AN OBJECT-ORIENTED DYNAMICS SIMULATOR

by

JONG-PIL PARK, B.S.M.E., M.S.

A DISSERTATION

IN

MECHANICAL ENGINEERING

Submitted to the Graduate Faculty
of Texas Tech University in
Partial Fulfillment of
the Requirements for
the Degree of

DOCTOR OF PHILOSOPHY

Approved

Chairperson of the Committee

Accepted

Dean of the Graduate School

August, 1991

HL
801
T3
1991
No. 97
Cop 2

Copyright © 1991, Jong-Pil Park

ACKNOWLEDGEMENTS

I am very grateful for the guidance and support of my advisor Dr. Aik-Siong Koh, without whom this work would not have been possible. I also wish to thank the other members of my doctoral committee, Dr. Herbert J. Carper, Dr. Atila Ertas, Dr. Timothy T. Maxwell, and Dr. James W. Oler.

I gratefully acknowledge the research and scholarship support from Ford Motor Company. The financial support has been immensely important.

I also would like to acknowledge the strong support from the president and staffs of Defense Quality Assurance Agency in Korea.

Finally, my gratitude is extended to my family for their enduring encouragement and understanding.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	ii
ABSTRACT	vii
LIST OF TABLES	viii
LIST OF FIGURES.....	ix
LIST OF SYMBOLS	xiii
CHAPTER	
I. INTRODUCTION	1
1.1 Introductory Comments.....	1
1.2 Survey of Existing Dynamics Simulators	2
1.3 Object-Oriented Programming (OOP)	3
1.3.1 Background and Objectives	3
1.3.2 Concepts and Characteristics of OOP	4
1.3.3 Structured Programming Versus OOP	6
1.3.4 Features of Smalltalk-80	8
1.4 Scope of the Research	9
II. SIMULATION OF VEHICLE RESPONSE TO CROSSWINDS.....	16
2.1 Introductory Comments.....	16
2.2 Simulation Scheme.....	16
2.3 Factors Affecting Crosswind Sensitivity.....	17
2.3.1 Varying Aerodynamic Coefficients	17
2.3.2 Comparing Taurus Sedan and Taurus Wagon.....	18
2.3.2.1 Yaw, Pitch, and Roll Responses.....	18
2.3.2.2 Tire Forces and Moments During Crosswind.	19
2.3.3 Parametric Study on Suspension and Aerodynamics.....	19

2.3.3.1 Dependency of Response on Suspension System	20
2.3.3.2 Further Study on Aerodynamic Coefficients	20
2.3.4 Interpretation of Vehicle Responses	21
2.4 Conclusions of Simulation Study	22
III. FORMULATION OF SYSTEM EQUATIONS	39
3.1 Introductory Comments.....	39
3.2 System Elements	39
3.3 Coordinate System	40
3.4 System Equations	41
3.4.1 State Variables.....	41
3.4.2 Lagrange's Equation	41
3.4.3 Force equations	48
3.4.3.1 Virtual Work and Generalized Forces.....	48
3.4.3.2 External Forces and Torques.....	50
3.4.4 Constraint Equations and Constraint Forces	53
3.4.4.1 At-Point Constraint	53
3.4.4.2 In-Plane Constraint.....	53
3.4.4.3 Perpendicular Constraint	54
3.4.4.4 Generalized Constraint Forces	55
IV. SOLUTION SCHEME OF SYSTEM EQUATIONS.....	74
4.1 Introductory Comments.....	74
4.2 Linearization of System Equations	74
4.3 Corrector Method	76
4.3.1 Gear Integration Formulas	76
4.3.2 Iteration Scheme.....	77
4.4 Predictor-Corrector Schemes	80

4.4.1 Predictor-Corrector I.....	80
4.4.2 Predictor-Corrector II.....	81
V. INITIAL CONDITIONS ANALYSIS.....	85
5.1 Introductory Comments.....	85
5.2 Position Initial Conditions.....	85
5.3 Velocity Initial Conditions.....	87
5.4 Acceleration Initial Conditions.....	89
VI. OBJECT-ORIENTED DYNAMICS SIMULATOR (OODS).....	91
6.1 Introductory Comments.....	91
6.2 Program Organization.....	91
6.2.1 Classes of OODS.....	91
6.2.2 Message Sendings.....	92
6.2.3 Input to OODS	93
6.3 Program Test Runs.....	95
6.3.1 One-Body Systems.....	96
6.3.1.1 Bar Pendulum in Plane Motion.....	96
6.3.1.2 Bar Pendulum in Conical Motion.....	97
6.3.1.3 Bar-Spring System.....	98
6.3.1.4 Bar-Spring-Damper System.....	98
6.3.1.5 Unconstrained Body.....	98
6.3.2 Two-Body Systems.....	99
6.3.2.1 Closed Two-Bar System.....	99
6.3.2.2 Double Pendulum.....	99
6.3.2.3 Unconstrained Two-Body System	100
6.4 Comments on Program Test Results.....	100
VII. CONCLUSIONS AND RECOMMENDATIONS.....	157

7.1 Conclusions	157
7.2 Recommendations	158
BIBLIOGRAPHY	159
APPENDICES	
A. EXPANDED EXPRESSIONS	161
B. PROGRAMS	181

ABSTRACT

An Object-Oriented Dynamics Simulator (OODS) has been developed by incorporating dynamics theory and numerical methods into the concepts of object-oriented programming (OOP). OODS is a response to the need for easily readable, reusable, and extensible general purpose multibody simulation programs.

In the development of the dynamics theory, emphasis is placed on the automatic formulation of system equations and their numerical solutions. The system equations consist of the equations of motion, generalized momentum definitions, constraint equations, and generalized force definitions. These equations are highly nonlinear and loosely coupled differential-algebraic equations. The Newton-Raphson method and a stiffly-stable integration method are combined to build a corrector method. Two standard predictor methods are adopted to complete the solution scheme of the system equations.

In the development of the new program, the above simulation algorithms are organized using objects and classes to take advantage of the power of OOP. The new program has been written in Smalltalk-80 which is one of the pure OOP languages. It has been shown that the new program is easy to understand, debug, and extend.

Test runs for several types of one- and two-body systems have been successfully completed using OODS. The test results reveal that OODS is reliable in its formulation of system equations and their numerical solutions. It has been also concluded that OODS is applicable to larger systems because the simulation algorithms and programming of OODS have been designed in general terms. Finally, some recommendations are made for potential improvement of OODS. Specific applications in mechanical engineering practice and education are encouraging.

LIST OF TABLES

2.1	Effect of Aerodynamic Coefficients on Taurus Sedan.....	24
2.2	Simulation of Taurus Sedan with Wagon's Aerodynamic Coefficients.....	25
3.1	Joint Types and Constraint Equations.....	58
4.1	Gear Coefficients.....	83
4.2	Interpolation Predictor Coefficients.....	83

LIST OF FIGURES

1.1	Unit of Modularity	12
1.2	System Browser in Smalltalk-80 (Class Selection).....	13
1.3	System Browser in Smalltalk-80 (Method Selection).....	14
1.4	Debugger Window in Smalltalk-80.....	15
2.1	Crosswind Simulation Scheme and Simulated Wind Input	26
2.2	Vehicle Axes and Wind Velocities	27
2.3	Yaw of Taurus Sedan and Wagon.....	28
2.4	Pitch of Taurus Sedan and Wagon	29
2.5	Roll of Taurus Sedan and Wagon	30
2.6	Force and Moment Vectors on Taurus Sedan	31
2.7	Force and Moment Vectors on Taurus Wagon	32
2.8	Net Lateral Front Tire Force of Taurus Sedan and Wagon.....	33
2.9	Force and Moment Vectors on Taurus Sedan at 0.8 Seconds.....	34
2.10	Force and Moment Vectors on Taurus Sedan at 1.2 Seconds.....	35
2.11	Force and Moment Vectors on Taurus Sedan at 1.7 Seconds.....	36
2.12	Force and Moment Vectors on Taurus Wagon at 1.2 Seconds	37
2.13	Force and Moment Vectors on Taurus Wagon at 1.7 Seconds	38
3.1	Multibody System	59
3.2	Axes Systems	60
3.3	Angular Velocity in Euler Axes System	61
3.4	External Force and Torque	62
3.5	Gravitational Force.....	63
3.6	Action-Only Force and Torque	64
3.7	Action-Reaction Force	65

3.8	Action-Reaction Torques	66
3.9	Linear Spring and Damper	67
3.10	Spherical Joint and Its Constraints	68
3.11	Universal Joint and Its Constraints.....	69
3.12	Revolute Joint and Its Constraints.....	70
3.13	Translational Joint and Its Constraints	71
3.14	Cylindrical Joint and Its Constraints	72
3.15	Planar Joint and Its Constraints.....	73
4.1	Predictor-Corrector Scheme	84
6.1	Class Hierarchy within OODS	102
6.2	Problem Solving in OODS.....	103
6.3	Message Sendings Diagram	104
6.4	Bar Pendulum in Plane Motion	105
6.5	Input Data of Bar Pendulum in Plane Motion.....	106
6.6	Effect of Integration Orders on Bar Pendulum	107
6.7	Effect of Step Sizes on Bar Pendulum	108
6.8	Y, Z Responses of Bar Pendulum in Plane Motion.....	109
6.9	Φ Response of Bar Pendulum in Plane Motion	110
6.10	$\dot{Y}, \dot{Z}$ Responses of Bar Pendulum in Plane Motion	111
6.11	$\dot{\Psi}$ Response of Bar Pendulum in Plane Motion	112
6.12	Bar pendulum in Conical Motion.....	113
6.13	Input Data of Bar Pendulum in Conical Motion	114
6.14	X, Y Responses of Bar Pendulum in Conical Motion.....	115
6.15	$\dot{X}, \dot{Y}$ Responses of Bar Pendulum in Conical Motion	116
6.16	Constraint Forces on Bar Pendulum in Conical Motion	117
6.17	Bar-Spring System	118

6.18	Input Data of Bar-Spring System.....	119
6.19	Z, Φ, Θ, Ψ Responses of Bar-Spring System.....	120
6.20	$\dot{Z}, \ddot{Z}$ Responses of Bar-Spring System	121
6.21	Constraint Forces of Bar-Spring System.....	122
6.22	Bar-Spring-Damper System	123
6.23	Input Data of Bar-Spring-Damper System.....	124
6.24	Z Response of Bar-Spring-Damper System.....	125
6.25	$\dot{Z}, \ddot{Z}$ Responses of Bar-Spring-Damper System	126
6.26	Unconstrained Body	127
6.27	Input Data of Unconstrained Body.....	128
6.28	Y, Z Responses of Unconstrained Body	129
6.29	Z/Y Response of Unconstrained Body.....	130
6.30	$\dot{Z}, \ddot{Z}$ Responses of Unconstrained Body	131
6.31	Closed Two-Bar System.....	132
6.32	Input Data of Closed Two-Bar System	133
6.33	X, Y Responses of Closed Two-Bar System.....	134
6.34	Z, Φ Responses of Closed Two-Bar System.....	135
6.35	Θ, Ψ Responses of Closed Two-Bar System.....	136
6.36	$\dot{X}, \dot{Z}$ Responses of Closed Two-Bar System	137
6.37	$\dot{\Theta}$ Response of Closed Two-Bar System	138
6.38	$\ddot{X}, \ddot{Z}$ Responses of Closed Two-Bar System	139
6.39	Constraint Forces of Closed Two-Bar System.....	140
6.40	Constraint Force Acting on Bar 1 of Closed Two-Bar System.....	141
6.41	Double Pendulum (Mode 1).....	142
6.42	Input Data of Double Pendulum (Mode 1).....	143
6.43	Y, Ψ Responses of Double Pendulum (Mode 1).....	144

6.44	$\dot{Y}, \dot{\Psi}$ Responses of Double Pendulum (Mode 1).....	145
6.45	$\ddot{Y}$ Response of Double Pendulum (Mode 1).....	146
6.46	Double Pendulum (Mode 2).....	147
6.47	Input Data of Double Pendulum (Mode 2).....	148
6.48	Y Response of Double Pendulum (Mode 2)	149
6.49	Ψ Response of Double Pendulum (Mode 2)	150
6.50	$\dot{Y}, \dot{\Psi}$ Responses of Double Pendulum (Mode 2).....	151
6.51	$\ddot{Y}, \ddot{\Psi}$ Responses of Double Pendulum (Mode 2).....	152
6.52	Unconstrained Two-Body System	153
6.53	Input Data of Unconstrained Two-Body System	154
6.54	Y, Z Responses of Unconstrained Two-Body System	155
6.55	Z/Y Response of Unconstrained Two-Body System	156

LIST OF SYMBOLS

c	Iteration counter if used as a subscript, otherwise damping coefficient
C_F 's	Constraint force components C_{F_x} , C_{F_y} , and C_{F_z}
C_M 's	Constraint torque components C_{M_x} , C_{M_y} , and C_{M_z}
E_1	Objective function for position initial conditions
E_2	Objective function for velocity initial conditions
F_{i_f}	An arbitrary applied force
G	Constraint equations of system
G_{i_c}	A constraint equation
I, J, K	Unit vectors of ground frame
i, j, k	Unit vectors of part frame
i', j', k'	Unit vectors of marker frame
i_c	Index for constraint equations
i_f	Index for applied forces on parts
i_p	Index for parts
i_q	Index for generalized coordinates
i_s	Index for state variables
i_t	Index for applied torques on parts
I_{dam}	Damper associated with marker I
I_m	Marker I
I_p	Part associated with marker I
I_{spr}	Spring associated with marker I
k	integration order or interpolation order if used as a subscript, otherwise spring constant
K, I', K''	Unit vectors of Euler axes

l_{IJ}	Relative position of marker I with respect to marker J
l_{IJ}	magnitude of l_{IJ}
L	Implicit function for generalized force definitions of system
L_{i_p}	Implicit function for generalized force definitions of part i_p
m_{i_p}	mass of part i_p
M	Implicit function for equations of motion of system
M_{i_p}	Implicit function for equations of motion of part i_p
n_c	Number of system constraint equations
n_f	Number of applied forces on parts
n_p	Number of parts
n_q	Number of system generalized coordinates ($= 6n_p$)
n_s	Number of system state variables ($= 16n_p + n_c$)
n_t	Number of applied torques on parts
P	Implicit function for generalized momentum definitions of system
P_{i_p}	Implicit function for generalized momentum definitions of part i_p
q	Generalized coordinates of system ($= \{q\}$)
q_{i_p}	Generalized coordinates of part i_p containing $X_{i_p}, Y_{i_p}, Z_{i_p}, \Phi_{i_p}, \Theta_{i_p}, \Psi_{i_p}$
Q	Generalized forces of system ($= \{Q\}$)
Q_{i_p}	Generalized forces of part i_p containing $Q_{X_{i_p}}, Q_{Y_{i_p}}, Q_{Z_{i_p}}, Q_{\Phi_{i_p}}, Q_{\Theta_{i_p}}, Q_{\Psi_{i_p}}$
T_{i_p}	An arbitrary applied torque
T	Kinetic energy of system
u	Generalized momenta of system ($= \{u\}$)
u_{i_p}	Generalized momenta of part i_p containing $u_{X_{i_p}}, u_{Y_{i_p}}, u_{Z_{i_p}}, u_{\Phi_{i_p}}, u_{\Theta_{i_p}}, u_{\Psi_{i_p}}$
V_{i_p}	Absolute velocity of center-of-mass of part i_p
x, y, z	Cartesian coordinates of part frame

X,Y,Z	Cartesian coordinates of ground frame
x',y',z'	Cartesian coordinates of marker frame
ϕ,θ,ψ	Euler angles measured from part frame
Φ,Θ,Ψ	Euler angles measured from ground frame
λ	Lagrange multipliers of system ($= \{\lambda\}$)
ω_{i_p}	Absolute angular velocity of part i_p
$[A]_{i_p}$	Transformation matrix from i_p part frame to ground frame
$[A]_{I_m}$	Transformation matrix from I marker frame to i_p part frame
$[B]_{i_p}$	Transformation matrix from Euler axes to i_p part frame
$[C]_{i_p}$	Transformation matrix from generalized speeds to generalized momenta of part i_p
$[J]_{i_p}$	Principal moment-of-inertia tensor containing J_{xx}, J_{yy}, J_{zz} of part i_p
$[I]$	Identity matrix
$[W_1]$	Weight matrix for initial position conditions
$[W_2]$	Weight matrix for initial velocity conditions
$[]^T$	Transpose of matrix

CHAPTER I

INTRODUCTION

1.1 Introductory Comments

The dynamic analysis of interconnecting rigid bodies that undergo large displacements is a research area with applications in many engineering fields. These fields include the study of vehicles, robots, and machines. Dynamic analysis allows better understanding of the motions and forces involved, and permits faster, better, and cheaper designs. This need has led to the development of several general purpose multibody dynamics simulators.

These dynamics simulators are capable of assembling systems with many parts connected by joints and acted upon by internal or external forces. The exact equations governing the systems are automatically formulated and solved numerically for the responses of the systems. Time histories of positions, velocities, accelerations, and constraint forces are easily produced. These systems are quite general. They can handle many degrees of freedom in two or three spatial dimensions. Kinematic, static, and dynamic analyses are usually possible. Real joints like spherical, revolute, cylindrical, translational, universal, and planar joints are accurately simulated as are other components like springs and dampers. Arbitrarily prescribed forces and motions can be accepted by the simulators too. The computed responses are accurate to within the assumptions of the models and numerical errors.

The first general purpose dynamics simulators appeared in the early 1970's. Those having major impact were DRAM [Chace 1971], IMP [Sheth 1972], and ADAMS [Orlandea 1973]. Numerous programs have since developed, but they generally represent variations on the above algorithms [Liegeois 1976, Williams 1979, Nikravesh 1982]. Some

comparison studies on existing programs have been reported [Haug 1984, Unda 1987, Schiehlen 1990].

1.2 Survey of Existing Dynamics Simulators

The analysis methods used in existing dynamics simulators may be categorized by their use of relative versus absolute coordinates, sparse versus dense equations, and stiff versus non-stiff solvers.

A major difference between the analysis methods lies in the type of coordinate systems chosen to specify the position and orientation of every element in the multibody system. Relative or Lagrangian coordinates define the position of each body relative to the neighboring one in the kinematic chain [Paul 1979]. This method requires the same number of coordinates as the degrees of freedom in open-loop mechanisms. In closed-loop mechanisms the number of coordinates exceeds the degrees of freedom because of the constraint equations arising from loop closures. Major disadvantages of relative coordinates are that they do not directly determine the positions of elements, and they also need topological preprocessing to identify the independent loops [Sheth 1972]. Moreover, the computer implementation of these coordinates is rather complicated. By contrast, absolute or Cartesian coordinates determine the position and orientation of each body relative to an inertial reference frame [Haug 1984]. The number of coordinates is much higher than the degrees of freedom, but the formulation is simpler than with relative coordinates. In this case, constraint equations arise from the joints between contiguous bodies.

Another major difference lies in the structure of the coefficient matrices in the system equations. A matrix is termed sparse if less than 5% of the elements in the matrix are non-zeros. Sparse coefficient matrices are produced when the system equations are formulated using absolute coordinates. In addition, the number of equations is typically large. Some sparse matrix techniques can be applied to enhance program efficiency [Chua 1975].

However, solving large number of equations is inherently time-consuming even when advantage is taken of matrix sparsity. In fact, sparse matrix formulations put emphasis on program generality rather than program efficiency. By contrast, formulations based on relative coordinates have dense coefficient matrices. In general, for a given problem, the number of equations in dense matrix formulations is much smaller than in sparse matrix formulations. Unfortunately, dense matrix formulations require program codes that are more difficult to understand and modify.

Finally, a major difference between the dynamics simulators is whether the numerical integrators in the programs can solve stiff systems or not. A system is stiff if it possesses widely varying natural frequencies. The ratio of the highest frequency to the lowest frequency is called the stiffness ratio. When the stiffness ratio is greater than 10, the system may be called stiff [Rampalli 1987]. Stiff integrators ignore the high frequencies in the system without any loss of stability. An example of stiff integrators is Gear's algorithm [Gear 1971] which is a multistep method with variable order and step size. Although more stable, stiff integrators are more complicated to code. By contrast, non-stiff integrators are easy to code, but the step size must be smaller than the smallest natural period. For stability reasons, non-stiff integrators must track the highest frequency in the system even though it is dormant. Examples of non-stiff integrators include the multistep Adams-Bashforth and Adams-Moulton predictor-corrector formulas. Stability theory, especially for stiff systems, is discussed comprehensively by Gear [1971] and Chua [1975].

1.3 Object-Oriented Programming (OOP)

1.3.1 Background and Objectives

An extensive dynamic simulation study was done using ADAMS (Automatic Dynamic Analysis of Mechanical Systems) at the beginning of the author's PhD program, and the results have been reported by Koh et al. [1989]. During the study, the author

gained insight into computer-aided dynamics analysis and was also motivated to develop a dynamics simulator using more innovative programming techniques. The major objectives of this research are

- (1) To develop the theory and numerical methods for multibody dynamics,
- (2) To incorporate this theory with object-oriented programming (OOP) concepts into an Object-Oriented Dynamics Simulator (OODS).

Existing dynamics simulators are coded using procedural or structural programming languages and concepts. Regardless of the simulation capabilities and limitations of existing simulators, the program codes are usually inaccessible or not well understandable to the users. This deficiency can be alleviated with OOP. The application of OOP in engineering is quite new, but some applications have been developed in other areas such as database, graphics, information system, etc. The OOP language used in this research is Smalltalk-80 [ParcPlace 1988]. To the best of the author's knowledge, OODS is the first dynamics simulator to use OOP concepts fully.

1.3.2 Concepts and Characteristics of OOP

OOP tries to create, in software, entities which are exact or good replicas of real world entities. This similarity will allow humans to apply their natural thinking modes to programming, thus improving program reliability.

OOP introduces the concepts of objects, message sendings, and class hierarchies to programming. An object represents a collection of data and a set of operations permitted on those data. Therefore, an OOP object can be made to behave like a real world object. This is the power of OOP.

To get an object to act, a message is sent to the object. The object will respond to the message if it finds within itself a procedure or method to direct its response or behavior. Otherwise, the message is ignored. Programming involves creating objects with suitable

behaviors and writing a script of messages for the objects so that the result of the objects' behaviors produces the desired computation.

To further improve programming productivity, OOP encourages objects to be grouped into classes according to their behaviors. The objects have the properties defined by their classes. Some of these properties are private to each object, while others are shared with other objects of the same class. The private properties are called instance variables and the shared properties are called class variables. The resulting class hierarchy helps the programmer organize the code, and it makes the program more understandable and easier to modify.

In more formal terms, the power of OOP comes from its ability to support abstraction, encapsulation, inheritance, and polymorphism [Pinson 1988]. Abstraction is understanding a complex behavior or object in simpler terms. A programmer does not have to go into any more detail than necessary to comprehend his problem. The behavior of an object can be understood purely from the messages it responds to, without need to see the procedures for each message. Therefore, using and understanding an abstraction enables higher-level human functioning.

Encapsulation is the complete description of all properties, features, and messages in a single entity. The unit of encapsulation is an object which is an instance of the corresponding class. Encapsulation provides a way of protecting information within a module. A collection of data structures and procedures is accessible only from within the scope of the module. Therefore, without sending a relevant message, the object cannot execute the message. This is important for ensuring reliability and modularity of a program by reducing interdependencies between the different behaviors of the different objects defined in the program.

Inheritance allows the subclasses to inherit the instance variables, class variables, and methods of the superclasses. When an instance of a class receives a message, a method

corresponding to that message is searched within the class protocol. If no correspondence is found, the methods in the superclasses are searched at ever higher levels until a suitable method is found. In addition, the subclasses can add new class variables, instance variables, and methods of their own without affecting the superclasses. This is important for ensuring reusability by extending program elements from simpler ones. Inheritance also makes the code easier to maintain and reduces redundant codes.

Polymorphism makes it possible to send the same message to different objects and have each object respond with its own behavior. Polymorphic redefinition is possible anywhere, but it is most useful when a subclass overrides the method defined in its superclasses. The ability to use the same message for similar behaviors on different objects is consistent with the way humans think in problem solving and helps programmers and users avoid errors in logic.

1.3.3 Structured Programming Versus OOP

Most computer languages, including Fortran, Pascal, and C, are structured programming languages. They are familiar to most programmers and are seen to be natural and suitable for problem solving. Closer inspection, however, reveals that structured programming is not only awkward, but conceptually difficult to follow [Micallef 1988].

Structured languages often lead to program structures radically different from the structures of the problems to be solved. The reason is that the data and the functions that operate on the data are separated. The data is passive and can be manipulated in any way regardless of the validity of the manipulation. In addition, structured languages do not discourage creation of poor functions that do not check the validity of the data adequately.

A programmer using structured languages must either map the problem domain into a set of functions, and then define the data structures needed by the functions, or he can map the problem domain to the data, and then define functions that transform the input data to

the output data. By contrast, OOP facilitates the creation of program components that closely parallel the application domain. That is, OOP treats functions and data as indivisible aspects of objects in the problem domain. Many programs can be designed by directly identifying the objects in the problem domain, and implementing the objects' behavior in the computer. Thus, the resulting programs are more natural, i.e., more readable and understandable.

Reusability is a major goal in software engineering, and it means that program components can be combined and modified easily to make new applications. The necessity of reusability is inherent in computer programming itself because most programs need to perform the basic tasks such as sorting, searching, reading, writing, comparing, and allocating. A survey shows that more than 85open-loop of a new program is used to carry out these basic tasks [Jones 1984]. High reusability ensures that the codes for the basic tasks need not be written from scratch.

Many structured programming languages have included features to promote and support their reusability by providing certain types of modularity. The unit of modularity is usually the procedure, but the modes of communication between procedures are highly variable. This uncertainty of communication makes it difficult to modify a module. Finally, this modularity applies to the functions only, not the data.

By contrast, OOP permits highly modular programming by describing problems in terms of objects and classes. Objects in turn respond only to distinct messages. Thus, modularity of functions and data is achieved. The degree of reusability of OOP codes is very high and is inherent in the language. Therefore, OOP may reduce the high cost of building large software systems. Figure 1.1 shows the modular units of object-oriented and structured languages.

OOP has some disadvantages also. The most often debated one is run-time cost. A message sending takes more time than a straight procedure call. Some studies have shown

that a message sending needs approximately 1.75 times the cost of a standard procedure call [Cox 1983, Cox 1984]. But, actual differences in program execution speed may not prove to be very significant because a message sending usually accomplishes more work than a procedure call. Another potential problem is that a programmer must learn an extensive library of classes before becoming proficient in an OOP environment. As a result, OOP may be more dependent on good documentation and programming tools.

1.3.4 Features of Smalltalk-80

Smalltalk-80 is more than a purely OOP language; it is an integrated programming environment with a great variety of tools for more convenient software development. Many articles and books have been written describing the features of Smalltalk-80 [Tesler 1981, Goldberg 1984, Kaehler 1986, Diederich 1987]. In this section, only two important programming tools are explained to give the reader an idea of the programming environment offered.

The system browser is the main interface for programming in Smalltalk-80. It contains several panes, each with its own menu of actions (Figures 1.2-1.3). The top four panes contain class categories, classes, message categories, and messages, respectively. The bottom pane displays either the corresponding class definition when only a class is selected (Figure 1.2) or the corresponding codes when a method is selected (Figure 1.3). Classes and methods are created and modified using the system browser.

A method in Smalltalk-80 is created using the system browser in the following four steps:

- (1) From the first top pane, select a class category for the new method. If a new category is needed, select the menu option 'add category' and type the name of the category when prompted.

- (2) From the second top pane, select a class for the new method. If a new class is needed, type the class definition in the bottom pane (Figure 1.2). When done, select the menu option 'accept'.
- (3) From the third top pane, select a message category for the new method. If a new category is needed, select the menu option 'add protocol' and type the name of the category when prompted.
- (4) Define the new method in the bottom pane by typing the method name followed by a set of executable statements. When done, select the menu option 'accept'. Now, the new method name (message) appears in the fourth top pane, and the message is ready to be sent to relevant objects for execution.

A debugger window appears whenever an error or a breakpoint is encountered during method execution. The debugger window has several panes also (Figure 1.4). The top pane lists the classes with the corresponding methods that are not completed due to the halt. The middle pane contains the codes of the method selected on the top pane. The location of the halt in the method is highlighted. The two bottom panes contain the status or values of the instance variables and temporary variables used in the selected message.

1.4 Scope of the Research

A general and easily extensible dynamics simulator has been developed. The simulator is able to formulate the system equations and solves them numerically. The simulation algorithms are organized using objects and classes to take full advantages of OOP.

In Chapter II, the simulation study on crosswind sensitivity of passenger cars is covered. The chosen vehicles are the Taurus sedan and the Taurus station wagon. The responses of the vehicles are simulated for many variations and combinations of

aerodynamic coefficients and suspension systems. The responses of the two vehicles are compared and summarized. Finally, conclusions of the study are presented.

The experience gained in this simulation study provided the motivation for the development of an easily accessible and reusable Object-Oriented Dynamics Simulator (OODS).

In Chapter III, the theoretical foundations for OODS are established. A set of state variables is defined to describe system behaviors. These state variables include six generalized coordinates, six generalized momenta, and six generalized forces for every part. Lagrange multipliers form the additional state variables.

The system equations consist of the equations of motion, generalized momentum definitions, constraint equations, and generalized force definitions. The equations of motion are derived using Lagrange's equation. These equations are reduced to first order by defining the generalized momenta. Next, three fundamental constraint equations called At-Point, In-Plane, and Perpendicular are defined. These constraints are combined to represent the joints available in OODS. OODS does not reduce the number of coordinates down to a minimal set of independent coordinates, but rather appends all the constraint equations to the system equations. Finally, the generalized forces are derived using the principle of virtual work. Contributions of gravitational force, user-defined constant forces and torques, translational spring forces and damping forces to these generalized forces are given specifically.

The system equations formulated govern the dynamic system completely. The equations are highly nonlinear and loosely coupled differential-algebraic equations. Thus, their solutions rely on numerical integration.

In Chapter IV, the system equations are linearized using the Newton-Raphson method and are expressed in a matrix equation. This matrix equation consists of the coefficient matrix, state vector column, and a right-hand side column. The coefficient

matrix and right-hand side column are called the system Jacobian and error column, respectively. Next, the matrix equation is incorporated with a stiffly-stable integration algorithm known as Gear's method to build a corrector algorithm. Finally, two predictor methods (Euler formula and an interpolation formula) are introduced. As a result, two predictor-corrector schemes are established.

In Chapter V, the initial condition analysis is discussed in detail. The theory is useful to assure that the input data are compatible with the real system. Compatible initial positions, velocities, and accelerations that satisfy the equations of motion and constraint equations are produced. These initial conditions are necessary to begin the predictor-corrector schemes.

In Chapter VI, the organization of the Smalltalk-80 classes defined for OODS is explained using diagrams of class hierarchy and message sendings. The way to input a mechanical system into OODS and how to obtain the system responses are then shown. Some test runs using one- and two-body systems were done, and plots of position, velocity, acceleration, and constraint force were produced. The simulation models were selected such that all constraints available in OODS were involved and checked. An attempt was also made to independently check some computations in OODS using a spreadsheet.

The test results are in good agreement with the numerically or analytically exact solutions. This proves that the new program is reliable in terms of its formulation of system equations and numerical solution algorithms. The author is confident that OODS will simulate systems with more than two bodies because the formulation and programming have been done in general terms. Only the long computation time required of a microcomputer has limited an actual simulation.

Finally, in Chapter VII conclusions of the research are made along with the simulation capabilities and limitations of OODS. Some potential improvements are suggested as continuing research.

Figure 1.1: Unit of Modularity

OS-Streaming	Marker	formulation	
OS-Unix	Part	initialize	
OS-Dos	Request	private 1	
OS-Mac	Spring	private 2	
System-Printing	System	predictor-correct	
Multibody System	instance class	initial condition	
Object subclass: #System instanceVariableNames: 'modelName ' classVariableNames: 'DamperDictionary EndTime ForceDictionary GearCoeffArray GearConstant Gravity InitialTime IntegrationOrder JointDictionary Lambda MatrixdGdq MatrixdPdq MaxIntegrationOrder NoOfSteps NumberOfConstraints NumberOfForces NumberOfParts NumberOfRequests NumberOfSprings PartDictionary' category: 'Multibody System'			

Figure 1.2: System Browser in Smalltalk-80 (Class Selection)

OS-Streaming	Marker	formulation	-----
OS-Unix	Part	initialize	accError
OS-Dos	Request	private 1	accJacobian
OS-Mac	Spring	private 2	analyzeAcc
System-Printing	System	predictor-corre	analyzePos
Multibody Syst	instance class	initial condition	analyzeVel

analyzePos

"starts the position initial conditions analysis."


```
| q c arrayq0 posLambda maxIterations matrixA arrayY
arrayX arraydelq arrayq1 |
```


"----- sets up the initial {q0} for iterations -----"

q ← 6*NumberOfParts.

c ← NumberOfConstraints.

Figure 1.3: System Browser in Smalltalk-80 (Method Selection)

System>>put:into:at:			
System>>analyzePos			
System>>evaluateIC			
analyzePos "starts the position initial conditions analysis." q c arrayq0 posLambda maxIterations matrixA arrayY arrayX arraydelq arrayq1 "----- sets up the initial {q0} for iterations -----" q ← 6*NumberOfParts.			
----- self modelN:	'Bar Pendulum in 3-D'	q arrayq0	3

Figure 1.4: Debugger Window in Smalltalk-80

CHAPTER II

SIMULATION OF VEHICLE RESPONSE TO CROSSWINDS

2.1 Introductory Comments

An extensive simulation study was done at the beginning of this research. The objective of the simulation study was to better understand how the crosswind sensitivity of passenger cars is determined by their aerodynamic coefficients and suspension systems. The factors that are dominant in affecting the vehicle response to crosswinds were determined. Also, methods to reduce the crosswind sensitivity of a car were investigated.

The simulations were done on ADAMS which is a commercially available dynamics simulator. The ADAMS input datasets for the Taurus Sedan and Taurus Station Wagon were provided by Ford Motor Company. But, the subroutines for aerodynamic loads had to be written and incorporated here at Texas Tech University. This chapter summarizes the important results and conclusions of this practical simulation study.

2.2 Simulation Scheme

In the simulation, the car travels at a constant speed on a flat and straight road with its steering wheel held fixed at a central position. No driver reaction is included. A steady wind blows across a certain portion of the road at a constant angle. When the car hits the crosswind, it is allowed to veer naturally (Figure 2.1).

The car speed V_{car} is 70 mph. The crosswind wind speed input as a function of time is shown in Figure 2.1. The absolute crosswind wind speed V_w and the absolute wind angle ψ_{abs} are 23.97 mph and 60° , respectively. These wind conditions are equivalent to the relative wind speed V_{res} of 84.57 mph and the relative wind angle ψ of 14.21° (Figure 2.2). The wind forces acting on the car were deduced by interpolating the steady-state aerodynamic coefficients obtained from the full-scale wind tunnel tests. The axes

system used to measure the vehicle responses is shown in Figure 2.2, also. The origin lies at road level, mid-wheelbase, and mid-track of the vehicle. The x -axis points backward, the y -axis points to the right, and the z -axis points up.

2.3 Factors Affecting Crosswind Sensitivity

2.3.1 Varying Aerodynamic Coefficients

Many simulations were done on the Taurus sedan to understand how changes in aerodynamic coefficients affect the vehicle response to crosswind. Each of the six aerodynamic coefficients was varied from 75% to 125% of real values one at a time. The angular velocity, lateral acceleration, and lateral deviation of the car were plotted to show the responses of the vehicle to crosswind. In addition, plots of the force transmitted to the steering rack were included because the force may be a major source of feedback to the driver. Finally, a relationship matrix showing the cause and effect was developed. The relationship matrix in Table 2.1 shows the changes in response variables for the aerodynamic coefficients changed to 125 %.

From this study, it was found that most of the response variables are influenced primarily by the yaw moment coefficient (C_{M_y}) and side force coefficient (C_{F_y}), and the majority of the change in roll velocity is attributed to the roll moment coefficient (C_{M_r}). The contributions of the drag coefficient (C_{F_x}), lift force coefficient (C_{F_z}), and pitch moment coefficient (C_{M_z}) to the changes in the variables are negligible. The simulations with the aerodynamic coefficients changed to 75% reveal very similar changes in the variables but in the opposite direction. In short, the vehicle response is almost linear to aerodynamic load changes within the ranges considered. More details are found in the reports by Koh et al. [1989].

2.3.2 Comparing Taurus Sedan and Taurus Wagon

Since the factors that affect the crosswind sensitivity of a vehicle are not known clearly at present, it was decided that by comparing the crosswind responses of two vehicles with different aerodynamic behaviors, one could learn which factors correlate or do not correlate with crosswind responses. One vehicle, the Taurus station wagon, has been described to have a qualitatively superior crosswind behavior, while the other, the Taurus sedan, has a mediocre one. The same crosswind condition was used in the two simulations (Figure 2.1). The only differences between the two simulations were the aerodynamic coefficients and the suspension system. The response characteristics of the vehicles were compared using outputs of the angular displacements of the vehicle body, the tire forces and moments, and the front spindle angles.

2.3.2.1 Yaw, Pitch, and Roll Responses

The yaw of the sedan (Figure 2.3) increases steadily away from the wind for the duration of the gust and reaches a magnitude of 2.5° at the end of the gust. Since there is no driver input or correction, the vehicle continues its heading after the gust. The wagon behaves similarly, but the yaw reached is 1.8° .

The sedan pitches (Figure 2.4) its nose down during the gust and does so proportionally to the gust strength. The maximum change in pitch is 0.18° . But, for the wagon, the pitch at the beginning of the gust shows an overshoot followed by an oscillating decay to a steady-state value of 0.14° . And, the pitch at the end of the gust also shows an overshoot followed by an oscillating decay to the initial level.

The sedan rolls (Figure 2.5) away from the wind briefly but then rolls into the wind for the duration of the gust. At the end, it rolls back passing the neutral and away from the wind briefly before settling in the neutral position. The wagon rolls more away from the wind at the start of the gust, but remains almost neutral for the duration of the gust. At the

end of the gust, the wagon rolls into the wind before returning to the neutral position. The lack of roll during the crosswind may account for the better ride in the wagon.

2.3.2.2 Tire Forces and Moments During Crosswind

Realizing that it is not possible to locate the aerodynamic center of pressure and the center of tire resistance [Koh 1989], it was decided to plot the force and moment vectors acting on the vehicle. The vectors include the resultant aerodynamic force and moment applied at the mid-track and mid-wheelbase and the tire force and moment acting on each tire-ground patch. Orthographic plots are used to illustrate the three dimensional nature of these vectors and how they change with time (Figures 2.6-2.7).

In the sedan, the net lateral front tire force augments the crosswind force to push it further away from the wind (Figure 2.6). In the wagon, however, the net lateral front tire force resists the crosswind force slightly at the beginning and end of the crosswind, and remains almost neutral during the steady portion of the gust (Figure 2.7). For both the sedan and the wagon, the lateral tire forces on the rear are negligible compared to those on the front.

Comparing the roll response (Figure 2.5) with the net lateral front tire force (Figure 2.8) reveals a definite relationship. When the vehicle rolls to the right, the net lateral front tire force points to the left and its magnitude is proportional to the amount of roll. This means that the sedan exhibits reverse roll-steer. Similar observations are found in the wagon.

2.3.3 Parametric Study on Suspension and Aerodynamics

The studies on the responses of the sedan and wagon gave some indications of the factors affecting crosswind sensitivity. But, to further isolate the factors, the sedan was run with the six aerodynamic coefficients of the wagon, and the wagon with the

coefficients of the sedan. In addition, the sedan was run with one of its coefficients replaced by the corresponding wagon's.

2.3.3.1 Dependency of Response on Suspension System

The Taurus sedan and station wagon have different rear suspension systems. To see how the suspension system affects the crosswind response, the sedan was simulated with the wagon aerodynamic coefficients ('sedan-wagon'). Conversely, the wagon was simulated with the sedan aerodynamic coefficients ('wagon-sedan'). The results show that the responses of 'sedan-wagon' are nearly the same as those of the wagon with the wagon aerodynamic coefficients ('wagon-wagon'). Also, the responses of 'wagon-sedan' are nearly the same as those of the sedan with the sedan aerodynamic coefficients ('sedan-sedan'). It seems that the rear suspension systems have essentially the same dynamic properties, or that the rear suspension systems do not affect the crosswind response significantly. To study further the dependency of responses on suspension system, simulations on a great variety of vehicle types are needed.

2.3.3.2 Further Study on Aerodynamic Coefficients

The 'sedan-wagon' and 'wagon-sedan' simulations show that the aerodynamic coefficients are the main determinants of vehicle crosswind sensitivity. To check the contribution of each coefficient to the vehicle responses, the Taurus sedan was run with one of its coefficients replaced by the corresponding wagon's. The results are summarized in a relationship matrix (Table 2.2).

From this study, it was found that most of the variables are influenced primarily by C_{M_z} and C_{M_y} of the wagon, and the majority of the changes in roll velocity of the sedan is affected by C_{M_z} of the wagon. Also, more nose down pitching moment reduces the yaw and lateral responses of the sedan. The contributions of C_{F_z} and C_{F_y} of the wagon to the changes in the responses of the sedan are negligible. These observations have good agreement with those in Section 2.2.1 (Table 2.1).

2.3.4 Interpretation of Vehicle Responses

The simulated sequence of events that take place when the Taurus sedan and station wagon encounter the crosswind model (Figure 2.1) are now described in detail.

From 0 to 1 second, the sedan travels in a straight line at a steady 70 mph. The state of aerodynamic and tire forces and moments is shown in Figure 2.9.

Between 1 and 1.2 seconds, when the crosswind speed increases smoothly from 0 to 23.97 mph, the sedan rolls away from the wind very briefly before beginning to roll into the wind. The initial roll away from the wind is due to the aerodynamic roll moment (Figure 2.5). Concurrently, the sedan begins to yaw away from the wind due to the aerodynamic yaw moment (Figure 2.3). The net lateral front and rear tire forces resist this yaw, but the moments they produce are small, 7% and 28%, respectively compared to the aerodynamic yaw moment (Figure 2.10). Therefore, reverse roll-steer is slightly beneficial at this stage. Once the sedan begins to curve away, the centrifugal force produces an overturning moment that opposes the aerodynamic roll moment.

Between 1.2 and 2.2 seconds, the crosswind wind speed is constant at 23.97 mph. The aerodynamic yaw moment continues to veer the sedan away and the centrifugal overturning moment overcomes the aerodynamic roll moment to cause the sedan to roll into the wind for this duration. This roll also causes the net lateral front tire forces to augment the yawing effect of the aerodynamic yaw moment (Figure 2.11). Therefore, reverse roll-steer is detrimental at this stage. The roll exhibits an overshoot followed by an oscillating decay to a steady state value.

Between 2.2 and 2.4 seconds, when the crosswind dies down to 0 mph, the roll increases into the wind because the aerodynamic roll moment decreases (Figure 2.5), while the centrifugal overturning moment is still strong. But as the aerodynamic yaw moment

decreases (Figure 2.3), the sedan curves less; the overturning moment reduces accordingly, and the sedan returns to its upright position.

After 2.4 seconds, there is no more crosswind; so the sedan returns to a straight and steady path. The heading is, however, no longer the same as the initial one because no driver input was introduced to correct the vehicle heading.

The sedan pitches its nose down in an overdamped fashion when the crosswind comes on and rebounds slowly to the initial level after the crosswind dies out (Figure 2.3).

The station wagon responds similarly in the simulation. But important differences exist that help to explain why it has reduced crosswind sensitivity. The initial roll away from the wind is larger than the sedan's because of the larger roll coefficient (Figure 2.4). But the smaller yaw moment coefficient causes the wagon to curve away from the wind, less than the sedan. The centrifugal overturning moment is, therefore, less and is almost balanced by the aerodynamic roll moment. Hence, the wagon maintains an almost upright position during the steady portion of the crosswind. Also, the larger initial roll causes a larger net lateral front tire force to resist the aerodynamic yaw moment (Figure 2.12). During the crosswind the upright position of the wagon results in no net lateral front tire force (Figure 2.13). In the wagon's case, reverse roll is more beneficial at the start of the crosswind and has neutral value during the steady portion of the crosswind.

From this comparison, it is found that the lateral responses of the cars depend mainly on the aerodynamic roll and yaw moments.

2.4 Conclusions of Simulation Study

The aerodynamic yaw moment is the most important factor affecting crosswind sensitivity. It is much bigger than the yaw moment due to the tire forces. Therefore, the resultant yaw moment on the vehicle, referred to the point at mid-wheelbase, mid-track, and road level, is mainly determined by the aerodynamic yaw moment.

The Taurus sedan and station wagon exhibit reverse roll-steer. That is, when the vehicle rolls to the right the net lateral front tire force is to the left, and vice versa. Reverse roll-steer is beneficial in reducing yaw and lateral responses only if the vehicle rolls away from the wind.

More nose down pitching moment is beneficial in reducing yaw and lateral response.

The aerodynamic roll moment competes with the centrifugal overturning moment to determine the roll of the vehicle.

Finally, the author gained some experience and ideas applicable to the development of OODS.

Table 2.1: Effect of Aerodynamic Coefficients on Taurus Sedan

Responses Parameters	Roll Velocity ω_x	Pitch Velocity ω_y	Yaw Velocity ω_z	Lateral Acceleration $\ddot{y}$	Lateral Deviation y	Steering System Reaction
$\uparrow C_{Fx}$	—	$\uparrow$	—	—	—	—
$\uparrow C_{Fy}$	$\downarrow$	—	$\uparrow$	—	—	$\downarrow$
$\uparrow C_{Fz}$	—	$\uparrow$	—	—	—	—
$\uparrow C_{Mx}$	$\uparrow$	—	—	—	—	—
$\uparrow C_{My}$	—	—	—	—	—	—
$\uparrow C_{Mz}$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$

= 100 % Increase in Magnitude

— : Negligible Change

Table 2.2: Simulation of Taurus Sedan with Wagon's Aerodynamic Coefficients

Responses Parameters		Roll Velocity ω_x	Pitch Velocity ω_y	Yaw Velocity ω_z	Lateral Acceleration $\ddot{y}$	Lateral Deviation y	Steering System Reaction
Wagon's Aerodynamic Coefficients	C_{Fx}	—	↑	—	—	—	—
	C_{Fy}	*	—	↑	↑	↑	↓
	C_{Fz}	—	—	—	—	—	—
	C_{Mx}	Fig. 4.13	—	—	—	—	—
	C_{My}	↓	↑	↓	↓	—	*
	C_{Mz}	↓	—	↓	↓	↓	↓
$C_{Mz} = 0$		Fig. 4.14	—	Fig. 4.15	*	*	*

↑ = 100 % Increase in Magnitude

* : The pattern is quite different.

— : The change is negligible.

Figure 2.1: Crosswind Simulation Scheme and Simulated Wind Input

$$F_i = \frac{1}{2} C_{F_i} A \rho V_{res}^2 \quad \text{where} \quad C_{F_i} = C_{F_i}(\psi_{res}) \quad i = x, y, z.$$

$$M_i = \frac{1}{2} C_{M_i} A W \rho V_{res}^2 \quad \text{where} \quad C_{M_i} = C_{M_i}(\psi_{res}).$$

Figure 2.2: Vehicle Axes and Wind Velocities

Figure 2.3: Yaw of Taurus Sedan and Wagon

Figure 2.4: Pitch of Taurus Sedan and Wagon

Figure 2.5: Roll of Taurus Sedan and Wagon

TITLE : Aero and Tire Forces on Taurus Sedan

TIME : 2.20 sec

YAW : 14.13 deg

→ : 1.0E+3 (N)

→ : 1.5E+5 (N-mm)

→ : Relative Wind Velocity, 36.69 m/sec (82.09 mph)

Small forces and moments do not have arrowhead.

Figure 2.6: Force and Moment Vectors on Taurus Sedan

TITLE : Aero and Tire Forces on Taurus

TIME : 2.20 sec

YAW : 13.96 deg

→ : $1.0\text{E}+3$ (N)

→ : $1.5\text{E}+5$ (N-mm)

→ : Relative Wind Velocity, 36.59 m/sec (81.86 mph)

Small forces and moments do not have arrowhead.

Figure 2.7: Force and Moment Vectors on Taurus Wagon

Figure 2.8: Net Lateral Front Tire Force of Taurus Sedan and Wagon

Figure 2.9: Force and Moment Vectors on Taurus Sedan at 0.8 Seconds

TITLE : Aero and Tire Forces on Taurus Sedan

TIME : 1.20 sec

YAW : 13.55 deg

→ : $1.0\text{E}+3$ (N)

→ : $1.5\text{E}+5$ (N-mm)

→ : Relative Wind Velocity, 37.17 m/sec (83.17 mph)

Small forces and moments do not have arrowhead.

Figure 2.10: Force and Moment Vectors on Taurus Sedan at 1.2 Seconds

TITLE : Aero and Tire Forces on Taurus Sedan
TIME : 1.70 sec
YAW : 13.96 deg
 —→ : 1.0E+3 (N)
 —→ : 1.5E+5 (N-mm)
 —→ : Relative Wind Velocity, 36.94 m/sec (82.64 mph)
Small forces and moments do not have arrowhead.

Figure 2.11: Force and Moment Vectors on Taurus Sedan at 1.7 Seconds

TITLE : Aero and Tire Forces on Taurus

TIME : 1.20 sec

YAW : 13.52 deg

→ : 1.0E+3 (N)

→ : 1.5E+5 (N-mm)

→ : Relative Wind Velocity, 37.10 m/sec (83.01 mph)

Small forces and moments do not have arrowhead.

Figure 2.12: Force and Moment Vectors on Taurus Wagon at 1.2 Seconds

TITLE : Aero and Tire Forces on Taurus Wagon
 TIME : 1.70 sec
 YAW : 13.79 deg
 $\longrightarrow$: $1.0E+3$ (N)
 $\longrightarrow$: $1.5E+5$ (N-mm)
 $\longrightarrow$: Relative Wind Velocity, 36.84 m/sec (82.43 mph)
 Small forces and moments do not have arrowhead.

Figure 2.13: Force and Moment Vectors on Taurus Wagon at 1.7 Seconds

CHAPTER III

FORMULATION OF SYSTEM EQUATIONS

3.1 Introductory Comments

The behavior of a dynamic system consisting of rigid bodies, joints, and force elements can be described in terms of state variables and system equations relating the state variables. The system may undergo large displacements and require a lot of computational work for its simulation. The essential computational work includes the automated formulation and numerical solution of a set of simultaneous nonlinear ordinary differential-algebraic equations with initial conditions.

This chapter explains the dynamics theory applied to OODS for the formulation of system equations. The system equations consist of the equations of motion, generalized momentum definitions, constraint equations, and generalized force definitions.

3.2 System Elements

A multibody system is a collection of dynamic elements such as ground, parts, joints, springs, dampers, applied forces and torques, prescribed motions, etc. (Figure 3.1). The ground holds the inertial frame. It can be approximated by the earth for most engineering problems.

A part is a rigid body which can move through space. It may be constrained to other parts or the ground with joints, springs, or dampers. The part has mass, moment-of-inertia, a center-of-mass marker, and other markers. It also has kinematic variables such as positions, velocities, accelerations, and prescribed motions, as well as dynamic variables such as applied forces and torques.

A marker is used to identify a point fixed on a part. It can be associated with the center-of-mass, point of force application, or point of joint attachment.

A joint allows two neighboring parts to move relative to each other in a predefined fashion by imposing constraints between them. The joints most often used include the spherical, revolute, universal, cylindrical, translational, and planar joints. A spherical or ball joint permits all three rotations but no translations. A revolute or pin joint allows only one rotation. A universal joint allows two rotations. A cylindrical joint permits one rotation and one translation. A translational joint allows only one translation. A planar joint permits two translations in a plane and one rotation about the normal to the plane. The exact constraint equations describing these joints are given later.

Springs and dampers produce compliant forces or torques between bodies as functions of relative positions and velocities. They will not be represented physically in the formulation. Instead, the spring forces, damping forces, gravitational forces, applied forces, and applied torques are all considered to be external forces acting on parts.

Finally, a prescribed motion imposes a translational or rotational motion on a part as a function of time.

3.3 Coordinate System

The spatial coordinates employed in OODS are specified by a ground frame, part frames, and marker frames (Figure 3.2). The ground frame is the inertial frame from which the kinematic variables and dynamic variables are measured. It employs three Cartesian coordinates (X, Y, Z) which have the corresponding unit vectors $I-J-K$.

A part frame is fixed to a part. Its origin lies at the center-of-mass, and its axes are oriented with the principal directions of the part. The part frame is located and oriented relative to the ground frame by three Cartesian coordinates (X, Y, Z) and three Euler angles (Φ, Θ, Ψ). The Euler angles are measured using the Z-X-Z convention. Each part frame employs three Cartesian coordinates (x, y, z) which have the corresponding unit vectors $i-j-k$.

The origin of the marker frame is attached to the point it is used to identify. Its axes can be oriented in any fashion for the purpose of specifying joints or applied forces. The marker frame is located relative to the part frame by $(x, y, z, \phi, \theta, \psi)$. Each marker frame employs three Cartesian coordinates (x', y', z') which have the corresponding unit vectors $i'-j'-k'$.

3.4 System Equations

3.4.1 State Variables

The state variables used in OODS are the generalized coordinates, generalized momenta, generalized forces, and Lagrange multipliers. As mentioned in Section 1.2, there are two major ways of choosing generalized coordinates for the formulation of system equations. OODS adopts the absolute coordinate system. The generalized coordinates for each part are $(X, Y, Z, \Phi, \Theta, \Psi)$.

The generalized momenta are used to reduce the order of the equations of motion derived using Lagrange's equation. Each part has six generalized momenta.

Lagrange multipliers are associated with constraint equations. Instead of substituting them into the equations of motion, OODS treats them as additional state variables of the system equations to increase the simplicity and generality of the formulation.

Finally, generalized forces are derived directly from their definition through the principle of virtual work.

3.4.2 Lagrange's Equation

Consider a dynamic system with n_p parts and n_c constraints. The number of system coordinates n_q is $6n_p$. Using Lagrange's equation, one obtains the equations of motion for q_{i_q} .

$$\frac{d}{dt} \left(\frac{\partial T}{\partial \dot{q}_{i_q}} \right) - \frac{\partial T}{\partial q_{i_q}} - Q_{i_q} - \sum_{i_c=1}^{n_c} \frac{\partial G_{i_c}}{\partial q_{i_q}} \lambda_{i_c} = 0 \quad (i_q = 1, 2, 3, \dots, n_q) \quad (3.1)$$

where T is the total kinetic energy of the system, Q_{i_q} is the generalized force associated with q_{i_q} , G_{i_c} is the constraint equation, and λ_{i_c} is the Lagrange multiplier associated with G_{i_c} . This equation arises from the force or moment balance in the q_{i_q} direction. The generalized momentum u_{i_q} is defined as

$$u_{i_q} = \frac{\partial T}{\partial \dot{q}_{i_q}} \quad (i_q = 1, 2, 3, \dots, n_q) . \quad (3.2)$$

Therefore, the equations of motion become

$$\dot{u}_{i_q} - \frac{\partial T}{\partial q_{i_q}} - Q_{i_q} - \sum_{i_c=1}^{n_c} \frac{\partial G_{i_c}}{\partial q_{i_q}} \lambda_{i_c} = 0 \quad (i_q = 1, 2, 3, \dots, n_q) . \quad (3.3)$$

The total kinetic energy of the system can be expressed as the sum of translational and rotational kinetic energy of every part.

$$T = \sum_{i_p=1}^{n_p} \frac{1}{2} m_{i_p} \mathbf{V}_{i_p} \cdot \mathbf{V}_{i_p} + \sum_{i_p=1}^{n_p} \frac{1}{2} \boldsymbol{\omega}_{i_p} \cdot \mathbf{J}_{i_p} \cdot \boldsymbol{\omega}_{i_p} \quad (3.4)$$

where m_{i_p} is the mass of part i_p and

$$\mathbf{V}_{i_p} = \dot{X}_{i_p} \mathbf{I} + \dot{Y}_{i_p} \mathbf{J} + \dot{Z}_{i_p} \mathbf{K} \quad (3.5)$$

$$\boldsymbol{\omega}_{i_p} = \omega_{i_p,x} \mathbf{i}_{i_p} + \omega_{i_p,y} \mathbf{j}_{i_p} + \omega_{i_p,z} \mathbf{k}_{i_p} \quad (3.6)$$

$$\mathbf{J}_{i_p} = J_{xx} \mathbf{i}_{i_p} \mathbf{i}_{i_p} + J_{yy} \mathbf{j}_{i_p} \mathbf{j}_{i_p} + J_{zz} \mathbf{k}_{i_p} \mathbf{k}_{i_p} . \quad (3.7)$$

Here, $\dot{X}_{i_p}$, $\dot{Y}_{i_p}$, and $\dot{Z}_{i_p}$ are the components of the absolute translational velocity of the center-of-mass, $\mathbf{V}_{i_p}$, expressed in the ground frame. $\omega_{i_p,x}$, $\omega_{i_p,y}$, and $\omega_{i_p,z}$ are the components of the absolute angular velocity, $\boldsymbol{\omega}_{i_p}$, expressed in the part frame. J_{xx} , J_{yy} , and J_{zz} are the three dyadic components of the principal mass moment-of-inertia $\mathbf{J}_{i_p}$.

Figure 3.3 shows that

$$\boldsymbol{\omega}_{i_r} = \dot{\Phi}_{i_r} \mathbf{K}_{i_r} + \dot{\Theta}_{i_r} \mathbf{I}'_{i_r} + \dot{\Psi}_{i_r} \mathbf{K}''_{i_r} \quad (3.8)$$

where the unit vectors $\mathbf{K}_{i_r}$, $\mathbf{I}'_{i_r}$, and $\mathbf{K}''_{i_r}$ form Euler axes, and $\dot{\Phi}_{i_r}$, $\dot{\Theta}_{i_r}$, and $\dot{\Psi}_{i_r}$ are the absolute angular velocity components along the Euler axes. Note that $\mathbf{K}_{i_r}$, $\mathbf{I}'_{i_r}$, and $\mathbf{K}''_{i_r}$ do not form a mutually orthogonal vector triad in general. Using the transformation matrices for the individual rotations

$$\begin{Bmatrix} \mathbf{I}' \\ \mathbf{J}' \\ \mathbf{K}' \end{Bmatrix}_{i_r} = \begin{bmatrix} \cos \Phi & \sin \Phi & 0 \\ -\sin \Phi & \cos \Phi & 0 \\ 0 & 0 & 1 \end{bmatrix}_{i_r} \begin{Bmatrix} \mathbf{I} \\ \mathbf{J} \\ \mathbf{K} \end{Bmatrix}_{i_r} = [\Phi]_{i_r} \begin{Bmatrix} \mathbf{I} \\ \mathbf{J} \\ \mathbf{K} \end{Bmatrix}_{i_r} \quad (3.9)$$

$$\begin{Bmatrix} \mathbf{I}'' \\ \mathbf{J}'' \\ \mathbf{K}'' \end{Bmatrix}_{i_r} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \Theta & \sin \Theta \\ 0 & -\sin \Theta & \cos \Theta \end{bmatrix}_{i_r} \begin{Bmatrix} \mathbf{I}' \\ \mathbf{J}' \\ \mathbf{K}' \end{Bmatrix}_{i_r} = [\Theta]_{i_r} \begin{Bmatrix} \mathbf{I}' \\ \mathbf{J}' \\ \mathbf{K}' \end{Bmatrix}_{i_r} = [\Theta]_{i_r} [\Phi]_{i_r} \begin{Bmatrix} \mathbf{I} \\ \mathbf{J} \\ \mathbf{K} \end{Bmatrix}_{i_r} \quad (3.10)$$

$$\begin{aligned} \begin{Bmatrix} \mathbf{i} \\ \mathbf{j} \\ \mathbf{k} \end{Bmatrix}_{i_r} &= \begin{bmatrix} \cos \Psi & \sin \Psi & 0 \\ -\sin \Psi & \cos \Psi & 0 \\ 0 & 0 & 1 \end{bmatrix}_{i_r} \begin{Bmatrix} \mathbf{I}'' \\ \mathbf{J}'' \\ \mathbf{K}'' \end{Bmatrix}_{i_r} \\ &= [\Psi]_{i_r} \begin{Bmatrix} \mathbf{I}'' \\ \mathbf{J}'' \\ \mathbf{K}'' \end{Bmatrix}_{i_r} = [\Psi]_{i_r} [\Theta]_{i_r} \begin{Bmatrix} \mathbf{I}' \\ \mathbf{J}' \\ \mathbf{K}' \end{Bmatrix}_{i_r} = [\Psi]_{i_r} [\Theta]_{i_r} [\Phi]_{i_r} \begin{Bmatrix} \mathbf{I} \\ \mathbf{J} \\ \mathbf{K} \end{Bmatrix}_{i_r}, \end{aligned} \quad (3.11)$$

one obtains

$$\mathbf{K}_{i_r} = \sin \Theta_{i_r} \sin \Psi_{i_r} \mathbf{i}_{i_r} + \sin \Theta_{i_r} \cos \Psi_{i_r} \mathbf{j}_{i_r} + \cos \Theta_{i_r} \mathbf{k}_{i_r} \quad (3.12)$$

$$\mathbf{I}'_{i_r} = \cos \Psi_{i_r} \mathbf{i}_{i_r} - \sin \Psi_{i_r} \mathbf{j}_{i_r} \quad (3.13)$$

$$\mathbf{K}''_{i_r} = \mathbf{k}_{i_r}. \quad (3.14)$$

Substituting Eq.(3.12)-(3.14) into Eq.(3.8) and comparing with Eq.(3.6) give

$$\begin{Bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{Bmatrix}_{i_p} = \begin{bmatrix} \sin \Theta \sin \Psi & \cos \Psi & 0 \\ \sin \Theta \cos \Psi & -\sin \Psi & 0 \\ \cos \Theta & 0 & 1 \end{bmatrix}_{i_p} \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i_p} = [B]_{i_p} \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i_p} \quad (3.15)$$

where $[B]_{i_p}$ is the transformation matrix from the Euler axes to the part frame. Noting Eq.(3.4) and Eq.(3.15), one can express the total kinetic energy of the system in matrix form.

$$T = \sum_{i_p=1}^{n_p} \frac{1}{2} m_{i_p} (\dot{X}_{i_p}^2 + \dot{Y}_{i_p}^2 + \dot{Z}_{i_p}^2) + \sum_{i_p=1}^{n_p} \frac{1}{2} \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i_p}^T [B]_{i_p}^T [J]_{i_p} [B]_{i_p} \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i_p} \quad (3.16)$$

where $[J]_{i_p}$ is

$$[J]_{i_p} = \begin{bmatrix} J_{xx} & 0 & 0 \\ 0 & J_{yy} & 0 \\ 0 & 0 & J_{zz} \end{bmatrix}_{i_p} \quad J_{xx} \leq J_{yy} \leq J_{zz} . \quad (3.17)$$

Now, substituting the kinetic energy expression into the equations of motion (Eq.(3.3)) and considering part i_p ,

$$\frac{\partial T}{\partial X_{i_p}} = 0 \quad (3.18)$$

$$\frac{\partial T}{\partial Y_{i_p}} = 0 \quad (3.19)$$

$$\frac{\partial T}{\partial Z_{i_p}} = 0 \quad (3.20)$$

$$\frac{\partial T}{\partial \Phi_{i_p}} = 0 \quad (3.21)$$

$$\frac{\partial T}{\partial \Theta_{i_p}} = \left\{ \begin{matrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{matrix} \right\}_{i_p}^T \left(\frac{\partial}{\partial \Theta_{i_p}} [B]_{i_p}^T \right) [J]_{i_p} [B]_{i_p} \left\{ \begin{matrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{matrix} \right\}_{i_p} \quad (3.22)$$

$$\frac{\partial T}{\partial \Psi_{i_p}} = \left\{ \begin{matrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{matrix} \right\}_{i_p}^T \left(\frac{\partial}{\partial \Psi_{i_p}} [B]_{i_p}^T \right) [J]_{i_p} [B]_{i_p} \left\{ \begin{matrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{matrix} \right\}_{i_p} . \quad (3.23)$$

Eq.(3.22)-(3.23) employ the symmetricity of $[B]_{i_p}^T [J]_{i_p} [B]_{i_p}$ to achieve the simpler form.

Finally, the equations of motion for part i_p are

$$\left\{ \begin{matrix} \dot{u}_x \\ \dot{u}_y \\ \dot{u}_z \\ \dot{u}_\Phi \\ \dot{u}_\Theta \\ \dot{u}_\Psi \end{matrix} \right\}_{i_p} - \left\{ \begin{matrix} 0 \\ 0 \\ 0 \\ 0 \\ \frac{\partial T}{\partial \Theta} \\ \frac{\partial T}{\partial \Psi} \end{matrix} \right\}_{i_p} - \left\{ \begin{matrix} Q_x \\ Q_y \\ Q_z \\ Q_\Phi \\ Q_\Theta \\ Q_\Psi \end{matrix} \right\}_{i_p} - \left\{ \begin{matrix} \frac{\partial G_{1_c}}{\partial X} \lambda_{1_c} + \frac{\partial G_{2_c}}{\partial X} \lambda_{2_c} + \dots + \frac{\partial G_{n_c}}{\partial X} \lambda_{n_c} \\ \frac{\partial G_{1_c}}{\partial Y} \lambda_{1_c} + \frac{\partial G_{2_c}}{\partial Y} \lambda_{2_c} + \dots + \frac{\partial G_{n_c}}{\partial Y} \lambda_{n_c} \\ \frac{\partial G_{1_c}}{\partial Z} \lambda_{1_c} + \frac{\partial G_{2_c}}{\partial Z} \lambda_{2_c} + \dots + \frac{\partial G_{n_c}}{\partial Z} \lambda_{n_c} \\ \frac{\partial G_{1_c}}{\partial \Phi} \lambda_{1_c} + \frac{\partial G_{2_c}}{\partial \Phi} \lambda_{2_c} + \dots + \frac{\partial G_{n_c}}{\partial \Phi} \lambda_{n_c} \\ \frac{\partial G_{1_c}}{\partial \Theta} \lambda_{1_c} + \frac{\partial G_{2_c}}{\partial \Theta} \lambda_{2_c} + \dots + \frac{\partial G_{n_c}}{\partial \Theta} \lambda_{n_c} \\ \frac{\partial G_{1_c}}{\partial \Psi} \lambda_{1_c} + \frac{\partial G_{2_c}}{\partial \Psi} \lambda_{2_c} + \dots + \frac{\partial G_{n_c}}{\partial \Psi} \lambda_{n_c} \end{matrix} \right\}_{i_p} = \left\{ \begin{matrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{matrix} \right\} . \quad (3.24)$$

This equation may be condensed to

$$\{\dot{u}\}_{i_r} - \left[\frac{\partial T}{\partial \{q\}_{i_r}} \right]^T - \{Q\}_{i_r} - \left[\frac{\partial \{G\}}{\partial \{q\}_{i_r}} \right]^T \{\lambda\} = \{0\} \quad (3.25)$$

using

$$\{q\}_{i_r} = \begin{Bmatrix} X \\ Y \\ Z \\ \Phi \\ \Theta \\ \Psi \end{Bmatrix}_{i_r} \quad \{u\}_{i_r} = \begin{Bmatrix} u_x \\ u_y \\ u_z \\ u_\Phi \\ u_\Theta \\ u_\Psi \end{Bmatrix}_{i_r} \quad \{Q\}_{i_r} = \begin{Bmatrix} Q_x \\ Q_y \\ Q_z \\ Q_\Phi \\ Q_\Theta \\ Q_\Psi \end{Bmatrix}_{i_r} \quad \{\lambda\} = \begin{Bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_{i_r} \\ \vdots \\ \lambda_{n_r} \end{Bmatrix}. \quad (3.26)$$

Substituting the kinetic energy expression into the definition of generalized momentum (Eq.(3.2)), one obtains

$$\begin{Bmatrix} u_x \\ u_y \\ u_z \\ u_\Phi \\ u_\Theta \\ u_\Psi \end{Bmatrix}_{i_r} = \begin{bmatrix} m[I] & [0] \\ [0] & [B]^T [J] [B] \end{bmatrix}_{i_r} \begin{Bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \\ \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i_r} = [C]_{i_r} \begin{Bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \\ \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i_r} \quad (3.27)$$

or

$$\{u\}_{i_r} - [C]_{i_r} \{\dot{q}\}_{i_r} = \{0\} \quad (3.28)$$

where $[C]_{i_r}$ transforms the generalized speeds to the generalized momenta. The expressions for the rotational momenta are quite complicated because $[B]_{i_r}$ contains trigonometric functions of rotational coordinates. If the generalized momenta were not defined as state variables, one could not avoid the direct differentiation of these complicated expression as required by the first term of Lagrange's equation (Eq.(3.3)). Also,

$$\begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i_p} = \left[[B]^T [J] [B] \right]_{i_p}^{-1} \begin{Bmatrix} u_{\Phi} \\ u_{\Theta} \\ u_{\Psi} \end{Bmatrix}_{i_p}. \quad (3.29)$$

The equations of motion and the momentum definitions for the whole system are obtained by stacking the corresponding equations of each part. That is,

$$\{\dot{u}\} - \left[\frac{\partial T}{\partial \{q\}} \right]^T - \{Q\} - \left[\frac{\partial \{G\}}{\partial \{q\}} \right]^T \{\lambda\} = \{0\} \quad (3.30)$$

where

$$\{q\} = \begin{Bmatrix} \{q\}_1 \\ \{q\}_2 \\ \vdots \\ \{q\}_{i_p} \\ \vdots \\ \{q\}_{n_p} \end{Bmatrix} \quad \{\dot{u}\} = \begin{Bmatrix} \{\dot{u}\}_1 \\ \{\dot{u}\}_2 \\ \vdots \\ \{\dot{u}\}_{i_p} \\ \vdots \\ \{\dot{u}\}_{n_p} \end{Bmatrix} \quad \{Q\} = \begin{Bmatrix} \{Q\}_1 \\ \{Q\}_2 \\ \vdots \\ \{Q\}_{i_p} \\ \vdots \\ \{Q\}_{n_p} \end{Bmatrix}. \quad (3.31)$$

Furthermore, Eq.(3.30) can be represented by an implicit function

$$\{M(\{q\}, \{\dot{q}\}, \{\dot{u}\}, \{\lambda\}, \{Q\})\} = \{0\}. \quad (3.32)$$

Similarly, the equations for system momenta can be

$$\{u\} - [C]\{\dot{q}\} = \{0\} \quad (3.33)$$

where

$$[C] = \begin{bmatrix} [C]_1 & \cdots & [0] \\ \vdots & \ddots & \vdots \\ [0] & \cdots & [C]_{n_p} \end{bmatrix}. \quad (3.34)$$

Eq.(3.33) may be represented by an implicit function

$$\{P(\{q\},\{\dot{q}\},\{\dot{u}\})\} = \{0\} . \quad (3.35)$$

3.4.3 Force equations

3.4.3.1 Virtual Work and Generalized Forces

Consider the whole system having a set of external forces F_{i_f} and torques T_{i_t} (Figure 3.4). The virtual work of the system can be expressed as

$$\delta W = \sum_{i_f=1}^{n_f} F_{i_f} \cdot \delta S_{i_f} + \sum_{i_t=1}^{n_t} T_{i_t} \cdot \delta \alpha_{i_t} \quad (3.36)$$

where n_f and n_t are the number of applied forces and torques. δS_{i_f} is the virtual displacement for F_{i_f} , and $\delta \alpha_{i_t}$ is the virtual rotation for T_{i_t} .

