AUTOMATIC IMPLEMENTATION OF THE KINEMATIC CONSTRAINTS OF A ROLLING DISK IN *ADAMS* USING *MATHEMATICA*

Mathias Lidberg[*]

[*]Chalmers University of Technology
Department of Mechanics, SE-412 96, Göteborg, Sweden
e-mail: mlidb@mec.chalmers.se, web page: http://www.mec.chalmers.se/~lidberg

1   INTRODUCTION

Imposing constraints on multibody systems (MBS) restricts the configuration and motion of mechanical systems. These constraints are physically recognizable like revolute joints or geometric in nature such as the inplane joint primitive. A number of these displacement constraints and other constraints are available in MBS-software such as ADAMS. The user of ADAMS also has the opportunity to program a user-defined constraint through a user-written subroutine. This procedure requires the programming of a number of partial derivatives. Users of ADAMS have found this to be a tedious and error prone method. By using symbolic mathematics software we have defined a couple of kinematic constraints and automatically generated and implemented the necessary subroutines to be used by ADAMS. In this paper we outline the details of the procedure applied to the modeling and simulation of a rolling disk. The procedure is simple and efficient.

2   MATHEMATICAL MODEL

The procedure is very well illustrated by the simple example of a rolling disk. Let us assume that the disk is rolling without slip on a horizontal plane. The disk has no other

constraints associated with it. It is free to tilt and roll in any arbitrary direction. The rolling disk is illustrated in Figure 1.
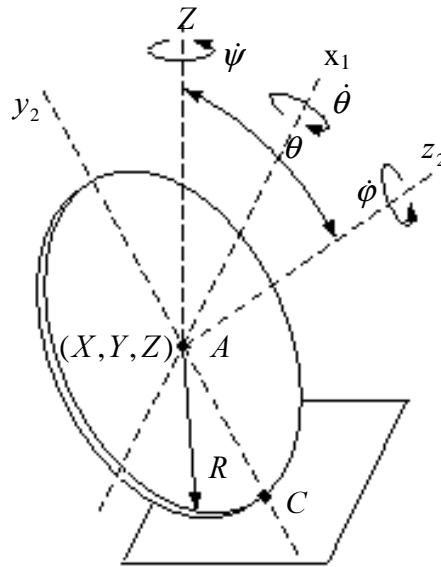
Figure 1: Mathematical model of a rolling disk.

The configuration of a single rigid body is defined using six generalized coordinates. Three Cartesian coordinates (X, Y, Z) define the location and three Euler Angles (ψ, θ, φ) define the orientation. ADAMS is using body-fixed Z-x_1-z_2 Euler angles to define the orientation of a rigid body, see Figure 1.

The requirement of the disk rolling on the horizontal plane without slip means that the point of contact of the disk has zero velocity. We call the angular velocity of the disk $\boldsymbol{\omega}$ and the vector of the point of contact relative to the center of mass of the disk $\boldsymbol{r}_{C/A}$. Then the no-penetration, no-slip condition can be formulated as three constraint equations:

$$\boldsymbol{V_C} = \boldsymbol{V_A} + \boldsymbol{\omega} \times \boldsymbol{r}_{C/A} =$$
$$(\dot{X} + R\cos\psi(\dot{\varphi} + \dot{\psi}\cos\theta) - R\dot{\theta}\sin\psi\sin\theta,$$
$$\dot{Y} + R\sin\psi(\dot{\varphi} + \dot{\psi}\cos\theta) - R\dot{\theta}\cos\psi\sin\theta,$$
$$\dot{Z} - R\dot{\theta}\cos\theta) = (0,0,0)$$

The resulting constraint equations include generalized velocities and are not integrable. Thus this constraint is non-holonomic. There is no standard constraint in ADAMS available to model this type of motion constraint. Instead this constraint is realized as a custom constraint using the residues (error relative to zero) of the constraint equations as equation expressions.

3 ADAMS MODEL

The standard constraint statements in ADAMS are sufficient for the definition of commonly occurring displacement constraints between rigid bodies. For displacement constraint not obtainable through combinations of standard constraints or for constraints involving generalized velocity variables the user needs to implement a user defined constraint (UCON).

