

GEOS-Chem-Adjoint-V8

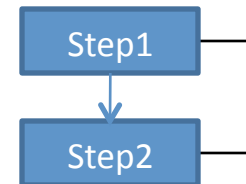
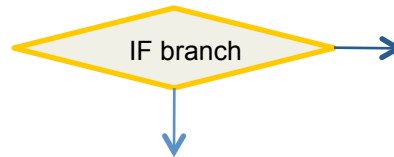
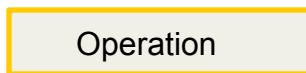
Flowchart descriptions of

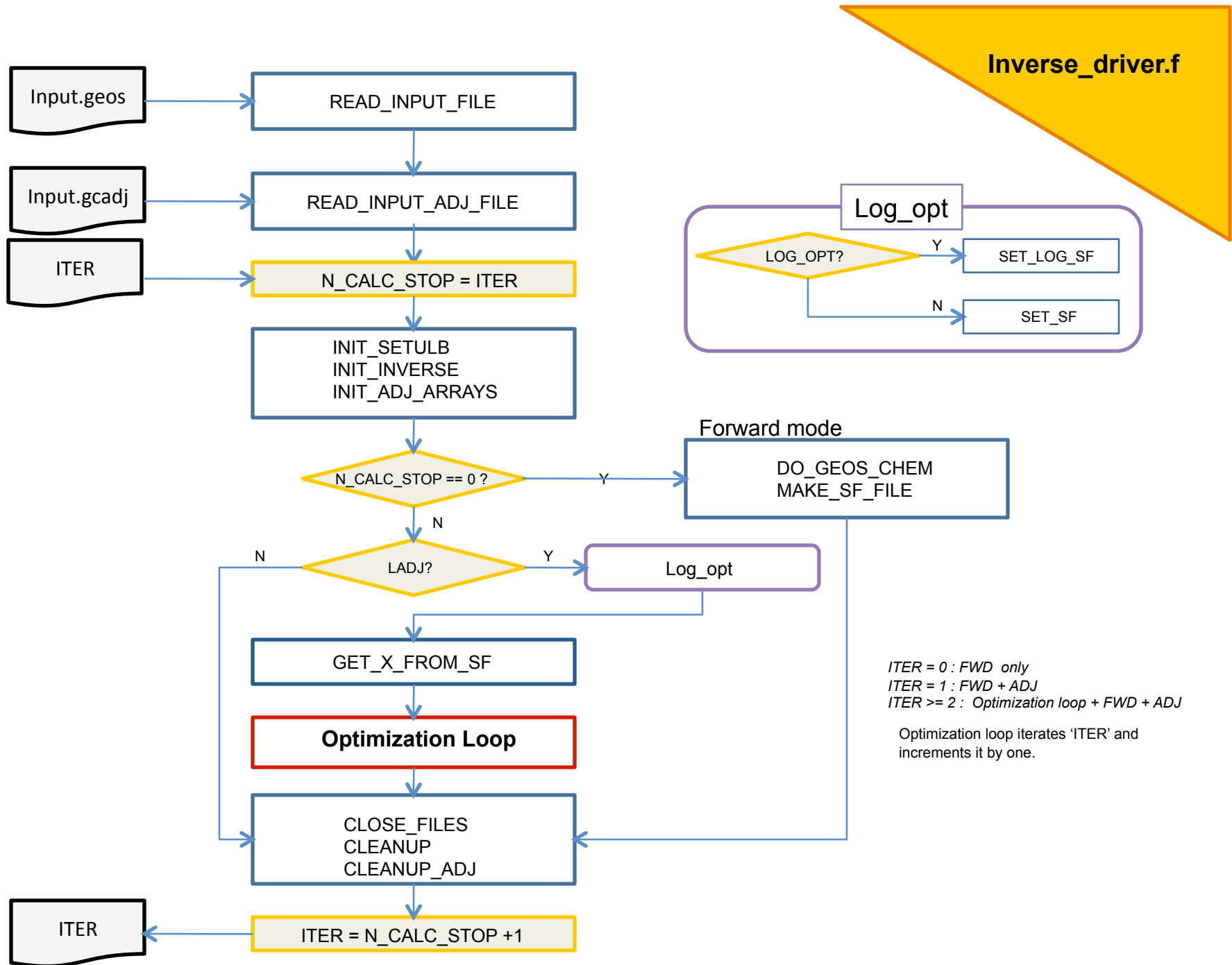
- Inverse_driver.f
- Geos_chem_mod.f
- Chemistry_mod.f
- Chemdr.f
- Geos_chem_adj.f

