GEOS-Chem Reference, Vol. 4: Global Terrestrial Mercury Model

GEOS-Chem Support Team

24 July 2012

Contents

1 Routine/Function Prologues

!PROGRAM: GTMM Based on the CASA (Carnegie, Ames, Stanford Approach) terrestrial biogeochemical model designed to simulate the terrestrial carbon cycle using satellite data

INTERFACE:

```
PROGRAM GTMM
```

USES:

```
  USE      defineConstants ! modify defineConstants.f90 to choose
                           ! parameters for your run

  USE      loadCASAinput

  USE      defineArrays

  USE      DORESTART_MOD ! New module to save/read Hg data and
                         ! CASA data for continuation runs and GEOS-CHEM.
                         ! (ccc, 11/3/09)

  USE      INPUT_GTMM_MOD

  IMPLICIT NONE
```

AUTHOR:

```
GTMM (Global Terrestrial Mercury Model) Developed by
Nicole Smith-Downey (nicolevdowney@gmail.com) 2006-2009
See Smith-Downey, Sunderland and Jacob, JGR Biogeosciences, 2009
Based on the CASA (Carnegie, Ames, Stanford Approach) terrestrial
biogeochemical model designed to simulate the terrestrial
carbon cycle using satellite data
Original program written by Potter and Randerson
See: Potter, C.S., J.T. Randerson, C.B. Field, P.A. Matson,
     P.M.Vitousek, H.A. Mooney, and S.A. Klooster, 1993.
     Terrestrial ecosystem production: A process model
     based on satellite and surface data.  Global
     Biogeochemical Cycles (7) 811-841.
```

REVISION HISTORY:

```
  ( 1) Translated into Matlab and accounted for fires by Guido van
       der Werf.
See:   van der Werf, G.R., J.T. Randerson, G.J. Collatz and L.
       Giglio, 2003.  Carbon emissions from fires in tropical
       and subtropical ecosystems.  Global Change Biology 9
       (4) 547-562.
  ( 2) Translated into Fortran90 and added Mercury simulation by
       Nicole Smith Downey - 2006
```

```
( 3) Main program for offline simulations. Added coupling to GEOS-Chem
     (see GTMM_coupled.f90) (ccc, 7/9/10)
( 4) Added capacity to restart runs. (ccc, 7/9/10)
```

1.1 Fortran: Module Interface CasaRegridModule

Module CasaRegridModule contains arrays and variables used to regrid the GEOS-5 data from 1 x 1 Generic to 2 x 2.5, and 4 x 5 geos grids.

INTERFACE:

```
MODULE CasaRegridModule
```

USES:

```
IMPLICIT NONE
PRIVATE
```

PUBLIC MEMBER FUNCTIONS:

```
PUBLIC :: regrid1x1to4x5
PUBLIC :: regrid4x5to1x1
PUBLIC :: regrid1x1to2x25
PUBLIC :: regrid2x25to1x1
PUBLIC :: CasaRegridInit
```

PUBLIC DATA MEMBERS:

```
PUBLIC :: I1x1,      J1x1,      L1x1
PUBLIC :: I2x25,     J2x25,     L2x25
PUBLIC :: I4x5,      J4x5,      L4x5
```

DEFINED PARAMETERS:

```
INTEGER, PARAMETER  :: I1x1     = 360,  J1x1     = 180,  L1x1     = 72
INTEGER, PARAMETER  :: I2x25    = 144,  J2x25    = 91,   L2x25    = 72
INTEGER, PARAMETER  :: I4x5     = 72,   J4x5     = 46,   L4x5     = 72
```

PRIVATE TYPES:

```
! Degrees to Radians
REAL*8,  PARAMETER  :: D2R = 3.1415926589793 23d0 / 180d0

!---------------------
! 1 x 1 grid
!---------------------

! Lon edges
REAL*8 :: xedge_1x1( I1x1 + 1 )

! Lat edges
```

```
  REAL*8 :: yedge_1x1( J1x1 + 1 )

  ! Sine of latitude
  REAL*8 :: sine_1x1( J1x1 + 1 )

  !---------------------
  ! 2 x 2.5 grid
  !---------------------

  ! Longitude edges
  REAL*8 :: xedge_2x25( I2x25 + 1 )

  ! Latitude edges
  REAL*8 :: yedge_2x25( J2x25 + 1 )

  ! Latitude edges
  REAL*8 :: sine_2x25( J2x25 + 1 )

  !---------------------
  ! 4 x 5 grid
  !---------------------

  ! Longitude edges
  REAL*8 :: xedge_4x5( I4x5 + 1 )

  ! Latitude edges
  REAL*8 :: yedge_4x5( J4x5 + 1 )

  ! Latitude edges
  REAL*8 :: sine_4x5( J4x5 + 1 )
```

REMARKS:

```
CasaRegridModule uses the regridding software "MAP_A2A" from S-J Lin.
This is area-preserving mapping.  For example, if you have a quantity
such as kg/m2/s or W/m2, MAP_A2A will multiply by the area on the
input grid, then regrid, and divide by the area on the output grid,
such that the total quantity is preserved.
```

REVISION HISTORY:

```
14 Jan 2008 - R. Yantosca - Initial version
(1 ) Modify regriddGeos5To* routines so that if all values are zero,
      then we just fill the output data array with zeros and return.
      This ought to speed up program execution. (bmy, 11/14/06)
```

1.1.1 regrid4x5to1x1

Subroutine regrid4x5to1x1 is a wrapper for MAP_A2A, which is called to regrid from the GEOS-5 4x5 grid to the GENERIC 1x1 grid.

INTERFACE:

```
SUBROUTINE regrid4x5to1x1( iv, q1, q2 )
```

INPUT PARAMETERS:

```
INTEGER, INTENT(IN)  :: iv
REAL*8,  INTENT(IN)  :: q1( I4x5, J4x5 )
```

OUTPUT PARAMETERS:

```
REAL*8,  INTENT(OUT) :: q2( I1x1, J1x1 )
```

REVISION HISTORY:

```
08 Nov 2006 - R. Yantosca - Initial version
```

1.1.2 regrid1x1to4x5

Subroutine regrid1x1to4x5 is a wrapper for MAP_A2A, which is called to regrid from the GENERIC 1x1 grid to the GEOS-5 4x5 grid.

INTERFACE:

```
SUBROUTINE regrid1x1to4x5( iv, q1, q2 )
```

INPUT PARAMETERS:

```
INTEGER, INTENT(IN)  :: iv
REAL*8,  INTENT(IN)  :: q1( I1x1, J1x1 )
```

OUTPUT PARAMETERS:

```
REAL*8,  INTENT(OUT) :: q2( I4x5, J4x5 )
```

REVISION HISTORY:

```
08 Nov 2006 - R. Yantosca - Initial version
```

1.1.3 regrid2x25to1x1

Subroutine regrid2x25to1x1 is a wrapper for MAP_A2A, which is called to regrid from the GENERIC 1x1 grid to the GEOS 2 x 2.5 grid. (bmy, 11/8/06)

!INTERFACE
SUBROUTINE regrid2x25to1x1(iv, q1, q2) **INPUT PARAMETERS:**

```
INTEGER, INTENT(IN)  :: iv
REAL*8,  INTENT(IN)  :: q1( I2x25, J2x25 )
```

OUTPUT PARAMETERS:

```
REAL*8,  INTENT(OUT) :: q2( I1x1,  J1x1  )
```

REVISION HISTORY:

```
08 Nov 2006 - R. Yantosca - Initial version
```

1.1.4 regrid1x1to2x25

Subroutine regridGeos5to2x25 is a wrapper for MAP_A2A, which regrids from the GEOS-5 1x1 grid to the GEOS 2 x 2.5 grid.

INTERFACE:

```
SUBROUTINE regrid1x1to2x25( iv, q1, q2 )
```

INPUT PARAMETERS:

```
INTEGER, INTENT(IN)  :: iv
REAL*8,  INTENT(IN)  :: q1( I1x1,  J1x1  )
```

OUTPUT PARAMETERS:

```
REAL*8,  INTENT(OUT) :: q2( I2x25, J2x25 )
```

REVISION HISTORY:

```
08 Nov 2006 - R. Yantosca - Initial version
```

1.1.5 map_a2a

Subroutine MAP_A2A is a orizontal arbitrary grid to arbitrary grid conservative high-order mapping regridding routine by S-J Lin.

INTERFACE:

```
SUBROUTINE map_a2a( im, jm, lon1, sin1, q1, &
                    in, jn, lon2, sin2, q2, ig, iv)
```

INPUT PARAMETERS:

```
INTEGER, INTENT(IN)  :: im, jm, in, jn, ig, iv
REAL*8,  INTENT(IN)  :: lon1(im+1), lon2(in+1)
REAL*8,  INTENT(IN)  :: sin1(jm+1), sin2(jn+1)
REAL*8,  INTENT(IN)  :: q1(im,jm)
```

OUTPUT PARAMETERS:

```
 REAL*8,  INTENT(OUT) :: q2(in,jn)
```

REVISION HISTORY:

```
(1) Original subroutine by S-J Lin.  Converted to F90 freeform format
    and inserted into "Geos3RegridModule" by Bob Yantosca (9/21/00)
(2) Added F90 type declarations to be consistent w/ TypeModule.f90.
    Also updated comments. (bmy, 9/21/00)
21 Sep 2000 - R. Yantosca - Initial version
```

1.1.6 ymap

Routine to perform area preserving mapping in N-S from an arbitrary resolution to another.

