

```

*****
MODULE HVTRANSFC
  definition of data structure for engine, transmission,
  body , test cycle, fuel maps
*****
module hvtransfc

-----
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
  real(8) gvw      ! GVW, kg
  real(8) ric      ! highway traveling ratio(0 - 1)
  integer crew     ! capacity, persons
end type sspec

-----
TYPE SENG
  engine specification
-----
type seng
  real(8) nex      ! max speed, rpm
  real(8) nrate    ! rated speed, rpm
  real(8) nidle    ! idling speed, rpm
  real(8) nes      ! starting speed, rpm
  real(8) nec      ! clutch cut speed, rpm
  real(8), pointer :: nel(:) ! minimum speed, rpm(for each gear)
end type seng

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

-----
TYPE stransmission
  transmission
-----
type stransmission
  integer at       ! torque converter AT or not
  integer grs      ! starting gear
  integer grb      ! bottom gear
  integer skiplimit ! limit of skip shift
  integer ngr      ! number of gear (main x sub)
  integer nmgr     ! number of gear (main)
  integer nsgr     ! number of gear (sub)
  real(8) fgr      ! final ratio
  real(8) efgr     ! efficiency of final gear
  type (sgdat), pointer :: gri(:) ! array of gear setting
end type stransmission

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

-----
TYPE SCYCLE
  definition of test cycle for evaluation
-----
type scycle
  integer ndat     ! number of test cycle data
  real(8), pointer :: v(:) ! test cycle, km/h
  real(8), pointer :: grade(:) ! grade, %
end type scycle

-----
TYPE SFCMAP
  fuel map (measured data for input)
-----
type sfcmap
  real(8) fcidle   ! idling fuel consumption, L/h
  integer ndata    ! number of data point
  real(8), pointer :: rev(:) ! engine speed, rpm
  real(8), pointer :: tq(:)  ! torque, Nm
  real(8), pointer :: fc(:)  ! fuel consumption, l/h
end type sfcmap

```

```

! TYPE SGRIDMAP
! grid interpolated fuel map (calculated in the simulation)
! -----
type sgridmap
  real(8) fcidle                                ! idling fuel consumption, l/h
  integer nx                                    ! number of point in X-axis(eng speed)
  integer ny                                    ! number of point in Y-axis(torque)
  real(8), pointer :: rev(:)                   ! engine speed, rpm
  real(8), pointer :: tq(:)                    ! torque, Nm
  real(8), pointer :: fc(:, :)                ! fuel consumption, l/h
end type sgridmap
end module hvtransfc

```

```

module MODCONST
  implicit none
  real(8), parameter :: PDWT      = 0.07_8
  real(8), parameter :: PDWE      = 0.03_8
  real(8), parameter :: PMEET     = 0.05_8
  real(8), parameter :: PRELEASE  = 0.04_8
  real(8), parameter :: PSGW      = 55.00_8
  integer, parameter :: GRB       = 2
  integer, parameter :: THOLD     = 3
  integer, parameter :: SKIPLIMIT = 4
  real(8), parameter :: EGR_DIRECT = 0.98_8
  real(8), parameter :: EGR_ND    = 0.95_8
  real(8) :: nmins(1:4)
  real(8) :: tmup8t(1:3)
  real(8) :: tmunder8t(1:3)
  data nmins / 0.05_8, 0.11_8, 0.19_8, 0.26_8 /
  data tmup8t / 2.0_8, 1.7_8, 1.3_8 /
  data tmunder8t / 2.4_8, 1.7_8, 1.6_8 /
end module MODCONST

! rotational mass(tire) : WO*7%
! rotational mass(engine) : WO*3%
! crutch meet speed : 5%
! crutch release speed : 4%
! passenger's weight (kg)
! bottom gear
! minimum gear hold time (s)
! upper limit of skip shift
! transmission efficiency (connected directly)
! transmission efficiency
! lower limit of engine speed
! torque margin, GVW>=8t
! torque margin, GVW<8t

```

```

module TESTCYCLE
  integer, parameter :: nurban = 1830
  integer, parameter :: nhighway = 3120
  real(8), target :: vurban(0:nurban)
  real(8), target :: gurban(0:nurban)
  real(8), target :: vhighway(0:nhighway)
  real(8), target :: ghighway(0:nhighway)

```

```

DATA vurban / &
0.00_8, &
0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, &
0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, &
0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 4.19_8, 8.32_8, 12.33_8, 16.05_8, 18.74_8, &
20.28_8, 21.48_8, 23.13_8, 25.17_8, 27.19_8, 28.97_8, 30.43_8, 31.46_8, 32.24_8, 33.16_8, &
34.29_8, 35.40_8, 36.57_8, 38.08_8, 39.65_8, 40.59_8, 40.87_8, 41.03_8, 41.23_8, 41.24_8, &
41.15_8, 41.11_8, 41.02_8, 40.97_8, 41.25_8, 41.78_8, 42.20_8, 42.54_8, 42.96_8, 43.37_8, &
43.84_8, 44.73_8, 46.10_8, 47.57_8, 48.85_8, 49.89_8, 50.56_8, 50.81_8, 50.84_8, 50.87_8, &
50.88_8, 50.71_8, 50.31_8, 49.79_8, 49.16_8, 48.09_8, 46.37_8, 44.14_8, 41.46_8, 38.22_8, &
34.76_8, 31.55_8, 28.16_8, 23.82_8, 18.88_8, 14.51_8, 11.13_8, 8.59_8, 7.36_8, 8.01_8, &
9.99_8, 12.29_8, 14.48_8, 16.35_8, 17.11_8, 15.78_8, 12.39_8, 7.15_8, 1.80_8, 0.00_8, &
0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, &
0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, &
0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, &
0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, &
0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, &
0.00_8, 0.00_8, 0.00_8, 3.70_8, 8.97_8, 10.99_8, 11.48_8, 15.12_8, 20.34_8, 23.32_8, &
25.11_8, 27.74_8, 30.38_8, 32.93_8, 36.44_8, 39.59_8, 40.72_8, 41.41_8, 43.50_8, 44.40_8, &
45.24_8, 45.41_8, 45.17_8, 44.76_8, 44.36_8, 44.01_8, 43.54_8, 42.85_8, 42.35_8, 42.47_8, &
42.94_8, 43.20_8, 43.31_8, 43.57_8, 43.96_8, 44.49_8, 45.41_8, 46.55_8, 47.53_8, 48.52_8, &
49.86_8, 51.32_8, 52.56_8, 53.69_8, 54.81_8, 55.85_8, 56.88_8, 57.88_8, 58.67_8, 59.31_8, &
59.92_8, 60.14_8, 59.88_8, 59.70_8, 59.85_8, 59.86_8, 59.62_8, 59.59_8, 59.81_8, 59.79_8, &
59.49_8, 59.24_8, 59.05_8, 58.78_8, 58.53_8, 58.37_8, 58.22_8, 58.08_8, 58.06_8, 58.09_8, &
58.05_8, 57.89_8, 57.72_8, 57.61_8, 57.52_8, 57.37_8, 57.14_8, 56.80_8, 56.53_8, 56.71_8, &
57.39_8, 57.96_8, 57.98_8, 57.78_8, 57.82_8, 58.01_8, 58.06_8, 57.80_8, 56.98_8, 55.49_8, &
53.69_8, 51.95_8, 50.25_8, 48.70_8, 47.64_8, 47.06_8, 46.64_8, 46.30_8, 46.39_8, 47.18_8, &
48.55_8, 49.91_8, 50.85_8, 51.65_8, 52.81_8, 54.13_8, 55.10_8, 55.75_8, 56.29_8, 56.14_8, &
54.54_8, 51.61_8, 48.27_8, 45.40_8, 43.49_8, 42.66_8, 42.71_8, 43.29_8, 44.16_8, 45.28_8, &
46.64_8, 48.05_8, 49.42_8, 51.05_8, 52.97_8, 54.57_8, 55.57_8, 56.53_8, 57.67_8, 58.42_8, &
58.81_8, 59.56_8, 60.52_8, 60.89_8, 60.87_8, 61.27_8, 61.88_8, 62.11_8, 62.23_8, 62.39_8, &
61.87_8, 60.48_8, 59.06_8, 58.16_8, 57.46_8, 56.79_8, 56.36_8, 56.16_8, 56.09_8, 56.15_8, &
56.18_8, 56.00_8, 55.71_8, 55.60_8, 55.76_8, 56.26_8, 57.22_8, 58.37_8, 59.12_8, 59.37_8, &
59.53_8, 59.73_8, 59.74_8, 59.59_8, 59.56_8, 59.65_8, 59.86_8, 60.40_8, 61.23_8, 61.99_8, &
62.64_8, 63.32_8, 63.74_8, 63.61_8, 63.25_8, 62.88_8, 62.25_8, 61.48_8, 61.06_8, 60.78_8, &
60.00_8, 58.97_8, 58.32_8, 58.01_8, 57.65_8, 57.20_8, 56.65_8, 55.92_8, 55.27_8, 54.77_8, &
54.16_8, 53.49_8, 53.06_8, 52.74_8, 52.38_8, 52.25_8, 52.33_8, 52.21_8, 52.05_8, 52.32_8, &
52.64_8, 52.38_8, 51.61_8, 50.48_8, 48.76_8, 46.68_8, 44.77_8, 42.88_8, 40.60_8, 38.17_8, &
35.70_8, 32.76_8, 28.21_8, 23.82_8, 20.17_8, 16.37_8, 10.92_8, 4.99_8, 1.06_8, 0.00_8, &
0.00_8, 0.00_8, 1.78_8, 4.02_8, 7.51_8, 12.17_8, 16.29_8, 18.22_8, 19.22_8, 21.99_8, &
24.70_8, 26.87_8, 27.96_8, 28.32_8, 28.05_8, 27.45_8, 27.05_8, 26.82_8, 26.53_8, 26.69_8, &
27.80_8, 29.17_8, 29.87_8, 30.11_8, 30.63_8, 31.59_8, 32.84_8, 34.17_8, 35.18_8, 35.58_8, &
35.67_8, 36.07_8, 37.08_8, 38.37_8, 39.26_8, 39.60_8, 39.96_8, 40.58_8, 40.91_8, 40.73_8, &
40.53_8, 40.51_8, 40.37_8, 40.06_8, 39.76_8, 39.46_8, 39.41_8, 39.81_8, 39.89_8, 38.96_8, &
37.88_8, 37.95_8, 39.17_8, 40.68_8, 41.98_8, 43.09_8, 44.24_8, 45.66_8, 47.17_8, 48.25_8, &
48.61_8, 48.39_8, 47.83_8, 47.28_8, 46.95_8, 46.61_8, 46.14_8, 45.86_8, 45.89_8, 45.76_8, &
45.18_8, 44.31_8, 43.27_8, 41.85_8, 39.69_8, 36.81_8, 33.66_8, 30.55_8, 27.25_8, 23.77_8, &
20.85_8, 18.65_8, 16.41_8, 13.89_8, 11.80_8, 10.42_8, 9.38_8, 8.61_8, 8.14_8, 7.47_8, &
6.43_8, 4.35_8, 2.49_8, 1.27_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, &
0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, &
0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, &
0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, &
0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, &
0.00_8, 3.37_8, 9.10_8, 14.02_8, 17.20_8, 20.22_8, 23.49_8, 26.43_8, 28.90_8, 30.55_8, &
31.17_8, 31.42_8, 31.48_8, 30.84_8, 29.90_8, 29.66_8, 29.20_8, 28.45_8, 27.40_8, 26.21_8, &
25.27_8, 24.81_8, 24.97_8, 26.03_8, 27.81_8, 29.48_8, 30.48_8, 30.85_8, 30.59_8, 29.84_8, &
28.92_8, 27.47_8, 24.78_8, 21.41_8, 18.66_8, 16.85_8, 15.79_8, 16.08_8, 18.06_8, 21.01_8, &
24.26_8, 27.72_8, 31.07_8, 33.82_8, 35.90_8, 37.26_8, 37.71_8, 37.50_8, 37.07_8, 36.47_8, &
35.57_8, 34.41_8, 33.12_8, 31.87_8, 30.79_8, 29.85_8, 28.93_8, 28.08_8, 27.60_8, 28.02_8, &
29.68_8, 31.96_8, 33.94_8, 35.57_8, 37.21_8, 38.51_8, 39.39_8, 40.58_8, 42.20_8, 43.44_8, &
44.19_8, 44.96_8, 45.73_8, 46.29_8, 46.87_8, 47.51_8, 48.07_8, 48.82_8, 49.85_8, 50.68_8, &
51.26_8, 52.04_8, 52.82_8, 53.22_8, 53.53_8, 54.00_8, 54.31_8, 54.35_8, 54.37_8, 54.28_8, &
53.91_8, 53.18_8, 51.82_8, 49.83_8, 47.71_8, 45.39_8, 41.80_8, 37.47_8, 33.19_8, 30.27_8, &
26.16_8, 19.57_8, 13.81_8, 11.04_8, 9.11_8, 6.17_8, 3.13_8, 1.17_8, 0.00_8, 0.00_8, &
0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, &
0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, &
0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, 0.00_8, &

```

3.83.8, 9.38.8, 13.85.8, 14.91.8, 15.68.8, 19.52.8, 24.58.8, 27.20.8, 27.48.8, 27.85.8, &
29.15.8, 31.13.8, 33.52.8, 35.89.8, 37.09.8, 37.33.8, 37.10.8, 36.30.8, 35.03.8, 34.21.8, &
34.23.8, 34.31.8, 33.99.8, 33.82.8, 34.34.8, 35.49.8, 37.22.8, 39.53.8, 41.98.8, 44.08.8, &
45.69.8, 46.78.8, 47.45.8, 47.84.8, 47.82.8, 47.14.8, 46.06.8, 45.13.8, 44.55.8, 44.41.8, &
44.84.8, 45.56.8, 45.84.8, 45.28.8, 43.79.8, 41.57.8, 39.00.8, 36.35.8, 33.60.8, 30.97.8, &
28.86.8, 27.00.8, 24.95.8, 23.05.8, 21.71.8, 20.52.8, 19.39.8, 19.06.8, 19.70.8, 20.50.8, &
20.95.8, 21.18.8, 21.19.8, 20.66.8, 19.26.8, 16.67.8, 13.34.8, 10.48.8, 8.59.8, 6.93.8, &
4.36.8, 2.09.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
1.05.8, 5.67.8, 9.44.8, 13.24.8, 16.38.8, 18.36.8, 19.93.8, 22.25.8, 25.25.8, 28.34.8, &
31.32.8, 33.95.8, 35.96.8, 37.89.8, 40.21.8, 42.12.8, 42.93.8, 43.53.8, 44.80.8, 46.02.8, &
46.29.8, 46.15.8, 46.42.8, 47.03.8, 47.57.8, 48.10.8, 48.68.8, 49.16.8, 49.56.8, 50.16.8, &
50.97.8, 51.75.8, 52.42.8, 53.00.8, 53.38.8, 53.57.8, 53.70.8, 53.61.8, 53.06.8, 52.29.8, &
51.78.8, 51.48.8, 50.93.8, 49.93.8, 48.45.8, 46.42.8, 43.97.8, 41.48.8, 39.39.8, 38.18.8, &
38.09.8, 38.70.8, 39.19.8, 39.06.8, 38.27.8, 37.02.8, 35.67.8, 34.61.8, 33.89.8, 33.32.8, &
32.62.8, 31.41.8, 29.63.8, 27.83.8, 26.44.8, 25.40.8, 24.84.8, 25.24.8, 26.34.8, 27.09.8, &
27.12.8, 27.01.8, 27.21.8, 27.70.8, 28.48.8, 29.54.8, 30.60.8, 31.61.8, 32.80.8, 34.11.8, &
35.20.8, 36.10.8, 37.13.8, 38.13.8, 38.62.8, 38.60.8, 38.48.8, 38.23.8, 37.40.8, 35.99.8, &
34.45.8, 33.07.8, 31.81.8, 30.59.8, 29.40.8, 28.40.8, 27.63.8, 26.57.8, 24.25.8, 20.69.8, &
14.60.8, 8.99.8, 4.76.8, 1.64.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
0.00.8, 3.57.8, 8.28.8, 11.75.8, 13.06.8, 15.07.8, 18.64.8, 21.16.8, 22.19.8, 22.89.8, &
23.73.8, 23.37.8, 22.87.8, 22.73.8, 22.51.8, 22.01.8, 21.45.8, 21.23.8, 22.02.8, 23.88.8, &
25.74.8, 26.82.8, 27.78.8, 29.33.8, 31.26.8, 33.32.8, 35.53.8, 37.60.8, 39.26.8, 40.64.8, &
41.70.8, 42.23.8, 42.50.8, 42.75.8, 42.61.8, 41.89.8, 40.86.8, 39.56.8, 37.87.8, 36.03.8, &
34.13.8, 31.63.8, 27.79.8, 22.97.8, 18.01.8, 13.36.8, 9.31.8, 6.70.8, 5.31.8, 3.98.8, &
2.54.8, 1.40.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 2.62.8, 3.82.8, 4.08.8, 6.12.8, &
8.81.8, 9.73.8, 9.59.8, 9.44.8, 9.45.8, 9.35.8, 9.30.8, 9.75.8, 10.70.8, 11.61.8, &
12.02.8, 12.02.8, 11.71.8, 10.78.8, 9.34.8, 6.66.8, 4.63.8, 3.28.8, 1.70.8, 0.00.8, &
0.00.8, 0.00.8, 0.00.8, 0.00.8, 2.43.8, 4.63.8, 7.93.8, 9.13.8, 10.21.8, 11.28.8, &
12.87.8, 14.44.8, 15.28.8, 15.41.8, 15.33.8, 15.28.8, 14.97.8, 14.23.8, 13.70.8, 14.26.8, &
15.77.8, 17.25.8, 18.21.8, 18.82.8, 19.00.8, 18.44.8, 17.29.8, 16.12.8, 15.00.8, 13.52.8, &
11.83.8, 10.76.8, 10.49.8, 10.04.8, 8.94.8, 8.11.8, 8.15.8, 8.24.8, 7.77.8, 7.65.8, &
8.64.8, 10.04.8, 10.94.8, 11.29.8, 11.36.8, 11.01.8, 10.01.8, 8.54.8, 7.13.8, 6.41.8, &
6.79.8, 8.38.8, 10.73.8, 12.83.8, 14.04.8, 14.97.8, 16.40.8, 18.03.8, 19.52.8, 21.53.8, &
24.25.8, 26.42.8, 27.30.8, 27.75.8, 28.38.8, 28.62.8, 28.01.8, 26.91.8, 25.46.8, 23.49.8, &
20.45.8, 17.47.8, 14.80.8, 12.03.8, 9.34.8, 7.27.8, 5.43.8, 3.23.8, 1.22.8, 0.00.8, &
0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
0.00.8, 1.92.8, 3.93.8, 6.80.8, 9.57.8, 12.26.8, 13.88.8, 14.61.8, 15.12.8, 15.52.8, &
15.14.8, 13.51.8, 11.06.8, 8.82.8, 7.51.8, 7.24.8, 7.54.8, 7.69.8, 7.12.8, 5.85.8, &
3.90.8, 2.23.8, 1.49.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
0.00.8, 1.08.8, 1.34.8, 3.04.8, 3.84.8, 4.07.8, 5.12.8, 7.12.8, 9.07.8, 10.25.8, &
10.65.8, 10.61.8, 10.78.8, 11.61.8, 12.65.8, 13.20.8, 13.16.8, 12.95.8, 12.77.8, 12.50.8, &
12.07.8, 11.66.8, 11.35.8, 10.77.8, 9.56.8, 8.03.8, 6.72.8, 5.73.8, 4.94.8, 4.46.8, &
4.29.8, 4.15.8, 3.85.8, 3.31.8, 2.49.8, 1.33.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
1.28.8, 1.60.8, 2.63.8, 5.02.8, 8.68.8, 12.57.8, 15.07.8, 16.22.8, 17.46.8, 19.65.8, &
20.82.8, 21.47.8, 22.09.8, 22.09.8, 20.95.8, 18.99.8, 16.56.8, 14.08.8, 12.39.8, 11.84.8, &
11.86.8, 12.11.8, 13.01.8, 14.67.8, 16.56.8, 18.29.8, 20.07.8, 22.45.8, 25.37.8, 27.84.8, &
29.36.8, 30.76.8, 32.49.8, 33.61.8, 33.67.8, 33.55.8, 33.29.8, 32.04.8, 30.09.8, 28.23.8, &
26.18.8, 23.77.8, 22.06.8, 21.48.8, 21.25.8, 21.09.8, 21.08.8, 20.47.8, 18.82.8, 16.86.8, &
14.85.8, 11.76.8, 8.45.8, 5.33.8, 3.78.8, 2.45.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 1.86.8, &
6.31.8, 9.90.8, 12.02.8, 13.52.8, 15.04.8, 14.83.8, 13.43.8, 12.27.8, 12.79.8, 14.79.8, &
16.84.8, 18.64.8, 20.87.8, 23.02.8, 24.13.8, 24.60.8, 24.92.8, 24.67.8, 23.86.8, 22.97.8, &
21.50.8, 19.10.8, 16.70.8, 15.04.8, 13.91.8, 13.35.8, 13.40.8, 13.35.8, 12.77.8, 11.82.8, &
9.99.8, 7.19.8, 5.07.8, 4.85.8, 5.29.8, 4.82.8, 3.66.8, 1.87.8, 0.00.8, 0.00.8, &
0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, &
0.00.8, 0.00.8, 0.00.8, 0.00.8, 0.00.8, 3.50.8, 5.08.8, 5.97.8, 9.46.8, 13.96.8, &
15.88.8, 16.84.8, 19.06.8, 21.53.8, 23.63.8, 25.88.8, 28.25.8, 30.55.8, 32.83.8, 34.81.8, &
36.22.8, 37.19.8, 38.01.8, 38.69.8, 39.31.8, 40.16.8, 41.24.8, 42.33.8, 43.38.8, 44.56.8, &
45.85.8, 47.02.8, 47.93.8, 48.80.8, 49.73.8, 50.57.8, 51.32.8, 52.19.8, 53.16.8, 53.98.8, &
54.72.8, 55.55.8, 56.47.8, 57.48.8, 58.69.8, 60.00.8, 61.20.8, 62.42.8, 63.75.8, 65.05.8, &
66.16.8, 67.12.8, 67.89.8, 68.54.8, 69.22.8, 69.98.8, 70.71.8, 71.47.8, 72.36.8, 73.35.8, &
74.41.8, 75.52.8, 76.52.8, 77.39.8, 78.29.8, 79.22.8, 79.95.8, 80.45.8, 80.88.8, 81.25.8, &
81.56.8, 81.81.8, 81.86.8, 81.66.8, 81.19.8, 80.68.8, 80.44.8, 80.39.8, 80.29.8, 80.21.8, &
80.19.8, 80.03.8, 79.63.8, 79.25.8, 79.09.8, 79.08.8, 79.01.8, 78.84.8, 78.61.8, 78.44.8, &
78.34.8, 78.23.8, 78.15.8, 78.19.8, 78.28.8, 78.34.8, 78.46.8, 78.72.8, 79.03.8, 79.30.8, &
79.61.8, 79.99.8, 80.39.8, 80.75.8, 81.08.8, 81.39.8, 81.73.8, 82.05.8, 82.37.8, 82.74.8, &
83.10.8, 83.34.8, 83.46.8, 83.51.8, 83.42.8, 83.22.8, 83.08.8, 82.97.8, 82.72.8, 82.41.8, &
82.17.8, 81.84.8, 81.34.8, 80.89.8, 80.63.8, 80.42.8, 80.17.8, 79.93.8, 79.67.8, 79.45.8, &
79.42.8, 79.50.8, 79.50.8, 79.53.8, 79.72.8, 79.88.8, 79.81.8, 79.69.8, 79.75.8, 79.95.8, &
80.24.8, 80.68.8, 81.25.8, 81.84.8, 82.39.8, 82.90.8, 83.42.8, 83.92.8, 84.34.8, 84.67.8, &
84.94.8, 85.12.8, 85.09.8, 84.86.8, 84.51.8, 84.09.8, 83.66.8, 83.30.8, 82.94.8, 82.54.8, &
82.18.8, 81.96.8, 81.86.8, 81.85.8, 81.82.8, 81.64.8, 81.37.8, 81.15.8, 80.89.8, 80.50.8, &
80.25.8, 80.39.8, 80.83.8, 81.44.8, 82.31.8, 83.38.8, 84.39.8, 85.24.8, 86.00.8, 86.67.8, &
87.20.8, 87.55.8, 87.60.8, 87.39.8, 87.10.8, 86.87.8, 86.62.8, 86.35.8, 86.17.8, 85.99.8, &
85.77.8, 85.59.8, 85.51.8, 85.45.8, 85.43.8, 85.61.8, 85.99.8, 86.30.8, 86.50.8, 86.50.8, &
86.57.8, 86.66.8, 86.79.8, 86.98.8, 87.08.8, 86.85.8, 86.16.8, 85.28.8, 84.52.8, 83.98.8, &
83.51.8, 83.10.8, 82.77.8, 82.60.8, 81.91.8, 80.94.8, 79.82.8, 78.50.8, 77.00.8, 75.57.8, &

