Fortran Tyre Model Subroutines

B.1 Interpolation tyre model subroutine

```
    SUBROUTINE TIRSUB ( ID, TIME, TO, CPROP, TPROP, MPROP,
    &                     PAR, NPAR, STR, NSTR, DFLAG,
    &                     IFLAG, FSAE, TSAE, FPROP )
C
C   This program is part of the CUTyre system - M Blundell, Feb 1997
C   This version is based on an interpolation approach using measured
C  tyre test data which is include in SPLINE statements. The model is referred
to as the
C   Limited version based on the limited testing where camber and slip
are varied
C   independently.
C
C
C   The coefficients in the model assume the following units:
C   slip angle: degrees
C   camber angle: degrees
C   Fz (load): kg
C   Fy and Fx: N
C   Tz : Nm
C
C   Note this subroutine is developed to not account for offsets
C   twice. The offsets are include for slip interpolation
C   but for camber the offset at zero camber is subtracted.
C
C Inputs:
C
    INTEGER           ID, NPAR, NSTR
    DOUBLE PRECISION TIME, TO
    DOUBLE PRECISION CPROP(*), TPROP(*), MPROP(*), PAR(*)
    CHARACTER*80      STR(*)
    LOGICAL           DFLAG, IFLAG, ERRFLG
C
C Outputs:
C
    DOUBLE PRECISION FSAE(*), TSAE(*), FPROP(*), ARRAY(3)
C
```

```
C Local Variables:
C
    DOUBLE PRECISION  SLIP, ALPHA, DEFL, DEFLD
    DOUBLE PRECISION  R2, CZ, CS, CA, CR, DZ, AMASS, WSPIN
C
    DOUBLE PRECISION  GAMMA,CG,RALPHA,RGAMMA,FZL,TZL,TZLA,TZLG
    DOUBLE PRECISION CFY,DFY,EFY,SHFY,SVFY,PHIFY,TZLG0,TZLG1
    DOUBLE PRECISION CTZ,DTZ,ETZ,BTZ,SHTZ,SVTZ,PHITZ
    DOUBLE PRECISION CFX,DFX,EFX,BFX,SHFX,SVFX,PHIFX
C
    INTEGER           IORD
    DOUBLE PRECISION  ZERO, ONE, SCFACT, DELMAX,FYA,FYG,FYG0,FYG1
    DOUBLE PRECISION  FX, FY, FZ, FX1, FX2, TY, TZ, H, ASTAR, SSTAR
    DOUBLE PRECISION  U, FZDAMP, FZDEFL, WSPNMX, DTOR, RTOD
    LOGICAL           ERFLG
C
    PARAMETER        (ZERO=0.0)
    PARAMETER        (ONE=1.0)
    PARAMETER        (IORD=0)
    PARAMETER        (WSPNMX=5.0D-1)
    PARAMETER        (DTOR=0.017453292)
    PARAMETER        (RTOD=57.29577951)
C
C
C  EXECUTABLE CODE
C
C
C   Extract data from input arrays
C
    SLIP  = CPROP(1)
    DEFL  = CPROP(4)
    DEFLD = CPROP(5)
    WSPIN = CPROP(8)
C
    AMASS = MPROP(1)
C
     R2    = TPROP(2)
     CZ    = TPROP(3)
     CS    = TPROP(4)
     CA    = TPROP(5)
     CR    = TPROP(7)
     DZ    = TPROP(8)
     U     = TPROP(11)
C
     RALPHA = CPROP(2)
     RGAMMA = CPROP (3)
     CG = TPROP (6)
```

```
      ALPHA=RALPHA*RTOD
      GAMMA=RGAMMA*RTOD
C
C     Initialize force values
C
         FX = 0.D0
         FY = 0.D0
         FZ = 0.D0
         TY = 0.D0
         TZ = 0.D0
C
      IF(DEFL .LE. 0.D0) THEN
         GOTO 1000
      ENDIF
C
C     Calculate normal loads due to stiffness (always .LE. zero)
C
      FZDEFL = -DEFL*CZ
C
C     Calculate normal loads due to damping
C
      FZDAMP = - 2.D0*SQRT(AMASS*CZ)*DZ*(DEFLD)
C
C     Calculate total normal force (fz)
C
      FZ     =  MIN (0.0D0, (FZDEFL + FZDAMP) )
C
C     Convert to kg and change sign
C
      FZL = -FZ/9.81
C
C     Calculate critical longitudinal slip value
C
      SSTAR = ABS(U*FZ/(2.D0*CS))
C
C     Compute longitudinal force
C
      IF(ABS(SLIP) .LE. ABS(SSTAR)) THEN
        FX = -CS*SLIP
      ELSE
        FX1 = U*ABS(FZ)
        FX2 = (U*FZ)**2/(4.D0*ABS(SLIP)*CS)
        FX = -(FX1-FX2)*SIGN(1.0D0,SLIP)
      ENDIF
C
C     Compute lateral force
C
```

```
      CALL CUBSPL (ALPHA,FZL,100,0,ARRAY,ERRFLG)
      FYA=ARRAY(1)
      CALL CUBSPL (0,FZL,300,0,ARRAY,ERRFLG)
      FYG0=ARRAY(1)
      CALL CUBSPL (GAMMA,FZL,300,0,ARRAY,ERRFLG)
      FYG1=ARRAY(1)
      FYG=FYG1-FYG0
      FY=FYA+FYG
C
C     Compute self aligning moment
C
      CALL CUBSPL (ALPHA,FZL,200,0,ARRAY,ERRFLG)
      TZLA=ARRAY(1)
      CALL CUBSPL (0,FZL,400,0,ARRAY,ERRFLG)
      TZLG0=ARRAY(1)
      CALL CUBSPL (GAMMA,FZL,400,0,ARRAY,ERRFLG)
      TZLG1=ARRAY(1)
      TZLG=TZLG1-TZLG0
      TZL=TZLA+TZLG
C
C     Convert to Nmm
C
      TZ = TZL*1000.0
C
C     Copy the calculated values for FX, FY, FZ, TY & TZ to FSAE
C     and TSAE arrays
C
1000  FSAE(1) = FX
      FSAE(2) = FY
      FSAE(3) = FZ
C
      TSAE(1) = 0.0
      TSAE(2) = 0.0
      TSAE(3) = TZ
C
      FPROP(1) = 0.0
      FPROP(2) = 0.0
C
      RETURN
      END
```

B.2 Magic formula tyre model (version 3) subroutine

```
      SUBROUTINE TIRSUB ( ID, TIME, T0, CPROP, TPROP, MPROP,
     &                    PAR, NPAR, STR, NSTR, DFLAG,
     &                    IFLAG, FSAE, TSAE, FPROP )
C
```

```
C    This program is part of the CUTyre system - M Blundell, Feb 1997
C    This version is based on the Magic Formula tyre model (Version 3).
