10th European ADAMS Users' Conference
Frankfurt, November 14-15th 1995

A Link Between ADAMS and the Bath Constraint Modeller

Ben Twyman
University of Bath, UK
Sean Biggs
MDI Ltd, Leamington Spa, UK

1. Introduction

This presentation describes work being carried out as part of a research programme at the University of Bath, in collaboration with MDI Ltd, CIMIO, AMTRI, CMB Technology, and Unilever Research. The project aim is to create a software environment for machinery design, based on the constraint modelling ideas that resulted from one of the research group's previous projects. The software environment is called SWORDS.

2. The SWORDS system

The system basically consists of an environment where the user can build up the constraints on a design problem, with a graphics area where geometry can be displayed. The macro language behind it gives it the flexibility to handle a wide range of design problems.

An optimisation routine built into the software then makes changes to variables defined by the user, in an attempt to provide design solutions that meet all the constraints. Often constraints specified will need values returned from dynamic analysis in order to be evaluated. For this purpose the link with ADAMS has been created.

Another interesting feature is the ability to build and evaluate mechanisms synthesised by the CAMFORD software, which resulted from a previous research project. This allows the user to define the motion of the output of a mechanism as a function or planar path. It searches for mechanisms in the library which most closely match the desired output. The library contains files that cover four, five, and six-bar chains, slider-cranks, and sliders driven by four and five-bar chains. Parameterised models of these linkages have been built in ADAMS/View.

The following sections describe how these tools have been brought together. The result is a system which can select and then optimise mechanisms to perform particular tasks, using dynamic properties calculated in ADAMS in the constraint functions.

3. Parameterisation of linkage models in ADAMS

The parameterised ADAMS models are defined using the location information associated with markers on the ground part, positioned at the joints of the mechanism. These markers are called "Master Markers" and are used with the

parametric expressions introduced with version 8.0 of ADAMS/View to define every other marker in the model.

3.1 Expressions relating to marker location

Suppose we have a Master marker called d1 which defines the position of a joint. If we create another marker which is to have the same position as marker d1 then we can define the new marker's position using the following expression in ADAMS/View,

(.mod1.ground.d1.location)

Note that the expression is in brackets and has a ".location" added to the marker name. This expression will work for any marker to be created on a part which has its co-ordinate system coincident with the ground part, while local co-ordinates have to be handled with one of the following commands,

a)

(loc_loc(.mod1.par2.mar3.location,.mod1.par2,.mod1.par6))

This changes the location of mar3 in the co-ordinate system of part 2 to the position of mar3 expressed in the co-ordinate system of part 6, (both part 2 and part 6 have local co-ordinate systems not coincident with the ground part).

b)

(loc_local(.mod1.ground.d1.location,.mod1.par2))

This changes the location of ground marker d1 into the local co-ordinate system of part 2.

c)

(loc_global(.mod1.par2.mar3.location,.mod1.par2))

This changes the local co-ordinate position of marker 3 into the ground co-ordinates.

3.2 Expressions relating to marker orientation

Similar expressions exist for orientations. These must be input in the orientation field, not in the 'along_axis_orientation' or 'in_plane_orientation' fields.

For example the following expression will extract the Euler angle sequence for the marker in the expression and apply it to the marker being parameterised.

(.mod1.ground.d1.orientation)

The along axis orientation method is implemented with the following expression typed in to the orientation field.

(ori_along_axis(.mod1.ground.d1,.mod1.ground.d2,"Z"))

This will point the Z axis of the marker being parameterised along a vector from marker d1 to marker d2.

The axis and plane method is implemented using,

(ori_in_plane(.mod1.ground.d1,.mod1.ground.d2,.mod1.ground.d3,"Y_YX"))

This expression will orient the Y axis along a vector from marker d1 to d2 and position the X axis such that the vector from marker d1 to marker d3 is in the YX plane.

3.3 Changing the geometry

The geometry of the linkage modelled using the parametric expressions can be easily modified by shifting the position of the Master markers positioned on the ground part. The Master marker positions can be defined using Design Variables available under the Global menu in ADAMS/View and for a 2 dimensional problem we would only require two variables per Master marker with the Real_Value for each design variable dictating the position of the marker.
The location field for ground marker d1 would thus look as follows,

(d1x),(d1y),0.0

where (d1x) and (d1y) are Design Variables.

Thus to fully automate the modification of the parametric models we need to write an ADAMS/View macro that will modify the Real_Value statements for the Design Variables and this very simple macro can be seen in appendix 2.

The macro consists of a user entered command which is required to run the macro, a list of the macro input parameters (in this case the Real_Values for the Design Variables) and the commands to be executed. Note that the commands are a modified version of the commands to be found in the aview.log file which is written during the ADAMS/View session.

In order to run the macro we issue the user entered command and provide the numeric inputs for the macro parameters. The inputs to the macro can be provided by an ADAMS/View panel which is automatically created by the macro or by typing in at the command line or by reading the command in a command (.cmd) file. An example of the command to modify the four-bar is as follows.

```
geom d1x=34.5 d1y=0.0 d2x=0.0 d2y=100.0 d3x=200.0 d3y=150.0 &
d4x=200.0 d4y=0.0 d5x=45.0 d5y=-100.0
```

An example of a parameterised four-bar linkage can be found in the "aview/examples/parametric/4bar" directory of the installation media for version 8.1 of ADAMS. This can be modified as detailed in appendix 1 to allow the macro to be loaded from a custom panel.

4. Automatic link between the two programs

After trying a number of ways of linking the two programs to allow automatic control of ADAMS by SWORDS, the following was found to be the most reliable. It makes use of named pipes in the UNIX operating system to provide the means of communicating data.

From within a SWORDS macro, system commands are sent to load ADAMS/View and ADAMS/Solver as separate processes. The command to load ADAMS/Solver

forces it to read commands from a file called "pipe.acf" which is a UNIX named pipe. The "aview.cmd" file contains instructions to read a command file called "pipe.cmd", which is also a named pipe. The two processes therefore sit and wait for commands, in the form of text strings, to be sent down the pipe. Functions have been written in SWORDS to create formatted text strings which contain commands for ADAMS/View and ADAMS/Solver.

SWORDS opens each of these pipes, and when it needs ADAMS to perform an analysis, it sends commands to do the following.

Actions performed by ADAMS/View
- load a parameterised model for the relevant linkage type.
- update the model with the values for the particular mechanism to be simulated.
- write a data set file for ADAMS/Solver.
- write an empty text file to act as confirmation for SWORDS that the new data set has been prepared.

Actions performed by ADAMS/Solver
- load the data set file
- run the simulation
- write a file (actually a system states file), to confirm to SWORDS that the analysis is complete and the new results file has been written.

After this sequence of events, SWORDS can read the results file (actually a ".req" file) to decide on which set of parameter values to test next. Before starting on the next iteration of the process, it deletes the confirmation files written by ADAMS. In theory it is possible to pipe the results back to SWORDS, but doing so has not proved as reliable as simple file transfer.

A useful extension of the technique described here is to use sockets or remote procedure calls to handle the data communication, instead of the named pipes. This allows the software packages to be running on different machines.

5. Example Application

The example to be presented involves the selection and analysis of a linkage mechanism to compare with a cam driven mechanism.