74.34.8, 73.14.8, 71.88.8, 70.73.8, 69.59.8, 67.81.8, 64.91.8, 60.93.8, 56.12.8, 50.87.8, &
45.70.8, 40.78.8, 35.82.8, 30.85.8, 26.48.8, 23.12.8, 20.59.8, 18.47.8, 16.69.8, 15.82.8, &
15.57.8, 15.98.8, 17.14.8, 18.68.8, 20.11.8, 21.30.8, 22.22.8, 22.50.8, 22.13.8, 21.85.8, &
22.02.8, 22.17.8, 21.86.8, 21.08.8, 19.50.8, 16.78.8, 13.55.8, 11.03.8, 9.72.8, 9.38.8, &
9.55.8, 9.71.8, 9.65.8, 9.80.8, 10.85.8, 12.80.8, 15.13.8, 17.67.8, 20.63.8, 23.74.8, &
26.17.8, 27.49.8, 28.09.8, 28.36.8, 28.16.8, 27.31.8, 26.07.8, 24.71.8, 23.29.8, 22.00.8, &
20.98.8, 19.93.8, 18.57.8, 17.29.8, 16.69.8, 17.09.8, 17.92.8, 19.14.8, 20.34.8, &
21.10.8, 21.30.8, 21.06.8, 20.63.8, 20.33.8, 20.44.8, 20.97.8, 21.60.8, 21.76.8, 21.39.8, &
21.23.8, 21.63.8, 21.90.8, 21.45.8, 20.74.8, 20.26.8, 19.76.8, 19.11.8, 18.79.8, 18.97.8, &
19.31.8, 19.90.8, 21.06.8, 22.54.8, 23.80.8, 24.79.8, 25.59.8, 26.01.8, 25.83.8, 25.26.8, &
24.73.8, 24.39.8, 23.94.8, 23.30.8, 23.10.8, 23.72.8, 24.49.8, 24.77.8, 25.01.8, 25.58.8, &
25.92.8, 25.88.8, 26.08.8, 26.44.8, 26.17.8, 25.39.8, 24.87.8, 24.61.8, 24.22.8, 23.93.8, &
24.01.8, 24.00.8, 23.27.8, 22.03.8, 21.23.8, 21.51.8, 22.53.8, 23.61.8, 24.63.8, 25.66.8, &
26.14.8, 25.76.8, 25.08.8, 24.34.8, 22.99.8, 21.14.8, 19.79.8, 19.14.8, 18.49.8, 17.60.8, &
16.83.8, 16.34.8, 16.15.8, 16.24.8, 16.37.8, 16.26.8, 15.85.8, 15.12.8, 14.32.8, 13.93.8, &
13.94.8, 13.75.8, 13.41.8, 13.58.8, 14.32.8, 15.23.8, 16.18.8, 16.91.8, 16.85.8, 16.20.8, &
15.78.8, 15.84.8, 16.19.8, 16.95.8, 17.97.8, 18.49.8, 18.03.8, 16.97.8, 16.16.8, 16.41.8, &
17.91.8, 19.70.8, 20.54.8, 20.40.8, 20.22.8, 20.55.8, 21.16.8, 21.53.8, 21.28.8, 20.29.8, &
18.95.8, 17.79.8, 16.89.8, 15.98.8, 15.16.8, 13.24.8, 10.27.8, 5.06.8, 0.00.8, 0.00.8, &

DATA gurban / 1831*0.0_8 /

DATA vhighway / 3121*80.0_8 /

DATA ghighway

/ 0.000.8, &
0.000.8, 0.000.8, 0.000.8, 0.000.8, -0.228.8, -2.280.8, -2.280.8, -2.280.8, -2.280.8, -2.280.8, &
-2.280.8, -2.280.8, -0.669.8, -0.490.8, -0.490.8, -0.490.8, -0.490.8, -0.490.8, 0.300.8, 0.494.8, &
1.270.8, 1.270.8, 3.135.8, 5.000.8, 5.000.8, 1.400.8, -4.000.8, -4.000.8, -4.000.8, -1.780.8, &
0.712.8, -1.080.8, -1.080.8, -1.080.8, 2.200.8, 2.200.8, 2.200.8, 2.200.8, 2.200.8, 2.200.8, &
2.200.8, 2.200.8, 2.200.8, 1.162.8, -2.990.8, -2.990.8, -2.990.8, -2.990.8, -2.990.8, -2.639.8, &
0.520.8, 0.520.8, 0.520.8, 0.520.8, 0.520.8, 0.768.8, 0.830.8, 0.830.8, 0.830.8, 0.830.8, &
-2.910.8, -2.910.8, -2.910.8, -2.910.8, -2.910.8, 1.786.8, 2.960.8, 2.960.8, 2.960.8, 2.960.8, &
2.960.8, 0.576.8, -3.000.8, -3.000.8, -3.000.8, -3.000.8, -3.000.8, -3.000.8, 2.247.8, 2.830.8, &
2.830.8, 2.830.8, 2.830.8, 2.830.8, 2.830.8, 2.830.8, 1.122.8, 0.996.8, 2.410.8, &
2.410.8, 2.410.8, 2.410.8, 2.410.8, -0.586.8, -1.870.8, -1.600.8, -0.520.8, -0.520.8, -0.520.8, &
-0.288.8, -0.230.8, -0.230.8, 0.292.8, 2.380.8, 2.380.8, 2.380.8, -0.221.8, -0.510.8, -0.510.8, &
-0.510.8, -1.998.8, -2.370.8, -2.370.8, -2.370.8, -2.370.8, -2.370.8, -2.370.8, -1.582.8, &
-0.400.8, -0.400.8, -0.400.8, -0.400.8, -0.400.8, -0.400.8, 0.520.8, 0.520.8, 0.520.8, &
0.520.8, 0.520.8, 0.520.8, -0.472.8, -1.960.8, -1.960.8, -1.006.8, 1.220.8, 1.220.8, 1.131.8, &
0.100.8, -1.970.8, -1.970.8, -1.970.8, -1.970.8, -1.970.8, -1.970.8, -1.970.8, -0.745.8, -0.220.8, &
-0.220.8, -0.036.8, 0.240.8, 0.240.8, 0.240.8, 0.240.8, 0.240.8, 0.090.8, -0.060.8, -0.060.8, &
-0.060.8, -0.060.8, -0.060.8, 0.868.8, 1.100.8, 1.100.8, 0.456.8, -0.510.8, -0.510.8, -0.510.8, &
-0.510.8, -0.249.8, 0.360.8, 0.360.8, 0.360.8, 0.360.8, 0.360.8, -0.540.8, -0.640.8, -0.122.8, &
1.950.8, 1.950.8, 1.680.8, 1.410.8, 1.410.8, 1.410.8, 1.410.8, 1.455.8, 1.860.8, 1.860.8, &
1.860.8, 1.860.8, 1.860.8, -0.316.8, -0.860.8, -0.860.8, -0.860.8, -0.860.8, -1.135.8, -1.410.8, &
-1.410.8, -1.410.8, -1.410.8, -1.239.8, 0.300.8, 0.300.8, 0.126.8, 0.010.8, 0.010.8, 0.010.8, &
0.010.8, 0.155.8, 0.300.8, 0.300.8, 0.300.8, 0.760.8, 2.600.8, 2.600.8, 1.300.8, 1.300.8, &
1.300.8, 1.300.8, 1.300.8, 1.492.8, 3.220.8, 3.220.8, 3.220.8, 3.220.8, 1.596.8, &
0.900.8, 0.900.8, 0.900.8, 0.900.8, 0.778.8, -0.320.8, -0.320.8, -2.092.8, -4.750.8, -4.750.8, &
-4.750.8, 2.585.8, 3.400.8, 2.334.8, -1.930.8, -1.930.8, -1.930.8, -1.930.8, -1.930.8, -1.004.8, &
2.700.8, 2.700.8, 2.700.8, 2.700.8, 2.970.8, 3.000.8, 3.000.8, 3.000.8, -4.011.8, -4.790.8, &
-3.546.8, -1.680.8, -1.680.8, -1.680.8, -1.680.8, -1.680.8, -1.680.8, -1.680.8, -1.515.8, -1.130.8, &
-0.757.8, 2.600.8, 2.600.8, -2.144.8, -2.797.8, 2.108.8, 2.540.8, 3.070.8, 3.600.8, 2.652.8, &
0.440.8, 0.440.8, -1.926.8, -2.940.8, 0.012.8, 1.530.8, 1.080.8, 1.233.8, 1.250.8, 1.250.8, &
1.724.8, 2.040.8, 2.040.8, 2.040.8, 4.740.8, 5.040.8, 5.040.8, 5.040.8, 5.040.8, 3.164.8, &
2.360.8, 2.360.8, 2.360.8, 2.360.8, 1.754.8, 1.350.8, 1.350.8, 1.350.8, 1.350.8, 1.350.8, &
1.548.8, 1.680.8, 1.680.8, 1.244.8, -0.500.8, -0.500.8, -0.500.8, -0.563.8, -0.710.8, -0.710.8, &
-0.048.8, 2.600.8, 2.600.8, 2.600.8, 2.600.8, 2.600.8, 1.370.8, 0.550.8, 0.550.8, 1.040.8, &
3.000.8, 3.000.8, 3.000.8, 3.000.8, 3.000.8, 1.120.8, 1.120.8, 1.120.8, 1.120.8, 1.120.8, &
1.120.8, 2.880.8, 2.880.8, 2.880.8, 2.880.8, 2.880.8, 2.515.8, -0.770.8, -0.770.8, -0.770.8, &
-0.770.8, -0.770.8, -0.770.8, -0.770.8, -0.770.8, -0.770.8, 4.000.8, 4.000.8, 4.000.8, 4.000.8, &
4.000.8, 4.000.8, 4.000.8, 2.696.8, 2.370.8, 2.370.8, 2.370.8, -0.126.8, -0.750.8, -0.750.8, &
-0.750.8, -0.750.8, -0.750.8, -0.510.8, 1.650.8, 1.650.8, 1.347.8, 0.640.8, 0.640.8, 0.640.8, &
0.640.8, 0.640.8, -0.180.8, -1.000.8, -1.000.8, -1.000.8, -0.325.8, -0.250.8, -0.250.8, -0.250.8, &
-1.218.8, -1.460.8, -1.460.8, -1.460.8, -1.460.8, -1.873.8, -2.050.8, -2.050.8, -2.050.8, -2.050.8, &
-2.050.8, -2.090.8, -2.450.8, -2.450.8, -2.450.8, -1.855.8, -1.260.8, -1.260.8, -1.260.8, -1.260.8, &
-1.260.8, -1.260.8, -1.260.8, -1.260.8, -1.260.8, -1.260.8, -2.644.8, -2.990.8, -2.990.8, &
-2.990.8, -2.990.8, -2.917.8, -2.260.8, -2.260.8, -2.260.8, -2.260.8, -2.260.8, -2.260.8, -2.260.8, &
-2.260.8, -2.260.8, -2.260.8, -2.260.8, -2.260.8, -2.260.8, -2.260.8, -2.260.8, -2.630.8, &
-3.000.8, -3.000.8, -3.000.8, -3.000.8, -3.000.8, -3.000.8, 0.770.8, 0.770.8, 0.770.8, 0.770.8, &
0.770.8, 0.770.8, 0.770.8, 0.770.8, 0.770.8, 0.770.8, 0.770.8, 0.770.8, 0.149.8, -1.300.8, &
-1.300.8, -1.300.8, -1.300.8, -1.300.8, -0.180.8, 0.300.8, 0.300.8, 0.300.8, 0.300.8, 0.300.8, &
0.300.8, 0.300.8, 0.300.8, 0.300.8, 0.300.8, 0.300.8, 0.012.8, -0.660.8, -0.660.8, &
-0.660.8, -0.660.8, -0.866.8, -2.720.8, -2.720.8, -2.720.8, -2.720.8, -2.250.8, -0.370.8, -0.370.8, &
-0.370.8, -0.370.8, -0.370.8, -0.370.8, -0.370.8, -0.370.8, -0.632.8, -1.680.8, -1.680.8, -1.680.8, &
-1.680.8, -1.680.8, -1.680.8, -1.680.8, -1.680.8, -0.154.8, 0.500.8, 0.500.8, 0.500.8, 0.500.8, &
0.500.8, 0.500.8, -0.217.8, -1.890.8, -1.890.8, -1.890.8, -1.890.8, 1.251.8, 1.600.8, 1.600.8, &
1.600.8, 1.600.8, 1.600.8, 1.222.8, -0.290.8, -0.290.8, -0.290.8, -0.290.8, -0.290.8, 1.020.8, &
2.330.8, 2.330.8, 2.330.8, 2.330.8, -0.030.8, -0.620.8, -0.620.8, -0.620.8, -0.620.8, 1.300.8, &
1.300.8, 1.300.8, 0.600.8, -0.450.8, -0.450.8, -0.450.8, -0.208.8, 0.760.8, 0.760.8, 0.760.8, &
0.760.8, 0.760.8, 0.760.8, 0.760.8, 0.760.8, 0.760.8, -2.760.8, -2.760.8, -2.760.8, &
-2.760.8, -2.760.8, -2.760.8, -2.760.8, -2.760.8, -2.760.8, -2.760.8, -2.760.8, -0.192.8, &
0.450.8, 0.450.8, 0.450.8, 0.880.8, 2.600.8, 2.600.8, 2.600.8, 2.600.8, 2.600.8, 2.600.8, &
2.600.8, 2.600.8, 2.600.8, 0.500.8, -2.650.8, -2.650.8, -2.650.8, -2.650.8, -2.650.8, -2.650.8, &
-2.650.8, -2.650.8, -2.650.8, -2.335.8, 0.500.8, 0.500.8, 0.500.8, 0.500.8, 0.500.8, 0.500.8, &
0.500.8, 0.050.8, -0.400.8, -0.400.8, -0.400.8, -0.400.8, -0.400.8, -0.400.8, -0.400.8, -1.040.8, &
-2.000.8, -2.000.8, -1.800.8, 0.000.8, 0.000.8, 0.000.8, 0.150.8, 0.500.8, 0.500.8, 0.500.8, &
0.248.8, -0.760.8, -0.760.8, -0.760.8, -0.760.8, -0.760.8, -0.760.8, -0.760.8, -0.228.8, &
0.000.8, 0.000.8, 0.000.8, 0.000.8, 2.232.8, 2.790.8, 2.790.8, 2.790.8, 2.790.8, 1.772.8, &
-2.300.8, -2.300.8, -2.300.8, -2.300.8, -2.300.8, -2.300.8, -2.300.8, -2.300.8, 0.962.8, 2.360.8, &
2.360.8, 2.360.8, 2.360.8, 2.360.8, 2.360.8, 2.360.8, -0.692.8, -2.000.8, -2.000.8, &
-2.000.8, 0.403.8, 0.670.8, 0.670.8, 0.670.8, -0.257.8, -0.360.8, -0.360.8, -0.360.8, -0.360.8, &
-0.297.8, -0.290.8, -0.290.8, -0.290.8, -0.290.8, -0.346.8, -0.370.8, -0.370.8, -0.370.8, -0.370.8, &
-0.370.8, -0.370.8, -0.370.8, -0.370.8, -0.370.8, -0.370.8, -0.370.8, -0.370.8, -0.370.8, &
0.080.8, 0.130.8, 0.130.8, 0.130.8, 0.130.8, 0.130.8, 0.130.8, 1.230.8, 2.330.8, 2.330.8, &
2.330.8, 2.330.8, 2.330.8, 2.330.8, 2.330.8, 2.330.8, 2.330.8, -0.541.8, -0.860.8, &

