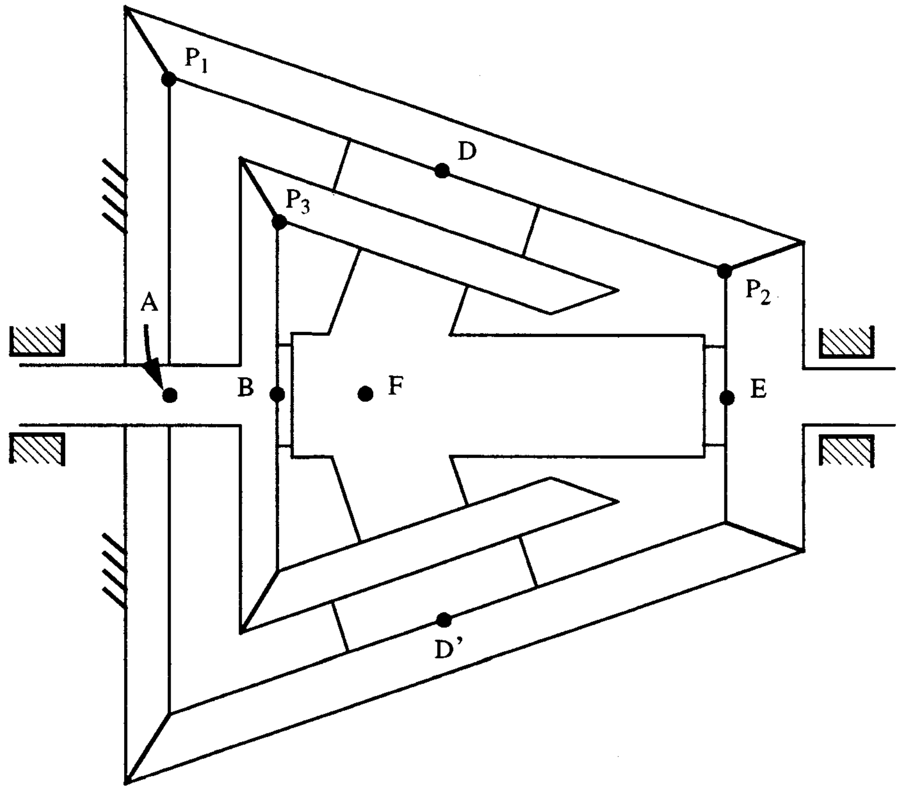
Modeling Gear Trains Using SD/FAST

Symbolic Dynamics, Inc.

SD/FAST Application Note #005

FIGURE 1 Humpage's Reduction Gear



1.0 Introduction

Gear trains are important components of many mechanical systems which must transmit power and motion between rotating shafts. For example, the drive train in an automobile contains a transmission and a differential, both of which are complicated gear trains. Of interest to the engineer who must design such systems is the velocity ratio between the input and output, the mechanical advantage of the gear train, the contact force between the gears, and power flow through the gear train.

FIGURE 1 is a drawing of a gear reducer called *Humpage's reduction gear.* In the literature this device is also called a bevel-gear epicyclic, or planetary gear train. It consists of a driven gear E (the sun gear), moving planet gears D and D', output gear B, and an arm F. Gears D and D' are pinned to F. They are in contact with the fixed gear A which is rigidly mounted to the housing of the gear box. Also, D and D' are multiply geared, maintaining gear contact with B as well as with A and E. In this Application Note we will use SD/FAST to model the gear train of FIGURE 1.

To simulate gear pairs with SD/FAST, the user-constraint facility of SD/FAST will be used. This Note will show how to prepare the SD/FAST description of the mechanism, and how to write the driver program which exercises the model. The remainder of the Note is organized as follows:

- The geometry of the gear train is discussed in Section 2.0.
- The SD/FAST System Description File is described in Section 3.0.
- A model of gear contact between bodies is presented and the appropriate constraints are developed in Section 4.0.
- Velocity Analysis, Mechanical Advantage, and Power Flow Analysis of the gear train are computed in Section 5.0.

All software necessary to analyze this mechanism is developed and presented here. The resulting code is also supplied on the SD/FAST release media. The necessary files are called `humpage.sd` and `humpage.f`.

2.0 The Geometry of the Gear Train

Because of the symmetry of the gear train we do not need to include D' in the remainder of the Note. We will build a model in which all the load is carried by the single planet gear D. This is a typical assumption for such epicyclic gear systems. Load sharing can be assumed to split the computed results based on the symmetry of the system. With multiple planets included in the model the tooth force computation becomes statically indeterminate, so neglecting all but one planet is reasonable and conservative.