The motion profile of the output was designed using the CAMFORD software. It included a dwell period followed by a period of constant velocity. The profile was assumed to be a translational displacement, and the library files of four and five-bar linkages searched to find a suitable match. It was clear that the motion defined was particularly suitable for a linkage mechanism, as there were very good matches with both linkage types. The parameter values for the best of the four-bar linkages were taken, and inserted in the ADAMS/View model to run kinetostatic analyses.

A cam mechanism, with a slider output connected to a rocker follower was also generated by the system, and built in ADAMS. It is possible to investigate the trade-offs of using one concept as opposed to another; for example, to look at the effect on drive torque and shaking forces by driving the mechanisms at higher speeds.

6. Conclusions

The presentation has shown the following.

- an example of interfacing ADAMS with another software package.
- the parameterisation of models of linkage mechanisms in ADAMS/View.
- a system whereby a linkage can be selected, and if needs be modified, to perform a particular task, yet still exhibit favourable dynamic properties.

Acknowledgement

The work described here is part of a project carried out under the LINK programme in the Design of High Speed Machinery, funded by EPSRC and DTI. The authors gratefully acknowledge this support, and that of the collaborating companies.

Appendix 1 - modifying "para4bar.cmd"

The process to modify the para4bar.cmd model to allow geometry to be input from a custom panel is as follows.

1. Create the "aview.cmd" file (as in appendix 3) and place it in the same directory as the "para4bar.cmd" file. ADAMS/View will automatically read the "aview.cmd" file and load the macro and set up the button to display the panel.

2. Create the macro file called "geom.mac" (as in appendix 2) and also have this in the directory with the "aview.cmd".

3. Load the "para4bar.cmd" model into ADAMS/View.

4. Create the Design Variables to drive the Master markers.
   Design variables are created using the following menu picks.
   Global, Next_Page, Variable, Create.
   Backspace Design Variable name and create variables
   d1x,d1y,d2x,d2y,d3x,d3y,d4x and d4y.
   The Real_Values of each Design Variable define the x or y locations for Master markers.
   Master markers are all on the ground part and are as follows,
   CRANK_GROUND
   CRANK_COUPLER
   CRANK_FOLLOWER
   FOLLOWER_GROUND

5. Modify the Master markers locations by clearing the references to actual co-ordinates and defining positions as below

   (d1x),(d1y),0.0

   where (d1x) and (d1y) are the Design Variables that locate marker CRANK_GROUND. etc

6. Save the model as a command file so that the changes are saved.

Once all the Master markers are positioned in this manner it should be possible to modify the model by typing numbers into the panel displayed by pushing the "Modify fourbar" button under the main menu.

Appendix 2 - geom.mac

```
!USER_ENTERED_COMMAND geom
! $d1x:t=real:u=0.0
! $d1y:t=real:u=0.0
! $d2x:t=real:u=0.0
! $d2y:t=real:u=11.0
! $d3x:t=real:u=23.0
! $d3y:t=real:u=17.0
! $d4x:t=real:u=23.0
! $d4y:t=real:u=0.0

 variable modify variable_name=.param4bar.d1x real_value=($d1x) units=length
 variable modify variable_name=.param4bar.d1y real_value=($d1y) units=length
 variable modify variable_name=.param4bar.d2x real_value=($d2x) units=length
 variable modify variable_name=.param4bar.d2y real_value=($d2y) units=length
 variable modify variable_name=.param4bar.d3x real_value=($d3x) units=length
 variable modify variable_name=.param4bar.d3y real_value=($d3y) units=length
 variable modify variable_name=.param4bar.d4x real_value=($d4x) units=length
 variable modify variable_name=.param4bar.d4y real_value=($d4y) units=length
```

Appendix 3 - aview.cmd

```
! Read in DESIGN VARIABLE modify macro
! for use with para4bar.cmd.

 macro read macro_name=.MAC1 file_name="geom.mac"&
 user_entered_command=geom wrap_in_undo=yes create_panel=yes

! Create menu button on root menu to access change panel

 menu create menu_name=.gui.root row_number=4  &
 label_string="MODIFY FOURBAR"  &
 command_string="PANEL DISPLAY PANEL =.gui.macro_MAC1"
```