INTERFACE:

```
SUBROUTINE ymap(im, jm, sin1, q1, jn, sin2, q2, ig, iv)
```

INPUT PARAMETERS:

```
 INTEGER, INTENT(IN)  :: im            ! original E-W dimension
 INTEGER, INTENT(IN)  :: jm            ! original N-S dimension
 INTEGER, INTENT(IN)  :: jn            ! Target N-S dimension
 INTEGER, INTENT(IN)  :: ig            ! ig=0: scalars from SP to NP
                                       ! D-grid v-wind is also ig 0
                                       ! ig=1: D-grid u-wind
 INTEGER, INTENT(IN)  :: iv            ! iv=0 scalar; iv=1: vector
 REAL*8,  INTENT(IN)  :: sin1(jm+1-ig) ! original southern edge of
                                       !  the cell sin(lat1)
 REAL*8,  INTENT(IN)  :: q1(im,jm)     ! original data at center of
                                       !  the cell
 REAL*8,  INTENT(IN)  :: sin2(jn+1-ig) ! Target cell's southern edge
                                       !  sin(lat2)
```

OUTPUT PARAMETERS:

```
 REAL*8,  INTENT(OUT) :: q2(im,jn)     ! Mapped data at the
                                       !  target resolution
```

REMARKS:

```
sin1 (1) = -1 must be south pole; sin1(jm+1)=1 must be N pole.

sin1(1) < sin1(2) < sin1(3) < ... < sin1(jm) < sin1(jm+1)
sin2(1) < sin2(2) < sin2(3) < ... < sin2(jn) < sin2(jn+1)
```

AUTHOR:

```
S.-J. Lin
First version: piece-wise constant mapping
Apr 1, 2000
Last modified:
```

REVISION HISTORY:

```
21 Sep 2000 - R. Yantosca - Initial version
```

1.1.7 ppm_lat

Subroutine PPM_LAT is called by YMAP. Written by S-J Lin, and converted to F90 freeform format by Bob Yantosca.

INTERFACE:

```
SUBROUTINE ppm_lat(im, jm, ig, q, al, ar, a6, jord, iv)
INPUT PARAMETERS:
  INTEGER           :: im, jm          !  Dimensions
  INTEGER           :: ig
  INTEGER           :: jord
  INTEGER           :: iv              ! iv=0 scalar
                                       ! iv=1 vector
```

INPUT/OUTPUT PARAMETERS:

```
  REAL*8            :: q(im,jm-ig)
```

OUTPUT PARAMETERS:

```
  REAL*8            :: al(im,jm-ig)
  REAL*8            :: ar(im,jm-ig)
  REAL*8            :: a6(im,jm-ig)
```

REVISION HISTORY:

```
21 Sep 2000 - R. Yantosca - Initial version
```

1.1.8 xmap

Routine to perform area preserving mapping in E-W from an arbitrary resolution to another. Periodic domain will be assumed, i.e., the eastern wall bounding cell im is lon1(im+1) = lon1(1); Note the equal sign is true geographysically.

INTERFACE:

```
SUBROUTINE xmap(im, jm, lon1, q1, in, lon2, q2)
```

INPUT PARAMETERS:

```
  INTEGER, INTENT(IN)  :: im          ! original E-W dimension
  INTEGER, INTENT(IN)  :: in          ! Target E-W dimension
  INTEGER, INTENT(IN)  :: jm          ! original N-S dimension
  REAL*8,  INTENT(IN)  :: lon1(im+1)  ! original western edge of
                                      !  the cell
  REAL*8,  INTENT(IN)  :: q1(im,jm)   ! original data at center of
                                      !  the cell
  REAL*8,  INTENT(IN)  :: lon2(in+1)  ! Target cell's western edge
```

OUTPUT PARAMETERS:

```
REAL*8,  INTENT(OUT) :: q2(in,jm)    ! Mapped data at the
                                     !  target resolution
```

REVISION HISTORY:

21 Sep 2000 - R. Yantosca - Initial version

1.1.9 ppm_cycle

PPM_CYCLE is called by XMAP

INTERFACE:

```
subroutine ppm_cycle(im, q, al, ar, a6, p, iord)
```

INPUT PARAMETERS:

```
INTEGER, INTENT(IN)  :: im, iord
REAL*8,  INTENT(IN)  :: q(1)
```

OUTPUT PARAMETERS:

```
REAL*8,  INTENT(OUT) :: al(1), ar(1), a6(1), p(0:im+1)
```

REVISION HISTORY:

21 Sep 2000 - R. Yantosca - Initial version

1.1.10 lmppm

Subroutine LMPPM is called by PPM_CYCLE.

INTERFACE:

```
SUBROUTINE lmppm(dm, a6, ar, al, p, im, lmt)
```

INPUT PARAMETERS:

```
INTEGER           :: im, lmt
REAL*8            :: p(im),dm(im)
```

INPUT/OUTPUT PARAMETERS:

```
REAL*8            :: a6(im),ar(im),al(im)
```

REMARKS:

```
LMT = 0: full monotonicity
LMT = 1: semi-monotonic constraint (no undershoot)
LMT = 2: positive-definite constraint
```

REVISION HISTORY:

21 Sep 2000 - R. Yantosca - Initial version

1.1.11 huynh

Subroutine HUYNH enforces Huynh's 2nd constraint in 1D periodic domain

INTERFACE:

```
SUBROUTINE huynh(im, ar, al, p, d2, d1)
```

INPUT PARAMETERS:

```
INTEGER :: im
REAL*8  :: p(im)
```

INPUT/OUTPUT PARAMETERS:

```
REAL*8  :: ar(im), al(im), d2(im), d1(im)
```

REVISION HISTORY:

```
21 Sep 2000 - R. Yantosca - Initial version
```

1.1.12 CasaRegridInit

Subroutine CasaRegridInit initializes the longitude and latitude edge arrays for 0.5 x 0.666, 1 x 1.25, 2 x 2.5, and 4 x 5 grids.

INTERFACE:

```
SUBROUTINE CasaRegridInit
```

REMARKS:

```
Computation is done in REAL*8 and then casted to REAL*4 in order
to get correct values for the high-resolution grids.
```