The housing of the gear train, body A, is fixed to ground. Bodies B and E are pinned to A at their mass centers. The arm F is supported by bearings fixed in B and E. Body D is pinned to F at D's mass center. In addition to its pin connection to F, body D maintains gear contact with the housing A, and bodies B and E. These contacts take place at the points labelled P_1, P_2, and P_3 in FIGURE 1. The coordinates of the body mass centers and the three contact points are given in TABLE 1. Note that the contact point between the gear pairs does not take place at a fixed point of D, B, A, or E. (If it did, then only one tooth of each gear would be used!) Observe though, that the contact points *are* fixed in the arm F as it rotates. As we will see, Body F plays a central role in the analysis of the gear train because it provides a rotating reference which aids in the analysis of the gear train kinematics.

TABLE 1 Coordinates of Body Mass Centers and Contact Points

Point	**X**	**Y**	**Z**	**Radius**
A	0.00	0.00	0.00	60
B	22.402258	0.00	0.00	30
D	57.628117	40.00	0.00	61, 30
F	43.746005	0.00	0.00	-
E	115.256236	0.00	0.00	20
P_1	0.00	60.00	0.00	-
P_2	22.402258	30.00	0.00	-
P_3	115.256236	20.00	0.00	-

To complete the geometry of the gear train, the axis of gear D must be specified. This axis, fixed in body F, makes an angle of 70.86053 degrees from the X axis, as shown in FIGURE 1. For the purpose of this Note, realistic mass properties for the bodies are not required. If dynamic analyses were to be performed mass and inertia properties would need to be specified by the engineer. Here we will use unit mass and inertia for each body. In the next section we will show how to prepare the SD/FAST System Description File, and then run SD/FAST.

3.0 The SD/FAST System Description File

FIGURE 2 shows the complete System Description File for the gear train. This input file contains the necessary information for SD/FAST to generate equations of motion for the gear train. This includes information describing the bodies and their arrangement into the mechanism, mass properties, joint descriptions, location of hinge points, and the number of user constraints which will be added to the system. The file is organized in paragraphs, with each paragraph describing a new body.

The first paragraph describes body B. The keyword `body` tells SD/FAST that a new body is being added to the system. In this case it is the body B, whose name is given after the `body` keyword. The next keyword, `inb` (which is short for 'inboard') tells SD/FAST the name of B's inboard body. B is connected to the housing, which is considered to be part of the ground frame. SD/FAST reserves the name `$ground` to mean the inertial frame. The keyword `joint` followed by the value `pin` tells SD/FAST that this joint is a pin joint. The next line in the file gives the mass of body B and its central inertia matrix. As mentioned earlier all the bodies will be given unit mass and inertia matrices. The next line in the file contains two keywords `bodytojoint` and `inbtojoint`. These keywords take as values the vector from the mass center of B to the hinge point, and the vector from the ground reference point to the hinge point,

FIGURE 2 System Description File humpage.sd

```
body = B   inb = $ground joint = pin
   mass = 1   inertia = 1 1 1
   bodytojoint = 0 0 0  inbtojoint = 22.402258 0 0
   pin = 1 0 0

body = E inb = $ground joint = pin
   mass = 1   inertia = 1 1 1
   bodytojoint = 0 0 0  inbtojoint =  115.256236 0 0
   pin = 1 0 0

body = F inb = $ground joint = pin
   mass = 1  inertia = 1 1 1
   bodytojoint = 0 0 0  inbtojoint =  43.746005 0 0
   pin = 1 0 0

body = D inb = F joint = pin
   mass = 1  inertia = 1 1 1
   bodytojoint = 0 0 0  inbtojoint = 13.882112  40.0 0
   pin =  0.32786885245902   0.94472324814583 0

constraints = 3
```

respectively. These vectors can be formed from the values shown in TABLE 1, keeping in mind that the hinge point for this (and all the other joints) is located at the mass center of each gear. The ground reference point is at A. The last entry in the paragraph is the keyword `pin`. This is used to define the joint axis, by giving three numbers which define a vector parallel to the joint axis. For body B the joint axis is along *x*.

The next three paragraphs are similar to the first. They are used to introduce bodies E, F, and D. E and F are both connected to ground by pin joints, while D is pinned to F. The `inbtojoint` vector for D is the vector from the mass center of F to the hinge point, which is coincident with the mass center of D.

Note that in the SD/FAST input file no explicit mention of the gears is made. The gears are introduced by way of the user-defined constraint facility. SD/FAST is told only to allow for three constraints to be included in the model. This is done by adding the keyword `constraints` followed by a number, in this case `3`. The gear contact geometry specified in TABLE 1 will be incorporated in the constraints as they are developed in the next section.

