Implementing the NRLMSISe00 and NRLMSISe2.0 Models into the GEODYN II Program on the CCMC-AWS Servers

Overview

- This document details the changes that were made to include the NRLMSISe00 and NRLMSISe2.0 models into GEODYN II.
- The majority of the changes take place in the IIE/MODS folders, but some changes were required in IIE/ORIG to keep the MSISe86 subroutines from overriding the new models.
- We construct a different version of GEODYN for each MSIS model. This is done by creating an independent IIE/MODS folder for each model. This is much simpler than trying to push things through IIS.
- Each MSIS model is called using the "ATMDEN 86" switch. The IIE executable in the respective MODs folders then determines which MSIS model is being used.

Contents

A MSIS 86

Steps

A.1 Modify the MSISe86 subroutines in IIE/ORIG

- Since the new MSIS versions have similar (or identical) function names as those in MSIS86, we needed to modify these duplicate routines. We did this by changing the names of the MSIS86 subroutines and their internal calls to have an "86" suffix (i.e. `MSIS.f90` becomes `MSIS86.f90`)
- The following subroutine names (and calls) were changed:
  - `DNET.f90`
  - `TSELEC.f90`
  - `CCOR.f90`
  - `MSIS.f90`
  - There has to be an `MSIS.f90` (and `DRAG.f90`) without the 86 suffix in the ORIG folder. I added a print statement to write to Unit 6 to double check that these are NOT being used when MSIS 2 is being called
  - Be sure all internal function names are also changed
  - Subroutines that were NOT changed:
    * `GLOBE5.f90` - not called in future MSIS versions
    * `SPLINE.f90` - this is used elsewhere by GEODYN, so instead we changed the SPLINE name in the other versions of MSIS.

B MSIS 00

B.1 Preparation Steps

- MSISe00 source Code:
  - Needed an unmodified, F90 version of the MSISe00 code. The CCMC version had been modified to suit their browser system and was in F77
  - https://github.com/graziano-giuliani/Meteostuff/tree/master/NRLMSIS_F90
- Construct a separate IIE/MODS folder for the MSISe00 run of GEODYN:
  - `IIE/MODS_msis00_f90`
- Place the following .F90 source files into the `MODS_msis00_f90` directory
  - `utils_constants.F90`
  - `utils_spline.F90`
  - `physics_constants.F90`
  - `physics_msis.F90`

B.2 Modifications for MSIS 00

B.2.1 Modifications to DRAG.f90

- Separated the reading of DTM87, MSIS86, and JAACHIA71 into distinct `IF THEN` statements
- Added checks to print to `UNIT6` (IIEOUT) that the correct model is being used
- Added a write statement to output density data to `Unit 99` :
  - `'FSSTRT','IYMD','IHMS','XLATD','XLOND','ALTI',`

    `'RHO','DRHODZ', 'X','Y','Z','XDOT','YDOT','ZDOT'`

B.2.2 Modifications to MSIS.f90

- Include the msis00 modules at start of code
  - Line 55: `use utils_constants`
  - Line 56: `use physics_msis`
- Fix Dimensions
  - Line 66: Change the dimension of the `DEN` array from 8 to 9
- Fix Data types

  - Line 69: `INTEGER(4) ::  IYYDDD`
  - Line 70: `INTEGER(4) ::  mass`

- MSIS option settings
  - Call `tselec(SWI)` instead of MSIS86's `TSELEC(SWI)`

    Line 103: `CALL tselec(SWI)`

- Call to MSIS subroutine
  - Mass should be put into a variable so that it can be converted to INT(4)

    Line 179: `mass=48`
  - We must call `gtd7d` instead of `gtd7`. This will include anomalous oxygen above 500 km which is important for satellite drag above that altitude.

    `CALL gtd7d(IYYDDD,UTSEC,ALTKM,GLAT,GLON,STLOC,` ...

