"satellite"
version 1.0.0
User Manual

July 21, 2010

V. Eymet, LAPLACE, Université Paul Sabatier, 31062 Toulouse, France

Contents

1 Introduction

This document is intended to be used by any new user of the code "satellite" . The basic linux commands are supposed to be known (file system commands, environment variables use, basic shell scripts).

The user is free to use and modify the sources of the code. It has been writen in fortran 77, which imposes some limitations, mostly for memory management.

Any question, remark, or improvement suggestion is welcome, and should be submitted by e-mail to the author (Vincent Eymet, eymet@laplace.univ-tlse.fr).

1.1 Purpose of the code

"satellite" is intended at computing the solar radiative flux density (W/m^2) received by a satellite detector. It uses as an input the angular distribution of top of atmosphere fluxes that results from solar radiative transfer simulations that have been performed using the "planet_EMC" computation code.

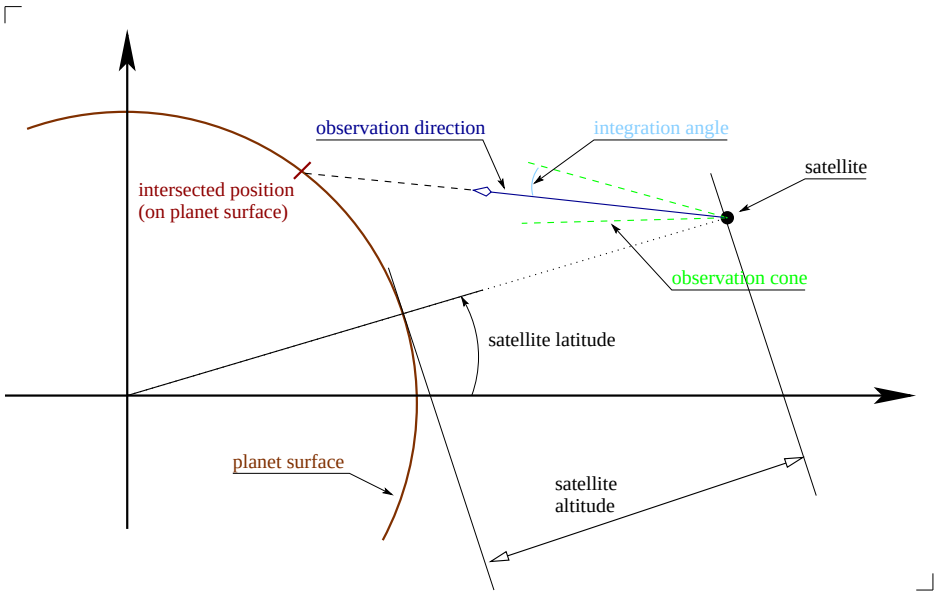

Figure 1: Basic sketch of the geometric configuration

As shown in figure 2.2, the satellite is defined by its position, the direction the detector is observing, and the detector's observation cone. Variables used for the description of the geometric configuration are described in section 2.

Results can be interpreted as an integrated radiance: "satellite" computes the solar power density (Watts per square meter) that reach the detector (surface normal to the observation direction), in a given spectral interval, within the observation cone; units is Watts per square meter of normal surface.

1.2 Informatics

Compiling the program "satellite" does not require external librairies: all source files are included in the package. "satellite" is actually a part of the "planet_EMC" project.

"satellite" is a fully parallel program. Specific parallelization instructions require having MPICH2 installed and functionnal.

2 Input data files

This section describes the input data files that are needed by the code. Example files are provided.

2.1 data.in

The "data.in" file resides into the "satellite" main folder. It contains:

- The satellite's latitude, in degrees. Latitude is defined from the equatorial plane, and can therefore be positive or negative.
- The satellite's longitude, in degrees. Longitude is defined from the planet's reference latitudinal plane.
- The satellite's altitude, in kilometers. Altitude is defined from the planet's ground level.
- The latitude and longitude of the position that is intersected by the observation direction, on the planet's ground.
- The integration angle, i.e. the angle that defines the observation cone around the observation direction.