The virtual displacements can be expressed in generalized coordinates;

$$\delta S_{i_f} = \sum_{i_q=1}^{n_q} \frac{\partial S_{i_f}}{\partial q_{i_q}} \delta q_{i_q} \quad (3.37)$$

$$\delta \alpha_{i_t} = \sum_{i_q=1}^{n_q} \frac{\partial \alpha_{i_t}}{\partial q_{i_q}} \delta q_{i_q} . \quad (3.38)$$

Substituting these expressions into Eq.(3.36) gives

$$\delta W = \sum_{i_q=1}^{n_q} \left(\sum_{i_f=1}^{n_f} F_{i_f} \cdot \frac{\partial S_{i_f}}{\partial q_{i_q}} + \sum_{i_t=1}^{n_t} T_{i_t} \cdot \frac{\partial \alpha_{i_t}}{\partial q_{i_q}} \right) \delta q_{i_q} = \sum_{i_q=1}^{n_q} Q_{i_q} \delta q_{i_q} . \quad (3.39)$$

Then, the generalized force associated with q_{i_q} is

$$Q_{i_q} = \sum_{i_f=1}^{n_f} F_{i_f} \cdot \frac{\partial S_{i_f}}{\partial q_{i_q}} + \sum_{i_t=1}^{n_t} T_{i_t} \cdot \frac{\partial \alpha_{i_t}}{\partial q_{i_q}} \quad (3.40)$$

which can be expanded as

$$\begin{aligned}
Q_{i_p} = & \sum_{i_f=1}^{n_f} [F_{i_f X} \quad F_{i_f Y} \quad F_{i_f Z}] \frac{\partial}{\partial q_{i_p}} \left\{ \begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix}_{i_p} + [A]_{i_p} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_{i_f} \right\} \\
& + \sum_{i_p=1}^{n_p} [T_{i_p X} \quad T_{i_p Y} \quad T_{i_p Z}] [A]_{i_p} [B]_{i_p} \frac{\partial}{\partial q_{i_p}} \left\{ \begin{Bmatrix} \Phi \\ \Theta \\ \Psi \end{Bmatrix}_{i_p} \right\}
\end{aligned} \tag{3.41}$$

where $[A]_{i_p}$ is the transformation matrix from the part frame to the ground frame

$$[A]_{i_p} = [[\Psi][\Theta][\Phi]]_{i_p}^T. \tag{3.42}$$

Finally, the six generalized forces acting on part i_p are

$$Q_{i_p X} = \sum_{i_f=1}^{\text{Forces on } i_p} F_{i_f X} \tag{3.43}$$

$$Q_{i_p Y} = \sum_{i_f=1}^{\text{Forces on } i_p} F_{i_f Y} \tag{3.44}$$

$$Q_{i_p Z} = \sum_{i_f=1}^{\text{Forces on } i_p} F_{i_f Z} \tag{3.45}$$

$$Q_{i_p \Phi} = \sum_{i_f=1}^{\text{forces on } i_p} \begin{Bmatrix} F_{i_f X} \\ F_{i_f Y} \\ F_{i_f Z} \end{Bmatrix}^T \frac{\partial [A]_{i_p}}{\partial \Phi_{i_p}} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_{i_p} + \sum_{i_p=1}^{\text{torque on } i_p} \begin{Bmatrix} T_{i_p X} \\ T_{i_p Y} \\ T_{i_p Z} \end{Bmatrix}^T [A]_{i_p} [B]_{i_p} \begin{Bmatrix} 1 \\ 0 \\ 0 \end{Bmatrix}_{i_p}. \tag{3.46}$$

$$Q_{i_p \Theta} = \sum_{i_f=1}^{\text{forces on } i_p} \begin{Bmatrix} F_{i_f X} \\ F_{i_f Y} \\ F_{i_f Z} \end{Bmatrix}^T \frac{\partial [A]_{i_p}}{\partial \Theta_{i_p}} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_{i_p} + \sum_{i_p=1}^{\text{torque on } i_p} \begin{Bmatrix} T_{i_p X} \\ T_{i_p Y} \\ T_{i_p Z} \end{Bmatrix}^T [A]_{i_p} [B]_{i_p} \begin{Bmatrix} 0 \\ 1 \\ 0 \end{Bmatrix}_{i_p} \tag{3.47}$$

$$Q_{i,\Psi} = \sum_{i_f=1}^{\text{forces on } i} \begin{Bmatrix} F_{i_f,x} \\ F_{i_f,y} \\ F_{i_f,z} \end{Bmatrix}^T \frac{\partial [A]_{i_f}}{\partial \Psi_{i_f}} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_{i_f} + \sum_{i_f=1}^{\text{torque on } i} \begin{Bmatrix} T_{i_f,x} \\ T_{i_f,y} \\ T_{i_f,z} \end{Bmatrix}^T [A]_{i_f} [B]_{i_f} \begin{Bmatrix} 0 \\ 0 \\ 1 \end{Bmatrix}_{i_f} . \quad (3.48)$$

3.4.3.2 External Forces and Torques

The external forces or torques available in OODS are gravitational forces, user-defined constant forces and torques, translational spring forces, and translational damping forces. Using Eq.(3.43)-(3.48), the contributions of these external forces and torques to the generalized forces are determined.

In OODS, the gravitational force on a part $F_{i,g}$ acts through the center-of-mass of the part, and it points in the negative Z direction of the ground frame (Figure 3.5), i.e.,

$$F_{i,g} = -m_i g K . \quad (3.49)$$

Substituting Eq.(3.49) into Eq.(3.43)-(3.48) gives

$$(Q_{i,z})_g = -m_i g \quad (3.50)$$

where $(Q_{i,z})_g$ is the contribution of the gravitational force in the Z direction and other Q 's receive zero contributions.

There are two kinds of applied forces and torques: action-only and action-reaction. The definitions are clarified in Figures 3.6-3.8. I and J markers are the points of application of the action and reaction. The action-only force or torque acts in the z' direction of the I marker only. The action-reaction forces occur in equal and opposite pairs that are colinear with the line joining I and J . The action-reaction torques can be defined in two ways. In the first case, the action-reaction torques occur in equal and opposite pairs that are parallel with the line joining I and J . In the second case, the torques are parallel with the z' axis of I .

The translational spring force always acts between two parts as a function of relative position (Figure 3.9). Consider a linear translational spring force between two markers I and J on the parts i_p and j_p . The spring force acting on the marker I is

$$F_{I_{\leftarrow}} = -k(l_{IJ} - l_0) \frac{l_v}{l_{IJ}}. \quad (3.51)$$

Here, k is the spring stiffness, l_0 is the free length of the spring and l_v is a vector representing the position of marker I relative to marker J . Note that $F_{I_{\leftarrow}}$ is positive when the spring is extended, and vice versa. Also,

$$F_{I_{\leftarrow}} = F_{I_{\leftarrow x}} I + F_{I_{\leftarrow y}} J + F_{I_{\leftarrow z}} K \quad (3.52)$$

$$l_v = l_{vx} I + l_{vy} J + l_{vz} K \quad (3.53)$$

$$l_{IJ} = \sqrt{l_{Ix}^2 + l_{Iy}^2 + l_{Iz}^2} \quad (3.54)$$

where

$$\begin{Bmatrix} l_{Ix} \\ l_{Iy} \\ l_{Iz} \end{Bmatrix} = \begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix}_{I_p} + [A]_{i_p} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_{I_m} - \begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix}_{J_p} - [A]_{j_p} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_{J_m}. \quad (3.55)$$

Then, Eq.(3.51) becomes

$$\begin{Bmatrix} F_{I_{\leftarrow x}} \\ F_{I_{\leftarrow y}} \\ F_{I_{\leftarrow z}} \end{Bmatrix} = -\frac{k(l_{IJ} - l_0)}{l_{IJ}} \begin{Bmatrix} l_{Ix} \\ l_{Iy} \\ l_{Iz} \end{Bmatrix}. \quad (3.56)$$

The generalized forces contributed by the spring forces on marker I can be obtained by substituting Eq.(3.56) into Eq.(3.43)-(3.48). The spring force acting on marker J ($F_{J_{\leftarrow}}$) has the same magnitude but opposite direction to $F_{I_{\leftarrow}}$. That is,

$$F_{J_{\leftarrow}} = -F_{I_{\leftarrow}}. \quad (3.57)$$

Similarly, the translational damping force always acts between two parts as a function of relative velocity (Figure 3.9). Consider a linear translational damping force between two markers I and J on the parts i_p and j_p . The damping force acting on the marker I $F_{I_{\text{dam}}}$ is

$$F_{I_{\text{dam}}} = -c \dot{l}_{IJ} \frac{l_{IJ}}{l_{IJ}} . \quad (3.58)$$

Here, c is the damping coefficient and $\dot{l}_{IJ}$ is the magnitude of the velocity of marker I relative to marker J . Also,

$$F_{I_{\text{dam}}} = F_{I_{\text{dam},x}} I + F_{I_{\text{dam},y}} J + F_{I_{\text{dam},z}} K \quad (3.59)$$

Differentiating Eq.(3.54) gives

$$\dot{l}_{IJ} = \frac{1}{l_{IJ}} (l_{IJ,x} \dot{l}_{IJ,x} + l_{IJ,y} \dot{l}_{IJ,y} + l_{IJ,z} \dot{l}_{IJ,z}) . \quad (3.60)$$

The components of the relative velocity vector are

$$\begin{Bmatrix} \dot{l}_{IJ,x} \\ \dot{l}_{IJ,y} \\ \dot{l}_{IJ,z} \end{Bmatrix} = \begin{Bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{Bmatrix}_{I_r} + \left(\frac{d}{dt}[A]_{I_r} \right) \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_{I_m} - \begin{Bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{Bmatrix}_{J_r} - \left(\frac{d}{dt}[A]_{J_r} \right) \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_{J_m} . \quad (3.61)$$

Then, Eq.(3.58) becomes

$$\begin{Bmatrix} F_{I_{\text{dam},x}} \\ F_{I_{\text{dam},y}} \\ F_{I_{\text{dam},z}} \end{Bmatrix} = -\frac{c \dot{l}_{IJ}}{l_{IJ}} \begin{Bmatrix} l_{IJ,x} \\ l_{IJ,y} \\ l_{IJ,z} \end{Bmatrix} \quad (3.62)$$

The generalized forces contributed by the damping forces on marker I can be obtained by substituting Eq.(3.62) into Eq.(3.43)-(3.48). The damping force acting on marker J $F_{J_{\text{dam}}}$ is

$$F_{J_{\text{dam}}} = -F_{I_{\text{dam}}} . \quad (3.63)$$

Finally, the generalized forces acting on the system are simply denoted by

$$\{Q\} = \{F(\{q\}, \{\dot{q}\})\} \quad (3.64)$$

where $\{F\}$ is a function of position and velocity. Eq.(3.64) can be expressed by an implicit function

$$\{L(\{Q\},\{q\},\{\dot{q}\})\} = \{Q\} - \{F(\{q\},\{\dot{q}\})\} = \{0\} . \quad (3.65)$$

3.4.4 Constraint Equations and Constraint Forces

The constraint equations that arise for each of the six joints described above are combinations of the At-Point, In-Plane, and Perpendicular constraints. These constraint equations are not explicit in time. Therefore, they are scleronomic constraints of the form

$$\{G(\{q\})\} = \{0\} . \quad (3.66)$$

3.4.4.1 At-Point Constraint

The At-Point constraint requires a point on one part to be in contact with a point on another part at all times. That is,

$$l_{IJ} = 0 \quad (3.67)$$

or

$$\begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix}_{i_p} + [A]_{i_p} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_{J_m} - \begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix}_{j_p} - [A]_{j_p} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_{J_m} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix} . \quad (3.68)$$

3.4.4.2 In-Plane Constraint

The In-Plane constraint requires a point on one part to lie in a plane fixed on another part. For example, marker I is always on the $y'z'$ -plane of the J marker frame if

$$l_{IJ} \cdot i'_{J_m} = 0 . \quad (3.69)$$

Other possible In-Plane constraints are

$$l_{IJ} \cdot j'_{J_m} = 0 \quad (3.70)$$

$$l_{IJ} \cdot k'_{J_m} = 0 . \quad (3.71)$$

In order to expand these constraints, one needs to transform the unit vectors of the J marker frame to the ground frame. That is,

$$\begin{Bmatrix} i' \\ j' \\ k' \end{Bmatrix}_{J_m} = [A]_{J_m}^T [A]_{J_r}^T \begin{Bmatrix} I \\ J \\ K \end{Bmatrix} = [D] \begin{Bmatrix} I \\ J \\ K \end{Bmatrix} = \begin{bmatrix} D_{11} & D_{12} & D_{13} \\ D_{21} & D_{22} & D_{23} \\ D_{31} & D_{32} & D_{33} \end{bmatrix} \begin{Bmatrix} I \\ J \\ K \end{Bmatrix}. \quad (3.72)$$

Then,

$$l_{IJ} \cdot i'_{J_m} = l_{IJ_x} D_{11} + l_{IJ_y} D_{12} + l_{IJ_z} D_{13} = 0 \quad (3.73)$$

$$l_{IJ} \cdot j'_{J_m} = l_{IJ_x} D_{21} + l_{IJ_y} D_{22} + l_{IJ_z} D_{23} = 0 \quad (3.74)$$

$$l_{IJ} \cdot k'_{J_m} = l_{IJ_x} D_{31} + l_{IJ_y} D_{32} + l_{IJ_z} D_{33} = 0. \quad (3.75)$$

3.4.4.3 Perpendicular Constraint

The Perpendicular constraint requires a unit vector fixed on one part to be always perpendicular to a unit vector fixed on another part. For example, the x' -axis of marker I is always perpendicular to the y' -axis of marker J if

$$i'_{I_m} \cdot j'_{J_m} = 0. \quad (3.76)$$

Other possible Perpendicular constraints are

$$k'_{I_m} \cdot i'_{J_m} = 0 \quad (3.77)$$

$$k'_{I_m} \cdot j'_{J_m} = 0 \quad (3.78)$$

$$k'_{I_m} \cdot k'_{J_m} = 0. \quad (3.79)$$

In order to expand these constraints, one needs to transform the unit vectors of the J marker frame to the I marker frame. That is,

$$\begin{Bmatrix} i' \\ j' \\ k' \end{Bmatrix}_{J_m} = [A]_{J_m}^T [A]_{J_r}^T [A]_{I_r} [A]_{I_m} \begin{Bmatrix} i' \\ j' \\ k' \end{Bmatrix}_{I_m} = [E] \begin{Bmatrix} i' \\ j' \\ k' \end{Bmatrix}_{I_m} = \begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{21} & E_{22} & E_{23} \\ E_{31} & E_{32} & E_{33} \end{bmatrix} \begin{Bmatrix} i' \\ j' \\ k' \end{Bmatrix}_{I_m}. \quad (3.80)$$

Note that $[A]_{J_m}^T$ and $[A]_{I_m}$ in $[D]$ and $[E]$ are constant. The set of Perpendicular constraints needed to represent the joints available are

$$i'_{I_m} \cdot j'_{J_m} = E_{21} = 0 \quad (3.81)$$

$$k'_{I_m} \cdot i'_{J_m} = E_{13} = 0 \quad (3.82)$$

$$k'_{I_m} \cdot j'_{J_m} = E_{23} = 0 \quad (3.83)$$

$$k'_{I_m} \cdot k'_{J_m} = E_{33} = 0. \quad (3.84)$$

Figures 3.10-3.15 show the six joints implemented in OODS. The I and J markers are shown with their unit vectors satisfying the possible necessary constraints. Table 3.1 summarizes all the independent constraint equations for the six joints.

3.4.4.4 Generalized Constraint Forces

The generalized constraint forces are derived with λ_{i_c} which are Lagrange multipliers consistent with the corresponding constraints G_{i_c} . Since the constraints do no virtual work,

$$\sum_{i_c}^{n_c} \lambda_{i_c} \delta G_{i_c} = 0. \quad (3.85)$$

Substituting

$$\delta G_{i_c} = \sum_{i_q}^{6n_p} \frac{\partial G_{i_c}}{\partial q_{i_q}} \delta q_{i_q} \quad (3.86)$$

into Eq.(3.85)

$$\sum_{i_c}^{n_c} \lambda_{i_c} \delta G_{i_c} = \sum_{i_c}^{n_c} \left(\lambda_{i_c} \sum_{i_q}^{6n_p} \frac{\partial G_{i_c}}{\partial q_{i_q}} \delta q_{i_q} \right) = \sum_{i_q}^{6n_p} \left(\sum_{i_c}^{n_c} \frac{\partial G_{i_c}}{\partial q_{i_q}} \lambda_{i_c} \right) \delta q_{i_q} = \sum_{i_q}^{6n_p} C_{i_q} \delta q_{i_q} \quad (3.87)$$

where C_{i_q} is the generalized constraint force associated with the generalized coordinate q_{i_q} .

$$C_{i_q} = \sum_{i_c}^{n_c} \frac{\partial G_{i_c}}{\partial q_{i_q}} \lambda_{i_c}. \quad (3.88)$$

Note that this expression is the same as the last term of the Lagrange's equation (Eq.(3.3)) in the associated coordinate.

One may need to express the constraint forces in the corresponding part frame or in the ground frame, instead of the generalized coordinates. The constraint forces and torques in the part frame of part i_p can be shown.

$$\begin{Bmatrix} C_{F_x} \\ C_{F_y} \\ C_{F_z} \end{Bmatrix}_{i_p} = [A]_{i_p}^T \begin{Bmatrix} C_X \\ C_Y \\ C_Z \end{Bmatrix}_{i_p} = [A]_{i_p}^T \begin{Bmatrix} \sum_{i_c}^{n_c} \frac{\partial G_{i_c}}{\partial X} \lambda_{i_c} \\ \sum_{i_c}^{n_c} \frac{\partial G_{i_c}}{\partial Y} \lambda_{i_c} \\ \sum_{i_c}^{n_c} \frac{\partial G_{i_c}}{\partial Z} \lambda_{i_c} \end{Bmatrix}_{i_p} \quad (3.89)$$

where C_{F_x} , C_{F_y} , and C_{F_z} are the constraint forces in the part frame. Also,

$$\begin{Bmatrix} C_{M_x} \\ C_{M_y} \\ C_{M_z} \end{Bmatrix}_{i_p} = [B]_{i_p} \begin{Bmatrix} C_\Phi \\ C_\Theta \\ C_\Psi \end{Bmatrix}_{i_p} = [B]_{i_p} \begin{Bmatrix} \sum_{i_c}^{n_c} \frac{\partial G_{i_c}}{\partial \Phi} \lambda_{i_c} \\ \sum_{i_c}^{n_c} \frac{\partial G_{i_c}}{\partial \Theta} \lambda_{i_c} \\ \sum_{i_c}^{n_c} \frac{\partial G_{i_c}}{\partial \Psi} \lambda_{i_c} \end{Bmatrix}_{i_p} \quad (3.90)$$

where C_{M_x} , C_{M_y} , and C_{M_z} are the constraint torques in the part frame. Likewise, the constraint forces and torques in the ground frame can be shown.

$$\begin{Bmatrix} C_{F_x} \\ C_{F_y} \\ C_{F_z} \end{Bmatrix}_{i_p} = \begin{Bmatrix} \sum_{i_c}^{n_c} \frac{\partial G_{i_c}}{\partial X} \lambda_{i_c} \\ \sum_{i_c}^{n_c} \frac{\partial G_{i_c}}{\partial Y} \lambda_{i_c} \\ \sum_{i_c}^{n_c} \frac{\partial G_{i_c}}{\partial Z} \lambda_{i_c} \end{Bmatrix}_{i_p} \quad (3.91)$$

where C_{F_x} , C_{F_y} , and C_{F_z} are the constraint forces in the ground frame. Also,

$$\begin{Bmatrix} C_{M_x} \\ C_{M_y} \\ C_{M_z} \end{Bmatrix}_{i_p} = [A]_{i_p} [B]_{i_p} \begin{Bmatrix} C_\Phi \\ C_\Theta \\ C_\Psi \end{Bmatrix}_{i_p} = [A]_{i_p} [B]_{i_p} \begin{Bmatrix} \sum_{i_c}^{n_c} \frac{\partial G_{i_c}}{\partial \Phi} \lambda_{i_c} \\ \sum_{i_c}^{n_c} \frac{\partial G_{i_c}}{\partial \Theta} \lambda_{i_c} \\ \sum_{i_c}^{n_c} \frac{\partial G_{i_c}}{\partial \Psi} \lambda_{i_c} \end{Bmatrix}_{i_p, i_p} \quad (3.92)$$

where C_{M_x} , C_{M_y} , and C_{M_z} are the constraint torques in the ground frame.

In this chapter, the equations of motion (Eq.(3.32)), momentum definitions (Eq.(3.35)), generalized force definitions (Eq.(3.65)), constraint equations (Eq.(3.66)), and constraint force equations (Eq.(3.88)-(3.92)) have been derived. These equations constitute the complete set of system equations.

Table 3.1: Joint Types and Constraint Equations

Joint Types	Constraint Equations			Constraints
	At-Point	In-Plane	Perpendicular	
Spherical	$l_v = 0$			3
Universal	$l_v = 0$		$k'_{I_m} \cdot k'_{J_m} = 0$	4
Revolute	$l_v = 0$		$k'_{I_m} \cdot i'_{J_m} = 0$ $k'_{I_m} \cdot j'_{J_m} = 0$	5
Translational		$l_v \cdot i'_{J_m} = 0$ $l_v \cdot j'_{J_m} = 0$	$i'_{I_m} \cdot j'_{J_m} = 0$ $k'_{I_m} \cdot i'_{J_m} = 0$ $k'_{I_m} \cdot j'_{J_m} = 0$	5
Cylindrical		$l_v \cdot i'_{J_m} = 0$ $l_v \cdot j'_{J_m} = 0$	$k'_{I_m} \cdot i'_{J_m} = 0$ $k'_{I_m} \cdot j'_{J_m} = 0$	4
Planar		$l_v \cdot k'_{J_m} = 0$	$k'_{I_m} \cdot i'_{J_m} = 0$ $k'_{I_m} \cdot j'_{J_m} = 0$	3

Figure 3.1: Multibody System

Figure 3.2: Axes Systems

Figure 3.3: Angular Velocity in Euler Axes System

Figure 3.4: External Force and Torque

Figure 3.5: Gravitational Force

Action-Only Force

Action-Only Torque

Figure 3.6: Action-Only Force and Torque

Figure 3.7: Action-Reaction Force

Definition 1

Definition 2

Figure 3.8: Action-Reaction Torques

Figure 3.9: Linear Spring and Damper

$$l_U = 0$$

Figure 3.10: Spherical Joint and Its Constraints

l_u	$=$	0
$k'_{i_n} \cdot k'_{j_n}$	$=$	0

Figure 3.11: Universal Joint and Its Constraints

l_{ij}	$=$	0
$k'_{i_m} \cdot i'_{j_m}$	$=$	0
$k'_{i_m} \cdot j'_{j_m}$	$=$	0

Figure 3.12: Revolute Joint and Its Constraints

Figure 3.13: Translational Joint and Its Constraints

$l_U \cdot i'_{J_m}$	$=$	0
$l_U \cdot j'_{J_m}$	$=$	0
$k'_{I_m} \cdot i'_{J_m}$	$=$	0
$k'_{I_m} \cdot j'_{J_m}$	$=$	0

Figure 3.14: Cylindrical Joint and Its Constraints

$l_u \cdot k'_{j_m}$	$=$	0
$k'_{i_m} \cdot i'_{j_m}$	$=$	0
$k'_{i_m} \cdot j'_{j_m}$	$=$	0

Figure 3.15: Planar Joint and Its Constraints

CHAPTER IV

SOLUTION SCHEME FOR SYSTEM EQUATIONS

4.1 Introductory Comments

The system equations are a set of highly nonlinear and loosely coupled differential-algebraic equations. They pose an initial-value problem and have to rely on a numerical integration method for their solution. A solution scheme requires the simultaneous solution of the nonlinear algebraic equations and nonlinear differential equations. In OODS, the Newton-Raphson method and a stiffly-stable integration method are combined to build a corrector method. Two standard predictor methods are adopted to complete the solution scheme.

4.2 Linearization of System Equations

The system state variables can be summarized as

$$q = \begin{Bmatrix} q_1, \\ q_2, \\ \vdots \\ q_i, \\ \vdots \\ q_n, \end{Bmatrix} \quad u = \begin{Bmatrix} u_1, \\ u_2, \\ \vdots \\ u_i, \\ \vdots \\ u_n, \end{Bmatrix} \quad Q = \begin{Bmatrix} Q_1, \\ Q_2, \\ \vdots \\ Q_i, \\ \vdots \\ Q_n, \end{Bmatrix} \quad \lambda = \begin{Bmatrix} \lambda_{1c} \\ \lambda_{2c} \\ \vdots \\ \lambda_{ic} \\ \vdots \\ \lambda_{nc} \end{Bmatrix} \quad (4.1)$$

where

$$q_i = \{q\}_{i,} = \begin{Bmatrix} X \\ Y \\ Z \\ \Phi \\ \Theta \\ \Psi \end{Bmatrix}_{i,} \quad u_i = \{u\}_{i,} = \begin{Bmatrix} u_x \\ u_y \\ u_z \\ u_\phi \\ u_\theta \\ u_\psi \end{Bmatrix}_{i,} \quad Q_i = \{Q\}_{i,} = \begin{Bmatrix} Q_x \\ Q_y \\ Q_z \\ Q_\phi \\ Q_\theta \\ Q_\psi \end{Bmatrix}_{i,} \quad (4.2)$$

for part i_p .

The system equations in implicit forms (Eq.(3.32), (3.35), (3.65), and (3.66)) are

$$M(q, \dot{q}, \dot{u}, \lambda, Q) = \dot{u} - \left[\frac{\partial T}{\partial \dot{q}} \right]^T - Q - \left[\frac{\partial G}{\partial \dot{q}} \right]^T \lambda = 0 \quad (4.3)$$

$$P(q, \dot{q}, u) = u - [C] \dot{q} = 0 \quad (4.4)$$

$$G(q) = 0 \quad (4.5)$$

$$N(Q, q, \dot{q}) = Q - F(q, \dot{q}) = 0. \quad (4.6)$$

Using Taylor series and truncating the second and higher order terms,

$$M|_c + \frac{\partial M}{\partial q} \Big|_c \Delta q + \frac{\partial M}{\partial \dot{q}} \Big|_c \Delta \dot{q} + \frac{\partial M}{\partial \dot{u}} \Big|_c \Delta \dot{u} + \frac{\partial M}{\partial \lambda} \Big|_c \Delta \lambda + \frac{\partial M}{\partial Q} \Big|_c \Delta Q = 0 \quad (4.7)$$

$$P|_c + \frac{\partial P}{\partial q} \Big|_c \Delta q + \frac{\partial P}{\partial \dot{q}} \Big|_c \Delta \dot{q} + \frac{\partial P}{\partial u} \Big|_c \Delta u = 0 \quad (4.8)$$

$$G|_c + \frac{\partial G}{\partial q} \Big|_c \Delta G = 0 \quad (4.9)$$

$$N|_c + \frac{\partial N}{\partial q} \Big|_c \Delta q + \frac{\partial N}{\partial \dot{q}} \Big|_c \Delta \dot{q} + \frac{\partial N}{\partial Q} \Big|_c \Delta Q = 0 \quad (4.10)$$

where

$$\Delta q = q_{c+1} - q_c \quad (4.11)$$

$$\Delta u = u_{c+1} - u_c \quad (4.12)$$

$$\Delta \lambda = \lambda_{c+1} - \lambda_c \quad (4.13)$$

$$\Delta Q = Q_{c+1} - Q_c. \quad (4.14)$$

and the subscript c denotes the iteration counter. Eq.(4.7)-(4.10) can be combined into a matrix form

$$\begin{bmatrix} \frac{\partial M}{\partial q} & \frac{\partial M}{\partial \dot{q}} & [0] & \frac{\partial M}{\partial \dot{u}} & \frac{\partial M}{\partial \lambda} & \frac{\partial M}{\partial Q} \\ \frac{\partial P}{\partial q} & \frac{\partial P}{\partial \dot{q}} & \frac{\partial P}{\partial u} & [0] & [0] & [0] \\ \frac{\partial G}{\partial q} & [0] & [0] & [0] & [0] & [0] \\ \frac{\partial N}{\partial q} & \frac{\partial N}{\partial \dot{q}} & [0] & [0] & [0] & \frac{\partial N}{\partial Q} \end{bmatrix} \begin{Bmatrix} \Delta q \\ \Delta \dot{q} \\ \Delta u \\ \Delta \dot{u} \\ \Delta \lambda \\ \Delta Q \end{Bmatrix}_c = - \begin{Bmatrix} M(q, \dot{q}, \dot{u}, \lambda, Q) \\ P(q, \dot{q}, u) \\ G(q) \\ N(Q, q, \dot{q}) \end{Bmatrix}_c \quad (4.15)$$

where the coefficient matrix is called the system Jacobian, and its size is $3n_q + n_c$.

4.3 Corrector Method

4.3.1 Gear Integration Formulas

The Gear implicit multistep integration formula has to be incorporated into the linearized system equations to solve the differential equations [Gear 1971]. A general form of the Gear formula is

$$\dot{y}^{t+1} = - \frac{1}{h\beta_0} \left(y^{t+1} - \sum_{j=1}^k \alpha_j y^{t-j+1} \right) \quad (4.16)$$

where y stands for any set of state variables, h is the integration-step size, and t denotes the integration step counter. β_0 and α_j are the Gear coefficients given in Table 4.1 such that Eq.(4.16) gives a polynomial solution of $f(y, \dot{y}, t) = 0$ exact up to order k . In other words, the formula gives the exact $\dot{y}$ at time $t+1$ in terms of the present value and k past values, provided y is a polynomial of degree k or less. This method is called implicit or closed because y^{t+1} and $\dot{y}^{t+1}$ are both unknowns in Eq.(4.16). The Gear coefficients have been proven to give a stiffly stable algorithm for $k \leq 6$ [Gear 1971].

Stiff differential equations frequently arise in engineering problems from systems with greatly differing response time constants. The time constant is one over the magnitude of the real part of a system eigenvalue. Lambert [1978] defined the stiffness ratio by

$$s = \frac{\text{Max} |\text{Re } e|}{\text{Min} |\text{Re } e|} \quad (4.17)$$

where e is the eigenvalues of the system Jacobian. Dynamic simulation with stiffness ratios up to 10^6 may arise in real engineering problems.

In non-stiff methods, the step size must not be greater than the smallest time constant to ensure numerical stability. This requirement results in wasteful computation if the fast components of response are actually negligible when compared with the slower components of response. Small step size can make roundoff errors significant too.

Stiff methods, however, are designed to ignore the negligible but fast components of response without showing numerical instability. Larger step sizes can be used and fewer computations can be achieved. Details of stability criteria are discussed in [Gear 1971] and [Chua 1975].

4.3.2 Iteration Scheme

Eq.(4.16) can be used to eliminate $\dot{q}$ and $\dot{u}$ in Eq.(4.15). Taking the total differential of Eq.(4.16),

$$\Delta \dot{y}^{t+1} = - \frac{1}{h\beta_0} \Delta y^{t+1} . \quad (4.18)$$

This is true because the terms in the summation in Eq.(4.16) remain constant at each iteration at $t+1$.

Substituting

$$\Delta \dot{q}^{t+1} = - \frac{1}{h\beta_0} \Delta q^{t+1} \quad (4.19)$$

$$\Delta \dot{u}^{t+1} = - \frac{1}{h\beta_0} \Delta u^{t+1} \quad (4.20)$$

into Eq.(4.15) gives

$$\begin{bmatrix}
\frac{\partial M}{\partial q} - \left(\frac{1}{h\beta_0}\right) \frac{\partial M}{\partial \dot{q}} & -\left(\frac{1}{h\beta_0}\right) \frac{\partial M}{\partial \dot{u}} & \frac{\partial M}{\partial \lambda} & \frac{\partial M}{\partial Q} \\
\frac{\partial P}{\partial q} - \left(\frac{1}{h\beta_0}\right) \frac{\partial P}{\partial \dot{q}} & \frac{\partial P}{\partial u} & [0] & [0] \\
\frac{\partial G}{\partial q} & [0] & [0] & [0] \\
\frac{\partial N}{\partial q} - \left(\frac{1}{h\beta_0}\right) \frac{\partial N}{\partial \dot{q}} & [0] & [0] & \frac{\partial N}{\partial Q}
\end{bmatrix}
\begin{Bmatrix}
\Delta q \\
\Delta u \\
\Delta \lambda \\
\Delta Q
\end{Bmatrix}_c =
\begin{Bmatrix}
M(q, \dot{q}, \dot{u}, \lambda, Q) \\
P(q, \dot{q}, u) \\
G(q) \\
N(Q, q, \dot{q})
\end{Bmatrix}_c. \quad (4.21)$$

The system Jacobian has a fixed and sparse structure. The column matrix on the right contains the residual errors at each iteration.

Each partition of the system Jacobian can be expressed as

$$\left[\frac{\partial M}{\partial q} \right] = -\frac{\partial}{\partial q} \left[\frac{\partial T}{\partial q} \right]^T - \frac{\partial}{\partial q} \left(\left[\frac{\partial G}{\partial q} \right]^T \lambda \right) \quad (4.22)$$

$$\left[\frac{\partial M}{\partial \dot{q}} \right] = -\frac{\partial}{\partial \dot{q}} \left[\frac{\partial T}{\partial q} \right]^T \quad (4.23)$$

$$\left[\frac{\partial M}{\partial \dot{u}} \right] = [I] \quad (4.24)$$

$$\left[\frac{\partial M}{\partial \lambda} \right] = -\left[\frac{\partial G}{\partial q} \right]^T \quad (4.25)$$

$$\left[\frac{\partial M}{\partial Q} \right] = -[I] \quad (4.26)$$

$$\left[\frac{\partial P}{\partial q} \right] = -\frac{\partial}{\partial q} ([C] \dot{q}) \quad (4.27)$$

$$\left[\frac{\partial P}{\partial \dot{q}} \right] = -[C] \quad (4.28)$$

$$\left[\frac{\partial P}{\partial u} \right] = [I] \quad (4.29)$$

$$\left[\frac{\partial N}{\partial q} \right] = -\frac{\partial F}{\partial q} \quad (4.30)$$

$$\left[\frac{\partial N}{\partial \dot{q}} \right] = -\frac{\partial F}{\partial \dot{q}} \quad (4.31)$$

$$\left[\frac{\partial N}{\partial Q} \right] = [I] . \quad (4.32)$$

Here, $[I]$ stands for an identity matrix. Substituting these partitions into Eq.(4.21) gives

$$\begin{bmatrix} \frac{\partial M}{\partial q} - \left(\frac{1}{h\beta_0} \right) \frac{\partial M}{\partial \dot{q}} & -\frac{1}{h\beta_0}[I] & -\left[\frac{\partial G}{\partial q} \right]^T & -[I] \\ \frac{\partial P}{\partial q} - \left(\frac{1}{h\beta_0} \right) \frac{\partial P}{\partial \dot{q}} & [I] & [0] & [0] \\ \frac{\partial \bar{G}}{\partial q} & [0] & [0] & [0] \\ \frac{\partial N}{\partial q} - \left(\frac{1}{h\beta_0} \right) \frac{\partial N}{\partial \dot{q}} & [0] & [0] & [I] \end{bmatrix}_c^{t+1} \begin{Bmatrix} \Delta q \\ \Delta u \\ \Delta \lambda \\ \Delta Q \end{Bmatrix}_c^{t+1} =$$

$$-\begin{Bmatrix} M(q, \dot{q}, \dot{u}, \lambda, Q) \\ P(q, \dot{q}, u) \\ G(q) \\ N(Q, q, \dot{q}) \end{Bmatrix}_c^{t+1} \quad (4.33)$$

at the integration step $t+1$.

Eq.(4.33) is used as a corrector to solve the state variables q , u , λ , and Q iteratively. During the corrector iterations, the successive iteration scheme (Eq.(4.11)-(4.14)) is used,

and $\dot{q}$ and $\dot{u}$ are obtained from Eq.(4.16). The iteration process stops either when convergence criteria is satisfied or when the maximum iteration number is reached. The convergence criteria used is

$$\frac{\sum_{i_s=1}^{n_s} |y_{i_s,(c+1)} - y_{i_s,(c)}|}{n_s \max |y_{i_s,(c+1)}|} < \text{Tolerance} \quad (4.34)$$

where y_{i_s} represents a state variable and n_s is the number of state variables.

4.4 Predictor-Corrector Schemes

The corrector process converges if the starting values at each iteration are sufficiently close to the solutions. There are many possible predictor methods to make good guesses for the corrector. OODS offers two predictor methods; a single step Euler method and a multistep interpolation method. These two predictor methods are combined with the above corrector method to build two separate predictor-corrector schemes.

4.4.1 Predictor-Corrector I

Figure 4.1 shows the Predictor-Corrector I of OODS. The predictor used is the single step Euler method

$$y_p^{t+1} = y' + h\dot{y}' \quad (4.35)$$

where the subscript p denotes the predicted values. q and u are predicted using this equation. However, the equation cannot be used to predict λ and Q since $\dot{\lambda}$ and $\dot{Q}$ are not available. Instead, λ and Q are predicted with

$$y_p^{t+1} = y' . \quad (4.36)$$

Predictor-Corrector I

- (1) sets the integration step counter to zero, i.e., $t = 0$.
- (2) sets the integration order to one, i.e., $k = 1$.
- (3) solves for q_p^{t+1} , u_p^{t+1} using q' , u' , $\dot{q}'$, $\dot{u}'$ in the Euler formula (Eq.(4.35)).

- (4) solves for λ_p^{i+1} , Q_p^{i+1} using λ' and Q' in Eq.(4.36).
- (5) solves for $\dot{q}_p^{i+1}$, $\dot{u}_p^{i+1}$ using the Gear formula (Eq.(4.16)).
- (6) sets the iteration counter to one, i.e., $c = 1$.
- (7) sets q_c^{i+1} , u_c^{i+1} , λ_c^{i+1} , Q_c^{i+1} equal to q_p^{i+1} , u_p^{i+1} , λ_p^{i+1} , Q_p^{i+1} , $\dot{q}_p^{i+1}$, $\dot{u}_p^{i+1}$.
- (8) solves for q_{c+1}^{i+1} , u_{c+1}^{i+1} , λ_{c+1}^{i+1} , Q_{c+1}^{i+1} using q_c^{i+1} , u_c^{i+1} , λ_c^{i+1} , Q_c^{i+1} in the corrector (Eq.(4.33)).
- (9) solves for $\dot{q}_{c+1}^{i+1}$ and $\dot{u}_{c+1}^{i+1}$ in the Gear formula (Eq.(4.16)).
- (10) checks whether the convergence criterion (Eq.(4.35)) is satisfied or $c = c_{\max}$.
If true, it sets q^{i+1} , u^{i+1} , λ^{i+1} , Q^{i+1} , $\dot{q}^{i+1}$, $\dot{u}^{i+1}$ equal to q_{c+1}^{i+1} , u_{c+1}^{i+1} , λ_{c+1}^{i+1} , Q_{c+1}^{i+1} , $\dot{q}_{c+1}^{i+1}$, $\dot{u}_{c+1}^{i+1}$ and then, goes to step 13.
- (11) increases the iteration counter by one, i.e., $c = c + 1$.
- (12) repeats steps 8, 9, 10, 11.
- (13) checks whether $t = t_{\max}$. If true, it stops the predictor-corrector process.
Otherwise, it increases t by one, i.e., $t = t + 1$.
- (14) checks whether $k = k_{\max}$. If true, it sets k equal to $k_{\max}$. Otherwise, it increases k by one, i.e., $k = k + 1$.
- (15) repeats steps from 3 to 14.

4.4.2 Predictor-Corrector II

Figure 4.1 also shows the flowchart of Predictor-Corrector II if the Euler predictor block is substituted with the interpolation block. The predictor used is the interpolation formula [Chua 1975]

$$y_p^{i+1} = \sum_{j=1}^{k+1} \gamma_j y^{i-j+1} \quad (4.37)$$

where k is the order of the polynomial fit through the past values and γ_j is a set of coefficients. Table 4.2 shows the values of γ 's for $k \leq 6$.

Predictor-Corrector II

- (1) follows steps 1 and 2 of Predictor-Corrector I,
- (2) solves for q_p^{i+1} , u_p^{i+1} , λ_p^{i+1} , Q_p^{i+1} using q' , u' , λ' , Q' in the interpolation formula (Eq.(4.37)).
- (3) follows steps 5 to 15 of Predictor-Corrector I.

The order k of the interpolation formula is actually independent of the order k in the Gear formula of the corrector. In OODS, however, they have been set to be equal.

Table 4.1: Gear Coefficients

k	β_0	α_1	α_2	α_3	α_4	α_5	α_6
1	-1	1					
2	-2/3	4/3	-1/3				
3	-6/11	18/11	-9/11	2/11			
4	-12/25	48/25	-36/25	16/25	-3/25		
5	-60/137	300/137	-300/137	200/137	-75/137	12/137	
6	-60/147	360/147	-450/147	400/147	-225/147	72/147	-10/147

Table 4.2: Interpolation Predictor Coefficients

k	γ_1	γ_2	γ_3	γ_4	γ_5	γ_6
1	1					
2	2	-1				
3	3	-3	1			
4	4	-6	4	-1		
5	5	-10	10	-5	1	
6	6	-15	20	-15	6	-1

Figure 4.1: Predictor-Corrector Scheme

CHAPTER V

INITIAL CONDITIONS ANALYSIS

5.1 Introductory Comments

The system equations contain differential equations which require initial conditions for a complete solution. These initial conditions which specify the initial system states must be provided by the user. To ensure that they are correct and accurate, they are checked to satisfy the system constraints. If they do not satisfy the constraints, they may be adjusted automatically in some optimum way, or the user is informed of the problem. Only when the initial conditions are correct, can the simulation begin and produce meaningful results.

This chapter describes how the initial positions, velocities, accelerations, and Lagrange multipliers are checked for consistency.

5.2 Position Initial Conditions

User input initial positions can be a combination of exact, approximate, or omitted data. A consistent set of initial positions is obtained by a weighted least squares method. A large weight is assigned to an input of high accuracy, a low weight to an approximate input, and zero to an omitted data.

The objective function to be minimized is

$$E_1 = \frac{1}{2} \sum_{i_q=1}^{n_q} W_{1,i_q} (q_{i_q} - q_{i_q}^0)^2 + \sum_{i_c=1}^{n_c} \mu_{i_c} G_{i_c} \quad (5.1)$$

where n_q is the number of system coordinates, $q_{i_q}^0$ is a user input initial position, n_c is the number of system constraints, W_{1,i_q} is a weight factor, and μ_{i_c} is a Lagrange multiplier associated with a position constraint. For a minimum,

$$\frac{\partial E_1}{\partial q_{i_q}} = 0 \quad (i_q = 1, 2, 3, \dots, n_q) \quad (5.2)$$

$$\frac{\partial E_1}{\partial \mu_{i_c}} = 0 \quad (i_c = 1, 2, 3, \dots, n_c) \quad (5.3)$$

which give the implicit functions

$$f(q, \mu) = [W_1](q - q^0) + \left[\frac{\partial G}{\partial q} \right]^T \mu = 0 \quad (5.4)$$

$$G(q) = 0. \quad (5.5)$$

$[W_1]$ is a square diagonal matrix containing the weight factors.

Linearizing these implicit functions using Taylor series,

$$f|_c + \frac{\partial f}{\partial q}|_c \Delta q + \frac{\partial f}{\partial \mu}|_c \Delta \mu = 0 \quad (5.6)$$

$$G|_c + \frac{\partial G}{\partial q}|_c \Delta q = 0 \quad (5.7)$$

where

$$\Delta q = q_{c+1} - q_c \quad (5.8)$$

$$\Delta \mu = \mu_{c+1} - \mu_c \quad (5.9)$$

and the subscript c denotes the iteration counter. Eq.(5.6)-(5.7) can be combined into a matrix form

$$\begin{bmatrix} \frac{\partial f}{\partial q} & \frac{\partial f}{\partial \mu} \\ \frac{\partial G}{\partial q} & [0] \end{bmatrix}_c \begin{Bmatrix} \Delta q \\ \Delta \mu \end{Bmatrix}_c = - \begin{Bmatrix} f(q, \mu) \\ G(q) \end{Bmatrix}_c \quad (5.10)$$

where the coefficient matrix is called the position Jacobian, and its size is $n_q + n_c$.

Differentiating Eq.(5.4) gives

$$\left[\frac{\partial f}{\partial \mathbf{q}} \right] = [\mathbf{W}_1] + \frac{\partial}{\partial \mathbf{q}} \left(\left[\frac{\partial \mathbf{G}}{\partial \mathbf{q}} \right]^T \boldsymbol{\mu} \right) \quad (5.11)$$

$$\left[\frac{\partial f}{\partial \boldsymbol{\mu}} \right] = \left[\frac{\partial \mathbf{G}}{\partial \mathbf{q}} \right]^T. \quad (5.12)$$

Substituting Eq.(5.11)-(5.12) into Eq.(5.6) gives

$$\left[\begin{array}{c|c} \left[[\mathbf{W}_1] + \left[\frac{\partial}{\partial \mathbf{q}} \left(\left[\frac{\partial \mathbf{G}}{\partial \mathbf{q}} \right]^T \boldsymbol{\mu} \right) \right] \right] & \left[\frac{\partial \mathbf{G}}{\partial \mathbf{q}} \right]^T \\ \hline \left[\frac{\partial \mathbf{G}}{\partial \mathbf{q}} \right] & [0] \end{array} \right]_c \begin{Bmatrix} \Delta \mathbf{q} \\ \Delta \boldsymbol{\mu} \end{Bmatrix}_c = - \begin{Bmatrix} f(\mathbf{q}, \boldsymbol{\mu}) \\ \mathbf{G}(\mathbf{q}) \end{Bmatrix}_c. \quad (5.13)$$

From Eq.(5.4),

$$f(\mathbf{q}, \boldsymbol{\mu})|_c = [\mathbf{W}_1](\mathbf{q} - \mathbf{q}^0)|_c + \left[\frac{\partial \mathbf{G}}{\partial \mathbf{q}} \right]^T \boldsymbol{\mu} \Big|_c. \quad (5.14)$$

Substituting Eq.(5.9) and Eq.(5.14) into Eq.(5.13) gives

$$\left[\begin{array}{c|c} \left[[\mathbf{W}_1] + \left[\frac{\partial}{\partial \mathbf{q}} \left(\left[\frac{\partial \mathbf{G}}{\partial \mathbf{q}} \right]^T \boldsymbol{\mu} \right) \right] \right] & \left[\frac{\partial \mathbf{G}}{\partial \mathbf{q}} \right]^T \\ \hline \left[\frac{\partial \mathbf{G}}{\partial \mathbf{q}} \right] & [0] \end{array} \right]_c \begin{Bmatrix} \Delta \mathbf{q}_c \\ \boldsymbol{\mu}_{c+1} \end{Bmatrix} = - \begin{Bmatrix} [\mathbf{W}_1](\mathbf{q} - \mathbf{q}^0) \\ \mathbf{G} \end{Bmatrix}_c. \quad (5.15)$$

This matrix equation is solved for consistent initial positions iteratively. The second term in the upper left quadrant contains rather complicated trigonometric functions.

5.3 Velocity Initial Conditions

A consistent set of initial velocities is computed in a similar fashion. However, only one iteration is needed in the weighted least squares method.

The objective function to be minimized is

$$E_2 = \frac{1}{2} \sum_{i_q=1}^{n_q} W_{2,i_q} (\dot{q}_{i_q} - \dot{q}_{i_q}^0)^2 + \sum_{i_c=1}^{n_c} v_{i_c} \frac{dG_{i_c}}{dt} \quad (5.16)$$

where $\dot{q}_{i_q}^0$ is a user input initial velocity, W_{2,i_q} is a weight factor, and v_{i_c} is a Lagrange multiplier associated with a velocity constraint. For a minimum,

$$\frac{\partial E_2}{\partial \dot{q}_{i_q}} = 0 \quad (i_q = 1, 2, 3, \dots, n_q) \quad (5.17)$$

$$\frac{\partial E_2}{\partial v_{i_c}} = 0 \quad (i_c = 1, 2, 3, \dots, n_c) \quad (5.18)$$

which give the implicit functions

$$\begin{aligned} x(\dot{q}, \mathbf{v}) &= [W_2](\dot{q} - \dot{q}^0) + \frac{\partial}{\partial \dot{q}} \left(\left[\frac{dG}{dt} \right]^T \mathbf{v} \right) \\ &= [W_2](\dot{q} - \dot{q}^0) + \frac{\partial}{\partial \dot{q}} \left(\left[\frac{\partial G}{\partial \mathbf{q}} \dot{q} + \frac{\partial G}{\partial t} \right]^T \mathbf{v} \right) \\ &= [W_2](\dot{q} - \dot{q}^0) + \left[\frac{\partial G}{\partial \mathbf{q}} \right]^T \mathbf{v} = 0 \end{aligned} \quad (5.19)$$

$$y(q, \dot{q}) = \frac{dG}{dt} = \frac{\partial G}{\partial \mathbf{q}} \dot{q} + \frac{\partial G}{\partial t} = 0. \quad (5.20)$$

Since the constraints are scleronomic,

$$\frac{\partial G}{\partial t} = 0. \quad (5.21)$$

Substituting Eq.(5.20) into Eq.(5.21) gives

$$y(q, \dot{q}) = \frac{\partial G}{\partial \mathbf{q}} \dot{q} = 0. \quad (5.22)$$

Combining Eq.(5.19) and Eq.(5.22) gives

$$\begin{bmatrix} [W_2] & \left[\frac{\partial G}{\partial q} \right]^T \\ \left[\frac{\partial G}{\partial q} \right] & [0] \end{bmatrix} \begin{Bmatrix} \dot{q} \\ v \end{Bmatrix} = \begin{Bmatrix} [W_2] \dot{q}^0 \\ 0 \end{Bmatrix} \quad (5.23)$$

which is a system of $(n_q + n_c)$ equations. The consistent initial velocities are solved for immediately.

5.4 Acceleration Initial Conditions

The initial accelerations and Lagrange multipliers are not necessary to start the predictor-corrector process. However, they may be needed as part of the output, or they can be used by some predictor methods to make good guesses for the corrector.

The initial accelerations and Lagrange multipliers are obtained from the equations of motion and the second derivatives of constraint equations. Differentiating Eq.(3.27) gives

$$\begin{aligned} \begin{Bmatrix} \ddot{u}_x \\ \ddot{u}_y \\ \ddot{u}_z \\ \ddot{u}_\phi \\ \ddot{u}_\theta \\ \ddot{u}_\psi \end{Bmatrix}_{i,} &= \begin{bmatrix} m[I] & [0] \\ [0] & [B]^T [J] [B] \end{bmatrix}_{i,} \begin{Bmatrix} \ddot{X} \\ \ddot{Y} \\ \ddot{Z} \\ \ddot{\Phi} \\ \ddot{\Theta} \\ \ddot{\Psi} \end{Bmatrix}_{i,} + \begin{bmatrix} [0] & [0] \\ [0] & \frac{\partial}{\partial t} ([B]^T [J] [B]) \end{bmatrix}_{i,} \begin{Bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \\ \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i,} \\ &= [C]_{i,} \ddot{q}_{i,} + [D]_{i,} \dot{q}_{i,} . \end{aligned} \quad (5.24)$$

Note that $[C]_{i,}$ is a function of positions only while $[D]_{i,}$ is a function of positions and velocities. Substituting Eq.(5.24) into the equations of motion and rearranging gives

$$[C]_{i,} \ddot{q}_{i,} - \left[\frac{\partial G}{\partial q_{i,}} \right]^T \lambda = Q_{i,} - [D]_{i,} \dot{q}_{i,} + \left[\frac{\partial T}{\partial q_{i,}} \right]^T . \quad (5.25)$$

For the whole system,

$$[C]\ddot{q} - \left[\frac{\partial G}{\partial q} \right]^T \lambda = Q - [D]\dot{q} + \left[\frac{\partial T}{\partial q} \right]^T. \quad (5.26)$$

Differentiating Eq.(5.22) gives

$$\frac{d}{dt} \left(\frac{\partial G}{\partial q} \dot{q} \right) = \left(\frac{d}{dt} \frac{\partial G}{\partial q} \right) \dot{q} + \frac{\partial G}{\partial q} \ddot{q} = \left(\frac{\partial^2 G}{\partial q^2} \dot{q} + \frac{\partial}{\partial t} \frac{\partial G}{\partial q} \right) \dot{q} + \frac{\partial G}{\partial q} \ddot{q} = 0. \quad (5.27)$$

Since G is a function of positions only,

$$\frac{\partial}{\partial t} \frac{\partial G}{\partial q} = 0. \quad (5.28)$$

Substituting Eq.(5.28) into Eq.(5.27) gives

$$\left(\frac{\partial}{\partial q} \left(\frac{\partial G}{\partial q} \dot{q} \right) \right) \dot{q} + \frac{\partial G}{\partial q} \ddot{q} = 0. \quad (5.29)$$

Eq.(5.26) and Eq.(5.29) can be combined into a matrix form

$$\begin{bmatrix} -[C] & \left[\frac{\partial G}{\partial q} \right]^T \\ \left[\frac{\partial G}{\partial q} \right] & [0] \end{bmatrix} \begin{Bmatrix} \ddot{q} \\ \lambda \end{Bmatrix} = \begin{Bmatrix} -Q + [D]\dot{q} - \left[\frac{\partial T}{\partial q} \right]^T \\ - \left(\frac{\partial}{\partial q} \left(\frac{\partial G}{\partial q} \dot{q} \right) \right) \dot{q} \end{Bmatrix}, \quad (5.30)$$

which is a system of $(n_q + n_c)$ equations. All the terms in the coefficient matrix and the right hand side can be evaluated from the initial positions and velocities calculated above. Hence $\ddot{q}$ and λ are easily solved for without iteration.

Note that the coefficient matrices in Eq.(5.15), Eq.(5.23), and Eq.(5.30) are similar to each other in that they contain the same partial derivatives of the constraint equations. Note also that λ is not related to μ and ν in any way; only λ arises in the equations of motion.

CHAPTER VI

OBJECT-ORIENTED DYNAMICS SIMULATOR (OODS)

6.1 Introductory Comments

OODS incorporates the dynamics theory and numerical methods developed with the power of OOP. The following are explained in this chapter.

First, the organization of OODS is explained in terms of its classes and messages. The classes available at present are System, Part, Joint, Spring, Damper, Marker, UserForce, and Request. The objects from these classes interact through message sendings to achieve the dynamic simulation of a mechanical system.

Next, the input to OODS is explained. The input is usually written in a workspace of Smalltalk-80; it can be modified and saved separately from the other workspaces.

Finally, several program test runs are discussed. The test systems are first illustrated, and the inputs to OODS are shown. Then the positions, velocities, accelerations, and constraint forces of the systems are plotted and checked with known results.

6.2 Program Organization

6.2.1 Classes of OODS

The organization of the classes in OODS is shown in Figure 6.1. The class Object is the top superclass of all other classes in Smalltalk-80. The subclass System is defined to represent a dynamic system. The class System has methods to formulate the system equations, solve the equations numerically, and store the system states in class variables. It also has dictionaries of parts, joints, springs, dampers, user forces, and requests. In general, System deals with variables and behaviors that are global to the dynamic system.

The classes Part, Marker, Joint, Spring, Damper, UserForce, and Request are defined as subclasses of System. Part is the class of all parts, i.e., all parts are instances of Part. Instances of Part have methods to initialize themselves with masses, moment-of-

inertias, etc., in instance variables. They also have methods to return their states and contributions to system dynamics when required.

The class Marker is a subclass of Part. Marker contains all markers in the system as its instances. The markers are used to identify center-of-masses, force application points, or joint attachment points. Instances of Marker know of parts to which they belong. They also carry transformation matrices between their own frames and part frames or ground frame as instance variables. They have methods to calculate relative positions and velocities between them.

The class Joint contains all joints in the system as its instances. Each joint knows it is one of the six possible joints. Its joint type is stored in an instance variable. Knowing its type, it will apply the proper combination of At-Point, In-Plane, and Perpendicular constraints when requested to do so.

The classes Spring and Damper contain all the translational springs and dampers in the system. Each spring and damper is able to determine its force acting on parts and return its contribution to the system equations when requested to do so.

The class Request contains objects that output numerical information into external files. Each request instance represents a user request for output of position, velocity, acceleration, momentum, constraint force, or generalized force of a part throughout the simulation. Each user request is written to a separate external file. Unlike other classes, Request represents conceptual (invisible) objects.

6.2.2 Message Sendings

The flowchart of the solution procedure of OODS is shown in Figure 6.2. A message sending diagram is introduced in Figure 6.3 where the arrows show flows of messages between the instances of the classes of OODS.

6.2.3 Input to OODS

The input to OODS is actually a highly readable program in Smalltalk-80. Therefore, no parser is needed to translate the input into executable code. This is a major advantage of Smalltalk-80 over other languages. The input program is usually written in a workspace, but it can be made a method too. Several examples of input programs are shown in Figures 6.5, 6.13, 6.18, 6.23, 6.27, 6.32, 6.42, 6.47, and 6.53.

By Smalltalk-80 convention, the program begins with a comment about the problem to be solved. A comment is any text within two double quotes. Additional comments can be put anywhere in the program. For example,

```
" This provides the input for the simulation of a bar pendulum in plane motion. "
```

Next, the names of the system, all the parts, joints, springs, or markers are declared in between two vertical bars. Each name must have no space in itself because the names are separated by spaces. The names may be listed in any order. For example,

```
| barSpring bar ground marker0101 marker0102 marker0001 marker0002 spring joint |
```

Next, an instance of System is created by sending the message 'new' to System, and it is assigned a name. The name is a string within two single quotes. This instance now represents the whole dynamic system in the input. For example,

```
barSpring := System new.  
barSpring initializeSystem: 'Bar-Spring in 3-D'.
```

Simulation duration, step size, and order are usually provided after system initialization. For example,

```
barSpring setInitialTime: 0 endTime: 5 stepSize: 0.05 maxIntegrationOrder: 3.
```

Then, all system components such as ground, parts, joints, springs, dampers, user forces, or markers are defined and initialized. A ground is created and initialized as a special instance of Part. It does not have any inertial property and dynamics variable; it just serves as an inertia frame. For example,

```
ground := Part new.  
ground initializeGround.
```

A part is created and initialized with its number, mass, moment-of-inertia, initial velocity, center-of-mass marker, and initial position. The moment-of-inertia is expressed in the three principal values. The initial velocity of the part is provided in an array. The six elements of the velocity array are three translational components expressed in the ground frame and three rotational components expressed in the part frame. The initial position of the part is specified in terms of the location and orientation of the center-of-mass marker frame. These values are provided by an array also. The six elements of the position array are three Cartesian coordinates and three Euler angles in degrees measured from the ground frame. For example,

```
bar := Part new.  
bar initializePartNo: '01'  
  mass: 1  
  momentOfInertia: # (0.01 8.3333333 8.3333333)  
  velocity: # (0 0 0 0 0 0)  
  centerOfMassMarkerNo: '0100'  
  centerOfMassMarkerAt: # (0 0 83 90 90 -90).
```

Markers are used to identify points for the attachments of joints, springs, dampers or application of forces. A marker is created and initialized with its number, associated part, and its position on the part. The position array contains three Cartesian coordinates and three Euler angles in degrees measured from the part center-of-mass marker. For example,

```
marker0102 := Marker new.  
marker0102 initializeMarkerOn: bar markerNo: '0102' position: # (0 0 0 90 -90 0).
```

A joint is created and initialized with its attachment markers and type. The joint types available are 'spherical', 'revolute', 'translational', 'universal', 'cylindrical', and 'planar'.

```
joint := Joint new.  
joint initializeJointBetween: marker0102 and: marker0002 type: 'translational'.
```

A spring is created and initialized with its attachment markers, stiffness, and free length. For example,

```
spring := Spring new.  
spring initializeSpringBetween: marker0101 and: marker0001 springConstant: 9.8  
freeLength: 10.
```

A damper is created and initialized with its attachment markers and damping coefficients. For example,

```
damper := Damper new.  
damper initializeDamperBetween: marker0101 and: marker0001 dampingCoeff: 1.
```

A user-defined force is created and initialized with its force type, magnitude, force pattern, and application points. For example,

```
force := UserForce new.  
force initializeUserForce: 'torque' pattern: 'action-only' magnitude: 10 between: marker0101  
and: marker0001.
```

A request is created and initialized with data type, part instance, and external file name. For example,

```
(Request new) request: 'position' of: bar onto: 'qFile'.
```

When the system is completely defined, the simulation can be started by sending the message 'solveSystem' to the instance of System. For example,

```
barSpring solveSystem.
```

6.3 Program Test Runs

Several one- and two-body systems are simulated to test OODS. The test models are either constrained or unconstrained. In the constrained models, three types of joints are considered so that the At-Point, In-Plane, and Perpendicular constraints are included. In the unconstrained models, the constraint equations are not involved, and therefore the size of the resulting system equations is smaller than the constrained case.

6.3.1 One-Body Systems

These test runs check if the expressions for dynamics equations and numerical algorithms for each part are correct. Several types of one-body systems are chosen as test models.

6.3.1.1 Bar Pendulum in Plane Motion

The initial configuration of a bar pendulum model is shown in Figure 6.4. The bar remains in YZ-plane during the oscillation. Two types of joints between the bar and ground are used in two separate simulations. They are a spherical joint and a revolute joint. The input data are shown in Figure 6.5.

During this simulation, a search for the optimal integration order and step-size is done first. The study shows that the Gear's first- and second-order formulas give significantly damped solutions even though the system is actually nondissipative. This situation is shown in Figure 6.6. The first and second orders damp the Y response amplitude by 45% and 2% per cycle, respectively. The third or higher orders prove to be satisfactory. Note that reducing the step size does not eliminate the damping in the solutions appreciably. Thus, it is not useful to control step size without having a proper integration order. Figure 6.7 shows that the Ψ response becomes overshoot and unstable as the step size increases at the third integration order. The one-body simulations use a third order Gear formula and a step size of 0.1.

Figures 6.8-6.9 show that the period and amplitudes are in agreement with the numerically exact solution. The numerically exact solution is obtained from solving the same problem with one degree of freedom using the modified Euler method. The equation of motion is

$$\ddot{\Psi} + \frac{3g}{2l} \cos \Psi = 0 \quad (6.1)$$

where Ψ is the oscillating angle, g is gravity, and l is the length of the bar (Figure 6.4). The period of Ψ is 5.55 seconds. The position values are also compatible because they satisfy the constraint equations. In addition, the velocities (Figures 6.10-6.11) are checked with the numerically exact solution.

The algorithm of OODS for this problem is reproduced in a spreadsheet for an independent check. The Smalltalk output and the spreadsheet output are compared exactly up to at least seven significant digits throughout the simulation.

6.3.1.2 Bar Pendulum in Conical Motion

The initial configuration of another bar pendulum model is shown in Figure 6.12. The bar swings around the Z-axis with the constant angular velocity while the angle between the Z-axis and bar remains constant. The joint between the bar and ground is a spherical joint. The input data are shown in Figure 6.13.

The plots of positions (Figure 6.14) show that the period and amplitudes are in agreement with the exact solution. The exact solution is

$$X = \frac{l}{2} \cos \Psi \sin \left(\sqrt{\frac{mgl}{2(I_{yy} - I_{xx}) \sin \Psi}} t \right) \quad (6.2)$$

$$Y = -\frac{l}{2} \cos \Psi \cos \left(\sqrt{\frac{mgl}{2(I_{yy} - I_{xx}) \sin \Psi}} t \right) \quad (6.3)$$

where m is mass, t is time, and I 's are moment-of-inertias in x and y axes of the bar. The exact period of the positions is 4.36 seconds. The position values also satisfy the constraint equations. In addition, the velocities (Figure 6.15) and constraint forces (Figure 6.16) are compared with the exact values.

6.3.1.3 Bar-Spring System

The initial configuration of a bar-spring system is shown in Figure 6.17. The bar oscillates parallel to the Z-axis. The spring, however, is not parallel to the Z-axis. The joint between the bar and ground is a translational joint. The input data are shown in Figure 6.18.

Figure 6.19 shows the motion of the bar. The period and amplitude look reasonable and are checked with the linear solution where the spring is parallel to the Z-axis. The linear period of oscillation is 2.01 seconds. Figure 6.20 shows the velocity and acceleration. Figure 6.21 shows the constraint force and moment. They all look reasonable when compared with the linear case.

6.3.1.4 Bar-Spring-Damper System

The initial configuration of a bar-spring-damper system is shown in Figure 6.22. The bar is in a damped oscillatory motion parallel to the Z-axis. Two different damping ratios are used at two separate simulations. The input data for the case $\zeta = 0.16$ is shown in Figure 6.23.

Figure 6.24 shows the motions of the bar at two different damped cases. The damping ratios are $\zeta = 0.48$ and $\zeta = 0.16$. The damped periods agree with the exact solution. The periods of both cases are 2.29 and 2.03 seconds, respectively. The damped amplitudes are also checked with the exact logarithmic decays of any two successive amplitudes at each case. The logarithmic decays are 3.44 and 1.02, respectively. Figure 6.25 shows the velocities and accelerations. They are also in agreement with the exact solution.

6.3.1.5 Unconstrained Body

The initial configuration of an unconstrained body is shown in Figure 6.26. The body is free to travel in a uniform gravitational field. The body has initial velocity of 14.14 m/s at an initial angle of 45° above the horizontal. The input data are shown in Figure 6.27.

Figures 6.28-6.29 show that the trajectories are in agreement with the exact solution. The flight time of the body above the ground is 2.04 seconds, the horizontal range is 20.4 m, and the maximum height is 5.10 m, respectively. The velocity and acceleration also agree with the exact solution (Figure 6.30).

6.3.2 Two-Body Systems

These tests check if OODS assembles the whole system equations of two or more parts properly. Several two-body systems with known analytical solutions are chosen. The spherical and revolute joints are tested here too. The integration order of 3 and step size of 0.05 seconds are used in these simulations.

6.3.2.1 Closed Two-Bar System

The initial configuration of a system which forms a closed loop consisting of two bars and the ground is shown in Figure 6.31. The two bars oscillate together in and out of the YZ-plane. The joints between each bar and the ground are spherical joints, and the joint connecting the bars is a revolute joint. The input data are shown in Figure 6.32.

Figures 6.33-6.35 show that the period and amplitudes look reasonable. All the constraints are satisfied throughout the oscillations. The velocities (Figures 6.36-6.37), accelerations (Figure 6.38), and constraint forces (Figures 6.39-6.40) look reasonable too.

6.3.2.2 Double Pendulum

A double pendulum having two mode shapes is simulated. The initial conditions are given such that the pendulums have small oscillations in one of the two possible mode shapes. The two joints imposed are revolute joints.

The initial configuration of the pendulum for the first mode is shown in Figure 6.41. The input data are shown in Figure 6.42. Figure 6.43 shows that the period and amplitudes satisfy the analytical solution. The amplitude ratio of the Ψ 's is 0.699, and the period is

7.42 seconds. All the constraints are satisfied throughout the oscillations. In addition, the velocities (Figure 6.44) and accelerations (Figure 6.45) are checked with the exact values.

