Robotica (1997) volume 15, pp 49–62. © 1997 Cambridge University Press

MDSF – A generic development and simulation facility for flexible, complex robotic systems

Ou Ma, Kerman Buhariwala, Neil Roger, John MacLean & Robert Carr

Spar Space System, Spar Aerospace Limited, 9445 Airport Road, Brampton, Ontario (Canada) L6S 4J3
E-mail: oma@spar.ca, jmaclean@spar.ca, rcarr@spar.ca

SUMMARY
MDSF is a generic manipulator development and simulation facility developed by Spar Aerospace Limited, as a prime contractor to the Canadian Space Agency. It is an integral part of Canada's contribution to the International Space Station (ISS) program. MDSF is capable of simulating general robotic systems with complex and flexible structures undergoing all kinds of tasks including payload hand-over and contact with the environment. Rigorous validation and configuration control have been imposed throughout the development course of the simulation facility. Examples of current applications in the ISS program are included.

KEYWORDS: Flexible robotic systems; Simulation facility; ISS Program; MDSF; Dynamics.

1. INTRODUCTION

Spar Aerospace Limited is the prime-contractor to the Canadian Space Agency (CSA) in the development of the Mobile Servicing System (MSS) which will be used to support the construction, operation and maintenance[1] of the International Space Station. The robotic elements of the MSS, as shown in Figure 1, are the SSRMS (Space Station Remote Manipulator System) and the SPDM (Special Purpose Dexterous Manipulator). The SSRMS is a large (17.6 m long) and self-relocatable manipulator arm with seven active joints. The SPDM consists of a body, with one active joint on its waist, and a pair of about 3.5 m long arms, each with seven active joints, mounted on the upper part of the body. Each SPDM arm ends with a special end effector. The SSRMS will be used for the station assembly and large scale manipulation of payloads and astronauts. It will also be used to position and support the operation of the SPDM. In such cases, the SSRMS is the conduit of structural stiffness, signals, and power. Both ends of the SSRMS have an identical latching end-effector. The SSRMS is symmetric with respect to its 4th (elbow) joint; this allows it to be attached and operated from either end and self-relocate by alternating both ends as either base or free tip. The SSRMS operates either from the station's Mobile Transporter or attached to one of the station modules. The SPDM is used for small, precise manipulation and its main role is maintenance replacement of specific station elements. The SPDM operates mounted on (attached to the tip of) the SSRMS, either with both arms free or with one arm anchored to the space station for structural stabilization.

MDSF, an acronym for Manipulator Development and Simulation Facility, is an integrated facility capable of providing both non-real and real-time dynamics simulations of arbitrarily configured robotic manipulator systems or combinations thereof. Its primary objective is the life cycle support for the development, verification, and operation of the space station robotic systems. The MDSF capabilities were dictated by a variety of tasks assigned to the two MSS robotic systems, their operational configurations and their interaction. The interaction – payload handover – with yet another robotic system, the Shuttle Remote Manipulator System (SRMS, also known as the Canadarm), whenever the space shuttle is docked to the station, had to be accommodated as well. MDSF has to be able to dynamically simulate (in both real-time and non-real time):

(i) arbitrary manipulator motion;
(ii) berthing, de-berthing, capturing, and releasing payloads;
(iii) handling the same payload by more than one manipulator arm;
(iv) maintenance replacement of small station elements;
(v) assembly together of separately flown, large station elements;
(vi) self-relocating of a manipulator;
(vii) stand alone operation of a single manipulator;
(viii) coordinated operation of two (or more) stand-alone manipulators;
(ix) operation of one manipulator on the top (attached to the tip) of another;
(x) anchoring one arm into the surrounding environment for higher resulting operational stiffness at the root of the other (operational) arm; and
(xi) transitions between various operational configurations, topologies and payloads.

Some of the above listed operational scenarios are graphically illustrated in Figure 2. When the development of MDSF was initiated in 1986 – and perhaps even now – there were no commercially available simulators that could meet all the needs listed above. As a result, MDSF became an integral part of the MSS program. The development of MDSF was completed at the end of 1995 and it is now in its sutaining engineering phase. Because

https://doi.org/10.1017/S0263574797000076 Published online by Cambridge University Press

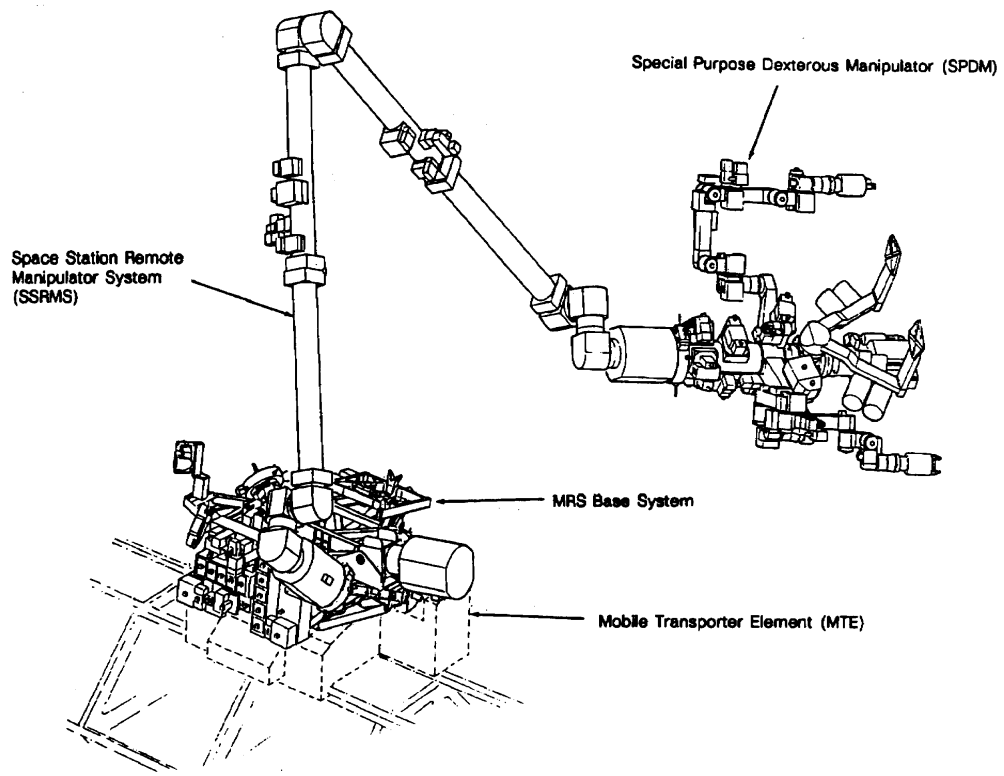


Fig. 1. Mobile Servicing System (MSS).

it was developed in increments, it first became operational in 1988 and it has been used in the MSS program ever since. Its current main uses in the ISS program are: acceptance verification of SSRMS, performance (envelop) study and characterization of SSRMS, design and development of SPDM. The MDSF-based SSRMS dynamics simulation model has been accepted by both CSA and NASA as the truth model for any SSRMS simulation model used in the ISS program. Because the MDSF-based SSRMS model includes some of the MSS flight software, the MDSF has been and continues to be used as an effective software test-bed.

The remarkable features of MDSF stem from the variety of tasks, robotic systems, operational configurations and interactions it had to accommodate. It has a versatile, robust and proven multibody dynamics kernel. Because it has to simulate space station assembly and maintenance operations, the MDSF has the capability of simulating the dynamics of intermittent contact regimes

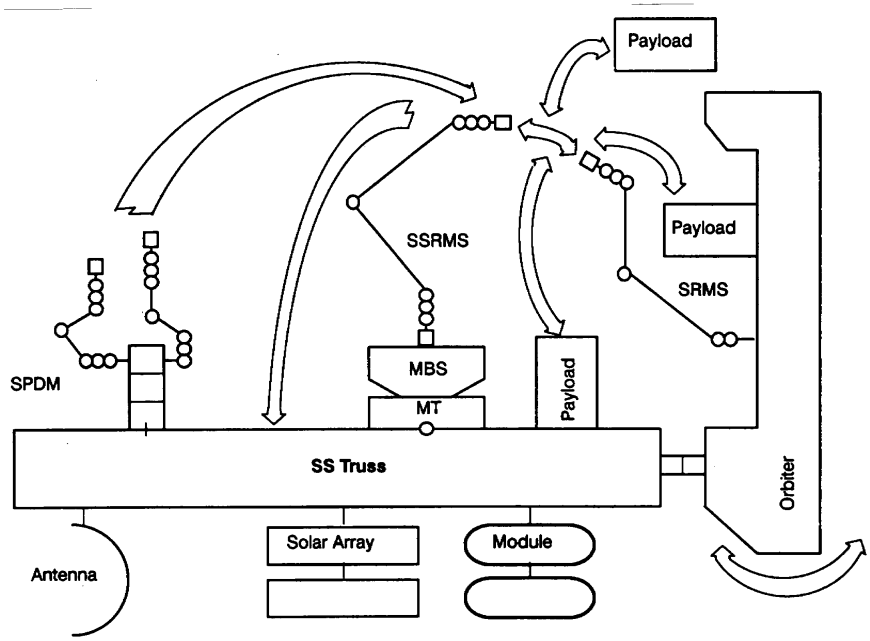


Fig. 2. Typical ISS activities performed by the SSRMS and SPDM.

https://doi.org/10.1017/S0263574797000076 Published online by Cambridge University Press

with payloads or surrounding environment. It has model order reduction at both body and system levels. It has a sophisticated numerical integration architecture. And it has a basic but sufficient robotic task definition language. The MDSF-based models incorporate a variety of (robotic) control systems at various hierarchical levels. Finally, the MSS simulation models include flight software items. MDSF's weak features stem from the fact that its development started a long time ago (relative to the fast evolving software practices) and re-writes of the software could not be cost-justified. As a result, the interface with the user is not graphical and the design/coding is not object based.

