TerrSysMP-PDAF
User Guide

Version 1.0

Wolfgang Kurtz[1,2]

[1]Forschungszentrum Jülich GmbH, Institute for Bio- and Geosciences (IBG-3): Agrosphere, Jülich, Germany

[2]Centre for High-Performance Scientific Computing in Terrestrial Systems, HPSC-TerrSys, Geoverbund ABC/J, Jülich, Germany

February 22, 2016

Contents

1 Installing TerrSysMP-PDAF **3**
- 1.1 Obtaining the code . . . 3
- 1.2 Code structure . . . 3
- 1.3 Configuring TerrSysMP . . . 4
- 1.4 Patching TerrSysMP . . . 5
- 1.5 Compiling TerrSysMP . . . 5
- 1.6 Compiling PDAF . . . 7
- 1.7 Compiling TerrSysMP-PDAF . . . 7

2 Running TerrSysMP-PDAF **9**
- 2.1 Input files for CLM . . . 9
- 2.2 Input files for ParFlow . . . 9
- 2.3 Input files for OASIS-MCT . . . 10
- 2.4 Control file `enkfpf.par` . . . 10
- 2.5 Observation files . . . 13
- 2.6 Command line options . . . 15

3 Relevant publications **17**

Bibliography **18**

1 Installing TerrSysMP-PDAF

1.1 Obtaining the code

TerrSysMP

The source code of TerrSysMP can be obtained from a git-repository hosted by the Meteorological Institute of the University of Bonn (registration required): https://git.meteo.uni-bonn.de/
This git-repository provides wiki pages on how to download, configure and install TerrSysMP. The latter two points are also explained in this manual in Sections 1.3 and 1.5 for convenience. Additionally, a few modifications need to be done for TerrSysMP in order to work within TerrSysMP-PDAF (see Section 1.4). The availability of TerrSysMP from the git repository will soon be replaced by a release version of TerrSysMP which will be available from the home page of the Centre for High-Performance Scientific Computing in Terrestrial Systems (HPSC-TerrSys): www.hpsc-terrsys.de

PDAF

The source code of PDAF is available from the following web page (registration required): http://pdaf.awi.de/trac/wiki
Install instructions are provided in the README files included in the downloadable archives. The installation procedure is also described in Section 1.6 in this user guide. The current version of TerrSysMP-PDAF is build with PDAF version 1.10.

All the codes should ideally be placed into one installation directory which is termed `$SVAROOT` in the following.

1.2 Code structure

After sucessfully downloading the different parts of TerrSysMP-PDAF, the following directory structure should be present in the install directory `$SVAROOT`:

```
$SVAROOT/bldsva
$SVAROOT/clm
$SVAROOT/parflow
$SVAROOT/oasis3
$SVAROOT/PDAF-D_V1.10
$SVAROOT/tsmp-pdaf
```

The `bldsva` directory includes the scripts for configuring and compiling TerrSysMP. The directories `clm`, `parflow` and `oasis3` contain the model source codes of the respective TerrSysMP model components. The directory `PDAF-D_v1.10` contains the PDAF source code. Finally, directory `tsmp-pdaf` contains install scripts and the source code for TerrSysMP-PDAF. `tsmp-pdaf` has the following internal structure:

```
$SVAROOT/tsmp-pdaf/bin
$SVAROOT/tsmp-pdaf/libs
$SVAROOT/tsmp-pdaf/install
$SVAROOT/tsmp-pdaf/src
```

`bin` and `libs` contain the finally compiled model executable and the model libraries. `install` contains the configuration and building scripts for TerrSysMP-PDAF and `src` contains the TerrSysMP-PDAF source code.