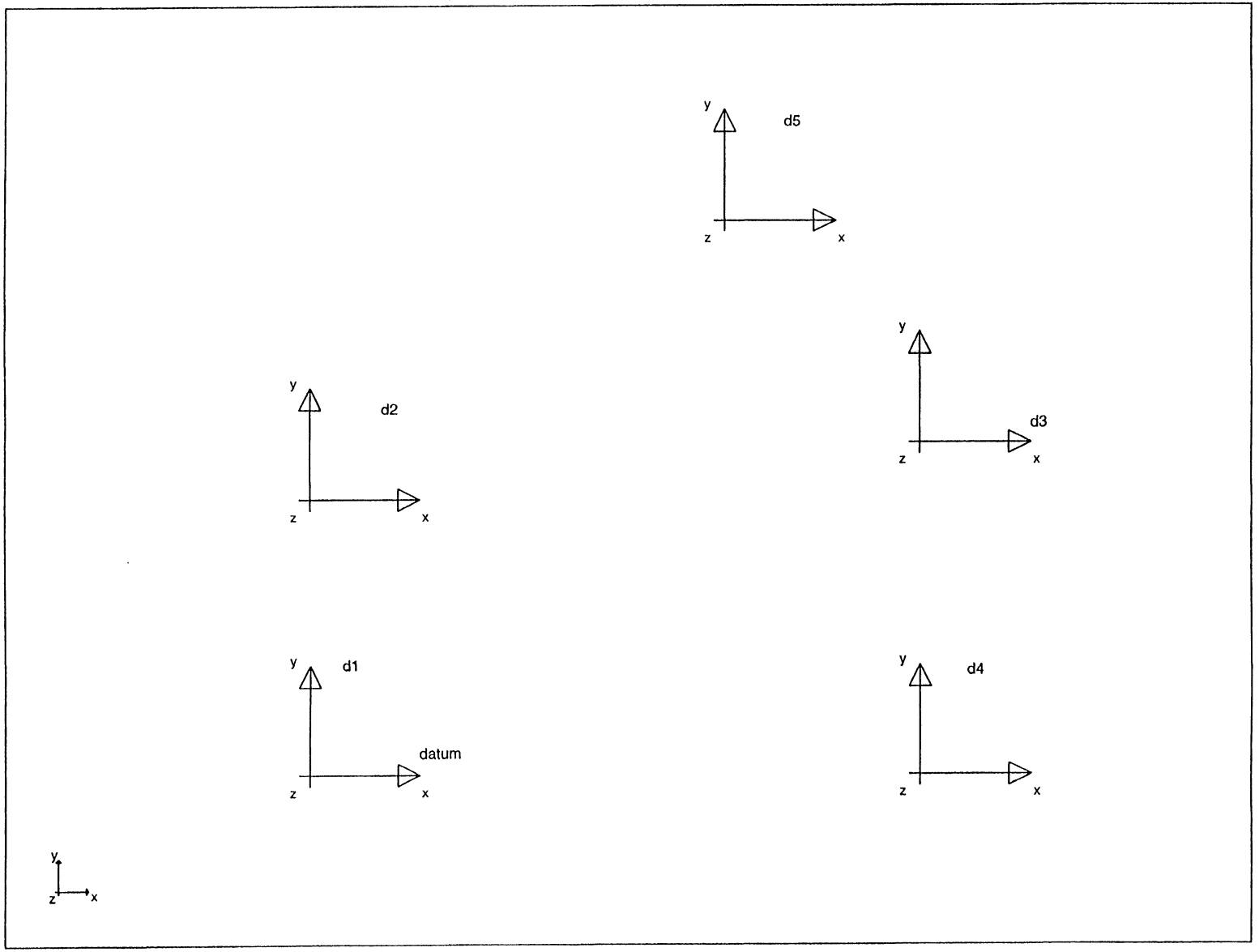

Figure 1 - Master markers for the four-bar chain

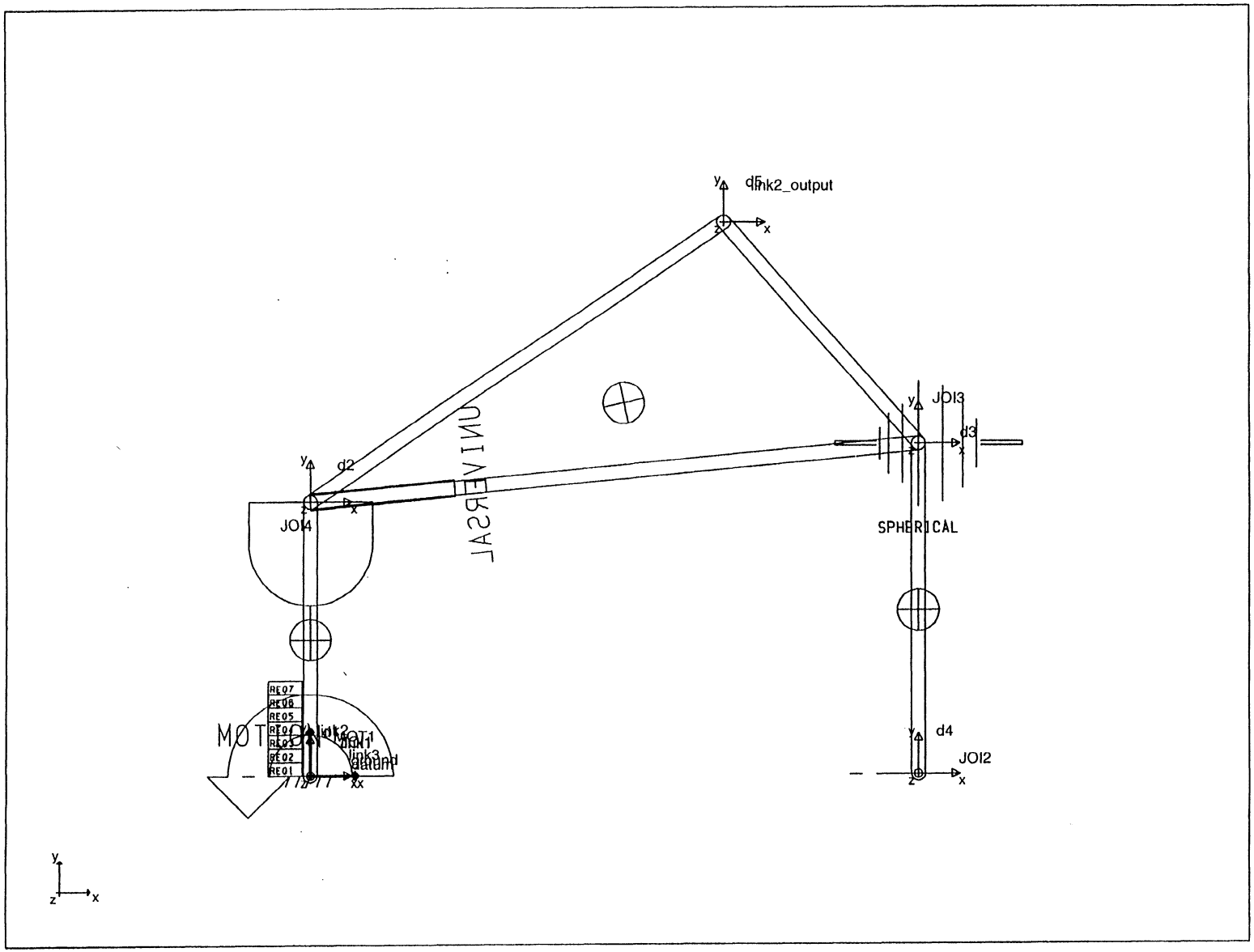

Figure 2 .e complete model

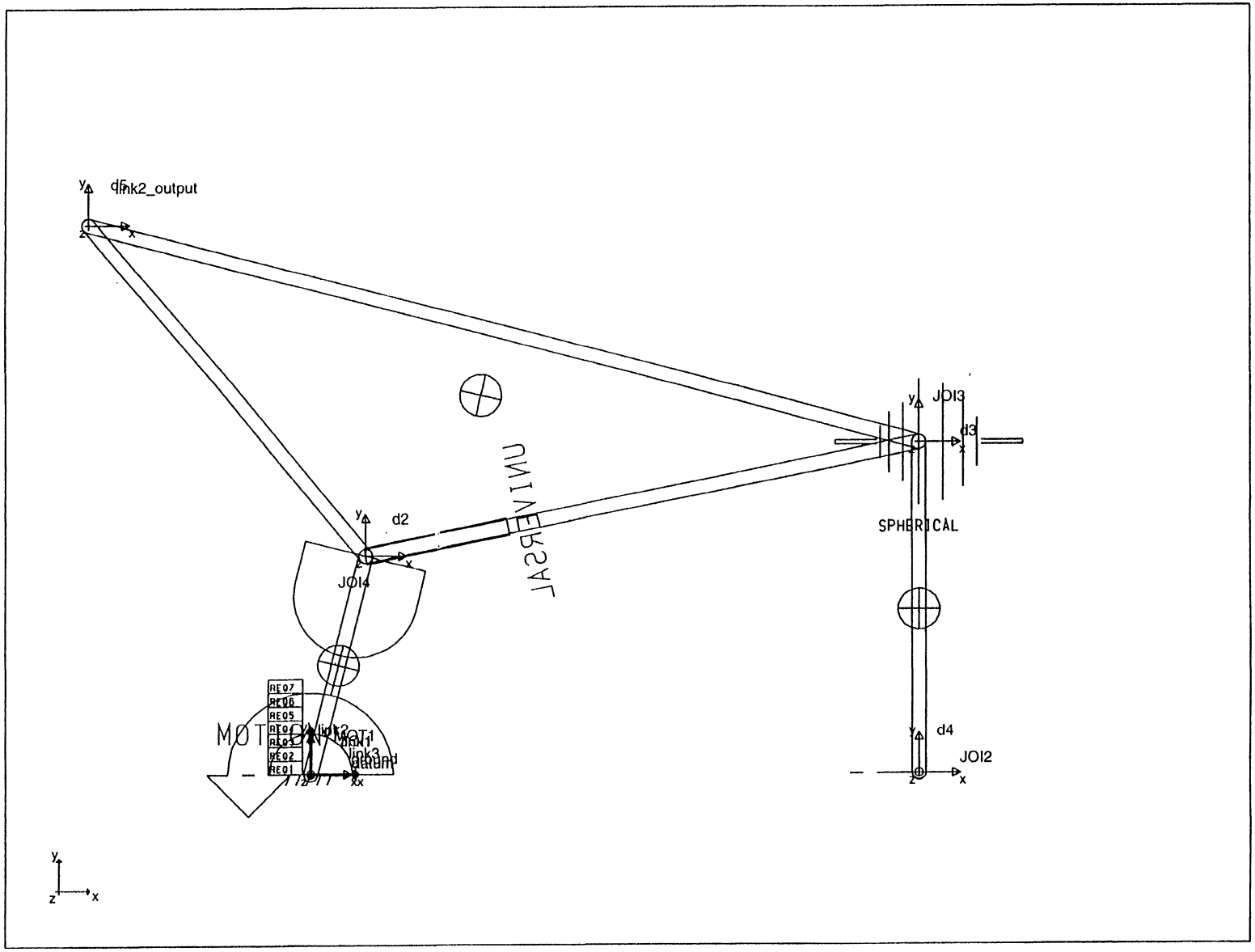

Figure 3 - Master marker 'd2' and 'd5' moved to change the mechanism

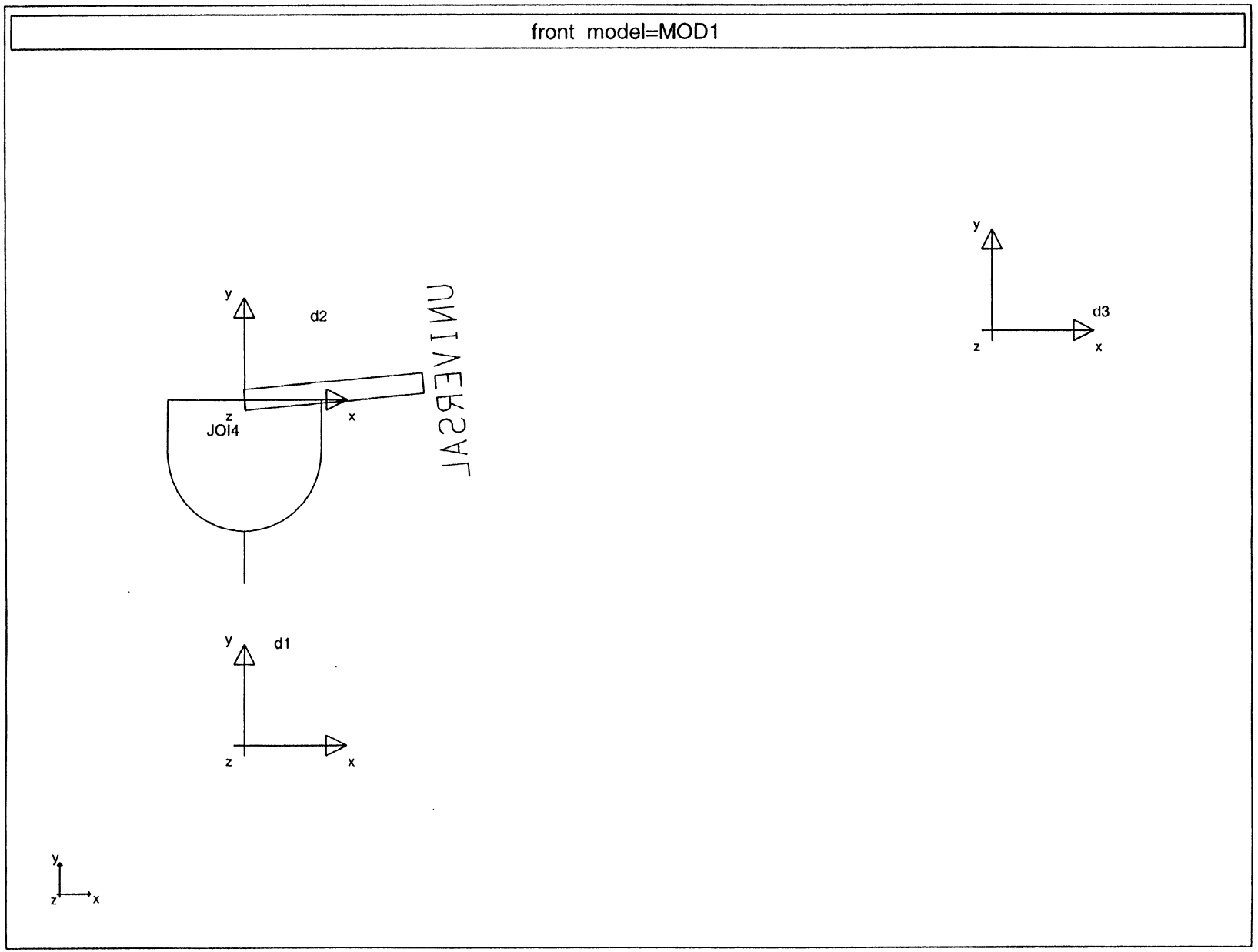

Figure 4 - . .ıding up the universal joint

```
constraint create joint universal  &
  joint_name = .MOD1.JOI4  &
  adams_id = 4  &
  i_marker_name = .MOD1.link1.MAR17  &
  j_marker_name = .MOD1.link2.MAR18

marker modify  &
  marker_name = .MOD1.link1.MAR17  &
  location =  &
    (.MOD1.ground.d2.location)  &
  orientation =  &
    (ORI_IN_PLANE(.MOD1.ground.d1, .MOD1.ground.d2, .MOD1.ground.d3, "Y_YZ"))  &
  relative_to = link1

defaults coordinate_system  &
  default_coordinate_system = .MOD1.ground

marker modify  &
  marker_name = .MOD1.link2.MAR18  &
  location =  &
    (.MOD1.ground.d2.location)  &
  orientation =  &
    (ORI_IN_PLANE(.MOD1.ground.d3, .MOD1.ground.d2, .MOD1.ground.d1, "Y_YX"))  &
  relative_to = link2
```

