```
module modconvg
!   ------------------------------------------------------------
!   TYPE SSPEC
!   vehicle specification
!   ------------------------------------------------------------
    type sspec
        real(8) mua                                                             ! air drag coefficient,
N/(km/h2)
        real(8) mur                                                             ! rolling resistance coeff,
N/kg
        real(8) rt                                                              ! tire radius, m
        real(8) tcl                                                             ! tire circumference, m
        real(8) bw, bh                                                          ! body width, height, m
        real(8) w0                                                              ! empty weight, kg
        real(8) wld                                                             ! load capacity, kg
        real(8) wt                                                              ! test weight, kg
        integer crew                                                            ! capacity, persons
    end type sspec

!   ------------------------------------------------------------
!   TYPE SENG
!   engine specification
!   ------------------------------------------------------------
    type sengine
        real(8) nex                                                             ! max speed, rpm
        real(8) nidle                                                           ! idling speed, rpm
        real(8) nes                                                             ! starting speed, rpm
    end type sengine

!   ------------------------------------------------------------
!   TYPE SGDAT
!   gear setting ( for each position )
!   ------------------------------------------------------------
    type sgdat
        real(8) gr                                                              ! ratio
        real(8) egr                                                             ! transmission efficiency
        real(8) dw                                                              ! rotating mass, kg
    end type sgdat

!   ------------------------------------------------------------
!   TYPE stransmission
!   transmission
!   ------------------------------------------------------------
    type stransmission
        integer grs                                                             ! starting gear
        integer ngr                                                             ! number of gear
        integer topgr                                                           ! top gear
        real(8) fgr                                                             ! final gear ratio
        real(8) efgr                                                            ! efficiency of final gear
        type (sgdat), pointer :: gri(:)                                         ! gear ratio
        real(8), pointer :: supv(:), sdownv(:), crelv(:)                        ! shiftup, down, clutch
release speed
    end type stransmission

!   ------------------------------------------------------------
!   TYPE STQCURVE
!   torque curves ( for maximam, frictional )
!   ------------------------------------------------------------
    type stqcurve
        integer ndata                                                           ! number of data
        real(8), pointer :: rev(:)                                              ! engine speed, rpm
        real(8), pointer :: tq(:)                                               ! engine torque, Nm
    end type stqcurve

end module

module version
    real(8), parameter :: VERNO = 1.4_8
end module

module subg
    interface
        subroutine readinput( spec, eng, tm, tqc , t, vdest, sp, n, inputf )
            use modconvg
            type (sspec) :: spec
            type (sengine) :: eng
            type (stqcurve) :: tqc
            type (stransmission) :: tm
            real(8), pointer :: vdest(:)
            integer, pointer :: t(:), sp(:)
            integer n
            character (len=*) :: inputf
        end subroutine

        subroutine readpattern ( uid, patname, n, t, vdest, sp )
            integer n, uid
            real(8), pointer :: vdest(:)
            integer, pointer :: t(:), sp(:)
            character (len=*) :: patname
        end subroutine

        subroutine reads ( uid, specf, spec, tm, eng )
            use modconvg
            type (sspec) :: spec
            type (sengine) :: eng
            type (stransmission) :: tm
            character (len=*) :: specf
            integer uid
        end subroutine

        subroutine readtqg ( uid, tqf, tqc )
            use modconvg
            integer uid
            character (len=*) :: tqf
            type (stqcurve) :: tqc
        end subroutine readtqg

        subroutine writeres ( n, t, vdest, vreal, s, ne, te, nne, nte, outf )
            real(8), pointer :: vdest(:)
            integer, pointer :: t(:), s(:)
            real(8) ne(:), te(:), vreal(:), nne(:), nte(:)
            integer n
            character (len=*) :: outf
        end subroutine

        subroutine setparameter ( spec, eng, tm )
            use modconvg
            type ( sspec ) :: spec
            type (sengine) :: eng
            type (stransmission) :: tm
        end subroutine

        recursive subroutine runmodeG ( i, n, vdest, vp, spp, evi, eti, vi, spi, verr, sww, ct, spec, tm,
eng, tqc, starting )
            use modconvg
            type (sspec) :: spec
            type (sengine) :: eng
            type (stqcurve) :: tqc
            type (stransmission) :: tm
            real(8), pointer :: vdest(:)
            integer, pointer :: sww(:)
            integer i, spp, spi, ct, n, starting
            real(8) vi, evi, eti, vp, verr
        end subroutine

        real(8) function maxtqlin( tqc, rev )
            use modconvg
            type (stqcurve) :: tqc
            real(8) rev
        end function maxtqlin

        function drvfrc ( spec, tm, sp, v1, v2, grade )
            use modconvg
            type (sspec) :: spec
            type (stransmission) :: tm
            integer sp
            real(8) drvfrc, v1, v2, grade
        end function drvfrc

        subroutine engstatG ( spec, tm, eng, evi, eti, s, dp, vp, vv, maxt, sw, tqc, starting )
```

```
            use modconvg
            type (sspec) :: spec
            type (stransmission) :: tm
            type (sengine) :: eng
            type (stqcurve) :: tqc
            real(8) evi, eti, fgr, dp, vp, vv, maxt
            integer s, sw, starting
        end subroutine

        subroutine sftdwnG ( i, vdest, vp, sp, spec, eng, tm, tqc, cnt, spout, minerr )
            use modconvg
            type (sspec) :: spec
            type (sengine) :: eng
            type (stransmission) :: tm
            type (stqcurve) :: tqc
            real(8), pointer :: vdest(:)
            integer i, sp, cnt, spout
            real(8) vp, minerr
        end subroutine

        subroutine sftupg( i, vdest, vp, sp, spec, eng, tm, tqc, cnt, spout, minerr )
            use modconvg
            type (sspec) :: spec
            type (sengine) :: eng
            type (stransmission) :: tm
            type (stqcurve) :: tqc
            real(8), pointer :: vdest(:)
            integer i, sp, cnt, spout
            real(8) vp, minerr
        end subroutine

        subroutine sftovrn( i, vdest, vp, sp, spec, eng, tm, tqc, spout, mkt )
            use modconvg
            type (sspec) :: spec
            type (sengine) :: eng
            type (stransmission) :: tm
            type (stqcurve) :: tqc
            real(8), pointer :: vdest(:)
            integer i, sp, spout, mkt
            real(8) vp
        end subroutine

        subroutine showinputdata( spec, eng, tm )
            use modconvg
            type (sspec) :: spec
            type (sengine) :: eng
            type (stransmission) :: tm
        end subroutine

    end interface

end module



! *************************************************************
! Conversion program for spark ignition engine (Fortran90)
! convG : main program
!