C    Coefficients are for TYRE B
C
C    The coefficients in the model assume the following units:
C    slip angle: radians
C    camber angle: radians
C    slip ratio %
C    Fz (load): N
C    Fy and Fx: N
C    Tz : Nm
C    Note sign changes between Paceka formulation and SAE convention
C   If camber is not included set A5,A10,A13,A14,A15,A16
C   and C6,C10,C13,C16,C17,C18,C19,C20 to zero
C
C Inputs:
C
      INTEGER          ID, NPAR, NSTR
      DOUBLE PRECISION TIME, T0
      DOUBLE PRECISION CPROP(*), TPROP(*), MPROP(*), PAR(*)
      CHARACTER*80     STR(*)
      LOGICAL          DFLAG, IFLAG
C
C Outputs:
C
      DOUBLE PRECISION FSAE(*), TSAE(*), FPROP(*)
C
C Local Variables:
C
      DOUBLE PRECISION  SLIP, ALPHA, DEFL, DEFLD
      DOUBLE PRECISION  R2, CZ, CS, CA, CR, DZ, AMASS, WSPIN
C
C
C
      DOUBLE PRECISION GAMMA,CG,RALPHA,RGAMMA,FXP,FZP,FYP,TZP
      DOUBLE PRECISION A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13
      DOUBLE PRECISION A14,A15,A16,A17,SLIPCENT
      DOUBLE PRECISION C0,C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11,C12,C13
      DOUBLE PRECISION C14,C15,C16,C17,C18,C19,C20
      DOUBLE PRECISION CFY,DFY,EFY,SHFY,SVFY,PHIFY
      DOUBLE PRECISION CTZ,DTZ,ETZ,BTZ,SHTZ,SVTZ,PHITZ
      DOUBLE PRECISION CFX,DFX,EFX,BFX,SHFX,SVFX,PHIFX,DUMTZ,DUMFY
C
      INTEGER           IORD
      DOUBLE PRECISION  ZERO, ONE, SCFACT, DELMAX
      DOUBLE PRECISION  FX, FY, FZ, FX1, FX2, TY, TZ, H, ASTAR, SSTAR
      DOUBLE PRECISION  U, FZDAMP, FZDEFL, WSPNMX, DTOR, RTOD
      LOGICAL           ERFLG
```

```
C
      PARAMETER         (ZERO=0.0)
      PARAMETER         (ONE=1.0)
      PARAMETER         (IORD=0)
      PARAMETER         (WSPNMX=5.0D-1)
      PARAMETER         (DTOR=0.017453292)
      PARAMETER         (RTOD=57.29577951)
C
C       Define Pacejka Coefficients
C
      A0=.103370E+01
      A1=-.224482E-05
      A2=.132185E+01
      A3=.604035E+05
      A4=.877727E+04
      A5=0.0
      A6=.458114E-04
      A7=.468222
      A8=.381896E-06
      A9=.516209E-02
      A10=0.00
      A11=-.366375E-01
      A12=-.568859E+02
      A13=0.00
      A14=0.00
      A15=0.00
      A16=0.00
      A17=.379913
C
C
      C0=.235000E+01
      C1=.266333E-05
      C2=.249270E-02
      C3=-.159794E-03
      C4=-.254777E-01
      C5=.142145E-03
      C6=0.00
      C7=.197277E-07
      C8=-.359537E-03
      C9=.630223
      C10=0.00
      C11=.120220E-06
      C12=.275062E-02
      C13=0.00
      C14=-.172742E-02
      C15=.544249E+01
      C16=0.00
      C17=0.00
```

```
      C18=0.00
      C19=0.00
      C20=0.00
C
C
C     EXECUTABLE CODE
C
C
C     Extract data from input arrays
C
      SLIP   = CPROP(1)
      DEFL   = CPROP(4)
      DEFLD  = CPROP(5)
      WSPIN  = CPROP(8)
C
      AMASS  = MPROP(1)
C
      R2     = TPROP(2)
      CZ     = TPROP(3)
      CS     = TPROP(4)
      CA     = TPROP(5)
      CR     = TPROP(7)
      DZ     = TPROP(8)
      U      = TPROP(11)
C
C     Convert sign on alpha
C
      RALPHA = CPROP(2)
      RGAMMA = CPROP (3)
      CG = TPROP (6)
      ALPHA=-RALPHA
      GAMMA=RGAMMA
C
C    Initialize force values
C
        FX = 0.D0
        FY = 0.D0
        FZ = 0.D0
        TY = 0.D0
        TZ = 0.D0
C
      IF(DEFL .LE. 0.D0) THEN
        GOTO 1000
      ENDIF
C
C     Calculate normal loads due to stiffness (always .LE. zero)
C
      FZDEFL = -DEFL*CZ
```

```
C
C     Calculate normal loads due to damping
C
      FZDAMP = - 2.D0*SQRT(AMASS*CZ)*DZ*(DEFLD)
C
C     Calculate total normal force (fz)
C
      FZ     =  MIN (0.0D0, (FZDEFL + FZDAMP) )
C
C     Convert to kN and change sign
C
      FZP = -FZ
C
C     Compute longitudinal force
C
      IF(ABS(SLIP) .LE. ABS(SSTAR)) THEN
         FX = -CS*SLIP
      ELSE
         FX1 = U*ABS(FZ)
         FX2 = (U*FZ)**2/(4.D0*ABS(SLIP)*CS)
         FX  = -(FX1-FX2)*SIGN(1.0D0,SLIP)
        ENDIF
C
C     Compute lateral force
C
      CFY=A0
      SHFY=A8*FZP+A9+A10*GAMMA
      DFY=(A1*FZP+A2)*(1-A15*GAMMA**2)*FZP
      IF(ALPHA+SHFY.LT.0.0)THEN
        DUMFY=-1.0
      ELSE
        DUMFY=1.0
      ENDIF
      EFY=(A6*FZP+A7)*(1-(A16*GAMMA+A17)*DUMFY)
      BFY=((A3*SIN(2*ATAN(FZP/A4)))*(1-A5*ABS(GAMMA)))/(CFY+DFY)
      SVFY=A11*FZP+A12+(A13*FZP**2+A14*FZP)*GAMMA
      PHIFY=(1-EFY)*(ALPHA+SHFY)+(EFY/BFY)*ATAN(BFY*(ALPHA+SHFY))
      FYP=DFY*SIN(CFY*ATAN(BFY*PHIFY))+SVFY
C
C     Change sign
C
      FY=FYP
C
C     Compute self aligning moment
C
      CTZ=C0
      SHTZ=C11*FZP+C12+C13*GAMMA
      DTZ=(C1*FZP**2+C2*FZP)*(1-C18*GAMMA**2)
```

```
      IF(ALPHA+SHTZ.LT.0.0)THEN
        DUMTZ=-1.0
      ELSE
        DUMTZ=1.0
      ENDIF
      ETZ=(C7*FZP**2+C8*FZP+C9)*(1-(C19*GAMMA+C20)*DUMTZ)
      ETZ=ETZ/(1-C10*ABS(GAMMA))
      BTZ=((C3*FZP**2+C4*FZP)*(1-C6*ABS(GAMMA))*EXP(-C5*FZP))/(CTZ+DTZ)
      SVTZ=C14*FZP+C15+(C16*FZP**2+C17*FZP)*GAMMA
      PHITZ=(1-ETZ)*(ALPHA+SHTZ)+(ETZ/BTZ)*ATAN(BTZ*(ALPHA+SHTZ))
      TZP=DTZ*SIN(CTZ*ATAN(BTZ*PHITZ))+SVTZ
C
C     Convert to Nmm and change sign
C
      TZ = TZP*1000.0
C
C     Copy the calculated values for FX, FY, FZ, TY & TZ to FSAE
C     and TSAE arrays
C
1000  FSAE(1) = FX
      FSAE(2) = FY
      FSAE(3) = FZ
      TSAE(1) = 0.0
      TSAE(2) = 0.0
      TSAE(3) = TZ
      FPROP(1) = 0.0
      FPROP(2) = 0.0
C
      RETURN
      END
```

B.3 The Harty tyre model subroutine TYR501

```
C  MDI TYR501 : Concept Tyre Model
C
C
C  A Quick & Dirty Tyre Model which plugs in as the FIALA
C  model does, with a "TIRE" statement.
C
C  Unlike FIALA, critical slip angle is broadly independent
C  of load and initial cornering stiffness is strongly
C  load dependent.
C
C  These attributes better represent a modern radial tyre
C  than does either the FIALA or University of Arizona
C  model.
C
```

```
C  The model does handle comprehensive slip.  Lateral force
C  generation is zero at peak longitudinal force slip ratio
C  (typically about 20%) but returns to a value around one
C  tenth of the peak lateral force as the wheel progresses
C  beyond that limit.  This may result in poor post-spin
C  performance.  The force generated with locked wheels is
C  aligned with the wheel plane; this is incorrect.
C
C  Longitudinal force generation is assumed to be symmetric
C  for tractive and braking slip.  This is not generally
C  true beyond the critical slip ratio for real tyres but
C  is reasonable up to that point.  This tyre will over
C  estimate longitudinal forces for tractive slip and
C  slightly underestimate them for braking slip in the
C  post-critical regions.
C
C  -- 29th December 2000 --
C
C  Camber thrust is included as for the motorcycle tire
C  model using "taut string" logic.  Lateral migration of
C  the contact patch is also included, as for the motorcycle
C  tyre model.
C
C  Aligning Torque calculation includes the lateral force
C  due to camber.  This is not quite right as the camber
C  force mechanism has no pneumatic trail associated with
C  it.  Pay attention if using this for motorcycle work;
C  consider reworking it so that TZ does not include the
C  camber force.  The form of the aligning torque is a
C  bit poor and would benefit from some more thought;
C  pneumatic trail collapses linearly with lateral force.
C
C  -- 10th January 2001 --
C
C  Unsuitable Aligning Moment behaviour substantially improved
C  for motorcycle use.
C
C  --
C
C  Relaxation Length is externally imposed as with the
C  Fiala tyre.
C
C  Tyre Data is taken from the tyre parameter file (.tpf)
C  but note that not all the data is used.  The other
C  parameters are passed in via the UPARAMETERS argument
C  on the TIRE statement inside an ADAMS deck.
C
```

```
C  The model is quite empirical and has no basis in any sort
C  of established fact or theory.  It may or may not bear a
C  passing resemblance to "Maltyre", a Malcolm Burgess model
C  implemented at Lotus to the same end.  I don't care, I
C  did it all myself without a grown-up to help with the
C  pointy bits.
C
C  -- 24th April 2001 --
C
C  Banner and zero parameter check added in IFLAG loop.
C
C  --  7th July 2001 --
C
C  Improved representation of behaviour outside friction
C  circle.  Correct differentiation between lock and
C  wheelspin in terms of force vector.
C
C  --  6th October 2004 --
C
C  Improved aligning moment form - was significantly too high.
C  Uses passed in paremeter for Pneumatic Trail on-centre.
C  Note that passed-in parameter can be negative, giving
C  pneumatic "lead".
C
C  -- 10th March 2006 --
C
C  Pneumatic lead introduced for camber forces to match
C  measured motorcycle data. Minor error in limit camber
C  clipping corrected
C
C  -- 30th March 2006 --
C
C  Minor error with form of camber clipping  (asymmetric)
C  corrected.
