N94-14625

Modelling Robotic Systems with DADS

L. W. Churchill*, I. Sharf*

March 30, 1992

Abstract

With the appearance of general off-the-shelf software packages for the simulation of mechanical systems, modelling and simulation of mechanisms has become an easier task. The authors have recently learned one such package, DADS, to model the dynamics of rigid and flexible-link robotic manipulators. In this paper, we present this overview of our learning experiences with DADS, in the hope that it will shorten the learning process for others interested in this software.

1 Introduction

The practice of robotic-systems simulation is presently undergoing a transition from user-customized to off-the-shelf software. Of course, the development of new methods will always require the development of new computer programs to test them. But with the appearance of general, (relatively) easy to use, simulation packages it is no longer necessary to write your own program every time you need to simulate a new mechanical system.

One such package is DADS[1], the Dynamic Analysis and Design System. The University of Victoria's department of mechanical engineering has recently acquired

*Department of Mechanical Engineering, University of Victoria, Victoria, British Columbia, V8W 2Y2 Canada.

[1]DADS is a product of Computer Aided Design Software, Incorporated, P.O. Box 203, Oakdale, Iowa 52319.

DADS for research use. As the authors' work on recursive algorithms for dynamics simulation came to fruition, we decided to use DADS as a standard in accuracy and performance tests of our own software. Thus, our primary objective was to check the results of our simulation of flexible-body open chains, in particular robot arms, against those of a completely different approach. As a secondary objective, we wished to use the package's plotting and animating capabilities, with both our own and DADS' data, to generate graphical and animated output.

We have developed a simulation implementing recursive solutions to both the *inverse* and *forward* dynamics problems for rigid-link open chains. These simulations have been extensively tested and provide highly accurate results. Our inverse dynamics code uses, as input, the angular displacements, joint rates and accelerations and generates, as output, the control forces to be applied to the actuators of the chain in order to obtain those trajectories. The forward dynamics code uses, as input, the control forces to be applied to the actuators and generates, as output, the angular displacements and rates of the joints and the resultant trajectories of the end-points of the links. The control forces can be supplied either as a prescribed set of data points or by the inverse dynamics routine. The inverse and forward algorithms differ sufficiently that a good indication of the accuracy of the simulation can be obtained from the rms error of the integrated solution vector, the set of angular displacements and joint rates for the whole chain.

We planned to use our inverse and forward dynamics programs to establish a confidence level for DADS' rigid-link dynamics, after which we would assume a "lesser than or equal" corresponding confidence level for DADS' flexible-link dynamics.

Thus, we wished to employ DADS to:

- Solve the inverse dynamics problem for a rigid-link model and compare the resultant control forces with ours.
- Solve the forward (simulation) dynamics for a rigid-link model and compare the resultant solution vector, its rms error and the trajectories of the end-points of the links to ours.

- Include link flexibility in the DADS model and simulate the motion of the system.
- Plot and animate DADS' output.

2 Getting Started

DADS comes complete with three accompanying texts: the theoretical, user's and examples manuals. The theoretical manual [1] is actually a hardcover book describing the theory underlying the program's algorithms. The user's manual [2] is really a reference manual. It includes a brief introduction to the program and some instructions in the use of the postprocessor (plotting) and the geometry and animation routines, but consists primarily of detailed outlines of the elements with which mechanical systems are modelled. The examples manual [3] consists of a large number of DADS models of varying complexity, but for the most part, the descriptions are limited to hard copies of intermediate data files, output files and plots. This set of manuals becomes quite useful once one achieves sufficient familiarity with the software.

In our version of DADS, Revision 6.1, the most commonly used portions of the program have been implemented via a graphic user interface with the remaining portions running in ordinary text windows. DADS must be run from OpenWindows (a Sun interpretation of X-Windows) and a resource/defaults file named daDS must be present in the home directory at startup.

It is worth making a few remarks on DADS' implementation. DADS can be heavy on system resources and doesn't always work smoothly with other applications. It can shut itself down if it finds another program running at startup and can interfere with other programs' operations, particulary those using color graphics. The older portions of the package work well within their text windows; though DGE, the animation routine, won't permit line editting after file input is completed. A minor problem with both the new and old command windows is that they automatically close upon job completion. This is convenient if the job ends successfully, but if the

termination was abnormal, any error messages presented flash offscreen too quickly to be read. Error messages from the routine performing the *analysis* may be recovered from an information file, but error messages from other portions of the program (*i.e.* model definition, plotting, graphics and animation) are lost.

