
- calculation of required motor forces and torques in cases when the motion of a system (of a robot arm, for example) is prescribed.

MESA VERDE is written in PASCAL (ANSI 1983 standard). It generates symbolical expressions either in PASCAL or in FORTRAN 77 code as is desired by the program user. MESA VERDE is easily adaptable to computers having a PASCAL compiler. Installations exist for the following machines: Siemens 78xx (BS3000), IBM 30xx (MVS), HP 835 (UNIX), HP 1000 (RTE-A), VAX (MVS), APOLLO (AEGIS), SUN 3/xx (UNIX), CELERITY C1260 (UNIX) and PC AT (MS-DOS).

In chapter 2 of this paper general characteristics of multibody systems are described. Chapter 3 explains input data of multibody systems for MESA VERDE. In chapter 4 the theory underlying MESA VERDE is outlined. Chapter 5 is devoted to illustrative examples.

2. General Characteristics of Multibody Systems

For many multibody systems it is not the motion relative to

inertial space but relative to some moving reference frame that is of interest. Typical examples are multibody systems mounted on a moving vehicle and multibody spacecraft moving relative to an orbiting reference frame. In MESA VERDE the moving reference frame is referred to as body 0. Input data for body 0 are functions of time which describe its three-dimensional motion.

Multibody systems are composed of only three types of elements, namely of bodies, of joints and of force elements.

2.1. Bodies: The individual bodies of a multibody system are assumed to be rigid.

2.2. Joints: Any type of technical joint between two bodies can be treated by MESA VERDE. Joints have a certain number smaller than six of degrees of freedom of motion of one body relative to its neighboring body. For each joint as many variables must be chosen by the program user as there are degrees of freedom in the joint. The variables describe the position and the angular orientation of one body relative to the neighboring body. Example 1: The cylindrical joint in Fig.1 has two degrees of freedom. Suitable variables are the displacement z and the rotation angle ϕ . Example 2: In Fig.2 the bodies 1 and 2 are connected by a crank-and-slider mechanism with bars 3 and 4. Body 2 slides along bar 4. If the bars have negligible mass then the whole mechanism represents a single joint between the bodies 1 and 2. Suitable joint variables are the displacement x and the crank angle ϕ .

If all bodies of a multibody system and also the reference body 0 are interconnected by joints then the variables of all joints together suffice to specify the position of the entire system. If there are not sufficiently many joints for total connection as is the case in Fig.3 then additional variables of relative position of bodies must be defined. In the figure, for example, six variables for the position and angular orientation of body 1 relative to body 0 might be defined. The connection thus established between the bodies 0 and 1 will be called a joint. With this generalized notion of joint all variables used in MESA VERDE are joint variables.

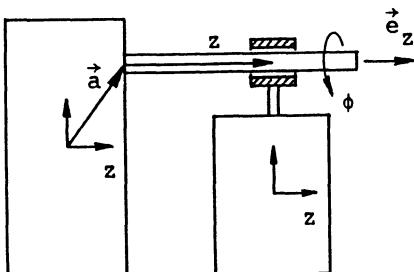


Fig.1: Cylindrical joint with joint variables z and ϕ . For other quantities see Chap.4.2

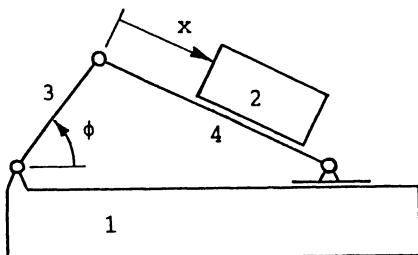


Fig.2: Two massless bars 3 and 4 represent a joint between bodies 1 and 2 with joint variables ϕ and x

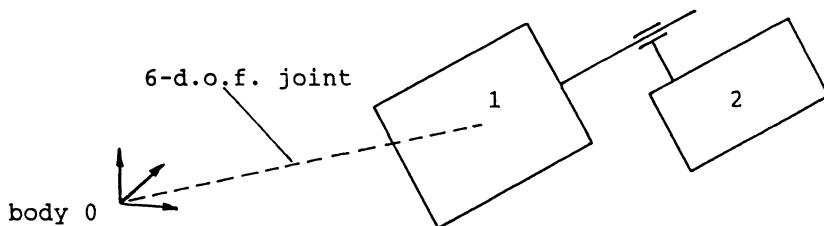


Fig.3: The system of bodies 1 and 2 is in free flight in inertial space (body 0). Between bodies 0 and 1 a joint with 6 joint variables is defined

If a system of bodies and joints with joint variables q_1, \dots, q_m has tree structure then the system has m degrees of freedom and MESA VERDE will generate m equations of motion. For various reasons there may exist constraint equations of the general form $f(q_1, \dots, q_m, t) = 0$ which reduce the number of degrees of freedom. Two important causes of such constraints are kinematical loops and the prescription that one or more of the variables q_1, \dots, q_m be specified functions of time. If constraint equations are given as part of the input data MESA VERDE will generate a minimal set of equations of motion for the reduced number of degrees of freedom. For closed kinematical loops the functions $f(q_1, \dots, q_m, t) = 0$ are not even required.

MESA VERDE can also treat nonholonomic constraints due to rolling conditions or to other causes.

2.3. Force Elements: Force elements are springs or dampers or actuators which connect bodies of a system. They may be linear or nonlinear. Forces exerted by actuators may be given functions of time or of other variables. They may also be determined by differential equations which govern a control system built into the multibody system. MESA VERDE allows any kind of force law.

For other capabilities of MESA VERDE see the appendix to this handbook.

3. Input Data for MESA VERDE

Since MESA VERDE generates symbolical expressions for subsequent numerical calculations in separate programs one must distinguish input data for MESA VERDE from input data for these numerical programs. The MESA VERDE input data to be supplied by the program user is very simple. As an example we consider the robot in Fig.4. It is composed of bodies 1, 2 and 3 (tower, arm and hand) and of joints 1, 2 and 3 (two cylindrical and one revolute joint) with five joint variables q_1, \dots, q_5 . Three motor torques and two motor forces act along the joint axes. All bodies are subject to weight. Body 0 is inertial space. The complete list of input data is shown on the next page. Underscore designates key words. Text following the symbol # is comment written by the program user.

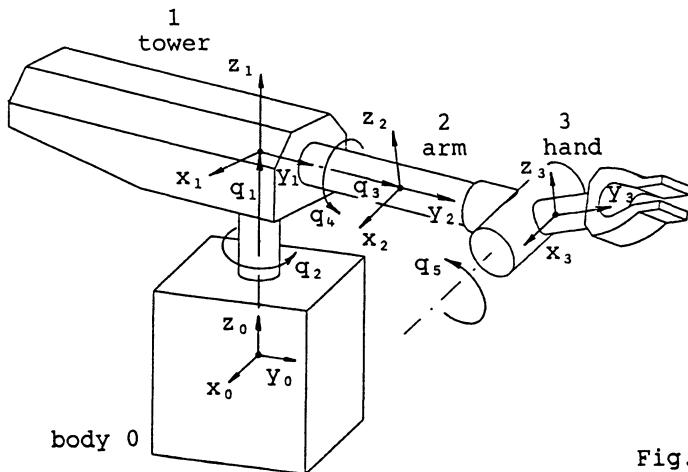


Fig.4: Robot

```

# input for bodies and for external forces

body 0 alias inertial
          motion zero

body 1 alias tower
          force 1 (z) frame inertial # weight

body 2 alias arm
          force 1 (z) frame inertial # weight

body 3 alias hand
          force 1 (z) frame inertial # weight
          marker 1

# input for joints

joint 1 type transzrotz # cylindrical joint; z-axis
          from inertial to tower
          torque 1 (z) # motor torque about joint axis

joint 2 type transyrotz # cylindrical joint; y-axis
          from tower to arm
          torque 1 (y) # motor torque about joint axis

joint 3 type rotx # revolute joint; x-axis
          from arm to hand
          torque 1 (x) # motor torque about joint axis

# input for force elements

force element 1 from inertial to tower # motor forces for
force element 2 from base to arm # translations in joints

# libraries for joint kinematics and for motions of body 0

include (joint.lib.)
include (motion.lib.)

# end of input data

```

The words "inertial", "tower", "arm" and "hand" as well as all comments are the program user's own words. The underscored key words have the effect that MESA VERDE automatically defines symbols for all system parameters, for all joint variables and for the first time derivatives of all joint variables. In what follows this is explained in more detail. Each entry body results in the definition of symbols for mass, moments and products of inertia (products of inertia are not defined if an additional key word indicates that principal axes of inertia are used). The entry "force 1 (z) frame inertial" in combination with body 3 results in the definition of one symbol representing the z-component in inertial space of an external force labeled

1 on body 3. The entry marker 1 in combination with body 3 results in the definition of symbols for the three cartesian coordinates of a marker point labeled 1 in the body 3 frame of reference. A marker is a point fixed on the body for which MESA VERDE generates symbolical expressions for position, velocity and acceleration relative to body 0. Type rotx together with the library (joint.lib.) results in the definition of symbols for six kinematical quantities for a revolute joint whose axis is parallel to the x-axes of the two coupled bodies. What these six quantities are will be explained in Chap.4.2. The input for all three joints together also results in the definition of the symbols $q(1), \dots, q(5)$ for the joint variables and of the symbols $qdot(1), \dots, qdot(5)$ for their first time derivatives. The entry "from arm to hand" associated with joint 3 expresses the user's own decision that $q(5)$ is the angle of rotation of the hand relative to the arm and not vice versa ("from arm to hand" describes the arrow shown in Fig.6). The entry "torque 1 (x)" results in the definition of one symbol for the internal torque about the joint axis x. The torque is applied with opposite signs to the arm and to the hand (with positive sign to the from body). Analogously, the entry "force element 1 from tower to arm" results in the definition of a symbol for the scalar magnitude of an internal force labeled 1 which is applied with opposite signs to the tower and to the arm (with positive sign to the from body). This entry also results in the definition of symbols for the constant cartesian coordinates of the points of application of these forces on the two bodies. The direction of the force need not be specified. It is given by the vector connecting the two points of application. This vector together with its length and with the time derivative of its length are automatically generated by MESA VERDE.

These comments on the input data for MESA VERDE also explain what the input data for subsequent numerical programs will be. Such input data is necessary for all symbols defined by MESA VERDE. Many symbols represent constant system parameters. Many others are components of vectors in specified frames of reference (specified by the body name following the key word frame in the input above). Many symbols represent functions and must

be defined as such. As an example suppose that the "force element 2 from tower to arm" is a linear spring with stiffness k and with unstretched length ℓ_0 . Then the scalar magnitude of the force would be the function $k(\ell - \ell_0)$ of the distance ℓ between the attachment points of the spring. This length is part of the output data of MESA VERDE.

4. The Theory Underlying MESA VERDE

4.1. The Principle of Virtual Power: For a multibody system of rigid bodies $i=1,\dots,n$ the principle of virtual power reads

$$\sum_{i=1}^n [\delta \dot{\vec{r}}_i \cdot (\underline{m}_i \ddot{\vec{r}}_i - \vec{F}_i) + \delta \dot{\vec{\omega}}_i \cdot (\underline{J}_i \ddot{\vec{\omega}}_i - \vec{M}_i^*)] = 0 \quad \text{with} \quad \vec{M}_i^* = \vec{M}_i - \vec{\omega}_i \times \underline{J}_i \vec{\omega}_i \quad (1)$$

(\underline{m}_i mass, \underline{J}_i inertia tensor, $\ddot{\vec{r}}_i$ absolute acceleration of center of mass, $\dot{\vec{\omega}}_i$ absolute angular acceleration, $\delta \dot{\vec{r}}_i$ virtual change of absolute velocity of center of mass, $\delta \dot{\vec{\omega}}_i$ virtual change of absolute angular velocity, \vec{F}_i resultant force and \vec{M}_i resultant torque about the center of mass; all quantities for body i). It should be noted that constraint forces and torques in joints do not appear in (1) since they are normal to $\delta \dot{\vec{r}}_i$ and to $\delta \dot{\vec{\omega}}_i$, respectively. (1) is given the matrix form

$$\delta \dot{\underline{r}}^T \cdot (\underline{m} \ddot{\underline{r}} - \underline{F}) + \delta \dot{\underline{\omega}}^T \cdot (\underline{J} \ddot{\underline{\omega}} - \underline{M}^*) = 0 \quad (2)$$

where \underline{m} and \underline{J} are ($n \times n$) diagonal matrices of masses and of inertia tensors, respectively; $\delta \dot{\underline{r}}$, $\ddot{\underline{r}}$, \underline{F} , $\delta \dot{\underline{\omega}}$, $\ddot{\underline{\omega}}$ and \underline{M}^* are column matrices of n vectors each and the exponent T indicates transposition. For more details on these as well as on the following equations the reader is referred to Wittenburg [1,2].

According to Chap.2.2 the position of a multibody system is described in terms of joint variables. Let these be arranged in the column matrix \underline{q} . The quantities $\dot{\underline{r}}$, $\ddot{\underline{r}}$, $\dot{\underline{\omega}}$ and $\ddot{\underline{\omega}}$ of (2) are linear forms

$$\dot{\underline{r}} = \underline{a}_1 \dot{\underline{q}} + \underline{f}_1(t), \quad \dot{\underline{\omega}} = \underline{a}_2 \dot{\underline{q}} + \underline{f}_2(t) \quad (3a)$$

$$\ddot{\underline{r}} = \underline{a}_1 \ddot{\underline{q}} + \underline{b}_1, \quad \ddot{\underline{\omega}} = \underline{a}_2 \ddot{\underline{q}} + \underline{b}_2 \quad (3b)$$

with as yet unknown coefficient matrices \underline{a}_1 , \underline{b}_1 , \underline{f}_1 , \underline{a}_2 , \underline{b}_2 and

$\ddot{\underline{q}}$. These matrices will be determined in Chap.4.3. Substitution of (3) into (2) yields

$$\delta \dot{\underline{q}}^T [\underbrace{(\dot{\underline{a}}_1^T \cdot \underline{m} \dot{\underline{a}}_1 + \dot{\underline{a}}_2^T \cdot \underline{J} \dot{\underline{a}}_2)}_{\underline{A}}] \ddot{\underline{q}} + \underbrace{[\dot{\underline{a}}_1^T \cdot (\underline{m} \dot{\underline{b}}_1 - \dot{\underline{F}}) + \dot{\underline{a}}_2^T \cdot (\underline{J} \dot{\underline{b}}_2 - \dot{\underline{M}}^*)]}_{-\underline{B}} = 0. \quad (4)$$

If the joint variables are not subject to constraint equations then follows

$$\underline{A} \dot{\underline{q}} = \underline{B}. \quad (5)$$

This is the desired minimal set of differential equations of motion. If there are constraint equations then (5) does not follow from (4). As was said in Chap.2.2 constraint equations reducing the number of degrees of freedom have the general form

$$f_i(q_1, \dots, q_m, t) = 0 \quad (i=1, \dots, v) \quad (6)$$

where m is the total number of joint variables and v is the number of constraint equations. An example: In the mechanism in Fig.5 the variables q_1 , q_2 and q_3 are subject to the independent constraint equations

$$f_1(q_1, q_2, q_3) = l_1 \cos q_1 - l_2 \cos(q_1 - q_2) - l_3 \cos q_3 + a = 0 \quad (7a)$$

$$f_2(q_1, q_2, q_3) = l_1 \sin q_1 - l_2 \sin(q_1 - q_2) + l_3 \sin q_3 - b = 0. \quad (7b)$$

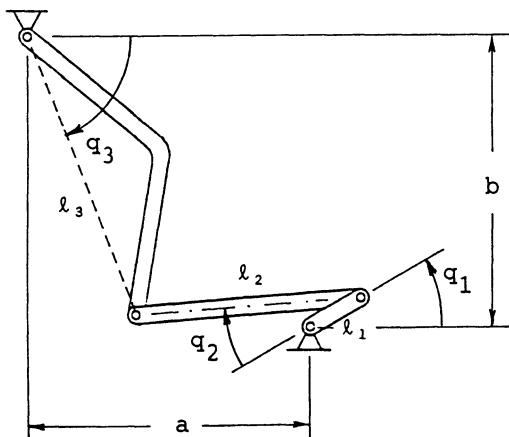


Fig.5: Mechanism with one degree of freedom. This is a section of the system of Fig.8

The second total time derivative of (6) represents v linear equations for $\ddot{q}_1, \dots, \ddot{q}_m$ with coefficients which depend on q_1, \dots, q_m and on t . Thus, v elements out of \ddot{q} can be expressed in terms of the remaining $m-v$ independent elements which form a column matrix \ddot{q}^* . The v expressions are combined with the identity relationship $\ddot{q}^* = \ddot{q}^*$ in the matrix equation

$$\ddot{q} = \underline{D}\ddot{q}^* + \underline{H} \quad (8a)$$

which implies

$$\dot{q} = \underline{D}\dot{q}^*. \quad (8b)$$

\underline{D} is an $[m \times (m-v)]$ matrix and \underline{H} is an $(m \times 1)$ column matrix. Substitution of (8) into (4) results in

$$\delta \dot{q}^* \underline{D}^T [\underline{A}(\underline{D}\ddot{q}^* + \underline{H}) - \underline{B}] = 0 \quad (9)$$

whence follows

$$\underline{A}^* \ddot{q}^* = \underline{B}^* \quad \text{with} \quad \underline{A}^* = \underline{D}^T \underline{A} \underline{D}, \quad \underline{B}^* = \underline{D}^T (\underline{B} - \underline{A} \underline{H}). \quad (10)$$

This is the desired minimal set of differential equations for the $m-v$ independent joint variables q^* . The matrix \underline{A}^* depends on all q_1, \dots, q_m and \underline{B}^* also on $\dot{q}_1, \dots, \dot{q}_m$. The dependent elements among these are substituted by means of (6) and (8b).

Constraint equations caused by closed loops are, in general, much more complicated than in the example (7). It is therefore important that MESA VERDE can generate the matrices \underline{A}^* and \underline{B}^* without the constraint equations (6). Instead, only the matrices $\underline{\dot{a}}_1, \underline{\dot{b}}_1, \underline{\dot{a}}_2$ and $\underline{\dot{b}}_2$ are required which are necessary for (5) anyway. They are needed only for systems without closed loops. These matrices are the subject of Chaps. 4.2 and 4.3.

4.2. Kinematics of Individual Joints: Fig. 6 depicts two bodies coupled by joint j . The kinematics description of a joint consists of six quantities. For many types of joints they are provided automatically by the library (joint.lib.) in the input data file of MESA VERDE (see Chap. 3). The definition of the six quantities requires the following decisions to be made first:

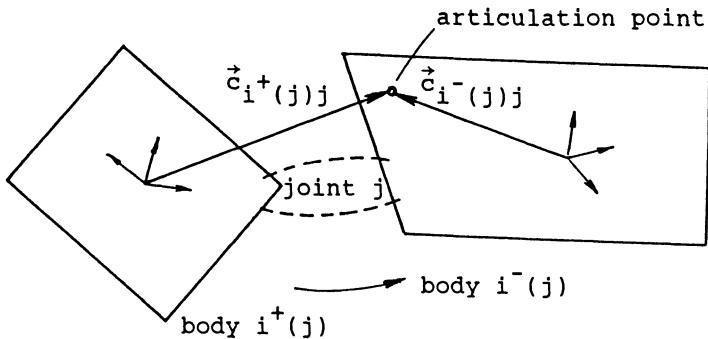


Fig.6: Two bodies coupled by joint j

(i) choose as many joint variables as there are degrees of freedom in the joint; (ii) choose on each body at the center of mass a body-fixed reference frame (these frames are referred to by the key word frame in the input data); (iii) choose one body as reference body and the other as body moving relative to the reference body; in the figure the choice is indicated by the arrow which points from the reference body to the other body (hence the key words from to in the input data); in what follows the index of the from body will be referred to as $i^+(j)$ and the index of the to body as $i^-(j)$; (iv) choose a point fixed on body $i^-(j)$ as so-called articulation point for joint j ; its location on body $i^-(j)$ is defined by the vector $\vec{c}_{i^-(j)j}$.

Three of the six kinematical quantities are the location vector $\vec{c}_{i^+(j)j}$, the velocity \vec{v}_j and the acceleration \vec{a}_j of the articulation point relative to the reference frame fixed on body $i^+(j)$. The other three are the (3×3) direction cosine matrix G_j relating the two body-fixed reference frames, the angular velocity $\vec{\omega}_j$ and the angular acceleration $\vec{\epsilon}_j$ of body $i^-(j)$ relative to body $i^+(j)$.

The decisions to be made by the program user should be such that these six quantities are the simplest possible functions of joint variables. An example: For the cylindrical joint in Fig.1 the body on the left is chosen as body $i^+(j)$, the point at the tip of the z-arrow is chosen as articulation point and the two reference frames are chosen such that the joint axis is

parallel to the z-axis of each of them. Then the six quantities are

$$\vec{c}_{i+}(j) = \vec{a} + z\vec{e}_z, \quad \vec{v}_j = \dot{z}\vec{e}_z, \quad \vec{a}_j = \ddot{z}\vec{e}_z, \quad \underline{G}_j = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (11)$$

$$\vec{\Omega}_j = \dot{\phi}\vec{e}_z, \quad \vec{\epsilon}_j = \ddot{\phi}\vec{e}_z,$$

For any type of joint and for any choice made by the program user \vec{v}_j , \vec{a}_j , $\vec{\Omega}_j$ and $\vec{\epsilon}_j$ have the forms

$$\vec{v}_j = \sum_{\ell} \vec{k}_{j\ell} \dot{q}_{\ell}, \quad \vec{\Omega}_j = \sum_{\ell} \vec{p}_{j\ell} \dot{q}_{\ell}, \quad (12a)$$

$$\vec{a}_j = \sum_{\ell} \vec{k}_{j\ell} \ddot{q}_{\ell} + \vec{s}_j, \quad \vec{\epsilon}_j = \sum_{\ell} \vec{p}_{j\ell} \ddot{q}_{\ell} + \vec{w}_j. \quad (12b)$$

In these expressions the q_{ℓ} are joint variables, $\vec{k}_{j\ell}$ and $\vec{p}_{j\ell}$ are vectors which do or do not depend on the joint variables and \vec{s}_j and \vec{w}_j depend on the \dot{q}_{ℓ} if they are not zero. The matrix \underline{G}_j and the vectors $\vec{c}_{i+}(j)$, $\vec{c}_{i-}(j)$, $\vec{k}_{j\ell}$, $\vec{p}_{j\ell}$, \vec{s}_j and \vec{w}_j are part of the input data. These quantities are automatically produced by the library (joint.lib.) once key words defining the joint are provided. An example: The key words joint 1 type transzrotz produce the vectors and the matrix \underline{G}_j shown in (11). The library can be expanded by the program user to include additional types of joints.

4.3. Kinematics of the System as a Whole: The absolute angular velocity $\vec{\omega}_i$ of an arbitrary body $i=1,\dots,n$ is the sum of the absolute angular velocity $\vec{\omega}_0(t)$ of body 0 and of the relative angular velocities $+ \vec{\Omega}_j$ or $- \vec{\Omega}_j$ of all joints located between the bodies 0 and i. This is expressed in the form

$$\vec{\omega}_i = \vec{\omega}_0(t) - \sum_{j=1}^n T_{ji} \vec{\Omega}_j. \quad (13)$$

The scalar T_{ji} ($j,i=1,\dots,n$) is +1 or -1 or 0. It is +1 (-1) if joint j is located between bodies 0 and i and if the arrow defined in Fig.6 is pointing toward body 0 (toward body i). Otherwise $T_{ji}=0$. With (13) the column matrices $\underline{\omega}$ and $\underline{\Omega}$ of all n angular velocities are related by the equation

$$\underline{\omega} = \vec{\omega}_0(t) \underline{1} - \underline{T} \underline{\Omega} \quad (14)$$

with the matrix \underline{T} of the elements T_{ji} and with a column matrix $\underline{1}$ of n unit elements. Because of (12) $\underline{\dot{q}}$ has the form $\underline{\dot{q}} = \underline{p}^T \underline{\dot{q}}$ where \underline{p} is a matrix containing only vectors \vec{p}_{jl} and zeros. Hence

$$\underline{\dot{\omega}} = \vec{\omega}_o(t) \underline{1} - \underline{T}^T \underline{p}^T \underline{\dot{q}}. \quad (15)$$

Comparison with (3a) yields two of the six matrices:

$$\underline{\dot{a}}_2 = -(\underline{p}t)^T, \quad \underline{\dot{x}}_2 = \vec{\omega}_o(t) \underline{1}. \quad (16)$$

Starting point for $\underline{\dot{a}}_1$ and $\underline{\dot{x}}_1$ is an expression for the velocity of the body i center of mass. In analogy to (13) we have

$$\dot{\underline{r}}_i = \dot{\underline{r}}_o(t) - \sum_{j=1}^n T_{ji} (\dot{\underline{c}}_i^+ (j) j - \dot{\underline{c}}_i^- (j) j). \quad (17)$$

With

$$\dot{\underline{c}}_i^+ (j) j = \vec{\omega}_i^+ (j) \times \vec{c}_i^+ (j) j + \vec{v}_j \quad \text{and} \quad \dot{\underline{c}}_i^- (j) j = \vec{\omega}_i^- (j) \times \vec{c}_i^- (j) j \quad (18)$$

follows in analogy to (14)

$$\dot{\underline{r}} = \dot{\underline{r}}_o(t) \underline{1} - \underline{T}^T \underline{v} - \underline{T}^T \underline{C}^T \times \underline{\dot{\omega}} \quad (19)$$

with a matrix \underline{C} whose elements are the vectors $\vec{c}_{ij} = +\vec{c}_{ij}$ for $i = i^+(j)$, $\vec{c}_{ij} = -\vec{c}_{ij}$ for $i = i^-(j)$ and $\vec{c}_{ij} = \vec{0}$ else. With $\underline{\dot{\omega}}$ from (15) and with $\underline{\dot{v}} = \underline{k}^T \underline{\dot{q}}$ from (12a) one finally gets

$$\dot{\underline{r}} = \dot{\underline{r}}_o(t) \underline{1} - \underline{T}^T \underline{C}^T \times \vec{\omega}_o(t) \underline{1} - \underline{T}^T \underline{k}^T \underline{\dot{q}} + \underline{T}^T \underline{C}^T \times \underline{T}^T \underline{p}^T \underline{\dot{q}}. \quad (20)$$

and by comparison with (3b)

$$\underline{\dot{a}}_1 = -[\underline{T} \underline{k} + \underline{p} \underline{T} \times \underline{C} \underline{T}]^T, \quad \underline{\dot{x}}_1 = \dot{\underline{r}}_o(t) \underline{1} + \vec{\omega}_o(t) \times (\underline{C} \underline{T})^T \underline{1}. \quad (21)$$

The matrices $\underline{\dot{b}}_1$ and $\underline{\dot{b}}_2$ are found by differentiating (15) and (20) once more with respect to time. For details see [1,2].

With the expressions for the matrices $\underline{\dot{a}}_1$, $\underline{\dot{b}}_1$, $\underline{\dot{x}}_1$, $\underline{\dot{a}}_2$, $\underline{\dot{b}}_2$ and $\underline{\dot{x}}_2$ the matrices \underline{A} and \underline{B} in (5) and \underline{A}^* and \underline{B}^* in (10) are expressed in terms of joint variables and of known functions of time.

MESA VERDE carries out the matrix multiplications. In this process the sparseness of the matrices as well as recurrence relationships indicated by the factors \underline{T} and \underline{T}^T are used. The various vectors in the matrices are transformed into a common re-

ference frame by means of the direction cosine matrices \underline{G}_j defined for the joints. The resulting symbolical expressions are kept short by expressing them in terms of a hierarchy of substitution variables which are generated in the process.

4.4. Constraint Forces and Torques in Joints: MESA VERDE generates expressions for constraint forces and torques in joints for systems with and without closed loops. In this paper only system without closed loops are discussed. All bodies $i=1,\dots,n$ of a multibody system are isolated by cutting all joints. The constraint forces in joint j are distributed over the contact area of the two bodies $i^+(j)$ and $i^-(j)$. Let the distributed forces be reduced to a single constraint force \vec{F}_j^C at the articulation point and to a single constraint torque \vec{M}_j^C . Let \vec{F}_j^C and \vec{M}_j^C be applied to body $i^+(j)$ and $-\vec{F}_j^C$ and $-\vec{M}_j^C$ to body $i^-(j)$. Define $S_{ij} = +1$ for $i=i^+(j)$, $S_{ij} = -1$ for $i=i^-(j)$ and $S_{ij} = 0$ else ($i,j=1,\dots,n$). With these quantities Newton's and Euler's laws for body i read

$$\ddot{\underline{m}}_i \ddot{\underline{r}}_i = \vec{F}_i + \sum_{j=1}^n S_{ij} \vec{F}_j^C, \quad \dot{J}_i \dot{\underline{\omega}}_i = \vec{M}_i^* + \sum_{j=1}^n S_{ij} (\vec{C}_{ij} \times \vec{F}_j^C + \vec{M}_j^C) \quad (i=1,\dots,n) \quad (22)$$

where \underline{m}_i , $\ddot{\underline{r}}_i$, \vec{F}_i , J_i , $\dot{\underline{\omega}}_i$ and \vec{M}_i^* are the same quantities as in (1). The sets of n equations each are written in the matrix forms

$$\ddot{\underline{m}}\ddot{\underline{r}} = \vec{F} + \underline{S}\vec{F}^C, \quad \dot{J}\dot{\underline{\omega}} = \vec{M}^* + \underline{C} \times \vec{F}^C + \underline{M}^C. \quad (23)$$

\underline{C} is the same matrix as in (19) and \vec{F}^C and \vec{M}^C are the column matrices of all constraint forces and torques, respectively. An important theorem states that \underline{T} is the inverse of \underline{S} (see [1]). This allows to solve (23) for \vec{F}^C and \vec{M}^C explicitly:

$$\vec{F}^C = \underline{T}(\ddot{\underline{m}}\ddot{\underline{r}} - \vec{F}), \quad \vec{M}^C = \underline{T}(\dot{J}\dot{\underline{\omega}} - \vec{M}^* - \underline{C} \times \vec{F}^C). \quad (24)$$

$\ddot{\underline{r}}$ and $\dot{\underline{\omega}}$ are known in terms of $\ddot{\underline{q}}$ from (3). Thus the numerical integration of (5) yields also \vec{F}^C and \vec{M}^C .

5. Illustrative Examples

All authors of this handbook were asked to simulate the robot shown in Fig.4 and the planar seven-body mechanism shown in Fig.8.