1.3 Configuring TerrSysMP

As a first step, TerrSysMP needs to be configured for the specific compiler suite (Gnu, Intel, IBM) and for the desired OASIS coupling scheme. This is done by executing the configure script `configure_machine` in the `bldsva` directory:

```
cd $SVAROOT/bldsva
./configure_machine
```

A command line prompt will appear which lets you choose the desired compiler:

```
Supported platforms. Select your machine. (This patch is
    irriversible)
  0 exit
  1 Linux x86_64, GNU compilers
  2 Linux x86_64, Intel compilers
  3 IBM BG/Q (JUQUEEN), IBM XL compilers
```

Choose your compiler suite (Gnu compilers recommended) by typing the corresponding number. Intel is the default compiler for the building procedure. For the other two options (Gnu and IBM) a patch is applied to TerrSysMP so that the respective compiler can be used. After choosing the compiler, the user can choose which version of the OASIS coupler should be used:

```
Would you also like to ... (This patches are irriversible but you
    can also do them later)
  1 patch Oasis3-MCT
  2 patch Oasis3-MCT + parallel CLM coupling
  3 no Thanks
Enter your selection [1-3]:
```

For TerrSysMP-PDAF, option 1 or 2 needs to be chosen. In both cases, the source code of OASIS (coupler works as separate executable) will be replaced by OASIS-MCT (coupler is a library function). Option 2 is recommended here, because it provides a faster coupling scheme between CLM and ParFlow.

1.4 Patching TerrSysMP

In order to make TerrSysMP work with the TerrSysMP-PDAF framework, some minor changes in the source code of ParFlow and OASIS-MCT and in the build script of TerrSysMP are necessary. The necessary configure script is:
`$SVAROOT/tsmp-pdaf/install/configure_machine_pdaf_with_terrsysmp.sh`
The header information in this script needs to be modified according to the needs of the user:

```
########################## user setup ##########################
export COUPLE_TYPE=terrsysmp
export SVAROOT=$HOME/terrsysmp-pdaf
################################################################
```

The variable `COUPLE_TYPE` defines which TerrSysMP models are used in TerrSysMP-PDAF. Three configurations are currently possible:

- `terrsysmp`
- `parflow_stand_alone`
- `clm_stand_alone`

With `terrsysmp` the data assimilation is run with CLM and ParFlow coupled via OASIS-MCT. With the other two option only ParFlow (`parflow_standalone`) or only CLM (`clm_stand_alone`) is used in TerrSysMP-PDAF. The variable `SVAROOT` specifies the installation directory of TerrSysMP.
After settings this variables appropriately, the TerrSysMP-patch for usage in TerrSysMP-PDAF can be applied:

```
cd $SVAROOT/tsmp-pdaf/install
./configure_machine_pdaf_with_terrsysmp.sh
```

1.5 Compiling TerrSysMP

After TerrSysMP is configured and patched for data assimilation, it can be compiled with the build script `$SVAROOT/bldsva/build_oas3`. Several settings in the 'USER SETTINGS' section of this build script need to be adapted to the specific computational environment:

```
setenv sva_compiler   Gnu
setenv MPIDIR         /usr
```

```
setenv NETCDF         /usr
setenv HYPRE_DIR      /usr
setenv SILO_DIR       /usr
setenv SVAROOT        $HOME/terrsysmp-pdaf
```

The options for variable `sva_compiler` are `Gnu`, `Intel` and `Ibm`. The choice of this variable must be in correspondence with the choice in the configuration of TerrSysMP (see Section 1.3). The variables `MPIDIR`, `NETCDF`, `HYPRE_DIR` and `SILO_DIR` specify the paths to the mpi, netcdf, hypre and silo installations on the system. In the specified paths, a `lib` and a `include` directory with the corresponding header files and libraries must be present, e.g., if netcdf header files are located in `/usr/netcdf/include` and NetCDF libraries are installed in `/usr/netcdf/lib`, the variable `NETCDF` should be set to `/usr/netcdf`.
The build script for TerrSysMP is executed as follows:

```
cd $SVAROOT/bldsva
./build_oas3
```

After executing the build script, the user will be asked for the model configuration of TerrSysMP:

```
SVA Configuration: TR32/Z4
0 Only COSMO
1 Only CLM
2 Only ParFlow
3 COSMO + CLM
4 CLM + ParFlow
5 COSMO + CLM + ParFlow

Enter your selection [0-5]:
```

Enter 1 (CLM standalone), 2 (ParFlow standalone) or 4 (CLM and ParFlow coupled via OASIS-MCT). All other options are currently not supported by TerrSysMP-PDAF. After the choice of model configuration the user is asked, whether the source/build directories should be cleaned before compilation:

```
Clean Compile (0/1) :
```