The new command shell windows also have another fault. DADS generally uses Xterm window conventions but has preset many of the usual options. In general, these are matters of little consequence but, in combination with other factors, one of these option settings has proven a consistent source of error: DADS sets the input field to follow the mouse-pointer location. When using the data entry windows, the mouse is useful in moving quickly from one input cell to another (though it would work just as well with mouse-selected input and, for general data entry, the tab key is often more convenient anyway). However, in the new command shells, data entry is recognized only from the command line at the bottom of the window. Unfortunately, input is also allowed, prompted and echoed in the dialogue section of the window. Of course, the programs' text messages are displayed in this dialogue pane and, after a menu selection, the mouse-pointer ends up there as well. These attributes combine with the dialogue pane's larger size and central location to make it a natural site for keyboard input, even though it is completely nonfunctional. The problem is compounded by the choice of white text on dark blue and black backgrounds for the dialogue pane and command line, respectively[2]. On a black and white display, the boundary between the two fields is indistinguishable and input is very easily misdirected, resulting in corrupted data and incomplete models.

DADS is organized in program segments corresponding to the different tasks involved in building a mathematical model of a mechanical system. A model is defined and initially configured in the *preprocessor*. The simulation is then performed by the analysis package. If desired, plotting and data manipulation can be done in the *postprocessor*. Graphic representations of the model elements are developed with the geometry routines. And, finally, the simulation results and graphic representations

[2]These default settings can be changed by modifying the daDS file. Try changing *DADSMessage*background from navy to grey55 or slate blue.

are visually integrated via the animation program. In addition, DADS is equipped with a number of supplementary conversion routines which translate data and output files from one program segment for use in another. The heart of the package, however, consists of the preprocessor and analysis routines with which DADS performs simulations.

3 Simulation

Before one begins to model with DADS, it is helpful to develop an overview of the way the program works. In particular, we mention some assumptions which seem to be implicit in the program's operation and provide an outline of the different types and functions of DADS' elements.

Many of the difficulties we encountered in modelling our systems were rooted in the choice of reference frames we made for the links. DADS seems to have been originally designed to use a global reference frame in combination with local body center-of-mass frames. The system is assumed to operate with a given orientation to the ground and under the influence of gravity and possibly dissipative forces as well. These properties are undoubtedly those most appropriate for arbitrary mechanical systems but, in the analysis of robot arms, other situations are also common. The recursive algorithms we use make joint-based reference frames a natural choice for the links. And, since our main application of interest is the Canadarm, we deal with all external forces as special cases and default to a weightless, frictionless environment. DADS seemed capable of adapting to our point of view so we elected to define the DADS models in the same way we defined our recursive models. But, in places, DADS still implicitly relies on body center-of-mass reference frames, which led to several problems.

A DADS model is defined in terms of a variety of program "elements." The shear size of the program's element library can be overwhelming, leaving a new user bewildered as to which elements would be appropriate to build and drive a model.

At the highest level is the *system* element which defines the type of analysis to be performed—static, dynamic, inverse or kinematic—and sets global parameters such

as units, the run time and printing intervals. The gravitational field vector must be set here (it helps to define the units), though a scaling factor is also provided to adjust the magnitude. Tolerances for DADS assembly analysis and LU factorization are specified. Matrix operations should be set to SPARSE as the alternative, since FULL matrix operations, doesn't seem to work correctly. DADS is capable of performing a (useful) check that the model will assemble correctly to within the given assembly tolerance. Since we deal with relatively straightforward assemblies, we tend to set this tolerance value quite small (10^{-6}.) The type of reference frame to be used in the analysis is also selected here. Possible choices are global, local (body center of mass) or NCBF (non-centroidal body frames.) Contrary to our expectations, this option mainly affects the interpretation of reference points in the input data, except for reaction forces, whose coordinate frames are specified elsewhere. Output data is given in terms of the global and, sometimes, the local (body center-of-mass) frames. Finally, a debug flag may be set here. We found this useful mainly because it turns on the time echoing in the analysis window.