5.1. Robot: Fig.4 shows the inertial reference frame x_o, y_o, z_o and the reference frames x_i, y_i, z_i attached to the centers of mass of bodies $i=1, 2, 3$. Also shown are the joint variables q_1, \dots, q_5 . In the configuration $q_1=q_2=q_3=q_4=q_5=0$ the reference frames of bodies 0, 1 and 2 coincide and the one on body 3 is parallel to the other three. System parameters: The distances of the centers of mass of bodies 2 and 3 from the q_5 -axis are 0.5m and 0.05m, respectively. The body masses and the moments of inertia are

body	m [kg]	I _x	I _y	I _z	[kgm ²]
1	250	90	10	90	
2	150	13	0.75	13	
3	100	4	1	4.3	

Motor torques [Nm] and forces [N] associated with the variables q_1, \dots, q_5 are the following functions of time t [s]:

$$q_1: F_1 = \begin{cases} 4348 & 0 \leq t \leq 0.5 \\ 4905 & 0.5 < t \leq 1.5 \\ 3462 & 1.5 < t \leq 2.0 \end{cases} \text{ applied to body 1}$$

$$q_2: M_2 = \begin{cases} 673 \cdot t - 508 & 0 \leq t \leq 0.5 \\ 148 \cdot \exp[-5.5(t-0.5)] + 8 & 0.5 < t \leq 1.5 \\ 240 & 1.5 < t \leq 2.0 \end{cases} \text{ applied to body 1}$$

$$q_3: F_3 = \begin{cases} 36 \cdot t + 986 & 0 \leq t \leq 0.5 \\ -2 & 0.5 < t \leq 1.5 \\ -1019 & 1.5 < t \leq 2.0 \end{cases} \text{ applied to body 2}$$

$$q_4: M_4 \equiv 0$$

$$q_5: M_5 = \begin{cases} 63.5 & 0 \leq t \leq 0.5 \\ 49.05 & 0.5 < t \leq 1.5 \\ 34.6 & 1.5 < t \leq 2.0 \end{cases} \text{ applied to body 3}$$

Initial conditions are $q_{10}=2.25\text{m}$, $q_{20}=-0.5236\text{rad}$, $q_{30}=0.75\text{m}$, $q_{40}=q_{50}=0$ and $\dot{q}_{io}=0$ ($i=1, \dots, 5$).

The input data for MESA VERDE as well as for the subsequent numerical integration program was discussed in Chap.3. Of the results only $q_1(t)$ is shown (see Fig.7).

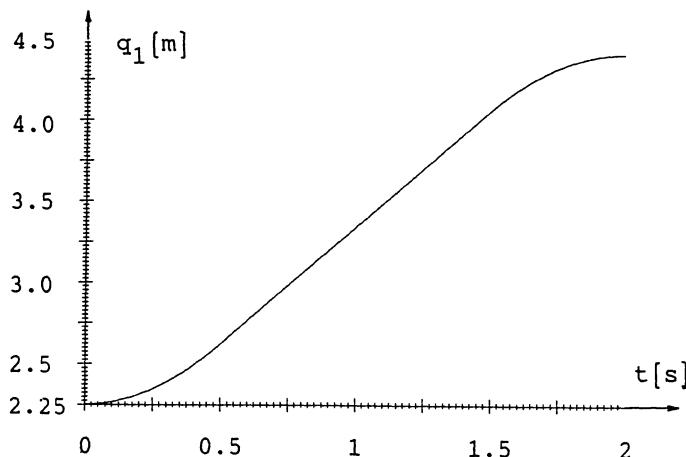


Fig.7

5.2. Planar Seven-Body Mechanism: The mechanism in Fig.8 has a single degree of freedom. As independent variable the crank angle q_1 is chosen. MESA VERDE generates a single differential

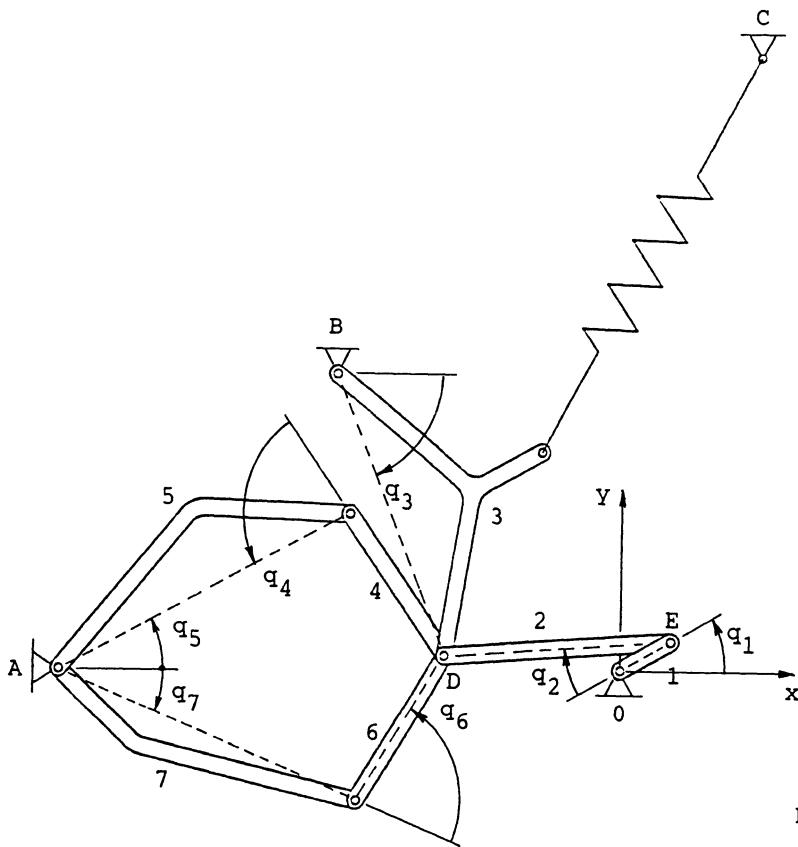


Fig.8

equation of motion in the form (10) which in this case reads $A^*\ddot{q}_1 = B^*$ with scalars A^* and B^* . (10) resulted from the equations of motion (5) for a system without closed kinematical loops and from constraint equations (6) which represent the loop closure conditions. There are many ways of producing from Fig.8 a system without closed kinematical loops. A simple way is to eliminate the support at point B and to disconnect bodies 4 and 6 at their right-hand side end points from the system and from one another. This produces two double pendulums (bodies 5 and 4 and bodies 7 and 6) and a triple pendulum (bodies 1, 2 and 3) which are connected to body 0 (represented by the supports A and 0). The MESA VERDE input data for this system is as simple as the one for the robot discussed in Chap.3. To give only one detail the input data for joint 3 between bodies 2 and 3 reads

joint 3 type rotz from bar2 to bar3 (25)

When the support at B is reconstructed the triple pendulum takes the form of Fig.5. For this mechanism the constraint equations (7a,b) were formulated. When body 4 is re-attached to the system bodies 5, 4 and 3 together represent the same type of mechanism as the one in Fig.5. Consequently there exist two similar constraint equations for q_5 , q_4 and q_3 . The same argument yields two more constraint equations of the same type for q_7 , q_6 and q_3 . For the six constraint equations the matrices D and H of (8a) must be formulated. This is simple. As to input data one must again distinguish the symbolical program MESA VERDE from subsequent numerical programs. In the MESA VERDE input data the dependency of a joint variable, say q_3 , on q_1 requires not more than to add the line "constraint (rz) to joint 1 (rz)" to the line (25). This results in the definition of symbols for the elements in the third row of the matrices D and H. The actual functions representing these elements are part of the input data of subsequent numerical programs.

The system parameters for the mechanism of Fig.8 including the driving torque and initial conditions can be found elsewhere in this book. The numerical results are illustrated by the diagrams for $q_1(t)$, $\dot{q}_1(t)$ and $\ddot{q}_1(t)$ in Fig.9.

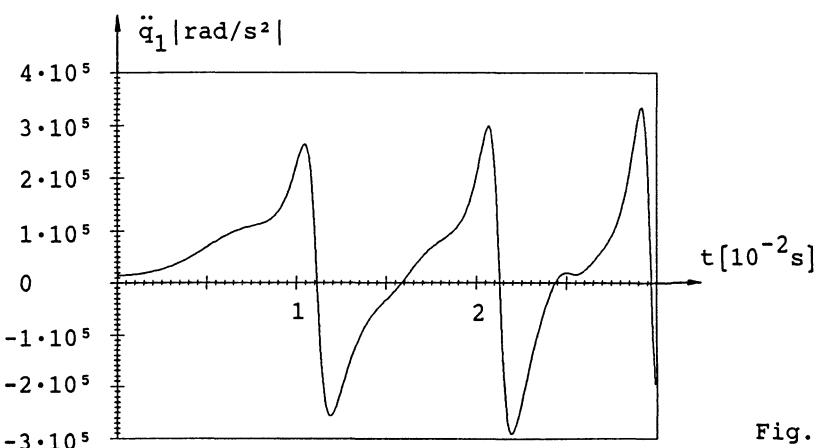
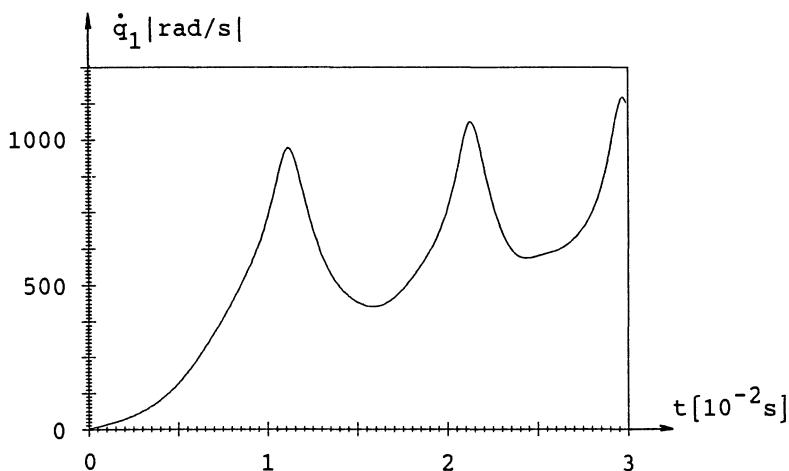
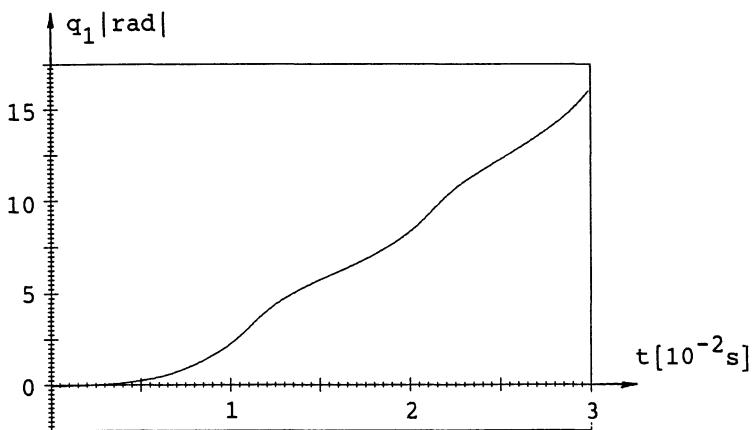


Fig. 9

In the mechanism with the prescribed link lengths the relationships between q_2, \dots, q_7 on the one hand and q_1 on the other are without much interest. More interesting from a kinematics as well as from a dynamics point of view is the combination of link length shown by the solid lines in Fig.10 ($l_2 = l_3 = \overline{AD} = l$, $l_1 = l/3$, $l_4 = l_5 = l_6 = l_7 = 5l/4$). Dashed lines show the same mechanism in two more positions. In one of them the bars 1, 2 and 3 are in one straight line. In this position inertia forces and external forces decide in which of two possible directions the point D moves when the crank angle is changed. Motions are possible which go through one complete cycle every two revolutions of the crank. Fig.11 shows simulation results obtained for $q_1(t)$ and $\dot{q}_1(t)$ for this kind of motion under the following conditions: Body centers of mass not along the drawn bar axes; spring as in Fig.8; weight on all bodies; motor torque = 0; initial velocity sufficiently high for a complete cycle.

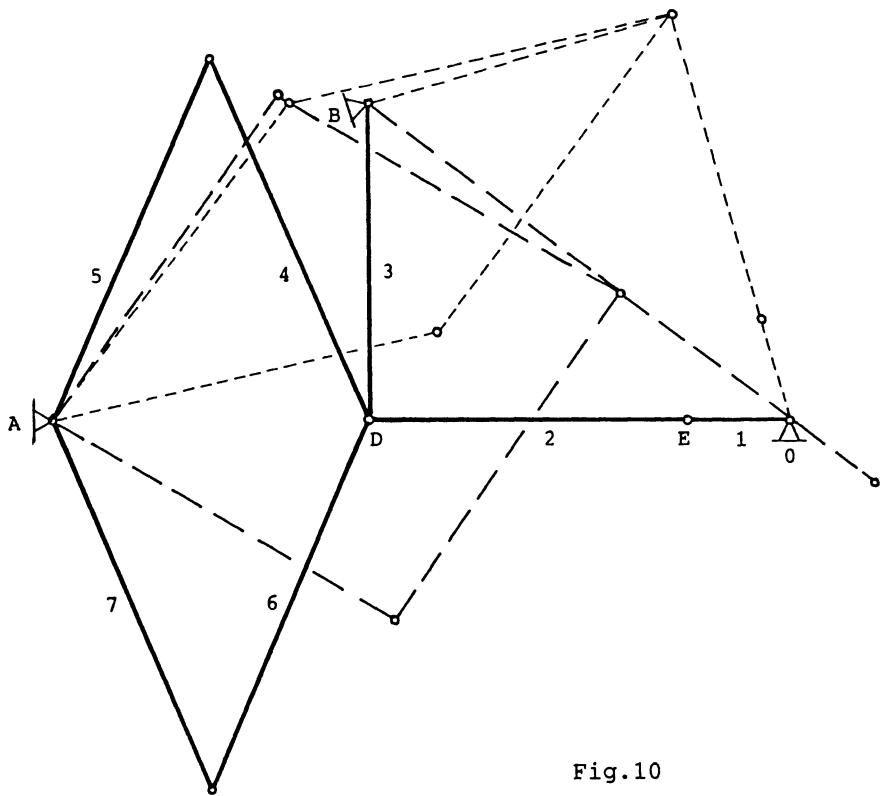


Fig.10

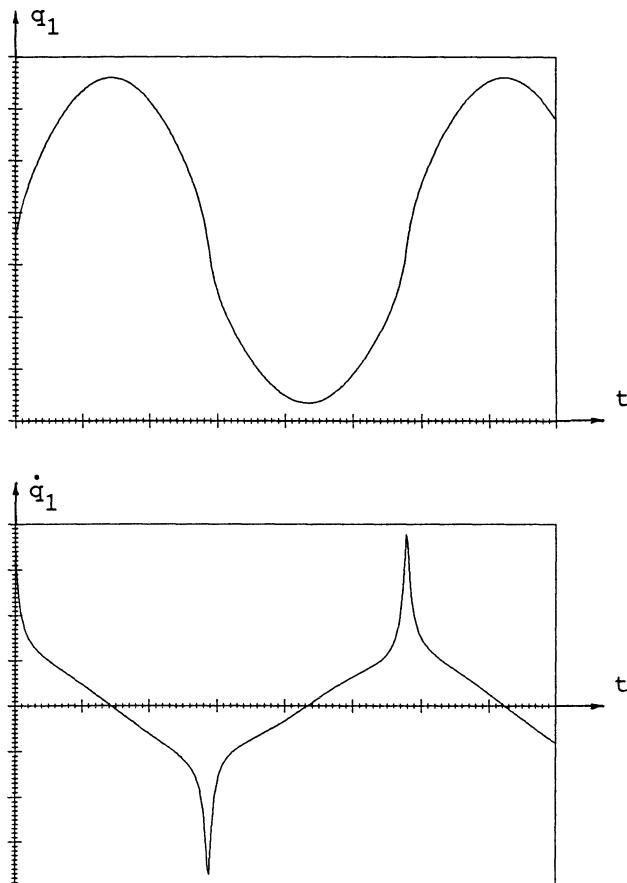


Fig. 11

Literature:

1. Wittenburg,J., The Dynamics of Systems of Rigid Bodies, Stuttgart: Teubner 1977, russ. and chin. transls.
2. Wittenburg,J., Analytical Methods in Mechanical System Dynamics, appeared in Haug,E.J.(Ed.), Computer Aided Analysis and Optimization of Mechanical System Dynamics, NATO ASI ser. Springer 1984
3. Schmidt,A., Wolz,U., Simulation komplexer dynamischer Systeme mit dem Programm paket MESA VERDE am Beispiel eines Fahrzeugsatzsystems, Automobil Industrie 31(1986),305-317
4. Schmidt,A., Methoden und Werkzeuge zur Erstellung von fahr-dynamischen Simulationsmodellen, Automobil Industrie 34 (1989),83-89

ADAMS - Multibody System Analysis Software

R. R. RYAN *

*Mechanical Dynamics, Inc.
Ann Arbor, Michigan 48105, USA*

Abstract

Multibody system analysis software forms an integral part of modern mechanical computer aided engineering (MCAE) practice in providing a means to simulate the operating performance of a product design prior to building a prototype. In this way, it reduces product development costs, allows evaluations of more alternative designs, and shortens the time it takes to bring a new product to the marketplace. This paper reviews the genesis, application, technical features, underlying analytical formulation, and future prospects of the most widely used multibody system analysis package, ADAMS.

I. Introduction

In the process of designing and analyzing complicated mechanical systems, a number of fundamental questions are often raised concerning the intended and actual behavior of the system designs being studied. When the systems involve parts and subassemblies that can articulate with respect to one another to allow large overall rotation and translation, such as the four systems shown in Figure 1, many of the questions raised by designers and engineers focus on possible motions of the systems. For instance, for the satellite system shown in the upper left portion of the figure, the effect that the astronaut/mechanic would have on satellite attitude motion during maintenance was of interest. For the jet aircraft ejection sequence depicted in the upper right, the engineering team developing the rocket motors attached to the ejection seat was interested in designing a rocket motor burn profile that would insure that the pilot and seat as well as the canopy would avoid hitting the tail of the aircraft during an emergency operation occurring in an environment involving unsteady aerodynamic loads. They needed a way of evaluating the effect of changes in burn profile on seat trajectories. In the case of the vehicle and driver included in the lower left, a suspension design engineer was interested in determining the response and settling time of the initial vehicle design subsequent to a driver-imposed lane change maneuver. Lastly, for the front-end loader illustrated in the lower right portion of the figure, an engineer needed to know the load capacity required of the hydraulic cylinders for the range of motion shown.

* Vice President, Product Development

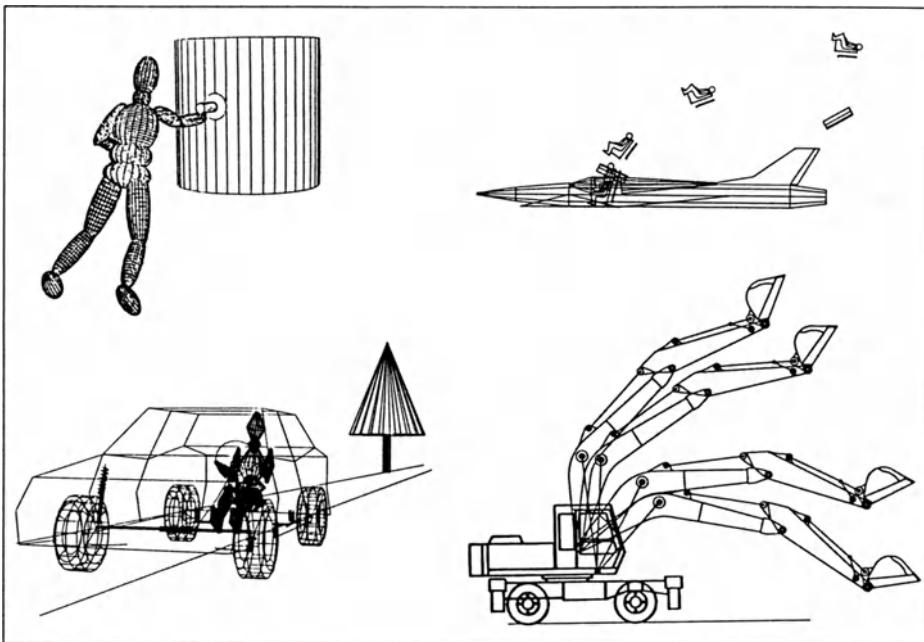


Fig. 1. Examples of Multibody System Models

These informational needs and concerns are typical in the design cycle of any product that is intended to be subjected to significant motion. Fortunately, present-day multibody system analysis tools, such as the one to be described in this paper, have made the process of answering these questions relatively straightforward. Such tools, which were unavailable thirty years ago, have revolutionized the way engineers perform kinematic, static, and dynamic analyses of mechanical systems.

For nearly two hundred years after the pioneering technical advancements of Newton and Lagrange, the process of analyzing systems of rigid bodies undergoing overall motions involving large-amplitude rotations remained an intimidating undertaking. First, one had to develop a mathematical model involving idealized components. Then, one was faced with the daunting task of manually formulating equations governing the behavior of the model. This typically involved a substantial amount of tedious algebraic manipulations, was prone to errors, and was virtually impossible to accomplish for any system with more than just a few degrees of freedom. Lastly, if an analyst was skilled enough to complete the first two tasks, the final obstacle of cleverly finding a closed-form solution of the equations using tabulated mathematical functions was difficult enough to restrict the entire process to only the most skilled academicians addressing the very simplest mechanical systems.

The advent of the digital computer brought forth a new dimension in engineering problem solving. The computer not only made it possible to routinely solve systems of mathematical equations, but also opened up the realm of automatically formulating equations of motion governing systems described by analysts and designers. This dramatic improvement in computational capabilities prompted the beginnings of a new technical field of study referred to as *multibody system analysis*. The new field focuses on more efficient and accurate techniques to harness the power of the digital computer to formulate and solve equations governing large overall motions of complex rigid and flexible mechanical systems. As in the field of structural finite element analysis, the product of focused research in the multibody system analysis field is often a software system embedding the new capabilities developed. The objective of the present paper is to provide an overview of one such software system, named ADAMS, which has evolved from many man years of research and development to become the world's most widely used software for large-displacement kinematic, static, and dynamic analysis.

The paper is organized as follows. The section that immediately follows this introduction defines the scope of the ADAMS Software System and describes its origin. Section three outlines the domain of commercial application of ADAMS and summarizes problem areas where ADAMS has been used extensively. In section four, the detailed program features of ADAMS are itemized and described briefly; while section five provides a short overview of the analytical and computational methods that underlie ADAMS. Development directions for ADAMS are reviewed in section six, followed by a synopsis of various test example problems used to demonstrate program capability and to provide results verification. Finally, the paper concludes with acknowledgements and technical references.

II. Background

ADAMS, which is an acronym for *Automatic Dynamic Analysis of Mechanical Systems*, is a software system consisting of a number of integrated programs that aid an analyst in performing three-dimensional kinematic, static, quasi-static, or dynamic analyses of mechanical systems. The mechanical systems may comprise any number of rigid or flexible bodies that are interconnected by joints allowing for arbitrarily large relative rotational and translational motions and that are subjected to any variety of internal or external forces or prescribed motions. There is no restriction as to topological interconnection of bodies. Thus, chain, tree, cluster, closed-loop, and multiple closed-loop configurations are treated in an identical fashion. The system identifies redundant constraints and eliminates them automatically.

The input to the program consists of part geometry and mass properties, reference frame definitions, body types, body compliance descriptions, topological and analytical constraints, force and motion actuator and sensor models, elastic restraints and connectors, control laws, and graphic entities. Typical output includes the time-dependent positions, velocities, accelerations, forces, and values of user-defined variables. This output can take the form of tabulated data, two-dimensional plots, still-frame system configurations, superimposed images of the system at various instants in time, and continuous graphic animations of system motion. More details of the features of ADAMS are included in section four.

The ADAMS software system encompasses ten separate, but integrated, programs as shown in Figure 2. The ADAMS program is the primary analysis engine providing the capabilities for general kinematic, static, and dynamic analyses.

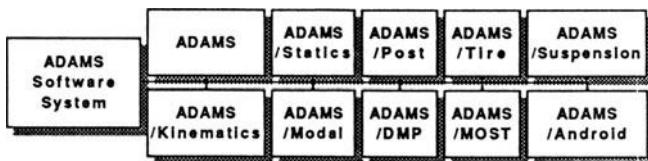


Fig. 2. ADAMS Software System Components

ADAMS/Kinematics and ADAMS/Statics are subsets of the ADAMS engine and, as such, are embedded in many turnkey MCAE software systems to provide integrated kinematic and static mechanism analysis capabilities within the framework of a CAE database. They are, to be precise, libraries of callable subroutines rather than stand-alone software packages.

ADAMS/Modal is a product that automatically linearizes the nonlinear ADAMS equations of motion about a nominal operating state defined by the user. The program provides the ability to either transfer this linearized state-space *plant* model to a control system design package for control law formulation or to solve the classical eigenvalue problem associated with the linearized equations. When the latter choice is made, ADAMS/Modal produces natural frequencies and complex mode shapes of the linearized system, which are particularly useful for correlating a system model with experimental modal analysis results. The program requires input in the form of an ADAMS model and provides output in the form of tabular data, static or animated mode shapes (see Fig. 3), open- and closed-loop pole/zero plots, and real and imaginary eigendata plots.

The ADAMS/Post software consists of a nearly device-independent graphic post-processor for

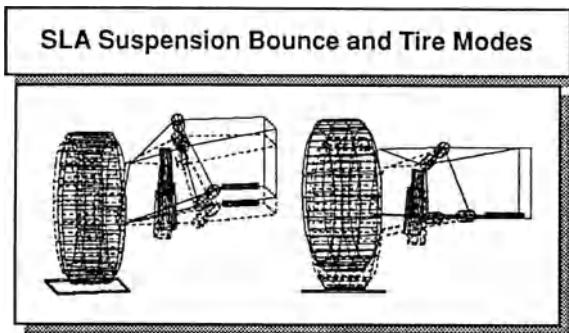


Fig. 3. ADAMS/Modal and ADAMS/Tire

plotting two-dimensional data and producing still-frame pictures of the system, as well as three device-specific continuous graphic animation programs for the Evans & Sutherland PS300 and 390, the HP9000/300 and 800, and the Silicon Graphics Iris workstations. In addition, a data conversion module is available that transforms ADAMS tabular output into the proper format to be read, manipulated, rearranged, and displayed (plotted) by popular IBM PC and Apple Mac II spreadsheet and graphics packages. All of the images appearing in Figure 1, as well as most of the other images and plots included in the remainder of this paper, are end-products of modules of ADAMS/Post.

ADAMS/DMP is a data modification program allowing for the formation and use of macros to automatically create and modify large, complicated data sets that may be used as input to ADAMS. The program provides for the systematic creation of symmetric and anti-symmetric assemblies based on user-supplied parameter specifications.

The ADAMS/Tire program provides a capability for modeling vehicle tire forces to be included in vehicle handling, ride quality, or durability analyses. The user enters road profile data and tire properties and applies traction and braking forces to the tire; the program then determines longitudinal reaction forces, lateral sliding forces, and all resulting torques to be applied to the attached vehicle (Fig. 3).

ADAMS/MOST (see top of Fig. 4) is a mechanism optimal synthesis tool that performs dimensional synthesis by varying an ADAMS model iteratively to minimize the deviation between a desired mechanism path and the existing mechanism path. The program accepts a mechanism as described in ADAMS, and directs the user to specify *tracer* points (indicated by \otimes) on the mechanism to allow the kinematic solver to sketch out the true path of the mechanism. The user then adds target points (marked by $+$) through which the tracer points are supposed to travel. Finally, the program solves an optimization problem to iteratively alter joint lo-

cations and link dimensions in order to define a new ADAMS mechanism model that accomplishes the design objective to the extent possible.

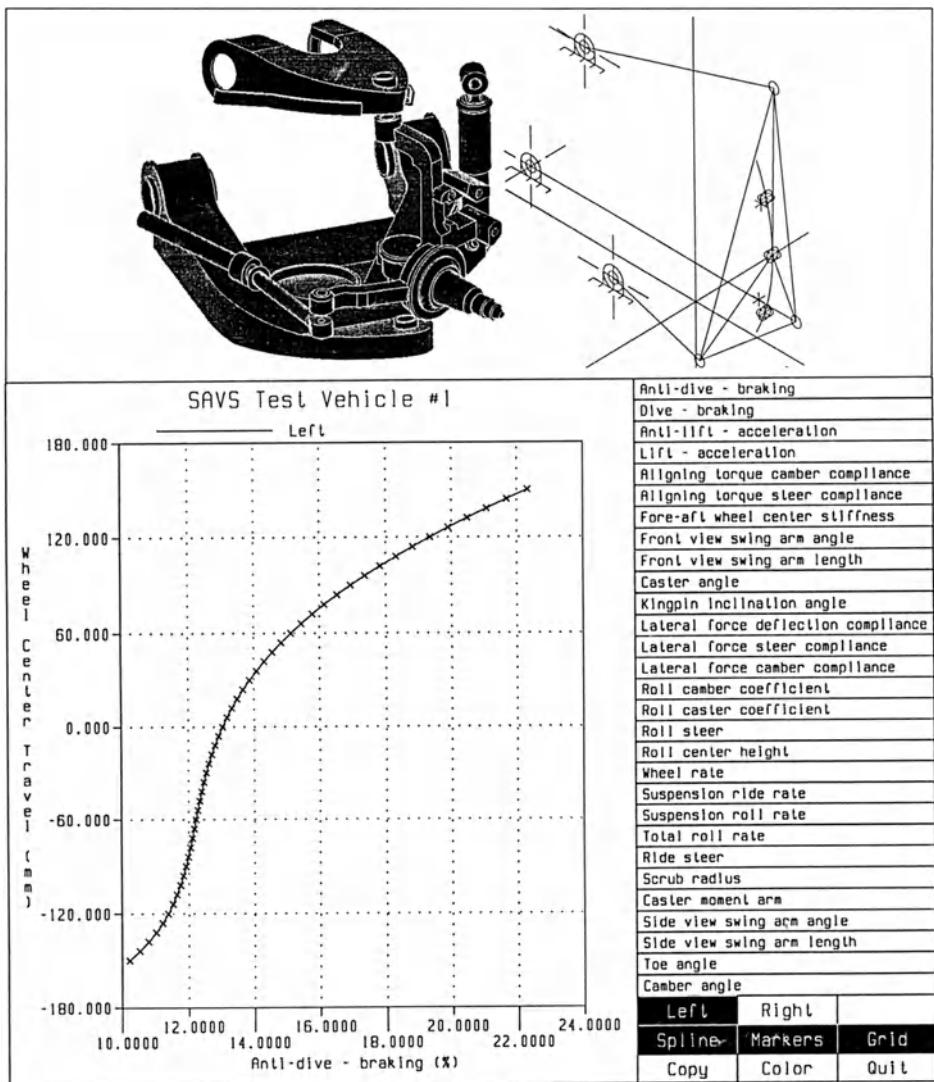


Fig. 4. ADAMS/MOST and ADAMS/Suspension

ADAMS/Android and ADAMS/Suspension are two prototype software programs, which are currently being converted into commercial form. ADAMS/Android provides a method for easily modeling a human body undergoing a variety of isolated motions or motions involving interaction with a mechanical system. Input to the program consists of age, sex, height and weight of the human as a percentile of the population, the desired initial configura-

tion, and the time-dependent motions to be applied to the system. The program provides all of the necessary geometry, mass, inertia, and connectivity data to ADAMS along with initial part locations and joint motion definitions so as to allow for full kinematic, static, and dynamic analyses of man-machine systems as illustrated in Figure 1. ADAMS/Suspension is a program that allows an analyst to very simply build a full ADAMS model of a standard or non-standard automotive or truck suspension by using input terminology and standard component elements that are familiar to suspension designers. The program then automatically invokes ADAMS to calculate twenty-nine different suspension characteristics (see lower portion of Fig. 4) as a function of vertical wheel travel as the suspension moves between the *jounce* (lowest) and *rebound* (highest) positions. Typical characteristics calculated include caster, camber, toe-in, scrub radius, ride steer, wheel rate, etc. These characteristics are provided to the user in tabular and plotted form.