2.2 integrated_solar_TOA_fluxes.txt

This file resides within the "data" folder of "satellite" . It is an output file of the "planet_EMC" simulation code. This file contains:

- A description of the longitude/latitude grid used for the simulation.
- A description of the spactral grid used for the simulation.
- The simulated upward solar flux density angular distribution at the top of the atmosphere, for each spectral interval.

From the solar flux angular distributions, "satellite" 's main task is to identify which angular sectors contribute to the flux that will be received by the detector, and then to compute the total flux density at the detector's position, for each spectral interval.

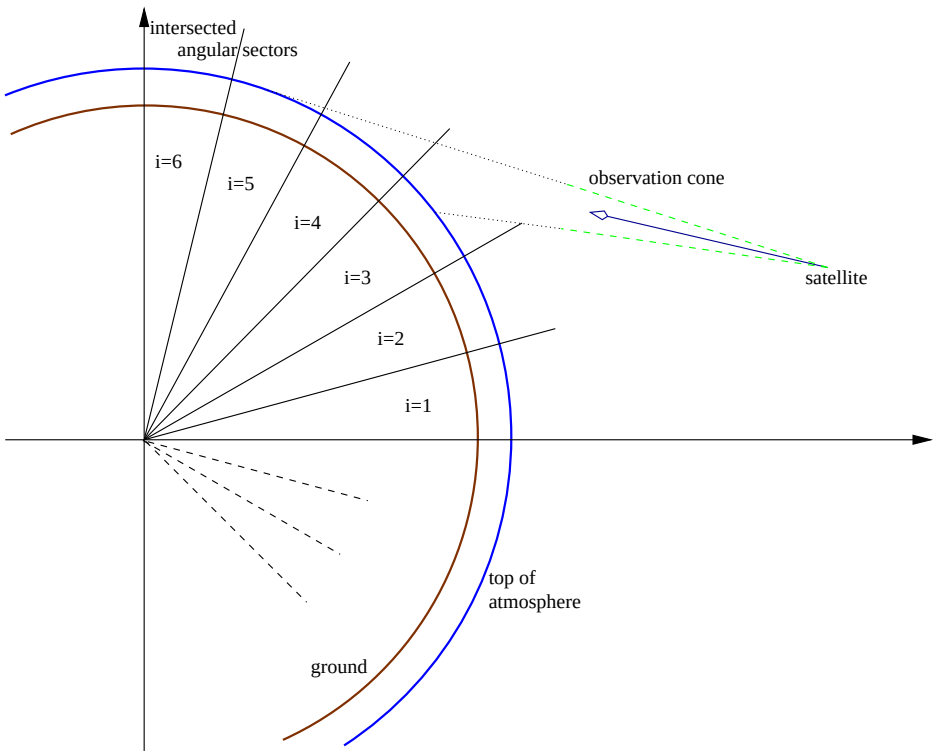

Figure 2: What is done: "satellite" first identifies which angular sectors contribute to the radiation received by the detector: in this example, only sectors 3 to 5 do send radiation into the observation cone. For each one of these angular sectors, the code has to compute the part of the surface (at top of atmosphere level) that contributes, and to integrate the angular distribution of radiation emitted into the observation cone.

3 Compiling and using the code

As mentionned in section 1, “satellite” requires MPICH version 2 to be installed, because the code has been parallelized.

3.1 Installation of MPICH

If you do not already have MPICH installed on the machine / group of machines you want to run “satellite” on, you will first have to download mpich2 from http://www.mcs.anl.gov/research/projects/mpich2 ; make sure you download version 1.0.7. or newer. Next, untar the downloaded archive, and install it on every system that will be part of your cluster:

First, you may have to set environment variables CFLAGS, FFLAGS, F90FLAGS, CXXFLAGS, F77 and F90 according to your architecture and the compilers that are installed on your system.

> export CFLAGS=”-m32” (use “-m64” on 64 bits systems)
> export FFLAGS=”-m32” (idem)
> export CXXFLAGS=”-m32” (idem)
> export F77=”ifort” (use any other compiler)
> export F90=”ifort” (idem)
> ./configure –prefix=/path/to/installation/directory
> make
> make install

Before running the MPD daemon, you must create a “.mpd.conf” file in your home folder:

> echo secretword=[secretword] » /.mpd.conf
> chmod 600 .mpd.conf

using any “secretword”.