Because of its generic capabilities, MDSF can be readily used to model and simulate any space or terrestrial manipulator system. The associated techniques and software can also be used for applications other than the ISS program. As an example, SparNRTSIM, a direct derivative of the MDSF non-real time dynamics simulation capability is now being used to model and simulate the Light Duty Utility Arm (LDUA) and its derivatives. The LDUA is a large terrestrial manipulator, just passed all the operational tests, built by Spar Aerospace for the US Department of Energy. It will be used for environmental cleanup/restoration. A capability to interface SparNRTSIM with $Matrix_x$ (a commercial control system design and analysis package) has also been recently added. The LDUA developers are implementing the joint (level one) controllers and actuators in $Matrix_x$, importing in the actual arm (level two) controllers and using the SparNRTSIM multibody dynamics. The LDUA team will use the simulator to carry out quick performance studies of modified, larger, LDUA-like manipulators and for the development of a force-moment control feature. But it was mainly the contact dynamics capability that made SparNRTSIM the preferred simulation vehicle, because during operation, the LDUA comes in contact with either the ''gooey'' contaminated residue or with the solid walls within which the latter is contained.

2. FUNCTIONAL CAPABILITIES

This section describes the major functional capabilities of the MDSF.

2.1 Model definition

A *simulation model* of a dynamic system in MDSF is defined by specifying

a. geometry, inertia, and structure properties of the multibody system;
b. articulation/constraint properties of the system;
c. mapping of control systems to the multibody system; and
d. geometries of contact interfaces.

An interactive menu system called SYSDEF provides the capability to define the above listed items. Following the robotics convention, MDSF considers *bodies and articulations* (or joints) as elements of a manipulator system. A manipulator can have an arbitrary number of bodies connected by articulations. A body in MDSF can have arbitrary geometry, arbitrary flexible degrees of freedom (DOFs), and arbitrary mass-stiffness distribution. Several functionalities within SYSDEF are provided for the definition of flexible bodies. A body may be defined by directly entering its inertia, stiffness, and geometry data or by creating a simple beam. A complex flexible body may be built up from simpler flexible or rigid bodies by body-synthesis function. Additional springs representing flexibilities may be added in the synthesis process. A flexible body may also be defined by importing data from NASTRAN. Where appropriate, a model order reduction function based on given boundary conditions and modal frequency characteristics may be used to reduce the number of DOFs in a complex body. Each body has an inboard node and one or more outboard nodes as well as their associated coordinate frames arbitrarily defined by the user. All the data of a body is represented with respect to the inboard frame of the body. An articulation defines the kinematic connection between a node of one body and a node of another body. Each articulation may allow from zero to six DOFs of relative motion between the two involved nodes. An example of a 0-DOF articulation is a completely locked joint and that of 6-DOF articulation is the ''joint'' between a flying spacecraft and the ground. Coupled motions (constraints) within an articulation, such as ball/screw joint, are allowed. In articulations with combined translational and rotational DOFs, the rotational motion may be defined to occur about the node of either body. Geometries of contact interfaces may be defined graphically using 3-D visualization software. An MDSF utility program extracts the required information from the graphical definitions to create a contact geometry data file for use at run time. Currently Deneb's IGRIP package is being used for this function.

A multibody system is defined when all the separately defined bodies and articulations are assembled into a system topology table. The bodies of a system can be articulated into three basic topologies, namely, the *chain, tree,* and *closed-loop topologies* as well as a combination of them. A multibody system of the chain topology has an open end, such as the SSRMS; a system of tree topology has at least one body being articulated to two or more body chains with open ends, such as the SPDM when both of its arms are free; and a system of closed-loop tolology has at least one multibody chain without an open end, such as the SPDM when one of its arms grasp the other arm. MDSF also allows on-line change of topology, through the fourth type of topology-*transient topology*. The data of a defined system is output by SYSDEF to three system initialization files for use at build and run time.

An essential part of a robotic system is its associated control systems. SYSDEF defines the mapping between a multi-body system and its associated multi-level control systems. It allows up to five hierarchical levels of control systems. The lowest level corresponds to the control of a specific articulation (joint controller). Higher levels may

https://doi.org/10.1017/S0263574797000076 Published online by Cambridge University Press

correspond to coordinated control of a group of articulations, supervisory control of multiple manipulators, and so on. Each level of control will be mapped onto its immediate upper level controller. This provides a convenient structure within which simulation models of one or more automatic control systems can be integrated with the multi-body dynamics independently of the details or overall scope of the multi-body system. It also helps organize the multi-body system, which may represent a number of independent or coordinated manipulators, space craft, and payloads, into physically meaningful systems. Therefore, control system interface does not change with redefinition of the multi-body system. The mapping is also used by the task definition language, as described in Section 2.3.

The SYSDEF can be invoked in batch mode. It is convenient to use the batch mode for minor changes to an already defined large system. The batch mode may also be interfaced to Matlab to automate the end-to-end procedure from raw structural data to system initialization data, including intermediate mathematical manipulation of the data. SYSDEF also contains a full set of editorial functions such as ''validate'', ''copy'', ''rename'', ''compare'', ''delete'', ''print'', etc., at either body, articulation, or system level.

2.2 Build

The MDSF's capability to simulate a wide variety of multi-body systems is provided by its *generic run-time software* (so-called because it does not change from manipulator to manipulator) which calculates the dynamics response of the multi-body system and performs such functions as parameter initialization and data gathering. *Building* consists of interfacing the generic software with the control systems, compilation, and linking to create a simulation executable software ready for execution.

The MDSF provides a communication capability between the controls systems and the generic software through definition of a common data base consisting of global variables. Pre-compilers provide the capability for defining interface variables such that the common data base as a whole is hidden. The build procedures provide interfaces to standard tools such as source code analysers, performance profilers, and code management systems to help develop and maintain the simulation model.

Although MDSF contains some library files for controller elements, the control systems models may be completely system specific. They are usually written in *Fortran* but *C* or *Ada* may also be used. An arbitrary number of control systems, using multiple languages may be integrated. The MDSF optimizes executable size by using the system and contact array definition files at compile time to size arrays for global variables in the generic software. If the system definition is changed it is only necessary to rebuild the executable without going through a full-scale build if the complexity of the multi-body system or contact geometry has not increased.

2.3 Execution and analysis

The MDSF provides the capability to define initialization data, output variables, and a task definition at execution time.

The system and contact initialization data files generated during system definition are used to initialize parameters in the generic dynamics software. Additional initialization files may be added to set control system parameters and run-time dynamics and numerical solution options. Those options include, for example, integration method, step size, model order reduction criteria, orbital dynamics and so on. Any variable declared as part of the common data base can be gathered in a binary output file. Optionally, visualization data may be gathered also. The time interval for output data is arbitrary and may be changed on-line.

A task definition is created using the *TDL* (*Task Definition Language*). This is a script language used to set up manipulator initial motion conditions, provide commands to the controls systems, and reset common data base variables. A high level, manipulator independent command syntax provides commands like ''brakes off/on'', ''move_to'', ''move_at rate'', ''lock_articulation'', and so on. Strings may be passed to the controller models for system specific commands. TDL takes advantage of the controls system mapping created during system definition to maintain independence from the specifics of the multi-body dynamics system. The mapping is also used to address specific controllers when more than one manipulator is being simulated simultaneously.

MDSF simulations are normally executed in a nonreal-time mode on VAX workstations because non-real time simulation does not require high computing power and human-machine interface. An arbitrary number of nonreal-time simulators can be run simultaneously and may be spread across a network encompassing multiple clusters of workstations. The simulation model may be exercised in real-time as well, with or without a human-in-the-loop, by building it on the MDSF's IBM 591 computer. Silicon Graphics workstations are used to provide real-time visualization and control/command screens. A pair of three degree of freedom hand controllers is used for operator input.

The simulation output is converted to Matlab format for post-simulation viewing and analysis. Post-simulation graphics visualization data is rendered on Silicon Graphics workstations using IGRIP. While the simulation is executing, dynamics variables (such as matrices and vectors) may be exported to Matlab format for debugging and analysis. Output data can be gathered and analyzed while the simulation is still running. Simulation may be temporarily frozen for analysis.

3. DYNAMICS FORMULATION

All dynamics equations in this section are expressed in invariant form without involving particular coordinate frames. As a convention, matrices (i.e. 2-D tensors in this context), vectors (1-D tensors), and scalars are represented by boldfaced upper-case letters, boldfaced lower-case letters, and lower-case letters, respectively.

https://doi.org/10.1017/S0263574797000076 Published online by Cambridge University Press

3.1 System dynamics model

As described in Section 2.1, MDSF considers bodies and joints as elements of a manipulator system. A manipulator can have an arbitrary number of bodies (including the base and payloads) being arbitrarily articulated by a set of joints. The articulated multibody system may have open-chain, tree, closed-chain, or even transient topology. Each joint can have from zero to six fixed DOFs or time-varying DOFs (transient joint). Such a broad definition of topologies and joints enables MDSF to model and simulate virtually all kinds of multibody systems. This requires the dynamics algorithms implemented in MDSF to be correspondingly general enough. The Newton-Euler equations of motion were used to derive dynamics algorithms.