Combined, these ten programs constitute the ADAMS software system, which is one of the oldest and most developed three-dimensional, type-variant, commercial multibody software systems available today. The first tools of this kind were conceived in 1964, starting with the dynamic formalisms of Hooker and Margulies^[1] and Roberson and Wittenburg^[2]. These developments were mainly targeted toward aerospace multibody systems having open-loop, tree-type topologies. At nearly the same time, planar and spatial multibody programs (KAM and COM-MEND) were being developed by Cooper et al.^[3] and Knappe^[4] to perform kinematic analyses of both open- and closed-loop industrial linkages. Together these parallel efforts provided the framework for developments by Chace and Korybalski^[5] of the formalism underlying the two-dimensional kinematic and dynamic multibody analysis code DRAM (Dynamic Response of Articulated Machinery), that represented the first generalized software program to provide time response of multifreedom, constrained, open- and closed-loop, mechanical machinery undergoing large-displacement behavior. Four years later, in 1973, Orlandea^[6] broke new ground in the development of a three-dimensional, type-variant, simulation program based on electrical network analogs, newly-developed sparse matrix processing, and advanced stiff integration algorithms. This program, which used absolute coordinates and an equation assembly procedure similar to most finite element programs, was a precursor to the present ADAMS software.

Much of the research and development work mentioned above, as well as a profusion of concurrent multibody research and development carried out by many other leading researchers in the field, was directed at extending and improving the efficiency of the underlying formalisms. These efforts often resulted in public domain software that was not well-suited to widespread general commercial usage.

In 1977, a U.S.-based private corporation, Mechanical Dynamics, Inc., (MDI), was formed to further develop multibody software and to distribute it commercially as an integrated element of corporate Mechanical Computer Aided Engineering (MCAE) software systems. MDI has its worldwide headquarters in Ann Arbor, Michigan, where it enjoys close affiliation with a number of the outstanding research centers existing within the University of Michigan academic system. The company began by redesigning and releasing a two-dimensional software program that was a successor to the university-developed DRAM program.

With the help of a dedicated team of multibody experts, computer interface development experts, technical writers, and hardware specialists, MDI released a fully-functional, commercial, three-dimensional, large-displacement, multibody kinematic, static, and dynamic simulation tool, named ADAMS, in 1980. Since then, over forty man-years of development effort^[7] has been expended in dramatically increasing the structure and capabilities of the proprietary software to address active control systems, part flexibility, static analysis, impact, linearization and modal analysis, pre- and post-processing, complicated internal and environmental forces of both a linear and nonlinear nature, as well as complex joint types and nonstandard constraints. This work has led to four major revisions and releases of the software since its commercial debut. Along the way, many other auxiliary software programs were developed and released to expand the applicability of the code. The continual enhancement of the software has led to a steadily increasing base of commercial and university installations, as shown in Figure 5. Major contributions to the development of this software were provided by Mr. John Angell, Dr. Nicolae Orlandea, Dr. Donald Calahan, Dr. Milton Chace, Mr. Rajiv Rampalli, Dr. David Benson, Dr. Thomas Wielenga, Dr. Joseph Whitesell, Mr. Michael Steigerwald, Mr. Jerry Green, Dr. Vikram Sodhani, Mr. Kristen Overmyer, Mr. Jeffrey Flinn, Mr. Mark Zykin, Dr. Liang Tang, and Dr. Robert Ryan.

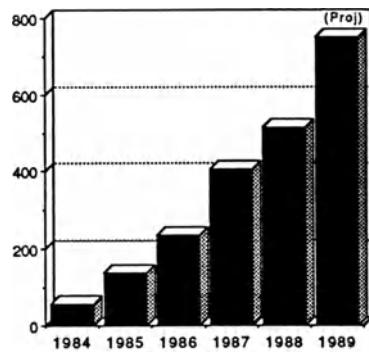


Fig. 5. MDI Software Installations

III. Applications

Before the details of the features and formulations of ADAMS are discussed, an overview of the commercial applicability of the software will be presented. This will begin with a client classification indicating who uses the software, what they use it for, where they use it, when they apply it, why they require its capabilities, and how they employ it in the MCAE process.

Following this is a breakdown of the applications into general categories.

Typical users of ADAMS software include engineering analysts and designers with educational backgrounds in physics, mechanical engineering, aerospace engineering, or mathematics. A significant percentage of these users have advanced graduate degrees in one or more of the fields mentioned and have studied rigid-body mechanics at some point in their career. For individuals to use the programs effectively, they need only be able to physically describe a mechanical system's geometry, mass, inertia, compliance, damping, and external forces in terms of idealized elements and then be able to interpret kinematic and kinetic output results.

Figure 6 provides an indication of what types of systems are most frequently analyzed with ADAMS. As shown, over 40% of the software license revenue is obtained from automotive manufacturers or suppliers. Roughly one-quarter of the license

revenue comes from aerospace/aeronautic companies, while the rest is split between general machinery manufacturers, construction and agricultural machinery producers, electro-mechanical device manufacturers, universities, and various other organizations. The software is used extensively in more than twenty countries in North America, Europe, and Asia. Companies that use the code are, more often than not, very large organizations employing hundreds or thousands of engineers and are often classified in the international Fortune 500.

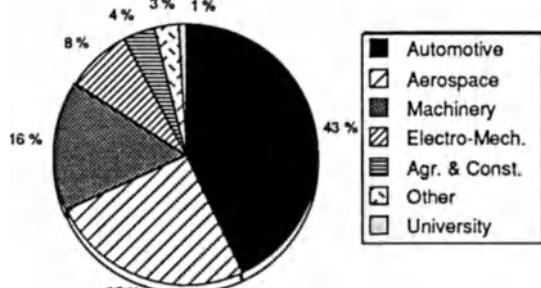


Fig. 6. ADAMS User Base

The engineering development phase in which ADAMS is most often required varies significantly from one application to the next. In general, however, ADAMS is applied in one of three modes: to predict characteristics and motion of a new design, to evaluate performance of an existing design, or to reconstruct the operation of a hardware system exhibiting anomalous behavior. Thus, ADAMS may be used at the outset of a design cycle to validate system concepts and to obtain approximate loads, then again after detailed component design has been completed to predict and refine system motion, and finally after full prototypes have been constructed and the system has been manufactured.

ADAMS is used in a variety of configurations from a stand-alone product to a fully integrated

MCAE package. Many organizations have embedded ADAMS software into the heart of their computerized engineering design procedures.

The rationale for relying on ADAMS is that it reduces the number of hardware prototypes required by allowing for accurate computer-based predictions (simulations) of system behavior. The software has been fully validated with an extensive library of example problems having known solutions. By alleviating the need for specialized programming and debugging, quicker turn-around on results can be obtained and overall project schedules can be shortened. The net effect is that better mechanical system designs are produced for less cost with ADAMS.

To provide the reader with a clearer picture of the types of mechanical systems that have been modeled and analyzed with ADAMS, the following few paragraphs describe specific examples from the general categories of automotive; aeronautics and astronautics; agriculture, construction, and off-highway equipment; biomechanics; and general machinery and mechanisms.

Figure 7 illustrates a small subset of automotive systems modeled with ADAMS. The systems shown include full vehicle models with suspension linkages, torsion bars, stabilizers, struts, steering mechanisms, drivetrains, engines, tires, cam and valve mechanisms, as well as electronic controls for speed, braking, and variable-power steering. These systems are used to predict and evaluate nonlinear, large-displacement, vehicle response to inputs from the following: drivers; wind gusts; aerodynamic lift and drag; road imperfections affecting ride quality, handling, and durability; and obstacles. References [8-12] discuss details of a few of these analyses. Also shown in the figure is a full dynamic model of a motorcycle used to study handling over bumps and to investigate specialized phenomena such as high-speed weave. Other automotive systems often modeled, but not shown, include windshield wipers, convertible top mechanisms, sunroof retraction devices, controlled seating systems, quick-disconnect fueling apparatus, door and window mechanisms, trunk and hood articulation systems, and differential gearing assemblies.

Aeronautical systems provide perfect applications (see Refs. [13,14]) for ADAMS since, by design, they typically undergo large overall motions involving coupled rotations and translations. A few select aeronautical systems modeled by ADAMS are illustrated in Figure 8. These include prop-driven aircraft, rotorcraft (helicopters), and jet aircraft whose dynamic performance is predicted during take-off, flight, and landing maneuvers in order to insure stability, absence of flutter characteristics, and proper durability. Modeled subassemblies include landing gear and flexible rotor-blade/bucket assemblies, as shown, as well as guidance and control systems, ailerons and rudders, turbine and impellers, hatch and safety systems, and weapon system release mechanisms.

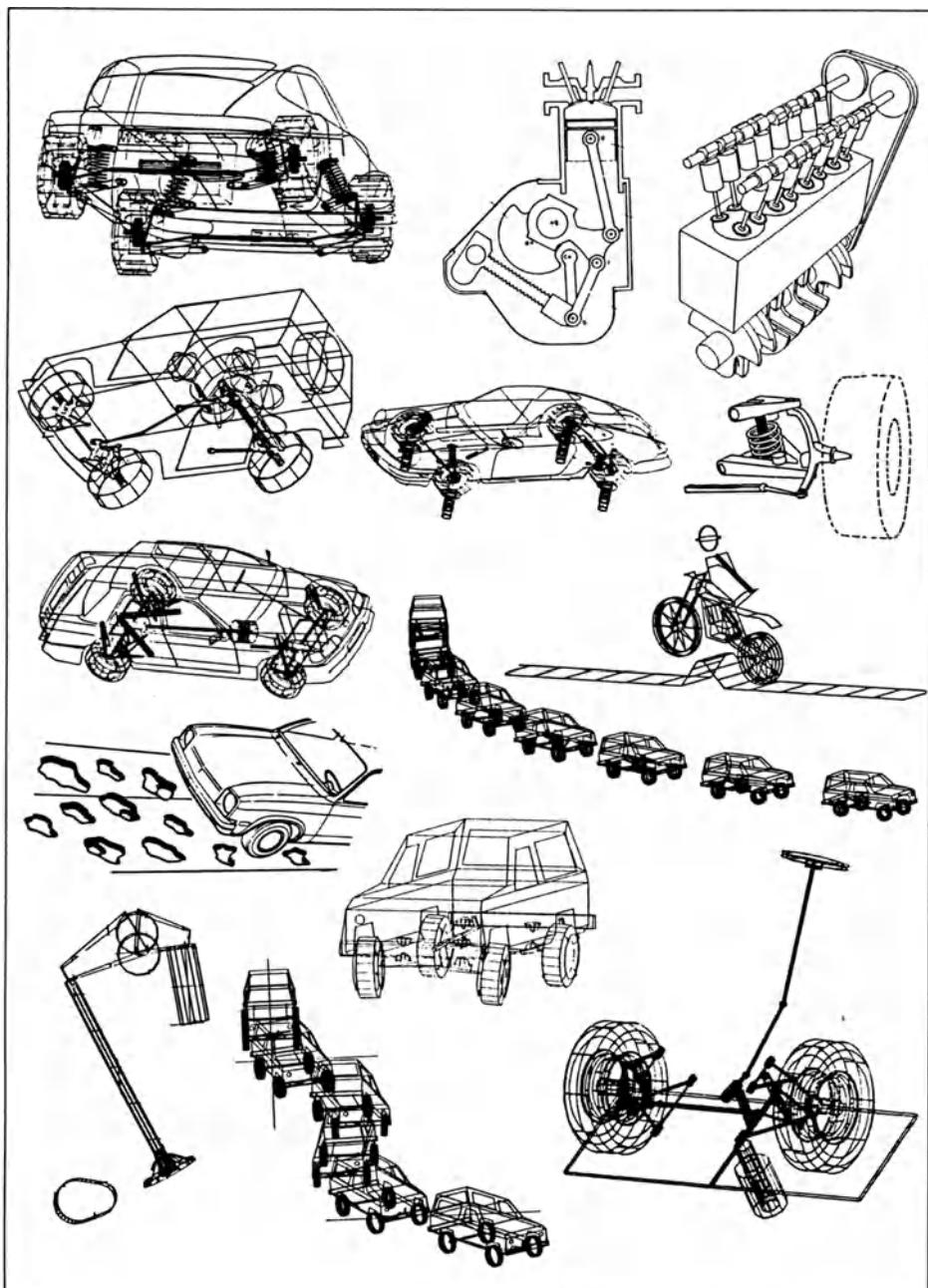


Fig. 7. Automotive Systems Modeled Using ADAMS

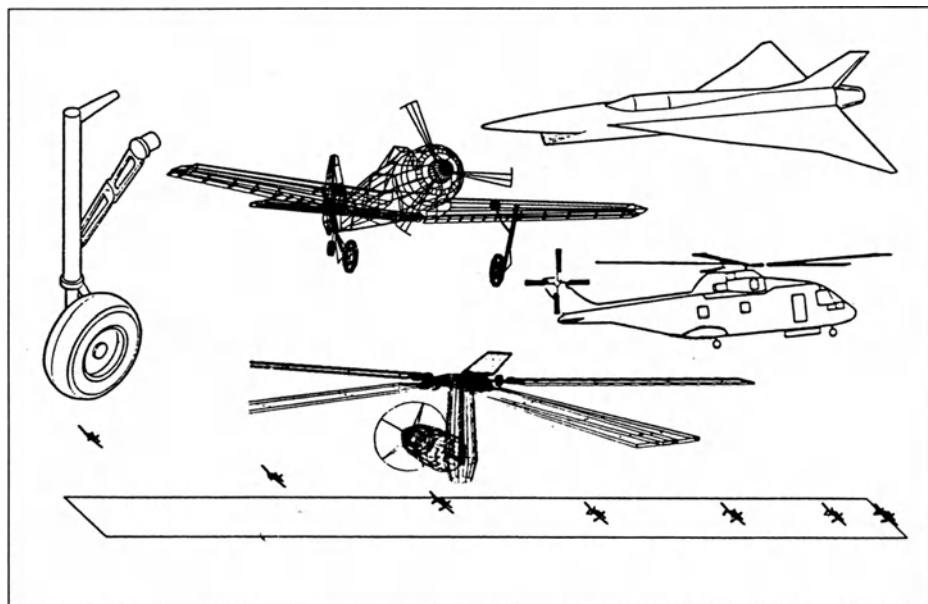


Fig. 8. Aeronautical Systems Modeled Using ADAMS

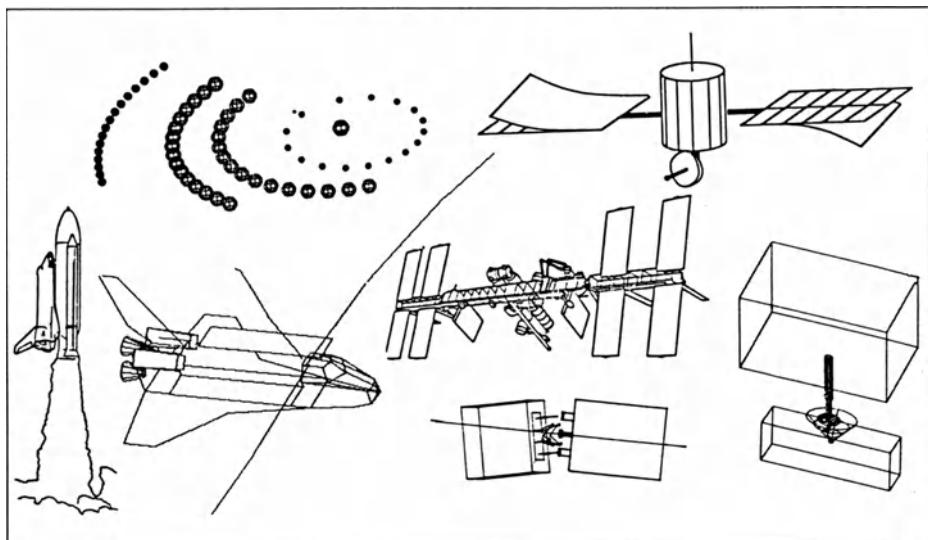


Fig. 9. Astronautical Systems Modeled Using ADAMS

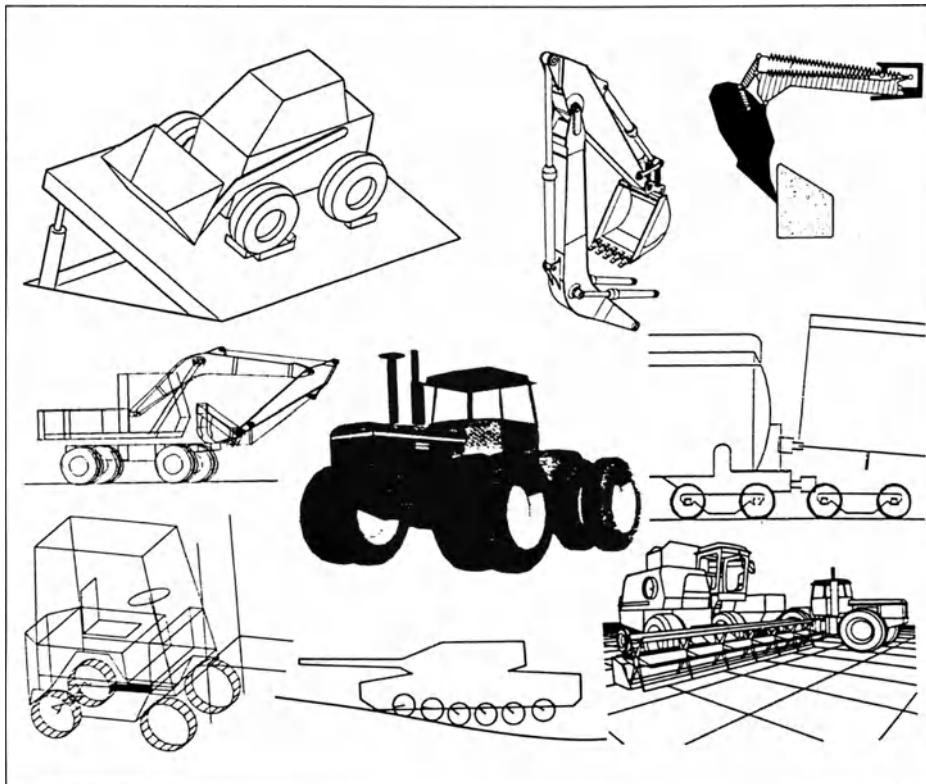


Fig. 10. Agricultural, Construction, and Off-Highway Systems Modeled Using ADAMS

Astronautical or aerospace applications include studies of orbital and attitude dynamics of satellites, space transportation vehicles, interplanetary probes, and space stations. Launch, deployment, docking, retrieval, and maintenance of these systems are also typically analyzed. For many of these systems, it is not easy to carry out appropriate experimental tests on earth; hence, simulation provides one of the few cost-effective ways of validating the design and studying implications of possible failure modes. Figure 9 illustrates a space shuttle launch procedure studied to determine launch loads, a planetary model of the inner solar system used to predict the path of Halley's comet, a flexible dipole antenna/orbiter configuration analyzed to determine the effect of antenna flexibility on overall attitude motion, as well as clamp and boom docking mechanisms investigated for feasibility of complete docking. The space station model shown was used to evaluate the effect of solar panel articulation on system orientation; and the earth-orbiting communications satellite was analyzed to determine the influence of solar radiation (thermal heating) and atmospheric drag on system deformation dur-

ing orbit. For additional information regarding the use of ADAMS in aerospace systems, see references [15-18].

Many of the early applications of ADAMS involved agricultural^[19], construction^[20], and off-highway vehicles as depicted in Figure 10. These systems require great durability; hence, accurate load determination is paramount in the design process to insure meaningful results from fatigue life predictions. ADAMS is the perfect tool to allow an engineer to model heavy equipment, to subject the system to standard operating maneuvers as determined from field tests and telemetry, and then to calculate the load path and magnitude of forces acting on all components. The resulting forces provide the raw data for further stress and strain finite element calculations. The systems shown in the figure include an inclined bulldozer model used in a stability analysis, an articulating front-end loader studied to determine reach and interference, a forklift system investigated for maneuverability, and a shovel assembly that provided for hydraulic cylinder capacity determination. Also shown are a plow mechanism used to simulate avoidance of imbedded rock and the extraction of surface rock, a tracked tank vehicle with a controlled turret and gun that required a design that isolated terrain disturbances from the gun pointing mechanism, a tractor modeled to evaluate and improve handling and ride quality, a pair of railcars investigated in an accident reconstruction legal case to determine speed of impact required to puncture the fuel tanker car, and lastly, an agricultural harvester assembly that was studied to determine failure mechanisms.

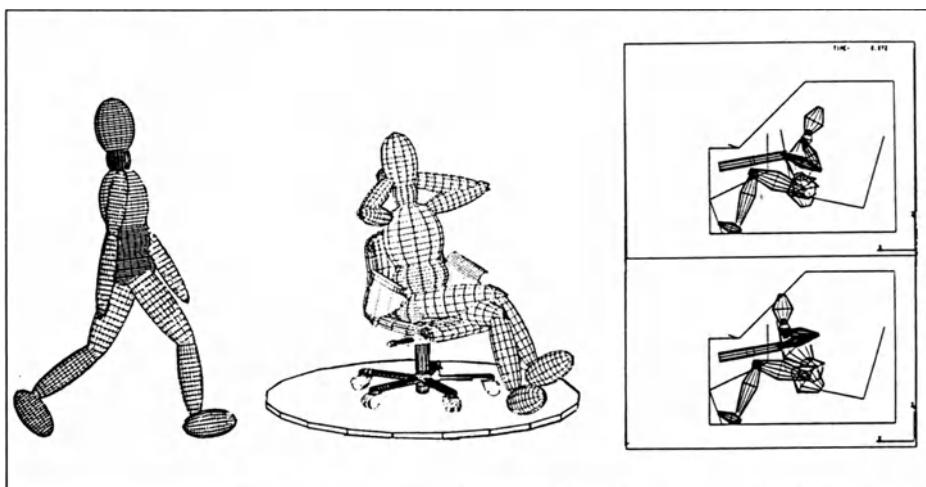


Fig. 11. Biomechanical Systems Modeled Using ADAMS

Biomechanical systems are perhaps the most natural multibody systems, but, until recently, they have not been extensively modeled because of the complexities involved with accurate specification and management of the data defining such models. Figure 11 displays a few sample ADAMS models of human bodies that were created by using the ADAMS/Android program to obtain joint definition, mass and inertia properties, geometry, initial conditions, and prescribed joint motions. The computer graphics included here illustrate a human body model which could be used to study gait, a model of a human sitting in a chair^[21] that provided data concerning chair stability and response, and a human driver model shown in a vehicle crash simulation.

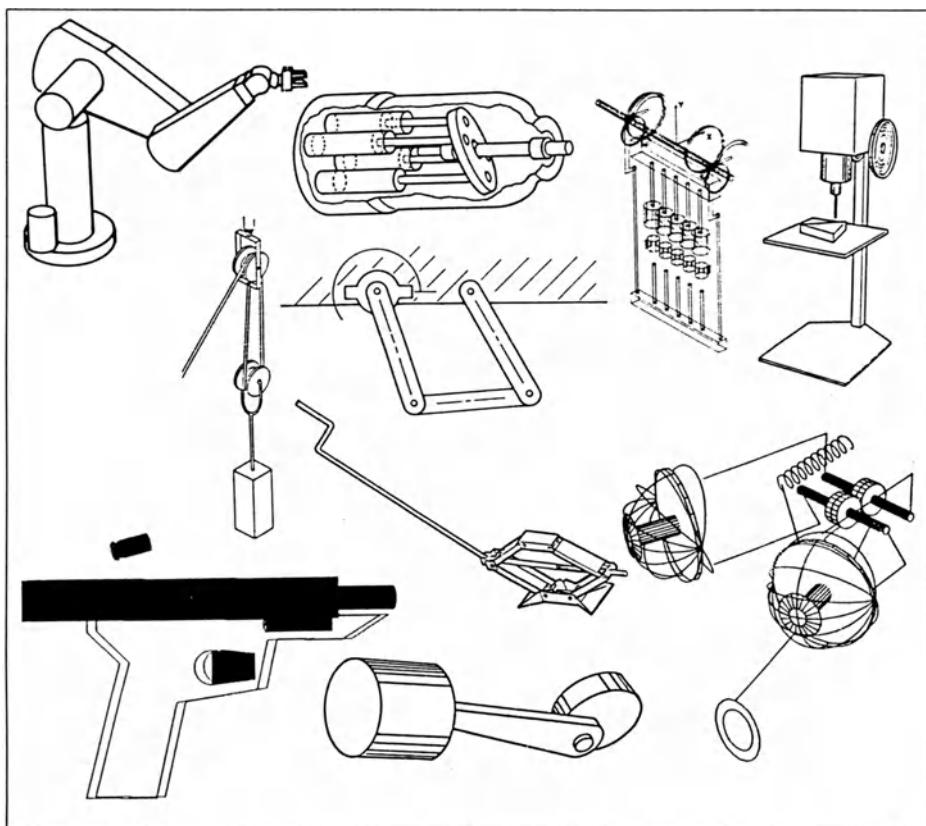


Fig. 12. General Machinery and Mechanism Systems Modeled Using ADAMS

Figure 12 includes many other general mechanisms that have been modeled by ADAMS, but that do not fall naturally into one of the previous categories. The handgun shown was modeled to determine cartridge trajectories as well as to evaluate jamming problems. The block and

tackle and the manual lifting jack were analyzed to calculate mechanical advantage and to determine loads. The Stirling engine model was used to predict motion and actuation of the swash plate, while the juicer mechanism shown next to the machine tool was studied to insure proper part alignment and machine operation. The figure in the lower right is a model of a puppet's eyelid-opening mechanism actuated by a finger-pull. The system was modeled to insure proper mechanical eyelid closure with a one-inch finger travel.

The systems shown and described above constitute only a small fraction of the mechanical devices modeled and analyzed by ADAMS and DRAM in a commercial setting during the past twelve years. The software has been employed to model systems containing as few as one part or as many as a couple of hundred parts. Because of the systematic equation assembly process employed, there is no software-inherent limit to the number of system degrees of freedom or state equations that can be included in an ADAMS model. Recently-constructed ADAMS models of automotive and railway systems have exceeded five hundred degrees of freedom and have been characterized by more than ten thousand state equations. Complex phenomena such as higher-pair contact, nonlinear flexibility, unsteady aerodynamics, discrete digital control, flexible body deployment, cable routing, and impact are routinely treated with ADAMS. Apart from commercial applications, the software has also been used in over fifty teaching and research universities worldwide to assist professors in instructing students in rigid body mechanics.

IV. Program Features

A. Overview

ADAMS' primary advantage over its competitors lies in its generality, its extensive built-in capabilities^[22], and its open-ended architecture that allows users to easily create extensions to ADAMS functionality using data set features and external subroutines. Like NASTRAN in the finite element arena, there are very few multibody system analyses that cannot be performed in ADAMS. ADAMS does not restrict a user to any class of mechanism, but instead provides techniques to treat systems with rigid or flexible bodies arranged in open-chain patterns, tree patterns, clusters, closed-loops, and multiple closed-loop configurations. Thus, *free-free* systems such as satellites and space probes are treated in the same manner as open-chain robotic manipulators, and closed-loop linkages and trusses. The user is not expected or required to be knowledgeable about topology theory or constraint checking. The program automatically performs an initialization pass on all user-supplied data; inconsistent or improper data is flagged and redundant constraints^[23] are automatically eliminated. To provide for the handling of a great variety of mechanism systems, ADAMS employs the most advanced *stiff* integration and sparse solution algorithms^[24–26]. These algorithms have been

demonstrated^[6] to be as much as one hundred times more efficient than other commonly used algorithms in multibody system analysis. The user may start, stop, or restart the simulation at any time. A full suite of program diagnostics, help facilities, and debugging aids are embedded into the software to guide the user, and complete working files are saved to permit full-feature restarts.

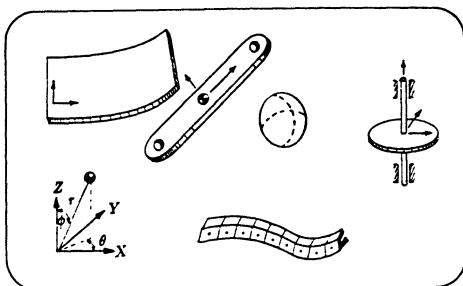
In order to simplify the process of modeling mechanical systems of many different types, a wide array of analytical and physical model-building elements have been incorporated into ADAMS. For the sake of discussion, these will be separated into the following groups: reference frames and coordinate systems, parts, connectors, sensors, functions, differential equations, user-accessible subroutines, and graphic icons. The individual ADAMS elements available in each of these categories are summarized in Figure 13 and discussed below.

B. *Modeling Elements*

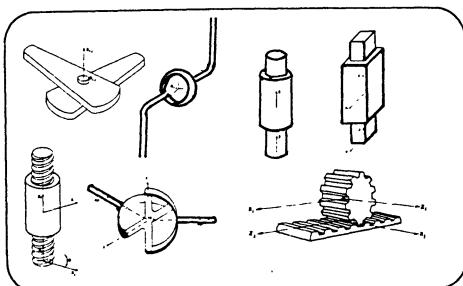
ADAMS users construct system models by first defining reference frames and coordinate systems. The program allows these to be inertially fixed or moving. Next, physical components of the mechanical system are modeled using idealized ADAMS parts, which can be either rigid bodies, flexible bodies^[27,28], or particles.

Using one of a number of simple algorithms supplied by ADAMS for defining location and orientation, the user then positions the parts in three-dimensional space. After this, the user describes the possible relative motions between parts of the system by specifying the type of physical mechanical joints (see Fig. 13) such as revolutes, ball and socket joints, sliders, universals, etc., that connect the bodies. This is easier and quicker than mathematically defining the number of relative degrees of freedom and generalized coordinates as in more abstract multibody modeling software. To analyze systems in which bodies are connected in nonstandard ways, ADAMS provides joint primitive elements that allow for any type of constraint to be constructed between parts. For systems having parts that move in some prescribed fashion relative to one another, ADAMS offers elements to model translational and rotational motion drivers and path followers.

