Space Call 3 - FP7-SPACE-2010-1
Grant Agreement Number 261948

ATMOP project has received
EC research funding

Advance Thermosphere Modelling for Orbit Prediction

ATMOP

Release of DTM 2013

SW Installation and User Manual

Code : ATMOP-DMS-RPT-021
Issue : 1.0
Date : 18/11/2013

	Name	Function	Organisation
Prepared by	Raúl Domínguez	Project Engineer	DMS
Reviewed by	Noelia Sánchez-Ortiz Sean Bruinsma	Project Coordinator WP4 leader	DMS CNES
Approved by	Noelia Sánchez-Ortiz	Project Coordinator	DMS

Participant no. *	Participant organisation name	Part. short name	Country
1 (Coordinator)	DEIMOS Space	**DMS**	Spain
2	Centre National de la Recherche Scientifique	**CNRS**	France
3	Collecte Localisation Satellites	**CLS**	France
4	Met Office	**MET**	United Kingdom
5	Centre National d'Etudes Spatiales	**CNES**	France
6	University College London	**UCL**	United Kingdom
8	Royal Obesrvatory of Belgium	**ROB**	Belgium

This page intentionally left blank

Document Information

Project Data	
Type of funding scheme:	Collaborative Project
FP7 Call	Space Call 3 - FP7-SPACE-2010-1
Work programme topic	Strengthening of Space foundations / Reducing the vulnerability of space assets: Security of space assets from space weather events (Objective SPA.2010.2.3-01)
Grant Agreement Number:	261948

ATMOP Internal Distribution			
Name	Organisation	Name	Organisation
ATMOP Team Members			
External Distribution			
Name	Organisation	Name	Organisation
Web sites visitors			
Confidentiality Level			
Public ☑		ATMOP team ☐	Steering Committee ☐

Archiving	
Word Processor:	MS Word 2000
File Name:	ATMOP-DMS-RPT-021-DTM2013 Command-line tool. Software Installation and User Manual-18112013.doc

Document Status Log

Issue	Change description	Date	Approved
1.0	Upgrades from DTM2012 documentation	18/11/2013	Noelia Sánchez

Table of Contents

List of Tables

1. INTRODUCTION

1.1. Purpose and Scope

The purpose of this document is that to facilitate the potential user of DTM-2013 pre-compiled library with the means to adequately make use of it. This SUM serves many purposes as to:

- Present the main functionality of DTM-2013 pre-compiled library
- Describe the procedure to link the library in all supported platforms
- Describe the interfaces of the tool with the user:
    1. Switches used to configure the behaviour of the library
    2. Input and output parameters of the main DTM subroutines and auxiliary routines
- Serve also as general guide of the tool architecture (for detailed information on internal routines, the SW reference manual shall be read.)
- Overall, serve as reference manual for the use of the DTM-2013 pre-compiled library.

1.2. Intended readership

The DTM-2013 pre-compiled library is provided free of charge to any user. It can be downloaded from the ATMOP website (http://www.atmop.eu). Prior to the use of the library, the user must accept all terms and conditions stated in the license agreement

The user should be familiar with the FORTRAN-90 programming language. One of the supported compilation environment should also be available and set-up (See list of supported environments at 4.1), and the user should be familiar with the environment.

1.3. Document Structure

This document is organised as follows:

- ❑ First chapter is this introduction
- ❑ Second chapter provides a summary of the contents of the SW package
- ❑ Third section provides the guidelines for the package installation in the supported platforms
- ❑ Fifth chapter contains the User manual for the two external routines, DTM and DTM wrapper.
- ❑ Finally, annexes provide the list of Event Messages, and a template to raise Software Problem Reports

1.4. Problem reporting instructions

In case an execution problem is encountered during DTM-2013 library utilisation and the user is convinced that the malfunction is motivated by a software problem, it is recommended to fill in a *Software Problem Report* (SPR) and send it to the tool responsible at Deimos Space for analysis and assessment of the possible solutions.

A standard form for the SPR is given in 2. The fill in of a number of fields in the SPR form is of special importance to allow the software developer to address the specific problem:

- Data identifying the user of the utility at the time of the failure
- Data identifying the utility where the problem was observed
- Description of the process carried out until the occurrence of the event
- The output error or warning prompted by the utility (if any)
- Description of any ancillary information that might be of interest in the case

In case of application, the utilised input files and the obtained output files in the process (if any) shall be sent along with the completed SPR form to enable an efficient and quick assessment of the problem.

1.5. Applicability statement

This release of DTM-2013 pre-compiled library is applicable to the software version 1.1 dated 17/10/12. The document explains how to install the software tool in the target platforms, how to make use of it and how the tool is set up and structured.

1.6. Acronyms and Abbreviations

The acronyms and abbreviations used in this document are the following ones:

Table 1: Table of Acronyms and Abbreviations

Acronym	Meaning
ATMOP	Advanced Thermosphere Modelling for Orbit Prediction
IDE	Integrated Development Environment
WP	Work Package

1.7. Related Documents

Table 2: Applicable Documents

Ref.	Code	Title	Date
[AD.1]	ATMOP-GA	ATMOP Grant Agreement	
[AD.2]	ATMOP-DoW	ATMOP Annex I - "Description of Work"	2010-08-13

Table 3: Reference Documents

Ref.	Reference Documents	Date
[RD.1]	SPENVIS Documentation http://www.spenvis.oma.be/help/background/indices.html	2012-08-13
[RD.2]	Barlier, F., Berger, C., Falin, J.L., Kockarts, G., Thuillier, G., 1978. A thermospheric model based on satellite drag data, Ann. Geophys, 34, 9-24.	1978
[RD.3]	Berger, C., Biancale, R., Ill, M., Barlier, F., 1998. Improvement of the empirical thermospheric model DTM: DTM-94- comparative review on various temporal variations and prospects in space geodesy applications, J of Geod., 72, 161- 178.	1998
[RD.4]	Bruinsma, S.L., Thuillier, G., Barlier, F., 2003. The DTM-2000 empirical thermosphere model with new data assimilation and constraints at lower boundary : accuracy and properties, J of Atmospheric and Solar-Terrestrial Physics, 65, 1053-1070.	2003
[RD.5]	Bruinsma, S.L., Tamagnan, D., Biancale, R., 2004. Atmospheric densities derived from CHAMP/STAR accelerometer observations, Plan. Space Sci., 52, 297-312.	2004
[RD.6]	Bruinsma, S.L., Sánchez-Ortiz, N., Olmedo, E., Guijarro, N. (2012) Evaluation of the DTM-2009 thermosphere model for benchmarking purposes, J. Space Weather Space Clim.	2012
[RD.7]	Dudok de Wit, T., S. Bruinsma (2011) Determination of the most pertinent EUV proxy for use in thermosphere modeling, Geophys. Res. Lett., doi:10.1029/2011GL049028	2011

2. SW PACKAGE OVERVIEW

The pre-compiled DTM2013 library is distributed in a compressed package. This package contains a static library file and several .mod files. In addition to this, an example source file is provided.