    ... `AVGFLX,FLUX,AP,mass,DEN,TEMP)`

B.2.3 Mods to the MSIS00 Compile Script

- The most important thing to do in the compile step is to remove the old MSIS.f90 files that are being linked from `ORIG/` that link back to MSISe86.
- MSISe00's specific compile order is listed in the attached script
- Modifications to SUMRY.f90: Fixed a write statement to Unit 6 to reflect that MSISe00 is being used instead of MSIS86

```
chmod a+w *mod *.o
rm *mod *.o
cp ../ORIG/*mod .
chmod a+w *mod *.o
ln -s ../ORIG/*.o .
#rm MSIS.o
rm MSIS86.o
rm MSIS.mod
rm MSIS86.mod

gfortran -c -O2 utils_constants.F90 utils_spline.F90 physics_constants.F90 physics_msis.F90>err1 2> err2

gfortran -c -std=legacy -fdefault-integer-8 -fcray-pointer -O2 *.f90>err3 2>err4

#ln -s ../ORIG/*.o .
gfortran *.o -o giie2002_gfortran -llapack -lblas
#rm *mod *.o
rm *.o
```

Listing 1: MSIS00 Compile Script

C MSIS 2

C.1 Preparation Steps

- MSIS 2.0 source code:

  `https://map.nrl.navy.mil/map/pub/nrl/NRLMSIS/NRLMSIS2.0/`

- Construct a separate IIE/MODS folder for the MSIS 2.0 run of GEODYN:
  - `IIE/MODS_msis2`
- Place the following .F90 source files into the `MODS_msis2` directory
  - `alt2gph.F90`
  - `msis_calc.F90`
  - `msis_constants.F90`
  - `msis_dfn.F90`
  - `msis_gfn.F90`
  - `msis_gtd8d.F90`
  - `msis_init.F90`
  - `msis_tfn.F90`
  - `msis20.parm`

C.2 Modifications for MSIS 2

C.2.1 Mods to DRAG.f90

- Separated the reading of DTM87, MSIS86, and JAACHIA71 into distinct `IF THEN` statements
- Added checks to print to `UNIT6` (IIEOUT) that the correct model is being used
- Added a write statement to output density data to `Unit 99` :
  - `'FSSTRT','IYMD','IHMS','XLATD','XLOND','ALTI',`

    `'RHO','DRHODZ', 'X','Y','Z','XDOT','YDOT','ZDOT'`

C.2.2 Mods to MSIS.f90

- Include the msis2 modules at start of code
  - Line 55 `use msis_init, only : msisinit`
- Dimensions

  - Line 64: Change the dimension of the DEN array from 8 to 9
  - Line 64: Add SWI here

    `DIMENSION AP(7),SWI(25),DEN(9),TEMP(2)`
  - Remove SWI from other dimension call in line 69
- Data Types
  - `INTEGER(4) :: IYYDDD`
  - `INTEGER(4) :: mass`
  - `REAL(4) :: SWI`
  - `INTEGER(4) :: iiun`
- MSIS2 Option Setting
  - Msis2 must be initialized. This replaces the TSELEC function from before with the switches stored in SWI being part of the input to the init function.
  - Line 96: `SWI=1`
  - Line 119: `iiun=117`
  - Line 120: Comment out TSELEC
  - Line 123:

    `CALL msisinit(parmpath='/data/geodyn_code/IIE/MODS_msis2/',`
    ... `parmfile='msis20.parm',switch_legacy=SWI )`
  - I had lots of issues with the parmfile. Be sure you are giving the right name for the parm file, and it seems to require the absolute path.
- Call to MSIS2's subroutine
  - Mass should be put into a variable so that it can be converted to proper data type

    Line 204: mass=48
  - The `gtd8d` function is a legacy wrapper for `msis_calc()` . This allows us to call MSIS with the same input parameters as in previous versions.

    `CALL CALL gtd8d(IYYDDD,UTSEC,ALTKM,` ...

    ... `GLAT,GLON,STLOC,AVGFLX,FLUX,AP, mass,DEN,TEMP)`

C.2.3 Mods to msis_gtd8d.F90

I had to change the datatype from real(8) to real(4) for many of the legacy inputs.

```
! MSIS Legacy subroutine arguments
 integer, intent(in)        :: iyd
 real(4), intent(in)        :: sec
 real(4), intent(in)        :: alt
 real(4), intent(in)        :: glat
 real(4), intent(in)        :: glong
 real(4), intent(in)        :: stl
 real(4), intent(in)        :: f107a
 real(4), intent(in)        :: f107
 real(4), intent(in)        :: ap(7)
 integer, intent(in)        :: mass
 real(4), intent(out)       :: d(9), t(2)
```