Jan/21, 2010

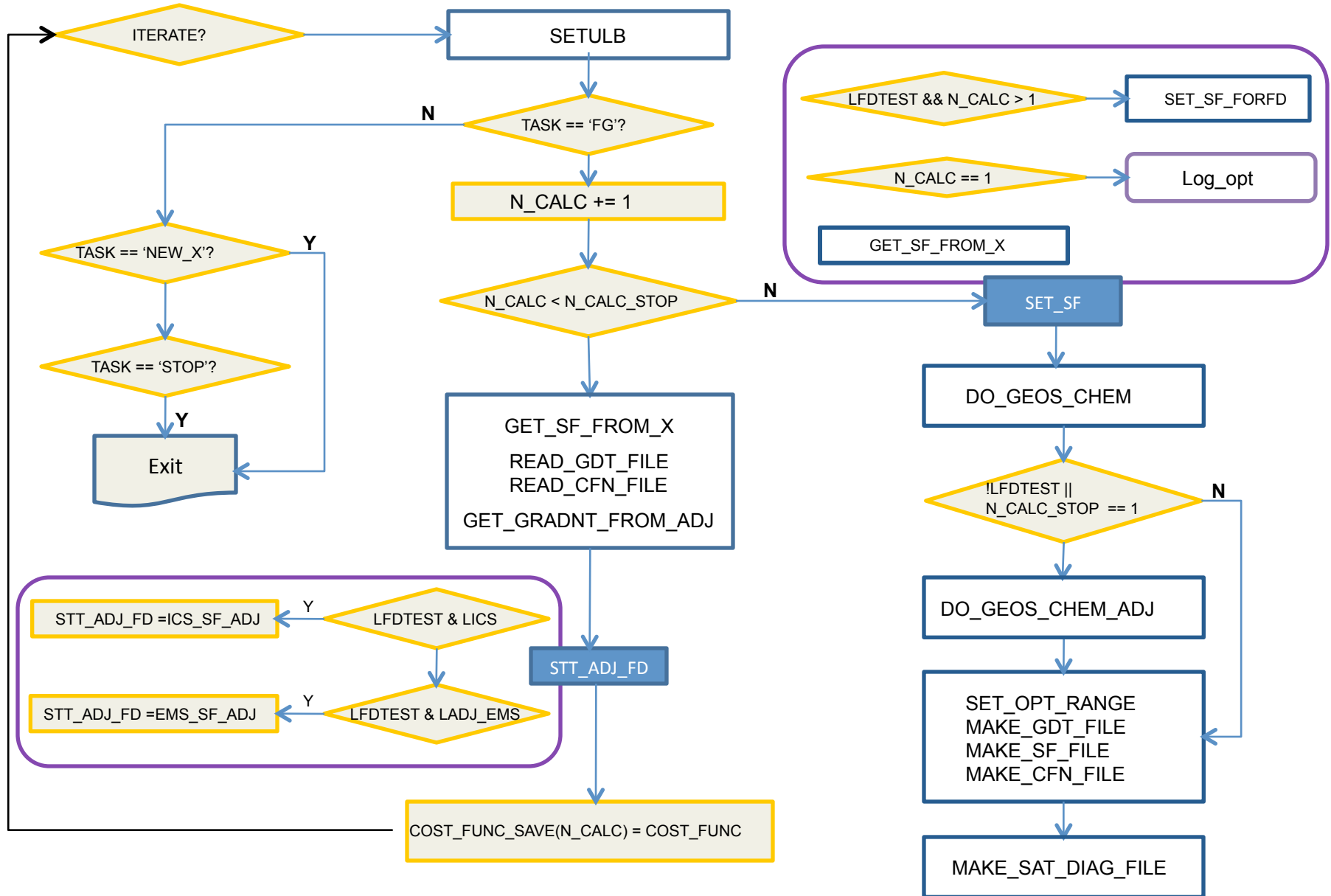
Meemong Lee

Shape & Color convention

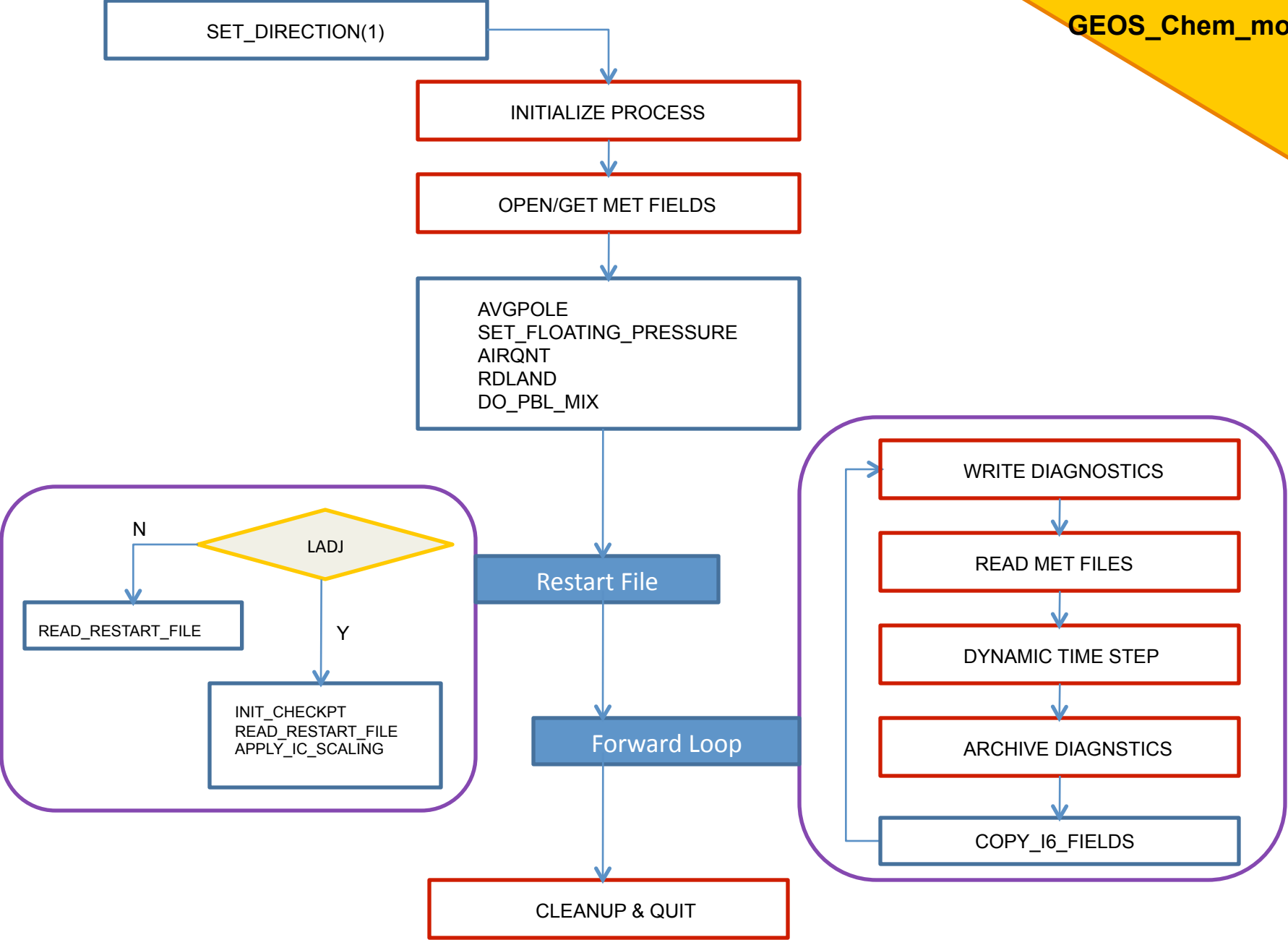
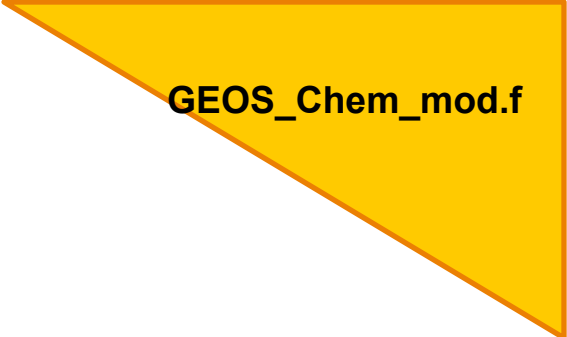