C
C
C  -- 16th February 2009 --
C
C  Damian made the mistake of letting someone else have a go at
C  his model and so I am attempting to migrate it to TYR501
C  since the TIRSUB routine will become defunct at the next release.
C
C  (Teena Gade)
C
C  -- 7th September 2009 --
C
C  Migration to TYR501 completed by DAH after Teena did all the
C  nasty bits getting the right data into the right place.
C
```

```
C  -- 22nd May 2013 --
C
C  Sign error in forces carried over from original MSC TYR501
C  sample file has led to erratic behaviour of TYR501 until
C  pinned down, now fixed. Not that FORCES and TORQUE are the
C  variables which actually deliver forces back to the solution.
C  VARINF is associated with VPG Tire, a mode of usage I have
C  never successfully invoked. VARINF information is of unknown
C  provenance and should be used without checking.
C
C  Also uncovered some strange behaviour of original TYR501
C  that didn't allow it to run backwards - fixed with velocity
C  sign check just before FORCES is returned. VARINF not
C  corrected.
C
C  Resulting bug with rolling resistance fixed, works
C  correctly (has it ever done before?)
C
C  --
C
C  (c) DAH 24 Oct 1999-2013
C
        SUBROUTINE TYR501( NDEV, ISWTCH, JOBFLG, IDTYRE,
     +             TIME, DIS, TRAMAT, ANGTWC, VEL, OMEGA, OMEGAR,
     +             NDEQVR, DEQVAR, NTYPAR, TYPARR,
     +             NCHTDS, CHTDST, ROAD, IDROAD,
     +             NROPAR, ROPAR, NCHRDS, CHRDST,
     +             FORCES, TORQUE, DEQINI, DEQDER, TYRMOD,
     +             NVARS, VARINF, NWORK, WRKARR,
     +             NIWORK, IWRKAR, IERR )
C
C Inputs:
      INTEGER          NDEV
      INTEGER          ISWTCH
      INTEGER          JOBFLG
      INTEGER          IDTYRE
      DOUBLE PRECISION TIME
      DOUBLE PRECISION DIS(3)
      DOUBLE PRECISION TRAMAT(3,3)
      DOUBLE PRECISION ANGTWC
      DOUBLE PRECISION VEL(3)
      DOUBLE PRECISION OMEGA(3)
      DOUBLE PRECISION OMEGAR
      INTEGER          NDEQVR
      DOUBLE PRECISION DEQVAR(NDEQVR)
      INTEGER          NTYPAR
      DOUBLE PRECISION TYPARR(NTYPAR)
```

```
      INTEGER          NCHTDS
      CHARACTER*256    CHTDST
      INTEGER          IDROAD
      INTEGER          NROPAR
      DOUBLE PRECISION ROPAR(NROPAR)
      INTEGER          NCHRDS
      CHARACTER*256    CHRDST
C Outputs:
      DOUBLE PRECISION FORCES(3)
      DOUBLE PRECISION TORQUE(3)
      DOUBLE PRECISION DEQINI(NDEQVR)
      DOUBLE PRECISION DEQDER(NDEQVR)
      CHARACTER*256    TYRMOD
      INTEGER          NVARS
      DOUBLE PRECISION VARINF(NVARS)
      INTEGER          NWORK
      DOUBLE PRECISION WRKARR(NWORK)
      INTEGER          NIWORK
      INTEGER          IWRKAR(NIWORK)
      INTEGER          IERR
C
C
C
C  Local Variables:
C
C  Locals:
      INTEGER  I
C       DOUBLE PRECISION C_SLIP
      DOUBLE PRECISION C_ALPHA
      DOUBLE PRECISION C_GAMMA
      DOUBLE PRECISION U1
      DOUBLE PRECISION U0
      DOUBLE PRECISION GAIN
      DOUBLE PRECISION R_LEN
      DOUBLE PRECISION URAD(3)
      DOUBLE PRECISION U
      DOUBLE PRECISION F(6)
      DOUBLE PRECISION FCP(3)
      DOUBLE PRECISION TCP(3)
      INTEGER          ARRPTR
      INTEGER          UMODE
      DOUBLE PRECISION RAD(3)
      DOUBLE PRECISION RADIUS
      INTEGER          NROAD
      DOUBLE PRECISION RCP(3)
      DOUBLE PRECISION RNORM(3)
      DOUBLE PRECISION SURFAC
```

```
      DOUBLE PRECISION CN
      DOUBLE PRECISION RDR
C     DOUBLE PRECISION CRR
      DOUBLE PRECISION CPMTX(3,3)
      DOUBLE PRECISION VCPLON
      DOUBLE PRECISION VCPLAT
      DOUBLE PRECISION VCPVRT
      DOUBLE PRECISION VLON
C     DOUBLE PRECISION ALPHA
        DOUBLE PRECISION ALPHA_L
      DOUBLE PRECISION KAPPA
      DOUBLE PRECISION KAPPA_L
      DOUBLE PRECISION GAMMA
      DOUBLE PRECISION FRCRAD
      DOUBLE PRECISION FRCVRT
      DOUBLE PRECISION FRCLON
      DOUBLE PRECISION FRCLAT
      DOUBLE PRECISION TRQALN
      DOUBLE PRECISION FZMAG
C
C  ------------------------------------------------------------------
C  -- Carried across from tirsub --
      DOUBLE PRECISION  FX, FY, FZ, TX, TY, TZ

      DOUBLE PRECISION  SLIP, ALPHA, DEFL, DEFLD
      DOUBLE PRECISION  R1, R2, CZ, CS, C_MX, CR, DZ, AMASS, WSPIN

      DOUBLE PRECISION  ALPHA_C, Ay, By, R_LOAD, dB_dFz, B
      DOUBLE PRECISION  SLIP_C, Ax, SLIP_M, FR_ELLIP, CP_LEN
      DOUBLE PRECISION  LSLIP, USLIP, UNLRAD

      DOUBLE PRECISION  THRSH, CAMB_C, CAMB_INC, A_INT, B_INT, C_INT
      DOUBLE PRECISION  SLIPSQ, TAN_ALPHA_SQ, DIVISOR
      DOUBLE PRECISION  FX1, FY1, FX2, FY2, ABSLIP, PTRAILC, PNOFFSET
      DOUBLE PRECISION  PLEAD, FY_CAMBER, FYWAS
C  ------------------------------------------------------------------
C
C
C  Scaling parameters

C     DOUBLE PRECISION SCLRR
      DOUBLE PRECISION SCLFY
      DOUBLE PRECISION SCLMX
      DOUBLE PRECISION SCLMZ

C  Drift array parameters

      DOUBLE PRECISION PLYFRC
      DOUBLE PRECISION CONFRC
```

```
      DOUBLE PRECISION PLYTRQ
      DOUBLE PRECISION CONTRQ

      LOGICAL          ERRFLG
C
C  Road Declarations:
      INTEGER          MAXDIV
      PARAMETER       (MAXDIV = 10)
      INTEGER          N_T_SHAPE
      DOUBLE PRECISION T_SHAPE(2, MAXDIV)
      DOUBLE PRECISION EFFVOL
      DOUBLE PRECISION EFFPEN
      CHARACTER*256    ERRMSG
      CHARACTER*80     ERRTMP
C
      DOUBLE PRECISION STARTUP
      DOUBLE PRECISION OFF_GRND
C
C  Useful Parameters:
      DOUBLE PRECISION ZERO_VAL
      PARAMETER       (ZERO_VAL = 0.D0)
      DOUBLE PRECISION ONE
      PARAMETER       (ONE = 1.D0)
      DOUBLE PRECISION ZERLIM
      PARAMETER       (ZERLIM = 1.0E-10)
      DOUBLE PRECISION TFULL
      PARAMETER       (TFULL = 0.5)
      DOUBLE PRECISION NO_FRC
      PARAMETER       (NO_FRC = 448)
      DOUBLE PRECISION WSPNMX
      PARAMETER       (WSPNMX = 5.0D-1)
      INTEGER          DYNAMIC
      PARAMETER       (DYNAMIC = 1)
      INTEGER          STATIC
      PARAMETER       (STATIC  = 0)
      INTEGER          IORD
      PARAMETER       (IORD=0)
      INTEGER          IMODE
C
      include 'ac_tir_jobflg.inc'
      include 'abg_varptr.inc'
      include 'tyrHarty_501.inc'
C  Functions:
      DOUBLE PRECISION DOT
      EXTERNAL         DOT
C
```

```
      EXTERNAL         ROAD

      LOGICAL          STAFLG
      SAVE             STAFLG

      DATA   STAFLG   /.FALSE./

      IERR = 0
C  Read the tire property file during initialization:

      IF ( JOBFLG .EQ. INIT ) THEN
      CALL USRMES( .TRUE.,
   +  '  ', 0,
   +  'INFO_NOPAD' )
      CALL USRMES( .TRUE.,
   +  '  ', 0,
   +  'INFO_NOPAD' )
      CALL USRMES( .TRUE.,
   +  '*************************************************', 0,
   +  'INFO_NOPAD' )
      CALL USRMES( .TRUE.,
   +  'TYR501 Harty Model: Compiled 12 Aug 2013', IDTYRE,
   +  'INFO_NOPAD' )
