Getting started with UNIX & DALES

Pim van Dorp
p.a.vandorp@student.tudelft.nl

April 20, 2016

UNDER CONSTRUCTION
Latest version available at `http://bit.ly/1kFbdUQ`

1 Introduction

This guide starts by providing all essential steps to get started with DALES 4.1, from obtaining the source code to running your first case. The second part consists of some tips on how to begin modifying and extending DALES to suite your own needs. This guide is mostly written from my experiences obtained during my master research project, in which I have worked a lot with DALES. Also, some parts are taken from an *Introduction to DALES v3.1*, written by Thijs Heus, Chiel van Heerwaarden and Johan van der Dussen.

2 Editing text in the terminal

In setting up and using DALES, often viewing and editing text files in the terminal is required. Nano and Vim are two of the many options. Nano is quite straightforward, but Vim takes some getting used to. However, after mastering the basics, Vim can already be a very powerful editor and save you a lot of time. Nano comes pre-installed with Ubuntu and Vim can be installed by

```
sudo apt-get install vim
```

The built-in tutorial can be used to learn the basics and can be easily accessed by typing

```
vimtutor
```

Nano and Vim are also available on the AtmosPhys network. Personally I use Vim, but from now on you can replace Vim in any command with the name of your text editor of choice.

3 Getting access to the local network

There are basically two options to get access to the local network of the Atmospheric Physics (AtmosPhys) group: login to one of the VR-Lab computers (for example inside the GRS student room, CiTG 1.33.1) or connect from a remote computer (e.g. your laptop or desktop) using the SSH protocol. Although there are Windows SSH clients around, it is a lot more convenient to use a (free) UNIX operating system

(Ubuntu, openSUSE, etc.) which can be easily installed alongside with Windows on virtually any computer. Note that Apple's operating system OS X is also UNIX based.

3.1 Setting up a SSH connection

Open a terminal and run the following command

```
ssh -Y username@bastion.grs.tudelft.nl
```

When prompted type your password and that's it, your now on the local network. Next, connect to your own computer at the group

```
ssh -Y computername
```

or, when you don't know the name of your personal computer, connect to Cirrus

```
ssh -Y cirrus
```

Generating a SSH key pair prevents having to type your password at every login. Open a new terminal on your own computer and do not login to the local network. Instead type

```
ssh-keygen -t rsa
```

when asked for a directory use the default (press Enter) and press Enter twice again when asked for a passphrase to set an empty, or no, passphrase. Next, create a new directory on the AtmosPhys network

```
ssh username@bastion.grs.tudelft.nl mkdir -p .ssh
```

When prompted type your password. Finally, add the locally generated key to the `authorized_keys` file:

```
cat .ssh/id_rsa.pub | ssh username@bastion.grs.tudelft.nl ...
'cat >> .ssh/authorized_keys'
```

Again, when prompted type your password. From now on you can login to the AtmosPhys network without a password by typing

```
ssh -Y username@bastion.grs.tudelft.nl
```

To make the login process even easier you might want to add an alias to the command above to your `.bashrc` file. First, start a new terminal and open this file

```
vim .bashrc
```

Next, add the following line to the bottom of the file

```
alias bastion='ssh -X  bastion.grs.tudelft.nl'
```

After you saved and closed the file, type

```
source .bashrc
```

Now, when typing

```
bastion
```

in a local terminal you will directly connect to the AtmosPhys network.

4 Obtaining the source code

Login to the AtmosPhys network and connect to your personal computer or Cirrus (see section 3). Type

```
pwd
```

to make sure the current directory is `/home/username`. Now obtain DALES by typing

```
git clone git://github.com/dalesteam/dales.git
```

Move to the (newly created) DALES folder:

```
cd dales
```

We can switch to branch 4.1 by

```
git checkout 4.1
```

DALES 4.1 is successfully obtained if

```
git status
```

reads `On branch 4.1`.

5 Compiling DALES

5.1 Loading modules

First we need to load the right modules by adding them to the `.bashrc` file. Open this file

```
vim .bashrc
```

and add to bottom of the file the following lines

```
module load netcdf/gcc
module load openmpi/gcc
```

Save and close and source the `.bashrc` file

```
source .bashrc
```

5.2 Building DALES

To keep track of different DALES builds, experiments and cases, we will use the folder structure as suggested by Stephan de Roode. Start by creating a directory that will hold our first DALES build

```
mkdir -p ~/Les/Les_versions/Dales4.1/test_version/dales.build
```

Note that ~ is equivalent to `/home/username` and `-p` makes sure all parent directories are created. Navigate to the build directory

```
cd ~/Les/Les_versions/Dales4.1/test_version/dales.build/
```

and type

```
cmake ~/dales
```

Now we can compile DALES 4.1 by typing

```
make
```

6 Preparing the first DALES run

As of April 2016 all DALES simulations should be run from a temporal ('scratch') folder on the network. The contents of this folder will be deleted after some time to prevent old experiments from clogging up disk space. The first step of performing a DALES run is hence to make a folder for the experiment inside your personal scratch folder:

```
mkdir -p /net/scratch/username/Experiments/experiment_name/experiment_number
```

A DALES run is initialized from a so-called jobscript (section 6.1), after which it reads options from a `namoptions` file (section 6.2) in the run directory we have just created.