The initial configuration of the pendulum for the second mode is shown in Figure 6.46. The input data are shown in Figure 6.47. Figures 6.48-6.49 show that the period and amplitudes satisfy the analytical solution. The amplitude ratio of the responses Ψ 's is -0.477 and the period is 2.77 seconds. The positions also satisfy the constraint equations. In addition, the velocities (Figure 6.50) and accelerations (Figure 6.51) are checked with the exact values.

6.3.2.3 Unconstrained Two-Body System

The initial configuration of two unconstrained bodies is shown in Figure 6.52. The two bodies are free to travel independently in a uniform gravitational field. They have the same initial speed of 11.55 m/s, but different initial angles of 30° and 60° above the horizontal. The input data are shown in Figure 6.53.

Figures 6.54-6.55 show that the trajectories agree with the exact solution. The flight time above the ground for each body is 1.18 and 2.04 seconds. The maximum height for each body is 1.70 and 5.07 m. The horizontal range is 11.79 m, the same for both bodies.

6.4 Comments on Program Test Results

Although not shown, many other simulations have been done on the foregoing one- and two-body systems with different initial conditions. The results have been checked with known results whenever possible.

The test runs reveal that OODS can formulate the system equations and solve them correctly. OODS is applicable to simulations of three or more body systems because the simulation algorithms and programming of OODS have been done in general terms.

All the above simulations have been done on a Macintosh SE/30 with 8 MB of RAM using Smalltalk-80 Version 2.5 from ParcPlace Systems. The computation time for each

one-body simulation was 20 to 25 minutes per 100 time steps with 5 iterations at each time step. For the two-body simulations, computation time was 90 to 100 minutes for the same number of time steps and iterations. These times are long because the trigonometric functions are evaluated every time they occur in different methods of OODS. Great improvement can be made if the trigonometric evaluations are done once and the values are reused by many methods. The research has concentrated on getting OODS to do the correct formulation and numerical solution. More effort will be made to increase computational efficiency as continuing research.

Figure 6.1: Class Hierarchy within OODS

Figure 6.2: Problem Solving in OODS

Figure 6.3: Message Sendings Diagram

Figure 6.4: Bar Pendulum in Plane Motion

"This provides the input for the simulation of a bar pendulum in plane motion."

```
l barPendulum bar ground marker0101 marker0001 joint0100 l
```

"----- INPUT STATEMENTS -----"

```
barPendulum := System new.
```

```
barPendulum initializeSystem: 'Bar Pendulum in Plane Motion'.
```

```
barPendulum setInitialTime: 0 endTime: 0.5 stepSize: 0.05 maxIntegrationOrder: 3.
```

```
ground := Part new.
```

```
ground initializeGround.
```

```
bar := Part new.
```

```
bar initializePartNo: '01' mass: 1.0 momentOfInertia: #(0.01 8.33333 8.33333)
```

```
velocity: #(0.0 0.0 0.0 0.0 0.0 0.0)
```

```
centerOfMassMarkerNo: '0100' centerOfMassMarkerAt: #(0.0 4.33013 97.5 90 90 -30).
```

```
marker0101 := Marker new.
```

```
marker0101 initializeMarkerOn: bar markerNo: '0101' position: #(-5.0 0.0 0.0 0.0 0.0 0.0).
```

```
marker0001 := Marker new.
```

```
marker0001 initializeMarkerOn: ground markerNo: '0001' position: #(0.0 0.0 100.0 90.0 90.0 0.0).
```

```
joint0100 := Joint new.
```

```
joint0100 initializeJointBetween: marker0101 and: marker0001 type: 'spherical'.
```

"----- OUTPUT STATEMENTS -----"

```
(Request new) request: 'position' of: bar onto: 'qFile'.
```

```
(Request new) request: 'velocity' of: bar onto: 'qdotFile'.
```

```
(Request new) request: 'acceleration' of: bar onto: 'qdotdotFile'.
```

"----- EXECUTION -----"

```
^ barPendulum solveSystem
```

Figure 6.5: Input Data of Bar Pendulum in Plane Motion

Figure 6.6: Effect of Integration Orders on Bar Pendulum

Figure 6.7: Effect of Step Sizes on Bar Pendulum

Figure 6.8: Y, Z Responses of Bar Pendulum in Plane Motion

Figure 6.9: Φ Response of Bar Pendulum in Plane Motion

Figure 6.10: $\dot{Y}$, $\dot{Z}$ Responses of Bar Pendulum in Plane Motion

Figure 6.11: $\dot{\Psi}$ Response of Bar Pendulum in Plane Motion

Figure 6.12: Bar pendulum in Conical Motion

"This provides the input for the simulation of a bar pendulum swinging around the Z axis of the ground frame."

```
l barPendulum bar ground marker0101 marker0001 joint0100 l
```

"----- INPUT STATEMENTS -----"

```
barPendulum := System new.
```

```
barPendulum initializeSystem: 'Bar Pendulum in Conical Motion'.
```

```
barPendulum setInitialTime: 0 endTime: 0.2 stepSize: 0.05 maxIntegrationOrder: 3.
```

```
ground := Part new.
```

```
ground initializeGround.
```

```
bar := Part new.
```

```
bar initializePartNo: '01' mass: 1 momentOfInertia: #(0.01 8.3333333 8.3333333)
```

```
velocity: #(-5.0984283 0 0 -1.0196857 1.0196857 0)
```

```
centerOfMassMarkerNo: '0100' centerOfMassMarkerAt: #(0 3.5355339 96.4644661 90 90 -45).
```

```
marker0101 := Marker new.
```

```
marker0101 initializeMarkerOn: bar markerNo: '0101' position: #(-5 0 0 0 0 0).
```

```
marker0001 := Marker new.
```

```
marker0001 initializeMarkerOn: ground markerNo: '0001' position: #(0 0 100 90 90 0).
```

```
joint0100 := Joint new.
```

```
joint0100 initializeJointBetween: marker0101 and: marker0001 type: 'spherical'.
```

"----- OUTPUT STATEMENTS -----"

```
(Request new) request: 'position' of: bar onto: 'qFile'.
```

```
(Request new) request: 'velocity' of: bar onto: 'qdotFile'.
```

```
(Request new) request: 'acceleration' of: bar onto: 'qdotdotFile'.
```

```
(Request new) request: 'momentum' of: bar onto: 'uFile'.
```

```
(Request new) request: 'momentumDot' of: bar onto: 'udotFile'.
```

```
(Request new) request: 'lambda' of: bar onto: 'LambdaFile'.
```

"----- EXECUTION -----"

```
barPendulum solveSystemSmooth
```

Figure 6.13: Input Data of Bar Pendulum in Conical Motion

Figure 6.14: X, Y Responses of Bar Pendulum in Conical Motion

Figure 6.15: $\dot{X}, \dot{Y}$ Responses of Bar Pendulum in Conical Motion

Figure 6.16: Constraint Forces on Bar Pendulum in Conical Motion

Figure 6.17: Bar-Spring System

"This provides the input for the simulation of a bar-spring whose one end is attached to the ground and oscillates up and down along Z-axis. The spring is out of Z-axis by 2 m."

```
l barSpring ground bar marker0101 marker0102 marker0001 marker0002 spring joint l
```

"----- INPUT STATEMENTS -----"

```
barSpring := System new.
```

```
barSpring initializeSystem: 'Bar-Spring Out of Z-axis'.
```

```
barSpring setInitialTime: 0 endTime: 5 stepSize: 0.05 maxIntegrationOrder: 3.
```

```
ground := Part new.
```

```
ground initializeGround.
```

```
bar := Part new.
```

```
bar initializePartNo: '01' mass: 1 momentOfInertia: #(0.01 8.3333333 8.3333333)
```

```
velocity: #(0 0 0 0 0 0)
```

```
centerOfMassMarkerNo: '0100' centerOfMassMarkerAt: #(0 0 83 90 90 -90).
```

```
marker0101 := Marker new.
```

```
marker0101 initializeMarkerOn: bar markerNo: '0101' position: #(-5 0 0 0 0 0).
```

```
marker0102 := Marker new.
```

```
marker0102 initializeMarkerOn: bar markerNo: '0102' position: #(0 0 0 90 -90 0).
```

```
marker0001 := Marker new.
```

```
marker0001 initializeMarkerOn: ground markerNo: '0001' position: #(0 -2 100 90 90 0).
```

```
marker0002 := Marker new.
```

```
marker0002 initializeMarkerOn: ground markerNo: '0002' position: #(0 0 83 90 0 0).
```

```
spring := Spring new.
```

```
spring initializeSpringBetween: marker0101 and: marker0001 springConstant: 9.8 freeLength: 10.
```

```
joint := Joint new.
```

```
joint initializeJointBetween: marker0102 and: marker0002 type: 'translational'.
```

"----- OUTPUT STATEMENTS -----"

```
(Request new) request: 'position' of: bar onto: 'qFile'.
```

```
(Request new) request: 'velocity' of: bar onto: 'qdotFile'.
```

```
(Request new) request: 'acceleration' of: bar onto: 'qdotdotFile'.
```

```
(Request new) request: 'constraintForce' of: bar onto: 'constraintForceFile'.
```

"----- EXECUTION -----"

```
barSpring solveSystem
```

Figure 6.18: Input Data of Bar-Spring System

Figure 6.19: Z , Φ , Θ , Ψ Responses of Bar-Spring System

Figure 6.20: $\dot{Z}$, $\ddot{Z}$ Responses of Bar-Spring System

Figure 6.21: Constraint Forces of Bar-Spring System

Figure 6.22: Bar-Spring-Damper System

"This provides the input for the simulation of a bar-spring-damper in a damped oscillatory motion parallel to the Z-axis. The damping ratio is 0.16."

```
l barSpring ground bar marker0101 marker0102 marker0001 marker0002 spring damper joint l
```

```
"----- INPUT STATEMENTS -----"
```

```
barSpring := System new.
```

```
barSpring initializeSystem: 'Bar-Spring-Damper System'.
```

```
barSpring setInitialTime: 0 endTime: 5 stepSize: 0.05 maxIntegrationOrder: 3.
```

```
ground := Part new.
```

```
ground initializeGround.
```

```
bar := Part new.
```

```
bar initializePartNo: '01' mass: 1 momentOfInertia: #(0.01 8.3333333 8.3333333)
```

```
velocity: #(0 0 0 0 0 0)
```

```
centerOfMassMarkerNo: '0100' centerOfMassMarkerAt: #(0 0 83 90 90 -90).
```

```
marker0101 := Marker new.
```

```
marker0101 initializeMarkerOn: bar markerNo: '0101' position: #(-5 0 0 0 0 0).
```

```
marker0102 := Marker new.
```

```
marker0102 initializeMarkerOn: bar markerNo: '0102' position: #(0 0 0 90 -90 0).
```

```
marker0001 := Marker new.
```

```
marker0001 initializeMarkerOn: ground markerNo: '0001' position: #(0 0 100 90 90 0).
```

```
marker0002 := Marker new.
```

```
marker0002 initializeMarkerOn: ground markerNo: '0002' position: #(0 0 83 90 0 0).
```

```
spring := Spring new.
```

```
spring initializeSpringBetween: marker0101 and: marker0001 springConstant: 9.8 freeLength: 10.
```

```
damper := Damper new.
```

```
damper initializeDamperBetween: marker0101 and: marker0001 dampingCoeff: 1.
```

```
joint := Joint new.
```

```
joint initializeJointBetween: marker0102 and: marker0002 type: 'translational'.
```

```
"----- OUTPUT STATEMENTS -----"
```

```
(Request new) request: 'position' of: bar onto: 'qFile'.
```

```
(Request new) request: 'velocity' of: bar onto: 'qdotFile'.
```

```
(Request new) request: 'acceleration' of: bar onto: 'qdotdotFile'.
```

```
(Request new) request: 'constraintForce' of: bar onto: 'constraintForceFile'.
```

```
"----- EXECUTION -----"
```

```
barSpring solveSystem
```

Figure 6.23: Input Data of Bar-Spring-Damper System

Figure 6.24: Z response of Bar-Spring-Damper System

Figure 6.25: $\dot{Z}$, $\ddot{Z}$ responses of Bar-Spring-Damper System

Bar
 Center-Of-Mass Marker : (0 0 0 90 90 0)
 Mass : 1 kg
 Moment-Of-Inertia : (0.01 8.33 8.33)
 Initial Velocity : (0 10 10 0 0 0)

Figure 6.26: Unconstrained Body

"This provides the input for the simulation of an unconstrained body."

| system bar ground |

"----- INPUT STATEMENTS -----"

system := System new.

system initializeSystem: 'Unconstrained Body'.

system setInitialTime: 0 endTime: 0.3 stepSize: 0.05 maxIntegrationOrder: 3.

ground := Part new.

ground initializeGround.

bar := Part new.

bar initializePartNo: '01' mass: 1 momentOfInertia: #(0.01 8.3333333 8.3333333)

velocity: #(0 0 0 0 0)

centerOfMassMarkerNo: '0100' centerOfMassMarkerAt: #(0 0 100 90 90 -90).

"----- OUTPUT STATEMENTS -----"

(Request new) request: 'position' of: bar onto: 'qFile'.

(Request new) request: 'velocity' of: bar onto: 'qdotFile'.

(Request new) request: 'acceleration' of: bar onto: 'qdotdotFile'.

(Request new) request: 'momentum' of: bar onto: 'uFile'.

(Request new) request: 'momentumDot' of: bar onto: 'udotFile'.

"----- EXECUTION -----"

system solveSystem

Figure 6.27: Input Data of Unconstrained Body

Figure 6.28: Y, Z Responses of Unconstrained Body

Figure 6.29: Z/Y Response of Unconstrained Body

Figure 6.30: $\dot{Z}$, $\ddot{Z}$ Responses of Unconstrained Body

Spherical Joint

Marker I on Bar 1 : (-5 0 0 0 0 0)

Marker J on Ground : (0 -5 100 90 90 0)

Spherical Joint

Marker I on Bar 2 : (-5 0 0 0 0 0)

Marker J on Ground : (0 5 100 90 90 0)

Bar 1

Center-Of-Mass Marker :

(3.75 -2.5 97.83 90 30 -60)

Mass : 1 kg

Moment-Of-Inertia :

(0.01 8.33 8.33)

Initial Velocity : (0 0 0 0 0 0)

Bar 2

Center-Of-Mass Marker :

(3.75 2.5 97.83 -90 -30 60)

Mass : 1 kg

Moment-Of-Inertia :

(0.01 8.33 8.33)

Initial Velocity : (0 0 0 0 0 0)

Revolute Joint

Marker I on Bar 1 : (5 0 0 0 0 0)

Marker J on Bar 2 : (5 0 0 0 0 0)

Figure 6.31: Closed Two-Bar System

"This provides the input for the simulation of a closed two-bar system."

```
l barPendulum bar1 bar2 ground marker0101 marker0102 marker0201 marker0202 marker0001 marker0002
joint0100 joint0200 joint0102 l
```

"----- INPUT STATEMENTS -----"

```
barPendulum := System new.
barPendulum initializeSystem: 'Closed Two-Bar System'.
barPendulum setInitialTime: 0 endTime: 0.12 stepSize: 0.05 maxIntegrationOrder: 3.
```

```
ground := Part new. ground initializeGround.
```

```
bar1 := Part new.
bar1 initializePartNo: '01' mass: 1.0 momentOfInertia: #(0.01 8.33333 8.33333)
    velocity: #(0.0 0.0 0.0 0.0 0.0 0.0)
    centerOfMassMarkerNo: '0100' centerOfMassMarkerAt: #(3.749999 -2.5 97.834936 90 30 -60).
marker0101 := Marker new.
marker0101 initializeMarkerOn: bar1 markerNo: '0101' position: #(-5.0 0.0 0.0 0.0 0.0 0.0).
marker0102 := Marker new.
marker0102 initializeMarkerOn: bar1 markerNo: '0102' position: #(5.0 0.0 0.0 0.0 0.0 0.0).
```

```
bar2 := Part new.
bar2 initializePartNo: '02' mass: 1.0 momentOfInertia: #(0.01 8.33333 8.33333)
    velocity: #(0.0 0.0 0.0 0.0 0.0 0.0)
    centerOfMassMarkerNo: '0200' centerOfMassMarkerAt: #(3.749999 2.5 97.834936 -90 -30 60).
marker0201 := Marker new.
marker0201 initializeMarkerOn: bar2 markerNo: '0201' position: #(-5.0 0.0 0.0 0.0 0.0 0.0).
marker0202 := Marker new.
marker0202 initializeMarkerOn: bar2 markerNo: '0202' position: #(5.0 0.0 0.0 0.0 0.0 0.0).
marker0001 := Marker new.
marker0001 initializeMarkerOn: ground markerNo: '0001' position: #(0 -5 100 90 90 0).
marker0002 := Marker new.
marker0002 initializeMarkerOn: ground markerNo: '0002' position: #(0 5 100 90 90 0).
```

```
joint0100 := Joint new. joint0100 initializeJointBetween: marker0101 and: marker0001 type: 'spherical'.
joint0200 := Joint new. joint0200 initializeJointBetween: marker0201 and: marker0002 type: 'spherical'.
joint0102 := Joint new. joint0102 initializeJointBetween: marker0102 and: marker0202 type: 'revolute'.
```

"----- OUTPUT STATEMENTS -----"

```
(Request new) request: 'position' of: bar1 onto: 'qBar1'.
(Request new) request: 'position' of: bar2 onto: 'qBar2'.
(Request new) request: 'velocity' of: bar1 onto: 'qdotBar1'.
(Request new) request: 'velocity' of: bar2 onto: 'qdotBar2'.
(Request new) request: 'acceleration' of: bar1 onto: 'qdotdotBar1'.
(Request new) request: 'acccleration' of: bar2 onto: 'qdotdotBar2'.
```

"----- EXECUTION -----"

```
barPendulum solveSystemSmooth
```

Figure 6.32: Input Data of Closed Two-Bar System

Figure 6.33: X, Y Responses of Closed Two-Bar System

Figure 6.34: Z , Φ Responses of Closed Two-Bar System

Figure 6.35: Θ, Ψ Responses of Closed Two-Bar System

Figure 6.36: $\dot{X}$, $\dot{Z}$ Responses of Closed Two-Bar System

Figure 6.37: $\dot{\Theta}$ Response of Closed Two-Bar System

Figure 6.38: $\ddot{X}$, $\ddot{Z}$ Responses of Closed Two-Bar System

Figure 6.39: Constraint Forces of Closed Two-Bar System

Figure 6.40: Constraint Force Acting on Bar 1 of Closed Two-Bar System

Figure 6.41: Double Pendulum (Mode 1)

"This provides the input for the simulation of a double pendulum (Mode 1)."

```
l barPendulum bar1 bar2 ground marker0101 marker0102 marker0201 marker0001 joint0100 joint0102 l
```

"----- INPUT STATEMENTS -----"

```
barPendulum := System new.
```

```
barPendulum initializeSystem: 'Double Pendulum (Mode 1)'.
```

```
barPendulum setInitialTime: 0 endTime: 10 stepSize: 0.05 maxIntegrationOrder: 3.
```

```
ground := Part new.
```

```
ground initializeGround.
```

```
bar1 := Part new.
```

```
bar1 initializePartNo: '01' mass: 1.0 momentOfInertia: #(0.01 8.33333 8.33333)
```

```
velocity: #(0.0 0.0 0.0 0.0 0.0 0.0)
```

```
centerOfMassMarkerNo: '0100' centerOfMassMarkerAt: #(0 0.60848055 95.037163 90 90 -83.01).
```

```
marker0101 := Marker new.
```

```
marker0101 initializeMarkerOn: bar1 markerNo: '0101' position: #(-5.0 0.0 0.0 0.0 0.0 0.0).
```

```
marker0102 := Marker new.
```

```
marker0102 initializeMarkerOn: bar1 markerNo: '0102' position: #(5.0 0.0 0.0 0.0 0.0 0.0).
```

```
bar2 := Part new.
```

```
bar2 initializePartNo: '02' mass: 1.0 momentOfInertia: #(0.01 8.33333 8.33333)
```

```
velocity: #(0.0 0.0 0.0 0.0 0.0 0.0)
```

```
centerOfMassMarkerNo: '0200' centerOfMassMarkerAt: #(0 2.085202 85.1502872 90 90 -80).
```

```
marker0201 := Marker new.
```

```
marker0201 initializeMarkerOn: bar2 markerNo: '0201' position: #(-5.0 0.0 0.0 0.0 0.0 0.0).
```

```
marker0001 := Marker new.
```

```
marker0001 initializeMarkerOn: ground markerNo: '0001' position: #(0 0 100 90 90 0).
```

```
joint0100 := Joint new.
```

```
joint0100 initializeJointBetween: marker0101 and: marker0001 type: 'revolute'.
```

```
joint0102 := Joint new.
```

```
joint0102 initializeJointBetween: marker0102 and: marker0201 type: 'revolute'.
```

"----- OUTPUTSTATEMENTS -----"

```
(Request new) request: 'position' of: bar1 onto: 'qBar1'.
```

```
(Request new) request: 'position' of: bar2 onto: 'qBar2'.
```

```
(Request new) request: 'velocity' of: bar1 onto: 'qdotBar1'.
```

```
(Request new) request: 'velocity' of: bar2 onto: 'qdotBar2'.
```

```
(Request new) request: 'acceleration' of: bar1 onto: 'qdotdotBar1'.
```

```
(Request new) request: 'acceleration' of: bar2 onto: 'qdotdotBar2'.
```

```
(Request new) request: 'lambda' onto: 'lambdaBar'.
```

"----- EXECUTION -----"

```
barPendulum solveSystemSmooth
```

Figure 6.42: Input Data of Double Pendulum (Mode 1)

Figure 6.43: Y , Ψ Responses of Double Pendulum (Mode 1)

Figure 6.44: $\dot{Y}, \dot{\Psi}$ Responses of Double Pendulum (Mode 1)

Figure 6.45: $\ddot{Y}$ Response of Double Pendulum (Mode 1)

Figure 6.46: Double Pendulum (Mode 2)

"This provides the input for the simulation of a double pendulum (mode 2)."

```
l barPendulum bar1 bar2 ground marker0101 marker0102 marker0201 marker0001
joint0100 joint0102 l
```

"----- INPUT STATEMENTS -----"

```
barPendulum := System new.
```

```
barPendulum initializeSystem: 'Double Pendulum (Mode 2)'.
```

```
barPendulum setInitialTime: 0 endTime: 5 stepSize: 0.05 maxIntegrationOrder: 3.
```

```
ground := Part new.
```

```
ground initializeGround.
```

```
bar1 := Part new.
```

```
bar1 initializePartNo: '01' mass: 1.0 momentOfInertia: #(0.01 8.33333 8.33333)
```

```
velocity: #(0.0 0.0 0.0 0.0 0.0 0.0)
```

```
centerOfMassMarkerNo: '0100' centerOfMassMarkerAt: #(0 0.415780345 95.0173173 90 90 -85.23).
```

```
marker0101 := Marker new.
```

```
marker0101 initializeMarkerOn: bar1 markerNo: '0101' position: #(-5.0 0.0 0.0 0.0 0.0 0.0).
```

```
marker0102 := Marker new.
```

```
marker0102 initializeMarkerOn: bar1 markerNo: '0102' position: #(5.0 0.0 0.0 0.0 0.0 0.0).
```

```
bar2 := Part new.
```

```
bar2 initializePartNo: '02' mass: 1.0 momentOfInertia: #(0.01 8.33333 8.33333)
```

```
velocity: #(0.0 0.0 0.0 0.0 0.0 0.0)
```

```
centerOfMassMarkerNo: '0200' centerOfMassMarkerAt: #(0 -0.03668021 85.1105959 -90 -90 80).
```

```
marker0201 := Marker new.
```

```
marker0201 initializeMarkerOn: bar2 markerNo: '0201' position: #(-5.0 0.0 0.0 0.0 0.0 0.0).
```

```
marker0001 := Marker new.
```

```
marker0001 initializeMarkerOn: ground markerNo: '0001' position: #(0 0 100 90 90 0).
```

```
joint0100 := Joint new.
```

```
joint0100 initializeJointBetween: marker0101 and: marker0001 type: 'revolute'.
```

```
joint0102 := Joint new.
```

```
joint0102 initializeJointBetween: marker0102 and: marker0201 type: 'revolute'.
```

"----- OUTPUT STATEMENTS -----"

```
(Request new) request: 'position' of: bar1 onto: 'qBar1'.
```

```
(Request new) request: 'position' of: bar2 onto: 'qBar2'.
```

```
(Request new) request: 'velocity' of: bar1 onto: 'qdotBar1'.
```

```
(Request new) request: 'velocity' of: bar2 onto: 'qdotBar2'.
```

```
(Request new) request: 'acceleration' of: bar1 onto: 'qdotdotBar1'.
```

```
(Request new) request: 'acceleration' of: bar2 onto: 'qdotdotBar2'.
```

```
(Request new) request: 'lambda' onto: 'lambdaBar'.
```

"----- EXECUTION -----"

```
barPendulum solveSystemSmooth
```

Figure 6.47: Input Data of Double Pendulum (Mode 2)

Figure 6.48: Y Response of Double Pendulum (Mode 2)

Figure 6.49: Ψ Response of Double Pendulum (Mode 2)

Figure 6.50: $\dot{Y}, \dot{\Psi}$ Responses of Double Pendulum (Mode 2)

Figure 6.51: $\ddot{Y}, \ddot{\Psi}$ Responses of Double Pendulum (Mode 2)

Bar 1

Center-Of-Mass Marker :

 $(0 \ 0 \ 0 \ 90 \ 90 \ 0)$

Mass : 1 kg

Moment-Of-Inertia : $(0.01 \ 8.33 \ 8.33)$ Initial Velocity : $(0 \ 5.77 \ 10 \ 0 \ 0 \ 0)$ Bar 2

Center-Of-Mass Marker :

 $(0 \ 0 \ 0 \ 90 \ 90 \ 0)$

Mass : 1 kg

Moment-Of-Inertia : $(0.01 \ 8.33 \ 8.33)$ Initial Velocity : $(0 \ 10 \ 5.77 \ 0 \ 0 \ 0)$

Figure 6.52: Unconstrained Two-Body System

```

! system bullet1 bullet2 ground !

"----- INPUT STATEMENTS -----"
system := System new.
system initializeSystem: 'Unconstrained Two-Body System'.
system setInitialTime: 0 endTime: 2.1 stepSize: 0.05 maxIntegrationOrder: 3.

ground := Part new.
ground initializeGround.

bullet1 := Part new.
bullet1 initializePartNo: '01' mass: 1 momentOfInertia: #(0.01 8.33333333 8.33333333)
    velocity: #(0 5.7735 10 0 0 0)
    centerOfMassMarkerNo: '0100' centerOfMassMarkerAt: #(0 0 0 90 90 0).
bullet2 := Part new.
bullet2 initializePartNo: '02' mass: 1 momentOfInertia: #(0.01 8.33333333 8.33333333)
    velocity: #(0 10 5.7735 0 0 0)
    centerOfMassMarkerNo: '0200' centerOfMassMarkerAt: #(0 0 0 90 90 0).

"----- OUTPUT STATEMENTS -----"
(Request new) request: 'position' of: bullet1 onto: 'qFile1'.
(Request new) request: 'velocity' of: bullet1 onto: 'qdotFile1'.
(Request new) request: 'acceleration' of: bullet1 onto: 'qdotdotFile1'.
(Request new) request: 'position' of: bullet2 onto: 'qFile2'.

"This provides the input for the simulation of an unconstrained two-body system."

(Request new) request: 'velocity' of: bullet2 onto: 'qdotFile2'.
(Request new) request: 'acceleration' of: bullet2 onto: 'qdotdotFile2'.

"----- EXECUTION -----"
system solveSystem

```

Figure 6.53: Input Data of Unconstrained Two-Body System

Figure 6.54: Y, Z Responses of Unconstrained Two-Body System

Figure 6.55: Z/Y Response of Unconstrained Two-Body System

CHAPTER VII

CONCLUSIONS AND RECOMMENDATIONS

OODS has resulted from incorporating dynamics simulation theory and numerical analysis into object-oriented programming. A theory for deriving and formulating the governing equations of a mechanical system has been developed in very general terms. Then, numerical solution schemes to solve these equations have been presented. Finally, these operations have been automated in Smalltalk-80 which takes full advantage of object-oriented programming. OODS has been carefully checked out with several test runs involving one- and two-body systems.

7.1 Conclusions

From this research, the following conclusions can be made:

- (1) To the best of the author's knowledge, OODS is the first dynamics simulator using an OOP concept and language.
- (2) OODS is relatively easy to understand and use. Like all Smalltalk-80 programs, it is embedded in the Smalltalk-80 environment. The entire source code of the environment and OODS is available for user perusal, modifications, or additions.
- (3) OODS is reliable in its formulation of system equations and their numerical solutions. It has simulated many one- and two-body systems correctly.
- (4) OODS currently recognizes rigid parts, six joints, translational springs, translational dampers, and user-defined constant forces and torques. The six joints are spherical, revolute, cylindrical, universal, translational, and planar joints.
- (5) OODS is applicable to larger systems because the simulation algorithms and programming of OODS have been designed in general terms. The systems may

undergo large displacements and have multiple degrees of freedom in three dimensional space. The systems can have closed- or open-loops which may be constrained or free between parts and ground. Larger systems have not been simulated due to the limitation of the long computation time.

- (6) OODS can be used for kinematic or static analysis. For kinematic analysis, only the motions of the systems are considered, and they must satisfy the system constraint equations. For static analysis, the velocity and acceleration terms of the system equations are assumed to be zeros. Thus, static equilibrium positions can be found, if they exist.
- (7) This dissertation contains the explicit expressions for the derivation of the system equations.

7.2 Recommendations

The capabilities of OODS can be enhanced over time, and Smalltalk-80 makes it easier to do that. Along this line, the following features can be added to OODS:

- (1) Algorithms for automatic control of integration step size and order.
- (2) Sparse matrix algorithms for efficient storage and solution of the system equations.
- (3) A postprocessor to produce graphical outputs.
- (4) Finally, specific applications in mechanical engineering practice and education are encouraging.

BIBLIOGRAPHY

- ADAMS User's Manual (Version 5.2), Vol. 1, Mechanical Dynamics, Ann Arbor, MI, 1987.
- Applications and Services Catalog (Objectworks for Smalltalk-80 Version 2.5), ParcPlace Systems, Mountain View, CA, July, 1990.
- Chace, M. A. and Smith, D. A. DAWN - A Digital Computer Program for the Dynamic Analysis of Generalized Mechanical Systems, SAE paper 710244, January, 1971.
- Chua, L. O. and Lin, P. M., Computer-Aided Analysis of Electronic Circuits: Algorithms and Computational Techniques, Prentice-Hall, Englewood Cliffs, NJ, 1975.
- Cox, B. J. Object-Oriented Programming in C, Unix Review, October/November, 1983.
- Cox, B. J. Message/Object Programming: An Evolutionary Change in Programming Technology, IEEE Software, Vol. 1, No. 1, January, 1984.
- Diederich, J. and Milton, J. Experimental Prototyping in Smalltalk, IEEE Software, pp 50-64, May, 1987.
- Gear, C. W. Simultaneous Numerical Solution of Differential-Algebraic Equations, IEEE Transactions on Circuit Theory, Vol. CT-18, No. 1, pp 89-95, January, 1971.
- Gear, C. W. Numerical Initial Value Problems in Ordinary Differential Equations, Prentice-Hall, Englewood Cliffs, NJ, 1971.
- Goldberg, A. Smalltalk-80: The Interactive Programming Environment, Addison-Wesley, 1984.
- Greenwood, D. T. Principles of Dynamics, Prentice-Hall, Englewood Cliffs, 1988.
- Haug, E. J. Computer Aided Analysis and Optimization of Mechanical System Dynamics, NATO ASI Series, Vol. F9, Springer-Verlag, Berlin, 1984.
- Jones, T. C., Reusability in Programming: A Survey of the State of the Art, IEEE Transaction. Software Engineering, pp 488-494, September, 1984.
- Kaehler, T. and Patterson, D. A Taste of Smalltalk, W.W. Norton & Company, New York, NY, 1986.
- Koh, A-S., Park, J-P., and Hsiung, C-M. Simulation Study of Crosswind Sensitivity, TR-FMC-89-6, Ford Motor Co., December, 1989.
- Lambert, J. D. Computational Method in Ordinary Differential Equations, John Wiley & Sons, 1987.

- Leigois, A., Khalil, W., and Dumas, J. M. Mathematical and Computer Models of Interconnected Mechanical Systems, Proceedings of the Symposium on Theory and Practice of Robots and Manipulators, Warsaw, Poland, September, 1976.
- Micallef, J. Encapsulation, Reusability, and Extensibility in Object-Oriented Programming Languages, Journal of Object-Oriented Programming, April/May, 1988.
- Nikravesh, P. E. and Chung, I. S. Application of Euler Parameters to the Dynamic Analysis of Three Dimensional Constrained Mechanical Systems - A General Purpose Computer Program, ASME Journal of Mechanical Design, Vol. 104, pp 785-791, October, 1982.
- Orlande, N., Chace, M. A., and Calahan, D. A. A Sparsity-Oriented Approach to the Dynamic Analysis and Design of Mechanical Systems, Parts I and II, ASME Journal of Engineering for Industry, Ser. B, Vol. 99, pp 773-784, 1977.
- Paul, B. Kinematics and Dynamics of Planar Machinery, Prentice-Hall, Englewood Cliffs, NJ, 1979.
- Pinson, L. J. and Wiener, R. S. An Introduction to Object-Oriented Programming and Smalltalk, Addison-Wesley, 1988.
- Rampalli, R. ADAMS - A Sparse Matrix Approach to Solving Multi-Body Dynamics Problems, SDIO/NASA Workshop on Multi-Body Simulation, Pasadena, CA, 1987.
- Schiehlen, W. Multibody System Handbook, Springer-Verlag, Berlin, 1990.
- Sheth, P. N. and Uicker, J. J., Jr. IMP - A Computer Aided Design Analysis System for Mechanisms and Linkages, ASME Journal of Engineering for Industry, Ser. B, Vol. 94, pp 454-464, 1972.
- Tesler, L. The Smalltalk Environment, Byte, pp 90-145, August, 1981.
- Unda, J., Garcia de Jalon, J., Losantos, F., and Enparantza, R. A Comparative Study on Some Different Formulations of the Dynamic Equations of Constrained Mechanical Systems, ASME Journal of Mechanisms, Transmissions, and Automation in Design, Vol. 109, December, 1987.
- Williams, R. J. and Seireg, A. Interactive Modeling and Analysis of Open or Closed Loop Dynamic Systems with Redundant Actuators, ASME Journal of Machine Design, Vol. 101, pp 407-416, July, 1979.

APPENDIX A

EXPANDED EXPRESSIONS

This appendix provides the expanded expressions of the complicated terms used in the formulation and numerical analysis of system equations.

A.1 Expressions for Formulation of System Equations

A.1.1 Transformation Matrices

The transformation matrix from the part i_p frame to the ground frame is shown as

$$[A]_{i_p} = \begin{bmatrix} c\Phi c\Psi - s\Phi c\Theta s\Psi & -c\Phi s\Psi - s\Phi c\Theta c\Psi & s\Phi s\Theta \\ s\Phi c\Psi + c\Phi c\Theta s\Psi & -s\Phi s\Psi + c\Phi c\Theta c\Psi & -c\Phi s\Theta \\ s\Theta s\Psi & s\Theta c\Psi & c\Theta \end{bmatrix}_{i_p}. \quad (\text{A.1})$$

The trigonometric functions 'sin' and 'cos' are denoted by s and c for simplicity. Note that the individual angles Φ and Ψ are not determinant when the second rotation angle Θ is zero.

The transformation matrix from the generalized speeds to the generalized momenta of part i_p , for rotational components, is

$$[B]_{i_p}^T [J]_{i_p} [B]_{i_p} = \begin{bmatrix} J_x s^2 \Theta s^2 \Psi & J_x s \Theta s \Psi c \Psi & J_x c \Theta \\ +J_y s^2 \Theta c^2 \Psi + J_z c^2 \Theta & -J_y s \Theta s \Psi c \Psi & \\ \hline J_x s \Theta s \Psi c \Psi & J_x c^2 \Psi + J_y s^2 \Psi & 0 \\ -J_y s \Theta s \Psi c \Psi & & \\ \hline J_x c \Theta & 0 & J_z \end{bmatrix}_{i_p}. \quad (\text{A.2})$$

A.1.2 Equations of Motion

The kinetic energy of a system is

$$\begin{aligned}
 T &= \sum_{i,r=1}^{n_r} \frac{1}{2} m_{i,r} (\dot{X}_{i,r}^2 + \dot{Y}_{i,r}^2 + \dot{Z}_{i,r}^2) + \sum_{i,r=1}^{n_r} \frac{1}{2} \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i,r}^T [B]_{i,r}^T [J]_{i,r} [B]_{i,r} \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i,r} \\
 &= \sum_{i,r=1}^{n_r} \frac{1}{2} m_{i,r} (\dot{X}_{i,r}^2 + \dot{Y}_{i,r}^2 + \dot{Z}_{i,r}^2) \\
 &+ \sum_{i,r=1}^{n_r} \frac{1}{2} \left(J_{xx} (\dot{\Phi}_s \Theta_s \Psi + \dot{\Theta}_c \Psi)^2 + J_{yy} (\dot{\Phi}_s \Theta_c \Psi - \dot{\Theta}_s \Psi)^2 + J_{zz} (\dot{\Phi}_c \Theta + \dot{\Psi})^2 \right). \quad (A.3)
 \end{aligned}$$

The derivatives of T with respect to Θ and Ψ have the following expressions;

$$\begin{aligned}
 \frac{\partial T}{\partial \Theta_{i,r}} &= \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i,r}^T \left(\frac{\partial}{\partial \Theta_{i,r}} [B]_{i,r}^T \right) [J]_{i,r} [B]_{i,r} \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i,r} \\
 &= \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i,r}^T \begin{bmatrix} c\Theta_s \Psi & 0 & 0 \\ c\Theta_c \Psi & 0 & 0 \\ -s\Theta & 0 & 0 \end{bmatrix}_{i,r} [J]_{i,r} [B]_{i,r} \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i,r} \\
 &= J_{xx} (\dot{\Phi}_s \Theta_s \Psi + \dot{\Theta}_c \Psi) \dot{\Phi}_c \Theta_s \Psi \\
 &+ J_{yy} (\dot{\Phi}_s \Theta_c \Psi - \dot{\Theta}_s \Psi) \dot{\Phi}_c \Theta_c \Psi \\
 &- J_{zz} (\dot{\Phi}_c \Theta + \dot{\Psi}) \dot{\Phi}_s \Theta \quad (A.4) \\
 \frac{\partial T}{\partial \Psi_{i,r}} &= \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i,r}^T \left(\frac{\partial}{\partial \Psi_{i,r}} [B]_{i,r}^T \right) [J]_{i,r} [B]_{i,r} \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i,r}
 \end{aligned}$$

$$\begin{aligned}
&= \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i,}^T \begin{bmatrix} s\Theta c\Psi & -s\Psi & 0 \\ -s\Theta s\Psi & -c\Psi & 0 \\ 0 & 0 & 0 \end{bmatrix}_{i,} [J]_{i,} [B]_{i,} \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i,} \\
&= J_{xx} (\dot{\Phi} s\Theta s\Psi + \dot{\Theta} c\Psi) (\dot{\Phi} s\Theta c\Psi - \dot{\Theta} s\Psi) \\
&\quad - J_{yy} (\dot{\Phi} s\Theta c\Psi - \dot{\Theta} s\Psi) (\dot{\Phi} s\Theta s\Psi + \dot{\Theta} c\Psi) .
\end{aligned} \tag{A.5}$$

A.2 Expressions for Numerical Solution

A.2.1 System Jacobian Submatrices

$$-\frac{\partial}{\partial q_{i,}} \left[\frac{\partial T}{\partial q_{i,}} \right]^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{\partial^2 T}{\partial \Theta^2} & -\frac{\partial^2 T}{\partial \Theta \partial \Psi} \\ 0 & 0 & 0 & 0 & -\frac{\partial^2 T}{\partial \Theta \partial \Psi} & -\frac{\partial^2 T}{\partial \Psi^2} \end{bmatrix}_{i,} \tag{A.6}$$

where the four nonzero elements can be expressed as

$$\begin{aligned}
-\frac{\partial^2 T}{\partial \Theta^2} &= - \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i,}^T \left(\frac{\partial^2}{\partial \Theta_{i,}^2} [B]_{i,}^T \right) [J]_{i,} [B]_{i,} \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i,} \\
&\quad - \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i,}^T \left(\frac{\partial}{\partial \Theta_{i,}} [B]_{i,}^T \right) [J]_{i,} \left(\frac{\partial}{\partial \Theta_{i,}} [B]_{i,} \right) \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i,} \\
&= -\dot{\Phi} [J_{xx} (-\dot{\Phi} s^2 \Theta s^2 \Psi + \dot{\Phi} c^2 \Theta s^2 \Psi - \dot{\Theta} s \Theta s \Psi c \Psi) \\
&\quad + J_{yy} (-\dot{\Phi} s^2 \Theta c^2 \Psi + \dot{\Phi} c^2 \Theta c^2 \Psi + \dot{\Theta} s \Theta s \Psi c \Psi) \\
&\quad + J_{zz} (-\dot{\Phi} c^2 \Theta + \dot{\Phi} s^2 \Theta - \dot{\Psi} c \Theta)]
\end{aligned} \tag{A.7}$$

$$\begin{aligned}
-\frac{\partial^2 T}{\partial \Theta \partial \Psi} &= -\left\{ \begin{matrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{matrix} \right\}_{i_i}^T \left(\frac{\partial^2}{\partial \Psi_i \partial \Theta_i} [B]_{i_i}^T \right) [J]_{i_i} [B]_{i_i} \left\{ \begin{matrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{matrix} \right\}_{i_i} \\
&\quad - \left\{ \begin{matrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{matrix} \right\}_{i_i}^T \left(\frac{\partial}{\partial \Theta_i} [B]_{i_i}^T \right) [J]_{i_i} \left(\frac{\partial}{\partial \Psi_i} [B]_{i_i} \right) \left\{ \begin{matrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{matrix} \right\}_{i_i} \\
&= -\dot{\Phi} [J_{xx} (2\dot{\Phi}_s \Theta_c \Theta_s \Psi_c \Psi + \dot{\Theta}_c \Theta_c^2 \Psi - \dot{\Theta}_c \Theta_s^2 \Psi) \\
&\quad + J_{yy} (-2\dot{\Phi}_s \Theta_c \Theta_s \Psi_c \Psi + \dot{\Theta}_c \Theta_s^2 \Psi - \dot{\Theta}_c \Theta_c^2 \Psi)] \tag{A.8}
\end{aligned}$$

$$\begin{aligned}
-\frac{\partial^2 T}{\partial \Psi^2} &= -\left\{ \begin{matrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{matrix} \right\}_{i_i}^T \left(\frac{\partial^2}{\partial \Psi_i^2} [B]_{i_i}^T \right) [J]_{i_i} [B]_{i_i} \left\{ \begin{matrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{matrix} \right\}_{i_i} \\
&\quad - \left\{ \begin{matrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{matrix} \right\}_{i_i}^T \left(\frac{\partial}{\partial \Psi_i} [B]_{i_i}^T \right) [J]_{i_i} \left(\frac{\partial}{\partial \Psi_i} [B]_{i_i} \right) \left\{ \begin{matrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{matrix} \right\}_{i_i} \\
&= -\dot{\Phi} [J_{xx} (-\dot{\Phi}_s^2 \Theta_s^2 \Psi + \dot{\Phi}_s^2 \Theta_c^2 \Psi - 2\dot{\Theta}_s \Theta_s \Psi_c \Psi) \\
&\quad + J_{yy} (-\dot{\Phi}_s^2 \Theta_c^2 \Psi + \dot{\Phi}_s^2 \Theta_s^2 \Psi + 2\dot{\Theta}_s \Theta_s \Psi_c \Psi)] \\
&\quad - \dot{\Theta} [J_{xx} (-2\dot{\Phi}_s \Theta_s \Psi_c \Psi - \dot{\Theta}_c^2 \Psi + \dot{\Theta}_s^2 \Psi) \\
&\quad + J_{yy} (\dot{\Phi}_s \Theta_s \Psi_c \Psi + \dot{\Phi}_s \Theta_s \Psi_c \Psi - \dot{\Theta}_s^2 \Psi + \dot{\Theta}_c^2 \Psi)] \tag{A.9}
\end{aligned}$$

$$-\frac{\partial}{\partial q_{i,}} \left(\left[\frac{\partial G}{\partial q_{i,}} \right]^T \lambda \right) = -\frac{\partial}{\partial q_{i,}} \left\{ \begin{array}{c} \frac{\partial G_1}{\partial X} \lambda_1 + \dots + \frac{\partial G_{n_c}}{\partial X} \lambda_{n_c} \\ \frac{\partial G_1}{\partial Y} \lambda_1 + \dots + \frac{\partial G_{n_c}}{\partial Y} \lambda_{n_c} \\ \frac{\partial G_1}{\partial Z} \lambda_1 + \dots + \frac{\partial G_{n_c}}{\partial Z} \lambda_{n_c} \\ \frac{\partial G_1}{\partial \Phi} \lambda_1 + \dots + \frac{\partial G_{n_c}}{\partial \Phi} \lambda_{n_c} \\ \frac{\partial G_1}{\partial \Theta} \lambda_1 + \dots + \frac{\partial G_{n_c}}{\partial \Theta} \lambda_{n_c} \\ \frac{\partial G_1}{\partial \Psi} \lambda_1 + \dots + \frac{\partial G_{n_c}}{\partial \Psi} \lambda_{n_c} \end{array} \right\}_{i,} =$$

$$-\left[\begin{array}{ccc} \frac{\partial}{\partial X} \left(\frac{\partial G_1}{\partial X} \lambda_1 + \dots + \frac{\partial G_{n_c}}{\partial X} \lambda_{n_c} \right) & \dots & \frac{\partial}{\partial \Psi} \left(\frac{\partial G_1}{\partial X} \lambda_1 + \dots + \frac{\partial G_{n_c}}{\partial X} \lambda_{n_c} \right) \\ \vdots & \ddots & \vdots \\ \frac{\partial}{\partial X} \left(\frac{\partial G_1}{\partial \Psi} \lambda_1 + \dots + \frac{\partial G_{n_c}}{\partial \Psi} \lambda_{n_c} \right) & \dots & \frac{\partial}{\partial \Psi} \left(\frac{\partial G_1}{\partial \Psi} \lambda_1 + \dots + \frac{\partial G_{n_c}}{\partial \Psi} \lambda_{n_c} \right) \end{array} \right]_{i,} \quad (\text{A.10})$$

$$-\frac{\partial}{\partial \dot{q}_{i,}} \left[\frac{\partial T}{\partial q_{i,}} \right]^T = \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{\partial^2 T}{\partial \Theta \partial \dot{\Phi}} & -\frac{\partial^2 T}{\partial \Theta \partial \dot{\Theta}} & -\frac{\partial^2 T}{\partial \Theta \partial \dot{\Psi}} \\ 0 & 0 & 0 & -\frac{\partial^2 T}{\partial \Psi \partial \dot{\Phi}} & -\frac{\partial^2 T}{\partial \Psi \partial \dot{\Theta}} & 0 \end{array} \right]_{i,} \quad (\text{A.11})$$

where the five nonzero elements are

$$-\frac{\partial^2 T}{\partial \Theta \partial \dot{\Phi}} = -\left\{ \begin{array}{c} 1 \\ 0 \\ 0 \end{array} \right\} \left(\frac{\partial}{\partial \Theta_{i,}} [B]_{i,}^T \right) [J]_{i,} [B]_{i,} \left\{ \begin{array}{c} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{array} \right\}_{i,}$$

$$\begin{aligned}
& - \left\{ \begin{matrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{matrix} \right\}_{i_i}^T \left(\frac{\partial}{\partial \Theta_{i_i}} [B]_{i_i}^T \right) [J]_{i_i}, [B]_{i_i}, \left\{ \begin{matrix} 1 \\ 0 \\ 0 \end{matrix} \right\} \\
& = -2\dot{\Phi}(J_{xx}s^2\Psi + J_{yy}c^2\Psi - J_{zz})s\Theta c\Theta \\
& \quad - \dot{\Theta}(J_{xx} - J_{yy})c\Theta s\Psi c\Psi + \dot{\Psi}J_{zz}s\Theta
\end{aligned} \tag{A.12}$$

$$\begin{aligned}
-\frac{\partial^2 T}{\partial \Theta \partial \dot{\Theta}} &= - \left\{ \begin{matrix} 0 \\ 1 \\ 0 \end{matrix} \right\} \left(\frac{\partial}{\partial \Theta_{i_i}} [B]_{i_i}^T \right) [J]_{i_i}, [B]_{i_i}, \left\{ \begin{matrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{matrix} \right\}_{i_i} \\
& - \left\{ \begin{matrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{matrix} \right\}_{i_i}^T \left(\frac{\partial}{\partial \Theta_{i_i}} [B]_{i_i}^T \right) [J]_{i_i}, [B]_{i_i}, \left\{ \begin{matrix} 0 \\ 1 \\ 0 \end{matrix} \right\} \\
& = -\dot{\Phi}(J_{xx} - J_{yy})c\Theta s\Psi c\Psi
\end{aligned} \tag{A.13}$$

$$\begin{aligned}
-\frac{\partial^2 T}{\partial \Theta \partial \dot{\Psi}} &= - \left\{ \begin{matrix} 0 \\ 0 \\ 1 \end{matrix} \right\} \left(\frac{\partial}{\partial \Theta_{i_i}} [B]_{i_i}^T \right) [J]_{i_i}, [B]_{i_i}, \left\{ \begin{matrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{matrix} \right\}_{i_i} \\
& - \left\{ \begin{matrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{matrix} \right\}_{i_i}^T \left(\frac{\partial}{\partial \Theta_{i_i}} [B]_{i_i}^T \right) [J]_{i_i}, [B]_{i_i}, \left\{ \begin{matrix} 0 \\ 0 \\ 1 \end{matrix} \right\} \\
& = \dot{\Phi}J_{zz}s\Theta
\end{aligned} \tag{A.14}$$

$$-\frac{\partial^2 T}{\partial \Psi \partial \dot{\Phi}} = - \left\{ \begin{matrix} 1 \\ 0 \\ 0 \end{matrix} \right\} \left(\frac{\partial}{\partial \Psi_{i_i}} [B]_{i_i}^T \right) [J]_{i_i}, [B]_{i_i}, \left\{ \begin{matrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{matrix} \right\}_{i_i}$$

$$\begin{aligned}
& - \left\{ \begin{matrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{matrix} \right\}_{i_i}^T \left(\frac{\partial}{\partial \Psi_{i_i}} [B]_{i_i}^T \right) [J]_{i_i}, [B]_{i_i}, \begin{Bmatrix} 1 \\ 0 \\ 0 \end{Bmatrix} \\
& = -2\dot{\Phi}(J_{xx} - J_{yy})s^2\Theta s\Psi c\Psi - \dot{\Theta}(J_{xx} - J_{yy})s\Theta c2\Psi \quad (A.15)
\end{aligned}$$

$$\begin{aligned}
-\frac{\partial^2 T}{\partial \Psi \partial \dot{\Theta}} &= - \begin{Bmatrix} 0 \\ 1 \\ 0 \end{Bmatrix} \left(\frac{\partial}{\partial \Psi_{i_i}} [B]_{i_i}^T \right) [J]_{i_i}, [B]_{i_i}, \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i_i} \\
& - \left\{ \begin{matrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{matrix} \right\}_{i_i}^T \left(\frac{\partial}{\partial \Psi_{i_i}} [B]_{i_i}^T \right) [J]_{i_i}, [B]_{i_i}, \begin{Bmatrix} 0 \\ 1 \\ 0 \end{Bmatrix} \\
& = -\dot{\Phi}(J_{xx} - J_{yy})s\Theta c2\Psi - 2\dot{\Theta}(-J_{xx} + J_{yy})s\Psi c\Psi \quad (A.16)
\end{aligned}$$

$$\frac{\partial}{\partial q_{i_i}} (-[C]_{i_i}, \dot{q}_{i_i}) = -\frac{\partial}{\partial q_{i_i}} \left\{ \begin{matrix} C_{11}\dot{X} + C_{12}\dot{Y} + C_{13}\dot{Z} + C_{14}\dot{\Phi} + C_{15}\dot{\Theta} + C_{16}\dot{\Psi} \\ C_{21}\dot{X} + C_{22}\dot{Y} + C_{23}\dot{Z} + C_{24}\dot{\Phi} + C_{25}\dot{\Theta} + C_{26}\dot{\Psi} \\ C_{31}\dot{X} + C_{32}\dot{Y} + C_{33}\dot{Z} + C_{34}\dot{\Phi} + C_{35}\dot{\Theta} + C_{36}\dot{\Psi} \\ C_{41}\dot{X} + C_{42}\dot{Y} + C_{43}\dot{Z} + C_{44}\dot{\Phi} + C_{45}\dot{\Theta} + C_{46}\dot{\Psi} \\ C_{51}\dot{X} + C_{52}\dot{Y} + C_{53}\dot{Z} + C_{54}\dot{\Phi} + C_{55}\dot{\Theta} + C_{56}\dot{\Psi} \\ C_{61}\dot{X} + C_{62}\dot{Y} + C_{63}\dot{Z} + C_{64}\dot{\Phi} + C_{65}\dot{\Theta} + C_{66}\dot{\Psi} \end{matrix} \right\}_{i_i},$$

$$= \begin{bmatrix} -\frac{\partial}{\partial X}(C_{11}\dot{X} + \dots + C_{16}\dot{\Psi}) & \dots & -\frac{\partial}{\partial \Psi}(C_{11}\dot{X} + \dots + C_{16}\dot{\Psi}) \\ \vdots & \ddots & \vdots \\ -\frac{\partial}{\partial X}(C_{61}\dot{X} + \dots + C_{66}\dot{\Psi}) & \dots & -\frac{\partial}{\partial \Psi}(C_{61}\dot{X} + \dots + C_{66}\dot{\Psi}) \end{bmatrix}_{i_i},$$

$$= \left[\left\{ -\frac{\partial}{\partial X}[C]\dot{q}_{i_i} \right\} \quad \left\{ -\frac{\partial}{\partial Y}[C]\dot{q}_{i_i} \right\} \quad \dots \quad \left\{ -\frac{\partial}{\partial \Psi}[C]\dot{q}_{i_i} \right\} \right]_{i_i},$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \left\{ -\frac{\partial}{\partial \Theta} [C] \dot{q}_i, \right\} \left\{ -\frac{\partial}{\partial \Psi} [C] \dot{q}_i, \right\} \quad (A.17)$$

where the two nonzero columns can be shown as

$$\begin{aligned} \left\{ -\frac{\partial}{\partial \Theta} [C] \dot{q}_i, \right\} &= -\left(\frac{\partial}{\partial \Theta_{i_r}} [B]_{i_r}^T \right) [J]_{i_r} [B]_{i_r} \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i_r} - [B]_{i_r}^T [J]_{i_r} \left(\frac{\partial}{\partial \Theta_{i_r}} [B]_{i_r} \right) \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i_r} \\ &= \begin{Bmatrix} 0 \\ 0 \\ 0 \\ -2\dot{\Phi}(J_{xx}s^2\Psi + J_{yy}c^2\Psi - J_{zz})s\Theta c\Theta \\ -\dot{\Phi}(J_{xx} - J_{yy})c\Theta s\Psi c\Psi \\ \dot{\Psi}J_{zz}s\Theta \end{Bmatrix}_{i_r} \end{aligned} \quad (A.18)$$

$$\begin{aligned} \left\{ -\frac{\partial}{\partial \Psi} [C] \dot{q}_i, \right\} &= -\left(\frac{\partial}{\partial \Psi_{i_r}} [B]_{i_r}^T \right) [J]_{i_r} [B]_{i_r} \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i_r} - [B]_{i_r}^T [J]_{i_r} \left(\frac{\partial}{\partial \Psi_{i_r}} [B]_{i_r} \right) \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i_r} \\ &= \begin{Bmatrix} 0 \\ 0 \\ 0 \\ -2\dot{\Phi}(J_{xx} - J_{yy})s^2\Theta s\Psi c\Psi - \dot{\Theta}(J_{xx} - J_{yy})s\Theta c2\Psi \\ -\dot{\Phi}(J_{xx} - J_{yy})s\Theta c2\Psi - 2\dot{\Theta}(-J_{xx} + J_{yy})s\Psi c\Psi \\ 0 \end{Bmatrix}_{i_r} \end{aligned} \quad (A.19)$$

Note that comparing Eq.(A.11) with Eq.(A.17) gives

$$\frac{\partial}{\partial q_i}(-[C]_i, \dot{q}_i) = \left[-\frac{\partial}{\partial \dot{q}_i} \left[\frac{\partial T}{\partial q_i} \right]^T \right]^T. \quad (\text{A.20})$$

$$\frac{\partial L_i}{\partial q_i} = \begin{bmatrix} \frac{\partial L_{i,x}}{\partial X_i} & \frac{\partial L_{i,x}}{\partial Y_i} & \frac{\partial L_{i,x}}{\partial Z_i} & \frac{\partial L_{i,x}}{\partial \Phi_i} & \frac{\partial L_{i,x}}{\partial \Theta_i} & \frac{\partial L_{i,x}}{\partial \Psi_i} \\ \frac{\partial L_{i,y}}{\partial X_i} & \frac{\partial L_{i,y}}{\partial Y_i} & \frac{\partial L_{i,y}}{\partial Z_i} & \frac{\partial L_{i,y}}{\partial \Phi_i} & \frac{\partial L_{i,y}}{\partial \Theta_i} & \frac{\partial L_{i,y}}{\partial \Psi_i} \\ \frac{\partial L_{i,z}}{\partial X_i} & \frac{\partial L_{i,z}}{\partial Y_i} & \frac{\partial L_{i,z}}{\partial Z_i} & \frac{\partial L_{i,z}}{\partial \Phi_i} & \frac{\partial L_{i,z}}{\partial \Theta_i} & \frac{\partial L_{i,z}}{\partial \Psi_i} \\ \frac{\partial L_{i,\phi}}{\partial X_i} & \frac{\partial L_{i,\phi}}{\partial Y_i} & \frac{\partial L_{i,\phi}}{\partial Z_i} & \frac{\partial L_{i,\phi}}{\partial \Phi_i} & \frac{\partial L_{i,\phi}}{\partial \Theta_i} & \frac{\partial L_{i,\phi}}{\partial \Psi_i} \\ \frac{\partial L_{i,\theta}}{\partial X_i} & \frac{\partial L_{i,\theta}}{\partial Y_i} & \frac{\partial L_{i,\theta}}{\partial Z_i} & \frac{\partial L_{i,\theta}}{\partial \Phi_i} & \frac{\partial L_{i,\theta}}{\partial \Theta_i} & \frac{\partial L_{i,\theta}}{\partial \Psi_i} \\ \frac{\partial L_{i,\psi}}{\partial X_i} & \frac{\partial L_{i,\psi}}{\partial Y_i} & \frac{\partial L_{i,\psi}}{\partial Z_i} & \frac{\partial L_{i,\psi}}{\partial \Phi_i} & \frac{\partial L_{i,\psi}}{\partial \Theta_i} & \frac{\partial L_{i,\psi}}{\partial \Psi_i} \end{bmatrix} \quad (\text{A.21})$$

where L is highly dependent on the external forces. The contribution of each force to the the above submatrix can be expressed as below. Gravitational forces contribute nothing to the submatrix.

Each spring force contributes the following expressions;

$$\begin{aligned} \frac{\partial}{\partial X_i} \left\{ \begin{matrix} L_{i,x} \\ L_{i,y} \\ L_{i,z} \end{matrix} \right\}_{spr} &= \frac{\partial}{\partial X_i} \left(\frac{k(l_{ij} - l_0)}{l_{ij}} \begin{bmatrix} l_{ijx} \\ l_{ijy} \\ l_{ijz} \end{bmatrix} \right) \\ &= \frac{kl_0 l_{ijx}}{l_{ij}^3} \begin{bmatrix} l_{ijx} \\ l_{ijy} \\ l_{ijz} \end{bmatrix} + \frac{k(l_{ij} - l_0)}{l_{ij}} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \end{aligned} \quad (\text{A.22})$$

$$\frac{\partial l_{ij}}{\partial X_i} = \frac{\partial}{\partial X_i} \sqrt{l_{ijx}^2 + l_{ijy}^2 + l_{ijz}^2} = \frac{l_{ijx}}{l_{ij}}. \quad (\text{A.23})$$

Similar expressions obtained for the partial derivatives with respect to Y_{i_r} and Z_{i_r} .

$$\begin{aligned} \frac{\partial}{\partial \Phi_{i_r}} \left\{ \begin{matrix} L_{i_r, x} \\ L_{i_r, y} \\ L_{i_r, z} \end{matrix} \right\}_{spr} &= \frac{\partial}{\partial \Phi_{i_r}} \left(\frac{k(l_{ij} - l_0)}{l_{ij}} \left\{ \begin{matrix} l_{ij, x} \\ l_{ij, y} \\ l_{ij, z} \end{matrix} \right\} \right) \\ &= \frac{kl_0}{l_{ij}^2} \frac{\partial l_{ij}}{\partial \Phi_{i_r}} \left\{ \begin{matrix} l_{ij, x} \\ l_{ij, y} \\ l_{ij, z} \end{matrix} \right\} + \frac{k(l_{ij} - l_0)}{l_{ij}} \frac{\partial}{\partial \Phi_{i_r}} \left\{ \begin{matrix} l_{ij, x} \\ l_{ij, y} \\ l_{ij, z} \end{matrix} \right\} \end{aligned} \quad (A.24)$$

$$\frac{\partial l_{ij}}{\partial \Phi_{i_r}} = \frac{1}{l_{ij}} [l_{ij, x} \quad l_{ij, y} \quad l_{ij, z}] \frac{\partial}{\partial \Phi_{i_r}} \left\{ \begin{matrix} l_{ij, x} \\ l_{ij, y} \\ l_{ij, z} \end{matrix} \right\} \quad (A.25)$$

$$\frac{\partial}{\partial \Phi_{i_r}} \left\{ \begin{matrix} l_{ij, x} \\ l_{ij, y} \\ l_{ij, z} \end{matrix} \right\} = \frac{\partial [A]_{i_r}}{\partial \Phi_{i_r}} \left\{ \begin{matrix} x \\ y \\ z \end{matrix} \right\}_{I_m}. \quad (A.26)$$

Similar expressions obtained for the partial derivatives with respect to Θ_{i_r} and Ψ_{i_r} .

$$\begin{aligned} \frac{\partial}{\partial X_{i_r}} \left\{ \begin{matrix} L_{i_r, \Phi} \\ L_{i_r, \Theta} \\ L_{i_r, \Psi} \end{matrix} \right\}_{spr} &= \frac{\partial}{\partial X_{i_r}} \left(\frac{k(l_{ij} - l_0)}{l_{ij}} [l_{ij, x} \quad l_{ij, y} \quad l_{ij, z}] \right) \left\{ \begin{matrix} \frac{\partial [A]_{i_r}}{\partial \Phi_{i_r}} \\ \frac{\partial [A]_{i_r}}{\partial \Theta_{i_r}} \\ \frac{\partial [A]_{i_r}}{\partial \Psi_{i_r}} \end{matrix} \right\} \left\{ \begin{matrix} x \\ y \\ z \end{matrix} \right\}_{I_m} \\ &= \left(\frac{kl_0 l_{ij, x}}{l_{ij}^3} [l_{ij, x} \quad l_{ij, y} \quad l_{ij, z}] + \frac{k(l_{ij} - l_0)}{l_{ij}} [1 \quad 0 \quad 0] \right) \left\{ \begin{matrix} \frac{\partial [A]_{i_r}}{\partial \Phi_{i_r}} \\ \frac{\partial [A]_{i_r}}{\partial \Theta_{i_r}} \\ \frac{\partial [A]_{i_r}}{\partial \Psi_{i_r}} \end{matrix} \right\} \left\{ \begin{matrix} x \\ y \\ z \end{matrix} \right\}_{I_m}. \end{aligned} \quad (A.27)$$

Similar expressions obtained for the partial derivatives with respect to Y_{i_r} and Z_{i_r} .

$$\begin{aligned}
\frac{\partial}{\partial \Phi_{i_r}} \left\{ \begin{matrix} L_{i_r, \Phi} \\ L_{i_r, \Theta} \\ L_{i_r, \Psi} \end{matrix} \right\}_{spr} &= \frac{\partial}{\partial \Phi_{i_r}} \left(\frac{k(l_{IJ} - l_0)}{l_{IJ}} [l_{IJx} \quad l_{IJy} \quad l_{IJz}] \right) \left\{ \begin{matrix} \frac{\partial[A]_{i_r}}{\partial \Phi_{i_r}} \\ \frac{\partial[A]_{i_r}}{\partial \Theta_{i_r}} \\ \frac{\partial[A]_{i_r}}{\partial \Psi_{i_r}} \end{matrix} \right\} \left\{ \begin{matrix} x \\ y \\ z \end{matrix} \right\}_{I_m} \\
&= \frac{kl_0}{l_{IJ}^2} \frac{\partial l_{IJ}}{\partial \Phi_{i_r}} [l_{IJx} \quad l_{IJy} \quad l_{IJz}] \left\{ \begin{matrix} \frac{\partial[A]_{i_r}}{\partial \Phi_{i_r}} \\ \frac{\partial[A]_{i_r}}{\partial \Theta_{i_r}} \\ \frac{\partial[A]_{i_r}}{\partial \Psi_{i_r}} \end{matrix} \right\} \left\{ \begin{matrix} x \\ y \\ z \end{matrix} \right\}_{I_m} \\
&\quad + \frac{k(l_{IJ} - l_0)}{l_{IJ}} \frac{\partial}{\partial \Phi_{i_r}} [l_{IJx} \quad l_{IJy} \quad l_{IJz}] \left\{ \begin{matrix} \frac{\partial[A]_{i_r}}{\partial \Phi_{i_r}} \\ \frac{\partial[A]_{i_r}}{\partial \Theta_{i_r}} \\ \frac{\partial[A]_{i_r}}{\partial \Psi_{i_r}} \end{matrix} \right\} \left\{ \begin{matrix} x \\ y \\ z \end{matrix} \right\}_{I_m} \\
&\quad + \frac{k(l_{IJ} - l_0)}{l_{IJ}} [l_{IJx} \quad l_{IJy} \quad l_{IJz}] \left\{ \begin{matrix} \frac{\partial^2[A]_{i_r}}{\partial \Phi_{i_r}^2} \\ \frac{\partial^2[A]_{i_r}}{\partial \Phi_{i_r} \partial \Theta_{i_r}} \\ \frac{\partial^2[A]_{i_r}}{\partial \Phi_{i_r} \partial \Psi_{i_r}} \end{matrix} \right\} \left\{ \begin{matrix} x \\ y \\ z \end{matrix} \right\}_{I_m} .
\end{aligned} \tag{A.28}$$

Similar expressions obtained for the partial derivatives with respect to Θ_{i_r} and Ψ_{i_r} . The expressions for the partial derivatives with respect to the generalized coordinates associated with the J marker are negative of the above expressions.