After preparing the System Description File, SD/FAST is run to generate the equations of motion for the gear train. SD/FAST is normally run by giving the command:

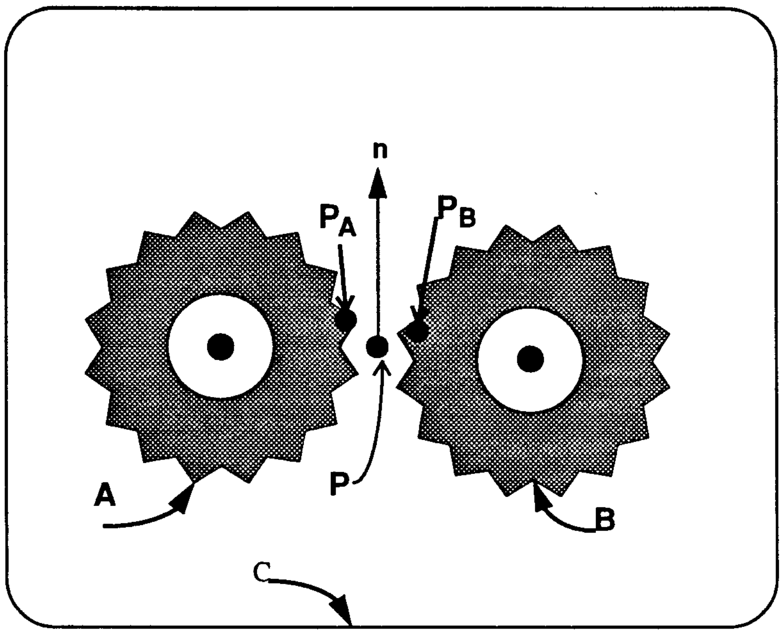
```
sdfast humpage.sd
```

This causes SD/FAST to process the input file we have just discussed, named `humpage.sd`. SD/FAST will generate three output files: `humpage_info`, `humpage_dyn.f`, and `humpage_sar.f`. `humpage_info` contains annotation and descriptive information about the gear train, especially regarding the choice of coordinates for the problem. Here is the file:

```
SD/FAST Information File: humpage.sd
Generated 23-Aug-1990 16:31:05 by SD/Fast, Kane's formulation
(sdfast BX.1.6 #10617) on machine ID 1700d49f

ROADMAP (humpage.sd)

Bodies        Inb
No  Name      body Joint type  Coords q         Multipliers
--- --------- ---- ----------- ---------------- -----------------------
  0 $ground                                    |
  1 b           0  Pin(1D)        1            |
  2 e           0  Pin(1D)        2            |
  3 f           0  Pin(1D)        3            |
  4 d           3  Pin(1D)        4            |

User Constraints

  1 USER_1                                     |  1
  2 USER_2                                     |  2
  3 USER_3                                     |  3

STATE INDEX TO JOINT/AXIS MAP (humpage.sd)

 Index
  q|u    Joint  Axis   Joint type    Axis type
 -----   -----  ----   -----------   ----------
  1|5      1      1    Pin(1D)       rotate
  2|6      2      1    Pin(1D)       rotate
  3|7      3      1    Pin(1D)       rotate
  4|8      4      1    Pin(1D)       rotate

SYSTEM PARAMETERS (humpage.sd)

Parameter  Value  Description

nbod          4   no. bodies (also, no. of tree joints)
njnt          4   total number of joints (tree+loop)
ndof          4   no. degrees of freedom allowed by tree joints
nloop         0   no. loop joints
nldof         0   no. degrees of freedom allowed by loop joints

nq            4   no. position coordinates in state (tree joints)
nu            4   no. rate coordinates in state (tree joints)
nlq           0   no. position coordinates describing loop joints
nlu           0   no. rate coordinates describing loop joints
nc            3   total no. constraints defined
nlc           0   no. loop joint constraints
npresc        0   no. prescribed motion constraints
nuserc        3   no. user constraints
```

The 'roadmap' describes the bodies, their ordering and connections, the coordinates assigned to each joint, and so on.

FIGURE 3 A Gear Pair



The file `humpage_dyn.f` contains the equations of motion for the gear train. The file `humpage_sar.f` contains analysis routines which can be used for many purposes. We will use the Velocity Analysis routine `SDINITVEL()` in a later section. In the next section we will look at a model for a pair of gears and show how to develop the necessary constraint equations.

4.0 A Model of a Gear Pair

FIGURE 3 shows two gears in contact. The gears are in rolling contact with each other, which means that the points in contact have the same velocity in the direction of the common tooth surface normal vector. This model of rolling contact does not preclude slipping of the teeth in the tooth tangential direction. In fact, if the tooth shape is not correct slipping must occur as the gears rotate. The model enforces the condition that the teeth do not penetrate each other. While the condition of rolling contact can be expressed in any reference frame, it is always true that a particular reference frame is available which simplifies the description of the rolling motion. This frame is called the *case*. For the gear train, body F, the arm, plays the role of the case for each of the gear pairs. The case frame simplifies the gear description because the contact point of the gears is fixed in the case frame, and the tooth surface normal is fixed in the case as well. In simple gear systems the case frame is obvious, because both gears are directly pinned to the case. This is not true in epicyclic gear trains, because the planet gears are carried on moving shafts. In FIGURE 3 the contact point is called P. The point of gear A which is instantaneously coincident with P is called P_A, and the point of gear B coincident with P is called P_B. The unit vector normal to the tooth surface at P is called **n**. The velocity constraint imposed by the gear may be expressed:

$$(v^A - v^B) \bullet n = 0 \qquad \text{(EQ 1)}$$

Where v^A is the linear velocity of P_A in inertial space and v^B is the linear velocity of P_B in inertial space. In our SD/FAST model of the gear, the bodies are assumed to be connected to the rest of the system by previously defined joints. The gear constraint itself does not constitute a joint. The bodies must be held together properly by the other joints in the system. Associated with the velocity constraint equation is an acceleration constraint. This equation is formed by differentiating the velocity constraint. The acceleration constraint must also be satisfied by the model during motion of the system. Upon differentiating the velocity constraint, we obtain the equation:

$$(a^A - a^B) \bullet n + (v^A - v^B) \bullet (\omega^C \times n) = 0 \qquad \text{(EQ 2)}$$

a^A is the inertial linear acceleration of P_A, a^B is the inertial linear acceleration of P_B, and ω^C is the inertial angular velocity of the case frame. This result is obtained by using the chain rule of differential calculus, and the rule for forming the inertial derivative of a vector fixed in a moving frame. The tooth surface normal vector is fixed in the case, and its derivative is found by forming the vector cross product with the case angular velocity vector and the surface normal. It is convenient to use a vector identity which allows the above expression to be rearranged:

$$(a^A - a^B + (v^A - v^B) \times \omega^C) \bullet n = 0 \qquad \text{(EQ 3)}$$

In addition to the velocity and acceleration constraints, a force model of the gear contact must be developed. The model is a meshing force applied at the contact point of each gear. The force vector applied to the contact point of gear A is $f^A = \lambda n$, where λ is the magnitude of the force, and $f^B = -\lambda n$ applied to the contact point of gear B. λ is not known as a function of the system coordinates or velocities. It is one of the unknowns, along with the system accelerations, whose value will be computed by SD/FAST.

4.1 Implementing the Gear Constraint

To implement the gear constraint in SD/FAST, the user-constraint facility must be used. This requires the user to program four subroutines, one to calculate velocity constraint errors, one to calculate acceleration constraint errors, one for the constraint force model, and a position constraint error routine. For our application, a position constraint was not included in the gear model, so the position constraint routine will return zero error. The four subroutines must be written according to certain conventions documented completely in the SD/FAST User's Manual. In this Note we will primarily illustrate how to write these routines for this specific case, rather than giving a general recipe. First we'll develop a few utility subroutines to be re-used for each gear by the SD/FAST-required constraint routines. We will begin with the velocity constraint routine.

4.1.1 Velocity Constraint Error Routine

```
                                              subroutine gearverr(case,A,B,fp,n,verr)
                                              integer GROUND,case,A,B,i
                                              parameter (GROUND=0)
                                              real*8 fp(3),n(3),verr
                                              real*8 pa(3),pb(3)
                                              real*8 va(3),vb(3),vab(3)

get coordinates of contact pt. on A and B     call getcoord(case,A,B,fp,pa,pb)

get inertial velocities of these points       call SDVEL(A,pa,va)
                                              call SDVEL(B,pb,vb)

compute relative velocity                     do 10 i = 1,3
                                       10        vab(i) = va(i) - vb(i)

transform to case frame                       call SDTRANS(GROUND,vab,case,vab)

dot with normal to get error                  verr = vab(1)*n(1) + vab(2)*n(2) + vab(3)*n(3)

                                              return
                                              end
```

This subroutine takes as input the body numbers of a case body and two gears, the vector from the case mass center to the contact point, and the tooth surface normal (expressed in the case frame), and returns the velocity constraint error shown in (EQ 1). The subroutine makes use of several SD/FAST-generated subroutines which compute the global position and velocity of points fixed in bodies, and a subroutine to transform vectors from one frame to another. (All subroutines whose name begins with SD are generated by SD/FAST. We also follow the convention of writing the SD/FAST-generated routine names in uppercase.) To compute the velocity constraint error, (EQ 1), the subroutine computes the velocity of the point of each gear coincident with the contact point. These velocities are then subtracted, and the difference is dot-multiplied with the tooth normal vector. The result of the dot product is returned as the velocity constraint error.

Subroutine `getcoord()` is called by several other routines, so it is not coded in-line with the velocity subroutine. Its purpose is to return the vector from the mass center of gear A to the contact point, expressed in the gear frame, and to return the vector from the mass center of gear B to the contact point, expressed in the gear B frame. These are the two vectors `pa` and `pb`. The coordinates of `pa` and `pb` are constantly changing in the gear-fixed frames as the gears spin.