      CALL USRMES( .TRUE.,
   +  '*************************************************', 0,
   +  'INFO_NOPAD' )
      CALL USRMES( .TRUE.,
   +  '  ', 0,
   +  'INFO_NOPAD' )
      CALL USRMES( .TRUE.,
   +  '  ', 0,
   +  'INFO_NOPAD' )
       CALL RPF501( NCHTDS, CHTDST, IDTYRE, NTYPAR, TYPARR )
      ENDIF

C  Set DEQINI:
      IF ( JOBFLG .EQ. INQUIRE ) THEN
       DEQINI(1) = 0.0D0
       DEQINI(2) = 0.0D0
      ENDIF

  IF  (JOBFLG .NE. ENDSIM) THEN


C  Decode TYPARR Array:
      UMODE  = NINT( TYPARR( use_mode ) )

      UNLRAD  = TYPARR( unloaded_radius )
      TIREW   = TYPARR( width )
      TIREK   = TYPARR( vertical_stiffness )
```

```
      TIREC   = TYPARR( vertical_damping )
      CR      = TYPARR( rolling_resistance )
      CA      = TYPARR( calpha )
      C_GAMMA = TYPARR( cgamma )
      U_MIN   = TYPARR( umin )
      U_MAX   = TYPARR( umax )
      R_LEN   = TYPARR( relaxation_length )
      ALPHA_C = TYPARR( alpha_critical )
      Ay      = TYPARR( curvature_factor_angle )
      By      = TYPARR( scale_factor_lateral )
      R_LOAD  = TYPARR( rated_load )
      dB_dFz  = TYPARR( scale_factor_dim )
      SLIP_C  = TYPARR( slip_ratio_critical )
      Ax      = TYPARR( curvature_factor_ratio )
      PTRAILC = TYPARR( pneum_trailing_scaling )
      PLEAD   = TYPARR( pneumatic_lead_camber )
      THRSH   = TYPARR( limit_camber_onset_fric )
C
      N_T_SHAPE  = NINT( TYPARR( n_shape ) )
C
      IF ( JOBFLG .EQ. INIT .OR. JOBFLG .EQ. RESET ) THEN
C     -- Debug only - check we're getting what we think --
C---------------------------------------------------------------------
C        WRITE(*,*) 'UNLRAD ', UNLRAD, ', TIREK ', TIREK
c        WRITE(*,*) ', TIREW ', TIREW
c        WRITE(*,*) 'TIREC', TIREC, ', CR ', CR
c        WRITE(*,*) 'CA ', CA
c        WRITE(*,*) 'C_GAMMA ', C_GAMMA, 'U_min ', U_min
c        WRITE(*,*) 'U_max ', U_max, 'R_LEN ', R_LEN
c        WRITE(*,*) 'ALPHA_C ', ALPHA_C, 'Ay ', Ay
c        WRITE(*,*) 'By ', By, 'R_LOAD ', R_LOAD
c        WRITE(*,*) 'dB_dFz ', dB_dFz
c        WRITE(*,*) 'SLIP_C ', SLIP_C
c        WRITE(*,*) 'Ax ', Ax
C      WRITE(*,*) 'PTRAILC ', PTRAILC, 'PLEAD ', PLEAD, 'THRSH ', THRSH
C---------------------------------------------------------------------
C
C
      ENDIF
C


C=====================================================================
C
C  -- All this is standard TYR501 stuff
C     - dynamic or static
C     - soft start to calculations
```

```
C     - road/tyre interaction including profile
C     - states for the tyre model
C
C======================================================================

C  Initialize mode (STATIC or DYNAMIC)
       IMODE = DYNAMIC
       IF ( ISWTCH .EQ. 0 ) IMODE = STATIC

C  Set flag for quasi-statice analyses

       IF ( ISWTCH .EQ. 2 ) STAFLG = .TRUE.

C  Setup Smoothing Function:
C
C  The MDI tire models include a feature for smoothing the
C  tire forces around time=0.0.  So, for example, if there's
C  some initial slip angle at time=0.0, the lateral force
C  builds up slowly instead of acting like a step input.
C  This helps the integrator get started. UMODE comes
C  from the tire property file.

       IF(UMODE .GE. 2 .AND.(.NOT.STAFLG) )THEN
         CALL STEP(TIME,ZERO_VAL,ZERO_VAL,TFULL,ONE,0,
     +            STARTUP,ERRFLG)
       ELSE
         STARTUP = ONE
       ENDIF

C  Setup The Tire Carcase (Cross Section) Shape
C  for use by the durability tire road contact
C  model:

       IF (N_T_SHAPE.EQ.0) THEN
         T_SHAPE(1,1) = 0.D0
         T_SHAPE(2,1) = 0.D0
       ELSE
         ARRPTR = SHAPE
       DO I=1,N_T_SHAPE
         T_SHAPE(1,I)=TYPARR(ARRPTR)
   T_SHAPE(2,I)=TYPARR(ARRPTR+1)
         ARRPTR = ARRPTR + 2
       ENDDO
     ENDIF

C  Offset rolling radius - this is in the original code but
C  I don't know why.
C
C    UNLRAD = UNLRAD + SCLRR
```

```
C  Call ROAD routine
C
C  The road routine calculates the local road normal, the
C  road contact point (contact patch location), the
C  local surface coefficent of friction and the tire’s
C  vertical deflection.  The STI passes in the name of
C  the subroutine to be called.  Hence "ROAD" is just a
C  placeholder.

        CALL ROAD(JOBFLG, IDTYRE,
     &            TIME, DIS, TRAMAT,
     &            IDROAD, NROPAR, ROPAR, NCHRDS, CHRDST,
     &            N_T_SHAPE, T_SHAPE, UNLRAD, TIREW,
     &            NROAD, EFFVOL, EFFPEN, RCP,
     &            RNORM, SURFAC, IERR, ERRMSG )



C  Call the TIRE Kinematics Routine (ACTCLC):
C
C  The ACTCLC routine calculates the slip angle (ALPHA),
C  inclination (camber) angle (GAMMA), longitudinal slip
C  (KAPPA), the longitudinal (VCPLON) and lateral (VCPLAT)
C  slip velocities, the longitudinal velocity of wheel
C  center (VLON), the vertical velocity of the wheel center
C  normal to the road (VCPVRT), the unit vector directed
C  from the wheel center to the contact patch (URAD) expressed
c  in global coordinates, and the transformation
C  matrix from SAE contact patch coordinates (CPMTX) to
C  global (ground part) coordinates.
C
C  Calculate the tire kinematics if:
C     The tire is in contact with road (e.g. not flying)
C
C     - and -
C
C     The job is normal execution or differencing for
C     derivatives.
C
     IF(
  .       NROAD .EQ. 1 .AND. IERR .NE. 3
  .       .AND.
  .            (
  .              JOBFLG .EQ. NORMAL .OR.
  .              JOBFLG .EQ. DIFF
  .            )
  .    ) THEN
```

```
      RADIUS = UNLRAD - EFFPEN


      CALL ACTCLC(TRAMAT, VEL, OMEGA, OMEGAR, RADIUS, RNORM,
     &             VLON, VCPLON, VCPLAT, VCPVRT,
     &             ALPHA, GAMMA, KAPPA,
     &             URAD, CPMTX)

c
c  Lag The slip angle to for tire relaxation effects:
c
c  d( Alpha_lagged )/dt = (VLON/Relaxation_Length)*( Alpha - Alpha_lagged )
c
c  If the relaxation length is less than 1e-4 Meters, then don't lag the
c  slips.
c
     IF ( R_LEN .LT. 1D-4  .OR.
  .        IMODE .EQ. STATIC ) THEN

      ALPHA_L  = ALPHA
      KAPPA_L  = KAPPA

     ELSE

      GAIN      = ABS(VLON)/R_LEN
      ALPHA_L   = DEQVAR(1) + DEQINI(1)
      KAPPA_L   = DEQVAR(2) + DEQINI(2)
      DEQDER(1) = GAIN*(ALPHA - ALPHA_L)
      DEQDER(2) = GAIN*(KAPPA - KAPPA_L)

     ENDIF

C======================================================================
C
C  -- End of the Standard TYR501 Stuff --
C
C======================================================================




C  -- Now the tyre modelling proper can start --
C  All forces calculated in SAE reference frame and transformed to TYDEX
format
C  afterwards - ease of continuity with previous model (also true of
reference
C  TYR501 model provided by MSC)
```

```
C  -- SAE Vertical Force like original tirsub calculations --

C     Normal Loads; simple calculations as with sample tirsub.f;
C     Penetrations to hub are not accounted for.