A clean compilation (1) is recommended.
After the build script successfully finished, the model libraries for CLM, ParFlow and OASIS-MCT (depending on the model configuration) should be present in the directory `$SVAROOT/tsmp-pdaf/libs`. For compile option 4 (CLM+ParFlow), `$SVAROOT/tsmp-pdaf/libs` should contain the following libraries:

```
libclm.a
libamps.a libamps_common.a libkinsol.a libparflow.a
libmct.a libmpeu.a libpsmile.MPI1.a libscrip.a
```

1.6 Compiling PDAF

A detailed description of the installation procedure for PDAF is given in the file `README` in the PDAF distribution. As a short summary, system specific settings have to be set in a header file `*.h`. Examples of such header files can be found in the directory:

```
$SVAROOT/PDAF-D_V1.10/make.arch
```

In such a header file, particularly the settings for compilers and the path to the lapack library have to be set. Before PDAF can be compiled via the command `make` in the directory `$SVAROOT/PDAF-D_v1.10/src`, the two environmental variables `PDAF_DIR` and `PDAF_ARCH` have to be set:

```
export PDAF_DIR=$SVAROOT/PDAF-D_V1.10
export PDAF_ARCH=<yourarchitecture>
```

Here, `<yourarchitecture>` corresponds to the chosen header file, i.e., the header file containing the system specific settings should be present as:

```
$PDAF_ARCH/make.arch/<yourarchitecture>.h
```

1.7 Compiling TerrSysMP-PDAF

Before TerrSysMP-PDAF can be compiled, the enviromental variables `PDAF_DIR` and `PDAF_ARCH` also need to be set. Two additional header files (following the naming convention of `PDAF_ARCH` also need to be present in the directories `$SVAROOT/tsmp-pdaf/src/model` and `$SVAROOT/tsmp-pdaf/src/model`. These additional header file contain, e.g., information on compiler settings, location of the netCDF installation, etc. In total, three header files with the same name need to be present in:

```
$SVAROOT/PDAF-D_V1.10/make.arch
$SVAROOT/tsmp-pdaf/src/model
$SVAROOT/tsmp-pdaf/src/framework
```

Note that the content of these files is different. An example is given in the header file `linux_gfortran_openmpi.h` present in these three directories.
After the header files are properly configured and the necessary environmental variables are set, the build script `build_pdaf_model.sh` has to be configured. This build script can be found in the directory:

```
$SVAROOT/tsmp-pdaf/install/
```

The user specific settings at the beginning of the file are as follows:

```
################## USER DEFINE SECTION ##########################
export COUPLE_TYPE=clm_stand_alone
export PDAF_DIR=$HOME/terrsysmp-pdaf/PDAF-D_V1.10
export EXE_NAME=tsmp-pdaf
```

```
PROFILING=false
DEBUG=false
CLEAN_BUILD=true
#################################################################
```

The variable `COUPLE_TYPE` sets the TerrSysMP model configuration and can take the following values:

- `clm_stand_alone` (only CLM)
- `parflow_stand_alone` (only ParFlow)
- `terrsysmp` (CLM and ParFlow coupled via OASIS-MCT)

The variable `PDAF_DIR` sets the path to the PDAF installation (in case PDAF is not installed in `$SVAROOT`). The variable `EXE_NAME` defines the name of the compiled TerrSysMP-PDAF executable. When the installation was successfull, the executable can be found in the directory `$SVAROOT/tsmp-pdaf/bin`. The variables `PROFILING`, `DEBUG` and `CLEAN_BUILD` are logical variables indicating whether TerrSysMP-PDAF should be instrumented with a profiling tools (`PROFILING`), whether TerrSysMP-PDAF should contain debugging information (`DEBUG`) and whether the project should be cleaned before compilation (`CLEAN_BUILD`).
TerrSysMP-PDAF is then compiled by executing the configured build script:

```
cd  $SVAROOT/tsmp-pdaf/install/
./build_pdaf_model.sh
```

2 Running TerrSysMP-PDAF

The execution of TerrSysMP-PDAF and the necessary input is closely related to the one of TerrSysMP. For executing TerrSysMP, a ParFlow database file `*.pfidb` (see ParFlow documentation for more details), an input file for CLM (`lnd.stdin`) and several input files for OASIS-MCT need to be present in the run directory. TerrSysMP-PDAF follows a similar approach. The only difference is that for each realisation a different set of ParFlow/ CLM input files needs to be present which follow a certain naming convention (see Sections 2.1 and 2.2). Additionally, a control file for the data assimilation (`enkfpf.par`, see Section 2.4) and observation files (Section 2.5) need to be present in the run directory. Furthermore, some command line options (Section 2.6) need to be specified when TerrSysMP-PDAF is executed.