`SDVEL()` is used to compute the velocity of the gear contact points. These velocities are expressed in the global frame, so after subtracting them, the velocity difference is transformed into the case frame by `SDTRANS()`. The tooth normal (given in the case frame) is then dot-multiplied with the difference vector to form the velocity constraint error.

Before we study the acceleration constraint routine, we will discuss the subroutine `getcoord()`.

```
      subroutine getcoord(case,A,B,fp,pa,pb)
      integer GROUND,case,A,B,i
      parameter (GROUND=0)
      real*8 fp(3),com(3),ra(3),rb(3),pa(3),pb(3)
      real*8 phat(3)
```

get contact point location

```
      call SDPOS(case,fp,phat)
```

get coordinates of gears' c.m.

```
      call SDPOS(A,com,ra)
      call SDPOS(B,com,rb)
```

vectors from gear cm's to contact

```
      do 10 i = 1,3
         pa(i) = phat(i) - ra(i)
         pb(i) = phat(i) - rb(i)
10    continue
```

transform to gear frame

```
      call SDTRANS(GROUND,pa,A,pa)
      call SDTRANS(GROUND,pb,B,pb)

      return
      end
```

This subroutine returns the coordinates of the point of each gear that is coincident with the contact point. The coordinates are returned expressed as the vector from the gear mass center to the contact point, expressed in the rotating gear frame. Note that this is a vector that is constantly changing in the gear frame, even though it is a constant vector in the case frame. SDPOS() is first used to compute the global position of the contact point. (All vectors returned by SDPOS() are global coordinates expressed in the global frame.) Next, the coordinates of the gear's mass centers are computed. Then the vector from the mass center of each gear to the contact point is found by subtracting their coordinates. Finally, SDTRANS is used to bring each vector into its own body-fixed frame.

4.1.2 Acceleration Constraint Error Routine

The acceleration constraint error is computed by subroutine `gearaerr`, shown below.

```
      subroutine gearaerr(case,A,B,fp,n,aerr)
      integer GROUND,case,A,B,i
      parameter (GROUND=0)
      real*8 fp(3),n(3),aerr
      real*8 pa(3),pb(3)
      real*8 va(3),vb(3),vab(3)
      real*8 aa(3),ab(3),aab(3)
      real*8 wc(3),vec(3)
```

```
get coordinates of contact pt on A, B     call getcoord(case,A,B,fp,pa,pb)

get inertial velocities of A and B        call SDVEL(A,pa,va)
                                          call SDVEL(B,pb,vb)

get their inertial accelerations          call SDACC(A,pa,aa)
                                          call SDACC(B,pb,ab)

                                          do 10 i = 1,3
                                              vab(i) = va(i) - vb(i)
                                              aab(i) = aa(i) - ab(i)
                                    10    continue

                                          call SDTRANS(GROUND,aab,case,aab)
                                          call SDTRANS(GROUND,vab,case,vab)

                                          call SDANGVEL(case,wc)

                                          vec(1) = aab(1) + vab(2)*wc(3) - vab(3)*wc(2)
                                          vec(2) = aab(2) + vab(3)*wc(1) - vab(1)*wc(3)
                                          vec(3) = aab(3) + vab(1)*wc(2) - vab(2)*wc(1)

                                          aerr = vec(1)*n(1) + vec(2)*n(2) + vec(3)*n(3)

                                          return
                                          end
```

This subroutine implements the acceleration constraint, (EQ 3). It is very much like the velocity constraint routine, except that it additionally computes a term involving the acceleration of the gear contact points, and a term involving the case angular velocity vector. Subroutine `SDACC()` is used to compute inertial acceleration of points (returned expressed in the global frame), while `SDANGVEL()` is used to compute the angular velocity of the case (returned in the case frame.)

4.1.3 Gear Tooth Force Routine

Next, we will look at a routine `gearfrc()` to calculate the constraint force.

```
subroutine gearfrc(case,A,B,fp,n,mult,mesh)
integer GROUND,case,A,B,i
parameter (GROUND=0)
real*8 fp(3),n(3),mult
real*8 pa(3),pb(3)
real*8 mesh(3),mesha(3),meshb(3)
```

```
get coordinates of contact points A, B         call getcoord(case,A,B,fp,pa,pb)

calculate meshing force                        do 10 i = 1,3
                                        10        mesh(i) = mult*n(i)

transform into gear A frame                    call SDTRANS(case,mesh,A,mesha)

apply at tooth contact point                   call SDPOINTF(A,pa,mesha)

Newton's Third Law                             do 20 i = 1,3
                                        20        mesha(i) = -mesha(i)

transform into gear B frame                    call SDTRANS(A,mesha,B,meshb)

apply at tooth contact point                   call SDPOINTF(B,pb,meshb)

                                               return
                                               end
```

`gearfrc()` causes a force to be applied to the contacting teeth of a gear pair. The force is computed using the passed in Lagrange Multiplier `mult` (the magnitude of the force) and the tooth normal vector `n`, which defines the direction of the meshing force. First `getcoord()` is used to locate the gear contact point in the gear frames. The tooth force is formed by multiplying the tooth normal vector by the multiplier. Note that this force is returned as the output parameter `mesh`. At this point the force will be expressed in the case frame, since the tooth normal vector is fixed in the case frame. `SDTRANS()` is used to transform the force from the case to the first gear. `SDPOINTF()` is used to apply the force to the contact point of the first gear. The force is then negated in accordance with Newton's Third Law, transformed into the second gear frame, and then applied to the second gear at its contact point.