C  -- Calculate normal loads due to stiffness (always .LE. zero) --

      FZDEFL = -EFFPEN*TIREK

C  -- Calculate normal loads due to damping --

      FZDAMP = -VCPVRT*TIREC

C  -- Note the startup modification that was present in the tirsub
model is
C     no longer needed --



C  -- Sum for total normal force --

      FZ  =  MIN (0.0D0, (FZDEFL + FZDAMP) )



      IF ( IMODE .EQ. DYNAMIC ) THEN

C          Coefficient of friction as function of combined slip:

           U = U_MAX+SQRT(KAPPA_L**2+(TAN(ALPHA_L))**2)*(U_MIN-U_MAX)

C          Modify coefficient of friction based on road surface
C          factor:

           U = U * SURFAC


C  Longitudinal Loads

C  -- We're working in percent --

       SLIP=KAPPA*100

       IF(ABS(SLIP) .LE. ABS(SLIP_C)) THEN
```

```
C  -- Exponential Rise (1-e^-x) below critical slip ratio --

       FX = (1-EXP(-Ax*ABS(SLIP)/SLIP_C))*U*ABS(FZ)*SIGN(1.0D0,SLIP)

       ELSE

C  -- Linear Decay to Sliding Friction above critical slip ratio --

       FX = ABS(FZ)*(1-EXP(-Ax))*U*SIGN(1.0D0,SLIP)

       ENDIF


C  Lateral force and aligning torque (FY & TZ)

C  -- Scale Factor Diminished with Load FZ --

  B = By+(ABS(FZ)-R_LOAD)*dB_dFz


C  -- We're working in degrees --

  ALPHA_L=ALPHA_L*45/ATAN(1.0)

C  -- Don't let alpha go beyond 80 - the TAN functions go kinda wild --

  IF(ALPHA_L.GT.80.) THEN
     ALPHA_L = 80.0
  ENDIF
  IF(ALPHA_L.LT.-80.) THEN
     ALPHA_L = -80.0
  ENDIF

  IF(ABS(ALPHA_L) .LE. 1.D-10) THEN

     FY = 0.D0
     TZ = 0.D0

  ELSE IF( ABS(ALPHA_L) .LE. ALPHA_C ) THEN

C  -- As for longitudinal forces, Exponential Rise (1-e^-x) below
C     critical slip angle --

C  -- This line contains an even number of minus-sign errors --

      FY = (1-EXP(-Ay*ABS(ALPHA_L)/ALPHA_C))
   +        *U*B*FZ*SIGN(1.0D0,ALPHA_L)
```

```
  ELSE

C  -- As for longitudinal forces, Linear Decay to Sliding Friction
C     above critical slip ratio --

C     FY = FZ*U*B*SIGN(1.0D0,SLIP)*(1-(ABS(ALPHA_L)-ALPHA_C)/800)

C  -- Simplified - ADAMS handles transition from static to sliding
C     friction in the calling routine --

     FY = FZ*(1-EXP(-Ay))*U*B*SIGN(1.0D0,ALPHA_L)

   ENDIF



C  Aligning Torque based on intermediate FY excluding camber force.

C  -- Contact Patch Length --

  R1=UNLRAD
  R2=TIREW

  CP_LEN = (R1**2 - (R1-ABS(FZ)/TIREK)**2)**0.5 * 2.0

  IF(ABS(ALPHA_L) .GT. 1.D-10) THEN

    IF( ABS(ALPHA_L) .LE. ALPHA_C ) THEN

      TZ = -FY*CP_LEN/6*(1-ABS(ALPHA_L)/ALPHA_C)*PTRAILC

C  -- Divisor is because lever arm is not the entire contact patch
length. --

C  -- Parameter PTRAILC should be set to 1.0 for tyres with recetangular
C  contact patches (i.e. car tyres) and 0.5 for tyres with elliptical
C  contact patches (i.e. motorcycle tyres.) --

      ELSE

        TZ = 0.0

      ENDIF

      ENDIF
```

```
C    -- Add camber force to FY - "Taut String" --

C  DAH Sign of Camber Component Changed 13-11-00
C    FY  =  FY - FZ * TAN(GAMMA)

C        CAMBER=GAMMA


C    -- "Clipped" Camber model - improved limit behaviour --
C        DAH 10-01-00

C    -- THRSH represents aggression of departure at limit; high value
C       implies high limit & aggressive departure, lower value implies
C       progression.

C    -- was hard-coded, now user parameter
C    THRSH=0.8

     IF (ABS(GAMMA) .LT. ATAN(THRSH*U/C_GAMMA)) THEN

C     -- Camber term now held separate for aligning moment calculation

      FY_CAMBER = - FZ * TAN(GAMMA) * C_GAMMA



     ELSE

      CAMB_C=ATAN(THRSH*U/C_GAMMA)
      CAMB_INC=ABS(GAMMA)-CAMB_C

      A_INT=(1/(1-THRSH*C_GAMMA))/(COS(ATAN(THRSH*U/C_GAMMA)))**2
      B_INT=-(1-THRSH)*U*C_GAMMA

C     -- Needed when C_GAMMA is not equal to unity --

      C_INT= - FZ * TAN(CAMB_C) * C_GAMMA /
  &           (
  &           SIGN(1.,CAMB_C)*FZ*B_INT*(1-EXP(-A_INT*CAMB_INC)) -
  &           SIGN(1.,CAMB_C)*THRSH*U*FZ*C_GAMMA
  &           )
C     MUX=C_INT


      FY_CAMBER =SIGN(1.,GAMMA)*FZ*B_INT*(1-EXP(-A_INT*CAMB_INC)) -
  &               SIGN(1.,GAMMA)*THRSH*U*FZ*C_GAMMA * C_INT

      ENDIF
```

```
      FY = FY + FY_CAMBER
      FYWAS = FY


C     Mitigate FY depending on "Friction Ellipse"


      FR_ELLIP = (FX/(FZ*U))**2 + (FY/(FZ*U*B))**2



      X_SIGN = SIGN(1.0D0,FX)
      Y_SIGN = SIGN(1.0D0,FY)


      IF ( FR_ELLIP .GT. 1.0 ) THEN

         LSLIP=50.0
         USLIP=100.0
         ABSLIP=ABS(SLIP)


C     -- Friction Ellipse treatment for comprehensive slip below
C     critical slip ratio - revised over previous calculations
C     to preserve ratio of FX, FY but bring them inside the
C     friction ellipse --

      DIVISOR=1 + (FY/FX)**2/B**2

      FX1 = ( (U*FZ)**2 / DIVISOR )**0.5 * X_SIGN



C     -- Alternative term; longitudinal force is preserved at the
C     expense of lateral; seems intuitively more correct but
C     produces apparently poorer results. --
C     FX1 = FX

      FY1 =  ( ( 1-(FX1/(FZ*U))**2 ) * (FZ*U*B)**2 )**0.5*Y_SIGN




C     -- Revised formulation for highest slip ratios arrived at
C     by consideration of contact patch velocity.  Gives pleasing
```

```
C     results for wheels locked and wheels spinning cases.  Note
C     conversion of ALPHA from degrees back to radians for
C     this calculation and SLIP back from percent. --

      SLIPSQ = (SLIP/100)**2
      TAN_ALPHA_SQ=(TAN(ALPHA_L*ATAN(1.0)/45))**2

      DIVISOR=( 1 + TAN_ALPHA_SQ/SLIPSQ )

      FX2 = ( (U*FZ)**2 / DIVISOR )**0.5 * X_SIGN
      FY2 =  ((1-(FX2/(FZ*U))**2)*(FZ*U)**2)**0.5*Y_SIGN


C     -- Smear between two models using slip ratio --

      CALL STEP(ABSLIP,LSLIP,FX1,USLIP,FX2,IORD,FX,ERRFLG)
      CALL STEP(ABSLIP,LSLIP,FY1,USLIP,FY2,IORD,FY,ERRFLG)
C       FX=FX1
C       FY=FY1


C     -- Mitigate Camber forces too, for subsequent aligning moment
calculations

C     CALL STEP(ABSLIP,LSLIP,FY1,USLIP,FY2,IORD,FY_CAMBER,ERRFLG)

C     Is this right? Doesn't it significantly corrupt aligning torque for
C     a locking wheel at a high slip angle?

      FY_CAMBER=FY_CAMBER*FY/FYWAS


    ENDIF

C  C  -- The real MUX and MUY; all others are for debug only --
C  C      MUX = (FX/(FZ*U))
C  C      MUY = (FY/(FZ*U*B))




C    Rolling resistance moment (TY) as FIALA Tyre:


C    IF ( OMEGAR .GE. 0.0 ) THEN
C
C      TY = -CR * FZ
```

```
C
C    ELSE
C
C       TY =  CR * FZ
C
C    ENDIF

C    No need for loop above - velocity change below takes care of it

     TY =  CR * FZ




C    Compute righting moment due to lateral Contact Patch Shift (TX)
C    Use CA as "shape factor" to add to or subtract righting moment
C    from ADAMS' Toroidal assumption.  CA > 1 = fatter than toroid
C    CA < 1 = more like blade.