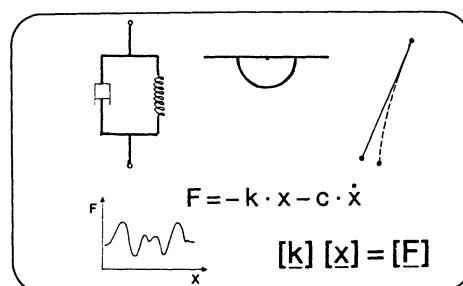
Having identified joint constraints and a ground part, an ADAMS user typically defines the internal compliance elements and external forces influencing the behavior of the system. To facilitate this task, which is perhaps the most difficult in system modeling, a long list of compliant connectors, actuators, and environmental forcing elements are available as can be seen in Figure 13. These compliant connectors include springs, dampers, beams, bushings, bearings, tires, impact elements, and extensive functions to permit modeling of nearly any other connector, such as hydraulic and pneumatic cylinders, friction pads, plates, shells, etc. Actua-



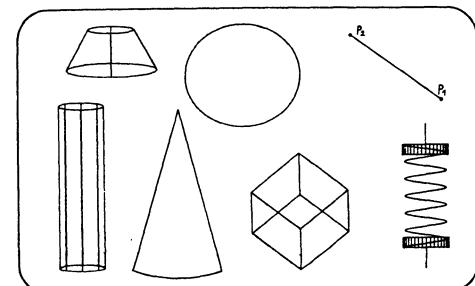
- Coordinate Systems : Inertially -Fixed or Moving
 - Input
 - Translation : Cartesian
 - Orientation : Euler Angles, 3x3 Matrices
 - Output
 - Translation : Cartesian, Cylindrical, Spherical
 - Orientation : 24 Euler Angles, 3x3 Matrices
 - Euler, Rodriguez Parameters, Y- P- R
- Parts
 - Rigid Bodies, Particles*, Momentum Wheels*
 - Flexible Bodies : Physical / Galerkin*



- Standard Joints
 - Fixed, Revolute, Prismatic, Universal, Planar
 - Cylindrical, Spherical, Screw, Rack & Pinion
- Non-Standard Joints
 - Joint Primitives, Couplers, C. V. Joint
 - Spur, Helical, Planetary Gears
- User Constraints
 - Holonomic : Path Following, General
 - Non-Holonomic : Roll Without Slip



- Standard Forces
 - Springs, Dampers, Bushings, Beams
 - Impact, Hydraulic /Pneumatic Cylinders, Fields
 - Plates, General Matrix, FEA Superelement
- User Defined Forces
 - Functions : Step, Harmonic, Gap, Haversine, Polynomial, Spline, Fourier Series
 - Subroutines : Cam and Surface Contact Models Solar Pressure, Aerodynamics Non-Deterministic Forces
- Tires : Handling, Durability, Ride Models
- Actuators : Action-Reaction Pairs Action-Only Thrusters Prescribed Force and/or Torque



- Arc, Box
- Cylinder, Circle,
- Outline, Force Vector,
- Frustum, Spring-Damper

Fig. 13. ADAMS Modeling Elements

tors can be modeled as servomotors supplying action/reaction torques, or strut rods supplying action/reaction translational forces. Users can also model action-only forces and torques as well. Environmental gravity forces can be defined in any direction, and other environmental forces such as unsteady aerodynamic effects^[29] and atmospheric drag can be defined in terms of program-supplied functions, table look-up facilities, and user-written subroutines linked to the software.

When the topological, geometrical, kinematic, and kinetic system descriptions are complete, a user often finds it advantageous to define measurement sensors as output facilities or as input to forces described in the system. ADAMS extends to the user a set of capabilities to measure nearly any system variable by using simple request statements. The sensors can provide part orientation information in any one of more than twenty different standard forms, and can give location data in Cartesian, cylindrical, or spherical coordinates. Kinematic sensors describing position, velocity, and acceleration, as well as kinetic sensors indicating applied, reactive, and inertial forces, are available.

When a user needs to model a complicated force, motion, or constraint that is not easily described with the elements listed above, ADAMS' "FUNCTION" capability provides the perfect solution. The user simply describes the force, motion, or constraint in terms of a combination of user-defined constants, system constants, system variables, arithmetic IF's, FORTRAN-77 library functions, standard mathematical functions, or ADAMS-supplied functions. The user can describe these with normal data set statements or, in more complicated cases, by writing a user-written subroutine and linking it with ADAMS to form an executable program.

When a user already has a detailed dynamic model of a particular component or control system defined in terms of algebraic or differential equations, ADAMS provides a facility to allow the user to input these equations directly to the database and to have them solved along with the system equations that ADAMS automatically formulates based on the user-defined topology. Lastly, in order to obtain a better physical interpretation of the motion and configuration of the system, ADAMS permits the user to describe components of the system in terms of graphic primitives and icons, which are then used to track the motion of the system in a visual way. External to ADAMS, these geometrical descriptions may take any form whatsoever, including solid parametric models. Inside ADAMS, the elements shown in Table 1 are supported.

Table 1

GRAPHICS
• Arc
• Box
• Circle
• Cylinder
• Outline
• Force Vector
• Frustum
• Spring-Damper

C. Analysis Types

Table 2, at right, lists the analysis types^[30] available to ADAMS users. If the system model has been completely constrained so that there are zero degrees of freedom, but perhaps one or many prescribed motions, the system is said to be *kinematic*. In this case, ADAMS allows the user to perform a kinematic analysis wherein the program determines the configurations at any point in time based on user-supplied, time-dependent information and the system constraints. The program also calculates the time-dependent velocities and accelerations of points in the system as well as the reaction forces. No numerical integration is necessary.

Table 2

ANALYSIS TYPES
• Kinematics
• Initial Conditions
• Statics
• Quasi-Statics
• Modal Analysis
• Nonlinear Dynamics
• Design Sensitivity
• Optimization

If the system has at least one degree of freedom, ADAMS permits the determination of static equilibrium configurations, modal eigendata, or transient dynamic response behavior. For any of these analyses, ADAMS first performs an initial assembly analysis wherein the initial positions and velocities are checked for consistency with system constraints, and the system is perturbed in the least possible fashion to achieve proper assembly. The user is free to specify some initial condition data as "exact" and other values as approximate. The program varies only the approximate quantities during assembly.

For static analyses, all inertia forces are assumed to be zero, and the applied loads are balanced against the reaction forces to determine an equilibrium position, if one exists. Successive static equilibrium configurations, based on different time-dependent motions, constraints, or forces are also possible.

When a user wishes to determine modal data concerning a system, ADAMS/Modal linearizes the fully nonlinear equations of motion in sparse form about a user-defined operating state and reduces the equations to a smaller set of independent equations in classical eigenvalue form^[31]. An eigenvalue problem is solved, and natural frequencies, damping ratios, and mode shapes are produced.

For transient dynamic analysis, ADAMS includes all inertia forces and torques in the simulation and obtains a solution to the equations of motion by integrating forward in time.

ADAMS can produce design sensitivity matrices for use in evaluating effects of changes in specific design variables and also for inclusion in system optimization software packages^[32-34].

Lastly, as mentioned above, optimization can be performed with the ADAMS/MOST package, which offers the capability of optimizing the kinematic performance of a planar or spatial mechanism.

D. *Input/Output*

To communicate effectively with ADAMS, a user must become familiar with the **content** and **form** of the input that ADAMS expects and the output that it produces. This section begins with a description of the content of the input and output, and then summarizes the form of each.

The input for ADAMS (see Table 3) consists of three essential types of information: execution instructions, bulk modeling data, and load case data. Execution instructions provided by the user inform ADAMS of the type of analysis to be performed, the duration of the analysis, and the data files to be manipulated. Bulk modeling data consists of the system description in terms of reference frames, geometry, body types, mass, inertia, compliance, constraints, topology, component models, differential equations, and graphic entities. Finally, load case data involves the specification of initial conditions, applied forces, applied motions, and sensor requests for each case to be studied.

Table 3

INPUT	OUTPUT
<ul style="list-style-type: none"> • Execution Instructions <ul style="list-style-type: none"> • Analysis Control • Query System • Plot Response • Draw System • Bulk Modeling Data <ul style="list-style-type: none"> • Reference Frames • Geometry • Mass Properties • Topology • Compliance • Component Models • Graphics • Load Case Data <ul style="list-style-type: none"> • Initial Conditions • Applied Forces • Applied Motions • Sensor Requests 	<ul style="list-style-type: none"> • Nonlinear Results <ul style="list-style-type: none"> • System Displacements • System Velocities • System Accelerations • Reaction Forces • Linear Results <ul style="list-style-type: none"> • Mode Shapes • Natural Frequencies • System Jacobian(s) • State Space Matrices • Design <ul style="list-style-type: none"> • Sensitivity Matrices • Output Files <ul style="list-style-type: none"> • Diagnostics • FEA Loads • Graphics • Requests • Restart • Tabular Output

Output from ADAMS includes time-dependent system configuration information, velocities, accelerations, reaction forces (loads), values of user-defined variables, equilibria, natural frequencies, mode shapes, modal damping, state space matrices, design sensitivity matrices, and diagnostics.

In an effort to provide for maximum ease of interfacing ADAMS with other software, all data exchange with ADAMS is conducted in the form of data files. All pre- and post-processors, as well as complementary software, read and write these files, which interact with ADAMS. The

individual data formats included in these files vary significantly from one data type to the next, but data associativity is kept consistent to the extent possible.

ADAMS input files consist of a command file containing execution commands, an ADAMS Data Language (ADL) file containing bulk data, and an initial conditions file with load case-specific data. The ADL file, which transfers most data to ADAMS, consists of a set of statements defining system elements. The statements, which can be included in any order, conform to a standard format and syntax. An example of an ADL input file is included in section VII. The general format for all ADAMS statements is:

$$\text{NAME/id, } \text{ARG}_1 \left[= \begin{Bmatrix} v_1, \dots, v_n \\ c(v_1, \dots, v_n) \\ e \end{Bmatrix} \right], \dots, \text{ARG}_n \left[= \begin{Bmatrix} v_1, \dots, v_n \\ c(v_1, \dots, v_n) \\ e \end{Bmatrix} \right]$$

where the braces $\{\dots\}$ indicate that the user should select one item, and the brackets $[.\dots.]$ denote that the information is optional. NAME indicates the statement type. Each statement type describes a supported ADAMS element (see section IV.B above). The statement arguments, namely ARG_1 through ARG_n , may indicate a condition, values (v_1, \dots, v_n) , a character string (c) , or an expression (e) . For example, to specify that a spherical (ball and socket) joint connects a point labelled 21 on body number 2 to a point labelled 31 on body 3, requires the following statement:

JOINT/1,I = 21,J = 31,SPHERICAL

where the number 1 indicates the label of the joint, while I and J are standard arguments of the JOINT statement and represent the two connecting points. For complete details, refer to the ADAMS User's Manual^[22].

ADAMS output files include (1) a formatted, tabular output data file, (2) a message file of diagnostics, (3) a graphics file containing the location and orientation of graphic icons at various instants in time, (4) a request output file containing output to be plotted or displayed, (5) an initial condition file for restarts, (6) a Jacobian file for sensitivity analyses, (7) a condensed results file for interaction with third-party software, (8) a FEM file for interaction with finite element programs, and (9) a modal file containing the state space matrices and modal data calculated from the linearized equations of motion.

E. MCAE Interfaces

At this point, it is probably abundantly clear to the reader that the use of modern multibody tools such as ADAMS can help produce better mechanical products and can significantly increase productivity of design groups. However, realizing these benefits requires that the tools used for multibody system simulation interact effectively with the multi-

tude of other MCAE tools being used by members of design teams for various other tasks necessary for complete analysis and design of a product. This is the impetus behind the aggressive efforts of MDI to both develop a common communications protocol for interaction with ADAMS and to work with developers of other leading worldwide MCAE software programs to build seamless links between their software and ADAMS. These efforts will result in a file interaction process that will allow an engineer to easily use ADAMS in conjunction with other popular programs specializing in geometric modeling, structural analysis, component actuation modeling, analytical and experimental modal analysis, control system design and analysis, and optimization. Figure 14 illustrates the interfaces that have already been constructed or are under development. Those not completed to date are marked by an asterisk.

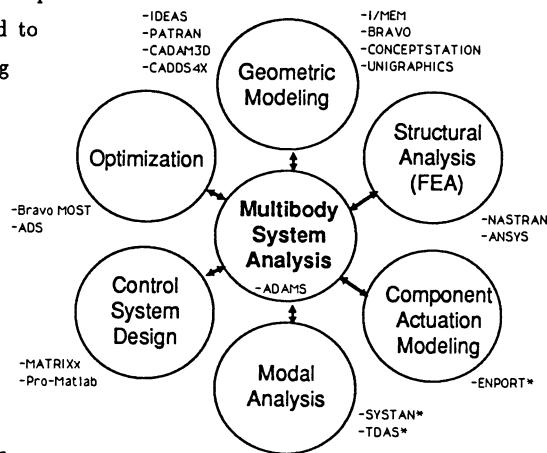


Fig. 14. MCAE Software
Interfaces to ADAMS

While the file interaction process described in the previous subsection unifies data transfer and promotes software interconnectivity, a typical user desires interactions with the software in a more intuitive manner. Therefore, a number of MDI and third-party pre- and post-processors have been developed to permit an ADAMS user to visually create part geometry, establish part connectivity, apply forces, and display output results in an animated fashion. In other words, the software provides a modeling environment that conforms to the operating mode that the user prefers rather than forcing users to communicate all data in a structured textual format. These interfaces then do the work of converting the data between the established file structure of ADAMS and the graphical user interfaces. Figure 15 provides a glimpse of one such modeling interface under development at MDI. At present, graphics-based commercial pre- and post-processors^[36] for ADAMS are available from Aries Technology, Inc.; Auto-trol Technology, Inc.; Cadam, Inc.; Intergraph Corporation; McDonnell Douglas Manufacturing & Engineering Systems Company; Mechanical Dynamics, Inc.; PDA Engineering; Prime Computer, Inc.; Schlumberger Technologies; Structural Dynamics Research Corporation; and TEDAS GmbH. In addition, many other interfaces have been developed within large engineering organizations, such as the HAL^[19] sys-

tem at Deere & Co., AMIGO at Audi^[37], and MOGESSA^[33] at Volkswagen. With the wealth of geometric modeling interfaces available, users are able to choose the input format most natural to their applications.



Fig. 15. ADAMS Pre- and Post-Processing

At some stage of a mechanical design cycle, engineering analysts are interested in determining stresses, strains, and the strength of each component. This is commonly done using finite element analysis (FEA) procedures. One of the most troublesome aspects of finite element structural analysis, however, is the specification of system loads and boundary conditions. ADAMS provides interfaces^[38] to finite element codes like ANSYS and NASTRAN, thereby allowing an analyst to perform a time-domain simulation in ADAMS to determine loads and then transfer these directly to FEA codes as input forces. Accordingly, interfaces are being developed to also permit elemental compliances obtained from FEA codes to be used in ADAMS to describe elastic parts. In other words, a user would be able to form a superelement stiffness and mass matrix of a component in an FEA code and transfer it to ADAMS to describe a physically-discretized body.

In many engineering companies, it is typical for certain groups within the organization to have responsibility for modeling various subsystems of a mechanical design. ADAMS permits these subsystems to be modeled as separate ADAMS models and combined later, or allows users

to transfer models created in other software to ADAMS through a differential and algebraic equation facility embedded in ADAMS. Also under development are specific links to third-party component actuator modeling packages, such as those based on bond graph techniques.

Although ADAMS provides the capability to linearize system equations and to calculate modal properties, it is often advantageous to use component modal synthesis techniques to combine modes from various systems and to perform forced response analyses of the combined linear systems. To facilitate this, future versions of ADAMS/Modal will supply data in a format that can be read by modal synthesis programs.

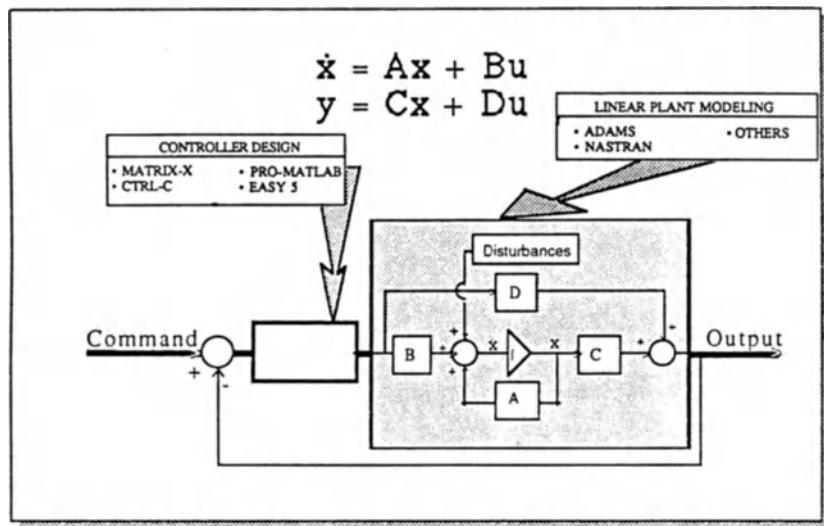


Fig. 16. Use of ADAMS in Control System Design

Despite the fact that ADAMS can easily incorporate a control system model^[35] in a mechanical system model and evaluate its effect on system behavior, multibody programs are not, in general, well-suited to serve as full-fledged control system design packages. There are a number of such programs on the market, and the best of these provide very extensive classical and modern control design techniques to aid the user in developing an appropriate control law complete with sensors and actuators. When a user begins working with these programs, one of the most difficult challenges that appear almost immediately is that of specifying a high-fidelity plant model governing motion of the mechanical system. ADAMS helps the control system designer over this hurdle by^[39] automating the process of forming accurate equations of a complicated nonlinear system, by linearizing these equations about an appropriate operating state, and by writing out state-space matrices in the standard A, B, C, D form (see Fig. 16) required by

the control design software packages. Once the control law is designed, the user can bring the control law, along with the sensor and actuator designs, back into ADAMS via differential and algebraic equation translation in order to validate the control algorithm in a true nonlinear environment.

Lastly, to interact with optimization packages^[33,34,40], ADAMS provides the user with the ability to precisely control kinematic, static, and dynamic simulations, and to write out sensitivity matrices for use in gradient procedures.

F. Documentation, Training, and Support

Multibody system analysis is not a trivial procedure. To perform it effectively often requires that the user have a good understanding of modeling methods and assumptions, rigid body dynamics theory, and other issues. To aid the user in this regard, there is an extensive network of documentation, training classes, and expert consulting support to complement ADAMS.

Available documentation includes a complete user's manual, an applications manual, and theory notes. Verification and benchmark sheets indicating the performance of the code on a suite of test problems is also available.

Classroom training provided by the distributors of ADAMS has been conducted for over seven years and has resulted in more than one thousand engineers obtaining instruction in the use of the code. There are classes for introductory and advanced users. These classes cover topics on general modeling, mechanism theory, applications, and program operation. In addition to classroom presentations, there are roughly five yearly international users conferences held in locations throughout the world. ADAMS is also getting more and more exposure in universities as software is being integrated into curricula for graduate studies in rigid body mechanics.

Even with superb training and documentation, new applications sometimes present a need for expert assistance in the use of the ADAMS software. ADAMS distributors handle this situation by providing users with telephone support and specialized consulting services supplied by ADAMS experts.

G. Availability

The availability of a software program is a function of the established distribution network and the hardware platforms on which the software executes, among other factors. This subsection begins by describing computer hardware currently supporting the ADAMS software and ends with a summary of the worldwide ADAMS distribution network.

Table 4 to the right illustrates the computer hardware vendors whose supercomputers, mainframes, and workstations support ADAMS software. The following models and operating systems currently apply: (1) all FX models of Alliant with the Concentrix operating system, (2) all Convex systems with Unix operating systems, (3) Cray models 1A, 1S, X-MP, and Y-MP with Cos versions 1.11 and higher or Unicos versions 4.0 and higher, (4) IBM models 30xx, 43xx, and plug compatibles (e.g., Fujitsu, Amdahl) with MVS/TSO or VM/CMS operating systems, (5) CDC Cyber with NOS VE, (6) all DEC models with VMS and Ultrix, (7) Prime 50 series with Primos operating system, (8) all Apollo models, except DN10000, with Aegis operating systems SR9.x and SR10.x, (9) HP 9000 series 300 and 800 with HP-UX, (10) Intergraph model IP-32C with Unix, (11) SGI series 3000 and 4D with Unix, and (12) Sun models 3 and 4 with SunOS(Unix). In addition, computer terminals from Digital Equipment Corp., Evans & Sutherland, and Tektronix, Inc. provide capabilities to serve as front-ends and graphic display devices for the software.

Table 4

ADAMS-Supported Hardware Platforms
<ul style="list-style-type: none"> • Supercomputers <ul style="list-style-type: none"> • Cray, NEC, IBM, Alliant, Convex • Mainframes <ul style="list-style-type: none"> • DEC, Prime, IBM, CDC • Workstations <ul style="list-style-type: none"> • Apollo, DEC, HP, Intergraph, SGI, SUN

ADAMS is distributed in North America, Europe, and Asia by Mechanical Dynamics, Inc., and affiliates (see Fig. 17 below). This network is quickly expanding to other locations throughout the world.

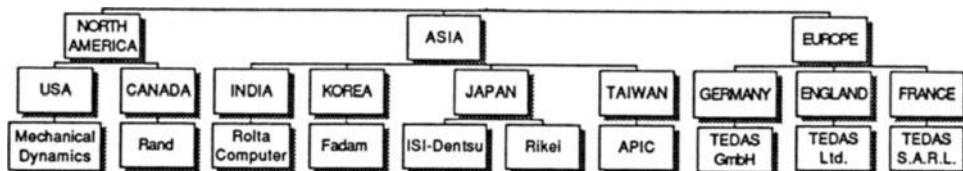


Fig. 17. ADAMS Worldwide Distribution Network

V. Analysis Methods

Many of the analytical and computational methods employed in ADAMS have been widely published in the past. Therefore, the details of the methods will not be reproduced here in their entirety; instead, this section will provide a summary of the general formulation and solution approaches used in ADAMS in order to put them into context with others. The interested reader may refer to references [7, 8, 30, 41-44] for more specifics.

As in finite element methods, ADAMS treats each part of a common type (e.g., rigid body) in an identical fashion. Using a variational form (Lagrange's equations of the first kind) of the classical Newton-Euler equations, the kinematical and dynamical equations of motion for these body types (unconstrained) have been derived *a priori* and pre-coded into the algorithm to be recalled whenever needed. This provides for a straightforward, systematic equation assembly process and a standard set of equations regardless of mechanical system application. As the solution of linear algebraic equations nearly always consumes the majority of CPU time in a multibody simulation, ADAMS has been specifically designed to maximize the efficiency of the decomposition and back substitution process by taking full advantage of sparse matrix processing techniques and by symbolically encoding the decomposition path within the code prior to run time. These efficiency considerations have continually guided the development of ADAMS.

The spatial generalized coordinates employed in ADAMS consist of three Cartesian coordinates of each part's mass center relative to a global origin and three orientation angles (Euler Angles) defining the attitude of the central principal axes of inertia of each part relative to mutually-perpendicular, inertially-fixed axes. In addition to these coordinates, ADAMS also employs part angular velocities and center of mass translational velocities, as well as applied force variables, and constraint forces (also known as Lagrange multipliers) as additional state variables.

When constraints act between various parts of the system, ADAMS does not reduce the number of coordinates down to a minimal independent set, but rather appends all of the configuration (joint) and motion (actuator) constraint equations onto the set of existing first-order differential equations defining the kinematical and dynamical equations for each part. The algebraic constraint equations have been pre-derived for each available ADAMS joint (i.e., revolute, spherical, etc.) and are recorded in the algorithm for use as necessary.

Additionally, instead of embedding the complete form of complex external forces into the dynamical differential equations, ADAMS identifies these forces as state variables in the dynamical equations. Then, ADAMS adds additional algebraic equations defining the full mathematical form of these forces. Any user-defined differential equations are appended last.

Thus, the system equations governing behavior of any mechanism in ADAMS consist of:

- six first-order dynamical equations of each part (relating forces to accelerations),
- six first-order kinematical equations of each part (relating positions to velocities),
- a single algebraic constraint for each scalar, holonomic, configuration constraint,
- a single differential equation for each scalar, nonholonomic, motion constraint,
- a single algebraic equation for each complicated, scalar force, and
- any number of user-defined algebraic or first-order differential equations.

These equations may be rearranged into the general form shown below.

$$\underline{M}(\underline{q}, \underline{u}, \dot{\underline{q}}, \underline{f}, t) = \underline{0} \quad (1)$$

$$\underline{N}(\underline{u}, \dot{\underline{q}}) = \underline{u} - \dot{\underline{q}} = \underline{0} \quad (2)$$

$$\underline{\Phi}(\underline{q}, \underline{f}, t) = \underline{0} \quad (3)$$

where an underscore denotes a matrix, a dot specifies a temporal derivative, and the parentheses indicate functional relationships. The symbol \underline{q} denotes the column matrix of generalized coordinates, \underline{u} represents the column matrix of generalized speeds, \underline{f} is the column matrix of constraint forces and applied forces, t represents time, \underline{M} is the column matrix of all differential dynamical equations and any user-defined differential equations, \underline{N} is the column matrix of differential kinematical equations and differential motion constraints, and $\underline{\Phi}$ is the column matrix of algebraic equations describing joint constraints and forces.

Eqs. (1-3) represent a very sparse (highly-uncoupled) set of mixed, nonlinear differential-algebraic equations (DAE's) governing behavior of an ADAMS model. These equations are solved in various combinations during kinematic, static, and dynamic analyses. Since the solution of sets of DAE's involves techniques that are quite different from those employed in solving standard ordinary differential equations (ODE's) found in most minimal-coordinate multi-body formalism^[44], some brief details of the solution process will be discussed here.

If a column matrix \underline{y} of state variables is defined as $\underline{y} = [\underline{q}, \underline{u}, \underline{f}]^T$, then Eqs. (1-3) can be written as the following single matrix equation.

$$\underline{G}(\underline{y}, \dot{\underline{y}}, t) = \underline{0} \quad (4)$$

This equation represents a set of nonlinear equations which must, in general, be solved in an iterative fashion to obtain a solution at any point in time. ADAMS solves these equations using a modified Newton-Raphson procedure. The equations are expanded in a Taylor series about an initial set of guesses for the state variables. At time zero, the initial guesses for the iteration procedure are obtained from the initial conditions; afterwards, they are set equal to past values of the state variables scaled in an appropriate manner. If \underline{y}_j and $\dot{\underline{y}}_j$ denote the j^{th} iteration for the state variables and derivatives, then the Taylor series expansion of the equations appears as (at a particular fixed instant in time):

$$\underline{G}_j + [\partial \underline{G} / \partial \underline{y}]_j \Delta \underline{y}_j + [\partial \underline{G} / \partial \dot{\underline{y}}]_j \Delta \dot{\underline{y}}_j + \dots = \underline{0} \quad (5)$$

where the brackets indicate a matrix, the bracket subscript j indicates that the quantity contained within the bracket is to be evaluated with all of the state variables and their derivatives set equal to their corresponding values at the j^{th} iteration, the symbol $\Delta \underline{y}_j$ is defined as $\Delta \underline{y}_j = \underline{y}_{j+1} - \underline{y}_j$, and the ellipses indicate terms of degree two and higher

in $\underline{\Delta y}_j$. If the higher degree terms are assumed negligible, then the following equality holds:

$$[\partial \underline{G} / \partial \underline{y}]_j \underline{\Delta y}_j + [\partial \underline{G} / \partial \dot{\underline{y}}]_j \underline{\Delta \dot{y}}_j = -\underline{G}_j \quad (6)$$

Equation (6) can be transformed from being a set of algebraic-differential equations to a set of standard linear algebraic equations if a numerical integration formula is introduced. ADAMS employs the implicit Gear variable-order, variable-step stiff integration algorithm which takes the following form:

$$\underline{y}_{n+1} = \sum_{i=1}^k \alpha_i \underline{y}_{n-i+1} - h \beta_o \underline{\dot{y}}_{n+1} \quad (7)$$

where n represents the time step counter (t_n is the value of time after the n^{th} integration time step), \underline{y}_{n+1} is the numerical approximation for $\underline{y}(t)$ at $t = t_{n+1}$, h is the step size given by $h = t_{n+1} - t_n$, k is the *order* of the algorithm employed, and β_o, α_i ($i = 1, \dots, k$) are real numbers referred to as the Gear integration coefficients.

Since the change in $\dot{\underline{y}}$ can be related to the change in \underline{y} by the relationship,

$$\underline{\Delta \dot{y}}_j = [\partial \dot{\underline{y}} / \partial \underline{y}]_j \underline{\Delta y}_j \quad (8)$$

it follows, from Eq.(7), that

$$\underline{\Delta \dot{y}}_j = -(1/h\beta_o) \underline{I} \underline{\Delta y}_j \quad (9)$$

where \underline{I} is an identity matrix.

Substituting the right-hand side of Eq.(9) back into Eq.(6) yields the following set of standard linear algebraic equations which can be solved using decomposition and back-substitution.

$$\left[[\partial \underline{G} / \partial \underline{y}] - (1/h\beta_o) [\partial \underline{G} / \partial \dot{\underline{y}}] \right]_j \underline{\Delta y}_j = -\underline{G}_j \quad (10)$$

This whole process corresponds to a predictor-corrector solution scheme that can be summarized as follows:

Predict

1. Predict values for \underline{y} and $\dot{\underline{y}}$ that lie on a polynomial passing through past state variable values.

Correct

2. Evaluate \underline{G} . If zero, the solution is acceptable and the corrector is unnecessary.
3. Evaluate the Jacobian matrix defined as $\left[[\partial \underline{G} / \partial \underline{y}] - (1/h\beta_o) [\partial \underline{G} / \partial \dot{\underline{y}}] \right]_j$

4. Factorize the Jacobian. (In ADAMS, this is performed symbolically for efficiency.)
5. Solve for $\Delta\mathbf{y}_j$. Calculate \mathbf{y}_{j+1} and $\dot{\mathbf{y}}_{j+1}$.
6. Repeat steps 2 through 5 until convergence criteria are satisfied.

Integration Error Control

7. Find best step-size and order for next step.
8. Estimate integration error. If too much, reject last step, reduce h , and go to step 1.
9. If end time has not been reached, start new time step by going back to step 1.

These steps describe a general transient dynamic analysis. For a static analysis, the term $(1/h\beta_0)$ can be set to zero to effectively cancel out all time derivatives. In practice, ADAMS sets this term to a very small number to improve calculation of neutrally-stable equilibria.

For kinematics, only the algebraic constraints of the system are solved iteratively to yield a solution.

VI. Development Directions

The field of multibody system analysis is changing rapidly. In the past few years, there has been an exponential growth in the number of research papers and proposals concerning aspects of the field. While this is an exciting trend, it also poses a significant challenge to multibody software developers to stay abreast of the latest technology and to provide users with the most accurate, efficient, and advanced techniques and capabilities. There are still quite a few different schools of thought concerning fundamental aspects of the field, and each provides a unique perspective. To take proper advantage of the wealth of knowledge being accumulated, the developers of ADAMS have not only formed their own in-house research teams, but also established key contacts with leading universities in this field throughout the world. From early exclusive ties with the University of Michigan, the affiliations have grown to encompass universities throughout the United States, Canada, Germany, Switzerland, the Netherlands, and other countries. It is no longer wise or feasible to rely totally on one research group.