! *************************************************************
PROGRAM convG
    use modconvg
    use subg
    implicit none

    type (sspec) :: spec
    type (sengine) :: eng
    type (stqcurve) :: tqc
    type (stransmission) :: tm
    character (len=1024) :: inputf, outf
    real(8), pointer :: vdest(:)                                                ! target speed
    integer, pointer :: t(:), sp(:)                                             ! time, initial sp
    integer n                                                                   ! length of test cycle
    real(8) maxt                                                                ! maximum torque
    integer spp                                                                 ! past shift position
    real(8) vpast                                                               ! past speed
    real(8) verr                                                                ! velocity error
    real(8), allocatable :: ne(:), te(:), vreal(:)                              ! result, engine, torque,
speed
    real(8), allocatable :: nne(:), nte(:)                                      ! result, normalized
    integer, pointer :: s(:)                                                    ! result, shift
    integer, pointer :: sww(:)                                                  ! gear holded time
    integer i, ct, starting

!   ----- set initial condition -------------------------
    call GETARG( 1, inputf )
    call GETARG( 2, outf)

    call readinput( spec, eng, tm, tqc , t, vdest, sp, n, inputf )             ! read input data

    allocate ( vreal(n), ne(n), te(n), s(n) )                                   ! for calculation result
    allocate ( nne(n), nte(n), sww(0:n) )

    call setparameter ( spec, eng, tm )                                         ! parameter setting
    call showinputdata( spec, eng, tm )                                         ! show parameters

!   ----- follow pattern -------------------------------
    starting = 0
    vpast = vdest(1)
    spp = 0
    sww(0) = 3                                                                  ! initial holding time =
3sec

    do i = 1, n
        ct = 1                                                                  ! initialize recursion
counter
        verr = 0.0_8                                                            ! initialize velocity error

        call runmodeG ( i, n, vdest, vpast, spp, ne(i), te(i), vreal(i), s(i), verr, sww, ct, spec, tm,
eng, tqc, starting )

        maxt = maxtqlin ( tqc, ne(i) )                                          ! maximum torque
        nne(i) = ( ne(i) - eng%nidle ) / ( eng%nex - eng%nidle ) * 100.0_8      ! normalized engine speed
        nte(i) = te(i) / maxt * 100.0_8                                         ! normalized engine torque
        vpast = vreal(i)
        spp = s(i)
    end do

!   ----- output ---------------------------------------
    call writeres ( n, t, vdest, vreal, s, ne, te, nne, nte, outf )

    deallocate ( t, vdest, sp, sww )
    deallocate ( vreal, s, ne, te, nne, nte )

end program convG




! ************************************************************
! SUBROUTINE RUNMODEG
!  i        : time index
!  n        : number of test cycle data
!  vdest    : target speed
!  vpast    : past speed
!  sppast   : past shift position
!  evi      : engine speed
!  eti      : engine torque
!  vi       : output speed
!  spi      : output shift position
!  verr     : cumulative error
!  sww      : gear holded time
!  ct       : recursive counter
!  spec     : vehicle spec
!  tm       : transimission spec
!  eng      : engine spec
!  tqc      : torque curve
!  starting : starting switch
! ************************************************************
recursive subroutine runmodeG ( i, n, vdest, vpast, sppast, evi, eti, vi, spi, verr, sww, ct, spec, tm,
eng, tqc, starting )
    use modconvg
    use subg, only : engstatG, sftupG, sftovrn, sftdwnG
    implicit none

    type (sspec) :: spec
    type (sengine) :: eng
    type (stqcurve) :: tqc
    type (stransmission) :: tm
    real(8), pointer :: vdest(:)
    integer, pointer :: sww(:)
    integer n, spi, sppast, ct, starting, starting2
    integer s, i, j, ct2, mkt, spi2, sw, sw2, reacc
    real(8) vi, evi, eti, vpast, verr
    real(8) errmin, maxt, dv
    real(8) evi2, eti2, vi2, verr2

!   ----- end of recursive calculation ------------------------
    ct2 = 3 - ct
    if ( ct > 3 .or. i > n ) then
        spi = sppast
        verr = verr
        vi = vpast
        return
    end if

!   ----- stop ------------------------------------------------
    errmin = verr
    if ( vdest(i) <= 0.0_8 ) then
        vi = vdest(i)
        spi = 0
        evi = eng%nidle
        eti = 0.0_8
        verr= errmin + ( vdest(i-1) + vdest(i) ) / 2.0_8 - ( vi + vpast ) / 2.0_8
        sww(i) = 3                                                              ! enable change shift
        return
    end if

!   ----- vdest > 0 -------------------------------------------
    dv = vdest(i) - vpast                                                       ! acceleration km/h/s
    spi = sppast                                                                ! initial shift position
    reacc = 0

!   ----- neutral---------------------------------------------
    if ( sppast == 0 ) then
        if ( vpast == 0.0_8 ) then                                              ! starting
            spi = tm%grs
            starting = 1                                                        ! starting sw
            call engstatG ( spec, tm, eng, evi, eti, spi, vdest(i), vpast, vi, maxt, sw, tqc, starting )
            sww(i) = 3
            verr = errmin + ( vdest(i-1) + vdest(i) ) / 2.0_8 - ( vi + vpast ) / 2.0_8
            return                                                              ! stop recursion
        else
            if ( dv >= 0.0_8 ) then                                             ! re-acceleration
                spi = tm%topgr                                                  ! shift to top gear
(temporal)
                reacc = 1                                                       ! reacceleration sw
            else                                                                ! braking
                vi = vdest(i)                                                   ! v = target velocity
                spi = 0                                                         ! keep neutral
                evi = eng%nidle                                                 ! idling speed
                eti = 0.0_8                                                     ! engine torque = 0(Nm)
                verr = errmin + ( vdest(i-1) + vdest(i) ) / 2.0_8 - ( vi + vpast ) / 2.0_8
                sww(i) = 3                                                      ! enable shift change
                return                                                          ! stop recursion
            end if
        end if
    end if

!   ----- gear hold time < 3sec, or braking --------------------
    if ( sww(i-1) < 3 .or. dv < 0.0_8 ) then

        do s = spi, tm%topgr
            call engstatG ( spec, tm, eng, evi, eti, s, vdest(i), vpast, vi, maxt, sw, tqc, starting )
            verr2 = errmin + ( vdest(i-1) + vdest(i) ) / 2.0_8 - ( vi + vpast ) / 2.0_8
            if (sw /= 2) then
                exit                                                            ! overrun?