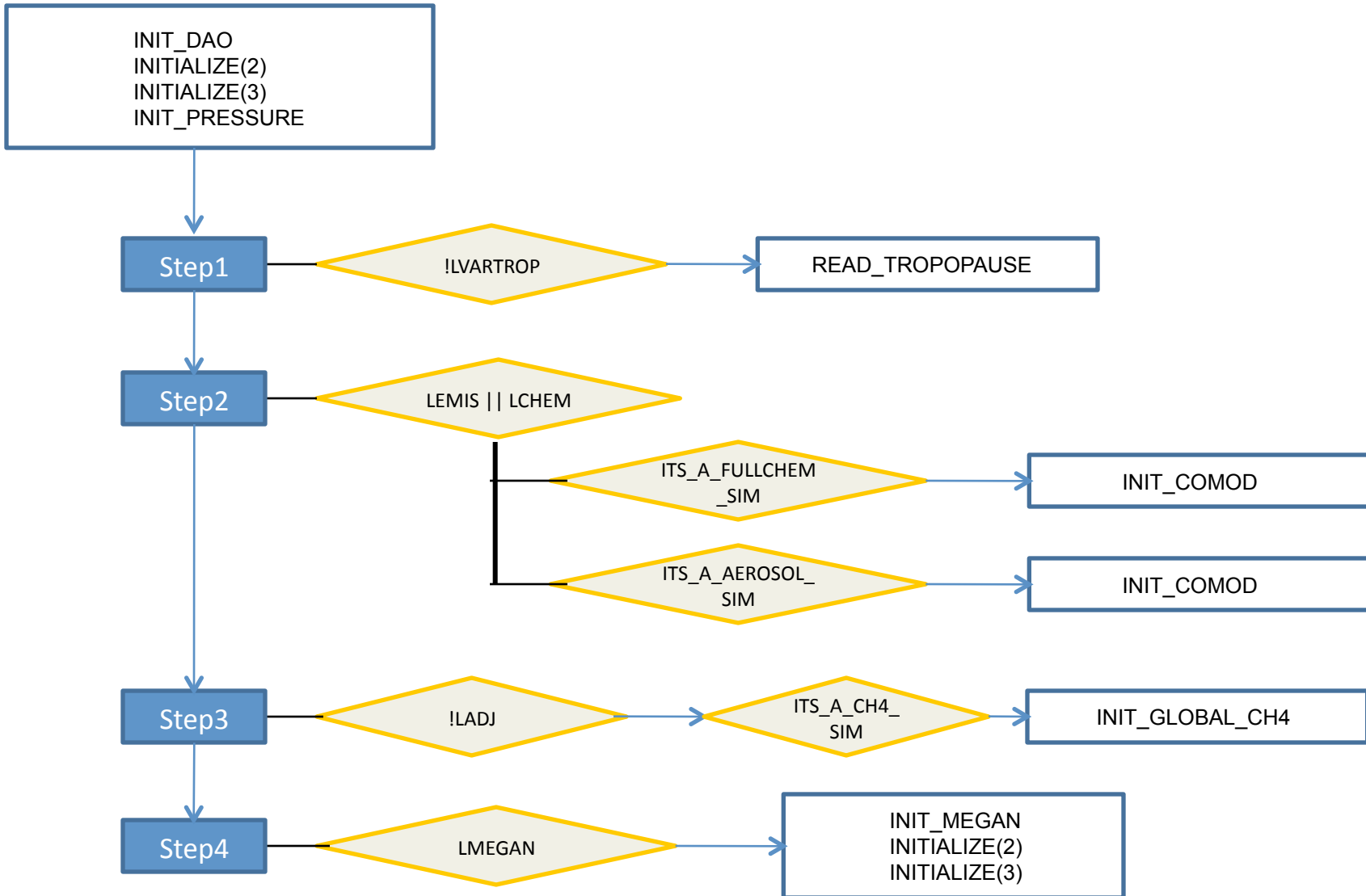
Optimization Loop



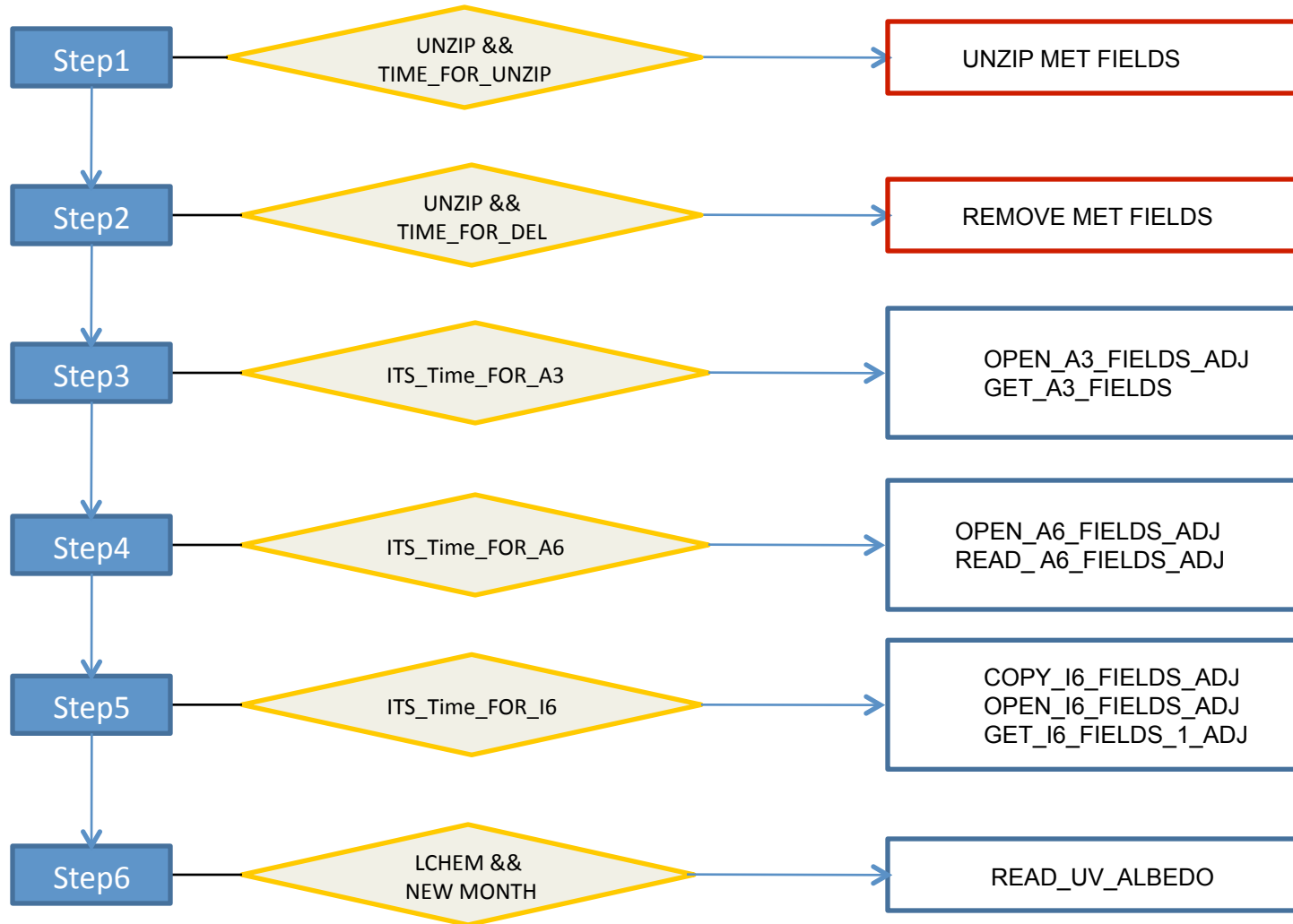
DO_GEOS_CHEM



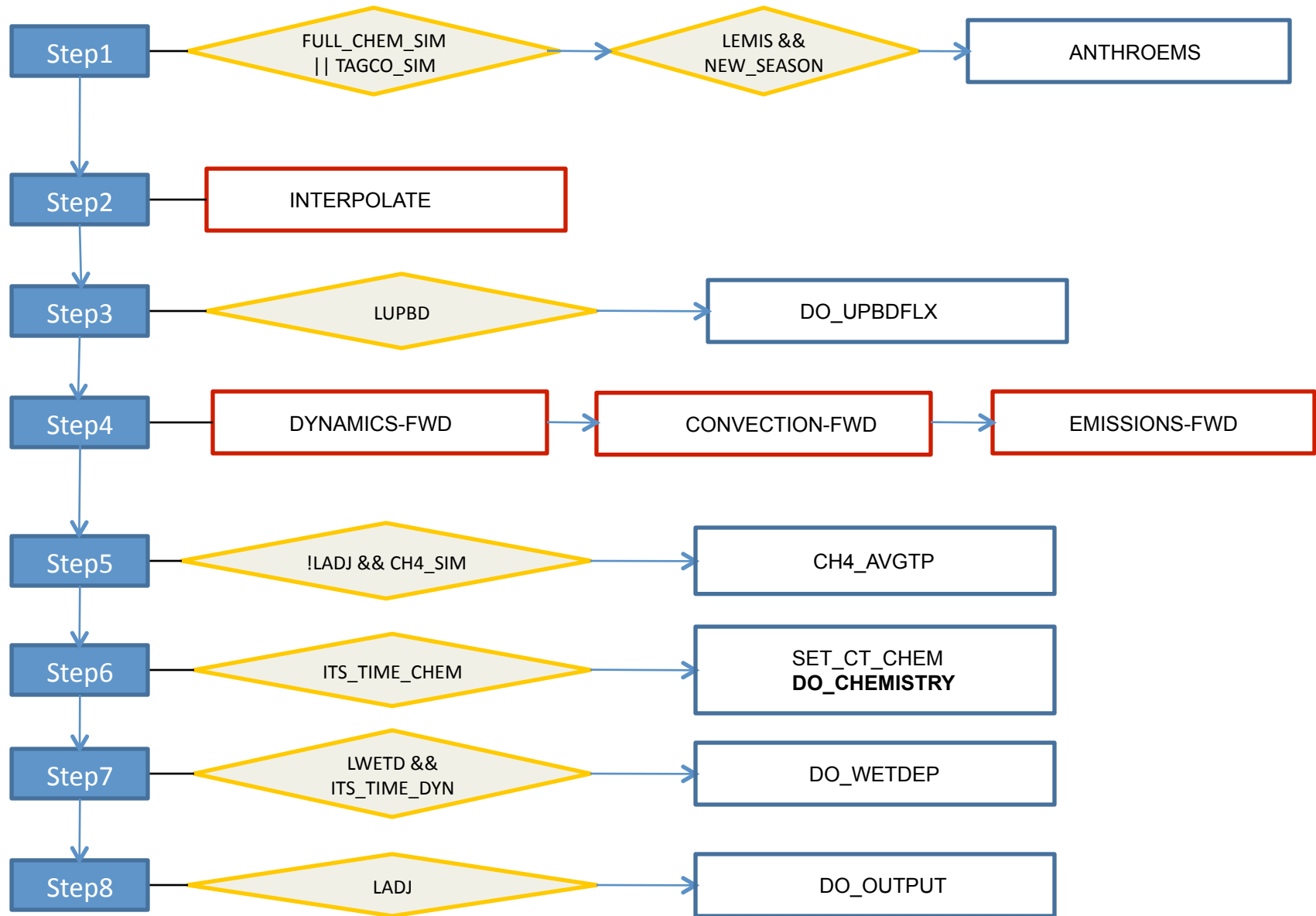
INITIALIZE PROCESS



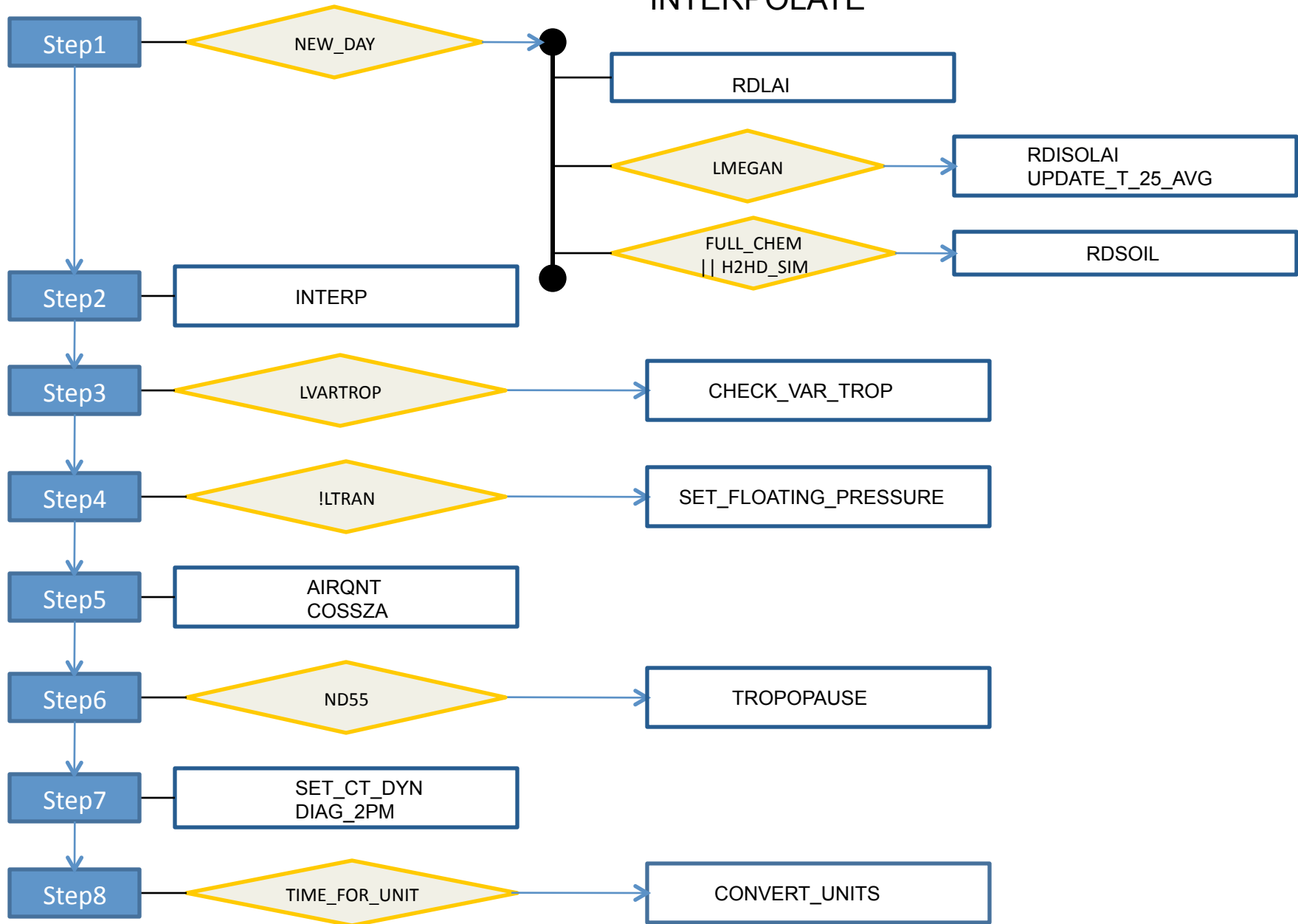
READ MET FIELDS