The philosophy adopted by ADAMS developers is to continue to aggressively develop software, while at the same time employing individuals to carry out sponsored research in strategic areas of interest. Present areas in which the developers of ADAMS are either performing or sponsoring research include human body modeling and ergonomics, large-displacement elastodynamics, symbolic computation in a sparse environment, numerical integration algorithm development, design sensitivity, optimization, controls, and parallel processing. MDI is currently funding multiple research projects at U.S. universities and is, in turn, being funded by government agencies for research in a variety of areas. In 1989, MDI was named as a "premier research company" in Michigan.

Having excellent research contacts and affiliations is necessary, but not sufficient to insure success of the software as a viable product. As mentioned earlier, MDI is making significant strides in interweaving the ADAMS software with other MCAE tools to form a seamless computing environment. This is crucial to promote widespread use of multibody system analysis software. Graphics processing of input and output remains a focal point of development in this field as engineers attempt to make the technology easier to use and understand.

In addition to research and interface strategies, an advanced development environment provides the needed link to guarantee quality software. The ADAMS software is one of the few systems that is being developed within a structured computer science environment wherein computer aided software engineering (CASE) tools are employed. ADAMS software developers prepare problem statements, write specifications, review designs, and produce code using both C and FORTRAN languages. The code is continually subjected to the following types of tests: accuracy verification, functional, boundary, biased random, stress, and regression. These tests are developed from the code specifications.

Over thirty full-time people are employed in the product development and maintenance of ADAMS. Their educational backgrounds range from B.S. to Ph.D. in fields such as computer science, mathematics, physics, aerospace engineering, and mechanical engineering. Many of the actual products currently under development are described briefly in section two.

VII. Test Example Problems (Verification)

This section serves two purposes, namely (1) to provide a glimpse of typical input, output, and performance for benchmark problems supplied in conjunction with Springer-Verlag's Multibody Systems Handbook, edited by W. Schiehlen, and (2) to present more comprehensive validation and performance of the ADAMS software.

Because of its longevity, ADAMS has been tested extensively in a variety of different ways. Over one hundred man-years have been expended by external research groups as well as the developers in validating the results of the code through (1) correlation with experiment (see Table 5), (2) comparison with other nonlinear finite element codes (NL-FEA), (3) correlation with other multibody and special-purpose programs, and (4) comparison with closed form solutions. Currently the ADAMS test library contains over five hundred of these accuracy verification tests for which complete solutions are known. Table 5 lists some of the published correlations.

Table 5. References of Published Accuracy Verifications of ADAMS

Correlation	ADAMS Validations		
With:	Automotive Systems	Aerospace Systems	Off-Highway Systems
<u>Experiment</u>	Orlandea[8]	Roux[15]	McConville[20]
<u>NL-FEA</u>	Zamow[12]/ABAQUS	Ryan[27], Simo/FEAP Elliot[29]/GRASP	
<u>Other Codes</u>	Giannopoulos[9]/IMP	Ryan[27]/Vigneron[45]	

To demonstrate the range of ADAMS, four benchmark problems will be presented. The first two were provided as test problems for a large suite of multibody software programs, while the last two exhibit the strength of ADAMS.

Figure 18 depicts a planar seven-link closed-loop mechanism with one driven link. The driving torque is shown on the figure, and all system properties are provided in the handbook mentioned earlier. Plotted time histories are included at the top of the figure, and superimposed static images of the system at various instants in time are shown in Figure 20. This model tests the ability of a program to easily model closed loops. The model was built and results obtained within two hours. More details of the simulation are included in Figure 18.

Figure 19 illustrates a spatial robot manipulator driven by joint motors as shown. This is an open-loop system involving three-dimensional dynamic effects. Time history plots are shown in the figure and superimposed images are illustrated in Figure 20. This was a routine problem for ADAMS to solve. Table 6 contains the entire ADAMS input model description.

Figure 21 displays a much more complicated three-dimensional multibody system representing a full automotive vehicle. The system has fifty-four degrees of freedom. Results from ADAMS were compared directly with experimental results for a ramp steer input of 210-degree occurring at forty-five miles per hour^[30]. As can be seen from the inset plots, excellent correlation was achieved.

Lastly, Figure 22 shows a Japanese bullet train traversing a bridge. An ADAMS model of the train/bridge assembly contained over ten thousand equations and more than five hundred degrees of freedom. The bridge was modeled completely with elastic structural elements in ADAMS and all nonlinear effects were captured. A time history plot of the acceleration of a point on the bridge during a train traversal is shown versus time in the inset of the figure.

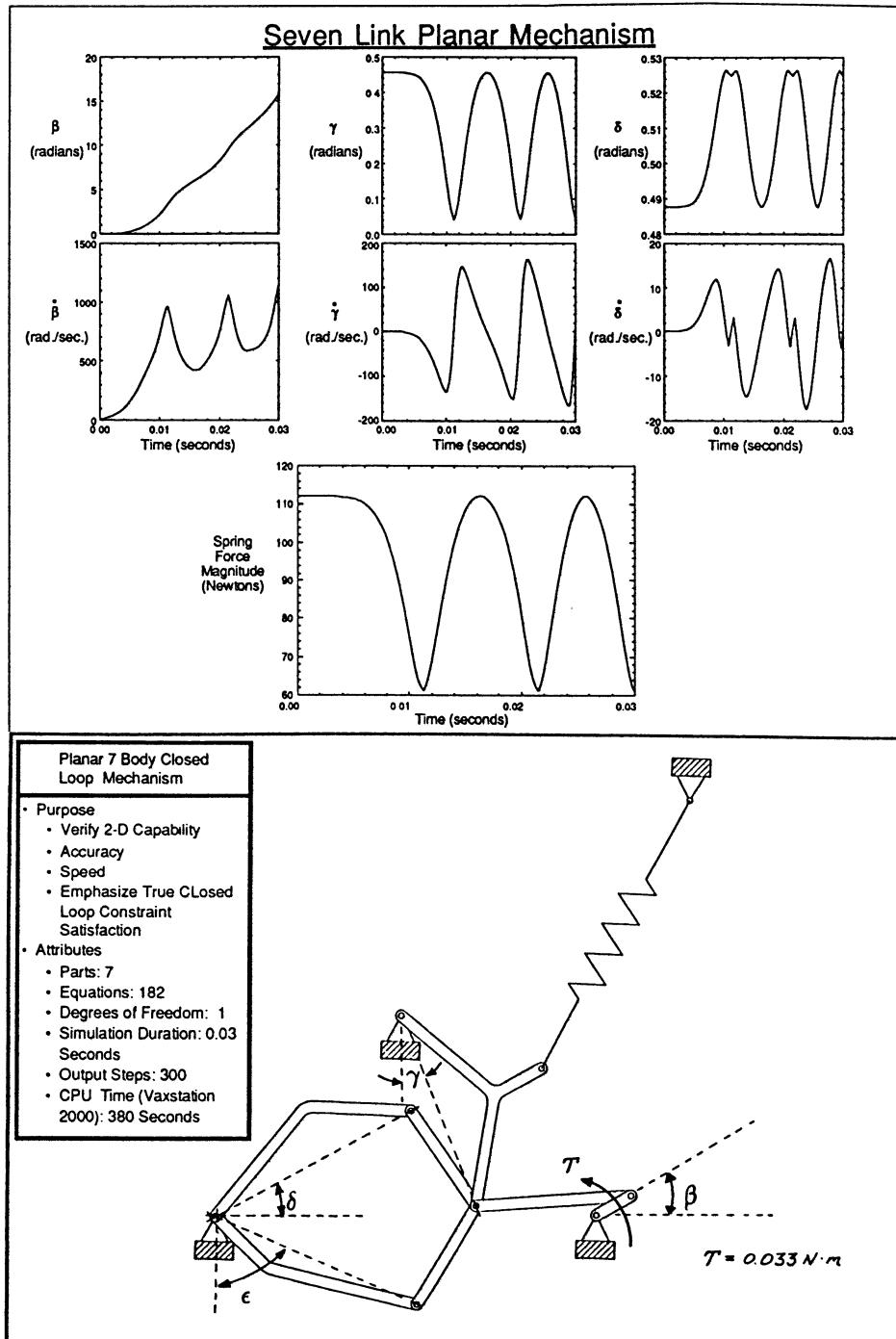


Fig. 18. Seven-Link Planar Mechanism Benchmark Problem

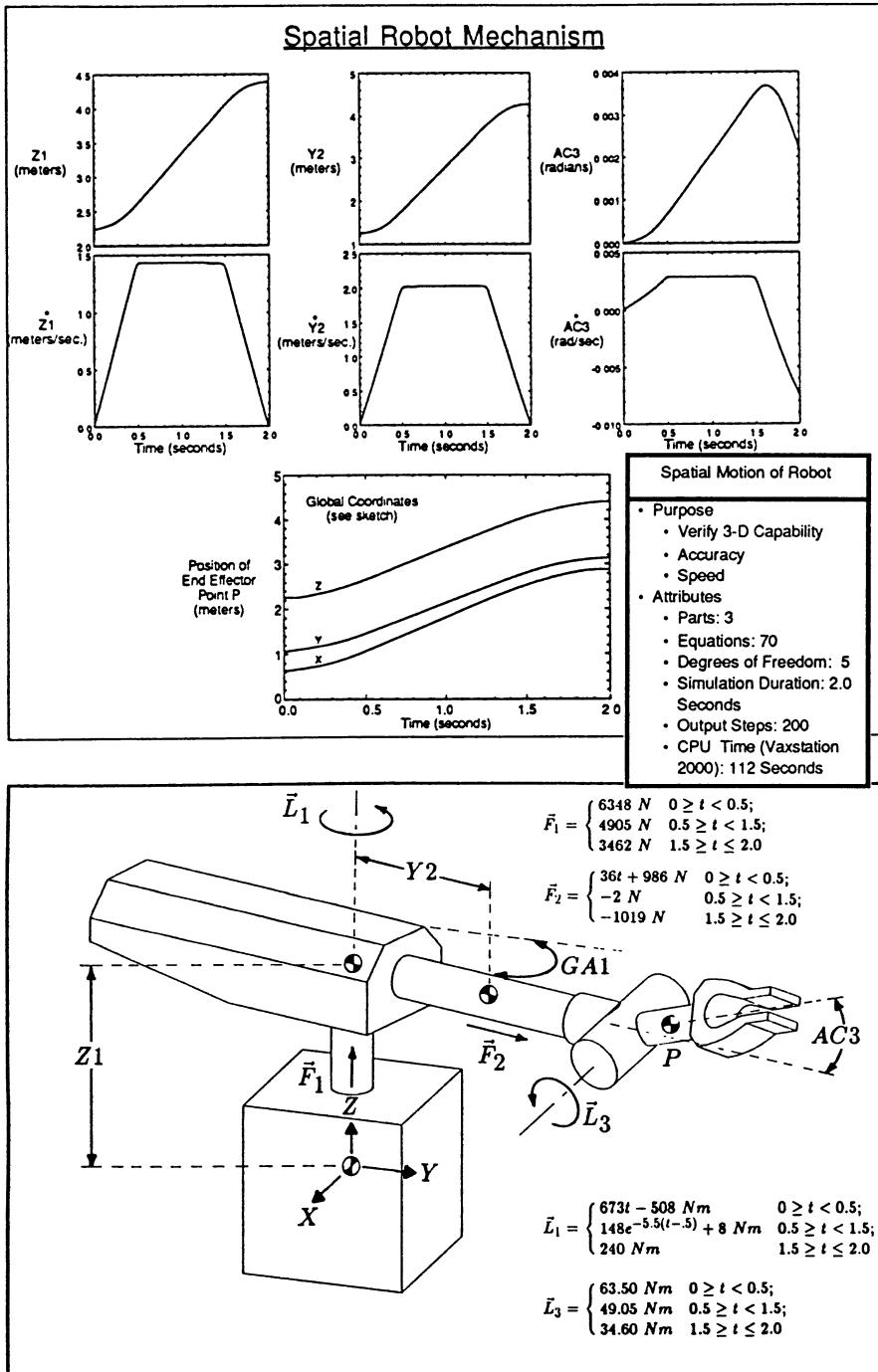


Fig. 19. Spatial Robot Manipulator Benchmark Problem

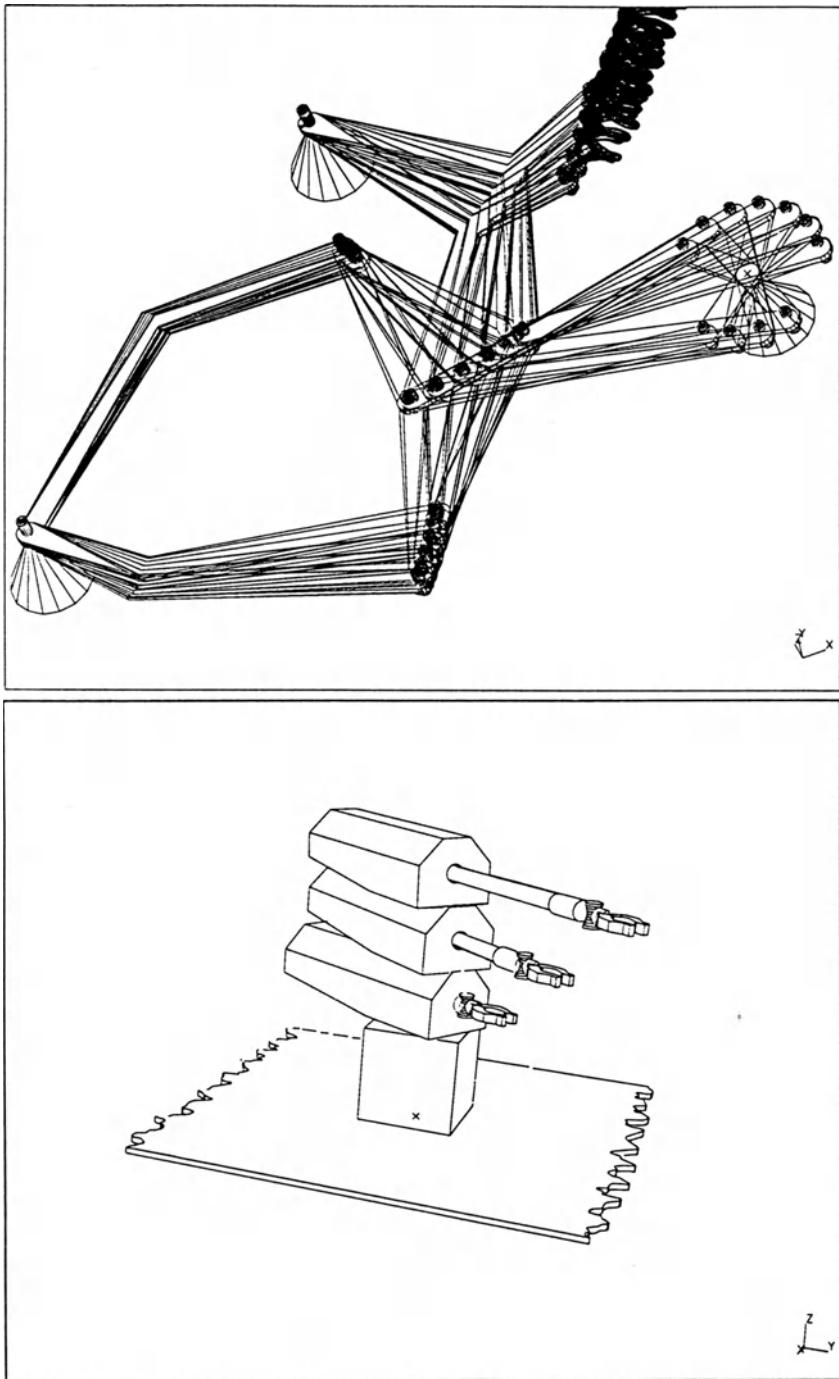


Fig. 20. Superimposed Images of Benchmark Problems

Table 6. Input ADL file for Robot Benchmark

```

FIVE DEGREE OF FREEDOM ROBOT

! [ UNITS - KG, M, N, SEC ]
PARTS
PART/99, GROUND
MARKER/9900, REULER = 0.5236, 0.0, 0.0

PART/01, MASS = 250.0, CM = 0100, IP = 90.0, 10.0, 90.0
, QG= 0.0, 0.0, 2.25
MARKER/0100

PART/02, MASS = 150.0, CM = 0200, IP = 13.0, 0.75, 13.0
, QG= 0.0, 0.75, 2.25
MARKER/0200

PART/03, MASS = 100.0, CM = 0300, IP = 4.0, 1.0, 4.3
, QG= 0.0, 1.30, 2.25
MARKER/0300

PROGRAM CONTROL

ATTACHMENTS
JOINT/0199, I = 0101, J = 9901, CYLINDRICAL
MARKER/0101, PART = 01
MARKER/9901, PART = 99, QP = 0.0, 0.0, 2.25

END

JOINT/0201, I = 0202, J = 0102, CYLINDRICAL
MARKER/0202, PART = 02, QP = 0.0, -0.75, 0.0, REULER = 0, -90D, 0
MARKER/0102, PART = 01, REULER = 0, -90D, 0

JOINT/0302, I = 0303, J = 0203, REVOLUTE
MARKER/0303, PART = 03, QP = 0.0, -0.05, 0.0, REULER = 90D, 90D, 0
MARKER/0203, PART = 02, QP = 0.0, 0.50, 0.0, REULER = 90D, 90D, 0

DRIVERS

SFORCE/01, I = 0104, J=9904, TRANSLATION
, FUNCTION = 6348.0 * IF( ( TIME - 0.5 ) : 1, 1, 0 )
, + 4905.0 * IF( ( TIME - 0.5 ) : 0, 0, 1 )
, * IF( ( TIME - 1.5 ) : 1, 1, 0 )
, + 3462.0 * IF( ( TIME - 1.5 ) : 0, 0, 1 )
MARKER/0104, PART = 01
MARKER/9904, PART = 99

SFORCE/02, I = 0105, J=9905, ROTATION
, FUNCTION = POLY( TIME, 0.0, -508.0, 673.0 )
, * IF( ( TIME - 0.5 ) : 1, 1, 0 )
, + ( 148.0 * EXP( POLY( TIME, 0.5, 0.0, -5.5 ) ) + 8.0 )
, * IF( ( TIME - 0.5 ) : 0, 0, 1 )
, * IF( ( TIME - 1.5 ) : 1, 1, 0 )
, + 240.0 * IF( ( TIME - 1.5 ) : 0, 0, 1 )
MARKER/0105, PART = 01
MARKER/9905, PART = 99, QP = 0.0, 0.0, 2.25

SFORCE/03, I = 0206, J=0106, TRANSLATION
, FUNCTION = POLY( TIME, 0.0, 986.0, 36.0 )
, * IF( ( TIME - 0.5 ) : 1, 1, 0 )
, + (-2.0) * IF( ( TIME - 0.5 ) : 0, 0, 1 )
, * IF( ( TIME - 1.5 ) : 1, 1, 0 )
, + (-1019.0) * IF( ( TIME - 1.5 ) : 0, 0, 1 )
MARKER/0206, PART = 02, QP = 0.0, -0.75, 0.0, REULER = 0, -90D, 0
MARKER/0106, PART = 01, QP = 0.0, -10.0, 0.0, REULER = 0, -90D, 0

SFORCE/04, I = 0307, J=0207, ROTATION
, FUNCTION = 63.5 * IF( ( TIME - 0.5 ) : 1, 1, 0 )
, + 49.05 * IF( ( TIME - 0.5 ) : 0, 0, 1 )
, * IF( ( TIME - 1.5 ) : 1, 1, 0 )
, + 34.6 * IF( ( TIME - 1.5 ) : 0, 0, 1 )
MARKER/0307, PART = 03, QP = 0.0, -0.05, 0.0, REULER = 90D, 90D, 0
MARKER/0207, PART = 02, QP = 0.0, 0.50, 0.0, REULER = 90D, 90D, 0

REQUESTS

```

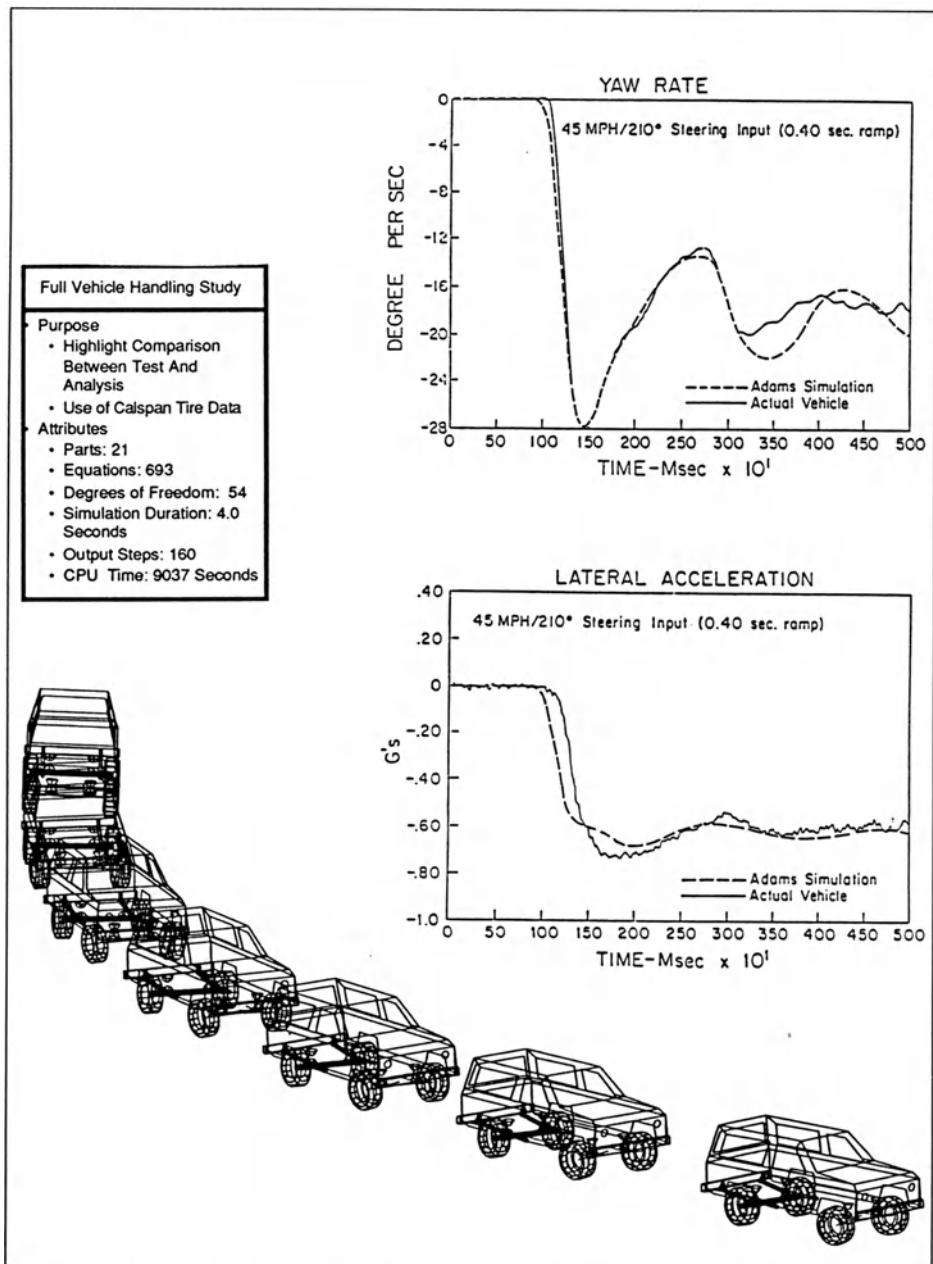


Fig. 21. Ford Bronco II Full Vehicle Simulation Benchmark

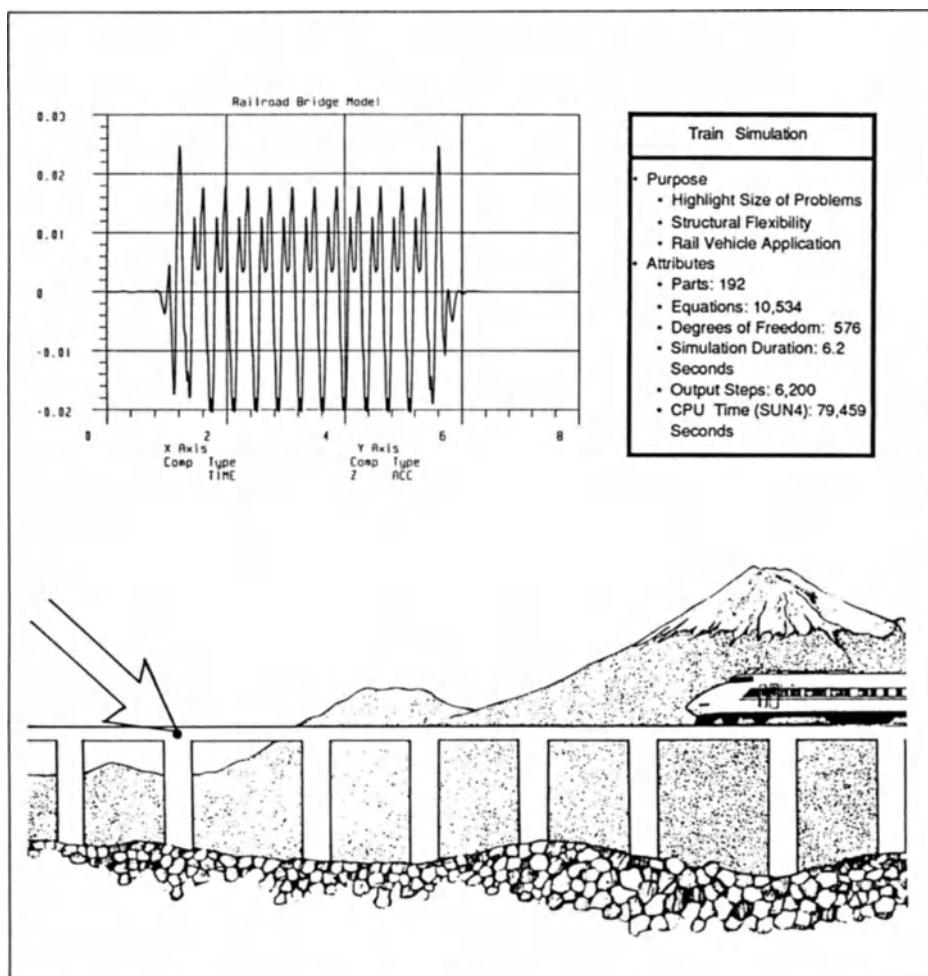


Fig. 22. Railway Benchmark Problem

The sheer size of the rail problem, not to mention the extreme elasticity, would prevent a user from obtaining simulation results with many of the competing multibody software programs.

The models included here are intended to give the reader a feeling for the generality and versatility of the ADAMS software when applied to mechanical multibody systems.

VIII. Acknowledgements

A number of people deserve credit for their contribution to the creation of this ADAMS overview. In particular, out of great appreciation for their significant efforts, I would like to thank the following individuals: [from MDI] David Andrews, Itshak Barkon, Carol Evert, Jeanne Kelley, Jim Molnar, Jim McConville, Sam McDonald, Rajiv Rampalli, Chuck Wilson, Michael Young; [from Schlumberger] Patrick Turner; [from SDRC] Matthew Peffley; and [from TEDAS] Michael Bartels and Ewald Fischer.

Also, I would like to express my appreciation to the thousands of engineers and designers who have used ADAMS to aid them in their work. By their ingenious use of the software and their continual feedback, they have helped improve the quality of the code immeasurably and have provided the raw data and graphics that form the basis of much of this paper.

Lastly, a special thanks goes out to MDI for providing an environment for multibody system analysis to flourish and grow in its applicability. If only Newton could have seen ADAMS!

IX. References

- [1] Hooker, W. W., and Margulies, G., "The Dynamical Attitude Equations for an n-Body Satellite." *The Journal of Astronautical Sciences*. Vol. XII, No. 4, 1965, pp. 123-128.
- [2] Roberson, R. E., and Wittenburg, J.. "A Dynamical Formalism for an Arbitrary Number of Inter-connected Rigid Bodies. with Reference to the Problem of Satellite Attitude Control." *Proceedings 3rd International Congress on Automatic Control* (London, 1966) London, 1967. pp. 46D.1-46D.8.
- [3] Cooper, D.W., Bitonti, F., Frayne, D.N., and Hansen, H.H., "Kinematic Analysis Method (KAM)." Society of Automotive Engineers, Paper No. SP-272, May 1965.
- [4] Knappe, L.F., "A Computer-Oriented Mechanical Design System," ASME Paper No. 64-MECH-30, *Mechanical Engineering*. Vol. 87, No. 5. pp. 35-40, May 1965.
- [5] Chace, M.A., and Korybalski, M. E.. "Computer Graphics in the Schematic Representation of Nonlinear, Constrained, Multifreedom Mechanical Systems," Computer Graphics 70 Conference, Brunel University, April 1970.
- [6] Orlandea, N., Chace, M.A., Calahan, D.A., "A Sparsity-Oriented Approach to the Dynamic Analysis and Design of Mechanical Systems - Parts 1 and 2," *ASME Journal of Engineering for Industry*, Paper No. 76-DET-19, ASME Mechanisms Conference, Montreal, Canada, Oct. 1976.
- [7] Rampalli, R., "ADAMS - A Sparse Matrix Approach to Solving Multibody Dynamics Problems," *Proceedings of the SDIO/NASA Workshop on Multibody Simulation*, Jet Propulsion Laboratory, Pasadena, California, Sept. 1-3, 1987.
- [8] Orlandea, N., and Chace, M., "Simulation of a Vehicle Suspension with the ADAMS Computer Program," SAE Paper No. 770053, *Proceedings of the 1977 International Automotive Engineering Congress and Exposition*, Detroit, Michigan, Feb. 28 - Mar. 4, 1977.
- [9] Giannopoulos, F., and Rao, A.K., "Dynamic Loads on Suspension Components Using Mechanisms Programs," SAE Paper 811307, 1981.