C    Lateral Contact Patch Shift clips at tyre extremity
C
C    Add aligning torque based on lateral offset of contact patch and
C    longitudinal forces to give "stand up under braking" behaviour for
C    motorcycles or tramlining for cars.

     PNOFFSET = 2 * GAMMA * R2/2 * (CA - 1)
     IF (ABS(PNOFFSET) .LT. R2/2 ) THEN
        TX = FZ * PNOFFSET
        TZ = TZ - FX * PNOFFSET
       ELSE
        TX = FZ * R2/2 * SIGN(1.,PNOFFSET)
        TZ = TZ - FX * R2/2  * SIGN(1.,PNOFFSET)
       ENDIF

C      Measured data shows evidence of significant "pneumatic lead" on
C      camber force data, aligning moment further modified to reflect
this.
C      Real data shows small dependency on load, some dependency on
camber
C      angle at low cambers; constant lead formulation neglects load
C      dependency and may overestimate torques at small cambers.
However,
C      camber forces are low and so torques are low too.

       TZ = TZ + FY_CAMBER*PLEAD
```

```
      ELSE
c  For static equilibrium zero the forces.

        FX  = ZERO_VAL
        FY  = ZERO_VAL
        TX  = ZERO_VAL
        TY  = ZERO_VAL
        TZ  = ZERO_VAL

      ENDIF



C======================================================================
C
C  -- After the tyre model giving forces & moments in SAE co-ordinates,
C  the long and arduous business of giving them back to ADAMS. Is this
C  *really* progress? --
C
C  -- Below here, all is standard TYR501 code except for sign mapping in
C  FCP and TCP --
C
C======================================================================


C  Apply the start-up transient smoothing and force the
C  all other tire forces to zero when the vertical force goes to
C  zero (e.g. when the tire is flying).

      FZMAG = DABS(FZ)

      CALL STEP(FZMAG,ZERO_VAL,ZERO_VAL,NO_FRC,ONE,
   +            0,OFF_GRND,ERRFLG)

      IF( IMODE .EQ. DYNAMIC ) THEN

        FCP(1) = -FX
        FCP(2) = FY
        FCP(3) = FZ

        TCP(1) = TX
        TCP(2) = TY
        TCP(3) = TZ
```

```
        ELSE


          FCP(1) = 0.0
          FCP(2) = 0.0
          FCP(3) = FZ

          TCP(1) = 0.0
          TCP(2) = 0.0
          TCP(3) = 0.0

        ENDIF

C  Transform the contact patch forces and moments to hub
C  coordinates:
C
C    Inputs:
C      FCP tire forces at contact patch in SAE contact patch
c          coordinates.
c
c      TCP Tire moments (torques) at contact patch in SAE
c          contact patch coordinates.
c
c      RAD Tire radius vector express in global (ground)
c          coordinates.
c
c      CPMTX Transformation from SAE contact patch coordinates
c          to ground.
c
c    Outputs:
c
c      F  tire forces at hub in global coordinates
c      T  Tire moments (torques) at hub (wheel center) in
c         global coordinates.
c
c      {F} = [CPMTX]{FCP}
c
c      {T} = [CPMTX]{TCP} + {RAD} X {F}
c
       CALL SVEC( RAD, URAD, RADIUS, 3 )
       CALL XCP2HB(FCP, TCP, RAD, CPMTX, F(1), F(4) )

C  Transformation of forces/torques from global to wheelcarrier
C  axes
       CALL M3T1(FORCES, TRAMAT, F(1) )
       CALL M3T1(TORQUE, TRAMAT, F(4) )

       XVEL=VEL(1)
       IF (XVEL .GE. 0.0) THEN
C         WRITE(*,*) 'Reverse'
          FORCES(1)=-FORCES(1)
```

```
          TORQUE(2)=-TORQUE(2)
C         WRITE(*,*) FORCES
C         WRITE(*,*) TORQUE
       ELSE
C         WRITE(*,*) 'Forwards'
C         WRITE(*,*) FORCES
C         WRITE(*,*) TORQUE
       ENDIF


C  *******************************************************
C  Assigning output quantities to VARINF array
C  *******************************************************

         IF(NVARS .GE. 75) THEN

C  Contact Patch Forces/Torques
C    VARINF in ISO coordinates, FCP and TCP in SAE coordinates
C    - Note the VARINF array is for examination inside A/View and
C      does not represent the forces passed back to the solver
C    - Arrays FORCES and TORQUE are the ones which influence
C      the solution

            VARINF(FX_ISO_PTR) =  FCP(1)
            VARINF(FY_ISO_PTR) = -FCP(2)
            VARINF(FZ_ISO_PTR) = -FCP(3)

            VARINF(MX_ISO_PTR) = TCP(1)
            VARINF(MY_ISO_PTR) = -TCP(2)
            VARINF(MZ_ISO_PTR) = -TCP(3)

C  Derivatives of state variables

            CALL COPYD(VARINF(DUDT_PTR), DEQDER(1), 2)

C  Slip quantities

C    Kinematic:

            VARINF(SLIPX_PTR) =KAPPA
            VARINF(SLIPI_PTR) =-ALPHA

C    Dynamic:
            VARINF(SLIPX_D_PTR) =  KAPPA_L
            VARINF(SLIPI_D_PTR) = -ALPHA_L
```

```
C  Friction coefficients

          VARINF(MUXTYR_PTR) = DABS(FRCLON/FCP(3))
          VARINF(MUYTYR_PTR) = DABS(FRCLAT/FCP(3))

C  Tire characteristics

          VARINF(PT_PTR) = 0.D0
          VARINF(MZR_PTR) = 0.D0
          VARINF(S_PTR) = 0.D0
          VARINF(SIGKPO_PTR) = 0.D0
          VARINF(SIGALO_PTR) = 0.D0
          VARINF(MGYR_PTR) = 0.D0
          VARINF(SVYKAP_PTR) = 0.D0
          VARINF(SVX_PTR) = 0.D0
          VARINF(SVY_PTR) = 0.D0

C  Contact Point

          CALL COPYD(VARINF(RCP1_PTR), RCP, 3)

C  Road Normal

          CALL COPYD(VARINF(RNORM1_PTR), RNORM, 3)

C  Surface Friction

          VARINF(SURFAC1_PTR) = SURFAC
          VARINF(SURFAC2_PTR) = SURFAC
          VARINF(SURFAC3_PTR) = SURFAC

C  Tire kinematics

          VARINF(CAMB_PTR)   = GAMMA
          VARINF(EFFPEN_PTR) = EFFPEN
          VARINF(VCPVRT_PTR) = VCPVRT
          VARINF(RADIUS_PTR) = RADIUS
          VARINF(VCPLON_PTR) = VCPLON
          VARINF(VCPLAT_PTR) = VCPLAT
          VARINF(VLON_PTR)   = VLON
        ELSE
          CALL ZERO(VARINF(1), NVARS)
          IERR = 2
          ERRMSG = 'TYR501: Incorrect Dimension on VARINF Array'
        ENDIF
```

```
C  ******************************************************
C  Use these values if tire is FLYING
C                              ^^^^^^

      ELSE
         CALL ZERO(VARINF(1), NVARS)
         IF(NVARS .GE. 75) THEN
           CALL COPYD(VARINF(DUDT_PTR), DEQDER(1), 2)
           VARINF(RADIUS_PTR) = UNLRAD
         ELSE
           CALL ZERO(VARINF(1), NVARS)
           IERR = 2
           ERRMSG = 'TYR501: Incorrect Dimension on VARINF Array'
         ENDIF
         ENDIF
      ELSE
         CALL ZERO(VARINF(1), NVARS)
      ENDIF

C  Error Handling

      IF(IERR .EQ. 0) THEN
        TYRMOD = 'TYR501 -> Harty Tyre Model '
      ELSE
        TYRMOD = ERRMSG(1:256)
      ENDIF
C
      RETURN


      END
```

RPF501

```
      SUBROUTINE RPF501( NCHTDS, CHTDST, TIR_ID, NTYPAR, TYPARR )

c  Copyright (C) 2000-1999
c  By Mechanical Dynamics, Inc. Ann Arbor, Michigan
c
c  All Rights Reserved, This code may not be copied or
c  reproduced in any form, in part or in whole, without the
c  explicit written permission of Mechanical Dynamics, Inc.
c

c DESCRIPTION:
c
c  Reads property file for user tire model based on
c  the Fiala Tire model and initializes the
c  tire parameter array (TYPARR).