Figure 5 - ADAMS/View code to create the parametric universal joint

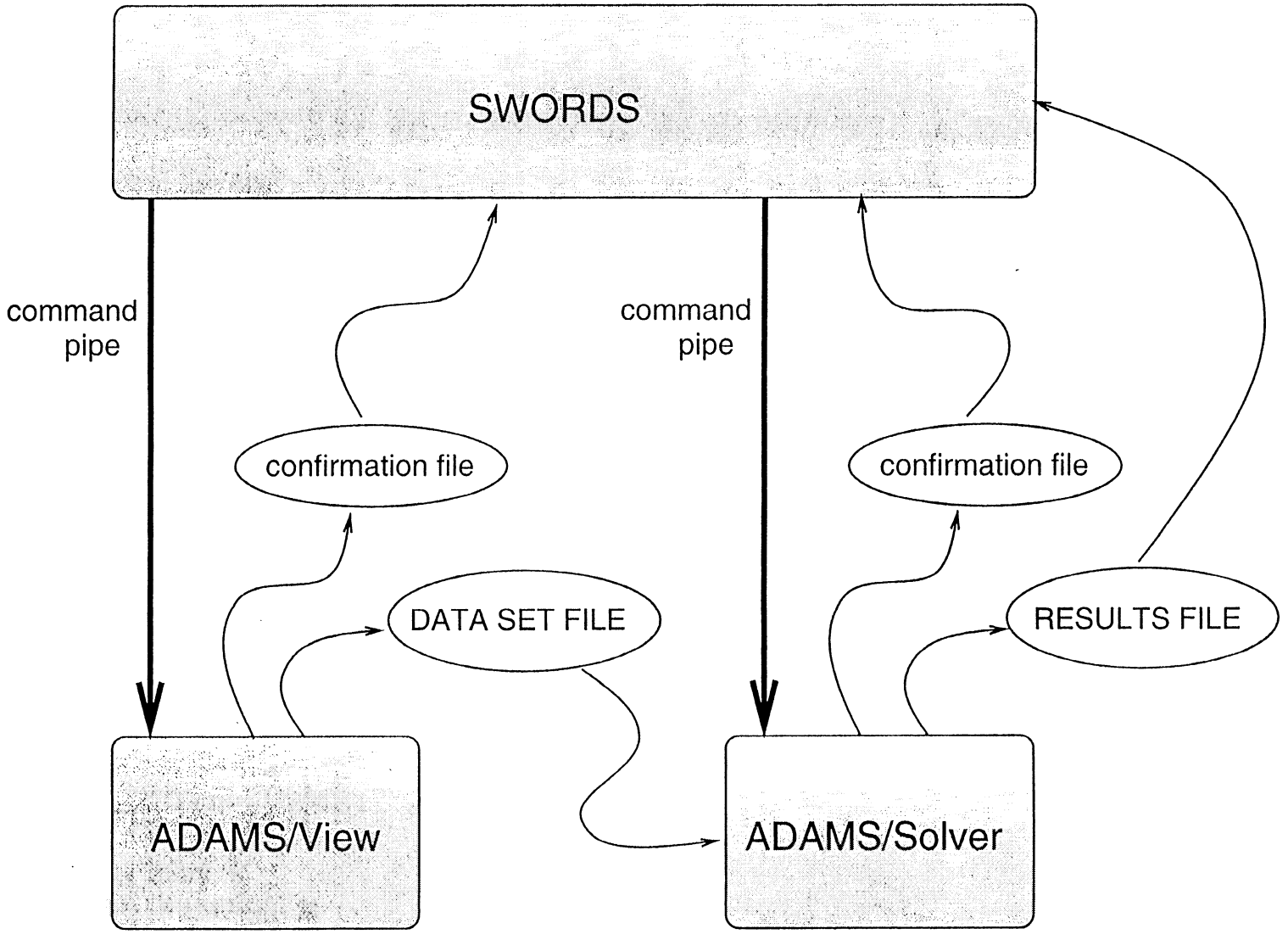

Figure 6 - Communication between SWORDS and ADAMS

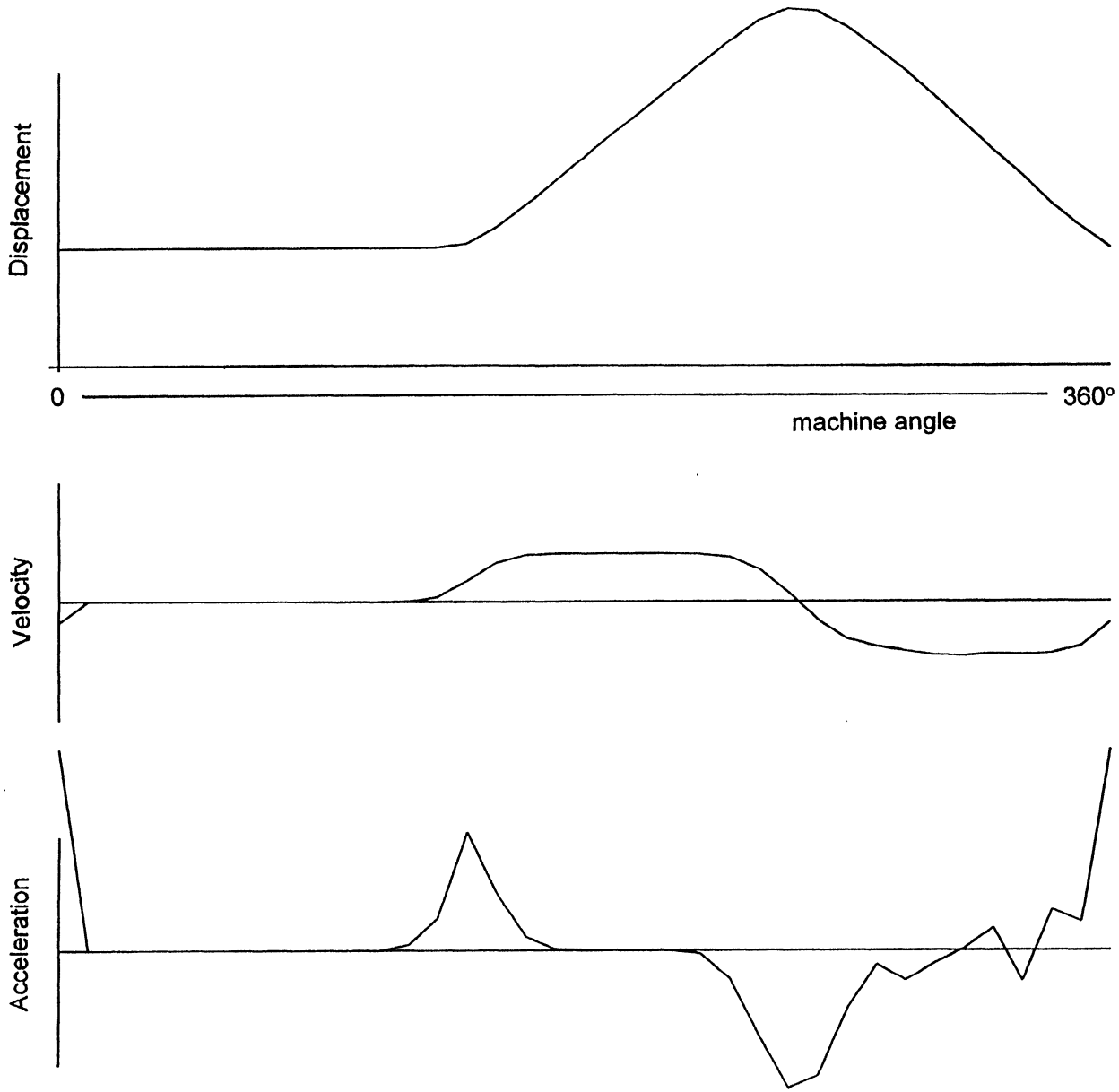