C.2.4 Mods to the MSIS 2 Compile Script

- Remove all linked instances of MSISe86 from the ORIG folder
- Msis2 must be compiled in a special order, and BEFORE the other geodyn subroutines that will be using it.

```
#
#### Script that Compiles modified subroutines and makes an IIE executable
#
#
chmod a+w *mod *.o
rm *mod *.o
cp ../ORIG/*mod .
chmod a+w *mod *.o
ln -s ../ORIG/*.o .

rm MSIS.o
rm MSIS86.o
rm MSIS.mod
rm MSIS86.mod

gfortran -c -O2 alt2gph.F90 msis_constants.F90 msis_init.F90 msis_gfn.F90
    msis_tfn.F90 msis_dfn.F90 msis_calc.F90 msis_gtd8d.F90>err3 2> err4
gfortran -c -std=legacy -fdefault-integer-8 -fcray-pointer -O2 *.f90>err
    2>err2

#ln -s ../ORIG/*.o .
gfortran *.o -o giie2002_gfortran -llapack -lblas
#rm *mod *.o
rm *.o
```

Listing 2: MSIS2 Compile Script

D The Calculation of DRHODZ (coming soon)

I will be updating this calculation soon.

The equation for computing $d\rho/dz$ (DRHODZ) in GEODYN is as follows:

```
!  Calculate drho/dz.
      IF(IDRV .NE. 0) THEN
!
!   D       O            D(1) - HE NUMBER DENSITY(CM-3)
!                        D(2) - O NUMBER DENSITY(CM-3)
!                        D(3) - N2 NUMBER DENSITY(CM-3)
!                        D(4) - O2 NUMBER DENSITY(CM-3)
!                        D(5) - AR NUMBER DENSITY(CM-3)
!                        D(6) - TOTAL MASS DENSITY(GM/CM3)
!                        D(7) - H NUMBER DENSITY(CM-3)
!                        D(8) - N NUMBER DENSITY(CM-3)
!   T                    T(1) - EXOSPHERIC TEMPERATURE
!                        T(2) - TEMPERATURE AT ALT
!
        TERM1 = -1.66D-24*(16.D0*DEN(1) + 256.D0*DEN(2) + 784.D0*DEN(3) &
     &           + DEN(7) + 196.D0*DEN(8) )
        TERM2 = GSURF/(TEMP(2)*RGAS)/(1.D0 + ZL/RE)**2
        TERM3 = ((RE+ZL)/(RE+ALTKM))**2
        DRHODZ = TERM1*TERM2*TERM3
!       DRHODZ IS NOW IN G/CC/KM.  THIS IS DIMENSIONALLY EQUAL TO KG/M4.

      ENDIF
```

In more readable math, this is:

$$\frac{d\rho}{dz} = -m_p\Big((m_{He}^2 n_{He})+(m_O^2 n_O)+(m_{N2}^2 n_{N2})+(m_H^2 n_H)+(m_N^2 n_N)\Big)\frac{\left(\frac{g_{surf}}{T_{alt}R_{gas}}\right)}{(1+Z_L/R_E)^2}\left(\frac{R_E+Z_L}{R_E+Z_{alt}}\right)$$

Where:

- m_p is the mass of a proton ($= 1.66E^{-24}$ grams)
- m_{He} is atomic mass of helium ($= 4$ amu)
- m_O is atomic mass of oxygen ($= 16$ amu)
- m_{N2} is atomic mass of molecular nitrogen ($= 28$ amu)
- m_H is atomic mass of hydrogen ($= 1$ amu)
- m_N is atomic mass of atomic nitrogen ($= 14$ amu)
- g_{surf} is the gravity at Earth's surface
- T_{alt} is the temperature at the given altitude
- R_{gas} is the gass constant ($= 831.4$)

- Z_L is the lower boundary for diffusive equilibrium in the MSIS model ($= 120$)
- R_E is radius of earth
- Z_{alt} is the altitude of the satellite

A few things to note:

1. For completeness sake, we should include O_2
2. Anomalous oxygen should also be included in the calculation for the new versions of MSIS
3. If including AnomO, we have to do the calculation with a separate temperature from the ones assumed for the other gasses (4000 K). If including Anomalous oxygen then it should be computed separately with its own scale height given the anomalous temperature.
4. Geodyn's GSURF and ZL should be made consistent.