            end if
        end do

        if( s > tm%topgr ) s = tm%topgr

        sww(i) = sww(i-1) + 1
        starting2 =  starting
        call runmodeG ( i+1, n, vdest, vi, s, evi2, eti2, vi2, spi2, verr2, sww, ct+1, spec, tm, eng, tqc,
starting )
        verr = verr2
        spi = s

        if ( sw == 4 ) then                                                     ! sw=4 means clutch release
            sww(i) = 3                                                          ! enable change shift
            spi = 0                                                             ! neutral
        else if ( spi /= sppast )then
            sww(i) = 1                                                          ! update gear hold time
        else
            sww(i) = sww(i-1) + 1                                               ! update gear hold time
        end if
        return

    end if

!   ----- accelerate, or gear change permitted -----------------
    call engstatG ( spec, tm, eng, evi, eti, spi, vdest(i), vpast, vi, maxt, sw, tqc, starting )
    verr2 = ( vdest(i-1) + vdest(i) ) / 2.0_8 - ( vi + vpast ) / 2.0_8          ! calculate error

    if ( reacc ==1 ) then
!       ----- check shiftup condition ------------------------------
        call sftupG( i, vdest, vpast, spi, spec, eng, tm, tqc, ct2, spi2, verr2 )
        if ( spi2 /= spi .and. errmin >= verr + verr2) then
            spi = spi2                                                          ! shift-up
            errmin = verr + verr2                                               ! update error value
        end if

    else
!       ------ calculation of gear holded case ---------------------
        sww(i) = sww(i-1) + 1                                                   ! update gear hold time
        starting2 =  starting                                                   ! copy starting sw
        call runmodeG ( i+1, n, vdest, vi, spi, evi2, eti2, vi2, spi2, verr2, sww, ct+1, spec, tm, eng,
tqc, starting2 )
        errmin = verr2                                                          ! minimum error

!       ----- sw=0 normal condition --------------------------------
        if ( sw == 0 ) then                                                     ! check shiftup condition
            call sftupG( i, vdest, vpast, spi, spec, eng, tm, tqc, ct2, spi2, verr2 )
            if ( spi2 /= spi .and. errmin >= verr + verr2) then
                spi = spi2                                                      ! update gear position
                errmin = verr + verr2                                           ! update error
            end if

!       ----- sw=2 engine overrun ---------------------------------
        else if ( sw == 2 ) then                                                ! sw=2 means overrun
            if ( spi < tm%topgr ) then                                          ! calculation of shiftup
                call sftovrn( i, vdest, vpast, spi, spec, eng, tm, tqc, spi2, mkt )
                starting2 =  starting
                call engstatG( spec, tm, eng, evi, eti, spi2, vdest(i), vpast, vi, maxt, sw, tqc, starting2
)
                spi = spi2                                                      ! update shift position
                sww(i) = 1 + ( 3 - mkt )                                        ! update gear hold time
            else
                sww(i) = sww(i-1) + 1                                           ! case of top gear
            end if
            verr2 = verr + ( vdest(i-1) + vdest(i) ) / 2.0_8 - ( vi + vpast ) / 2.0_8

            call runmodeG ( i+1, n, vdest, vi, spi, evi2, eti2, vi2, spi2, verr2, sww, ct+1, spec, tm, eng,
tqc, starting )
            errmin = verr2
        end if

!       ----- calculation of shiftdown ( poor torque ) ------------
        if ( sw == 1 ) then
```

```
            call sftdwnG ( i, vdest, vpast, spi, spec, eng, tm, tqc, ct2, spi2, verr2 )
            if ( spi > spi2 .and. errmin >= verr + verr2 ) then                 ! error minimum?