The static library file provides two subroutines, which provide the main functionalities:

- `dtm2013` executes the DTM-2013 model. The user must provide the location of the point where the model is to be executed (latitude, longitude, altitude and local solar time) and the values of the solar and geomagnetic proxies (F30, FBAR, and AKP)
- `dtm_wrapper` is a wrapper for the former subroutine. This routine requires only the location and the universal time. The routine internally computes the local solar time and the values of the geomagnetic and solar activity proxies.

2.1. Summary of DTM 2013 model

2.1.1. The pre-ATMOP DTM models

The Drag Temperature Model (DTM) is a semi empirical model describing the temperature, density and composition of the Earth's thermosphere.

Its first version DTM-78 (Barlier et al., 1978) used direct measurements of exospheric temperature and atmospheric densities derived from satellite drag data. It has been officially upgraded twice: DTM-94 (Berger et al., 1998) and DTM-2000 (Bruinsma et al., 2003). DTM-2000 was the first model to have been constructed not with the usual solar activity proxy F10.7, but with the MgII index converted to F10.7 units (sfu). These models are mainly used in orbit determination software. The model is maintained and developed by CNES. The latest pre-ATMOP version of DTM is DTM-2009 (Bruinsma et al., 2012), which was the only model to have assimilated high-accuracy and high-resolution densities inferred from accelerometers onboard CHAMP (Bruinsma et al., 2004) and GRACE. This model is the starting point of the new developments in the framework of the "Advanced Thermosphere Model for Orbit Prediction (ATMOP)" research project.

2.1.2. The DTM-2013 model

DTM-2013 is the first of two models that are developed within the ATMOP project. The objective is to improve the model in two stages, first by replacing the solar activity proxy index F10.7 with a more representative index of chromospheric activity, such as MgII, or directly with measurements of the HeII line in the solar EUV band. The second and final ATMOP model will be released in the fall of 2013, and at that time a new geomagnetic index, which is currently being elaborated within ATMOP, will be used instead of the 3-hourly planetary geomagnetic index am. A study by Dudok de Wit and Bruinsma (2011) showed that the best modeling results are obtained with HeII measurements from SOHO/SEM on all time scales (i.e. smaller and larger than a solar rotation). Therefore, two versions of DTM-2013 were constructed under identical conditions except for the solar activity indices used, either F10.7 or S10.7 (SOHO/SEM converted to F10.7 units). Initially the F10.7 model was simply constructed for

reference purposes in order to quantify the gain in accuracy entirely due to the solar activity index, but after full evaluation of both models it was decided to select the F10.7 model as the new DTM-2013. In short, there is a problem in the slowly evolving component of HeII/S10.7 when using the complete SOHO time series starting in 1996. Dudok de Wit and Bruinsma (2011) did not detect the problem in part because they analyzed a shorter time span.

A model evaluation is given in the Evaluation Report, which shows comparisons to data for DTM2013, as well as the a-priori model DTM-2009, and the COSPAR reference models JB2008 and NRLMSISE-00 available at ATMOP web page.

2.2. Main Routines

The main functionality of the DTM2013 model is provided by the `dtm2013` subroutine. This routine requires the user to provide some values of the solar and geomagnetic proxies. As this process is not straightforward, a wrapper routine for `dtm2013` is provided. This routine, called `dtm_wrapper`, is called with the universal time, and it computes the proxies required for `dtm2013`.

In addition to the two main subroutines listed above, the pre-compiled library contains other auxiliary routines. These routines are described briefly here for the sake of completeness.

- load_config: allows the user to set input file paths and fortran unit descriptors by means of a configuration file
- `P_ReadDTM13` reads input files required for the DTM2013 routine. It must be called once before the execution of `dtm2013`. It is not required, however, for the execution of `dtm_wrapper` (as `dtm_wrapper` calls this routine internally)
- `density_uncertainty` computes the uncertainties in the value of the density computed in a previous call to `dtm_wrapper`. This routine is also called internally by `dtm_wrapper`

3. SW PACKAGE CONTENTS

The package contains several files:

- Two source files (.f90) which must be compiled by the user in order to use the library
- An example program, which can be compiled by the user in order to test the compilation environment
- The library itself
- With gfortran versions, an interface which allows linking DTM-2013 from C/C++ programs is also provided. These files comprise two header files (for C and C++) and an example program which illustrates how to link the DTM-2013 library with a C program.

Table 4: Files provided in the DTM-2013 precompiled library packages

Platform	Compiler used to compile the library	Archived file format	Files
All			dtm_interfaces.f90 t_dtm_date.f90 dtm2013_example_program.f90 config.cfg Reference manual Data folder (6 files)
Linux x86 (gfortran)	4.4.5 (Debian 4.4.5-8)	.tar.gz	libdtm2013_linux_x86.a DTM2013.h DTM2013.hpp dtm2013_c_example.c
Linux AMD64 (gfortran)	4.4.5 (Debian 4.4.5-8)	.tar.gz	libdtm2013_linux_x86.a DTM2013.h DTM2013.hpp dtm2013_c_example.c
Windows x86 (mingw gfortran)	4.5.2	.zip	libdtm2013_mingw_x86.a DTM2013.h DTM2013.hpp dtm2013_c_example.c
Windows x86 (intel Fortran compiler)	ifort 12.0.5.221	.zip	libdtm2013_ifortran_x86.lib
Unix SunOS 5.8 + SUN workshop 6	Sun WorkShop(TM) 6 update 2 Fortran 95 compiler.	.tar.gz	libdtm2013_solaris.a

4. INSTALLATION PROCEDURE

In this section, generic procedures for linking the pre-compiled library with the provided example program are explained for all platforms. These procedures allow the user to link by using command-line tools. It is possible to add the pre-compiled library to an existing project by means of an integrated development environment, such as Microsoft Visual Studio or Eclipse. However, due to the large number of IDEs and versions, no information about compilation in IDEs is provided. In case you want to include the library in an IDE, please refer to your IDE documentation.