The overall motion of a robot system can be broken down into three major sub-motions, namely, the base motion, the rigid/joint motion, and the elastic motion. The three types of motions are governed by the following equations of motion:

$$\begin{aligned}
&\boldsymbol{M}_{bb}\ddot{\boldsymbol{q}}_b + \boldsymbol{M}_{br}\ddot{\boldsymbol{q}}_r + \boldsymbol{M}_{be}\ddot{\boldsymbol{q}}_e \\
&\qquad = \boldsymbol{f}_{b,ext} + \boldsymbol{f}_{b,non} + \boldsymbol{f}_{b,grav} + \boldsymbol{f}_{b,driv} + \boldsymbol{f}_{b,cd} \\
&\boldsymbol{M}_{br}^T\ddot{\boldsymbol{q}}_b + \boldsymbol{M}_{rr}\ddot{\boldsymbol{q}}_r + \boldsymbol{M}_{re}\ddot{\boldsymbol{q}}_e \\
&\qquad = \boldsymbol{f}_{r,ext} + \boldsymbol{f}_{r,non} + \boldsymbol{f}_{r,grav} + \boldsymbol{f}_{r,cons} + \boldsymbol{f}_{r,driv} + \boldsymbol{f}_{r,cd} \\
&\boldsymbol{M}_{be}^T\ddot{\boldsymbol{q}}_b + \boldsymbol{M}_{re}^T\ddot{\boldsymbol{q}}_r + \boldsymbol{M}_{ee}\ddot{\boldsymbol{q}}_e + \boldsymbol{K}_{ee}\boldsymbol{q}_e \\
&\qquad = \boldsymbol{f}_{e,ext} + \boldsymbol{f}_{e,non} + \boldsymbol{f}_{e,grav} + \boldsymbol{f}_{e,cons} + \boldsymbol{f}_{e,cd}
\end{aligned} \tag{1}$$

subject to constraints:

$$\boldsymbol{\Lambda}_r\dot{\boldsymbol{q}}_r + \boldsymbol{\Lambda}_e\dot{\boldsymbol{q}}_e = \boldsymbol{0} \tag{2}$$

It is assumed that the manipulator base has k DOFs and the manipulator itself has n DOFs; the whole system has m elastic DOFs and l independent closed loops. The symbols of the above equations are then defined below:

$\boldsymbol{q}_b$: k-D vector of generalized rigid coordinates of the manipulator base (i.e., orbiter).

$\boldsymbol{q}_r$: n-D vector of generalized rigid coordinates of the manipulator (joint coordinates).

$\boldsymbol{q}_e$: m-D vector of generalized elastic coordinates.

$\boldsymbol{f}_{b,ext}$: k-D vector of generalized external forces exerted on the manipulator base.

$\boldsymbol{f}_{b,non}$: k-D vector of nonlinear forces associated with the inertia of the base (i.e., the centrifugal and Coriolis forces). Here the term "nonlinear" refers to the nonlinearity in terms of velocities.

$\boldsymbol{f}_{b,grav}$: k-D vector of the generalized gravity/orbital forces associated with the base.

$\boldsymbol{f}_{b,driv}$: k-D vector of driving forces exerted on the manipulator base to drive the base.

$\boldsymbol{f}_{b,cd}$: k-D vector of generalized contract forces exerted on the manipulator base.

$\boldsymbol{f}_{r,ext}$: n-D vector of generalized external forces associated with joint coordinates.

$\boldsymbol{f}_{r,non}$: n-D vector of nonlinear forces associated with joint motion.

$\boldsymbol{f}_{r,driv}$: n-D vector of driving forces applied to actuated joints. These forces are the net forces output from the actuator dynamics model which includes control/actuator forces, joint flexibility, joint friction, brakes, gear train, and all other joint hardware effects.

$\boldsymbol{f}_{r,cd}$: n-D vector of generalized contact forces exerted on the manipulator.

$\boldsymbol{f}_{r,cons}$: n-D vector of generalized constraint forces associated with joint coordinates. These forces come into picture only when the manipulator has closed-loop topology.

$\boldsymbol{f}_{e,ext}$: m-D vector of generalized external forces associated with elastic coordinates.

$\boldsymbol{f}_{e,non}$: m-D vector of nonlinear forces associated with elastic coordinates. These forces are quadratic in terms of the speeds of elastic vibrations.

$\boldsymbol{f}_{e,cd}$: m-D vector of generalized contact forces associated with elastic coordinates.

$\boldsymbol{f}_{e,cons}$: m-D vector of generalized constraint forces associated with elastic coordinates. Like $\boldsymbol{f}_{r,cons}$, these forces exist only when the manipulator has closed-loop topology.

$\boldsymbol{M}_{bb}$: $k \times k$ generalized inertia matrix associated with the rigid motion of the base.

$\boldsymbol{M}_{rr}$: $n \times n$ generalized inertia matrix associated with joint motions. This is the generalized inertia matrix of the manipulator itself if it is a rigid-body manipulator.

$\boldsymbol{M}_{ee}$: $m \times m$ generalized inertia matrix associated with the elastic motion.

$\boldsymbol{M}_{be}$; $k \times m$ generalized inertia matrix coupling the orbiter motion and the elastic motion.

$\boldsymbol{M}_{br}$: $k \times n$ generalized inertia matrix coupling the orbiter motion and the joint motion.

$\boldsymbol{M}_{re}$: $n \times m$ generalized inertia matrix coupling the arm motion and the elastic motion.

$\boldsymbol{K}_{ee}$: $m \times m$ generalized stiffness matrix associated with elasticity.

$\boldsymbol{\Lambda}_r$: $6l \times n$ constraint matrix representing joint-motion constraints.

$\boldsymbol{\Lambda}_e$: $6l \times m$ constraint matrix representing elastic-motion constraints.

The dimensions indicated in the above list are for the most general case. Dimensions may be different for many special multibody systems such as a planar mechanism. The coupling of the three different types of motion can be clearly seen from the above equations. Although the coupling in accelerations is linear, the implicit coupling in velocities and positions (displacements) are nonlinear. This coupling and nonlinear nature, along with the essential requirement of real-time operation of manipulators, provides a challenge for the development of efficient simulation algorithms. The detailed dynamics formulation for MDSF can be found in references 2–4 and a series of Spar internal technical reports. In the above dynamics model, the rank (independency) of the constraint equations depends on the type of each closed kinematic loop. For a manipulator of open-chain type

https://doi.org/10.1017/S0263574797000076 Published online by Cambridge University Press

(i.e., chain or tree topology), the constraint equations simply vanish because all generalized coordinates are independent from each other. Joint flexibility is modelled in a separate motor/joint dynamics model and its results are passed into the system dynamics through the generalized driving forces $\boldsymbol{f}_{r,driv}$.

3.2 Elimination of constraint forces and equations

When a manipulator to be simulated has closed kinematic loops, its dynamics model, as presented in Section 3.1, contains generalized constraint forces and subjects to kinematic constraints. Also the generalized coordinates are not all independent from each other because of the presence of kinematic constraints. It is not necessary to compute the internal and nonworking constraint forces for dynamics simulation. From a control point of view, only the motions associated with the independent generalized coordinates are of interest and, from a simulation point of view, solving for both independent and dependent coordinates simultaneously will involve integration of the constraint equations, which has been proven to be numerically unstable and time consuming. Therefore, it is desirable to derive the dynamics equations in terms of independent variables without involving constraint equations and constraint forces. Summarized next is the method implemented in MDSF for this purpose.

First, let us re-write equation (1) by combining its last two vector equations:

$$\begin{aligned}&\boldsymbol{M}_{bb}\ddot{\boldsymbol{q}}_b + \boldsymbol{M}_{bm}\ddot{\boldsymbol{q}}_m = \boldsymbol{f}_{b,ext} + \boldsymbol{f}_{b,non} + \boldsymbol{f}_{b,grav} + \boldsymbol{f}_{b,driv} + \boldsymbol{f}_{b,cd}\\ &\boldsymbol{M}_{bm}^T\ddot{\boldsymbol{q}}_b + \boldsymbol{M}_{mm}\ddot{\boldsymbol{q}}_m + \boldsymbol{K}_{mm}\boldsymbol{q}_m\\ &\quad = \boldsymbol{f}_{m,ext} + \boldsymbol{f}_{m,non} + \boldsymbol{f}_{m,grav} + \boldsymbol{f}_{m,cons} + \boldsymbol{f}_{m,driv} + \boldsymbol{f}_{m,cd}\end{aligned} \tag{3}$$

subject to

$$\boldsymbol{\Lambda}\dot{\boldsymbol{q}}_m = \boldsymbol{0} \tag{4}$$

where

$$\begin{aligned}&\boldsymbol{M}_{mm} = \begin{bmatrix}\boldsymbol{M}_{rr} & \boldsymbol{M}_{re}\\ \boldsymbol{M}_{re}^T & \boldsymbol{M}_{ee}\end{bmatrix},\quad \boldsymbol{M}_{bm} = [\boldsymbol{M}_{br}\ \ \boldsymbol{M}_{be}],\quad \boldsymbol{q}_m = \begin{vmatrix}\boldsymbol{q}_r\\ \boldsymbol{q}_e\end{vmatrix}\\ &\boldsymbol{\Lambda} = [\boldsymbol{\Lambda}_r\quad \boldsymbol{\Lambda}_e],\qquad \boldsymbol{f}_{m,ext} = \begin{bmatrix}\boldsymbol{f}_{r,ext}\\ \boldsymbol{f}_{e,ext}\end{bmatrix},\quad \boldsymbol{f}_{m,driv} = \begin{bmatrix}\boldsymbol{f}_{r,driv}\\ \boldsymbol{0}\end{bmatrix}\end{aligned} \tag{5}$$

