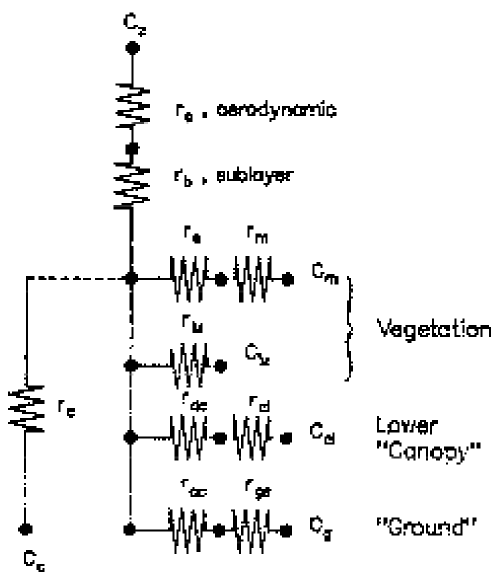
How does the current implementation of dry deposition in GEOS-Chem work?

GSF Tutorial Talk

Bess Sturges Corbitt

Oct. 20th, 2011

"Dry deposition is the process by which atmospheric trace chemicals are transferred by air motions to the surface of the Earth." (Wesely & Hicks 2000)



Wesely 1989

Resistance in series model

$$v_d \equiv -F_c/C_z$$

$$v_d = (r_a + r_b + r_c)^{-1}$$

Want to output dry deposition fluxes and velocities in GEOS-Chem? Select in input.geos diagnostic menu:

ND44: Drydep flx/vel : 1 all

1D array (surface)

species you want to output (all or #s for tracers)

Things you need to know to add dry deposition for a new species:

r_c values calculated based SO_2 and O_3, then modified for other gases

Mesophyll resistance: $r_m = (H^*/3000 + 100f_0)^{-1}$

f_0 ≡ “reactivity factor for oxidation of biological substances”

Typical values:

0 : non-reactive, e.g. SO_2, NO, HCHO

0.1 : slightly reactive, e.g. NO_2 ,PAN, HNO_2

1 : highly reactive, e.g. O_3, H_2O_2

Code examples

```
! O3 (as part of Ox)
ELSE IF ( N == IDTOX ) THEN
   NUMDEP          = NUMDEP + 1
   NTRAIND(NUMDEP) = IDTOX
   NDVZIND(NUMDEP) = NUMDEP
   DEPNAME(NUMDEP) = 'O3'
   HSTAR(NUMDEP)   = 0.01d0
   F0(NUMDEP)      = 1.0d0
   XMW(NUMDEP)     = 48d-3
   AIROSOL(NUMDEP) = .FALSE.
```

```
! Hg0 -- Elemental Mercury
IF ( N == ID_Hg0(ID_Hg_tot) ) THEN
  NUMDEP          = NUMDEP + 1
  NTRAIND(NUMDEP) = ID_Hg0(ID_Hg_tot)
  NDVZIND(NUMDEP) = NUMDEP
  DEPNAME(NUMDEP) = 'Hg0'
  HSTAR(NUMDEP)   = 0.11
  ! F0 consistent with Lin et al (2006)
  F0(NUMDEP)      = 1.0d-5
  XMW(NUMDEP)     = 201d-3
  AIROSOL(NUMDEP) = .FALSE.
ENDIF
```

GEOS-Chem: drydep_mod.F

```
!  Module Routines
!  ===========================================================================
!  (1 ) DO_DRYDEP         : Dry deposition driver routine
!  (2 ) DVZ_MINVAL        : Sets minimum drydep velocities for SULFATE tracers
!  (3 ) METERO            : Computes meterological fields for dry deposition
!  (4 ) DRYFLX            : Applies drydep losses from SMVGEAR to tracer array
!  (5 ) DRYFLXRnPbBe      : Applies drydep losses to 210Pb and 7Be
!  (6 ) DRYFLXH2HD        : Applies drydep losses to H2 and HD
!  (7 ) DEPVEL            : Computes dry deposition velocities (by D. Jacob)
!  (8 ) DIFFG             : Computes diffusion coefficient for a gas
!  (9 ) MODIN             : Reads inputs for DEPVEL from "drydep.table"
!  (10) RDDRYCF           : Reads drydep polynomial coeffs from "drydep.coef"
!  (11) AERO_SFCRSI       : Computes dust sfc resistance ff Seinfeld et al 86
!  (12) AERO_SFCRSII      : Computes dust sfc resistance ff Zhang et al 2001
!  (13) INIT_DRYDEP       : Initializes and allocates module arrays
!  (14) CLEANUP_DRYDEP    : Deallocates module arrays
```

SUBROUTINE DO_DRYDEP (called from main.F)

(1) Reads in drydep coefficients & land types: CALL RDDRYCF, CALL MODIN
(2) Reads in 1-D met fields: CALL METERO
(3) Calculates dry deposition velocity [m/s]: CALL DEPVEL
(4) Calculates dry deposition frequencies [1/s]: (over entire PBL)
(5) Archive diagnostics in AD44.