REVISION HISTORY:

```
09 Nov 2006- R. Yantosca - Initial version
```

1.2 Fortran: Module Interface defineArrays

Module defineArrays defines all allocatable arrays for GTMM

INTERFACE:

```
MODULE defineArrays
```

USES:

```
USE defineConstants

IMPLICIT NONE
```

PUBLIC DATA MEMBERS:

```
! in getSoilParams
CHARACTER(5), dimension(20000) :: years
REAL*8, ALLOCATABLE :: clay(:,:)
REAL*8, ALLOCATABLE :: silt(:,:)
REAL*8, ALLOCATABLE :: sand(:,:)
REAL*8, ALLOCATABLE :: litcn(:,:)
REAL*8, ALLOCATABLE :: lignin(:,:)
REAL*8, ALLOCATABLE :: lrage(:,:)
REAL*8, ALLOCATABLE :: woodage(:,:)

!in getSoilMoistureParams
REAL*8, ALLOCATABLE :: SMparams(:,:)
REAL*8, ALLOCATABLE :: last_soilm(:,:)

! in doPET
REAL*8, ALLOCATABLE :: PET(:,:)
REAL*8, ALLOCATABLE :: AHI(:,:)
REAL*8              :: coef(4,12)

! in doSoilMoisture
REAL*8, ALLOCATABLE :: last_pack(:,:)
REAL*8, ALLOCATABLE :: spack(:,:)
REAL*8, ALLOCATABLE :: bgmoist(:,:)
REAL*8, ALLOCATABLE :: NPPmoist(:,:)
REAL*8, ALLOCATABLE :: EET(:,:)
REAL*8, ALLOCATABLE :: NPPmoist_temp(:,:)
REAL*8, ALLOCATABLE :: bgmoist_temp(:,:)
REAL*8, ALLOCATABLE :: bgmoistpret(:,:)
REAL*8, ALLOCATABLE :: NPPmoistpret(:,:)
REAL*8, ALLOCATABLE :: soilm(:,:)
REAL*8, ALLOCATABLE :: rdr(:,:)
REAL*8, ALLOCATABLE :: current_ppt(:,:)
REAL*8, ALLOCATABLE :: eeta(:,:)
REAL*8, ALLOCATABLE :: eetb(:,:)
REAL*8, ALLOCATABLE :: this_soilm(:,:)
REAL*8, ALLOCATABLE :: bgratio(:,:)

! in doFPARandLAI
!constraints used to determine the simple ratio for each
!grid cell from code written by Sietse Los in Jan 93
REAL*8              :: SRMAX1  = 4.2448d0
REAL*8              :: SRMAX2  = 4.5970d0
REAL*8              :: SRMAX3  = 4.5970d0
REAL*8              :: SRMAX4  = 4.5970d0
REAL*8              :: SRMAX5  = 4.5970d0
REAL*8              :: SRMAX6  = 4.2448d0
```

```
REAL*8              :: SRMAX7  = 3.8387d0
REAL*8              :: SRMAX8  = 4.5970d0
REAL*8              :: SRMAX9  = 3.8387d0
REAL*8              :: SRMAX10 = 3.8387d0
REAL*8              :: SRMAX11 = 3.8387d0
REAL*8              :: SRMAX12 = 3.8387d0
REAL*8              :: SRMIN   = 1.0d0    ! for unvegetated land

!maximum and minimum possible FPAR
REAL*8              :: FPARMAX=0.9500d0
REAL*8              :: FPARMIN=0.0000d0 ! changed from original
! code, was 0.0010

!maximum possible LAI for each biome
REAL*8              :: LAIMAX1  = 7.0000d0
REAL*8              :: LAIMAX2  = 7.0000d0
REAL*8              :: LAIMAX3  = 7.5000d0
REAL*8              :: LAIMAX4  = 8.0000d0
REAL*8              :: LAIMAX5  = 8.0000d0
REAL*8              :: LAIMAX6  = 5.0000d0
REAL*8              :: LAIMAX7  = 5.0000d0
REAL*8              :: LAIMAX8  = 5.0000d0
REAL*8              :: LAIMAX9  = 5.0000d0
REAL*8              :: LAIMAX10 = 5.0000d0
REAL*8              :: LAIMAX11 = 5.0000d0
REAL*8              :: LAIMAX12 = 6.0000d0

REAL*8              :: Ae

!arrays for later use
REAL*8, ALLOCATABLE :: srmax(:,:)
REAL*8, ALLOCATABLE :: LAImax(:,:)
REAL*8, ALLOCATABLE :: LAI_temp(:,:)
REAL*8, ALLOCATABLE :: FPAR(:,:)
REAL*8, ALLOCATABLE :: LAI(:,:)
REAL*8, ALLOCATABLE :: sr(:,:)
!in doOptimumTemperature
REAL*8, ALLOCATABLE :: topt(:,:)
REAL*8, ALLOCATABLE :: maxlai(:,:)
REAL*8, ALLOCATABLE :: lais(:,:)

!in doLeafRootShedding
REAL*8, ALLOCATABLE :: LTCON(:,:)
REAL*8, ALLOCATABLE :: LTVAR(:,:)

!in doTreeCarbon and doHerbCarbon
INTEGER             :: n_wood_pools=13
INTEGER             :: n_herb_pools=10
```

```
!woody pools
REAL*8, ALLOCATABLE :: abovewoodpool(:,:)
REAL*8, ALLOCATABLE :: belowwoodpool(:,:)
REAL*8, ALLOCATABLE :: leafpool(:,:)
REAL*8, ALLOCATABLE :: frootpool(:,:)
REAL*8, ALLOCATABLE :: cwdpool(:,:)
REAL*8, ALLOCATABLE :: surfstrpool(:,:)
REAL*8, ALLOCATABLE :: surfmetpool(:,:)
REAL*8, ALLOCATABLE :: surfmicpool(:,:)
REAL*8, ALLOCATABLE :: soilstrpool(:,:)
REAL*8, ALLOCATABLE :: soilmetpool(:,:)
REAL*8, ALLOCATABLE :: soilmicpool(:,:)
REAL*8, ALLOCATABLE :: slowpool(:,:)
REAL*8, ALLOCATABLE :: armoredpool(:,:)

!woody pools Hg
REAL*8, ALLOCATABLE :: abovewoodpool_Hg(:,:)
REAL*8, ALLOCATABLE :: belowwoodpool_Hg(:,:)
REAL*8, ALLOCATABLE :: leafpool_Hg(:,:)
REAL*8, ALLOCATABLE :: frootpool_Hg(:,:)
REAL*8, ALLOCATABLE :: cwdpool_Hg(:,:)
REAL*8, ALLOCATABLE :: surfstrpool_Hg(:,:)
REAL*8, ALLOCATABLE :: surfmetpool_Hg(:,:)
REAL*8, ALLOCATABLE :: surfmicpool_Hg(:,:)
REAL*8, ALLOCATABLE :: soilstrpool_Hg(:,:)
REAL*8, ALLOCATABLE :: soilmetpool_Hg(:,:)
REAL*8, ALLOCATABLE :: soilmicpool_Hg(:,:)
REAL*8, ALLOCATABLE :: slowpool_Hg(:,:)
REAL*8, ALLOCATABLE :: armoredpool_Hg(:,:)
REAL*8, ALLOCATABLE :: total_tree_hg(:,:)

!herb pools
REAL*8, ALLOCATABLE :: hleafpool(:,:)
REAL*8, ALLOCATABLE :: hfrootpool(:,:)
REAL*8, ALLOCATABLE :: hsurfstrpool(:,:)
REAL*8, ALLOCATABLE :: hsurfmetpool(:,:)
REAL*8, ALLOCATABLE :: hsurfmicpool(:,:)
REAL*8, ALLOCATABLE :: hsoilstrpool(:,:)
REAL*8, ALLOCATABLE :: hsoilmetpool(:,:)
REAL*8, ALLOCATABLE :: hsoilmicpool(:,:)
REAL*8, ALLOCATABLE :: hslowpool(:,:)
REAL*8, ALLOCATABLE :: harmoredpool(:,:)

!herb pools Hg
REAL*8, ALLOCATABLE :: hleafpool_Hg(:,:)
REAL*8, ALLOCATABLE :: hfrootpool_Hg(:,:)
REAL*8, ALLOCATABLE :: hsurfstrpool_Hg(:,:)
```

```
REAL*8, ALLOCATABLE :: hsurfmetpool_Hg(:,:)
REAL*8, ALLOCATABLE :: hsurfmicpool_Hg(:,:)
REAL*8, ALLOCATABLE :: hsoilstrpool_Hg(:,:)
REAL*8, ALLOCATABLE :: hsoilmetpool_Hg(:,:)
REAL*8, ALLOCATABLE :: hsoilmicpool_Hg(:,:)
REAL*8, ALLOCATABLE :: hslowpool_Hg(:,:)
REAL*8, ALLOCATABLE :: harmoredpool_Hg(:,:)
REAL*8, ALLOCATABLE :: total_herb_hg(:,:)

!max hg woody pool
REAL*8, ALLOCATABLE :: max_hg_leaf(:,:)
REAL*8, ALLOCATABLE :: max_hg_surfstr(:,:)
REAL*8, ALLOCATABLE :: max_hg_surfmet(:,:)
REAL*8, ALLOCATABLE :: max_hg_surfmic(:,:)
REAL*8, ALLOCATABLE :: max_hg_soilstr(:,:)
REAL*8, ALLOCATABLE :: max_hg_soilmet(:,:)
REAL*8, ALLOCATABLE :: max_hg_soilmic(:,:)
REAL*8, ALLOCATABLE :: max_hg_slow(:,:)
REAL*8, ALLOCATABLE :: max_hg_armored(:,:)

!max hg herb pools
REAL*8, ALLOCATABLE :: max_hg_hleaf(:,:)
REAL*8, ALLOCATABLE :: max_hg_hsurfstr(:,:)
REAL*8, ALLOCATABLE :: max_hg_hsurfmet(:,:)
REAL*8, ALLOCATABLE :: max_hg_hsurfmic(:,:)
REAL*8, ALLOCATABLE :: max_hg_hsoilstr(:,:)
REAL*8, ALLOCATABLE :: max_hg_hsoilmet(:,:)
REAL*8, ALLOCATABLE :: max_hg_hsoilmic(:,:)
REAL*8, ALLOCATABLE :: max_hg_hslow(:,:)
REAL*8, ALLOCATABLE :: max_hg_harmored(:,:)

!woody pools
REAL*8, ALLOCATABLE :: abovewoodpools(:,:)
REAL*8, ALLOCATABLE :: belowwoodpools(:,:)
REAL*8, ALLOCATABLE :: leafpools(:,:)
REAL*8, ALLOCATABLE :: frootpools(:,:)
REAL*8, ALLOCATABLE :: cwdpools(:,:)
REAL*8, ALLOCATABLE :: surfstrpools(:,:)
REAL*8, ALLOCATABLE :: surfmetpools(:,:)
REAL*8, ALLOCATABLE :: surfmicpools(:,:)
REAL*8, ALLOCATABLE :: soilstrpools(:,:)
REAL*8, ALLOCATABLE :: soilmetpools(:,:)
REAL*8, ALLOCATABLE :: soilmicpools(:,:)
REAL*8, ALLOCATABLE :: slowpools(:,:)
REAL*8, ALLOCATABLE :: armoredpools(:,:)