Figure 7 - Displacement, velocity, acceleration graphs for the required motion

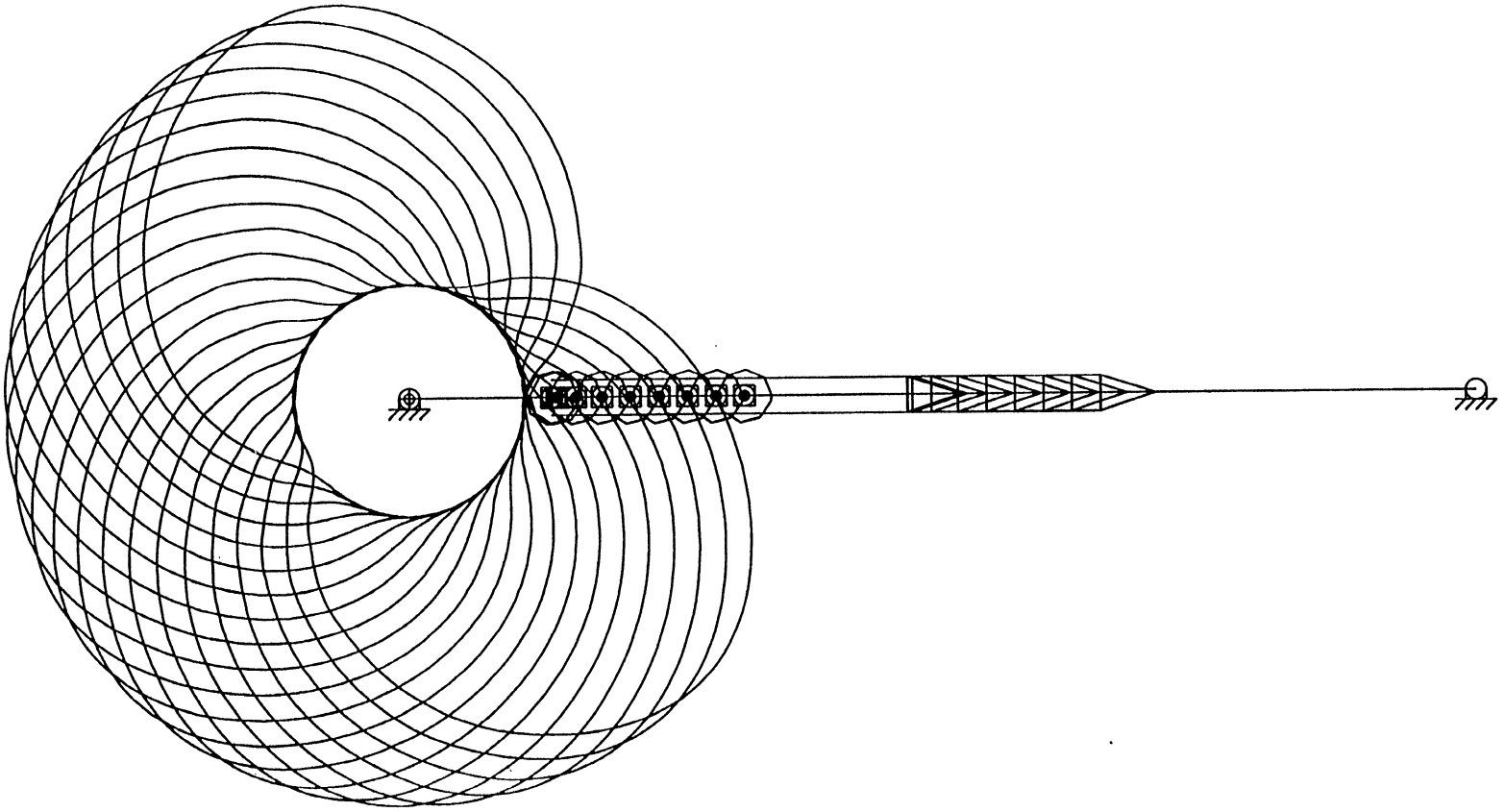

Figure 8 - Cam/slider achieving required motion

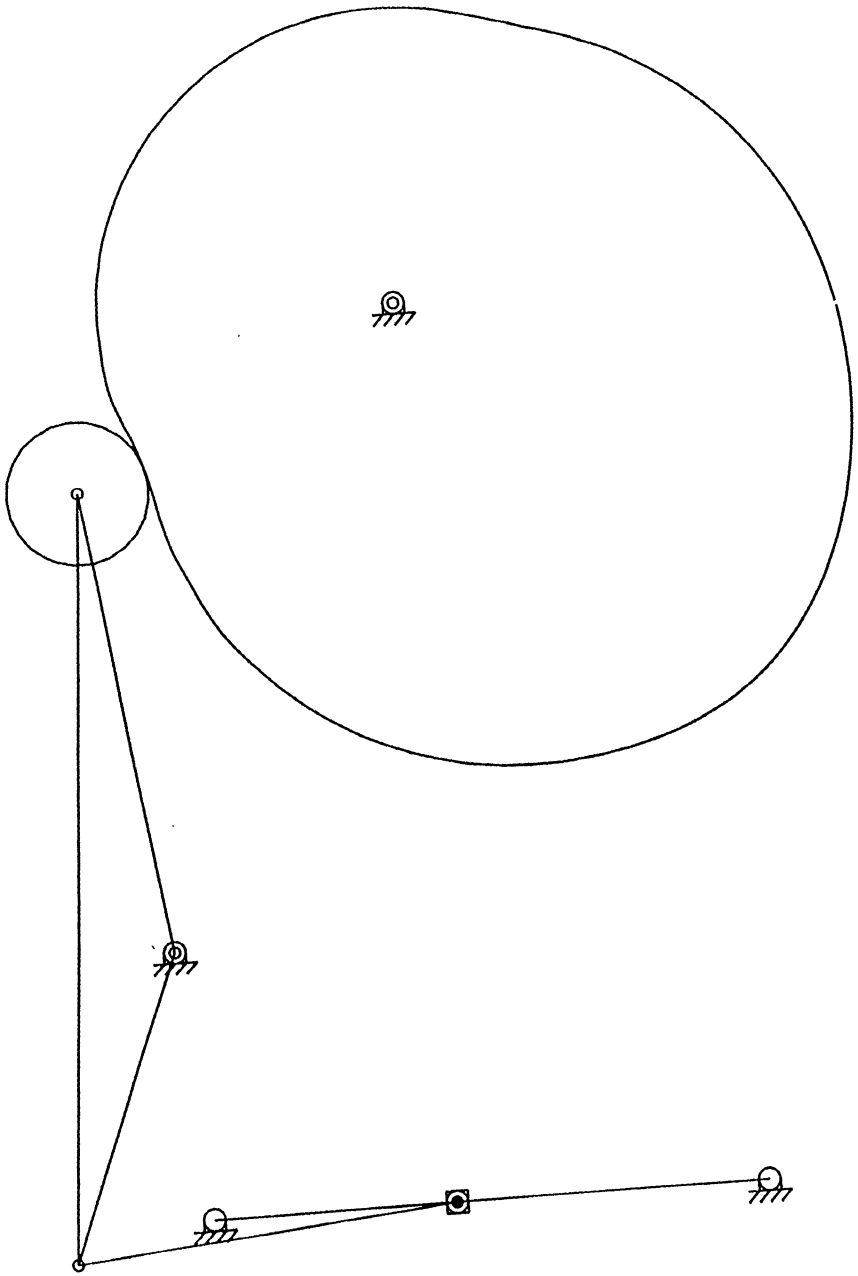

Figure 9 - Cam/rocker achieving required motion