                spi = spi2                                                      ! update gear position
                errmin = verr + verr2                                           ! update error
            end if
        end if

    end if

!   ----- final condition -------------------------------------
    if ( spi == 0 ) then
        vi = vdest(i)
        spi = 0
        evi = eng%nidle
        eti = 0.0_8
        return
    end if

    call engstatG ( spec, tm, eng, evi, eti, spi, vdest(i), vpast, vi, maxt, sw, tqc, starting )

    if ( dv < 0.0_8 .and. vi < tm%crelv(spi) ) then
        evi = eng%nidle                                                         ! clutch release
        eti = 0.0_8
        spi = 0
    end if

    if ( spi == sppast ) then                                                   ! update gear hold time
        sww(i) = sww(i-1) + 1
    else
        sww(i) = 1
    end if
    verr = verr + errmin                                                        ! update error

end subroutine runmodeG




! *************************************************************
! SUBROUTINE setparameter : set all parameters
!  spec  : vehicle spec
!  eng   : engine spec
!  tm    : transimission spec
! *************************************************************
SUBROUTINE setparameter ( spec, eng, tm )
    use modconvg
    implicit none

    type ( sspec ) :: spec
    type (sengine) :: eng
    type (stransmission) :: tm
    integer i, nloop
    real(8) maxs
    real(8), target :: supv(5), sdownv(4), crelv(5)
    DATA supv   /  0.0_8, 15.0_8, 30.0_8, 50.0_8, 70.0_8    /                   ! shift up speed (km/h)
    DATA sdownv / 10.0_8, 20.0_8, 40.0_8, 60.0_8            /                   ! upper limit when shift
down (km/h)
    DATA crelv  /  5.0_8, 10.0_8, 15.0_8, 20.0_8, 30.0_8    /                   ! clutch release speed
(km/h)

!   ----- start gear, shift condition -------------------------
    spec%tcl = 2.0_8 * spec%rt * 3.14_8                                         ! tire circumference (m)
    tm%grs = 1                                                                  ! initial start gear = 1
    if ( tm%ngr < 5 ) then
        nloop = tm%ngr - 1                                                      ! case of ngear < 5
    else
        nloop = 5 - 1                                                           ! case of ngear >= 5
    end if

    do i = tm%grs, nloop                                                        ! verify shiftup speed
        maxs = supv(i+1) / 3.6_8 / spec%tcl * 60.0_8 * tm%gri(i)%gr * tm%fgr
        if ( maxs > eng%nex ) then
            tm%grs = 2                                                          ! choose 2nd, when overrun
occurs
            exit
        end if
    end do

    tm%topgr = min ( tm%grs+4, tm%ngr )                                         ! top gear
    allocate ( tm%supv(tm%grs:tm%topgr) )
    allocate ( tm%sdownv(tm%grs:(tm%topgr-1)) )
    allocate ( tm%crelv(tm%grs:tm%topgr) )
    tm%supv =   supv                                                            ! shiftup condition
    tm%sdownv = sdownv                                                          ! shiftdown condition
    tm%crelv =  crelv                                                           ! clutch release condition

    eng%nes = 0.05_8 * ( eng%nex - eng%nidle ) + eng%nidle                      ! starting engine speed
(rpm)

!   ----- vehicle spec ----------------------------------------
    spec%wt= spec%w0 + spec%wld / 2.0_8 + 55.0_8                                ! test weight
    spec%mua = 2.99D-3 * spec%bw * spec%bh - 8.32D-4                            ! air drug
    spec%mur = 5.13D-3 + ( 17.6_8 / spec%wt )                                   ! rolling resistance
    do i = 1, tm%ngr                                                            ! rotational mass
        tm%gri(i)%dw = spec%w0 * ( 0.07_8 + 0.03_8 * tm%gri(i)%gr * tm%gri(i)%gr )
    end do

!   ----- transmission spec -----------------------------------
    tm%efgr = 0.95_8                                                            ! final efficiency
    do i = 1, tm%ngr
        if ( tm%gri(i)%gr == 1.0_8 ) then
            tm%gri(i)%egr = 0.98_8                                              ! direct
        else
            tm%gri(i)%egr = 0.95_8                                              ! other all gears
        end if
    end do

END SUBROUTINE setparameter




! ************************************************************
! SUBROUTINE ENGSTATG
!  calculate engine running condition
!  output engine speed, engine torque, maximum torque,
!  vehicle speed, engine condition
!  sw : return code
!   sw=0 engine speed is in range
!   sw=1 lack of torque
!   sw=2 engine overrun
!   sw=3 engine speed under limit
!   sw=4 clutch off
! ************************************************************
subroutine engstatG ( spec, tm, eng, evi, eti, s, dp, vp, vv, maxt, sw, tqc, starting )
    use modconvg
    use subg
    implicit none

    real(8), parameter :: pi = 3.14_8
    type ( sspec ) :: spec
    type ( stqcurve ) :: tqc
    type (sengine) :: eng
    type ( stransmission ) :: tm
    real(8) evi, eti, dp, vp, vv, maxt
    integer s, sw, starting

    real(8) dv, tqdif, df, ds
    integer flg

    dv = dp - vp
    if ( s == 0 ) then
        if (dv < 0.0_8 ) then
            evi = eng%nidle
            eti = 0.0_8
            vv = dp
            sw = 0
        else
            vv = 0.0_8
            evi = 0.0_8
            eti = 0.0_8
            sw = -1
        end if
        return
    end if

    vv = dp
    df = drvfrc ( spec, tm, s, vp, vv, 0.0_8 )
    eti = df * spec%tcl / ( 2.0_8 * pi * tm%gri(s)%gr * tm%fgr * tm%gri(s)%egr * tm%efgr )
    evi = ( vv * 60.0_8 ) * ( tm%gri(s)%gr * tm%fgr ) / ( 3.6_8 * spec%tcl )

!   ----- braking ---------------------------------------------
    if ( dv < 0.0_8 ) then
!       ----- neutral -----------------------------------------
        if ( s == 0 ) then
            evi = eng%nidle