Each damping force contributes the following expressions;

$$\begin{aligned}
\frac{\partial}{\partial X_{i_r}} \begin{Bmatrix} L_{i_r,x} \\ L_{i_r,y} \\ L_{i_r,z} \end{Bmatrix}_{dam} &= \frac{\partial}{\partial X_{i_r}} \left(\frac{c \dot{l}_{ij}}{l_{ij}} \begin{Bmatrix} l_{ij,x} \\ l_{ij,y} \\ l_{ij,z} \end{Bmatrix} \right) \\
&= \frac{c}{l_{ij}} \frac{\partial \dot{l}_{ij}}{\partial X_{i_r}} \begin{Bmatrix} l_{ij,x} \\ l_{ij,y} \\ l_{ij,z} \end{Bmatrix} - \frac{c \dot{l}_{ij}}{l_{ij}^2} \frac{\partial l_{ij}}{\partial X_{i_r}} \begin{Bmatrix} l_{ij,x} \\ l_{ij,y} \\ l_{ij,z} \end{Bmatrix} + \frac{c \dot{l}_{ij}}{l_{ij}} \frac{\partial}{\partial X_{i_r}} \begin{Bmatrix} l_{ij,x} \\ l_{ij,y} \\ l_{ij,z} \end{Bmatrix} \\
&= \frac{c}{l_{ij}^2} \left(\dot{l}_{ij,x} - 2 \frac{l_{ij,x} \dot{l}_{ij}}{l_{ij}} \right) \begin{Bmatrix} l_{ij,x} \\ l_{ij,y} \\ l_{ij,z} \end{Bmatrix} + \frac{c \dot{l}_{ij}}{l_{ij}} \begin{Bmatrix} 1 \\ 0 \\ 0 \end{Bmatrix}
\end{aligned} \tag{A.29}$$

$$\begin{aligned}
\frac{\partial \dot{l}_{ij}}{\partial X_{i_r}} &= \frac{\partial}{\partial X_{i_r}} \left(\frac{1}{l_{ij}} (l_{ij,x} \dot{l}_{ij,x} + l_{ij,y} \dot{l}_{ij,y} + l_{ij,z} \dot{l}_{ij,z}) \right) \\
&= -\frac{l_{ij,x}}{l_{ij}^2} \dot{l}_{ij} + \frac{\dot{l}_{ij,x}}{l_{ij}}.
\end{aligned} \tag{A.30}$$

Similar expressions obtained for the partial derivatives with respect to Y_{i_r} and Z_{i_r} .

$$\begin{aligned}
\frac{\partial}{\partial \Phi_{i_r}} \begin{Bmatrix} L_{i_r,x} \\ L_{i_r,y} \\ L_{i_r,z} \end{Bmatrix}_{dam} &= \frac{\partial}{\partial \Phi_{i_r}} \left(\frac{c \dot{l}_{ij}}{l_{ij}} \begin{Bmatrix} l_{ij,x} \\ l_{ij,y} \\ l_{ij,z} \end{Bmatrix} \right) \\
&= \frac{c}{l_{ij}} \frac{\partial \dot{l}_{ij}}{\partial \Phi_{i_r}} \begin{Bmatrix} l_{ij,x} \\ l_{ij,y} \\ l_{ij,z} \end{Bmatrix} - \frac{c \dot{l}_{ij}}{l_{ij}^2} \frac{\partial l_{ij}}{\partial \Phi_{i_r}} \begin{Bmatrix} l_{ij,x} \\ l_{ij,y} \\ l_{ij,z} \end{Bmatrix} + \frac{c \dot{l}_{ij}}{l_{ij}} \frac{\partial}{\partial \Phi_{i_r}} \begin{Bmatrix} l_{ij,x} \\ l_{ij,y} \\ l_{ij,z} \end{Bmatrix}
\end{aligned} \tag{A.31}$$

$$\begin{aligned}
\frac{\partial \dot{l}_{ij}}{\partial \Phi_{i_r}} &= \frac{\partial}{\partial \Phi_{i_r}} \left(\frac{1}{l_{ij}} (l_{ij,x} \dot{l}_{ij,x} + l_{ij,y} \dot{l}_{ij,y} + l_{ij,z} \dot{l}_{ij,z}) \right) \\
&= -\frac{\dot{l}_{ij}}{l_{ij}} \frac{\partial l_{ij}}{\partial \Phi_{i_r}} \\
&\quad + \begin{bmatrix} \dot{l}_{ij,x} & \dot{l}_{ij,y} & \dot{l}_{ij,z} \end{bmatrix} \frac{\partial}{\partial \Phi_{i_r}} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_{I_m} + \begin{bmatrix} l_{ij,x} & l_{ij,y} & l_{ij,z} \end{bmatrix} \frac{\partial}{\partial \Phi_{i_r}} \begin{Bmatrix} \dot{l}_{ij,x} \\ \dot{l}_{ij,y} \\ \dot{l}_{ij,z} \end{Bmatrix}_{I_m}.
\end{aligned} \tag{A.32}$$

Similar expressions obtained for the partial derivatives with respect to Θ_{i_r} and Ψ_{i_r} .

$$\begin{aligned}
\frac{\partial}{\partial X_{i_r}} \begin{Bmatrix} L_{i_r, \Phi} \\ L_{i_r, \Theta} \\ L_{i_r, \Psi} \end{Bmatrix}_{dam} &= \frac{\partial}{\partial X_{i_r}} \left(\frac{c \dot{l}_\mu}{l_\mu} [l_{\mu_x} \quad l_{\mu_y} \quad l_{\mu_z}] \right) \begin{Bmatrix} \frac{\partial[A]_{i_r}}{\partial \Phi_{i_r}} \\ \frac{\partial[A]_{i_r}}{\partial \Theta_{i_r}} \\ \frac{\partial[A]_{i_r}}{\partial \Psi_{i_r}} \end{Bmatrix} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_{I_m} \\
&= \left(-\frac{c \dot{l}_{\mu_x}}{l_\mu^2} [l_{\mu_x} \quad l_{\mu_y} \quad l_{\mu_z}] + \frac{c \dot{l}_\mu}{l_\mu} [1 \quad 0 \quad 0] \right) \begin{Bmatrix} \frac{\partial[A]_{i_r}}{\partial \Phi_{i_r}} \\ \frac{\partial[A]_{i_r}}{\partial \Theta_{i_r}} \\ \frac{\partial[A]_{i_r}}{\partial \Psi_{i_r}} \end{Bmatrix} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_{I_m}. \quad (A.33)
\end{aligned}$$

Similar expressions obtained for the partial derivatives with respect to Y_{i_r} and Z_{i_r} .

$$\begin{aligned}
\frac{\partial}{\partial \Phi_{i_r}} \begin{Bmatrix} L_{i_r, \Phi} \\ L_{i_r, \Theta} \\ L_{i_r, \Psi} \end{Bmatrix}_{dam} &= \frac{\partial}{\partial \Phi_{i_r}} \left(\frac{c \dot{l}_\mu}{l_\mu} [l_{\mu_x} \quad l_{\mu_y} \quad l_{\mu_z}] \right) \begin{Bmatrix} \frac{\partial[A]_{i_r}}{\partial \Phi_{i_r}} \\ \frac{\partial[A]_{i_r}}{\partial \Theta_{i_r}} \\ \frac{\partial[A]_{i_r}}{\partial \Psi_{i_r}} \end{Bmatrix} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_{I_m} \\
&= \left(\frac{c}{l_\mu} \frac{\partial \dot{l}_\mu}{\partial \Phi_{i_r}} [l_{\mu_x} \quad l_{\mu_y} \quad l_{\mu_z}] \right) \begin{Bmatrix} \frac{\partial[A]_{i_r}}{\partial \Phi_{i_r}} \\ \frac{\partial[A]_{i_r}}{\partial \Theta_{i_r}} \\ \frac{\partial[A]_{i_r}}{\partial \Psi_{i_r}} \end{Bmatrix} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_{I_m} \\
&\quad - \left(\frac{c \dot{l}_\mu}{l_\mu^2} \frac{\partial l_\mu}{\partial \Phi_{i_r}} [l_{\mu_x} \quad l_{\mu_y} \quad l_{\mu_z}] \right) \begin{Bmatrix} \frac{\partial[A]_{i_r}}{\partial \Phi_{i_r}} \\ \frac{\partial[A]_{i_r}}{\partial \Theta_{i_r}} \\ \frac{\partial[A]_{i_r}}{\partial \Psi_{i_r}} \end{Bmatrix} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_{I_m}
\end{aligned}$$

$$\begin{aligned}
& + \left(\frac{c\dot{l}_\mu}{l_\mu} \frac{\partial}{\partial \Phi_{i,\mu}} [l_{\mu x} \quad l_{\mu y} \quad l_{\mu z}] \right) \left\{ \begin{array}{c} \frac{\partial[A]_{i,\mu}}{\partial \Phi_{i,\mu}} \\ \frac{\partial[A]_{i,\mu}}{\partial \Theta_{i,\mu}} \\ \frac{\partial[A]_{i,\mu}}{\partial \Psi_{i,\mu}} \end{array} \right\} \left\{ \begin{array}{c} x \\ y \\ z \end{array} \right\}_{I_\mu} \\
& + \left(\frac{c\dot{l}_\mu}{l_\mu} [l_{\mu x} \quad l_{\mu y} \quad l_{\mu z}] \right) \left\{ \begin{array}{c} \frac{\partial^2[A]_{i,\mu}}{\partial \Phi_{i,\mu}^2} \\ \frac{\partial^2[A]_{i,\mu}}{\partial \Phi_{i,\mu} \partial \Theta_{i,\mu}} \\ \frac{\partial^2[A]_{i,\mu}}{\partial \Phi_{i,\mu} \partial \Psi_{i,\mu}} \end{array} \right\} \left\{ \begin{array}{c} x \\ y \\ z \end{array} \right\}_{I_\mu} .
\end{aligned} \tag{A.34}$$

Similar expressions obtained for the partial derivatives with respect to $\Theta_{i,\mu}$ and $\Psi_{i,\mu}$. The expressions for the partial derivatives with respect to the generalized coordinates associated with the J marker are negative of the above expressions.

$$\frac{\partial L_{i,\mu}}{\partial \dot{q}_{i,\mu}} = \begin{bmatrix} \frac{\partial L_{i,\mu}}{\partial \dot{X}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{Y}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{Z}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{\Phi}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{\Theta}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{\Psi}_{i,\mu}} \\ \frac{\partial L_{i,\mu}}{\partial \dot{X}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{Y}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{Z}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{\Phi}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{\Theta}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{\Psi}_{i,\mu}} \\ \frac{\partial L_{i,\mu}}{\partial \dot{X}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{Y}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{Z}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{\Phi}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{\Theta}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{\Psi}_{i,\mu}} \\ \frac{\partial L_{i,\mu}}{\partial \dot{X}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{Y}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{Z}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{\Phi}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{\Theta}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{\Psi}_{i,\mu}} \\ \frac{\partial L_{i,\mu}}{\partial \dot{X}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{Y}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{Z}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{\Phi}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{\Theta}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{\Psi}_{i,\mu}} \\ \frac{\partial L_{i,\mu}}{\partial \dot{X}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{Y}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{Z}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{\Phi}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{\Theta}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{\Psi}_{i,\mu}} \\ \frac{\partial L_{i,\mu}}{\partial \dot{X}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{Y}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{Z}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{\Phi}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{\Theta}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{\Psi}_{i,\mu}} \\ \frac{\partial L_{i,\mu}}{\partial \dot{X}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{Y}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{Z}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{\Phi}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{\Theta}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{\Psi}_{i,\mu}} \\ \frac{\partial L_{i,\mu}}{\partial \dot{X}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{Y}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{Z}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{\Phi}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{\Theta}_{i,\mu}} & \frac{\partial L_{i,\mu}}{\partial \dot{\Psi}_{i,\mu}} \end{bmatrix} \tag{A.35}$$

where L is highly dependent on the external forces. The contribution of each force to the the above submatrix can be expressed as below. Gravitational forces and spring forces contribute nothing to the submatrix.

Each damping force contributes the following expressions;

$$\frac{\partial}{\partial \dot{X}_{i_r}} \begin{Bmatrix} L_{i_r,x} \\ L_{i_r,y} \\ L_{i_r,z} \end{Bmatrix}_{dam} = \frac{\partial}{\partial \dot{X}_{i_r}} \left(\frac{c \dot{l}_{ij}}{l_{ij}} \begin{Bmatrix} l_{ij,x} \\ l_{ij,y} \\ l_{ij,z} \end{Bmatrix} \right) = \frac{c}{l_{ij}} \frac{\partial \dot{l}_{ij}}{\partial \dot{X}_{i_r}} \begin{Bmatrix} l_{ij,x} \\ l_{ij,y} \\ l_{ij,z} \end{Bmatrix} = \frac{c l_{ij,x}}{l_{ij}^2} \begin{Bmatrix} l_{ij,x} \\ l_{ij,y} \\ l_{ij,z} \end{Bmatrix} \quad (A.36)$$

$$\frac{\partial \dot{l}_{ij}}{\partial \dot{X}_{i_r}} = \frac{\partial}{\partial \dot{X}_{i_r}} \left(\frac{1}{l_{ij}} (l_{ij,x} \dot{l}_{ij,x} + l_{ij,y} \dot{l}_{ij,y} + l_{ij,z} \dot{l}_{ij,z}) \right) = \frac{l_{ij,x}}{l_{ij}}. \quad (A.37)$$

Similar expressions obtained for the partial derivatives with respect to Y_{i_r} and Z_{i_r} .

$$\frac{\partial}{\partial \dot{\Phi}_{i_r}} \begin{Bmatrix} L_{i_r,x} \\ L_{i_r,y} \\ L_{i_r,z} \end{Bmatrix}_{dam} = \frac{\partial}{\partial \dot{\Phi}_{i_r}} \left(\frac{c \dot{l}_{ij}}{l_{ij}} \begin{Bmatrix} l_{ij,x} \\ l_{ij,y} \\ l_{ij,z} \end{Bmatrix} \right) = \frac{c}{l_{ij}} \frac{\partial \dot{l}_{ij}}{\partial \dot{\Phi}_{i_r}} \begin{Bmatrix} l_{ij,x} \\ l_{ij,y} \\ l_{ij,z} \end{Bmatrix} \quad (A.38)$$

$$\begin{aligned} \frac{\partial \dot{l}_{ij}}{\partial \dot{\Phi}_{i_r}} &= \frac{1}{l_{ij}} \left(l_{ij,x} \frac{\partial \dot{l}_{ij,x}}{\partial \dot{\Phi}_{i_r}} + l_{ij,y} \frac{\partial \dot{l}_{ij,y}}{\partial \dot{\Phi}_{i_r}} + l_{ij,z} \frac{\partial \dot{l}_{ij,z}}{\partial \dot{\Phi}_{i_r}} \right) \\ &= \frac{1}{l_{ij}} \begin{bmatrix} l_{ij,x} & l_{ij,y} & l_{ij,z} \end{bmatrix} \begin{Bmatrix} \frac{\partial \dot{l}_{ij,x}}{\partial \dot{\Phi}_{i_r}} \\ \frac{\partial \dot{l}_{ij,y}}{\partial \dot{\Phi}_{i_r}} \\ \frac{\partial \dot{l}_{ij,z}}{\partial \dot{\Phi}_{i_r}} \end{Bmatrix}. \end{aligned} \quad (A.39)$$

Similar expressions obtained for the partial derivatives with respect to Θ_{i_r} and Ψ_{i_r} .

$$\begin{aligned}
\frac{\partial}{\partial \dot{X}_{i_r}} \left\{ \begin{matrix} L_{i_r, \Phi} \\ L_{i_r, \Theta} \\ L_{i_r, \Psi} \end{matrix} \right\}_{dam} &= \frac{\partial}{\partial \dot{X}_{i_r}} \left(\frac{c \dot{l}_{ij}}{l_{ij}} \begin{bmatrix} l_{ijx} & l_{ijy} & l_{ijz} \end{bmatrix} \right) \left\{ \begin{matrix} \frac{\partial[A]_{i_r}}{\partial \Phi_{i_r}} \\ \frac{\partial[A]_{i_r}}{\partial \Theta_{i_r}} \\ \frac{\partial[A]_{i_r}}{\partial \Psi_{i_r}} \end{matrix} \right\} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_{I_m} \\
&= \frac{c l_{ijx}}{l_{ij}^2} \begin{bmatrix} l_{ijx} & l_{ijy} & l_{ijz} \end{bmatrix} \left\{ \begin{matrix} \frac{\partial[A]_{i_r}}{\partial \Phi_{i_r}} \\ \frac{\partial[A]_{i_r}}{\partial \Theta_{i_r}} \\ \frac{\partial[A]_{i_r}}{\partial \Psi_{i_r}} \end{matrix} \right\} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_{I_m}. \tag{A.40}
\end{aligned}$$

Similar expressions obtained for the partial derivatives with respect to Y_{i_r} and Z_{i_r} .

$$\frac{\partial}{\partial \dot{\Phi}_{i_r}} \left\{ \begin{matrix} L_{i_r, \Phi} \\ L_{i_r, \Theta} \\ L_{i_r, \Psi} \end{matrix} \right\}_{dam} = \frac{c}{l_{ij}} \frac{\partial \dot{l}_{ij}}{\partial \dot{\Phi}_{i_r}} \begin{bmatrix} l_{ijx} & l_{ijy} & l_{ijz} \end{bmatrix} \left\{ \begin{matrix} \frac{\partial[A]_{i_r}}{\partial \Phi_{i_r}} \\ \frac{\partial[A]_{i_r}}{\partial \Theta_{i_r}} \\ \frac{\partial[A]_{i_r}}{\partial \Psi_{i_r}} \end{matrix} \right\} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_{I_m} \tag{A.41}$$

$$\frac{\partial}{\partial \dot{\Phi}_{i_r}} \left\{ \begin{matrix} \dot{l}_{ijx} \\ \dot{l}_{ijy} \\ \dot{l}_{ijz} \end{matrix} \right\} = \frac{\partial[A]_{i_r}}{\partial \Phi_{i_r}} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_{I_m}. \tag{A.42}$$

Similar expressions obtained for the partial derivatives with respect to Θ_{i_r} and Ψ_{i_r} . The expressions for the partial derivatives with respect to the generalized coordinates associated with the J marker are negative of the above expressions.

A.2.3 Initial Condition Analysis

$$\left(\frac{\partial}{\partial \mathbf{q}} \left(\frac{\partial G}{\partial \dot{\mathbf{q}}} \right) \right) \dot{\mathbf{q}} = \begin{Bmatrix} \frac{\partial}{\partial X_{i_r}} \left(\frac{\partial G_1}{\partial X_{i_r}} \dot{X}_{i_r} + \dots + \frac{\partial G_1}{\partial \Psi_{j_r}} \dot{\Psi}_{j_r} \right) \dot{X}_{i_r} + \dots + \frac{\partial}{\partial \Psi_{j_r}} \left(\frac{\partial G_1}{\partial X_{i_r}} \dot{X}_{i_r} + \dots + \frac{\partial G_1}{\partial \Psi_{j_r}} \dot{\Psi}_{j_r} \right) \dot{\Psi}_{j_r} \\ \vdots \\ \frac{\partial}{\partial X_{i_r}} \left(\frac{\partial G_{n_c}}{\partial X_{i_r}} \dot{X}_{i_r} + \dots + \frac{\partial G_{n_c}}{\partial \Psi_{j_r}} \dot{\Psi}_{j_r} \right) \dot{X}_{i_r} + \dots + \frac{\partial}{\partial \Psi_{j_r}} \left(\frac{\partial G_{n_c}}{\partial X_{i_r}} \dot{X}_{i_r} + \dots + \frac{\partial G_{n_c}}{\partial \Psi_{j_r}} \dot{\Psi}_{j_r} \right) \dot{\Psi}_{j_r} \end{Bmatrix}. \quad (\text{A.43})$$

This expression is needed in the initial acceleration evaluation. This expression also can be represented by the following matrix multiplication. For each row (i.e., for each constraint),

$$\begin{aligned} & \frac{\partial}{\partial X_{i_r}} \left(\frac{\partial G_{i_c}}{\partial X_{i_r}} \dot{X}_{i_r} + \dots + \frac{\partial G_{i_c}}{\partial \Psi_{j_r}} \dot{\Psi}_{j_r} \right) \dot{X}_{i_r} + \dots + \frac{\partial}{\partial \Psi_{j_r}} \left(\frac{\partial G_{i_c}}{\partial X_{i_r}} \dot{X}_{i_r} + \dots + \frac{\partial G_{i_c}}{\partial \Psi_{j_r}} \dot{\Psi}_{j_r} \right) \dot{\Psi}_{j_r} \\ &= \left[\begin{bmatrix} \frac{\partial^2 G_{i_c}}{\partial X_{i_r}^2} & \dots & \frac{\partial^2 G_{i_c}}{\partial X_{i_r} \partial \Psi_{j_r}} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 G_{i_c}}{\partial \Psi_{j_r} \partial X_{i_r}} & \dots & \frac{\partial^2 G_{i_c}}{\partial \Psi_{j_r}^2} \end{bmatrix} \begin{Bmatrix} \dot{X}_{i_r} \\ \vdots \\ \dot{\Psi}_{j_r} \end{Bmatrix} \right]^T \begin{Bmatrix} \dot{X}_{i_r} \\ \vdots \\ \dot{\Psi}_{j_r} \end{Bmatrix}. \end{aligned} \quad (\text{A.44})$$

Consider the relationship between the time derivative of momentum and the acceleration.

The translational and rotational momenta have been defined as

$$\begin{Bmatrix} \dot{u}_x \\ \dot{u}_y \\ \dot{u}_z \end{Bmatrix}_{i_r} = m_{i_r} \begin{Bmatrix} \ddot{X} \\ \ddot{Y} \\ \ddot{Z} \end{Bmatrix}_{i_r}, \quad \begin{Bmatrix} u_\phi \\ u_\theta \\ u_\psi \end{Bmatrix}_{i_r} = [B]_{i_r}^T [J]_{i_r} [B]_{i_r} \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i_r}. \quad (\text{A.45})$$

Differentiating the momenta gives

$$\begin{Bmatrix} \dot{u}_x \\ \dot{u}_y \\ \dot{u}_z \end{Bmatrix}_{i,} = m_{i,} \begin{Bmatrix} \ddot{X} \\ \ddot{Y} \\ \ddot{Z} \end{Bmatrix}_{i,} \quad (\text{A.46})$$

$$\begin{aligned} \begin{Bmatrix} \dot{u}_\Phi \\ \dot{u}_\Theta \\ \dot{u}_\Psi \end{Bmatrix}_{i,} &= \left(\frac{d}{dt} [B]_{i,}^T \right) [J]_{i,} [B]_{i,} \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i,} + [B]_{i,}^T [J]_{i,} \left(\frac{d}{dt} [B]_{i,} \right) \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i,} \\ &+ [B]_{i,}^T [J]_{i,} [B]_{i,} \begin{Bmatrix} \ddot{\Phi} \\ \ddot{\Theta} \\ \ddot{\Psi} \end{Bmatrix}_{i,}. \end{aligned} \quad (\text{A.47})$$

Here,

$$\begin{aligned} \left(\frac{d}{dt} [B]_{i,}^T \right) [J]_{i,} [B]_{i,} &= \\ \begin{bmatrix} \dot{\Theta} c \Theta s \Psi + \dot{\Psi} s \Theta c \Psi & \dot{\Theta} c \Theta c \Psi - \dot{\Psi} s \Theta s \Psi & -\dot{\Theta} s \Theta \\ -\dot{\Psi} s \Psi & -\dot{\Psi} c \Psi & 0 \\ 0 & 0 & 0 \end{bmatrix}_{i,} &\begin{bmatrix} J_{xx} s \Theta s \Psi & J_{xx} c \Psi & 0 \\ J_{yy} s \Theta c \Psi & -J_{yy} s \Psi & 0 \\ J_{zz} c \Theta & 0 & J_{zz} \end{bmatrix}_{i,} \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i,}, \end{aligned}$$

$$= \begin{bmatrix} J_x s \Theta s \Psi (\dot{\Theta} c \Theta s \Psi + \dot{\Psi} s \Theta c \Psi) & J_x c \Psi (\dot{\Theta} c \Theta s \Psi + \dot{\Psi} s \Theta c \Psi) & -J_z \dot{\Theta} s \Theta \\ J_y s \Theta c \Psi (\dot{\Theta} c \Theta c \Psi - \dot{\Psi} s \Theta s \Psi) & -J_y s \Psi (\dot{\Theta} c \Theta c \Psi - \dot{\Psi} s \Theta s \Psi) & 0 \\ J_z c \Theta (-\dot{\Theta} s \Theta) & 0 & 0 \end{bmatrix}_{i,} \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i,}$$

(A.48)

$$[B]_{i,}^T [J]_{i,} \left(\frac{d}{dt} [B]_{i,} \right) \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i,} =$$

$$= \begin{bmatrix} s \Theta s \Psi & s \Theta c \Psi & c \Theta \\ c \Psi & -s \Psi & 0 \\ 0 & 0 & 1 \end{bmatrix}_{i,} \begin{bmatrix} J_x (\dot{\Theta} c \Theta s \Psi + \dot{\Psi} s \Theta c \Psi) & -J_x \dot{\Psi} s \Psi & 0 \\ J_y (\dot{\Theta} c \Theta c \Psi - \dot{\Psi} s \Theta s \Psi) & -J_y \dot{\Psi} c \Psi & 0 \\ -J_z \dot{\Theta} s \Theta & 0 & 0 \end{bmatrix}_{i,} \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i,}$$

$$= \begin{bmatrix} J_x s \Theta s \Psi (\dot{\Theta} c \Theta s \Psi + \dot{\Psi} s \Theta c \Psi) & J_x \dot{\Psi} s \Theta s^2 \Psi & 0 \\ J_y s \Theta c \Psi (\dot{\Theta} c \Theta c \Psi - \dot{\Psi} s \Theta s \Psi) & -J_y \dot{\Psi} s \Theta c^2 \Psi & 0 \\ J_z c \Theta (-\dot{\Theta} s \Theta) & 0 & 0 \end{bmatrix}_{i,} \begin{Bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{Bmatrix}_{i,}.$$

(A.49)

Here, it is noted that

$$\left(\frac{d}{dt}[B]_i^T\right)[J]_i,[B]_i = \left[[B]_i^T[J]_i,\left(\frac{d}{dt}[B]_i\right)\right]^T. \quad (\text{A.50})$$

APPENDIX B

PROGRAMS

This appendix contains the messages defined in the classes of OODS.

B.1 System Class

```
Object subclass: #System
  instanceVariableNames: 'modelName '
  classVariableNames: 'DamperDictionary EndTime ForceDictionary GearCoeffArray
    GearConstant Gravity InitialTime IntegrationOrder JointDictionary Lambda MatrixdGdq MatrixdPdq
    MaxIntegrationOrder NoOfSteps NumberOfConstraints NumberOfForces NumberOfParts
    NumberOfRequests NumberOfSprings PartDictionary RequestDictionary SpringDictionary Stepsize '
  poolDictionaries: ''
  category: 'Multibody System'

modelName
  "returns the name of the simulation model."
  ^modelName

setErrorColumn
  "returns the error column on the right-hand side of the system equations;i.e, {E} = - [M P G L]t. "
  | q c arrayM arrayP arrayG arrayL arrayE |
  q _ 6*NumberOfParts. c _ NumberOfConstraints.
  arrayM _ self systemEOMFunction. arrayP _ self systemMomentumFunction.
  (c ~= 0) ifTrue: [arrayG _ self systemConstraintFunction].
  arrayL _ self systemForceFunction. arrayE _ Array new: (q+q+c+q).
  self put: arrayM into: arrayE at: (1@1). self put: arrayP into: arrayE at: (1@(q+1)).
  (c ~= 0) ifTrue: [self put: arrayG into: arrayE at: (1@(q+q+1))].
  self put: arrayL into: arrayE at: (1@(q+q+c+1)).
  arrayE _ self multiply: arrayE by: (-1).
  ^ arrayE

setSubJacobian11
  "returns the submatrix [dM/dq] - (1/h*beta)[dM/dqdot]. Note that [dM/dq] = - [d(dT/dq)/dq] -
  <Lambda>[d(dG/dq)/dq]."
  | q c matrix1 matrix2 matrix3 sub11 |
  q _ 6*NumberOfParts.
  "----- For - <Lambda>[d(dG/dq)/dq] -----"
  (NumberOfConstraints ~= 0) ifTrue: [
    matrix1 _ (Array new: q) collect: [: each | Array new: q withAll: 0].
    c _ 0.
    1 to: (JointDictionary size) do: [: whichJoint | | aJoint aMarkerI aPartI aMarkerJ aPartJ dumMatrix aMatrix sub1
    sub2 sub3 sub4 sub5 partitionI partitionJ partitionIJ partitionJI whereI whereJ dum |
      aJoint _ JointDictionary at: whichJoint.
      ((aJoint type) = 'spherical') ifTrue: [
        sub1 _ self multiply: (aJoint sub11AtPoint1) by: (Lambda at: (c+1)).
        sub2 _ self multiply: (aJoint sub11AtPoint2) by: (Lambda at: (c+2)).
        sub3 _ self multiply: (aJoint sub11AtPoint3) by: (Lambda at: (c+3)).
        dumMatrix _ self add: sub1 with: sub2. aMatrix _ self add: dumMatrix with: sub3. c _ c + 3].
      ((aJoint type) = 'universal') ifTrue: [
        sub1 _ self multiply: (aJoint sub11AtPoint1) by: (Lambda at: (c+1)).
        sub2 _ self multiply: (aJoint sub11AtPoint2) by: (Lambda at: (c+2)).
        sub3 _ self multiply: (aJoint sub11AtPoint3) by: (Lambda at: (c+3)).
        sub4 _ self multiply: (aJoint sub11Perpendicular3) by: (Lambda at: (c+4)).
        dumMatrix _ self add: sub1 with: sub2. dumMatrix _ self add: dumMatrix with: sub3.
```

```

aMatrix _ self add: dumMatrix with: sub4. c _ c + 4].
((aJoint type) = 'revolute') ifTrue: [
    sub1 _ self multiply: (aJoint sub11AtPoint1) by: (Lambda at: (c+1)).
    sub2 _ self multiply: (aJoint sub11AtPoint2) by: (Lambda at: (c+2)).
    sub3 _ self multiply: (aJoint sub11AtPoint3) by: (Lambda at: (c+3)).
    sub4 _ self multiply: (aJoint sub11Perpendicular1) by: (Lambda at: (c+4)).
    sub5 _ self multiply: (aJoint sub11Perpendicular2) by: (Lambda at: (c+5)).
    dumMatrix _ self add: sub1 with: sub2. dumMatrix _ self add: dumMatrix with: sub3.
    dumMatrix _ self add: dumMatrix with: sub4. aMatrix _ self add: dumMatrix with: sub5. c _ c + 5].
((aJoint type) = 'translational') ifTrue: [
    sub1 _ self multiply: (aJoint sub11InPlane1) by: (Lambda at: (c+1)).
    sub2 _ self multiply: (aJoint sub11InPlane2) by: (Lambda at: (c+2)).
    sub3 _ self multiply: (aJoint sub11Perpendicular1) by: (Lambda at: (c+3)).
    sub4 _ self multiply: (aJoint sub11Perpendicular2) by: (Lambda at: (c+4)).
    sub5 _ self multiply: (aJoint sub11Perpendicular4) by: (Lambda at: (c+5)).
    dumMatrix _ self add: sub1 with: sub2. dumMatrix _ self add: dumMatrix with: sub3.
    dumMatrix _ self add: dumMatrix with: sub4. aMatrix _ self add: dumMatrix with: sub5. c _ c + 5].
((aJoint type) = 'cylindrical') ifTrue: [
    sub1 _ self multiply: (aJoint sub11InPlane1) by: (Lambda at: (c+1)).
    sub2 _ self multiply: (aJoint sub11InPlane2) by: (Lambda at: (c+2)).
    sub3 _ self multiply: (aJoint sub11Perpendicular1) by: (Lambda at: (c+3)).
    sub4 _ self multiply: (aJoint sub11Perpendicular2) by: (Lambda at: (c+4)).
    dumMatrix _ self add: sub1 with: sub2. dumMatrix _ self add: dumMatrix with: sub3.
    aMatrix _ self add: dumMatrix with: sub4. c _ c + 4].
((aJoint type) = 'planar') ifTrue: [
    sub1 _ self multiply: (aJoint sub11InPlane3) by: (Lambda at: (c+1)).
    sub2 _ self multiply: (aJoint sub11Perpendicular1) by: (Lambda at: (c+2)).
    sub3 _ self multiply: (aJoint sub11Perpendicular2) by: (Lambda at: (c+3)).
    dumMatrix _ self add: sub1 with: sub2. aMatrix _ self add: dumMatrix with: sub3. c _ c + 3].
"Here, aMatrix has the size 12 by 12 (symmetric matrix)."
aMatrix _ self multiply: aMatrix by: (-1).
partitionI _ self partition: aMatrix from: (1@1) extension: (6@6).
partitionIJ _ self partition: aMatrix from: (7@1) extension: (6@6).
partitionJ _ self partition: aMatrix from: (7@7) extension: (6@6).
partitionJI _ self partition: aMatrix from: (1@7) extension: (6@6).
aMarkerI _ aJoint markerI. aMarkerJ _ aJoint markerJ.
aPartI _ aMarkerI part. aPartJ _ aMarkerJ part.
whereI _ 6*(((aPartI partNo) asNumber) - 1). whereJ _ 6*(((aPartJ partNo) asNumber) - 1).
dum _ (Array new: q) collect: [: each | Array new: q withAll: 0].
(((aPartI partNo) ~= '00') & ((aPartJ partNo) ~= '00'))
ifTrue: [
    self put: partitionI into: dum at: ((whereI+1)@(whereI+1)).
    self put: partitionIJ into: dum at: ((whereJ+1)@(whereI+1)).
    self put: partitionJ into: dum at: ((whereJ+1)@(whereJ+1)).
    self put: partitionJI into: dum at: ((whereI+1)@(whereJ+1))]
ifFalse: [
    ((aPartI partNo) ~= '00') ifTrue: [ self put: partitionI into: dum at: ((whereI+1)@(whereI+1))].
    ((aPartJ partNo) ~= '00') ifTrue: [ self put: partitionJ into: dum at: ((whereJ+1)@(whereJ+1))].
matrix1 _ self add: matrix1 with: dum]].
"----- For -[d(dT/dq)/dq] and -(1/h*beta)[dM/dqdot] -----"
matrix2 _ (Array new: q) collect: [: each | Array new: q withAll: 0].
matrix3 _ (Array new: q) collect: [: each | Array new: q withAll: 0].
1 to: NumberOfParts do: [: whichPart | aPart where |
    aPart _ PartDictionary at: whichPart. where _ 6*(((aPart partNo) asNumber) - 1).
    self put: (aPart sub11Inertia1) into: matrix2 at: ((where+1)@(where+1)).
    self put: (aPart sub11Inertia2) into: matrix3 at: ((where+1)@(where+1))].
MatrixdPdQ _ self transpose: matrix3. matrix3 _ self multiply: matrix3 by: (-1/(Stepsize*GearConstant)).
(NumberOfConstraints ~= 0)
ifTrue: [ sub11 _ self add: (self add: matrix1 with: matrix2) with: matrix3]
ifFalse: [ sub11 _ self add: matrix2 with: matrix3].
^ sub11

```

setSubJacobian12

```
"returns the submatrix  $-(1/h*\beta)[dM/d\dot{u}] = -1/h*\beta[I]$ ."
| q matrixI sub12 |
q _ 6*NumberOfParts. matrixI _ self identityMatrix: q.
sub12 _ self multiply: matrixI by: (-1/(Stepsize*GearConstant)).
^ sub12
```

setSubJacobian21

```
"returns the submatrix  $[dP/dq] - (1/h*\beta)[dP/d\dot{q}] = -d([M]\{\dot{q}\})/dq - (1/h*\beta)[-M]$ .
Note that  $[dP/dq] = [dM/d\dot{q}]$ . This term is represented by the class variable 'MatrixdPdq'."
| q dum sub21 |
q _ 6*NumberOfParts. dum _ (Array new: q) collect: [: each | Array new: q withAll: 0].
1 to: NumberOfParts do: [: whichPart | | aPart where matrixMi matrix |
  aPart _ PartDictionary at: whichPart. where _ 6*(((aPart partNo) asNumber) - 1).
  matrixMi _ aPart toMomentum. matrix _ self multiply: matrixMi by: (1/(Stepsize*GearConstant)).
  self put: matrix into: dum at: ((where+1)@(where+1)).
sub21 _ self add: MatrixdPdq with: dum.
^ sub21
```

setSubJacobian31

```
"returns the submatrix  $[dG/dq]$  where G represents constraints."
| q c sub31 y |
q _ 6*NumberOfParts. c _ NumberOfConstraints.
sub31 _ (Array new: c) collect: [: each | Array new: q withAll: 0].
y _ 1.
1 to: (JointDictionary size) do: [: whichJoint |
  | aJoint arrayOfaJoint aMarkerI aPartI aMarkerJ aPartJ partitionI partitionJ whereI whereJ |
  aJoint _ JointDictionary at: whichJoint.
  ((aJoint type) = 'spherical') ifTrue: [ | a1 a2 a3 |
    a1 _ aJoint sub31AtPoint1. a2 _ aJoint sub31AtPoint2. a3 _ aJoint sub31AtPoint3. arrayOfaJoint _
    (a1,a2),a3].
    ((aJoint type) = 'universal') ifTrue: [ | a1 a2 a3 a4 |
      a1 _ aJoint sub31AtPoint1. a2 _ aJoint sub31AtPoint2. a3 _ aJoint sub31AtPoint3.
      a4 _ aJoint sub31Perpendicular3. arrayOfaJoint _ ((a1,a2),a3),a4].
    ((aJoint type) = 'revolute') ifTrue: [ | a1 a2 a3 a4 a5 |
      a1 _ aJoint sub31AtPoint1. a2 _ aJoint sub31AtPoint2. a3 _ aJoint sub31AtPoint3.
      a4 _ aJoint sub31Perpendicular1. a5 _ aJoint sub31Perpendicular2. arrayOfaJoint _ (((a1,a2),a3),a4),a5].
    ((aJoint type) = 'translational') ifTrue: [ | a1 a2 a3 a4 a5 |
      a1 _ aJoint sub31InPlane1. a2 _ aJoint sub31InPlane2. a3 _ aJoint sub31Perpendicular1.
      a4 _ aJoint sub31Perpendicular2. a5 _ aJoint sub31Perpendicular4. arrayOfaJoint _ (((a1,a2),a3),a4),a5].
    ((aJoint type) = 'cylindrical') ifTrue: [ | a1 a2 a3 a4 |
      a1 _ aJoint sub31InPlane1. a2 _ aJoint sub31InPlane2. a3 _ aJoint sub31Perpendicular1.
      a4 _ aJoint sub31Perpendicular2. arrayOfaJoint _ ((a1,a2),a3),a4].
    ((aJoint type) = 'planar') ifTrue: [ | a1 a2 a3 |
      a1 _ aJoint sub31InPlane3. a2 _ aJoint sub31Perpendicular1. a3 _ aJoint sub31Perpendicular2.
      arrayOfaJoint _ (a1,a2),a3].
    "Here, arrayOfaJoint has the size 12 by (3, 4, or 5)."
    partitionI _ self partition: arrayOfaJoint from: (1@1) extension: (6@(arrayOfaJoint size)).
    partitionJ _ self partition: arrayOfaJoint from: (7@1) extension: (6@(arrayOfaJoint size)).
    aMarkerI _ aJoint markerI. aMarkerJ _ aJoint markerJ.
    aPartI _ aMarkerI part. aPartJ _ aMarkerJ part.
    whereI _ 6*(((aPartI partNo) asNumber) - 1). whereJ _ 6*(((aPartJ partNo) asNumber) - 1).
    (((aPartI partNo) == '00') & ((aPartJ partNo) == '00'))
    ifTrue: [ self put: partitionI into: sub31 at: ((whereI+1)@y). self put: partitionJ into: sub31 at:
    ((whereJ+1)@y)]
    ifFalse: [ ((aPartI partNo) == '00') ifTrue: [ self put: partitionI into: sub31 at: ((whereI+1)@y)].
      ((aPartJ partNo) == '00') ifTrue: [ self put: partitionJ into: sub31 at: ((whereJ+1)@y)].
    y _ y + (arrayOfaJoint size)].
MatrixdGdq _ sub31.
^ sub31
```

setSubJacobian41

"returns the Jacobian submatrix [dL/dq] where $\{L(Q,q,\dot{q})\} = \{Q\} - \{F(q,\dot{q})\}$."

! q sub41 |

q _ 6*NumberOfParts.

sub41 _ (Array new: q) collect: [: each | Array new: q withAll: 0].

"----- Springs -----"

(SpringDictionary size) ~= 0 ifTrue: [| matrixSpring dum |

matrixSpring _ (Array new: q) collect: [: each | Array new: q withAll: 0].

dum _ (Array new: q) collect: [: each | Array new: q].

1 to: NumberOfSprings do: [: whichSpring | | aSpring partI partJ partNoI partNoJ whereI whereJ matrixIJ matrixI matrixJ |

aSpring _ SpringDictionary at: whichSpring.

partI _ (aSpring markerI) part. partJ _ (aSpring markerJ) part. partNoI _ partI partNo. partNoJ _ partJ partNo.

whereI _ 6*((partNoI asNumber) - 1). whereJ _ 6*((partNoJ asNumber) - 1). matrixIJ _ aSpring sub41Spring.

matrixI _ self partition: matrixIJ from: (1@1) extension: (6@6).

matrixJ _ self partition: matrixIJ from: (7@1) extension: (6@6).

((partNoI ~= '00') & (partNoJ ~= '00')) ifTrue: [

self put: matrixI into: dum at: ((whereI+1)@(whereI+1)).

self put: matrixJ into: dum at: ((whereJ+1)@(whereJ+1)).

((partNoI = '00') & (partNoJ ~= '00')) ifTrue: [self put: matrixJ into: dum at: ((whereJ+1)@(whereJ+1)).

((partNoI ~= '00') & (partNoJ = '00')) ifTrue: [self put: matrixI into: dum at: ((whereI+1)@(whereI+1)).

matrixSpring _ self add: matrixSpring with: dum].

sub41 _ matrixSpring].

"----- Dampers -----"

(DamperDictionary size) ~= 0 ifTrue: [| matrixDamper dum |

matrixDamper _ (Array new: q) collect: [: each | Array new: q withAll: 0].

dum _ (Array new: q) collect: [: each | Array new: q].

1 to: (DamperDictionary size) do: [: whichDamper | | aDamper partI partJ partNoI partNoJ whereI whereJ matrixIJ matrixI matrixJ |

aDamper _ DamperDictionary at: whichDamper.

partI _ (aDamper markerI) part. partJ _ (aDamper markerJ) part. partNoI _ partI partNo. partNoJ _ partJ partNo.

whereI _ 6*((partNoI asNumber) - 1). whereJ _ 6*((partNoJ asNumber) - 1). matrixIJ _ aDamper sub41Damper.

matrixI _ self partition: matrixIJ from: (1@1) extension: (6@6).

matrixJ _ self partition: matrixIJ from: (7@1) extension: (6@6).

((partNoI ~= '00') & (partNoJ ~= '00')) ifTrue: [

self put: matrixI into: dum at: ((whereI+1)@(whereI+1)).

self put: matrixJ into: dum at: ((whereJ+1)@(whereJ+1)).

((partNoI = '00') & (partNoJ ~= '00')) ifTrue: [self put: matrixJ into: dum at: ((whereJ+1)@(whereJ+1)).

((partNoI ~= '00') & (partNoJ = '00')) ifTrue: [self put: matrixI into: dum at: ((whereI+1)@(whereI+1)).

matrixDamper _ self add: matrixDamper with: dum].

sub41 _ self add: sub41 with: matrixDamper].

"----- Forces and Torques -----"

(ForceDictionary size) ~= 0 ifTrue: [| matrixForce dum |

matrixForce _ (Array new: q) collect: [: each | Array new: q withAll: 0].

dum _ (Array new: q) collect: [: each | Array new: q].

1 to: (ForceDictionary size) do: [: whichForce | | aForce partI partJ partNoI partNoJ whereI whereJ aMatrix matrixI matrixJ |

aForce _ ForceDictionary at: whichForce.

partI _ (aForce actionMarker) part. partJ _ (aForce reactionMarker) part.

partNoI _ partI partNo. partNoJ _ partJ partNo.

whereI _ 6*((partNoI asNumber) - 1). whereJ _ 6*((partNoJ asNumber) - 1).

(aForce type) = 'force' ifTrue: [aMatrix _ aForce sub41Force].

(aForce type) = 'torque' ifTrue: [aMatrix _ aForce sub41Torque].

((aForce reactionMarker) ~= nil) ifTrue: [

matrixI _ self partition: aMatrix from: (1@1) extension: (6@6).

matrixJ _ self partition: aMatrix from: (7@1) extension: (6@6).

self put: matrixI into: dum at: ((whereI+1)@(whereI+1)).

self put: matrixJ into: dum at: ((whereJ+1)@(whereJ+1)).

((aForce reactionMarker) = nil) ifTrue: [self put: aMatrix into: dum at: ((whereI+1)@(whereI+1)).

matrixForce _ self add: matrixForce with: dum].

sub41 _ self add: sub41 with: matrixForce].

^ sub41

setSubJacobian42

"no damper"

| q sub42 |

q _ 6*NumberOfParts. sub42 _ (Array new: q) collect: [: each | Array new: q withAll: 0].

^ sub42

setSystemJacobian

"constructs the system Jacobian by stacking the submatrices for the constraint equations, generalized speed definition, equations of motion, and generalized forces."

| n q c dumJ matrixI sub11 sub12 sub13 sub14 sub21 sub22 sub31 sub41 sub42 sub44 systemJacobian |
NumberOfParts _ PartDictionary size.

q _ 6*NumberOfParts. c _ NumberOfConstraints. n _ (3*q) + c.

dumJ _ (Array new: n) collect: [: each | Array new: n withAll: 0].

matrixI _ self identityMatrix: q.

sub11 _ self setSubJacobian11. sub12 _ self setSubJacobian12. sub21 _ self setSubJacobian21.

(c ~= 0) ifTrue: [sub31 _ self setSubJacobian31].

sub41 _ self setSubJacobian41. sub42 _ self setSubJacobian42.

(c ~= 0) ifTrue: [sub13 _ self multiply: (self transpose: sub31) by: (-1)].

sub14 _ self multiply: matrixI by: (-1). sub22 _ matrixI. sub44 _ matrixI.

self put: sub11 into: dumJ at: (1@1). self put: sub12 into: dumJ at: ((q+1)@1).

(c ~= 0) ifTrue: [self put: sub13 into: dumJ at: ((q+q+1)@1)].

self put: sub14 into: dumJ at: ((q+q+c+1)@1). self put: sub21 into: dumJ at: (1@(q+1)).

self put: sub22 into: dumJ at: ((q+1)@(q+1)).

(c ~= 0) ifTrue: [self put: sub31 into: dumJ at: (1@ (q+q+1))].

self put: sub41 into: dumJ at: (1 @ (q+q+c+1)). self put: sub42 into: dumJ at: ((q+1) @ (q+q+c+1)).

systemJacobian _ self put: sub44 into: dumJ at: ((q+q+c+1) @ (q+q+c+1)).

^ systemJacobian

systemConstraintFunction

"returns the constraints G of the system."

| arrayG y arraySpherical arrayUniversal arrayRevolute arrayTranslational arrayCylindrical arrayPlanar |

arrayG _ Array new: NumberOfConstraints.

arraySpherical _ Array new: 3. arrayUniversal _ Array new: 4. arrayRevolute _ Array new: 5.

arrayTranslational _ Array new: 5. arrayCylindrical _ Array new: 4. arrayPlanar _ Array new: 3.

y _ 1.

1 to: (JointDictionary size) do: [: whichJoint | | aJoint anArray |

aJoint _ JointDictionary at: whichJoint.

((aJoint type) = 'spherical') ifTrue: [| a1 a2 a3 |

a1 _ aJoint atPoint1. a2 _ aJoint atPoint2. a3 _ aJoint atPoint3.

arraySpherical at: 1 put: a1; at: 2 put: a2; at: 3 put: a3. anArray _ arraySpherical].

((aJoint type) = 'universal') ifTrue: [| a1 a2 a3 a4 |

a1 _ aJoint atPoint1. a2 _ aJoint atPoint2. a3 _ aJoint atPoint3. a4 _ aJoint perpendicular3.

arrayUniversal at: 1 put: a1; at: 2 put: a2; at: 3 put: a3; at: 4 put: a4. anArray _ arrayUniversal].

((aJoint type) = 'revolute') ifTrue: [| a1 a2 a3 a4 a5 |

a1 _ aJoint atPoint1. a2 _ aJoint atPoint2. a3 _ aJoint atPoint3. a4 _ aJoint perpendicular1.

a5 _ aJoint perpendicular2.

arrayRevolute at: 1 put: a1; at: 2 put: a2; at: 3 put: a3; at: 4 put: a4; at: 5 put: a5. anArray _ arrayRevolute].

((aJoint type) = 'translational') ifTrue: [| a1 a2 a3 a4 a5 |

a1 _ aJoint inPlane1. a2 _ aJoint inPlane2. a3 _ aJoint perpendicular1. a4 _ aJoint perpendicular2.

a5 _ aJoint perpendicular4.

arrayTranslational at: 1 put: a1; at: 2 put: a2; at: 3 put: a3; at: 4 put: a4; at: 5 put: a5.

anArray _ arrayTranslational].

((aJoint type) = 'cylindrical') ifTrue: [| a1 a2 a3 a4 |

a1 _ aJoint inPlane1. a2 _ aJoint inPlane2. a3 _ aJoint perpendicular1. a4 _ aJoint perpendicular2.

arrayCylindrical at: 1 put: a1; at: 2 put: a2; at: 3 put: a3; at: 4 put: a4. anArray _ arrayCylindrical].

((aJoint type) = 'planar') ifTrue: [| a1 a2 a3 |

a1 _ aJoint inPlane3. a2 _ aJoint perpendicular1. a3 _ aJoint perpendicular2.

arrayPlanar at: 1 put: a1; at: 2 put: a2; at: 3 put: a3. anArray _ arrayPlanar].

self put: anArray into: arrayG at: (1@y). y _ y + (anArray size)].

^ arrayG

systemEOMFunction

"returns the submatrix {M} of the error column matrix {E}; i.e., $\{M\} = \text{udot} - [dT/dq]t - \{Q\} - [dG/dq]t\{\text{Lambda}\} = \{0\}$."

| q arrayM array1 array2 |

q _ 6*NumberOfParts. array1 _ Array new: q.

1 to: NumberOfParts do: [: whichPart || aPart where array1 |

aPart _ PartDictionary at: whichPart. where _ 6*(((aPart partNo) asNumber) - 1).

array _ aPart partEOMFunction. self put: array into: array1 at: (1@(where+1)).

arrayM _ array1.

(NumberOfConstraints ~= 0) ifTrue: [

array2 _ self multiply: (self transpose: MatrixdGdq) with: Lambda. arrayM _ self subtract: arrayM with: array2].

^ arrayM

systemForceFunction

"returns the submatrix {L} of the system error column; i.e., $\{L(Q,q,\dot{q})\} = \{Q\} - \{F(q,\dot{q})\}$."

| q arrayQ arraySystemGenForces arrayL |

q _ 6*NumberOfParts. arrayQ _ Array new: q.

1 to: NumberOfParts do: [: whichPart || aPart where arrayQi |

aPart _ PartDictionary at: whichPart. where _ 6*(((aPart partNo) asNumber) - 1).

arrayQi _ aPart stateVariableQ. self put: arrayQi into: arrayQ at: (1@(where+1)).

arraySystemGenForces _ self systemGenForces. arrayL _ self subtract: arrayQ with: arraySystemGenForces.

^ arrayL

systemMomentumFunction

"returns the submatrix {P} of the error column; i.e., $\{P\} = \{P(u,q,\dot{q})\} = \{u\} - [M]\{\dot{q}\}$ "

| q arrayP |

q _ 6*NumberOfParts. arrayP _ Array new: q.

1 to: NumberOfParts do: [: whichPart || aPart where arrayPi |

aPart _ PartDictionary at: whichPart. where _ 6*(((aPart partNo) asNumber) - 1).

arrayPi _ aPart partMomentumFunction. self put: arrayPi into: arrayP at: (1@(where+1)).

^ arrayP

InitializeSystem: aString

"Initializes a dynamic system before the input data is read in.

The unit of the gravity is (m/sec*sec) and the gravity always acts in the negative Z direction of the ground frame."

modelName _ aString.

Gravity _ -9.8 asGoodDouble. NumberOfParts _ 0. NumberOfConstraints _ 0.

NumberOfSprings _ 0. NumberOfForces _ 0. NumberOfRequests _ 0.

PartDictionary _ OrderedCollection new. JointDictionary _ OrderedCollection new.

ForceDictionary _ OrderedCollection new. SpringDictionary _ OrderedCollection new.

DamperDictionary _ OrderedCollection new. RequestDictionary _ OrderedCollection new.

setInitialTime: t0 endTime: tn stepSize: h maxIntegrationOrder: k

"sets the time, stepsize, and order for numerical integration."

Stepsize _ h asGoodDouble. InitialTime _ t0. EndTime _ tn. NoOfSteps _ ((tn - t0)/h) asInteger.

MaxIntegrationOrder _ k.

add: matrixA with: matrixB

"Returns the result of the matrix addition; i.e., $\{C\} = \{A\} + \{B\}$; (l by m) = (l by m) + (l by m).

It is assumed that each matrix be created by the following syntax;

(Array new: noOfRow) collect: [: each | Array new: noOfColumn]."

lA mA lB mB sum matrixC |

lA _ matrixA size.

(matrixA at: 1) isSequenceable ifTrue: [mA _ (matrixA at:1) size] ifFalse: [mA _ 1].

lB _ matrixB size.

(matrixB at: 1) isSequenceable ifTrue: [mB _ (matrixB at:1) size] ifFalse: [mB _ 1].

matrixC _ (Array new: lA) collect: [: each | Array new: mA].

(lA = lB) & (mA = mB)

ifTrue: [

"----- (l by 1) + (l by 1) -----"

(mA = 1) ifTrue: [(1 to: lA) do: [: i | sum _ (matrixA at: i) + (matrixB at: i). matrixC at: i put: sum].

^ matrixC].

"----- (l by m) + (l by m) -----"

```
(1 to: lA) do: [: i |
  (1 to: mA) do: [: j | sum _ ((matrixA at: i) at: j) + ((matrixB at: i) at: j). (matrixC at: i) at: j put: sum]].
  ^ matrixC]
ifFalse: [ ^ 'The matrices are nonconformable for matrix addition.']
```

identityMatrix: aSize

"Returns the identity matrix with the size of aSize."

| matrixI |

matrixI _ (Array new: aSize) collect: [: each | Array new: aSize withAll: 0].

(1 to: aSize) do: [: i | (matrixI at: i) at: i put: 1].

^ matrixI

inverse: matrixA

"Returns the result of the matrix inversion using the Gauss-Jordan method.

This algorithm is based on the one in the Chapra's book (Chapter 8); i.e, $[A][I] = [I][B]$ "

| n augmentedMatrix matrixB |

n _ (matrixA at: 1) size.

augmentedMatrix _ (Array new: n) collect: [: each | Array new: (2*n)].

self put: matrixA into: augmentedMatrix at: (1@1).

self put: (self identityMatrix: n) into: augmentedMatrix at: ((n+1)@1).

1 to: n do: [: k | dum _ (augmentedMatrix at: k) at: k.

1 to: (2*n) do: [: j | (augmentedMatrix at: k) at: j put: (((augmentedMatrix at: k) at: j)/dum)].

1 to: n do: [: i |

(i ~= k) ifTrue: [dum _ (augmentedMatrix at: i) at: k.

1 to: (2*n) do: [: j |

(augmentedMatrix at: i) at: j put: (((augmentedMatrix at: i) at: j) - (dum*((augmentedMatrix at: k) at: j))))].

matrixB _ self partition: augmentedMatrix from: ((n+1)@1) extension: (n@n).

^ matrixB

inversePivot: matrixA

"Returns the result of the matrix inversion using the Gauss-Jordan method with the application of partial pivoting.

This algorithm is based on the one in the Chapra's book (Chapter 8); i.e, $[A][I] = [I][B]$

Note that the order of pivot rows must be considered when applied to the related matrix multiplication."

| n augmentedMatrix matrixB |

n _ (matrixA at: 1) size.

augmentedMatrix _ (Array new: n) collect: [: each | Array new: (2*n)].

self put: matrixA into: augmentedMatrix at: (1@1).

self put: (self identityMatrix: n) into: augmentedMatrix at: ((n+1)@1).

1 to: n do: [: k | dum _

self pivotForInverse: augmentedMatrix columnNo: k size: n.

dum _ (augmentedMatrix at: k) at: k.

1 to: (2*n) do: [: j | (augmentedMatrix at: k) at: j put: (((augmentedMatrix at: k) at: j)/dum)].

1 to: n do: [: i |

(i ~= k) ifTrue: [

dum _ (augmentedMatrix at: i) at: k.

1 to: (2*n) do: [: j |

(augmentedMatrix at: i) at: j put: (((augmentedMatrix at: i) at: j) - (dum*((augmentedMatrix at: k) at: j))))].

matrixB _ self partition: augmentedMatrix from: ((n+1)@1) extension: (n@n).

^ matrixB

multiply: matrixA by: aConstant

"Returns the result of the matrix multiplied by aConstant; i.e, $[C] = a[A]$

Note that whenever an element is 0 the corresponding multiplication is avoided to save computation time."

| m n matrixC |

m _ matrixA size.

(matrixA at: 1) isSequenceable

ifTrue: [

n _ (matrixA at: 1) size.

matrixC _ (Array new: m) collect: [: each | Array new: n withAll: 0].

1 to: m do: [: i |

1 to: n do: [: j | ai j |

```

    aij _ (matrixA at: i) at: j.
    aij = 0 ifFalse: [(matrixC at: i) at: j put: (aConstant*((matrixA at: i) at: j))]]]
ifFalse: [
    matrixC _ Array new: m withAll: 0.
    1 to: m do: [: i | | ai |
        ai _ matrixA at: i.
        ai = 0 ifFalse: [matrixC at: i put: (aConstant*(matrixA at: i))]].
^ matrixC

```

multiply: matrixA with: matrixB

"Returns the result of the matrix multiplication; i.e., [C] = [A][B]; (l by n) = (l by m)(m by n).

Note that whenever an element is 0 the corresponding multiplication is avoided to save computation time."

ll mA mB m n sum matrixC |

l _ matrixA size.

(matrixA at: 1) isSequenceable ifTrue: [mA _ (matrixA at:1) size] ifFalse: [mA _ 1].

mB _ matrixB size.

(matrixB at: 1) isSequenceable ifTrue: [n _ (matrixB at:1) size] ifFalse: [n _ 1].

(mA = mB)

ifTrue: [matrixC _ (Array new: 1) collect: [: each | Array new: n]. m _ mA.

"----- (1 by 1) = (1 by m)(m by 1) -----"

"Note that this case can be for a vector dot-product."

((l = 1) & (n = 1)) ifTrue: [sum _ 0.

(1 to: m) do: [: k |

((matrixA at: 1) at: k) = 0 | (matrixB at: k) = 0) ifFalse: [
 sum _ sum + (((matrixA at: 1) at: k)*(matrixB at: k))].

matrixC _ sum.

^ matrixC].

"----- (1 by n) = (1 by m)(m by n) -----"

(l = 1) ifTrue: [

(1 to: n) do: [: j |

sum _ 0.

(1 to: m) do: [: k |

((matrixA at: 1) at: k) = 0 | ((matrixB at: k) at: j) = 0)

ifFalse: [sum _ sum + (((matrixA at: 1) at: k)*(matrixB at: k) at: j))].

(matrixC at: 1) at: j put: sum].

^ matrixC].

"----- (l by 1) = (l by m)(m by 1) -----"

(n = 1) ifTrue: [

(1 to: l) do: [: i |

sum _ 0.

(1 to: m) do: [: k |

((matrixA at: i) at: k) = 0 | (matrixB at: k) = 0) ifFalse: [

sum _ sum + (((matrixA at: i) at: k)*(matrixB at: k))].

matrixC at: i put: sum].

^ matrixC].

"----- (l by n) = (l by m)(m by n) -----"

(1 to: l) do: [: i |

(1 to: n) do: [: j |

sum _ 0.

(1 to: m) do: [: k |

((matrixA at: i) at: k) = 0 | ((matrixB at: k) at: j) = 0)

ifFalse: [sum _ sum + (((matrixA at: i) at: k)*(matrixB at: k) at: j))].

(matrixC at: i) at: j put: sum]].

^ matrixC]

ifFalse: [

^ 'The matrices are nonconformable for matrix multiplication.'

partition: matrixA from: aPoint1 extension: aPoint2

"partitions and returns the partitioned matrix from the covered area by the arguments in matrixA."

l originY originX down across partMatrix |

originY _ aPoint1 y.

originX _ aPoint1 x.


```

down _ aPoint2 y.
across _ aPoint2 x.
(matrixA at: 1) isSequenceable
  ifFalse: [ partMatrix _ Array new: down.
    1 to: down do: [: i | partMatrix at: i put: (matrixA at: (originY+i-1))]]
  ifTrue: [
    (across = 1) ifTrue: [
      partMatrix _ Array new: down.
      1 to: down do: [: i | partMatrix at: i
        put: ((matrixA at: (originY+i-1)) at: originX)].
    (across ~= 1) ifTrue: [
      partMatrix _ (Array new: down) collect: [: each | Array new: across].
      1 to: down do: [: i |
        1 to: across do: [: j | (partMatrix at: i) at: j
          put: ((matrixA at: (originY+i-1)) at: (originX+j-1))]]].
    ^ partMatrix

pivotForInverse: augmentedMatrix columnNo: k size: n
"pivots the augmented matrix columnwise."
| big pivotI dum |
pivotI _ k.
big _ ((augmentedMatrix at: k) at: k) abs.
(k+1) to: n do: [: i | dum _ (( augmentedMatrix at: i) at: k) abs.
  (dum > big) ifTrue: [big _ dum. pivotI _ i]].
pivotI ~= k ifTrue: [
  1 to: (2*n) do: [: j | dum _ (augmentedMatrix at: pivotI) at: j.
    (augmentedMatrix at: pivotI) at: j put: ((augmentedMatrix at: k) at: j). (augmentedMatrix at: k) at: j put: dum]].
  ^ augmentedMatrix

put: aSubMatrix into: aMatrix at: aPoint
"puts aSubMatrix into aMatrix at aPoint and returns the new matrix.
Here, 'put' means that aMatrix is substituted by aSubMatrix in the corresponding partition."
| m n m1 n1 down across |
m _ aMatrix size.
(aMatrix at: 1) isSequenceable ifTrue: [ n _ (aMatrix at: 1) size] ifFalse: [ n _ 1].
m1 _ aSubMatrix size.
(aSubMatrix at: 1) isSequenceable ifTrue: [ n1 _ (aSubMatrix at: 1) size] ifFalse: [ n1 _ 1].
down _ aPoint y. across _ aPoint x.
((m1 + down - 1) > m | (n1 + across - 1) > n) ifTrue: [ ^ 'The main matrix can not accomodate the submatrix.'].
((n1 > 1) & (n > 1)) ifTrue: [
  1 to: m1 do: [: row | | x y |
    1 to: n1 do: [: column | y _ down + row - 1. x _ across + column - 1.
      (aMatrix at: y) at: x put: ((aSubMatrix at: row) at: column)]]].
((n1 = 1) & (n > 1)) ifTrue: [
  1 to: m1 do: [: row | | x y | y _ down + row - 1. x _ across. (aMatrix at: y) at: x put: (aSubMatrix at: row)]].
((n1 = 1) & (n = 1)) ifTrue: [
  1 to: m1 do: [: row | | y | y _ down + row - 1. aMatrix at: y put: (aSubMatrix at: row)]].
  ^ aMatrix

subtract: matrixA with: matrixB
"Returns the result of the matrix subtraction; i.e, [C] = [A] - [B]; (l by m) = (l by m) - (l by m).
It is assumed that each matrix be created by the following syntax;
(Array new: noOfRow) collect: [: each | Array new: noOfColumn]."
| lA mA lB mB diff matrixC |
lA _ matrixA size.
(matrixA at: 1) isSequenceable ifTrue: [mA _ (matrixA at:1) size] ifFalse: [mA _ 1].
lB _ matrixB size.
(matrixB at: 1) isSequenceable ifTrue: [mB _ (matrixB at:1) size] ifFalse: [mB _ 1].
matrixC _ (Array new: lA) collect: [: each | Array new: mA].
(lA = lB) & (mA = mB) ifTrue: [
  "----- (l by 1) - (l by 1) -----"
  (mA = 1) ifTrue: [(1 to: lA) do: [: i | diff _ (matrixA at: i) - (matrixB at: i). matrixC at: i put: diff]].

```

```

^ matrixC].
"----- (l by m) - (l by m) -----"
(1 to: lA) do: [: i |
  (1 to: mA) do: [: j | diff _ ((matrixA at: i) at: j) - ((matrixB at: i) at: j). (matrixC at: i) at: j put: diff]].
^ matrixC]
ifFalse: [
  ^ 'The matrices are nonconformable for matrix subtraction.']

```

transpose: aMatrix

"Returns the result of the matrix transpose. It is assumed that the matrix be created by the following syntax;
(Array new: noOfRow) collect: [: each | Array new: noOfColumn]."

```

| m transposedMatrix |
l _ aMatrix size.
(aMatrix at: 1) isSequenceable ifTrue: [m _ (aMatrix at: 1) size] ifFalse: [m _ 1].
transposedMatrix _ (Array new: m) collect: [: each | Array new: l].
(m = 1) ifTrue: [(1 to: l) do: [: i | (transposedMatrix at: 1) at: i put: (aMatrix at: i)]. ^ transposedMatrix].
(1 to: l) do: [: i | (1 to: m) do: [: j | (transposedMatrix at: j) at: i put: ((aMatrix at: i) at: j)]].
^ transposedMatrix

```

pivot: matrixA columnNo: j rowOrder: order size: n

"returns the partially pivoted matrix for columnJ. matrixA is a n by n matrix and matrixA1 is a (n+1) by n."

```

| max pivotI dum1 dum2 matrixA1 |
pivotI _ j. max _ ((matrixA at: j) at: j) abs.
(j+1) to: n do: [: i | ((matrixA at: i) at: j) abs > max ifTrue: [max _ ((matrixA at: i) at: j) abs. pivotI _ i]].
pivotI = j ifFalse: [
  1 to: n do: [: k |
    dum1 _ (matrixA at: j) at: k.
    (matrixA at: j) at: k put: ((matrixA at: pivotI) at: k). (matrixA at: pivotI) at: k put: dum1].
  dum2 _ order at: j. order at: j put: (order at: pivotI). order at: pivotI put: dum2].
matrixA1 _ Array new: (n+1).
1 to: n do: [: i | matrixA1 at: i put: (matrixA at: i)]. matrixA1 at: (n+1) put: order.
^ matrixA1

```

pivotLU: matrixA

"1.decompose L, 2.pivots the matrix columnwise, 3.decomposes U, and 4.continue steps 1-3 until LU is completed."

```

| n dum order sum matrixA1 |
n _ matrixA size. order _ Array new: n.
1 to: n do: [: i | order at: i put: i].
dum _ self pivot: matrixA columnNo: 1 rowOrder: order size: n.
1 to: n do: [: i | matrixA at: i put: (dum at: i)].
order _ dum at: (n+1).
2 to: n do: [: k | (matrixA at: 1) at: k put: (((matrixA at: 1) at: k)/((matrixA at: 1) at: 1))].
2 to: (n-1) do: [: j |
  j to: n do: [: i |
    sum _ 0.
    1 to: (j - 1) do: [: k | sum _ sum + (((matrixA at: i) at: k)*((matrixA at: k) at: j))].
    (matrixA at: i) at: j put: (((matrixA at: i) at: j) - sum)].
  dum _ self pivot: matrixA columnNo: j rowOrder: order size: n.
  1 to: n do: [: i | matrixA at: i put: (dum at: i)].
  order _ dum at: (n+1).
  (j + 1) to: n do: [: k |
    sum _ 0.
    1 to: (j - 1) do: [: i | sum _ sum + (((matrixA at: j) at: i)*((matrixA at: i) at: k))].
    (matrixA at: j) at: k put: (((matrixA at: j) at: k) - sum)/((matrixA at: j) at: j)].
  sum _ 0.
  1 to: (n-1) do: [: k | sum _ sum + (((matrixA at: n) at: k)*((matrixA at: k) at: n))].
  (matrixA at: n) at: n put: (((matrixA at: n) at: n) - sum).
  matrixA1 _ Array new: (n+1).
  1 to: n do: [: i | matrixA1 at: i put: (matrixA at: i)]. matrixA1 at: (n+1) put: order.
^ matrixA1

```

pivotLUSolve: matrixA with: arrayB

"[A]{X} = [L][U]{X} = [L]{X1} = {Y} where [U]{X} = {X1}; 1. solves {X1}, and 2. solves {X}. "

| n matrixA1 matrixLU order arrayX arrayY sum arrayX1 |

n _ matrixA size.

"-----[L]<Pivot>[U] -----"

matrixA1 _ self pivotLU: matrixA.

matrixLU _ self partition: matrixA1 from: (1@1) extension: (n@n).

order _ matrixA1 at: (n+1).

"----- Rearrange R.H.S -----"

arrayY _ Array new: n.

1 to: n do: [: i | j | j _ order at: i. arrayY at: i put: (arrayB at: j)].

"----- Solve for {X1} -----"

arrayX1 _ Array new: n withAll: 0.

arrayX1 at: 1 put: ((arrayY at: 1)/((matrixLU at: 1) at: 1)).

2 to: n do: [: i |

sum _ 0.

1 to: (i - 1) do: [: k | sum _ sum + (((matrixLU at: i) at: k)*(arrayX1 at: k))].

arrayX1 at: i put: (((arrayY at: i) - sum)/((matrixLU at: i) at: i)).

"----- Solve for {X} -----"

arrayX _ Array new: n withAll: 0.

arrayX at: n put: (arrayX1 at: n).

(n-1) to: 1 by: -1 do: [: j | sum _ 0.

(j+1) to: n do: [: k | sum _ sum + (((matrixLU at: j) at: k)*(arrayX at: k))].

arrayX at: j put: ((arrayX1 at: j) - sum)].

^ arrayX

scale: matrixA

"scales matrixA so that the largest element in each row is unity. matrixB contains the scaled matrix and scaleFactor."

| n scaledMatrix scaleFactor matrixB |

n _ matrixA size.

scaledMatrix _ (Array new: n) collect: [: each | Array new: n].

scaleFactor _ Array new: n.

1 to: n do: [: i | max _ max _ ((matrixA at: i) at: i) abs.

2 to: n do: [: j | dum _ dum _ ((matrixA at: i) at: j) abs. (dum > max) ifTrue: [max _ dum]].

scaleFactor at: i put: (1/max)].

1 to: n do: [: i |

1 to: n do: [: j | (scaledMatrix at: i) at: j put: (((matrixA at: i) at: j)*(scaleFactor at: i))].

matrixB _ Array new: (n+1).