This subroutine is typical of user-constraint force routines. In general, these routines must be written in terms of unknown multipliers. SD/FAST uses the constraint force routine, the acceleration constraint error routine and the generated equations of motion to solve for all the accelerations in the system, as well as for all the multiplier values.

The velocity constraint routine, the acceleration constraint routine, and the constraint force routine we have just presented form a generic set of routines for any gear pair. That is, they are not written specifically for the gear train we have been discussing. To use these routines we must write a problem-specific velocity error routine, acceleration error routine, and a constraint force routine. These routines will simply make calls to the low-level generic routines we have just seen. Since there are three gear contacts in the problem, each routine will make three calls to the low-level routines.

compute velocity constraint errors

```
subroutine sduverr(t,q,u,verr)
integer GROUND,B,E,F,D
parameter (GROUND=0,B=1,E=2,F=3,D=4)
real*8 t,q(*),u(*),verr(*),n(3)
real*8fp1(3),fp2(3),fp3(3)
common /vectors/ n,fp1,fp2,fp3
```

one call for each gear pair

```
call gearverr(F,GROUND,D, fp1,n,verr(1))
call gearverr(F,     E,D, fp2,n,verr(2))
call gearverr(F,     B,D, fp3,n,verr(3))

return
end
```

compute accel. constraint errors

```
subroutine sduaerr(t,q,u,udot,aerr)
integer GROUND,B,E,F,D
parameter (GROUND=0,B=1,E=2,F=3,D=4)
real*8 t,q(*),u(*),udot(*),aerr(*),n(3)
real*8fp1(3),fp2(3),fp3(3)
common /vectors/ n,fp1,fp2,fp3
```

one call for each gear pair

```
call gearaerr(F,GROUND,D, fp1,n,aerr(1))
call gearaerr(F,     E,D, fp2,n,aerr(2))
call gearaerr(F,     B,D, fp3,n,aerr(3))

return
end
```

compute constraint forces

```
subroutine sduconsfrc(t,q,u,mult)
integer GROUND,B,E,F,D
parameter (GROUND=0,B=1,E=2,F=3,D=4)
real*8 t,q(*),u(*),mult(*)
real*8 n(3),fp1(3),fp2(3),fp3(3)
real*8 frc1(3),frc2(3),frc3(3)
common /vectors/ n,fp1,fp2,fp3
common /mesh/ frc1,frc2,frc3
```

one call for each gear pair

```
call gearfrc(F,GROUND,D, fp1,n,mult(1),frc1)
call gearfrc(F,     E,D, fp2,n,mult(2),frc2)
call gearfrc(F,     B,D, fp3,n,mult(3),frc3)

return
end
```

The names `sduverr()`, `sduaerr()`, and `sduconsfrc()` for the problem-specific routines are required by SD/FAST. SD/FAST will make calls to these routines as needed. The names and argument list must be exactly as shown.

In the above routines the vectors `fp1`, `fp2`, `fp3` are the vectors from the mass center of the arm F to the contact point of each gear pair, expressed in the F frame. This data can be formed by reference to TABLE 1. In the program, the vectors are stored in the named `common` block `vectors`. They are initialized by the following `BLOCK DATA` subprogram:

```
block data
real*8 n(3),fp1(3),fp2(3),fp3(3)
common /vectors/ n,fp1,fp2,fp3
data n  /0d0, 0d0, 1d0/
data fp1/-43.746005d0, 60d0,   0d0/
data fp2/ 71.510230d0, 20.0d0, 0d0/
data fp3/-21.343747d0, 30.0d0, 0d0/
end
```

4.1.4 Position Constraint Error Routine

Finally, we must code a position constraint subroutine which simply sets all the position errors to zero. In other problems the position constraints might be nonzero, but for this problem we can treat the gears as velocity constraints only. Again, this routine must be called `sduperr()` and must have the arguments shown.

```
subroutine sduperr(t,q,perr)
real*8 t,q(*),perr(*)
perr(1) = 0d0
perr(2) = 0d0
perr(3) = 0d0
return
end
```