Each of the analysis types have their own element. We discuss only the elements relevant to the present application—inverse and dynamic. In the *inverse* element, one specifies the coordinates used to output the reaction forces. The analysis step size and solution tolerance is also determined here. We generally found the default values adequate for a first run. Only after the system was in working order did we try for greater precision. In the *dynamic* element one specifies the maximum integration step and the solution and integration tolerances. The defaults are adequate to get the system working but the tolerances had to be lowered to get the accuracies we desired.

Data elements describe the physical components of the system. In a *body* element one describes the physical properties of an individual component, defined in local (not NCBF) coordinates. One can apply external forces directly to the center of mass of a body but, for our system, we found another mechanism more convenient. If NCBF is the type of reference frame selected in the system element,

then all bodies must be associated with a corresponding *reference frame* element defining their coordinate system. Similarly, if a body is specified to be flexible it must be associated with a corresponding *flexible body* element which permits the definition of damping and/or external forces and points to a data file containing vibrational mode information from a finite elements program. Other data elements include an *initial condition* element (if absent, the associated value defaults to zero) and a *curve* element which defines a function in terms of a prescribed set of data points. We usually specified our control forces via curve elements. Curve elements can read data from a text file but, afterwards, such data cannot be edited. For this reason we suggest saving a model's configuration and curve elements separately. This greatly simplifies switching between sets of control forces, for example. We also suggest frequent saving when loading large files into curve element sets, as DADS can shut down unexpectedly during these operations.

The joint-constraint elements make up a large library of the various means for joining the bodies in the model. The name is indicative of their function—these elements are connectors with specified degrees of freedom but are *passive,* not active. Note that, despite the description in the user's manual, the order in which the bodies connected by the joint are specified can be significant. The robot arms that we have modelled with DADS use only *bracket joint* elements (connectors with no degrees of freedom) and *revolute joint* elements.

Most of the other-constraint elements enable one to model the physical consequences of the bodies' dimensions and the system's overall geometry. As our models assume an idealized geometry, we found most of these unneccesary. In the inverse dynamics analysis, joints are made to move with the *driver* element. A driver element may be specified as one of several types, driving: any of the coordinates, any component of the velocity, a distance, a relative angle or a relative translation. The driving function may be specified in a variety of ways: a curve element, a simple polynomial, a simple harmonic function, as the control output of a large set of control elements or in terms of a user-supplied subroutine. We used relative angle drivers in our inverse

dynamics models. The polynomial and harmonic functions were adequate for test cases though we will likely need to develop our own subroutines, eventually.

There are seven force elements which provide a wide variety of options for applying forces in the forward dynamics analysis. We applied control forces to our revolute joints by means of *rotational spring-damper-actuator* elements. RSDA's are, effectively, damped motors with torsion. The three types of torque contribution may have both constant and time-varying components. However, contrary to expectation, the actuator torque is applied backwards so as to be dissipative. This may be corrected in several ways. One is to change both the direction of the rotation axis or the order of the connecting bodies in the associated revolute joint element definition. Simpler, however, is adjusting the curve element to scale the applied torques by a value of -1.

To summarize, the set of elements we used in our inverse analysis consisted of a system and inverse element with a set of body and reference frame pairs connected by revolute joint-driver pairs. For dynamic analysis we used a system and dynamic element with a set of body and reference frame pairs connected by revolute joint-RSDA pairs. Initial condition elements were used and control forces were specified to the RSDA's via curve elements. For flexible-link dynamics we moved to body, reference frame, flexible body element triples.

4 Visual Presentation

As well as performing dynamical analyses, DADS is capable of plotting and/or animating the results. Plotting is accomplished with the postprocessor. Essentially any value associated with a component of the model may be plotted. DADS is also capable of combining data from different runs and data may also be read from (or saved to) text files. Plots may be displayed on screen or directed to files in a wide variety of formats.

Before animating a model, it is necessary to describe the geometries of the components. This is done with Geomake, one of the older portions of DADS. A disadvantage of this routine is that once one has created the pieces that make up an

animation, they cannot be edited. As a result, we soon learned not to build large parts files. Instead we defined command files to create the parts and saved them in separate files to be combined later as desired. DADS geometry definition is general and seems capable of generating virtually any desired object. As we were building a simple robot arm without the end effector we dealt with ordinary cylinders. In order to smooth the appearance of the arm's joints we added spheres to the proximal end of each cylinder. Also, since the geometric body description needn't correspond to its modelled dimensions, our payload was modelled as a cylinder much shorter than its actual length, enabling a clearer view of the motion of the small links at the end of the arm. These simple additions transformed the original collection of stubby cylinders into a continuous articulated arm. Geomake can create body-geometries in one or more colors but the animation routine requires one color per part. To begin with, we suggest white.