1 to: n do: [: i | matrixB at: i put: (scaledMatrix at: i)].

matrixB at: (n+1) put: scaleFactor.

^ matrixB

scalePivotLUSolve: matrixA0 with: arrayC

"[A0]{X0} = [A]{X} = [L][U]{X} = [L]{X1} = {Y} where [U]{X} = {X1};

1. scale [A0] and {X0}, 2. solves {X1}, and 3. solves {X}. "

| n matrixB matrixA scaleFactor arrayB matrixA1 matrixLU order arrayX arrayY sum arrayX1 |

n _ matrixA0 size.

"----- Scale -----"

matrixB _ self scale: matrixA0.

matrixA _ self partition: matrixB from: (1@1) extension: (n@n).

scaleFactor _ matrixB at: (n+1).

arrayB _ Array new: n.

1 to: n do: [: i | arrayB at: i put: ((scaleFactor at: i)*(arrayC at: i))].

"-----[L]<Pivot>[U] -----"

matrixA1 _ self pivotLU: matrixA.

matrixLU _ self partition: matrixA1 from: (1@1) extension: (n@n).

order _ matrixA1 at: (n+1).

"----- Rearrange R.H.S -----"

arrayY _ Array new: n.

1 to: n do: [: i | j | j _ order at: i. arrayY at: i put: (arrayB at: j)].

"----- Solve for {X1} -----"

```

arrayX1 _ Array new: n withAll: 0.
arrayX1 at: 1 put: ((arrayY at: 1)/((matrixLU at: 1) at: 1)).
2 to: n do: [: i | sum _ 0.
    1 to: (i - 1) do: [: k | sum _ sum + (((matrixLU at: i) at: k)*(arrayX1 at: k))].
    arrayX1 at: i put: (((arrayY at: i) - sum)/((matrixLU at: i) at: i)).
"----- Solve for {X} -----"
arrayX _ Array new: n withAll: 0.
arrayX at: n put: (arrayX1 at: n).
(n-1) to: 1 by: -1 do: [: j | sum _ 0.
    (j+1) to: n do: [: k | sum _ sum + (((matrixLU at: j) at: k)*(arrayX at: k))].
    arrayX at: j put: ((arrayX1 at: j) - sum)].
^ arrayX

```

changeIntegrationOrder: k

"changes the order of the Gear's integration formula."

IntegrationOrder _ k. GearCoeffArray _ Array new: k.

(k = 1) ifTrue: [GearConstant _ -1. GearCoeffArray at: 1 put: 1].

(k = 2) ifTrue: [GearConstant _ -2/3. GearCoeffArray at: 1 put: (4/3). GearCoeffArray at: 2 put: (-1/3)].

(k = 3) ifTrue: [GearConstant _ -6/11. GearCoeffArray at: 1 put: (18/11). GearCoeffArray at: 2 put: (-9/11).

GearCoeffArray at: 3 put: (2/11)].

(k = 4) ifTrue: [GearConstant _ -12/25. GearCoeffArray at: 1 put: (48/25). GearCoeffArray at: 2 put: (-36/25).

GearCoeffArray at: 3 put: (16/25). GearCoeffArray at: 4 put: (-3/25)].

(k = 5) ifTrue: [GearConstant _ -60/137. GearCoeffArray at: 1 put: (300/137). GearCoeffArray at: 2 put: (-300/137).

GearCoeffArray at: 3 put: (200/137). GearCoeffArray at: 4 put: (-75/137). GearCoeffArray at: 5 put: (12/137)].

(k = 6) ifTrue: [GearConstant _ -60/147. GearCoeffArray at: 1 put: (360/147). GearCoeffArray at: 2 put: (-450/147).

GearCoeffArray at: 3 put: (400/147). GearCoeffArray at: 4 put: (-225/147). GearCoeffArray at: 5 put: (72/147).

GearCoeffArray at: 6 put: (-10/147)].

correctSystem

"formulates system equations and solves the system equations numerically; [J]{delX} = - {Y}."

| matrixJ arrayY maxIterations arrayX0 arraydelX arrayX1 |

arrayX0 _ self setUpStartingVector.

"----- Solve System Equations -----"

maxIterations _ 5.

1 to: maxIterations do: [: each |

matrixJ _ self setSystemJacobian. arrayY _ self setErrorColumn.

arraydelX _ self pivotLUSolve: matrixJ with: arrayY. arrayX1 _ self add: arraydelX with: arrayX0.

self updateVariables: arrayX1.

self qdotGearFormula. self udotGearFormula.

arrayX0 _ arrayX1].

self saveVariables.

InterpolatingCoeff: k

"solves and returns the interpolating coefficients of the predictor formulas depending on the argument anInteger;

i.e., uses the formula in the Chua's book (Eq. (17-81)). $(Y_p)_i = C_1(Y)_{i-1} + C_2(Y)_{i-2} + C_3(Y)_{i-3} + \dots + C_k(Y)_{i-k}$

"

| matrixA arrayB coeffArray |

matrixA _ (Array new: k) collect: [: each | Array new: k].

1 to: k do: [: j | 1 to: k do: [: i | (matrixA at: i) at: j put: (j*(i-1))]].

arrayB _ Array new: k withAll: 0. arrayB at: 1 put: 1. coeffArray _ self LUSolve: matrixA with: arrayB.

^ coeffArray

predictLambda

"predicts {Lambda} by applying the algorithm used in the acceleration initial condition evaluation."

| q c matrixA arrayY arrayX predictedLambda |

q _ 6*NumberOfParts. c _ NumberOfConstraints.

matrixA _ self accJacobian. arrayY _ self accError. arrayX _ self pivotLUSolve: matrixA with: arrayY.

predictedLambda _ self partition: arrayX from: (1@(q+1)) extension: (1@c).

Lambda _ predictedLambda.

predictSystem2

"predicts {q}, {u} using the Euler method, {Lambda} using the algorithm used in the initial acceleration evaluation, and {qdot} and {udot} using the Gear formulas."

"----- For {q} and {u} -----"

1 to: NumberOfParts do: [: whichPart || aPart pastq pastqdot delq predictedq pastu pastudot delu predictedu |
aPart _ PartDictionary at: whichPart.

"For {q}"

pastq _ (aPart centerOfMassMarker) position. pastqdot _ aPart velocity. delq _ self multiply: pastqdot by: Stepsize.

predictedq _ self add: pastq with: delq.

"For {u}"

pastu _ aPart stateVariableu. pastudot _ aPart stateVariableudot. delu _ self multiply: pastudot by: Stepsize.

predictedu _ self add: pastu with: delu.

"updates {q} and {u}"

aPart position: predictedq. aPart stateVariableu: predictedu].

"----- For {qdot} and {udot} -----"

self qdotGearFormula. self udotGearFormula.

"----- For {Lambda} -----"

(NumberOfConstraints ~= 0) ifTrue: [self predictLambda].

predictSystem3

"predicts the future state variables by the backward interpolaton formula in the Chua's book (Eq.(17-78))."

! coeffArray !

coeffArray _ self interpolatingCoeff: IntegrationOrder.

1 to: NumberOfParts do: [: whichPart || aPart q dumq u dumu |

aPart _ PartDictionary at: whichPart.

q _ Array new: 6 withAll: 0. u _ Array new: 6 withAll: 0.

"----- For {q}, {u} -----"

1 to: IntegrationOrder do: [: i |

dumq _ self multiply: ((aPart positionArray) at: (IntegrationOrder+1-i)) by: (coeffArray at: i).

q _ self add: q with: dumq.

dumu _ self multiply: ((aPart stateVariableuArray) at: (IntegrationOrder+1-i)) by: (coeffArray at: i).

u _ self add: u with: dumu].

aPart position: q. aPart stateVariableu: u].

"----- For {qdot}, {udot} -----"

self qdotGearFormula. self udotGearFormula.

"----- For {Lambda} -----"

(NumberOfConstraints ~= 0) ifTrue: [self predictLambda].

qdotGearFormula

"updates the array {qdot} using the Gear's implicit formula;

The maximum order is defined in the input. The possible maximum order in the system is 6."

! aPart presentq sum terms qdot !

1 to: NumberOfParts do: [: whichPart |

aPart _ PartDictionary at: whichPart.

presentq _ (aPart centerOfMassMarker) position.

sum _ Array new: 6 withAll: 0.

1 to: IntegrationOrder do: [: i || pastqi dum |

pastqi _ (aPart positionArray) at: (IntegrationOrder+1-i). dum _ self multiply: pastqi by: (GearCoeffArray at: i).

sum _ self add: sum with: dum].

terms _ self subtract: presentq with: sum.

qdot _ self multiply: terms by: (-1/(GearConstant*Stepsize)).

aPart velocity: qdot].

saveVariables

"stores the corrected state variables for the next predictions and outputs."

1 to: NumberOfParts do: [: whichPart || aPart array1 array2 matrix1 arrayqdotdot | aPart _ PartDictionary at: whichPart.

(IntegrationOrder < MaxIntegrationOrder) ifTrue: [

"----- {q} -----"

(aPart positionArray) add: ((aPart centerOfMassMarker) position).

"----- {u} -----"

```

(aPart stateVariableuArray) add: (aPart stateVariableu)).
(IntegrationOrder = MaxIntegrationOrder) ifTrue: [
  "---- {q} ----"
  (aPart positionArray) add: ((aPart centerOfMassMarker) position). (aPart positionArray) removeFirst.
  "---- {u} ----"
  (aPart stateVariableuArray) add: (aPart stateVariableu). (aPart stateVariableuArray) removeFirst].
"---- {qdotdot} ----"
array1 _ Array new: 6 withAll: 0. self put: (aPart inertial1) into: array1 at: (1@4).
array2 _ self subtract: (aPart stateVariableudot) with: array1. matrix1 _ self inversePivot: (aPart toMomentum).
arrayqdotdot _ self multiply: matrix1 with: array2. aPart acceleration: arrayqdotdot].

```

setUpIntegration

"gets the values at the 1st and 2nd time steps to use them as the past values in the subsequent integrations. The time step used in this initial integration stage is 20% of the regular time step which is specified in input. The order of integration is up to 3."

```

| dum maxOrder steps aRequest |
dum _ Stepsize.
Stepsize _ 0.2*Stepsize. "sets 20% of the regular step size"
maxOrder _ 3.
steps _ 10. "sets 10 small steps in the first two regular steps"
aRequest _ Request new.
1 to: steps do: [: whichStep |
  (whichStep < maxOrder) ifTrue: [ self changeIntegrationOrder: whichStep].
  (whichStep = maxOrder) ifTrue: [ self changeIntegrationOrder: maxOrder].
  self predictSystem2. self correctSystem.
  ((whichStep = 5) | (whichStep = 10)) ifTrue: [
    "outputs the first two variable columns"
    aRequest openOutput.
    "stores the first two variable columns for the regular integration"
    1 to: NumberOfParts do: [: whichPart | | aPart | aPart _ PartDictionary at: whichPart.
      (aPart positionArrayDum) add: ((aPart positionArray) last).
      (aPart stateVariableuArrayDum) add: ((aPart stateVariableuArray) last)]]].
"sets the regular integration"
Stepsize _ dum.
1 to: NumberOfParts do: [: whichPart | | aPart |
  1 to: 3 do: [: i | aPart _ PartDictionary at: whichPart.
    (aPart positionArray) removeFirst. (aPart stateVariableuArray) removeFirst].
    (aPart positionArray) add: ((aPart positionArrayDum) at: 1).
    (aPart positionArray) add: ((aPart positionArrayDum) at: 2).
    (aPart positionArray) add: ((aPart positionArrayDum) at: 3).
    (aPart stateVariableuArray) add: ((aPart stateVariableuArrayDum) at: 1).
    (aPart stateVariableuArray) add: ((aPart stateVariableuArrayDum) at: 2).
    (aPart stateVariableuArray) add: ((aPart stateVariableuArrayDum) at: 3)].

```

setUpStartingVector

"sets up the starting Values {X0} for the first iteration at each time step."

```

| q c n arrayX0 |
q _ 6*NumberOfParts. c _ NumberOfConstraints. n _ q+q+c+q.
arrayX0 _ Array new: n withAll: 0.
1 to: NumberOfParts do: [: whichPart | | aPart where arrayq arrayu arrayQ |
  aPart _ PartDictionary at: whichPart. where _ 6*(((aPart partNo) asNumber) - 1).
  arrayq _ (aPart centerOfMassMarker) position. arrayu _ aPart stateVariableu. arrayQ _ aPart stateVariableQ.
  self put: arrayq into: arrayX0 at: (1@(where+1)). self put: arrayu into: arrayX0 at: (1@(q+where+1)).
  self put: arrayQ into: arrayX0 at: (1@(q+q+c+where+1))].
(c ~= 0) ifTrue: [self put: Lambda into: arrayX0 at: (1@(q+q+1))].
^ arrayX0

```

solveSystem

"executes the predictor-corrector porocess with the initial conditions and stores the result."

```

| aRequest |
self evaluateIC.
aRequest _ Request new.

```

```

aRequest openOutput.
1 to: NoOfSteps do: [: whichStep |
    (whichStep < MaxIntegrationOrder) ifTrue: [ self changeIntegrationOrder: whichStep].
    (whichStep = MaxIntegrationOrder) ifTrue: [ self changeIntegrationOrder: MaxIntegrationOrder].
    self predictSystem2. self correctSystem.
    aRequest openOutput].
aRequest closeOutput.

```

solveSystemSmooth

"starts the predictor-corrector process using the initial conditions and smooth starting values, and stores the result. To use this method, the Gear integration order must be three or higher since the values at the first and second time steps are obtained from the method 'setUpIntegration'."

```

| aRequest |
self evaluateIC.
"stores the initial conditions to be used for the regular integration"
1 to: NumberOfParts do: [: whichPart | aPart | aPart _ PartDictionary at: whichPart.
    (aPart positionArrayDum) add: ((aPart positionArray) last).
    (aPart stateVariableuArrayDum) add: ((aPart stateVariableuArray) last)].
aRequest _ Request new.
aRequest openOutput.
"----- For Initial Integration Stage -----"
self setUpIntegration.
"----- For Regular Integration Stage -----"
3 to: NoOfSteps do: [: whichStep |
    (whichStep < MaxIntegrationOrder) ifTrue: [ self changeIntegrationOrder: whichStep].
    (whichStep = MaxIntegrationOrder) ifTrue: [ self changeIntegrationOrder: MaxIntegrationOrder].
    self predictSystem2. self correctSystem.
    aRequest openOutput].
aRequest closeOutput.

```

uDotGearFormula

"updates the array {uDot} using the Gear's implicit formula.

The maximum order is defined in the input. The possible maximum order in the input is 6."

```

| aPart presentu sum dum terms uDot |
1 to: NumberOfParts do: [: whichPart | aPart _ PartDictionary at: whichPart.
    presentu _ aPart stateVariableu.
    sum _ Array new: 6 withAll: 0.
    1 to: IntegrationOrder do: [: i |
        dum _ self multiply: ((aPart stateVariableuArray) at: (IntegrationOrder+1-i)) by: (GearCoeffArray at: i).
        sum _ self add: sum with: dum].
    terms _ self subtract: presentu with: sum.
    uDot _ self multiply: terms by: (-1/(GearConstant*Stepsize)).
    aPart stateVariableuDot: uDot].

```

updateVariables: arrayX1

"updates the state variables during the iterations."

```

| q c |
q _ 6*NumberOfParts. c _ NumberOfConstraints.
1 to: NumberOfParts do: [: whichPart | aPart where arrayq arrayu arrayQ |
    aPart _ PartDictionary at: whichPart. where _ 6*(((aPart partNo) asNumber) - 1).
    arrayq _ self partition: arrayX1 from: (1@(where+1)) extension: (1@6).
    arrayu _ self partition: arrayX1 from: (1@(q+where+1)) extension: (1@6).
    arrayQ _ self partition: arrayX1 from: (1@(q+q+c+where+1)) extension: (1@6).
    aPart position: arrayq. aPart stateVariableu: arrayu. aPart stateVariableQ: arrayQ].
(c ~= 0) ifTrue: [Lambda _ self partition: arrayX1 from: (1@(q+q+1)) extension: (1@c)].
self systemGenForces.

```

accError

"returns the right-hand side of a set of initial acceleration analysis equations."

```

| q arrayA arrayQ upper c arrayB arrayE |
q _ 6*NumberOfParts.
arrayA _ Array new: q withAll: 0.

```

```

1 to: NumberOfParts do: [: whichPart || aPart where array |
  aPart _ PartDictionary at: whichPart. where _ 6*(((aPart partNo) asNumber) -1).
  array _ self subtract: (aPart inertia1) with: (aPart inertia3). self put: array into: arrayA at: (1@(where+1+3)).
arrayQ _ self systemGenForces. upper _ self subtract: arrayA with: arrayQ.
c _ 0.
(NumberOfConstraints ~= 0) ifTrue: [
arrayB _ Array new: NumberOfConstraints.
1 to: (JointDictionary size) do: [: whichJoint || aJoint aMarkerI aMarkerJ aPartI aPartJ aPartNoI aPartNoJ dum1
dum2 arrayV arrayVI arrayVJ |
  aJoint _ JointDictionary at: whichJoint. aMarkerI _ aJoint markerI. aMarkerJ _ aJoint markerJ.
  aPartI _ aMarkerI part. aPartJ _ aMarkerJ part.
  aPartNoI _ (aPartI partNo) asNumber. aPartNoJ _ (aPartJ partNo) asNumber.
  (aPartNoI = 0) ifTrue: [ arrayVI _ Array new: 6 withAll: 0] ifFalse: [ arrayVI _ aPartI velocity].
  (aPartNoJ = 0) ifTrue: [ arrayVJ _ Array new: 6 withAll: 0] ifFalse: [ arrayVJ _ aPartJ velocity].
  arrayV _ arrayVI, arrayVJ.
  ((aJoint type) = 'spherical') ifTrue: [
    dum1 _ self multiply: (aJoint sub11AtPoint1) with: arrayV.
    dum2 _ self multiply: (self transpose: dum1) with: arrayV. arrayB at: (c+1) put: dum2.
    dum1 _ self multiply: (aJoint sub11AtPoint2) with: arrayV.
    dum2 _ self multiply: (self transpose: dum1) with: arrayV. arrayB at: (c+2) put: dum2.
    dum1 _ self multiply: (aJoint sub11AtPoint3) with: arrayV.
    dum2 _ self multiply: (self transpose: dum1) with: arrayV. arrayB at: (c+3) put: dum2. c _ c + 3].
  ((aJoint type) = 'universal') ifTrue: [
    dum1 _ self multiply: (aJoint sub11AtPoint1) with: arrayV.
    dum2 _ self multiply: (self transpose: dum1) with: arrayV. arrayB at: (c+1) put: dum2.
    dum1 _ self multiply: (aJoint sub11AtPoint2) with: arrayV.
    dum2 _ self multiply: (self transpose: dum1) with: arrayV. arrayB at: (c+2) put: dum2.
    dum1 _ self multiply: (aJoint sub11AtPoint3) with: arrayV.
    dum2 _ self multiply: (self transpose: dum1) with: arrayV. arrayB at: (c+3) put: dum2.
    dum1 _ self multiply: (aJoint sub11Perpendicular3) with: arrayV.
    dum2 _ self multiply: (self transpose: dum1) with: arrayV. arrayB at: (c+4) put: dum2. c _ c + 4].
  ((aJoint type) = 'revolute') ifTrue: [
    dum1 _ self multiply: (aJoint sub11AtPoint1) with: arrayV.
    dum2 _ self multiply: (self transpose: dum1) with: arrayV. arrayB at: (c+1) put: dum2.
    dum1 _ self multiply: (aJoint sub11AtPoint2) with: arrayV.
    dum2 _ self multiply: (self transpose: dum1) with: arrayV. arrayB at: (c+2) put: dum2.
    dum1 _ self multiply: (aJoint sub11AtPoint3) with: arrayV.
    dum2 _ self multiply: (self transpose: dum1) with: arrayV. arrayB at: (c+3) put: dum2.
    dum1 _ self multiply: (aJoint sub11Perpendicular1) with: arrayV.
    dum2 _ self multiply: (self transpose: dum1) with: arrayV. arrayB at: (c+4) put: dum2.
    dum1 _ self multiply: (aJoint sub11Perpendicular2) with: arrayV.
    dum2 _ self multiply: (self transpose: dum1) with: arrayV. arrayB at: (c+5) put: dum2. c _ c + 5].
  ((aJoint type) = 'translational') ifTrue: [
    dum1 _ self multiply: (aJoint sub11InPlane1) with: arrayV.
    dum2 _ self multiply: (self transpose: dum1) with: arrayV. arrayB at: (c+1) put: dum2.
    dum1 _ self multiply: (aJoint sub11InPlane2) with: arrayV.
    dum2 _ self multiply: (self transpose: dum1) with: arrayV. arrayB at: (c+2) put: dum2.
    dum1 _ self multiply: (aJoint sub11Perpendicular1) with: arrayV.
    dum2 _ self multiply: (self transpose: dum1) with: arrayV. arrayB at: (c+3) put: dum2.
    dum1 _ self multiply: (aJoint sub11Perpendicular2) with: arrayV.
    dum2 _ self multiply: (self transpose: dum1) with: arrayV. arrayB at: (c+4) put: dum2.
    dum1 _ self multiply: (aJoint sub11Perpendicular4) with: arrayV.
    dum2 _ self multiply: (self transpose: dum1) with: arrayV. arrayB at: (c+5) put: dum2. c _ c + 5].
  ((aJoint type) = 'cylindrical') ifTrue: [
    dum1 _ self multiply: (aJoint sub11InPlane1) with: arrayV.
    dum2 _ self multiply: (self transpose: dum1) with: arrayV. arrayB at: (c+1) put: dum2.
    dum1 _ self multiply: (aJoint sub11InPlane2) with: arrayV.
    dum2 _ self multiply: (self transpose: dum1) with: arrayV. arrayB at: (c+2) put: dum2.
    dum1 _ self multiply: (aJoint sub11Perpendicular1) with: arrayV.
    dum2 _ self multiply: (self transpose: dum1) with: arrayV. arrayB at: (c+3) put: dum2.
    dum1 _ self multiply: (aJoint sub11Perpendicular2) with: arrayV.
    dum2 _ self multiply: (self transpose: dum1) with: arrayV. arrayB at: (c+4) put: dum2. c _ c + 4].

```



```
((aJoint type) = 'planar') ifTrue: [
    dum1 _ self multiply: (aJoint sub11InPlane3) with: arrayV.
    dum2 _ self multiply: (self transpose: dum1) with: arrayV. arrayB at: (c+1) put: dum2.
    dum1 _ self multiply: (aJoint sub11Perpendicular1) with: arrayV.
    dum2 _ self multiply: (self transpose: dum1) with: arrayV. arrayB at: (c+2) put: dum2.
    dum1 _ self multiply: (aJoint sub11Perpendicular2) with: arrayV.
    dum2 _ self multiply: (self transpose: dum1) with: arrayV. arrayB at: (c+3) put: dum2. c _ c + 3]].
arrayB _ self multiply: arrayB by: (-1)]. arrayE _ Array new: (q+c). self put: upper into: arrayE at: (1@1).
(NumberOfConstraints ~= 0) ifTrue: [ self put: arrayB into: arrayE at: (1@(q+1))].
^ arrayE
```

accJacobian

"returns the updated acceleration-Lagrange multipliers Jacobian [A] for the determination of initial conditions."

```
| q c matrixA matrixM matrixMi a12 a21 |
q _ 6*NumberOfParts. c _ NumberOfConstraints.
matrixM _ (Array new: q) collect: [: each | Array new: q withAll: 0].
1 to: NumberOfParts do: [: whichPart | | aPart where |
    aPart _ PartDictionary at: whichPart. where _ 6*(((aPart partNo) asNumber) - 1).
    matrixMi _ self multiply: (aPart toMomentum) by: (-1).
    self put: matrixMi into: matrixM at: ((where+1)@(where+1))].
matrixA _ (Array new: (q+c)) collect: [: each | Array new: (q+c) withAll: 0].
self put: matrixM into: matrixA at: (1@1).
(c ~= 0) ifTrue: [a21 _ MatrixdGdq. a12 _ self transpose: a21.
    self put: a12 into: matrixA at: ((q+1)@1). self put: a21 into: matrixA at: (1@(q+1))].
^ matrixA
```

analyzeAcc

"starts the initial conditions analysis for acceleration and Lagrange multiplier, and then stores the result. The acceleration is in Xdotdot Ydotdot Zdotdot Phidotdot Thetadotdot Psidotdot."

```
| q c matrixA arrayX arrayY |
q _ 6*NumberOfParts. c _ NumberOfConstraints.
"solves for {qdotdot} and {Lambda}"
matrixA _ self accJacobian. arrayY _ self accError. arrayX _ self pivotLUSolve: matrixA with: arrayY.
"updates {Lambda}"
(c ~= 0) ifTrue: [Lambda _ self partition: arrayX from: (1@(q+1)) extension: (1@c)].
"updates {qdotdot}, {udot}, and {Q}."
1 to: NumberOfParts do: [: whichPart | | aPart where arrayqdotdot arrayudot array1 array2 |
    aPart _ PartDictionary at: whichPart. where _ 6*(((aPart partNo) asNumber) - 1).
    arrayqdotdot _ self partition: arrayX from: (1@(where+1)) extension: (1@6).
    aPart acceleration: arrayqdotdot.
    arrayudot _ Array new: 6 withAll: 0.
    array1 _ self put: (aPart inertial1) into: arrayudot at: (1@4).
    array2 _ self multiply: (aPart toMomentum) with: arrayqdotdot.
    arrayudot _ self add: array1 with: array2.
    aPart stateVariableudot: arrayudot.
    self systemGenForces.
    aPart stateVariableQ: (aPart generalizedForce) "initial value" ].
```

analyzePos

"starts the position initial conditions analysis."

```
| q c arrayq0 posLambda maxIterations matrixA arrayY arrayX arraydelq arrayq1 |
"----- sets up the initial {q0} for iterations -----"
q _ 6*NumberOfParts. c _ NumberOfConstraints.
arrayq0 _ Array new: q.
1 to: NumberOfParts do: [: whichPart | | aPart where |
    aPart _ PartDictionary at: whichPart. where _ 6*(((aPart partNo) asNumber) - 1).
    self put: (aPart initialPos) into: arrayq0 at: (1@(where+1))].
posLambda _ Array new: c withAll: 0.
"----- solves for {q} -----"
maxIterations _ 5.
1 to: maxIterations do: [: i |
    matrixA _ self posJacobian: posLambda. arrayY _ self posError.
```

```

arrayX _ self scalePivotLUSolve: matrixA with: arrayY.
arraydelq _ self partition: arrayX from: (1@1) extension: (1@q). arrayq1 _ self add: arraydelq with: arrayq0.
"updates {q}."
1 to: NumberOfParts do: [: whichPart || aPart where updatedPos |
    aPart _ PartDictionary at: whichPart. where _ 6*(((aPart partNo) asNumber) - 1).
    updatedPos _ self partition: arrayq1 from: (1@(where+1)) extension: (1@6). aPart position: updatedPos.
    "stores {q} and updates the class variable 'MatrixdGdq' after the iterations"
    (i = maxIterations) ifTrue: [ (aPart positionArray) add: updatedPos. (c ~= 0) ifTrue: [self setSubJacobian31]].
arrayq0 _ arrayq1. posLambda _ self partition: arrayX from: (1@(q+1)) extension: (1@c)].

```

analyzeVel

```

"starts the velocity initial conditions analysis."
| matrixA arrayX arrayY arrayqdot |
"----- solves for {qdot} -----"
matrixA _ self velJacobian. arrayY _ self velError. arrayX _ self scalePivotLUSolve: matrixA with: arrayY.
"updates and stores {qdot} and {u}"
1 to: NumberOfParts do: [: whichPart || aPart where momentum |
    aPart _ PartDictionary at: whichPart. where _ 6*(((aPart partNo) asNumber) - 1).
    arrayqdot _ self partition: arrayX from: (1@(where+1)) extension: (1@6).
    aPart velocity: arrayqdot.
    momentum _ self multiply: (aPart toMomentum) with: arrayqdot.
    aPart stateVariableu: momentum. "initial value"
    (aPart stateVariableuArray) add: momentum].

```

evaluateIC

```

"evaluates the prescribed initial conditions for position, velocity, acceleration, and Lagrange multipliers."
self analyzePos. self analyzeVel. self analyzeAcc.

```

posError

```

"returns the right-hand side of a set of initial position analysis equations."
| q c arrayG matrixW1 arrayq arrayq0 dq upper arrayE |
q _ 6*NumberOfParts. c _ NumberOfConstraints.
matrixW1 _ self posWeights. arrayq0 _ Array new: q. arrayq _ Array new: q.
1 to: NumberOfParts do: [: whichPart || aPart where arrayX0 arrayX |
    aPart _ PartDictionary at: whichPart. where _ 6*(((aPart partNo) asNumber) - 1).
    arrayX0 _ aPart initialPos. self put: arrayX0 into: arrayq0 at: (1@(where+1)).
    arrayX _ (aPart centerOfMassMarker) position. self put: arrayX into: arrayq at: (1@(where+1))).
dq _ self subtract: arrayq with: arrayq0. upper _ self multiply: matrixW1 with: dq. arrayE _ Array new: (q+c).
self put: upper into: arrayE at: (1@1).
(c ~= 0) ifTrue: [ arrayG _ self systemConstraintFunction. self put: arrayG into: arrayE at: (1@(q+1))].
arrayE _ self multiply: arrayE by: (-1).
^ arrayE

```

posJacobian11: posLambda

```

"returns the updated position Jacobian partition [A11] for the determination of initial conditions."
| q c matrixA11 matrix |
q _ 6*NumberOfParts.
"----- For weights matrix [W] -----"
matrixA11 _ self posWeights.
"----- For <Lambda>[d(dG/dq)/dq] -----"
(NumberOfConstraints ~= 0) ifTrue: [
    matrix _ (Array new: q) collect: [: each | Array new: q withAll: 0].
    c _ 0.
    1 to: (JointDictionary size) do: [: whichJoint || aJoint aMarkerI aPartI aMarkerJ aPartJ dumMatrix aMatrix sub1
sub2 sub3 sub4 sub5 partitionI partitionJ partitionIJ partitionJI whereI whereJ dum |
        aJoint _ JointDictionary at: whichJoint.
        ((aJoint type) = 'spherical') ifTrue: [
            sub1 _ self multiply: (aJoint sub11AtPoint1) by: (posLambda at: (c+1)).
            sub2 _ self multiply: (aJoint sub11AtPoint2) by: (posLambda at: (c+2)).
            sub3 _ self multiply: (aJoint sub11AtPoint3) by: (posLambda at: (c+3)).
            dumMatrix _ self add: sub1 with: sub2. aMatrix _ self add: dumMatrix with: sub3. c _ c + 3].
        ((aJoint type) = 'universal') ifTrue: [

```

```

sub1 _ self multiply: (aJoint sub11AtPoint1) by: (posLambda at: (c+1)).
sub2 _ self multiply: (aJoint sub11AtPoint2) by: (posLambda at: (c+2)).
sub3 _ self multiply: (aJoint sub11AtPoint3) by: (posLambda at: (c+3)).
sub4 _ self multiply: (aJoint sub11Perpendicular3) by: (posLambda at: (c+4)).
dumMatrix _ self add: sub1 with: sub2. dumMatrix _ self add: dumMatrix with: sub3.
aMatrix _ self add: dumMatrix with: sub4. c _ c + 4].
((aJoint type) = 'revolute') ifTrue: [
  sub1 _ self multiply: (aJoint sub11AtPoint1) by: (posLambda at: (c+1)).
  sub2 _ self multiply: (aJoint sub11AtPoint2) by: (posLambda at: (c+2)).
  sub3 _ self multiply: (aJoint sub11AtPoint3) by: (posLambda at: (c+3)).
  sub4 _ self multiply: (aJoint sub11Perpendicular1) by: (posLambda at: (c+4)).
  sub5 _ self multiply: (aJoint sub11Perpendicular2) by: (posLambda at: (c+5)).
  dumMatrix _ self add: sub1 with: sub2. dumMatrix _ self add: dumMatrix with: sub3.
  dumMatrix _ self add: dumMatrix with: sub4. aMatrix _ self add: dumMatrix with: sub5. c _ c + 5].
((aJoint type) = 'translational') ifTrue: [
  sub1 _ self multiply: (aJoint sub11InPlane1) by: (posLambda at: (c+1)).
  sub2 _ self multiply: (aJoint sub11InPlane2) by: (posLambda at: (c+2)).
  sub3 _ self multiply: (aJoint sub11Perpendicular1) by: (posLambda at: (c+3)).
  sub4 _ self multiply: (aJoint sub11Perpendicular2) by: (posLambda at: (c+4)).
  sub5 _ self multiply: (aJoint sub11Perpendicular4) by: (posLambda at: (c+5)).
  dumMatrix _ self add: sub1 with: sub2. dumMatrix _ self add: dumMatrix with: sub3.
  dumMatrix _ self add: dumMatrix with: sub4. aMatrix _ self add: dumMatrix with: sub5. c _ c + 5].
((aJoint type) = 'cylindrical') ifTrue: [
  sub1 _ self multiply: (aJoint sub11InPlane1) by: (posLambda at: (c+1)).
  sub2 _ self multiply: (aJoint sub11InPlane2) by: (posLambda at: (c+2)).
  sub3 _ self multiply: (aJoint sub11Perpendicular1) by: (posLambda at: (c+3)).
  sub4 _ self multiply: (aJoint sub11Perpendicular2) by: (posLambda at: (c+4)).
  dumMatrix _ self add: sub1 with: sub2. dumMatrix _ self add: dumMatrix with: sub3.
  aMatrix _ self add: dumMatrix with: sub4. c _ c + 4].
((aJoint type) = 'planar') ifTrue: [
  sub1 _ self multiply: (aJoint sub11InPlane3) by: (posLambda at: (c+1)).
  sub2 _ self multiply: (aJoint sub11Perpendicular1) by: (posLambda at: (c+2)).
  sub3 _ self multiply: (aJoint sub11Perpendicular2) by: (posLambda at: (c+3)).
  dumMatrix _ self add: sub1 with: sub2. aMatrix _ self add: dumMatrix with: sub3. c _ c + 3].
"Here, aMatrix has the size 12 by 12 (symmetric matrix)."
partitionI _ self partition: aMatrix from: (1@1) extension: (6@6).
partitionIJ _ self partition: aMatrix from: (7@1) extension: (6@6).
partitionJ _ self partition: aMatrix from: (7@7) extension: (6@6).
partitionJI _ self partition: aMatrix from: (1@7) extension: (6@6).
aMarkerI _ aJoint markerI. aMarkerJ _ aJoint markerJ. aPartI _ aMarkerI part. aPartJ _ aMarkerJ part.
whereI _ 6*(((aPartI partNo) asNumber) - 1). whereJ _ 6*(((aPartJ partNo) asNumber) - 1).
dum _ (Array new: q) collect: [: each | Array new: q withAll: 0].
(((aPartI partNo) ~= '00') & ((aPartJ partNo) ~= '00'))
ifTrue: [
  self put: partitionI into: dum at: ((whereI+1)@(whereI+1)).
  self put: partitionIJ into: dum at: ((whereJ+1)@(whereI+1)).
  self put: partitionJ into: dum at: ((whereJ+1)@(whereJ+1)).
  self put: partitionJI into: dum at: ((whereI+1)@(whereJ+1))]
ifFalse: [
  ((aPartI partNo) ~= '00') ifTrue: [ self put: partitionI into: dum at: ((whereI+1)@(whereI+1)).
  ((aPartJ partNo) ~= '00') ifTrue: [ self put: partitionJ into: dum at: ((whereJ+1)@(whereJ+1))].
  matrix _ self add: matrix with: dum]].
(NumberOfConstraints ~= 0) ifTrue: [ matrixA11 _ self add: matrixA11 with: matrix].
^ matrixA11

```

posJacobian: posLambda

"returns the updated position Jacobian [A] for the determination of initial conditions."

| q c matrixA matrixA11 a12 a21 |

q _ 6*NumberOfParts. c _ NumberOfConstraints.

matrixA _ (Array new: (q+c)) collect: [: each | Array new: (q+c) withAll: 0].

matrixA11 _ self posJacobian11: posLambda. self put: matrixA11 into: matrixA at: (1@1).

(c ~= 0) ifTrue: [a21 _ self setSubJacobian31. a12 _ self transpose: a21.

```

    self put: a12 into: matrixA at: ((q+1)@1). self put: a21 into: matrixA at: (1@(q+1)).
    ^ matrixA

```

posWeights

"returns the weigh factors for the initial position analysis [W1]."

```

    | q matrixW1 |
    q _ 6*NumberOfParts.
    matrixW1 _ (Array new: q) collect: [: each | Array new: q withAll: 0].
    1 to: NumberOfParts do: [: whichPart | | aPart where weights |
        aPart _ PartDictionary at: whichPart. where _ 6*(((aPart partNo) asNumber) - 1).
        weights _ aPart positionWeights.
        1 to: 6 do: [: i | (matrixW1 at: (where+i)) at: (where+i) put: (weights at: i)]]].
    ^ matrixW1

```

systemGenForces

"returns the matrix {F(q,qdot)} for the generalized forces of the system.

Gravitational forces are included. Note that the gravitational forces acts always in the negative Z of the ground frame. "

```

    | q systemGenForceArray gravityArray |

```

```

    q _ 6*NumberOfParts.

```

```

    "----- For Gravity -----"

```

```

    gravityArray _ Array new: q withAll: 0.

```

```

    1 to: NumberOfParts do: [: whichPart | | aPart where mg |

```

```

        aPart _ PartDictionary at: whichPart. where _ 6*(((aPart partNo) asNumber) - 1).

```

```

        mg _ (aPart mass)*Gravity. gravityArray at: (where+3) put: mg].

```

```

    systemGenForceArray _ gravityArray.

```

```

    "----- For Translational Spring Forces -----"

```

```

    (NumberOfSprings ~= 0) ifTrue: [| springForceTorqueArray dum |

```

```

        springForceTorqueArray _ Array new: q withAll: 0. dum _ Array new: q withAll: 0.

```

```

        1 to: NumberOfSprings do: [: whichSpring | | aSpring partI partJ partNoI partNoJ whereI whereJ forceTorque
arrayI arrayJ |

```

```

            aSpring _ SpringDictionary at: whichSpring.

```

```

            partI _ (aSpring markerI) part. partJ _ (aSpring markerJ) part. partNoI _ partI partNo. partNoJ _ partJ partNo.

```

```

            whereI _ 6*(((partNoI asNumber) - 1). whereJ _ 6*(((partNoJ asNumber) - 1).

```

```

            forceTorque _ aSpring forceTorqueOnPartIJ. "12 by 1"

```

```

            ((partNoI ~= '00') & (partNoJ ~= '00')) ifTrue: [

```

```

                arrayI _ self partition: forceTorque from: (1@1) extension: (1@6).

```

```

                arrayJ _ self partition: forceTorque from: (1@7) extension: (1@6).

```

```

                self put: arrayI into: dum at: (1@(whereI+1)). self put: arrayJ into: dum at: (1@(whereJ+1))].

```

```

            ((partNoI = '00') & (partNoJ ~= '00')) ifTrue: [

```

```

                arrayJ _ self partition: forceTorque from: (1@7) extension: (1@6).

```

```

                self put: arrayJ into: dum at: (1@(whereJ+1))].

```

```

            ((partNoI ~= '00') & (partNoJ = '00')) ifTrue: [

```

```

                arrayI _ self partition: forceTorque from: (1@1) extension: (1@6).

```

```

                self put: arrayI into: dum at: (1@(whereI+1))].

```

```

            springForceTorqueArray _ self add: springForceTorqueArray with: dum].

```

```

            systemGenForceArray _ self add: gravityArray with: springForceTorqueArray].

```

```

    "----- For Forces or Torques -----"

```

```

    (NumberOfForces ~= 0) ifTrue: [| forceArray dum |

```

```

        forceArray _ Array new: q withAll: 0.

```

```

        dum _ Array new: q.

```

```

        1 to: NumberOfForces do: [: whichForce | | aForce actionPart reactionPart whereI whereJ array partition1
partition2 |

```

```

            aForce _ ForceDictionary at: whichForce.

```

```

            (aForce pattern) = 'action-reaction' ifTrue: [

```

```

                actionPart _ (aForce actionMarker) part. whereI _ 6*(((actionPart partNo) asNumber) - 1).

```

```

                reactionPart _ (aForce reactionMarker) part. whereJ _ 6*(((reactionPart partNo) asNumber) - 1)].

```

```

            (aForce pattern) = 'action-only' ifTrue: [

```

```

                actionPart _ (aForce actionMarker) part. whereI _ 6*(((actionPart partNo) asNumber) - 1)].

```

```

            (aForce type) = 'force' ifTrue: [ array _ aForce translationalForce.

```

```

                (aForce pattern) = 'action-reaction' ifTrue: [

```

```

                    partition1 _ self partition: array from: (1@1) extension: (1@6).

```

```

    partition2 _ self partition: array from: (1@7) extension: (1@6).
    self put: partition1 into: dum at: (1@(whereI+1)). self put: partition2 into: dum at: (1@(whereJ+1))).
    (aForce pattern) = 'action-only' ifTrue: [ self put: array into: dum at: (1@(whereI+1))].
    (aForce type) = 'torque' ifTrue: [ array _ aForce directedTorque.
    (aForce pattern) = 'action-reaction' ifTrue: [
        partition1 _ self partition: array from: (1@1) extension: (1@3).
        partition2 _ self partition: array from: (1@4) extension: (1@3).
        self put: partition1 into: dum at: (1@(whereI+1+3)). self put: partition2 into: dum at: (1@(whereJ+1+3))].
    (aForce pattern) = 'action-only' ifTrue: [ self put: array into: dum at: (1@(whereI+1+3))].
    forceArray _ self add: forceArray with: dum].
    systemGenForceArray _ self add: systemGenForceArray with: forceArray].
"stores generalized forces onto each part."
1 to: NumberOfParts do: [: whichPart | | aPart where array |
    aPart _ PartDictionary at: whichPart. where _ 6*(((aPart partNo) asNumber) - 1).
    array _ self partition: systemGenForceArray from: (1@(where+1)) extension: (1@6). aPart generalizedForce:
    array].
^ systemGenForceArray

```

velError

```

"returns the right-hand side of a set of initial velocity analysis equations."
| q c matrixW2 arrayV0 upper arrayE |
q _ 6*NumberOfParts. c _ NumberOfConstraints.
matrixW2 _ self velWeights.
arrayV0 _ Array new: q.
1 to: NumberOfParts do: [: whichPart | | aPart where array |
    aPart _ PartDictionary at: whichPart. where _ 6*(((aPart partNo) asNumber) - 1).
    array _ aPart initialVel. self put: array into: arrayV0 at: (1@(where+1))].
upper _ self multiply: matrixW2 with: arrayV0. arrayE _ Array new: (q+c) withAll: 0.
self put: upper into: arrayE at: (1@1).
^ arrayE

```

velJacobian

```

"returns the updated velocity Jacobian [A] for the determination of initial conditions."
| q c matrixA matrixW2 a21 a12 |
q _ 6*NumberOfParts. c _ NumberOfConstraints.
matrixA _ (Array new: (q+c)) collect: [: each | Array new: (q+c) withAll: 0].
matrixW2 _ self velWeights. self put: matrixW2 into: matrixA at: (1@1).
(c ~= 0) ifTrue: [ a21 _ MatrixdGdq. a12 _ self transpose: a21.
    self put: a12 into: matrixA at: ((q+1)@1). self put: a21 into: matrixA at: (1@(q+1))].
^ matrixA

```

velWeights

```

"returns the weigh factors for the initial velocity analysis [W2]."
| q matrixW2 |
q _ 6*NumberOfParts. matrixW2 _ (Array new: q) collect: [: each | Array new: q withAll: 0].
1 to: NumberOfParts do: [: whichPart | | aPart where |
    aPart _ PartDictionary at: whichPart. where _ 6*(((aPart partNo) asNumber) - 1).
    1 to: 6 do: [: i | (matrixW2 at: (where+i)) at: (where+i) put: ((aPart velocityWeights) at: i)]].
^ matrixW2

```

B.2 Spring Class

System subclass: #Spring

```

    instanceVariableNames: 'freeLength springConstant markerI markerJ type '
    classVariableNames: ''
    poolDictionaries: ''
    category: 'Multibody System'

```

```

initializeSpringBetween: marker1 and: marker2 springConstant: aConst freeLength: aLength

```

"Initializes a spring using the arguments"
 markerI _ marker1. markerJ _ marker2.
 springConstant _ aConst asGoodDouble.
 freeLength _ aLength asGoodDouble.
 NumberOfSprings _ NumberOfSprings + 1.
 SpringDictionary add: self

freeLength
 "Returns the unstressed length of a spring."
 ^ freeLength

markerI
 "Returns the i-marker instance related to a spring."
 ^ markerI

markerJ
 "Returns the j-marker instance related to a spring."
 ^ markerJ

springConstant
 "Returns the spring constant of a spring."
 ^ springConstant

forceTorqueOnPartIJ
 "returns the force and torque contribution on part I and J (12 by 1) done by a spring."
 | partI partJ aMarkerI aMarkerJ x y z phi the psi sA sB sC cA cB cC t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t18 t19 t20
 t21 dLxdPhi dLxdThe dLxdPsi dLydPhi dLydThe dLydPsi dLzdThe dLzdPsi dLdPhi dLdThe dLdPsi forceArrayt term1
 term2 term3 forceArray torqueArray forceTorqueArrayI forceTorqueArrayJ forceTorqueArrayIJ |
 "----- For Part I -----"
 "Force Contribution"
 forceArray _ self springForceOnMarkerI.
 "Torque Contribution"
 aMarkerI _ self markerI.
 partI _ aMarkerI part.
 x _ (aMarkerI position) at: 1. y _ (aMarkerI position) at: 2. z _ (aMarkerI position) at: 3.
 phi _ ((partI centerOfMassMarker) position) at: 4.
 the _ ((partI centerOfMassMarker) position) at: 5.
 psi _ ((partI centerOfMassMarker) position) at: 6.
 sA _ phi sin. sB _ the sin. sC _ psi sin. cA _ phi cos. cB _ the cos. cC _ psi cos.
 t1 _ -1*sA*sB.
 t2 _ cA*sC + (sA*cB*cC).
 t3 _ sA*cB*sC - (cA*cC).
 t4 _ cA*sB.
 t5 _ sA*sC - (cA*cB*cC).
 t6 _ -1*(cA*cB*sC + (sA*cC)).
 t7 _ cA*cB.
 t8 _ cA*sB*cC.
 t9 _ cA*sB*sC.
 t10 _ sA*cB.
 t11 _ sA*sB*cC.
 t12 _ sA*sB*sC.
 t18 _ -1*sB*cC.
 t19 _ -1*sB*sC.
 t20 _ cB*cC.
 t21 _ -1*cB*sC.
 dLxdPhi _ t4*z + (t5*y) + (t6*x).
 dLydPhi _ -1*t1*z - (t2*y) - (t3*x).
 dLdPhi _ Array new: 3.
 dLdPhi at: 1 put: dLxdPhi; at: 2 put: dLydPhi; at: 3 put: 0.
 dLxdThe _ t10*z + (t11*y) + (t12*x).
 dLydThe _ -1*t7*z - (t8*y) - (t9*x).
 dLzdThe _ -1*sB*z + (t20*y) - (t21*x).

```

dLdThe _ Array new: 3.
dLdThe at: 1 put: dLxdThe; at: 2 put: dLydThe; at: 3 put: dLzdThe.
dLxdPsi _ t3*y - (t2*x).
dLydPsi _ t6*y - (t5*x).
dLzdPsi _ t19*y - (t18*x).
dLdPsi _ Array new: 3.
dLdPsi at: 1 put: dLxdPsi; at: 2 put: dLydPsi; at: 3 put: dLzdPsi.
forceArrayt _ super transpose: forceArray.
term1 _ super multiply: forceArrayt with: dLdPhi.
term2 _ super multiply: forceArrayt with: dLdThe.
term3 _ super multiply: forceArrayt with: dLdPsi.
torqueArray _ Array new: 3.
torqueArray at: 1 put: term1; at: 2 put: term2; at: 3 put: term3.
forceTorqueArrayI _ forceArray,torqueArray.
"----- For Part J -----"
"Force Contribution"
forceArray _ super multiply: forceArray by: (-1).
"Torque Contribution"
aMarkerJ _ self markerJ.
partJ _ aMarkerJ part.
x _ (aMarkerJ position) at: 1. y _ (aMarkerJ position) at: 2. z _ (aMarkerJ position) at: 3.
phi _ ((partJ centerOfMassMarker) position) at: 4.
the _ ((partJ centerOfMassMarker) position) at: 5.
psi _ ((partJ centerOfMassMarker) position) at: 6.
sA _ phi sin. sB _ the sin. sC _ psi sin. cA _ phi cos. cB _ the cos. cC _ psi cos.
t1 _ -1*sA*sB.
t2 _ cA*sC + (sA*cB*cC).
t3 _ sA*cB*sC - (cA*cC).
t4 _ cA*sB.
t5 _ sA*sC - (cA*cB*cC).
t6 _ -1*(cA*cB*sC + (sA*cC)).
t7 _ cA*cB.
t8 _ cA*sB*cC.
t9 _ cA*sB*sC.
t10 _ sA*cB.
t11 _ sA*sB*cC.
t12 _ sA*sB*sC.
t18 _ -1*sB*cC.
t19 _ -1*sB*sC.
t20 _ cB*cC.
t21 _ -1*cB*sC.
dLxdPhi _ t4*z + (t5*y) + (t6*x).
dLydPhi _ -1*t1*z - (t2*y) - (t3*x).
dLdPhi _ Array new: 3.
dLdPhi at: 1 put: dLxdPhi; at: 2 put: dLydPhi; at: 3 put: 0.
dLxdThe _ t10*z + (t11*y) + (t12*x).
dLydThe _ -1*t7*z - (t8*y) - (t9*x).
dLzdThe _ -1*sB*z + (t20*y) - (t21*x).
dLdThe _ Array new: 3.
dLdThe at: 1 put: dLxdThe; at: 2 put: dLydThe; at: 3 put: dLzdThe.
dLxdPsi _ t3*y - (t2*x).
dLydPsi _ t6*y - (t5*x).
dLzdPsi _ t19*y - (t18*x).
dLdPsi _ Array new: 3.
dLdPsi at: 1 put: dLxdPsi; at: 2 put: dLydPsi; at: 3 put: dLzdPsi.
forceArrayt _ super transpose: forceArray.
term1 _ super multiply: forceArrayt with: dLdPhi.
term2 _ super multiply: forceArrayt with: dLdThe.
term3 _ super multiply: forceArrayt with: dLdPsi.
torqueArray at: 1 put: term1; at: 2 put: term2; at: 3 put: term3.
forceTorqueArrayJ _ forceArray,torqueArray.
forceTorqueArrayIJ _ forceTorqueArrayI,forceTorqueArrayJ.

```

^ forceTorqueArrayIJ

springForceOnMarkerI

"Solves the spring force acting on marker I and returns the force matrix (3 by 1)."

| lengthArray sum length magnitude forceArray |

sum _ 0.

lengthArray _ (self markerI) relativePositionWith: (self markerJ). lengthArray do: [: elem | sum _ sum + (elem*elem)].

length _ sum sqrt. magnitude _ (self springConstant)*((self freeLength) - length).

forceArray _ lengthArray collect: [: elem | magnitude*elem/length].

^ forceArray

sub41Spring

"returns the matrix [dF/dq] for a translational spring force. This matrix is to be put into the Jacobian submatrix sub41. The matrix has the size of 6 by 12. Note that {F(spring)} = {Q} - {f(spring)}."

| matrixI matrixJ partNoI partNoJ dumMatrix matrixIJ |

matrixI _ (Array new: 6) collect: [: each | Array new: 6 withAll: 0].

matrixJ _ (Array new: 6) collect: [: each | Array new: 6 withAll: 0].

partNoI _ ((self markerI) part) partNo. partNoJ _ ((self markerJ) part) partNo.

(partNoI ~= '00') ifTrue: [matrixI _ self sub41SpringForMarker: (self markerI)].

(partNoJ ~= '00') ifTrue: [

dumMatrix _ self sub41SpringForMarker: (self markerJ). matrixJ _ super multiply: dumMatrix by: (-1)].

matrixIJ _ (Array new: 6) collect: [: each | Array new: 12].

super put: matrixI into: matrixIJ at: (1@1). super put: matrixJ into: matrixIJ at: (7@1).

^ matrixIJ

sub41SpringForMarker: aMarker

| matrix k free lengthArray lengthX lengthY lengthZ sum length lengthArrayt c1 c2 aPart c3 c4 c5 x y z phi the psi
sA sB sC cA cB cC t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t13 t14 t15 t16 t17 t18 t19 t20 t21 dLxdPhi dLxdThe
dLxdPsi dLydPhi dLydThe dLydPsi dLzdThe dLzdPsi dLdPhi dLdThe dLdPsi term1 term2 term3 term4 term5 term6
term7 term8 term9 term10 term11 term12 term13 term14 term15 term16 term17 term18 term19 term20 term21
term22 term23 term24 sub11 sub12 sub13 sub14 sub15 sub16 sub21 sub22 sub23 sub24 sub25 sub26 dum1 dum2
dum3 sub11t sub12t sub13t e1 e2 e3 e4 e5 e6 e7 e8 e9 e10 e11 e12 e13 e14 e15 sum1 sum2 sum3 sum4 sum5 sum6
sum7 sum8 sum9 array1 array2 array3 array4 array5 array6 |

matrix _ (Array new: 6) collect: [: each | Array new: 6 withAll: 0].

k _ self springConstant. free _ self freeLength.

lengthArray _ (self markerI) relativePositionWith: (self markerJ).

lengthX _ lengthArray at: 1. lengthY _ lengthArray at: 2. lengthZ _ lengthArray at: 3.

sum _ 0.

lengthArray do: [: elem | sum _ sum + (elem*elem)]. length _ sum sqrt.

lengthArrayt _ super transpose: lengthArray.

c1 _ k*free/(length**3). c2 _ k*(length-free)/length.

aPart _ aMarker part.

x _ (aMarker position) at: 1. y _ (aMarker position) at: 2. z _ (aMarker position) at: 3.

phi _ ((aPart centerOfMassMarker) position) at: 4.

the _ ((aPart centerOfMassMarker) position) at: 5.

psi _ ((aPart centerOfMassMarker) position) at: 6.

sA _ phi sin. sB _ the sin. sC _ psi sin. cA _ phi cos. cB _ the cos. cC _ psi cos.

t1 _ -1*sA*sB.

t2 _ cA*sC + (sA*cB*cC).

t3 _ sA*cB*sC - (cA*cC).

t4 _ cA*sB.

t5 _ sA*sC - (cA*cB*cC).

t6 _ -1*(cA*cB*sC + (sA*cC)).

t7 _ cA*cB.

t8 _ cA*sB*cC.

t9 _ cA*sB*sC.

t10 _ sA*cB.

t11 _ sA*sB*cC.

t12 _ sA*sB*sC.

t13 _ sA*cB*cC.

t14 _ sA*cB*sC.


```

t15 _ -1*cA*cB*cC.
t16 _ -1*cA*cB*sC.
t17 _ -1*cB.
t18 _ -1*sB*cC.
t19 _ -1*sB*sC.
t20 _ cB*cC.
t21 _ -1*cB*sC.
dLxdPhi _ t4*z + (t5*y) + (t6*x).
dLydPhi _ -1*t1*z - (t2*y) - (t3*x).
dLdPhi _ Array new: 3.
dLdPhi at: 1 put: dLxdPhi; at: 2 put: dLydPhi; at: 3 put: 0.
dLxdThe _ t10*z + (t11*y) + (t12*x).
dLydThe _ -1*t7*z - (t8*y) - (t9*x).
dLzdThe _ -1*sB*z + (t20*y) - (t21*x).
dLdThe _ Array new: 3.
dLdThe at: 1 put: dLxdThe; at: 2 put: dLydThe; at: 3 put: dLzdThe.
dLxdPsi _ t3*y - (t2*x).
dLydPsi _ t6*y - (t5*x).
dLzdPsi _ t19*y - (t18*x).
dLdPsi _ Array new: 3.
dLdPsi at: 1 put: dLxdPsi; at: 2 put: dLydPsi; at: 3 put: dLzdPsi.
" For 1st quadrant "
term1 _ super multiply: lengthArray by: (c1*lengthX). term2 _ Array new: 3.
term2 at: 1 put: c2; at: 2 put: 0; at: 3 put: 0.
sub11 _ super add: term1 with: term2.
term3 _ super multiply: lengthArray by: (c1*lengthY). term4 _ Array new: 3.
term4 at: 1 put: 0; at: 2 put: c2; at: 3 put: 0.
sub12 _ super add: term3 with: term4.
term5 _ super multiply: lengthArray by: (c1*lengthZ). term6 _ Array new: 3.
term6 at: 1 put: 0; at: 2 put: 0; at: 3 put: c2.
sub13 _ super add: term5 with: term6.
" For 2nd quadrant "
dum1 _ super multiply: lengthArrayt with: dLdPhi.
c3 _ c1*dum1.
term1 _ super multiply: lengthArray by: c3. term2 _ super multiply: dLdPhi by: c2.
sub14 _ super add: term1 with: term2.
dum2 _ super multiply: lengthArrayt with: dLdThe.
c4 _ c1*dum2.
term3 _ super multiply: lengthArray by: c4. term4 _ super multiply: dLdThe by: c2.
sub15 _ super add: term3 with: term4.
dum3 _ super multiply: lengthArrayt with: dLdPsi.
c5 _ c1*dum3.
term5 _ super multiply: lengthArray by: c5. term6 _ super multiply: dLdPsi by: c2.
sub16 _ super add: term5 with: term6.
" For 3rd quadrant "
sub11t _ super transpose: sub11.
term1 _ super multiply: sub11t with: dLdPhi.
term2 _ super multiply: sub11t with: dLdThe.
term3 _ super multiply: sub11t with: dLdPsi.
sub21 _ Array new: 3. sub21 at: 1 put: term1; at: 2 put: term2; at: 3 put: term3.
sub12t _ super transpose: sub12.
term4 _ super multiply: sub12t with: dLdPhi.
term5 _ super multiply: sub12t with: dLdThe.
term6 _ super multiply: sub12t with: dLdPsi.
sub22 _ Array new: 3. sub22 at: 1 put: term4; at: 2 put: term5; at: 3 put: term6.
sub13t _ super transpose: sub13.
term7 _ super multiply: sub13t with: dLdPhi.
term8 _ super multiply: sub13t with: dLdThe.
term9 _ super multiply: sub13t with: dLdPsi.
sub23 _ Array new: 3. sub23 at: 1 put: term7; at: 2 put: term8; at: 3 put: term9.
" For 4th quadrant "
term1 _ c3*dum1.

```

```

term2 _ c3*dum2.
term3 _ c3*dum3.
term4 _ c2*(super multiply: (super transpose: dLdPhi) with: dLdPhi).
term5 _ c2*(super multiply: (super transpose: dLdPhi) with: dLdThe).
term6 _ c2*(super multiply: (super transpose: dLdPhi) with: dLdPsi).
e1 _ t1*z + (t2*y) + (t3*x).
e2 _ t4*z + (t5*y) + (t6*x).
array1 _ Array new: 3.
array1 at: 1 put: e1; at: 2 put: e2; at: 3 put: 0.
term7 _ c2*(super multiply: lengthArrayt with: array1).
e3 _ t7*z + (t8*y) + (t9*x).
e4 _ t10*z + (t11*y) + (t12*x).
array2 _ Array new: 3.
array2 at: 1 put: e3; at: 2 put: e4; at: 3 put: 0.
term8 _ c2*(super multiply: lengthArrayt with: array2).
e5 _ -1*t6*y + (t5*x).
e6 _ t3*y - (t2*x).
array3 _ Array new: 3.
array3 at: 1 put: e5; at: 2 put: e6; at: 3 put: 0.
term9 _ c2*(super multiply: lengthArrayt with: array3).
sum1 _ term1 + term4 + term7.
sum2 _ term2 + term5 + term8.
sum3 _ term3 + term6 + term9.
sub24 _ Array new: 3.
sub24 at: 1 put: sum1; at: 2 put: sum2; at: 3 put: sum3.
term10 _ c4*dum1.
term11 _ c4*dum2.
term12 _ c4*dum3.
term13 _ c2*(super multiply: (super transpose: dLdThe) with: dLdPhi).
term14 _ c2*(super multiply: (super transpose: dLdThe) with: dLdThe).
term15 _ c2*(super multiply: (super transpose: dLdThe) with: dLdPsi).
e7 _ t1*z + (t13*y) + (t14*x).
e8 _ t4*z + (t15*y) + (t16*x).
e9 _ t17*z + (t18*y) + (t19*x).
array4 _ Array new: 3.
array4 at: 1 put: e7; at: 2 put: e8; at: 3 put: e9.
term16 _ c2*(super multiply: lengthArrayt with: array4).
e10 _ -1*t12*y + (t11*x).
e11 _ t9*y - (t8*x).
e12 _ t21*y + (t20*x).
array5 _ Array new: 3.
array5 at: 1 put: e10; at: 2 put: e11; at: 3 put: e12.
term17 _ c2*(super multiply: lengthArrayt with: array5).
sum4 _ term10 + term13 + term8.
sum5 _ term11 + term14 + term16.
sum6 _ term12 + term15 + term17.
sub25 _ Array new: 3.
sub25 at: 1 put: sum4; at: 2 put: sum5; at: 3 put: sum6.
term18 _ c5*dum1.
term19 _ c5*dum2.
term20 _ c5*dum3.
term21 _ c2*(super multiply: (super transpose: dLdPsi) with: dLdPhi).
term22 _ c2*(super multiply: (super transpose: dLdPsi) with: dLdThe).
term23 _ c2*(super multiply: (super transpose: dLdPsi) with: dLdPsi).
e13 _ t2*y + (t3*x).
e14 _ t5*y + (t6*x).
e15 _ t18*y + (t19*x).
array6 _ Array new: 3.
array6 at: 1 put: e13; at: 2 put: e14; at: 3 put: e15.
term24 _ c2*(super multiply: lengthArrayt with: array6).
sum7 _ term18 + term21 + term9.
sum8 _ term19 + term22 + term17.

```

```

sum9 _ term20 + term23 + term24.
sub26 _ Array new: 3.
sub26 at: 1 put: sum7; at: 2 put: sum8; at: 3 put: sum9.
super put: sub11 into: matrix at: (1@1). super put: sub12 into: matrix at: (2@1).
super put: sub13 into: matrix at: (3@1). super put: sub14 into: matrix at: (4@1).
super put: sub15 into: matrix at: (5@1). super put: sub16 into: matrix at: (6@1).
super put: sub21 into: matrix at: (1@4). super put: sub22 into: matrix at: (2@4).
super put: sub23 into: matrix at: (3@4). super put: sub24 into: matrix at: (4@4).
super put: sub25 into: matrix at: (5@4). super put: sub26 into: matrix at: (6@4).
^ matrix

```

B.3 Request Class

System subclass: #Request

```

instanceVariableNames: 'type part fileStream '
classVariableNames: ''
poolDictionaries: ''
category: 'Multibody System'

```

closeOutput

"closes the output files."

```

1 to: NumberOfRequests do: [: whichRequest | | aRequest outputFile |
    aRequest _ RequestDictionary at: whichRequest. outputFile _ aRequest fileStream. outputFile flush close].

```

openOutput

"files out the numerical data onto the specified files."

```

1 to: NumberOfRequests do: [: whichRequest | | aRequest aType aPart outputFile array dGdqtLambda where phi the
psi sA sB sC cA cB cC a11 a12 a13 a21 a22 a23 a31 a32 a33 b11 b12 b21 b22 b31 ab11 ab12 ab13 ab21 ab22
ab23 ab31 ab32 ab33 matrixAB dum |

```

```

    aRequest _ RequestDictionary at: whichRequest. aType _ aRequest type.

```

```

    aPart _ aRequest part. outputFile _ aRequest fileStream.

```

```

    (aType = 'lambda')

```

```

        ifTrue: [1 to: NumberOfConstraints do: [: i | outputFile nextPutAll: (Lambda at: i) printString; tab]]

```

```

        ifFalse: [

```

```

            (aType = 'position') ifTrue: [ array _ (aPart centerOfMassMarker) position].

```

```

            (aType = 'velocity') ifTrue: [ array _ aPart velocity].

```

```

            (aType = 'acceleration') ifTrue: [ array _ aPart acceleration].

```

```

            (aType = 'momentum') ifTrue: [ array _ aPart stateVariableu].

```

```

            (aType = 'momentumDot') ifTrue: [ array _ aPart stateVariableudot].

```

```

            (aType = 'force') ifTrue: [ array _ aPart generalizedForce].

```

```

            (aType = 'constraintForce') ifTrue: [

```

```

                dGdqtLambda _ super multiply: (super transpose: MatrixdGdq) with: Lambda.

```

```

                where _ 6*((aPart partNo) asNumber) - 1.

```

```

                array _ super partition: dGdqtLambda from: (1@(where+1)) extension: (1@6).

```

```

                phi _ ((aPart centerOfMassMarker) position) at: 4.

```

```

                the _ ((aPart centerOfMassMarker) position) at: 5.

```

```

                psi _ ((aPart centerOfMassMarker) position) at: 6.

```

```

                sA _ phi sin. sB _ the sin. sC _ psi sin. cA _ phi cos. cB _ the cos. cC _ psi cos.

```

```

                a11 _ cA*cC - (sA*cB*sC). a12 _ -1*cA*sC - (sA*cB*cC). a13 _ sA*sB.

```

```

                a21 _ sA*cC + (cA*cB*sC). a22 _ -1*sA*sC + (cA*cB*cC). a23 _ -1*cA*sB.

```

```

                a31 _ sB*sC. a32 _ sB*cC. a33 _ cB.

```

```

                b11 _ sB*sC. b12 _ cC.

```

```

                b21 _ sB*cC. b22 _ -1*sC.

```

```

                b31 _ cB.

```

```

                ab11 _ a11*b11 + (a12*b21) + (a13*b31). ab12 _ a11*b12 + (a12*b22). ab13 _ a13.

```

```

                ab21 _ a21*b11 + (a22*b21) + (a23*b31). ab22 _ a21*b12 + (a22*b22). ab23 _ a23.

```

```

                ab31 _ a31*b11 + (a32*b21) + (a33*b31). ab32 _ a31*b12 + (a32*b22). ab33 _ a33.

```

```

                matrixAB _ (Array new: 3) collect: [: each | Array new: 3].

```

```

                (matrixAB at: 1) at: 1 put: ab11; at: 2 put: ab12; at: 3 put: ab13.

```

```

(matrixAB at: 2) at: 1 put: ab21; at: 2 put: ab22; at: 3 put: ab23.
(matrixAB at: 3) at: 1 put: ab31; at: 2 put: ab32; at: 3 put: ab33.
dum _ super multiply: matrixAB with: (super partition: array from: (1@4) extension: (1@3)).
array _ super put: dum into: array at: (1@4)].
1 to: 6 do: [: i | outputFile nextPutAll: (array at: i) printString; tab]].
outputFile cr].

```

request: aString1 of: aPart onto: aString2

"initializes a request and stores the request in the class variable RequestDictionary."

type _ aString1. part _ aPart. fileStream _ aString2 asFilename writeStream.

NumberOfRequests _ NumberOfRequests + 1. RequestDictionary add: self.

request: aString1 onto: aString2

"initializes a request and stores the request in the class variable RequestDictionary. Here, the request is for Lagrange multipliers."

type _ aString1. fileStream _ aString2 asFilename writeStream.

NumberOfRequests _ NumberOfRequests + 1. RequestDictionary add: self.

fileStream

"returns the output-file stream."

^ fileStream

part

"returns the instance of the part to which a request is related."

^ part

type

"returns the type of a request."

^ type

B.4 Part Class

System subclass: #Part

instanceVariableNames: 'mass partNo momentOfInertia centerOfMassMarker velocity stateVariableu
stateVariableudot generalizedForce stateVariableQ acceleration initialPos initialVel positionWeights
velocityWeights positionArray stateVariableuArray positionArrayDum stateVariableuArrayDum '

classVariableNames: "

poolDictionaries: "

category: 'Multibody System'

acceleration

"Returns the acceleration of the part;

i.e, $\{R\} = [X\ddot{} Y\ddot{} Z\ddot{} \Phi\ddot{} \Theta\ddot{} \Psi\ddot{}]t$."

^ acceleration

accelerationArray

"Returns the acceleration array which contains a series of accelerations of the part in time domain;

$\{q\ddot{}\} = [X\ddot{} Y\ddot{} Z\ddot{} \Phi\ddot{} \Theta\ddot{} \Psi\ddot{}]$ "

^ accelerationArray

centerOfMassMarker

"Returns the internal center-of-mass marker instance of a part."

^ centerOfMassMarker

generalizedForce

"Returns the generalized forces of the part."

^ generalizedForce

initialPos

"Returns the initial position of a part."

^ initialPos

initialVel

"Returns the initial velocity of a part."

^ initialVel

mass

"Returns the mass of a part."

^ mass

momentOfInertia

"Returns the principal moment of inertia of a part."

^ momentOfInertia

partNo

"Returns the number of a part."

^ partNo

positionArray

"Returns the position array which contains a series of positions of the part in time domain;

{q} = [X Y Z Phi Theta Psi] "

^ positionArray

positionArrayDum

"Returns the position array {q} which contains the values in the first two time steps."