Furthermore, files containing the initial vertical profiles of several quantities and the large scale tendencies ('case files') are required. Since the case files will probably be used in multiple runs, they are best kept in a separate folder. The home folder is not accessible from the local cluster so it is recommend to keep the case files on `labdata` as well:

```
mkdir -p /net/labdata/username/Cases/
```

The location of the executable should also be accessible from the cluster. Therefore it might be convenient to keep a copy of the folder with the DALES build on `labdata`:

```
mkdir /net/labdata/username/Les_versions/
rsync -aP /home/username/Les/Les_versions/ /net/labdata/username/Les_versions/
```

6.1 Jobscript

Below a simple example job script is shown with a short description of every command. This job script is sufficient to run a DALES simulation on the local cluster (Cirrus). Note that this script assumes the folder structure used in this document and some modifications may be required.

```
#!/bin/bash
#$ -pe mpi 8
#$ -q gpcpu
#$ -l h_rt=500000

. /etc/bash.bashrc.local
module load openmpi/gcc
module load netcdf/gcc

expnr=001
Exp_dir=$(pwd)
Les_dir=/net/labdata/pim/Les_versions/Dales4.1/test_version/dales.build/src
Case_dir=/net/labdata/pim/Cases/case_title/case_subtitle

cp $Exp_dir/namoptions.$expnr $Exp_dir/namoptions
cp $Case_dir/prof.inp$case_ext $Exp_dir/prof.inp.$expnr
cp $Case_dir/lscale.inp$case_ext $Exp_dir/lscale.inp.$expnr

mkdir -p $Exp_dir/INIT
```

```
cp $Les_dir/dales4 .

mpirun dales4

mv init* $Exp_dir/INIT
```

6.2 Namoptions

Some of the options in `namoptions` are required, while most have default values and are thus optional. The options are divided into different so-called namelists. The standard package contains four of them: run, domain, physics and dynamics. The options belonging to a namelist must be preceded by an ampersand (&) followed directly by the name of the namelist in capitals. There should be a backslash on the line below the last option of a namelist to close it, e.g.:

```
&NAMELIST
option1 = value1
option2 = value2
/
```

6.2.1 Setting the number of cpu's

The total number of cpu's is set on the second line of the job file (8 in this case): `#$ -pe mpi 8`. As of DALES 4.1, the computational domain can be split up in both the x- and y-direction. Each part of the domain is then handled by a different cpu. The number of cpu's in the x- and y-direction are set in `namoptions`, within the `RUN` namelist:

```
&RUN
iexpnr      =  007
lwarmstart  =  .false.
...
nprocx      =  2
nprocy      =  4
/
```

When `nprocx` and `nprocy` are not defined in `namoptions`, DALES uses the default configuration, i.e. `nprocx=1` and `nprocy=0`. This setting corresponds to the old 1D 'slab' parallelization of the previous versions.

If `nprocx` and `nprocy` are both set to zero, DALES tries to find a suitable configuration. This is recommended when a large number of processors is used. In other cases it may be best to use the default, since the code is not symmetrical in `nprocx` and `nprocy`. For example, setting `nprocx=0` and `nprocy=1` will be slower than the default configuration (`nprocx=1` and `nprocy=0`).

6.2.2 Surface parametrization

DALES can calculate surface conditions via different methods. A specific method can be chosen by setting the `isurf` option in the `surface` namelist, to an integer between one and four. Each value requires different input, see table 1.

The liquid water potential temperature flux at the surface can be prescribed by setting a specific value of `wtsurf` in the `namoptions` file, or in the `lsflux.inp` file in case of a time-dependent flux. The temperature flux has to be provided in units

Table 1: Overview of the different surface parametrization options.

isurf	Required inputs	Additional information
1	z0	thls is constant, surface fluxes are calculated
2	z0	thls is constant, surface fluxes are calculated
3	ustin, wtsurf, wqsurf, z0	
4	wtsurf, wqsurf, z0	ustin is calculated

Km/s. Hence, when we want to prescribe a certain heat flux q in W/m^2, we have to convert it to the right dimension by first dividing by the density and heat capacity of air.