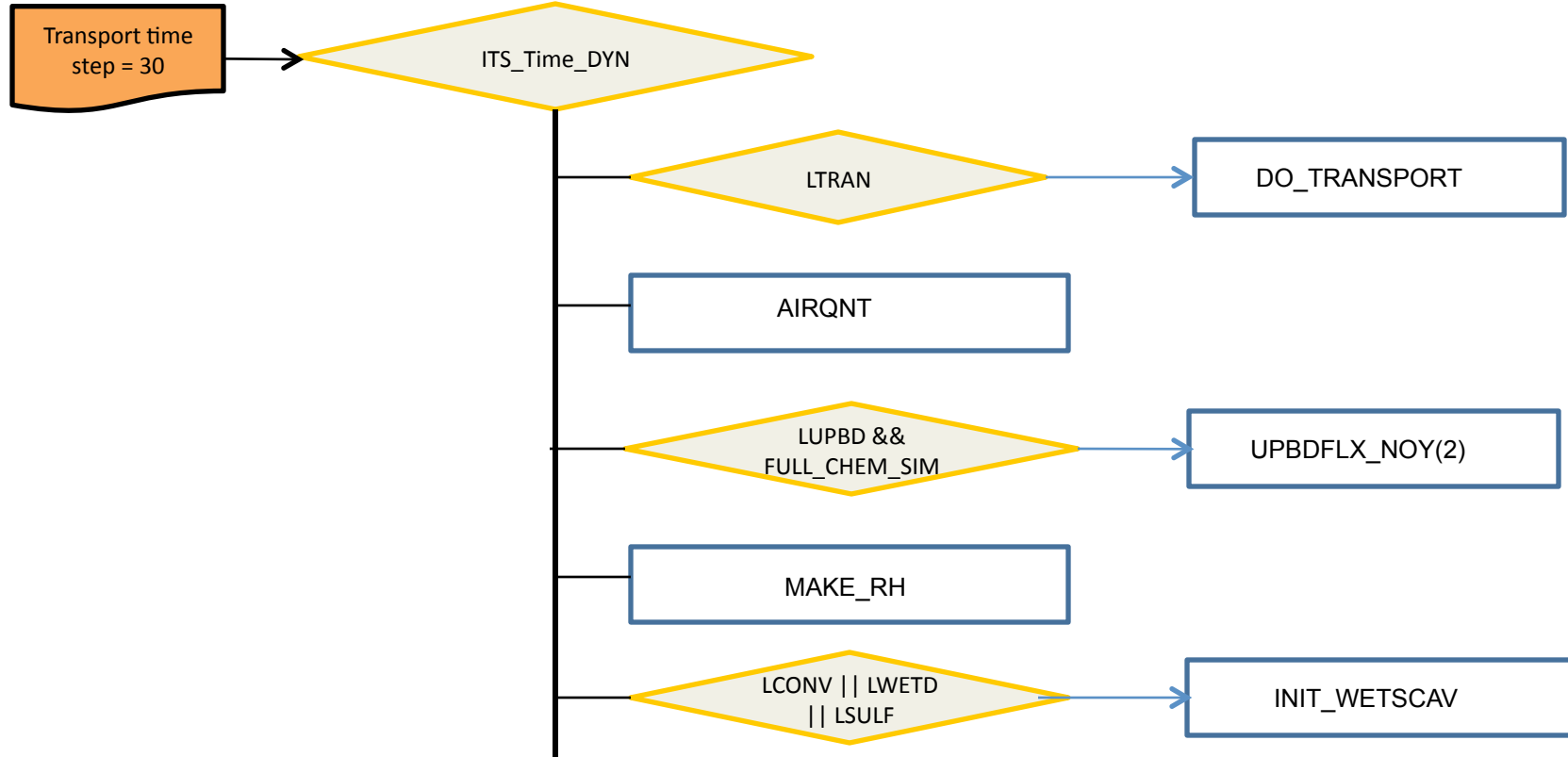
DYNAMIC TIMESTEP



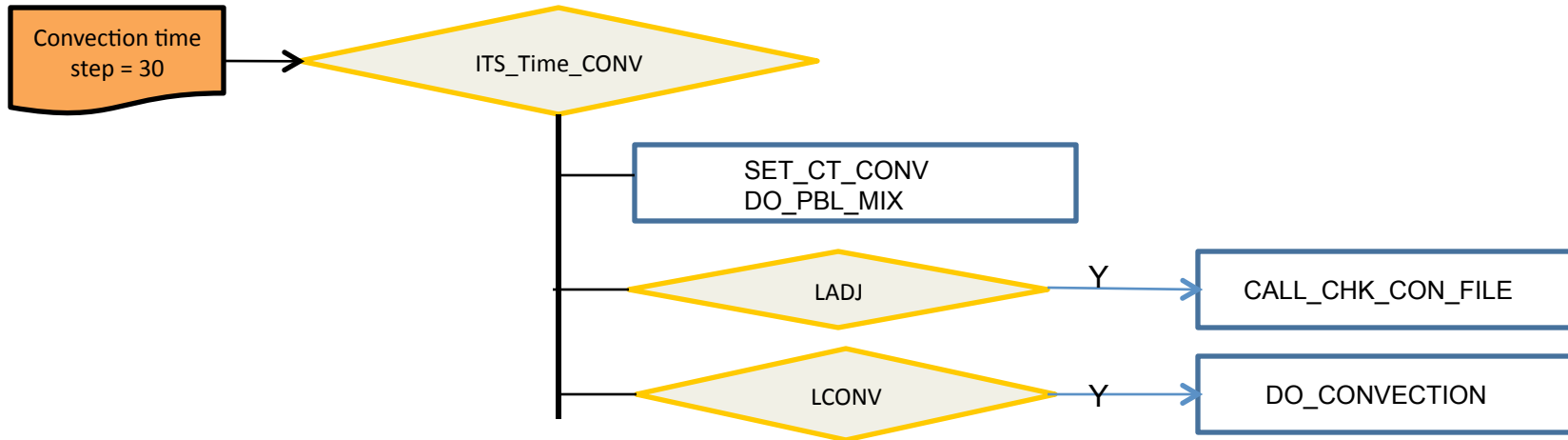
INTERPOLATE



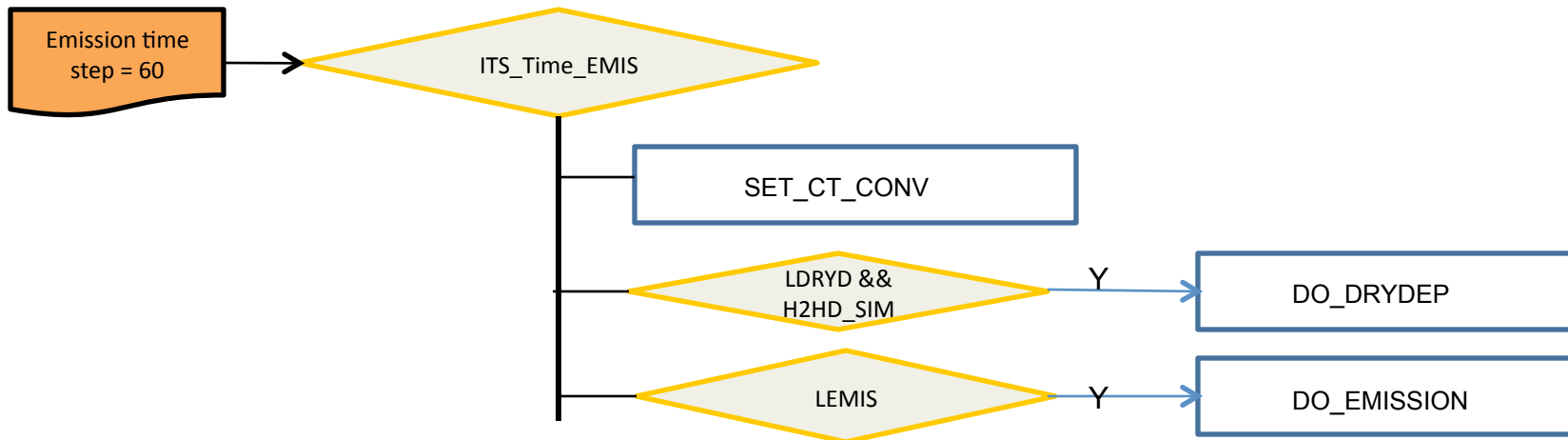
DYNAMICS-FWD



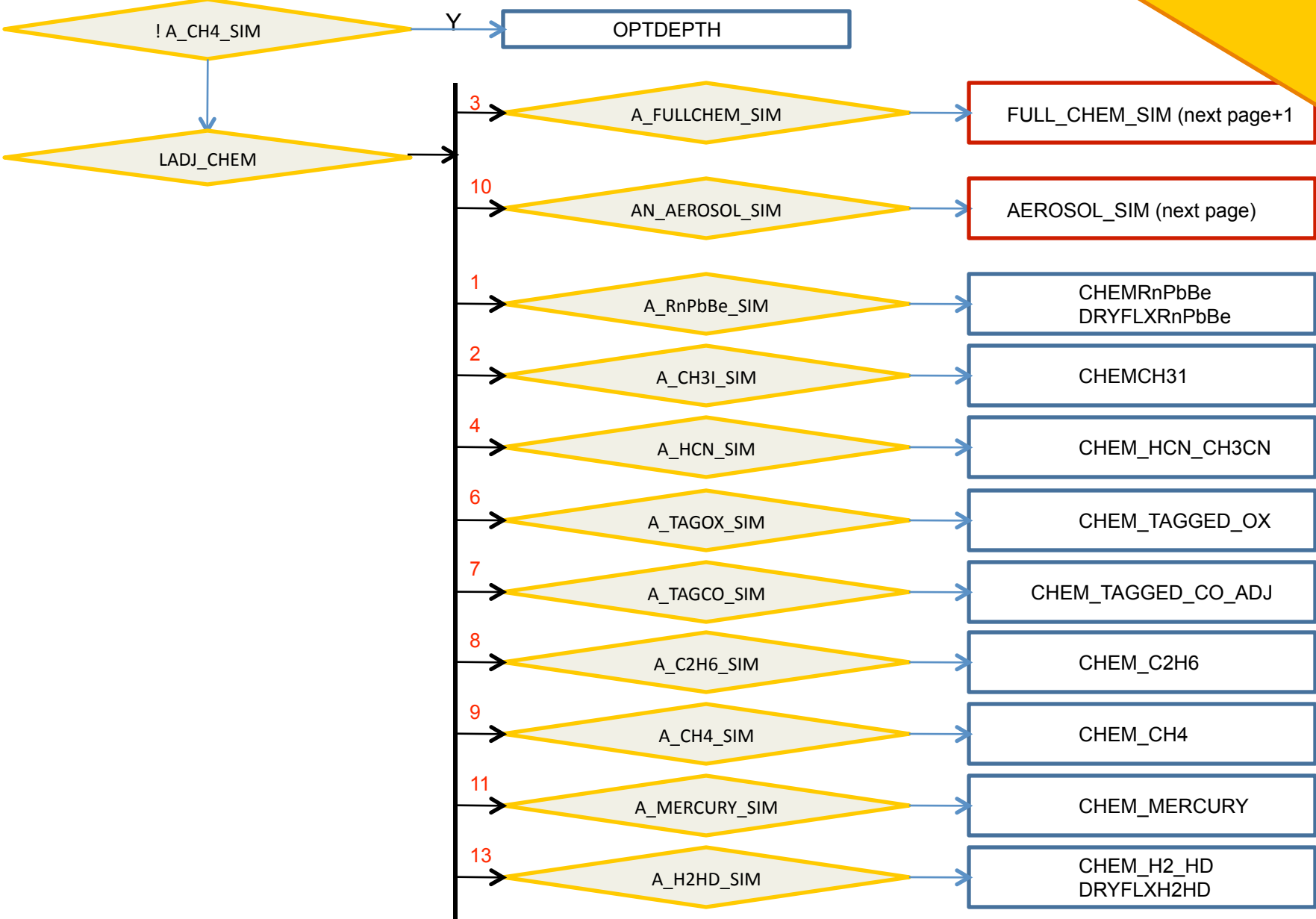
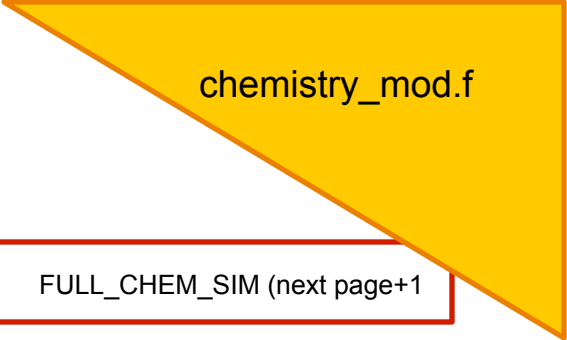
CONVECTION-FWD



