1 Introduction

This guide starts by providing all essential steps to get started with the Dutch Atmospheric Large Eddy Simulation (DALES) model version 4.1 on your own computer, from compiling the source code to running your first case. This description has been adapted from an early description of DALES 3.1 by Thijs Heus, Chiel van Heerwaarden and Johan van der Dussen.
DALES is written in Fortran 90. This language is still frequently used in the scientific communities of geophysics and astronomy basically because in the 1950s many early scientific programs were developed in Fortran. For example, numerical weather prediction (NWP) was one of the first intensive computer applications, and many current state-of-the-art NWP models have their roots in these early Fortran versions. DALES can be run using multiple CPUs, and it has the option to write output in the so-called NCDF format. This however requires quite some software packages, which are often conveniently available in Linux environments. To perform numerical simulations you will have to install the VirtualBox[1] and the virtual Linux environment including DALES4.1[2]. Sometimes you want to install a package which requires the root password: in this application it has been set to `dales`. Some convenient Linux commands will be summarized in Appendix A. In case you happen to operate Ubuntu on your computer you can also find a recipe to install Dales on Ubuntu 16.04 in Appendix B.

[1]available for downloading from `https://www.virtualbox.org`. In case you are already using Ubuntu or OpenSuSe you can install VirtualBox from `zypper install virtualbox` or `sudo apt-get install virtualbox`, respectively.

[2]available for downloading from `http://www.srderoode.nl/Teaching/LES_course/Dales-4.1_3.ova`

2 Running DALES in the VirtualBox

In the virtual Linux environment you will see eight icons at the top left of the screen. Open a terminal by clicking on the terminal icon. Type `ls` to see which files and directories are present. The source code of DALES is stored in `dales/src`. You can go to this directory by typing `cd dales/src`. If you want to go back to your home folder simply type `cd`. If you want to read the contents of a file type `atom filename.f90`. If you open the file `program.f90`[3] you will learn about the skeleton of the DALES model in the sense that it calls many subroutines. Note that every `*f90` routine starts with a brief description. Since DALES is a collaboration between different institutions (Heus et al., 2010) it also states the authors of the subroutines, which can be handy in case one has specific questions about the code. For example, subroutines including the effect of plants on evaporation and atmospheric chemistry have been developed by our colleagues from Wageningen University. The bulk of the routines relevant to clouds have been written (e.g. precipitation, moist thermodynamics) or incorporated (radiative transfer) by team members from the TU Delft and KNMI.

2.1 Make an executable of the code

To make an executable of the model code create a new directory by typing

```
mkdir dales.build
```

Go to this directory and type

`cmake ~/dales`[4].

You will see that the directory contains some directories. In addition the script `Makefile` has been generated. It compiles the Fortran file and makes an executable. Execute this file by typing

`make`[5].

The `dales4` model has been put in the directory `~\dales.build\src`.

2.2 Perform a simulation with DALES

2.2.1 Input files

Some examples of input files have been posted on the internet. As you will often use the same input files for different experiments it is therefore convenient to put them in a separate directory. To this end create the directory `~/Cases/CBL`, go to this subdirectory by typing `cd ~/Cases/CBL` and download the files `prof.inp.cbl_fixed_grad`, `lscale.inp.cbl_fixed_grad` and `namoptions.coarse` by typing, as an example,

`wget www.srderoode.nl/Teaching/LES_course/CBL/namoptions.coarse`[6]

[3]Linux tip: type `atom pr` followed by a click on the 'tab' of your keyboard. You will see that the system automatically finishes the filename.

[4]At some computers, like Macs, the $\sim$ symbol can be obtained from the Shift-§-button

[5]If your computer has multiple CPUs you can type `make -j n` to use a number of n CPUs, with $1 \leq n \leq N$ an integer that should be smaller than the number N of CPUs on your machine. If you type `make -j` the number of processors is automatically set to N. Be careful if you do this on a computer cluster as your task will use all CPUs which may lead to a significant speed reduction of the other running processes. You may loose friends if you do so.

[6]You can obtain the history of your previous commands by using the upwards pointing arrow at your keyboard.

We will perform various experiments of the Convective Boundary Layer (CBL) and we will store the results in the directory `~/Experiments/CBL` (create it if it is not present). We wish to collect the output from our first experiment in the subdirectory `~/Experiments/CBL/H001`. Before we can do a run, we have to copy some obligatory input files to this subdirectory. Go to the subdirectory `H001` (`cd ~/Experiments/CBL/H001`) and copy the following files

```
cp ~/Cases/CBL/prof.inp.cbl_fixed_grad prof.inp.001
cp ~/Cases/CBL/lscale.inp.cbl_fixed_grad lscale.inp.001
cp ~/Cases/CBL/namoptions.coarse namoptions
cp ~/dales.build/src/dales4 .
```

In the last command the dot indicates that the name of the copied file remains the same. Always check if the experiment number `iexpnr` in the second line of `namoptions` is the same as the number of the subdirectory, otherwise change to `iexpnr = 001`, for example with aid of `atom namoptions`. The `namoptions` file tells the model to do a simulation on a horizontal domain size of `xsize=ysize`=1.6 km using `itot=jtot`=32 grid points that lasts only `runtime`= 300 s with a time step of `dtmax`=1 s . Except for some testing purposes such a small number of grid points in the horizontal plane (`itot=jtot`=32) is actually never used in practice. However, here we find it convenient to quickly see whether the model runs smoothly.

2.2.2 Execute a simulation

Start a model run by typing

```
mpirun -np 2 dales4 |tee output.001.
```

The command tells the computer to use 2 CPUs and to send information about the run both to screen and an output file. The last line presents the total duration of the simulation (W =`TOTAL wall time`). The model experiment can be sped up if you have the opportunity to use more than 2 CPUs.

Exercise 1: Applying LES to the entire Earth?

1a. Use the information given above to calculate the computation time for one grid point per computational time step per CPU. The total number of grid points N can be found in `namoptions` and is equal to
$N_p =$ `itot`$\cdot$ `jtot`$\cdot$ `kmax`
The number of time steps is equal to the ratio $N_t =$ `runtime/dtmax`. With aid of the Wall clock time W the computational time per time step can be calculated from $t^* = \frac{W N_{\mathrm{CPU}}}{N_t N_p}$. We have to multiply by N_{CPU} as the wall clock time is related to its reciprocal (as we assume that a number of n CPUs reduce the wall clock time by about a factor n).
1b. Use the radius of the Earth (6400 km) to compute the global surface area. Compute the total time needed to repeat simulation `H001` for the entire Earth and for a period of 24 hours using only one CPU.
1c. Suppose that you can make a network consisting of the same CPUs as you are using, and that the calculation time will be reduced inversely proportional to the number of CPUs. How many CPUs are needed to finish the calculation within 24 hours?