5.0 Using the SD/FAST Model

Having now written the gear constraint subroutines, the gear contacts are 'built in' to the SD/FAST model. The routines will be exercised by other SD/FAST generated routines as required. In this section we will analyze the gear train by performing a complete velocity analysis, torque transmission, and power flow through the gear train. This section is organized into three Exercises which make use of SD/FAST routines, along with the user-written main program. The program begins with a variable declaration section.

```
program humpage
integer NQ,NU,NEQ,NC,GROUND,B,E,F,D
parameter (NQ=4,NU=4,NEQ=NQ+NU,NC=3)
parameter (GROUND=0,B=1,E=2,F=3,D=4)
real*8 y(NEQ),dy(NEQ),t,ctol,torque,vel(3)
real*8 pa(3),pb(3),power
real*8 frc1(3),frc2(3),frc3(3),n(3)
real*8 fp1(3),fp2(3),fp3(3)
integer lock(NU),fcnt,err,maxeval,i,SDINDX
common /vectors/ n,fp1,fp2,fp3
common /mesh/ frc1,frc2,frc3
common /load/ torque
```

5.1 Exercise 1: Velocity Analysis

The user program must always make a call to SDINIT before any other SD/FAST routine can be called. We also set time to 0 here.

```
call SDINIT
t = 0d0
```

We begin by setting the angular velocity of shaft B to one rad/sec. We will then do a Velocity Analysis to find the compatible angular velocity of all the other bodies. Velocity Analysis is performed by the routine SDINITVEL(). This routine takes the user's passed-in initial guess at the velocities and finds a set of velocities which satisfy all velocity constraints. In this case, we have three velocity constraints, one for each gear contact.

The lock parameter is used here to prevent the angular velocity of B from being altered during the velocity analysis. Ctol indicates the required accuracy with which we wish to solve the velocity constraint equations. The solution makes use of a built-in root-finder, and the parameter maxeval limits the number of velocity error evaluations the routine is allowed to make. Velocity errors are computed by the sduverr() subroutine, which in turn calls gearverr() as shown above.

Y is an array containing the elements of the system state vector. For this problem, the first four elements are rotation angles of the bodies, while the last four elements are rotation rates. Elements of the state array are accessed by using the integer function SDINDX(). This function takes as input the index of a joint and a joint axis within the joint. It returns the index of the appropriate element of the state vector. Note that to access velocities in the state vector we add NQ (the number of rotation angle states) to the value from SDINDX(), to skip over the angles.

set B to one rad/sec and lock it

```
y(NQ+SDINDX(B,1)) = 1d0
```

set velocity analysis parameters

```
lock(SDINDX(B,1)) = 1
ctol = 1d-10
maxeval = 500
```

run velocity analysis

```
      call SDINITVEL(t,y,lock,ctol,maxeval,fcnt,err)

      write(6,*)   'BODY     ANG. VEL.'
      write(6,*)   ''
      write(6,100) '  B   ',  y(NQ+SDINDX(B,1))
      write(6,100) '  F   ',  y(NQ+SDINDX(F,1))
      write(6,100) '  E   ',  y(NQ+SDINDX(E,1))
      write(6,*)   ''
      write(6,*)   '  Planet rate relative to arm'
      write(6,100) '  D   ',  y(NQ+SDINDX(D,1))
100   format(a6, f14.8)
```

Running the above program to this point produces the following output:

```
BODY     ANG. VEL.

  B        1.00000000
  F       60.99999017
  E      243.99996162

Planet rate relative to arm

  D      -59.99999048
```

From these results we see that the overall velocity reduction is 244:1. Gear B is turning in the same direction as E, since both have the same sign on their angular velocity. Body F is turning one fourth as fast as gear E, also in the same direction. The planet gear D is turning relative to the arm F at -60.0 rad/sec. This means that its angular velocity relative to the arm is anti-parallel to the pin axis vector which defines its connection to F.

5.2 Exercise 2: Mechanical Advantage

Another important calculation for the gear train is Mechanical Advantage. It is well-known that mechanical advantage between the input and output of a gear train is the inverse of the velocity ratio. Thus, if we apply a unit torque to B, a torque of magnitude 1/244 applied to E should prevent any acceleration of the gears. This test is performed by the following code:

```
compute holding torque          torque = - y(NQ+SDINDX(B,1)) / y(NQ+SDINDX(E,1))

register state vector           call SDSTATE(t,y,y(NQ+1))
apply torques                   call sduforce(t,y,y(NQ+1))
compute accelerations           call SDDERIV(dy,dy(NQ+1))

                                write(6,*) ''
                                write(6,*) 'STATIC TEST'
                                write(6,*) ''
                                write(6,*) '   GEAR ANGULAR ACCELERATIONS:'
                                write(6,*) ''
                                write(6,101) (dy(NQ+i),i=1,NU)
                          101   format (3x,e12.5)
```

The variable `torque` is computed as the inverse ratio between input and output speeds. It is passed in common to the routine `sduforce()`. This subroutine is used to encode all forces and torques acting on any of the bodies (except for the gear constraint forces). The code for `sduforce()` will be shown after the main program. Before calling `sduforce()` the routine `SDSTATE()` must be called to provide the current value of the system state vector. After calling `sduforce()`, a call is made to `SDDERIV()`.