            eti = 0.0_8
            sw = 0
!       ----- release clutch ----------------------------------
        else if ( dp < tm%crelv(s) ) then
            evi = eng%nidle
            eti = 0.0_8
            sw = 4
!       ----- normal condition --------------------------------
        else
            sw = 0
        end if
        maxt = maxtqlin ( tqc, evi )
        return
    end if

!   ----- acceleration ----------------------------------------
!   ----- starting mode ---------------------------------------
    if ( ( starting == 1 ) .and. ( evi < eng%nes ) .and. ( s == tm%grs ) ) then
        evi = eng%nes
        maxt = maxtqlin ( tqc, evi )

        if ( eti > maxt ) then
            sw = 1                                                              ! poor torque
        else
            sw = 0
            return
        end if
!   ----- over speed ------------------------------------------
    else if ( evi > eng%nex ) then
        starting = 0
        evi = eng%nex
        vv = evi * spec%tcl * 3.6_8 / ( 60.0_8 * tm%gri(s)%gr * tm%fgr )
        df = drvfrc ( spec, tm, s, vp, vv, 0.0_8 )
        eti = df * spec%tcl / ( 2.0_8 * pi * tm%gri(s)%gr * tm%fgr * tm%gri(s)%egr * tm%efgr )
        maxt = maxtqlin ( tqc, evi )
        if ( eti > maxt ) then
            sw = 1                                                              ! poor torque
        else
            sw = 2                                                              ! over speed
            if( s < tm%topgr ) then
                vv = 0.0_8                                                      ! no choise of gear
            end if
            return
        end if
!   ----- other cases -----------------------------------------
    else
        maxt = maxtqlin ( tqc, evi )
        if ( eti > maxt ) then
            sw = 1
        else
            sw = 0
            starting = 0
            return
        end if
    end if

!   ----- poor torque -----------------------------------------
    if ( eti > maxt ) then
        ds = 1.0_8
        flg = 0
        do while ( flg == 0 )
            df = drvfrc ( spec, tm, s, vp, vv, 0.0_8 )
            eti = df * spec%tcl / ( 2.0_8 * pi * tm%gri(s)%gr * tm%fgr * tm%gri(s)%egr * tm%efgr )
            evi = vv / 3.6_8 / spec%tcl * 60.0_8 * tm%gri(s)%gr * tm%fgr

            if ( ( starting == 1 ) .and. ( s == tm%grs ) .and. ( evi < eng%nes ) ) then
                maxt = maxtqlin ( tqc, eng%nes )
               evi = eng%nes
            else
                maxt = maxtqlin ( tqc, evi )
            end if

            tqdif = maxt - eti
            if ( ( tqdif < 1.0e-6 ) .and. ( tqdif >= 0.0_8 ) ) then
                flg = 1
            else if ( tqdif < 0.0_8 ) then
                vv = vv - ds
            else
                ds = ds / 2.0_8
                vv = vv + ds
            end if
        end do
    end if

    if ( ( starting == 1 ) .and. ( evi <= eng%nes ) .and. ( s == tm%grs ) ) then
        starting = 1
    else
        starting = 0
    end if
end subroutine




! *************************************************************
! FUNCTION MAXTQLIN, interpolation of engine torque
!  tqc      : torque curve
!  rev      : input speed rpm
!  maxtqlin : torque output Nm
! *************************************************************
real(8) function maxtqlin( tqc, rev )
    use modconvg
    implicit none

    type ( stqcurve ) :: tqc
    real(8) rev
    integer x

    if ( rev < tqc%rev(1) ) then
        x = 1                                                                   ! extrapolation(low)
    else if ( rev >= tqc%rev(tqc%ndata) ) then
        x = tqc%ndata -1                                                        ! extrapolation(high)
    else
        do x = 1, tqc%ndata
            if ( rev >= tqc%rev(x) .and. rev < tqc%rev(x+1) ) exit
        end do
    end if

    maxtqlin = tqc%tq(x) + ( tqc%tq(x+1)-tqc%tq(x) ) * ( rev-tqc%rev(x) ) / ( tqc%rev(x+1)-tqc%rev(x) )

end function maxtqlin




! *************************************************************
! FUNCTION DRVFRC, calculation of driving force
!  spec  : vehicle spec
!  tm    : transmission spec
!  sp    : shift position
!  v1    : initial speed (km/h)
!  v2    : target speed (km/h)
!  grade : no use
! *************************************************************
```

```fortran
real(8) function drvfrc ( spec, tm, sp, v1, v2, grade )
    use modconvg
    implicit none

    type ( sspec ) :: spec
    type ( stransmission ) :: tm
    integer sp
    real(8) v1, v2, grade, rr, mass, acc, fgrade

    rr =  9.8_8 * ( spec%mur * spec%wt + spec%mua * ( v2 * v2 ) )
    mass = spec%wt + tm%gri(sp)%dw
    acc = ( v2 - v1 ) / ( 3.6_8 * 1.0_8 )
    fgrade = spec%wt * dsin( datan( grade / 100.0_8 ) ) * 9.8_8

    drvfrc = rr + mass * acc  + fgrade

end function drvfrc




! ************************************************************
! SUBROUTINE SFTDWNG, downshift calculation
!  i      : index
!  vdest  : target speed
!  vp     : past speed
!  sp     : past shift position
!  spec   : vehicle spec
!  eng    : engine spec
!  tm     : transmission spec
!  tqc    : torque curve
!  cnt    : recursive counter
!  spout  : output shift posision
!  minerr : minimum error
! ************************************************************
subroutine sftdwnG ( i, vdest, vp, sp, spec, eng, tm, tqc, cnt, spout, minerr )
    use modconvg
    use subg
    implicit none

    real(8), pointer :: vdest(:)
    type (sspec) :: spec
    type (sengine) :: eng
    type (stransmission) :: tm
    type (stqcurve) :: tqc
    integer i, sp, cnt, spout, s, sw, x, starting
    real(8) vp, minerr, verr, vold, vnew, ev, et, maxt

    starting = 0
    spout = sp                                                                  ! initial shift position
    do s = tm%grs, sp - 1                                                       ! test shiftdown
        vold = vp
        verr = 0.0_8
        do x = 0, cnt                                                           ! running condition for cnt
sec.