In order for the library to run, a number of data files must be available. The routine assumes that all data files are stored in a folder called `data`, which must be in the same subdirectory from where the program is executed. It is possible, however, to configure the library to find these files in another location.

4.1. Target Platforms

The DTM-2013 precompiled library has been developed to be executed under a number of environments that are common among researchers. Focus has been put on the Linux platforms and the *gfortran* compiler, since *gfortran* is free software, and is available on almost any Linux distribution.

The supported platforms are:

- ❑ Linux x86 + gfortran
- ❑ Linux AMD64 + gfortran
- ❑ Windows (xp or newer) x86 + gfortran (mingw)
- ❑ Windows (xp or newer) x86 + intel fortran compiler
- ❑ SUN (SunOS 5.8)

The binaries provided in the library should be compatible among compilers of the same family (for example, binaries generated with gfortran 4.5.x should be compatible with gfortran 4.7.*), However, it is not possible to ensure that the library is compatible with every compiler. In case of binary compatibility problems, please contact the support team.

4.1.1. Linux (x86 and AMD64) + gfortran

Follow these steps for the use of the routines:

- Check that gfortran is installed in your system. If it is not, refer to your linux distribution documentation

```
gfortran --version
```

- Navigate to the directory where the package file was downloaded

```
cd /path/to/download/directory
```

- Extract the contents of the package (use the first command if the environment is 32-bit, or the second if the environment is 64-bit)

```
tar xvzf libdtm2013_x86_linux_1.3.tar.gz
tar xvzf libdtm2013_amd64_linux_1.3.tar.gz
```

- Compile the example program (use the first command if the environment is 32-bit, or the second if it is 64-bit)

```
gfortran -Wall -O2 lib/t_dtm_date.f90 lib/dtm_interfaces.f90 lib/dtm2013_example_program.f90
                        lib/libdtm2013_linux_x86.a -o test_program
gfortran -Wall -O2 -  lib/t_dtm_date.f90 lib/dtm_interfaces.f90 lib/dtm2013_example_program.f90
                        lib/libdtm2013_linux_x86.a -o test_program
```

- Run the program to test the library

```
./test_program
```

- An output like the following should be printed to the screen. Note that due to the differences in the compiler implementations, your results might be slightly different from these. Also bear in mind that the example program expects to find a file named "config.cfg" (provided) in the directory it is called from

```
-----------CALL TO DTM2013-----------


 inputs
   Date:             4263.3559517  (  3/ 9/2011 ) at  8:32:34.230000000
   Day of year:      246.355957
   local time (hours):    8.2761755
   latitude:           40.0000000
   longitude:          -4.0000000
   altitude:          200.0000000
   f :                120.0678558    0.0000000
   fbar :             103.3559494    0.0000000
   akp :                2.6002905    0.0000000    1.6877123    0.0000000


 outputs
   Temp at altitude : 893.1622925
   exospheric tmp :   968.7214355
   atomic hydrogen :    2.7588559E-19
   helium :             5.1591831E-17
   atomic oxygen :      7.0344270E-14
   molecular nitro :    1.8052648E-13
   molecular oxygen :   1.6177149E-14
   density (g/cm^3) :   2.6709974E-13  (+-   3.02% )
   mean molec mass :    2.3506405E+01
```

4.1.2. Windows (x86) + mingw gfortran

Follow these steps for the use of the routines:

- Extract the downloaded package to a folder of your choice
- Open a mingw shell

```
Start > All Programs > MinGW > MinGW Shell
```

- Navigate to the directory where the package was extracted (remember than in mingw shell, `/c` corresponds to the C: drive letter)

```
cd /c/path/to/download/directory
```

- Compile the example program

```
gfortran -Wall -O2 -static lib/t_dtm_date.f90 lib/dtm_interfaces.f90
lib/dtm2013_example_program.f90 lib/libdtm2013_linux_x86.a -o test_program
```

- Run the program to test the library

```
./test_program
```

- An output similar to that from the Linux version should appear

4.1.3. Windows (x86) + intel fortran compiler

Follow these steps for the use of the routines:

- Unzip the DTM-2013 pre-compiled package to a folder of your choice.
- Open an intel terminal

```
Start > All programs > Intel Parallel Studio XE 2011 > Command Prompt >
Parallel Studio XE with intel compiler > IA-32 Visual Studio 2010 mode
```

- Navigate to the folder where you unzipped the package

```
cd path/to/download/directory
```

- Compile the example program with the following command

```
ifort lib/t_dtm_date.f90 lib/dtm_interfaces.f90 lib/dtm2013_example_program.f90
lib/libdtm2013_ifortran_x86.libs /o test_program
```

- Execute the example program

```
test_program
```

- An output similar to that from the Linux version should appear

4.1.4. Unix (SunOS 5.8) + SUN workshop 6

Follow these steps for the use of the routines:

- Navigate to the directory where the package file was downloaded

```
cd /path/to/download/directory
```

- Extract the contents of the package

```
gunzip libdtm2013_solaris_v1.1.tar.gz
tar xvf libdtm2013_solaris_v1.1.tar
```

- Navigate to the source directory

```
cd src
```

- Convert the source file end line delimitators to SunOS format.

```
dos2unix DTM_2013_subroutines.f90 DTM_2013_subroutines.f90
dos2unix DTM_config.f90 DTM_config.f90
dos2unix dtm_wrapper.f90 dtm_wrapper.f90
dos2unix dtm2013_example_program.f90 dtm2013_example_program.f90
```

- Compile the test program.

```
f90 -O2 lib/t_dtm_date.f90 lib/dtm_interfaces.f90 lib/dtm2013_example_program.f90
lib/libdtm2013_solaris.a –Mlib -o test_program
```

- Run the program

```
./test_program
```

- An output similar to the linux version should be printed to the screen

5. SW USER'S MANUAL

In this section we describe the general architecture and all the public interfaces to the DTM2013 pre-compiled library. For each of the subroutines, we provide the complete subroutine header as well as a brief description of the purpose of the routine. For the modules, their contents are listed and described.