6.3 Case files

6.3.1 prof.inp

6.3.2 lscale.inp

6.3.3 lsflux.inp

The `lsflux.inp` file is only required when `ltimedep` is enabled and allows for time-dependent large scale forcings.

6.3.4 scalar.inp

6.3.5 lsfluxsv.inp

7 Running DALES

After creating the jobscript, namoptions and all case dependent input files, move to the run folder

```
cd /net/scratch/username/Experiments/experiment_name/experiment_number
```

and type

```
qsub job.001
```

You can check the status of your run by

```
qstat
```

8 Output files

It is important to first copy/move any output you want to store from the scratch folder to the 'labdata' folder. Unlike in the scratch folder, files in the labdata folder will not be automatically deleted. To keep things organized it might be preferable to maintain the same folder structure in labdata. First make the folder:

```
mkdir -p /net/labdata/username/Experiments/experiment_name/experiment_number
```

And move any files you want to keep to this folder by using the following command from inside the run directory:

```
mv file1 file2 file3 /net/labdata/username/
```

8.1 Job output and error files

8.2 Fielddumps

The fielddump files are labeled as follows: `fielddump.001.003.007.nc`. The first (`001`) and second (`003`) integer refer, respectively, to the identification of nth cpu in the x- and y-direction. The cpu's are labeled with respect to the origin, i.e. `fielddump.000.000.007.nc` contains the origin. The third integer (`007`) indicates the experiment number.

8.3 Profiles.nc

The `profiles.nc` file contains slab averaged values (in the x, y-plane) of almost all relevant quantities, conveniently stored in a single `.nc` file. The variables are given as a function of time and either the half or full level height (see section 10.1). See the section about post-processing for more information on working with `.nc` files. The table below shows an overview of the most important quantities that are stored in `profiles.nc`.

Table 2: Overview of the most important quantities stored in the `profiles.nc` output file.

Quantity	Symbol	Units	Name in DALES	Function of
Pressure	P	Pa	`presh`	z^t, t
Liq. water pot. temp.	θ_l	K	`thl`	z^t, t
Vir. pot. temp.	θ_v	K	`thv`	z^t, t
Tot. water spec. humidity	q_t	kg/kg	`qt`	z^t, t
Liq. water spec. humidity	q_l	kg/kg	`ql`	z^t, t
Tot. momentum flux	$\overline{uw}$	$\mathrm{m}^2/\mathrm{s}^2$	`uwt`	z^m, t
Tot. momentum flux	$\overline{vw}$	$\mathrm{m}^2/\mathrm{s}^2$	`vwt`	z^m, t
Res. vertical velocity var.	$\overline{w'w'}$	$\mathrm{m}^2/\mathrm{s}^2$	`w2r`	z^m, t
Tot. buoyancy flux	$\overline{\theta_v w}$	Km/s	`wthvt`	z^m, t
SFS-TKE	e	$\mathrm{m}^2/\mathrm{s}^2$	`w2s`	z^m, t

8.4 Tmser.nc

The `tmser.nc` file contains time series of several derived variables calculated by DALES.

Table 3: Overview of the most important quantities stored in the `tmser.nc` output file.

Quantity	Symbol	Units	Name in DALES	Function of
Cloud-base height		m	`zb`	t
Av. cloud-top height		m	`zc_av`	t
Boundary layer height	z_i	m	`zi`	t
Liquid water path	LWP	$\mathrm{kg/m}^2$	`lwp_bar`	t
Obukhov length	L	m	`obukh`	t

9 Modifying and extending DALES

The DALES source code, written in Fortran, contains numerous modules and can be found at

```
cd ~/dales/src
```

From the file `program.f90` all subroutines are called, and it contains the main time loop. These subroutines are written in the different modules.

9.1 Basics of Git

A large part of the fun is of course to modify the existing DALES source code or to add your own modules. Probably the best way to do this is to start your own branch using Git. In this branch you can edit and extend the DALES code, but you will always be able to return to the 'clean' DALES 4.1 if things don't work out as planned, or if you just want to perform a clean run.

Working with Git has a large number of advantages, but requires some introduction. A simple and helpful guide can be found at `http://gitimmersion.com`.

Creating a new (local) branch is straightforward

```
git checkout -b name_of_your_branch
```

Now, a list of the available branches can be printed

```
git branch
```

and it should read

```
4.1
* name_of_your_branch
master
```

where the asterisk indicates the branch you are currently working on. After editing a file or creating a new file, the change can be added to Git

```
git add filename
```

or to add all changes

```
git add .
```

To commit changes type

```
git commit -m 'commit_message'
```

Withing the commit message you can provide some comments on the changes. More on adding and committing changes: `http://gitref.org/basic/`.