!herb poolss
REAL*8, ALLOCATABLE :: hleafpools(:,:)
```

```
REAL*8, ALLOCATABLE :: hfrootpools(:,:)
REAL*8, ALLOCATABLE :: hsurfstrpools(:,:)
REAL*8, ALLOCATABLE :: hsurfmetpools(:,:)
REAL*8, ALLOCATABLE :: hsurfmicpools(:,:)
REAL*8, ALLOCATABLE :: hsoilstrpools(:,:)
REAL*8, ALLOCATABLE :: hsoilmetpools(:,:)
REAL*8, ALLOCATABLE :: hsoilmicpools(:,:)
REAL*8, ALLOCATABLE :: hslowpools(:,:)
REAL*8, ALLOCATABLE :: harmoredpools(:,:)

REAL*8, ALLOCATABLE :: fuelshortage(:,:)

REAL*8, ALLOCATABLE :: LtN(:,:)
REAL*8, ALLOCATABLE :: annK_leaf(:,:)
REAL*8, ALLOCATABLE :: annK_leaf_hg(:,:)
REAL*8, ALLOCATABLE :: annK_wood(:,:)
REAL*8, ALLOCATABLE :: annK_froot(:,:)
REAL*8, ALLOCATABLE :: K_wood(:,:)
REAL*8, ALLOCATABLE :: K_froot(:,:)
REAL*8, ALLOCATABLE :: K_leaf(:,:)
REAL*8, ALLOCATABLE :: K_hleaf(:,:)
REAL*8, ALLOCATABLE :: K_hfroot(:,:)
REAL*8, ALLOCATABLE :: K_surfmet(:,:)
REAL*8, ALLOCATABLE :: K_surfstr(:,:)
REAL*8, ALLOCATABLE :: K_soilmet(:,:)
REAL*8, ALLOCATABLE :: K_soilstr(:,:)
REAL*8, ALLOCATABLE :: K_cwd(:,:)
REAL*8, ALLOCATABLE :: K_surfmic(:,:)
REAL*8, ALLOCATABLE :: K_soilmic(:,:)
REAL*8, ALLOCATABLE :: K_slow(:,:)
REAL*8, ALLOCATABLE :: K_armored(:,:)
REAL*8, ALLOCATABLE :: slitscalar(:,:)
REAL*8, ALLOCATABLE :: shlitscalar(:,:)
REAL*8, ALLOCATABLE :: srootlitscalar(:,:)
REAL*8, ALLOCATABLE :: sabiotic(:,:)
REAL*8, ALLOCATABLE :: sabiotsmc(:,:)
REAL*8, ALLOCATABLE :: sabiotlign(:,:)
REAL*8, ALLOCATABLE :: metabfract(:,:)
REAL*8, ALLOCATABLE :: structuralLignin(:,:)
REAL*8, ALLOCATABLE :: lignineffect(:,:)
REAL*8, ALLOCATABLE :: soilmicDecayFactor(:,:)
REAL*8, ALLOCATABLE :: slowDecayFactor(:,:)
REAL*8, ALLOCATABLE :: armoredDecayFactor(:,:)
REAL*8, ALLOCATABLE :: fid(:,:)
REAL*8, ALLOCATABLE :: decayClayFactor(:,:)
REAL*8, ALLOCATABLE :: eff_soilmic2slow(:,:)

! rate constants for pools
```

```
REAL*8 :: annK_hleaf=2.000d0   !turnover time for grass leaves
REAL*8 :: annK_hfroot=2.000d0  !turnover time for grass roots
REAL*8 :: annK_surfmet=14.800d0
REAL*8 :: annK_surfstr=3.900d0
REAL*8 :: annK_soilmet=18.500d0
REAL*8 :: annK_soilstr=4.800d0
REAL*8 :: annK_cwd=0.2500d0
REAL*8 :: annK_surfmic=6.000d0
REAL*8 :: annK_soilmic=7.300d0
REAL*8 :: annK_slow=0.200d0
REAL*8 :: annK_armored=0.0045d0
REAL*8 :: annK_hleaf_hg=2.000d0   !turnover time for grass leaves
REAL*8 :: annK_surfmet_hg=14.800d0
REAL*8 :: annK_surfstr_hg=3.900d0
REAL*8 :: annK_soilmet_hg=18.500d0
REAL*8 :: annK_soilstr_hg=4.800d0
REAL*8 :: annK_surfmic_hg=6.000d0
REAL*8 :: annK_soilmic_hg=7.300d0
REAL*8 :: annK_slow_hg=0.200d0
REAL*8 :: annK_armored_hg=0.0045d0


REAL*8 :: eff_surfstr2slow=0.700d0
REAL*8 :: eff_surfstr2surfmic=0.400d0
REAL*8 :: eff_soilstr2slow=0.700d0
REAL*8 :: eff_soilstr2soilmic=0.4500d0
REAL*8 :: eff_cwd2slow=0.700d0
REAL*8 :: eff_surfmic2slow=0.400d0
REAL*8 :: eff_surfmet2surfmic=0.400d0
REAL*8 :: eff_soilmet2soilmic=0.4500d0
REAL*8 :: eff_slow2soilmic=0.4500d0
REAL*8 :: eff_armored2soilmic=0.4500d0
REAL*8 :: woodligninfract=0.400d0 !estimate that lignin content of
!wood is 40%

REAL*8, ALLOCATABLE :: latitude(:,:)
REAL*8, ALLOCATABLE :: latitude1(:,:)

REAL*8, ALLOCATABLE :: fuelwooddemand(:,:)

!in doNPP
REAL*8, ALLOCATABLE :: T1(:,:)
REAL*8, ALLOCATABLE :: T2low(:,:)
REAL*8, ALLOCATABLE :: T2high(:,:)
REAL*8, ALLOCATABLE :: NPPtemp(:,:)
REAL*8, ALLOCATABLE :: IPAR(:,:)
REAL*8, ALLOCATABLE :: NPP(:,:)
```

```
REAL*8, ALLOCATABLE :: epsilona(:,:)
REAL*8, ALLOCATABLE :: bgtemp(:,:)
REAL*8, ALLOCATABLE :: abiotic(:,:)

!in doHerbivory
REAL*8, ALLOCATABLE :: grass_herbivory(:,:)
REAL*8, ALLOCATABLE :: trees_herbivory(:,:)
REAL*8, ALLOCATABLE :: herb_seasonality(:,:)

!in doLeafRootShedding
REAL*8, ALLOCATABLE :: MINLAI(:,:)
REAL*8, ALLOCATABLE :: SUMLAI(:,:)
REAL*8, ALLOCATABLE :: AVELAI(:,:)
REAL*8, ALLOCATABLE :: LTVARSUM(:,:)
REAL*8, ALLOCATABLE :: SUMLAInew(:,:)
REAL*8, ALLOCATABLE :: litterscalar(:,:)
REAL*8, ALLOCATABLE :: hlitterscalar(:,:)
REAL*8, ALLOCATABLE :: rootlitscalar(:,:)

!in getFireParams
REAL*8               :: CC(4,2)
REAL*8, ALLOCATABLE :: ccWood(:,:)
REAL*8, ALLOCATABLE :: ccLeaf(:,:)
REAL*8, ALLOCATABLE :: PET_current(:,:)
REAL*8, ALLOCATABLE :: CCratio_current(:,:)
REAL*8, ALLOCATABLE :: ccFineLitter(:,:)
REAL*8, ALLOCATABLE :: ccCWD(:,:)
REAL*8, ALLOCATABLE :: CCratio_previous(:,:)
REAL*8, ALLOCATABLE :: mortality_tree(:,:)
REAL*8, ALLOCATABLE :: mortality_hfroot(:,:)

!in doTreeCarbon and doHerbCarbon
REAL*8, ALLOCATABLE :: leafinput(:,:)
REAL*8, ALLOCATABLE :: woodinput(:,:)
REAL*8, ALLOCATABLE :: frootinput(:,:)
REAL*8, ALLOCATABLE :: herbivory(:,:)
REAL*8, ALLOCATABLE :: carbonout_leaf(:,:)
REAL*8, ALLOCATABLE :: carbonout_abovewood(:,:)
REAL*8, ALLOCATABLE :: carbonout_belowwood(:,:)
REAL*8, ALLOCATABLE :: carbonout_froot(:,:)
REAL*8, ALLOCATABLE :: carbonout_cwd(:,:)
REAL*8, ALLOCATABLE :: carbonout_surfmet(:,:)
REAL*8, ALLOCATABLE :: carbonout_surfstr(:,:)
REAL*8, ALLOCATABLE :: carbonout_soilmet(:,:)
REAL*8, ALLOCATABLE :: carbonout_soilstr(:,:)
REAL*8, ALLOCATABLE :: carbonout_surfmic(:,:)
REAL*8, ALLOCATABLE :: carbonout_soilmic(:,:)
REAL*8, ALLOCATABLE :: carbonout_slow(:,:)
```

```
REAL*8, ALLOCATABLE :: carbonout_armored(:,:)

REAL*8, ALLOCATABLE :: hgout_surfmet(:,:)
REAL*8, ALLOCATABLE :: hgout_surfstr(:,:)
REAL*8, ALLOCATABLE :: hgout_leaf(:,:)
REAL*8, ALLOCATABLE :: hgout_soilstr(:,:)
REAL*8, ALLOCATABLE :: hgout_surfmic(:,:)
REAL*8, ALLOCATABLE :: hgout_soilmic(:,:)
REAL*8, ALLOCATABLE :: hgout_slow(:,:)
REAL*8, ALLOCATABLE :: hgout_armored(:,:)
REAL*8, ALLOCATABLE :: hgout_soilmet(:,:)

REAL*8, ALLOCATABLE :: Hg_pool_fluxes1(:,:)
REAL*8, ALLOCATABLE :: Hg_pool_fluxes2(:,:)
REAL*8, ALLOCATABLE :: Hg_pool_fluxes3(:,:)
REAL*8, ALLOCATABLE :: Hg_pool_fluxes4(:,:)
REAL*8, ALLOCATABLE :: Hg_pool_fluxes5(:,:)
REAL*8, ALLOCATABLE :: Hg_pool_fluxes6(:,:)

REAL*8, ALLOCATABLE :: Hg_hpool_fluxes1(:,:)
REAL*8, ALLOCATABLE :: Hg_hpool_fluxes2(:,:)
REAL*8, ALLOCATABLE :: Hg_hpool_fluxes3(:,:)
REAL*8, ALLOCATABLE :: Hg_hpool_fluxes4(:,:)
REAL*8, ALLOCATABLE :: Hg_hpool_fluxes5(:,:)
REAL*8, ALLOCATABLE :: Hg_hpool_fluxes6(:,:)

REAL*8, ALLOCATABLE :: resppool_surfstr_hg(:,:)
REAL*8, ALLOCATABLE :: resppool_surfmet_hg(:,:)
REAL*8, ALLOCATABLE :: resppool_surfmic_hg(:,:)
REAL*8, ALLOCATABLE :: resppool_soilstr_hg(:,:)
REAL*8, ALLOCATABLE :: resppool_soilmic_hg(:,:)
REAL*8, ALLOCATABLE :: resppool_soilmet_hg(:,:)
REAL*8, ALLOCATABLE :: resppool_slow_hg(:,:)
REAL*8, ALLOCATABLE :: resppool_armored_hg(:,:)

REAL*8, ALLOCATABLE :: resp_surfstr_hg(:,:)
REAL*8, ALLOCATABLE :: resp_surfmet_hg(:,:)
REAL*8, ALLOCATABLE :: resp_surfmic_hg(:,:)
REAL*8, ALLOCATABLE :: resp_soilstr_hg(:,:)
REAL*8, ALLOCATABLE :: resp_soilmic_hg(:,:)
REAL*8, ALLOCATABLE :: resp_soilmet_hg(:,:)
REAL*8, ALLOCATABLE :: resp_slow_hg(:,:)
REAL*8, ALLOCATABLE :: resp_armored_hg(:,:)