and other force iterms are similarly defined. Then, a full-rank matrix $\boldsymbol{\Pi}$, is found such that $\boldsymbol{\Lambda\Pi} = \boldsymbol{0}$. Obviously, $\boldsymbol{\Pi}$ is the orthogonal complement of the constraint matrix $\boldsymbol{\Lambda}$. In other words, $\boldsymbol{\Pi}$ defines the null space of $\boldsymbol{\Lambda}$. Physically, the columns of matrix $\boldsymbol{\Pi}$ form a basis of the feasible motion space of the manipulator, and hence, the projection of all constraint forces on this space must vanish. This fact can be easily understood because constraint forces never exist in the motion directions. Similarly, the rows of constraint matrix $\boldsymbol{\Lambda}$ define the constraint space (non-motion space) of the manipulator and hence, there should be no velocity components in this space. Based on this fact, it is apparent that the constraint forces appearing in the dynamics model can be eliminated by premultiplying both sides of the equations of motion by $\boldsymbol{\Pi}^T$. The next task is to eliminate constraint equations. To this end, a set of independent generalized coordinates, $\boldsymbol{q}_\pi$, is defined in the sense that $\dot{\boldsymbol{q}}_m = \boldsymbol{\Pi}\dot{\boldsymbol{q}}_\pi$. It can be readily shown that any $\boldsymbol{q}_m$ defined in such a way will always satisfy the constraint equation because, by definition, columns of matrix $\boldsymbol{\Pi}$ span the null space of matrix $\boldsymbol{\Lambda}$ and hence, such-defined $\boldsymbol{q}_m$ must lie in the null space of $\boldsymbol{\Lambda}$. In other words, for an arbitrarily given $\boldsymbol{q}_\pi$, the corresponding $\boldsymbol{q}_m$ must be inside the motion space and thus, one needs not worry about constraints any more if he uses $\boldsymbol{q}_\pi$ instead of $\boldsymbol{q}_m$ in dynamics modelling.

Substituting the above relation into the original dynamics equations, we obtain a new set of equations in terms of the independent generalized coordinates, namely,

$$\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{K}\boldsymbol{q} = \boldsymbol{f} \tag{6}$$

where

$$\begin{aligned}&\boldsymbol{q} = \begin{bmatrix}\boldsymbol{q}_b\\ \boldsymbol{q}_\pi\end{bmatrix}\\ &\boldsymbol{M} = \begin{bmatrix}\boldsymbol{M}_{bb} & \boldsymbol{M}_{bm}\boldsymbol{\Pi}\\ (\boldsymbol{M}_{bm}\boldsymbol{\Pi})^T & \boldsymbol{\Pi}^T\boldsymbol{M}_{mm}\boldsymbol{\Pi}\end{bmatrix}\\ &\boldsymbol{K} = \begin{bmatrix}\boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{\Pi}^T\boldsymbol{K}_{mm}\boldsymbol{\Pi}\end{bmatrix}\\ &\boldsymbol{f} = \begin{bmatrix}\boldsymbol{f}_{b,ext} + \boldsymbol{f}_{b,non} + \boldsymbol{f}_{b,grav} + \boldsymbol{f}_{b,driv} + \boldsymbol{f}_{b,cd}\\ \boldsymbol{\Pi}^T(\boldsymbol{f}_{m,ext} + \boldsymbol{f}_{m,non} + \boldsymbol{f}_{m,grav} + \boldsymbol{f}_{m,driv} + \boldsymbol{f}_{m,cd})\end{bmatrix}\end{aligned} \tag{7}$$

Matrix $\boldsymbol{\Pi}$ is calculated using a singular value decomposition method for achieving optimal numerical performance.

The above described method of eliminating constraints is conditional on the fact that the motion speed of the manipulator to be modelled is relatively small. If the dynamic system undergoes a fast speed manoeuvre, motion equations (3) have to be treated as perturbation equations (with respect to a known recent configuration). The rest of the method remains the same except that the operational configuration for the perturbation has to be updated periodically. Update period depends on motion speed and required accuracy. Once the perturbation reference configuration is updated, the position constraints need to be examined again. If any position constraint error is found, the generalized coordinates have to be adjusted in order to eliminate the constraint error.

3.3 Model order reduction

After eliminating the kinematic constraints, the resulting dynamics equations are independent and hence, can be readily integrated for simulation. However, due to the presence of high frequency flexible modes in the dynamics model, very small integration step sizes have to be used in order to keep the integration procedure stable. A small integration step size will lead to an inefficient simulation, which is undesirable, especially for real-time simulation. To overcome this problem, the following model order reduction technique is applied:

https://doi.org/10.1017/S0263574797000076 Published online by Cambridge University Press

Define a set of modal coordinates $\boldsymbol{\eta}$, such that

$$\boldsymbol{q} = \boldsymbol{E}\boldsymbol{\eta} \tag{8}$$

The columns of matrix $\boldsymbol{E}$ are generalized eigenvectors of the generalized inertia matrix $\boldsymbol{M}$ and stiffness matrix $\boldsymbol{K}$ normalized in the way that

$$\boldsymbol{E}^T\boldsymbol{M}\boldsymbol{E} = \mathbf{1} \quad \text{and} \quad \boldsymbol{E}^T\boldsymbol{K}\boldsymbol{E} = \boldsymbol{\Omega}^2 \tag{9}$$

In the above equations $\mathbf{1}$ denotes an identity matrix and $\boldsymbol{\Omega} = \mathrm{diag}(\omega_1, \omega_2, \omega_3, \ldots)$ *where* ω_i is the modal frequency of the ith flexible mode. In modal space, the dynamics equations reduce to

$$\ddot{\boldsymbol{\eta}} + \boldsymbol{\Omega}^2\boldsymbol{\eta} = \boldsymbol{E}^T\boldsymbol{f} \tag{10}$$

Appropriate boundary conditions have to be applied in model order reduction. In MDSF boundary conditions are known as different mode types. There are a total of six different mode types currently available for the user to select. A fundamental condition for the above described model order reduction is that the motion of the manipulator is slow so that the time derivatives of the eigenmatrix $\boldsymbol{E}$ are small. If this condition does not hold, extra nonlinear terms involving the time derivatives of matrix $\boldsymbol{E}$ must be added to the right hand side of equation (10).

In modal space the dynamics equations corresponding to different flexible modes have been decoupled. The next step is to select the flexible modes which have significant contributions to the motion of the system and neglect those without much contributions. Several criteria have been introduced in reference 5 for mode selection. Basically, these criteria fall into three categories, namely, (a) criteria based on modal frequencies; (b) criteria based on momentum coefficients; and (c) criteria based on modal costs. In general, the higher the frequency of a mode, the less contribution it makes to the manipulator motion. Therefore, removing the flexible modes with highest frequencies from numerical integration has been a common and practical approach. For the unselected modes, three options are available in MDSF. One option is to completely discard them; the second is to take the steady-state response of those modes; and the third one is to take a filtered dynamic response of those modes.

3.4 Contact dynamics modelling

Here the word "contact" includes both intermittent contact (impact) and continuous contact with or without sliding. By its purpose, a manipulator at work will be frequently required to contact something, such as when performing an assembly. To simulate a contact situation, contact mechanics has to be incorporated into the dynamics model. The modelling of contact dynamics in MDSF is based on an iterative geometry interference method which was originally developed at Spar[6]. The basic strategy of the method is: at each simulation step, a set of local contact forces between contacting bodies is computed in terms of the amount of geometry interferences and material stiffness between each pair of contacting bodies. The locally calculated contact forces are then applied to the system dynamics model as external forces and the immediate past integration step is repeated. After repeating the integration, the local contact forces are re-calculated and compared against their previous values. If the contact forces converge, the simulation takes one step forward, otherwise, the above iteration continues. The integration step size may be reduced on-line if the step size is found inappropriate for a large impact force. Such an approach avoids explicitly including the state variables of all contacting areas in the equations of motion which is, otherwise, very difficult, but it still guarantees the satisfaction of the computed contact forces to the equations of motion. An essential requirement of such a solution methodology to the simulation software is to have a feature of jumping back in time during numerical integration.

In the contact model, the total contact force-moment exerted on a contacting body is calculated by summing up all the local contact forces and moments exerted on this body, namely,

$$\boldsymbol{f}_c = \begin{bmatrix} \boldsymbol{\Sigma} - k_i d_i \boldsymbol{n}_i \\ \boldsymbol{\Sigma} k_i d_i \boldsymbol{r}_i \times \boldsymbol{n}_i \end{bmatrix} \tag{11}$$

where k_i, d_i, and $\boldsymbol{n}_i$ are, respectively, the contact stiffness, the depth of interference, and the unit surface normal vector of the ith contact point. $\boldsymbol{r}_i$ is the vector pointing to the ith contact point from the reference point used to represent the total contact force. Contact stiffness k_i is estimated based on Hertz contact theory,[7] which is a function of material stiffness of the two contacting bodies and the size of contacted surface area. Since each contact surface area depends nonlinearly on the depth of the geometric interference, the contact stiffness is obviously a nonlinear function of the depth of geometry interference and, therefore, the contact force is also nonlinear as well. MDSF also provides a linear contact force model which is very useful for supporting characteristics analyses of dynamics and control systems for many practical cases. The determination of a contact stiffness is mainly dependent on the material and structural properties of the contact interfaces but it is also subject to other constraints such as: very high stiffness may cause numerical instability during integration but too low stiffness may lead to large geometric interference which is against the basic assumption that the surface deformations between contact bodies must be much smaller than the sizes of the contacting bodies.