For each system constraint, ADAMS/Solver formulates a governing constraint equation in implicit, Jacobian form. In order to proceed smoothly during the simulation process a number of partial derivatives of the constraint equation expressions are needed, see *Using ADAMS/Solver 9.1* and *Using ADAMS/Solver Subroutines 9.1*. For standard ADAMS/Solver constraint statements, symbolic partial derivatives of the constraint equation with respect to the system variables have been pre-computed and stored in the program for use in the system Jacobian when the particular constraint is involved. A user-written subroutine (UCOSUB) computes the value of the constraint expression and its derivatives for the UCON statement during simulation.

The generalized coordinates and velocities in the constraint equations are measured at the origin of the principal axes system with respect to the global coordinate system. The principal axes of a part are the axes about which products of inertia are zero and therefor the inertia matrix is diagonal with respect to this system, see *Using ADAMS/Solver 9.1* page 2-75. The principal axes system in ADAMS is located at the center of mass.

Here q represents the principal axes system variables used in the expressions, t stands for time and F refers to the constraint equation expressions. n is the number of constraint expressions and m is the number of principal axes system variables included in the constraint expressions. These variables are either generalized coordinates or velocities. (We are assuming that all constraint equations contain the same variables.)

Partial derivatives of equation expressions needed by ADAMS:

- The $n \cdot m$ first order partial derivatives with respect to the principal axes variables ($\frac{\partial F}{\partial q}$)

  and the $n \cdot m$ second order partial derivatives with respect to principal axes variables and time ($\frac{\partial^2 F}{\partial t \partial q}$).

- The $n \cdot m \cdot m$ second order partial derivatives of the constraint expressions with respect to the principal axes variables ($\frac{\partial^2 F}{\partial q \partial q}$).

- The n constraint expressions ($F(t)$).

- The n portion of the constraint expressions independent of the principal system variables ($f(t)$).

- The *n* first and *n* second order partial derivatives of the constraint expressions with respect to time ($\frac{\partial \boldsymbol{F}}{\partial t}, \frac{\partial^2 \boldsymbol{F}}{\partial t \partial t}$).

In total ADAMS requires $n \cdot (4 + 2 \cdot m + m^2)$ partial derivatives. (In general the actual number is less because not all partial derivatives are independent.)

5 MATHEMATICA PROGRAM

The number of partial derivatives needed by ADAMS even for small systems with simple constraints like the rolling disk makes it impractical to derive all expressions by hand calculation. This statement is true even if we take into account the fact that many of these partial derivatives are simply equal to zero. By using symbolic algebra software these derivatives can be calculated fast and easy. Starting with the zero velocity condition in vector form, Mathematica calculates 49 non-zero partial derivatives symbolically, see Appendix A.

Mathematica finally exports the constraint expressions and all partial derivatives to separate textfiles, see Appendix A. Some minor manual editing is currently necessary to format the output to the FORTRAN subroutine (ucosub.f) needed by ADAMS.

6 NUMERICAL RESULTS AND DISCUSSION

Simulating a couple of scenarios and inspecting animations and numerical results has validated the new constraint. One of the simulations involves a disk made of steel with radius 100 mm and thickness 3 mm. The disk starts out in the positive X-direction with the speed 100 mm/s. The disk is affected by an impact after 2 seconds of simulation. (The sole purpose of the impact is to get interesting dynamics for visualization.) The disk looks very much like a coin travelling across a table. The simulation proceeds smoothly without numerical difficulties. (The numerical results have not been included in this paper but are available at the author's homepage together with the ADAMS model and all other related material.)
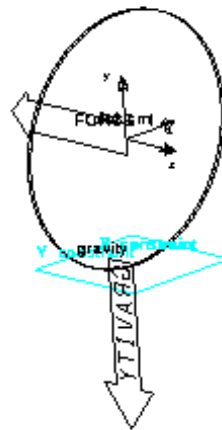
Figure 2: ADAMS model of a rolling disk.

Even though the user defined constraints capability in ADAMS is flexible and powerful some potential difficulties have been detected:

- Because UCONs are applied to the principal axes system of a part rather then the center of mass marker it is not possible to obtain the reaction forces of the constraint. This might also be a potential problem while creating the user defined constraint because in general the orientation of the principal axes system is not known in advance.

- During the simulation process Euler angle singularities must be avoided. ADAMS reorients the center of mass marker and the principal axes system whenever this is about to happen. The implementation of the UCOSUB therefor needs functionality to handle the event of reorientation of principal axes system due to Euler angle singularities. (The center of mass marker has been rotated 90° about the X-axis in order to avoid an initial Euler angle singularity, see Figure 2.)