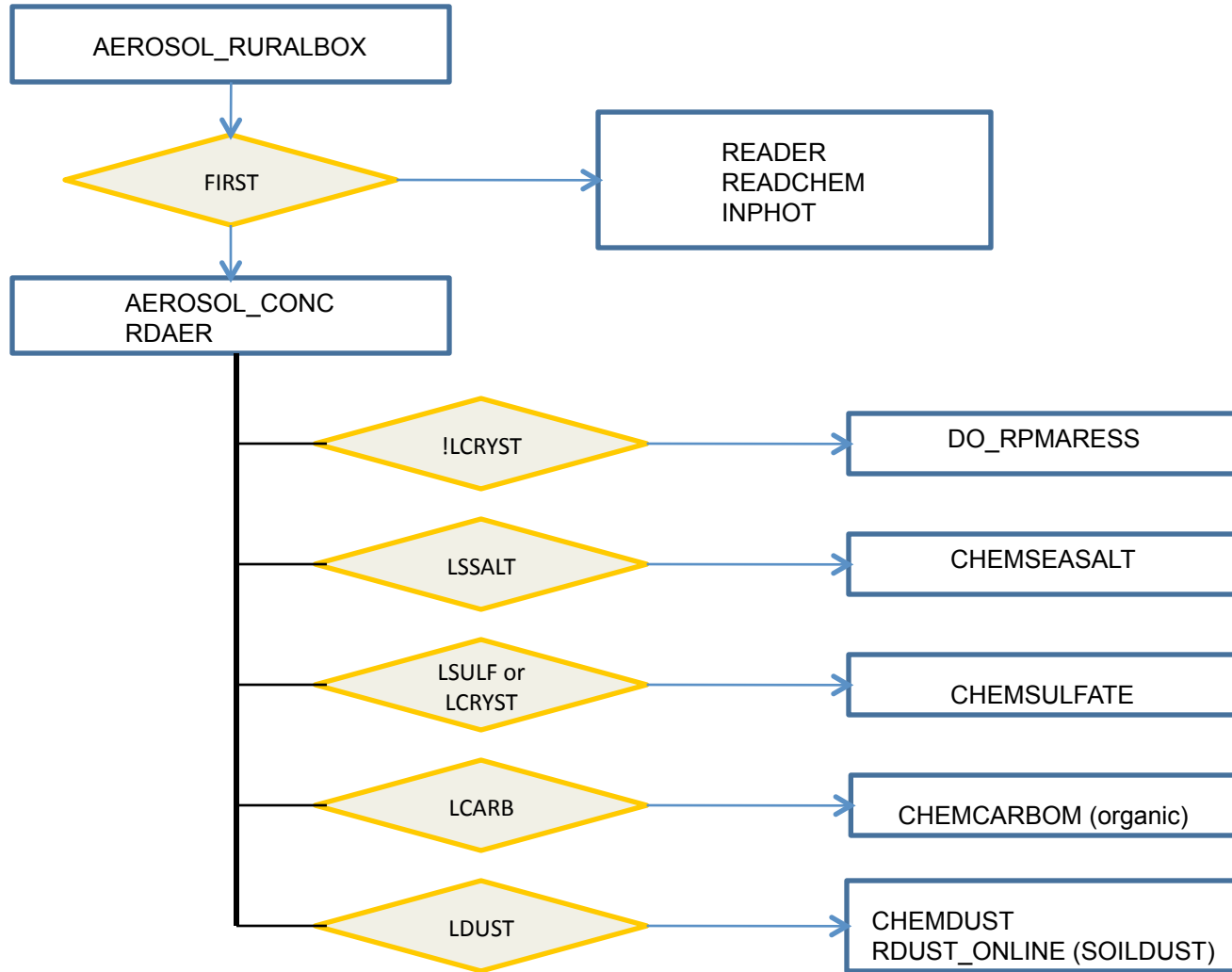
EMISSION-FWD



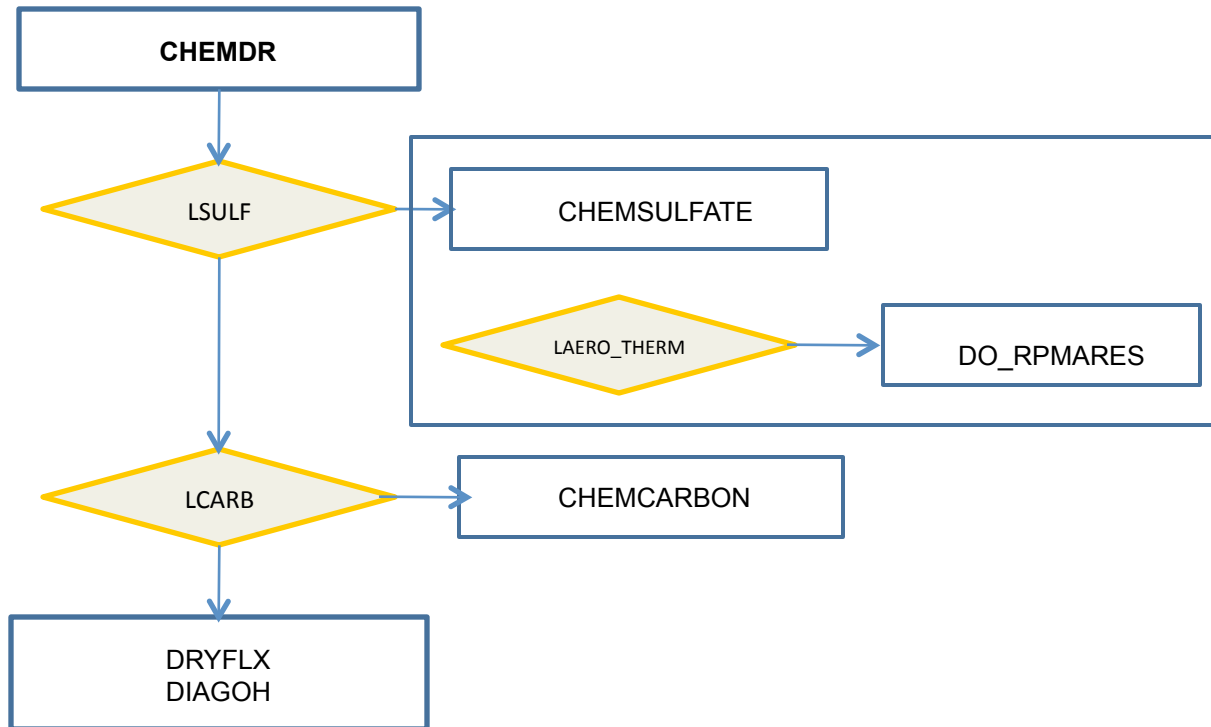
DO_CHEMISTRY

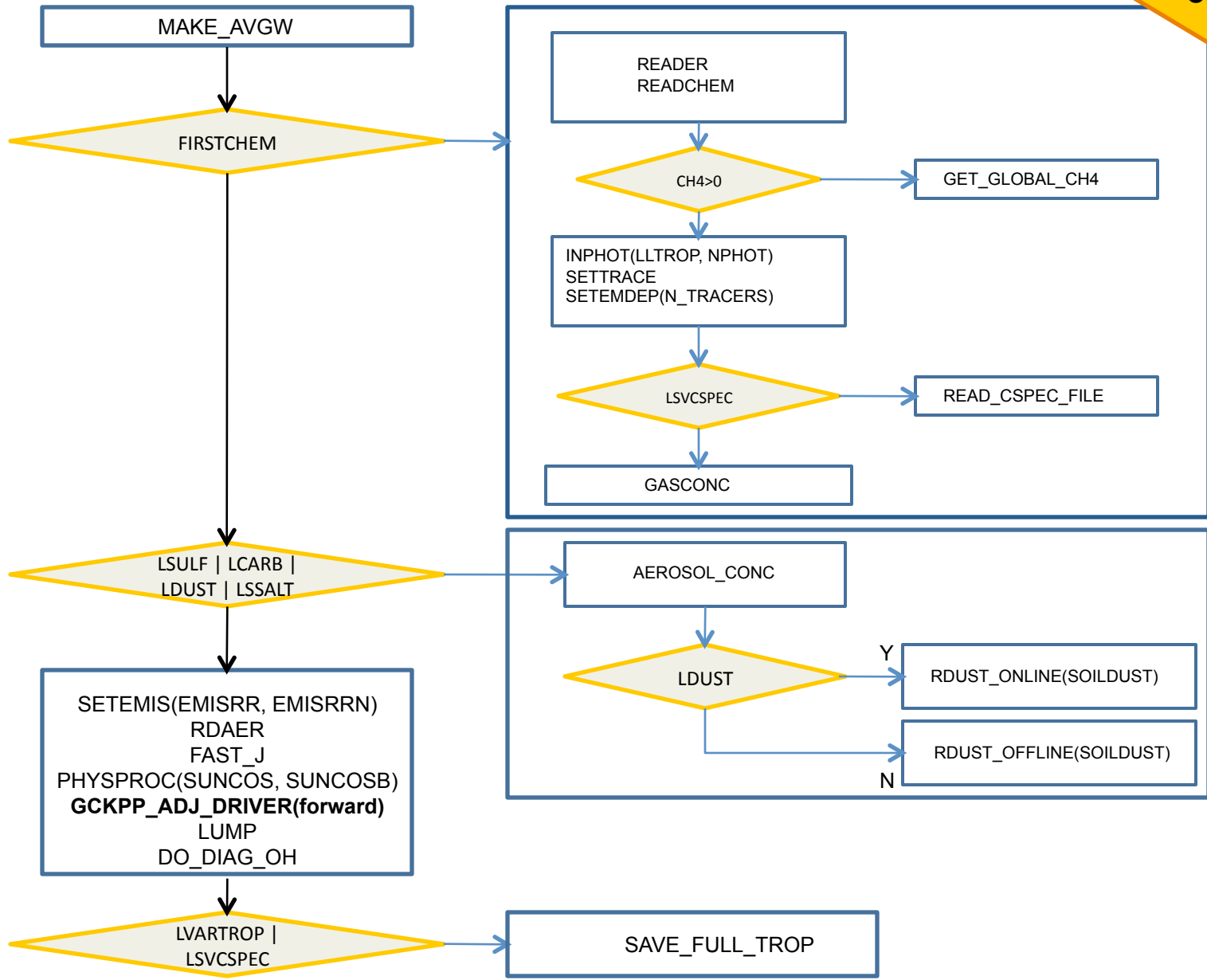


AEROSOL_SIM



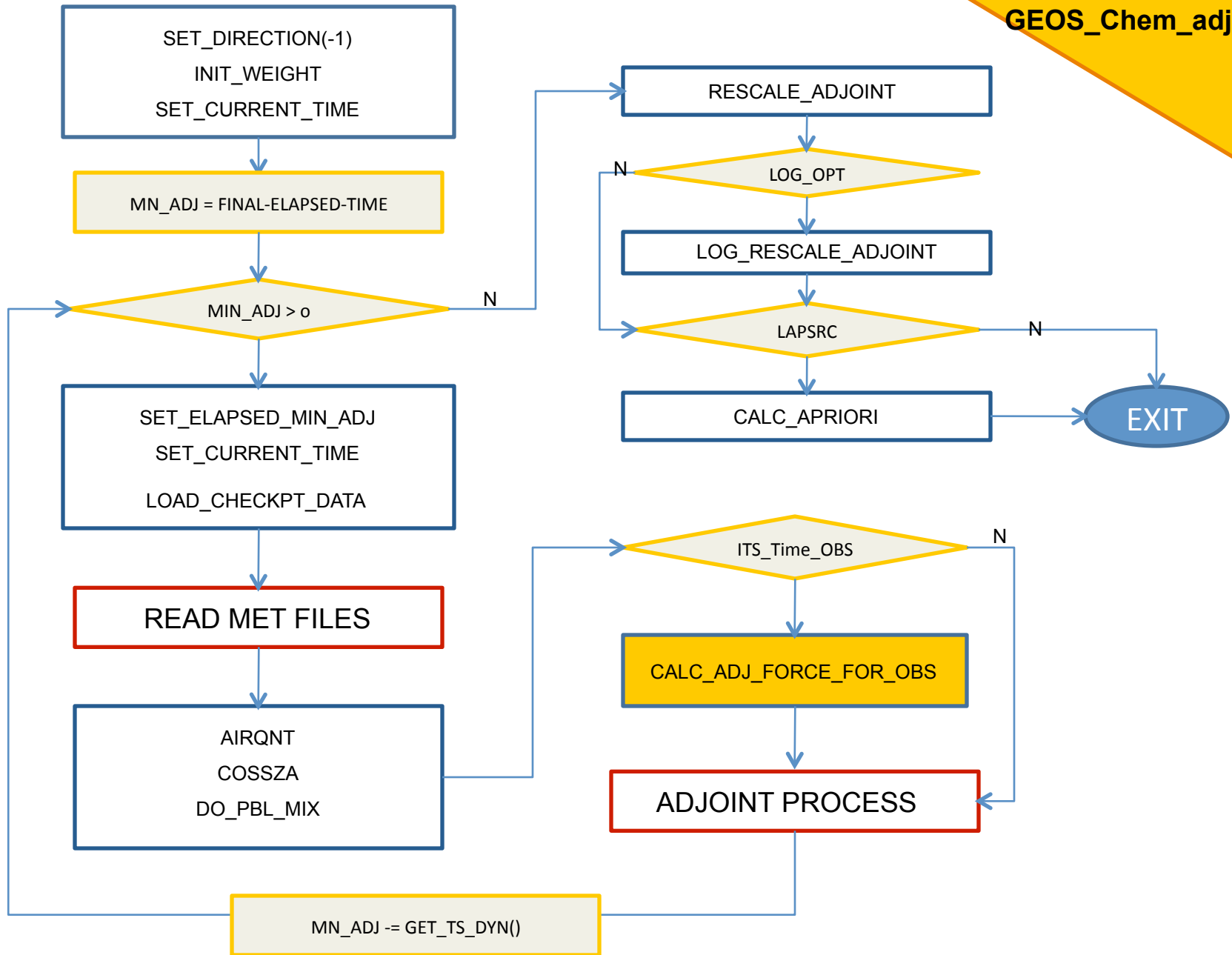
FULL_CHEM_SIM



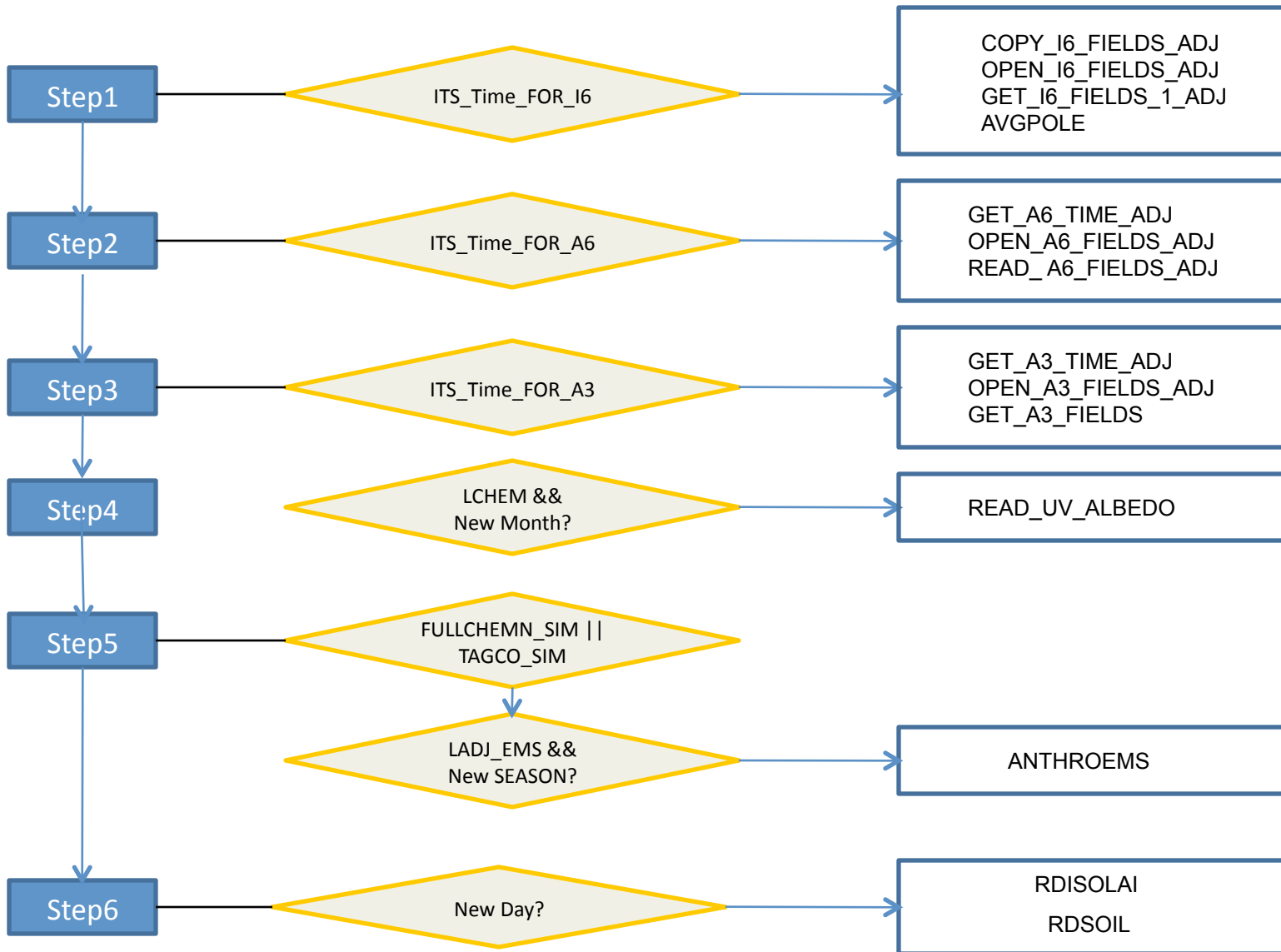


GEOS_CHEM_ADJ

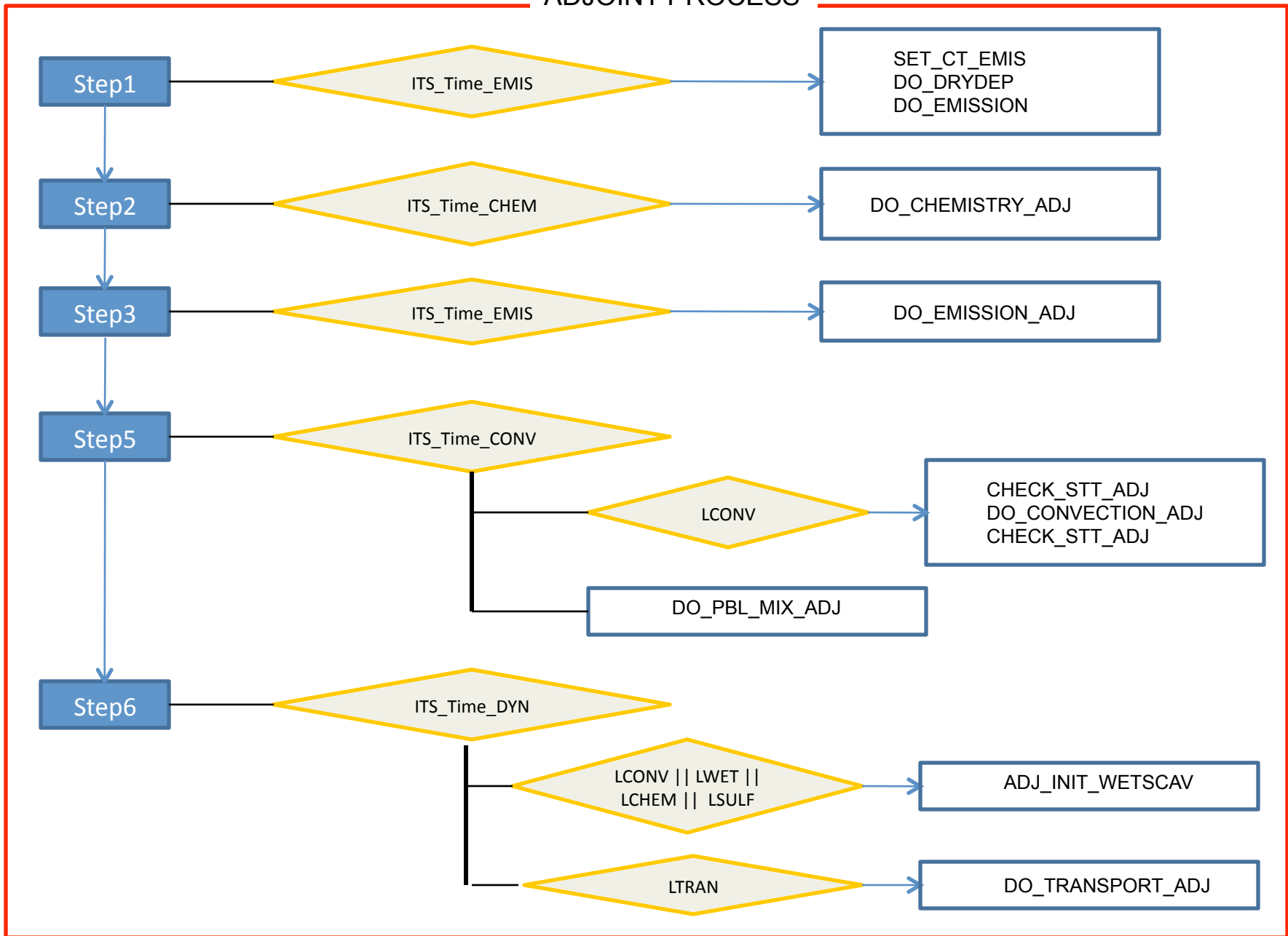
GEOS_Chem_adj.f

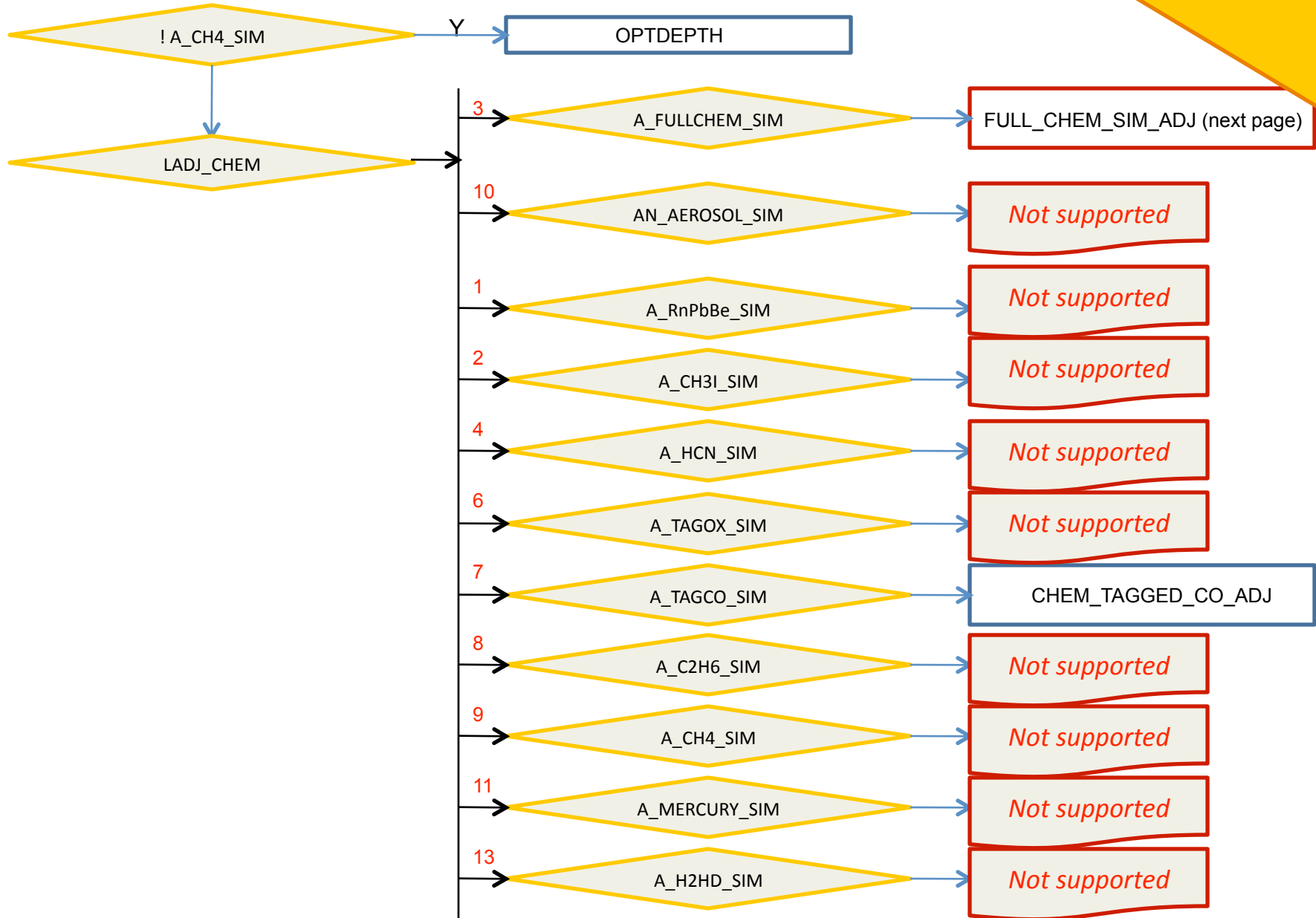


READ MET FILES

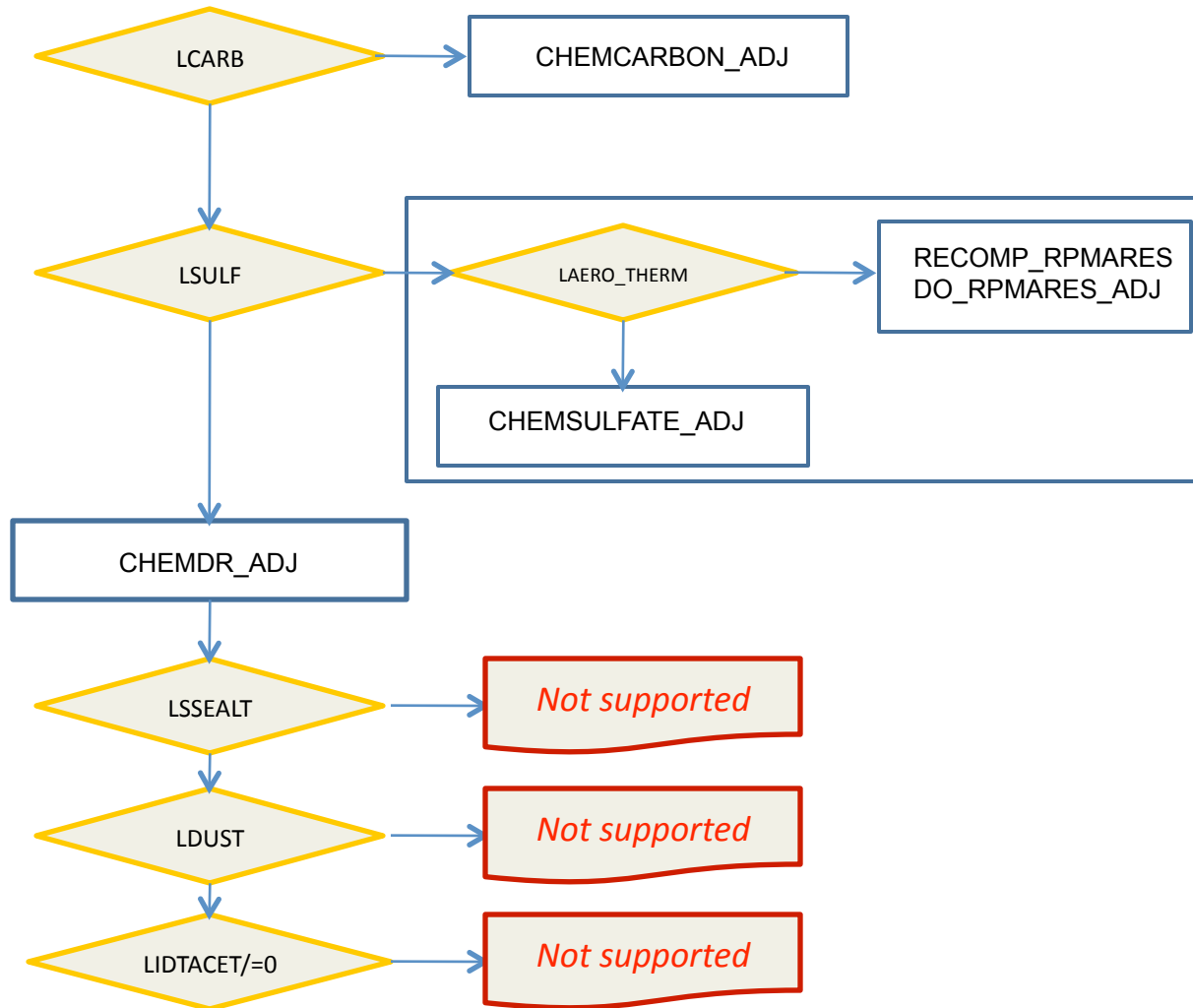


ADJOINT PROCESS

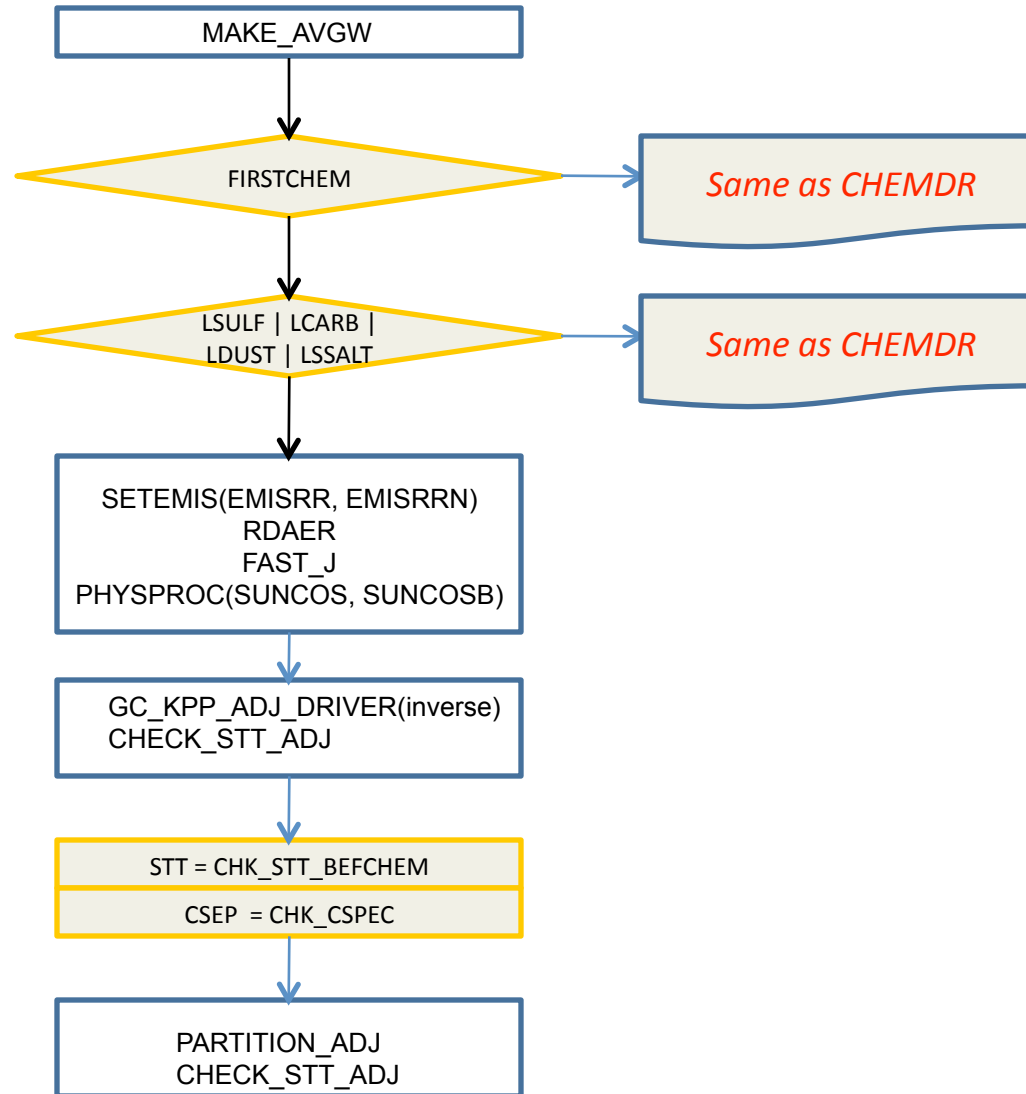




FULL_CHEM_SIM_ADJ



chemdr_adj.f



CALC_ADJ_FORCE_FOR_OBS

TES NH3 case

CALC_ADJ_FORCE_FOR_OBS- 1/4

! Calculate the interpolation weight matrix

```
MAP(1:LLPAR,1:LLNT) = GET_INTMAP( LLPAR, GC_PRES(:), GC_PSURF, LLNT, TES(NT)%PRES(1:LLNT), GC_PSURF )
```

! Get NH3 values at native model resolution

```
GC_NH3_NATIVE(:) = CHK_STT(I,J,,:IDTNH3)
```

! Convert from kg/box to ppm

```
GC_NH3_NATIVE(:) = GC_NH3_NATIVE(:) * TCVV(IDTNH3) / AD(I,J,:) * 1d6
```

```
NH3_SAVE(:,NT) = GC_NH3_NATIVE(:)