The contact status between every pair of potentially contactable bodies has to be frequently checked during a simulation. It is a rather challenging problem to efficiently determine the contact status of general 3-D objects undergoing arbitrary motions. Conventionally, this was done by computing the distances between the low level geometry primitives like vertices, edges, and surfaces of one body and those of another body. Such an approach suffers tremendous computational burden and algorithmic complications when the contacting geometry is even slightly complicated. MDSF applies an efficient contact detection algorithm[8] which works at the body

https://doi.org/10.1017/S0263574797000076 Published online by Cambridge University Press

level as opposed to the above-mentioned low primitive levels. The method finds the minimum distance between every pair of contacting bodies from the following constrained optimization problem:

$$\min \tfrac{1}{2}(\boldsymbol{p}_1^T - \boldsymbol{p}_2)^T \boldsymbol{W}(\boldsymbol{p}_1 - \boldsymbol{p}_2) \tag{12}$$

subject to

$$\boldsymbol{g}_1(\boldsymbol{p}_1) \le \boldsymbol{0} \quad \text{and} \quad \boldsymbol{g}_2(\boldsymbol{p}_2) \le \boldsymbol{0} \tag{13}$$

where $\boldsymbol{p}_1$ is the position vector of a point of the first contacting body and $\boldsymbol{p}_2$ is a point of the second body. $\boldsymbol{W}$ is a weighting matrix. The equalities of equations (13) are the bounding surface equations of the two contact bodies, respectively. A mathematical condition for this method is that both bodies have to be of convex shape. This, in fact, is not a problem, because in practice a concave body can always be approximated by a set of convex subbodies. Therefore, a concave body may be virtually cut into several convex subbodies at the stage of creating the geometry model of the body. Theoretically, this method works for any type of bounding surfaces. However, the required computational time closely depends on the complexity of the surfaces involved. For this reason, MDSF limits the surface type to either linear or quadratic. As a matter of fact, these two types of surfaces are sufficient for most practical applications.

When a contact between two bodies is detected, the next is to evaluate the amount of geometry interference at each contact area. This is done by first calculating the depth of interference and then computing the size of the contact area. The depth of interference is computed using the following ''negative distance'' algorithm:[9]

$$\min -d \tag{14}$$

subject to

$$\boldsymbol{g}_1(\boldsymbol{p}) \le -\frac{d}{2}\boldsymbol{e}_1 \quad \text{and} \quad \boldsymbol{g}_2(\boldsymbol{p}) \le -\frac{d}{2}\boldsymbol{e}_2 \tag{15}$$

where $\boldsymbol{p}$ is the position vector of a common point of the two contacting bodies; $\boldsymbol{e}_1 = [1, 1, \ldots]^T$ and $\boldsymbol{e}_2 = [1, 1, \ldots]^T$ whose dimensions are the same as the geometry constraints $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$, respectively. Variable d describes the depth of the geometry interference between the two contacting bodies at the contact point. This ''negative distance'' algorithm has similarities to the distance algorithm and, thus, its application conditions and its software implementation are similar to the former as well. The size of each contact area is estimated using the depth of interference and the known geometry of the associated contacting bodies.

Modelling friction, especially the static friction (stiction), between two arbitrarily contacting 3-D objects is another challenging problem. Conventional Coulomb friction model not only fails to produce stiction or jamming situations but sometimes also causes unstable dynamics simulation. Haessig and Friedland proposed a novel bristle stiction model in reference 10 but their model was limited to a single dimensional case only. We extended the idea of bristle friction to the general 3-D case, as described next. For each contact point, the friction force, $\boldsymbol{f}_f$, is calculated as

$$\boldsymbol{f}_f = -k_f \boldsymbol{s} \tag{16}$$

where k_f is the contact bristle stiffness and $\boldsymbol{s}$ is the vector of the bristle displacement,

$$\boldsymbol{s}(t) = \begin{cases} \boldsymbol{s}(t_0) + \displaystyle\int_{t_0}^{t} \boldsymbol{v}_t \, dt, & \text{if } |\boldsymbol{s}| < s_{\max} \\ s_{\max} \dfrac{\boldsymbol{v}_t}{|\boldsymbol{v}_t|}, & \text{otherwise} \end{cases} \tag{17}$$

where t and t_0 are, respectively, the current time and the start time of the last sticking at the contact point; $\boldsymbol{v}_t$ is the relative tangential velocity between the two contact bodies at the contact point; and $s_{\max}$ is the maximum allowed deflection of the bristle, namely,

$$s_{\max} = \frac{\mu |\boldsymbol{f}_n|}{k_f} \tag{18}$$

where μ is the friction coefficient and $\boldsymbol{f}_n$ is the vector of normal contact force. Note that $s_{\max}$ is time-varying and t_0 is different for each individual contact point and it has to be reset for each new impact or each new sticking. Besides the contact friction force, contact damping force is also modelled in MDSF to take into account of energy dissipation phenomenon arising from contacting.

4. SOFTWARE

4.1 Integration architecture

The integration architecture has been designed to support a number of features, as described next. The basic ''building block'' within the non-realtime integration architecture is the SMM, or Simulation Modelling Module. This is a subroutine which must conform to certain coding standards. One generic SMM is provided, containing the general dynamcis and kinematics equations. All other SMMs are used-provided, and represent the user's control system (perhaps one for joint-level control, another for arm-level control, and so on) or environmental forces (such as hydrodynamic forces for underwater manipulators). The main purpose of the SMMs is to compute the derivatives of the associated state variables. Discrete equations may also be included in the SMMs. The SMMs may be coupled together into one ''equation set'', and integrated together by the selected integration method. Alternatively they may be split up into one or more equation sets. The equation sets may then be integrated with different integration methods, or even different step sizes. The user's code does not need to change when using these features, it is all handled through data files at runtime.

Each equation set has its own time variable. Since only one equation set can be integrated at a time, the equation sets will in general be slightly out of step with each other. It therefore has to be assumed that the coupling between the equation sets is sufficiently weak so

https://doi.org/10.1017/S0263574797000076 Published online by Cambridge University Press

that accurate results can still be obtained. Strongly coupled equations should be put in the same equation set, of course. The equation sets will be periodically synchronised, so that output may be generated with a consistent time stamp. When and how the equation sets are integrated and synchronised is under user control through runtime data files.

Typically, the equations representing the output side of the gearboxes on a robot arm will need to be coupled to the system dynamics and integrated with a small step size. The remaining control equations may be put into other equation sets and perhaps integrated with a larger step size. To check that this is acceptable, the user would first put all the equations into a single equation set and integrate with a small step size. Then he would compare this run with other runs using different step sizes and equation sets until an acceptable tradeoff between accuracy and speed of execution was obtained.

The integration step size may be changed during a run, by a request from any SMM. Additionally, any SMM may request that the current integration step be repeated, perhaps with a smaller step size. In this case, the equation set containing the requesting SMM will jump back in time, reset the step size and go again. A priority system based on smallest step size is used to handle simultaneous requests from different SMMs. The capability to change the step size and jump back in time is utilized in simulating friction and contact dynamics.

A library of integration methods is available, consisting of various first and second order, single step, mulsti-step and predictor-corrector methods. Additional methods can be easily added. The user selects which method to use for each equation set via his runtime data files. The SMM code does not have to change in any way when a different integration method is selected.

4.2 Equation set synchronisation

An example using three equation sets is shown in Figure 3. Equation sets A, B and C have nominal step sizes δ_a, δ_b and δ_c respectively (set by the user), where:

$$0 < \delta_a \leq \delta_b \leq \delta_c \tag{19}$$

The actual step sizes used are free to vary at any time, and the equation set may jump back and repeat each step a number of times, due to requests from the SMMs. Let

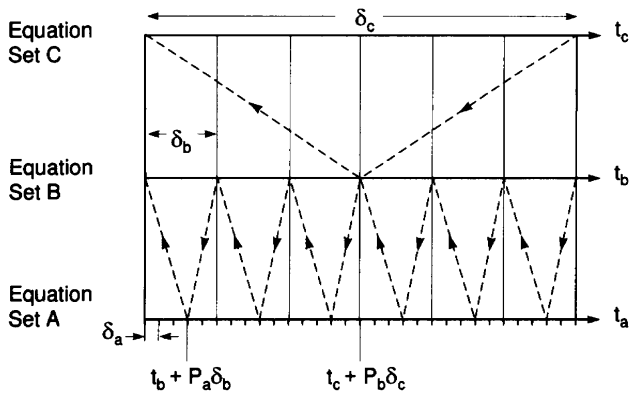


Fig. 3. Equation set synchronisation diagram.

the time variables for the three equation sets A, B and C be t_a, t_b and t_c respectively. The synchronisation between the equation sets is determined by the parameters p_a and p_b, where:

$$0 \leq p_a \leq 1 \quad \text{and} \quad 0 \leq p_b \leq 1 \tag{20}$$

Then equation set B will move ahead by one step whenever the condition:

$$t_a \geq t_b + p_a\delta_b \tag{21}$$

Thus equation set A integrates forward using its step size δ_a (or less), taking data from equation sets B and C, which are stationary at times t_b and t_c respectively, until it reaches some point $t_b + p_a\delta_b$ midway between equation set B's current time and its next time. At this point, equation set B integrates forward, and the process repeats, with equation set A taking data from equation set B at its new time.