7 CONCLUSIONS

The broad classes of real and practical important systems must be considered and modeled as MBS having both holonomic and non-holonomic constraints. Commercially available MBS software, e.g. ADAMS, can not be directly used for modeling the motion of mechanical systems with non-holonomic constraints. To solve this problem we propose a methodology based on utilization of possibilities available from both ADAMS and Mathematica. It seems reasonable to explore Mathematica for the symbolic generation of non-holonomic constraints and calculation of the respective derivatives to be used by ADAMS. The efficiency of the proposed methodology has been illustrated by the modeling and simulation of the motion of a rolling disk without slip on a horizontal plane. We also used our methodology in teaching. It makes it possible to demonstrate the effectiveness of interaction between ADAMS and Mathematica for the study of dynamics of non-holonomic systems.

REFERENCES

[1] Mechanical Dynamics Inc., *Using ADAMS/Solver 9.1*, Ann Arbor (1998).
[2] Mechanical Dynamics Inc., *Using ADAMS/Solver Subroutines 9.1*, Ann Arbor (1998).

APPENDIX A: Mathematica Program

<u>NOTE:</u> For readability reasons only part of the program has been included. The entire program can be downloaded at the authors homepage, www://mec.chalmers.se/~lidberg.

Automatic Implementation of the Kinematic Constraints of a Rolling Disk in *ADAMS* Using *Mathematica*.

```
Off[General::spell]
```

R_ψ **transformation matrix due to first rotation about the original Z-axis:**
(psi etc. used to allow automatic export to FORTRAN code)

$$R_{psi} = \begin{pmatrix} Cos[psi] & Sin[psi] & 0 \\ -Sin[psi] & Cos[psi] & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} \cos(psi) & \sin(psi) & 0 \\ -\sin(psi) & \cos(psi) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

R_θ **transformation matrix due to second rotation about the** x_1 **-axis.**

$$R_{theta} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & Cos[theta] & Sin[theta] \\ 0 & -Sin[theta] & Cos[theta] \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(theta) & \sin(theta) \\ 0 & -\sin(theta) & \cos(theta) \end{pmatrix}$$

Transformation of unit vectors from the original XYZ coordinate system to the
$x_1\, y_1\, z_1$ coordinate system :

$$\begin{pmatrix} e_{x_1} \\ e_{y_1} \\ e_{z_1} \end{pmatrix} = R_{psi} \cdot \begin{pmatrix} e_X \\ e_Y \\ e_Z \end{pmatrix}$$

$$\begin{pmatrix} \cos(psi)\, e_X + \sin(psi)\, e_Y \\ \cos(psi)\, e_Y - \sin(psi)\, e_X \\ e_Z \end{pmatrix}$$

Transformation of unit vectors from the original XYZ coordinate system to the
$x_2\, y_2\, z_2$ coordinate system :

$$\begin{pmatrix} e_{x_2} \\ e_{y_2} \\ e_{z_2} \end{pmatrix} = R_{theta} \cdot \left(R_{psi} \cdot \begin{pmatrix} e_X \\ e_Y \\ e_Z \end{pmatrix} \right)$$

$$\begin{pmatrix} \cos(psi)\, e_X + \sin(psi)\, e_Y \\ \cos(theta)\,(\cos(psi)\, e_Y - \sin(psi)\, e_X) + \sin(theta)\, e_Z \\ \cos(theta)\, e_Z - \sin(theta)\,(\cos(psi)\, e_Y - \sin(psi)\, e_X) \end{pmatrix}$$

Angular velocity of the rolling disk:

ω = psidot e_Z + thetadot e_{x_1} + phidot e_{z_2}

thetadot $(\cos(\text{psi})\,e_X + \sin(\text{psi})\,e_Y) + \text{psidot}\,e_Z +$
$\quad$ phidot $(\cos(\text{theta})\,e_Z - \sin(\text{theta})\,(\cos(\text{psi})\,e_Y - \sin(\text{psi})\,e_X))$

Angular velocity of the rolling disk in global coordinates:

ω = {Coefficient[ω, e_X], Coefficient[ω, e_Y],
$\quad$ Coefficient[ω, e_Z]}

{thetadot $\cos(\text{psi}) +$ phidot $\sin(\text{psi})\sin(\text{theta})$,
$\quad$ thetadot $\sin(\text{psi}) -$ phidot $\cos(\text{psi})\sin(\text{theta})$, psidot $+$ phidot $\cos(\text{th}$