- [10] Antoun, R.J., Hackert, P.B., O'Leary, M.C., Sitchin, A., "Vehicle Dynamic Handling Computer Simulation – Model Development, Correlation, and Application Using ADAMS," SAE Technical Paper No. 860574, *International Congress and Exposition*, Detroit, Michigan, Feb. 24-28, 1986.
- [11] Agelidis, N., "3-D Crash Analysis Using ADAMS," Ford Motor Company Report, SAE Paper No. SAE 885076, *Proceedings of FISITA*. Detroit, Michigan, Sept. 1988.
- [12] Zamow, J., "Use of the Rigid Body Program ADAMS at PORSCHE," *Proceedings of the 1989 Far East ADAMS Users' Conference*, Tokyo, Japan, Mar. 7, 1989.
- [13] McConvilie, J.B., "Aircraft Stability and Control Using ADAMS," *Proceedings of the 1988 International ADAMS Users' Conference*, Mechanical Dynamics, Inc., Ann Arbor, Michigan, Sept. 13-15, 1988.
- [14] Sukarie, G., "Influence of the Flexibility of the Landing Gears on the Impact Forces of an Aeroplane Under Ultimate Landing Conditions," *Proceedings of the 5th European ADAMS News Conference*, TEDAS GmbH, Marburg, FR Germany, Oct. 4, 1988.
- [15] Roux, C., and Flament, P., "Solar Array Deployment Simulation Using ADAMS Software," *Proceedings of the Second European Space Mechanisms & Tribology Symposium*, Meersburg, FR Germany, Oct. 9-11, 1985.
- [16] Sohoni, V.N., and Chace, M.A., "Pseudo-Prototyping of Aerospace Mechanical Dynamic Systems with a Generalized Computer Program." *Proceedings of the 20th Aerospace Mechanisms Symposium*, NASA Lewis Research Center, Cleveland, Ohio, May 7-9, 1986.
- [17] Brazzini, G., Broustet, Y., Garnier, C., Picard, P., "Kinematic Analysis of a Large Deployable Truss Antenna (Using ADAMS)." *Aerospatiale Paper*. No. IAF 85-74. 1985.
- [18] Panin, F., "Modeling Flexibility of (Aerospace) Mechanisms with ADAMS," European Scientific and Technical Memorandum No. ESA STM-242, European Space Agency. Noordwijk, The Netherlands, July, 1988.
- [19] Smith, D.W., Light, R.A., Romig, B.E., Berenyi, T.A., Orlandea, N.V., Wiley, J.C., Portillo, N., O'Brien, S.E., Hertema, D.J., "Automated Simulation and Display of Mechanism and Vehicle Dynamics," ASAE Paper No. 82-5019, *1982 ASAE Summer Meeting*, Madison, Wisconsin, June 27-30, 1982.
- [20] McConvilie, J.B., Angell, J. C., "The Dynamic Simulation of a Moving Vehicle Subject to Transient Steering Inputs Using the ADAMS Computer Program," ASME Paper No. 84-DET-2, 1984.
- [21] DeFouw, J.A., and Dowell, W.R., "Analysis of a Seating System Using ADAMS and the Anthropoid Preprocessor," *Proceedings of the 1988 International ADAMS Users' Conference*, Mechanical Dynamics, Inc., Ann Arbor, Michigan, Sept. 13-15, 1988.
- [22] Anon., *ADAMS User's Guide*. Mechanical Dynamics. Inc.. 3055 Plymouth Rd., Ann Arbor, Michigan, 1987.
- [23] Duff, I.S., and Reid, J.K., "A Comparison of Some Methods for the Solution of Sparse Overdetermined Systems of Linear Equations," *J. Inst. Math. Appl.*, Vol. 17, pp. 267-280, 1976.
- [24] Gear, C.W., "The Simultaneous Numerical Solution of Differential Algebraic Systems," *IEEE Transactions on Circuit Theory*, Vol. 1, pp. 89-95. 1971.
- [25] Pissanetzky, S., *Sparse Matrix Technology*. Academic Press, London, 1984.
- [26] Chua, L., and Lin, Pen-Min, *Computer-Aided Analysis of Electronic Circuits*, Prentice-Hall, Englewood Cliffs, New Jersey, 1975.
- [27] Ryan, R. R., "Flexible Multibody Dynamics: Problems and Solutions," *Proceedings of the NASA/SDIO Workshop on Multibody Dynamics*, Pasadena, California, Sept. 1-3, 1987.
- [28] Wielenga, T., "Simplifications in the Simulation of Mechanisms Containing Flexible Members," Ph.D. Thesis, University of Michigan, 1984.
- [29] Elliott, A.S., and McConvilie, J.B., "Application of a General-Purpose Mechanical Systems Analysis Code (ADAMS) to Rotorcraft Dynamics Problems," *Proceedings of the 1989 AHS Conference*, Nov. 13-14, 1989.

- [30] Chace, M.A., "Methods and Experience in Computer Aided Design of Large-Displacement Mechanical Systems," *Computer Aided Analysis and Optimization of Mechanical System Dynamics*, NATO ASI Series, Vol. F9, Springer-Verlag Berlin Heidelberg, 1984.
- [31] Sohoni, V.N., and Whitesell, J., "Automatic Linearization of Constrained Dynamical Models," *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, Vol. 108, No. 3, pp. 300-304, Sept. 1986.
- [32] Sohoni, V.N., "Design Optimization of Mechanical Systems," *Proceedings of the 1989 Far East ADAMS Users' Conference*, Tokyo, Japan, Mar. 7, 1989.
- [33] Terlinden, M.W., Langner, W., and Hache, M., "MOGESSA - A Full Vehicle Simulation System Based on ADAMS," *Proceedings of the 4th European ADAMS News Conference*, TEDAS GmbH, Marburg, FR Germany, Sept. 1987.
- [34] Langner, W., "Sensitivity Analysis and Optimization of Mechanical System Design," *Proceedings of the 5th European ADAMS News Conference*, TEDAS GmbH, Marburg, FR Germany, Oct. 4, 1988.
- [35] Steigerwald, M.F., "Implementing Digital Control Algorithms in an ADAMS Model," *Proceedings of the 1988 International ADAMS Users' Conference*, Mechanical Dynamics, Inc., Ann Arbor, Michigan, Sept. 13-15, 1988.
- [36] Turner, P.R., "Workstation Integration for Large Displacement Mechanical Analysis," *U.M. Conference on Computer Aided Engineering of Vehicles and Machinery*, Ann Arbor, Michigan, July 25-29, 1988.
- [37] Hudi, J., "AMIGO - A Modular System for Generating ADAMS Models," *Proceedings of the 5th European ADAMS News Conference*, TEDAS GmbH, Marburg, FR Germany, Oct. 4, 1988.
- [38] McConville, J.B., Steigerwald, M.F., "The Use of ADAMS to Create Realistic Boundary Conditions for Finite Element Evaluation of Mechanisms - The ADAMS/ANSYS Interface," *Proceedings of the ANSYS Users' Conference*, Pittsburgh, Pennsylvania, April 23-25, 1985.
- [39] Sohoni, V., Winkelmann, J., "Control Systems Analysis Using ADAMS and MATRIXx," *Proceedings of the 1988 International ADAMS Users' Conference*, Mechanical Dynamics, Inc., Ann Arbor, Michigan, Sept. 13-15, 1988.
- [40] Turner, P.R., and Bodner, M.E., "Optimization and Synthesis for Mechanism Design," Paper No. MS88-711, *Proceedings of the Society of Manufacturing Engineers AUTOFAC'T '88 Conference and Exposition*, Chicago, Illinois, Oct. 1988.
- [41] Ryan, R.R., and Steigerwald, M.F., "ADAMS: Theory and Application," *Proceedings of the 1988 ADAMS Software Theory Seminar*, Mechanical Dynamics, Inc., Ann Arbor, Michigan, 1988.
- [42] Wielenga, T.W., "Analysis Methods and Model Representation in ADAMS," MDI Technical Paper No. 41, 1987.
- [43] Kane, T. R., and Levinson, D. A., "Multibody Dynamics," *Journal of Applied Mechanics*, Vol. 50, Dec. 1983, pp. 1071-1078.
- [44] Kane, T. R., and Levinson, D. A., *Dynamics, Theory and Applications*, McGraw-Hill Book Co., New York, N.Y., 1985, p. 24.
- [45] Lips, K. W., Graham, W. B., Vigneron, F. R., and Hunter, D. G., "Dynamics and Control Characteristics for the WISP 300m Dipole Antenna/Shuttle Configuration," *Advances in the Astronautical Sciences*, Vol. 58, No. 1, 1985, pp. 349-369.

PLEXUS - Software for the Numerical Analysis of the Dynamical Behavior of Rigid and Flexible Mechanisms

by A. BARRACO, B. CUNY, E.N.S.A.M. - Laboratoire de Robotique de Paris
and : A. HOFFMANN, P. JAMET, A. COMBESCURE, M. LEPAREUX, H. BUNG,
L.A.M.S. / D.E.M.T. / S.M.T.S. - C.E.A. - Centre d'Etudes Nucléaires de Saclay

0 - Introduction to the mechanical analysis of multibody systems

In most computer codes developed for the dynamic analysis of *mechanisms* , such as the serial manipulators used in Robotics, differential equations of motion are derived from the hypothesis that the links are rigid while flexibilities, if any, are located at the joints. This simplification is widely used since it correctly models the sturdy members found in present robots while limiting the number of unknown parameters and of the related kinematical and dynamical equations. Indeed, a rigid body shows six degrees of freedom, and any connection between two adjacent members removes up to five possibilities of relative motion : the total number of independent position parameters of any complex mechanism remains small, and elaborating the *kinematical models* for the end effector is then a straightforward procedure (the geometrical model relates its position and orientation to the joint parameters, while the differential model expresses its linear and angular velocities in terms of the time derivatives of joint parameters) , [1] .

Concerning the *dynamical model* of such manipulators, several formulations are available which all lead, in a more or less systematic way, to the same set of coupled non-linear differential equations : the Newton-Euler approach provides a dynamic balance of forces and torques, including inertia effects, for each arm link considered separately, while the Lagrangian formulation, a special form of the Principal of Virtual Work, describes the system's dynamics in terms of work and energy . Theses equations relate applied forces and moments to the second time derivatives of the linear and angular absolute positions of rigid bodies, [2], [3], [4], and lead to two opposite uses : for given input variations of driving forces and moments, joint accelerations are computed with the *direct* model to yield, after time integration, joint motions

and end effector trajectory and velocity changes; conversely, in order to achieve prescribed tasks specified as end effector motions, driving efforts are needed to overcome gravity and inertia actions : their values are provided by the *inverse* model.

While these rigid body codes are today used to dimension actuators and structural links, they show two limitations to achieve realistic dynamical analyses of present and future manipulators :

- a) when the flexibility of some members is important, the rigid body hypothesis is no longer valid since the kinematics and dynamics of the system are deeply influenced by *déformation* , [5] . Since this feature can no longer be ignored, its control at both simulation and execution levels needs be investigated, for instance on highly flexible manipulators in research labs. While the accurate positioning precision of today's industrial robots mainly results from their overdimensioned member stiffness and is paid for by a very small payload to dead weight ratio, the tendency to increase this ratio and to reduce the amount of energy used to merely move the structure will result in lighter and more flexible robot arms. Appropriate simulation tools are thus needed to foresee the influence of deformation-induced phenomena on the system's dynamics and to master them.
- b) the actual driving efforts applied to manipulator joints do not simply reproduce the variations of forces and moments obtained in a previous dynamical analysis simulation of the same motions : these computed values indeed show errors due to badly modeled parameters and phenomena, (wrong dimensions and inertia values, ignored dry friction, omitted motor and gear dynamics, ...) , so that the motion obtained from their strict enforcement on the structure would be fairly different from that expected. Automatism is instead used to compensate these deficiencies : the real efforts are generated by continuously checking how far away the real behavior is from the desired one : sensors deliver real time data on positions, velocities, ..., and the error signals are supplied to a feedback controller that monitors actuator inputs in such a way as to optimize trajectory tracking , [6] , [7] . In this closed loop control process, the variations of the driving efforts and of the induced end effector motions highly depend on the gain values : simulation capabilities become compulsory to analyze these major aspects of robot dynamics .

While we simultaneously developed a rigid body analysis code based on the the Principle of Virtual Work and could have enlarged it with some account of deformation (mode superposition, simple finite element model), we decided that the best way to meet the above double challenge was to complete an existing structural analysis finite element code, called PLEXUS, with mechanism features such as the possibility of relative motions at substructure joints. This program has been developped and used since 1977 at the Mechanical and Thermal Studies Department of the Nuclear Energy Agency center in Saclay, France. This code is integrated in a software environment called CASTEM 2000 , which provides it with pre- and

post-processors making it easier first to input data (free format keyword instructions to define geometry, properties, boundary conditions, loading cases, and mesh visualization), and then to analyze simulation results on paper drawings or color screen graphics (displaced and deformed structure, space distribution of variables, time variation diagrams of any quantity).

PLEXUS was originally designed to solve fast dynamics problems leading to structural ruin, such as projectile impacts, container or pipe explosions, for which mesh updating was necessary. Any law of material behavior can be imposed to fluids and solids, either usual or from experimental origin, [8], [9]. Full size experiments were achieved on real structures to check PLEXUS numerical results and validate this code, [10]. The possibility to impose relations between nodal degrees of freedom, and thereby ensure the partial or total connection of distinct substructures, appeared as a favorable base for extending the code potentialities to the dynamical analysis of any spatial articulated mechanical system, and especially manipulators used in Robotics. PLEXUS makes it now possible to study them as if their component parts were rigid, but further to analyze how the input of flexibility in some parts changes the system's dynamics, and to check how controller parameters interfere with its response.

After the first section, devoted to the theoretical background of the code, the second part presents the developments specifically made for robotics applications, and is followed by the specification of program structure and syntax rules.

1 - PLEXUS implementation of basic continuum mechanics concepts

1.1 - Configurations of a deformable body

To analyze the evolutions of a deformable body \mathcal{D} with time, [2], [5], we designate by :

- \mathcal{D}_0 the domain which it occupies at time t_0 , arbitrarily chosen as the initial instant and designed to serve as reference for the calculations at later times,
- \mathcal{D}_τ and \mathcal{D}_t the two configurations reached at consecutive times τ et $t = \tau + \Delta t$,
- $\mathcal{R} = (\vec{O}, \overset{\rightarrow}{e_i})$ a fixed orthonormal frame used to decompose vectors and tensors.

A given point M_0 (resp. curve Λ_0 , surface Γ_0 , or volume Ω_0) of the \mathcal{D}_0 domain is transformed at time t into a homologous point M_t (resp. curve Λ_t , surface Γ_t , or volume Ω_t) in \mathcal{D}_t . The positions of M_0 and M_t in \mathcal{R} are given by the column matrices of their coordinates :

$$\overset{\rightarrow}{OM_0} = \{a\} . \overset{T}{\rightarrow} \{e\} = a_i . \overset{\rightarrow}{e_i} \quad & \quad \overset{\rightarrow}{OM_t} = \{x\} . \overset{T}{\rightarrow} \{e\} = x_i . \overset{\rightarrow}{e_i}$$

expression in which the Einstein implicit summation convention on repeated indices is used.

The transformation from M_0 to M_t of a given point M in the domain \mathcal{D} is expressed by vectorial

bijections \vec{f} and \vec{g} , inverse of each other and defined by their components as :

$$\{x\} = \{ f(\{a\}, t_0, t) \} \quad \{a\} = \{ g(\{x\}, t, t_0) \} \quad (1)$$

Since each point M in \mathcal{D} is continuously transformed, two approaches are available to express physical quantities or mechanical laws in this point :

- * the *material* standpoint : it consists in following each particle M along its trajectory, from its initial reference position M_a at t_0 , with components $\{a\}$, and to express any quantity or relation $G(\{a\}, t)$ as a function of the 'material' *Lagrange variables*, $\{a\}$ and t .
- * the *geometrical* standpoint : it considers that different particles successively go through any given point P in space, defined by its components $\{x\}$ in \mathcal{R} , to express any quantity or relation $G(\{x\}, t)$ at P in terms of its 'geometrical' *Euler variables*, $\{x\}$ and t .

Let $\{x\}$ be the position at time t of the particle M which started in $\{a\}$ at t_0 : the two expressions G and G are obviously identical, i.e. : $G(\{a\}, t) = G(\{f(\{a\}, t_0, t)\}, t)$ (2)
a relation satisfied *at all t* if the geometrical point P is moved along with particle M.

In a Total Lagrangian Model, the reference configuration \mathcal{D}_0 is kept the same during the whole evolution : such a description is well suited for statical and vibratory analyses of solids, for which position differences between the initial and deformed configuration remain small. If the body undergoes large motions, the reference position can be changed : several successive configurations \mathcal{D}_a are then substituted to the previous unique \mathcal{D}_0 and give rise to an Updated Lagrangian Model. This is the approach used in PLEXUS, [8], [9], [10], for the dynamical analysis of mechanisms : updating occurs at the end of each time step by taking as the new reference configuration at time t the one actually reached at the previous instant $\tau = t - \Delta t$. Finally, in the Eulerian description, the present configuration at time t is chosen as reference, without consideration to what previously happened : although this is the only model in which present quantities are introduced 'naturally', its application is limited to fluids and avoided for solids. Nevertheless, since the real configuration at time t and its related quantities remain of utmost importance, transportation relations from and to the reference will be defined. In the following, the ' a ' subscript is uniformly attached to the reference configuration (either initial, i.e. at t_0 , or updated, at τ) and to the quantities and relations there defined, whereas the ' t ' subscript for the present configuration and related quantities is omitted for brevity.

The vectorial difference : $\vec{\Delta u} = \vec{OM} - \vec{OM}_a$ between positions of homologous points in the present and reference configurations thus stands for either the whole displacement in the Total Lagrangian description ($\mathcal{D}_a = \mathcal{D}_0$), or the displacement increment between two consecutive instants τ and $t = \tau + \Delta t$ in the Updated Lagrangian description ($\mathcal{D}_a = \mathcal{D}_\tau$).

1.2 - Stress vector definition

Internal forces arise at time t in any point M of the moving deformable body : they represent

interactions between particles located in the vicinity of M and are expressed as the ratio between an elementary force $\vec{T} \cdot d\Gamma$ and the infinitely small surface $d\Gamma$ on which it acts. This vector is called the *stress vector* and varies both with the location of M inside the body and, for a given M, with the orientation of the normal vector n pointing away from that part of the body on which the stress is computed.

The state of stress at point M, ie set of all stress vectors $\vec{T}(M, n)$ for all directions n , is completely specified by knowing the symmetrical *stress tensor* at this point, tensor explicitly defined by its six distinct components in the reference frame \mathcal{R} in two possible ways, according again to which configuration is used as a reference :

- in the Eulerian description of the movement, the present configuration \mathcal{D} is taken as the reference : the state of stress in \mathcal{D} at time t is given by the Cauchy stress tensor Σ , with components σ_{ij} ; the resulting force on a given surface Γ in \mathcal{D} is thus integrated as :

$$\vec{F}(t, \mathcal{D}) = \int_{\Gamma} \vec{\Sigma} \cdot \vec{n} \cdot d\Gamma = \int_{\Gamma} T_i \cdot e_i \cdot d\Gamma = \int_{\Gamma} \sigma_{ij} \cdot n_j \cdot e_i \cdot d\Gamma \quad (3)$$

- in the *Lagrangian* description of the movement, the reference configuration \mathcal{D}_a is one which occurred before the present configuration \mathcal{D} : the state of stress in \mathcal{D} at time t is expressed in \mathcal{D}_a by the *Piola-Kirchhoff stress tensor* Π , with components $\pi_{\alpha\beta}$.

The stress integral at time t on a given surface Γ in \mathcal{D} is computed in two steps using the gradient tensor \mathcal{F} of the vectorial transformation f in M_a which relates homologous vectors in the reference and present configurations :

$$\vec{F}(t, \mathcal{D}) = \vec{\mathcal{F}} \cdot \vec{F}(t, \mathcal{D}_a) \quad (4)$$

$$\vec{F}(t, \mathcal{D}_a) = \int_{\Gamma_a} \vec{\Pi} \cdot \vec{n}_a \cdot d\Gamma_a = \int_{\Gamma_a} T_{a\alpha} \cdot \vec{e}_\alpha \cdot d\Gamma_a = \int_{\Gamma_a} \pi_{\alpha\beta} \cdot n_{a\beta} \cdot \vec{e}_\alpha \cdot d\Gamma_a \quad (4')$$

$$\Rightarrow \vec{F}(t, \mathcal{D}) = \int_{\Gamma_a} \pi_{\alpha\beta} \cdot n_{a\beta} \cdot \vec{\mathcal{F}} \cdot \vec{e}_\alpha \cdot d\Gamma_a = \int_{\Gamma_a} \pi_{\alpha\beta} \cdot n_{a\beta} \cdot \vec{E}_\alpha \cdot d\Gamma_a \quad (4'')$$

where the E_α vectors are computed using the components $F_{ia} = \partial x_i / \partial a_\alpha$ of the gradient tensor matrix $[F]$. It is well known that the components of the two stress tensors are related by:

$$\det(\mathcal{F}) \cdot \sigma_{ij} = F_{ia} \cdot F_{jb} \cdot \pi_{ab} = \partial x_i / \partial a_\alpha \cdot \partial x_j / \partial a_\beta \cdot \pi_{ab} \quad (5)$$

with : $\det(\mathcal{F}) = \rho_a / \rho$ ratio of mass densities at points M_a in \mathcal{D}_a and M in \mathcal{D} .

1.3 - Dynamics equations

Two sets of equations need then be written for a moving deformable body : first the *equations of motion*, for points located inside the volume under study, and secondly *boundary conditions* at points on the volume surface where forces or displacements are prescribed. For the two types of reference configurations described above, volume and surface forces are then integrated as :

- gravity body forces :

$$\vec{f} = \int_{\Omega_a} \rho_a \cdot g_\alpha \cdot \vec{e}_\alpha \cdot d\Omega_a = \int_{\Omega} \rho \cdot g_i \cdot \vec{e}_i \cdot d\Omega \quad (6)$$

- boundary forces on that part Γ_F of the surface Γ of \mathcal{D} where forces are applied :

$$\vec{F} = \int_{\Gamma_F} F_{a_\alpha} \cdot \vec{e}_\alpha \cdot d\Gamma_{F_a} = \int_{\Gamma_F} F_i \cdot \vec{e}_i \cdot d\Gamma_F \quad (7)$$

The dynamics equations involving Cauchy stresses are referred to the evolving present configuration \mathcal{D} , which must be known at time t to carry out derivation of stress components σ_{ij} with respect to position parameters x_j . Despite their simplicity due to their linearity :

$$\partial\sigma_{ij}/\partial x_j + \rho \cdot g_i = \rho \cdot \partial^2 x_i / \partial t^2 \quad \text{in } \mathcal{D} \quad ; \quad \sigma_{ij} \cdot n_j = F_i \quad \text{on } \Gamma_F \quad (8)$$

they are often preferred the dynamics equations in terms of Piola-Kirchhoff stresses since these refer to the permanently known 'previous' configuration \mathcal{D}_a . The latter yield the more complex form :

$$\left\{ \begin{array}{ll} \partial(\pi_{\alpha\beta} \cdot \partial x_i / \partial a_\alpha) / \partial a_\beta + \rho_a \cdot g_i = \rho_a \cdot \partial^2 x_i / \partial t^2 & \text{in } \mathcal{D}_a \\ \partial x_i / \partial a_\alpha \cdot \pi_{\alpha\beta} \cdot n_{a_\beta} = F_i \cdot (1 + \Xi) & \text{on } \Gamma_{F_a} \end{array} \right. \quad (9)$$

where Ξ is the change of surface ratio at point M_0 in \mathcal{D}_0 . In these non linear expressions, the derivatives are computed with respect to the position parameters a_α of the known reference configuration \mathcal{D}_a . In an *explicit* numerical integration code, such as PLEXUS, the present configuration \mathcal{D} , i.e. the actual one at time t , is known by the very choice of the integration scheme : equations (8) are therefore used to compute those accelerations that balance internal stresses in the configuration \mathcal{D} itself. This no longer holds for an *explicit* type code.

1.4 - Variational formulation . Principle of Virtual Work (P.V.W.)

The Principle asserts that, at any given time t and for any distribution of virtual displacement δu or virtual velocity v^* imposed to any mechanical system at this instant, the following relation on the developed virtual work or power holds :

$$\delta W_{acc} = \delta W_{int} + \delta W_{ext} \quad \text{or} \quad P^{*acc} = P^{*int} + P^{*ext} \quad (10)$$

From the arbitrary virtual velocity field : $\vec{v}^* = \{v^*\} \cdot \{e\} = v^*_i \cdot e_i^T$, Eulerian since defined on \mathcal{D}_t , components of the rate of change in virtual strain Eulerian tensor are deduced to compute δW_{int} or P^*_{int} due to internal forces : $D^*_{ij} = 1/2 \cdot (\partial v^*_i / \partial x_j + \partial v^*_j / \partial x_i)$ (11)

Equation (10) for the P.V.W. is strictly equivalent to the dynamics equations of the previous section and results from integrating equations (8) or (9), depending on which reference configuration has been chosen, namely :

- with Euler variables, in the present configuration \mathcal{D} in which Cauchy stresses are defined :

$$\int_{\Omega} \rho \cdot \partial^2 x_i / \partial t^2 \cdot v^*_i \cdot d\Omega = - \int_{\Omega} \sigma_{ij} \cdot D^*_{ij} \cdot d\Omega + \int_{\Gamma_F} F_i \cdot v^*_i \cdot d\Gamma_F + \int_{\Omega} \rho \cdot g_i \cdot v^*_i \cdot d\Omega \quad (12)$$

- with Lagrange variables, by transportation to the \mathcal{D}_a reference configuration in which Piola-Kirchhoff stresses are defined :

$$\int_{\Omega_a} \rho_a \cdot \partial^2 x_i / \partial t^2 \cdot v^*_i \cdot d\Omega_a = - \int_{\Omega_a} \pi_{\alpha\beta} \cdot D^*_{\alpha\beta} \cdot d\Omega_a + \int_{\Gamma_{Fa}} F_{ai} \cdot v^*_i \cdot d\Gamma_{Fa} + \int_{\Omega_a} \rho_a \cdot g_i \cdot v^*_i \cdot d\Omega_a \quad (13)$$

the first term on the right hand side of this equation is transformed using, firstly, relation (5) between Cauchy stresses σ_{ij} and Piola-Kirchhoff stresses $\pi_{\alpha\beta}$ and, secondly, the rate of change in virtual strain Lagrangian tensor $D^*_{\alpha\beta}$, defined as the counterpart in \mathcal{D}_a of the Eulerian tensor D^*_{ij} by :

$$D^*_{\alpha\beta} = D^*_{ij} \cdot \frac{\partial x_i}{\partial a_\alpha} \cdot \frac{\partial x_j}{\partial a_\beta} \quad (14)$$

$$- \int_{\Omega_a} \pi_{\alpha\beta} \cdot D^*_{\alpha\beta} \cdot d\Omega_a = - \int_{\Omega_a} \pi_{\alpha\beta} \cdot \frac{\partial x_i}{\partial a_\alpha} \cdot \frac{\partial v^*_i}{\partial a_\beta} \cdot d\Omega_a$$

Since gradient tensor matrix components F_{ia} can be expressed from displacement components between the reference and the present configurations as : $\frac{\partial x_i}{\partial a_\alpha} = \delta_{i\alpha} + \partial \Delta u_i / \partial a_\alpha$ (15)

the resulting non linear quantities : $\partial \Delta u_i / \partial a_\alpha \cdot \partial v^*_i / \partial a_\beta$ in (14) produce buckling matrix terms, when discretized in an implicit code. In an explicit code, computations are instead achieved on the known situation \mathcal{D} at time t , avoiding non linear terms with equations (12).

1.5 - Stress-strain relationship

In order to cope with any type of material behavior, elastic, plastic or of any other type, for instance recovered from experimental measurements, PLEXUS is designed to handle them in an incremental way. Let \mathcal{D}_τ and \mathcal{D}_t be the configurations reached by a deformable body \mathcal{D} at successive times τ and $t = \tau + \Delta t$. For a zero strain change rigid body transformation between these two instants, such as a rotation, the $\sigma_{ij}(t)$ components of the Cauchy stress tensor at time t are in general different from the same quantities $\sigma_{ij}(\tau)$ at the previous instant, whereas the

two Piola-Kirchhoff stress tensor components at both times $\pi_{\alpha\beta}(t)$ and $\pi_{\alpha\beta}(\tau)$ are identical.

Therefore, the incremental law must be written : $\pi_{\alpha\beta}(t) = \pi_{\alpha\beta}(\tau) + f_{\alpha\beta}(\Delta\varepsilon_{kl})$ (16)

The reference configuration \mathcal{D}_a at time t_a is also, by definition, the only one for which the two types of stresses are identical : $\sigma_{ij}(t_a) = \pi_{ij}(t_a)$ (17)

When the latter is changed during the movement, in an Updated Lagrangian type formulation, in such a way that at any time t the reference configuration be the previous one actually occupied at time τ , the incremental law becomes : $\pi_{ij}(t) = \sigma_{ij}(\tau) + f_{ij}(\Delta\varepsilon_{kl})$ (18)

where strain increments are computed from the usual orthonormal frame relations with Euler variables : $\Delta\varepsilon_{kl} = 1/2 \cdot (\partial\Delta u_k / \partial x_l + \partial\Delta u_l / \partial x_k + \partial\Delta u_m / \partial x_k \cdot \partial\Delta u_m / \partial x_l)$ (19)

In PLEXUS, displacements increments Δu are assumed small with respect to body dimensions : the product terms, normally to be kept in equation (19), are of order 2 and therefore vanish so that strain increments reduce to : $\Delta\varepsilon_{kl} = 1/2 \cdot (\partial\Delta u_k / \partial x_l + \partial\Delta u_l / \partial x_k)$ (20)

Piola-Kirchhoff stresses $\pi_{ij}(t)$ are thus computed at time t in the reference configuration \mathcal{D}_τ at time τ , using equations (18) and (20), and then converted into Cauchy stresses $\sigma_{ij}(t)$ at time t in the present configuration \mathcal{D}_t : these are required during the next time step to compute Piola-Kirchhoff stresses $\pi_{ij}(t+\Delta t)$ at time $t+\Delta t$ in the updated reference configuration \mathcal{D}_t . This conversion results from relation (5) :

$$\sigma_{ij}(t) = \rho / \rho_a \cdot \partial x_i / \partial a_\alpha \cdot \partial x_j / \partial a_\beta \cdot \pi_{\alpha\beta}(t) \quad (5')$$

with partial derivatives substituted from (18). Discarding terms of order greater or equal to 2 results in :

$$\sigma_{ij}(t) = \rho / \rho_a \cdot [\pi_{ij}(t) + \partial\Delta u_i / \partial a_\alpha \cdot \pi_{\alpha j}(t) + \partial\Delta u_j / \partial a_\beta \cdot \pi_{i\beta}(t)] \quad (21)$$

Replacing partial derivatives w.r.t. the reference coordinates by those w.r.t. the present ones :

$$\sigma_{ij}(t) = (1 - \partial\Delta u_m / \partial x_m) \cdot \pi_{ij}(t) + \partial\Delta u_i / \partial x_k \cdot \pi_{kj}(t) + \partial\Delta u_j / \partial x_l \cdot \pi_{il}(t) \quad (22)$$

and after reintroducing Cauchy stresses : $\pi_{ij}(t) = \sigma_{ij}(\tau) + f_{ij}(\{\Delta\varepsilon\})$ this approximation reduces to : (23)

$$\sigma_{ij}(t) = \sigma_{ij}(\tau) + f_{ij}(\{\Delta\varepsilon\}) - \partial\Delta u_m / \partial x_m \cdot \sigma_{ij}(\tau) + \partial\Delta u_i / \partial x_k \cdot \sigma_{kj}(\tau) + \partial\Delta u_j / \partial x_l \cdot \sigma_{il}(\tau)$$

or, in the final matrix form :

$$[\Delta D] = [\Delta D_{ij}] = [\partial\Delta u_i / \partial x_j]$$

$$[\Delta\varepsilon] = [\Delta\varepsilon_{ij}] = 1/2 \cdot ([\Delta D] + [\Delta D]^T) \quad (24)$$

$$[\sigma(t)] = [\sigma(\tau)] + [f[\Delta\varepsilon]] - \text{Tr}([\Delta D] \cdot [\sigma(\tau)] + [\Delta D]^T \cdot [\sigma(\tau)] + [\sigma(\tau)]^T \cdot [\Delta D])$$

1.6 - Time integration and stability criterion

Let us consider that in the \mathcal{D}_n configuration at time t_n all dynamic quantities are known :

geometry $\{x_n\}$, velocity $\{v_n\} = \partial\{x_n\} / \partial t$, and acceleration $\{\gamma_n\} = \partial^2\{x_n\} / \partial t^2$,

Their computation at the next instant during numerical integration : $t_{n+1} = t_n + \Delta t_{n+1}$ is based on the well known *explicit integration algorithm* called the 'Central Difference Method', using the following set of equations, in which variable time steps Δt_{n+1} can be used :

- prediction of new geometry : $\{x_{n+1}\} = \{x_n\} + \Delta t_{n+1} \cdot \{v_n\} + 1/2 \cdot \Delta t_{n+1}^2 \cdot \{\gamma_n\}$ (25)

$$\text{or : } \{\Delta u_{n+1}\} = \{x_{n+1}\} - \{x_n\} = \Delta t_{n+1} \cdot \{v_n\} + 1/2 \cdot \Delta t_{n+1}^2 \cdot \{\gamma_n\} \quad (25')$$

for the displacement change. The *new configuration* \mathcal{D}_{n+1} is therefore *completely known*.