REAL*8, ALLOCATABLE :: combusted_leaf_hg(:,:)
REAL*8, ALLOCATABLE :: combusted_surfstr_hg(:,:)
REAL*8, ALLOCATABLE :: combusted_surfmet_hg(:,:)
REAL*8, ALLOCATABLE :: combusted_surfmic_hg(:,:)
```

```
REAL*8, ALLOCATABLE :: nonCombusted_leaf_hg(:,:)
REAL*8, ALLOCATABLE :: combusted_soilstr_hg(:,:)
REAL*8, ALLOCATABLE :: combusted_soilmet_hg(:,:)
REAL*8, ALLOCATABLE :: combusted_soilmic_hg(:,:)
REAL*8, ALLOCATABLE :: combusted_slow_hg(:,:)
REAL*8, ALLOCATABLE :: combusted_armored_hg(:,:)
REAL*8, ALLOCATABLE :: fuelwoodout_hg(:,:)
REAL*8, ALLOCATABLE :: wresp_hg(:,:)
REAL*8, ALLOCATABLE :: wcomb_hg(:,:)
REAL*8, ALLOCATABLE :: wherb_hg(:,:)
REAL*8, ALLOCATABLE :: wbiof_hg(:,:)
REAL*8, ALLOCATABLE :: hresp_hg(:,:)
REAL*8, ALLOCATABLE :: hcomb_hg(:,:)
REAL*8, ALLOCATABLE :: hherb_hg(:,:)
REAL*8, ALLOCATABLE :: veg_burn(:,:)
REAL*8, ALLOCATABLE :: f_carbonout_surfmet(:,:)
REAL*8, ALLOCATABLE :: f_carbonout_surfstr(:,:)
REAL*8, ALLOCATABLE :: f_carbonout_leaf(:,:)
REAL*8, ALLOCATABLE :: f_carbonout_soilstr(:,:)
REAL*8, ALLOCATABLE :: f_carbonout_surfmic(:,:)
REAL*8, ALLOCATABLE :: f_carbonout_soilmic(:,:)
REAL*8, ALLOCATABLE :: f_carbonout_soilmet(:,:)
REAL*8, ALLOCATABLE :: f_carbonout_slow(:,:)
REAL*8, ALLOCATABLE :: f_carbonout_armored(:,:)
REAL*8, ALLOCATABLE :: resppool_surfstr(:,:)
REAL*8, ALLOCATABLE :: resppool_surfmet(:,:)
REAL*8, ALLOCATABLE :: resppool_surfmic(:,:)
REAL*8, ALLOCATABLE :: resppool_soilstr(:,:)
REAL*8, ALLOCATABLE :: resppool_soilmet(:,:)
REAL*8, ALLOCATABLE :: resppool_soilmic(:,:)
REAL*8, ALLOCATABLE :: resppool_slow(:,:)
REAL*8, ALLOCATABLE :: resppool_armored(:,:)
REAL*8, ALLOCATABLE :: resp_surfstr(:,:)
REAL*8, ALLOCATABLE :: resp_surfmet(:,:)
REAL*8, ALLOCATABLE :: resp_surfmic(:,:)
REAL*8, ALLOCATABLE :: resp_soilstr(:,:)
REAL*8, ALLOCATABLE :: resp_soilmet(:,:)
REAL*8, ALLOCATABLE :: resp_soilmic(:,:)
REAL*8, ALLOCATABLE :: resp_slow(:,:)
REAL*8, ALLOCATABLE :: resp_armored(:,:)
REAL*8, ALLOCATABLE :: temp(:,:)
REAL*8, ALLOCATABLE :: combusted_leaf(:,:)
REAL*8, ALLOCATABLE :: combusted_abovewood(:,:)
REAL*8, ALLOCATABLE :: combusted_cwd(:,:)
REAL*8, ALLOCATABLE :: combusted_surfstr(:,:)
REAL*8, ALLOCATABLE :: combusted_surfmet(:,:)
REAL*8, ALLOCATABLE :: combusted_surfmic(:,:)
REAL*8, ALLOCATABLE :: combusted_soilstr(:,:)
```

```
REAL*8, ALLOCATABLE :: combusted_soilmet(:,:)
REAL*8, ALLOCATABLE :: combusted_soilmic(:,:)
REAL*8, ALLOCATABLE :: combusted_slow(:,:)
REAL*8, ALLOCATABLE :: combusted_armored(:,:)
REAL*8, ALLOCATABLE :: nonCombusted_leaf(:,:)
REAL*8, ALLOCATABLE :: nonCombusted_abovewood(:,:)
REAL*8, ALLOCATABLE :: nonCombusted_belowwood(:,:)
REAL*8, ALLOCATABLE :: nonCombusted_froot(:,:)
REAL*8, ALLOCATABLE :: fuelwoodout(:,:)
REAL*8, ALLOCATABLE :: wresp(:,:)
REAL*8, ALLOCATABLE :: wcomb(:,:)
REAL*8, ALLOCATABLE :: wherb(:,:)
REAL*8, ALLOCATABLE :: wbiof(:,:)
REAL*8, ALLOCATABLE :: hresp(:,:)
REAL*8, ALLOCATABLE :: hcomb(:,:)
REAL*8, ALLOCATABLE :: hherb(:,:)

!in getAgeClassBF
REAL*8, ALLOCATABLE :: ageClassIndex(:,:)
REAL*8, ALLOCATABLE :: BFallClasses(:,:)
REAL*8, ALLOCATABLE :: BFleftCurrentMonth(:,:)
REAL*8, ALLOCATABLE :: BFtemp(:,:)
REAL*8, ALLOCATABLE :: ageCurrentClass(:,:)

!in organizeAgeClasses
REAL*8, ALLOCATABLE :: ageClassSorted(:,:)
REAL*8, ALLOCATABLE :: ageClassSortedInd(:,:)
REAL*8, ALLOCATABLE :: tempAge(:,:)

!in processData
REAL*8, ALLOCATABLE :: NPPmonthly(:,:)
REAL*8, ALLOCATABLE :: respmonthly(:,:)
REAL*8, ALLOCATABLE :: combmonthly(:,:)
REAL*8, ALLOCATABLE :: herbmonthly(:,:)
REAL*8, ALLOCATABLE :: biofmonthly(:,:)
REAL*8, ALLOCATABLE :: respEQ(:,:)
REAL*8, ALLOCATABLE :: combEQ(:,:)
REAL*8, ALLOCATABLE :: herbEQ(:,:)
REAL*8, ALLOCATABLE :: biofEQ(:,:)
REAL*8, ALLOCATABLE :: NPPmonthly_hg(:,:)
REAL*8, ALLOCATABLE :: respmonthly_hg(:,:)
REAL*8, ALLOCATABLE :: combmonthly_hg(:,:)
REAL*8, ALLOCATABLE :: herbmonthly_hg(:,:)
REAL*8, ALLOCATABLE :: biofmonthly_hg(:,:)
REAL*8, ALLOCATABLE :: respEQ_hg(:,:)
REAL*8, ALLOCATABLE :: combEQ_hg(:,:)
REAL*8, ALLOCATABLE :: herbEQ_hg(:,:)
REAL*8, ALLOCATABLE :: biofEQ_hg(:,:)
```

```
REAL*8, ALLOCATABLE :: evapEQ_hg(:,:)
REAL*8, ALLOCATABLE :: reemitEQ_hg(:,:)
REAL*8, ALLOCATABLE :: photoEQ_hg(:,:)
REAL*8, ALLOCATABLE :: leafpoolEQ_hg(:,:)
REAL*8, ALLOCATABLE :: slowpoolEQ_hg(:,:)
REAL*8, ALLOCATABLE :: armoredpoolEQ_hg(:,:)
REAL*8, ALLOCATABLE :: surfstrpoolEQ_hg(:,:)
REAL*8, ALLOCATABLE :: soilstrpoolEQ_hg(:,:)
REAL*8, ALLOCATABLE :: surfmetpoolEQ_hg(:,:)
REAL*8, ALLOCATABLE :: soilmetpoolEQ_hg(:,:)
REAL*8, ALLOCATABLE :: surfmicpoolEQ_hg(:,:)
REAL*8, ALLOCATABLE :: soilmicpoolEQ_hg(:,:)
REAL*8, ALLOCATABLE :: HgAqEQ_hg(:,:)
REAL*8, ALLOCATABLE :: reemmonthly_hg(:,:)
REAL*8, ALLOCATABLE :: photmonthly_hg(:,:)
REAL*8, ALLOCATABLE :: slowmonthly(:,:)
REAL*8, ALLOCATABLE :: armoredmonthly(:,:)
REAL*8, ALLOCATABLE :: surfstrmonthly(:,:)
REAL*8, ALLOCATABLE :: surfmetmonthly(:,:)
REAL*8, ALLOCATABLE :: surfmicmonthly(:,:)
REAL*8, ALLOCATABLE :: soilstrmonthly(:,:)
REAL*8, ALLOCATABLE :: soilmetmonthly(:,:)
REAL*8, ALLOCATABLE :: soilmicmonthly(:,:)
REAL*8, ALLOCATABLE :: leafmonthly(:,:)
REAL*8, ALLOCATABLE :: slowmonthly_hg(:,:)
REAL*8, ALLOCATABLE :: armoredmonthly_hg(:,:)
REAL*8, ALLOCATABLE :: surfstrmonthly_hg(:,:)
REAL*8, ALLOCATABLE :: surfmetmonthly_hg(:,:)
REAL*8, ALLOCATABLE :: surfmicmonthly_hg(:,:)
REAL*8, ALLOCATABLE :: soilstrmonthly_hg(:,:)
REAL*8, ALLOCATABLE :: soilmetmonthly_hg(:,:)
REAL*8, ALLOCATABLE :: soilmicmonthly_hg(:,:)
REAL*8, ALLOCATABLE :: leafmonthly_hg(:,:)
REAL*8, ALLOCATABLE :: HgAqmonthly(:,:)
REAL*8, ALLOCATABLE :: leafpoolEQ(:,:)
REAL*8, ALLOCATABLE :: slowpoolEQ(:,:)
REAL*8, ALLOCATABLE :: armoredpoolEQ(:,:)
REAL*8, ALLOCATABLE :: surfstrpoolEQ(:,:)
REAL*8, ALLOCATABLE :: soilstrpoolEQ(:,:)
REAL*8, ALLOCATABLE :: surfmetpoolEQ(:,:)
REAL*8, ALLOCATABLE :: soilmetpoolEQ(:,:)
REAL*8, ALLOCATABLE :: surfmicpoolEQ(:,:)
REAL*8, ALLOCATABLE :: soilmicpoolEQ(:,:)
REAL*8, ALLOCATABLE :: biomeAnnual_Hg(:,:)
!in doHgDeposition
REAL*8, ALLOCATABLE :: Hg0dry(:,:)
REAL*8, ALLOCATABLE :: HgIIdry(:,:)
REAL*8, ALLOCATABLE :: HgIIwet(:,:)
```

```
REAL*8, ALLOCATABLE :: Hg0dry_mo(:,:,:)
REAL*8, ALLOCATABLE :: HgIIdry_mo(:,:,:)
REAL*8, ALLOCATABLE :: HgIIwet_mo(:,:,:)
REAL*8, ALLOCATABLE :: HgP(:,:)
REAL*8, ALLOCATABLE :: HgAq(:,:)
REAL*8, ALLOCATABLE :: hHgAq(:,:)
REAL*8, ALLOCATABLE :: Hg0_surf_leaf(:,:)
REAL*8, ALLOCATABLE :: Hg0_surf_soil(:,:)
REAL*8, ALLOCATABLE :: HgII_surf_leaf(:,:)
REAL*8, ALLOCATABLE :: HgII_surf_soil(:,:)
REAL*8, ALLOCATABLE :: maxallLAI(:)
REAL*8, ALLOCATABLE :: fstom(:,:)
REAL*8, ALLOCATABLE :: fleaf(:,:)
REAL*8, ALLOCATABLE :: fsoil(:,:)
REAL*8, ALLOCATABLE :: fsum(:,:)
REAL*8, ALLOCATABLE :: freemitted(:,:)
REAL*8, ALLOCATABLE :: reemitted(:,:)
REAL*8, ALLOCATABLE :: temp_hg(:,:)
REAL*8, ALLOCATABLE :: photoreduced(:,:)
REAL*8, ALLOCATABLE :: Hg0out(:,:)
```