Velocity of center of mass of rolling disk in global coordinates:

v_0 = {Xdot, Ydot, Zdot}
{Xdot, Ydot, Zdot}

Displacement of point of contact relative center of mass:

$r_{c/o}$ = -R e_{y_2}
$-R\,(\cos(\text{theta})\,(\cos(\text{psi})\,e_Y - \sin(\text{psi})\,e_X) + \sin(\text{theta})\,e_Z)$

Displacement of point of contact relative center of mass in global coordinates:

$r_{c/o}$ = {Coefficient[$r_{c/o}$, e_X], Coefficient[$r_{c/o}$, e_Y]
$\quad$ Coefficient[$r_{c/o}$, e_Z]}
{$R\cos(\text{theta})\sin(\text{psi})$, $-R\cos(\text{psi})\cos(\text{theta})$, $-R\sin(\text{theta})$}

Velocity of rolling disk at point of contact in global coordinates:

v_c = FullSimplify[v_0 + ω × $r_{c/o}$]
{Xdot $+ R\cos(\text{psi})\,(\text{phidot} + \text{psidot}\cos(\text{theta})) -$
$\quad R$ thetadot $\sin(\text{psi})\sin(\text{theta})$, Ydot $+$
$\quad R\,(\text{phidot} + \text{psidot}\cos(\text{theta}))\sin(\text{psi}) + R$ thetadot $\cos(\text{psi})\sin(\text{th}$
Zdot $- R$ thetadot $\cos(\text{theta})$}

The resulting constraint equations equals the velocity of the point of contact since the velocity of this point equals zero:

F = v_c
{Xdot $+ R\cos(\text{psi})\,(\text{phidot} + \text{psidot}\cos(\text{theta})) -$
$\quad R$ thetadot $\sin(\text{psi})\sin(\text{theta})$, Ydot $+$
$\quad R\,(\text{phidot} + \text{psidot}\cos(\text{theta}))\sin(\text{psi}) + R$ thetadot $\cos(\text{psi})\sin(\text{th}$
Zdot $- R$ thetadot $\cos(\text{theta})$}

First partial derivatives of the constraint equations with respect to the principal axes variables:

dFdq = FullSimplify$\left[\left\{\dfrac{\partial F}{\partial \text{Xdot}}, \dfrac{\partial F}{\partial \text{Ydot}}, \dfrac{\partial F}{\partial \text{Zdot}}, \dfrac{\partial F}{\partial \text{psi}}, \dfrac{\partial F}{\partial \text{theta}},\right.\right.$

$\left.\left.\dfrac{\partial F}{\partial \text{psidot}}, \dfrac{\partial F}{\partial \text{thetadot}}, \dfrac{\partial F}{\partial \text{phidot}}\right\}\right]$

$$
\left(\begin{array}{c}
1 \\
0 \\
0 \\
-R\,((\text{phidot} + \text{psidot}\cos(\text{theta}))\sin(\text{psi}) + \text{thetadot}\cos(\text{psi})\sin(\text{the} \\
-R\,(\text{thetadot}\cos(\text{theta})\sin(\text{psi}) + \text{psidot}\cos(\text{psi})\sin(\text{theta})) \\
R\cos(\text{psi})\cos(\text{theta}) \\
-R\sin(\text{psi})\sin(\text{theta}) \\
R\cos(\text{psi})
\end{array}\right.
$$

Second partial derivatives of the constraint equations with respect to the principal axes variables:

$$
\text{ddFdqdq} = \text{FullSimplify}\Big[\Big\{\frac{\partial\,\text{dFdq}}{\partial\,\text{Xdot}},\ \frac{\partial\,\text{dFdq}}{\partial\,\text{Ydot}},\ \frac{\partial\,\text{dFdq}}{\partial\,\text{Zdot}},\ \frac{\partial\,\text{dFdq}}{\partial\,\text{psi}},
$$

$$
\frac{\partial\,\text{dFdq}}{\partial\,\text{theta}},\ \frac{\partial\,\text{dFdq}}{\partial\,\text{psidot}},\ \frac{\partial\,\text{dFdq}}{\partial\,\text{thetadot}},\ \frac{\partial\,\text{dFdq}}{\partial\,\text{phidot}}\Big\}\Big]
$$