All the functionalities are provided by the following subroutines:

- `dtm2013`: Main functionality
- `dtm_wrapper`: Wrapper for the dtm2013 subroutine
- `P_ReadDTM13`: Initialize `dtm2013`
- `density_uncertainty`: Compute density uncertainty after `dtm2013`
- `load config:` Configure DTM2013 from a file

The last two subroutines are auxiliary. `P_ReadDTM13` must be called prior to `dtm2013`. `density_uncertainty` may be called (if required) after `dtm2013`. The wrapper (`dtm_wrapper`) does call these functions automatically, so no user interaction is required in this case. The call to `load config` is optional, and is performed before calling any other routine in the package.

In order to call these routines, the user must compile two source files: `t_dtm_date` and `dtm_intefaces`.

- `t_dtm_date` provides the `dtm_date` custom type. This type provides a convenient way to pass dates to the `dtm_wrapper` routine. Therefore, every routine that calls `dtm_wrapper` must use this module.
- The `dtm_interfaces` module provides two modules with explicit interfaces for all the routines in the library and, although optional, it is advisable to use it in order to prevent wrong calls to the subroutines.

5.1. Subroutines, modules and headers provided by the library

The contents of the pre-compiled library are listed in Table 5

Table 5: Contents of the DTM2013 pre-compiled library

Name	Type	Description
dtm2013	Subroutine	Main dtm2013 subroutine
dtm_wrapper	Subroutine	Wrapper for the dtm2013 subroutine
density_uncertainty	Subroutine	Computes the density uncertainty for a given position
P_ReadDTM13	Subroutine	Initializes some values required for the dtm2013 subroutine
load_config	Subroutine	Loads a configuration from a file
dtm_interfaces	Module	Provides explicit interfaces to DTM routines
t_dtm_date	Module	Provides the "dtm_date" custom type
DTM2013.h	C header	Header to call DTM2013 from C programs
DTM2013.hpp	C++ header	Header to call DTM2013 from C++ programs

5.1.1. dtm2013

The `dtm2013` subroutine is the subroutine which implements the DTM2013 model. In must be always taken into account that it is necessary to initialize the DTM2013 model before calling the `dtm2013` subroutine. To do this, call the subroutine `P_ReadDTM13`

```
!------------------------------------------------------------------------------
!
! ROUTINE: dtm2013
!
!> @author Sean Bruinsma
!>
!> @brief calculation of temperature and density with DTM2009
!
! PROTOTYPE:
!                     dtm2013 (
!                              real, intent(in) :: day
!                              real, intent(in) :: f(2)
!                              real, intent(in) :: fbar(2)
!                              real, intent(in) :: akp(4)
!                              real, intent(in) :: alti
!                              real, intent(in) :: hl
```

```
!                              real, intent(in) :: alat
!                              real, intent(in) :: xlon
!                              real, intent(out) :: tz
!                              real, intent(out) :: tinf
!                              real, intent(out) :: tp120
!                              real, intent(out) :: ro
!                              real, intent(out) :: d(6)
!                              real, intent(out) :: wmm
!                             )
! INPUT ARGUMENTS:
!>                    @param[in] day day-of-year [0-366]
!>                    @param[in] f(2) f(1)=instantaneous flux at (day - 1) / f(2)=0
!>                    @param[in] fbar(2) fbar(1)=average flux at t and derivative / fbar(2)=0.
!>                    @param[in] akp(4) kp delayed by 3 hours, akp(3)=mean of last 24 hours / akp(2) &
                      akp(4)=0
!>                    @param[in] alti altitude (in km) greater than 120 km
!>                    @param[in] hl local time (in radian)
!>                    @param[in] alat latitude (in radian)
!>                    @param[in] xlon longitude (in radian)
!
!
! OUTPUT ARGUMENTS:
!>                    @param[out] ro          density (g/cm^3) at the given position
!>                    @param[out] tinf        exosphere temperature
!>                    @param[out] tz          temperature at the given height
!>                    @param[out] xmm         mean molecular mass
!>                    @param[out] tp120
!>                    @param[out] d(6)        concentrations in atomic hydrogen (1)
!>                                            concentration in helium (2)
!>                                            concentration in atomic oxygen (3)
!>                                            concentration in molecular nitrogen (4)
!>                                            concentration in molecular oxygen (5)
!>                                            concentration in atomic nitrogen (6) (unused)
!
!> @date 06/2012
!
!-----------------------------------------------------------------------
```

5.1.2. density_uncertainty

This subroutine computes the uncertainty of the density value due to the low model resolution (i.e. 1-sigma small scale variability) returned by the DTM2013 at a given position. It accesses three files. The location of these files and the units used to access them can be configured by using the `dtm_config` module.

```
!------------------------------------------------------------------------
!
! ROUTINE: densisty_uncertainty
!
!> @author Sean Bruinsma
!>
!> @brief  calculation of density uncertainty
!
! PROTOTYPE:
!         density_uncertainty(
!                              real, intent(in) :: alt
!                              real, intent(in) :: lat
!                              real, intent(in) :: lst
!                              real, intent(in) :: flux
!                              real, intent(in) :: kp
!                              real, intent(out) :: unc
!                             )
!
!
! INPUT ARGUMENTS:
!>                    @param[in] alt altitude in km (NB: uncertainties NOT correct - too big - for
                                altitudes<300 km)
!>                    @param[in] lat latitude in degrees
!>                    @param[in] lst local solar time in hours (0. - 23.999)
!>                    @param[in] flux mean solar flux (sfu)
!>                    @param[in] kp geomagnetic activity (0. - 9.0)
!
!
! OUTPUT ARGUMENTS:
!>                    @param[out] unc density uncertainty (i.e. 1-sigma small scale variability)
!
!> @date
!
!------------------------------------------------------------------------
```

5.1.3. P_ReadDTM13

`P_ReadDTM13` initializes some values required by `dtm2013`. Therefore, it must be called before calling `dtm2013`. This routine needs to read a file. The location of this file and the unit used to access it can be configured by using the `dtm_config` module.

```
!-------------------------------------------------------------------------
!
! ROUTINE: P_ReadDTM12
!
!> @author Sean Bruinsma
!>
!> @brief  read DTM format/version 2012
!
! PROTOTYPE:
!                 P_ReadDTM13 (
!                             )
!
!
! INPUT ARGUMENTS:
!
!
! OUTPUT ARGUMENTS:
!
!> @date 03/2012
!
!-------------------------------------------------------------------------
```

5.1.4. dtm_wrapper

`dtm_wrapper` provides the results returned by `dtm2013` for a location (defined as longitude, latitude and altitude) and a universal time (defined by either a MJD2000 date or a calendar date)

The first time the routine is called, it reads the proxy files and stores all their contents into memory. Next it processes the user input date by calling the `process_input_date` subroutine. Next, it computes the local solar time and the day of the year.