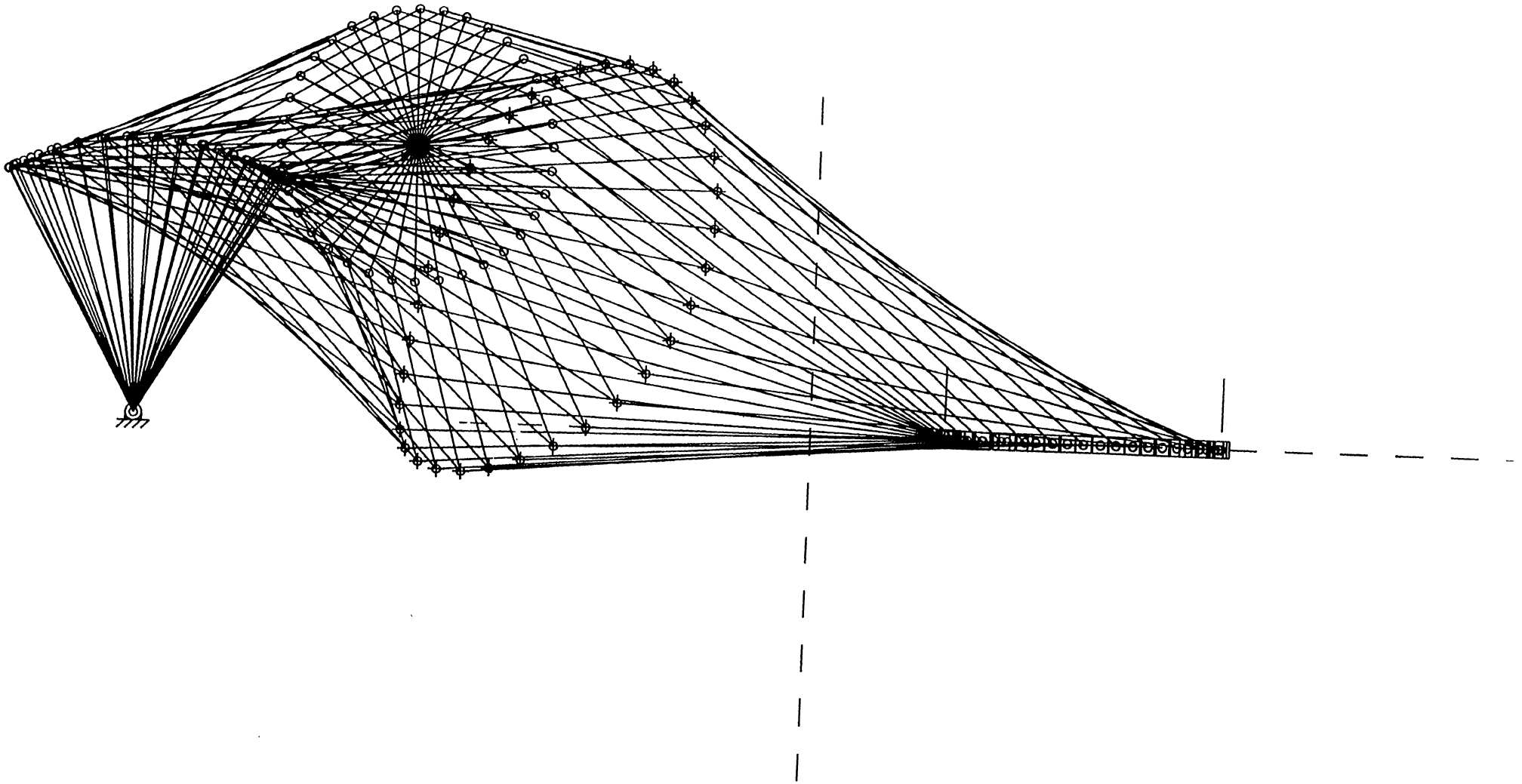

Figure 10 - Slider driven by four-bar chain approximating required motion

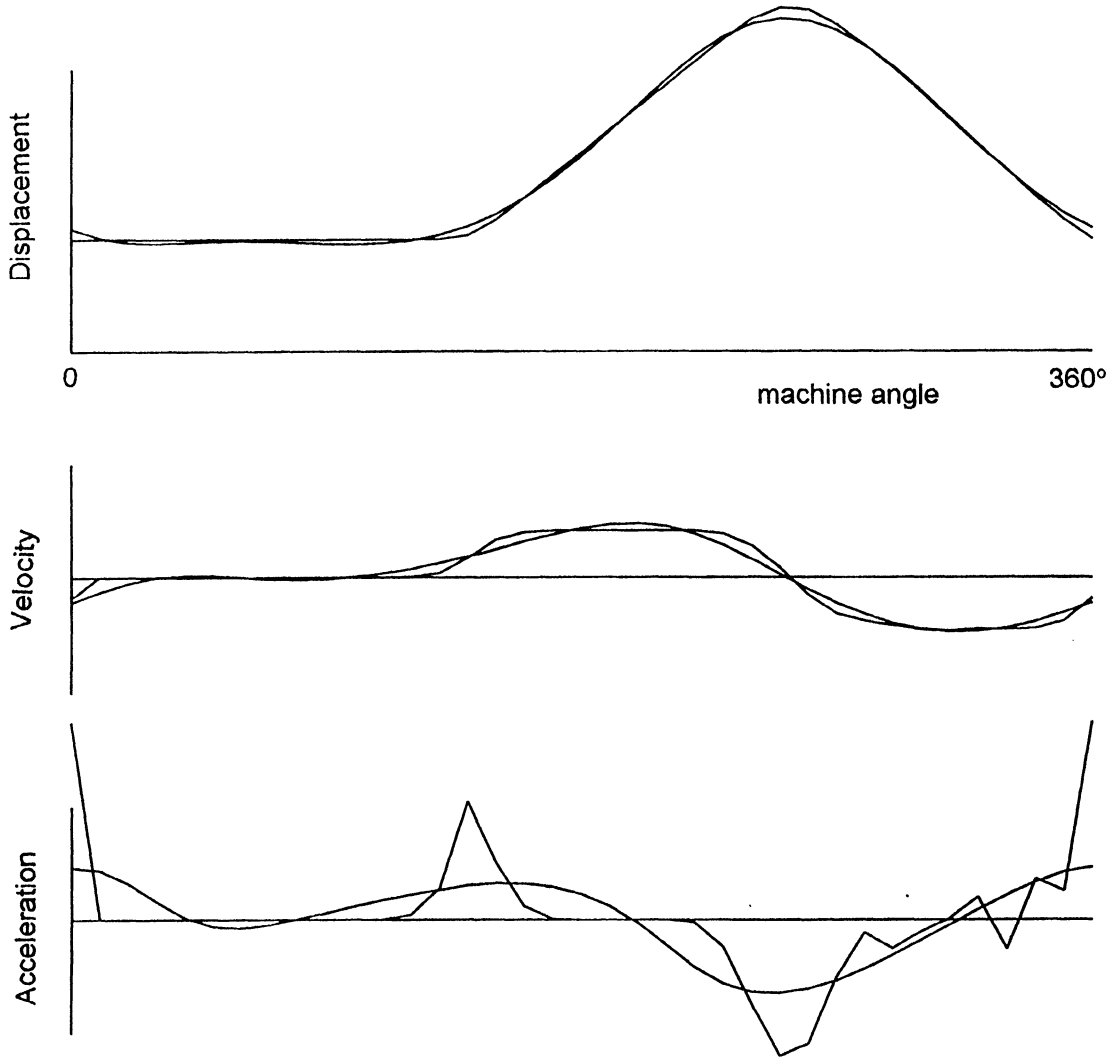

Figure 11 - Comparison of actual motion