-0.860_8, -0.860_8, -0.860_8, -1.105_8, -1.350_8, -1.350_8, -1.350_8, -1.350_8, -1.350_8, -1.350_8, &
 -1.350_8, -1.350_8, -1.350_8, -1.350_8, -1.350_8, -0.618_8, -0.130_8, -0.130_8, 0.293_8, 0.340_8, &
 0.340_8, 0.340_8, 0.340_8, 0.340_8, 0.340_8, 0.340_8, 0.340_8, -0.045_8, -0.430_8, -0.430_8, &
 -0.430_8, -0.430_8, -0.430_8, 0.102_8, 0.900_8, 0.900_8, 0.900_8, 0.366_8, -0.880_8, -0.880_8, &
 -0.880_8, -0.880_8, -0.880_8, -0.880_8, -0.880_8, -0.880_8, 0.208_8, 1.840_8, 1.840_8, 1.840_8, &
 1.840_8, 1.840_8, 1.840_8, 1.840_8, 1.840_8, 1.406_8, -2.500_8, -2.500_8, -2.500_8, -2.500_8, &
 -2.500_8, -0.100_8, -0.100_8, -0.100_8, -0.100_8, -0.100_8, -0.100_8, -0.100_8, -0.100_8, -0.100_8, &
 -0.100_8, 0.140_8, 0.200_8, 0.200_8, 0.200_8, 0.200_8, 0.200_8, 0.220_8, 0.300_8, 0.300_8, 0.300_8, &
 0.250_8, -0.200_8, -0.200_8, -0.200_8, -0.200_8, -0.200_8, -0.200_8, 1.240_8, 1.600_8, 1.600_8, 1.240_8, &
 -2.000_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, -0.020_8, &
 0.200_8, 0.200_8, 0.200_8, 0.200_8, 0.200_8, 0.200_8, 0.083_8, 0.070_8, 0.070_8, 0.070_8, &
 0.070_8, 0.070_8, 0.070_8, 0.070_8, 0.222_8, 0.450_8, 0.450_8, 0.450_8, 0.450_8, 0.450_8, &
 0.450_8, 0.450_8, 0.450_8, 0.450_8, 0.385_8, -0.200_8, -0.200_8, -0.200_8, -0.200_8, -0.200_8, &
 -0.200_8, -0.200_8, -0.200_8, -0.200_8, -0.398_8, -2.180_8, -2.180_8, -2.180_8, -2.180_8, 1.048_8, &
 3.200_8, 3.200_8, 3.200_8, 3.200_8, 1.555_8, 0.850_8, 0.850_8, 0.850_8, -1.090_8, -4.000_8, &
 -4.000_8, -4.000_8, -4.000_8, -4.000_8, -4.000_8, -4.000_8, -4.000_8, -4.000_8, -4.000_8, 2.800_8, &
 4.500_8, 4.500_8, 4.500_8, 4.500_8, 4.500_8, 4.500_8, 2.500_8, 2.000_8, 2.000_8, 2.000_8, &
 2.000_8, 2.000_8, 2.000_8, 2.000_8, 2.000_8, 1.686_8, 0.430_8, 0.430_8, 0.430_8, 0.430_8, &
 0.430_8, 0.430_8, 0.430_8, -1.714_8, -4.930_8, -4.930_8, -4.930_8, -4.930_8, -4.930_8, &
 -1.740_8, -1.740_8, -1.740_8, -1.740_8, -2.370_8, -3.000_8, -3.000_8, -2.719_8, -0.190_8, -0.190_8, &
 -0.190_8, -0.190_8, -0.190_8, -0.190_8, -0.190_8, -0.190_8, 1.840_8, 1.840_8, 1.840_8, 1.840_8, &
 1.732_8, 0.760_8, 0.760_8, 0.760_8, -0.423_8, -0.930_8, -0.930_8, -0.604_8, 0.700_8, 0.700_8, &
 0.700_8, 0.700_8, -0.920_8, -1.100_8, -1.100_8, -1.100_8, -1.100_8, -0.828_8, -0.420_8, -0.420_8, &
 -0.420_8, -0.420_8, -0.420_8, -0.420_8, -0.420_8, -0.078_8, 0.720_8, 0.720_8, 0.720_8, -0.370_8, &
 -0.370_8, -0.370_8, -0.370_8, -0.812_8, -2.580_8, -2.580_8, -2.580_8, -2.580_8, -2.580_8, &
 -2.580_8, -2.580_8, -2.580_8, -2.580_8, -2.580_8, -2.580_8, -0.683_8, 0.130_8, 0.130_8, 0.130_8, &
 0.130_8, 0.130_8, 0.130_8, 0.130_8, -0.493_8, -0.760_8, -0.760_8, -0.760_8, -0.760_8, &
 -0.760_8, -0.760_8, -0.394_8, 0.460_8, 0.460_8, 0.460_8, 0.460_8, -0.317_8, -0.650_8, -0.650_8, &
 -0.650_8, -0.650_8, -0.650_8, -0.650_8, -0.650_8, 0.136_8, 0.660_8, 0.660_8, 0.660_8, 0.660_8, &
 0.660_8, 3.000_8, 3.000_8, 3.000_8, 3.000_8, 3.000_8, 1.068_8, -1.830_8, -1.830_8, -1.830_8, &
 -1.830_8, -1.830_8, 0.126_8, 1.430_8, 1.430_8, 1.430_8, 1.430_8, 0.576_8, -2.840_8, -2.840_8, &
 -2.840_8, -2.840_8, -2.840_8, -2.840_8, -1.270_8, 0.300_8, 0.300_8, 0.300_8, 0.300_8, 0.300_8, &
 0.300_8, -0.510_8, -0.510_8, -0.510_8, -0.510_8, 0.200_8, 0.200_8, 0.200_8, 0.200_8, 0.200_8, &
 0.200_8, -0.087_8, -0.210_8, -0.210_8, -0.210_8, 0.036_8, 0.200_8, 0.200_8, 0.200_8, 0.060_8, &
 0.000_8, 0.000_8, 0.000_8, -0.120_8, -0.400_8, -0.400_8, -0.400_8, -0.400_8, 0.800_8, 2.600_8, &
 2.600_8, 2.600_8, 2.600_8, 2.600_8, 2.600_8, 2.600_8, 1.420_8, 0.240_8, 0.240_8, 1.235_8, &
 2.230_8, 2.230_8, 2.230_8, 2.230_8, 2.230_8, 2.230_8, 2.230_8, 2.230_8, 2.230_8, 2.230_8, &
 2.230_8, 2.230_8, 0.529_8, -0.200_8, -0.200_8, -0.200_8, -0.200_8, -0.200_8, -0.760_8, -3.000_8, &
 -3.000_8, -3.000_8, 0.210_8, 2.350_8, 2.350_8, 2.350_8, 2.350_8, 0.541_8, 0.340_8, 0.340_8, &
 0.340_8, -0.170_8, -0.510_8, -0.510_8, -0.510_8, -1.302_8, -1.830_8, -1.830_8, -1.830_8, -1.830_8, &
 -1.830_8, -1.830_8, -1.830_8, -1.830_8, -0.006_8, -0.302_8, -1.430_8, -1.430_8, -1.430_8, -1.100_8, &
 -1.100_8, -1.100_8, -1.100_8, 0.300_8, 1.700_8, 1.700_8, 1.700_8, 1.700_8, 1.700_8, 1.228_8, &
 -0.660_8, -0.660_8, -0.660_8, -0.660_8, 0.339_8, 2.670_8, 2.670_8, 0.674_8, -2.320_8, -2.320_8, &
 -1.356_8, 2.500_8, 2.500_8, 2.500_8, 2.500_8, 2.500_8, -1.000_8, -2.500_8, -2.500_8, -2.500_8, &
 -2.500_8, -2.500_8, -2.500_8, -2.500_8, -2.500_8, -2.500_8, -2.090_8, 1.600_8, 1.600_8, 1.600_8, &
 1.600_8, 1.600_8, 1.600_8, 2.230_8, 2.500_8, 2.500_8, 1.573_8, -0.590_8, -0.533_8, -0.400_8, &
 -0.400_8, -0.400_8, 0.240_8, 0.400_8, 0.400_8, 0.220_8, -0.500_8, -0.500_8, -0.968_8, -1.670_8, &
 -1.670_8, -1.670_8, -1.670_8, -0.985_8, -0.300_8, -0.960_8, -2.500_8, -2.220_8, -1.800_8, -1.760_8, &
 -1.600_8, -1.600_8, -1.600_8, -1.600_8, -1.600_8, -1.600_8, -1.600_8, -0.820_8, -0.300_8, &
 -1.920_8, -3.000_8, -3.000_8, -3.000_8, -3.000_8, -3.000_8, -3.000_8, -0.282_8, 0.020_8, 0.020_8, &
 1.155_8, 2.110_8, 2.110_8, 1.805_8, -0.940_8, -0.772_8, 0.180_8, 0.250_8, 0.250_8, 0.250_8, &
 0.400_8, 0.580_8, 1.300_8, 0.800_8, 0.300_8, 1.200_8, 1.800_8, 1.640_8, 1.000_8, 1.000_8, &
 -0.400_8, -1.000_8, -0.280_8, -0.200_8, -0.200_8, -0.200_8, -0.080_8, 1.000_8, 1.000_8, 1.000_8, &
 1.000_8, -1.240_8, -2.200_8, -2.200_8, -2.200_8, 1.800_8, 1.800_8, 1.800_8, 1.800_8, 1.800_8, &
 2.790_8, 2.900_8, 2.900_8, 2.900_8, 1.720_8, -3.000_8, -3.000_8, -3.000_8, 1.000_8, 1.000_8, &
 1.800_8, 3.000_8, 3.000_8, 1.500_8, 0.500_8, 0.920_8, 1.200_8, 1.280_8, 1.600_8, 1.600_8, &
 2.720_8, 3.000_8, 1.945_8, 0.890_8, 0.861_8, 0.600_8, 0.600_8, 0.600_8, 1.288_8, 1.460_8, &
 -0.670_8, -2.800_8, -2.800_8, -2.800_8, -2.800_8, -2.060_8, -1.320_8, -1.101_8, 0.870_8, 0.870_8, &
 -0.544_8, -1.150_8, -1.150_8, -1.420_8, -1.450_8, -1.765_8, -1.900_8, -1.900_8, -1.900_8, -1.900_8, &
 -0.600_8, -0.600_8, -0.744_8, -0.760_8, -0.760_8, -1.230_8, -1.230_8, -1.230_8, -1.230_8, &
 -1.230_8, -1.230_8, -1.230_8, 0.246_8, 0.410_8, 0.978_8, 1.120_8, 1.120_8, 1.120_8, 1.120_8, &
 -0.608_8, -0.800_8, -0.020_8, 0.600_8, 1.000_8, 1.000_8, 1.900_8, 2.000_8, 2.000_8, 2.000_8, &
 0.950_8, -1.500_8, -1.350_8, -0.750_8, -0.750_8, -0.750_8, -0.750_8, -0.750_8, -0.675_8, -0.500_8, &
 -0.500_8, -0.300_8, 1.500_8, 1.500_8, 1.500_8, 1.500_8, 1.500_8, 1.950_8, 2.000_8, 0.600_8, &
 -1.500_8, -1.500_8, 1.200_8, 3.000_8, 1.100_8, -0.800_8, -0.980_8, -1.100_8, -2.000_8, -2.000_8, &
 -2.000_8, -2.000_8, -2.000_8, -0.926_8, -0.210_8, -0.210_8, -0.210_8, -0.912_8, -0.990_8, -0.621_8, &
 0.240_8, 0.240_8, 0.966_8, 1.450_8, 1.450_8, 0.890_8, 0.050_8, 0.050_8, 0.050_8, 0.050_8, &
 0.050_8, 0.995_8, 1.100_8, 1.100_8, -1.412_8, -2.040_8, -2.040_8, -2.040_8, -2.040_8, -0.870_8, &
 0.300_8, 0.300_8, 0.300_8, 0.300_8, 0.300_8, 0.300_8, -0.180_8, -0.300_8, -0.295_8, -0.290_8, &
 -0.290_8, -0.290_8, -0.290_8, -0.290_8, -0.290_8, -0.290_8, -0.290_8, 2.671_8, 3.000_8, &
 -0.970_8, -0.970_8, -0.970_8, -0.970_8, -0.970_8, -0.970_8, -0.970_8, -0.970_8, -0.970_8, &
 -0.970_8, -0.970_8, -0.182_8, 1.000_8, 1.000_8, 1.000_8, 1.600_8, 2.000_8, 2.000_8, 1.040_8, &
 0.400_8, 0.128_8, -0.280_8, -0.280_8, -0.280_8, -0.772_8, -1.100_8, -1.100_8, -1.100_8, -1.100_8, &
 0.350_8, 0.350_8, 0.350_8, 0.350_8, -1.630_8, -1.630_8, -1.630_8, -1.365_8, -1.100_8, -1.100_8, &
 -1.100_8, 0.850_8, 2.800_8, 2.800_8, 2.800_8, 2.745_8, 2.250_8, 2.250_8, 1.340_8, -2.300_8, &
 -2.300_8, -2.180_8, -2.000_8, -2.000_8, -0.536_8, 0.600_8, 0.600_8, 0.600_8, 0.600_8, 0.600_8, &
 0.600_8, 0.600_8, 0.600_8, 0.600_8, 0.600_8, -0.840_8, -0.840_8, -0.144_8, 0.900_8, 0.900_8, &
 0.900_8, 0.900_8, 0.900_8, -0.189_8, -0.310_8, -0.310_8, -0.310_8, -0.310_8, -0.310_8, &
 0.683_8, 3.000_8, 3.000_8, 3.000_8, 3.000_8, 3.000_8, 3.000_8, 3.000_8, 3.000_8, 3.000_8, &
 3.000_8, 3.000_8, 3.000_8, 2.000_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, 0.800_8, &
 2.000_8, 2.000_8, 2.000_8, 0.144_8, -0.320_8, -0.320_8, -0.320_8, 0.608_8, 2.000_8, 0.512_8, &
 0.140_8, 0.140_8, 0.140_8, 0.140_8, 0.140_8, 0.140_8, 0.140_8, 0.140_8, 1.814_8, &
 2.000_8, 2.000_8, 2.000_8, 2.000_8, 2.000_8, 2.000_8, 2.000_8, 2.000_8, 2.000_8, &
 0.215_8, -1.570_8, -1.570_8, -1.570_8, -1.570_8, -1.570_8, -1.570_8, -0.331_8, 0.200_8, &
 0.200_8, -0.250_8, -0.700_8, -0.630_8, 0.000_8, 0.000_8, 0.000_8, 0.000_8, 0.000_8, -0.414_8, &
 0.414_8, 0.000_8, 0.000_8, 0.000_8, 0.000_8, 0.630_8, 0.700_8, 0.250_8, -0.200_8, &
 -0.200_8, 0.331_8, 1.570_8, 1.570_8, 1.570_8, 1.570_8, 1.570_8, 1.570_8, 1.570_8, -0.215_8, &
 -2.000_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, &
 -1.814_8, -0.140_8, -0.140_8, -0.140_8, -0.140_8, -0.140_8, -0.140_8, -0.140_8, -0.140_8, &
 -0.512_8, -2.000_8, -0.608_8, 0.320_8, 0.320_8, 0.320_8, -0.144_8, -2.000_8, -2.000_8, &
 -0.800_8, 2.000_8, 2.000_8, 2.000_8, 2.000_8, 2.000_8, -2.000_8, -3.000_8, -3.000_8, &
 -3.000_8, -3.000_8, -3.000_8, -3.000_8, -3.000_8, -3.000_8, -3.000_8, -3.000_8, -0.683_8, &
 0.310_8, 0.310_8, 0.310_8, 0.310_8, 0.189_8, -0.900_8, -0.900_8, -0.900_8, -0.900_8, &
 -0.900_8, -0.900_8, 0.144_8, 0.840_8, 0.840_8, -0.600_8, -0.600_8, -0.600_8, -0.600_8, &
 -0.600_8, -0.600_8, -0.600_8, -0.600_8, -0.600_8, 0.536_8, 2.000_8, 2.000_8, 2.180_8, &
 2.300_8, -1.340_8, -2.250_8, -2.250_8, -2.250_8, -2.745_8, -2.800_8, -2.800_8, -0.850_8, &
 1.100_8, 1.100_8, 1.365_8, 1.630_8, 1.630_8, 1.630_8, -0.350_8, -0.350_8, -0.350_8, &
 1.100_8, 1.100_8, 1.100_8, 1.100_8, 0.772_8, 0.280_8, 0.280_8, 0.280_8, -0.128_8, -0.400_8, &