```

```
c

c ARGUMENT LIST:
c
c     name   type  storage  use  description
c     ====== ====  =======  ===  ===============================
c     NCHTDS  I.S.    -       R    Number of characters in tire
c                                  property file name.
c     CHTDST  C.A.            R    Tire property file name
c     TIR_ID  I.S.    1       R    Tire GFORCE id
c     NTYPAR  I.S.    1       R    Dimension of TYPARR
c     TYPARR  D.A.  NTYPAR    E    Tire parameter array
c
c  *** Legend:  I integer           S scalar  R referenced
c               D double precision  A array   E evaluated
c               C character


C Inputs:
      INTEGER        NCHTDS
      CHARACTER*(*)  CHTDST
      INTEGER        TIR_ID
       INTEGER       NTYPAR

C  Outputs:

       DOUBLE RECISION TYPARR( NTYPAR )

C  Locals:

C  Units conversions:

       CHARACTER*(12)  UNITS(5)
       DOUBLE PRECISION CV2MDL(5)
       DOUBLE PRECISION CV2SI(5)
       DOUBLE PRECISION FCVSI
       DOUBLE PRECISION LCVSI
       DOUBLE PRECISION MCVSI
       DOUBLE PRECISION ACVSI
       DOUBLE PRECISION TCVSI

c  Fiala Property File Map

       INCLUDE 'tyrHarty_501.inc'

C  RTO variables:

       INTEGER          RETURN_VAL
       DOUBLE PRECISION TMPREAL
```

```
C  Shape Array RTO Stuff:

      DOUBLE PRECISION TMP1, TMP2
      INTEGER          N_NODES
      INTEGER          ARRPTR
      CHARACTER*80     FORM
      INTEGER FLEN
      CHARACTER*80     TABLE
      INTEGER TLEN

      LOGICAL          ERRFLG
      CHARACTER*80     MESSAG

c+--------------------------------------------------------------------*
c
c  Open the file:

      CALL RTO_OPEN_FILE_F2C ( CHTDST, NCHTDS, RETURN_VAL )

      ERRFLG = RETURN_VAL .EQ. 0
      MESSAG = 'Harty Tyre 501:  No Error opening tire property file.'
      CALL ERRMES ( ERRFLG, MESSAG, TIR_ID, 'STOP' )

c  Read [UNITS] block from property file:

c   Parameters in the property file may be given in any consistent
c   set of units.  The [UNITS] block identifies those units.
c   During evaluation, however, SI Units are used. So as parameters
c   are read from the property file they are converted
c   to SI units.

c   SI unit system.
c      LENGTH = meter
c      FORCE  = newton
c      ANGLE  = radians
c      MASS   = kg
c      TIME   = second

c   UNITS(1)-> FORCE  UNITS(2)-> MASS  UNITS(3)-> LENGTH
c   UNITS(4)-> TIME   UNITS(5)-> ANGLE

      CALL ATRTOU( TIR_ID, UNITS )
      CALL ACUNFN( UNITS, CV2MDL, CV2SI )

      FCVSI = CV2SI(1)
C   Force Conversion
      MCVSI = CV2SI(2)
C   Mass Conversion
      LCVSI = CV2SI(3)
```

```
C    Length Conversion
       TCVSI = CV2SI(4)
C    Time Conversion
       ACVSI = CV2SI(5)
C    Angle Conversion

C************************  TIRPRP POPULATION  ***********************

c  Read [MODEL] block:

       CALL RTO_READ_REAL_F2C
  .     (
  .       'MODEL', 5, 'USE_MODE', 8,
  .        TYPARR( USE_MODE ), RETURN_VAL
  .     )

       ERRFLG = RETURN_VAL .EQ. 0
       CALL ERRMES( ERRFLG ,
  .       'Harty Tyre 501:  No Use_mode?'
  .       ,TIR_ID,'STOP')


c  Read [DIMENSION] block:
       CALL RTO_READ_REAL_F2C
      . (
      .  'DIMENSION', 9, 'UNLOADED_RADIUS', 15,
      .   TMPREAL, RETURN_VAL
      . )

       ERRFLG = RETURN_VAL .EQ. 0
       CALL ERRMES( ERRFLG,
      .  'Harty Tyre 501:  No UNLOADED_RADIUS?'
      .  ,TIR_ID,'STOP')

       TYPARR( UNLOADED_RADIUS ) = TMPREAL  *  LCVSI

       CALL RTO_READ_REAL_F2C
      . (
      .  'DIMENSION', 9, 'WIDTH', 5,
      .   TMPREAL, RETURN_VAL
      . )

       ERRFLG = RETURN_VAL .EQ. 0
       CALL ERRMES( ERRFLG, 'Harty Tyre 501:  No WIDTH?'
  .                  ,TIR_ID,'STOP')

       TYPARR( WIDTH )  = TMPREAL  *  LCVSI
```

```
c  Read [PARAMETER] block

       CALL RTO_READ_REAL_F2C
      . (
      .  'PARAMETER', 9, 'VERTICAL_STIFFNESS', 18,
      .   TMPREAL, RETURN_VAL
      . )

       ERRFLG = RETURN_VAL .EQ. 0
C-------------------------------------------------------------------- ----
       CALL ERRMES( ERRFLG, 'Harty Tyre 501:  No VERTICAL_STIFFNESS?'
  .                 ,TIR_ID,'STOP')

       TYPARR( VERTICAL_STIFFNESS )  = TMPREAL * (FCVSI / LCVSI)

       CALL RTO_READ_REAL_F2C
      . (
      .  'PARAMETER', 9, 'VERTICAL_DAMPING', 16,
      .   TMPREAL, RETURN_VAL
      . )

       ERRFLG = RETURN_VAL .EQ. 0
       CALL ERRMES( ERRFLG, 'Harty Tyre 501:  No VERTICAL_DAMPING?'
  .                 ,TIR_ID,'STOP')

       TYPARR( VERTICAL_DAMPING ) = TMPREAL * (FCVSI * TCVSI / LCVSI)


       CALL RTO_READ_REAL_F2C
      . (
      .  'PARAMETER', 9, 'ROLLING_RESISTANCE', 18,
      .   TMPREAL, RETURN_VAL
      . )

       ERRFLG = RETURN_VAL .EQ. 0
       CALL ERRMES( ERRFLG,
      .  'Harty Tyre 501:  No ROLLING_RESISTANCE?',
      .   TIR_ID,'STOP')

       TYPARR( ROLLING_RESISTANCE )  = TMPREAL


       CALL RTO_READ_REAL_F2C
      . (
      .  'PARAMETER', 9, 'CMX', 3,
      .   TMPREAL, RETURN_VAL
      . )
```

```
 ERRFLG = RETURN_VAL .EQ. 0
 CALL ERRMES( ERRFLG,
.  'rpf501: CMX undefined.',
.    TIR_ID,'STOP')

 TYPARR( CMX )  = TMPREAL


 CALL RTO_READ_REAL_F2C
. (
.  'PARAMETER', 9, 'CGAMMA', 6,
.    TMPREAL, RETURN_VAL
. )

 ERRFLG = RETURN_VAL .EQ. 0
 CALL ERRMES( ERRFLG,
.  'Harty Tyre 501:  No CGAMMA?',
.    TIR_ID,'STOP' )

 TYPARR( CGAMMA ) = TMPREAL * (FCVSI / ACVSI)

 CALL RTO_READ_REAL_F2C
. (
.  'PARAMETER', 9, 'UMIN', 4,
.    TMPREAL, RETURN_VAL
. )

 ERRFLG = RETURN_VAL .EQ. 0
 CALL ERRMES( ERRFLG,
.  'Harty Tyre 501:  No UMIN?',
.    TIR_ID,'STOP')

 TYPARR( UMIN ) = TMPREAL


 CALL RTO_READ_REAL_F2C
. (
.  'PARAMETER', 9, 'UMAX', 4,
.    TMPREAL, RETURN_VAL
. )

 ERRFLG = RETURN_VAL .EQ. 0
 CALL ERRMES( ERRFLG,
.  'Harty Tyre 501:  No UMAX?',
.    TIR_ID,'STOP')

 TYPARR( UMAX ) = TMPREAL
```

```
 CALL RTO_READ_REAL_F2C
. (
.  'PARAMETER', 9, 'RELAXATION_LENGTH', 17,
.   TMPREAL, RETURN_VAL
. )

 ERRFLG = RETURN_VAL .EQ. 0
 CALL ERRMES( ERRFLG,
.  'Harty Tyre 501:  No RELAXATION_LENGTH?',
.   TIR_ID,'STOP')

 TYPARR( RELAXATION_LENGTH ) = ABS(TMPREAL * LCVSI)


 CALL RTO_READ_REAL_F2C
. (
.  'PARAMETER', 9, 'ALPHA_CRITICAL', 14,
.   TMPREAL, RETURN_VAL
. )

 ERRFLG = RETURN_VAL .EQ. 0
 CALL ERRMES( ERRFLG,