Any changes made to the source code will only exist in this newly created branch, and you can return anytime to the unmodified version of DALES 4.1 by

```
git checkout 4.1
```

9.2 Compiling your modified DALES version

After making any changes to the source code, DALES should be compiled again. To have direct access to all your different DALES builds, it may be convenient to do this in a new build directory

```
mkdir -p ~/Les/Les_versions/Dales4.1/version_name/dales.build
```

and follow the steps described in section 5. Now, after making changes within modules it is sufficient to only perform

```
make -j4
```

in the build directory. When, however, new modules are added it is required to perform all steps given in section 5 again, to make sure all paths are included. Finally, any previously compiled files from the build directory can be deleted by

```
make clean
```

9.3 Test runs

Successfully modifying DALES requires a lot of testing. Short test runs to check if your code is working properly can be prioritized in the job scheduler by selecting a run time of less than one hour in the jobscript.

10 Additional information

10.1 The computational grid

Like any CFD code, DALES distinguishes half and full levels. Quantities like temperature, humidity and velocity are in general defined at full levels,

$$
\begin{aligned}
x^t_{i,j,k} &\in \{\Delta x/2, 3\Delta x/2, \ldots, D_x - \Delta x/2\} \\
y^t_{i,j,k} &\in \{\Delta y/2, 3\Delta y/2, \ldots, D_y - \Delta y/2\} \\
z^t_{i,j,k} &\in \{\Delta z/2, 3\Delta z/2, \ldots, D_z - \Delta z/2\},
\end{aligned}
$$

and fluxes at half levels,

$$
\begin{aligned}
x^m_{i,j,k} &\in \{0, \Delta x, \ldots, D_x - \Delta x\} \\
y^m_{i,j,k} &\in \{0, \Delta y, \ldots, D_y - \Delta y\} \\
z^m_{i,j,k} &\in \{0, \Delta z, \ldots, D_z - \Delta z\},
\end{aligned}
$$

where D indicates the domain size in each direction. Figure 1 illustrates the grid layout.

In the vertical direction, a non-equidistant grid can be prescribed, i.e. $\Delta z = \Delta z_k$. In general, the half level thickness $\Delta z_{\mathrm{h},k}$ then needs to be distinguished from the full level thickness $\Delta z_{\mathrm{f},k}$.

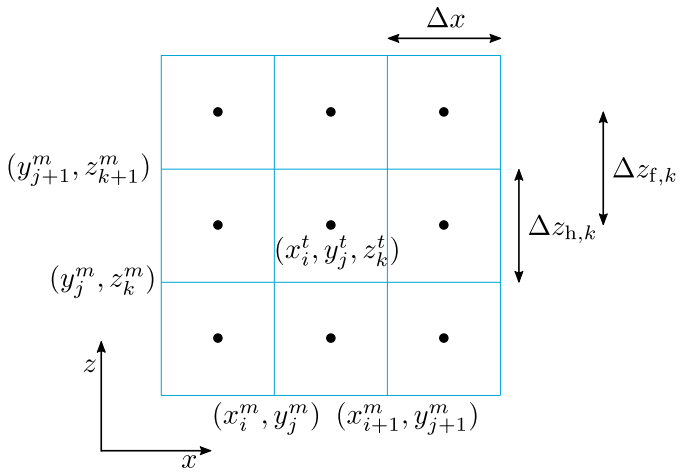

Figure 1: Schematic view of the computational grid in the x, z-plane.

11 Pre- and post-processing

11.1 Working with NETCDF files

11.2 Example programs

Example programs for generating the DALES input files, written in Python:
`https://github.com/pimvandorp/dales4.1inputgen.git`
Example programs for analyzing and plotting fielddump files, written in Python:
`https://github.com/pimvandorp/dales4.1analyzedata.git`

12 Links

12.1 DALES

Introduction to DALES v3.1:
`http://www.srderoode.nl/Students/Dales_manual.pdf`
Formulation of the Dutch Atmospheric Large-Eddy Simulation (DALES) and overview of its applications:
`http://www.geosci-model-dev.net/3/415/2010/gmd-3-415-2010.pdf`

12.2 UNIX, Ubuntu, etc.

Linux command line cheat sheet:
`http://bit.ly/1imoOPS`
Official Ubuntu documentation:
`https://help.ubuntu.com/`
Using the terminal:
`https://help.ubuntu.com/community/UsingTheTerminal`

12.3 Vim

Information about Vim:
`http://vimdoc.sourceforge.net/`
`http://www.truth.sk/vim/vimbook-OPL.pdf`
Configuring Vim:
`https://amix.dk/vim/vimrc.html`

12.4 Git

Basics of Git:
`http://gitref.org/basic/`
Git tutorial:
`http://gitimmersion.com`