            call engstatG( spec, tm, eng, ev, et, s, vdest(i+x), vold, vnew, maxt, sw, tqc, starting )

            if( sw == 2 .or. sw == 3 ) then
                exit                                                            ! overrun(2),
out-of-range(3)
            end if
            verr = verr + ( vdest(i+x) + vdest(i+x-1) ) / 2.0_8 - ( vnew + vold ) / 2.0_8
            vold = vnew                                                         ! update vehicle speed
        end do

        if( sw == 2 .or. sw == 3 ) then                                         ! overrun(2),
out-of-range(3)
            cycle
        else if ( spout == sp .or. minerr >= verr ) then
            spout = s                                                           ! update shift position
            minerr = verr                                                       ! update error
        end if
    end do

end subroutine sftdwnG
```

```
! ************************************************************
! SUBROUTINE SFTUPG, calculate shiftup
!  i     : index
!  vdest : target speed
!  vp    : past speed
!  sp    : past shift position
!  spec  : vehicle spec
!  eng   : engine spec
!  tm    : transmission spec
!  tqc   : torque curve
!  cnt   : recursive counter
!  spout : output shift posision
!  minerr : minimum error
! ************************************************************
subroutine sftupG( i, vdest, vp, sp, spec, eng, tm, tqc, cnt, spout, minerr )
    use modconvg
    use subg
    implicit none

    real(8), pointer :: vdest(:)
    type (sspec) :: spec
    type (sengine) :: eng
    type (stransmission) :: tm
    type (stqcurve) :: tqc
    integer i, sp, cnt, spout, s, sw, x, starting, sw2
    real(8) vp, minerr, verr, vold, vnew, ev, et, maxt

    spout = sp
    minerr = 0.0_8
    starting = 0

!   ----- calculation of shiftup ------------------------------
    do s = sp + 1, tm%topgr
        if ( vdest(i) < tm%supv(s) ) cycle                                      ! vdest < shiftup condition

        vold = vp
        verr = 0.0_8
        do x = 0, cnt                                                           ! running condition of next
cnt sec.
            call engstatG( spec, tm, eng, ev, et, s, vdest(i+x), vold, vnew, maxt, sw, tqc, starting )
            if( sw == 2 .or. sw == 3 ) then
                exit                                                            ! overrun(2),
out-of-range(3)
            end if
            verr = verr + ( vdest(i+x) + vdest(i+x-1) ) / 2.0_8 - ( vnew + vold ) / 2.0_8
            vold = vnew
        end do

        if( sw == 2 .or. sw == 3 ) then                                         ! overrun(2),
out-of-range(3)
            cycle
        else if ( spout == sp .or. (s>spout .and. minerr >= verr) ) then
            spout = s                                                           ! update shift position
            minerr = verr                                                       ! update error
        end if
    end do

!   ----- reacceleration -------------------------------------
    sw2 = 0
    do s = tm%grs, sp - 1                                                       ! start from starting gear
        if ( vdest(i) >= tm%sdownv(s) ) then
            cycle
        else
            sw2 = 1
            exit
        end if
    end do

    if ( sw2 == 1 ) then
        do s = s, tm%grs, -1
```

```
            if ( sw2 == 0 ) then
                if ( vdest(i) >= tm%sdownv(s) ) then
                    cycle
                else
                    sw2 = 1
                end if
            end if

            vold = vp
            verr = 0.0_8
            do x = 0, cnt                                                       ! running condition of next
cnt sec.
                call engstatG( spec, tm, eng, ev, et, s, vdest(i+x), vold, vnew, maxt, sw, tqc, starting )
                if( sw == 2 .or. sw == 3 ) then
                    exit                                                        ! overrun(2),
out-of-range(3)
                end if
                verr = verr + ( vdest(i+x) + vdest(i+x-1) ) / 2.0_8 - ( vnew + vold ) / 2.0_8
                vold = vnew                                                     ! update speed
            end do

            if( sw == 2 .or. sw == 3 ) then                                     ! overrun(2),
out-of-range(3)
                cycle                                                           ! skip impossible condition
            else if ( spout == sp .or. ( s<spout .and. minerr > verr) ) then
                spout = s                                                       ! update shift position
                minerr = verr                                                   ! update error
            end if
        end do
    end if
end subroutine sftupG




! *************************************************************
!  SUBROUTINE SFTOVRN, calculation of gear position
!  Case of engine overrun
!  i     : index
!  vdest : target speed
!  vp    : past speed
!  sp    : past shift position
!  spec  : vehicle spec
!  eng   : engine spec
!  tm    : transmission spec
!  tqc   : torque curve
!  cnt   : recursive counter
!  spout : output shift posision
!  mkt   : minimum gear hold time
! *************************************************************
subroutine sftovrn( i, vdest, vp, sp, spec, eng, tm, tqc, spout, mkt )
    use modconvg
    use subg
    implicit none

    real(8), pointer :: vdest(:)
    type (sspec) :: spec
    type (sengine) :: eng
    type (stransmission) :: tm
    type (stqcurve) :: tqc
    integer i, sp, spout, mkt
    integer x, y, z, sw, starting
    real(8) vp, verr, minerr, vold, vnew, maxt, ev, et

    starting = 0
    spout = sp
    do x = 3, 1, -1                                                             ! minimum gear hold time (3
to 1)
        minerr = 0.0_8
        do y = sp + 1, tm%topgr
            vold = vp
            verr = 0.0_8
            do z = 0, x - 1
                call engstatG( spec, tm, eng, ev, et, y, vdest(i+z), vold, vnew, maxt, sw, tqc, starting )
                if( sw == 2 .or. sw == 3 ) then
```

```
                    exit                                                        ! overrun(2), 
out-of-range(3)
                end if
                verr = verr + ( vdest(i+z) + vdest(i+z-1) ) / 2.0_8 - ( vnew + vold ) / 2.0_8
                vold = vnew
            end do

            if( sw == 2 .or. sw == 3 ) then                                     ! overrun(2), 
out-of-range(3)
                cycle                                                           ! skip this gear
            else if ( spout == sp ) then
                spout = y
                minerr = verr
                if ( minerr == 0.0_8 ) exit
            else if ( minerr > verr ) then
                spout = y
                minerr = verr
            end if
        end do
        if ( spout /= sp ) then
            mkt = x                                                             ! final gear hold time
            exit
        end if
    end do

end subroutine sftovrn



!***************************************************************