from four-bar with required motion

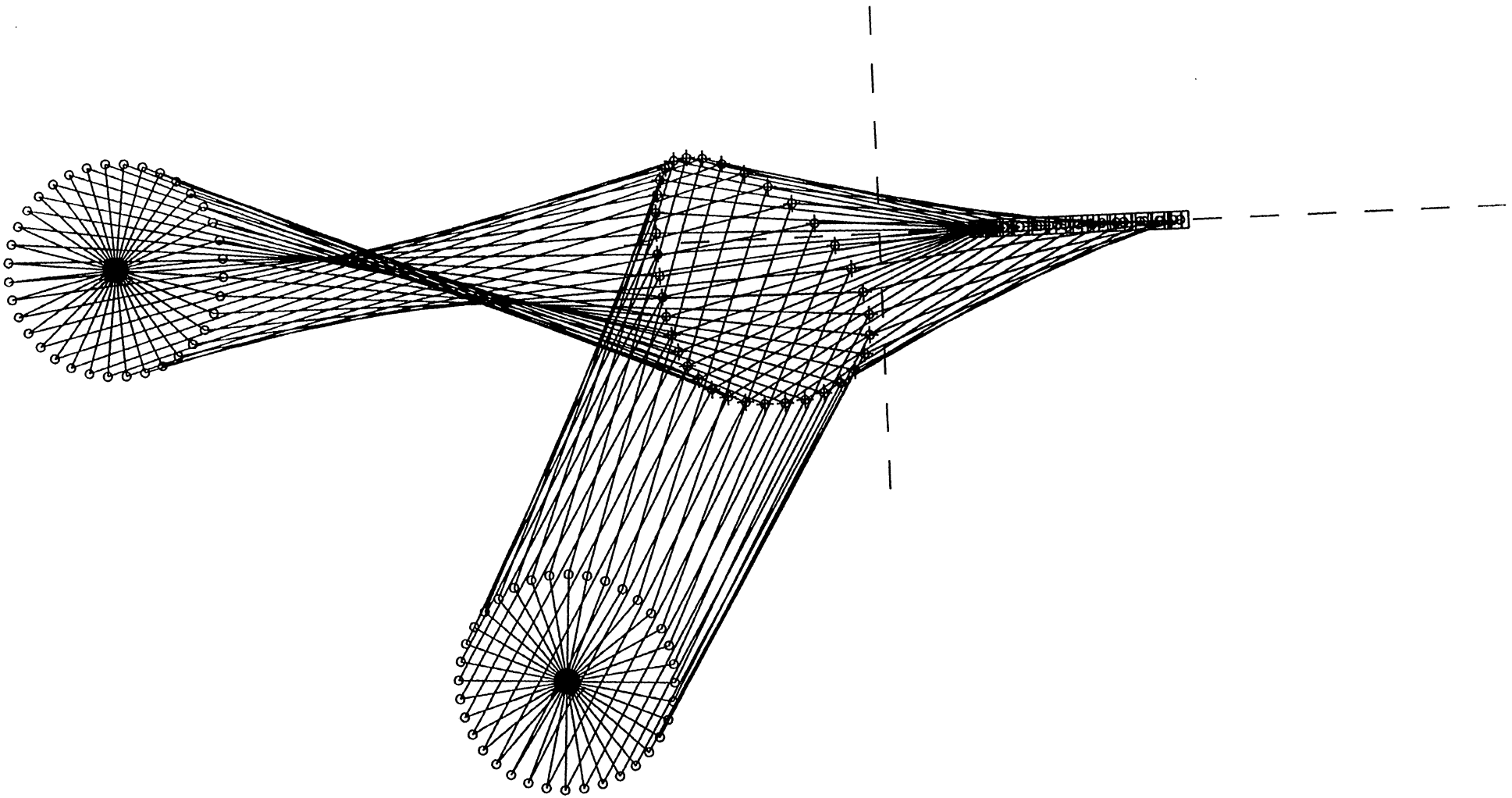

Figure 12 - Geared five-bar chain approximating required motion

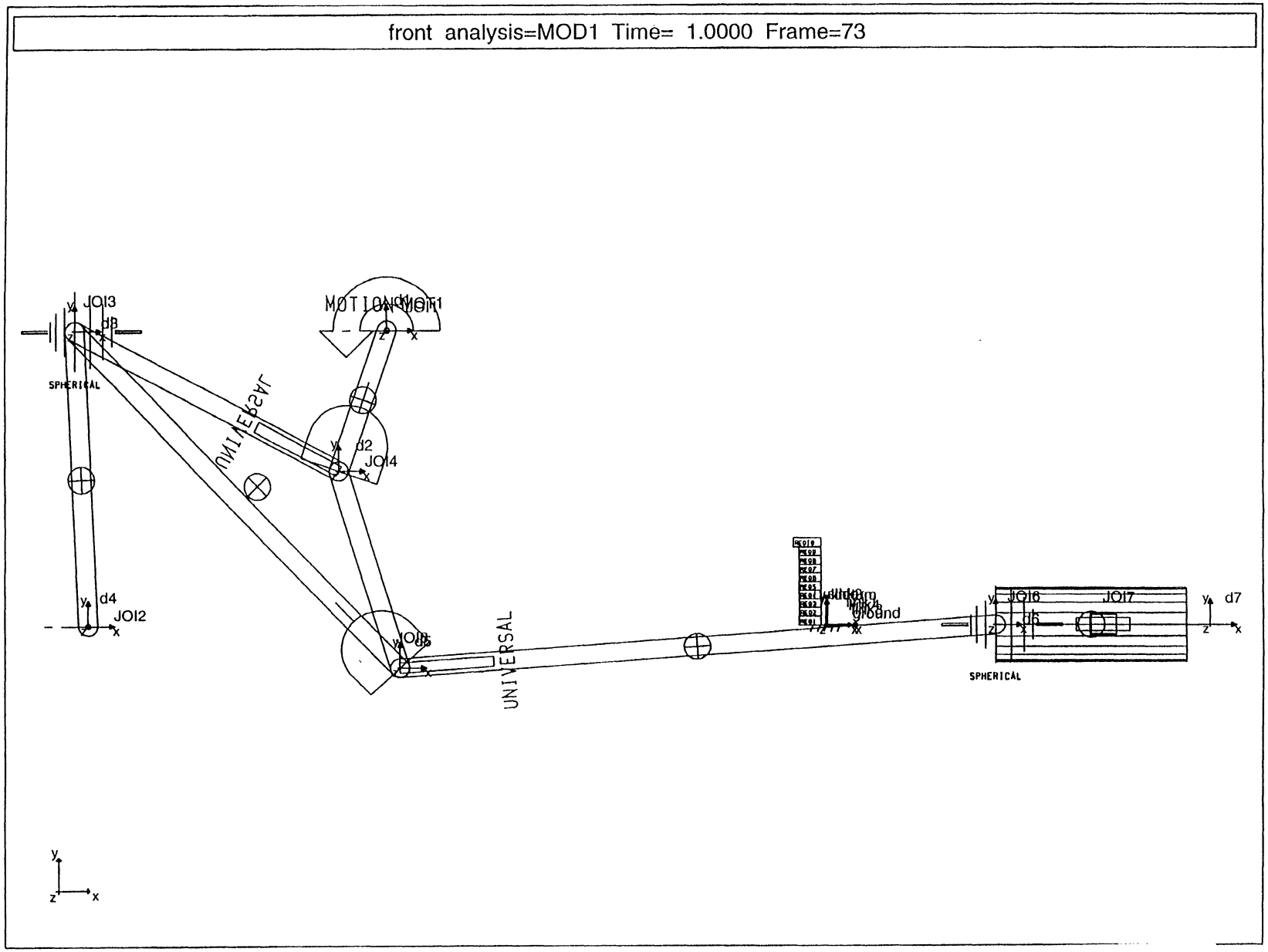

Figure 13 - ADAMS model of the four-bar