SUBROUTINE RDDRYCF

Reads in the file data_dir/drydep_200203/drydep.coef :

```
Baldocchi drydep
 -3.58E-01  3.02E+00  3.85E+00 -9.78E-02 -3.66E+00  1.20E+01  2.52E-01 -7.80E+00
  2.26E-01  2.74E-01  1.14E+00 -2.19E+00  2.61E-01 -4.62E+00  6.85E-01 -2.54E-01
  4.37E+00 -2.66E-01 -1.59E-01 -2.06E-01
```

These are polynomial coefficients for use in calculating stomatal resistance

SUBROUTINE MODIN

Reads in Olson land data from data_dir/drydep_200203/drydep.table :

```
    !=================================================================
    ! For each of the NVEGTYPE Olson land types, read:
    !
    ! IOLSON (INTEGER) : Olson surface type ID #
    ! IDEP   (INTEGER) : Drydep ID # corresponding to IOLSON
    ! IZO    (INTEGER) : Roughness height [1e-4 m]
    !=================================================================
```

In drydep.table :

```
Olson ID# - deposition ID# - z0 in 1.E-4 m
   1    11    10    Water
   2    10 25000    Urban
   3     5   100    Shrub                                  etc.
```

```
    !=================================================================
    ! Read in resistances for each surface type (see "depvel.f")
    ! IRI,IRLU,IRAC,IRGSS,IRGSO,IRCLS,IRCLO,IVSMAX
    !=================================================================
```

In drydep.table :

```
**  Resistances (s m-1) for each deposition surface type,
**  and maximum deposition velocity Vsmax(1.E-2 cm s-1) for aerosol
  Type  Ri  Rlu  Rac RgsS RgsO RclS RclO Vsmax
    1 9999 9999    0  100 3500 9999 1000  100   Snow/Ice (Wesely) - listed first.
    2  200 9000 2000  500  200 2000 1000  100   Deciduous forest (Wesely)
    3  400 9000 2000  500  200 2000 1000  100   Coniferous forest (Wesely)
    4  200 9000  200  150  150 2000 1000  100   Agricultural land (Wesely)      etc.
```

We already have the map of Olson land types from the subroutine RLAND (in rland.F) which is called in main.F when initializing.

“Map” is in vegtype.global in the data directory for the resolution you're running in.

This is a confusing file format that looks something like this:

I,J   # land types in gridbox

**land type #s   **% of grid box (i.e. 91.0%, 1.3%, 7.7%)

```
22  10   3   0  63  23 910  13  77
23  10   8   0  63  68  65  23  55  25  52 434  61  12  37 164  25 101 166
24  10   4   0  65  25  52 902  12  61  25
25  10   2   0  41 987  13
```

***I think! Don't trust me completely on this. Verify!*

Note: This is something you will want to change if you use other land models / climate.

Back in DO_DRYDEP....

SUBROUTINE METERO

Calculates useful meteorological values such as the: surface temperature, Monin-Obhukov length, solar radiation, roughness heights, PBL, ice/snow flag, etc.