2.1 Input files for CLM

For executing TerrSysMP-PDAF, the standard CLM input file `lnd.stdin` is replaced by separate input files for the individual CLM realisations. The naming convention for these input files is a user defined prefix followed by a formatted realisation number. For example, if the chosen prefix is `clminput`, the file names for the first three realisations are:
`clminput_00000`
`clminput_00001`
`clminput_00002`

All these files follow the same structure as the standard CLM input file `lnd.stdin`. The timing information needs to be the same for all these CLM input files but all other variables representing CLM input and output can differ between the realisations. It is important to note that the variable `caseid` should be an individual identifier for each CLM realisation because this variable determines the names of the CLM output files. If certain realisation share the same `caseid` the name of the CLM output files will be the same for these realisation (i.e., these realisation will produce the same output files) which results in undefined behaviour.

2.2 Input files for ParFlow

Similar to the input for CLM, the used has to create inidiviual ParFlow data base files for each model realisation which share the same file name prefix. For

example, if the chosen ParFlow file name prefix is `pfinput`, the file names for the first three model realisation are:
`pfinput_00000.pfidb`
`pfinput_00001.pfidb`
`pfinput_00002.pfidb`
As for the CLM input, the timing information contained in the ParFlow input files needs to be consistent among the different realisation. Other input information like initial conditions, parameters, etc. can differ between the realisations.

2.3 Input files for OASIS-MCT

When CLM and ParFlow are coupled via OASIS-MCT, certain input files for the coupler need to be provided for a regular TerrSysMP run. These files are principally the same for the execution of TerrSysMP-PDAF and are shared among the different realisations. These input files include the file `namcouple` which defines the coupling between CLM and ParFlow, the netCDF file `clmgrid.nc` which is identical to the CLM input file containing the grid data (see CLM documentation) and the remapping files for the coupling (`rmp_*.nc`). See the OASIS-MCT documentation for more details. It is important to note here that the remapping files should be created by a single deterministic TerrSysMP run of the model. The so created remapping files should then be copied to the TerrSysMP-PDAF run directory. If the remapping files are not present when TerrSysMP-PDAF is executed, each realisation will try to create these files. However, because the naming convention for the OASIS-MCT remapping files is fixed, this would lead to an undefined overwriting of these files from the different realisations.

2.4 Control file `enkfpf.par`

An additional file named `enkfpf.par` needs to be present in the TerrSysMP-PDAF run directory. This file contains information about the different model components (ParFlow, CLM and data assimilation) like, e.g., the timing information of the models, the prefixes for the ParFlow/ CLM input files, etc. This input is structure in three items: `[PF]` which holds information about ParFlow, `[CLM]` which holds information about CLM and `[DA]` which holds information about the data assimilation process. An example of this parameter file is given below. Note that the sequence of the entries within these three categories can be changed if desired.

```
[PF]
problemname      = ""
nprocs           =
starttime        =
```

```
dt               =
endtime          =
updateflag       =
paramupdate      =
aniso_perm_y     =
aniso_perm_z     =
printensemble    =
printstat        =
paramprintensemble =
paramprintstat   =

[CLM]
problemname = ""
nprocs     =
update_swc =
print_swc  =

[DA]
nreal =
outdir = ""
stat_dumpinterval =
da_interval     =
stat_dumpoffset =
```

In the following the individual entries of `enkfpf.par` are described:

`[PF]`

`problemname` (string) Problem prefix for ParFlow.

`nprocs` (integer) Number of processors per ParFlow instance. Must match with the specifications in the `*.pfidb` input.

`starttime` (real) ParFlow start time. Must match with the specifications in the `*.pfidb` input.

`dt` (real) Length of ParFlow time step. Must match with the specifications in the `*.pfidb` input. It is implicitly assumed that ParFlow and CLM calculate the same amount of time steps (i.e., have the same time step length).

`endtime` (real) Total simulation time (in terms of ParFlow timing). Must match with the specifications in the `*.pfidb` input.