After that, it computes the f, fbar and akp coefficients by interpolation, as required by the `dtm2013` subroutine.

Once all inputs to dtm have been found, it calls `dtm2013` and `density_uncertainty`. The outputs from `dtm_wrapper` are those returned by `dtm2013` and `density_uncertainty.`

The last five arguments of the routine are optional, and provide the results of the intermediate computations performed by the wrapper prior to calling dtm2013.

```
!------------------------------------------------------------------------
 !
 ! ROUTINE: dtm_wrapper
 !
 !> @author Raul Dominguez
 !>
 !> @brief  compute the required arguments for dtm2013 subroutine, call it
 !>              and return the results.
 !
 ! INPUT ARGUMENTS:
 !>                   @param[in] alat          latitude in degrees
 !>                   @param[in] xlon          longitude in degrees
 !>                   @param[in] alti          altitude in km (must be greater than 120)
 !>                   @param[in] in_date       UT date in MJD2000 or calendar date
 !
 ! OUTPUT ARGUMENTS:
 !>                   @param[out] ro           density (g/cm^3) at the given position
 !>                   @param[out] ro_unc       density uncertainty
 !>                   @param[out] tinf         exospheric temperature
 !>                   @param[out] tp120
 !>                   @param[out] tz           temperature at the given height
 !>                   @param[out] xmm          mean molecular mass
 !>                   @param[out] d(1)         concentration in atomic hydrogen
 !>                   @param[out] d(2)         concentration in helium
 !>                   @param[out] d(3)         concentration in atomic oxygen
 !>                   @param[out] d(4)         concentration in molecular nitrogen
 !>                   @param[out] d(5)         concentration in molecular oxygen
 !>                   @param[out] d(6)         concentration in atomic nitrogen
```

```
 !>                  @param[out] f_out         [optional] f values computed by the DTM2013 wrapper
 !>                  @param[out] fbar_out      [optional] fbar values computed by the DTM2013 wrapper
 !>                  @param[out] akp_out       [optional] akp values computed by the DTM2013 wrapper
 !>                  @param[out] hl_out        [optional] local solar time (in days) computed by the
DTM2013 wrapper
 !>                  @param[out] dayofyear_out [optional] day of year (in days) computed by the DTM2013
wrapper
 !
 ! REVISION HISTORY:
 !
 ! version 1.0
 ! date 31/07/2012
 ! Initial version
 !
 ! version 1.1
 ! date 10/08/2012
 ! Added density uncertainty computation
 !
 ! version 1.2
 ! date 10/08/2012
 ! Changed internal variable dayofyear from integer to real to match its kind
 ! in DTM2012 subroutine
 ! Changed date input, to allow the user to use either MJD2000 or calendar date
 !
 ! version 1.3
 ! date 11/09/2012
 ! Bugfix. The density uncertainty was being called with local solar time in
 ! radians, and should be hours
 !
 ! version 1.4
 ! date 14/09/2012
 ! Change. Input parameters are now real (before they were real*8)
 !
 ! version 1.6
 ! date 19/09/2012
 ! Change in the solar proxy file format (email from Sean Bruinsma at 19/09/2012)
 ! Fixed a bug: The am values where not being read properly
 !
 ! @version 1.7
 ! @date 28/09/2012
 ! Bugfix. The interpolated akp value was not being computed properly. In order to
 ! interpolate, the minutes and seconds of the user input hour were not being
```

```
 ! considered
 !
 ! @version 1.8
 ! @date 15/0/2013
 ! Took this routine out from dtm_wrap module. This way, compatibility
 ! problems among several compiler versions are prevented.
 ! Now the values f_out,fbar_out,akp_out,hl_out,dayofyear_out
 ! are optional arguments, instead of shared through common memory
 !-------------------------------------------------------------------------
```

5.1.5. load_config

This subroutine opens a file (by using the unit descriptor provided in the header), and loads a configuration from it. The call to this routine is optional (if it is not called, the library provides default values)

The configuration file may have comments (lines beginning with '#') and configuration keys. These must be written as CONFIG_VARIABLE = "VALUE" (see example below, which includes the default values). It is possible to fill in paths to all files required by DTM2013, and to set a range of file unit descriptors to handle files within the fortran code.

```
!-------------------------------------------------------------------------
!
! SUBROUTINE: load_config
!
!> @brief this routine reads a configuration from the config.cfg file
!         and stores it in the dtm_config module
!
!> @author Raul Dominguez
!
! INPUT ARGUMENTS:
!>                    @param[in] unit          Fortran channel used to access the
config file
!>                    @param[in] file          Path to the configuration file
!
! REVISION HISTORY: 18/07/2013 Initial version
!
!-------------------------------------------------------------------------
```

```
# DTM-2013 configuration file
# This file is used to set the paths of all files required by the DTM-2013 library
# In case some value is missing, default values are used (see table below)
#
# All values (even integer numbers) must be filled between double quotes (")
#
#    PATH_TO_A_FILE = "data/am_file_spider.dat"/
```

```
#     PATH_TO_F_FILE = "data/proxies_unadjusted.dat"/
#     PATH_TO_DTM2013_IN_FILE = "data/dtm_2013_NF.dat"
#     PATH_TO_DTM2013_OUT_FILE = "ZVERIF_xxx.DAT"
#     PATH_TO_DEN_UNC_1 = "data/smoothrelvar_350km_kpmax3"
#     PATH_TO_DEN_UNC_2 = "data/smoothratio_kpmin4-to-max3"
#     PATH_TO_DEN_UNC_3 = "data/smoothratio_500to350km"
#
#
#
    PATH_TO_A_FILE = "data/am_file_spider.dat"/
    PATH_TO_F_FILE = "data/proxies_unadjusted.dat"/
    PATH_TO_DTM2013_IN_FILE = "data/dtm_2013_NF.dat"
    PATH_TO_DTM2013_OUT_FILE = "ZVERIF_xxx.DAT"
    PATH_TO_DEN_UNC_1 = "data/smoothrelvar_350km_kpmax3"
    PATH_TO_DEN_UNC_2 = "data/smoothratio_kpmin4-to-max3"
    PATH_TO_DEN_UNC_3 = "data/smoothratio_500to350km"
    FILEUNIT_START = "98"
#
#
```