$$
\begin{array}{ccc}
(0,0,0) & (0,0,0) & (0,0,0) \\
(0,0,0) & (0,0,0) & (0,0,0) \\
(0,0,0) & (0,0,0) & (0,0,0) \\
(0,0,0) & (0,0,0) & (0,0,0)\ (R\,\text{thetadot}\sin(\text{psi})\sin(\text{theta}) - R\cos \\
(0,0,0) & (0,0,0) & (0,0,0)\ (\text{psidot}\,R\sin(\text{psi})\sin(\text{theta})) \\
(0,0,0) & (0,0,0) & (0,0,0) \\
(0,0,0) & (0,0,0) & (0,0,0) \\
(0,0,0) & (0,0,0) & (0,0,0)
\end{array}
$$

APPENDIX B: User-Written Subroutine (ucosub.f)

<u>NOTE:</u> For readability reasons only part of the program has been included. The entire program can be downloaded at the authors homepage, www://mec.chalmers.se/~lidberg.

```
      SUBROUTINE UCOSUB (ID, TIME, Q, PAR, NPAR,
     &                   IDRSEL, IFLAG, SCALAR, ARRAY,
     &                   XMATRX)
C
C Purpose: Disk must roll without slip at the xy-plane
C
C
C === Type and dimension statements ===================
C
C Note: For machines with 60 or more bits per word,
C       substitute "REAL" for "DOUBLE PRECISION".
C
C --- External variable definitions -------------------
C
      IMPLICIT NONE
      INTEGER                   ID
      DOUBLE PRECISION          TIME
      DOUBLE PRECISION          Q( 30 )
      DOUBLE PRECISION          PAR( * )
      INTEGER                   NPAR
      INTEGER                   IDRSEL( 3 )
      LOGICAL                   IFLAG
      DOUBLE PRECISION          SCALAR
      DOUBLE PRECISION          ARRAY( 30 )
      DOUBLE PRECISION          XMATRX( 30, 30 )
C
C ID       Identifier of calling UCON statement
C TIME     Current time
C PARVAR   Array of part state variables
C PAR      Array of passed statement parameters
C NPAR     Number of passed parameters
C IDRSEL   UCON values selection control flag
C IFLAG    Initialization pass flag
C SCALAR   Scalar value returned to ADAMS
C ARRAY    Partial derivatives
C XMATRX   Second partial derivatives
C
C --- Local variable definitions ---------------------
C
      INTEGER           PRTLST( 8 ), VARLST( 8 ), CONTYPE
      DOUBLE PRECISION R,Xdot,Ydot,Zdot,
     +                 psi,theta,psidot,thetadot,phidot
C
C === Executable code ===============================
C
C --- Assign parameters to readable variable names ----
C
      PRTLST(1) = NINT(PAR(1))
      PRTLST(2) = NINT(PAR(1))
      PRTLST(3) = NINT(PAR(1))
      PRTLST(4) = NINT(PAR(1))
```

```
      PRTLST(5) = NINT(PAR(1))
      PRTLST(6) = NINT(PAR(1))
      PRTLST(7) = NINT(PAR(1))
      PRTLST(8) = NINT(PAR(1))

      VARLST(1) = NINT(PAR(2))
      VARLST(2) = NINT(PAR(3))
      VARLST(3) = NINT(PAR(4))
      VARLST(4) = NINT(PAR(5))
      VARLST(5) = NINT(PAR(6))
      VARLST(6) = NINT(PAR(7))
      VARLST(7) = NINT(PAR(8))
      VARLST(8) = NINT(PAR(9))

      R         = PAR(10)

      CONTYPE   = NINT(PAR(11))
C
C --- Subroutine initialization ----------------------
C
      IF (IFLAG) THEN
C
C     Declare principal axes variables
C
         CALL UCOVAR(ID, 8, PRTLST, 8, VARLST)
      ENDIF
C
         Xdot     = Q( 1 )
         Ydot     = Q( 2 )
         Zdot     = Q( 3 )
         psi      = Q( 4 )
         theta    = Q( 5 )
       psidot   = Q( 6 )
       thetadot = Q( 7 )