```

! Get NH3 values from doubled emissions run [ppmv]

```
GC_NH3_NATIVE_DBL(:) = GET_DOUBLED_NH3( GET_NYMD(), GET_NHMS(), REAL(TES(NT)%LON,4), REAL(TES(NT)%LAT,4))
```

! Interpolate GC NH3 column to TES grid

```
DO LL = 1, LLNT
```

```
GC_NH3(LL) = 0d0
```

```
DO L = 1, LLPAR
```

```
GC_NH3(LL) = GC_NH3(LL) + MAP(L,LL) * GC_NH3_NATIVE(L)
```

```
ENDDO
```

```
ENDDO
```

! Interpolate doubled GC NH3 column to TES grid

```
DO LL = 1, LLNT
```

```
GC_NH3_DBL(LL) = 0d0
```

```
DO L = 1, LLPAR
```

```
GC_NH3_DBL(LL) = GC_NH3_DBL(LL) + MAP(L,LL) * GC_NH3_NATIVE_DBL(L)
```

```
ENDDO
```

```
ENDDO
```

CALC_ADJ_FORCE_FOR_OBS- 2/4

! x_m - x_a

```
DO L = 1, LLNT
  GC_NH3(L) = MAX(GC_NH3(L), 1d-10)
  NH3_PERT(L) = LOG(GC_NH3(L)) - LOG(TES(NT)%PRIOR(L))
ENDDO
```

! x_a + A_k * (x_m - x_a)

```
DO L = 1, LLNT
  NH3_HAT(L) = 0d0
  DO LL = 1, LLNT
    NH3_HAT(L) = NH3_HAT(L) + TES(NT)%AVG_KERNEL(L,LL) * NH3_PERT(LL)
  ENDDO
  NH3_HAT(L) = NH3_HAT(L) + LOG(TES(NT)%PRIOR(L))
ENDDO
```