- anticipation of new velocity : $\{v_{n+1}\} = \{v_{n+1/2}\} + \Delta t_{n+1}/2 \cdot \{\gamma_{n+1}\}$ (26)
 $= [\{v_n\} + \Delta t_{n+1}/2 \cdot \{\gamma_n\}] + \Delta t_{n+1}/2 \cdot \{\gamma_{n+1}\}$

- strain increment definition during time step Δt_{n+1} by referring the difference in displacement changes $\{\Delta u_{n+1}\}$ and $\{\Delta u_n\}$ at two consecutive instants to the corresponding position change $\{x_{n+1}\} - \{x_n\}$: $[\Delta D_{n+1}] = [\Delta\{\Delta u_{n+1}\} / \Delta\{x_{n+1}\}]$
 $[\Delta \epsilon_{n+1}] = 1/2 \cdot [[\Delta D_{n+1}] + [\Delta D_{n+1}]^T]$ (27)

- Cauchy stresses updating in the new configuration at time t_{n+1} using the incremental relation described in the previous paragraph :

$$[\sigma_{n+1}] = [\sigma_n] + [f[\Delta \epsilon_{n+1}]] \quad (28)$$

$$- \text{Trace}([\Delta D_{n+1}]) \cdot [\sigma_{n+1}] + [\Delta D_{n+1}]^T \cdot [\sigma_{n+1}] + [\sigma_{n+1}]^T \cdot [\Delta D_{n+1}]$$

- dynamics equations at time t_{n+1} : $\forall v^* \text{ K.A.}$ (29)

$$\int_{\Omega_{n+1}} \rho \cdot \gamma_{i_{n+1}} \cdot v^*_i \cdot d\Omega = \int_{\Omega_{n+1}} \rho \cdot g_i \cdot v^*_i \cdot d\Omega + \int_{\Gamma_{F_{n+1}}} F_{i_{n+1}} \cdot v^*_i \cdot d\Gamma_F + \int_{\Omega_{n+1}} \sigma_{ij_{n+1}} \cdot D^*_{ij} \cdot d\Omega$$

Since the second member of this equation is known, the acceleration $\{\gamma_{n+1}\}$ is computed at time t_{n+1} and the velocity $\{v_{n+1}\}$ is deduced from (26). This way, all dynamical quantities are known at the end of this time step, and the same procedure is reproduced for the next instant $t_{n+2} = t_{n+1} + \Delta t_{n+1}$, at the end of the $(n+2)^{\text{th}}$ time increment.

However, explicit codes, which appear in the above procedure as easy-to-chain calculations, conversely show the major disadvantage of their *conditional stability* : computation results will only converge to stable and exact values if the time increment Δt used for numerical integration is smaller than a given limit value Δt_{crit} , shown to be inversely proportional to the stiffness of the elements that build up the structure : $\Delta t_{\text{crit}} = \text{Inf}(L_e/c_e)$ (30)

with : L_e : smallest dimension of element e , c_e : sound velocity in the element's material.

This stability limit time is thus enforced by the stiffer part of a material system and induces long and costly calculations, especially for the more flexible parts for which a much larger time step could be afforded. Since a nearly rigid substructure, as often encountered in manipulator links, would drag the limit time step to tiny, and therefore unacceptable, levels, we implemented in PLEXUS a rigid body model which is detailed in the next part of this document.

1.7 - Space integration

Since each particle M of a deformable body has three degrees of freedom, namely three translations measured in the motion's reference frame \mathcal{R} , the real displacement field $\{u_r(M)\}$ theoretically contains an infinite number of independent parameters. In practice, such a size cannot be handled and an acceptable approximation of the problem is required to model the system's behavior with a finite number of parameters. At the poorest end of kinematical models, the rigid body one only possesses six d.o.f. to describe the movements of all its points, namely three translations and three rotations, so that no deformability can be modelled, which may sometimes prove insufficient. More realistic models consider that, during motion, neighboring particles in a deformable body are neither rigidly linked, nor fully free to move away from each other because of the intrinsic stiffness in the material that keeps separating distances 'close' to their original values. And instead of approximating the real displacement field $\{u_r(M)\}$ as a unique continuous function over the entire body, piecewise models are preferred, based on subdividing the volume of the system under study into a large number of contiguous pieces, small in size with respect to its overall dimensions. These elements are interconnected at a limited number of nodal points, the displacements of which become the problem unknowns. This type of approximation has given birth to many widely used *Finite Elements* computer codes, each with its own library of elements, individually specialized for the different types of problems at hand : 1-D, 2-D or 3-D approximation of the structure; number of internal and/or boundary nodes; number and nature of nodal degrees of freedom; types of materials : solid, fluid, coupling; The interference of this technique with the dynamics equations is shortly presented below for the PLEXUS code .

Let $\{u(t)\}_e$ be the column matrix containing the time varying translational and/or rotational d.o.f.'s at the nodal points of an arbitrary element e in a given mesh. At a current point M in this element, the displacements are interpolated from these nodal unknowns using polynomial shape functions $\phi_i(\{x\})$, the variables of which are the coordinates $\{x_M\}$ of point M in the frame local to element e . The values $\phi_i(\{x_M\})$ show how the i^{th} nodal d.o.f. of the element influences the displacement components at point M : $\phi_i(\{x_M\})$ varies between 1. and 0. with the location of M . Applied to displacement (the same holds for velocity and acceleration) , the shape function

interpolation reads in matrix notation :

$$\{u(\{x_M\}, t)\}_e = [\phi(\{x_M\})] \cdot \{u(t)\}_e \quad (31)$$

Derivating displacements w.r.t. position variables yields strain components as :

$$\{\varepsilon(\{x_M\}, t)\}_e = [B(\{x_M\})] \cdot \{u(t)\}_e \quad (32)$$

where the $[B(\{x_M\})]$ matrix collects derivatives of shape functions $\phi_i(\{x_M\})$ with respect to position parameters in the element's local frame. The kinematically admissible virtual velocity field and the associated matrix of virtual strain rates , required in equation (29) of the P.V.W., are simply reproduced from the interpolation formulae used for the real displacement field :

$$\{v^*(\{x\})\}_e = [\phi(\{x\})] \cdot \{v^*\}_e \quad (33)$$

$$\{D^*(\{x\})\}_e = [B(\{x\})] \cdot \{v^*\}_e \quad (34)$$

Thanks to these hypotheses, the first member of equation (29) for element e is rewritten as :

$$\int_{\Omega_e_{n+1}} \rho \cdot \gamma_{n+1} \cdot v^*_i \cdot d\Omega = \{v^*\}_e^T \cdot \left(\int_{\Omega_e_{n+1}} \rho \cdot [\phi(\{x\})]^T \cdot [\phi(\{x\})] \cdot d\Omega \right) \cdot \{\gamma_{n+1}\}_e \quad (35)$$

The term between braces is the symmetrical *mass matrix* of element e, containing as many rows and columns as d.o.f.'s at the element nodes :

$$[M_e] = \int_{\Omega_e} \rho \cdot [\phi]^T \cdot [\phi] \cdot d\Omega \quad (36)$$

In the same way, the three terms on the right hand side of equation (29) become :

$$\int_{\Omega_e_{n+1}} \rho \cdot g_i \cdot v^*_i \cdot d\Omega = \{v^*\}_e^T \cdot \left(\int_{\Omega_e_{n+1}} \rho \cdot [\phi(\{x\})]^T \cdot \{g\} \cdot d\Omega \right) \quad (37)$$

$$\int_{\Gamma_{F_{en+1}}} F_{in+1} \cdot v^*_i \cdot d\Gamma_F = \{v^*\}_e^T \cdot \left(\int_{\Gamma_{F_{en+1}}} [\phi(\{x\})]^T \cdot \{F_{n+1}\} \cdot d\Gamma_F \right) \quad (38)$$

$$\int_{\Omega_e_{n+1}} \sigma_{ij_{n+1}} \cdot D^*_{ij} \cdot d\Omega = \{v^*\}_e^T \cdot \left(\int_{\Omega_e_{n+1}} [\phi(\{x\})]^T \cdot [\sigma_{n+1}] \cdot d\Omega \right) \quad (39)$$

Collecting as *external* (resp. *internal*) forces at time t_{n+1} all matrix quantities between braces in equations (37) and (38), (resp. (39)), these read :

$$\{F_{ext_{n+1}}\}_e = \int_{\Omega_e_{n+1}} \rho \cdot [\phi]^T \cdot \{g\} \cdot d\Omega + \int_{\Gamma_{F_{en+1}}} [\phi]^T \cdot \{F_{n+1}\} \cdot d\Gamma_F \quad (40)$$

$$\{F_{int_{n+1}}\}_e = \int_{\Omega_e_{n+1}} [B]^T \cdot [\sigma_{n+1}] \cdot d\Omega \quad (41)$$

so that equation (29) for the P.V.W. is extended in matricial form to all the elements building the system under study as :

$$\forall \vec{v^*} : \sum_e \{v^*\}_e^T \cdot \left\{ [M_e] \cdot \{\gamma_{n+1}\}_e - \{F_{ext_{n+1}}\}_e + \{F_{int_{n+1}}\}_e \right\} = 0 \quad (42)$$

Defining the three column matrices, extended to the whole dimension of the problem (i.e. with as many terms as the number \mathcal{N} of d.o.f.'s for all the nodes of the system), of the cumulated internal forces, cumulated external forces, and of the accelerations at all nodes of the mesh :

$$\{F_{int}\} = \sum_e \{F_{int}\}_e , \quad \{F_{ext}\} = \sum_e \{F_{ext}\}_e \quad \text{and} \quad \{\gamma\} = \sum_e \{F_{int}\}_e$$

and the square matrix of the structure's total mass obtained by adding the separate contributions of all the elements :

$$[M] = \sum_e [M_e]$$

the final linear system of \mathcal{N} equations to be solved for the \mathcal{N} unknown nodal accelerations at time t_{n+1} reads :

$$[M] \cdot \{\gamma_{n+1}\} = \{F_{ext_{n+1}}\} - \{F_{int_{n+1}}\} \quad (43)$$

To reduce calculation times, especially during inversion, the mass matrix is condensed into a diagonal form as :

$$[M_i^j]_e = \sum_j [M_{ij}]_e \quad (44)$$

1.8 - Couplings between degrees of freedom

If two bodies \mathcal{D}_1 and \mathcal{D}_2 are meshed separately, their interconnection, either total to represent a rigid body or partial to model a mechanism, requires that appropriate relationships be imposed to the different d.o.f.'s of the *distinct nodes* $N_j(e_1)$ and $N_j(e_2)$ that realize the junction of the elements e_1 and e_2 belonging to the two structures. Therefore, the number of unknowns is no longer minimal and some d.o.f.'s that were originally assumed to be independent parameters are *constrained* by these relations : the size of the linear system to be solved is thus larger. The constraint equations are designed to satisfy two goals :

- from the kinematical standpoint : having nodes $N_j(e_1)$ and $N_j(e_2)$ in coincidence at all times requires that their coordinate $\{x\}$ or displacement components $\{u\}$ be equal. Generally speaking, PLEXUS is capable to handle *couplings between nodal degrees of freedom* given in the following form :

$$[C] \cdot \{u\} \geq \{D\} \quad (45)$$

where $\{u\}$ is the column matrix containing all d.o.f.'s of the system,

$[C]$ is a coefficient matrix partially filled with constant or variable terms (varying with time or configuration) ; there usually is a small number of non-zero C_{ij} terms that correspond to the coupled d.o.f.'s

$\{D\}$ is the right hand side member, that can also be non-zero.

The specific application of this procedure to node full connection, as in a rigid body, or partial linkage, as in a kinematical mechanism, is detailed in the next part of this paper. In these cases, the equality sign prevails in equation (45) : $[C] \cdot \{u\} = \{D\}$ (45') in which the C_{ij} and D_i coefficients will be assumed time independent.

- from the dynamical standpoint : in order to ensure that the above kinematical relations be achieved for couples of nodes $N_j(e_1)$ and $N_j(e_2)$, complementary forces must be added to the dynamics equations on those rows refering to coupled d.o.f.'s.

The dynamical behavior of the whole structure with interconnected nodes is then expressed at time t_{n+1} by the matricial system (43), including all d.o.f.'s but with added complementary forces $\{F_{comp}\}$ such that the induced acceleration $\{\gamma_{n+1}\}$ produces a new geometry which fulfills node linkages :

$$[M] \cdot \{\gamma_{n+1}\} = \{F_{ext\ n+1}\} - \{F_{int\ n+1}\} + \{F_{comp\ n+1}\} \quad (46)$$

These couplings are dealt with in an *implicit* manner (whereas all other d.o.f.'s are still handled in an *explicit* way) : this requires that v new unknowns parameters $\{\lambda\}$ called *Lagrange multipliers* be solved for, one for each of the v constraint equations, which are such that :

$$\{F_{comp}\} = [C]^T \cdot \{\lambda\} \quad (47)$$

From a variational standpoint, equations (29) and (42) for the P.V.W. express that an energy type integral (the whole potential W of the structure) is minimized with respect to each of the system's \mathcal{N} position parameters u_j . If these unknown variables are subjected to v constraint equations $g_1(u)$, $g_2(u)$, ..., $g_v(u)$, mathematics say that each of the dynamics equations (43) : $\partial W / \partial u_i$, $i = 1, \mathcal{N}$, derived for each parameter as if they were all independent, must be completed by the sum of terms : $\lambda_k \cdot \partial g_k / \partial u_i$.

For linear constraints with constant coefficients $\{g\} = [C] \cdot \{u\}$ as in (45'), this reduces to :

$$\partial g_k / \partial u_i = C_{ki} \quad \Rightarrow \quad \sum \lambda_k \cdot \partial g_k / \partial u_i = \sum \lambda_k \cdot C_{ki} = \{C_i\}^T \cdot \{\lambda\}$$

$\{C_i\}$ denoting the i th row of matrix $[C]$. Collected for all equations as $[C]^T \cdot \{\lambda\}$, these complementary terms change the dynamics equations, at time t_{n+1} for the constrained system, into : $\{[M] \cdot \{\gamma_{n+1}\} - \{F_{ext\ n+1}\} + \{F_{int\ n+1}\}\} - [C_{n+1}]^T \cdot \{\lambda_{n+1}\} = 0 \quad (48)$

The computation of the $\lambda_{k\ n+1}$'s and their related connection forces $\{F_{comp\ n+1}\}$ results from eliminating unknown nodal accelerations $\{\gamma_{n+1}\}$ from the $\mathcal{N}+v$ equations (45') and (48).

Twice deriving kinematical constraints w.r.t. time as : $[C] \cdot \{\gamma_{n+1}\} = 0$

while inverting (48) into : $\{\gamma_{n+1}\} = [M]^{-1} \cdot \{\{F_{ext\ n+1}\} - \{F_{int\ n+1}\} + \{F_{comp\ n+1}\}\}$ one gets :

$$[C] \cdot [M]^{-1} \cdot \{\{F_{ext\ n+1}\} - \{F_{int\ n+1}\}\} = [C] \cdot [M]^{-1} \cdot [C]^T \cdot \{\lambda_{n+1}\}$$

$$\Rightarrow \{\lambda_{n+1}\} = [[C] \cdot [M]^{-1} \cdot [C]^T]^{-1} \cdot [C] \cdot [M]^{-1} \cdot \{\{F_{ext\ n+1}\} - \{F_{int\ n+1}\}\} \quad (49)$$

Reintroducing the computed Lagrange multipliers into (48) ends the calculation of accelerations

$\{\gamma_{n+1}\}$ at time t_{n+1} which duly account for node connections. And thanks to this treatment, substructure partial or total linkages are automatically fulfilled at all times of the integration process *at the condition* that they are satisfied at the initial instant t_0 .

2 - Specific developments for the dynamical analysis of mechanisms

2.1 - Rigid body model

Because of its implicit nature, the Central Difference integration algorithm imposes a stability limit Δt_{crit} which is so much time consuming when rigid parts are included in a mechanical structure that a new approach was imperative to handle that precise case.

The first step in building the rigid body model arises from the fact that for any previously defined subdivision of a solid into finite elements and nodal points, the mechanical properties (i.e. : mass M , location of the barycenter G and six components of the central inertia tensor in the problem's reference frame) will be conserved by replacing these many scattered nodes by an equivalent set of *four* appropriately located *equal point masses* at fixed distances from one another. The positions of these four points in the central principal frame \mathcal{R}_S attached to the solid, are computed as :

$$\begin{aligned} A_1 &= (0, -b, -c) & A_2 &= (0, b, -c) \\ A_3 &= (-a, 0, c) & A_4 &= (a, 0, c) \end{aligned} \quad (50)$$

a, b and c being deduced from the principal inertia components J_x, J_y, J_z in \mathcal{R}_S as :

$$a = (J_y + J_z - J_x) / M \quad b = (J_x - J_y + J_z) / M \quad c = (J_x + J_y - J_z) / 2 * M \quad (51)$$

The second step in 'rigidifying' the solid results from the condition of constant distances between any two points M_i and M_j in a rigid body, i.e. :

$$M_i^2 M_j^2 = L_{ij}^2 \Leftrightarrow \vec{M_i M_j} \cdot \frac{d\vec{M_i M_j}}{dt} = \vec{M_i M_j} \cdot (\vec{v_i} - \vec{v_j}) = 0 \quad (52)$$

equations which reduce the too many d.o.f.'s of the mesh nodes to the sufficient *six degrees of freedom* for the rigid body. In particular, when applied to the four points A_k , these kinematical constraints reduce the twelve d.o.f.'s so introduced to the six expected ones.^k

Therefore, to create a rigid body in PLEXUS, the user has to specify the group of mesh points M_i which will be replaced by the four point masses A_k . He must also take care to designate the necessary connection points P_j through which the rigid body will be, either completely or partially, linked to other substructures : these frontier nodes will be excluded from the reduction process. The mechanical properties of the solid are then either computed internally from the mass distribution at the remaining nodes M_i , or can be directly input by the user himself if precisely known values are to be employed.

Position parameters for both the four A_k points and the ξ chosen frontier nodes P_j therefore satisfy $(6 + 3*\xi)$ scalar conditions of type (52). If these relations are fulfilled at time t_n , the condition inferred on the position $\{x_{n+1}\}$ at t_{n+1} in order that the kinematical couplings remain true is obtained as follows :

- at the end of the n.th time increment : $[(M_i M_j)_n + dM_i M_j]^2 = (M_i M_j)_n^2 = L_{ij}^2$

so that relation (52) becomes :

$$[\overset{\rightarrow}{(M_i M_j)}_{n+1} + \overset{\rightarrow}{(M_i M_j)}_n] \cdot \overset{\rightarrow}{dM_i M_j} = 0 \quad (52')$$

or in terms of components, writing :

$$\{\Delta x_{ij}\} = \{x_j\} - \{x_i\}$$

$$\Rightarrow \{\{\Delta x_{ij}\}_{n+1}\} + \{\Delta x_{ij}\}_n\}^T \cdot \{\{\Delta x_{ij}\}_{n+1}\} - \{\Delta x_{ij}\}_n\} = 0 \quad (53)$$

- positions are deduced at the end of the $(n+1)^{th}$ time step from the Central Difference relation :

$$\{x_{n+1}\} = \{x_n\} + \Delta t_{n+1} \cdot \{v_n\} + 1/2 \cdot \Delta t_{n+1}^2 \cdot \{\gamma_n\} = \{x_n\} + \Delta t_{n+1} \cdot \{v_{n+1/2}\}$$

so that :

$$\{\Delta x_{ij}\}_{n+1} = \{\Delta x_{ij}\}_n + \Delta t_{n+1} \cdot \{\Delta v_{ij}\}_{n+1/2} \quad (54)$$

- reporting (54) into (53) leads to the set of relations that positions, and also displacements, must satisfy at t_{n+1} for the rigid body condition to hold :

$$2 \cdot \{\Delta v_{ij}\}_{n+1/2}^T \cdot \{\Delta x_{ij}\}_{n+1} = \Delta t_{n+1} \cdot \{\Delta v_{ij}\}_{n+1/2}^2 \quad (55)$$

After such relations have been written for all pairs of rigidly connected points A_k and P_j in the body, relations (55) are grouped in the following matrix form : $[C] \cdot \{u\} = \{D\}$ similar to equation (45') but in which the right hand side is time variable and the values for the $v * \mathcal{N}$ coefficients depend on the nodal coordinates at time t

- coupled with the similar equation written at instant t_{n+2} , relation (55) induces the condition that accelerations must satisfy at t_{n+1} for the solid to behave rigidly :

$$\{\Delta v_{ij}\}_{n+3/2}^T \cdot \{2 \cdot \{\Delta x_{ij}\}_{n+2} - \Delta t_{n+2} \cdot \{\Delta v_{ij}\}_{n+3/2}\} = 0 \quad (55')$$

with :

$$\{x_{n+2}\} = \{x_{n+1}\} + \Delta t_{n+2} \cdot \{v_{n+3/2}\} \quad \& \quad \{v_{n+3/2}\} = \{v_{n+1/2}\} + 1/2 \cdot (\Delta t_{n+1} + \Delta t_{n+2}) \cdot \{\gamma_{n+1}\}$$

This is ultimately simplified to the first order as :

$$\{\{\Delta x_{ij}\}_{n+1}\} + \Delta t_{n+2} \cdot \{\Delta v_{ij}\}_{n+1/2} \}^T \cdot \{\Delta \gamma_{n+1}\} = - \{\Delta v_{ij}\}_{n+1/2}^2 \quad (56)$$

which are the relations to be observed at time t_{n+1} by the accelerations $\{\gamma_{n+1}\}$ of the rigidly interconnected nodes :

$$[C_{n+1}] \cdot \{\gamma_{n+1}\} = \{S_{n+1}\} \quad (57)$$

These v relations, with a known non-zero right hand side $\{S_{n+1}\}$, induce again complementary forces : $\{F_{comp\ n+1}\} = [C_{n+1}]^T \cdot \{\lambda_{n+1}\}$

which modify the \mathcal{N} dynamics equations (46) at t_{n+1} , and the v involved Lagrange multipliers $\lambda_{k\ n+1}$ are computed as in section 1.8 by eliminating the unknown nodal accelerations $\{\gamma_{n+1}\}$ from the $\mathcal{N}+v$ equations (48) and (57) :

$$\{\lambda_{n+1}\} = [H]^{-1} \cdot ([C] \cdot [M]^{-1} \cdot \{\{F_{ext\ n+1}\} - \{F_{int\ n+1}\}\} - \{S_{n+1}\}) \quad (58)$$

with :

$$[H] = [[C] \cdot [M]^{-1} \cdot [C]^T]$$

Reintroducing the multipliers into (48) directly delivers the acceleration components $\{\gamma_{n+1}\}$.

2.2 - Elementary mechanism model

In articulated mechanisms, different rigid or deformable substructures are interconnected by suppressing from one to five degrees of freedom at shared joints. While a general instruction is available in PLEXUS to prescribe relations between d.o.f.'s, we have specialized it to the three most often employed elementary technological mechanisms : the *revolute* or *pin* joint, the *sliding* or *prismatic* joint and the *spherical* or *ball and socket* joint. Other models can easily be developed (universal joint, sliding plane, ...), according to the following concept which was used for the three previous ones.

An *elementary mechanism* is defined as a new type of element composed of :

- * two end nodes L_A and L_B , both having the same geometrical position, and possessing 6 d.o.f.'s to authorize the writing of any kinematical relation. \rightarrow
- * the unit direction vector of the relative motion axis at the joint, V_{AB} , through L_A and L_B .

Besides this information, the user must also input to which substructure \mathcal{A} (resp. \mathcal{B}) node L_A (resp. L_B) is rigidly connected, in order that the imposed kinematical relation be effectively applied to the two mechanism links. If the relevant substructure, say \mathcal{A} , has been divided into elements with 6 d.o.f.'s per node, the corresponding mechanism end point, say L_A , can be directly identified to one of the substructure nodal points. Conversely, if nodes in \mathcal{A} only have three translational d.o.f.'s, the user must choose a *vicinity area* including at least four non coplanar points of \mathcal{A} from which angular information can be transferred to L_A : only then can the appropriate kinematical relations be written for that elementary mechanism.

The following equations for a *pin joint* show how the related five scalar kinematic constraints are treated in the element's local frame based on V_{AB} . Let $\{\Delta r_{A\ n+1}\}$ and $\{\Delta\theta_{A\ n+1}\}$ be the translational and rotational displacement increment submatrices of $\{\Delta u_{A\ n+1}\}$ at node L_A during time step $n+1$. With the same notation for node L_B , the partial linkage of the two end nodes L_A and L_B during that time increment induces the following five scalar equations :

$$\text{- translation identity} \quad \vec{\Delta r}_{AB\ n+1} = \vec{\Delta r}_{A\ n+1} - \vec{\Delta r}_{B\ n+1} = \vec{0} \quad (59)$$

$$\text{- rotation freedom along } V_{AB} \quad \vec{\Delta\theta}_{AB\ n+1} = \vec{\Delta\theta}_{A\ n+1} - \vec{\Delta\theta}_{B\ n+1} \quad (60)$$

$$\vec{\Delta\theta}_{AB\ n+1} - [\vec{\Delta\theta}_{AB\ n+1} \cdot \vec{V}_{AB\ n+1/2}] \vec{V}_{AB\ n+1/2} = \vec{0} \quad (61)$$

The induced relations on d.o.f.'s, presenting coefficients that vary with nodal positions and orientations at time t_n , are again handled by the Lagrange multiplier technique. Since the V_{AB} axis is moved along by the displaced and deformed substructures which it joins, it is updated at

the end of this time step as :

$$\vec{V}_{AB\ n+1} = \vec{V}_{AB\ n} + \vec{\Delta\theta}_{A\ n+1} \wedge \vec{V}_{AB\ n} \quad (62)$$

2.3 - Control model

Since the dynamics equations incorrectly model, and even omit, actual phenomena (mass and inertia distribution, friction, motor dynamics), the results they yield cannot be directly applied to move mechanisms. Instead, automatism principles are used to control the different motion parameters of the structure by continuously checking how well they stick to their prescribed variations. This information, delivered by position and velocity sensors, is sampled at fast rates and the measured error signals are amplified to induce correcting motor actions tending to make them vanish. The most simple and widely used control techniques are based on constant gain linear P.I.D. controllers which, by proper tuning of their gain values, can be made quite effective to handle nonlinear phenomena. We therefore introduced the modeling of this feature in PLEXUS to enable the code to better simulate actual manipulators and to make it a good testing basis for new more sophisticated control algorithms. As for the presently available P.I.D. models for independant joint control, they are based on the following ideas :

- * *Proportional* : the position difference ($q_j(t) - q_{jd}(t)$) is used to define the correcting effort at time t according to :

$$F_j(t) \text{ or } C_j(t) = -K_{jp} \cdot (q_j(t) - q_{jd}(t)) \quad (63)$$

where $q_j(t)$ stands for every actual degree of freedom, linear or angular, at manipulator joints, and $q_{jd}(t)$ for its 'desired' counterpart.

Proportional control at a joint acts like a local spring with stiffness K_{jp} and with a varying equilibrium position $q_{jd}(t)$: the restoring force F_j or torque C_j tends to cancel the forward [$(q_j - q_{jd}) > 0$] or backward [$(q_j - q_{jd}) < 0$] difference existing at each instant.

- * *Derived* : a similar treatment applied to the first time derivatives $q'_j(t)$ and $q'_{jd}(t)$ of each individual joint's real and desired parameters leads to :

$$F_j(t) \text{ or } C_j(t) = -K_{jd} \cdot (q'_j(t) - q'_{jd}(t)) \quad (64)$$

- * *Integral* : extending the above relations to the time integration of joint parameters provides the third classical control component as :

$$F_j(t) \text{ or } C_j(t) = -K_{jl} \cdot (Q_j(t) - Q_{jd}(t)) \quad (65)$$

where the two areas spanned in the time domain ($t, q_i(t)$) are compared :

$$Q_i(t) = \int_{t_0}^t q_i(\tau) \cdot d\tau \quad \text{and} \quad Q_{id}(t) = \int_{t_0}^t q_{id}(\tau) \cdot d\tau$$

In PLEXUS, P.I.D. control simulation requires that constant gains K_{jp} , K_{jd} and K_{jl} be fixed for each one d.o.f. joint, and that a prescribed parameter variation $q_{id}(t)$ be input in functional or tabulated form. If \mathcal{A} and \mathcal{B} are the substructures connected by joint j , control simulation on this active d.o.f. at time t_n induces a correction effort along the motion vector $\vec{V}_{AB\ n}$ for that

joint : the resulting d.o.f. variation during time increment ($n+1$) is computed while the other five constrained d.o.f.'s for that joint satisfy relations, such as (59)-(61) above for a pin joint.

2.4 - Electrical motor and transmission model

The previous controller model is not still appropriate for two reasons : first, control is not actually exerted directly on the motoring force or torque, but rather on the quantity input to the motor : current voltage or intensity, fluid pressure, ... ; secondly, in order to fulfill the above requirement, the mechanical behavior of the motor, gear box and transmission equipment must be integrated in the dynamics of the entire system.