This is where it all happens. →

```
CALL DEPVEL( MAXIJ, RADIAT,  TC0,    SUNCOS, F0,  HSTAR,
&              XMW,   AIROSOL, USTAR, CZ1,    OBK, CFRAC,
&           ZH,    LSNOW,   DVEL,  AZO,    RHB, DVRA )
```

```
C     Subroutine computes the dry deposition velocities using
C      a resistance-in-series model.
C
C** Contact: D.J. Jacob, Harvard U. (djj@io.harvard.edu)
C** Modularized by G.M. Gardner, Harvard U.
C** Version 3.2:   5/27/97

! Need as landtype input for each grid square (I,J) see (RDLAND & F77_CMN_VEL):
!    IJREG(JLOOP)      - # of landtypes in grid square
!    IJLAND(IJLOOP,LDT) - Land type ID for element LDT =1, IJREG(IJLOOP)
!                         (could be from any source - mapped to deposition
!                         surface ID in input unit 65)
!    IJUSE(IJLOOP,LDT)  - Fraction ((per mil) of gridbox area occupied by
!                         land type element LDT
!
! Need as leaf area index see (RDLAI & F77_CMN_VEL):
!    XYLAI(IJLOOP,LDT)  - Leaf Area Index of land type element LDT
!
! Need as meteorological input for each grid square(I,J) (passed):
!    RADIAT(IJLOOP) - Solar radiation in W m-2
!    TEMP(IJLOOP)   - Surface air temperature in K
!    SUNCOS(IJLOOP) - Cosine of solar zenith angle
!    LSNOW(IJLOOP)  - Logical for snow and sea ice
!    RHB(IJLOOP)    - Relative humidity at the surface
```

```
! Need as input for each species K (passed):
!     F0(K)          - reactivity factor for oxidation of biological substances
!     HSTAR(K)       - Henry's Law constant
!     XMW(K)         - Molecular weight (kg/mole) of species K
!                      (used to calculate molecular diffusivities)
!     AIROSOL(K)     - LOGICAL flag (T = aerosol species;
!                                   F = gas-phase species)

! Some variables used in the subroutine (passed):
!     LRGERA(IJLOOP) T -> stable atmosphere; a high aerodynamic resistance
!                      (RA=1.E4 m s-1) is imposed; else RA is calculated
!     USTAR(IJLOOP)  - Friction velocity (m s-1)
!     CZ1(IJLOOP)    - Altitude (m) at which deposition velocity is computed
!     OBK(IJLOOP)    - Monin-Obukhov length (m): set to 1.E5 m under neutral
!                      conditions
!     CFRAC(IJLOOP)  - Fractional cloud cover
!     ZH(IJLOOP)     - Mixing depth (m)
!
! Some variables used in the subroutine:
!     MAXDEP         - the maximum number of species for which the dry
!                      deposition calculation is done
!     ZO(LDT)        - Roughness height (m) for specific surface type indexed
!                      by LDT
!     RSURFC(K,LDT)  - Bulk surface resistance (s m-1) for species K to
!                      surface LDT
!     C1X(K)         - Total resistance to deposition (s m-1) for species K
!
! Returned:
!     DVEL(IJLOOP,K) - Deposition velocity (m s-1) of species K
```

(1) Calculate bulk surface resistance for gases

```
RT = 1000.0D0*EXP(-TEMPC-4.0D0)
```

(2) Get the stomatal a.k.a. **internal resistance RI** for water vapor for each land type II, per unit area of leaf (we'll later modify for each gas)

```
RI(LDT) = DBLE(IRI(II))
```

(3) Get the **cuticular resistance RLU** per unit area leaf and divide by leaf area index, add on the surface resistance and get value for bulk canopy

```
RLU(LDT)= DBLE(IRLU(II))/XYLAI(IJLOOP,LDT)+RT
```

(4) Get the other land-type specific resistances: **resistance depending on canopy height and density RAC, ground surface resistance RGS*** (soil, leaf litter, snow, water), **lower canopy resistance RCL*** (leaves, twig, bark); where **S** is for SO_2 and **O** is for O_3

```
RAC(LDT)  = MAX(DBLE(IRAC(II)), 1.D0)
RGSS(LDT) = MAX(DBLE(IRGSS(II)) + RT ,1.D0)
RGSO(LDT) = MAX(DBLE(IRGSO(II)) + RT ,1.D0)
RCLS(LDT) = DBLE(IRCLS(II)) + RT
RCLO(LDT) = DBLE(IRCLO(II)) + RT
```

(5) Adjust stomatal resistance for insolation and temperature

```
      IF (TEMPC .GT. 0.D0 .AND. TEMPC .LT. 40.D0)
 *        GFACT = 400.D0/TEMPC/(40.0D0-TEMPC)
      GFACI = 100.D0
      IF (RAD0.GT.0.D0 .AND. XYLAI(IJLOOP,LDT).GT.0.D0) THEN
         GFACI=1.D0/BIOFIT(DRYCOEFF,XYLAI(IJLOOP,LDT),
 *            SUNCOS(IJLOOP),CFRAC(IJLOOP))
      ENDIF

      RIX = RIX*GFACT*GFACI
```