A similar process occurs between equation sets B and C, with C integrating forward when the following condition is satisfied:

$$t_b \geq t_c + p_b\delta_c \tag{22}$$

Any number of equation sets can be accommodated in this manner, using any desired step sizes. The parameters p_a, p_b, etc. may be set by the user, but a good value is 0.5, which is a compromise between using advanced data and retarded data from the other equation sets.

The nominal step sizes δ_a, δ_b and δ_c need not be integral multiples of each other, and indeed an SMM may request a smaller step to be used at any time. However, the equation set times t_a, t_b and t_c must be periodically synchronized (for example when generating output). To do this, the integration architecture will automatically divide up the remaining time into steps less than or equal to the nominal step size (a number of ways to do this are available to the user), so that periodic synchronisation is achieved.

4.3 Source code preprocessor

The simulation software is written in *Fortran 77*. However, a preprocessor is used to manage the Fortran common blocks. Directives to the preprocessor are either put in separate files, or in the Fortran source files. These directives will appear as comment statements to the Fortran compiler. The preprocessor provides the following functions:

(a) Generates the *Fortran* common blocks based on information in the preprocessor directives from the source files.

(b) Provides "information hiding" by giving routines access only to those labels in the common blocks that they actually need. This feature also has the effect of dramatically reducing the size of the executable file, compared to what it would be if each routine had access to every label.

(c) Create the connection between the ASCII string

https://doi.org/10.1017/S0263574797000076 Published online by Cambridge University Press

representing a label's name, and the location in the common block. This information is used for simulation I/O.

The Fortran compiler usually makes the decision as to where to put the labels within the common blocks. Usually, some control is provided to the user: packed, alignment on even byte addresses, alignment on 4-byte addresses, natural alignment, and so on. Since the preprocessor must be portable between many different platforms, it cannot just assume that it knows where the compiler is going to put them, or what compiler option the user will select.

Instead, the preprocessor makes the decision as to where all the labels will be located within the common blocks. This is achieved by having the preprocessor declare the common blocks as a single large byte array, and then equivalence the labels to the appropriate locations. The alignment methods used in the Microsoft, Vax and Alpha compilers have been duplicated in the preprocessor, for selection by the user. In addition, the labels may also be sorted by the preprocessor so that they are simultaneously packed to save space and aligned for maximum access speed.

The preprocessor can then provide information on each label's name, address, type, number of bytes, etc. for use at runtime.

4.4 Simulation runtime I/O

MDSF has three major I/O data flows. They are the initial input data, the time-dependent input data, and the simulation output data, as described below.

The simulation initial data consists of:

(a) data defining the physical system to be simulated, such as system topology, inertia and stiffness properties of bodies, kinematic data of links, joint articulation information, etc.;
(b) integration architecture data such as step sizes, equation sets, simulation start and end times, numerical solution control parameters, etc.;
(c) dynamics algorithm data such as model order reduction information, flags to include nonlinear terms, periods to use for updating the mass matrix, normalising the quaternions, orbital data;
(d) contact geometry and material data;
(e) any data needed for the user's control system or environmental forces.

All of this initialisation data will appear in ASCII data files in the general form "variable = value". The simulation input routines parse statements of this form and use information provided by the preprocessor to recognize the variables and put the value specified into the appropriate location within the common block. The value specified on the right could be an integer, real, logical or string value. The variable on the left could be a scalar or array element, of the same type. Additional flexibility is provided by using the Bauer-Samelson algorithm to handle expressions for the value on the right, and for array indices on the left. Common block labels may be used in the expressions, and temporary variables may also be declared and used on the fly. The resulting syntax is very similar to Fortran or C assignment statements.

Time-dependent data consists of ASCII statements written in the *MDSF TDL* language (*Task Definition Language*), intended to be acted upon by the arm or joint-level controllers. These statements are parsed separately and read in at the appropriate times during the run. For example, the joint angles could be commanded to move at a particular rate or to a particular position in joint-space, or the tip of the arm could be commanded to move through a series of points in 6-DOF Cartesian space (x, y, z, roll, pitch, yaw), and so on. These commands will only be executed when the simulation reaches the time specified in the TDL file.

The main output consists of a binary file containing the time-history of various simulation variables and array elements. Exactly which variables and array elements are to be output is specified as a list of label names in an ASCII file. Information from the preprocessor is used to recognize the label names and locate them in the common blocks. The values may be output periodically or non-periodically during a run. The binary files may be analyzed or plotted during or after the run, via MATLAB, or used for visualisation/animation via IGRIP. Additional ASCII output files are generated during the run, containing information/error messages, a summary of the run conditions, and so on.

4.5 Interfaces to Non-MDSF software

To facilitate the pre-simulation modelling and post-simulation analysis, MDSF provides interfaces for the user to communicate with the following well-known software tools.

(a) **NASTRAN.** Nastran may be used to generate the physical system properties, such as masses, inertias, stiffnesses, and so on. An interface is provided to convert the NASTRAN data into MDSF format.

(b) **MATLAB.** MATLAB is used for post-processing the simulation results. Plots of one variable against another may be generated, or various types of analysis may be performed.

(c) **IGRIP.** IGRIP is used for visualisation/animation of the non real-time simulation results, after the run has been performed.

(d) **Matrix$_x$.** The simulation kernel has been ported to Matrix$_x$, to run as a User Code Block. This allows the user to provide the control system via a Matrix$_x$ block diagram rather than having to write *Fortran* code, and provides immediate access to the control design and analysis tools of Matrix$_x$.

5. VALIDATION AND QUALITY/CONFIGURATION CONTROL

MDSF software development followed a rigorous software development life-cycle which was monitored under the auspices of the product and quality assurance procedure. Since the facility was developed for the design and verification of space flight items of the

https://doi.org/10.1017/S0263574797000076 Published online by Cambridge University Press

International Space Station Program all functional capabilities in the official software have undergone rigorous and extensive validation. The capabilities were introduced incrementally into the software over several versions of MDSF to mitigate risk to the overall program. Incremental introduction of capabilities meant enhancements and modifications to formally validated software routines. Thus any modification to existing routines, no matter how insignificant, had to involve retesting and revalidating preexisting capabilities to ensure that the software had not regressed. The software has been fully documented.

The most difficult task of validating all of MDSF's rich and complex multibody dynamics and control capabilities was to find validated reference results or validated simulators to compare against. It is evident that there has been no single reference or simulator available which is able to validate all the MDSF capabilities listed in Sections 1 and 2. As a result, validated or well recognized reference results and simulators had to be used, as shown in Figure 4. The dynamics of chain, tree and closed-loop topologies, including nonlinear inertia forces and orbital dynamics were validated against TREETOPS which is a public domain multibody dynamics pakage developed for NASA. The constraint forces of systems with closed-loop topology and structural properties of flexible multibody systems were compared against corresponding NASTRAN analysis results. The SRMS model implemented on MDSF has been compared against ASAD which is a NASA certified SRMS simulator developed by Spar in the 1980's. The validity of ASAD has been established by comparing its simulation results with real SRMS flight data. In addition, the SRMS model in MDSF was also compared directly to SRMS flight data, because the structural and stiction friction models along with the solution employed in MDSF to solve the dynamics equations are different from that implemented in ASAD. MDSF simulation results have also been compared to various engineering test results as well as many independent engineering analysis results.

Validation of MDSF's contact dynamics model was carried out in several steps. An early validation exercise was carried out by an outside group specialized in space dynamics and control. In that validation work the MDSF simulation of a simple flexible manipulator striking a wall was successfully compared against an analytical solution. The second validation effort was made by MDSF developers. This one compared the full-order contact dynamics simulation against a reduced-order simulation using a prototype of the SPDM berthing a payload onto the space station truss. This validation demonstrated that model order reduction techniques implemented in MDSF work well with the contact dynamics model. The latest validation was preformed by the MSS simulations group. In that validation exercise, Whitney's analytical and experimental peg-in-hole data, as presented in reference 11, were reproduced by MDSF. As an example, the insertion forces generated by MDSF simulation and those from Whiteny's paper are superimposed in Figure 5 for comparison. Small oscillatory responses can be noticed in the MDSF results because of the dynamic simulation (as opposed to just static analysis). This validation has demonstrated that the MDSF contact dynamics model, as applied to peg-in-hole situation, can predict the jamming situation at the right time, in the right place, and with the right insertion and reaction forces and moments.

Further validation efforts, especially validation against hardware tests and flight data will be made in future.

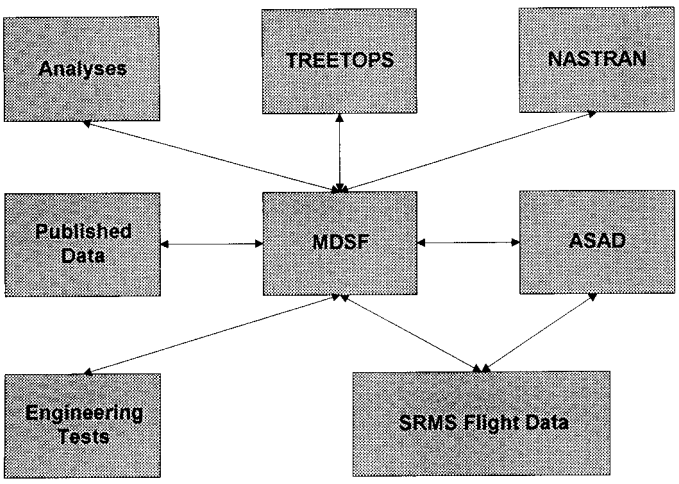


Fig. 4. MDSF validation flow.