Since this type of actuator is widely used in motoring mechanisms and presents well known equations, the direct current electrical motor has been implemented in PLEXUS. The following equations express how the torque C_a transmitted to the output shaft, and therefore available to move adjacent links, is related to the input motor tension V_a that is usually monitored through a P.I.D. controller as described above. They also stress why the dynamics of motoring devices cannot be discarded : a large part of the current induced torque C_m applied to the rotor is dissipated merely to overcome the inertia J_{eq} of its rotating parts and dry and viscous friction resisting torques C_f and $f \cdot \omega_m$. Considering the following notation : R , L : resistance and self-inductance of the coil ; E : counter-electromotrice force of the D.C motor ; K_e : its proportionality coefficient ; Ω_m : output angular velocity of the motor ; N : gear box reduction ratio ; V_a : input current voltage ; I : current intensity ; C_m : torque generated on the rotor ; K_m : torque proportionality coefficient (and i, general : $K_e = K_m = K$) ; the electrical and mechanical equations then reduce to :

$$V_a = R \cdot I + L \cdot dI/dt + E \quad ; \quad E = K_e \cdot \Omega_m \quad ; \quad C_m = K_m \cdot I \quad (66)$$

$$N \cdot C_m = J_{eq} \cdot d\omega_m / dt + f \cdot \omega_m + C_f + C_a \quad (67)$$

in which ω_m is the arm relative angular velocity ; J_{eq} is the equivalent inertia moment for all rotating parts, referred to the output shaft of the gear box ; f is the equivalent viscous friction coefficient referred to the same axis ; C_f is the dry friction torque in the same conditions ; and C_a is the available torque at the output of the gear box and thus applied between the two adjacent links. All these properties must be input by the user and considerably influence the structure dynamics and control responses, especially when high reduction ratios are involved.

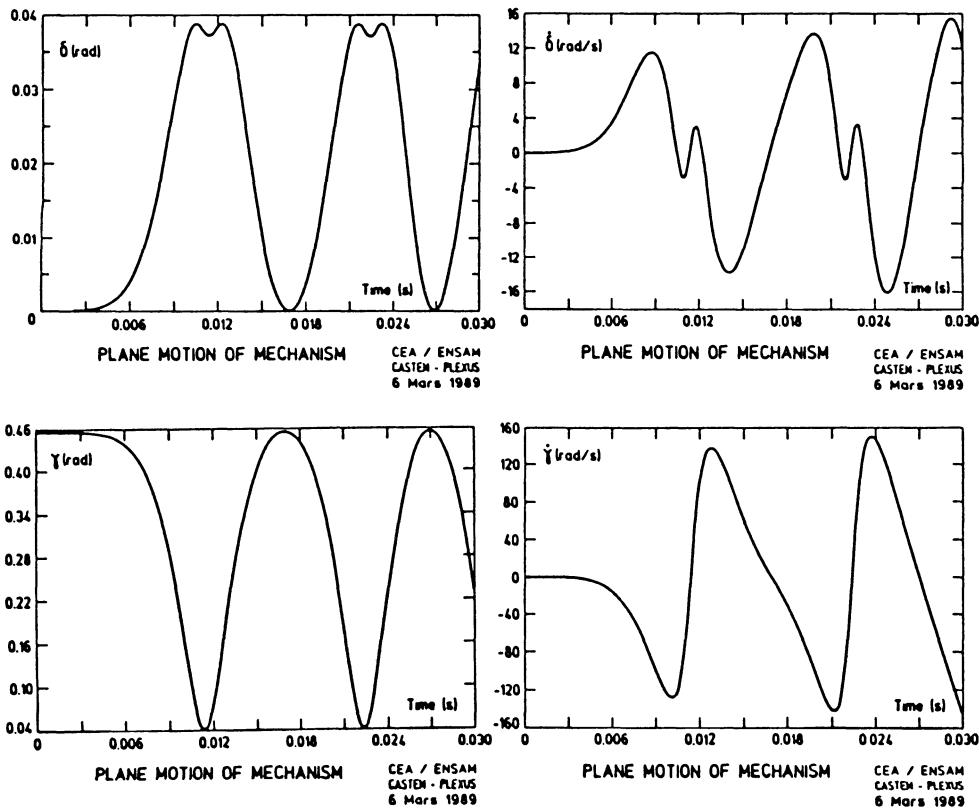
3 - Examples

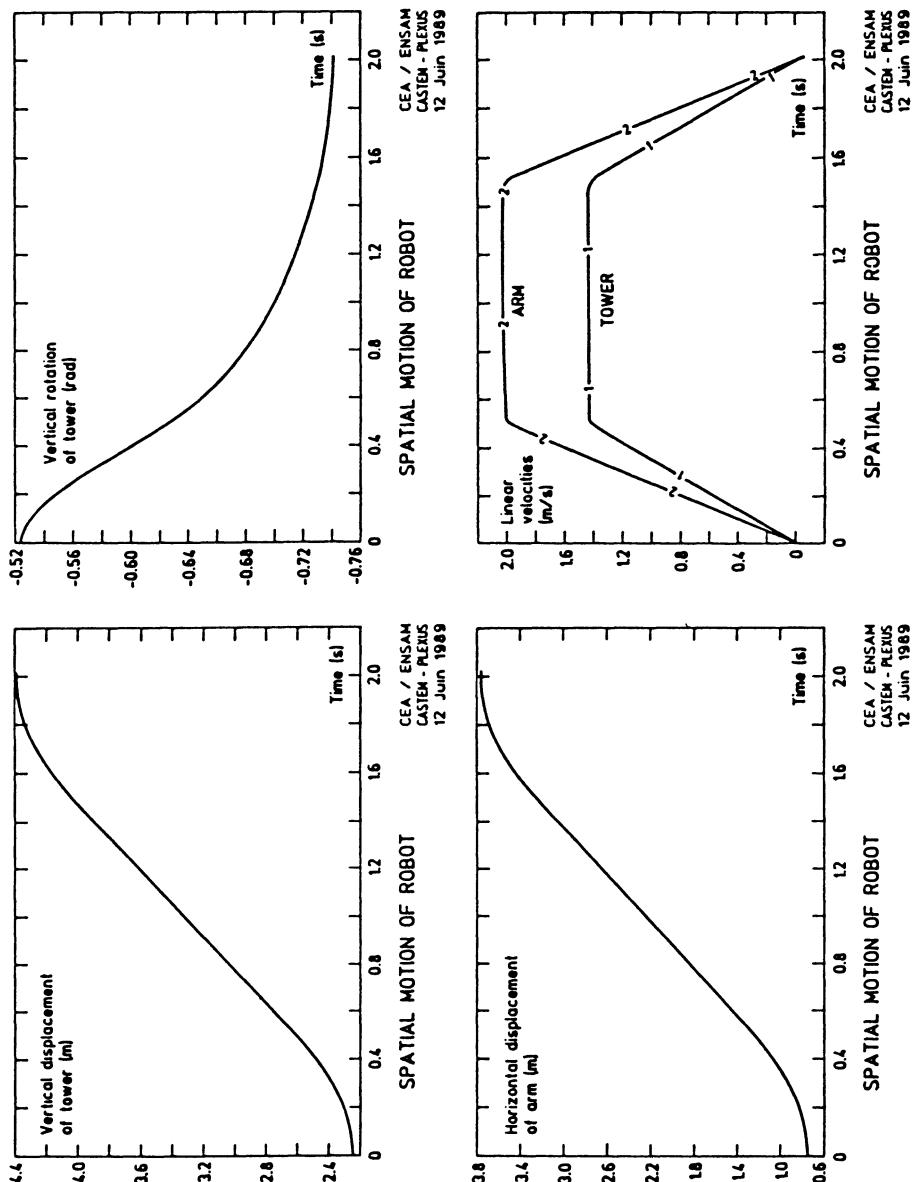
The data files for the 'Plane Motion Mechanism' test example defined in this handbook are displayed at the end of this document. Comments are spread all over these input files to make

them easy to read. In a first step, the geometrical definition and visualization of the system to be analyzed is achieved by the GIBI pre-processor and this file can be interactively corrected ; its output is recovered in a second step by a PLEXUS file, in which other analysis conditions are defined and execution started. Without going into free format input details, it is seen that understandable keywords and symbols are used to define all necessary data for the analysis : geometry, material properties, element types, loading conditions, calculation and output requirements. Named entities are introduced to create points, lines, surfaces and volumes, which can then automatically be subdivided into finite elements by means of condensed instructions. Separately meshed parts can easily be concatenated into a new named entity. Varying data are input either as predefined functions : sine, cosine, ... , or as tables of numerical values.

Before the whole calculation is started, a check run of one step displays in an easy-to-read form all data stored in PLEXUS internal files : all input values can thus be checked for correctness. And after simulation is completed, output diagrams can be drawn with sufficiently explicit comments for an easy understanding. Mechanism animations result from drawing deformed positions of the structure at selected time intervals , either superimposed or individually.

Four output curves for each of the two test examples are reported below, keeping the same notations as those suggested in the example definition file.





Output angular displacements and velocities for plane mechanism motion analysis

(figures 1 to 4 , previous page)

and output linear and angular displacements for spatial robot motion analysis

(figures 5 to 8 , above)

FILE: TEST1 DC1B1 AI WELCOME TO SHOTS W/S/SPS OPS 511

FILE: TEST1 DC1B1 AI WELCOME TO SHOTS W/S/SPS OPS 511

LR = AR D 1 BR;
RESORT = LR COUL; VERT;

OPT1 DIME 3;
OPT1 ELEM SEC1;

QEIL = 0 0 1000;
PO = -0.093444016 -0.0227576607 0;
PA = -0.063546-058 -0.02737386.91 0;
PB = -.014 .072 0;

DEFINITION DES POINTS MATERIELS

POA = HANIEL PO1 PO;
PFB = HANIEL PO1 PB;
PMC = PFO ET PFB ET PMC;

DEFINITION DES PIVOTS

PIV1 = HANIEL SEC2 PO A1;
PIV2 = HANIEL SEC2 BI A2;
PIV3 = HANIEL SEC2 BI B3;
PIV4 = HANIEL SEC2 BI A4;
PIV5 = HANIEL SEC2 AA A5;
PIV6 = HANIEL SEC2 BB B7;

PIV7 = HANIEL SEC2 AV PA;
PIV8 = HANIEL SEC2 PA A5;
PIV9 = HANIEL SEC2 BS B4;
PIV10= HANIEL SEC2 PB A3;
PIV11= HANIEL SEC2 D3 AC;
PIV12= HANIEL SEC2 BK PC;
PIV = PIV ET PIV2 ET PIV3 ET PIV4 ET PIV5 ET PIV6 ET PIV7;

PIV = PIV ET PIV5 ET PIV9 ET PIV10 ET PIV11 ET PIV12;

NEOPFL = CORPS ET PIV ET RESORT ET POA ET BATI;

*SORTIE MECHANIC:

DEFINITION DES PARAMETRES

CORPS 3

DC1 = -0.28; DA = 0.015; F = -0.2; EA = -0.1421; FF = -.02; FA = -.0421;
SC = -.018; SD = .02; ST = .035; SA = .019745; SB = .009163; U = .04;
UA = .01238; UB = -.004493;

DEFINITION DES CORPS (REFERE LOCAL)

CORPS 1

AI = 0 0 0; BI = R 0 0; CI = RA 0 0;
LI = AI D 3 C1 D 3 BI;
CORPS1 = LI COOL BLEU;

CORPS 2

A2 = 0 0 0; B2 = -D1 0 0; C2 = DA 0 0;
L2 = AD 3 C2 D 3 B2;
CORPS2 = L2 COOL ROUE;

CORPS 3

A3 = 0 0 0; B3 = -S 0 0; D3 = SC SD 0; C3 = SA SB 0;
L3 = AD 3 CD 3 B3; L32 = C3 D 3 D3;

CORPS4 = L31 ET L32 COUL. ROUE;

CORPS 4

A4 = 0 0 0; B4 = E 0 0; CA = EA 0 0;
CORPS4 = LA COOL JAHN;

CORPS 5

A5 = 0 0 0; B5 = ZZ 0 0; CS = EA 0 0;
L5 = AS D 3 CS D 3 B5;
CORPS5 = L5 COUL. VERT;

CORPS 6

A7 = 0 0 0; B7 = U 0 0; CT = UA UB 0;
L7 = AV D 3 C7 D 3 B7;
CORPS7 = L7 COOL BLEU;

RESORT

ANCR = 62.3071251; VECR = D;
DEPL. RESORT TORN ANCR AR (AR PLUS (0 0 1));

DEFINITION DES BATIS

P1 = -.0015 -0.003 0; P2 = -.0015 -.003 0; PO = 0 0 0;
BATIO = PO D 1 P1 D 1 P2 D 1 PO;
CORPS-CORPS1 ET CORPS2 ET CORPS3 ET CORPS4;
CORPS-CORPS ET CORPS5 ET CORPS6 ET CORPS7;

BATIC = BATIO PLUS PA;
BATIC = BATIO PLUS PB;
BATIC = BATIO PLUS PC; DEPL. BATIC TORN 180 PC (PC PLUS (0 0 1));
BATIC = BATIA ET BATIA ET BATIS ET BATIC;

AR = 0 0 0; BR = 5.2667E-2 0 0;

```

FILE: TEST1 DPLEXUS AI WELCOME TO SHOTS W/S/SPS OPS 511
... PLANE MOTION OF MECHANISM ...
ECHO
GIBI 9 MECHANISM
TRID LINE
6
DIMENSION
ZONE 4 POUT 200 BRAD 13 PHAT 7 MEGA 12 PFLM 56 PTFL 12
PHAT 4 FTHL 1 FDS 2
SOUL 7 PFL 15 PFLB 37 BLQD 24
TILLION 126 3045 DFLAT 45 EFLDU 438 TRAV 2498
OPTION TASSER
6
CLOM
CLOM LECT CORPS1 TERM
BLF LECT AS B6 TERM
INER MASS 0.00706 XC -0.24050E-1 YC -0.35949E-2 ZC 0
IXX 5.667E-7
IY 5.667E-7
IZZ 5.667E-7
CORPS 6 '
ELBN LECT CORPS6 TERM
PLLE LECT AS B6 TERM
INER MASS 0.00706 XC -0.24050E-1 YC -0.35949E-2 ZC 0
IXX 1.169E-5
IY 1.169E-5
IZZ 1.169E-5
CORPS 6 '
ELBN LECT CORPS6 TERM
PLLE LECT AS B6 TERM
INER MASS 0.00706 XC -0.24050E-1 YC -0.35949E-2 ZC 0
IXX 1.169E-5
IY 1.169E-5
IZZ 1.169E-5
CORPS 6 '
ELBN LECT CORPS7 TERM
PLLE LECT AS B7 TERM
INER MASS 0.05498 XC -0.59246E-1 YC -0.10606E-1 ZC 0
IXX 1.912E-5
IY 1.912E-5
IZZ 1.912E-5
ANT1
PLVO LECT PIVV TERM
ARE VR VY VZ 1
NOEU LECT PO TERM VOIS ABSENT
NOEU LECT AI TERM VOIS INDEF 1
ROTU LECT PIVV TERM
NOEU LECT BI TERM VOIS INDEF 1
NOEU LECT AJ TERM VOIS INDEF 1
ROTU LECT PIVV TERM
NOEU LECT BZ TERM VOIS INDEF 2
NOEU LECT B3 TERM VOIS INDEF 2
ROTU LECT PIVV TERM
NOEU LECT B4 TERM VOIS INDEF 1
NOEU LECT B5 TERM VOIS INDEF 1
ROTU LECT PIVV TERM
NOEU LECT B6 TERM VOIS INDEF 3
NOEU LECT AA TERM VOIS INDEF 4
ROTU LECT PIVV TERM
NOEU LECT AB TERM VOIS INDEF 4
NOEU LECT AC TERM VOIS INDEF 4
NOEU LECT AD TERM VOIS INDEF 6
NOEU LECT AE TERM VOIS INDEF 7
NOEU LECT AF TERM VOIS INDEF 7
ROTU LECT PIVV TERM
ARE VR VY VZ 1
NOEU LECT PA TERM VOIS ABSENT
NOEU LECT AV TERM VOIS INDEF 7
PLLE LECT CORPS1 TERM
INER MASS 0.04325 XC 0.92193E-3 YC -0.57227E-4 ZC 0
IXX 2.139E-6
IY 2.139E-6
IZZ 2.139E-6
CORPS 7
SOLDIES 7
CORPS 1,
ELBN LECT CORPS1 TERM
BLF LECT AS B1 TERM
INER MASS 0.04325 XC 0.92193E-3 YC -0.57227E-4 ZC 0
IXX 5.431E-6
IY 5.431E-6
IZZ 5.431E-6
CORPS 3
ELBN LECT CORPS3 TERM
BLF LECT AS B3 TERM
INER MASS 0.043273 XC -0.18743E-1 YC 0.20487E-1 ZC 0
IXX 5.435E-6
IY 5.435E-6
IZZ 5.435E-6
CORPS 4
ELBN LECT CORPS4 TERM
BLF LECT AS B4 TERM
INER MASS 0.043273 XC -0.30228E-1 YC 0.12073E-1 ZC 0
IXX 5.467E-7
IY 5.467E-7
IZZ 5.467E-7
CORPS 5
ELBN LECT CORPS5 TERM
BLF LECT AS B5 TERM
INER MASS 0.03246E-1 YC 0.16631E-1 ZC 0
IXX 5.467E-7
IY 5.467E-7
IZZ 5.467E-7
CORPS 6
ELBN LECT CORPS6 TERM
BLF LECT AS B6 TERM
INER MASS 0.07056 XC -0.53246E-1 YC 0.16631E-1 ZC 0

```

```

FILE: TEST1 DPLEXUS AI WELCOME TO SHOTS W/S/SPS OPS 511
... PLANE MOTION OF MECHANISM ...
ECHO
GIBI 9 MECHANISM
TRID LINE
6
DIMENSION
ZONE 4 POUT 200 BRAD 13 PHAT 7 MEGA 12 PFLM 56 PTFL 12
PHAT 4 FTHL 1 FDS 2
SOUL 7 PFL 15 PFLB 37 BLQD 24
TILLION 126 3045 DFLAT 45 EFLDU 438 TRAV 2498
OPTION TASSER
6
CLOM
CLOM LECT CORPS1 TERM
BLF LECT AS B6 TERM
INER MASS 0.00706 XC -0.24050E-1 YC -0.35949E-2 ZC 0
IXX 5.667E-7
IY 5.667E-7
IZZ 5.667E-7
CORPS 6 '
ELBN LECT CORPS6 TERM
PLLE LECT AS B6 TERM
INER MASS 0.00706 XC -0.24050E-1 YC -0.35949E-2 ZC 0
IXX 1.169E-5
IY 1.169E-5
IZZ 1.169E-5
CORPS 6 '
ELBN LECT CORPS7 TERM
PLLE LECT AS B7 TERM
INER MASS 0.05498 XC -0.59246E-1 YC -0.10606E-1 ZC 0
IXX 1.912E-5
IY 1.912E-5
IZZ 1.912E-5
ANT1
PLVO LECT PIVV TERM
ARE VR VY VZ 1
NOEU LECT PO TERM VOIS ABSENT
NOEU LECT AI TERM VOIS INDEF 1
ROTU LECT PIVV TERM
NOEU LECT BI TERM VOIS INDEF 1
NOEU LECT AJ TERM VOIS INDEF 1
ROTU LECT PIVV TERM
NOEU LECT BZ TERM VOIS INDEF 2
NOEU LECT B3 TERM VOIS INDEF 2
ROTU LECT PIVV TERM
NOEU LECT B4 TERM VOIS INDEF 1
NOEU LECT B5 TERM VOIS INDEF 1
ROTU LECT PIVV TERM
NOEU LECT B6 TERM VOIS INDEF 3
NOEU LECT AA TERM VOIS INDEF 4
ROTU LECT PIVV TERM
NOEU LECT AB TERM VOIS INDEF 4
NOEU LECT AC TERM VOIS INDEF 4
NOEU LECT AD TERM VOIS INDEF 6
NOEU LECT AE TERM VOIS INDEF 7
NOEU LECT AF TERM VOIS INDEF 7
ROTU LECT PIVV TERM
ARE VR VY VZ 1
NOEU LECT PA TERM VOIS ABSENT
NOEU LECT AV TERM VOIS INDEF 7
PLLE LECT CORPS1 TERM
INER MASS 0.04325 XC 0.92193E-3 YC -0.57227E-4 ZC 0
IXX 2.139E-6
IY 2.139E-6
IZZ 2.139E-6
CORPS 7
SOLDIES 7
CORPS 1,
ELBN LECT CORPS1 TERM
BLF LECT AS B1 TERM
INER MASS 0.04325 XC 0.92193E-3 YC -0.57227E-4 ZC 0
IXX 5.431E-6
IY 5.431E-6
IZZ 5.431E-6
CORPS 3
ELBN LECT CORPS3 TERM
BLF LECT AS B3 TERM
INER MASS 0.043273 XC -0.18743E-1 YC 0.20487E-1 ZC 0
IXX 5.435E-6
IY 5.435E-6
IZZ 5.435E-6
CORPS 4
ELBN LECT CORPS4 TERM
BLF LECT AS B4 TERM
INER MASS 0.043273 XC -0.30228E-1 YC 0.12073E-1 ZC 0
IXX 5.467E-7
IY 5.467E-7
IZZ 5.467E-7
CORPS 5
ELBN LECT CORPS5 TERM
BLF LECT AS B5 TERM
INER MASS 0.03246E-1 YC 0.16631E-1 ZC 0
IXX 5.467E-7
IY 5.467E-7
IZZ 5.467E-7
CORPS 6
ELBN LECT CORPS6 TERM
BLF LECT AS B6 TERM
INER MASS 0.07056 XC -0.53246E-1 YC 0.16631E-1 ZC 0

```

4 - References

- [1] - P. COIFFET - Les Robots . Tome 1 : Modélisation et Commande - Hermes - Paris 1981
- [2] - P. GERMAIN - Mécanique . Tome 1 - Ellipses - Paris 1986
- [3] - J. WITTENBURG - Dynamics of systems of rigid bodies - B.G. Teubner - Stuttgart 1977
- [4] - Y. BAMBERGER - Mécanique de l'ingénieur . Tome 1 : Systèmes de corps rigides - Hermann - Paris 1981
- [5] - Y. BAMBERGER - Mécanique de l'ingénieur . Tome 2 : Milieux déformables - Hermann - Paris 1981
- [6] - H. ASADA , J.J. SLOTINE - Robot Analysis and Control - John Wiley & Sons - New York 1985
- [7] - K.S.FU , R.C. GONZALES , C.S.G. LEE - Robotics : Control, Sensing, Vision and Intelligence - MacGraw Hill - New York 1987
- [8] - Système CASTEM - Programme PLEXUS . Manuel d'utilisation - C.E.A. - Update Spring 1989
- [9] - M. LEPIAREUX , H. BUNG - Système CASTEM . Programme PLEXUS . Présentation résumée des possibilités - DEMT/SMTS/LAMS/8384 Report - Spring 1983
- [10] - Calcul et analyse des Structures en Mécanique et Thermique . Fiches de validation - DEMT/SMTS/LAMS/8275 Report - Autumn 1982
- [11] - K.J. BATHE - Finite Element Procedures in Engineering Analysis - Prentice Hall - Englewood Cliffs 1982

5 - Related publications

- [1] - Conception Mécanique , Cinématique et Dynamique des Robots - Revue Française de Mécanique n° 1987-4 - Paris . Autumn 1987
- [2] - Computer Simulation of Manipulator Dynamics using differnt Control Laws - Third International Conference on Advanced Robotics - Versailles . October 1987
- [3] - Dynamic Behaviour of Deformable and Rigid Articulated systems - Structural Mechanics in Reactor Technology 10th Conf. - Los Angeles . August 1989

Contributors and Distributors

The contributing research groups are listed according to the date of the receipt of their contributions by the editorial office.

1. Dr.-Ing. E. Pankiewicz
BMW AG, Abt. EW402
Postfach 40 02 40
8000 München 40, FRG

Marketing organization of NUBEMM:

Contact author.

Telephone (089) 3183-2606

2. Prof. M. Vukobratovic, Ph.D.
Institut Mihailo Pupin
P.O.B. 15
Beograd 11000, YUGOSLAVIA

Marketing organization of SYM:

Contact author.

Telephone 776 222

Telefax 775 870

Telex 11584 YU IMP 36

3. Prof. Dr. L. Lilov
Sofia University "Kliment Ohridski"
15 Ruski Bd.
1000 Sofia, BULGARIA

Marketing organization of CAMS:

"MECHATRONINA"

P.O.Box 76

1126 Sofia, BULGARIA

4. Prof. T.R. Kane, Ph.D.
817 Lathrop Drive
Stanford, CA 94305, USA

Marketing organization of AUTOLEV:

Online Dynamics, Inc.

1605 Honfleur Drive

Sunnyvale, CA 94087, USA

5. Prof. R.L. Huston, Ph.D.
Department of Mech. and Ind. Engineering
University of Cincinnati
Cincinnati, OH 45221, USA

Marketing organization of UCIN-DYNOCOMBS:

Contact author.

Telephone (513) 556-6131

6. Prof. Dr. J.B. Jonker
Faculty of Mechanical Engineering
Twente University of Technology
P.O. Box 217
7500 AE Enschede, THE NETHERLANDS

Marketing organization of SPACAR:

Laboratory of Engineering Mechanics
Faculty of Mechanical Engineering
Delft University of Technology
2628 CD Delft, THE NETHERLANDS

7. J.P. Frisch, Ph.D.

NASA
Goddard Space Flight Center
Greenbelt, MD 20771, USA

Marketing organization of DISCOS & NBOD:

COSMIC Program No. GSG-12810, GSG-12846
NASA's Computer Software Management and
Information Center
The University of Georgia
382 East Broad Street
Athens, GA 30603, USA

Telephone (404) 542-3265

8. Prof. E.J. Haug, Ph.D.
Center for Computer Aided Design
College of Engineering
The University of Iowa
Iowa City, IA 52242, USA

Marketing organization of DADS:

Computer Aided Design Software, Inc.
P.O.Box 203
Oakdale, IA 52319, USA
Telephone (319) 337-8968

9. Prof. Dr.-Ing. W. Schiehlen
Institute B of Mechanics
University of Stuttgart
Pfaffenwaldring 9
7000 Stuttgart 80, FRG

Marketing organization of NEWEUL:

Institute B of Mechanics
University of Stuttgart
Pfaffenwaldring 9
7000 Stuttgart 80, FRG

Telephone (0711) 685-6388
Telefax (0711) 685-6400
Telex 7255445 univ d

10. Dipl.-Ing. O. Wallrapp
German Aerospace Research Establishment (DLR)
8031 Oberpfaffenhofen, FRG

Marketing organization of MEDYNA:

T-Programm GmbH
Technische Software and Engineering
Oskar-Kalbfell-Platz 8
7410 Reutlingen, FRG

Telephone (07121) 22095
Telefax (07121) 22090
Telex 729948

11. Prof. P.Y. Willems
Universite Catholique de Louvain
Place du Levant 2
1348 Louvain-La-Neuve, BELGIUM

Marketing organization of AUTODYN & ROBOTRAN:

Contact author.

Telephone (32) 10-472597
Telefax (32) 10-452692
Telex 59315 TELHY B

12. Dipl.-Ing. W. Rulka
MAN Technologie AG
Dachauer Straße 667
8000 München 50, FRG

Marketing organization of SIMPACK:

MAN Technologie AG
Abt. SDR-3
Dachauer Straße 667
8000 München 50, FRG

Telephone (08131) 907364

13. Prof. J. Garcia de Jalon
Department of Applied Mechanics
Universidad de Navarra
20006 San Sebastian, SPAIN

Marketing organization of COMPAMM:

CORITEL S.A.
Pl. Carlos Trias Bertran
s./n. Edificio Sollube Pl. O
28020 Madrid, SPAIN

Telephone (1) 59 73335
Telefax (1) 5970109

14. Prof. B. Paul, Ph.D.
Department of Mech.Eng. Appl.Mech.
University of Pennsylvania
111 Towne Building
Philadelphia, PA 19014-6315, USA

Marketing organization of DYMAG & DYSPAM:

B. Paul Associates
204 Dodds Lane
Princeton, NJ 08540, USA

15. Prof. Dr.-Ing. J. Wittenburg
Institut für Technische Mechanik
Universität Karlsruhe
Kaiserstraße 12
7500 Karlsruhe 1, FRG

Marketing organization of MESA VERDE:

Ingenieurgemeinschaft
Prof. Dr.-Ing. R. Gnädler
Kaiserallee 111
7500 Karlsruhe 21

Telephone (0721) 842297

16. R.R. Ryan, Ph.D.
Mechanical Dynamics, Inc.
3055 Plymouth Road
Ann Arbor, MI 48105-3203, USA

Marketing organization of ADAMS:

Mechanical Dynamics, Inc.
3055 Plymouth Road
Ann Arbor, MI 48105-3203, USA

Telephone (313) 994-3800
Telefax (313) 994 6418

17. Prof. A. Barraco
E.N.S.A.M.
151, boulevard de l'Hopital
75640 Paris Cedex 13

Marketing organization of PLEXUS:

Contact author, Telephone (1) 43364955

Index

- ADAMS 361
applications 6,9
AUTODYN 225
AUTOLEV 81
Avello, A. 285
Barraco, A. 403
Bekjarov, B. 61
CAMS 61
COMPAMM 285
computation methods 4,7
computers 9
coordinates 4,7
Cuny, B. 403
DADS 161
DISCOS 145
distribution of software 6
dynamical methods 4,8
DYMAC 305
DYNOCOMBS 103
DYSPAM 323
elements 4,7
equations 5,8
Frisch, H.P. 145
Führer, C. 203
Garcia-Alonso, A. 285
Garcia de Jalon, J. 285
Haug, E.J. 161
Huston, R.L. 103
installations 9
Jimenez, J.M. 285
Jonker, J.B. 123
Kamman, J.W. 103
Kane, T.R. 81
King, T.P. 103
Kircanski, M. 37
Kircanski, N. 37
Kreuzer, E.J. 181
languages 9
Levinson, D.A. 81
Lilov, L. 61
Lorer, M. 61
Maes, P. 225
MEDYNA 203
Meijaard, J.P. 123
MESA VERDE 341
NBOD 145
NEWEUL 181
NUBEMM 21
operating systems 9
overview 3
Pankiewicz, E. 21
Paul, B. 305
PLEXUS 403
postprocessing 6,9
preprocessing 5,8
ROBOTRAN 246
Rulka, W. 265
Ryan, R.R. 361
Samin, J.C. 225
Schaffa, R. 305
Schiehlen, W. 181
Schmidt, A. 341

signal analysis 5,8
SIMPACK 265
simulation 5,8
Smith, R.L. 161
software engineering 6
SPACAR 123
SYM 37
tables of information 6
test example mechanism 10
test example robot 16
Timcenko, A. 37
topology 3,7
UCIN-DYNOCOMBS 103
Vukobratovic, M. 37
Wallrapp, O. 203
Willems, P.Y. 225
Wittenburg, J. 341
Wolz, U. 341