5.1.6. t_dtm_date module

This module provides a custom type (dtm_date) which is required to call the `dtm_wrapper` subroutine.

```
type dtm_date
     integer type_flag !<1 for MJD2000 date, 2 for calendar date
     real*8 mjd2000 !< Date in MJD2000
     integer day !< Day
     integer month !< Month
     integer year !< Year
     integer hour !< Hour
     integer minute !< Minute
     real*8 second !< Seconds
end type
```

This custom type is used to provide the required universal time to `dtm_wrapper`. It is possible to use either a date in calendar format, or in MJD2000 format (commonly used in astrodynamics).

It is necessary to set the `type_flag` to the kind of format to be used. Thus:

- To use a MJD2000 date, set `type_flag` to 1, and mjd2000 to the actual date. The rest of the components are unused, therefore, it is not necessary to set them
- To use a calendar date, set `type_flag` to 2, and day, month, year, hour, minute and second to the required values. It is not necessary to set mjd2000.

5.1.7. dtm_interfaces

This module provides explicit interfaces to the routines provided in the DTM2013 library. Although it is not necessary to use it, it is highly recommended, because having the routine interfaces explicitly defined prevents many bugs.

5.2. Use of DTM-2013 wrapper subroutine

In this section we provide an example program which uses the `dtm_wrapper` subroutine. This example shows the required interface for the `dtm_wrapper` routine (specially, the date)

```
program dtm2013_example_program

!This program is an example on how to use the DTM 2013 wrapper

use dtm_wrapper_interfaces  !Load the interface for the "dtm_wrapper" subroutine
use dtm2013_interfaces      !Load the interface for the "dtm2013", "P_ReadDTM13", and
                            !"density uncertainty" subroutines
use dtm_config              !DTM configuration
use t_dtm_date              !Provides the dtm_date custom type

implicit none

!Define pi
real pi
parameter(pi=3.14159265)

!INPUT for dtm_wrapper
    type(dtm_date) today !the "dtm_date" type is defined within the t_dtm_date
                         !it provides a convenient way to input the universal time
                         !to the dtm wrapper routine

    real alti !altitude in km
    real alat !latitude in degrees
    real xlon !longitude in degrees

!OUTPUT from dtm_wrapper
      real :: ro     !density at the given altitude
      real :: tinf   !exospheric temperature
      real :: tz     !temperature at the given altitude
      real :: tp120  !vertical temperature gradient at 120 km
      real :: wmm    !mean molecular mass
      real :: ro_unc !Uncertainty in density
      real :: d(6)   !Concentrations of:
                     ! 1 atomic hydrogen
                     ! 2 helium
                     ! 3 atomic oxygen
                     ! 4 molecular nitrogen
                     ! 5 molecular oxygen
                     ! 6 atomic nitrogen (currently unused)

      !optional arguments for dtm_wrapper
      real, dimension(2) :: f_out
```

```
      real, dimension(2) :: fbar_out
      real, dimension(4) :: akp_out
      real :: hl_out
      real :: dayofyear_out


!Begin program

call load_config(784,"config.cfg")

!Set inputs
 alti=200.0 !real
 xlon=-4.0
 alat=40.0

! set a date

today%type_flag=2 ! Set this flag to 2 to input a calendar date
today%day=3
today%month=9
today%year=2011
today%hour=8
today%minute=32
today%second=34.23d0 ! warning! real*8


!call dtm wrapper

call dtm_wrapper(today,alti,alat,xlon,tz,tinf,tp120,ro,ro_unc,d,wmm, &
                 f_out=f_out,fbar_out=fbar_out,akp_out=akp_out, &
                 hl_out=hl_out,dayofyear_out=dayofyear_out)


!output the results

      write(*,*) " "
      write(*,'("-----------CALL TO DTM2013-----------")')
      write(*,*) " "


      write(*,*) "inputs"
       write(*,'("   Date:           ", f12.7, "  ( ",i2,"/",i2,"/",i4, " ) at ", i2, ":", i2, ":", f12.9)')&
                  today%mjd2000, today%day, today%month,today%year, today%hour, today%minute, today%second
      write(*,'("   Day of year:      ", f10.6)') dayofyear_out
      write(*,'("   local time (hours): ", f12.7)') hl_out
      write(*,'("   latitude:         ", f12.7)') alat
      write(*,'("   longitude:        ", f12.7)') xlon
      write(*,'("   altitude:         ", f12.7)') alti
      write(*,'("   f :               ", 2(f12.7,1x))') f_out
      write(*,'("   fbar :            ", 2(f12.7,1x))') fbar_out
      write(*,'("   akp :             ", 4(f12.7,1x))') akp_out
      write(*,*) " "
      write(*,*) "outputs"
      write(*,'("   Temp at altitude :", f12.7)') tz
      write(*,'("   exospheric tmp :  ", f12.7)') tinf
      write(*,'("   atomic hydrogen : ", ES16.7)') d(1)
      write(*,'("   helium :          ", ES16.7)') d(2)
      write(*,'("   atomic oxygen :   ", ES16.7)') d(3)
      write(*,'("   molecular nitro : ", ES16.7)') d(4)
      write(*,'("   molecular oxygen :", ES16.7)') d(5)
      !write(*,'("   atomic nitrogen : ", ES16.7)') d(6) Ignore this value
      write(*,'("   density (g/cm^3) :", ES16.7, "  (+- ", f6.2, "% )")') ro, ro_unc
      write(*,'("   mean molec mass : ", ES16.7)') wmm

end program
```