REVISION HISTORY:

```
9 July 2010 - C. Carouge  - Adapted for coupling with GEOS-Chem and restart
                            offline simulations.
```

1.2.1 readCASAparam

Subroutine readCASAparam reads some input for CASA **INTERFACE:**

```
SUBROUTINE readCASAparam
```

USES:

```
  USE defineConstants
```

REVISION HISTORY:

1.2.2 makeCASAarrays

Subroutine makeCASAarrays allocate all allocatable arrays.

INTERFACE:

```
SUBROUTINE makeCASAarrays
```

USES:

```
USE defineConstants
```

REVISION HISTORY:

1.2.3 initialize

Subroutine initialize initialize all allocatable arrays to 0

INTERFACE:

```
SUBROUTINE initialize
```

REVISION HISTORY:

1.3 Fortran: Module Interface defineConstants

Define some constants for the run, including input and output directories.

INTERFACE:

```
MODULE defineConstants
```

USES:

```
IMPLICIT NONE
```

REVISION HISTORY:

```
09 July 2010 - C. Carouge  - Adapted to restart simulations.
```

1.4 Fortran: Module Interface dorestart_mod

Module DORESTART_MOD contains subroutines to save and read data created by GTMM and used to restart a stand-alone run or a coupled run with GEOS-Chem. **INTERFACE:**

```
MODULE DORESTART_MOD
```

USES:

```
IMPLICIT NONE
```

REVISION HISTORY:

```
16 Dec 2009 - C. Carouge   - Initial version
```

1.4.1 doSaveHgforGC

Subroutine doSaveHgforGC saves Hg pools at equilibrium to be used when GTMM is coupled to GEOS-Chem.

INTERFACE:

```
SUBROUTINE doSaveHgforGC
```

USES:

```
USE defineConstants
USE defineArrays
```

REVISION HISTORY:

```
03 Nov 2009 - C. Carouge   - Initial version
```

1.4.2 doReadHgforGC

Subroutine doReadHgforGC reads Hg pools at equilibrium to use them in GTMM when coupled to GEOS-Chem.

INTERFACE:

```
SUBROUTINE doReadHgforGC
```

USES:

```
USE defineConstants
USE defineArrays
```

REVISION HISTORY:

```
03 Nov 2009 - C. Carouge   - Initial version
```

1.4.3 doSaveCASAforRestart

Subroutine doSaveCASAforRestart saves CASA values at equilibrium to be used when an equilibrium run for GTMM is continued.

INTERFACE:

```
SUBROUTINE doSaveCASAforRestart
```

USES:

```
USE defineConstants
USE defineArrays
```

REVISION HISTORY:

```
16 Dec 2009 - C. Carouge   - Initial version
```

1.4.4 doReadCASAfromRestart

Subroutine doReadCASAfromRestart reads CASA values at equilibrium to be used when an equilibrium run for GTMM is continued.

INTERFACE:

```
SUBROUTINE doReadCASAfromRestart
```

USES:

```
USE defineConstants
USE defineArrays
```

REVISION HISTORY:

```
16 Dec 2009 - C. Carouge   - Initial version
```

1.5 Fortran: Module Interface input_gtmm_mod

Module INPUT_GTMM_MOD contains subroutines to read input file for GTMM.

INTERFACE:

```
MODULE INPUT_GTMM_MOD
```

USES:

```
USE defineConstants
USE defineArrays

IMPLICIT NONE
PRIVATE
!PRIVATE DATA MEMBERS:

INTEGER, PARAMETER :: FIRSTCOL = 38
INTEGER, PARAMETER :: HEADER   = 8
CHARACTER(LEN=255) :: FILENAME = 'input.gtmm'
```

PUBLIC MEMBER FUNCTIONS:

```
PUBLIC :: READ_GTMM_INPUT_FILE
```

REVISION HISTORY:

```
17 Dec 2009 - C. Carouge   - Initial version
```

1.5.1 read_gtmm_input_file

Subroutine READ_GTMM_INPUT_FILE reads the input.gtmm file.

INTERFACE:

```
SUBROUTINE READ_GTMM_INPUT_FILE
```

REVISION HISTORY:

```
 17 Dec 2009 - C. Carouge  -- Initial version
```

1.6 Fortran: Module Interface loadCASAinput

Loads input files.