This causes the acceleration of all the gears to be computed. Running this piece of code produces the following output:

```
STATIC TEST

GEAR ANGULAR ACCELERATIONS:

-0.22204E-15
 0.24026E-15
-0.70557E-18
-0.27732E-15
```

We see that the gears have zero acceleration, within machine precision, as they should. We will next analyze the power flow through the gear train.

5.3 Exercise 3: Power Flow

The power flow is computed by summing the gear tooth force multiplied by the tooth velocity. The input power is one unit, since the input torque and shaft velocity were set to 1.0. Also, we know that no power flows to the ground, since it is not moving. Thus we can exclude the ground contact with gear D. The tooth force is computed by SD/FAST, and makes use of the gear constraint force routine, `sduconsfrc()`. This routine is called behind the scenes by `SDDERIV()`, which also calls the acceleration constraint error routine. We now show the code for the Power Flow analysis.

```
      write(6,*) ''
      write(6,*) 'POWER FLOW'
      write(6,*) ''
```

We will start with the gear between B and D. `getcoord()` is used to compute the coordinates of the gear contact point in the gear frames. Then `SDVEL()` is used to compute the velocity of one of these points. (Their velocities are the same to within constraint tolerance.) The contact force (which was placed in `common` variable `frc3` by `sduconsfrc()`, (see page 12) is transformed into the ground frame by `SDTRANS()`. Since only the *z* component of the velocity and the force are nonzero, we only print this component of each vector.

```
      call getcoord(F,B,D,fp3,pa,pb)
      call SDVEL(D,pb,vel)
      call SDTRANS(F,frc3,GROUND,frc3)
      power = -vel(3)*frc3(3)
      write(6,102) 'gear pair','gear velocity',
     1      'tooth force','power'
      write(6,*)   ''

      write(6,103) 'B_D',vel(3),-frc3(3),power
      write(6,*)   ''
```

We next compute the power flowing from D to E, by repeating a similar set of calls.

get velocity of E-D contact pt.

transform into case frame

compute power

```
      call getcoord(F,E,D,fp2,pa,pb)
      call SDVEL(D,pb,vel)
      call SDTRANS(F,frc2,GROUND,frc2)

      power = vel(3)*frc2(3)
      write(6,103) 'E_D',vel(3),frc2(3),power
```

This code produces the following output:

```
   gear pair       gear velocity       tooth force         power

    B_D               30.00000         0.33333E-01       1.00000

    E_D             4879.99923         0.20492E-03       1.00000
```

We see that power is conserved through the gear train.

```
 102  format(3x,a9,a20,a20,a20)
 103  format(a9,10x,f12.5,10x,e12.5,10x,f12.5)

      call SDPRINTERR(6)
      end
```

We end the program with a call to SDPRINTERR which reports any usage errors in calling the SD/FAST routines by writing a message to the indicated file number.

The subroutine sduforce() for this problem is the following:

Apply a unit torque to B

Apply passed-in torque to E

```
      subroutine sduforce(t,q,u)
      integer GROUND,B,E,F,D
      parameter (GROUND=0,B=1,E=2,F=3,D=4)
      real*8 t,q(*),u(*),torque
      common /load/ torque

      call SDHINGET(B,1,1d0)
      call SDHINGET(E,1,torque)
      return
```

Two calls are made to SDHINGET(), first to apply a unit torque to B, then to apply the torque computed above to E. Sduforce() is written by the user for each problem.

6.0 Summary

In this Application Note we have seen how to model a complicated gear reducer. The user-constraint feature of SD/FAST was used to add the gear connections to the system model. After developing a gear model, a study of the velocity and torque transmission characteristics of the gear reducer was performed using SD/FAST analysis routines. Using the general gear subroutines developed in this Note, almost any gear train can be modeled. The gear train can also be included as part of the overall model of a larger multibody system, in which full dynamic analysis could be modelled.

SD/FAST PURCHASING INFORMATION

U.S.A. and Canada:

Symbolic Dynamics, Inc.
561 Bush Street
Mountain View, CA 94041 U.S.A.
tel/fax: +1 650 960 1532 / 0338
info@symdyn.com
www.symdyn.com

Europe:

Rapid Data Ltd.
Amelia House, Crescent Road
Worthing, West Sussex
England, U.K. BN11 1RL
tel/fax: +44 1903 821266 / 820762
ali@radata.demon.co.uk
www.radata.demon.co.uk

Elsewhere:

Symbolic Dynamics, Inc.
561 Bush Street
Mountain View, CA 94041 U.S.A.
tel/fax: +1 650 960 1532 / 0338
info@symdyn.com
www.symdyn.com