5.3. Direct Use of DTM-2013 subroutine

Here, we provide an example of a program which calls the dtm2013 subroutine directly. In this case, the user must provide values for the solar and geomagnetic proxies (which should be computed elsewhere) and for the local solar time and the day of year

```
program dtm2013_example_direct_call

!This program is an example on how to use the DTM 2013 subroutine

use dtm2013_interfaces      !Load the interface for the "dtm2013", "P_ReadDTM13", and
                            !"density uncertainty" subroutines


implicit none

!Define pi
real pi
parameter(pi=3.14159265)

!Define input and output variables
!INPUT
    real alti !altitude in km
    real alat !latitude in degrees
    real xlon !longitude in degrees
    real f(2) !solar proxy
    real fbar(2) !solar proxy
    real akp(4) !geomagnetic proxy
    real local_solar_time !local solar time in radians
    real day_of_year !the day of year corresponding to the current date
!OUTPUT
      real :: ro     !density at the given altitude
      real :: tinf   !exospheric temperature
      real :: tz     !temperature at the given altitude
      real :: tp120  !vertical temperature gradient at 120 km
      real :: wmm    !mean molecular mass
      real :: ro_unc !Uncertainty in density
      real :: d(6)   !Concentrations of:
                     ! 1 atomic hydrogen
                     ! 2 helium
                     ! 3 atomic oxygen
                     ! 4 molecular nitrogen
                     ! 5 molecular oxygen
                     ! 6 atomic nitrogen (currently unused)

!Begin program
!Set inputs
 alti=200.0
 xlon=-4.0*pi/180.0
 alat=40.0*pi/180.0
 f(1)= 16.5746403
 f(2)=0.0
 fbar(1)=98.0711899
 fbar(2)=0.0
 akp(1)=2.6002905
 akp(2)=0.0
 akp(3)=1.6877123
 akp(4)=0.0
 local_solar_time=2.16669767922
 day_of_year=246.355957


!First, a call to P_ReadDTM13 is required to initialize the dtm2013 model
```

```
 call P_ReadDTM13 ()

!Now call dtm2013

 call dtm2013(day_of_year,f,fbar,akp,alti,local_solar_time,alat,xlon,tz,tinf,tp120,ro,d,wmm)

!Obtain the density uncertainty

!The local solar time must be in hours for the density_uncertainty routine

local_solar_time=local_solar_time*24.0/(2.0*pi)

 call density_uncertainty(alti,alat,local_solar_time,f(1),akp(1),ro_unc)

!output the results


      write(*,*) " "
      write(*,'("-----------CALL TO DTM2013-----------")')
      write(*,*) " "


      write(*,*) "inputs"
      !write(*,'("   Date:             ", f12.7, "  ( ",i2,"/",i2,"/",i4, " ) at ", i2, ":", i2, ":",
f12.9)') today%mjd2000, &
      !                                  today%day, today%month,today%year, today%hour, today%minute,
today%second
      !write(*,'("   Day of year:      ", f10.6)') dayofyear_out
      write(*,'("   local time (hours): ", f12.7)') local_solar_time
      write(*,'("   latitude:         ", f12.7)') alat
      write(*,'("   longitude:        ", f12.7)') xlon
      write(*,'("   altitude:         ", f12.7)') alti
      write(*,'("   f :               ", 2(f12.7,1x))') f
      write(*,'("   fbar :            ", 2(f12.7,1x))') fbar
      write(*,'("   akp :             ", 4(f12.7,1x))') akp
      write(*,*) " "
      write(*,*) "outputs"
      write(*,'("   Temp at altitude :", f12.7)') tz
      write(*,'("   exospheric tmp :  ", f12.7)') tinf
      write(*,'("   atomic hydrogen : ", ES16.7)') d(1)
      write(*,'("   helium :          ", ES16.7)') d(2)
      write(*,'("   atomic oxygen :   ", ES16.7)') d(3)
      write(*,'("   molecular nitro : ", ES16.7)') d(4)
      write(*,'("   molecular oxygen :", ES16.7)') d(5)
      !write(*,'("   atomic nitrogen : ", ES16.7)') d(6) Ignore this value
      write(*,'("   density (g/cm^3) :", ES16.7, "  (+- ", f6.2, "% )")') ro, ro_unc
      write(*,'("   mean molec mass : ", ES16.7)') wmm


end program dtm2013_example_direct_call
```

5.4. Use of DTM-2013 from C/C++

It is possible to use DTM-2013 in C or C++ programs (only with gcc/g++ compilers). In order to do so, two header files (`DTM2013.h` and `DTM2013.hpp`) are provided. The first one `DTM2013.h` is to be included in C programs (i.e., source files which are compiled with the gcc compiler). `DTM2013.hpp` is for source files compiled with g++

The header files provide a custom type "`dtm_date`" similar to the one described in former sections. Also, it provides headers to the five main routines (`dtm2013`, `dtm_wrapper`, `P_ReadDTM13`, `density_uncertainty`, and `load_config`).

The four routines provide the same functionalities as their fortran counterparts. The only difference is that the C `dtm_wrapper` does not provide the optional outputs provided be the fortran `dtm_wrapper`.

The following example program illustrates the usage of DTM2013 from C

```
#include<stdio.h>
#include<string.h>
#include<DTM2013.h>

int main() {

   //Path to configuration file
    char configFile[200];
    strncpy(configFile,"config.cfg",200);


   //DTM wrapper inputs
    struct dtm_date today; //struct provided by DTM2013.h
    float alti;
    float alat;
    float xlon;

    //DTM wrapper outputs
    float ro;
    float tinf;
    float tz;
    float tp120;
    float wmm;
    float ro_unc;
    float d[6];

    //Fill in the universal time in calendar date format
    today.type_flag=2;
    today.day=3;
    today.month=9;
    today.year=2011;
    today.hour=8;
    today.minute=32;
    today.second=34.23;

    //Fill in the other inputs
    alti=200.0;
    alat=40.0;
    xlon=-4.0;
```

```
    //load configuration
    load_config(configFile);

    //Call dtm wrapper
    dtm_wrapper (&today, &alti, &alat, &xlon, &tz, &tinf, &tp120, &ro, &ro_unc, d, &wmm);

    printf("\n");
    printf("-----------CALL TO DTM2013-----------\n");
    printf("--------------C WRAPPER--------------\n");
    printf("\n");


    printf("inputs\n");
    printf("   Date:             %f (%i/%i/%i) at %i : %i : %f\n", today.mjd2000, today.day,
today.month,today.year, today.hour, today.minute, today.second);
    printf("\n");
    printf("\n");
    printf("   latitude:            %f\n", alat);
    printf("   longitude:           %f\n", xlon);
    printf("   altitude:            %f\n", alti);
    printf("\n");
    printf("\n");


    printf("\n");
    printf("outputs\n");
    printf("   Temp at altitude : %f\n", tz);
    printf("   exospheric tmp :   %f\n", tinf);
    printf("   atomic hydrogen :  %g\n", d[0]);
    printf("   helium :           %g\n", d[1]);
    printf("   atomic oxygen :    %g\n", d[2]);
    printf("   molecular nitro :  %g\n", d[3]);
    printf("   molecular oxygen : %g\n", d[4]);
    printf("   atomic nitrogen :  %g\n", d[5]);
    printf("   density (g/cm^3) : %g   (+-  %f %% )\n", ro, ro_unc);
    printf("   mean molec mass :  %g\n", wmm);


   return 0;
}
```