^ positionArrayDum

positionWeights

"Returns the array positionWeights of the part"

^ positionWeights

stateVariableQ

"Returns the state variable {Q} of the part;

i.e, {Q} = [QX QY QZ QPhi QTheta QPsi]t."

^ stateVariableQ

stateVariableu

"Returns the state variable {u} of the part; i.e, {u} = [uX uY uZ uPhi uTheta uPsi]t."

^ stateVariableu

stateVariableuArray

"Returns the state variable array {u} which contains a series of state variables of the part in time domain;

{u} = [uX uY uZ uPhi uTheta uPsi]"

^ stateVariableuArray

stateVariableuArrayDum

"Returns the state variable array {u} which contains the values in the first two time steps."

^ stateVariableuArrayDum

stateVariableudot

"Returns the array {udot} of the part; i.e, {udot} = [udotX udotY udotZ udotPhi udotTheta udotPsi]t."

^ stateVariableudot

stateVariableudotArray

"Returns the array {udot} which contains a series of values in the time domain;

i.e, {udot} = [udotX udotY udotZ udotPhi udotTheta udotPsi]t."

^ stateVariableudotArray

velocity

"Returns the velocity of the part; i.e, {R} = [Xdot Ydot Zdot wx wy wz]t before IC evaluation.

{R} = [Xdot Ydot Zdot Phidot Thetadot Psidot]t after IC evaluation."

^ velocity

velocityArray

"Returns the velocity array which contains a series of velocities of the part in time domain;
{qdot} = [Xdot Ydot Zdot Phidot Thetadot Psidot] "

^ velocityArray

velocityWeights

"Returns the array velocityWeights of the part"

^ velocityWeights

acceleration: anArray

"updates the acceleration of the part. The accelerations are in Xdotdot Ydotdot Zdotdot Phidotdot Thetadotdot Psidotdot."

acceleration _ anArray

generalizedForce: anArray

"updates the generalized forces of the part."

generalizedForce _ anArray

initializeGround

"initializes the ground as the inertial frame of the dynamic system. Note that the ground is not stored in the class variable PartDictionary. The ground has an imaginary c.m. at the origin."

partNo _ '00'.

centerOfMassMarker _ (Marker new) initializeCMMarkerOn: self

markerNo: '0000' position: #(0 0 0 0 0).

initializePartNo: string1 mass: m momentOfInertia: arrayJ velocity: velArray

centerOfMassMarkerNo: string2 centerOfMassMarkerAt: posArray

"Initializes a part by using the the arguments. The undefined initial positions and velocities of the part are checked and assigned weight factors. But, the the positions of markers are must be defined in the input.

The translational velocity components are expressed in the ground frame while the rotational velocity components are expressed in the principal directions. Note that the all numbers are in double precision."

| arrayW1 arrayW2 arrayW arrayVr inverseB the psi a11 a12 a21 a22 a31 a32 a33 |

"----- Weight Factors -----"

arrayW1 _ Array new: 6.

1 to: 6 do: [: i | (posArray at: i) = nil

ifTrue: [arrayW1 at: i put: (1 asGoodDouble). posArray at: i put: (0 asGoodDouble)]

ifFalse: [arrayW1 at: i put: (1000 asGoodDouble)]]].

positionWeights _ arrayW1.

arrayW2 _ Array new: 6.

1 to: 6 do: [: i | (velArray at: i) = nil

ifTrue: [arrayW2 at: i put: (1 asGoodDouble). velArray at: i put: (0 asGoodDouble)]

ifFalse: [arrayW2 at: i put: (1000 asGoodDouble)]]].

velocityWeights _ arrayW2.

1 to: 3 do: [: i |

posArray at: i put: ((posArray at: i) asGoodDouble).

posArray at: (i+3) put: (((posArray at: (i+3)) degreesToRadians) asGoodDouble)].

initialPos _ posArray.

partNo _ string1.

mass _ m asGoodDouble.

1 to: 3 do: [: i | arrayJ at: i put: ((arrayJ at: i) asGoodDouble)].

momentOfInertia _ arrayJ.

"expresses the angular velocity in PhiDot, ThetaDot, and PsiDot; i.e, {Vr} = [B]-1 {W}."

arrayW _ super partition: velArray from: (1@4) extension: (1@3).

the _ (posArray at: 5) asGoodDouble. psi _ (posArray at: 6) asGoodDouble.

a11 _ (psi sin)/(the sin). a12 _ (psi cos)/(the sin).

a21 _ psi cos. a22 _ -1*(psi sin).

a31 _ -1*(the cos)*(psi sin)/(the sin). a32 _ -1*(the cos)*(psi cos)/(the sin). a33 _ 1.

inverseB _ (Array new: 3) collect: [: each | Array new: 3 withAll: 0].

(inverseB at: 1) at: 1 put: a11; at: 2 put: a12.

(inverseB at: 2) at: 1 put: a21; at: 2 put: a22.

```

(inverseB at: 3) at: 1 put: a31; at: 2 put: a32; at: 3 put: a33.
arrayVr _ super multiply: inverseB with: arrayW.
1 to: 3 do: [: i | velArray at: i put: ((velArray at: i) asGoodDouble). velArray at: (i+3) put: (arrayVr at: i)].
initialVel _ velArray.
velocity _ velArray.
centerOfMassMarker _ (Marker new) initializeCMMarkerOn: self markerNo: string2 position: posArray.
"prepares the spaces for past values in the initial integration set-up."
positionArrayDum _ OrderedCollection new.
stateVariableuArrayDum _ OrderedCollection new.
"prepares the spaces for past values in the regular integration."
positionArray _ OrderedCollection new.
stateVariableuArray _ OrderedCollection new.
NumberOfParts _ NumberOfParts + 1.
PartDictionary add: self

```

```

position: anArray
"updates the position of the center-of-mass of the part."
(self centerOfMassMarker) position: anArray.

```

```

stateVariableQ: anArray
"assigns the evaluated {Q} to the part."
stateVariableQ _ anArray

```

```

stateVariableu: anArray
"assigns the evaluated {u} to the part."
stateVariableu _ anArray

```

```

stateVariableudot: anArray
"assigns the evaluated {udot} to the part."
stateVariableudot _ anArray

```

```

velocity: anArray
"updates the velocity of the part."
velocity _ anArray

```

sub11Inertia1

"returns the matrix - [d(dT/dq)/dq] (6 by 6) of a part. The matrix is symmetric. The matrix will be put into the matrix <Lambda>[d(dG/dq)/dq] to set the sub11 matrix [dM/dq].

```

a55 = - d(dT/dTheta)/dTheta = - { Vr } t (d2[B]/dTheta2) [J] [B] { Vr } - { Vr } t (d[B]/dTheta) [J] (d[B]/dTheta) { Vr }
a56 = - d(dT/dTheta)/dPsi = - { Vr } t (d2[B]/dPsiTheta) [J] [B] { Vr } - { Vr } t (d[B]/dTheta) [J] (d[B]/dPsi) { Vr }
a66 = - d(dT/dPsi)/dPsi = - { Vr } t (d2[B]/dPsi2) [J] [B] { Vr } - { Vr } t (d[B]/dPsi) [J] (d[B]/dPsi) { Vr }
a65 = a56.

```

And, the rest of the elements are zero. It is noted that {Vr} is in Xdot, Ydot, Zdot, Phidot, Thetadot, Psidot."

```

| jxx jyy jzz theta psi phiDot thetaDot psiDot sB sC cB cC sB2 sC2 cB2 cC2 term1 term2 term3 term4 term5 term6
term7 term8 term9 a55 a56 a66 matrixA |

```

```

jxx _ (self momentOfInertia) at: 1. jyy _ (self momentOfInertia) at: 2. jzz _ (self momentOfInertia) at: 3.
theta _ ((self centerOfMassMarker) position) at: 5. psi _ ((self centerOfMassMarker) position) at: 6.
phiDot _ (self velocity) at: 4. thetaDot _ (self velocity) at: 5. psiDot _ (self velocity) at: 6.
sB _ theta sin. sC _ psi sin. cB _ theta cos. cC _ psi cos. sB2 _ sB*sB. sC2 _ sC*sC. cB2 _ cB*cB. cC2 _ cC*cC.
term1 _ -1*phiDot*sB2*sC2 + (phiDot*cB2*sC2) - (thetaDot*sB*sC*cC).
term2 _ -1*phiDot*sB2*cC2 + (phiDot*cB2*cC2) + (thetaDot*sB*sC*cC).
term3 _ -1*phiDot*cB2 + (phiDot*sB2) - (psiDot*cB).
a55 _ -1*phiDot*(jxx*term1 + (jyy*term2) + (jzz*term3)).
term4 _ 2*phiDot*sB*cB*sC*cC + (thetaDot*cB*cC2) - (thetaDot*cB*sC2).
term5 _ -2*phiDot*sB*cB*sC*cC + (thetaDot*cB*sC2) - (thetaDot*cB*cC2).
a56 _ -1*phiDot*(jxx*term4 + (jyy*term5)).
term6 _ -1*phiDot*sB2*sC2 + (phiDot*sB2*cC2) - (2*thetaDot*sB*sC*cC).
term7 _ -1*phiDot*sB2*cC2 + (phiDot*sB2*sC2) + (2*thetaDot*sB*sC*cC).
term8 _ -2*phiDot*sB*sC*cC - (thetaDot*cC2) + (thetaDot*sC2).
term9 _ 2*phiDot*sB*sC*cC - (thetaDot*sC2) + (thetaDot*cC2).
a66 _ -1*phiDot*(jxx*term6 + (jyy*term7)) - (thetaDot*(jxx*term8 + (jyy*term9))).
matrixA _ (Array new: 6) collect: [: each | Array new: 6 withAll: 0].

```

(matrixA at: 5) at: 5 put: a55. (matrixA at: 5) at: 6 put: a56. (matrixA at: 6) at: 5 put: a56.
(matrixA at: 6) at: 6 put: a66.
^ matrixA

sub11Inertia2

"returns the matrix $[dM/dqdot] = -[d(dT/dq)/dqdot]$ (6 by 6) of a part.

Note that this matrix is not symmetric.

a54 = - [1 0 0]t (d[B]/dTheta) [J] [B] {Vr} - {Vr}t (d[B]/dTheta) [J] [B] [1 0 0]t.

a55 = - [0 1 0]t (d[B]/dTheta) [J] [B] {Vr} - {Vr}t (d[B]/dTheta) [J] [B] [0 1 0]t.

a56 = - [0 0 1]t (d[B]/dTheta) [J] [B] {Vr} - {Vr}t (d[B]/dTheta) [J] [B] [0 0 1]t.

a64 = - [1 0 0]t (d[B]/dPsi) [J] [B] {Vr} - {Vr}t (d[B]/dPsi) [J] [B] [1 0 0]t.

a65 = - [0 1 0]t (d[B]/dPsi) [J] [B] {Vr} - {Vr}t (d[B]/dPsi) [J] [B] [0 1 0]t.

a66 = - [0 0 1]t (d[B]/dPsi) [J] [B] {Vr} - {Vr}t (d[B]/dPsi) [J] [B] [0 0 1]t."

| jxx jyy jzz theta psi phiDot thetaDot psiDot sB sC cB cC sB2 sC2 cB2 cC2 term1 term2 term3 a54 a55 a56 a64 a65 matrixA |

jxx _ (self momentOfInertia) at: 1. jyy _ (self momentOfInertia) at: 2. jzz _ (self momentOfInertia) at: 3.

theta _ ((self centerOfMassMarker) position) at: 5. psi _ ((self centerOfMassMarker) position) at: 6.

phiDot _ (self velocity) at: 4. thetaDot _ (self velocity) at: 5. psiDot _ (self velocity) at: 6.

sB _ theta sin. sC _ psi sin. cB _ theta cos. cC _ psi cos.

sB2 _ sB*sB. sC2 _ sC*sC. cB2 _ cB*cB. cC2 _ cC*cC.

term1 _ jxx*sB*cB*sC2 + (jyy*sB*cB*cC2) - (jzz*sB*cB). term2 _ (jxx - jyy)*cB*sC*cC. term3 _ jzz*sB.

a54 _ -2*phiDot*term1 - (thetaDot*term2) + (psiDot*term3).

a55 _ -1*phiDot*(jxx - jyy)*cB*sC*cC.

a56 _ phiDot*jzz*sB.

a64 _ -2*phiDot*(jxx - jyy)*sB2*sC*cC - (thetaDot*(jxx - jyy)*sB*cC2).

a65 _ -1*phiDot*(jxx - jyy)*sB*cC2 - (2*thetaDot*(jyy - jxx)*sC*cC).

matrixA _ (Array new: 6) collect: [: each | Array new: 6 withAll: 0].

(matrixA at: 5) at: 4 put: a54. (matrixA at: 5) at: 5 put: a55. (matrixA at: 5) at: 6 put: a56.

(matrixA at: 6) at: 4 put: a64. (matrixA at: 6) at: 5 put: a65.

^ matrixA

Inertia1

"returns the matrix $[d[B]/dt] [J] [B] \{Vr\} + [B]t [J] [d[B]/dt] \{Vr\}$.

{Vr} is in Phidot, Thetadot, Psidot. The size of the matrix is 3 by 1."

| jxx jyy jzz theta psi phiDot thetaDot psiDot sB sC cB cC sC2 cC2 term1 term2 term3 term4 term5 a1 a2 b1 b2 b3 c1 c2 c3 arrayC |

jxx _ (self momentOfInertia) at: 1. jyy _ (self momentOfInertia) at: 2. jzz _ (self momentOfInertia) at: 3.

theta _ ((self centerOfMassMarker) position) at: 5. psi _ ((self centerOfMassMarker) position) at: 6.

phiDot _ (self velocity) at: 4. thetaDot _ (self velocity) at: 5. psiDot _ (self velocity) at: 6.

sB _ theta sin. sC _ psi sin. cB _ theta cos. cC _ psi cos. sC2 _ sC*sC. cC2 _ cC*cC.

term1 _ jxx*sB*sC*(thetaDot*cB*sC + (psiDot*sB*cC)) + (jyy*sB*cC*(thetaDot*cB*cC - (psiDot*sB*sC))) - (jzz*thetaDot*sB*cB).

term2 _ jxx*cC*(thetaDot*cB*sC + (psiDot*sB*cC)) - (jyy*sC*(thetaDot*cB*cC - (psiDot*sB*sC))).

term3 _ -1*jzz*thetaDot*sB.

term4 _ -1*jxx*psiDot*sB*sC2 - (jyy*psiDot*sB*cC2).

term5 _ -1*jxx*psiDot*sC*cC + (jyy*psiDot*sC*cC).

a1 _ term1*phiDot + (term2*thetaDot) + (term3*psiDot). a2 _ term4*phiDot + (term5*thetaDot).

b1 _ term1*phiDot + (term4*thetaDot). b2 _ term2*phiDot + (term5*thetaDot). b3 _ term3*phiDot.

c1 _ a1 + b1. c2 _ a2 + b2. c3 _ b3.

arrayC _ Array new: 3. arrayC at: 1 put: c1. arrayC at: 2 put: c2. arrayC at: 3 put: c3.

^ arrayC

Inertia2

"returns the matrix $[B]t [J] [B]$. The size of the matrix is 3 by 3."

| jxx jyy jzz theta psi sB sC cB cC sB2 sC2 cB2 cC2 a11 a12 a13 a22 a33 matrixA |

jxx _ (self momentOfInertia) at: 1. jyy _ (self momentOfInertia) at: 2. jzz _ (self momentOfInertia) at: 3.

theta _ ((self centerOfMassMarker) position) at: 5. psi _ ((self centerOfMassMarker) position) at: 6.

sB _ theta sin. sC _ psi sin. cB _ theta cos. cC _ psi cos. sB2 _ sB*sB. sC2 _ sC*sC. cB2 _ cB*cB. cC2 _ cC*cC.

a11 _ jxx*sB2*sC2 + (jyy*sB2*cC2) + (jzz*cB2). a12 _ jxx*sB*sC*cC - (jyy*sB*sC*cC). a13 _ jzz*cB.

a22 _ jxx*cC2 + (jyy*sC2).

a33 _ jzz.

matrixA _ (Array new: 3) collect: [: each | Array new: 3 withAll: 0].


```
(matrixA at: 1) at: 1 put: a11. (matrixA at: 1) at: 2 put: a12. (matrixA at: 1) at: 3 put: a13.
(matrixA at: 2) at: 2 put: a22. (matrixA at: 3) at: 3 put: a33.
"symmetric"
(matrixA at: 2) at: 1 put: a12. (matrixA at: 3) at: 1 put: a13.
^ matrixA
```

inertia3

```
"returns the matrix {dT/dq}. The size of the matrix is 3 by 1; i.e. dT/dq = {Vr}t d([B]t)/dq [J] [B] {Vr}
It is noted that {Vr} is in Phidot, Thetadot, Psidot."
| jxx jyy jzz theta psi phiDot thetaDot psiDot sB sC cB cC term1 term2 term3 term4 term5 a2 a3 arrayA |
jxx _ (self momentOfInertia) at: 1. jyy _ (self momentOfInertia) at: 2. jzz _ (self momentOfInertia) at: 3.
theta _ ((self centerOfMassMarker) position) at: 5. psi _ ((self centerOfMassMarker) position) at: 6.
phiDot _ (self velocity) at: 4. thetaDot _ (self velocity) at: 5. psiDot _ (self velocity) at: 6.
sB _ theta sin. sC _ psi sin. cB _ theta cos. cC _ psi cos.
term1 _ jxx*(phiDot*sB*sC + (thetaDot*cC))*phiDot*cB*sC.
term2 _ jyy*(phiDot*sB*cC - (thetaDot*sC))*phiDot*cB*cC.
term3 _ -1*jzz*(phiDot*cB + psiDot)*phiDot*sB.
a2 _ term1 + term2 + term3.
term4 _ jxx*(phiDot*sB*sC + (thetaDot*cC))*(phiDot*sB*cC - (thetaDot*sC)).
term5 _ -1*jyy*(phiDot*sB*cC - (thetaDot*sC))*(phiDot*sB*sC + (thetaDot*cC)).
a3 _ term4 + term5.
arrayA _ Array new: 3. arrayA at: 1 put: 0. arrayA at: 2 put: a2. arrayA at: 3 put: a3.
^ arrayA
```

partEOMFunction

```
"returns the submatrix {Mi} of the part for numerical integration
i.e. {M}i = {u}i - [dT/dq]t - {Q}i + [dG/dq]t {Lambda} = {A} - {B} - {C} + {D}.
The last term is considered globally in the 'systemEOMFunction'."
| arrayA arrayB arrayC dumArray arrayMi |
arrayA _ self stateVariableudot. arrayB _ Array new: 6 withAll: 0. super put: (self inertia3) into: arrayB at: (1@4).
arrayC _ self stateVariableQ. dumArray _ super subtract: arrayA with: arrayB.
arrayMi _ super subtract: dumArray with: arrayC.
^ arrayMi
```

partMomentumFunction

```
"returns the submatrix {Pi} of the part for numerical integration; i.e. {P}i = {u}i - [M]{qdot}i "
| arrayA arrayB arrayPi |
arrayA _ self stateVariableu. arrayB _ super multiply: (self toMomentum) with: (self velocity).
arrayPi _ super subtract: arrayA with: arrayB.
^ arrayPi
```

toMomentum

```
"returns the transformation matrix[M] from the velocity to the momentum. The size of the matrix is 6 by 6"
| matrix1 matrix2 matrixMi |
matrix1 _ super multiply: (super identityMatrix: 3) by: (self mass). matrix2 _ self inertia2.
matrixMi _ (Array new: 6) collect: [: each | Array new: 6 withAll: 0].
super put: matrix1 into: matrixMi at: (1@1). super put: matrix2 into: matrixMi at: (4@4).
^ matrixMi
```

B.5 Marker Class

Part subclass: #Marker

```
instanceVariableNames: 'part markerNo position transformationMatrix '
classVariableNames: "
poolDictionaries: "
category: 'Multibody System'
```

fromCenterToGround

"returns the transformation matrix from the center-of-mass to the ground frame. This method is used only for the center-of-mass markers."

```
| phi the psi sA sB sC cA cB cC a11 a12 a13 a21 a22 a23 a31 a32 a33 matrixA |
phi _ (self position) at: 4. the _ (self position) at: 5. psi _ (self position) at: 6.
sA _ phi sin. sB _ the sin. sC _ psi sin. cA _ phi cos. cB _ the cos. cC _ psi cos.
a11 _ cA*cC - (sA*cB*sC). a12 _ -1*cA*sC - (sA*cB*cC). a13 _ sA*sB.
a21 _ sA*cC + (cA*cB*sC). a22 _ -1*sA*sC + (cA*cB*cC). a23 _ -1*cA*sB.
a31 _ sB*sC. a32 _ sB*cC. a33 _ cB.
matrixA _ (Array new: 3) collect: [: each | Array new: 3].
(matrixA at: 1) at: 1 put: a11; at: 2 put: a12; at: 3 put: a13.
(matrixA at: 2) at: 1 put: a21; at: 2 put: a22; at: 3 put: a23.
(matrixA at: 3) at: 1 put: a31; at: 2 put: a32; at: 3 put: a33.
^ matrixA
```

markerNo

"Returns a marker number. The first two digits are the part number where the marker exists."

^ markerNo

part

"Returns the part on which the marker exists."

^ part

position

"Returns the position of a marker; If the marker represents a part frame, {R}ground = [X Y Z PHI THETA PSI]t. If the marker represents a marker frame, {r}part = [x y z phi theta psi]t."

^ position

relativePositionWith: aMarker

"returns the relative position of the receiver with respect to aMarker. i.e, {Lij} = {Xi} + [Ai]{xi} - {Xj} - [Aj]{xj}."

```
| partI partJ posI posJ cmXI cmXJ markerXI markerXJ term1 term2 arrayL phi the psi sA sB sC cA cB cC a11 a12
a13 a21 a22 a23 a31 a32 a33 matrixA |
```

"----- For Marker I -----"

partI _ self part. posI _ (partI centerOfMassMarker) position.

cmXI _ super partition: posI from: (1@1) extension: (1@3).

markerXI _ super partition: (self position) from: (1@1) extension: (1@3).

phi _ posI at: 4. the _ posI at: 5. psi _ posI at: 6.

sA _ phi sin. sB _ the sin. sC _ psi sin. cA _ phi cos. cB _ the cos. cC _ psi cos.

a11 _ cA*cC - (sA*cB*sC). a12 _ -1*cA*sC - (sA*cB*cC). a13 _ sA*sB.

a21 _ sA*cC + (cA*cB*sC). a22 _ -1*sA*sC + (cA*cB*cC). a23 _ -1*cA*sB.

a31 _ sB*sC. a32 _ sB*cC. a33 _ cB.

matrixA _ (Array new: 3) collect: [: each | Array new: 3].

(matrixA at: 1) at: 1 put: a11; at: 2 put: a12; at: 3 put: a13.

(matrixA at: 2) at: 1 put: a21; at: 2 put: a22; at: 3 put: a23.

(matrixA at: 3) at: 1 put: a31; at: 2 put: a32; at: 3 put: a33.

term1 _ super add: cmXI with: (super multiply: matrixA with: markerXI).

"----- For Marker J -----"

partJ _ aMarker part. posJ _ (partJ centerOfMassMarker) position.

cmXJ _ super partition: posJ from: (1@1) extension: (1@3).

markerXJ _ super partition: (aMarker position) from: (1@1) extension: (1@3).

phi _ posJ at: 4. the _ posJ at: 5. psi _ posJ at: 6.

sA _ phi sin. sB _ the sin. sC _ psi sin. cA _ phi cos. cB _ the cos. cC _ psi cos.

a11 _ cA*cC - (sA*cB*sC). a12 _ -1*cA*sC - (sA*cB*cC). a13 _ sA*sB.

a21 _ sA*cC + (cA*cB*sC). a22 _ -1*sA*sC + (cA*cB*cC). a23 _ -1*cA*sB.

a31 _ sB*sC. a32 _ sB*cC. a33 _ cB.

matrixA _ (Array new: 3) collect: [: each | Array new: 3].

(matrixA at: 1) at: 1 put: a11; at: 2 put: a12; at: 3 put: a13.

(matrixA at: 2) at: 1 put: a21; at: 2 put: a22; at: 3 put: a23.

(matrixA at: 3) at: 1 put: a31; at: 2 put: a32; at: 3 put: a33.

term2 _ super add: cmXJ with: (super multiply: matrixA with: markerXJ).

arrayL _ super subtract: term1 with: term2.

^ arrayL

transformationMatrix

"returns the transformation matrix from the marker frame to the part frame. This method is used for the markers besides the center-of-mass markers."

^ transformationMatrix

InitializeCMMarkerOn: aPart markerNo: aString position: posArray

"Initializes a center-of-mass marker using the arguments. Note that posArray is in double precision and the rotational components are in radian already."

part _ aPart. markerNo _ aString. position _ posArray

InitializeMarkerOn: aPart markerNo: aString position: posArray

"Initializes a marker using the arguments. Note that a ground marker is initialized using this method also."

! phi theta psi sA sB sC cA cB cC a11 a12 a13 a21 a22 a23 a31 a32 a33 matrixA !

part _ aPart.

markerNo _ aString.

(1 to: 3) do: [: i |

posArray at: i put: ((posArray at: i) asGoodDouble).

posArray at: (i+3) put: (((posArray at: (i+3)) degreesToRadians) asGoodDouble)].

position _ posArray.

"the transformation matrix from the marker frame to the part frame"

phi _ (posArray at: 4) asGoodDouble. theta _ (posArray at: 5) asGoodDouble. psi _ (posArray at: 6) asGoodDouble.

sA _ phi sin. sB _ theta sin. sC _ psi sin. cA _ phi cos. cB _ theta cos. cC _ psi cos.

a11 _ cA*cC - (sA*sB*sC). a12 _ -1*cA*sC - (sA*cB*cC). a13 _ sA*sB.

a21 _ sA*cC + (cA*cB*sC). a22 _ -1*sA*sC + (cA*cB*cC). a23 _ -1*cA*sB.

a31 _ sB*sC. a32 _ sB*cC. a33 _ cB.

matrixA _ (Array new: 3) collect: [: each | Array new: 3].

(matrixA at: 1) at: 1 put: a11; at: 2 put: a12; at: 3 put: a13.

(matrixA at: 2) at: 1 put: a21; at: 2 put: a22; at: 3 put: a23.

(matrixA at: 3) at: 1 put: a31; at: 2 put: a32; at: 3 put: a33.

transformationMatrix _ matrixA.

position: anArray

"updates the position of the center-of-mass marker."

position _ anArray.

B.6 Joint Class

```

System subclass: #Joint
  instanceVariableNames: 'type markerI markerJ lambda '
  classVariableNames: "
  poolDictionaries: "
  category: 'Multibody System'

markerI
  "Returns the i-marker instance related to a joint."
  ^ markerI

markerJ
  "Returns the j-marker instance related to a joint."
  ^ markerJ

type
  "Returns the type of a joint."
  ^ type

initializeJointBetween: markerInstance1 and: markerInstance2 type: aString
  "Initializes a joint using the arguments and counts the local and global number of constraints of the system."
  markerI _ markerInstance1. markerJ _ markerInstance2. type _ aString.
  (aString = 'spherical') ifTrue: [ NumberOfConstraints _ NumberOfConstraints + 3].
  (aString = 'universal') ifTrue: [ NumberOfConstraints _ NumberOfConstraints + 4].
  (aString = 'revolute') ifTrue: [ NumberOfConstraints _ NumberOfConstraints + 5].
  (aString = 'translational') ifTrue: [ NumberOfConstraints _ NumberOfConstraints + 5].
  (aString = 'cylindrical') ifTrue: [ NumberOfConstraints _ NumberOfConstraints + 4].
  (aString = 'planar') ifTrue: [ NumberOfConstraints _ NumberOfConstraints + 3].
  JointDictionary add: self

atPoint1
  "Solves the constraint equation  $G1 = (dIJ)X$ "
  | partI partJ x phi theta psi sA sB sC cA cB cC xx yy zz termI termJ valueG |
  "----- For Marker I -----"
  partI _ (self markerI) part.
  x _ ((partI centerOfMassMarker) position) at: 1.
  phi _ ((partI centerOfMassMarker) position) at: 4.
  theta _ ((partI centerOfMassMarker) position) at: 5.
  psi _ ((partI centerOfMassMarker) position) at: 6.
  sA _ phi sin. sB _ theta sin. sC _ psi sin. cA _ phi cos. cB _ theta cos. cC _ psi cos.
  xx _ (markerI position) at: 1. yy _ (markerI position) at: 2. zz _ (markerI position) at: 3.
  termI _ x + (xx*(cA*cC - (sA*cB*sC))) + (yy*(-1*cA*sC - (sA*cB*cC))) + (zz*sA*sB).
  "----- For Marker J -----"
  partJ _ (self markerJ) part.
  x _ ((partJ centerOfMassMarker) position) at: 1.
  phi _ ((partJ centerOfMassMarker) position) at: 4.
  theta _ ((partJ centerOfMassMarker) position) at: 5.
  psi _ ((partJ centerOfMassMarker) position) at: 6.
  sA _ phi sin. sB _ theta sin. sC _ psi sin. cA _ phi cos. cB _ theta cos. cC _ psi cos.
  xx _ (markerJ position) at: 1. yy _ (markerJ position) at: 2. zz _ (markerJ position) at: 3.
  termJ _ x + (xx*(cA*cC - (sA*cB*sC))) + (yy*(-1*cA*sC - (sA*cB*cC))) + (zz*sA*sB).
  valueG _ termI - termJ.
  ^ valueG

atPoint2
  "Solves the constraint equation  $G2 = (dIJ)Y$ "
  | partI partJ y phi theta psi sA sB sC cA cB cC xx yy zz termI termJ valueG |
  "----- For Marker I -----"

```

```

partI _ (self markerI) part.
y _ ((partI centerOfMassMarker) position) at: 2.
phi _ ((partI centerOfMassMarker) position) at: 4.
theta _ ((partI centerOfMassMarker) position) at: 5.
psi _ ((partI centerOfMassMarker) position) at: 6.
sA _ phi sin. sB _ theta sin. sC _ psi sin. cA _ phi cos. cB _ theta cos. cC _ psi cos.
xx _ (markerI position) at: 1. yy _ (markerI position) at: 2. zz _ (markerI position) at: 3.
termI _ y + (xx*(sA*cC + (cA*cB*sC))) + (yy*(-1*sA*sC + (cA*cB*cC))) - (zz*cA*sB).
"----- For Marker I -----"
partJ _ (self markerJ) part.
y _ ((partJ centerOfMassMarker) position) at: 2.
phi _ ((partJ centerOfMassMarker) position) at: 4.
theta _ ((partJ centerOfMassMarker) position) at: 5.
psi _ ((partJ centerOfMassMarker) position) at: 6.
sA _ phi sin. sB _ theta sin. sC _ psi sin. cA _ phi cos. cB _ theta cos. cC _ psi cos.
xx _ (markerJ position) at: 1. yy _ (markerJ position) at: 2. zz _ (markerJ position) at: 3.
termJ _ y + (xx*(sA*cC + (cA*cB*sC))) + (yy*(-1*sA*sC + (cA*cB*cC))) - (zz*cA*sB).
valueG _ termI - termJ.
^ valueG

```

atPoint3

```

"Solves the constraint equation  $G3 = (dIJ)Z$ "
| partI partJ z theta psi sB sC cB cC xx yy zz termI termJ valueG |
"----- For Marker I -----"
partI _ (self markerI) part.
z _ ((partI centerOfMassMarker) position) at: 3.
theta _ ((partI centerOfMassMarker) position) at: 5.
psi _ ((partI centerOfMassMarker) position) at: 6.
sB _ theta sin. sC _ psi sin. cB _ theta cos. cC _ psi cos.
xx _ (markerI position) at: 1. yy _ (markerI position) at: 2. zz _ (markerI position) at: 3.
termI _ z + (xx*sB*sC) + (yy*sB*cC) + (zz*cB).
"----- For Marker J -----"
partJ _ (self markerJ) part.
z _ ((partJ centerOfMassMarker) position) at: 3.
theta _ ((partJ centerOfMassMarker) position) at: 5.
psi _ ((partJ centerOfMassMarker) position) at: 6.
sB _ theta sin. sC _ psi sin. cB _ theta cos. cC _ psi cos.
xx _ (markerJ position) at: 1. yy _ (markerJ position) at: 2. zz _ (markerJ position) at: 3.
termJ _ z + (xx*sB*sC) + (yy*sB*cC) + (zz*cB).
valueG _ termI - termJ.
^ valueG

```

InPlane1

```

"Solves the constraint equation;
 $G1 = (d)(i'J) = dX*C11 + dY*C12 + dZ*C13$ . See the thesis."
| matrixAtJm row1 partJ cmJ matrixAtJp row2 c11 c12 c13 dX dY dZ valueG |
matrixAtJm _ super transpose: ((self markerJ) transformationMatrix).
row1 _ super partition: matrixAtJm from: (1@1) extension: (3@1).
partJ _ (self markerJ) part. cmJ _ partJ centerOfMassMarker.
matrixAtJp _ super transpose: (cmJ fromCenterToGround).
row2 _ super multiply: row1 with: matrixAtJp.
c11 _ (row2 at: 1) at: 1. c12 _ (row2 at: 1) at: 2. c13 _ (row2 at: 1) at: 3.
dX _ self atPoint1. dY _ self atPoint2. dZ _ self atPoint3.
valueG _ dX*c11 + (dY*c12) + (dZ*c13).
^ valueG

```

InPlane2

```

"Solves the constraint equation;
 $G2 = (d)(j'J) = dX*C21 + dY*C22 + dZ*C23$ . See the thesis."
| matrixAtJm row1 partJ cmJ matrixAtJp row2 c21 c22 c23 dX dY dZ valueG |
matrixAtJm _ super transpose: ((self markerJ) transformationMatrix).
row1 _ super partition: matrixAtJm from: (1@2) extension: (3@1).

```

```

partJ _ (self markerJ) part. cmJ _ partJ centerOfMassMarker.
matrixAtJp _ super transpose: (cmJ fromCenterToGround).
row2 _ super multiply: row1 with: matrixAtJp.
c21 _ (row2 at: 1) at: 1. c22 _ (row2 at: 1) at: 2. c23 _ (row2 at: 1) at: 3.
dX _ self atPoint1. dY _ self atPoint2. dZ _ self atPoint3.
valueG _ dX*c21 + (dY*c22) + (dZ*c23).
^valueG

```

inPlane3

```

"Solves the constraint equation;
G3 = (d)(k'J) = dX*C31 + dY*C32 + dZ*C33. See the thesis."
| matrixAtJm row1 partJ cmJ matrixAtJp row2 c31 c32 c33 dX dY dZ valueG |
matrixAtJm _ super transpose: ((self markerJ) transformationMatrix).
row1 _ super partition: matrixAtJm from: (1@3) extension: (3@1).
partJ _ (self markerJ) part. cmJ _ partJ centerOfMassMarker.
matrixAtJp _ super transpose: (cmJ fromCenterToGround).
row2 _ super multiply: row1 with: matrixAtJp.
c31 _ (row2 at: 1) at: 1. c32 _ (row2 at: 1) at: 2. c33 _ (row2 at: 1) at: 3.
dX _ self atPoint1. dY _ self atPoint2. dZ _ self atPoint3.
valueG _ dX*c31 + (dY*c32) + (dZ*c33).
^valueG

```

perpendicular1

```

"Solves the constraint equation G1 = (kI)(iJ) = D13. See the thesis."
| matrixAtJm partJ cmJ matrixAtJp matrix1 row partI cmI matrixAIp matrixAlm matrix2 column valueG |
matrixAtJm _ super transpose: ((self markerJ) transformationMatrix).
partJ _ (self markerJ) part. cmJ _ partJ centerOfMassMarker.
matrixAtJp _ super transpose: (cmJ fromCenterToGround).
matrix1 _ super multiply: matrixAtJm with: matrixAtJp.
row _ super partition: matrix1 from: (1@1) extension: (3@1).
partI _ (self markerI) part. cmI _ partI centerOfMassMarker. matrixAIp _ cmI fromCenterToGround.
matrixAlm _ (self markerI) transformationMatrix. matrix2 _ super multiply: matrixAIp with: matrixAlm.
column _ super partition: matrix2 from: (3@1) extension: (1@3).
valueG _ super multiply: row with: column.
^valueG

```

perpendicular2

```

"Solves the constraint equation G2 = (kI)(jJ) = D23. See the thesis."
| matrixAtJm partJ cmJ matrixAtJp matrix1 row partI cmI matrixAIp matrixAlm matrix2 column valueG |
matrixAtJm _ super transpose: ((self markerJ) transformationMatrix).
partJ _ (self markerJ) part. cmJ _ partJ centerOfMassMarker.
matrixAtJp _ super transpose: (cmJ fromCenterToGround).
matrix1 _ super multiply: matrixAtJm with: matrixAtJp.
row _ super partition: matrix1 from: (1@2) extension: (3@1).
partI _ (self markerI) part. cmI _ partI centerOfMassMarker.
matrixAIp _ cmI fromCenterToGround. matrixAlm _ (self markerI) transformationMatrix.
matrix2 _ super multiply: matrixAIp with: matrixAlm.
column _ super partition: matrix2 from: (3@1) extension: (1@3).
valueG _ super multiply: row with: column.
^valueG

```

perpendicular3

```

"Solves the constraint equation G3 = (kI)(kJ) = D33. See the thesis."
| matrixAtJm partJ cmJ matrixAtJp matrix1 row partI cmI matrixAIp matrixAlm matrix2 column valueG |
matrixAtJm _ super transpose: ((self markerJ) transformationMatrix).
partJ _ (self markerJ) part. cmJ _ partJ centerOfMassMarker.
matrixAtJp _ super transpose: (cmJ fromCenterToGround).
matrix1 _ super multiply: matrixAtJm with: matrixAtJp.
row _ super partition: matrix1 from: (1@3) extension: (3@1).
partI _ (self markerI) part. cmI _ partI centerOfMassMarker.
matrixAIp _ cmI fromCenterToGround. matrixAlm _ (self markerI) transformationMatrix.
matrix2 _ super multiply: matrixAIp with: matrixAlm.

```

```
column _ super partition: matrix2 from: (3@1) extension: (1@3).
valueG _ super multiply: row with: column.
^valueG
```

perpendicular4

```
"Solves the constraint equation  $G4 = (iI)(jJ) = D21$ . See the thesis."
| matrixAtJm partJ cmJ matrixAtUp matrix1 row partI cmI matrixAIp matrixAIm matrix2 column valueG |
matrixAtJm _ super transpose: ((self markerJ) transformationMatrix).
partJ _ (self markerJ) part. cmJ _ partJ centerOfMassMarker.
matrixAtUp _ super transpose: (cmJ fromCenterToGround).
matrix1 _ super multiply: matrixAtJm with: matrixAtUp.
row _ super partition: matrix1 from: (1@2) extension: (3@1).
partI _ (self markerI) part. cmI _ partI centerOfMassMarker.
matrixAIp _ cmI fromCenterToGround. matrixAIm _ (self markerI) transformationMatrix.
matrix2 _ super multiply: matrixAIp with: matrixAIm.
column _ super partition: matrix2 from: (1@1) extension: (1@3).
valueG _ super multiply: row with: column.
^valueG
```

setConstraints

```
"returns the constraints G of the system."
| arrayG y arraySpherical arrayUniversal arrayRevolute arrayTranslational arrayCylindrical arrayPlanar |
arrayG _ Array new: NumberOfConstraints.
arraySpherical _ Array new: 3. arrayUniversal _ Array new: 4. arrayRevolute _ Array new: 5.
arrayTranslational _ Array new: 5. arrayCylindrical _ Array new: 4. arrayPlanar _ Array new: 3.
y _ 1.
1 to: (JointDictionary size) do: [: whichJoint | | aJoint arrayOfaJoint |
    aJoint _ JointDictionary at: whichJoint.
    ((aJoint type) = 'spherical') ifTrue: [ | a1 a2 a3 | a1 _ aJoint atPoint1. a2 _ aJoint atPoint2. a3 _ aJoint atPoint3.
        arraySpherical at: 1 put: a1; at: 2 put: a2; at: 3 put: a3. arrayOfaJoint _ arraySpherical].
    ((aJoint type) = 'universal') ifTrue: [ | a1 a2 a3 a4 | a1 _ aJoint atPoint1. a2 _ aJoint atPoint2.
        a3 _ aJoint atPoint3. a4 _ aJoint perpendicular3.
        arrayUniversal at: 1 put: a1; at: 2 put: a2; at: 3 put: a3; at: 4 put: a4. arrayOfaJoint _ arrayUniversal].
    ((aJoint type) = 'revolute') ifTrue: [ | a1 a2 a3 a4 a5 | a1 _ aJoint atPoint1. a2 _ aJoint atPoint2.
        a3 _ aJoint atPoint3. a4 _ aJoint perpendicular1. a5 _ aJoint perpendicular2.
        arrayOfaJoint _ arrayRevolute at: 1 put: a1; at: 2 put: a2; at: 3 put: a3; at: 4 put: a4; at: 5 put: a5.
        arrayOfaJoint _ arrayRevolute].
    ((aJoint type) = 'translational') ifTrue: [ | a1 a2 a3 a4 a5 | a1 _ aJoint inPlane1. a2 _ aJoint inPlane2.
        a3 _ aJoint perpendicular1. a4 _ aJoint perpendicular2. a5 _ aJoint perpendicular4.
        arrayOfaJoint _ arrayTranslational at: 1 put: a1; at: 2 put: a2; at: 3 put: a3; at: 4 put: a4; at: 5 put: a5.
        arrayOfaJoint _ arrayTranslational].
    ((aJoint type) = 'cylindrical') ifTrue: [ | a1 a2 a3 a4 | a1 _ aJoint inPlane1. a2 _ aJoint inPlane2.
        a3 _ aJoint perpendicular1. a4 _ aJoint perpendicular2.
        arrayOfaJoint _ arrayCylindrical at: 1 put: a1; at: 2 put: a2; at: 3 put: a3; at: 4 put: a4.
        arrayOfaJoint _ arrayCylindrical].
    ((aJoint type) = 'planar') ifTrue: [ | a1 a2 a3 |
        a1 _ aJoint inPlane3. a2 _ aJoint perpendicular1. a3 _ aJoint perpendicular2.
        arrayOfaJoint _ arrayPlanar at: 1 put: a1; at: 2 put: a2; at: 3 put: a3. arrayOfaJoint _ arrayPlanar].
    self put: arrayOfaJoint into: arrayG at: (1@y). y _ y + (arrayOfaJoint size)].
^arrayG
```

sub11AtPoint1

```
"returns the values to each element of the matrix  $d(dG1/dq)/dq$  for an At-Point constraint, where  $G1 = (dij)x = 0$ .
The symmetric matrix has the size of 12 by 12."
|matrixA xpi ypi zpi xqj yqj zqj ap bp cp aq bq cq sAp cAp sBp cBp sCp cCp sAq cAq sBq cBq sCq cCq partI partJ
a0404 a0405 a0406 a0505 a0506 a0606 a1010 a1011 a1012 a1111 a1112 a1212 t1 t2 t3 t5 t6 t7 t8 t9 t11 t12
t13 t14 |
matrixA _ (Array new: 12) collect: [: each | Array new: 12 withAll: 0].
partI _ (self markerI) part.
xpi _ ((self markerI) position) at: 1. ypi _ ((self markerI) position) at: 2. zpi _ ((self markerI) position) at: 3.
ap _ ((partI centerOfMassMarker) position) at: 4. bp _ ((partI centerOfMassMarker) position) at: 5.
cp _ ((partI centerOfMassMarker) position) at: 6.
```

```

sAp _ ap sin. sBp _ bp sin. sCp _ cp sin. cAp _ ap cos. cBp _ bp cos. cCp _ cp cos.
t1 _ -1*sAp*sBp.
t2 _ cAp*sCp + (sAp*cBp*cCp).
t3 _ sAp*cBp*sCp - (cAp*cCp).
t5 _ sAp*sCp - (cAp*cBp*cCp).
t6 _ -1*(cAp*cBp*sCp + (sAp*cCp)).
t7 _ cAp*cBp.
t8 _ cAp*sBp*cCp.
t9 _ cAp*sBp*sCp.
t11 _ sAp*sBp*cCp.
t12 _ sAp*sBp*sCp.
t13 _ sAp*cBp*cCp.
t14 _ sAp*cBp*sCp.
a0404 _ t1*zpi + (t2*yqi) + (t3*xpi). a0405 _ t7*zpi + (t8*yqi) + (t9*xpi). a0406 _ -1*t6*yqi + (t5*xpi).
a0505 _ t1*zpi + (t13*yqi) + (t14*xpi). a0506 _ -1*t12*yqi + (t11*xpi). a0606 _ t2*yqi + (t3*xpi).
partJ _ (self markerJ) part.
xqj _ ((self markerJ) position) at: 1. yqj _ ((self markerJ) position) at: 2. zqj _ ((self markerJ) position) at: 3.
aq _ ((partJ centerOfMassMarker) position) at: 4. bq _ ((partJ centerOfMassMarker) position) at: 5.
cq _ ((partJ centerOfMassMarker) position) at: 6.
sAq _ aq sin. sBq _ bq sin. sCq _ cq sin. cAq _ aq cos. cBq _ bq cos. cCq _ cq cos.
t1 _ -1*sAq*sBq.
t2 _ cAq*sCq + (sAq*cBq*cCq).
t3 _ sAq*cBq*sCq - (cAq*cCq).
t5 _ sAq*sCq - (cAq*cBq*cCq).
t6 _ -1*(cAq*cBq*sCq + (sAq*cCq)).
t7 _ cAq*cBq.
t8 _ cAq*sBq*cCq.
t9 _ cAq*sBq*sCq.
t11 _ sAq*sBq*cCq.
t12 _ sAq*sBq*sCq.
t13 _ sAq*cBq*cCq.
t14 _ sAq*cBq*sCq.
a1010 _ (-1)*(t1*zqj + (t2*yqj) + (t3*xqj)). a1011 _ (-1)*(t7*zqj + (t8*yqj) + (t9*xqj)). a1012 _ (-1)*(-1*t6*yqj + (t5*xqj)).
a1111 _ (-1)*(t1*zqj + (t13*yqj) + (t14*xqj)). a1112 _ (-1)*(-1*t12*yqj + (t11*xqj)). a1212 _ (-1)*(t2*yqj + (t3*xqj)).
(matrixA at: 4) at: 4 put: a0404. (matrixA at: 4) at: 5 put: a0405. (matrixA at: 4) at: 6 put: a0406.
(matrixA at: 5) at: 5 put: a0505. (matrixA at: 5) at: 6 put: a0506. (matrixA at: 6) at: 6 put: a0606.
(matrixA at: 10) at: 10 put: a1010. (matrixA at: 10) at: 11 put: a1011. (matrixA at: 10) at: 12 put: a1012.
(matrixA at: 11) at: 11 put: a1111. (matrixA at: 11) at: 12 put: a1112. (matrixA at: 12) at: 12 put: a1212.
"symmetric"
(matrixA at: 5) at: 4 put: a0405. (matrixA at: 6) at: 4 put: a0406. (matrixA at: 6) at: 5 put: a0506.
(matrixA at: 11) at: 10 put: a1011. (matrixA at: 12) at: 10 put: a1012. (matrixA at: 12) at: 11 put: a1112.
^ matrixA

```

sub11AtPoint2

"returns the values to each element of the matrix $d(dG2/dq)/dq$ for an At-Point constraint, where $G2 = (dij)y = 0$. The symmetric matrix has the size of 12 by 12."

```

lmatrixA xpi ypi zpi xqj yqj zqj ap bp cp aq bq cq sAp cAp sBp cBp sCp cCp sAq cAq sBq cBq sCq cCq partI partJ
a0404 a0405 a0406 a0505 a0506 a0606 a1010 a1011 a1012 a1111 a1112 a1212 t2 t3 t4 t5 t6 t8 t9 t10 t11 t12
t15 t16l

```

```
matrixA _ (Array new: 12) collect: [: each | Array new: 12 withAll: 0].
```

```
partI _ (self markerI) part.
```

```
xpi _ ((self markerI) position) at: 1. ypi _ ((self markerI) position) at: 2. zpi _ ((self markerI) position) at: 3.
```

```
ap _ ((partI centerOfMassMarker) position) at: 4. bp _ ((partI centerOfMassMarker) position) at: 5.
```

```
cp _ ((partI centerOfMassMarker) position) at: 6.
```

```
sAp _ ap sin. sBp _ bp sin. sCp _ cp sin. cAp _ ap cos. cBp _ bp cos. cCp _ cp cos.
```

```
t2 _ cAp*sCp + (sAp*cBp*cCp).
```

```
t3 _ sAp*cBp*sCp - (cAp*cCp).
```

```
t4 _ cAp*sBp.
```

```
t5 _ sAp*sCp - (cAp*cBp*cCp).
```

```
t6 _ -1*(cAp*cBp*sCp + (sAp*cCp)).
```



```

t8 _ cAp*sBp*cCp.
t9 _ cAp*sBp*sCp.
t10 _ sAp*cBp.
t11 _ sAp*sBp*cCp.
t12 _ sAp*sBp*sCp.
t15 _ -1*cAp*cBp*cCp.
t16 _ -1*cAp*cBp*sCp.
a0404 _ t4*zpi + (t5*yyp) + (t6*xpi). a0405 _ t10*zpi + (t11*yyp) + (t12*xpi). a0406 _ t3*yyp - (t2*xpi).
a0505 _ t4*zpi + (t15*yyp) + (t16*xpi). a0506 _ t9*yyp - (t8*xpi). a0606 _ t5*yyp + (t6*xpi).
partJ _ (self markerJ) part.
xqj _ ((self markerJ) position) at: 1. yqj _ ((self markerJ) position) at: 2. zqj _ ((self markerJ) position) at: 3.
aq _ ((partJ centerOfMassMarker) position) at: 4. bq _ ((partJ centerOfMassMarker) position) at: 5.
cq _ ((partJ centerOfMassMarker) position) at: 6.
sAq _ aq sin. sBq _ bq sin. sCq _ cq sin. cAq _ aq cos. cBq _ bq cos. cCq _ cq cos.
t2 _ cAq*sCq + (sAq*cBq*cCq).
t3 _ sAq*cBq*sCq - (cAq*cCq).
t4 _ cAq*sBq.
t5 _ sAq*sCq - (cAq*cBq*cCq).
t6 _ -1*(cAq*cBq*sCq + (sAq*cCq)).
t8 _ cAq*sBq*cCq.
t9 _ cAq*sBq*sCq.
t10 _ sAq*cBq.
t11 _ sAq*sBq*cCq.
t12 _ sAq*sBq*sCq.
t15 _ -1*cAq*cBq*cCq.
t16 _ -1*cAq*cBq*sCq.
a1010 _ (-1)*(t4*zqj + (t5*yqj) + (t6*xqj)). a1011 _ (-1)*(t10*zqj + (t11*yqj) + (t12*xqj)). a1012 _ (-1)*(t3*yqj
- (t2*xqj)).
a1111 _ (-1)*(t4*zqj + (t15*yqj) + (t16*xqj)). a1112 _ (-1)*(t9*yqj - (t8*xqj)). a1212 _ (-1)*(t5*yqj + (t6*xqj)).
(matrixA at: 4) at: 4 put: a0404. (matrixA at: 4) at: 5 put: a0405. (matrixA at: 4) at: 6 put: a0406.
(matrixA at: 5) at: 5 put: a0505. (matrixA at: 5) at: 6 put: a0506. (matrixA at: 6) at: 6 put: a0606.
(matrixA at: 10) at: 10 put: a1010. (matrixA at: 10) at: 11 put: a1011. (matrixA at: 10) at: 12 put: a1012.
(matrixA at: 11) at: 11 put: a1111. (matrixA at: 11) at: 12 put: a1112. (matrixA at: 12) at: 12 put: a1212.
"symmetric"
(matrixA at: 5) at: 4 put: a0405. (matrixA at: 6) at: 4 put: a0406. (matrixA at: 6) at: 5 put: a0506.
(matrixA at: 11) at: 10 put: a1011. (matrixA at: 12) at: 10 put: a1012. (matrixA at: 12) at: 11 put: a1112.
^ matrixA

```

sub11AtPoint3

"returns the values to each element of the matrix $d(dG3/dq)/dq$ for an At-Point constraint, where $G3 = (dij)z = 0$. The symmetric matrix has the size of 12 by 12."

```

!matrixA xpi ypi zpi xqj yqj zqj bp cp bq cq sBp cBp sCp cCp sBq cBq sCq cCq partI partJ a0505 a0506 a0606
a1111 a1112 a1212 t17 t18 t19 t20 t21 |

```

```
matrixA _ (Array new: 12) collect: [: each | Array new: 12 withAll: 0].
```

```
partI _ (self markerI) part.
```

```
xpi _ ((self markerI) position) at: 1. ypi _ ((self markerI) position) at: 2. zpi _ ((self markerI) position) at: 3.
```

```
bp _ ((partI centerOfMassMarker) position) at: 5. cp _ ((partI centerOfMassMarker) position) at: 6.
```

```
sBp _ bp sin. sCp _ cp sin. cBp _ bp cos. cCp _ cp cos.
```

```
t17 _ -1*cBp.
```

```
t18 _ -1*sBp*cCp.
```

```
t19 _ -1*sBp*sCp.
```

```
t20 _ cBp*cCp.
```

```
t21 _ -1*cBp*sCp.
```

```
a0505 _ t17*zpi + (t18*yyp) + (t19*xpi). a0506 _ t21*yyp + (t20*xpi). a0606 _ t18*yyp + (t19*xpi).
```

```
partJ _ (self markerJ) part.
```

```
xqj _ ((self markerJ) position) at: 1. yqj _ ((self markerJ) position) at: 2. zqj _ ((self markerJ) position) at: 3.
```

```
bq _ ((partJ centerOfMassMarker) position) at: 5. cq _ ((partJ centerOfMassMarker) position) at: 6.
```

```
sBq _ bq sin. sCq _ cq sin. cBq _ bq cos. cCq _ cq cos.
```

```
t17 _ -1*cBq.
```

```
t18 _ -1*sBq*cCq.
```

```
t19 _ -1*sBq*sCq.
```

```
t20 _ cBq*cCq.
```

```

t21 _ -1*cBq*sCq.
a1111 _ (-1)*(t17*zqj + (t18*yqj) + (t19*xqj)). a1112 _ (-1)*(t21*yqj + (t20*xqj)). a1212 _ (-1)*(t18*yqj +
(t19*xqj)).
(matrixA at: 5) at: 5 put: a0505. (matrixA at: 5) at: 6 put: a0506. (matrixA at: 6) at: 6 put: a0606.
(matrixA at: 11) at: 11 put: a1111. (matrixA at: 11) at: 12 put: a1112. (matrixA at: 12) at: 12 put: a1212.
"symmetric"
(matrixA at: 6) at: 5 put: a0506. (matrixA at: 12) at: 11 put: a1112.
^ matrixA

```

"The following definitions of the temporary variables are used for the the methods (sub11InPlane1, sub11InPlane2, sub11InPlane3, sub11Perpendicular1, sub11Perpendicular2, sub11Perpendicular3, sub11Perpendicular4)"

```

t1 _ sAq*sBq.
t2 _ -1*(cAq*sCq + (sAq*cBq*cCq)).
t3 _ cAq*cCq - (sAq*cBq*sCq).
t4 _ -1*sAp*sBp.
t5 _ cAp*sCp + (sAp*cBp*cCp).
t6 _ sAp*cBp*sCp - (cAp*cCp).
t7 _ -1*cAq*sBq.
t8 _ cAq*cBq*cCq - (sAq*sCq).
t9 _ cAq*cBq*sCq + (sAq*cCq).
t10 _ cAp*sBp.
t11 _ sAp*sCp - (cAp*cBp*cCp).
t12 _ -1*(cAp*cBp*sCp + (sAp*cCp)).
t13 _ sAp*cBp.
t14 _ sAp*sBp*cCp.
t15 _ sAp*sBp*sCp.
t16 _ cAp*cBp.
t17 _ cAp*sBp*cCp.
t18 _ cAp*sBp*sCp.
t27 _ cBq.
t28 _ sBq*cCq.
t29 _ sBq*sCq.
t30 _ -1*cBp.
t31 _ -1*sBp*cCp.
t32 _ -1*sBp*sCp.
t33 _ -1*cBp*sCp.
t34 _ cBp*cCp.
t35 _ -1*sAq*cBq.
t36 _ -1*sAq*sBq*cCq.
t37 _ -1*sAq*sBq*sCq.
t38 _ cAq*cBq.
t39 _ cAq*sBq*cCq.
t40 _ cAq*sBq*sCq.
t41 _ sBq.
t42 _ -1*cBq*cCq.
t43 _ -1*cBq*sCq.
t48 _ -1*sBp."

```

sub11InPlane1

"returns the values to each element of the matrix <lambda>d(dG1/dq)/dq for an In-Plane constraint, where $G1 = (dij)(xj) = 0$. The symmetric matrix has the size of 12 by 12."

```

!matrixA matrixZ partI partJ z11 z21 z31 xpi ypi zpi xp yp zp ap bp cp xqj yqj zqj xq yq zq aq bq cq sAp cAp sBp
cBp sCp cCp sAq cAq sBq cBq sCq cCq a0404 a0405 a0406 a0505 a0506 a0606 a0710 a0711 a0712 a0810 a0811
a0812 a0911 a0912 a1010 a1011 a1012 a1111 a1112 a1212 a0110 a0111 a0112 a0210 a0211 a0212 a0311
a0312 a0410 a0411 a0412 a0510 a0511 a0512 a0610 a0611 a0612 t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t13 t14
t15 t16 t17 t18 t19 t20 t23 t24 t25 t26 t27 t28 t29 t30 t31 t32 t33 t34 t35 t36 t37 t38 t39 t40 t41 t42 t43 t44
t45 t46 t47 t48 sum1 sum2 sum3 sum4 sum5 sum6 sum7 sum8 sum9 sum10 sum11 sum12 sum13 sum14 sum15
sum16 sum17 sum18 |
matrixA _ (Array new: 12) collect: [: each | Array new: 12 withAll: 0].
matrixZ _ (self markerJ) transformationMatrix.
z11 _ (matrixZ at: 1) at: 1. z21 _ (matrixZ at: 2) at: 1. z31 _ (matrixZ at: 3) at: 1.

```

```

partI _ (self markerI) part.
xpi _ (markerI position) at: 1. ypi _ (markerI position) at: 2. zpi _ (markerI position) at: 3.
xp _ ((partI centerOfMassMarker) position) at: 1. yp _ ((partI centerOfMassMarker) position) at: 2.
zp _ ((partI centerOfMassMarker) position) at: 3.
ap _ ((partI centerOfMassMarker) position) at: 4. bp _ ((partI centerOfMassMarker) position) at: 5.
cp _ ((partI centerOfMassMarker) position) at: 6.
sAp _ ap sin. sBp _ bp sin. sCp _ cp sin. cAp _ ap cos. cBp _ bp cos. cCp _ cp cos.
partJ _ (self markerJ) part.
xqj _ (markerJ position) at: 1. yqj _ (markerJ position) at: 2. zqj _ (markerJ position) at: 3.
xq _ ((partJ centerOfMassMarker) position) at: 1. yq _ ((partJ centerOfMassMarker) position) at: 2.
zq _ ((partJ centerOfMassMarker) position) at: 3.
aq _ ((partJ centerOfMassMarker) position) at: 4. bq _ ((partJ centerOfMassMarker) position) at: 5.
cq _ ((partJ centerOfMassMarker) position) at: 6.
sAq _ aq sin. sBq _ bq sin. sCq _ cq sin. cAq _ aq cos. cBq _ bq cos. cCq _ cq cos.
sum1 _ t1*z31 + (t2*z21) + (t3*z11). sum2 _ t1*zqj + (t2*yqj) + (t3*xqj). sum3 _ t7*z31 + (t8*z21) + (t9*z11).
sum4 _ t7*zqj + (t8*yqj) + (t9*xqj). sum5 _ t10*zpi + (t11*ypi) + (t12*xpi). sum6 _ t4*zpi + (t5*ypi) +
(t6*xpi).
sum7 _ t38*z31 + (t39*z21) + (t40*z11). sum8 _ t35*z31 + (t36*z21) + (t37*z11). sum9 _ t27*z31 + (t28*z21)
+ (t29*z11).
sum10 _ t41*z31 + (t42*z21) + (t43*z11). sum11 _ t27*zqj + (t28*yqj) + (t29*xqj). sum12 _ t30*zpi +
(t31*ypi) + (t32*xpi).
sum13 _ t41*zqj + (t42*yqj) + (t43*xqj). sum14 _ t35*zqj + (t36*yqj) + (t37*xqj). sum15 _ t38*zqj + (t39*yqj) +
(t40*xqj).
sum16 _ t48*zpi + (t34*ypi) - (t33*xpi). sum17 _ t13*zpi + (t14*ypi) + (t15*xpi). sum18 _ t16*zpi + (t17*ypi)
+ (t18*xpi).
a0404 _ sum1*sum6 + (sum3*sum5).
a0405 _ sum3*(t13*zpi + (t14*ypi) + (t15*xpi)) + (sum1*(t16*zpi + (t17*ypi) + (t18*xpi))).
a0406 _ sum1*(t19*ypi + (t20*xpi)) + (sum3*(t6*ypi - (t5*xpi))).
a0505 _ sum1*(t4*zpi + (t23*ypi) + (t24*xpi)) + (sum3*(t10*zpi + (t25*ypi) + (t26*xpi))) + (sum9*sum12).
a0506 _ sum1*(-1*t15*ypi + (t14*xpi)) + (sum3*(t18*ypi + (-1*t17*xpi))) + (sum9*(t33*ypi + (t34*xpi))).
a0606 _ sum1*(t5*ypi + (t6*xpi)) + (sum3*(t11*ypi + (t12*xpi))) + (sum9*(t31*ypi + (t32*xpi))).
a0710 _ sum3.
a0711 _ sum8.
a0712 _ t3*z21 - (t2*z11).
a0810 _ -1*sum1.
a0811 _ sum7.
a0812 _ t9*z21 - (t8*z11).
a0911 _ sum10.
a0912 _ t29*z21 - (t28*z11).
a1010 _ sum1*(sum6 + xq - xp) + (sum3*(sum5 + yq - yp)).
a1011 _ -1*sum7*(sum6 + xq - xp) + (sum8*(sum5 + yq - yp)).
a1012 _ -1*(t9*z21 - (t8*z11))*(sum6 + xq - xp) + ((t3*z21 - (t2*z11))*(sum5 + yq - yp)).
a1111 _ sum1*(t1*zqj - (t44*yqj) - (t45*xqj)) + ((t1*z31 - (t44*z21) - (t45*z11))*(sum2 + sum6 + xq - xp))
+ ((t7*z31 + (t46*z21) + (t47*z11))*(sum4 + sum5 + yq - yp)) + (sum3*(t7*zqj + (t46*yqj) + (t47*xqj)))
- (2*sum10*sum13) - (2*sum8*sum14) - (2*sum7*sum15) + (2*sum9*(t27*zqj + (t28*yqj) + (t29*xqj)))
+ (sum9*(sum12 + zq - zp)).
a1112 _ (t37*z21 - (t36*z11))*(-1*sum2 - sum6 - xq + xp) + ((t40*z21 - (t39*z11))*(-1*sum4 - sum5 - yq + yp))
+ ((-1*t29*z21 + (t28*z11))*sum13) + ((-1*t3*z21 + (t2*z11))*sum14) + ((-1*t9*z21 + (t8*z11))*sum15)
+ ((t43*z21 - (t42*z11))*(-1*sum11 - sum12 - zq + zp))
+ ((-1*t37*yqj + (t36*xqj))*sum1) + ((-1*t40*yqj + (t39*xqj))*sum3)
+ ((t29*yqj - (t28*xqj))*(-1*sum10)) + ((t3*yqj - (t2*xqj))*(-1*sum8))
+ ((t9*yqj - (t8*xqj))*(-1*sum7)) + ((-1*t43*yqj + (t42*xqj))*sum9).
a1212 _ (t2*z21 + (t3*z11))*(sum2 + sum6 + xq - xp) + ((t8*z21 + (t9*z11))*(sum4 + sum5 + yq - yp))
+ ((t28*z21 + (t29*z11))*(sum11 + sum12 + zq - zp)) + ((t2*yqj + (t3*xqj))*sum1) + ((t8*yqj + (t9*xqj))*sum3)
+ ((t28*yqj + (t29*xqj))*sum9) + (2*(t3*yqj - (t2*xqj))*(-1*t3*z21 + (t2*z11)))
+ (2*(t9*yqj - (t8*xqj))*(-1*t9*z21 + (t8*z11))) + (2*(t29*yqj - (t28*xqj))*(-1*t29*z21 + (t28*z11))).
a0110 _ -1*a0710. a0111 _ -1*a0711. a0112 _ -1*a0712.
a0210 _ -1*a0810. a0211 _ -1*a0811. a0212 _ -1*a0812.
a0311 _ -1*a0911. a0312 _ -1*a0912.
a0410 _ -1*a0404.
a0411 _ sum7*sum6.
a0412 _ sum6*(t9*z21 - (t8*z11)) + (sum5*(-1*t3*z21 + (t2*z11))).

```

```

a0510 _ -1*a0405.
a0511 _ -1*sum10*sum16 - (sum8*sum17) + (sum7*sum18).
a0512 _ (-1*t29*z21 + (t28*z11))*sum16 + ((-1*t3*z21 + (t2*z11))*sum17) + ((t9*z21 - (t8*z11))*sum18).
a0610 _ -1*a0406.
a0611 _ (-1*t32*ypi + (t31*xpi))*sum10 + ((-1*t6*ypi + (t5*xpi))*sum8) + ((-1*t12*ypi + (t11*xpi))*sum7).
a0612 _ (t6*ypi - (t5*xpi))*(-1*t3*z21 + (t2*z11)) + ((t12*ypi - (t11*xpi))*(-1*t9*z21 + (t8*z11)))
+ ((t32*ypi - (t31*xpi))*(-1*t29*z21 + (t28*z11))).
(matrixA at: 4) at: 4 put: a0404. (matrixA at: 4) at: 5 put: a0405. (matrixA at: 4) at: 6 put: a0406.
(matrixA at: 5) at: 5 put: a0505. (matrixA at: 5) at: 6 put: a0506. (matrixA at: 6) at: 6 put: a0606.
(matrixA at: 7) at: 10 put: a0710. (matrixA at: 7) at: 11 put: a0711. (matrixA at: 7) at: 12 put: a0712.
(matrixA at: 8) at: 10 put: a0810. (matrixA at: 8) at: 11 put: a0811. (matrixA at: 8) at: 12 put: a0812.
(matrixA at: 9) at: 11 put: a0911. (matrixA at: 9) at: 12 put: a0912. (matrixA at: 10) at: 10 put: a1010.
(matrixA at: 10) at: 11 put: a1011. (matrixA at: 10) at: 12 put: a1012. (matrixA at: 11) at: 11 put: a1111.
(matrixA at: 11) at: 12 put: a1112. (matrixA at: 12) at: 12 put: a1212. (matrixA at: 1) at: 10 put: a0110.
(matrixA at: 1) at: 11 put: a0111. (matrixA at: 1) at: 12 put: a0112. (matrixA at: 2) at: 10 put: a0210.
(matrixA at: 2) at: 11 put: a0211. (matrixA at: 2) at: 12 put: a0212. (matrixA at: 3) at: 11 put: a0311.
(matrixA at: 3) at: 12 put: a0312. (matrixA at: 4) at: 10 put: a0410. (matrixA at: 4) at: 11 put: a0411.
(matrixA at: 4) at: 12 put: a0412. (matrixA at: 5) at: 10 put: a0510. (matrixA at: 5) at: 11 put: a0511.
(matrixA at: 5) at: 12 put: a0512. (matrixA at: 6) at: 10 put: a0610. (matrixA at: 6) at: 11 put: a0611.
(matrixA at: 6) at: 12 put: a0612. (matrixA at: 5) at: 4 put: a0405. (matrixA at: 6) at: 4 put: a0406.
(matrixA at: 6) at: 5 put: a0506. (matrixA at: 10) at: 7 put: a0710. (matrixA at: 11) at: 7 put: a0711.
(matrixA at: 12) at: 7 put: a0712. (matrixA at: 10) at: 8 put: a0810. (matrixA at: 11) at: 8 put: a0811.
(matrixA at: 12) at: 8 put: a0812. (matrixA at: 11) at: 9 put: a0911. (matrixA at: 12) at: 9 put: a0912.
(matrixA at: 11) at: 10 put: a1011. (matrixA at: 12) at: 10 put: a1012. (matrixA at: 12) at: 11 put: a1112.
(matrixA at: 10) at: 1 put: a0110. (matrixA at: 11) at: 1 put: a0111. (matrixA at: 12) at: 1 put: a0112.
(matrixA at: 10) at: 2 put: a0210. (matrixA at: 11) at: 2 put: a0211. (matrixA at: 12) at: 2 put: a0212.
(matrixA at: 11) at: 3 put: a0311. (matrixA at: 12) at: 3 put: a0312. (matrixA at: 10) at: 4 put: a0410.
(matrixA at: 11) at: 4 put: a0411. (matrixA at: 12) at: 4 put: a0412. (matrixA at: 10) at: 5 put: a0510.
(matrixA at: 11) at: 5 put: a0511. (matrixA at: 12) at: 5 put: a0512. (matrixA at: 10) at: 6 put: a0610.
(matrixA at: 11) at: 6 put: a0611. (matrixA at: 12) at: 6 put: a0612.
^ matrixA

```

sub11InPlane2

"returns the values to each element of the matrix <lambda>d(dG2/dq)/dq for an In-Plane constraint, where G2 = (dij)(y_j) = 0. The symmetric matrix has the size of 12 by 12."

```

!matrixA matrixZ partI partJ z11 z21 z31 xpi ypi zpi xp yp zp ap bp cp xqj yqj zqj xq yq zq aq bq cq sAp cAp sBp
cBp sCp cCp sAq cAq sBq cBq sCq cCq a0404 a0405 a0406 a0505 a0506 a0606 a0710 a0711 a0712 a0810 a0811
a0812 a0911 a0912 a1010 a1011 a1012 a1111 a1112 a1212 a0110 a0111 a0112 a0210 a0211 a0212 a0311
a0312 a0410 a0411 a0412 a0510 a0511 a0512 a0610 a0611 a0612 t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t13 t14
t15 t16 t17 t18 t19 t20 t23 t24 t25 t26 t27 t28 t29 t30 t31 t32 t33 t34 t35 t36 t37 t38 t39 t40 t41 t42 t43 t44
t45 t46 t47 t48 sum1 sum2 sum3 sum4 sum5 sum6 sum7 sum8 sum9 sum10 sum11 sum12 sum13 sum14 sum15
sum16 sum17 sum18 |
matrixA _ (Array new: 12) collect: [: each | Array new: 12 withAll: 0].
matrixZ _ (self markerJ) transformationMatrix.
z11 _ (matrixZ at: 1) at: 2. z21 _ (matrixZ at: 2) at: 2. z31 _ (matrixZ at: 3) at: 2.
partI _ (self markerI) part.
xpi _ (markerI position) at: 1. ypi _ (markerI position) at: 2. zpi _ (markerI position) at: 3.
xp _ ((partI centerOfMassMarker) position) at: 1. yp _ ((partI centerOfMassMarker) position) at: 2.
zp _ ((partI centerOfMassMarker) position) at: 3.
ap _ ((partI centerOfMassMarker) position) at: 4. bp _ ((partI centerOfMassMarker) position) at: 5.
cp _ ((partI centerOfMassMarker) position) at: 6.
sAp _ ap sin. sBp _ bp sin. sCp _ cp sin. cAp _ ap cos. cBp _ bp cos. cCp _ cp cos.
partJ _ (self markerJ) part.
xqj _ (markerJ position) at: 1. yqj _ (markerJ position) at: 2. zqj _ (markerJ position) at: 3.
xq _ ((partJ centerOfMassMarker) position) at: 1. yq _ ((partJ centerOfMassMarker) position) at: 2.
zq _ ((partJ centerOfMassMarker) position) at: 3.
aq _ ((partJ centerOfMassMarker) position) at: 4. bq _ ((partJ centerOfMassMarker) position) at: 5.
cq _ ((partJ centerOfMassMarker) position) at: 6.
sAq _ aq sin. sBq _ bq sin. sCq _ cq sin. cAq _ aq cos. cBq _ bq cos. cCq _ cq cos.
sum1 _ t1*z31 + (t2*z21) + (t3*z11). sum2 _ t1*zqj + (t2*yqj) + (t3*xqj). sum3 _ t7*z31 + (t8*z21) - (t9*z11).
sum4 _ t7*zqj + (t8*yqj) + (t9*xqj). sum5 _ t10*zpi + (t11*ypi) + (t12*xpi). sum6 _ t4*zpi + (t5*ypi) +
(t6*xpi).