INTERFACE:

```
MODULE loadCASAinput
```

USES:

```
USE defineConstants
USE CasaRegridModule

IMPLICIT NONE
```

PUBLIC DATA MEMBERS:

```
CONTINUOUS FIELD MAPS
 REAL*8, ALLOCATABLE :: perc_tree(:,:)
 REAL*8, ALLOCATABLE :: perc_herb(:,:)

RESIZED CONTINUOUS FIELD MAPS
 REAL*8, ALLOCATABLE :: perc_tree1(:,:)
 REAL*8, ALLOCATABLE :: perc_herb1(:,:)
 REAL*8, ALLOCATABLE :: frac_tree(:,:)
 REAL*8, ALLOCATABLE :: frac_herb(:,:)
 REAL*8, ALLOCATABLE :: frac_veg(:,:)

CLIMATE FILES
 REAL*8, ALLOCATABLE :: airt(:,:,:)   !monthly air temperature
 REAL*8, ALLOCATABLE :: ppt(:,:,:)    !monthly precipitation
 REAL*8, ALLOCATABLE :: solrad(:,:,:) !monthly solar radiation,
                                      !Taken from Bishop and Rossow
                                      !average 83-99 from Jim Collatz

 REAL*8, ALLOCATABLE :: NDVI(:,:,:)   !monthly fraction PAR
 REAL*8, ALLOCATABLE :: BF(:,:,:)     !fraction gridcell tha burns
 REAL*8, ALLOCATABLE :: ppt_mo(:,:)   !sum of all precip in each mo
```

```
RESIZED CLIMATE FILES
 REAL*8, ALLOCATABLE :: airt1(:,:)
 REAL*8, ALLOCATABLE :: ppt1(:,:)
 REAL*8, ALLOCATABLE :: solrad1(:,:)
 REAL*8, ALLOCATABLE :: NDVI1(:,:)
 REAL*8, ALLOCATABLE :: BF1(:,:)
 REAL*8, ALLOCATABLE :: maxt(:,:)
 REAL*8, ALLOCATABLE :: mint(:,:)

OTHER FILES
 REAL*8, ALLOCATABLE :: soiltext(:,:)  ! soil type
 REAL*8, ALLOCATABLE :: veg(:,:)       ! vegetation map
 REAL*8, ALLOCATABLE :: fuelneed(:,:)  ! fuelwood needed
                                       !   per capita
 REAL*8, ALLOCATABLE :: popdens(:,:)   ! population density (/m2)
 REAL*8, ALLOCATABLE :: gridAreaa(:,:)
 REAL*8, ALLOCATABLE :: gridAreab(:,:)

RESIZED OTHER FILES
 REAL*8, ALLOCATABLE :: soiltext1(:,:)
 REAL*8, ALLOCATABLE :: veg1(:,:)
 REAL*8, ALLOCATABLE :: fuelneed1(:,:)
 REAL*8, ALLOCATABLE :: popdens1(:,:)


 REAL*8, ALLOCATABLE :: mask2(:,:)
```

REMARKS:

```
as 180 by 360 matrix for 1x1 degree, 360 by 720 for half by half
etc.  This routine will construct a 'mask' with vegetated gridcells
and will reshape them into an X by 1 matrix, where X is the total
number of vegetated grid cells.
The files which contain monthly varying parameters (i.e. precip)
should be given as (for 1x1) 180x360x12, and will be reshaped to
an X by 12 matrix, one column for each month
```

REVISION HISTORY:

```
9 July 2010 - C. Carouge  - Modified for coupled simulations with GEOS-Chem
                            or to restart offline simulations.
```

1.6.1 load_data

Reads input data.

INTERFACE:

```
SUBROUTINE load_data(LCPLE)
```

USES:

```
USE defineConstants
```

INPUT PARAMETERS:

```
LOGICAL, INTENT(IN)  :: LCPLE
```

REVISION HISTORY:

1.6.2 conv_to_1d

Convert maps into one column with only vegetated grid cells

INTERFACE:

```
SUBROUTINE CONV_TO_1D
```

USES:

```
USE defineConstants
```

REVISION HISTORY:

1.6.3 maskfile

This subroutine ...

INTERFACE:

```
function maskfile(dummy,mask3) result (masked_file)
```

USES:

```
USE defineConstants

implicit none
```

INPUT PARAMETERS:

```
real*8, dimension(columns, rows) :: dummy, mask3
```

RETURN VALUE:

```
real*8, dimension(n_veg,1)  :: masked_file
```

REVISION HISTORY:

1.6.4 mask12file

This subroutine ...

INTERFACE:

```
function mask12file(dummy12, mask3) result (masked_12file)
```

USES:

```
USE defineConstants

implicit none
```

INPUT PARAMETERS:

```
real*8, dimension(columns, rows, 12)  ::  dummy12
real*8, dimension(columns, rows)      ::  mask3
```

RETURN VALUE:

```
real*8, dimension(n_veg, 12)              ::  masked_12file
```

REVISION HISTORY:

1.7 Fortran: Module Interface Individual GTMM routines

Here follows a list of GTMM routines that do not belong to any F90 module.

1.7.1 CleanupCASAarrays

Subroutine CleanupCASAarrays deallocate all allocated arrays used for GTMM

INTERFACE:

```
SUBROUTINE CleanupCASAarrays
  !USES
  USE defineConstants
  USE loadCASAinput
  USE defineArrays

  implicit none
```

REVISION HISTORY:

1.7.2 GTMM_coupled

Main subroutine for GTMM when coupled to GEOS-Chem. Replace GTMM.f90.

INTERFACE:

```
 SUBROUTINE GTMM_coupled(year, month,  DD_Hg0, DD_HgII, WD_HgII, &
                         TS,   PREACC, RADSWG, Hg0reemit  )
```

USES:

```
   USE      defineConstants ! modify defineConstants.f90 to choose
                            ! parameters for your run

   USE      loadCASAinput

   USE      defineArrays

   USE      DORESTART_MOD     ! New module to save/read Hg data for
                              ! GEOS-CHEM. (ccc, 11/3/09)

   USE      INPUT_GTMM_MOD


   IMPLICIT NONE
```

INPUT PARAMETERS:

```
   REAL*8, INTENT(IN)           :: DD_Hg0(72, 46), DD_HgII(72, 46), &
                                   WD_HgII(72, 46)    ! Hg deposition info.

   REAL*8, INTENT(IN),  DIMENSION(72, 46)  :: TS, PREACC, RADSWG !Met field info
```

INPUT/OUTPUT PARAMETERS:

```
   INTEGER, INTENT(INOUT)       :: year, month
```

OUTPUT PARAMETERS:

```
   REAL*8, INTENT(OUT), DIMENSION(72, 46)  :: Hg0reemit  ! Reemitted flux,
                                                         ! output to GEOS-Chem
```

REVISION HISTORY:

```
    09 July 2010 - C. Carouge  - First version. Adapted from GTMM.f90
```

1.7.3 HgOutForGEOS

Subroutine HgOutForGEOS converts the (n_veg,1) data to 1x1, then to 4x5 grid then writes out the file for use by GEOS-Chem.

INTERFACE:

```
SUBROUTINE HgOutForGEOS(LCPLE, HgOreemit)
```

USES:

```
  USE defineConstants
  USE loadCASAinput
  USE defineArrays
  USE CasaRegridModule

  IMPLICIT NONE
```

INPUT PARAMETERS:

```
  LOGICAL, INTENT(IN) :: LCPLE
```

OUTPUT PARAMETERS:

```
  REAL*8, INTENT(OUT) :: HgOreemit(72, 46)
```

REVISION HISTORY:

```
   09 July 2010 - C. Carouge  - Modified to couple with GEOS-Chem
```

1.7.4 assignAgeClassToRunningPool

This subroutine ... running pool.

INTERFACE:

```
SUBROUTINE assignAgeClassToRunningPool
```

USES:

```
  USE defineConstants
  USE loadCASAinput
  USE defineArrays

  implicit none
```

REVISION HISTORY:

```
   09 July 2010 - C. Carouge  - Add parallelization.
```

1.7.5 assignRanPoolToAgeClass

This subroutine ...

INTERFACE:

```
SUBROUTINE assignRanPoolToAgeClass
```

USES:

```
USE defineConstants
USE loadCASAinput
USE defineArrays

implicit none
```

REVISION HISTORY:

```
09 July 2010 - C. Carouge  - Add parallelization.
```

1.7.6 doFPARandLAI

This subroutine ...

INTERFACE:

```
SUBROUTINE doFPARandLAI(FIRST)
```

USES:

```
USE defineConstants
USE loadCASAinput
USE defineArrays

implicit none
!INPUT PARAMETERS
LOGICAL, INTENT(IN) :: FIRST
```

REVISION HISTORY:

```
09 July 2010 - C. Carouge  - Adapted to restart simulations. Parallelization
```

1.7.7 doHerbCarbon

This subroutine ...

INTERFACE:

```
SUBROUTINE doHerbCarbon
```

USES:

```
USE defineConstants
USE loadCASAinput
USE defineArrays

IMPLICIT NONE
```

REVISION HISTORY:

```
09 July 2010 - C. Carouge  - Parallelized
```

1.7.8 doHerbCarbonHg

This subroutine ...

INTERFACE:

```
SUBROUTINE doHerbCarbonHg
```

USES:

```
  USE defineConstants
  USE loadCASAinput
  USE defineArrays

  IMPLICIT NONE
  !REVISION HISTORY
   09 July 2010 - C. Carouge  - Parallelization
```

1.7.9 doHerbivory

Subroutine doHerbivory calculate herbivory analog to McNaughton (Science, 1989) as fraction of foliage NPP.