6. APPLICATION EXAMPLES

In this section, two recent applications of the MDSF are described.

6.1 SSRMS berthing A payload

When the SSRMS is first installed on the International Space Station it will undergo a series of checkout manoeuvres. One of these manoeuvres is to test the ability of the SSRMS to berth a payload to the space station with its force-moment accommodation feature active. The payload involved in this test is called SLP (shown in Fig. 6), which has a mass of about 960 Kg and an approximate size of $3.3\,\text{m} \times 3.5\,\text{m} \times 4.6\,\text{m}$. It has a "U" cross-section structure used for packing the folded SSRMS inside the shuttle's cargo bay during launch. A pre-flight analysis of this manoeuvre was performed using MDSF for CSA to determine the SSRMS performance and dynamic loads.

The SSRMS dynamics model was built based on the design data which consists of a chain of elastic bodies with seven active joints and a total of 44 DOFs.[12] Each of the bodies were built by synthesizing together simple beams with springs. The payload was modelled as a flexible body based on the structure data provided by NASA. Joint hardware such as motors, brakes, and (nonlinear) flexible gear-trains with friction were separately modelled as a second order dynamic system at each joint. The manipulator's joint control software (in *Ada* and 80186 assembly code) was re-coded in MDSF using *Fortran* language. However, its arm control software (in Ada language), which is used to control the

https://doi.org/10.1017/S0263574797000076 Published online by Cambridge University Press

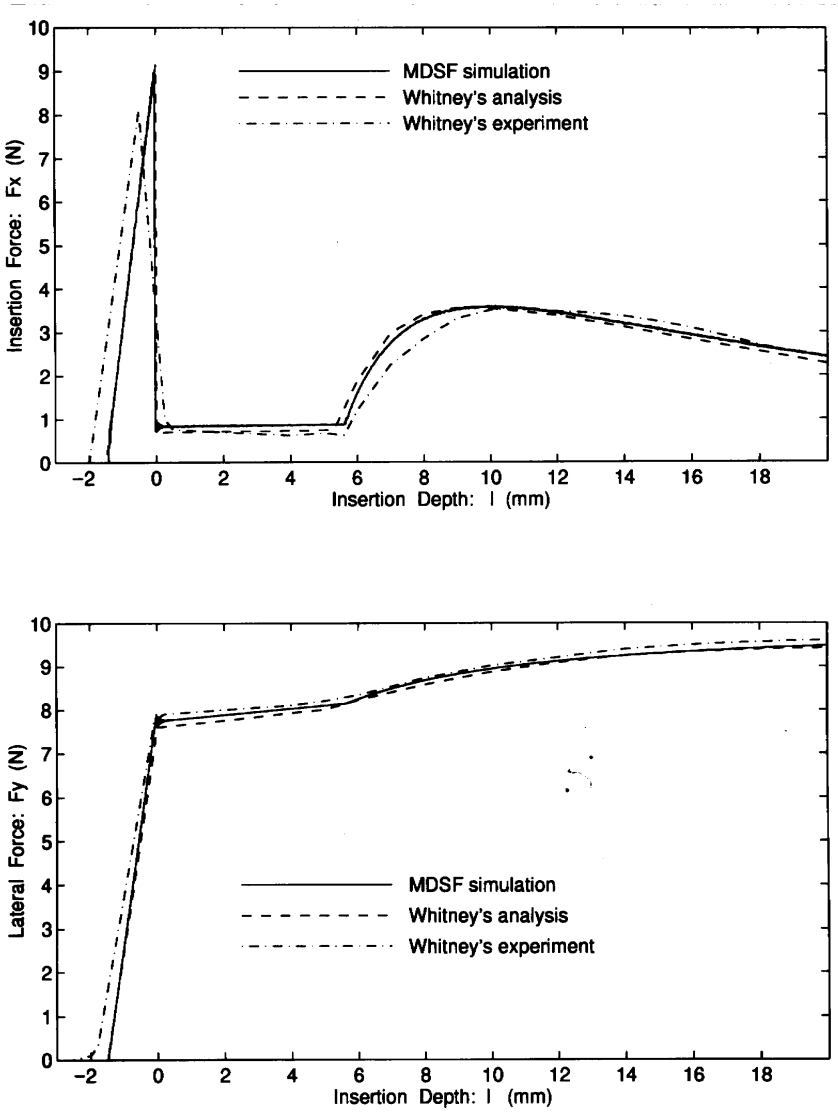


Fig. 5. Comparison between MDSF and Whitney's results.

coordinated joint motions, was directly used in MDSF as is. Actuators that require modelling were joint and motor resolvers at each joint, and the force-moment sensor on the end effector of the arm. In the simulation the system dynamics equations were integrated at a step size of 1.0 milli-second and the joint hardware models were integrated at a step size of 0.1 milli-second. Iterative integration was used during contact periods for producing accurate contact responses. Model order reduction was used, which retained all the dynamic responses of the modes equal or below 50 Hz and retained only the low frequency dynamic responses of the modes above 50 Hz.

The contact geometry of the berthing interface, as shown in Figure 7(a), consists of four trunnion/V-guide

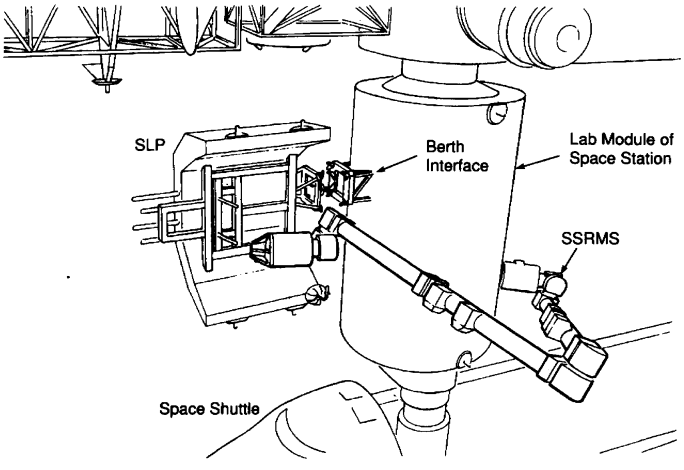


Fig. 6. SSRMS berthing SLP to the space station.

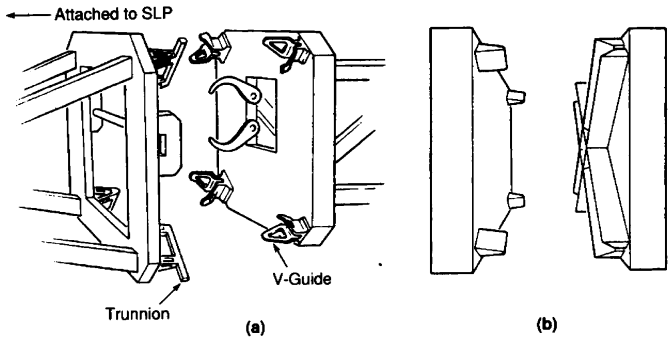


Fig. 7. SLP berthing interface (a) and its MDSF model (b).

https://doi.org/10.1017/S0263574797000076 Published online by Cambridge University Press

pairs. Because simulation execution time is proportional to the number of subbodies and surfaces in the contact geometry model,[6,8] every two opposite v-guides along the diagonal line of the berthing interface were modelled as a single long v-guide, as shown in Figure 7(b). While this does not affect the dynamics response of the model, by reducing the complexity it allows the model to be simulated in real-time. The contact stiffness is chosen in accordance with the discussion in Section 3.4.

The operation starts with the SSRMS holding the payload completely berthed to the interface installed in the lab module of the space station. The arm is first commanded to pull the payload away from the interface by approximately 36 inches. The arm is then commanded to move forward and push the payload towards the interface at a vernier speed with its force-moment controller active. Once all the trunnions have bottomed out in the corresponding v-guides an active mechanism provides fine alignment and rigidization of the interface. Simulation results indicated that the berthing task can be accomplished in accordance with the specification requirements. A full set of useful information has been generated from the simulation and provided to the system, design and operation teams for their uses.

6.2 SRMS to SSRMS payload handover

During assembly of the space station, sections of the station are delivered by the space shuttles to the station site and then handed over by the Shuttle Remote Manipulator System (SRMS) to the SSRMS for installation on the station. An example of this type of operations is the handling of the S3S4 section of the station, which is a large structure weighing approximately 15,000 Kg and sizing about 3.4 m × 4.8 m × 14.0 m, to be used for supporting some of the photovoltaic arrays at the starboard end of the station truss. During the handover, the SRMS is braked while the SSRMS end-effector approaches and then attaches to a grapple fixture mounted on the S3S4 (i.e., the payload). While attaching, the SSRMS is limped by opening the windings of the joint motors, and then the mechanism of end-effector pulls the arm onto the payload. The handover configuration of the two manipulators is shown in Figure 8. After the SRMS has handed over the S3S4 to the SSRMS, the latter carries the payload to its installation site and berths it onto its permanent position on the station (shown in dashed lines in Figure 8). An analysis was performed at Spar for CSA, using MDSF to determine dynamic loads encountered along the SSRMS and SRMS structures during this operation.