(6) Modify the resistances for the specific gas

```
         RIXX = RIX*DIFFG(TEMPK,PRESS,XMWH2O)/
 C             DIFFG(TEMPK,PRESS,XMW(K))
 C             + 1.D0/(HSTAR(K)/3000.D0+100.D0*F0(K))

         RLUXX = RLU(LDT)/(HSTAR(K)/1.0D+05 + F0(K))

         RGSX = 1.D0/(HSTAR(K)/1.0D+05/RGSS(LDT) +
 1             F0(K)/RGSO(LDT))
         RCLX = 1.D0/(HSTAR(K)/1.0D+05/RCLS(LDT) +
 1             F0(K)/RCLO(LDT))
```

(7) Calculate the **bulk surface resistance of the canopy RSURFC** by adding up all the parallel and in series resistances (not shown)

Warning!

```
155        IF (.NOT. AIROSOL(K)) GOTO 160
                     .
                     .
        (a bunch of stuff not for gases)
                     .
                     .
160     CONTINUE
```

(8) Calculate the aerodynamic and surface layer resistances (not shown). This is where it uses all the met data you read in and calculates the resistances according to the equations in the references. Tip: watch out for #if define (GCAP) etc. if you are using/adding met fields with different names for things, etc. Also lots of go to statements.

(9) Add the aerodynamic and surface layer resistance to the bulk canopy resistance:

```
C1X(K) = RA + RB + RSURFC(K,LDT)
```

(10) Finally add up the the weighted dry deposition velocities for each land type within a grid box.

```
VD(K) = VK(K) +.001D0*DBLE(IJUSE(IJLOOP,LDT))/C1X(K)
```

(11) Bug check etc. and we're done! Return the dry deposition velocity.

```
DVEL(IJLOOP,K) = VD(K)
```

Back in the subroutine DO_DRYDEP, we calculate dry dep frequency over the entire PBL.

```
! Dry deposition frequency [1/s]
DEPSAV(I,J,N) = ( DVZ / 100.d0 ) / THIK
```

Archive in the diagnostic ND44 if you asked for it in input.geos.

```
! ND44 diagnostic: drydep velocity [cm/s]
IF ( ND44 > 0 ) THEN
   AD44(I,J,N,2) = AD44(I,J,N,2) + DVZ
ENDIF
```

Back to main.F but we haven't actually applied dry deposition to your tracers yet. We've just saved the dry deposition velocities.

But first: A note on aerosols.

Aerosols are treated differently from gases.

FUNCTION ADUST_SFCRSII: Used for all aerosols except dust, suflate, and seasalt to calculate the aerodynamic resistance of non-size resolved aerosol. Ref. Zhange et al. 2001.

```
!...Model theroy
!    Vd = Vs + 1./(Ra+Rs)
!      where Vs is the gravitational settling velocity,
!      Ra is the aerodynamic resistance above the canopy
!      Rs  is the surface resistance
!    Here we calculate Rs only..
!    Rs = 1 / (Eo*Ustar*(Eb+Eim+Ein)*R1)
!      where Eo is an empirical constant ( = 3.)
!      Ustar is the friction velocity
!      Collection efficiency from
!        Eb,  [Brownian diffusion]
!        Eim, [Impaction]
!        Ein, [Interception]
!      R1 is the correction factor representing the fraction
!         of particles that stick to the surface.
```

For seasalt: FUNCTION AERO_SFCRSII which includes hygroscopic growth.
For dust: FUNCTION DUST_SFCRSII.

Assuming you are using non-local mixing scheme, there is a call to DO_PBL_MIX_2 back in main.F

Inside vdiff_mod.F90 we have the subroutine DO_PBL_MIX_2
… which then calls the subroutine VDIFFDR

After calculating emissions flux (eflx), we calculate dry deposition flux (dflx) for the PBL (not shown).

Total surface flux = emissions – dry deposition:

```
sflx(I,J,:) = eflx(I,J,:) - dflx(I,J,:) ! kg/m2/s
```

Pass this flux when you call the subroutine VDIFF...
Which renames sflx to cflx, does a bunch of other stuff and then calls the subroutine PBLDIF...

Computes kinematic surface flux (qfs) by dividing by air density

```
kqfs(:,m) = cflx(:,m)*rrho(:)
```

Computes counter-gradient terms

```
cgq(:,k,m) = kqfs(:,m)*cgs(:,k)
```

Back to VDIFF, this is used to calculate qmx which is then passed to the subroutine QVDIFF, for diffusing constituents, where it is called qm1...

Which is used to compute zfq... which is used to calculate qp1... which is the final constituent! This is put back into the as2 array in VDIFF, which is then passed all the way back up to VDIFFDR and DO_PBL_MIX2 which puts it in STT. THE END.