INTERFACE:

```
SUBROUTINE doHerbivory
```

USES:

```
  USE defineConstants
  USE loadCASAinput
  USE defineArrays

  IMPLICIT NONE
```

REMARKS:

```
   Herbivory analog to McNaughton is computed as:

      log C = 2.04*(log NFP)-4.8   -->  C = NFP^2.04*10^(-4.8)

   where C= consumption, NFP = Net foliage production (NPP
   delivered to leaves)  units kJ/m2/yr
```

REVISION HISTORY:

```
   09 July 2010 - C. Carouge  - Parallelization.
```

1.7.10 doHgDeposition

This subrouitine ...

INTERFACE:

```
SUBROUTINE doHgDeposition
```

USES:

```
  USE defineConstants
  USE loadCASAinput
  USE defineArrays

  implicit none
```

REVISION HISTORY:

```
   09 July 2010 - C. Carouge  - Parallelization
```

1.7.11 doLatitude

This subroutine ...

INTERFACE:

```
SUBROUTINE doLatitude
```

USES:

```
  USE defineConstants
  USE loadCASAinput
  USE defineArrays
```

REVISION HISTORY:

1.7.12 doLeafRootShedding

Overview

Subroutine doLeafRootShedding define the scalars that predict the seasonality of leaf shedding and root decay based on changes in LAI This needs improvement.

References

Randerson, Thompson, Malmstrom, Field and Fung 1996, GBC 10(4) p585

INTERFACE:

```
SUBROUTINE doLeafRootShedding
```

USES:

```
  USE defineConstants
  USE loadCASAinput
  USE defineArrays

  implicit none
```

REVISION HISTORY:

```
   09 July 2010 - C. Carouge  - Parallelization
```

1.7.13 doMAXHg

Calculates the maximum mercury storage (gHg/m2) for each soil pool

INTERFACE:

```
SUBROUTINE doMAXHg
```

USES:

```
  USE defineConstants
  USE loadCASAinput
  USE defineArrays

  implicit none
```

REVISION HISTORY:

```
   09 July 2010 - C. Carouge  - Parallelization
```

1.7.14 doNPP

Subroutin doNPP calculate net primary production (NPP)

INTERFACE:

```
SUBROUTINE doNPP
```

USES:

```
  USE defineConstants
  USE loadCASAinput
  USE defineArrays

  implicit none
```

REVISION HISTORY:

```
   09 July 2010 - C. Carouge  - Parallelization
```

1.7.15 doOptimumTemperature

Defines the optimum temperature; this is the air temperature in the month when the LAI is highest

INTERFACE:

```
SUBROUTINE doOptimumTemperature
```

USES:

```
  USE defineConstants
  USE loadCASAinput
  USE defineArrays

  implicit none
```

REVISION HISTORY:

```
   09 July 2010 - C. Carouge  - Parallelization
```

1.7.16 doPET

Calculate potential evapotranspiration (PET) and the annual heat index (AHI)

INTERFACE:

```
SUBROUTINE doPET
```

USES:

```
  USE defineConstants
  USE loadCASAinput
  USE defineArrays

  IMPLICIT NONE
```

REVISION HISTORY:

```
   09 July 2010 - C. Carouge  - Parallelization
```

1.7.17 doSoilMoisture

This subroutine ...

INTERFACE:

```
SUBROUTINE doSoilMoisture
```

USES:

```
   USE defineConstants
   USE loadCASAinput
   USE defineArrays

   IMPLICIT NONE
```

REVISION HISTORY:

```
    09 July 2010 - C. Carouge  - Parallelization
```

1.7.18 doTreeCarbon

This subroutine ...

INTERFACE:

```
 SUBROUTINE doTreeCarbon
```

USES:

```
   USE defineConstants
   USE loadCASAinput
   USE defineArrays

   IMPLICIT NONE
```

REVISION HISTORY:

```
    09 July 2010 - C. Carouge  - Parallelization
```

1.7.19 doTreeCarbonHg

This subroutine ...

INTERFACE:

```
 SUBROUTINE doTreeCarbonHg
```

USES:

```
   USE defineConstants
   USE loadCASAinput
   USE defineArrays

   IMPLICIT NONE
```

REVISION HISTORY:

```
    09 July 2010 - C. Carouge  - Adapted for restarting simulations.
                                 Parallelization
```

1.7.20 getAgeClassBF

Get the burned fraction for the running age class The idea is that the oldest part of the gridcell burns first, because these parts contain the highest fuel loads

INTERFACE:

```
SUBROUTINE getAgeClassBF
```

USES:

```
  USE defineConstants
  USE loadCASAinput
  USE defineArrays

  IMPLICIT NONE
```

REVISION HISTORY:

```
   09 July 2010 - C. Carouge  - Parallelization
```

1.7.21 getFireParams

Calculates combustion completeness (CC, aka combustion factor or combustion efficiency.

INTERFACE:

```
SUBROUTINE getFireParams
```

USES:

```
  USE defineConstants
  USE loadCASAinput
  USE defineArrays

  implicit none
```

REMARKS:

```
   Fuel is split into live wood, live leaves (inc grass), fine
   litter and coarse litter (cwd)

        min CC   max CC   fuel type
   CC= [0.2      0.3      live wood
        0.8      1.0      live leaves
        0.9      1.0      fine litter
        0.2,     0.4]     coarse litter

   Scaling is as follows: for live material the CC is scaled
   linearly with NPP moisture scalar, dead material is scaled
   using the PPT over PET ratio, with a running mean to
   include some memory, which is greater for CWD
```

REVISION HISTORY:

```
09 July 2010 - C. Carouge  - Parallelization
```

1.7.22 getFuelWood

This subroutine ...

INTERFACE:

```
SUBROUTINE getFuelWood
```

USES:

```
USE defineConstants
USE loadCASAinput
USE defineArrays

implicit none
```

REVISION HISTORY:

```
09 July 2010 - C. Carouge  - Parallelization
```

1.7.23 getSoilMoistParams

This subroutine ...

INTERFACE:

```
SUBROUTINE getSoilMoistParams
```

USES:

```
USE defineConstants
USE loadCASAinput
USE defineArrays

IMPLICIT NONE
```

REVISION HISTORY:

```
09 July 2010 - C. Carouge  - Parallelization
```

1.7.24 getSoilParams

This subroutine ...

INTERFACE:

```
SUBROUTINE getSoilParams
```

USES:

```
  USE defineConstants
  USE loadCASAinput
  USE defineArrays

  IMPLICIT NONE
```

REVISION HISTORY:

```
   09 July 2010 - C. Carouge  - Parallelization
```

1.7.25 loadHgDeposition

This code reads in output from geos chem 4x5 degree, converts it to 1x1 degree, then converts it to (n_veg,1) for CASA

INTERFACE:

```
SUBROUTINE loadHgDeposition(LCPLE, DD_Hg0, DD_HgII, WD_HgII)
```

USES:

```
  USE defineConstants
  USE loadCASAinput
  USE defineArrays
  USE CasaRegridModule

  IMPLICIT NONE
```

INPUT PARAMETERS:

```
  LOGICAL, INTENT(IN)           :: LCPLE

  REAL*8, INTENT(IN), OPTIONAL  :: DD_Hg0(72, 46), DD_HgII(72, 46)
  REAL*8, INTENT(IN), OPTIONAL  :: WD_HgII(72, 46)
```

REVISION HISTORY:

```
  15 Dec 09 - C. Carouge  - Add arguments for coupling with GEOS-Chem
                          - Change format of emission years file to
                            facilitate restart.
```

1.7.26 load_GC_data

Subroutine load_GC_data is only used with GTMM coupled to GEOS-Chem. The subroutine regrid the temperature, precipation and radiation fields to 1x1. The met fields are read in GTMM_DR in GEOS_Chem.

INTERFACE:

```
SUBROUTINE load_GC_data(month, TS, PREACC, RADSWG)
```

USES:

```
  USE defineConstants
  USE loadCASAinput
  USE CasaRegridModule

  implicit none
```

INPUT PARAMETERS:

```
  INTEGER, INTENT(IN) :: month
  REAL*8, INTENT(IN), DIMENSION(I4x5, J4x5)  :: TS, &
       PREACC, RADSWG
```

REVISION HISTORY:

```
   09 July 2010 - C. Carouge  - Initial version
```

1.7.27 organizeAgeClasses

This routine reorganizes the pools according to the age classes so that the oldest comes first (and will run and burn first)

INTERFACE:

```
SUBROUTINE organizeAgeClasses
```

USES:

```
  USE defineConstants
  USE loadCASAinput
  USE defineArrays

  IMPLICIT NONE
```

REVISION HISTORY:

```
   09 July 2010 - C. Carouge  - Parallelization
```

1.7.28 processData

This subroutine ...

INTERFACE:

```
SUBROUTINE processData
```

USES:

```
  USE defineConstants
  USE loadCASAinput
  USE defineArrays

  IMPLICIT NONE
```

REVISION HISTORY:

```
   09 Jul 2010 - C. Caroauge - Initial version
```

1.7.29 sort_pick_veg

This subroutine ...

INTERFACE:

```
SUBROUTINE sort_pick_veg(arr, ind)
```

USES:

```
  USE defineConstants

  implicit none
```

INPUT/OUTPUT PARAMETERS:

```
  REAL*8, dimension(n_veg, n_age_classes), intent(INOUT) :: ind
  REAL*8, dimension(n_veg, n_age_classes), intent(INOUT) :: arr
```

REVISION HISTORY:

```
   09 Jul 2010 - C. Carouge - Initial version
```