In order to link this program with the DTM-2013 library, it is necessary to instruct the gcc/g++ compiler to use the gnu libgfortran library. Therefore, the command required to compile this example (in x86 linux) would be:

```
gcc dtm2013_c_example.c libdtm2013_linux_x86.a -I. -o dtm2013_c_example -lgfortran
```

ANNEX A: EVENT MESSAGES

In case the `dtm_wrapper` subroutine encounters a fatal error during its execution, it displays a message to the screen (starting with the `string FATAL ERROR in dtm_wrap module`) and then stops program execution.

Table 6: Fatal error messages

Message	Explanation	Action to solve the problem
Cannot read 'f' from fluxes array. Dates in file do not match user input date	The user has requested dtm_wrapper to compute a solar proxy, but the solar proxy data file has not entries for the date required by the user	Either call dtm2013 directly (by providing solar proxies computed elsewhere) or update the solar proxies data file
Cannot read 'a' from geomagnetic data array. Dates in file do not match user input date	The user has requested dtm_wrapper to compute a geomagnetic proxy, but the geomagnetic proxy data file has not entries for the date required by the user	Either call dtm2013 directly (by providing solar proxies computed elsewhere) or update the solar proxies data file
Cannot open solar proxies file	The solar proxies file could not be opened, or an error happened while opening it	Check that the file exists. Take into account that Linux and Unix are case-sensitive. Set the full path to the file by setting the PATH_TO_F_FILE variable
I/O error while reading solar proxies file	There was an input/output error while reading the solar proxies file	The solar proxies file was successfully open, but an error rose while reading it. The file might have become corrupted, or it might have been edited. Check the file, or get a backup version. In case the error persists, report it.
Solar proxies file longer than max_lines parameter	The DTM-2013 library allocates enough memory to hold a reasonably large file. In this case, this limit has been reached.	Either remove unnecessary lines for the solar proxies file or contact support to get a version with a larger limit
Cannot open geomagnetic proxies file	The geomagnetic proxies file could not be opened, or an error happened while opening it	Check that the file exists. Take into account that Linux and Unix are case-sensitive. Set the full path to the file by setting the PATH_TO_A_FILE variable
I/O error while reading geomagnetic proxies file	There was an input/output error while reading the geomagnetic proxies file	The geomagnetic proxies file was successfully open, but an error rose while reading it. The file might have become corrupted, or it might have been edited. Check the file, or get a backup version. In case the error persists, report it.
Geomagnetic proxies file longer than max_lines parameter	The DTM-2013 library allocates enough memory to hold a reasonably large file. In this case, this limit has been reached.	Either remove unnecessary lines for the geomagnetic proxies file or contact support to get a version with a larger limit
Geomagnetic proxies file too new. The first record of this file must be 24 hrs or	In order to compute the adequate input paremeters for dtm2013, the dtm_wrapper	Either call dtm2013 directly, or get an updated version of the

more before the date	subroutine needs a record of the geomagnetic proxies which is 24 hours before the user-input date. If this is not possible, this error comes up	geomagnetic proxies file.
The provided calendar date is invalid. Please check it	The user provided dtm_wrapper with a non-existing calendar date (for example, 29/02/2001)	Check your program for the offending input
Geomagnetic proxy (am) file contains an invalid measurement. Check geomagnetic proxies file	dtm_wrapper has read a less than 0 value from the geomagnetic proxies file. Less than 0 values are used to mark not available data	Either get an upgraded version of the geomagnetic proxies file with the offending entries fixed, or call dtm2013 directly
Geomagnetic proxy (am) has a strange value (too big). Check geomagnetic proxies file	dtm_wrapper has read a value greater than 611.4 from the geomagnetic proxies file. This value is out of bounds.	Check that the input file is correct, get a newer version of the input file, or call dtm2013 directly
Unable to process configuration file	There was a problem while opening or while parsing the provided configuration file	Check that the path to the configuration file is correctly defined, and that the configuration file is correctly formed

In addition to fatal errors, warnings are also issued to the user when required. These warnings are displayed to the screen, but they do not stop program execution.

Table 7: Warning messages

Message	Explanation	Action to solve the problem
Unable to interpolate fbar. Returning a non-interpolated value	In order to compute the adequate input for dtm2013, dtm_wrapper must interpolate the solar proxies file between the user-input date and a date 24 hours later. This warning comes up when the user-input date matches the last record on the solar proxies file.	dtm_wrapper calls dtm2013 with a non-interpolated value of the fbar parameter. This fact should be taken into account when processing the results.
Suspicious value passed to a2K function. Rounded to zero	This warning arises when DTM wrapper receives an interpolated geomagnetic proxy with a near-zero negative value. (Within a threshold of 1e-7). When that happens, the value is truncated to 0.	Normally, this warning arises due to numerical roundoff errors, and can be ignored safely. However, in some cases it can be caused by incorrect geomagnetic proxies values

ANNEX B: SOFTWARE PROBLEM REPORT FORM

deimos SPACE	**Software Problem Report**	Code: ATMOP-DMS-SPR-### Date: Page: 1/1

Project Name: **Project Phase:**	**Module:** **Version:**
Opening Date:	**Originator:** **Signature:**
Closing Date:	**Responsible:** **Signature:**

Priority:	☐ Critical	☐ Urgent	☐ Routine	☐ Low
Impact:	☐ In scope	☐ Out of scope		
Decision:	☐ Correct	☐ Reject	☐ Use as is	
	☐ Waiver	☐ CCN	**No:**	

Environment description:

Problem description:

Problem analysis:

Recommended solution:

Attachments:

END OF DOCUMENT