-1.040_8, -2.000_8, -2.000_8, -1.600_8, -1.000_8, -1.000_8, -1.000_8, 0.182_8, 0.970_8, 0.970_8, &
0.970_8, 0.970_8, 0.970_8, 0.970_8, 0.970_8, 0.970_8, 0.970_8, 0.970_8, 0.970_8, 0.970_8, &
-3.000_8, -2.671_8, 0.290_8, 0.290_8, 0.290_8, 0.290_8, 0.290_8, 0.290_8, 0.290_8, 0.290_8, &
0.290_8, 0.295_8, 0.300_8, 0.180_8, -0.300_8, -0.300_8, -0.300_8, -0.300_8, -0.300_8, -0.300_8, &
0.870_8, 2.040_8, 2.040_8, 2.040_8, 2.040_8, 1.412_8, -1.100_8, -1.100_8, -0.995_8, -0.050_8, &
-0.050_8, -0.050_8, -0.050_8, -0.050_8, -0.890_8, -1.450_8, -1.450_8, -0.966_8, -0.240_8, -0.240_8, &
0.621_8, 0.990_8, 0.912_8, 0.210_8, 0.210_8, 0.210_8, 0.926_8, 2.000_8, 2.000_8, 2.000_8, &
2.000_8, 2.000_8, 1.100_8, 0.980_8, 0.800_8, -1.100_8, -3.000_8, -1.200_8, 1.500_8, 1.500_8, &
-0.600_8, -2.000_8, -1.950_8, -1.500_8, -1.500_8, -1.500_8, -1.500_8, -1.500_8, 0.300_8, 0.500_8, &
0.500_8, 0.675_8, 0.750_8, 0.750_8, 0.750_8, 0.750_8, 0.750_8, 1.350_8, 1.500_8, -0.950_8, &
-2.000_8, -2.000_8, -2.000_8, -1.900_8, -1.000_8, -1.000_8, -0.600_8, 0.020_8, 0.800_8, 0.608_8, &
-1.120_8, -1.120_8, -1.120_8, -1.120_8, -0.978_8, -0.410_8, -0.246_8, 1.230_8, 1.230_8, 1.230_8, &
1.230_8, 1.230_8, 1.230_8, 1.230_8, 1.230_8, 0.760_8, 0.760_8, 0.744_8, 0.600_8, 0.600_8, &
1.900_8, 1.900_8, 1.900_8, 1.900_8, 1.765_8, 1.450_8, 1.420_8, 1.150_8, 1.150_8, 0.544_8, &
-0.870_8, -0.870_8, 1.101_8, 1.320_8, 2.060_8, 2.800_8, 2.800_8, 2.800_8, 2.800_8, 0.670_8, &
-1.460_8, -1.288_8, -0.600_8, -0.600_8, -0.600_8, -0.861_8, -0.890_8, -1.945_8, -3.000_8, -2.720_8, &
-1.600_8, -1.600_8, -1.280_8, -1.200_8, -0.920_8, -0.500_8, -1.500_8, -3.000_8, -3.000_8, -1.800_8, &
-1.000_8, -1.000_8, 3.000_8, 3.000_8, 3.000_8, -1.720_8, -2.900_8, -2.900_8, -2.900_8, -2.790_8, &
-1.800_8, -1.800_8, -1.800_8, -1.800_8, -1.800_8, 2.200_8, 2.200_8, 2.200_8, 1.240_8, -1.000_8, &
-1.000_8, -1.000_8, -1.000_8, 0.080_8, 0.200_8, 0.200_8, 0.200_8, 0.280_8, 1.000_8, 0.400_8, &
-1.000_8, -1.000_8, -1.640_8, -1.800_8, -1.200_8, -0.300_8, -0.800_8, -1.300_8, -0.580_8, -0.400_8, &
-0.250_8, -0.250_8, -0.250_8, -0.180_8, 0.772_8, 0.940_8, -1.805_8, -2.110_8, -2.110_8, -1.155_8, &
-0.020_8, -0.020_8, 0.282_8, 3.000_8, 3.000_8, 3.000_8, 3.000_8, 3.000_8, 3.000_8, 1.920_8, &
0.300_8, 0.820_8, 1.600_8, 1.600_8, 1.600_8, 1.600_8, 1.600_8, 1.600_8, 1.600_8, 1.600_8, &
1.760_8, 1.800_8, 2.220_8, 2.500_8, 0.960_8, 0.300_8, 0.985_8, 1.670_8, 1.670_8, 1.670_8, &
1.670_8, 0.968_8, 0.500_8, 0.500_8, -0.220_8, -0.400_8, -0.400_8, -0.240_8, 0.400_8, 0.400_8, &
0.400_8, 0.533_8, 0.590_8, -1.573_8, -2.500_8, -2.500_8, -2.230_8, -1.600_8, -1.600_8, -1.600_8, &
-1.600_8, -1.600_8, 2.090_8, 2.500_8, 2.500_8, 2.500_8, 2.500_8, 2.500_8, 2.500_8, 2.500_8, &
2.500_8, 2.500_8, 2.500_8, 2.500_8, 1.000_8, -2.500_8, -2.500_8, -2.500_8, -2.500_8, 1.356_8, &
2.320_8, 2.320_8, -0.674_8, -2.670_8, -2.670_8, -0.339_8, 0.660_8, 0.660_8, 0.660_8, 0.660_8, &
-1.228_8, -1.700_8, -1.700_8, -1.700_8, -1.700_8, -1.700_8, -0.300_8, 1.100_8, 1.100_8, 1.100_8, &
1.100_8, 1.430_8, 1.430_8, 1.430_8, 0.302_8, 0.006_8, 1.830_8, 1.830_8, 1.830_8, 1.830_8, &
1.830_8, 1.830_8, 1.830_8, 1.830_8, 1.830_8, 0.510_8, 0.510_8, 0.510_8, 0.170_8, -0.340_8, &
-0.340_8, -0.340_8, -0.541_8, -2.350_8, -2.350_8, -2.350_8, -2.350_8, -0.210_8, 3.000_8, 3.000_8, &
3.000_8, 0.760_8, 0.200_8, 0.200_8, 0.200_8, 0.200_8, 0.200_8, -0.529_8, -2.230_8, -2.230_8, &
-2.230_8, -2.230_8, -2.230_8, -2.230_8, -2.230_8, -2.230_8, -2.230_8, -2.230_8, -2.230_8, &
-1.235_8, -0.240_8, -0.240_8, -1.420_8, -2.600_8, -2.600_8, -2.600_8, -2.600_8, -2.600_8, -2.600_8, &
-2.600_8, -0.800_8, 0.400_8, 0.400_8, 0.400_8, 0.400_8, 0.120_8, 0.000_8, 0.000_8, 0.000_8, &
-0.060_8, -0.200_8, -0.200_8, -0.200_8, -0.036_8, 0.210_8, 0.210_8, 0.210_8, 0.087_8, -0.200_8, &
-0.200_8, -0.200_8, -0.200_8, -0.200_8, -0.200_8, 0.510_8, 0.510_8, 0.510_8, 0.510_8, -0.300_8, &
-0.300_8, -0.300_8, -0.300_8, -0.300_8, -0.300_8, 1.270_8, 2.840_8, 2.840_8, 2.840_8, 2.840_8, &
2.840_8, 2.840_8, -0.576_8, -1.430_8, -1.430_8, -1.430_8, -1.430_8, -0.126_8, 1.830_8, 1.830_8, &
1.830_8, 1.830_8, 1.830_8, -1.068_8, -3.000_8, -3.000_8, -3.000_8, -3.000_8, -3.000_8, -0.660_8, &
-0.660_8, -0.660_8, -0.660_8, -0.660_8, -0.136_8, 0.650_8, 0.650_8, 0.650_8, 0.650_8, 0.650_8, &
0.650_8, 0.650_8, 0.317_8, -0.460_8, -0.460_8, -0.460_8, -0.460_8, 0.394_8, 0.760_8, 0.760_8, &
0.760_8, 0.760_8, 0.760_8, 0.760_8, 0.760_8, 0.493_8, -0.130_8, -0.130_8, -0.130_8, -0.130_8, &
-0.130_8, -0.130_8, -0.130_8, -0.130_8, 0.683_8, 2.580_8, 2.580_8, 2.580_8, 2.580_8, 2.580_8, &
2.580_8, 2.580_8, 2.580_8, 2.580_8, 0.812_8, 0.370_8, 0.370_8, 0.370_8, 0.370_8, &
0.370_8, -0.720_8, -0.720_8, -0.720_8, 0.078_8, 0.420_8, 0.420_8, 0.420_8, 0.420_8, 0.420_8, &
0.420_8, 0.420_8, 0.828_8, 1.100_8, 1.100_8, 1.100_8, 1.100_8, 0.920_8, -0.700_8, -0.700_8, &
-0.700_8, -0.700_8, 0.604_8, 0.930_8, 0.930_8, 0.423_8, -0.760_8, -0.760_8, -0.760_8, -1.732_8, &
-1.840_8, -1.840_8, -1.840_8, -1.840_8, 0.190_8, 0.190_8, 0.190_8, 0.190_8, 0.190_8, 0.190_8, &
0.190_8, 0.190_8, 2.719_8, 3.000_8, 3.000_8, 2.370_8, 1.740_8, 1.740_8, 1.740_8, 1.740_8, &
2.059_8, 4.930_8, 4.930_8, 4.930_8, 4.930_8, 4.930_8, 1.714_8, -0.430_8, -0.430_8, -0.430_8, &
-0.430_8, -0.430_8, -0.430_8, -0.430_8, -1.686_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, &
-2.000_8, -2.000_8, -2.000_8, -2.500_8, -4.500_8, -4.500_8, -4.500_8, -4.500_8, -4.500_8, -4.500_8, &
-2.800_8, 4.000_8, 4.000_8, 4.000_8, 4.000_8, 4.000_8, 4.000_8, 4.000_8, 4.000_8, 4.000_8, &
4.000_8, 1.090_8, -0.850_8, -0.850_8, -0.850_8, -1.555_8, -3.200_8, -3.200_8, -3.200_8, -3.200_8, &
-1.048_8, 2.180_8, 2.180_8, 2.180_8, 2.180_8, 0.398_8, 0.200_8, 0.200_8, 0.200_8, 0.200_8, &
0.200_8, 0.200_8, 0.200_8, 0.200_8, 0.200_8, -0.385_8, -0.450_8, -0.450_8, -0.450_8, -0.450_8, &
-0.450_8, -0.450_8, -0.450_8, -0.450_8, -0.450_8, -0.222_8, -0.070_8, -0.070_8, -0.070_8, -0.070_8, &
-0.070_8, -0.070_8, -0.070_8, -0.083_8, -0.200_8, -0.200_8, -0.200_8, -0.200_8, -0.200_8, -0.200_8, &
0.020_8, 2.000_8, 2.000_8, 2.000_8, 2.000_8, 2.000_8, 2.000_8, 2.000_8, 2.000_8, 2.000_8, &
-1.240_8, -1.600_8, -1.600_8, -1.240_8, 0.200_8, 0.200_8, 0.200_8, 0.200_8, 0.200_8, -0.250_8, &
-0.300_8, -0.300_8, -0.300_8, -0.220_8, -0.200_8, -0.200_8, -0.200_8, -0.200_8, -0.140_8, 0.100_8, &
0.100_8, 0.100_8, 0.100_8, 0.100_8, 0.100_8, 0.100_8, 0.100_8, 0.100_8, 0.100_8, 2.500_8, &
2.500_8, 2.500_8, 2.500_8, 2.500_8, -1.406_8, -1.840_8, -1.840_8, -1.840_8, -1.840_8, -1.840_8, &
-1.840_8, -1.840_8, -1.840_8, -0.208_8, 0.880_8, 0.880_8, 0.880_8, 0.880_8, 0.880_8, 0.880_8, &
0.880_8, 0.880_8, -0.366_8, -0.900_8, -0.900_8, -0.900_8, -0.900_8, -0.102_8, 0.430_8, 0.430_8, &
0.430_8, 0.430_8, 0.045_8, -0.340_8, -0.340_8, -0.340_8, -0.340_8, -0.340_8, -0.340_8, -0.340_8, &
-0.340_8, -0.293_8, 0.130_8, 0.130_8, 0.618_8, 1.350_8, 1.350_8, 1.350_8, 1.350_8, 1.350_8, &
1.350_8, 1.350_8, 1.350_8, 1.350_8, 1.350_8, 1.105_8, 0.860_8, 0.860_8, 0.860_8, 0.860_8, &
0.860_8, 0.541_8, -2.330_8, -2.330_8, -2.330_8, -2.330_8, -2.330_8, -2.330_8, -2.330_8, -2.330_8, &
-2.330_8, -2.330_8, -1.230_8, -0.130_8, -0.130_8, -0.130_8, -0.130_8, -0.130_8, -0.130_8, -0.080_8, &
0.370_8, 0.370_8, 0.370_8, 0.370_8, 0.370_8, 0.370_8, 0.370_8, 0.370_8, 0.370_8, 0.370_8, &
0.370_8, 0.370_8, 0.370_8, 0.370_8, 0.346_8, 0.290_8, 0.290_8, 0.290_8, 0.290_8, 0.297_8, &
0.360_8, 0.360_8, 0.360_8, 0.360_8, 0.257_8, -0.670_8, -0.670_8, -0.670_8, -0.403_8, 2.000_8, &
2.000_8, 2.000_8, 0.692_8, -2.360_8, -2.360_8, -2.360_8, -2.360_8, -2.360_8, -2.360_8, -2.360_8, &
-2.360_8, -0.962_8, 2.300_8, 2.300_8, 2.300_8, 2.300_8, 2.300_8, 2.300_8, 2.300_8, 2.300_8, &
-1.772_8, -2.790_8, -2.790_8, -2.790_8, -2.790_8, -2.232_8, 0.000_8, 0.000_8, 0.000_8, 0.000_8, &
0.228_8, 0.760_8, 0.760_8, 0.760_8, 0.760_8, 0.760_8, 0.760_8, 0.760_8, 0.760_8, -0.248_8, &
-0.500_8, -0.500_8, -0.500_8, -0.150_8, 0.000_8, 0.000_8, 0.000_8, 1.800_8, 2.000_8, 2.000_8, &
1.040_8, 0.400_8, 0.400_8, 0.400_8, 0.400_8, 0.400_8, 0.400_8, 0.400_8, -0.050_8, -0.500_8, &
-0.500_8, -0.500_8, -0.500_8, -0.500_8, -0.500_8, -0.500_8, 2.335_8, 2.650_8, 2.650_8, 2.650_8, &
2.650_8, 2.650_8, 2.650_8, 2.650_8, 2.650_8, 2.650_8, -0.500_8, -2.600_8, -2.600_8, -2.600_8, &
-2.600_8, -2.600_8, -2.600_8, -2.600_8, -2.600_8, -2.600_8, -0.880_8, -0.450_8, -0.450_8, -0.450_8, &
0.192_8, 2.760_8, 2.760_8, 2.760_8, 2.760_8, 2.760_8, 2.760_8, 2.760_8, 2.760_8, 2.760_8, &
2.760_8, 2.760_8, 2.760_8, 2.760_8, -0.760_8, -0.760_8, -0.760_8, -0.760_8, -0.760_8, -0.760_8, &
-0.760_8, -0.760_8, -0.760_8, 0.208_8, 0.450_8, 0.450_8, 0.450_8, -0.600_8, -1.300_8, -1.300_8, &
-1.300_8, -0.148_8, 0.620_8, 0.620_8, 0.620_8, 0.030_8, -2.330_8, -2.330_8, -2.330_8, -2.330_8, &
-1.020_8, 0.290_8, 0.290_8, 0.290_8, 0.290_8, 0.290_8, -1.222_8, -1.600_8, -1.600_8, -1.600_8, &
-1.600_8, -1.600_8, -1.251_8, 1.890_8, 1.890_8, 1.890_8, 1.890_8, 0.217_8, -0.500_8, -0.500_8, &
-0.500_8, -0.500_8, -0.500_8, -0.500_8, 0.154_8, 1.680_8, 1.680_8, 1.680_8, 1.680_8, 1.680_8, &
1.680_8, 1.680_8, 1.680_8, 0.632_8, 0.370_8, 0.370_8, 0.370_8, 0.370_8, 0.370_8, 0.370_8, &
0.370_8, 0.370_8, 2.250_8, 2.720_8, 2.720_8, 2.720_8, 2.720_8, 0.866_8, 0.660_8, 0.660_8, &
0.660_8, 0.660_8, -0.012_8, -0.300_8, -0.300_8, -0.300_8, -0.300_8, -0.300_8, -0.300_8, -0.300_8, &
-0.300_8, -0.300_8, -0.300_8, -0.300_8, -0.300_8, 0.180_8, 1.300_8, 1.300_8, 1.300_8, 1.300_8, &
1.300_8, -0.149_8, -0.770_8, -0.770_8, -0.770_8, -0.770_8, -0.770_8, -0.770_8, -0.770_8, -0.770_8, &

```

-0.770_8, -0.770_8, -0.770_8, -0.770_8, 3.000_8, 3.000_8, 3.000_8, 3.000_8, 3.000_8, 3.000_8, &
2.630_8, 2.260_8, 2.260_8, 2.260_8, 2.260_8, 2.260_8, 2.260_8, 2.260_8, 2.260_8, 2.260_8, &
2.260_8, 2.260_8, 2.260_8, 2.260_8, 2.260_8, 2.260_8, 2.260_8, 2.917_8, 2.990_8, 2.990_8, &
2.990_8, 2.990_8, 2.644_8, 1.260_8, 1.260_8, 1.260_8, 1.260_8, 1.260_8, 1.260_8, 1.260_8, &
1.260_8, 1.260_8, 1.260_8, 1.260_8, 1.855_8, 2.450_8, 2.450_8, 2.450_8, 2.090_8, 2.050_8, &
2.050_8, 2.050_8, 2.050_8, 2.050_8, 1.873_8, 1.460_8, 1.460_8, 1.460_8, 1.460_8, 1.218_8, &
0.250_8, 0.250_8, 0.250_8, 0.325_8, 1.000_8, 1.000_8, 1.000_8, 0.180_8, -0.640_8, -0.640_8, &
-0.640_8, -0.640_8, -0.640_8, -1.347_8, -1.650_8, -1.650_8, 0.510_8, 0.750_8, 0.750_8, 0.750_8, &
0.750_8, 0.750_8, 0.126_8, -2.370_8, -2.370_8, -2.370_8, -2.696_8, -4.000_8, -4.000_8, -4.000_8, &
-4.000_8, -4.000_8, -4.000_8, -4.000_8, 0.770_8, 0.770_8, 0.770_8, 0.770_8, 0.770_8, 0.770_8, &
0.770_8, 0.770_8, 0.770_8, -2.515_8, -2.880_8, -2.880_8, -2.880_8, -2.880_8, -2.880_8, -1.120_8, &
-1.120_8, -1.120_8, -1.120_8, -1.120_8, -1.120_8, -3.000_8, -3.000_8, -3.000_8, -3.000_8, -3.000_8, &
-1.040_8, -0.550_8, -0.550_8, -0.550_8, -1.370_8, -2.600_8, -2.600_8, -2.600_8, -2.600_8, -2.600_8, &
0.710_8, 0.710_8, 0.563_8, 0.500_8, 0.500_8, 0.500_8, 0.500_8, -1.244_8, -1.680_8, -1.680_8, -1.548_8, &
-1.350_8, -1.350_8, -1.350_8, -1.350_8, -1.350_8, -1.754_8, -2.360_8, -2.360_8, -2.360_8, -2.360_8, &
-3.164_8, -5.040_8, -5.040_8, -5.040_8, -5.040_8, -4.740_8, -2.040_8, -2.040_8, -2.040_8, -1.724_8, &
-1.250_8, -1.250_8, -1.233_8, -1.080_8, -1.530_8, -0.012_8, 2.940_8, 1.926_8, -0.440_8, -0.440_8, &
-2.652_8, -3.600_8, -3.070_8, -2.540_8, -2.108_8, 2.797_8, 2.144_8, -2.600_8, -2.600_8, 0.757_8, &
1.130_8, 1.515_8, 1.680_8, 1.680_8, 1.680_8, 1.680_8, 1.680_8, 1.680_8, 1.680_8, 3.546_8, &
4.790_8, 4.011_8, -3.000_8, -3.000_8, -3.000_8, -2.970_8, -2.700_8, -2.700_8, -2.700_8, -2.700_8, &
1.004_8, 1.930_8, 1.930_8, 1.930_8, 1.930_8, 1.930_8, -2.334_8, -3.400_8, -2.585_8, 4.750_8, &
4.750_8, 4.750_8, 2.092_8, 0.320_8, 0.320_8, -0.778_8, -0.900_8, -0.900_8, -0.900_8, -0.900_8, &
-1.596_8, -3.220_8, -3.220_8, -3.220_8, -3.220_8, -3.220_8, -1.492_8, -1.300_8, -1.300_8, -1.300_8, &
-1.300_8, -1.300_8, -2.600_8, -2.600_8, -0.760_8, -0.300_8, -0.300_8, -0.300_8, -0.155_8, -0.010_8, &
-0.010_8, -0.010_8, -0.010_8, -0.126_8, -0.300_8, -0.300_8, 1.239_8, 1.410_8, 1.410_8, 1.410_8, &
1.410_8, 1.135_8, 0.860_8, 0.860_8, 0.860_8, 0.860_8, 0.860_8, 0.316_8, -1.860_8, -1.860_8, -1.860_8, &
-1.860_8, -1.860_8, -1.455_8, -1.410_8, -1.410_8, -1.410_8, -1.410_8, -1.680_8, -1.950_8, -1.950_8, &
0.122_8, 0.640_8, 0.540_8, -0.360_8, -0.360_8, -0.360_8, -0.360_8, -0.360_8, 0.249_8, 0.510_8, &
0.510_8, 0.510_8, 0.510_8, -0.456_8, -1.100_8, -1.100_8, -0.868_8, 0.060_8, 0.060_8, 0.060_8, &
0.060_8, 0.060_8, -0.090_8, -0.240_8, -0.240_8, -0.240_8, -0.240_8, -0.240_8, 0.036_8, 0.220_8, &
0.220_8, 0.745_8, 1.970_8, 1.970_8, 1.970_8, 1.970_8, 1.970_8, 1.970_8, 1.970_8, -0.100_8, &
-1.131_8, -1.220_8, -1.220_8, 1.006_8, 1.960_8, 1.960_8, 0.472_8, -0.520_8, -0.520_8, -0.520_8, &
-0.520_8, -0.520_8, -0.520_8, 0.400_8, 0.400_8, 0.400_8, 0.400_8, 0.400_8, 0.400_8, 0.400_8, &
1.582_8, 2.370_8, 2.370_8, 2.370_8, 2.370_8, 2.370_8, 2.370_8, 2.370_8, 1.998_8, 0.510_8, &
0.510_8, 0.510_8, 0.221_8, -2.380_8, -2.380_8, -2.380_8, -0.292_8, 0.230_8, 0.230_8, 0.288_8, &
0.520_8, 0.520_8, 0.520_8, 1.600_8, 1.870_8, 0.586_8, -2.410_8, -2.410_8, -2.410_8, -2.410_8, &
-2.410_8, -0.996_8, -1.122_8, -2.830_8, -2.830_8, -2.830_8, -2.830_8, -2.830_8, -2.830_8, -2.830_8, &
-2.830_8, -2.247_8, 3.000_8, 3.000_8, 3.000_8, 3.000_8, 3.000_8, 3.000_8, -0.576_8, -2.960_8, &
-2.960_8, -2.960_8, -2.960_8, -2.960_8, -1.786_8, 2.910_8, 2.910_8, 2.910_8, 2.910_8, 2.910_8, &
-0.830_8, -0.830_8, -0.830_8, -0.830_8, -0.768_8, -0.520_8, -0.520_8, -0.520_8, -0.520_8, -0.520_8, &
2.639_8, 2.990_8, 2.990_8, 2.990_8, 2.990_8, 2.990_8, -1.162_8, -2.200_8, -2.200_8, -2.200_8, &
-2.200_8, -2.200_8, -2.200_8, -2.200_8, -2.200_8, 1.080_8, 1.080_8, 1.080_8, -0.712_8, &
1.780_8, 4.000_8, 4.000_8, 4.000_8, -1.400_8, -5.000_8, -5.000_8, -3.135_8, -1.270_8, -1.270_8, &
-0.494_8, -0.300_8, 0.490_8, 0.490_8, 0.490_8, 0.490_8, 0.490_8, 0.669_8, 2.280_8, 2.280_8, &
2.280_8, 2.280_8, 2.280_8, 2.280_8, 2.280_8, 0.228_8, 0.000_8, 0.000_8, 0.000_8, 0.000_8 &