.  'Harty Tyre 501:  No ALPHA_CRITICAL?',
.   TIR_ID,'STOP')

 TYPARR( ALPHA_CRITICAL ) = ABS(TMPREAL * ACVSI)


 CALL RTO_READ_REAL_F2C
. (
.  'PARAMETER', 9, 'CURVATURE_FACTOR_ANGLE', 22,
.   TMPREAL, RETURN_VAL
. )

 ERRFLG = RETURN_VAL .EQ. 0
 CALL ERRMES( ERRFLG,
.  'Harty Tyre 501:  No CURVATURE_FACTOR_ANGLE?',
.   TIR_ID,'STOP')

 TYPARR( curvature_factor_angle ) = ABS(TMPREAL)


 CALL RTO_READ_REAL_F2C
. (
.  'PARAMETER', 9, 'SCALE_FACTOR_LATERAL', 20,
.   TMPREAL, RETURN_VAL
. )
```

```
 ERRFLG = RETURN_VAL .EQ. 0
 CALL ERRMES( ERRFLG,
.  'Harty Tyre 501:  No SCALE_FACTOR_LATERAL?',
.    TIR_ID,'STOP')

 TYPARR( SCALE_FACTOR_LATERAL ) = ABS(TMPREAL)


 CALL RTO_READ_REAL_F2C
. (
.  'PARAMETER', 9, 'RATED_LOAD', 10,
.    TMPREAL, RETURN_VAL
. )

 ERRFLG = RETURN_VAL .EQ. 0
 CALL ERRMES( ERRFLG,
.  'Harty Tyre 501:  No RATED_LOAD?',
.    TIR_ID,'STOP')

 TYPARR( rated_load ) = ABS(TMPREAL * MCVSI)


 CALL RTO_READ_REAL_F2C
. (
.  'PARAMETER', 9, 'SCALE_FACTOR_DIM', 16,
.    TMPREAL, RETURN_VAL
. )

 ERRFLG = RETURN_VAL .EQ. 0
 CALL ERRMES( ERRFLG,
.  'Harty Tyre 501:  No SCALE_FACTOR_DIM?',
.    TIR_ID,'STOP')

 TYPARR( scale_factor_dim ) = ABS(TMPREAL)


 CALL RTO_READ_REAL_F2C
. (
.  'PARAMETER', 9, 'SLIP_RATIO_CRITICAL', 19,
.    TMPREAL, RETURN_VAL
. )

 ERRFLG = RETURN_VAL .EQ. 0
 CALL ERRMES( ERRFLG,
.  'Harty Tyre 501:  No SLIP_RATIO_CRITICAL?',
.    TIR_ID,'STOP')
```

```
      TYPARR( slip_ratio_critical ) = ABS(TMPREAL)


      CALL RTO_READ_REAL_F2C
     . (
     .  'PARAMETER', 9, 'CURVATURE_FACTOR_RATIO', 22,
     .   TMPREAL, RETURN_VAL
     . )

      ERRFLG = RETURN_VAL .EQ. 0
      CALL ERRMES( ERRFLG,
     .  'Harty Tyre 501:  No CURVATURE_FACTOR_RATIO?',
     .   TIR_ID,'STOP')

      TYPARR( curvature_factor_ratio ) = ABS(TMPREAL)


      CALL RTO_READ_REAL_F2C
     . (
     .  'PARAMETER', 9, 'PNEUM_TRAILING_SCALING', 22,
     .   TMPREAL, RETURN_VAL
     . )

      ERRFLG = RETURN_VAL .EQ. 0
      CALL ERRMES( ERRFLG,
     .  'Harty Tyre 501:  No PNEUM_TRAILING_SCALING?',
     .   TIR_ID,'STOP')

      TYPARR( pneum_trailing_scaling ) = ABS(TMPREAL)


      CALL RTO_READ_REAL_F2C
     . (
     .  'PARAMETER', 9, 'PNEUMATIC_LEAD_CAMBER', 21,
     .   TMPREAL, RETURN_VAL
     . )

      ERRFLG = RETURN_VAL .EQ. 0
      CALL ERRMES( ERRFLG,
     .  'Harty Tyre 501:  No PNEUMATIC_LEAD_CAMBER?',
     .   TIR_ID,'STOP')

      TYPARR( pneumatic_lead_camber ) = ABS(TMPREAL * LCVSI)
```

```
      CALL RTO_READ_REAL_F2C
     . (
     .  'PARAMETER', 9, 'LIMIT_CAMBER_ONSET_FRIC', 23,
     .   TMPREAL, RETURN_VAL
     . )

      ERRFLG = RETURN_VAL .EQ. 0
      CALL ERRMES( ERRFLG,
     .  'Harty Tyre 501:  No LIMIT_CAMBER_ONSET_FRIC?',
     .   TIR_ID,'STOP')

      TYPARR( limit_camber_onset_fric ) = ABS(TMPREAL * ACVSI)

      n_nodes  = 0
      arrptr   = shape

C  READ [SHAPE] BLOCK IF IT EXISTS:

      CALL RTO_START_TABLE_READ_F2C
     . (
     .  'SHAPE', 5, FORM, FLEN, RETURN_VAL
     . )

      IF ( RETURN_VAL .EQ. 1 ) THEN

800      CONTINUE

         CALL RTO_READ_TABLE_LINE_F2C( TABLE, TLEN, RETURN_VAL )
         if ( return_val . eq. 1.and. tlen .gt. 3 ) then

          call act_line_parse (table, tmp1, tmp2, tlen)

          if ( n_nodes .lt. max_shape .and. tlen .eq. 2) then
               n_nodes  = n_nodes  + 1
               typarr( arrptr ) = tmp1
               typarr( arrptr + 1) = tmp2
               arrptr = arrptr + 2
            else

              if ( n_nodes .gt. max_shape) then
      CALL ERRMES( .true.,
     .  'Harty Tyre 501:  Shape table has more than 10 nodes',
     .   TIR_ID, 'STOP' )
              endif
```

```
               if (tlen .ne. 2) then
                  CALL ERRMES( .true.,
     .  'Harty Tyre 501:  Error parsing line of SHAPE table',
     .   TIR_ID, 'STOP' )
               endif

            endif

         goto 800

       endif

       typarr( n_shape ) = n_nodes

  else

call usrmes( .true.,
  .  'Harty Tyre 501:  No shape table.  Cylinder will be used'
  .  ,tir_id, 'WARN')

  endif

C  Close tire property file:

       CALL RTO_CLOSE_FILE_F2C ( CHTDST, NCHTDS, RETURN_VAL )

       ERRFLG = RETURN_VAL .EQ. 0
       MESSAG = 'exa_fiaini: Error closing tire property file.'
       CALL ERRMES( ERRFLG, MESSAG, TIR_ID, 'STOP' )

       RETURN
       END
```

Sample .TIR file

```
$---------------------------------------------------------------
------MDI_HEADER
[MDI_HEADER]
 FILE_TYPE     =  'tir'
 FILE_VERSION  =  2.0
 FILE_FORMAT   =  'ASCII'
(COMMENTS)
{comment_string}
'Tyre      - Dunlop 100/90 19 D401'
'Pressure  - Unknown'
'Test Date - Estimated DAH 2004'
```

```
'Harty Tire Model 2013'
'New File Format v2.1'
$---------------------------------------------------------------------
------units
[UNITS]
LENGTH = 'mm'
FORCE  = 'newton'
ANGLE  = 'radians'
MASS   = 'kg'
TIME   = 'sec'
$-------------------------------------------------------------- ---model
[MODEL]
$              use mode   1   2
$            ---------------------------------
$              smoothing     X
$
  PROPERTY_FILE_FORMAT = 'USER'
  FUNCTION_NAME        = 'HTire501_2013::TYR501'
  USE_MODE             = 2.0
$------------------------------------------------ -----------dimension
  [DIMENSION]
  UNLOADED_RADIUS = 341
  WIDTH = 100.0
$------------------------------------------------------ ------parameter
  [PARAMETER]
  VERTICAL_STIFFNESS = 146.0
  VERTICAL_DAMPING   = 0.2
  ROLLING_RESISTANCE = 0.02
  CMX  = 1.70
  CGAMMA   = 1.00
  UMIN     = 1.40
  UMAX     = 1.30
  RELAXATION_LENGTH = 100.0
  ALPHA_CRITICAL = 10.0
  CURVATURE_FACTOR_ANGLE = 2.70
  SCALE_FACTOR_LATERAL = 1.6417
  RATED_LOAD = 1662
  SCALE_FACTOR_DIM = -1.0E-4
  SLIP_RATIO_CRITICAL = 20.0
  CURVATURE_FACTOR_RATIO = 5.5
  PNEUM_TRAILING_SCALING = 1.25
  PNEUMATIC_LEAD_CAMBER = 20.0
  LIMIT_CAMBER_ONSET_FRIC = 0.80
$
```

Sample build file

```
@echo off
 rem for Intel Fortran 2013 and ADAMS 2013.1

 dir /b *.f > build.lst

 call  \msc.software\adams_x64\2013_1\common\mdi  cr-us  n  @build.lst
HTire501_2013_for_Adams2013_1.dll

 del build.lst
 copy HTire501_2013_for_Adams2013_1.dll
 "C:\MSC.Software\Adams_x64\2013_1\win64\HTire501_2013.dll"
```