         phidot   = Q( 8 )
C
C --- Evaluate scalar -------------------------------
C
      IF ( IDRSEL(1) .EQ. 1 ) THEN
         IF ( CONTYPE .EQ. 1) THEN
            SCALAR = Xdot + R*Cos(psi)*(phidot
     +              + psidot*Cos(theta))
     +              - R*thetadot*Sin(psi)*Sin(theta)
         ELSE IF ( CONTYPE .EQ. 2) THEN
            SCALAR = Ydot + R*(phidot
     +              + psidot*Cos(theta))*Sin(psi)
     +              + R*thetadot*Cos(psi)*Sin(theta)
         ELSE
            SCALAR=Zdot - R*thetadot*Cos(theta)
         ENDIF
      ENDIF
C
C --- Evaluate first derivative array ----------------
C
      IF ( IDRSEL(2) .EQ. 1 ) THEN
         IF ( CONTYPE .EQ. 1 ) THEN
            ARRAY(1) = 1.0d0
            ARRAY(4) = -(R*((phidot + psidot*Cos(theta))*Sin(psi)
     +                  + thetadot*Cos(psi)*Sin(theta)))
```

```
            ARRAY(5) = -(R*(thetadot*Cos(theta)*Sin(psi)
     +                 + psidot*Cos(psi)*Sin(theta)))
            ARRAY(6) = R*Cos(psi)*Cos(theta)
            ARRAY(7) = -(R*Sin(psi)*Sin(theta))
            ARRAY(8) = R*Cos(psi)
         ELSE IF ( CONTYPE .EQ. 2 ) THEN
            ARRAY(2) = 1.0d0
            ARRAY(4) = R*Cos(psi)*(phidot + psidot*Cos(theta))
     +                 - R*thetadot*Sin(psi)*Sin(theta)
            ARRAY(5) = R*(thetadot*Cos(psi)*Cos(theta)
     +                 - psidot*Sin(psi)*Sin(theta))
            ARRAY(6) = R*Cos(theta)*Sin(psi)
            ARRAY(7) = R*Cos(psi)*Sin(theta)
            ARRAY(8) = R*Sin(psi)
         ELSE IF ( CONTYPE .EQ. 3 ) THEN
            ARRAY(3) = 1.0d0
            ARRAY(5) = R*thetadot*Sin(theta)
            ARRAY(7) = -(R*Cos(theta))
         ENDIF
       ENDIF
C
C --- Second partials derivatives
C
      IF ( IDRSEL(3) .EQ. 0 ) THEN
       IF ( CONTYPE .EQ. 1 ) THEN
         XMATRX(4,4) = -(R*Cos(psi)*(phidot + psidot*Cos(theta)))
     +                 + R*thetadot*Sin(psi)*Sin(theta)
         XMATRX(4,5) = -(R*thetadot*Cos(psi)*Cos(theta))
     +                 + psidot*R*Sin(psi)*Sin(theta)
         XMATRX(4,6) = -(R*Cos(theta)*Sin(psi))
         XMATRX(4,7) = -(R*Cos(psi)*Sin(theta))
         XMATRX(4,8) = -(R*Sin(psi))
         XMATRX(5,4) = -(R*thetadot*Cos(psi)*Cos(theta))
     +                 + psidot*R*Sin(psi)*Sin(theta)
         XMATRX(5,5) = -(psidot*R*Cos(psi)*Cos(theta))
     +                 + R*thetadot*Sin(psi)*Sin(theta)
         XMATRX(5,6) = -(R*Cos(psi)*Sin(theta))
         XMATRX(5,7) = -(R*Cos(theta)*Sin(psi))
         XMATRX(6,4) = -(R*Cos(theta)*Sin(psi))
         XMATRX(6,5) = -(R*Cos(psi)*Sin(theta))
         XMATRX(7,4) = -(R*Cos(psi)*Sin(theta))
         XMATRX(7,5) = -(R*Cos(theta)*Sin(psi))
         XMATRX(8,4) = -(R*Sin(psi))
        ELSE IF ( CONTYPE .EQ. 2 ) THEN
         XMATRX(4,4) = -(R*((phidot + psidot*Cos(theta))*Sin(psi)
     +                 + thetadot*Cos(psi)*Sin(theta)))
         XMATRX(4,5) = -(R*(thetadot*Cos(theta)*Sin(psi)
     +                 + psidot*Cos(psi)*Sin(theta)))
         XMATRX(4,6) = R*Cos(psi)*Cos(theta)
         XMATRX(4,7) = -(R*Sin(psi)*Sin(theta))
         XMATRX(4,8) = R*Cos(psi)
```