! x_m - x_a for doubled

```
DO L = 1, LLNT
  GC_NH3_DBL(L) = MAX(GC_NH3_DBL(L), 1d-10)
  NH3_PERT_DBL(L) = LOG(GC_NH3_DBL(L)) - LOG(TES(NT)%PRIOR(L))
ENDDO
```

! x_a + A_k * (x_m - x_a)

```
DO L = 1, LLNT
  NH3_HAT_DBL(L) = 0d0
  DO LL = 1, LLNT
    NH3_HAT_DBL(L) = NH3_HAT_DBL(L) + TES(NT)%AVG_KERNEL(L,LL) * NH3_PERT_DBL(LL)
  ENDDO
  NH3_HAT_DBL(L) = NH3_HAT_DBL(L) + LOG(TES(NT)%PRIOR(L))
ENDDO
```

CALC_ADJ_FORCE_FOR_OBS- 3/4

! Calculate difference between modeled and observed profile

```
DO L = 1, LLNT
  IF ( TES(NT)%NH3(L) > 0d0 ) THEN
    DIFF(L) = NH3_HAT(L) - LOG( TES(NT)%NH3(L) )
  ELSE
    DIFF(L) = 0d0
  ENDIF
ENDDO
```

! Calculate $1/2 * DIFF^2 * S_{\{obs\}}^{-1} * DIFF$

```
DO L = 1, LLNT
  FORCE(L) = 0d0
  DO LL = 1, LLNT
    FORCE(L) = FORCE(L) + TES(NT)%OER_INV(L,LL) * DIFF(LL)
  ENDDO
  NEW_COST(NT) = NEW_COST(NT) + 0.5d0 * DIFF(L) * FORCE(L)
ENDDO
```

ADJ_DIFF = FORCE

! Adjoint of difference

```
DO L = 1, LLNT
  IF ( TES(NT)%NH3(L) > 0d0 ) THEN
    ADJ_NH3_HAT(L) = ADJ_DIFF(L)