!  SHOWINPUTDATA, Display vehicle spec & input parameters
!  spec  : vehicle spec
!  eng   : engine spec
!  tm    : transimission spec
!***************************************************************
subroutine showinputdata( spec, eng, tm )
    use modconvg
    use version
    implicit none

    type (sspec) :: spec
    type (sengine) :: eng
    type (stransmission) :: tm
    integer i

    print '("[ VERSION ",F4.1," ]")', VERNO
    print*
    print '(" W0    =",F8.2, "[kg], Wtest =", F8.2, "[kg]")', spec%w0, spec%wt
    print '(" Width =",F8.3, "[m],  Height=", F8.3, "[m], Tire radius=", F8.3, "[m]")', spec%bw, spec%bh, 
spec%rt
    print '(" Crew =", I3)', spec%crew
    print*
    print '(" Nidle =", F8.2, "[rpm], Nex  =", F8.2, "[rpm]")', eng%nidle, eng%nex
    print '(" Nes   =", F8.2, "[rpm]")', eng%nes
    print '(" MuAir =", F10.6, " [kgf/(km/h)^2], MuRoll =", F10.6, " [kgf/kg]")', spec%mua, spec%mur
    print*
    print '(" Number of gear =", I3)', tm%ngr
    print '(" gear   ratio  efficiency  DW[kg]")'
    do i = 1, tm%ngr
        print '(I4, ": ", F8.3, F10.3, F12.3, F15.5)', i, tm%gri(i)%gr, tm%gri(i)%egr, tm%gri(i)%dw
    end do
    print '(" fin: ", F8.3, F10.3)', tm%fgr, tm%efgr
    print*

end subroutine showinputdata



! ************************************************************
! SUBROUTINE readinput
!  spec  : vehicle spec
!  eng   : engine spec
!  tm    : transimission spec
!  tqc   : torque curve
!  t     : time
!  vdest : target speed (km/h)
!  sp    : normal shift position
!  n     : number of pattern
! *************************************************************
subroutine readinput( spec, eng, tm, tqc , t, vdest, sp, n, inputf )
    use modconvg
    use subg
    implicit none

    type (sspec) :: spec
    type (sengine) :: eng
    type (stqcurve) :: tqc
    type (stransmission) :: tm
    character (len=*) :: inputf
    real(8), pointer :: vdest(:)
    integer, pointer :: t(:), sp(:)
    integer n
    character(len=1024) :: patf='', specf= '', tqf= ''

!   ----- read DATA file --------------------------------------
    if (inputf == '') then
        open ( 11, file = 'DATA', status = 'old', err=100 )
    else
        open ( 11, file = trim(inputf), status = 'old', err=100 )
    end if

    read ( 11, '(a)', err=110 ) patf                                             ! test cycle
    read ( 11, '(a)', err=110 ) specf                                            ! spec
    read ( 11, '(a)', err=110 ) tqf                                              ! torque curve
    close ( 11 )

!   ----- read input ------------------------------------------
    call readpattern ( 12, patf, n, t, vdest, sp )                               ! test cycle
    call reads ( 13, specf, spec, tm, eng )                                      ! spec
    call readtqg ( 14, tqf, tqc )                                                ! torque curve
    return

!   -----------------------------------------------------------
!   error
!   -----------------------------------------------------------
100 write ( 0, '(a)' ) " Error : Cannot open DATA file."
    stop
110 write ( 0, '(a)' ) " Error : Failed to read DATA file."
    stop

end subroutine




! *************************************************************
! SUBROUTINE readpattern, read test cycle
!  uid     : unit
!  patname : pattern filename
!  n       : number of data
!  t       : time
!  vdest   : target speed (km/h)
!  sp      : normal gear position
! *************************************************************
subroutine readpattern ( uid, patname, n, t, vdest, sp )
    implicit none

    character(len=*) :: patname
    real(8), pointer :: vdest(:)
    integer, pointer :: t(:), sp(:)
    integer uid, n, ios, i
    real(8) timeT, vdestT, shiftT
    character(len=1024) :: tmp

!   ----- count number of pattern data -------------------------
    open ( uid, file = patname, status = 'old', err=200 )
    read ( uid ,'(a)', err=210 ) tmp                                             ! label
    n = 0
    ios = 0
    do while ( ios == 0 )
        read ( uid, *, iostat=ios, err=210 ) timeT, vdestT, shiftT
        if ( ios == 0 ) n = n + 1
    end do

    allocate ( vdest(0:n), sp(0:n), t(1:n) )

!   ----- read data -------------------------------------------
    rewind (uid)
    read( uid ,'(a)', err=210 ) tmp                                              ! label
    do i = 1, n
        read ( uid, *, err=210 ) t(i), vdest(i), sp(i)                          ! test cycle
    end do
    close ( uid )

    vdest(0) = vdest(1)
    sp(0) = sp(1)

    return

!   -----------------------------------------------------------
!   error
!   -----------------------------------------------------------
200 write ( 0, '(3A)') " Error : Cannot open pattern file : [ ", trim( patname ), " ]"
    stop

210 write ( 0, '( A, i0, 3A )') " Error : Failed to read pattern data. Data No.= ", n+1, " in [ ",
trim(patname), " ]"
    stop

end subroutine readpattern




! ***************************************************************
! SUBROUTINE READS, read spec
!  uid   : unit
!  specf : spec filename
!  spec  : vehicle spec
!  eng   : engine spec
!  tm    : transmission spec
!  tqc   : torque curve
! ***************************************************************
subroutine reads ( uid, specf, spec, tm, eng )
    use modconvg
    implicit none

    type (sspec) :: spec
    type (sengine) :: eng
    type (stransmission) :: tm
    character (len=*) :: specf
    integer uid, i

    open ( uid, file = specf, status = 'old', err=300 )

    read ( uid, *, err=310 ) spec%w0                                            ! curb vehicle mass (kg)
    read ( uid, *, err=310 ) spec%wld                                           ! payload (kg)
    read ( uid, *, err=310 ) spec%crew                                          ! crew (persons) 55kgf /
1-passenger
    read ( uid, *, err=310 ) spec%bh                                            ! overall height (m)
    read ( uid, *, err=310 ) spec%bw                                            ! overall width (m)
    read ( uid, *, err=310 ) spec%rt                                            ! tire rolling radius (m)

    read ( uid, *, err=310 ) tm%ngr                                             ! number of gear
    allocate ( tm%gri( tm%ngr ) )                                               ! allocate gear ratio array
    do i = 1, tm%ngr
        read ( uid, *, err=310 ) tm%gri(i)%gr                                   ! read gear ratio data
    end do

    read ( uid, *, err=310 ) tm%fgr                                             ! final gear ratio
    read ( uid, *, err=310 ) eng%nidle                                          ! idling engine speed(rpm)
    read ( uid, *, err=310 ) eng%nex                                            ! governed engine speed
```

```
(rpm)

    close ( uid )
    return

!   -----------------------------------------------------------
!   error
!   -----------------------------------------------------------
300 write ( 0, '(3a)' ) " Error : Cannot open spec file : [ ", trim(specf), " ]"
    stop

310 write ( 0, '(A)' ) " Error : Failed to read spec data."