The dynamics model of the SRMS was created based on its hardware. This model has been validated against both ASAD and flight data. A simulation model was created, which includes the space station, the space shuttle, the S3S4, the SRMS, and the SSRMS. The SSRMS-to-payload attachment interface consists of an end effector and a grapple fixture. The handover operation starts when the three snaring cables inside the end effector cylinder catch the grapple fixture probe. After the capture of the probe, the handover procedure was approximated by splitting it into the following four phases:

a) the radial misalignment between the end effector and the grapple fixture is removed by tightening the snaring cables;
b) the grapple fixture is then pulled by the cables towards the end effector. Yaw and pitch misalignments are removed by the pivoting action of grapple fixture cams against the front rim of the end effector;
c) after the backplate of the grapple fixture makes contact with the front rim of the end effector, the yaw and pitch misalignments are corrected; and finally
d) the roll misalignment is removed by the reaction of the cams with the rim geometry and the connection between the S3S4 and the SSRMS end effector is rigidized and the connection between the S3S4 and the SRMS end effector is released.

Each phase of the attachment procedure was simulated by a pulling model except the pivoting action of the cams for yaw and pitch alignments. The latter was modelled using contact dynamics. Figure 9 (solid lines) shows the modelled contact geometries of the grapple fixture and the end effector. Contact between the grapple fixture probe and the inner surface of the end effector is not an issue in this study and hence, the probe was not included in the contact model in order to have high simulation

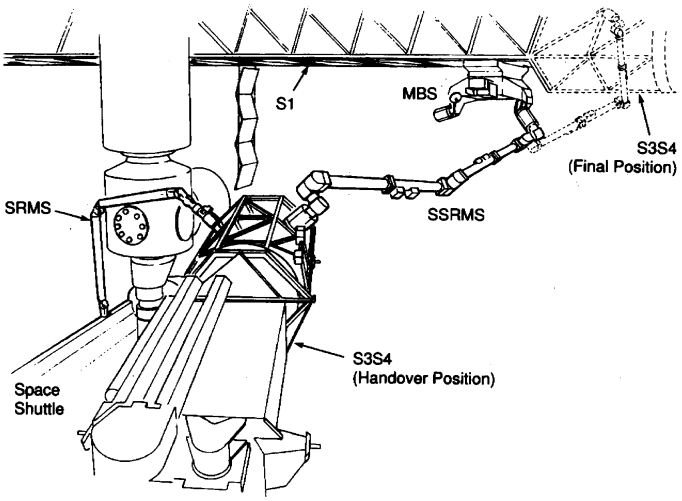


Fig. 8. Arm configurations for S3S4 handover and berthing.

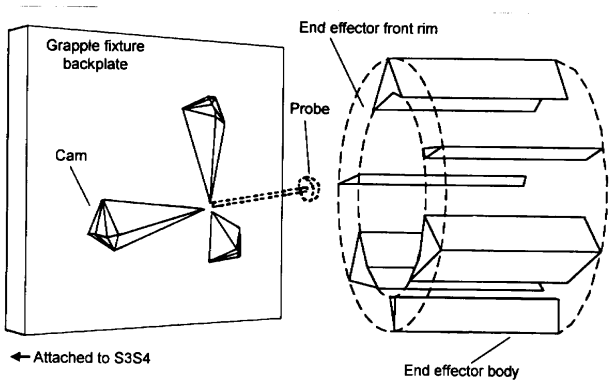


Fig. 9. End-effector/grapple-fixture attachment interface.

https://doi.org/10.1017/S0263574797000076 Published online by Cambridge University Press

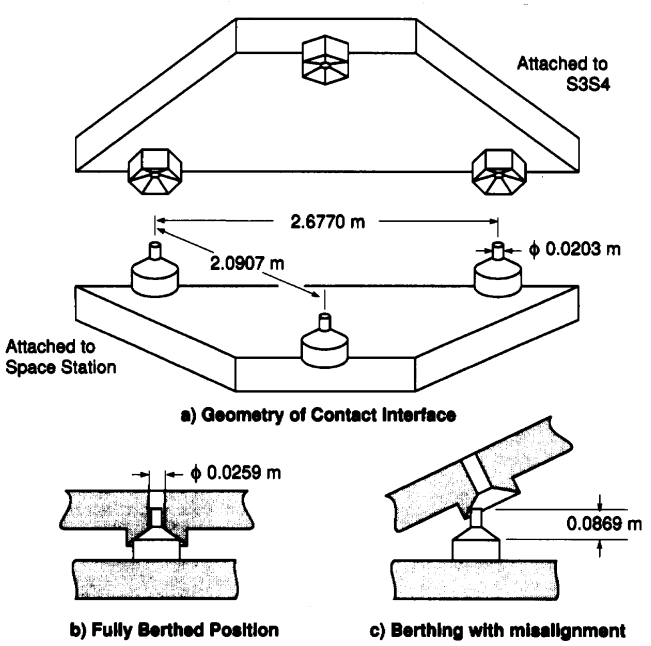


Fig. 10. S3S4 berthing interface model on MDSF.

efficiency. For the end effector, six sections (30° or 15° each) which may contact the grapple cams are modelled. In this way the circular geometry of the end effector can be better approximated by a limited number of planar sections. Specifications on initial misalignment between the grapple fixture and the end effector for successful capture put a limit of only 10 degrees about the roll axis which is much smaller than the modelled geometry, so that any contact outside of this 10° region is not of interest.

After the handover, the S3S4 is plugged onto the space station by the SSRMS. The berthing interface consists of three pairs of cones and cylinders, as shown in Figure 10. Because the female part of each cone has a concave shape, it has to be cut into subbodies of convex shape as required by the contact dynamics methodology. This is the reason why it is modelled as a hexagon instead. For this particular application the inner part of the hexagon, which consists of six linear surfaces, is sufficient for approximating the female conic surface.

7. CONCLUSIONS

A generic manipulator development and simulation system has been developed by Spar Aerospace Limited as a prime contractor to the Canadian Space Agency. It is an integral part of the mobile servicing system, Canada's contribution to the International Space Station Program. With the development of MDSF, several advanced dynamics and simulation techniques have been developed, implemented, and validated. Among others, the contact dynamics model, the 3-D contact friction/stiction model, the model order reduction, the solution of kinematic constraints for flexible closed-loop topology, the parallel integration of equation sets, and the jump-back in integration time are the key techniques which have made MDSF uniquely capable of simulating general robotic systems with complex and flexible structures undergoing all kinds of tasks including payloads hand-over and contact with the environment. Rigorous validation and quality control procedures have been imposed throughout the development course. The simulator is being used to support the development, verification, and operation-planning of the space station manipulators. As a technology spin-off, the simulator or its associated techniques are also being used in other space or non-space projects.

Acknowledgements

The authors acknowledge the essential support of the whole MDSF development team and the encouragement from Dr. C. Thornton, Manager of Spar System & Controls and Dr. L. Gregoris, director of Spar R & D in preparing this paper. Special thanks are given to Mr. Ken Podwalski of Canadian Space Agency for his support in performing the two simulation tasks described as examples in this paper. Finally, the authors wish to thank the Canadian Space Agency for granting permission to publish this paper.

References

1. K.H. Doetsch and J.A. Middleton, "Canada's Space Station Program" *Canadian Aeronautics and Space Journal* **33,** No. 4, 61–89 (1987).
2. D.F. Golla, K.J. Buhariwala, P.C. Hughes and G.M.T. D'Eleuterio, "Efficient algorithms for the dynamics simulation of structurally flexible manipulators" *CSME Journal* **13**(4)**,** 97–102 (1989).
3. P.C. Hughes and G.B. Sincarsin, "Dynamics of elastic chains: part a – body motion equations" *J. of Dyn. and Stability of Systs.,* **4**(3)**,** 209–226 (1989).
4. P.C. Hughes and G.B. Sincarsin, "Dynamics of elastic chains: part b – global dynamics" *J. of Dyn. and Stability of Systs.,* **4**(3)**,** 227–244 (1989).
5. P.C. Hughes, "Modal identities for elastic bodies, with application to vehicle dynamics and control" *ASME J. of Applied Mechanics* **47,** 177–184 (1980).
6. O. Ma, "Contact dynamics modelling for the simulation of the Space Station Manipulators handling payloads" *Proc. of IEEE Int. Conf. on Robotics & Auto.,* Nagoya, Japan (1995) **Vol. 2** pp. 1252–1258.
7. K.L. Johnson, *Contact Mechanics* (Cambridge Univ. Press, London, 1985).
8. O. Ma and M. Nahon, "A general method for computing the distance between two objects using optimization techniques" *Proc. of ASME Int. Conf. on Advances in Design and Auto.,* Phoenix, USA, **Vol. 1** (1992) pp. 109–117.
9. M. Nahon, "Determination of the interference distance between two objects using optimization techniques" *ASME Conf. on Advances in Design Auto.,* Minnesota (1994) **Vol. 1** pp. 1–6 (1994).
10. D.A. Haessig and B. Friedland, "On the modeling and simulation of friction, "*ASME Trans., J. of Dyn. Syst., Meas., and Ctrl* **113,** 354–362 (1991).
11. D.E. Whitney, "Quasi-static assembly of compliantly supported rigid parts" *ASME Trans. J. of Dyn. Syst., Meas., and Ctrl.* **104,** 65–77 (1982).
12. J. Maclean, C., Arnoldi, S. Chan and S. Assaf, "Simulation of the Space Station Manipulator System on MDSF, *The 2nd ESA Workshop on Simulators for European Space Programmes, ESTEC,* Noordwijk, October (1990) pp. 1–8.
13. J. Maclean and R. Carr, "Operational verification of dynamics systems using MDSF" *CASI Symposium on Space Mission Operations,* Montreal, Canada (1995) pp. 171–180.

https://doi.org/10.1017/S0263574797000076 Published online by Cambridge University Press