The output of Geomake must be converted to a format suitable for the animation routine. This is accomplished with a program called CONV, but this routine will not work if called from DADS' window menu (it builds an empty data file.) One shoul execute CONV from a system command line in order to get it to work properly. The other conversion routine, DADS2MOD works well.

DADS animation is impressive and easy to learn. Previously created and converted geometries are stored in .def files and may be modified and saved easily from within DADS Graphic Environment (DGE.) In fact, after one becomes familiar with DGE, the .def files become quite readable and may be simply modified with a text editor. The DGE is command driven, but a graphic interface may also be started and is to be highly recommended, though it should be called after the model has been "assembled" by viewing the first frame. DGE has several viewing modes and both the viewpoint and lighting may be changed at will. Color and shading are good and, while the animation was noticably jerky on our architecture (a SPARCstation IPC networked to a Sun 4), the speed was within acceptable limits. DGE's graphic window is small and the program will not resize it while running. However this flaw

may be overcome by executing DGE from a system command line (rather than DADS' window menu) where one may specify the starting size for the graphics window as a command option. We found DADS animations were often helpful in understanding the motions resulting from the application of arbitrary forces.

5 Results and Conclusions

Our first results with DADS were obtained for rigid-link models. We tested two systems, a single link rotating about a fixed base and a six-link system modelled after the Canadarm, with solution and integration tolerances of 10^{-6} and 10^{-8}, respectively. With the first model, we compared the joint reaction forces from DADS' inverse dynamics to the control forces generated with our software. We then used the joint reaction forces as input to DADS' forward dynamics and compared the integrated solution vector to the prescribed input trajectories. The agreement between these was good, the difference appearing only in the last decimal of DADS' single precision output. Curiously, the joint reaction forces agreed with our control forces only to an average of about four decimal digits. When DADS' forward dynamics was run with our control forces the integrated solution vector and resultant trajectories agreed with the originals to three or four digits. For the second more complicated system, even using DADS' joint reaction forces as input to the forward dynamics results in only four digits agreement between the output trajectories and the prescribed inputs. We concluded that, for a general robotic system, DADS' confidence level was about four digits.

Our results with flexible-link models began very poorly. The problem lay with an undocumented aspect of DADS' handling of flexible bodies. DADS is capable of using finite element data generated by a number of different programs. We used ANSYS and chose a body-fixed coordinate system coinciding with the NCBF coordinate system assigned to the body in DADS. However, for flexible bodies, DADS reinterprets the postion specified for the body center of mass as the origin of the flexible element coordinate system, with other complicated consequences as well. This prob-

lem was fixed by reassigning the finite element coordinate origin. A second difficulty was encountered when we modelled the flexible links as single-element cantilevered-beams. With this model, we could not obtain a solution in a reasonable amount of time. This problem was fixed by using a five-element cantilevered-beam model.

The data we were interested in comparing included the trajectories of the end-points of the links. DADS' output actually gives the trajectories of the centers of mass of the links. With flexible links, these values cannot be immediately converted to end-point trajectories, so we attempted to use a *point-of-interest* data element. This element is designed to provide information about nodes of interest in flexible bodies. Unfortunately, the output data for this element was quite incorrect, possibly a casualty of the use of two different coordinate systems, in DADS and the finite element package, for the link.

We have been able to obtain flexible-link output for a single-link system of a well-known spin-up problem. (Please refer to Kane *et al.* [4] for details.) The transverse tip deflection is shown in Figure 1 and displays the divergent behaviour previously obtained with other multibody simulation packages [4]. At present, we are involved in incorporating link flexibility into the more sophisticated six-link model of the Canadarm, for the purpose of doing detailed comparisons between DADS' results and those of our simulation.

To conclude, we have employed DADS to model a particular class of mechanical systems—robotic manipulators with rigid and/or flexible links. We have discussed the various elements available in the package for constructing the model of such a system. Also, some of the difficulties encountered in the process of using DADS were noted. A short description of DADS plotting and animation capabilities was given together with out experiences of using them. Comparison of DADS results for a rigid-link manipulator demonstrates good agreement with the results of our own inverse and forward dynamics software. Finally, results of the simulation of a flexible beam with DADS support the previous claim regarding the lack of geometric stiffening terms in the existing multibody computer programs.