Next, you will need to be able to connect via ssh to every other machine of your cluster, with no password request. For this, you must first create a DSA key on the machine you will run the code from :

> ssh-keygen -t dsa

leaving all fields blank (use the “enter” key to answer each question). Then you will have to add this DSA key to the list of authorized keys of every machine that will need to be accessed for computation :

> cd .ssh
> cat id_dsa.pub » authorized_keys

Finally, create the list of machines that belong to your cluster. This list must reside within the "mpd.host" file on your home folder. Each line must contain the name of the machine, by order of availability:

[host1].[domain]
[host2].[domain]
[host3].[domain]
etc

You can then try to run the MPD daemon:

> mpdboot -n [#]

with [#] the number of hosts you want to run MPD on (typically, the number of machines in your cluster). If you encounter no error, you can use command "mpdtrace" to check the number of hosts the MPD daemon is running on. This should give you the list of machines in your cluster.

3.2 Tweaking the "Makefile"

Before compiling, you will have to find out what compilation options are right for your compiler, and your machine. Open the "Makefile" file, and look at variables "FOR", "ARCH" and "OPTI". Variable "FOR" is used to specify your fortran 77 compiler. As "satellite" uses MPICH, you will most likely use the "mpif77" compilation command, that has been installed along with MPICH.

Variable "ARCH" is used to specify machine architecture. "-m486" is probably a good choice for a PC running a 32bits linux. On recent Mac computers, "i686 -m64" works. Use the documentation of your fortran compiler to find out what architecture option you can use.

Variable "OPTI" is used to specify code optimization options. The default options should be enough. Please note that you definitely must use option "-Wno-globals" for compiling parallel code.

You might also want to set variable "DEBUG" (look for its definition in the file). You can expect faster execution times if you leave it empty.

3.3 Setting up array sizes

One limitation of fortran 77 code is that you must define array sizes before compilation. Arrays sizes used by the present code are defined within the "includes/max.inc" file. You should at least look at it before compiling, and more precisely at the value of variables Ntheta_mx, Nphi_mx, Nz_mx,

Nb_clouds_mx, Nb_gas_mx and Nq_mx. Please note you should never modify the value of variable Nmat_mx (3).

3.4 Compilation

Once you checked compilation options and array size definitions, you can use the following command in order to compile the executable file:

> make all

If compilation fails, use the compiler error message to determine what went wrong. The most probable error causes are: a bad definition of architecture compilation option, or an inappropriate value in code optimization options.

If you ever need to modify the source files (in directory "source"), you can quickly recompile the code using "make all" again. This will only recompile the modified source files, and link objects files in order to produce the new executable file.

If you have to modify the value of any variable defined in includes files (directory "includes"), you will have to recompile the whole code from scratch. Use the following command to erase all objects files:

> make clean all

and then recompile them properly with "make all".

Odd errors may happen if you modify an include file and then recompile using only the "make all" command (old value of the modified variable will remain in the unchanged object files).

3.5 Running the code using MPICH

Once everything is installed and the executable file "satellite.exe" file has been compiled, you can try to run a computation. I would recommend that, for the first time, you run "satellite" using the provided example data files.

Use the following command to run the code:

> mpirun -np [#] satellite.exe

with [#] the number of processes that have to run.

Because communication times are small compared to computation times in "satellite" , it is a good idea to chose a number of processes equal to the number of (physical) processors of your cluster, plus one. One process, the master process, is dispatching computational loads to every other processes (slave processes), and gathering results from them. It does not require any

significant CPU time, therefore it is OK to have a number of slave processes equal to the number of processors, so that each slave process can use a processor (or each processor will have only one slave process running on it).

In practice, if your cluster is composed of n processors, you can use:

> mpirun -np n+1 satellite.exe

4 Results

Results are recorded into file "results/detector.txt". It contains:

- The number of spectral intervals
- Their limits in μm.
- The solar flux density (W/m^2) received by the detector, in each spectral interval, along with the associated numerical uncertainties.
- The total solar flux density (W/m^2) received by the detector and its uncertainty.