```

$\text{sum7} _ t38*z31 + (t39*z21) + (t40*z11).$ $\text{sum8} _ t35*z31 + (t36*z21) + (t37*z11).$ $\text{sum9} _ t27*z31 + (t28*z21) + (t29*z11).$
 $\text{sum10} _ t41*z31 + (t42*z21) + (t43*z11).$ $\text{sum11} _ t27*zqj + (t28*yqj) + (t29*xqj).$ $\text{sum12} _ t30*zpi + (t31*y\text{pi}) + (t32*x\text{pi}).$
 $\text{sum13} _ t41*zqj + (t42*yqj) + (t43*xqj).$ $\text{sum14} _ t35*zqj + (t36*yqj) + (t37*xqj).$ $\text{sum15} _ t38*zqj + (t39*yqj) + (t40*xqj).$
 $\text{sum16} _ t48*zpi + (t34*y\text{pi}) - (t33*x\text{pi}).$ $\text{sum17} _ t13*zpi + (t14*y\text{pi}) + (t15*x\text{pi}).$ $\text{sum18} _ t16*zpi + (t17*y\text{pi}) + (t18*x\text{pi}).$
 $\text{a0404} _ \text{sum1}*\text{sum6} + (\text{sum3}*\text{sum5}).$
 $\text{a0405} _ \text{sum3}*(t13*zpi + (t14*y\text{pi}) + (t15*x\text{pi})) + (\text{sum1}*(t16*zpi + (t17*y\text{pi}) + (t18*x\text{pi}))).$
 $\text{a0406} _ \text{sum1}*(t19*y\text{pi} + (t20*x\text{pi})) + (\text{sum3}*(t6*y\text{pi} - (t5*x\text{pi}))).$
 $\text{a0505} _ \text{sum1}*(t4*zpi + (t23*y\text{pi}) + (t24*x\text{pi})) + (\text{sum3}*(t10*zpi + (t25*y\text{pi}) + (t26*x\text{pi}))) + (\text{sum9}*\text{sum12}).$
 $\text{a0506} _ \text{sum1}*(-1*t15*y\text{pi} + (t14*x\text{pi})) + (\text{sum3}*(t18*y\text{pi} + (-1*t17*x\text{pi}))) + (\text{sum9}*(t33*y\text{pi} + (t34*x\text{pi}))).$
 $\text{a0606} _ \text{sum1}*(t5*y\text{pi} + (t6*x\text{pi})) + (\text{sum3}*(t11*y\text{pi} + (t12*x\text{pi}))) + (\text{sum9}*(t31*y\text{pi} + (t32*x\text{pi}))).$
 $\text{a0710} _ \text{sum3}.$
 $\text{a0711} _ \text{sum8}.$
 $\text{a0712} _ t3*z21 - (t2*z11).$
 $\text{a0810} _ -1*\text{sum1}.$
 $\text{a0811} _ \text{sum7}.$
 $\text{a0812} _ t9*z21 - (t8*z11).$
 $\text{a0911} _ \text{sum10}.$
 $\text{a0912} _ t29*z21 - (t28*z11).$
 $\text{a1010} _ \text{sum1}*(\text{sum6} + xq - xp) + (\text{sum3}*(\text{sum5} + yq - yp)).$
 $\text{a1011} _ -1*\text{sum7}*(\text{sum6} + xq - xp) + (\text{sum8}*(\text{sum5} + yq - yp)).$
 $\text{a1012} _ -1*(t9*z21 - (t8*z11))*(\text{sum6} + xq - xp) + ((t3*z21 - (t2*z11))*(\text{sum5} + yq - yp)).$
 $\text{a1111} _ \text{sum1}*(t1*zqj - (t44*yqj) - (t45*xqj)) + ((t1*z31 - (t44*z21) - (t45*z11))*(\text{sum2} + \text{sum6} + xq - xp))$
 $+ ((t7*z31 + (t46*z21) + (t47*z11))*(\text{sum4} + \text{sum5} + yq - yp))$
 $+ (\text{sum3}*(t7*zqj + (t46*yqj) + (t47*xqj))) - (2*\text{sum10}*\text{sum13}) - (2*\text{sum8}*\text{sum14}) - (2*\text{sum7}*\text{sum15})$
 $+ (2*\text{sum9}*(t27*zqj + (t28*yqj) + (t29*xqj))) + (\text{sum9}*(\text{sum12} + zq - zp)).$
 $\text{a1112} _ (t37*z21 - (t36*z11))*(-1*\text{sum2} - \text{sum6} - xq + xp) + ((t40*z21 - (t39*z11))*(-1*\text{sum4} - \text{sum5} - yq + yp))$
 $+ ((-1*t29*z21 + (t28*z11))*\text{sum13}) + ((-1*t3*z21 + (t2*z11))*\text{sum14}) + ((-1*t9*z21 + (t8*z11))*\text{sum15})$
 $+ ((t43*z21 - (t42*z11))*(-1*\text{sum11} - \text{sum12} - zq + zp))$
 $+ ((-1*t37*yqj + (t36*xqj))*\text{sum1}) + ((-1*t40*yqj + (t39*xqj))*\text{sum3})$
 $+ ((t29*yqj - (t28*xqj))*(-1*\text{sum10})) + ((t3*yqj - (t2*xqj))*(-1*\text{sum8}))$
 $+ ((t9*yqj - (t8*xqj))*(-1*\text{sum7})) + ((-1*t43*yqj + (t42*xqj))*\text{sum9}).$
 $\text{a1212} _ (t2*z21 + (t3*z11))*(\text{sum2} + \text{sum6} + xq - xp) + ((t8*z21 + (t9*z11))*(\text{sum4} + \text{sum5} + yq - yp))$
 $+ ((t28*z21 + (t29*z11))*(\text{sum11} + \text{sum12} + zq - zp)) + ((t2*yqj + (t3*xqj))*\text{sum1}) + ((t8*yqj + (t9*xqj))*\text{sum3})$
 $+ ((t28*yqj + (t29*xqj))*\text{sum9}) + (2*(t3*yqj - (t2*xqj))*(-1*t3*z21 + (t2*z11)))$
 $+ (2*(t9*yqj - (t8*xqj))*(-1*t9*z21 + (t8*z11))) + (2*(t29*yqj - (t28*xqj))*(-1*t29*z21 + (t28*z11))).$
 $\text{a0110} _ -1*\text{a0710}.$ $\text{a0111} _ -1*\text{a0711}.$ $\text{a0112} _ -1*\text{a0712}.$
 $\text{a0210} _ -1*\text{a0810}.$ $\text{a0211} _ -1*\text{a0811}.$ $\text{a0212} _ -1*\text{a0812}.$
 $\text{a0311} _ -1*\text{a0911}.$ $\text{a0312} _ -1*\text{a0912}.$
 $\text{a0410} _ -1*\text{a0404}.$
 $\text{a0411} _ \text{sum7}*\text{sum6}.$
 $\text{a0412} _ \text{sum6}*(t9*z21 - (t8*z11)) + (\text{sum5}*(-1*t3*z21 + (t2*z11))).$
 $\text{a0510} _ -1*\text{a0405}.$
 $\text{a0511} _ -1*\text{sum10}*\text{sum16} - (\text{sum8}*\text{sum17}) + (\text{sum7}*\text{sum18}).$
 $\text{a0512} _ (-1*t29*z21 + (t28*z11))*\text{sum16} + ((-1*t3*z21 + (t2*z11))*\text{sum17}) + ((t9*z21 - (t8*z11))*\text{sum18}).$
 $\text{a0610} _ -1*\text{a0406}.$
 $\text{a0611} _ (-1*t32*y\text{pi} + (t31*x\text{pi}))*\text{sum10} + ((-1*t6*y\text{pi} + (t5*x\text{pi}))*\text{sum8}) + ((-1*t12*y\text{pi} + (t11*x\text{pi}))*\text{sum7}).$
 $\text{a0612} _ (t6*y\text{pi} - (t5*x\text{pi}))*(-1*t3*z21 + (t2*z11)) + ((t12*y\text{pi} - (t11*x\text{pi}))*(-1*t9*z21 + (t8*z11)))$
 $+ ((t32*y\text{pi} - (t31*x\text{pi}))*(-1*t29*z21 + (t28*z11))).$
(matrixA at: 4) at: 4 put: a0404. (matrixA at: 4) at: 5 put: a0405. (matrixA at: 4) at: 6 put: a0406.
(matrixA at: 5) at: 5 put: a0505. (matrixA at: 5) at: 6 put: a0506. (matrixA at: 6) at: 6 put: a0606.
(matrixA at: 7) at: 10 put: a0710. (matrixA at: 7) at: 11 put: a0711. (matrixA at: 7) at: 12 put: a0712.
(matrixA at: 8) at: 10 put: a0810. (matrixA at: 8) at: 11 put: a0811. (matrixA at: 8) at: 12 put: a0812.
(matrixA at: 9) at: 11 put: a0911. (matrixA at: 9) at: 12 put: a0912. (matrixA at: 10) at: 10 put: a1010.
(matrixA at: 10) at: 11 put: a1011. (matrixA at: 10) at: 12 put: a1012. (matrixA at: 11) at: 11 put: a1111.
(matrixA at: 11) at: 12 put: a1112. (matrixA at: 12) at: 12 put: a1212. (matrixA at: 1) at: 10 put: a0110.
(matrixA at: 1) at: 11 put: a0111. (matrixA at: 1) at: 12 put: a0112. (matrixA at: 2) at: 10 put: a0210.
(matrixA at: 2) at: 11 put: a0211. (matrixA at: 2) at: 12 put: a0212. (matrixA at: 3) at: 11 put: a0311.
(matrixA at: 3) at: 12 put: a0312. (matrixA at: 4) at: 10 put: a0410. (matrixA at: 4) at: 11 put: a0411.

```

(matrixA at: 4) at: 12 put: a0412. (matrixA at: 5) at: 10 put: a0510. (matrixA at: 5) at: 11 put: a0511.
(matrixA at: 5) at: 12 put: a0512. (matrixA at: 6) at: 10 put: a0610. (matrixA at: 6) at: 11 put: a0611.
(matrixA at: 6) at: 12 put: a0612. (matrixA at: 5) at: 4 put: a0405. (matrixA at: 6) at: 4 put: a0406.
(matrixA at: 6) at: 5 put: a0506. (matrixA at: 10) at: 7 put: a0710. (matrixA at: 11) at: 7 put: a0711.
(matrixA at: 12) at: 7 put: a0712. (matrixA at: 10) at: 8 put: a0810. (matrixA at: 11) at: 8 put: a0811.
(matrixA at: 12) at: 8 put: a0812. (matrixA at: 11) at: 9 put: a0911. (matrixA at: 12) at: 9 put: a0912.
(matrixA at: 11) at: 10 put: a1011. (matrixA at: 12) at: 10 put: a1012. (matrixA at: 12) at: 11 put: a1112.
(matrixA at: 10) at: 1 put: a0110. (matrixA at: 11) at: 1 put: a0111. (matrixA at: 12) at: 1 put: a0112.
(matrixA at: 10) at: 2 put: a0210. (matrixA at: 11) at: 2 put: a0211. (matrixA at: 12) at: 2 put: a0212.
(matrixA at: 11) at: 3 put: a0311. (matrixA at: 12) at: 3 put: a0312. (matrixA at: 10) at: 4 put: a0410.
(matrixA at: 11) at: 4 put: a0411. (matrixA at: 12) at: 4 put: a0412. (matrixA at: 10) at: 5 put: a0510.
(matrixA at: 11) at: 5 put: a0511. (matrixA at: 12) at: 5 put: a0512. (matrixA at: 10) at: 6 put: a0610.
(matrixA at: 11) at: 6 put: a0611. (matrixA at: 12) at: 6 put: a0612.
^ matrixA

```

sub11InPlane3

"returns the values to each element of the matrix <lambda>d(dG3/dq)/dq for an In-Plane constraint, where $G3 = (dij)(zj) = 0$. The symmetric matrix has the size of 12 by 12."

```

!matrixA matrixZ partI partJ z11 z21 z31 xpi ypi zpi xp yp zp ap bp cp xqj yqj zqj xq yq zq aq bq cq sAp cAp sBp
cBp sCp cCp sAq cAq sBq cBq sCq cCq a0404 a0405 a0406 a0505 a0506 a0606 a0710 a0711 a0712 a0810 a0811
a0812 a0911 a0912 a1010 a1011 a1012 a1111 a1112 a1212 a0110 a0111 a0112 a0210 a0211 a0212 a0311
a0312 a0410 a0411 a0412 a0510 a0511 a0512 a0610 a0611 a0612 t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t13 t14
t15 t16 t17 t18 t19 t20 t23 t24 t25 t26 t27 t28 t29 t30 t31 t32 t33 t34 t35 t36 t37 t38 t39 t40 t41 t42 t43 t44
t45 t46 t47 t48 sum1 sum2 sum3 sum4 sum5 sum6 sum7 sum8 sum9 sum10 sum11 sum12 sum13 sum14 sum15
sum16 sum17 sum18 |
matrixA _ (Array new: 12) collect: [: each | Array new: 12 withAll: 0].
matrixZ _ (self markerJ) transformationMatrix.
z11 _ (matrixZ at: 1) at: 3. z21 _ (matrixZ at: 2) at: 3. z31 _ (matrixZ at: 3) at: 3.
partI _ (self markerI) part.
xpi _ (markerI position) at: 1. ypi _ (markerI position) at: 2. zpi _ (markerI position) at: 3.
xp _ ((partI centerOfMassMarker) position) at: 1. yp _ ((partI centerOfMassMarker) position) at: 2.
zp _ ((partI centerOfMassMarker) position) at: 3.
ap _ ((partI centerOfMassMarker) position) at: 4. bp _ ((partI centerOfMassMarker) position) at: 5.
cp _ ((partI centerOfMassMarker) position) at: 6.
sAp _ ap sin. sBp _ bp sin. sCp _ cp sin. cAp _ ap cos. cBp _ bp cos. cCp _ cp cos.
partJ _ (self markerJ) part.
xqj _ (markerJ position) at: 1. yqj _ (markerJ position) at: 2. zqj _ (markerJ position) at: 3.
xq _ ((partJ centerOfMassMarker) position) at: 1. yq _ ((partJ centerOfMassMarker) position) at: 2.
zq _ ((partJ centerOfMassMarker) position) at: 3.
aq _ ((partJ centerOfMassMarker) position) at: 4. bq _ ((partJ centerOfMassMarker) position) at: 5.
cq _ ((partJ centerOfMassMarker) position) at: 6.
sAq _ aq sin. sBq _ bq sin. sCq _ cq sin. cAq _ aq cos. cBq _ bq cos. cCq _ cq cos.
sum1 _ t1*z31 + (t2*z21) + (t3*z11). sum2 _ t1*zqj + (t2*yqj) + (t3*xqj). sum3 _ t7*z31 + (t8*z21) + (t9*z11).
sum4 _ t7*zqj + (t8*yqj) + (t9*xqj). sum5 _ t10*zpi + (t11*ypi) + (t12*xpi). sum6 _ t4*zpi + (t5*ypi) +
(t6*xpi).
sum7 _ t38*z31 + (t39*z21) + (t40*z11). sum8 _ t35*z31 + (t36*z21) + (t37*z11). sum9 _ t27*z31 + (t28*z21)
+ (t29*z11).
sum10 _ t41*z31 + (t42*z21) + (t43*z11). sum11 _ t27*zqj + (t28*yqj) + (t29*xqj). sum12 _ t30*zpi +
(t31*ypi) + (t32*xpi).
sum13 _ t41*zqj + (t42*yqj) + (t43*xqj). sum14 _ t35*zqj + (t36*yqj) + (t37*xqj). sum15 _ t38*zqj + (t39*yqj) +
(t40*xqj).
sum16 _ t48*zpi + (t34*ypi) - (t33*xpi). sum17 _ t13*zpi + (t14*ypi) + (t15*xpi). sum18 _ t16*zpi + (t17*ypi)
+ (t18*xpi).
a0404 _ sum1*sum6 + (sum3*sum5).
a0405 _ sum3*(t13*zpi + (t14*ypi) + (t15*xpi)) + (sum1*(t16*zpi + (t17*ypi) + (t18*xpi))).
a0406 _ sum1*(t19*ypi + (t20*xpi)) + (sum3*(t6*ypi - (t5*xpi))).
a0505 _ sum1*(t4*zpi + (t23*ypi) + (t24*xpi)) + (sum3*(t10*zpi + (t25*ypi) + (t26*xpi))) + (sum9*sum12).
a0506 _ sum1*(-1*t15*ypi + (t14*xpi)) + (sum3*(t18*ypi + (-1*t17*xpi))) + (sum9*(t33*ypi + (t34*xpi))).
a0606 _ sum1*(t5*ypi + (t6*xpi)) + (sum3*(t11*ypi + (t12*xpi))) + (sum9*(t31*ypi + (t32*xpi))).
a0710 _ sum3.
a0711 _ sum8.
a0712 _ t3*z21 - (t2*z11).

```

```

a0810 _ -1*sum1.
a0811 _ sum7.
a0812 _ t9*z21 - (t8*z11).
a0911 _ sum10.
a0912 _ t29*z21 - (t28*z11).
a1010 _ sum1*(sum6 + xq - xp) + (sum3*(sum5 + yq - yp)).
a1011 _ -1*sum7*(sum6 + xq - xp) + (sum8*(sum5 + yq - yp)).
a1012 _ -1*(t9*z21 - (t8*z11))*(sum6 + xq - xp) + ((t3*z21 - (t2*z11))*(sum5 + yq - yp)).
a1111 _ sum1*(t1*zqj - (t44*yqj) - (t45*xqj)) + ((t1*z31 - (t44*z21) - (t45*z11))*(sum2 + sum6 + xq - xp))
+ ((t7*z31 + (t46*z21) + (t47*z11))*(sum4 + sum5 + yq - yp))
+ (sum3*(t7*zqj + (t46*yqj) + (t47*xqj))) - (2*sum10*sum13) - (2*sum8*sum14) - (2*sum7*sum15)
+ (2*sum9*(t27*zqj + (t28*yqj) + (t29*xqj))) + (sum9*(sum12 + zq - zp)).
a1112 _ (t37*z21 - (t36*z11))*(-1*sum2 - sum6 - xq + xp) + ((t40*z21 - (t39*z11))*(-1*sum4 - sum5 - yq + yp))
+ ((-1*t29*z21 + (t28*z11))*sum13) + ((-1*t3*z21 + (t2*z11))*sum14) + ((-1*t9*z21 + (t8*z11))*sum15)
+ ((t43*z21 - (t42*z11))*(-1*sum11 - sum12 - zq + zp))
+ ((-1*t37*yqj + (t36*xqj))*sum1) + ((-1*t40*yqj + (t39*xqj))*sum3)
+ ((t29*yqj - (t28*xqj))*(-1*sum10)) + ((t3*yqj - (t2*xqj))*(-1*sum8))
+ ((t9*yqj - (t8*xqj))*(-1*sum7)) + ((-1*t43*yqj + (t42*xqj))*sum9).
a1212 _ (t2*z21 + (t3*z11))*(sum2 + sum6 + xq - xp) + ((t8*z21 + (t9*z11))*(sum4 + sum5 + yq - yp))
+ ((t28*z21 + (t29*z11))*(sum11 + sum12 + zq - zp)) + ((t2*yqj + (t3*xqj))*sum1) + ((t8*yqj + (t9*xqj))*sum3)
+ ((t28*yqj + (t29*xqj))*sum9) + (2*(t3*yqj - (t2*xqj))*(-1*t3*z21 + (t2*z11)))
+ (2*(t9*yqj - (t8*xqj))*(-1*t9*z21 + (t8*z11))) + (2*(t29*yqj - (t28*xqj))*(-1*t29*z21 + (t28*z11))).
a0110 _ -1*a0710. a0111 _ -1*a0711. a0112 _ -1*a0712.
a0210 _ -1*a0810. a0211 _ -1*a0811. a0212 _ -1*a0812.
a0311 _ -1*a0911. a0312 _ -1*a0912.
a0410 _ -1*a0404.
a0411 _ sum7*sum6.
a0412 _ sum6*(t9*z21 - (t8*z11)) + (sum5*(-1*t3*z21 + (t2*z11))).
a0510 _ -1*a0405.
a0511 _ -1*sum10*sum16 - (sum8*sum17) + (sum7*sum18).
a0512 _ (-1*t29*z21 + (t28*z11))*sum16 + ((-1*t3*z21 + (t2*z11))*sum17) + ((t9*z21 - (t8*z11))*sum18).
a0610 _ -1*a0406.
a0611 _ (-1*t32*yqi + (t31*xpi))*sum10 + ((-1*t6*yqi + (t5*xpi))*sum8) + ((-1*t12*yqi + (t11*xpi))*sum7).
a0612 _ (t6*yqi - (t5*xpi))*(-1*t3*z21 + (t2*z11)) + ((t12*yqi - (t11*xpi))*(-1*t9*z21 + (t8*z11)))
+ ((t32*yqi - (t31*xpi))*(-1*t29*z21 + (t28*z11))).
(matrixA at: 4) at: 4 put: a0404. (matrixA at: 4) at: 5 put: a0405. (matrixA at: 4) at: 6 put: a0406.
(matrixA at: 5) at: 5 put: a0505. (matrixA at: 5) at: 6 put: a0506. (matrixA at: 6) at: 6 put: a0606.
(matrixA at: 7) at: 10 put: a0710. (matrixA at: 7) at: 11 put: a0711. (matrixA at: 7) at: 12 put: a0712.
(matrixA at: 8) at: 10 put: a0810. (matrixA at: 8) at: 11 put: a0811. (matrixA at: 8) at: 12 put: a0812.
(matrixA at: 9) at: 11 put: a0911. (matrixA at: 9) at: 12 put: a0912.
(matrixA at: 10) at: 10 put: a1010. (matrixA at: 10) at: 11 put: a1011. (matrixA at: 10) at: 12 put: a1012.
(matrixA at: 11) at: 11 put: a1111. (matrixA at: 11) at: 12 put: a1112.
(matrixA at: 12) at: 12 put: a1212.
(matrixA at: 1) at: 10 put: a0110. (matrixA at: 1) at: 11 put: a0111. (matrixA at: 1) at: 12 put: a0112.
(matrixA at: 2) at: 10 put: a0210. (matrixA at: 2) at: 11 put: a0211. (matrixA at: 2) at: 12 put: a0212.
(matrixA at: 3) at: 11 put: a0311. (matrixA at: 3) at: 12 put: a0312.
(matrixA at: 4) at: 10 put: a0410. (matrixA at: 4) at: 11 put: a0411. (matrixA at: 4) at: 12 put: a0412.
(matrixA at: 5) at: 10 put: a0510. (matrixA at: 5) at: 11 put: a0511. (matrixA at: 5) at: 12 put: a0512.
(matrixA at: 6) at: 10 put: a0610. (matrixA at: 6) at: 11 put: a0611. (matrixA at: 6) at: 12 put: a0612.
(matrixA at: 5) at: 4 put: a0405. (matrixA at: 6) at: 4 put: a0406. (matrixA at: 6) at: 5 put: a0506.
(matrixA at: 10) at: 7 put: a0710. (matrixA at: 11) at: 7 put: a0711. (matrixA at: 12) at: 7 put: a0712.
(matrixA at: 10) at: 8 put: a0810. (matrixA at: 11) at: 8 put: a0811. (matrixA at: 12) at: 8 put: a0812.
(matrixA at: 11) at: 9 put: a0911. (matrixA at: 12) at: 9 put: a0912. (matrixA at: 11) at: 10 put: a1011.
(matrixA at: 12) at: 10 put: a1012. (matrixA at: 12) at: 11 put: a1112. (matrixA at: 10) at: 1 put: a0110.
(matrixA at: 11) at: 1 put: a0111. (matrixA at: 12) at: 1 put: a0112. (matrixA at: 10) at: 2 put: a0210.
(matrixA at: 11) at: 2 put: a0211. (matrixA at: 12) at: 2 put: a0212. (matrixA at: 11) at: 3 put: a0311.
(matrixA at: 12) at: 3 put: a0312. (matrixA at: 10) at: 4 put: a0410. (matrixA at: 11) at: 4 put: a0411.
(matrixA at: 12) at: 4 put: a0412. (matrixA at: 10) at: 5 put: a0510. (matrixA at: 11) at: 5 put: a0511.
(matrixA at: 12) at: 5 put: a0512. (matrixA at: 10) at: 6 put: a0610. (matrixA at: 11) at: 6 put: a0611.
(matrixA at: 12) at: 6 put: a0612.
^ matrixA

```

sub11Perpendicular1

"returns the values to each element of the matrix $\langle \lambda \rangle d(G1/dq)/dq$ for a perpendicular constraint, where $G1 = (z_i)(x_j) = 0$. The symmetric matrix has the size of 12 by 12."

```

matrixA matrixW matrixZ partI partJ ap bp cp aq bq cq sAp cAp sBp cBp sCp cCp sAq cAq sBq cBq sCq cCq a0404
a0405 a0406 a0505 a0506 a0606 a1010 a1011 a1012 a1111 a1112 a1212 a0410 a0411 a0412 a0510 a0511
a0512 a0610 a0611 a0612 t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t13 t14 t15 t16 t17 t18 t23 t24 t25 t26 t27 t28 t29
t30 t31 t32 t33 t34 t35 t36 t37 t38 t39 t40 t41 t42 t43 t44 t45 t46 t47 t48 sum1 sum3 sum7 sum8 sum9 sum10
sum19 sum20 w31 w32 w33 z11 z21 z31 |
matrixA _ (Array new: 12) collect: [: each | Array new: 12 withAll: 0].
matrixW _ (self markerI) transformationMatrix. matrixZ _ (self markerJ) transformationMatrix.
w31 _ (matrixW at: 3) at: 1. w32 _ (matrixW at: 3) at: 2. w33 _ (matrixW at: 3) at: 3.
z11 _ (matrixZ at: 1) at: 1. z21 _ (matrixZ at: 2) at: 1. z31 _ (matrixZ at: 3) at: 1.
partI _ (self markerI) part.
ap _ ((partI centerOfMassMarker) position) at: 4. bp _ ((partI centerOfMassMarker) position) at: 5.
cp _ ((partI centerOfMassMarker) position) at: 6.
sAp _ ap sin. sBp _ bp sin. sCp _ cp sin. cAp _ ap cos. cBp _ bp cos. cCp _ cp cos.
partJ _ (self markerJ) part.
aq _ ((partJ centerOfMassMarker) position) at: 4. bq _ ((partJ centerOfMassMarker) position) at: 5.
cq _ ((partJ centerOfMassMarker) position) at: 6.
sAq _ aq sin. sBq _ bq sin. sCq _ cq sin. cAq _ aq cos. cBq _ bq cos. cCq _ cq cos.
sum1 _ t1*z31 + (t2*z21) + (t3*z11). sum3 _ t7*z31 + (t8*z21) + (t9*z11). sum7 _ t38*z31 + (t39*z21) +
(t40*z11).
sum8 _ t35*z31 + (t36*z21) + (t37*z11). sum9 _ t27*z31 + (t28*z21) + (t29*z11). sum10 _ t41*z31 +
(t42*z21) + (t43*z11).
sum19 _ t1*z31 - (t44*z21) - (t45*z11). sum20 _ t7*z31 + (t46*z21) + (t47*z11).
a0404 _ w31*(t6*sum1 + (t12*sum3)) + (w32*(t5*sum1 + (t11*sum3))) + (w33*(t4*sum1 + (t10*sum3))).
a0405 _ w31*(t18*sum1 + (t15*sum3)) + (w32*(t17*sum1 + (t14*sum3))) + (w33*(t16*sum1 + (t13*sum3))).
a0406 _ w31*(t11*sum1 - (t5*sum3)) + (w32*(-1*t12*sum1 + (t6*sum3))).
a0505 _ w31*(t24*sum1 + (t26*sum3) + (t32*sum9)) + (w32*(t23*sum1 + (t25*sum3) + (t31*sum9)))
+ (w33*(t4*sum1 + (t10*sum3) + (t30*sum9))).
a0506 _ w31*(t14*sum1 - (t17*sum3) + (t34*sum9)) + (w32*(-1*t15*sum1 + (t18*sum3) + (t33*sum9))).
a0606 _ w31*(t6*sum1 - (t12*sum3) + (t32*sum9)) + (w32*(t5*sum1 + (t11*sum3) + (t31*sum9))).
a1010 _ a0404.
a1011 _ w31*(t12*sum8 + (-1*t6*sum7)) + (w32*(t11*sum8 + (-1*t5*sum7))) + (w33*(t10*sum8 + (-
1*t4*sum7))).
a1012 _ w31*(t12*(t3*z21 - (t2*z11)) + (t6*(-1*t9*z21 + (t8*z11))))
+ (w32*(t11*(t3*z21 - (t2*z11)) + (t5*(-1*t9*z21 + (t8*z11)))))
+ (w33*(t10*(t3*z21 - (t2*z11)) + (t4*(-1*t9*z21 + (t8*z11)))).
a1111 _ w31*(t6*sum19 + (t12*sum20) + (t32*sum9)) + (w32*(t5*sum19 + (t11*sum20) + (t31*sum9)))
+ (w33*(t4*sum19 + (t10*sum20) + (t30*sum9))).
a1112 _ w31*(-1*t6*(t37*z21 - (t36*z11)) + (-1*t12*(t40*z21 - (t39*z11))) + (-1*t32*(t43*z21 - (t42*z11))))
+ (w32*(-1*t5*(t37*z21 - (t36*z11)) + (-1*t11*(t40*z21 - (t39*z11))) + (-1*t31*(t43*z21 - (t42*z11)))))
+ (w33*(-1*t4*(t37*z21 - (t36*z11)) + (-1*t10*(t40*z21 - (t39*z11))) + (-1*t30*(t43*z21 - (t42*z11)))).
a1212 _ w31*(t12*(t8*z21 + (t9*z11)) + (t6*(t2*z21 + (t3*z11))) + (t32*(t28*z21 + (t29*z11))))
+ (w32*(t11*(t8*z21 + (t9*z11)) + (t5*(t2*z21 + (t3*z11))) + (t31*(t28*z21 + (t29*z11)))))
+ (w33*(t10*(t8*z21 + (t9*z11)) + (t4*(t2*z21 + (t3*z11))) + (t30*(t28*z21 + (t29*z11)))).
a0410 _ -1*a1010. a0411 _ -1*a1011. a0412 _ -1*a1012.
a0510 _ w31*(-1*t15*sum3 - (t18*sum1)) + (w32*(-1*t14*sum3 - (t17*sum1))) + (w33*(-1*t13*sum3 -
(t16*sum1))).
a0511 _ w31*(t33*sum10 - (t15*sum8) + (t18*sum7)) + (w32*(-1*t34*sum10 - (t14*sum8) + (t17*sum7)))
+ (w33*(-1*t48*sum10 - (t13*sum8) + (t16*sum7))).
a0512 _ w31*(t15*(-1*t3*z21 + (t2*z11)) + (t18*(t9*z21 - (t8*z11))) + (t33*(t29*z21 - (t28*z11))))
+ (w32*(t14*(-1*t3*z21 + (t2*z11)) + (t17*(t9*z21 - (t8*z11))) - (t34*(t29*z21 - (t28*z11)))))
+ (w33*(t13*(-1*t3*z21 + (t2*z11)) + (t16*(t9*z21 - (t8*z11))) - (t48*(t29*z21 - (t28*z11)))).
a0610 _ -1*a0406.
a0611 _ w31*(t31*sum10 + (t5*sum8) - (t8*sum7)) + (w32*(-1*t32*sum10 - (t6*sum8) + (t9*sum7))).
a0612 _ w31*(t5*(t3*z21 - (t2*z11)) + (t11*(t9*z21 - (t8*z11))) + (t31*(t29*z21 - (t28*z11))))
+ (w31*(-1*t6*(t3*z21 - (t2*z11)) + (-1*t12*(t9*z21 - (t8*z11))) + (-1*t32*(t29*z21 - (t28*z11)))).
(matrixA at: 4) at: 4 put: a0404. (matrixA at: 4) at: 5 put: a0405. (matrixA at: 4) at: 6 put: a0406.
(matrixA at: 5) at: 5 put: a0505. (matrixA at: 5) at: 6 put: a0506. (matrixA at: 6) at: 6 put: a0606.
(matrixA at: 10) at: 10 put: a1010. (matrixA at: 10) at: 11 put: a1011. (matrixA at: 10) at: 12 put: a1012.
(matrixA at: 11) at: 11 put: a1111. (matrixA at: 11) at: 12 put: a1112. (matrixA at: 12) at: 12 put: a1212.

```



```
(matrixA at: 4) at: 10 put: a0410. (matrixA at: 4) at: 11 put: a0411. (matrixA at: 4) at: 12 put: a0412.
(matrixA at: 5) at: 10 put: a0510. (matrixA at: 5) at: 11 put: a0511. (matrixA at: 5) at: 12 put: a0512.
(matrixA at: 6) at: 10 put: a0610. (matrixA at: 6) at: 11 put: a0611. (matrixA at: 6) at: 12 put: a0612.
(matrixA at: 5) at: 4 put: a0405. (matrixA at: 6) at: 4 put: a0406. (matrixA at: 6) at: 5 put: a0506.
(matrixA at: 11) at: 10 put: a1011. (matrixA at: 12) at: 10 put: a1012. (matrixA at: 12) at: 11 put: a1112.
(matrixA at: 10) at: 4 put: a0410. (matrixA at: 11) at: 4 put: a0411. (matrixA at: 12) at: 4 put: a0412.
(matrixA at: 10) at: 5 put: a0510. (matrixA at: 11) at: 5 put: a0511. (matrixA at: 12) at: 5 put: a0512.
(matrixA at: 10) at: 6 put: a0610. (matrixA at: 11) at: 6 put: a0611. (matrixA at: 12) at: 6 put: a0612.
^ matrixA
```

sub11Perpendicular2

"returns the values to each element of the matrix <lambda>d(dG2/dq)/dq for a perpendicular constraint, where $G2 = (z_i)(y_j) = 0$. The symmetric matrix has the size of 12 by 12."

```
!matrixA matrixW matrixZ partI partJ ap bp cp aq bq cq sAp cAp sBp cBp sCp cCp sAq cAq sBq cBq sCq cCq a0404
a0405 a0406 a0505 a0506 a0606 a1010 a1011 a1012 a1111 a1112 a1212 a0410 a0411 a0412 a0510 a0511
a0512 a0610 a0611 a0612 t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t13 t14 t15 t16 t17 t18 t23 t24 t25 t26 t27 t28 t29
t30 t31 t32 t33 t34 t35 t36 t37 t38 t39 t40 t41 t42 t43 t44 t45 t46 t47 t48 sum1 sum3 sum7 sum8 sum9 sum10
sum19 sum20 w31 w32 w33 z11 z21 z31 |
matrixA _ (Array new: 12) collect: [: each | Array new: 12 withAll: 0].
matrixW _ (self markerI) transformationMatrix.
matrixZ _ (self markerJ) transformationMatrix.
w31 _ (matrixW at: 3) at: 1. w32 _ (matrixW at: 3) at: 2. w33 _ (matrixW at: 3) at: 3.
z11 _ (matrixZ at: 1) at: 2. z21 _ (matrixZ at: 2) at: 2. z31 _ (matrixZ at: 3) at: 2.
partI _ (self markerI) part.
ap _ ((partI centerOfMassMarker) position) at: 4. bp _ ((partI centerOfMassMarker) position) at: 5.
cp _ ((partI centerOfMassMarker) position) at: 6.
sAp _ ap sin. sBp _ bp sin. sCp _ cp sin. cAp _ ap cos. cBp _ bp cos. cCp _ cp cos.
partJ _ (self markerJ) part.
aq _ ((partJ centerOfMassMarker) position) at: 4. bq _ ((partJ centerOfMassMarker) position) at: 5.
cq _ ((partJ centerOfMassMarker) position) at: 6.
sAq _ aq sin. sBq _ bq sin. sCq _ cq sin. cAq _ aq cos. cBq _ bq cos. cCq _ cq cos.
sum1 _ t1*z31 + (t2*z21) + (t3*z11). sum3 _ t7*z31 + (t8*z21) + (t9*z11). sum7 _ t38*z31 + (t39*z21) +
(t40*z11).
sum8 _ t35*z31 + (t36*z21) + (t37*z11). sum9 _ t27*z31 + (t28*z21) + (t29*z11). sum10 _ t41*z31 +
(t42*z21) + (t43*z11).
sum19 _ t1*z31 - (t44*z21) - (t45*z11). sum20 _ t7*z31 + (t46*z21) + (t47*z11).
a0404 _ w31*(t6*sum1 + (t12*sum3)) + (w32*(t5*sum1 + (t11*sum3))) + (w33*(t4*sum1 + (t10*sum3))).
a0405 _ w31*(t18*sum1 + (t15*sum3)) + (w32*(t17*sum1 + (t14*sum3))) + (w33*(t16*sum1 + (t13*sum3))).
a0406 _ w31*(t11*sum1 - (t5*sum3)) + (w32*(-1*t12*sum1 + (t6*sum3))).
a0505 _ w31*(t24*sum1 + (t26*sum3) + (t32*sum9)) + (w32*(t23*sum1 + (t25*sum3) + (t31*sum9)))
+ (w33*(t4*sum1 + (t10*sum3) + (t30*sum9))).
a0506 _ w31*(t14*sum1 - (t17*sum3) + (t34*sum9)) + (w32*(-1*t15*sum1 + (t18*sum3) + (t33*sum9))).
a0606 _ w31*(t6*sum1 - (t12*sum3) + (t32*sum9)) + (w32*(t5*sum1 + (t11*sum3) + (t31*sum9))).
a1010 _ a0404.
a1011 _ w31*(t12*sum8 + (-1*t6*sum7)) + (w32*(t11*sum8 + (-1*t5*sum7))) + (w33*(t10*sum8 + (-
1*t4*sum7))).
a1012 _ w31*(t12*(t3*z21 - (t2*z11)) + (t6*(-1*t9*z21 + (t8*z11))))
+ (w32*(t11*(t3*z21 - (t2*z11)) + (t5*(-1*t9*z21 + (t8*z11)))))
+ (w33*(t10*(t3*z21 - (t2*z11)) + (t4*(-1*t9*z21 + (t8*z11))))).
a1111 _ w31*(t6*sum19 + (t12*sum20) + (t32*sum9)) + (w32*(t5*sum19 + (t11*sum20) + (t31*sum9)))
+ (w33*(t4*sum19 + (t10*sum20) + (t30*sum9))).
a1112 _ w31*(-1*t6*(t37*z21 - (t36*z11)) + (-1*t12*(t40*z21 - (t39*z11))) + (-1*t32*(t43*z21 - (t42*z11))))
+ (w32*(-1*t5*(t37*z21 - (t36*z11)) + (-1*t11*(t40*z21 - (t39*z11))) + (-1*t31*(t43*z21 - (t42*z11))))
+ (w33*(-1*t4*(t37*z21 - (t36*z11)) + (-1*t10*(t40*z21 - (t39*z11))) + (-1*t30*(t43*z21 - (t42*z11)))).
a1212 _ w31*(t12*(t8*z21 + (t9*z11)) + (t6*(t2*z21 + (t3*z11))) + (t32*(t28*z21 + (t29*z11))))
+ (w32*(t11*(t8*z21 + (t9*z11)) + (t5*(t2*z21 + (t3*z11))) + (t31*(t28*z21 + (t29*z11)))))
+ (w31*(t10*(t8*z21 + (t9*z11)) + (t4*(t2*z21 + (t3*z11))) + (t30*(t28*z21 + (t29*z11)))).
a0410 _ -1*a1010. a0411 _ -1*a1011. a0412 _ -1*a1012.
a0510 _ w31*(-1*t15*sum3 - (t18*sum1)) + (w32*(-1*t14*sum3 - (t17*sum1)))
+ (w33*(-1*t13*sum3 - (t16*sum1))).
a0511 _ w31*(t33*sum10 - (t15*sum8) + (t18*sum7)) + (w32*(-1*t34*sum10 - (t14*sum8) + (t17*sum7)))
+ (w33*(-1*t48*sum10 - (t13*sum8) + (t16*sum7))).
```

```

a0512 _ w31*(t15*(-1*t3*z21 + (t2*z11)) + (t18*(t9*z21 - (t8*z11))) + (t33*(t29*z21 - (t28*z11))))
+ (w32*(t14*(-1*t3*z21 + (t2*z11)) + (t17*(t9*z21 - (t8*z11))) - (t34*(t29*z21 - (t28*z11))))
+ (w33*(t13*(-1*t3*z21 + (t2*z11)) + (t16*(t9*z21 - (t8*z11))) - (t48*(t29*z21 - (t28*z11))))).
a0610 _ -1*a0406.
a0611 _ w31*(t31*sum10 + (t5*sum8) - (t8*sum7)) + (w32*(-1*t32*sum10 - (t6*sum8) + (t9*sum7))).
a0612 _ w31*(t5*(t3*z21 - (t2*z11)) + (t11*(t9*z21 - (t8*z11))) + (t31*(t29*z21 - (t28*z11))))
+ (w31*(-1*t6*(t3*z21 - (t2*z11)) + (-1*t12*(t9*z21 - (t8*z11))) + (-1*t32*(t29*z21 - (t28*z11))))).
(matrixA at: 4) at: 4 put: a0404. (matrixA at: 4) at: 5 put: a0405. (matrixA at: 4) at: 6 put: a0406.
(matrixA at: 5) at: 5 put: a0505. (matrixA at: 5) at: 6 put: a0506. (matrixA at: 6) at: 6 put: a0606.
(matrixA at: 10) at: 10 put: a1010. (matrixA at: 10) at: 11 put: a1011. (matrixA at: 10) at: 12 put: a1012.
(matrixA at: 11) at: 11 put: a1111. (matrixA at: 11) at: 12 put: a1112. (matrixA at: 12) at: 12 put: a1212.
(matrixA at: 4) at: 10 put: a0410. (matrixA at: 4) at: 11 put: a0411. (matrixA at: 4) at: 12 put: a0412.
(matrixA at: 5) at: 10 put: a0510. (matrixA at: 5) at: 11 put: a0511. (matrixA at: 5) at: 12 put: a0512.
(matrixA at: 6) at: 10 put: a0610. (matrixA at: 6) at: 11 put: a0611. (matrixA at: 6) at: 12 put: a0612.
(matrixA at: 5) at: 4 put: a0405. (matrixA at: 6) at: 4 put: a0406. (matrixA at: 6) at: 5 put: a0506.
(matrixA at: 11) at: 10 put: a1011. (matrixA at: 12) at: 10 put: a1012. (matrixA at: 12) at: 11 put: a1112.
(matrixA at: 10) at: 4 put: a0410. (matrixA at: 11) at: 4 put: a0411. (matrixA at: 12) at: 4 put: a0412.
(matrixA at: 10) at: 5 put: a0510. (matrixA at: 11) at: 5 put: a0511. (matrixA at: 12) at: 5 put: a0512.
(matrixA at: 10) at: 6 put: a0610. (matrixA at: 11) at: 6 put: a0611. (matrixA at: 12) at: 6 put: a0612.
^ matrixA

```

sub11Perpendicular3

"returns the values to each element of the matrix <lambda>d(dG3/dq)/dq for a perpendicular constraint, where G3 = (zi)(zj) = 0. The symmetric matrix has the size of 12 by 12."

```

lmatrixA matrixW matrixZ partI partJ ap bp cp aq bq cq sAp cAp sBp cBp sCp cCp sAq cAq sBq cBq sCq cCq a0404
a0405 a0406 a0505 a0506 a0606 a1010 a1011 a1012 a1111 a1112 a1212 a0410 a0411 a0412 a0510 a0511
a0512 a0610 a0611 a0612 t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t13 t14 t15 t16 t17 t18 t23 t24 t25 t26 t27 t28 t29
t30 t31 t32 t33 t34 t35 t36 t37 t38 t39 t40 t41 t42 t43 t44 t45 t46 t47 t48 sum1 sum3 sum7 sum8 sum9 sum10
sum19 sum20 w31 w32 w33 z11 z21 z31 |
matrixA _ (Array new: 12) collect: [: each | Array new: 12 withAll: 0].
matrixW _ (self markerI) transformationMatrix. matrixZ _ (self markerJ) transformationMatrix.
w31 _ (matrixW at: 3) at: 1. w32 _ (matrixW at: 3) at: 2. w33 _ (matrixW at: 3) at: 3.
z11 _ (matrixZ at: 1) at: 3. z21 _ (matrixZ at: 2) at: 3. z31 _ (matrixZ at: 3) at: 3.
partI _ (self markerI) part.
ap _ ((partI centerOfMassMarker) position) at: 4. bp _ ((partI centerOfMassMarker) position) at: 5.
cp _ ((partI centerOfMassMarker) position) at: 6.
sAp _ ap sin. sBp _ bp sin. sCp _ cp sin. cAp _ ap cos. cBp _ bp cos. cCp _ cp cos.
partJ _ (self markerJ) part.
aq _ ((partJ centerOfMassMarker) position) at: 4. bq _ ((partJ centerOfMassMarker) position) at: 5.
cq _ ((partJ centerOfMassMarker) position) at: 6.
sAq _ aq sin. sBq _ bq sin. sCq _ cq sin. cAq _ aq cos. cBq _ bq cos. cCq _ cq cos.
sum1 _ t1*z31 + (t2*z21) + (t3*z11). sum3 _ t7*z31 + (t8*z21) + (t9*z11). sum7 _ t38*z31 + (t39*z21) +
(t40*z11).
sum8 _ t35*z31 + (t36*z21) + (t37*z11). sum9 _ t27*z31 + (t28*z21) + (t29*z11). sum10 _ t41*z31 +
(t42*z21) + (t43*z11).
sum19 _ t1*z31 - (t44*z21) - (t45*z11). sum20 _ t7*z31 + (t46*z21) + (t47*z11).
a0404 _ w31*(t6*sum1 + (t12*sum3)) + (w32*(t5*sum1 + (t11*sum3))) + (w33*(t4*sum1 + (t10*sum3))).
a0405 _ w31*(t18*sum1 + (t15*sum3)) + (w32*(t17*sum1 + (t14*sum3))) + (w33*(t16*sum1 + (t13*sum3))).
a0406 _ w31*(t11*sum1 - (t5*sum3)) + (w32*(-1*t12*sum1 + (t6*sum3))).
a0505 _ w31*(t24*sum1 + (t26*sum3) + (t32*sum9)) + (w32*(t23*sum1 + (t25*sum3) + (t31*sum9)))
+ (w33*(t4*sum1 + (t10*sum3) + (t30*sum9))).
a0506 _ w31*(t14*sum1 - (t17*sum3) + (t34*sum9)) + (w32*(-1*t15*sum1 + (t18*sum3) + (t33*sum9))).
a0606 _ w31*(t6*sum1 - (t12*sum3) + (t32*sum9)) + (w32*(t5*sum1 + (t11*sum3) + (t31*sum9))).
a1010 _ a0404.
a1011 _ w31*(t12*sum8 + (-1*t6*sum7)) + (w32*(t11*sum8 + (-1*t5*sum7))) + (w33*(t10*sum8 + (-
1*t4*sum7))).
a1012 _ w31*(t12*(t3*z21 - (t2*z11)) + (t6*(-1*t9*z21 + (t8*z11))))
+ (w32*(t11*(t3*z21 - (t2*z11)) + (t5*(-1*t9*z21 + (t8*z11))))
+ (w33*(t10*(t3*z21 - (t2*z11)) + (t4*(-1*t9*z21 + (t8*z11))))).
a1111 _ w31*(t6*sum19 + (t12*sum20) + (t32*sum9)) + (w32*(t5*sum19 + (t11*sum20) + (t31*sum9)))
+ (w33*(t4*sum19 + (t10*sum20) + (t30*sum9))).
a1112 _ w31*(-1*t6*(t37*z21 - (t36*z11)) + (-1*t12*(t40*z21 - (t39*z11))) + (-1*t32*(t43*z21 - (t42*z11))))

```

```

+ (w32*(-1*t5*(t37*z21 - (t36*z11)) + (-1*t11*(t40*z21 - (t39*z11))) + (-1*t31*(t43*z21 - (t42*z11))))
+ (w33*(-1*t4*(t37*z21 - (t36*z11)) + (-1*t10*(t40*z21 - (t39*z11))) + (-1*t30*(t43*z21 - (t42*z11))))).
a1212 _ w31*(t12*(t8*z21 + (t9*z11)) + (t6*(t2*z21 + (t3*z11))) + (t32*(t28*z21 + (t29*z11))))
+ (w32*(t11*(t8*z21 + (t9*z11)) + (t5*(t2*z21 + (t3*z11))) + (t31*(t28*z21 + (t29*z11))))
+ (w31*(t10*(t8*z21 + (t9*z11)) + (t4*(t2*z21 + (t3*z11))) + (t30*(t28*z21 + (t29*z11))))).
a0410 _ -1*a1010. a0411 _ -1*a1011. a0412 _ -1*a1012.
a0510 _ w31*(-1*t15*sum3 - (t18*sum1)) + (w32*(-1*t14*sum3 - (t17*sum1))) + (w33*(-1*t13*sum3 -
(t16*sum1))).
a0511 _ w31*(t33*sum10 - (t15*sum8) + (t18*sum7)) + (w32*(-1*t34*sum10 - (t14*sum8) + (t17*sum7)))
+ (w33*(-1*t48*sum10 - (t13*sum8) + (t16*sum7))).
a0512 _ w31*(t15*(-1*t3*z21 + (t2*z11)) + (t18*(t9*z21 - (t8*z11))) + (t33*(t29*z21 - (t28*z11))))
+ (w32*(t14*(-1*t3*z21 + (t2*z11)) + (t17*(t9*z21 - (t8*z11))) - (t34*(t29*z21 - (t28*z11))))
+ (w33*(t13*(-1*t3*z21 + (t2*z11)) + (t16*(t9*z21 - (t8*z11))) - (t48*(t29*z21 - (t28*z11))))).
a0610 _ -1*a0406.
a0611 _ w31*(t31*sum10 + (t5*sum8) - (t8*sum7)) + (w32*(-1*t32*sum10 - (t6*sum8) + (t9*sum7))).
a0612 _ w31*(t5*(t3*z21 - (t2*z11)) + (t11*(t9*z21 - (t8*z11))) + (t31*(t29*z21 - (t28*z11))))
+ (w31*(-1*t6*(t3*z21 - (t2*z11)) + (-1*t12*(t9*z21 - (t8*z11))) + (-1*t32*(t29*z21 - (t28*z11))))).
(matrixA at: 4) at: 4 put: a0404. (matrixA at: 4) at: 5 put: a0405. (matrixA at: 4) at: 6 put: a0406.
(matrixA at: 5) at: 5 put: a0505. (matrixA at: 5) at: 6 put: a0506. (matrixA at: 6) at: 6 put: a0606.
(matrixA at: 10) at: 10 put: a1010. (matrixA at: 10) at: 11 put: a1011. (matrixA at: 10) at: 12 put: a1012.
(matrixA at: 11) at: 11 put: a1111. (matrixA at: 11) at: 12 put: a1112. (matrixA at: 12) at: 12 put: a1212.
(matrixA at: 4) at: 10 put: a0410. (matrixA at: 4) at: 11 put: a0411. (matrixA at: 4) at: 12 put: a0412.
(matrixA at: 5) at: 10 put: a0510. (matrixA at: 5) at: 11 put: a0511. (matrixA at: 5) at: 12 put: a0512.
(matrixA at: 6) at: 10 put: a0610. (matrixA at: 6) at: 11 put: a0611. (matrixA at: 6) at: 12 put: a0612.
(matrixA at: 5) at: 4 put: a0405. (matrixA at: 6) at: 4 put: a0406. (matrixA at: 6) at: 5 put: a0506.
(matrixA at: 11) at: 10 put: a1011. (matrixA at: 12) at: 10 put: a1012. (matrixA at: 12) at: 11 put: a1112.
(matrixA at: 10) at: 4 put: a0410. (matrixA at: 11) at: 4 put: a0411. (matrixA at: 12) at: 4 put: a0412.
(matrixA at: 10) at: 5 put: a0510. (matrixA at: 11) at: 5 put: a0511. (matrixA at: 12) at: 5 put: a0512.
(matrixA at: 10) at: 6 put: a0610. (matrixA at: 11) at: 6 put: a0611. (matrixA at: 12) at: 6 put: a0612.
^ matrixA

```

sub11Perpendicular4

"returns the values to each element of the matrix <lambda>d(dG4/dq)/dq for a perpendicular constraint, where G4 = (xi)(yj) = 0. The symmetric matrix has the size of 12 by 12."

```

!matrixA matrixW matrixZ partI partJ ap bp cp aq bq cq sAp cAp sBp cBp sCp cCp sAq cAq sBq cBq sCq cCq a0404
a0405 a0406 a0505 a0506 a0606 a1010 a1011 a1012 a1111 a1112 a1212 a0410 a0411 a0412 a0510 a0511
a0512 a0610 a0611 a0612 t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t13 t14 t15 t16 t17 t18 t23 t24 t25 t26 t27 t28 t29
t30 t31 t32 t33 t34 t35 t36 t37 t38 t39 t40 t41 t42 t43 t44 t45 t46 t47 t48 sum1 sum3 sum7 sum8 sum9 sum10
sum19 sum20 w31 w32 w33 z11 z21 z31 |
matrixA _ (Array new: 12) collect: [: each | Array new: 12 withAll: 0].
matrixW _ (self markerI) transformationMatrix. matrixZ _ (self markerJ) transformationMatrix.
w31 _ (matrixW at: 1) at: 1. w32 _ (matrixW at: 1) at: 2. w33 _ (matrixW at: 1) at: 3.
z11 _ (matrixZ at: 1) at: 2. z21 _ (matrixZ at: 2) at: 2. z31 _ (matrixZ at: 3) at: 2.
partI _ (self markerI) part.
ap _ ((partI centerOfMassMarker) position) at: 4. bp _ ((partI centerOfMassMarker) position) at: 5.
cp _ ((partI centerOfMassMarker) position) at: 6.
sAp _ ap sin. sBp _ bp sin. sCp _ cp sin. cAp _ ap cos. cBp _ bp cos. cCp _ cp cos.
partJ _ (self markerJ) part.
aq _ ((partJ centerOfMassMarker) position) at: 4. bq _ ((partJ centerOfMassMarker) position) at: 5.
cq _ ((partJ centerOfMassMarker) position) at: 6.
sAq _ aq sin. sBq _ bq sin. sCq _ cq sin. cAq _ aq cos. cBq _ bq cos. cCq _ cq cos.
sum1 _ t1*z31 + (t2*z21) + (t3*z11). sum3 _ t7*z31 + (t8*z21) + (t9*z11). sum7 _ t38*z31 - (t39*z21) +
(t40*z11).
sum8 _ t35*z31 + (t36*z21) + (t37*z11). sum9 _ t27*z31 + (t28*z21) + (t29*z11). sum10 _ t41*z31 +
(t42*z21) + (t43*z11).
sum19 _ t1*z31 - (t44*z21) - (t45*z11). sum20 _ t7*z31 + (t46*z21) + (t47*z11).
a0404 _ w31*(t6*sum1 + (t12*sum3)) + (w32*(t5*sum1 + (t11*sum3))) + (w33*(t4*sum1 + (t10*sum3))).
a0405 _ w31*(t18*sum1 + (t15*sum3)) + (w32*(t17*sum1 + (t14*sum3))) + (w33*(t16*sum1 + (t13*sum3))).
a0406 _ w31*(t11*sum1 - (t5*sum3)) + (w32*(-1*t12*sum1 + (t6*sum3))).
a0505 _ w31*(t24*sum1 + (t26*sum3) + (t32*sum9)) + (w32*(t23*sum1 + (t25*sum3) + (t31*sum9)))
+ (w33*(t4*sum1 + (t10*sum3) + (t30*sum9))).
a0506 _ w31*(t14*sum1 - (t17*sum3) + (t34*sum9)) + (w32*(-1*t15*sum1 + (t18*sum3) + (t33*sum9))).

```

```

a0606 _ w31*(t6*sum1 - (t12*sum3) + (t32*sum9)) + (w32*(t5*sum1 + (t11*sum3) + (t31*sum9))).
a1010 _ a0404.
a1011 _ w31*(t12*sum8 + (-1*t6*sum7)) + (w32*(t11*sum8 + (-1*t5*sum7))) + (w33*(t10*sum8 + (-1*t4*sum7))).
a1012 _ w31*(t12*(t3*z21 - (t2*z11)) + (t6*(-1*t9*z21 + (t8*z11))))
+ (w32*(t11*(t3*z21 - (t2*z11)) + (t5*(-1*t9*z21 + (t8*z11)))))
+ (w33*(t10*(t3*z21 - (t2*z11)) + (t4*(-1*t9*z21 + (t8*z11))))).
a1111 _ w31*(t6*sum19 + (t12*sum20) + (t32*sum9)) + (w32*(t5*sum19 + (t11*sum20) + (t31*sum9)))
+ (w33*(t4*sum19 + (t10*sum20) + (t30*sum9))).
a1112 _ w31*(-1*t6*(t37*z21 - (t36*z11)) + (-1*t12*(t40*z21 - (t39*z11))) + (-1*t32*(t43*z21 - (t42*z11))))
+ (w32*(-1*t5*(t37*z21 - (t36*z11)) + (-1*t11*(t40*z21 - (t39*z11))) + (-1*t31*(t43*z21 - (t42*z11)))))
+ (w33*(-1*t4*(t37*z21 - (t36*z11)) + (-1*t10*(t40*z21 - (t39*z11))) + (-1*t30*(t43*z21 - (t42*z11)))).
a1212 _ w31*(t12*(t8*z21 + (t9*z11)) + (t6*(t2*z21 + (t3*z11))) + (t32*(t28*z21 + (t29*z11))))
+ (w32*(t11*(t8*z21 + (t9*z11)) + (t5*(t2*z21 + (t3*z11))) + (t31*(t28*z21 + (t29*z11)))))
+ (w31*(t10*(t8*z21 + (t9*z11)) + (t4*(t2*z21 + (t3*z11))) + (t30*(t28*z21 + (t29*z11)))).
a0410 _ -1*a1010. a0411 _ -1*a1011. a0412 _ -1*a1012.
a0510 _ w31*(-1*t15*sum3 - (t18*sum1)) + (w32*(-1*t14*sum3 - (t17*sum1)))
+ (w33*(-1*t13*sum3 - (t16*sum1))).
a0511 _ w31*(t33*sum10 - (t15*sum8) + (t18*sum7)) + (w32*(-1*t34*sum10 - (t14*sum8) + (t17*sum7)))
+ (w33*(-1*t48*sum10 - (t13*sum8) + (t16*sum7))).
a0512 _ w31*(t15*(-1*t3*z21 + (t2*z11)) + (t18*(t9*z21 - (t8*z11))) + (t33*(t29*z21 - (t28*z11))))
+ (w32*(t14*(-1*t3*z21 + (t2*z11)) + (t17*(t9*z21 - (t8*z11))) - (t34*(t29*z21 - (t28*z11)))))
+ (w33*(t13*(-1*t3*z21 + (t2*z11)) + (t16*(t9*z21 - (t8*z11))) - (t48*(t29*z21 - (t28*z11)))).
a0610 _ -1*a0406.
a0611 _ w31*(t31*sum10 + (t5*sum8) - (t8*sum7)) + (w32*(-1*t32*sum10 - (t6*sum8) + (t9*sum7))).
a0612 _ w31*(t5*(t3*z21 - (t2*z11)) + (t11*(t9*z21 - (t8*z11))) + (t31*(t29*z21 - (t28*z11))))
+ (w31*(-1*t6*(t3*z21 - (t2*z11)) + (-1*t12*(t9*z21 - (t8*z11))) + (-1*t32*(t29*z21 - (t28*z11)))).
(matrixA at: 4) at: 4 put: a0404. (matrixA at: 4) at: 5 put: a0405. (matrixA at: 4) at: 6 put: a0406.
(matrixA at: 5) at: 5 put: a0505. (matrixA at: 5) at: 6 put: a0506. (matrixA at: 6) at: 6 put: a0606.
(matrixA at: 10) at: 10 put: a1010. (matrixA at: 10) at: 11 put: a1011. (matrixA at: 10) at: 12 put: a1012.
(matrixA at: 11) at: 11 put: a1111. (matrixA at: 11) at: 12 put: a1112. (matrixA at: 12) at: 12 put: a1212.
(matrixA at: 4) at: 10 put: a0410. (matrixA at: 4) at: 11 put: a0411. (matrixA at: 4) at: 12 put: a0412.
(matrixA at: 5) at: 10 put: a0510. (matrixA at: 5) at: 11 put: a0511. (matrixA at: 5) at: 12 put: a0512.
(matrixA at: 6) at: 10 put: a0610. (matrixA at: 6) at: 11 put: a0611. (matrixA at: 6) at: 12 put: a0612.
(matrixA at: 5) at: 4 put: a0405. (matrixA at: 6) at: 4 put: a0406. (matrixA at: 6) at: 5 put: a0506.
(matrixA at: 11) at: 10 put: a1011. (matrixA at: 12) at: 10 put: a1012. (matrixA at: 12) at: 11 put: a1112.
(matrixA at: 10) at: 4 put: a0410. (matrixA at: 11) at: 4 put: a0411. (matrixA at: 12) at: 4 put: a0412.
(matrixA at: 10) at: 5 put: a0510. (matrixA at: 11) at: 5 put: a0511. (matrixA at: 12) at: 5 put: a0512.
(matrixA at: 10) at: 6 put: a0610. (matrixA at: 11) at: 6 put: a0611. (matrixA at: 12) at: 6 put: a0612.
^ matrixA

```

sub31AtPoint1

"returns the values of the elements of the matrix dG1/dq for an At-Point constraint, where $G1 = (dij)x = 0$. The size of the matrix is (1 by 12)."

| rowA partI xpi ypi zpi ap bp cp sAp sBp sCp cAp cBp cCp partJ xqj yqj zqj aq bq cq sAq sBq sCq cAq cBq cCq t2
t3 t4 t5 t6 t10 t11 t12 a01 a04 a05 a06 a07 a10 a11 a12 |

rowA _ (Array new: 1) collect: [: row | Array new: 12 withAll: 0].

partI _ (self markerI) part.

xpi _ ((self markerI) position) at: 1. ypi _ ((self markerI) position) at: 2. zpi _ ((self markerI) position) at: 3.

ap _ ((partI centerOfMassMarker) position) at: 4. bp _ ((partI centerOfMassMarker) position) at: 5.

cp _ ((partI centerOfMassMarker) position) at: 6.

sAp _ ap sin. sBp _ bp sin. sCp _ cp sin. cAp _ ap cos. cBp _ bp cos. cCp _ cp cos.

t2 _ cAp*sCp + (sAp*cBp*cCp).

t3 _ sAp*cBp*sCp - (cAp*cCp).

t4 _ cAp*sBp.

t5 _ sAp*sCp - (cAp*cBp*cCp).

t6 _ -1*(cAp*cBp*sCp + (sAp*cCp)).

t10 _ sAp*cBp.

t11 _ sAp*sBp*cCp.

t12 _ sAp*sBp*sCp.

a01 _ 1. a04 _ t4*zpi + (t5*ypi) + (t6*xpi). a05 _ t10*zpi + (t11*ypi) + (t12*xpi). a06 _ t3*ypi - (t2*xpi).

partJ _ (self markerJ) part.

```

xqj _ ((self markerJ) position) at: 1. yqj _ ((self markerJ) position) at: 2. zqj _ ((self markerJ) position) at: 3.
aq _ ((partJ centerOfMassMarker) position) at: 4. bq _ ((partJ centerOfMassMarker) position) at: 5.
cq _ ((partJ centerOfMassMarker) position) at: 6.
sAq _ aq sin. sBq _ bq sin. sCq _ cq sin. cAq _ aq cos. cBq _ bq cos. cCq _ cq cos.
t2 _ cAq*sCq + (sAq*cBq*cCq).
t3 _ sAq*cBq*sCq - (cAq*cCq).
t4 _ cAq*sBq.
t5 _ sAq*sCq - (cAq*cBq*cCq).
t6 _ -1*(cAq*cBq*sCq + (sAq*cCq)).
t10 _ sAq*cBq.
t11 _ sAq*sBq*cCq.
t12 _ sAq*sBq*sCq.
a07 _ -1. a10 _ (-1)*(t4*zqj + (t5*yqj) + (t6*xqj)). a11 _ (-1)*(t10*zqj + (t11*yqj) + (t12*xqj)). a12 _ (-
1)*(t3*yqj - (t2*xqj)).
(rowA at: 1) at: 1 put: a01. (rowA at: 1) at: 4 put: a04. (rowA at: 1) at: 5 put: a05. (rowA at: 1) at: 6 put: a06.
(rowA at: 1) at: 7 put: a07. (rowA at: 1) at: 10 put: a10. (rowA at: 1) at: 11 put: a11.
(rowA at: 1) at: 12 put: a12.
^rowA

```

sub31AtPoint2

"returns the values of the elements of the matrix dG2/dq for an At-Point constraint, where $G2 = (dij)y = 0$. The size of the matrix is (1 by 12)."

```

| rowA partI xpi ypi zpi ap bp cp sAp sBp sCp cAp cBp cCp partJ xqj yqj zqj aq bq cq sAq sBq sCq cAq cBq cCq t1
t2 t3 t5 t6 t7 t8 t9 a02 a04 a05 a06 a08 a10 a11 a12 |

```

```

rowA _ (Array new: 1) collect: [: row | Array new: 12 withAll: 0].

```

```

partI _ (self markerI) part.

```

```

xpi _ ((self markerI) position) at: 1. ypi _ ((self markerI) position) at: 2. zpi _ ((self markerI) position) at: 3.

```

```

ap _ ((partI centerOfMassMarker) position) at: 4. bp _ ((partI centerOfMassMarker) position) at: 5.