```

end module TESTCYCLE


```

*****
MAIN PROGRAM convDFC
*****
PROGRAM convDFC
-----
This program uses the GETARG function to acquire the
command line option. Please change this part by a suitable
method for your fortran90 compiler to use GETARG.
---for Compaq, Intel compiler---
USE DFLIB, ONLY:GETARG
---for Fujitsu compiler---
USE SERVICE_ROUTINES, ONLY:GETARG
-----

use modconst
use hvtransfc
use testcycle
use vehicleparams
use msubfc, only : calcfc, calcave, fcgrid
use dataio
use mshiftp, only : runmode, maxtqlin
implicit none

type (sspec) :: spec
type (sengine) :: eng
type (stransmission) :: tm
type (stqcurve) :: tqc, ftqc
type (sfcm) :: fcm
type (sgridmap) :: gfc
type (scycle) :: vpat, vpat2

! vehicle specification
! engine
! transmission
! maximum, frictional torque
! measured fuel map
! interpolated grid map
! test cycle

character (len = 1024) :: specf='', engf='', tmf=''
! spec, engine, transmission filename
character (len = 1024) :: tqf='', ftqf='', mapf=''
! maximum & frictional torque, map filename
character (len = 1024) :: outf=''
! output name
character (len = 1024) :: vhname='', tpname=''
! vehiclename, typename

real(8), allocatable :: vu(:), vh(:)
! vehicle speed(km/h)
real(8), allocatable :: neu(:), neh(:), nneu(:), nneh(:)
! engine speed(rpm), normalized
real(8), allocatable :: teu(:), teh(:), nteu(:), nteh(:)
! engine torque(Nm), normalized
real(8), allocatable :: fcu(:), fch(:)
! fuel(L/h)
integer, allocatable :: spu(:), sph(:)
! gear position

integer i, spp, spi, theld
real(8) vi, vp, eti, evi, verr, maxt
integer grs
! start gear
real(8), pointer :: gr(:), sgr(:)
! gear ratio (main, sub)
real(8) eu, eh, et, ece
real(8) avvu, avvh, avvt
integer tsout
! output time series data(1:Yes,0:No)

-----
Get Specification filename
-----
call GETARG(1, specf)
call GETARG(2, outf)
if (specf == '') then
  do while (specf == '')
    write(*, '(A,$)') 'TYPE FILENAME FOR INPUT : '
    read(*, *) specf
  end do
end if

if (outf == '') then
  do while (outf == '')
    write(*, '(A,$)') 'TYPE FILENAME FOR OUTPUT : '
    read(*, *) outf
  end do
end if

-----
read input data
-----
call readspecfc ( 11, specf, spec%rt, tm%fgr, tpname, vhname, engf, tmf, tsout )
call readgear ( 12, tmf, tm%nmgr, gr, tm%nsgr, sgr, grs, tm%at )
call readengfc ( 13, engf, mapf, tqf, ftqf, eng%idle, eng%rate, eng%nex )
call setparameters ( tpname, spec, eng, tm, gr, sgr, grs )

call readtqfc ( 14, trim(tqf), tqc )
! maximum torque
call readtqfc ( 15, trim(ftqf), ftqc )
! frictional torque
call readmapfc ( 17, trim(mapf), fcm )
! fuel map
call fcgrid ( gfc, fcm, ftqc )
! calculate grided fuel map
call showinputdata ( spec, eng, tm, vhname, tpname )
! display input dataset

-----
set test cycle for urban
-----
vpat%ndat = nurban
vpat%v => vurban
vpat%grade => gurban
allocate ( vu( vpat%ndat ) )
! calculated speed
allocate ( spu( vpat%ndat ) )
! shift position
allocate ( neu( vpat%ndat ) )
! engine speed (rpm)
allocate ( teu( vpat%ndat ) )
! engine torque (Nm)
allocate ( nneu( vpat%ndat ) )
! normalized engine speed (%)
allocate ( nteu( vpat%ndat ) )
! normalized engine torque (%)
allocate ( fcu( vpat%ndat ) )
! fuel consumption (l/h)

```

```

! calculate all gear position
! -----
spp = tm%ngr
theld = THOLD ! highest pos.
vp = vpat%v(1)
verr = 0.0_8
call runmode ( 1, vpat, vp, spp, theld, THOLD, spec, &
               eng, tm, tqc, evi, eti, spi, vi, verr ) ! calculate initial state

do i = 1, vpat%ndat
  spp = spi
  vp = vi
  verr = 0.0_8

  call runmode ( i, vpat, vp, spp, theld, THOLD, spec, &
                eng, tm, tqc, evi, eti, spi, vi, verr )
  maxt = maxtqlin ( tqc, evi )
  vu(i) = vi
  neu(i) = evi
  teu(i) = eti
  nneu(i) = ( neu(i) - eng%nidle ) &
            / ( eng%nrates - eng%nidle ) * 100.0_8
  nteu(i) = teu(i) / maxt * 100.0_8
  spu(i) = spi
end do

! -----
! calculate average fuel consumption
! -----
call calcfc ( neu, teu, spu, fcu, vpat%ndat, ftqc, gfc ) ! sum-up fuel consumption
call calcave ( vu, fcu, vpat%ndat, avvu, eu ) ! average (URBAN)
call calcave ( vu(645:), fcu(645:), 766, avvt, et ) ! average (MID-TOWN)

! -----
! set test cycle for highway
! -----
vpat2%ndat = nhighway
vpat2%v => vhighway
vpat2%grade => ghighway
allocate ( vh ( vpat2%ndat ) )
allocate ( sph ( vpat2%ndat ) )
allocate ( neh ( vpat2%ndat ) )
allocate ( teh ( vpat2%ndat ) )
allocate ( nneh ( vpat2%ndat ) )
allocate ( nteh ( vpat2%ndat ) )
allocate ( fch ( vpat2%ndat ) )

! -----
! calculate all gear position
! -----
theld = THOLD
spp = tm%ngr
vp = vpat2%v(1)
verr = 0.0_8
call runmode ( 1, vpat2, vp, spp, theld, THOLD, spec, &
               eng, tm, tqc, evi, eti, spi, vi, verr ) ! calculate initial state

do i = 1, vpat2%ndat
  spp = spi
  vp = vi
  verr = 0.0_8
  call runmode ( i, vpat2, vp, spp, theld, THOLD, spec, &
                eng, tm, tqc, evi, eti, spi, vi, verr )
  maxt = maxtqlin ( tqc, evi )
  vh(i) = vi
  neh(i) = evi
  teh(i) = eti
  nneh(i) = ( neh(i) - eng%nidle ) &
            / ( eng%nrates - eng%nidle ) * 100.0_8
  nteh(i) = teh(i) / maxt * 100.0_8
  sph(i) = spi
end do

! -----
! calculate average fuel consumption
! -----
call calcfc ( neh, teh, sph, fch, vpat2%ndat, ftqc, gfc ) ! sum-up fuel consumption
call calcave ( vh, fch, vpat2%ndat, avvh, eh ) ! average (INTERCITY)

print*, "-----"
if ( tm%at == 1 ) then
  eh = eh * 0.96_8
  eu = eu * 0.91_8
  et = et * 0.91_8
end if
ece = 1.0_8 / ( ( 1.0_8 - spec%ric ) / eu + spec%ric / eh )

print ' ( 2A, F8. 4, A, F0. 1 )', " URBAN, "FC(km/l) : ", eu, " Ave. Speed(km/h) : ", avvu
print ' ( 2A, F8. 4, A, F0. 1 )', " HIGHWAY, "FC(km/l) : ", eh, " Ave. Speed(km/h) : ", avvh
print ' ( 2A, F8. 4 )', " AVERAGE, "FC(km/l) : ", ece
print *
print ' ( 2A, F8. 4, A, F0. 1 )', " MID-TOWN, "FC(km/l) : ", et, " Ave. Speed(km/h) : ", avvt

! -----
! output calculation result
! -----
print*
print ' ( " OUTPUT FILE : ", A )', trim( outf )

```

```

call outaveragefc ( 21, outf, specf, vhname, tpname, engf, &
                    tmf, tm%fgf, spec%rt, avvu, eu, avvh, eh, avvt, et, spec%ric, ece )
if( tsout == 1 ) then
    call outtimeseriesfc (21, outf, vpat, vu, neu, teu, nneu, nteu, spu, fcu, 'URBAN')
    call outtimeseriesfc (21, outf, vpat2, vh, neh, teh, nneh, nteh, sph, fch, 'HIGHWAY')
end if

nullify ( vpat%v, vpat%grade, vpat2%v, vpat2%grade )
deallocate ( vu, spu, neu, teu, nneu, nteu, fcu )
deallocate ( vh, sph, neh, teh, nneh, nteh, fch )
deallocate ( eng%nel, tm%gri)
deallocate ( tqc%rev, tqc%tq, ftqc%rev, ftqc%tq )
deallocate ( fcm%rev, fcm%tq, fcm%fc, gfc%rev, gfc%tq, gfc%fc )

print*, ' complete.'

end program convDFC

```

```
module vehicleparams
```

```
interface
  subroutine setparameters( tpname, spec, eng, tm, gr, sgr, grs )
    use hvtransfc
    character (len=*) tpname
    type (stransmission) tm
    type (sengine) eng
    type (sspec) spec
    real(8), pointer :: gr(:), sgr(:)
    integer grs
  end subroutine

  subroutine showinputdata( spec, eng, tm, vname, tpname )
    use hvtransfc
    type (sspec) spec
    type (sengine) eng
    type (stransmission) tm
    character (len=*) vname, tpname
  end subroutine showinputdata
end interface
```

```
end module vehicleparams
```

```
!*****
! READ SPEC DATA, VEHICLE, ENGINE, TRANSMISSION, MAX&FRICTION TORQUE
!*****
```

```
subroutine setparameters( tpname, spec, eng, tm, gr, sgr, grs )
  use hvtransfc
  use modconst
  implicit none

  character (len=*) tpname
  type (stransmission) tm
  type (sengine) eng
  type (sspec) spec
  real(8), pointer :: gr(:), sgr(:)
  integer i, j, x, grs
  real(8) dwt, dwe
  real(8) nerange

  real(8) beta
  -----
  spec
  -----
  call setvehicletype( tpname, spec )

  spec%tcl = spec%rt * 2.0_8 * 3.14_8
  spec%gvw = spec%w0 + spec%wld + REAL(spec%crew, 8) * PSGW
  -----
  select case ( tpname(1:1) )
  case ( 'T' )
    spec%wt = spec%w0 + spec%wld / 2.0_8 + 55.0_8
    spec%mur = ( 0.00513_8 + 17.6_8 / spec%wt ) * 9.8_8
    spec%mu = ( 0.00299 * spec%bh * spec%bw - 0.000832_8 ) * 9.8_8
  case ( 'B' )
    spec%wt = spec%w0 + spec%crew * 55.0_8 / 2.0_8
    spec%mur = ( 0.00513_8 + 17.6_8 / spec%wt ) * 9.8_8
    spec%mu = ( 0.00299 * spec%bh * spec%bw - 0.000832_8 ) * 0.680_8 * 9.8_8
  case default
    write (0, '(3A)') &
      " Vehicle Type Error, [ ", trim(tpname), " ] is not defined."
    stop
  end select

  -----
  engine
  -----
  nerange = eng%nrte - eng%ndle
  eng%nes = PMEET * nerange + eng%ndle
  eng%nc = PRELEASE * nerange + eng%ndle
  -----
  transmission
  -----
  tm%grs = grs * tm%nsgr
  if( tm%at == 1 ) then
    tm%grb = 1
  else
    tm%grb = ( GRB - 1 ) * tm%nsgr + 1
  end if
  tm%skiplimit = ( skiplimit - 1 ) * tm%nsgr
  tm%efgr = EGR_ND
  tm%ngr = tm%nmgr * tm%nsgr
  -----
  parameter setting for each gear
  -----
  allocate ( tm%gri( tm%ngr ) )
  allocate ( eng%nel( tm%ngr ) )
  dwt = spec%w0 * PDWT
  dwe = spec%w0 * PDWE
  -----
  ! for gear ratio
  ! for minimum speed
  ! rotational inertia of tire
  ! rotational inertia of engine

  ! T10, T11, TT1の回転慣性質量補正
```

```

!
if ( tpname == 'T11' .or. tpname == 'T10' .or. tpname == 'T11' ) then
  if ( tm%nsgr == 1 .and. tm%nmgr == 7 ) then
    if ( gr(7) == 1.0_8 ) then
      beta = 0.6_8
    else if ( gr(6) == 1.0_8 ) then
      beta = 0.81_8
    else
      beta = 1.0_8
    end if
  else if ( tm%nsgr == 2 .and. tm%nmgr == 6 ) then
    if ( sgr(2) == 1.0_8 .and. gr(6) == 1.0_8 ) then
      beta = 0.6_8
    else
      beta = 1.0_8
    end if
  else
    beta = 1.0_8
  end if
end if

x = 1
do i = 1, tm%nmgr
  do j = 1, tm%nsgr

    tm%gri(x)%gr = gr(i) * sgr(j)
    tm%gri(x)%dw = dwt + dwe * ( tm%gri(x)%gr * beta )**2
    ! gear ratio
    ! rotational mass

    if ( gr(i) == 1.0_8 ) then
      tm%gri(x)%egr = EGR_DIRECT
    else
      tm%gri(x)%egr = EGR_ND
    end if
    ! transmission efficiency

    if ( spec%gvw >= 8000.0_8 ) then
      if ( i >= grs+2 ) then
        tm%gri(x)%tmargin = tmup8t(3)
      else if ( i <= grs ) then
        tm%gri(x)%tmargin = tmup8t(1)
      else
        tm%gri(x)%tmargin = tmup8t(2)
      end if
    else
      if ( i >= grs+2 ) then
        tm%gri(x)%tmargin = tmunder8t(3)
      else if ( i <= grs ) then
        tm%gri(x)%tmargin = tmunder8t(1)
      else
        tm%gri(x)%tmargin = tmunder8t(2)
      end if
    end if
    ! GVW < 8t

    if ( i >= grs+3 ) then
      eng%nel(x) = nmins(4) * nerange + eng%nidle
    else if ( i <= grs ) then
      eng%nel(x) = nmins(1) * nerange + eng%nidle
    else
      eng%nel(x) = nmins(i-grs+1) * nerange + eng%nidle
    end if
    ! minimum speed

    x = x + 1
  end do
end do
end subroutine

```

```

!*****
! SHOWINPUTDATA
! Display vehicle spec & input parameters
!*****
subroutine showinputdata( spec, eng, tm, vhname, tpname )
  use hvtransfc
  implicit none

  type (sspec) spec
  type (sengine) eng
  type (stransmission) tm
  character (len=*) vhname, tpname
  integer i

  print*, "-----"
  print '(2A)', " Vehicle Name      :", trim(vhname)
  select case ( tpname(1:1) )
  case ( 'T' )
    print '(A)', " Vehicle type   : TRUCK"
  case ( 'B' )
    print '(A)', " Vehicle type   : BUS"
  end select
  print '(2A)', " Category       :", trim(tpname)
  if ( tm%at == 1 ) then
    print '(A)', " Transmission  : Torque Converter AT"
  else
    print '(A)', " Transmission  : MT, AMT"
  end if
end subroutine

```

```

end if
print ' (A,F0.2,A)', " HIGHWAY RATIO : ", spec%ric * 100.0_8, " [%]"
print*

print* "-----"
print ' (A,F8.2,A)', " GVW =", spec%gvw, "[kg]"
print ' (A,F8.2,A,F8.2,A)', " Wcurb =", spec%w0, "[kg], Wtest =", spec%wt, "[kg]"
print ' (A,F8.3,A,F8.3,A,F8.3,A)', " Width =", spec%bw, "[m], Height =", spec%bh, "[m], Tire radius =", spec%rt, "[m]"
print ' (A,I3)', " Crew =", spec%crew
print*
print ' (A,F8.2,A,F8.2,A,F8.2,A)', " Nidle =", eng%nidle, "[rpm], Nrate =", eng%nrte, "[rpm], Nex =", eng%nex, "[rpm]"
print ' (A,F8.2,A,F8.2,A)', " Nes =", eng%nes, "[rpm], Nec =", eng%nec, "[rpm]"
print ' (A,F10.6,A)', " MuAir =", spec%muair, "[N/(km/h)^2]"
print ' (A,F10.6,A)', " MuRoll =", spec%mur, "[N/kg]"
print*

print ' (A,I3)', " NUMBER OF GEAR =", tm%ngr
print ' (A)', " GEAR RATIO EFFICIENCY TORQ MARGIN DW[kg]"
do i = 1, tm%ngr
    print ' (I4,A,F8.3,F10.3,F12.3,F15.5)', i, ": ", tm%gri(i)%gr, &
        tm%gri(i)%egr, tm%gri(i)%tmargin, tm%gri(i)%dw
end do
print ' (A,F8.3,F10.3)', " FIN: ", tm%fgr, tm%efgr
print*

end subroutine showinputdata

! *****
! SUBROUTINE TOUPPER
! convert small letters to capital letters
! buf : converted strings
! *****
subroutine toUpper( buf )
    implicit none
    integer strlen, i
    character ( len = * ) buf

    strlen = len( buf )
    do i = 1, strlen
        if( ichar(buf(i:i))>=97 .and. ichar(buf(i:i))<=122 ) then
            buf(i:i) = char(ichar(buf(i:i))-32)
        end if
    end do
end subroutine

! *****
! SUBROUTINE SETVEHICLETYPE
! set parameters from standard vehicle category name
! typename : vehicle category name
! spec : vehicle spec
! *****
subroutine setvehicletype( categoryname, spec )
    use hvtransfc
    implicit none
    type (sspec) :: spec
    character (len=*) :: categoryname

    call toUpper( categoryname ) ! convert category string to capital

    select case ( categoryname )
    ! ----- TRUCK TRACTOR, "Tx" -----
    case ("TT1") ! TT1 GVW <=20t
        spec%w0 = 10525.0_8
        spec%wld = 24000.0_8
        spec%crew = 2
        spec%bh = 2.927_8
        spec%bw = 2.490_8
        spec%ric = 0.2_8

    case ("TT2") ! TT2 GVW >20t
        spec%w0 = 19028.0_8
        spec%wld = 40000.0_8
        spec%crew = 2
        spec%bh = 2.890_8
        spec%bw = 2.490_8
        spec%ric = 0.1_8

    ! ----- GENERAL TRUCK, "Tx" -----
    case ("T1") ! T1 3.5t< GVW <= 7.5t
        ! Payload <=1.5t
        spec%w0 = 1957.0_8
        spec%wld = 1490.0_8
        spec%crew = 3
        spec%bh = 1.982_8
        spec%bw = 1.695_8
        spec%ric = 0.1_8

    case ("T2") ! T2 3.5t< GVW <= 7.5t
        ! 1.5t < Payload <=2t
        spec%w0 = 2356.0_8
        spec%wld = 2000.0_8
        spec%crew = 3