References

[1] Haug, Edward J., *Computer Aided Kinematics and Dynamics of Mechanical Systems,* Allyn and Bacon, 1989.

[2] *DADS User's Manual,* Rev. 6.0, CADSI, July 1989.

[3] *DADS Examples Manual,* Rev. 6.0, CADSI, July 1989.

[4] Kane, T. R., Ryan, R. R. & Banerjee, A. K. (1987). "Dynamics of a Cantilever Beam Attached to a Moving Base," *J. Guidance, Control, and Dynamics,* **10**, 2, pp. 139-151.

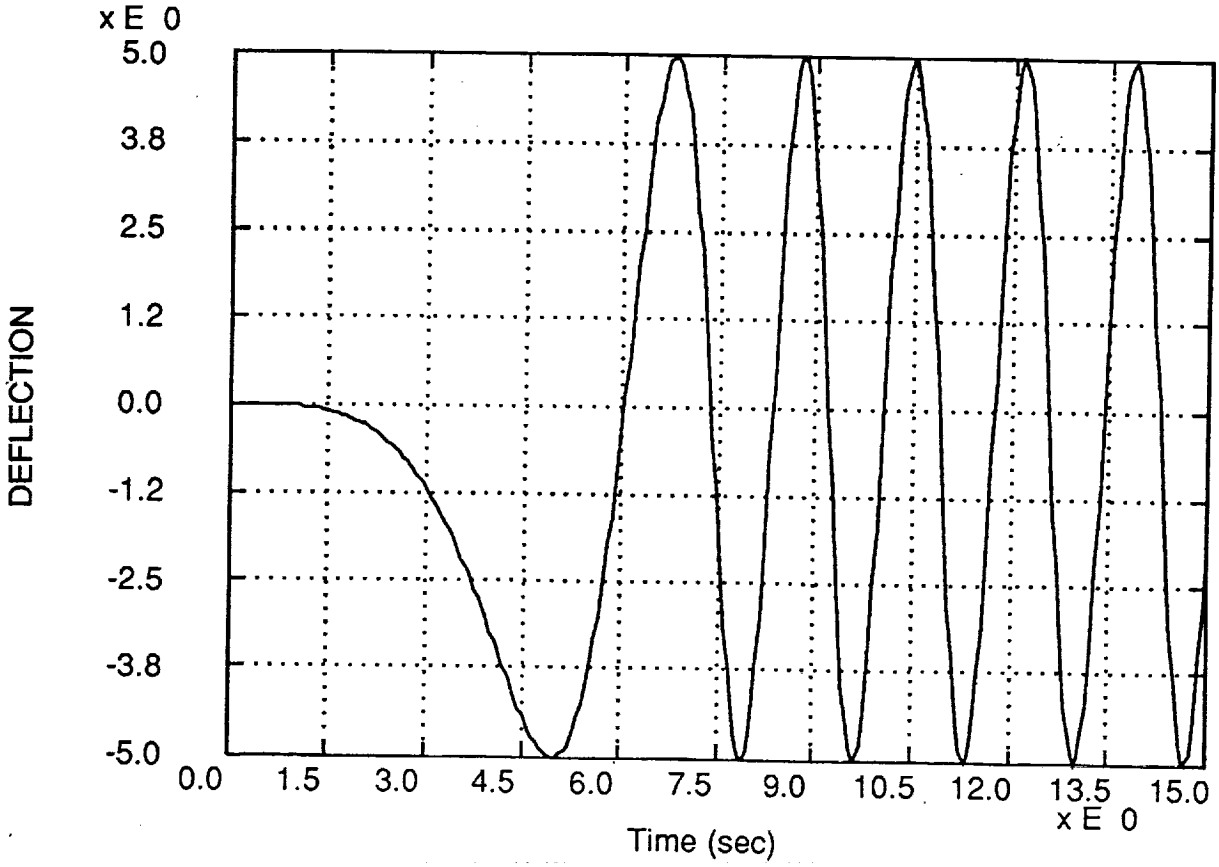

Figure 1: Transverse Deflection (m)