```

```

cp _ ((partI centerOfMassMarker) position) at: 6.

```

```

sAp _ ap sin. sBp _ bp sin. sCp _ cp sin. cAp _ ap cos. cBp _ bp cos. cCp _ cp cos.

```

```

t1 _ -1*sAp*sBp.

```

```

t2 _ cAp*sCp + (sAp*cBp*cCp).

```

```

t3 _ sAp*cBp*sCp - (cAp*cCp).

```

```

t5 _ sAp*sCp - (cAp*cBp*cCp).

```

```

t6 _ -1*(cAp*cBp*sCp + (sAp*cCp)).

```

```

t7 _ cAp*cBp.

```

```

t8 _ cAp*sBp*cCp.

```

```

t9 _ cAp*sBp*sCp.

```

```

a02 _ 1. a04 _ -1*t1*zpi - (t2*ypi) - (t3*xpi). a05 _ -1*t7*zpi - (t8*ypi) - (t9*xpi). a06 _ t6*ypi - (t5*xpi).

```

```

partJ _ (self markerJ) part.

```

```

xqj _ ((self markerJ) position) at: 1. yqj _ ((self markerJ) position) at: 2. zqj _ ((self markerJ) position) at: 3.

```

```

aq _ ((partJ centerOfMassMarker) position) at: 4. bq _ ((partJ centerOfMassMarker) position) at: 5.

```

```

cq _ ((partJ centerOfMassMarker) position) at: 6.

```

```

sAq _ aq sin. sBq _ bq sin. sCq _ cq sin. cAq _ aq cos. cBq _ bq cos. cCq _ cq cos.

```

```

t1 _ -1*sAq*sBq.

```

```

t2 _ cAq*sCq + (sAq*cBq*cCq).

```

```

t3 _ sAq*cBq*sCq - (cAq*cCq).

```

```

t5 _ sAq*sCq - (cAq*cBq*cCq).

```

```

t6 _ -1*(cAq*cBq*sCq + (sAq*cCq)).

```

```

t7 _ cAq*cBq.

```

```

t8 _ cAq*sBq*cCq.

```

```

t9 _ cAq*sBq*sCq.

```

```

a08 _ -1. a10 _ t1*zqj + (t2*yqj) + (t3*xqj). a11 _ t7*zqj + (t8*yqj) + (t9*xqj). a12 _ -1*t6*yqj + (t5*xqj).

```

```

(rowA at: 1) at: 2 put: a02. (rowA at: 1) at: 4 put: a04. (rowA at: 1) at: 5 put: a05. (rowA at: 1) at: 6 put: a06.

```

```

(rowA at: 1) at: 8 put: a08. (rowA at: 1) at: 10 put: a10. (rowA at: 1) at: 11 put: a11. (rowA at: 1) at: 12 put: a12.

```

```

^rowA

```

sub31AtPoint3

"returns the values of the elements of the matrix dG3/dq for an At-Point constraint, where $G3 = (dij)z = 0$. The size of the matrix is (1 by 12)."

```

| rowA partI xpi ypi zpi bp cp sBp sCp cBp cCp partJ xqj yqj zqj bq cq sBq sCq cBq cCq t18 t19 t20 t21 a03 a05
a06 a09 a11 a12 |

```

```

rowA _ (Array new: 1) collect: [: row | Array new: 12 withAll: 0].
partI _ (self markerI) part.
xpi _ ((self markerI) position) at: 1. ypi _ ((self markerI) position) at: 2. zpi _ ((self markerI) position) at: 3.
bp _ ((partI centerOfMassMarker) position) at: 5. cp _ ((partI centerOfMassMarker) position) at: 6.
sBp _ bp sin. sCp _ cp sin. cBp _ bp cos. cCp _ cp cos.
t18 _ -1*sBp*cCp.
t19 _ -1*sBp*sCp.
t20 _ cBp*cCp.
t21 _ -1*cBp*sCp.
a03 _ 1. a05 _ -1*sBp*zpi + (t20*ypi) - (t21*xpi). a06 _ t19*ypi - (t18*xpi).
partJ _ (self markerJ) part.
xqj _ ((self markerJ) position) at: 1. yqj _ ((self markerJ) position) at: 2. zqj _ ((self markerJ) position) at: 3.
bq _ ((partJ centerOfMassMarker) position) at: 5. cq _ ((partJ centerOfMassMarker) position) at: 6.
sBq _ bq sin. sCq _ cq sin. cBq _ bq cos. cCq _ cq cos.
t18 _ -1*sBq*cCq.
t19 _ -1*sBq*sCq.
t20 _ cBq*cCq.
t21 _ -1*cBq*sCq.
a09 _ -1.
a11 _ sBq*zqj - (t20*yqj) + (t21*xqj).
a12 _ -1*t19*yqj + (t18*xqj).
(rowA at: 1) at: 3 put: a03. (rowA at: 1) at: 5 put: a05. (rowA at: 1) at: 6 put: a06. (rowA at: 1) at: 9 put: a09.
(rowA at: 1) at: 11 put: a11. (rowA at: 1) at: 12 put: a12.
^rowA

```

sub31InPlane1

"returns the values of the elements of the matrix dG1/dq for an In-Plane constraint, where $G1 = (dij)(xj) = 0$. The size of the matrix is (1 by 12)."

```

krowA matrixZ partI partJ z11 z21 z31 xpi ypi zpi xp yp zp ap bp cp xqj yqj zqj xq yq zq aq bq cq sAp cAp sBp cBp
sCp cCp sAq cAq sBq cBq sCq cCq a01 a02 a03 a04 a05 a06 a07 a08 a09 a10 a11 a12 t1 t2 t3 t4 t5 t6 t7 t8 t9 t10
t11 t12 t13 t14 t15 t16 t17 t18 t19 t20 t21 t22 t23 t24 t25 t26 t27 t28 t29 t30 t31 t32 t33 t34 t35 t36 t37 t38 t39 t40 t41 t42 t43 t44
sum1 sum2 sum3 sum4 sum5 sum6 sum7 sum8 sum9 sum10 sum11 sum12 sum13 sum14 sum15 sum16 sum17 sum18 |
rowA _ (Array new: 1) collect: [: row | Array new: 12 withAll: 0].
matrixZ _ (self markerJ) transformationMatrix.
z11 _ (matrixZ at: 1) at: 1. z21 _ (matrixZ at: 2) at: 1. z31 _ (matrixZ at: 3) at: 1.
partI _ (self markerI) part.
xpi _ (markerI position) at: 1. ypi _ (markerI position) at: 2. zpi _ (markerI position) at: 3.
xp _ ((partI centerOfMassMarker) position) at: 1. yp _ ((partI centerOfMassMarker) position) at: 2.
zp _ ((partI centerOfMassMarker) position) at: 3.
ap _ ((partI centerOfMassMarker) position) at: 4. bp _ ((partI centerOfMassMarker) position) at: 5.
cp _ ((partI centerOfMassMarker) position) at: 6.
sAp _ ap sin. sBp _ bp sin. sCp _ cp sin. cAp _ ap cos. cBp _ bp cos. cCp _ cp cos.
partJ _ (self markerJ) part.
xqj _ (markerJ position) at: 1. yqj _ (markerJ position) at: 2. zqj _ (markerJ position) at: 3.
xq _ ((partJ centerOfMassMarker) position) at: 1. yq _ ((partJ centerOfMassMarker) position) at: 2.
zq _ ((partJ centerOfMassMarker) position) at: 3.
aq _ ((partJ centerOfMassMarker) position) at: 4. bq _ ((partJ centerOfMassMarker) position) at: 5.
cq _ ((partJ centerOfMassMarker) position) at: 6.
sAq _ aq sin. sBq _ bq sin. sCq _ cq sin. cAq _ aq cos. cBq _ bq cos. cCq _ cq cos.
sum1 _ t1*z31 + (t2*z21) + (t3*z11). sum2 _ t1*zqj + (t2*yqj) + (t3*xqj). sum3 _ t7*z31 + (t8*z21) + (t9*z11).
sum4 _ t7*zqj + (t8*yqj) + (t9*xqj). sum5 _ t10*zpi + (t11*ypi) + (t12*xpi). sum6 _ t4*zpi + (t5*ypi) +
(t6*xpi).
sum7 _ t38*z31 + (t39*z21) + (t40*z11). sum8 _ t35*z31 + (t36*z21) + (t37*z11). sum9 _ t27*z31 + (t28*z21)
+ (t29*z11).
sum10 _ t41*z31 + (t42*z21) + (t43*z11). sum11 _ t27*zqj + (t28*yqj) + (t29*xqj). sum12 _ t30*zpi +
(t31*ypi) + (t32*xpi).
sum13 _ t41*zqj + (t42*yqj) + (t43*xqj). sum14 _ t35*zqj + (t36*yqj) + (t37*xqj). sum15 _ t38*zqj + (t39*yqj) +
(t40*xqj).
sum16 _ t48*zpi + (t34*ypi) - (t33*xpi). sum17 _ t13*zpi + (t14*ypi) + (t15*xpi). sum18 _ t16*zpi + (t17*ypi)
+ (t18*xpi).
a01 _ sum1. a02 _ sum3. a03 _ sum9.
a04 _ -1*sum3*sum6 + sum1*sum5.

```

```

a05 _ sum9*sum16 + sum1*sum17 - sum3*sum18.
a06 _ (t6*ypi - (t5*xpi))*sum1 + ((t12*ypi - (t11*xpi))*sum3) + ((t32*ypi - (t31*xpi))*sum9).
a07 _ -1*a01. a08 _ -1*a02. a09 _ -1*a03.
a10 _ sum3*(sum6 + xq - xp) - (sum1*(sum5 + yq - yp)).
a11 _ sum8*(sum2 + sum6 + xq - xp) + (sum7*(sum4 + sum5 + yq - yp)) + (sum9*sum13) + (sum1*sum14) +
(sum3*sum15) + (sum10*(sum11 + sum12 + zq - zp)).
a12 _ (t3*z21 - (t2*z11))*(sum2 + sum6 + xq - xp) + ((t9*z21 - (t8*z11))*(sum4 + sum5 + yq - xp))
+ ((t29*z21 - (t28*z11))*(sum11 + sum12 + zq - zp)) + ((t3*yqj - (t2*xqj))*sum1)
+ ((t9*yqj - (t8*xqj))*sum3) + ((t29*yqj - (t28*xqj))*sum9).
(rowA at: 1) at: 1 put: a01. (rowA at: 1) at: 2 put: a02. (rowA at: 1) at: 3 put: a03. (rowA at: 1) at: 4 put: a04.
(rowA at: 1) at: 5 put: a05. (rowA at: 1) at: 6 put: a06. (rowA at: 1) at: 7 put: a07. (rowA at: 1) at: 8 put: a08.
(rowA at: 1) at: 9 put: a09. (rowA at: 1) at: 10 put: a10. (rowA at: 1) at: 11 put: a11. (rowA at: 1) at: 12 put: a12.
^rowA

```

sub31InPlane2

"returns the values of the elements of the matrix dG2/dq for an In-Plane constraint, where $G2 = (dij)(y_j) = 0$. The size of the matrix is (1 by 12)."

```

rowA matrixZ partI partJ z11 z21 z31 xpi ypi zpi xp yp zp ap bp cp xqj yqj zqj xq yq zq aq bq cq sAp cAp sBp cBp
sCp cCp sAq cAq sBq cBq sCq cCq a01 a02 a03 a04 a05 a06 a07 a08 a09 a10 a11 a12 t1 t2 t3 t4 t5 t6 t7 t8 t9 t10
t11 t12 t13 t14 t15 t16 t17 t18 t27 t28 t29 t30 t31 t32 t33 t34 t35 t36 t37 t38 t39 t40 t41 t42 t43 t48 sum1 sum2
sum3 sum4 sum5 sum6 sum7 sum8 sum9 sum10 sum11 sum12 sum13 sum14 sum15 sum16 sum17 sum18 |
rowA _ (Array new: 1) collect: [: row | Array new: 12 withAll: 0].
matrixZ _ (self markerJ) transformationMatrix.
z11 _ (matrixZ at: 1) at: 2. z21 _ (matrixZ at: 2) at: 2. z31 _ (matrixZ at: 3) at: 2.
partI _ (self markerI) part.
xpi _ (markerI position) at: 1. ypi _ (markerI position) at: 2. zpi _ (markerI position) at: 3.
xp _ ((partI centerOfMassMarker) position) at: 1. yp _ ((partI centerOfMassMarker) position) at: 2.
zp _ ((partI centerOfMassMarker) position) at: 3.
ap _ ((partI centerOfMassMarker) position) at: 4. bp _ ((partI centerOfMassMarker) position) at: 5.
cp _ ((partI centerOfMassMarker) position) at: 6.
sAp _ ap sin. sBp _ bp sin. sCp _ cp sin. cAp _ ap cos. cBp _ bp cos. cCp _ cp cos.
partJ _ (self markerJ) part.
xqj _ (markerJ position) at: 1. yqj _ (markerJ position) at: 2. zqj _ (markerJ position) at: 3.
xq _ ((partJ centerOfMassMarker) position) at: 1. yq _ ((partJ centerOfMassMarker) position) at: 2.
zq _ ((partJ centerOfMassMarker) position) at: 3.
aq _ ((partJ centerOfMassMarker) position) at: 4. bq _ ((partJ centerOfMassMarker) position) at: 5.
cq _ ((partJ centerOfMassMarker) position) at: 6.
sAq _ aq sin. sBq _ bq sin. sCq _ cq sin. cAq _ aq cos. cBq _ bq cos. cCq _ cq cos.
sum1 _ t1*z31 + (t2*z21) + (t3*z11). sum2 _ t1*zqj + (t2*yqj) + (t3*xqj). sum3 _ t7*z31 + (t8*z21) + (t9*z11).
sum4 _ t7*zqj + (t8*yqj) + (t9*xqj). sum5 _ t10*zpi + (t11*ypi) + (t12*xpi). sum6 _ t4*zpi + (t5*ypi) +
(t6*xpi).
sum7 _ t38*z31 + (t39*z21) + (t40*z11). sum8 _ t35*z31 + (t36*z21) + (t37*z11). sum9 _ t27*z31 + (t28*z21)
+ (t29*z11).
sum10 _ t41*z31 + (t42*z21) + (t43*z11). sum11 _ t27*zqj + (t28*yqj) + (t29*xqj). sum12 _ t30*zpi +
(t31*ypi) + (t32*xpi).
sum13 _ t41*zqj + (t42*yqj) + (t43*xqj). sum14 _ t35*zqj + (t36*yqj) + (t37*xqj). sum15 _ t38*zqj + (t39*yqj) +
(t40*xqj).
sum16 _ t48*zpi + (t34*ypi) - (t33*xpi). sum17 _ t13*zpi + (t14*ypi) + (t15*xpi). sum18 _ t16*zpi + (t17*ypi)
+ (t18*xpi).
a01 _ sum1. a02 _ sum3. a03 _ sum9.
a04 _ -1*sum3*sum6 + sum1*sum5.
a05 _ sum9*sum16 + sum1*sum17 - sum3*sum18.
a06 _ (t6*ypi - (t5*xpi))*sum1 + ((t12*ypi - (t11*xpi))*sum3) + ((t32*ypi - (t31*xpi))*sum9).
a07 _ -1*a01. a08 _ -1*a02. a09 _ -1*a03.
a10 _ sum3*(sum6 + xq - xp) - (sum1*(sum5 + yq - yp)).
a11 _ sum8*(sum2 + sum6 + xq - xp) + (sum7*(sum4 + sum5 + yq - yp)) + (sum9*sum13) + (sum1*sum14) +
(sum3*sum15) + (sum10*(sum11 + sum12 + zq - zp)).
a12 _ (t3*z21 - (t2*z11))*(sum2 + sum6 + xq - xp) + ((t9*z21 - (t8*z11))*(sum4 + sum5 + yq - xp))
+ ((t29*z21 - (t28*z11))*(sum11 + sum12 + zq - zp)) + ((t3*yqj - (t2*xqj))*sum1)
+ ((t9*yqj - (t8*xqj))*sum3) + ((t29*yqj - (t28*xqj))*sum9).
(rowA at: 1) at: 1 put: a01. (rowA at: 1) at: 2 put: a02. (rowA at: 1) at: 3 put: a03. (rowA at: 1) at: 4 put: a04.
(rowA at: 1) at: 5 put: a05. (rowA at: 1) at: 6 put: a06. (rowA at: 1) at: 7 put: a07. (rowA at: 1) at: 8 put: a08.

```

(rowA at: 1) at: 9 put: a09. (rowA at: 1) at: 10 put: a10. (rowA at: 1) at: 11 put: a11. (rowA at: 1) at: 12 put: a12.
^rowA

sub31InPlane3

"returns the values of the elements of the matrix dG3/dq for an In-Plane constraint, where $G3 = (dij)(zj) = 0$. The size of the matrix is (1 by 12)."

rowA matrixZ partI partJ z11 z21 z31 xpi ypi zpi xp yp zp ap bp cp xqj yqj zqj xq yq zq aq bq cq sAp cAp sBp cBp sCp cCp sAq cAq sBq cBq sCq cCq a01 a02 a03 a04 a05 a06 a07 a08 a09 a10 a11 a12 t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t13 t14 t15 t16 t17 t18 t27 t28 t29 t30 t31 t32 t33 t34 t35 t36 t37 t38 t39 t40 t41 t42 t43 t48 sum1 sum2 sum3 sum4 sum5 sum6 sum7 sum8 sum9 sum10 sum11 sum12 sum13 sum14 sum15 sum16 sum17 sum18 |

rowA _ (Array new: 1) collect: [: row | Array new: 12 withAll: 0].

matrixZ _ (self markerJ) transformationMatrix.

z11 _ (matrixZ at: 1) at: 3. z21 _ (matrixZ at: 2) at: 3. z31 _ (matrixZ at: 3) at: 3.

partI _ (self markerI) part.

xpi _ (markerI position) at: 1. ypi _ (markerI position) at: 2. zpi _ (markerI position) at: 3.

xp _ ((partI centerOfMassMarker) position) at: 1. yp _ ((partI centerOfMassMarker) position) at: 2.

zp _ ((partI centerOfMassMarker) position) at: 3.

ap _ ((partI centerOfMassMarker) position) at: 4. bp _ ((partI centerOfMassMarker) position) at: 5.

cp _ ((partI centerOfMassMarker) position) at: 6.

sAp _ ap sin. sBp _ bp sin. sCp _ cp sin. cAp _ ap cos. cBp _ bp cos. cCp _ cp cos.

partJ _ (self markerJ) part.

xqj _ (markerJ position) at: 1. yqj _ (markerJ position) at: 2. zqj _ (markerJ position) at: 3.

xq _ ((partJ centerOfMassMarker) position) at: 1. yq _ ((partJ centerOfMassMarker) position) at: 2.

zq _ ((partJ centerOfMassMarker) position) at: 3.

aq _ ((partJ centerOfMassMarker) position) at: 4. bq _ ((partJ centerOfMassMarker) position) at: 5.

cq _ ((partJ centerOfMassMarker) position) at: 6.

sAq _ aq sin. sBq _ bq sin. sCq _ cq sin. cAq _ aq cos. cBq _ bq cos. cCq _ cq cos.

sum1 _ t1*z31 + (t2*z21) + (t3*z11). sum2 _ t1*zqj + (t2*yqj) + (t3*xqj). sum3 _ t7*z31 + (t8*z21) + (t9*z11).

sum4 _ t7*zqj + (t8*yqj) + (t9*xqj). sum5 _ t10*zpi + (t11*yqi) + (t12*xpi). sum6 _ t4*zpi + (t5*yqi) + (t6*xpi).

sum7 _ t38*z31 + (t39*z21) + (t40*z11). sum8 _ t35*z31 + (t36*z21) + (t37*z11). sum9 _ t27*z31 + (t28*z21) + (t29*z11).

sum10 _ t41*z31 + (t42*z21) + (t43*z11). sum11 _ t27*zqj + (t28*yqj) + (t29*xqj). sum12 _ t30*zpi + (t31*yqi) + (t32*xpi).

sum13 _ t41*zqj + (t42*yqj) + (t43*xqj). sum14 _ t35*zqj + (t36*yqj) + (t37*xqj). sum15 _ t38*zqj + (t39*yqj) + (t40*xqj).

sum16 _ t48*zpi + (t34*yqi) - (t33*xpi). sum17 _ t13*zpi + (t14*yqi) + (t15*xpi). sum18 _ t16*zpi + (t17*yqi) + (t18*xpi).

a01 _ sum1. a02 _ sum3. a03 _ sum9.

a04 _ -1*sum3*sum6 + sum1*sum5.

a05 _ sum9*sum16 + sum1*sum17 - sum3*sum18.

a06 _ (t6*yqi - (t5*xpi))*sum1 + ((t12*yqi - (t11*xpi))*sum3) + ((t32*yqi - (t31*xpi))*sum9).

a07 _ -1*a01. a08 _ -1*a02. a09 _ -1*a03.

a10 _ sum3*(sum6 + xq - xp) - (sum1*(sum5 + yq - yp)).

a11 _ sum8*(sum2 + sum6 + xq - xp) + (sum7*(sum4 + sum5 + yq - yp)) + (sum9*sum13) + (sum1*sum14) + (sum3*sum15) + (sum10*(sum11 + sum12 + zq - zp)).

a12 _ (t3*z21 - (t2*z11))*(sum2 + sum6 + xq - xp) + ((t9*z21 - (t8*z11))*(sum4 + sum5 + yq - xp)) + ((t29*z21 - (t28*z11))*(sum11 + sum12 + zq - zp)) + ((t3*yqj - (t2*xqj))*sum1) + ((t9*yqj - (t8*xqj))*sum3) + ((t29*yqj - (t28*xqj))*sum9).

(rowA at: 1) at: 1 put: a01. (rowA at: 1) at: 2 put: a02. (rowA at: 1) at: 3 put: a03. (rowA at: 1) at: 4 put: a04.

(rowA at: 1) at: 5 put: a05. (rowA at: 1) at: 6 put: a06. (rowA at: 1) at: 7 put: a07. (rowA at: 1) at: 8 put: a08.

(rowA at: 1) at: 9 put: a09. (rowA at: 1) at: 10 put: a10. (rowA at: 1) at: 11 put: a11. (rowA at: 1) at: 12 put: a12.

^rowA

sub31Perpendicular1

"returns the values of the elements of the matrix dG1/dq for a Perpendicular constraint, where $G1 = (zi)(xj) = 0$. The size of the matrix is (1 by 12)."

rowA matrixW matrixZ partI partJ ap bp cp aq bq cq sAp cAp sBp cBp sCp cCp sAq cAq sBq cBq sCq cCq a04 a05 a06 a10 a11 a12 t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t13 t14 t15 t16 t17 t18 t27 t28 t29 t30 t31 t32 t33 t34 t35 t36 t37 t38 t39 t40 t41 t42 t43 t48 sum1 sum3 sum7 sum8 sum9 sum10 w31 w32 w33 z11 z21 z31 |

rowA _ (Array new: 1) collect: [: row | Array new: 12 withAll: 0].

matrixW _ (self markerI) transformationMatrix. matrixZ _ (self markerJ) transformationMatrix.


```

w31 _ (matrixW at: 3) at: 1. w32 _ (matrixW at: 3) at: 2. w33 _ (matrixW at: 3) at: 3.
z11 _ (matrixZ at: 1) at: 1. z21 _ (matrixZ at: 2) at: 1. z31 _ (matrixZ at: 3) at: 1.
partI _ (self markerI) part.
ap _ ((partI centerOfMassMarker) position) at: 4. bp _ ((partI centerOfMassMarker) position) at: 5.
cp _ ((partI centerOfMassMarker) position) at: 6.
sAp _ ap sin. sBp _ bp sin. sCp _ cp sin. cAp _ ap cos. cBp _ bp cos. cCp _ cp cos.
partJ _ (self markerJ) part.
aq _ ((partJ centerOfMassMarker) position) at: 4. bq _ ((partJ centerOfMassMarker) position) at: 5.
cq _ ((partJ centerOfMassMarker) position) at: 6.
sAq _ aq sin. sBq _ bq sin. sCq _ cq sin. cAq _ aq cos. cBq _ bq cos. cCq _ cq cos.
sum1 _ t1*z31 + (t2*z21) + (t3*z11). sum3 _ t7*z31 + (t8*z21) + (t9*z11). sum7 _ t38*z31 + (t39*z21) +
(t40*z11).
sum8 _ t35*z31 + (t36*z21) + (t37*z11). sum9 _ t27*z31 + (t28*z21) + (t29*z11). sum10 _ t41*z31 +
(t42*z21) + (t43*z11).
a04 _ w31*(t12*sum1 - (t6*sum3)) + (w32*(t11*sum1 - (t5*sum3))) + (w33*(t10*sum1 - (t4*sum3))).
a05 _ w31*(t15*sum1 - (t18*sum3) - (t33*sum9)) + (w32*(t14*sum1 - (t17*sum3) + (t34*sum9)))
+ (w33*(t13*sum1 - (t16*sum3) + (t48*sum9))).
a06 _ w32*(t6*sum1 + (t12*sum3) + (t32*sum9)) - (w31*(t5*sum1 + (t11*sum3) + (t31*sum9))).
a10 _ w31*(-1*t12*sum1 + (t6*sum3)) + (w32*(-1*t11*sum1 + (t5*sum3))) + (w33*(-1*t10*sum1 +
(t4*sum3))).
a11 _ w31*(t32*sum10 + (t6*sum8) + (t12*sum7)) + (w32*(t31*sum10 + (t5*sum8) + (t11*sum7)))
+ (w33*(t30*sum10 + (t4*sum8) + (t10*sum7))).
a12 _ w31*(t6*(t3*z21 - (t2*z11)) + (t12*(t9*z21 - (t8*z11))) + (t32*(t29*z21 - (t28*z11))))
+ (w32*(t5*(t3*z21 - (t2*z11)) + (t11*(t9*z21 - (t8*z11))) + (t31*(t29*z21 - (t28*z11))))
+ (w33*(t4*(t3*z21 - (t2*z11)) + (t10*(t9*z21 - (t8*z11))) + (t30*(t29*z21 - (t28*z11))))).
(rowA at: 1) at: 4 put: a04. (rowA at: 1) at: 5 put: a05. (rowA at: 1) at: 6 put: a06. (rowA at: 1) at: 10 put: a10.
(rowA at: 1) at: 11 put: a11. (rowA at: 1) at: 12 put: a12.
^rowA

```

sub31Perpendicular2

"returns the values of the elements of the matrix dG2/dq for a Perpendicular constraint, where $G2 = (z_i)(y_j) = 0$. The size of the matrix is (1 by 12)."

```

rowA matrixW matrixZ partI partJ ap bp cp aq bq cq sAp cAp sBp cBp sCp cCp sAq cAq sBq cBq sCq cCq a04 a05
a06 a10 a11 a12 t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t13 t14 t15 t16 t17 t18 t27 t28 t29 t30 t31 t32 t33 t34 t35
t36 t37 t38 t39 t40 t41 t42 t43 t48 sum1 sum3 sum7 sum8 sum9 sum10 w31 w32 w33 z11 z21 z31 |
rowA _ (Array new: 1) collect: [: row | Array new: 12 withAll: 0].
matrixW _ (self markerI) transformationMatrix. matrixZ _ (self markerJ) transformationMatrix.
w31 _ (matrixW at: 3) at: 1. w32 _ (matrixW at: 3) at: 2. w33 _ (matrixW at: 3) at: 3.
z11 _ (matrixZ at: 1) at: 2. z21 _ (matrixZ at: 2) at: 2. z31 _ (matrixZ at: 3) at: 2.
partI _ (self markerI) part.
ap _ ((partI centerOfMassMarker) position) at: 4. bp _ ((partI centerOfMassMarker) position) at: 5.
cp _ ((partI centerOfMassMarker) position) at: 6.
sAp _ ap sin. sBp _ bp sin. sCp _ cp sin. cAp _ ap cos. cBp _ bp cos. cCp _ cp cos.
partJ _ (self markerJ) part.
aq _ ((partJ centerOfMassMarker) position) at: 4. bq _ ((partJ centerOfMassMarker) position) at: 5.
cq _ ((partJ centerOfMassMarker) position) at: 6.
sAq _ aq sin. sBq _ bq sin. sCq _ cq sin. cAq _ aq cos. cBq _ bq cos. cCq _ cq cos.
sum1 _ t1*z31 + (t2*z21) + (t3*z11). sum3 _ t7*z31 + (t8*z21) + (t9*z11). sum7 _ t38*z31 + (t39*z21) +
(t40*z11).
sum8 _ t35*z31 + (t36*z21) + (t37*z11). sum9 _ t27*z31 + (t28*z21) + (t29*z11). sum10 _ t41*z31 +
(t42*z21) + (t43*z11).
a04 _ w31*(t12*sum1 - (t6*sum3)) + (w32*(t11*sum1 - (t5*sum3))) + (w33*(t10*sum1 - (t4*sum3))).
a05 _ w31*(t15*sum1 - (t18*sum3) - (t33*sum9)) + (w32*(t14*sum1 - (t17*sum3) + (t34*sum9)))
+ (w33*(t13*sum1 - (t16*sum3) + (t48*sum9))).
a06 _ w32*(t6*sum1 + (t12*sum3) + (t32*sum9)) - (w31*(t5*sum1 + (t11*sum3) + (t31*sum9))).
a10 _ w31*(-1*t12*sum1 + (t6*sum3)) + (w32*(-1*t11*sum1 + (t5*sum3))) + (w33*(-1*t10*sum1 -
(t4*sum3))).
a11 _ w31*(t32*sum10 + (t6*sum8) + (t12*sum7)) + (w32*(t31*sum10 + (t5*sum8) + (t11*sum7)))
+ (w33*(t30*sum10 + (t4*sum8) + (t10*sum7))).
a12 _ w31*(t6*(t3*z21 - (t2*z11)) + (t12*(t9*z21 - (t8*z11))) + (t32*(t29*z21 - (t28*z11))))
+ (w32*(t5*(t3*z21 - (t2*z11)) + (t11*(t9*z21 - (t8*z11))) + (t31*(t29*z21 - (t28*z11))))
+ (w33*(t4*(t3*z21 - (t2*z11)) + (t10*(t9*z21 - (t8*z11))) + (t30*(t29*z21 - (t28*z11))))).

```

(rowA at: 1) at: 4 put: a04. (rowA at: 1) at: 5 put: a05. (rowA at: 1) at: 6 put: a06. (rowA at: 1) at: 10 put: a10.
 (rowA at: 1) at: 11 put: a11. (rowA at: 1) at: 12 put: a12.
 ^rowA

sub31Perpendicular3

"returns the values of the elements of the matrix dG3/dq for a Perpendicular constraint, where $G3 = (z_i)(z_j) = 0$. The size of the matrix is (1 by 12)."

```
lrowA matrixW matrixZ partI partJ ap bp cp aq bq cq sAp cAp sBp cBp sCp cCp sAq cAq sBq cBq sCq cCq a04 a05
a06 a10 a11 a12 t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t13 t14 t15 t16 t17 t18 t27 t28 t29 t30 t31 t32 t33 t34 t35
t36 t37 t38 t39 t40 t41 t42 t43 t48 sum1 sum3 sum7 sum8 sum9 sum10 w31 w32 w33 z11 z21 z31 |
rowA _ (Array new: 1) collect: [: row | Array new: 12 withAll: 0].
matrixW _ (self markerI) transformationMatrix. matrixZ _ (self markerJ) transformationMatrix.
w31 _ (matrixW at: 3) at: 1. w32 _ (matrixW at: 3) at: 2. w33 _ (matrixW at: 3) at: 3.
z11 _ (matrixZ at: 1) at: 3. z21 _ (matrixZ at: 2) at: 3. z31 _ (matrixZ at: 3) at: 3.
partI _ (self markerI) part.
ap _ ((partI centerOfMassMarker) position) at: 4. bp _ ((partI centerOfMassMarker) position) at: 5.
cp _ ((partI centerOfMassMarker) position) at: 6.
sAp _ ap sin. sBp _ bp sin. sCp _ cp sin. cAp _ ap cos. cBp _ bp cos. cCp _ cp cos.
partJ _ (self markerJ) part.
aq _ ((partJ centerOfMassMarker) position) at: 4. bq _ ((partJ centerOfMassMarker) position) at: 5.
cq _ ((partJ centerOfMassMarker) position) at: 6.
sAq _ aq sin. sBq _ bq sin. sCq _ cq sin. cAq _ aq cos. cBq _ bq cos. cCq _ cq cos.
sum1 _ t1*z31 + (t2*z21) + (t3*z11). sum3 _ t7*z31 + (t8*z21) + (t9*z11). sum7 _ t38*z31 + (t39*z21) +
(t40*z11).
sum8 _ t35*z31 + (t36*z21) + (t37*z11). sum9 _ t27*z31 + (t28*z21) + (t29*z11). sum10 _ t41*z31 +
(t42*z21) + (t43*z11).
a04 _ w31*(t12*sum1 - (t6*sum3)) + (w32*(t11*sum1 - (t5*sum3))) + (w33*(t10*sum1 - (t4*sum3))).
a05 _ w31*(t15*sum1 - (t18*sum3) - (t33*sum9)) + (w32*(t14*sum1 - (t17*sum3) + (t34*sum9)))
+ (w33*(t13*sum1 - (t16*sum3) + (t48*sum9))).
a06 _ w32*(t6*sum1 + (t12*sum3) + (t32*sum9)) - (w31*(t5*sum1 + (t11*sum3) + (t31*sum9))).
a10 _ w31*(-1*t12*sum1 + (t6*sum3)) + (w32*(-1*t11*sum1 + (t5*sum3))) + (w33*(-1*t10*sum1 +
(t4*sum3))).
a11 _ w31*(t32*sum10 + (t6*sum8) + (t12*sum7)) + (w32*(t31*sum10 + (t5*sum8) + (t11*sum7)))
+ (w33*(t30*sum10 + (t4*sum8) + (t10*sum7))).
a12 _ w31*(t6*(t3*z21 - (t2*z11)) + (t12*(t9*z21 - (t8*z11))) + (t32*(t29*z21 - (t28*z11))))
+ (w32*(t5*(t3*z21 - (t2*z11)) + (t11*(t9*z21 - (t8*z11))) + (t31*(t29*z21 - (t28*z11))))
+ (w33*(t4*(t3*z21 - (t2*z11)) + (t10*(t9*z21 - (t8*z11))) + (t30*(t29*z21 - (t28*z11))))).
(rowA at: 1) at: 4 put: a04. (rowA at: 1) at: 5 put: a05. (rowA at: 1) at: 6 put: a06. (rowA at: 1) at: 10 put: a10.
(rowA at: 1) at: 11 put: a11. (rowA at: 1) at: 12 put: a12.
^rowA
```

sub31Perpendicular4

"returns the values of the elements of the matrix dG4/dq for a Perpendicular constraint, where $G4 = (x_i)(y_j) = 0$. The size of the matrix is (1 by 12)."

```
lrowA matrixW matrixZ partI partJ ap bp cp aq bq cq sAp cAp sBp cBp sCp cCp sAq cAq sBq cBq sCq cCq a04 a05
a06 a10 a11 a12 t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t13 t14 t15 t16 t17 t18 t27 t28 t29 t30 t31 t32 t33 t34 t35
t36 t37 t38 t39 t40 t41 t42 t43 t48 sum1 sum3 sum7 sum8 sum9 sum10 w31 w32 w33 z11 z21 z31 |
rowA _ (Array new: 1) collect: [: row | Array new: 12 withAll: 0].
matrixW _ (self markerI) transformationMatrix.
matrixZ _ (self markerJ) transformationMatrix.
w31 _ (matrixW at: 1) at: 1. w32 _ (matrixW at: 1) at: 2. w33 _ (matrixW at: 1) at: 3.
z11 _ (matrixZ at: 1) at: 2. z21 _ (matrixZ at: 2) at: 2. z31 _ (matrixZ at: 3) at: 2.
partI _ (self markerI) part.
(partI partNo) = '00'
  ifFalse: [
    ap _ ((partI centerOfMassMarker) position) at: 4.
    bp _ ((partI centerOfMassMarker) position) at: 5.
    cp _ ((partI centerOfMassMarker) position) at: 6]
  ifTrue: [ ap _ 0. bp _ 0. cp _ 0].
sAp _ ap sin. sBp _ bp sin. sCp _ cp sin. cAp _ ap cos. cBp _ bp cos. cCp _ cp cos.
partJ _ (self markerJ) part.
(partJ partNo) = '00'
```

```

ifFalse: [
    aq _ ((partJ centerOfMassMarker) position) at: 4.
    bq _ ((partJ centerOfMassMarker) position) at: 5.
    cq _ ((partJ centerOfMassMarker) position) at: 6]
ifTrue: [ aq _ 0. bq _ 0. cq _ 0].
sAq _ aq sin. sBq _ bq sin. sCq _ cq sin. cAq _ aq cos. cBq _ bq cos. cCq _ cq cos.
sum1 _ t1*z31 + (t2*z21) + (t3*z11). sum3 _ t7*z31 + (t8*z21) + (t9*z11). sum7 _ t38*z31 + (t39*z21) +
(t40*z11).
sum8 _ t35*z31 + (t36*z21) + (t37*z11). sum9 _ t27*z31 + (t28*z21) + (t29*z11). sum10 _ t41*z31 +
(t42*z21) + (t43*z11).
a04 _ w31*(t12*sum1 - (t6*sum3)) + (w32*(t11*sum1 - (t5*sum3))) + (w33*(t10*sum1 - (t4*sum3))).
a05 _ w31*(t15*sum1 - (t18*sum3) - (t33*sum9)) + (w32*(t14*sum1 - (t17*sum3) + (t34*sum9)))
+ (w33*(t13*sum1 - (t16*sum3) + (t48*sum9))).
a06 _ w32*(t6*sum1 + (t12*sum3) + (t32*sum9)) - (w31*(t5*sum1 + (t11*sum3) + (t31*sum9))).
a10 _ w31*(-1*t12*sum1 + (t6*sum3)) + (w32*(-1*t11*sum1 + (t5*sum3))) + (w33*(-1*t10*sum1 + (t4*sum3))).
a11 _ w31*(t32*sum10 + (t6*sum8) + (t12*sum7)) + (w32*(t31*sum10 + (t5*sum8) + (t11*sum7)))
+ (w33*(t30*sum10 + (t4*sum8) + (t10*sum7))).
a12 _ w31*(t6*(t3*z21 - (t2*z11)) + (t12*(t9*z21 - (t8*z11))) + (t32*(t29*z21 - (t28*z11))))
+ (w32*(t5*(t3*z21 - (t2*z11)) + (t11*(t9*z21 - (t8*z11))) + (t31*(t29*z21 - (t28*z11))))
+ (w33*(t4*(t3*z21 - (t2*z11)) + (t10*(t9*z21 - (t8*z11))) + (t30*(t29*z21 - (t28*z11))))).
(rowA at: 1) at: 4 put: a04. (rowA at: 1) at: 5 put: a05. (rowA at: 1) at: 6 put: a06.
(rowA at: 1) at: 10 put: a10. (rowA at: 1) at: 11 put: a11. (rowA at: 1) at: 12 put: a12.
^rowA

```

B.7 Damper Class

System subclass: #Damper

instanceVariableNames: 'dampingCoeff markerI markerJ type'

classVariableNames: "

poolDictionaries: "

category: 'Multibody System'

dampingForceOnMarkerI

"Solves the damping force acting on marker I and returns the force matrix (3 by 1)."

| lengthArray lengthDotArray sum dum coeff forceArray |

sum _ 0.

lengthArray _ (self markerI) relativePositionWith: (self markerJ).

lengthArray do: [: elem | sum _ sum + (elem*elem)].

lengthDotArray _ (self markerI) relativeVelocityWith: (self markerJ).

dum _ super multiply: (super transpose: lengthArray) with: lengthDotArray.

coeff _ -1*(self dampingCoeff)*dum/sum.

forceArray _ lengthArray collect: [: elem | coeff*elem].

^ forceArray

forceTorqueOnPartIJ

"returns the force and torque contribution on part I and J (12 by 1) done by a damper."

| partI partJ aMarkerI aMarkerJ x y z phi the psi sA sB sC cA cB cC t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t18 t19 t20

t21 dLxdPhi dLxdThe dLxdPsi dLydPhi dLydThe dLydPsi dLzdThe dLzdPsi dLdPhi dLdThe dLdPsi forceArrayt term1

term2 term3 forceArray torqueArray forceTorqueArrayI forceTorqueArrayJ forceTorqueArrayIJ |

"----- For Part I -----"

"Force Contribution"

forceArray _ self dampingForceOnMarkerI.

"Torque Contribution"

aMarkerI _ self markerI. partI _ aMarkerI part.

x _ (aMarkerI position) at: 1. y _ (aMarkerI position) at: 2. z _ (aMarkerI position) at: 3.

phi _ ((partI centerOfMassMarker) position) at: 4. the _ ((partI centerOfMassMarker) position) at: 5.

psi _ ((partI centerOfMassMarker) position) at: 6.

sA _ phi sin. sB _ the sin. sC _ psi sin. cA _ phi cos. cB _ the cos. cC _ psi cos.

t1 _ -1*sA*sB.

```

t2 _ cA*sC + (sA*cB*cC).
t3 _ sA*cB*sC - (cA*cC).
t4 _ cA*sB.
t5 _ sA*sC - (cA*cB*cC).
t6 _ -1*(cA*cB*sC + (sA*cC)).
t7 _ cA*cB.
t8 _ cA*sB*cC.
t9 _ cA*sB*sC.
t10 _ sA*cB.
t11 _ sA*sB*cC.
t12 _ sA*sB*sC.
t18 _ -1*sB*cC.
t19 _ -1*sB*sC.
t20 _ cB*cC.
t21 _ -1*cB*sC.
dLxdPhi _ t4*z + (t5*y) + (t6*x).
dLydPhi _ -1*t1*z - (t2*y) - (t3*x).
dLdPhi _ Array new: 3.
dLdPhi at: 1 put: dLxdPhi; at: 2 put: dLydPhi; at: 3 put: 0.
dLxdThe _ t10*z + (t11*y) + (t12*x).
dLydThe _ -1*t7*z - (t8*y) - (t9*x).
dLzdThe _ -1*sB*z + (t20*y) - (t21*x).
dLdThe _ Array new: 3.
dLdThe at: 1 put: dLxdThe; at: 2 put: dLydThe; at: 3 put: dLzdThe.
dLxdPsi _ t3*y - (t2*x).
dLydPsi _ t6*y - (t5*x).
dLzdPsi _ t19*y - (t18*x).
dLdPsi _ Array new: 3.
dLdPsi at: 1 put: dLxdPsi; at: 2 put: dLydPsi; at: 3 put: dLzdPsi.
forceArrayt _ super transpose: forceArray.
term1 _ super multiply: forceArrayt with: dLdPhi.
term2 _ super multiply: forceArrayt with: dLdThe.
term3 _ super multiply: forceArrayt with: dLdPsi.
torqueArray _ Array new: 3.
torqueArray at: 1 put: term1; at: 2 put: term2; at: 3 put: term3.
forceTorqueArrayI _ forceArray,torqueArray.
"----- For Part J -----"
"Force Contribution"
forceArray _ super multiply: forceArray by: (-1).
"Torque Contribution"
aMarkerJ _ self markerJ. partJ _ aMarkerJ part.
x _ (aMarkerJ position) at: 1. y _ (aMarkerJ position) at: 2. z _ (aMarkerJ position) at: 3.
phi _ ((partJ centerOfMassMarker) position) at: 4. the _ ((partJ centerOfMassMarker) position) at: 5.
psi _ ((partJ centerOfMassMarker) position) at: 6.
sA _ phi sin. sB _ the sin. sC _ psi sin.
cA _ phi cos. cB _ the cos. cC _ psi cos.
t1 _ -1*sA*sB.
t2 _ cA*sC + (sA*cB*cC).
t3 _ sA*cB*sC - (cA*cC).
t4 _ cA*sB.
t5 _ sA*sC - (cA*cB*cC).
t6 _ -1*(cA*cB*sC + (sA*cC)).
t7 _ cA*cB.
t8 _ cA*sB*cC.
t9 _ cA*sB*sC.
t10 _ sA*cB.
t11 _ sA*sB*cC.
t12 _ sA*sB*sC.
t18 _ -1*sB*cC.
t19 _ -1*sB*sC.
t20 _ cB*cC.
t21 _ -1*cB*sC.

```

```

dLxdPhi _ t4*z + (t5*y) + (t6*x).
dLydPhi _ -1*t1*z - (t2*y) - (t3*x).
dLdPhi _ Array new: 3.
dLdPhi at: 1 put: dLxdPhi; at: 2 put: dLydPhi; at: 3 put: 0.
dLxdThe _ t10*z + (t11*y) + (t12*x).
dLydThe _ -1*t7*z - (t8*y) - (t9*x).
dLzdThe _ -1*sB*z + (t20*y) - (t21*x).
dLdThe _ Array new: 3.
dLdThe at: 1 put: dLxdThe; at: 2 put: dLydThe; at: 3 put: dLzdThe.
dLxdPsi _ t3*y - (t2*x).
dLydPsi _ t6*y - (t5*x).
dLzdPsi _ t19*y - (t18*x).
dLdPsi _ Array new: 3.
dLdPsi at: 1 put: dLxdPsi; at: 2 put: dLydPsi; at: 3 put: dLzdPsi.
forceArrayt _ super transpose: forceArray.
term1 _ super multiply: forceArrayt with: dLdPhi.
term2 _ super multiply: forceArrayt with: dLdThe.
term3 _ super multiply: forceArrayt with: dLdPsi.
torqueArray at: 1 put: term1; at: 2 put: term2; at: 3 put: term3.
forceTorqueArrayJ _ forceArray,torqueArray.
forceTorqueArrayIJ _ forceTorqueArrayI,forceTorqueArrayJ.
^ forceTorqueArrayIJ

```

dampingCoeff
 "Returns the damping coefficient of a damper."
 ^ dampingCoeff

markerI
 "Returns the i-marker instance related to a damper."
 ^ markerI

markerJ
 "Returns j-marker instance related to a damper."
 ^ markerJ

initializeDamperBetween: markerInstance1 and: markerInstance2 dampingCoeff: aCoeff
 "Initializes a damper using the arguments"
 markerI _ markerInstance1. markerJ _ markerInstance2.
 dampingCoeff _ aCoeff asGoodDouble.
 NumberOfDampers _ NumberOfDampers + 1.
 DamperDictionary add: self

sub41Damper
 "returns the matrix [dL/dq] for a translational damping force. This matrix is to be put into the Jacobian submatrix sub41. The matrix has the size of 6 by 12. Note that {L(damper)} = {Q} - {F(damper)}."
 | matrixI matrixJ partNoI partNoJ dumMatrix matrixIJ |
 matrixI _ (Array new: 6) collect: [: each | Array new: 6 withAll: 0].
 matrixJ _ (Array new: 6) collect: [: each | Array new: 6 withAll: 0].
 partNoI _ ((self markerI) part) partNo. partNoJ _ ((self markerJ) part) partNo.
 (partNoI ~= '00') ifTrue: [matrixI _ self sub41DamperForMarker: (self markerI)].
 (partNoJ ~= '00') ifTrue: [
 dumMatrix _ self sub41DamperForMarker: (self markerJ). matrixJ _ super multiply: dumMatrix by: (-1)].
 matrixIJ _ (Array new: 6) collect: [: each | Array new: 12].
 super put: matrixI into: matrixIJ at: (1@1). super put: matrixJ into: matrixIJ at: (7@1).
 ^ matrixIJ

sub41DamperForMarker: aMarker
 "returns the contribution of a damping force on a marker to the matrix [dL/dq]."
 | matrix c lengthArray sum length lengthArrayt c1 c2 aPart c3 c4 c5 x y z phi the psi sA sB sC cA cB cC t1 t2 t3 t4
 t5 t6 t7 t8 t9 t10 t11 t12 t13 t14 t15 t16 t17 t18 t19 t20 t21 dLxdPhi dLxdThe dLxdPsi dLydPhi dLydThe dLydPsi
 dLzdThe dLzdPsi dLdPhi dLdThe dLdPsi lengthDotArray lengthDotArrayt lengthDotX lengthDotY lengthDotZ
 phiDot theDot psiDot term1 term2 term3 term4 term5 term6 term7 term8 term9 term10 term11 term12 term13

```

term14 term15 term16 term17 term18 term19 term20 term21 term22 term23 term24 sub11 sub12 sub13 sub14
sub15 sub16 sub21 sub22 sub23 sub24 sub25 sub26 dum dum1 dum2 dum3 elem1 elem2 elem3 dummy dummy1
dummy2 dummy3 dummy4 dummy5 dummy6 dummy7 dummy8 dummy9 e1 e2 e3 e4 e5 e6 e7 e8 e9 e10 e11 e12
e13 e14 e15 sum1 sum2 sum3 sum4 sum5 sum6 sum7 sum8 sum9 array1 array2 array3 array4 array5 array6 val1
val2 val3 val4 val5 val6 val7 val8 val9 val10 val11 val12 dLdtdPhi dLdtdThe dLdtdPsi |
c _ self dampingCoeff.
lengthArray _ (self markerI) relativePositionWith: (self markerJ).
sum _ 0.
lengthArray do: [: elem | sum _ sum + (elem*elem)].
length _ sum sqrt.
lengthArrayt _ super transpose: lengthArray.
lengthDotArray _ (self markerI) relativeVelocityWith: (self markerJ).
lengthDotX _ lengthDotArray at: 1. lengthDotY _ lengthDotArray at: 2. lengthDotZ _ lengthDotArray at: 3.
lengthDotArrayt _ super transpose: lengthDotArray.
dum _ super multiply: lengthArrayt with: lengthDotArray.
c1 _ c/(length**2).
c2 _ c1*dum.
aPart _ aMarker part.
x _ (aMarker position) at: 1. y _ (aMarker position) at: 2. z _ (aMarker position) at: 3.
phi _ ((aPart centerOfMassMarker) position) at: 4. the _ ((aPart centerOfMassMarker) position) at: 5.
psi _ ((aPart centerOfMassMarker) position) at: 6.
sA _ phi sin. sB _ the sin. sC _ psi sin. cA _ phi cos. cB _ the cos. cC _ psi cos.
t1 _ -1*sA*sB.
t2 _ cA*sC + (sA*cB*cC).
t3 _ sA*cB*sC - (cA*cC).
t4 _ cA*sB.
t5 _ sA*sC - (cA*cB*cC).
t6 _ -1*(cA*cB*sC + (sA*cC)).
t7 _ cA*cB.
t8 _ cA*sB*cC.
t9 _ cA*sB*sC.
t10 _ sA*cB.
t11 _ sA*sB*cC.
t12 _ sA*sB*sC.
t13 _ sA*cB*cC.
t14 _ sA*cB*sC.
t15 _ -1*cA*cB*cC.
t16 _ -1*cA*cB*sC.
t17 _ -1*cB.
t18 _ -1*sB*cC.
t19 _ -1*sB*sC.
t20 _ cB*cC.
t21 _ -1*cB*sC.
dLxdPhi _ t4*z + (t5*y) + (t6*x).
dLydPhi _ -1*t1*z - (t2*y) - (t3*x).
dLdPhi _ Array new: 3.
dLdPhi at: 1 put: dLxdPhi; at: 2 put: dLydPhi; at: 3 put: 0.
dLxdThe _ t10*z + (t11*y) + (t12*x).
dLydThe _ -1*t7*z - (t8*y) - (t9*x).
dLzdThe _ -1*sB*z + (t20*y) - (t21*x).
dLdThe _ Array new: 3.
dLdThe at: 1 put: dLxdThe; at: 2 put: dLydThe; at: 3 put: dLzdThe.
dLxdPsi _ t3*y - (t2*x).
dLydPsi _ t6*y - (t5*x).
dLzdPsi _ t19*y - (t18*x).
dLdPsi _ Array new: 3.
dLdPsi at: 1 put: dLxdPsi; at: 2 put: dLydPsi; at: 3 put: dLzdPsi.
e1 _ t1*z + (t2*y) + (t3*x).
e2 _ t4*z + (t5*y) + (t6*x).
e3 _ t7*z + (t8*y) + (t9*x).
e4 _ t10*z + (t11*y) + (t12*x).
e5 _ -1*t6*y + (t5*x).

```

```

e6 _ t3*y - (t2*x).
e7 _ t1*z + (t13*y) + (t14*x).
e8 _ t4*z + (t15*y) + (t16*x).
e9 _ t17*z + (t18*y) + (t19*x).
e10 _ -1*t12*y + (t11*x).
e11 _ t9*y - (t8*x).
e12 _ t21*y + (t20*x).
e13 _ t2*y + (t3*x).
e14 _ t5*y + (t6*x).
e15 _ t18*y + (t19*x).
phiDot _ (aPart velocity) at: 4.
theDot _ (aPart velocity) at: 5.
psiDot _ (aPart velocity) at: 6.
"---- For 1st quadrant ----"
dummy _ -2*c*dum/(length**4).
term1 _ super multiply: lengthArray by: (c1*lengthDotX).
term2 _ Array new: 3.
term2 at: 1 put: c2; at: 2 put: 0; at: 3 put: 0.
dummy1 _ dummy*(lengthArray at: 1).
term3 _ super multiply: lengthArray by: dummy1.
sub11 _ super add: (super add: term1 with: term2) with: term3.
term4 _ super multiply: lengthArray by: (c1*lengthDotY).
term5 _ Array new: 3.
term5 at: 1 put: 0; at: 2 put: c2; at: 3 put: 0.
dummy2 _ dummy*(lengthArray at: 2).
term6 _ super multiply: lengthArray by: dummy2.
sub12 _ super add: (super add: term4 with: term5) with: term6.
term7 _ super multiply: lengthArray by: (c1*lengthDotZ).
term8 _ Array new: 3.
term8 at: 1 put: 0; at: 2 put: 0; at: 3 put: c2.
dummy3 _ dummy*(lengthArray at: 3).
term9 _ super multiply: lengthArray by: dummy3.
sub13 _ super add: (super add: term7 with: term8) with: term9.
"---- For 2nd quadrant ----"
dum1 _ super multiply: lengthArrayt with: dLdPhi.
val1 _ -1*dum*dum1/(length**3).
val2 _ super multiply: lengthDotArrayt with: dLdPhi.
elem1 _ e1*phiDot + (e3*theDot) + (e5*psiDot).
elem2 _ e2*phiDot + (e4*theDot) + (e6*psiDot).
dLdotdPhi _ Array new: 3 withAll: 0.
dLdotdPhi at: 1 put: elem1; at: 2 put: elem2.
val3 _ super multiply: lengthArrayt with: dLdotdPhi.
val4 _ c*(val1 + val2 + val3)/length.
term1 _ super multiply: lengthArray by: val4.
c3 _ c2*dum1/(length**2).
term2 _ super multiply: lengthArray by: (-1*c3).
term3 _ super multiply: dLdPhi by: c2.
sub14 _ super add: (super add: term1 with: term2) with: term3.
dum2 _ super multiply: lengthArrayt with: dLdThe.
val5 _ -1*dum*dum2/(length**3).
val6 _ super multiply: lengthDotArrayt with: dLdThe.
elem1 _ e3*phiDot + (e7*theDot) + (e10*psiDot).
elem2 _ e4*phiDot + (e8*theDot) + (e11*psiDot).
elem3 _ e9*theDot + (e12*psiDot).
dLdotdThe _ Array new: 3 withAll: 0.
dLdotdThe at: 1 put: elem1; at: 2 put: elem2; at: 3 put: elem3.
val7 _ super multiply: lengthArrayt with: dLdotdThe.
val8 _ c*(val5 + val6 + val7)/length.
term1 _ super multiply: lengthArray by: val8.
c4 _ c2*dum2/(length**2).
term2 _ super multiply: lengthArray by: (-1*c4).
term3 _ super multiply: dLdThe by: c2.

```

```

sub15 _ super add: (super add: term1 with: term2) with: term3.
dum3 _ super multiply: lengthArrayt with: dLdPsi.
val9 _ -1*dum*dum3/(length**3).
val10 _ super multiply: lengthDotArrayt with: dLdPsi.
elem1 _ e5*phiDot + (e10*theDot) + (e13*psiDot).
elem2 _ e6*phiDot + (e11*theDot) + (e14*psiDot).
elem3 _ e12*theDot + (e15*psiDot).
dLdotdPsi _ Array new: 3 withAll: 0.
dLdotdPsi at: 1 put: elem1; at: 2 put: elem2; at: 3 put: elem3.
val11 _ super multiply: lengthArrayt with: dLdotdPsi.
val12 _ c*(val9 + val10 + val11)/length.
term1 _ super multiply: lengthArray by: val12.
c5 _ c2*dum3/(length**2).
term2 _ super multiply: lengthArray by: (-1*c5).
term3 _ super multiply: dLdPsi by: c2.
sub16 _ super add: (super add: term1 with: term2) with: term3.
"---- For 3rd quadrant ----"
dummy1 _ super multiply: lengthArray by: (-1*c1*lengthDotX).
dummy2 _ Array new: 3.
dummy2 at: 1 put: c2; at: 2 put: 0; at: 3 put: 0.
dummy3 _ super transpose: (super add: dummy1 with: dummy2).
term1 _ super multiply: dummy3 with: dLdPhi.
term2 _ super multiply: dummy3 with: dLdThe.
term3 _ super multiply: dummy3 with: dLdPsi.
sub21 _ Array new: 3.
sub21 at: 1 put: term1; at: 2 put: term2; at: 3 put: term3.
dummy4 _ super multiply: lengthArray by: (-1*c1*lengthDotY).
dummy5 _ Array new: 3.
dummy5 at: 1 put: 0; at: 2 put: c2; at: 3 put: 0.
dummy6 _ super transpose: (super add: dummy4 with: dummy5).
term4 _ super multiply: dummy6 with: dLdPhi.
term5 _ super multiply: dummy6 with: dLdThe.
term6 _ super multiply: dummy6 with: dLdPsi.
sub22 _ Array new: 3.
sub22 at: 1 put: term4; at: 2 put: term5; at: 3 put: term6.
dummy7 _ super multiply: lengthArray by: (-1*c1*lengthDotZ).
dummy8 _ Array new: 3.
dummy8 at: 1 put: 0; at: 2 put: 0; at: 3 put: c2.
dummy9 _ super transpose: (super add: dummy7 with: dummy8).
term7 _ super multiply: dummy9 with: dLdPhi.
term8 _ super multiply: dummy9 with: dLdThe.
term9 _ super multiply: dummy9 with: dLdPsi.
sub23 _ Array new: 3.
sub23 at: 1 put: term7; at: 2 put: term8; at: 3 put: term9.
"---- For 4th quadrant ----"
term1 _ (val4 - c3)*dum1.
term2 _ (val4 - c3)*dum2.
term3 _ (val4 - c3)*dum3.
term4 _ c2*(super multiply: (super transpose: dLdPhi) with: dLdPhi).
term5 _ c2*(super multiply: (super transpose: dLdPhi) with: dLdThe).
term6 _ c2*(super multiply: (super transpose: dLdPhi) with: dLdPsi).
array1 _ Array new: 3.
array1 at: 1 put: e1; at: 2 put: e2; at: 3 put: 0.
term7 _ c2*(super multiply: lengthArrayt with: array1).
array2 _ Array new: 3.
array2 at: 1 put: e3; at: 2 put: e4; at: 3 put: 0.
term8 _ c2*(super multiply: lengthArrayt with: array2).
array3 _ Array new: 3.
array3 at: 1 put: e5; at: 2 put: e6; at: 3 put: 0.
term9 _ c2*(super multiply: lengthArrayt with: array3).
sum1 _ term1 + term4 + term7.
sum2 _ term2 + term5 + term8.

```



```

sum3 _ term3 + term6 + term9.
sub24 _ Array new: 3.
sub24 at: 1 put: sum1; at: 2 put: sum2; at: 3 put: sum3.
term10 _ (val8 - c4)*dum1.
term11 _ (val8 - c4)*dum2.
term12 _ (val8 - c4)*dum3.
term13 _ c2*(super multiply: (super transpose: dLdThe) with: dLdPhi).
term14 _ c2*(super multiply: (super transpose: dLdThe) with: dLdThe).
term15 _ c2*(super multiply: (super transpose: dLdThe) with: dLdPsi).
array4 _ Array new: 3.
array4 at: 1 put: e7; at: 2 put: e8; at: 3 put: e9.
term16 _ c2*(super multiply: lengthArrayt with: array4).
array5 _ Array new: 3.
array5 at: 1 put: e10; at: 2 put: e11; at: 3 put: e12.
term17 _ c2*(super multiply: lengthArrayt with: array5).
sum4 _ term10 + term13 + term8.
sum5 _ term11 + term14 + term16.
sum6 _ term12 + term15 + term17.
sub25 _ Array new: 3.
sub25 at: 1 put: sum4; at: 2 put: sum5; at: 3 put: sum6.
term18 _ (val12 - c5)*dum1.
term19 _ (val12 - c5)*dum2.
term20 _ (val12 - c5)*dum3.
term21 _ c2*(super multiply: (super transpose: dLdPsi) with: dLdPhi).
term22 _ c2*(super multiply: (super transpose: dLdPsi) with: dLdThe).
term23 _ c2*(super multiply: (super transpose: dLdPsi) with: dLdPsi).
array6 _ Array new: 3.
array6 at: 1 put: e13; at: 2 put: e14; at: 3 put: e15.
term24 _ c2*(super multiply: lengthArrayt with: array6).
sum7 _ term18 + term21 + term9.
sum8 _ term19 + term22 + term17.
sum9 _ term20 + term23 + term24.
sub26 _ Array new: 3.
sub26 at: 1 put: sum7; at: 2 put: sum8; at: 3 put: sum9.
matrix _ (Array new: 6) collect: [: each | Array new: 6 withAll: 0].
super put: sub11 into: matrix at: (1@1). super put: sub12 into: matrix at: (2@1).
super put: sub13 into: matrix at: (3@1). super put: sub14 into: matrix at: (4@1).
super put: sub15 into: matrix at: (5@1). super put: sub16 into: matrix at: (6@1).
super put: sub21 into: matrix at: (1@4). super put: sub22 into: matrix at: (2@4).
super put: sub23 into: matrix at: (3@4). super put: sub24 into: matrix at: (4@4).
super put: sub25 into: matrix at: (5@4). super put: sub26 into: matrix at: (6@4).
^ matrix

```

sub42Damper

"returns the matrix [dL/dqdot] for a translational damping force. This matrix is to be put into the Jacobian submatrix sub42. The matrix has the size of 6 by 12. Note that {L(damper)} = {Q} - {F(damper)}."

```

| matrixI matrixJ partNoI partNoJ dumMatrix matrixIJ |
matrixI _ (Array new: 6) collect: [: each | Array new: 6 withAll: 0].
matrixJ _ (Array new: 6) collect: [: each | Array new: 6 withAll: 0].
partNoI _ ((self markerI) part) partNo. partNoJ _ ((self markerJ) part) partNo.
(partNoI ~= '00') ifTrue: [ matrixI _ self sub42DamperForMarker: (self markerI)].
(partNoJ ~= '00') ifTrue: [
    dumMatrix _ self sub42DamperForMarker: (self markerJ). matrixJ _ super multiply: dumMatrix by: (-1)].
matrixIJ _ (Array new: 6) collect: [: each | Array new: 12].
super put: matrixI into: matrixIJ at: (1@1). super put: matrixJ into: matrixIJ at: (7@1).
^ matrixIJ

```

sub42DamperForMarker: aMarker

"returns the contribution of a damping force on a marker to the matrix [dL/dqdot]."

```

| aPart matrix c lengthArray sum length lengthArrayt lengthX lengthY lengthZ c1 c3 c4 c5 x y z phi the psi sA sB
sC cA cB cC t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t18 t19 t20 t21 dLxdPhi dLxdThe dLxdPsi dLydPhi dLydThe

```

```

dLydPsi dLzdThe dLzdPsi dLdPhi dLdThe dLdPsi term1 term2 term3 sub11 sub12 sub13 sub14 sub15 sub16 sub21
sub22 sub23 sub24 sub25 sub26 dum1 dum2 dum3 |
c _ self dampingCoeff.
lengthArray _ (self markerI) relativePositionWith: (self markerJ).
sum _ 0.
lengthArray do: [: elem | sum _ sum + (elem*elem)].
lengthX _ lengthArray at: 1. lengthY _ lengthArray at: 2. lengthZ _ lengthArray at: 3.
length _ sum sqrt.
lengthArrayt _ super transpose: lengthArray.
c1 _ c/(length**2).
aPart _ aMarker part.
x _ (aMarker position) at: 1. y _ (aMarker position) at: 2. z _ (aMarker position) at: 3.
phi _ ((aPart centerOfMassMarker) position) at: 4. the _ ((aPart centerOfMassMarker) position) at: 5.
psi _ ((aPart centerOfMassMarker) position) at: 6.
sA _ phi sin. sB _ the sin. sC _ psi sin. cA _ phi cos. cB _ the cos. cC _ psi cos.
t1 _ -1*sA*sB.
t2 _ cA*sC + (sA*cB*cC).
t3 _ sA*cB*sC - (cA*cC).
t4 _ cA*sB.
t5 _ sA*sC - (cA*cB*cC).
t6 _ -1*(cA*cB*sC + (sA*cC)).
t7 _ cA*cB.
t8 _ cA*sB*cC.
t9 _ cA*sB*sC.
t10 _ sA*cB.
t11 _ sA*sB*cC.
t12 _ sA*sB*sC.
t18 _ -1*sB*cC.
t19 _ -1*sB*sC.
t20 _ cB*cC.
t21 _ -1*cB*sC.
dLxdPhi _ t4*z + (t5*y) + (t6*x).
dLydPhi _ -1*t1*z - (t2*y) - (t3*x).
dLdPhi _ Array new: 3.
dLdPhi at: 1 put: dLxdPhi; at: 2 put: dLydPhi; at: 3 put: 0.
dLxdThe _ t10*z + (t11*y) + (t12*x).
dLydThe _ -1*t7*z - (t8*y) - (t9*x).
dLzdThe _ -1*sB*z + (t20*y) - (t21*x).
dLdThe _ Array new: 3.
dLdThe at: 1 put: dLxdThe; at: 2 put: dLydThe; at: 3 put: dLzdThe.
dLxdPsi _ t3*y - (t2*x).
dLydPsi _ t6*y - (t5*x).
dLzdPsi _ t19*y - (t18*x).
dLdPsi _ Array new: 3.
dLdPsi at: 1 put: dLxdPsi; at: 2 put: dLydPsi; at: 3 put: dLzdPsi.
"----- For 1st quadrant -----"
sub11 _ super multiply: lengthArray by: (c1*lengthX).
sub12 _ super multiply: lengthArray by: (c1*lengthY).
sub13 _ super multiply: lengthArray by: (c1*lengthZ).
"----- For 2nd quadrant -----"
dum1 _ super multiply: lengthArrayt with: dLdPhi. c3 _ c*dum1/(length**2).
sub14 _ super multiply: lengthArray by: c3.
dum2 _ super multiply: lengthArrayt with: dLdThe. c4 _ c*dum2/(length**2).
sub15 _ super multiply: lengthArray by: c4.
dum3 _ super multiply: lengthArrayt with: dLdPsi. c5 _ c*dum3/(length**2).
sub16 _ super multiply: lengthArray by: c5.
"----- For 3rd quadrant -----"
term1 _ c1*lengthX*dum1. term2 _ c1*lengthX*dum2. term3 _ c1*lengthX*dum3.
sub21 _ Array new: 3. sub21 at: 1 put: term1; at: 2 put: term2; at: 3 put: term3.
term1 _ c1*lengthY*dum1. term2 _ c1*lengthY*dum2. term3 _ c1*lengthY*dum3.
sub22 _ Array new: 3. sub22 at: 1 put: term1; at: 2 put: term2; at: 3 put: term3.
term1 _ c1*lengthZ*dum1. term2 _ c1*lengthZ*dum2. term3 _ c1*lengthZ*dum3.

```

```

sub23 _ Array new: 3. sub23 at: 1 put: term1; at: 2 put: term2; at: 3 put: term3.
"----- For 4th quadrant -----"
term1 _ c1*dum1*dum1. term2 _ c1*dum1*dum2. term3 _ c1*dum1*dum3.
sub24 _ Array new: 3. sub24 at: 1 put: term1; at: 2 put: term2; at: 3 put: term3.
term1 _ c1*dum2*dum1. term2 _ c1*dum2*dum2. term3 _ c1*dum2*dum3.
sub25 _ Array new: 3. sub25 at: 1 put: term1; at: 2 put: term2; at: 3 put: term3.
term1 _ c1*dum3*dum1. term2 _ c1*dum3*dum2. term3 _ c1*dum3*dum3.
sub26 _ Array new: 3. sub26 at: 1 put: term1; at: 2 put: term2; at: 3 put: term3.
matrix _ (Array new: 6) collect: [: each | Array new: 6 withAll: 0].
super put: sub11 into: matrix at: (1@1). super put: sub12 into: matrix at: (2@1).
super put: sub13 into: matrix at: (3@1). super put: sub14 into: matrix at: (4@1).
super put: sub15 into: matrix at: (5@1). super put: sub16 into: matrix at: (6@1).
super put: sub21 into: matrix at: (1@4). super put: sub22 into: matrix at: (2@4).
super put: sub23 into: matrix at: (3@4). super put: sub24 into: matrix at: (4@4).
super put: sub25 into: matrix at: (5@4). super put: sub26 into: matrix at: (6@4).
^ matrix

```