```

```

spec%bh = 2.099_8
spec%bw = 1.751_8
spec%ric = 0.1_8

case ("T3")
spec%w0 = 2652.0_8
spec%wld = 2995.0_8
spec%crew = 3
spec%bh = 2.041_8
spec%bw = 1.729_8
spec%ric = 0.1_8
! T3 3.5t< GVW <= 7.5t
! 2t < Payload <=3t

case ("T4")
spec%w0 = 2979.0_8
spec%wld = 3749.0_8
spec%crew = 3
spec%bh = 2.363_8
spec%bw = 2.161_8
spec%ric = 0.1_8
! T4 3.5t< GVW <= 7.5t
! 3t < Payload

case ("T5")
spec%w0 = 3543.0_8
spec%wld = 4275.0_8
spec%crew = 2
spec%bh = 2.454_8
spec%bw = 2.235_8
spec%ric = 0.1_8
! T5 7.5t< GVW <= 8t

case ("T6")
spec%w0 = 3659.0_8
spec%wld = 5789.0_8
spec%crew = 2
spec%bh = 2.625_8
spec%bw = 2.239_8
spec%ric = 0.1_8
! T6 8t< GVW <= 10t

case ("T7")
spec%w0 = 4048.0_8
spec%wld = 7483.0_8
spec%crew = 2
spec%bh = 2.541_8
spec%bw = 2.350_8
spec%ric = 0.1_8
! T7 10t< GVW <= 12t

case ("T8")
spec%w0 = 4516.0_8
spec%wld = 7992.0_8
spec%crew = 2
spec%bh = 2.572_8
spec%bw = 2.379_8
spec%ric = 0.1_8
! T8 12t< GVW <= 14t

case ("T9")
spec%w0 = 5533.0_8
spec%wld = 8900.0_8
spec%crew = 2
spec%bh = 2.745_8
spec%bw = 2.480_8
spec%ric = 0.1_8
! T9 14t< GVW <= 16t

case ("T10")
spec%w0 = 8688.0_8
spec%wld = 11089.0_8
spec%crew = 2
spec%bh = 3.049_8
spec%bw = 2.490_8
spec%ric = 0.1_8
! T10 16t< GVW <= 20t

case ("T11")
spec%w0 = 8765.0_8
spec%wld = 15530.0_8
spec%crew = 2
spec%bh = 2.934_8
spec%bw = 2.490_8
spec%ric = 0.3_8
! T11 20t< GVW

! ----- ROOT BUS, CREW>=11persons, "BRx" -----
case ("BR1")
spec%w0 = 5186.0_8
spec%wld = 0.0_8
spec%crew = 39
spec%bh = 2.880_8
spec%bw = 2.072_8
spec%ric = 0.0_8
! BR1 6t< GVW <= 8t

case ("BR2")
spec%w0 = 6672.0_8
spec%wld = 0.0_8
spec%crew = 46
spec%bh = 2.947_8
spec%bw = 2.301_8
spec%ric = 0.0_8
! BR2 8t< GVW <= 10t

case ("BR3")
spec%w0 = 7324.0_8
spec%wld = 0.0_8
spec%crew = 62
spec%bh = 2.949_8
! BR3 10t< GVW <= 12t

```

```

spec%bw = 2.304_8
spec%ric = 0.0_8

case ("BR4")
spec%w0 = 8654.0_8
spec%wld = 0.0_8
spec%crew = 77
spec%bh = 2.969_8
spec%bw = 2.385_8
spec%ric = 0.0_8
! BR4 12t< GVW <= 14t

case ("BR5")
spec%w0 = 9790.0_8
spec%wld = 0.0_8
spec%crew = 79
spec%bh = 2.962_8
spec%bw = 2.490_8
spec%ric = 0.0_8
! BR5 14t< GVW

! ----- GENERAL BUS, CREW>=11persons, "Bx" -----
case ("B1")
spec%w0 = 3543.0_8
spec%wld = 0.0_8
spec%crew = 29
spec%bh = 2.593_8
spec%bw = 2.027_8
spec%ric = 0.1_8
! B1 3.5t< GVW <= 6t

case ("B2")
spec%w0 = 5622.0_8
spec%wld = 0.0_8
spec%crew = 29
spec%bh = 3.019_8
spec%bw = 2.197_8
spec%ric = 0.1_8
! B2 6t< GVW <= 8t

case ("B3")
spec%w0 = 6608.0_8
spec%wld = 0.0_8
spec%crew = 49
spec%bh = 3.105_8
spec%bw = 2.314_8
spec%ric = 0.1_8
! B3 8t< GVW <= 10t

case ("B4")
spec%w0 = 8022.0_8
spec%wld = 0.0_8
spec%crew = 58
spec%bh = 3.160_8
spec%bw = 2.399_8
spec%ric = 0.1_8
! B4 10t< GVW <= 12t

case ("B5")
spec%w0 = 9774.0_8
spec%wld = 0.0_8
spec%crew = 60
spec%bh = 3.168_8
spec%bw = 2.490_8
spec%ric = 0.1_8
! B5 12t< GVW <= 14t

case ("B6")
spec%w0 = 12110.0_8
spec%wld = 0.0_8
spec%crew = 62
spec%bh = 3.320_8
spec%bw = 2.490_8
spec%ric = 0.35_8
! B6 14t< GVW <= 16t

case ("B7")
spec%w0 = 14583.0_8
spec%wld = 0.0_8
spec%crew = 51
spec%bh = 3.668_8
spec%bw = 2.490_8
spec%ric = 0.35_8
! B7 16t< GVW

case default
write (0,'(3A)') " Category Error : [ ", trim(categoryname), " ] is not defined."
stop
end select

end subroutine

```



```

module mshiftp

interface

recursive subroutine runmode ( i, vpat, vp, sp, sht, nr, spec, eng, tm, tqc, evi, eti, spi, vi, verr )
  use hvtransfc
  type ( sspec ) :: spec
  type ( sengine ) :: eng
  type ( stransmission ) :: tm
  type ( stqcurve ) :: tqc
  type ( scycle ) :: vpat
  real(8) vi, evi, eti, vp, verr
  integer nr, i, sp, spi, sht
end subroutine

subroutine sftdwn ( i, vpat, vp, sp, spec, eng, tm, tqc, cnt, spout, verr )
  use hvtransfc
  type ( sspec ) :: spec
  type ( sengine ) :: eng
  type ( stransmission ) :: tm
  type ( stqcurve ) :: tqc
  type ( scycle ) :: vpat
  integer i, sp, spout, cnt
  real(8) vp, verr
end subroutine

subroutine sftuptm ( reacc, i, vpat, vp, sp, spec, eng, tm, tqc, cnt, spout, verr )
  use hvtransfc
  type ( sspec ) :: spec
  type ( sengine ) :: eng
  type ( stransmission ) :: tm
  type ( stqcurve ) :: tqc
  type ( scycle ) :: vpat
  integer i, sp, spout, cnt, reacc
  real(8) vp, verr
end subroutine

subroutine sftup ( reacc, i, vpat, vp, sp, spec, eng, tm, tqc, spout, mkt )
  use hvtransfc
  type ( sspec ) :: spec
  type ( sengine ) :: eng
  type ( stransmission ) :: tm
  type ( stqcurve ) :: tqc
  type ( scycle ) :: vpat
  integer i, sp, spout, reacc, mkt
  real(8) vp
end subroutine

integer function startgear ( i, vpat, spec, eng, tm, tqc, stime )
  use hvtransfc
  type ( sspec ) :: spec
  type ( sengine ) :: eng
  type ( stransmission ) :: tm
  type ( stqcurve ) :: tqc
  type ( scycle ) :: vpat
  integer i, stime
end function startgear

subroutine engstat ( ev, et, sp, spec, eng, tm, tqc, vdest, grade, vp, vv, maxt, sw )
  use hvtransfc
  type ( sspec ) :: spec
  type ( sengine ) :: eng
  type ( stransmission ) :: tm
  type ( stqcurve ) :: tqc
  real(8) ev, et, vdest, maxt, vp, vv, grade
  integer sp, sw
end subroutine engstat

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

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

end interface

end module mshiftp

```

```

! *****
! SUBROUTINE RUNMODE
! calculation of gear position & vehicle running conditions
! i : index
! vpat : test cycle
! vp : past speed (km/h)
! sp : past gear position
! theld : gear holding time (s)

```

```

! nrecur : recursion counter
! spec : vehicle spec
! eng : engine data
! tm : transmission data
! tqc : maximum torque curve
! evi : output engine speed
! eti : output engine torque
! spi : output gear position
! vi : output vehicle speed (km/h)
! verr : cumulative velocity error
! *****
recursive subroutine runmode ( i, vpat, vp, sp, theld, nrecur, spec, eng, tm, tqc, &
                             evi, eti, spi, vi, verr )

    use modconst
    use hvtransfc
    use mshiftp, only : sftup, sftuptm, sftdwn, startgear
    implicit none

    type (scycle) :: vpat
    type (sspec) :: spec
    type (sengine) :: eng
    type (stransmission) :: tm
    type (stqcurve) :: tqc
    integer i, sp, theld, nrecur, spi

    real(8) vp, evi, eti, vi, verr

    real(8) errmin
    integer sw
    integer stime
    real(8) maxt
    real(8) dv
    integer reacc
    integer spi2, theld2, ct, sw2, s2
    real(8) ev2, et2, vi2, verr2
    integer mkt

    ! minimum error
    ! engine condition return code
    ! starting time length
    ! maximum engine torque
    ! velocity difference
    ! reacceleration (0:OFF, 1:ON)
    ! for recursion
    ! estimated minimum keep time

! ===== DETECT END OF RECURSIVE CALCULATION =====
    ct = nrecur - 1
    if ( ct < 0 .or. i > vpat%ndat ) then
        spi = sp
        verr = verr
        return
    end if

! ===== VDEST(I) = 0, VEHICLE STOPS =====
    if ( vpat%v(i) <= 0.0_8 ) then
        ! vehicle stops
        vi = vpat%v(i)
        evi = eng%idle
        eti = 0.0_8
        spi = 0
        verr = verr
        + ( vpat%v(i-1) + vpat%v(i) ) / 2.0_8
        - ( vi + vp ) / 2.0_8
        return
    end if

! ===== VDEST(I) > 0 =====
    mkt = THOLD
    errmin = 0.0_8
    reacc = 0
    dv = vpat%v(i) - vp
    ! reacceleration

! ===== PAST SHIFT POSITION WAS NEUTRAL =====
    if ( sp == 0 ) then
        ! ===== VEHICLE STARTS TO MOVE, DETERMINE GEAR POSITION =====
        if ( vp == 0.0_8 ) then
            spi = startgear ( i, vpat, spec, eng, tm, tqc, stime )
            theld = THOLD - stime
            call engstat ( evi, eti, spi, spec, eng, tm, tqc, &
                          vpat%v(i), vpat%grade(i), vp, vi, maxt, sw2 )
        ! ===== REACCELERATION MODE =====
        else if ( dv >= 0 ) then
            reacc = 1
            theld = THOLD
            spi = tm%grb
            ! shift to bottom gear

! ===== DECELERATE WITH NEUTRAL POSITION =====
        else
            vi = vpat%v(i)
            evi = eng%idle
            eti = 0.0_8
            theld = THOLD
            verr = verr
            + ( vpat%v(i-1) + vpat%v(i) ) / 2.0_8
            - ( vi + vp ) / 2.0_8
            return
        end if

! ===== PAST SHIFT POSITION WAS NOT NEUTRAL =====
    else
        ! ----- check engine speed -----
        ev2 = vpat%v(i) * 60.0_8 * tm%gri(sp)%gr
            * tm%fgr / ( 3.6_8 * spec%tcl )
        &

        if ( ( dv >= 0.0_8 ) .and.
            ( ev2 < eng%nel(sp) ) .and.
            ( sp > tm%grs ) ) then
            spi = tm%grb
            & ! acc>=0
            & ! ev2 < lower limit
            & ! sp > start gear
            & ! shift to bottom gear
        end if
    end if

```

```

        reacc = 1                                ! reacceleration mode ON
    else
        spi = sp                                  ! normal condition
    end if
end if

! ===== CALCULATE ENGINE RUNNING CONDITION =====
call engstat ( evi, eti, spi, spec, eng, tm, tqc,      &
               vpat%v(i), vpat%grade(i), vp, vi, maxt, sw )

! ===== CASE THAT CANNOT CHANGE GEAR, DECELERATION =====
if ( ( theld < THOLD ) .or. ( dv < 0.0_8 ) ) then

    if ( sw == 4 .or. sw == 3 ) then                ! clutch off(SW=4) or lower limit(SW=3)
        spi = 0                                     ! neutral
        theld = THOLD                               ! enable gear change
        return
    end if

    if ( sw == 2 .and. spi < tm%ngr ) then           ! overrun(SW=2)
        vi = 0.0_8                                  ! cannot continue calculation
        return                                       ! (RETURN ABNORMAL RESULT)
    end if

    verr2 = verr
             + ( vpat%v(i-1) + vpat%v(i) ) / 2.0_8    &
             - ( vi + vp ) / 2.0_8                  &
    theld2 = theld + 1
    call runmode ( i+1, vpat, vi, spi, theld2, ct, spec,      &
                  eng, tm, tqc, ev2, et2, spi2, vi2, verr2 )
    verr = verr2
    theld = theld + 1                                     ! increment gear hold time
    return
end if

! ===== CASE THAT CAN CHANGE GEAR POSITION =====
! ===== REACCELERATION =====
if ( reacc == 1 ) then
    ! ----- TRY SHIFT-UP (WITH TORQUE MARGIN) -----
    call sftuptm ( reacc, i, vpat, vp, spi, spec,          &
                  eng, tm, tqc, ct, spi2, verr2 )
    if ( spi < spi2 ) then
        spi = spi2
    ! ----- FIND OPTIMAL GEAR THAT CAN KEEP 3 SECONDS -----
    else
        call sftup( reacc, i, vpat, vp, spi, spec,          &
                   eng, tm, tqc, s2, mkt )
        spi = s2
    end if
end if

! ===== OVER RUN. SHIFT-UP REQUIRED =====
else if ( sw == 2 ) then
    if ( spi < tm%ngr ) then
        ! ----- TRY SHIFT-UP (WITH TORQUE MARGIN) -----
        call sftuptm ( reacc, i, vpat, vp, spi, spec,      &
                      eng, tm, tqc, ct, s2, verr2 )
        ! ----- FIND OPTIMAL GEAR THAT CAN KEEP 3 SECONDS -----
        if ( spi == s2 ) then
            call sftup( reacc, i, vpat, vp, spi, spec,      &
                       eng, tm, tqc, s2, mkt )
        end if
        ! ----- ENGINE RUNNING CONDITION -----
        call engstat ( evi, eti, s2, spec, eng, tm, tqc,    &
                      vpat%v(i), vpat%grade(i), vp, vi, maxt, sw )
        spi = s2
        theld2 = 1 + ( THOLD - mkt )
    else
        theld2 = theld + 1                                ! top gear
    end if
    ! ----- CALCULATE CUMULATIVE ERROR -----
    verr2 = verr
             + ( vpat%v(i-1) + vpat%v(i) )              &
             / 2.0_8 - ( vi + vp ) / 2.0_8              &
    call runmode ( i+1, vpat, vi, spi, theld2, ct, spec,    &
                  eng, tm, tqc, ev2, et2, spi2, vi2, verr2 )
    errmin = verr2

! ===== ALL OTHER CASES =====
else
    ! ----- KEEPS SAME GEAR POSITION -----
    theld2 = theld + 1
    verr2 = verr
             + ( vpat%v(i-1) + vpat%v(i) )              &
             / 2.0_8 - ( vi + vp ) / 2.0_8              &
    call runmode ( i+1, vpat, vi, spi, theld2, ct, spec,    &
                  eng, tm, tqc, ev2, et2, spi2, vi2, verr2 )
    errmin = verr2

    ! ----- TRY SHIFT-UP (WITH TORQUE MARGIN) -----
    call sftuptm ( reacc, i, vpat, vp, spi, spec, eng,      &
                  tm, tqc, ct, spi2, verr2 )
    if ( spi2 > spi .and. errmin >= verr+verr2 ) then
        spi = spi2
        errmin = verr+verr2
    end if
end if

```

```

! ----- TRY SHIFT-DOWN -----
! if ( sp == spi .and. sw == 1 ) then
!   call sftdwn ( i, vpat, vp, spi, spec, eng,      &
!               tm, tqc, ct, s2, verr2 )
!   if ( spi > s2 .and. errmin >= verr+verr2 ) then
!     spi = s2
!     errmin = verr+verr2
!   end if
! end if
end if

! ===== FINAL RUNNING CONDITION =====
call engstat ( evi, eti, spi, spec, eng, tm, tqc,      &
              vpat%v(i), vpat%grade(i), vp, vi, maxt, sw )

if ( dv >= 0.0_8 .and. evi < eng%nes ) then           ! reacc by bottom gear
  evi = eng%nes                                       ! keep starting speed
else if ( evi < eng%nc ) then                         ! clutch off
  evi = eng%idle
  eti = 0.0_8
  spi = 0
end if

if ( spi == 0 ) then                                  ! neutral
  theld = THOLD                                       ! enable gear change
else if ( sp /= spi ) then
  theld = 1 + ( THOLD - mkt )                         ! reset hold time
else
  theld = theld + 1                                   ! increment hold time
end if
verr = errmin

end subroutine runmode

```

```

! *****
! FUNCTION STARTGEAR
! calculation of starting gear
! i : index
! vpat : test cycle
! spec : vehicle spec
! eng : engine data
! tm : transmission data
! tqc : maximum torque curve
! stime : time length of starting
! startgear : starting gear
! *****
integer function startgear ( i, vpat, spec, eng, tm, tqc, stime )
  use hvtransfc
  use mshiftp, only : engstat
  implicit none

  type (sspec) :: spec
  type (sengine) :: eng
  type (stransmission) :: tm
  type (stqcurve) :: tqc
  type (scycle) :: vpat
  integer i, stime
  integer i2, flg, flg2, sp, sw
  real(8) ev, et, vi, maxt

  flg2 = 0
  do sp = tm%grs, 1, -1                                ! test starting gear
    i2 = i
    flg = 0
    stime = 0
    startgear = sp
    do while ( flg == 0 )
      stime = stime + 1                                ! predict running condition
      call engstat ( ev, et, sp, spec, eng, tm, tqc, vpat%v(i2), &
                    vpat%grade(i2), vpat%v(i2-1), vi, maxt, sw )

      if ( sw == 0 .or. sw == 2 ) then
        flg = 1                                         ! normal(0) or overrun(2)
        flg2 = 1                                       ! fix startgear
      else if ( sw == 1 .or. sw == 3 ) then
        flg = 1                                         ! lack of torque(1), out of range(3)
      else
        i2 = i2 + 1                                     ! continue
      end if
    end do
    if ( flg2 == 1 ) exit                               ! exit loop
  end do
end function startgear

```

```

! *****
! SUBROUTINE SFTUPTM
! calculation of shift-up (with torque margin)
! reacc : reacceleration mode ( 1: reacceleration, 0: non )
! i : index
! vpat : test cycle
! vp : past speed
! sp : past gear position
! spec : vehicle spec