  ENDIF
ENDDO

DO L = 1, LLNT
  ADJ_NH3_PERT(L) = 0d0
  DO LL = 1, LLNT
    ADJ_NH3_PERT(L) = ADJ_NH3_PERT(L) + TES(NT)%AVG_KERNEL(LL,L) * ADJ_NH3_HAT(LL)
  ENDDO
ENDDO
```

CALC_ADJ_FORCE_FOR_OBS- 4/4

! Adjoint of $x_m - x_a$

```
DO L = 1, LLNT
  IF ( GC_NH3(L) > 1d-10 ) THEN
    ADJ_GC_NH3(L) = 1d0 / GC_NH3(L) * ADJ_NH3_PERT(L)
  ELSE
    ADJ_GC_NH3(L) = 0d0
  ENDIF
ENDDO
```

! Adjoint of interpolation

```
DO L = 1, LLPAR
  ADJ_GC_NH3_NATIVE(L) = 0d0
  DO LL = 1, LLNT
    ADJ_GC_NH3_NATIVE(L) = ADJ_GC_NH3_NATIVE(L) + MAP(L,LL) * ADJ_GC_NH3(LL)
  ENDDO
ENDDO
```

! Adjoint of unit conversion

```
ADJ_GC_NH3_NATIVE(:) = ADJ_GC_NH3_NATIVE(:) * TCVV(IDTNH3) / AD(I,J,:) * 1d6
! Pass adjoint back to adjoint tracer array
STT_ADJ(I,J,:,IDTNH3) = STT_ADJ(I,J,:,IDTNH3) + ADJ_GC_NH3_NATIVE(:)
```

!AD = (dry) mass of air in grid box (I,J,L) in kg,

!TCVV = Array containing [Air MW / Tracer MW] for tracers

! Update cost function

```
COST_FUNC = COST_FUNC + SUM(NEW_COST(NTSTOP:NTSTART))
```