    stop

end subroutine reads




! **************************************************************
! SUBROUTINE READTQG
!  uid   : unit
!  tqf   : filename
!  tqc   : torque curve
! **************************************************************
subroutine readtqg ( uid, tqf, tqc )
    use modconvg
    implicit none

    type (stqcurve) :: tqc
    character (len=*) :: tqf
    character (len=1024) :: tmp
    integer uid, i, j, gap, ios
    real(8) revt, tmaxt

!   ----- count number of torque data -------------------------
    tqc%ndata = 0                                                                ! initialize
    open ( uid, file = tqf, status = 'old', err=400 )
    read ( uid,'(A)', err=410 ) tmp                                              ! skip header

    ios = 0                                                                      ! status
    do while ( ios == 0 )
        read ( uid, *, iostat = ios, err=410 ) revt, tmaxt                       ! test reading of data set
        if( ios == 0 ) tqc%ndata = tqc%ndata + 1                                 ! count n of data
    end do

!   ----- read torque data ------------------------------------
    allocate ( tqc%tq( tqc%ndata ), tqc%rev( tqc%ndata ) )

    rewind ( uid )                                                               ! move to head of file
    read ( uid,'(A)', err=410 ) tmp                                              ! skip header

    do i = 1, tqc%ndata
        read ( uid, *, err=410 ) tqc%rev(i), tqc%tq(i)                           ! read torque data
    end do

    close ( uid )

!   ----- sort by engine speed --------------------------------
    gap = ( tqc%ndata + 1 ) / 2
    do while ( gap >= 1 )
        do i = gap+1, tqc%ndata
            do j = i-gap, 1, -gap
                if ( tqc%rev(j) <= tqc%rev(j+gap) ) exit
                revt               = tqc%rev(j)
                tqc%rev( j )       = tqc%rev(j+gap)
                tqc%rev( j + gap ) = revt
                tmaxt              = tqc%tq( j )
                tqc%tq( j )        = tqc%tq( j + gap )
                tqc%tq( j + gap )  = tmaxt
            end do
        end do
        gap = gap / 2
    end do
    return

!   -----------------------------------------------------------
!   error
!   -----------------------------------------------------------
400 write (0,'(3A)') " Cannot open torque data file [ ", trim(tqf), " ]"
    stop
410 write (0,'(A,i0,3A)') " Failed to read torque data. Data No.= ", tqc%ndata+1, " in [ ", trim(tqf), " ]"
    close ( uid )
    stop

end subroutine readtqg



! **************************************************************
! SUBROUTINE WRITERES
!  n        : number of data
!  t        : time
!  vdest    : target speed
!  vreal    : calculated speed
!  s        : shift position
!  ne, te   : engine speed, torque
!  nne, nte : normalized speed, torque
! **************************************************************
subroutine writeres ( n, t, vdest, vreal, s, ne, te, nne, nte, outf )
    implicit none

    character, parameter :: ht = char(9)
    real(8), pointer :: vdest(:)
    integer, pointer :: t(:), s(:)
    real(8) ne(:), te(:), vreal(:), nne(:), nte(:)
    integer i, n
    character (len=*) :: outf

    if ( outf == '' ) then
        do while ( outf == '' )
            write( *, '(A,$)' ) 'Type filename for output : '
            read ( *, * ) outf
        end do
    end if

    open ( 15, file = trim(outf), status = 'unknown', err=500 )

    write ( 15,'(15A)', err=510 ) 'time(s)',ht,'Vtarget(km/h)',ht,          &
                        'Vreal(km/h)',ht,'Ne(rpm)', ht,'Te(N-m)',ht,        &
                        'N_norm(%)',ht,'T_norm(%)',ht,'Shift'
    do i = 1, n
        if ( te(i) < 0.0_8 ) then                                               ! motoring
            write ( 15, '(I0,2(A,F0.2),A,F0.1,3A,F0.2,3A,I0)', err=510 )    &
                    t(i), ht, vdest(i), ht, vreal(i), ht, ne(i), ht,        &
                    'M', ht, nne(i), ht, 'M', ht, s(i)
        else
            write ( 15, '(I0,2(A,F0.2),2(A,F0.1),2(A,F0.2),A,I0)', err=510 ) &
                    t(i), ht, vdest(i), ht, vreal(i), ht, ne(i), ht,         &
                    te(i), ht, nne(i), ht, nte(i), ht, s(i)
        end if
    end do

    close ( 15 )
    return

!   -----------------------------------------------------------
!   error
!   -----------------------------------------------------------
500 write (0,'(3A)') " Cannot open output file [ ", trim(outf), " ]"
    stop
510 write (0,'(A,i0,3A)') " Failed to write output data. Data No.= ", i, " in [ ", trim(outf), " ]"
    stop

end subroutine writeres
```