```

```

! eng : engine data
! tm : transmission data
! tqc : maximum torque curve
! cnt : recursion counter
! spout : output gear position
! minerr : cumulative error
! *****
subroutine sftuptm( reacc, i, vpat, vp, sp, spec, eng, tm, tqc, cnt, spout, minerr )
  use hvtransfc
  use mshiftp, only : engstat
  implicit none

  type (sspec) :: spec
  type (sengine) :: eng
  type (stransmission) :: tm
  type (stqcurve) :: tqc
  type (scycle) :: vpat
  integer reacc, i, sp, cnt, spout
  real(8) vp, minerr
  integer s, sw, x
  real(8) verr, vold, vv, tqmargin
  real(8) ev, et, maxt

  spout = sp
  minerr = 0.0_8

  do s = sp + 1, tm%ngr
    verr = 0.0_8

    if ( ( s > sp + tm%skiplimit ) .and. ( reacc == 0 ) ) then
      exit ! check skip limit
    end if

    call engstat( ev, et, s, spec, eng, tm, tqc, &
      vpat%v(i), vpat%grade(i), vp, vv, maxt, sw )

    if ( sw == 2 .or. sw == 3 .or. sw == 5 ) cycle ! overrun(2), out-of-range(3), starting(5)

    if ( et > 0.0_8 ) then
      tqmargin = maxt / et ! torque margin
    else
      tqmargin = 100.0_8 ! if (te < 0)
    end if

    if ( tqmargin >= tm%gri(s)%tmargin .and. sw == 0 ) then ! try shift up
      vold = vp
      do x = 0, cnt ! predict running condition
        call engstat ( ev, et, s, spec, eng, tm, tqc, vpat%v(i+x), &
          vpat%grade(i+x), vold, vv, maxt, sw )
        if( sw == 2 .or. sw == 3 ) exit ! overrun(2), out-of-range(3)
        verr = verr &
          + ( vpat%v(i+x) + vpat%v(i+x-1) ) / 2.0_8 &
          - ( vv + vold ) / 2.0_8
        vold = vv
      end do

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

```

```

! *****
! SUBROUTINE SFTUP
! calculation of shift-up (except torque margin)
! reacc : reacceleration mode ( 1: reacceleration, 0: non )
! i : index
! vpat : test cycle
! vp : past speed
! sp : past gear position
! spec : vehicle spec
! eng : engine data
! tm : transmission data
! tqc : maximum torque curve
! spout : output gear position
! mkt : estimated minimum gear hold time
! *****
subroutine sftup ( reacc, i, vpat, vp, sp, spec, eng, tm, tqc, spout, mkt )
  use hvtransfc
  use mshiftp, only : engstat
  use modconst
  implicit none

  type (sspec) :: spec
  type (sengine) :: eng
  type (stransmission) :: tm
  type (stqcurve) :: tqc
  type (scycle) :: vpat

```

```

integer reacc, i, sp, spout, mkt
real(8) vp
integer x, y, z, sw
real(8) verr, minerr, vold, vv, maxt, ev, et

spout = sp
do x = THOLD, 1, -1                                ! gear hold time
  minerr = 0.0_8
  do y = sp + 1, tm%ngr                             ! gear position
    vold = vp
    verr = 0.0_8

    do z = 0, x - 1                                  ! predict running condition
      call engstat( ev, et, y, spec, eng, tm, tqc, &
                   vpat%v(i+z), vpat%grade(i+z), &
                   vold, vv, maxt, sw )
      if( sw == 2 .or. sw == 3 ) exit                ! overrun(2), out-of-range(3)
      verr = verr + ( vpat%v(i+z) + vpat%v(i+z-1) ) / 2.0_8 &
      - ( vv + vold ) / 2.0_8
      vold = vv
    end do

    if( sw == 2 .or. sw == 3 ) then                  ! overrun(2), out-of-range(3)
      cycle
    else if ( spout == sp ) then
      spout = y
      minerr = verr
      if ( minerr == 0.0_8 .and. reacc == 0 ) exit
    else if ( minerr > verr ) then
      spout = y
      minerr = verr
    end if
  end do
  if ( spout /= sp ) then
    mkt = x
    exit
  end if
end do
end subroutine sftup

```

```

! *****
! SUBROUTINE SFTDWN
! calculation of shift down
! i : index
! vpat : test cycle
! vp : past speed
! sp : past gear position
! spec : vehicle spec
! eng : engine data
! tm : transmission data
! tqc : maximum torque curve
! cnt : recursion counter
! spout : output gear position
! minerr : cumulative error
! *****
subroutine sftdwn ( i, vpat, vp, sp, spec, eng, tm, tqc, cnt, spout, minerr )
  use hvtransfc
  use mshiftp, only : engstat
  implicit none

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

  spout = sp                                ! initial position
  do s = tm%grb, sp - 1
    vold = vp
    verr = 0.0_8
    do x = 0, cnt
      call engstat ( ev, et, s, spec, eng, tm, tqc, vpat%v(i+x), &
                   vpat%grade(i+x), vold, vnew, maxt, sw )

      if( sw == 2 .or. sw == 3 ) exit          ! overrun(2), out-of-range(3)
      verr = verr + ( vpat%v(i+x) + vpat%v(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. minerr >= verr ) then
      spout = s
      minerr = verr
    end if
  end do
end subroutine sftdwn

```

```

! *****
! FUNCTION DRVFRFC
! calculate driving force
! spec : vehicle spec
! mis : transmission
! sp : gear position
! v1 : current speed (km/h)
! v2 : target speed (km/h)
! grade : grade (%)
! *****
real(8) function drvfrfc ( spec, tm, sp, v1, v2, grade )
  use hvtransfc
  implicit none

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

  rr = 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

  drvfrfc = rr + mass * acc + fgrade ! driving force (N)
end function drvfrfc

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

  type ( stqcurve ) :: tqc ! torque curve array
  real(8) rev ! input engine speed, rpm
  integer x ! index

  if ( rev < tqc%rev(1) ) then
    x = 1 ! rev<r[1]
  else if ( rev >= tqc%rev(tqc%ndata) ) then
    x = tqc%ndata - 1 ! rev>=r[ndata]
  else
    do x = 1, tqc%ndata
      if ( rev >= tqc%rev(x) .and. rev < tqc%rev(x+1) ) then
        exit
      end if
    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

! *****
! SUBROUTINE ENGSTAT
! calculate engine running condition
! output engine speed, engine torque,
! vehicle speed, engine condition
! ev : engine speed
! et : engine torque
! sp : gear position
! spec : vehicle spec
! eng : engine data
! tm : transmission data
! tqc : maximum torque curve
! vdest : target speed (km/h)
! grade : grade (%)
! vp : past speed (km/h)
! vv : output vehicle speed
! maxt : maximum torque
! 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
! sw=5 starting, Ne <= Nes, shift = 1st or 2nd
! *****
subroutine engstat ( ev, et, sp, spec, eng, tm, tqc, vdest, grade, vp, vv, maxt, sw )
  use hvtransfc
  use mshiftp, only : maxtqlin, drvfrfc
  implicit none

```



```

if ( df >= 0.0_8 ) then
    et = df * spec%rt / ( tgr * tegr )          ! engine torque(when driving)
else
    et = df * ( spec%rt * tegr ) / tgr          ! engine torque(when braking)
end if

ev = vv * 60.0_8 * tgr / ( spec%tcl * 3.6_8 )    ! engine speed

if ( sp > tm%grb .and. ev < eng%nel(sp) ) then
    maxt = maxtqlin ( tqc, eng%nel(sp) )          ! engine speed is under lower limit
else if ( ev < eng%nes ) then
    ev = eng%nes                                  ! starting engine speed
    maxt = maxtqlin ( tqc, ev )
else
    maxt = maxtqlin ( tqc, ev )
end if

tqdif = maxt - et
if ( ( tqdif < 1.0E-6 ) .and. ( tqdif >= 0.0_8 ) ) then ! check convergence calculation result
    flg = 1
else if ( tqdif < 0.0_8 ) then
    vv = vv - ds                                  ! decrease target speed
else
    ds = ds / 2.0_8                               ! increase target speed
    vv = vv + ds
end if
end do

! -----
! final engine running condition
! -----
if ( ev < eng%nel(sp) ) then
    vv = 0.0_8                                     ! engine speed is under the lower limit
    maxt = maxtqlin ( tqc, eng%nel(sp) )           ! cannot continue calculation
    sw = 3                                          ! [ RETURN CODE = 3 ], out of range
else if ( ev >= eng%nex ) then
    vv = 0.0_8                                     ! engine overrun
    maxt = maxtqlin ( tqc, eng%nex )               ! cannot continue calculation
    sw = 2                                          ! [ return code = 2 ], engine overrun
end if

end subroutine engstat

```

```

! *****
! MODULE MSUBFC
! definition of fuel consumption calculating subroutine
! *****
module msubfc

  interface

    subroutine fcgrid( gfc, fcm, ftqc )
      use hvtransfc
      type (sgridmap) gfc
      type (sfcm) fcm
      type (stqcurve) ftqc
    end subroutine fcgrid

    subroutine calcfc( rev, tq, sp, fc, n, ftqc, gfc )
      use hvtransfc
      type (sgridmap) gfc
      type (stqcurve) ftqc
      real(8) rev(:), tq(:), fc(:)
      integer n, sp(:)
    end subroutine calcfc

    subroutine calcave( v, fc, n, avv, avfc )
      integer n
      real(8) v(:), fc(:), avv, avfc
    end subroutine calcave

  end interface

end module msubfc

! *****
! SUBROUTINE CALCFC
! calculation of fuel consumption at every time step
! rev : engine speed array
! tq : engine torque array
! sp : shift position array
! fc : fuel consumption array (output)
! n : number of data
! ftqc : frictional torque
! gfc : grid fuel map
! *****
subroutine calcfc( rev, tq, sp, fc, n, ftqc, gfc )
  use hvtransfc
  use mshiftp, only : maxtqlin
  implicit none

  type ( sgridmap ) :: gfc
  type ( stqcurve ) :: ftqc
  real(8) :: rev(:), tq(:), fc(:)
  integer :: sp(:)
  integer n

  integer i, j, ierr
  real(8) ftq
  real(8), allocatable :: f(:), d2(:), d(:,:)
  logical skip

  allocate( d( gfc%nx, gfc%ny ) )
  allocate( f( gfc%ny ) )
  allocate( d2( gfc%ny ) )

  do j = 1, n
    if ( sp(j) == 0 ) then
      fc(j) = gfc%fcidle ! idling
      cycle
    end if

    ftq = maxtqlin( ftqc, rev(j) )
    if ( tq(j) <= ftq ) then
      fc(j) = 0.0_8
    else
      do i = 1, gfc%ny
        call DPCHIM( gfc%nx, gfc%rev, gfc%fc(1,i), d(1,i), 1, ierr )
        if( ierr < 0 ) then
          call errhermite ( 5, ierr )
        end if

        skip = .true.
        call DPCHFE( gfc%nx, gfc%rev, gfc%fc(1,i), d(1,i), &
                     1, skip, 1, rev(j), f(i), ierr ) ! torque=fuel curve
        if( ierr < 0 ) then
          call errhermite ( 6, ierr )
        end if
      end do

      skip=.true.
      call DPCHIM( gfc%ny, gfc%tq, f, d2, 1, ierr )
      if( ierr < 0 ) then
        call errhermite ( 7, ierr )
      end if

      call DPCHFE( gfc%ny, gfc%tq, f, d2, 1, skip, 1, tq(j), fc(j), ierr ) ! fc at tq(j)
      if( ierr < 0 ) then
        call errhermite ( 8, ierr )
      end if
    end if
  end do
end subroutine calcfc

```

```

        end if
        if( fc(j) < 0.0_8 ) then
            fc(j) = 0.0_8
        end if
    end if
end do

deallocate( d, d2, f )

end subroutine

```

```

! *****
! SUBROUTINE CALCAVE
! calculate average speed, average fuel consumption
! v : calculated speed
! fc : fuel consumption
! n : number of data
! avv : average speed
! avfc : average fuel consumption
! *****
subroutine calcave( v, fc, n, avv, avfc )
    implicit none

    integer n, i
    real(8) v(:), fc(:)
    real(8) avv, avfc, sumv, sumf

    sumv = 0.0_8
    sumf = 0.0_8
    do i = 1, n
        sumv = sumv + v(i)
        sumf = sumf + fc(i)
    end do

    avv = sumv / (real(n,8) * 1.0_8 )
    avfc = sumf / sumf

end subroutine

```

```

! *****
! SUBROUTINE FCGRID
! generate NXxNY grid fuel map
! gfcmap : type(sgridmap), grid fuel map
! fcm : type(sfcmap), input fuel map
! ftqc : type(stqcurve), frictional torque
! *****
subroutine fcgrid( gfcmap, fcm, ftqc )
    use hvtransfc
    use mshiftp, only : maxtqlin
    implicit none

    integer, parameter :: nx = 20, ny = 20
    type (stqcurve) :: ftqc
    type (sfcmap) :: fcm
    type (sgridmap) :: gfcmap

    integer i, j, ierr
    integer rn, tn, tnmax
    real(8), allocatable :: r(:), t(:, :), f(:, :), x(:)
    integer, allocatable :: tqn(:)
    real(8) nemin, nemax, tqmin, tqmax
    real(8) dne, dtq
    real(8) rr
    logical skip
    real(8), allocatable :: d1(:, :), f1(:, :)
    real(8), pointer :: pchval(:)

    allocate ( gfcmap%rev( nx ) )
    allocate ( gfcmap%tq ( ny ) )
    allocate ( gfcmap%fc ( nx, ny ) )
    gfcmap%nx = nx
    gfcmap%ny = ny
    gfcmap%fcidle = fcm%fcidle

    nemin = minval( fcm%rev )
    nemax = maxval( fcm%rev )
    tqmin = minval( fcm%tq )
    tqmax = maxval( fcm%tq )

    rn = 1
    tn = 1
    tnmax = 1
    rr = fcm%rev(1)
    do i = 1, fcm%ndata
        if ( rr /= fcm%rev(i) ) then
            rn = rn + 1
            rr = fcm%rev(i)
            if( tn > tnmax ) tnmax = tn
            tn = 1
        end if
        tn = tn + 1
    end do

```

! count points of input fuel map

```

allocate ( r(rn) )
allocate ( tqn(rn) )
allocate ( t(rn, tnmax) )
allocate ( f(rn, tnmax) )

j = 1
r(j) = fcm%rev(1)
tqn(j) = 1
t( j, tqn(j) ) = maxtqlin( ftqc, r(j) )
f( j, tqn(j) ) = 0.0_8
do i = 1, fcm%ndata
  if ( r(j) /= fcm%rev(i) ) then
    j = j + 1
    r(j) = fcm%rev(i)
    tqn(j) = 1
    t( j, tqn(j) ) = maxtqlin( ftqc, r(j) )
    f( j, tqn(j) ) = 0.0_8
    tqmin = min( tqmin, t(j, tqn(j)) )
  end if
  tqn(j) = tqn(j) + 1
  t( j, tqn(j) ) = fcm%tq(i)
  f( j, tqn(j) ) = fcm%fc(i)
end do

! -----
dne = ( nemax - nemin ) / real(nx-1, 8)
dtq = ( tqmax - tqmin ) / real(ny-1, 8)

do i = 1, nx
  gfc%rev(i) = nemin + dne * real(i-1, 8)
end do
do i = 1, ny
  gfc%tq(i) = tqmin + dtq * real(i-1, 8)
end do

! -----
allocate ( f1(rn, ny) )
allocate ( d1(rn, tnmax) )
do i = 1, rn
  allocate ( x( tqn(i) ) )
  do j = 1, tqn(i)
    x(j) = t( i, j )
  end do

  call DPCHIM( tqn(i), x, f(i,1), d1(i,1), rn, ierr )
  if( ierr < 0 ) call errhermite ( 1, ierr )

  skip = .true.
  allocate ( pchval( ny ) )
  call DPCHFE( tqn(i), x, f(i,1), d1(i,1), rn, skip,      &
    gfc%ny, gfc%tq, pchval, ierr )
  if( ierr < 0 ) call errhermite ( 2, ierr )

  do j = 1, ny
    f1(i, j) = pchval(j)
  end do

  deallocate ( pchval, x )
end do

deallocate ( d1 )

! -----
allocate ( d1(rn, ny) )
do i = 1, ny
  call DPCHIM( rn, r, f1(1,i), d1(1,i), 1, ierr )
  if( ierr < 0 ) call errhermite ( 3, ierr )

  skip = .true.
  allocate ( pchval( nx ) )
  call DPCHFE( rn, r, f1(1,i), d1(1,i), 1, skip,      &
    gfc%nx, gfc%rev, pchval, ierr )
  if( ierr < 0 ) call errhermite ( 4, ierr )

  do j = 1, nx
    gfc%fc(j, i) = pchval(j)
  end do

  deallocate ( pchval )
end do

deallocate ( d1, f1, r, tqn, t, f )

end subroutine

! *****
! SUBROUTINE ERRHERMITE
! error handler for hermite interpolation
! *****
subroutine errhermite ( errid, ierr )
  implicit none

  integer errid, ierr
  character (len=1024) msg

```

```

select case ( errid )
-----
at FCGRID
-----
case( 1 )
  msg = "Error : DPCHIM ( called from FCGRID, 1st )"
case( 2 )
  msg = "Error : DPCHFE ( called from FCGRID, 1st )"
case( 3 )
  msg = "Error : DPCHIM ( called from FCGRID, 2nd )"
case( 4 )
  msg = "Error : DPCHFE ( called from FCGRID, 2nd )"
-----
at CALCFC
-----
case( 5 )
  msg = "Error : DPCHIM ( called from CALCFC, 1st )"
case( 6 )
  msg = "Error : DPCHFE ( called from CALCFC, 1st )"
case( 7 )
  msg = "Error : DPCHIM ( called from CALCFC, 2nd )"
case( 8 )
  msg = "Error : DPCHFE ( called from CALCFC, 2nd )"
end select

write( 0, '( A )' ) msg
write ( 0, '( A, I )' ) "ierr = ", ierr
stop ' Terminated by error of Hermite function. Check input fuel map.'

end subroutine

```

```

! *****
! DPCHST: SIGN TESTING OF 2 ARGUMENTS
! Returns:
!   -1. if ARG1 and ARG2 are of opposite sign.
!   0. if either argument is zero.
!   +1. if ARG1 and ARG2 are of the same sign.
! *****
real(8) function DPCHST( arg1, arg2 )
  implicit none

  real(8) arg1, arg2

  if ( ( arg1 == 0.0_8 ) .or. ( arg2 == 0.0_8 ) ) then
    DPCHST = 0.0_8
  else
    DPCHST = dsign(1.0_8, arg1) * dsign(1.0_8, arg2)
  end if

end function

```

```

! *****
! DPCHIM: Piecewise Cubic Hermite Interpolation to Monotone data
! *****
subroutine DPCHIM ( n, x, f, d, incfd, ierr )
  implicit none
  interface
    real(8) function DPCHST( arg1, arg2 )
      real(8) arg1, arg2
    end function
  end interface

  integer n, incfd, ierr
  real(8) x(*), f(incfd,*), d(incfd,*)

  integer i, nsec
  real(8) dx1, dx2, del1, del2, dsave, dxsum, dxsumt3, dmax, dmin
  real(8) w1, w2, drat1, drat2, sgn

! ----- check argument -----
  ierr = 0
  if( n < 2 ) then
    ierr = -1
    return
  end if

  if( incfd < 1 ) then
    ierr = -2
    return
  end if

  do i = 2, n
    if( x(i) > x(i-1) ) then
      continue
    else
      ierr = -3
      return
    end if
  end do

! ----- Piecewise Cubic Hermite Interpolation -----
  nsec = n - 1

```

```

dx1 = x(2) - x(1)
del1 = ( f(1,2) - f(1,1) ) / dx1
dsave = del1

if( n == 2 )then
  d(1,1) = del1
  d(1,2) = del1
  return
end if

dx2 = x(3) - x(2)
del2 = ( f(1,3) - f(1,2) ) / dx2

dxsum = dx1 + dx2
w1 = ( dx1 + dxsum ) / dxsum
w2 = -dx1 / dxsum
d(1,1) = w1 * del1 + w2 * del2
if( DPCHST( d(1,1), del1 ) <= 0.0_8 ) then
  d(1,1) = 0.0_8
else if( DPCHST( del1, del2 ) < 0.0_8 ) then
  dmax = 3.0_8 * del1
  if( abs( d(1,1) ) > abs( dmax ) ) then
    d(1,1) = dmax
  end if
end if

do i = 2, nsec
  if( i /= 2 )then
    dx1 = dx2
    dx2 = x(i+1) - x(i)
    dxsum = dx1 + dx2
    del1 = del2
    del2 = ( f(1,i+1) - f(1,i) ) / dx2
  end if

  d( 1, i ) = 0.0_8
  sgn = DPCHST( del1, del2 )

  if( sgn == -1.0_8 ) then
    ierr = ierr + 1
    dsave = del2

  else if( sgn == 0.0_8 ) then
    if( del2 /= 0.0_8 ) then
      if( DPCHST( dsave, del2 ) < 0.0_8 ) then
        ierr = ierr + 1
      end if
      dsave = del2
    end if

  else
    dxsumt3 = dxsum * 3.0_8
    w1 = ( dxsum + dx1 ) / dxsumt3
    w2 = ( dxsum + dx2 ) / dxsumt3
    dmax = max( abs(del1), abs(del2) )
    dmin = min( abs(del1), abs(del2) )
    drat1 = del1 / dmax
    drat2 = del2 / dmax
    d(1,i) = dmin / ( w1 * drat1 + w2 * drat2 )
  end if
end do

w1 = -dx2 / dxsum
w2 = ( dx2 + dxsum ) / dxsum
d( 1, n ) = w1 * del1 + w2 * del2
if( DPCHST( d(1,n), del2 ) <= 0.0_8 ) then
  d(1,n) = 0.0_8
else if( DPCHST( del1, del2 ) < 0.0_8 ) then
  dmax = 3.0_8 * del2
  if( abs( d(1,n) ) > abs( dmax ) ) then
    d( 1, n ) = dmax
  end if
end if

end subroutine DPCHIM

! *****
! DPCHFE: Piecewise Cubic Hermite Function Evaluator
! *****
subroutine DPCHFE( n, x, f, d, incfd, skip, ne, xe, fe, ierr )
  implicit none

  integer n
  integer incfd, ne, ierr
  real(8) x(*), f(incfd,*), d(incfd,*), xe(*), fe(*)
  logical skip

  integer i, ierc, ir, j, j2, jfirst, next(2), nj

  ierr = 0
  ! ----- check argument -----
  if( skip ) then
    if( n < 2 ) then
      ierr = -1
      return
    end if
  end if

```

```

        if( incfd < 1 ) then
            ierr = -2
            return
        end if

        do i = 2, n
            if( x(i) <= x(i-1) ) then
                ierr = -3
                return
            end if
        end do

        if( ne < 1 ) then
            ierr = -4
            return
        end if
    end if

! ----- piecewise cubic hermite function evaluator -----
skip = .true.
jfirst = 1
ir = 2

do while ( jfirst <= ne )
    j2 = ne + 1
    do j = jfirst, ne
        if( xe(j) >= x(ir) ) then
            if( ir == n ) then
                j2 = ne + 1
            else
                j2 = j
            end if
            exit
        end if
    end do
    nj = j2 - jfirst

    if( nj /= 0 ) then
        call DCHF EV ( x(ir-1), x(ir), f(1,ir-1), f(1,ir),      &
                     d(1,ir-1), d(1,ir), nj, xe(jfirst),      &
                     fe(jfirst), next, ierc )

        if( ierc < 0 ) then
            ierr = -5
            return
        end if

        if( next(2) == 0 ) then

            else
                if( ir < n ) then
                    ierr = -5
                    return
                else
                    ierr = ierr + next(2)
                end if
            end if

            if( next(1) == 0 ) then

            else
                if( ir > 2 ) then
                    do i = jfirst, j2-1
                        if( xe(i) < x(ir-1) ) then
                            exit
                        end if
                    end do
                    if( i > j2-1 ) then
                        ierr = -5
                        return
                    end if
                    j2 = i
                    do i = 1, ir-1
                        if( xe(j2) < x(i) ) exit
                    end do
                    ir = max( 1, i-1 )
                else
                    ierr = ierr + next(1)
                end if
            end if
            jfirst = j
        end if

        ir = ir + 1
        if( ir > n ) exit
    end do
end subroutine

! *****
! DCHF EV: Cubic Hermite Function Evaluator
! *****
subroutine DCHF EV (x1, x2, f1, f2, d1, d2, ne, xe, fe, next, ierr)
    implicit none

```

```

integer ne, ierr, next(2)
real(8) x1, x2, f1, f2, d1, d2, xe(*), fe(*)
integer i
real(8) c2, c3, del1, del2, delta, h, x, xmi, xma

ierr = 0
! ----- check arguments -----
if( ne < 1 ) then
    ierr = -1
    return
end if

h = x2 - x1
if( h == 0.0_8 ) then
    ierr = -2
    return
end if

! ----- cubic hermite function evaluator -----
next(1) = 0
next(2) = 0
xmi = min( 0.0_8, h )
xma = max( 0.0_8, h )

delta = ( f2 - f1 ) / h
del1 = ( d1 - delta ) / h
del2 = ( d2 - delta ) / h
c2 = -( del1 + del1 + del2 )
c3 = ( del1 + del2 ) / h

do i = 1, ne
    x = xe(i) - x1
    fe(i) = f1 + x * ( d1 + x * ( c2 + x * c3 ) )
    if( x < xmi ) then
        next(1) = next(1) + 1
    end if
    if( x > xma ) then
        next(2) = next(2) + 1
    end if
end do

end subroutine

```



```
module dataio
```

```
interface
```

```
  subroutine readengfc( uid, engf, mapf, tqf, ftqf, nidle, nrate, nex )
    integer uid
    real(8) nidle, nrate, nex
    character (len=*) engf, mapf, tqf, ftqf
  end subroutine
```

```
  subroutine readgear( uid, tmf, ngr, gr, nsgr, sgr, grs, swat )
    integer uid, ngr, nsgr, grs, swat
    character (len=*) tmf
    real(8), pointer :: gr(:), sgr(:)
  end subroutine
```

```
  subroutine readspefc( uid, specname, rt, fgr, tpname, vname, engf, tmf, tsout )
    integer uid, tsout
    real(8) rt, fgr
    character (len=*) specname, tpname, vname, engf, tmf
  end subroutine
```

```
  subroutine readtqfc( uid, tqf, tqc )
    use hvtransfc
    integer uid
    type (stqcurve) tqc
    character (len=*) tqf
  end subroutine readtqfc
```

```
  subroutine readmapfc( uid, mapf, fcm )
    use hvtransfc
    integer uid
    type (sfcmap) fcm
    character (len=*) mapf
  end subroutine readmapfc
```

```
  subroutine outaveragefc( uid, fname, specf, vname, tpname, engf, tmf, &
    fgr, rt, avvu, eu, avvh, eh, avvt, et, ric, ece )
    integer uid
    character (len=*) fname, specf, vname, tpname, engf, tmf
    real(8) fgr, rt, eu, avvu, eh, avvh, et, avvt, ric, ece
  end subroutine
```

```
  subroutine outtimeseriesfc( uid, fname, vpat, vreal, ne, te, nne, nte, sp, fc, comments )
    use hvtransfc
    integer uid, sp(:)
    character (len=*) fname, comments
    type (scycle) vpat
    real(8) vreal(:), ne(:), te(:), fc(:), nne(:), nte(:)
  end subroutine
```

```
end interface
```

```
end module dataio
```

```
!*****
```

```
! SUBROUTINE READENGFC
```

```
! Read engine input data (map, torque, engine speed)
```

```
!*****
```

```
subroutine readengfc( uid, engf, mapf, tqf, ftqf, nidle, nrate, nex )
```

```
  implicit none
```

```
  integer uid
  character (len=*) tqf, ftqf, mapf, engf
  real(8) nidle, nrate, nex
```

```
  open ( uid, file = engf, status = 'old', access = 'sequential', err=880 )
```

```
  read ( uid, *, err=890 ) mapf           ! fuel map
  read ( uid, *, err=890 ) tqf            ! max torque
  read ( uid, *, err=890 ) ftqf           ! frictional torque
  read ( uid, *, err=890 ) nidle          ! idling speed
  read ( uid, *, err=890 ) nrate         ! rated speed
  read ( uid, *, err=890 ) nex            ! max speed
```

```
  close ( uid )
```

```
  return
```

```
! -----
880 write (0,'(3A)') " Cannot open engine file [ ", trim(engf), " ]"
```

```
  stop
890 write (0,'(3A)') " Failed to read engine file [ ", trim(engf), " ]"
  close ( uid )
  stop
```

```
end subroutine
```

```
!*****
```

```
! SUBROUTINE READGEAR
```

```
! Read transmission input data
```

```
!*****
```

```
subroutine readgear( uid, tmf, ngr, gr, nsgr, sgr, grs, swat )
```

```

implicit none

integer uid, i, ngr, nsgr, grs, swat
character(len=*) tmf                                ! transmission filename
real(8), pointer :: gr(:), sgr(:)

open ( uid, file = tmf, status = 'old', access = 'sequential', err=900 )

read ( uid, *, err=910 ) grs                        ! start gear
read ( uid, *, err=910 ) ngr                        ! n of main gear

allocate ( gr(ngr) )
do i = 1, ngr
    read ( uid, *, err=910 ) gr(i)                  ! gear ratio
end do

read ( uid, *, err=910 ) nsgr                        ! n of sub transmission

allocate( sgr( nsgr ) )
do i = 1, nsgr
    read ( uid, *, err=910 ) sgr(i)                  ! gear ratio
end do

read ( uid, *, err=910 ) swat                        ! torque converter AT(0:MT, AMT, 1:AT)

close( uid )
return

! -----
900 write (0,'(3A)') " Cannot open transmission file [ ", trim(tmf), " ]"
    stop
910 write (0,'(3A)') " Failed to read transmission file [ ", trim(tmf), " ]"
    close( uid )
    stop

end subroutine

!*****
! SUBROUTINE READSPECFC
! Read vehicle specification data
!*****
subroutine readspecfc ( uid, specf, rt, fgr, tpname, vhname, engf, tmf, tsout )
    implicit none

    character (len=*) specf, tpname, vhname, engf, tmf
    integer uid, ios, tsout
    real(8) rt, fgr

    open ( uid, file=trim(specf), status='OLD', access='sequential', iostat=ios )
    if ( ios /= 0 ) then
        write(0,*) ' Cannot open spec-file : ', trim(specf)
        stop
    end if

    read ( uid, * ) vhname                            ! vehicle name
    read ( uid, * ) tpname                            ! type name
    read ( uid, * ) engf                              ! engine filename
    read ( uid, * ) tmf                              ! transmission filename
    read ( uid, * ) fgr                              ! final ratio
    read ( uid, * ) rt                               ! tire radius (m)
    read ( uid, * ) tsout                            ! time series output 0:OFF, 1:ON

    close(uid)

end subroutine

!*****
! SUBROUTINE READTQFC
! Read engine torque input data
!*****
subroutine readtqfc ( uid, tqf, tqc )
    use hvtransfc
    implicit none

    integer uid
    character (len=*) tqf                            ! torque filename
    type (stqcurve) tqc                             ! torque curve

    character (len=1024) tmp
    integer i, j, gap, ios
    real(8) revt, tmaxt

! -----
! count number of torque data
! -----
    tqc%ndata = 0                                     ! initialize
    open ( uid, file = tqf, status = 'old', err=600 )
    read ( uid,'(A)' ) tmp                            ! skip header
    read ( uid,'(A)' ) tmp                            ! skip header

    ios = 0                                           ! status
    do while ( ios == 0 )
        read ( uid, *, iostat = ios, err=650 ) revt, tmaxt    ! test reading of data set
    end do

```

```

        if( ios == 0 ) tqc%ndata = tqc%ndata + 1          ! count n of data
    end do

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

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

    do i = 1, tqc%ndata
        read ( uid, *, err=650 ) tqc%rev(i), tqc%tq(i)    ! read torque data
    end do

    close ( uid )

!-----
! sort in ascending order
!-----
    gap = ( tqc%ndata + 1 ) / 2
    do while ( gap /= 0 )
        i = gap
        do while ( i <= tqc%ndata )
            j = i - gap
            do while ( ( j >= 1 ) .and. ( tqc%rev( j ) > tqc%rev( j + gap ) ) )
                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
                j = j - gap
            end do
            i = i + 1
        end do
        gap = gap / 2
    end do

    return

!-----
! エラー対応
!-----
600 write (0, '(3A)') " Cannot open torque data file [ ", trim(tqf), " ]"
    stop
650 write (0, '(A,i0,3A)') " Failed to read torque data. Data No.= ", &
        tqc%ndata+1, " in [ ", trim( tqf ), " ]"
    close ( uid )
    stop

end subroutine readtqfc

!-----
! *****
! SUBROUTINE READMAPFC
! 燃費マップデータの読み込みサブルーチン
! uid      : 装置番号
! mapf     : fuel map filename
! fcm      : 燃費マップ(構造体)
! *****
subroutine readmapfc( uid, mapf, fcm )
    use hvtransfc
    implicit none

    integer uid
    character (len=*) mapf          ! fuel map filename
    type (sfcmmap) fcm              ! fuel map

    character (len=1024) tmp
    real(8) idles, idlet, r, t, f
    integer gap, i, j, k, ios, head, ct

!-----
! count number of fuel map data
!-----
    fcm%ndata = 0
    open( uid, file = mapf, status = 'old', err=500 )    ! initialize n of data

    read ( uid, *, err=550 ) tmp          ! skip header
    read ( uid, *, err=550 ) tmp          ! skip header
    read ( uid, *, err=550 ) idles, idlet, fcm%fcidle    ! idling fuel consumption(L/h)

    ios = 0
    do while ( ios == 0 )
        read( uid, *, iostat = ios, err=550 ) r, t, f    ! test reading
        if( ios == 0 ) fcm%ndata = fcm%ndata + 1          ! count n of data
    end do

!-----
! read fuel map
!-----
    allocate( fcm%rev( fcm%ndata ) )
    allocate( fcm%tq( fcm%ndata ) )
    allocate( fcm%fc( fcm%ndata ) )

```

```

rewind( uid )
read ( uid, *, err=550 ) tmp
read ( uid, *, err=550 ) tmp
read ( uid, *, err=550 ) tmp
! move to head of file
! skip header
! skip header
! skip idling fuel

do i = 1, fcm%ndata
  read ( uid, *, err=550 ) fcm%rev(i), fcm%tq(i), fcm%fc(i)
end do

close( uid )

! ----- SORT MAP DATA WITH ENGINE SPEED -----
gap = ( fcm%ndata + 1 ) / 2
do while ( gap > 0 )
  i = gap
  do while ( i <= fcm%ndata )
    j = i - gap
    do while ( ( j >= 1 ) .and. ( fcm%rev(j) > fcm%rev(j+gap) ) )
      r = fcm%rev(j)
      fcm%rev(j) = fcm%rev(j+gap)
      fcm%rev(j+gap) = r
      t = fcm%tq(j)
      fcm%tq(j) = fcm%tq(j+gap)
      fcm%tq(j+gap) = t
      f = fcm%fc(j)
      fcm%fc(j) = fcm%fc(j+gap)
      fcm%fc(j+gap) = f
      j = j - gap
    end do
    i = i + 1
  end do
  gap = gap / 2
end do

! ----- SORT WITH ENGINE TORQUE -----
head = 1
r = fcm%rev( head )
ct = 0
do i = 1, fcm%ndata
  ct = ct + 1
  if ( ( r /= fcm%rev(i) ) .or. ( i == fcm%ndata ) ) then
    if ( r /= fcm%rev(i) ) then
      ct = ct - 1
    end if
    gap = ( ct + 1 ) / 2
    do while ( gap > 0 )
      k = gap
      do while ( k <= ct - 1 )
        j = head + k - gap
        do while ( ( j >= head ) .and. ( fcm%tq(j) > fcm%tq(j+gap) ) )
          r = fcm%rev(j)
          fcm%rev(j) = fcm%rev(j+gap)
          fcm%rev(j+gap) = r
          t = fcm%tq(j)
          fcm%tq(j) = fcm%tq(j+gap)
          fcm%tq(j+gap) = t
          f = fcm%fc(j)
          fcm%fc(j) = fcm%fc(j+gap)
          fcm%fc(j+gap) = f
          j = j - gap
        end do
        k = k + 1
      end do
      gap = gap / 2
    end do
    r = fcm%rev(i)
    ct = 1
    head=i
  end if
end do

return

! -----
! error handling
! -----
500 write (0,'(3A)') " Cannot open map data file [ ", trim(mapf), " ]"
stop
550 write (0,'(3A)') " Failed to read fuel map [ ", trim(mapf), " ]"
close ( uid )
stop

end subroutine

!*****
! SUBROUTINE OUTAVERAGEFC
! Output result of average speed & fuel consumption
!*****
subroutine outaveragefc ( uid, outf, specf, vname, tpname, engf, tmf, &
  fgr, rt, avvu, eu, avvh, eh, avvt, et, ric, ece )
  use hvtransfc
  implicit none

  character, parameter :: ht = char(9)

```

```

integer uid
character (len=*) outf, specf, vname, tname, engf, tmf
real(8) fgr, rt, eu, avvu, eh, avvh, avvt, et, ric, ece

open ( uid, file = outf, status = 'unknown' )

write ( uid, '(3A)' ) "VEHICLENAME", ht, trim(vname)
write ( uid, '(3A)' ) "TYPE", ht, trim(tname)
write ( uid, '(3A)' ) "SPEC FILE", ht, trim(specf)
write ( uid, '(3A)' ) "ENGINE FILE", ht, trim(engf)
write ( uid, '(3A)' ) "TRANSMISSION FILE", ht, trim(tmf)
write ( uid, '(2A,F8.5)' ) "FINAL GEAR RATIO", ht, fgr
write ( uid, '(2A,F8.5)' ) "TIRE RADIUS(m)", ht, rt
write ( uid, '*' )
write ( uid, '(4A,F8.4,3A,F5.1)' ) "URBAN ", ht, "FC(km/l)", ht, eu, ht, "Ave. Speed(km/h)", ht, avvu
write ( uid, '(4A,F8.4,3A,F5.1)' ) "HIGHWAY ", ht, "FC(km/l)", ht, eh, ht, "Ave. Speed(km/h)", ht, avvh
write ( uid, '(4A,F8.4,3A,F5.2)' ) "AVERAGE ", ht, "FC(km/l)", ht, ece, ht, "HIGHWAY RATIO", ht, ric
write ( uid, '*' )
write ( uid, '(4A,F8.4,3A,F5.1)' ) "MID-TOWN ", ht, "FC(km/l)", ht, et, ht, "Ave. Speed(km/h)", ht, avvt

close ( uid )

end subroutine outaveragefc

!*****
! SUBROUTINE OUTPUTTIMESERIES
! Output fuel consumption data of time series
!*****
subroutine outtimeseriesfc (uid,outf,vpat,vreal,ne,te,nne,nte,sp,fc,comments)
  use hvtransfc
  implicit none

  character, parameter :: ht = char(9)

  integer uid, i, sp(:)
  real(8) vreal(:), ne(:), te(:), fc(:), nne(:), nte(:)
  character (len=*) outf, comments
  type (scycle) vpat

  open ( uid, file = outf, status = 'unknown', position = 'APPEND' )

  write ( uid, '*' )
  write ( uid, '(A)' ) trim(comments)
  write ( uid, '(17A)' ) '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', ht, 'FC(l/h)'

  do i = 1, vpat%ndat
    write (uid, '(i0,2(a,f0.2),2(a,f0.1),2(a,f0.2),a,i0,a,f0.6)') &
      i, ht, vpat%v(i), ht, vreal(i), ht, ne(i), ht, te(i), &
      ht, nne(i), ht, nte(i), ht, sp(i), ht, fc(i)
  end do

  close ( uid )

end subroutine outtimeseriesfc

```