`updateflag` (integer) Type of state vector update in ParFlow.

1: Assimilation of pressure data. State vector consists of pressure values and is directly updated with pressure observations.
2: Assimilation of soil moisture data. State vector consists of soil moisture content values and is updated with soil moisture observations. The updated soil moisture content is transformed back to pressure via the inverse van Genuchten functions.
3: Assimilation of soil moisture data. State vector consists of soil moisture content and pressure values. Soil moisture data are used to update pressure indirectly.

`paramupdate` (integer) Flag for parameter update
0: No parameter update
1: Update of saturated hydraulic conductivity
2: Update of Mannings coefficient

`aniso_perm_y` (real) Anisotropy factor of saturated hydraulic conductivity in y-direction. Only used when hydraulic conductivity is updated (`[PF]paramupdate = 1`)

`aniso_perm_z` (real) Anisotropy factor of saturated hydraulic conductivity in z-direction. Only used when hydraulic conductivity is updated (`[PF]paramupdate = 1`)

`printensemble` (integer) If set to `1`, the updated state variables for all ensemble members is printed out as `pfb` files after each assimilation cycle. Files are printed to `[DA]outdir` and follow the file naming convention of ParFlow. They include the specifier `update` in the file name.

`printstat` (integer) If set to `1` the ensemble statistics (mean and standard deviation) of the forecasted state variable are calculated and printed out as `pfb` files after each assimilation cycle. Files are printed to `[DA]outdir` and follow the file naming convention of ParFlow. The output files include the specifiers `press.mean`/ `press.sd` in case of assimilated pressure update `[PF]updateflag = 1` and the specifiers `swc.mean`/ `swc.sd` in case of assimilated soil moisture data `[PF]updateflag = 2/3`

`paramprintensemble` (integer) Only used in case of parameter update. If set to `1`, the updated parameters are printed to `pfb` files (similar to `[PF]printensemble`). Output files include the specifier `update.param`.

`paramprintstat` (integer) Only used in case of parameter update. If set to `1` statistics on the updated parameters are printed to `pfb` files (similar to `[PF]printstat`).

`CLM`

`problemname` (string) Problem prefix for CLM

`nprocs` (integer) Number of processors for each CLM instance.

`update_swc` Flag for update of soil moisture content in CLM (standalone only).
- `0`: No update of soil moisture content
- `1`: Update of soil moisture content

`print_swc` (integer) If set to `1`, the updated soil moisture content in CLM for all ensemble members is printed out as `netcdf` files (one file per realisation for the whole simulation period). Files are printed to the run directory and are named according to the CLM problem prefix of the corresponding realisation and include the specifier `update` in the file name.

`[DA]`

`nreal` (integer) Number of realisations used in the simulation.

`outdir` (string) Directory where assimilation results should be written.

`da_interval` (double) Time lag (in terms of ParFlow timing) at which the models are stopped for data assimilation. At these time steps the output from the data assimilation module is performed. The command line option `delt_obs` (see Section 2.6) allows to specify an additional lag at which data assimilation is actually performed.

`stat_dumpoffset` File number offset for the data assimilation output files for ParFlow. Can be used when an assimilation run is restarted.

2.5 Observation files

Observation files for TerrSysMP-PDAF are netCDF files which follow a certain naming convention. Each observation file has a fixed prefix followed by the formatted number of the assimilation cycle which is determined by `[DA]da_inverval` (see Section 2.4). The prefix is chosen with the command line option `obs_filename` (see Section 2.6). For example, if the chosen prefix is `myobs`, the observation files are named `myobs.00001, myobs.00002, myobs.00003`, etc.
The observation files contain one dimension named `dim_obs` which should be set equal to the number of observations for the respective assimilation cycle. The observation files for ParFlow should contain the following variables which all should have the dimension `dim_obs` (except variable `dr` in CLM observations files):

`obs_pf` (real) Observations for ParFlow (either pressure or soil moisture)

Table 2.1: Default values for parameter file `enkfpf.par`

	parameter	value
[PF]		
	problemname	-
	nprocs	0
	starttime	0.0
	endtime	0
	dt	0.0
	updateflag	1
	paramupdate	0
	aniso_perm_y	1.0
	aniso_perm_z	1.0
	printensemble	1
	printstat	1
	paramprintensemble	1
	paramprintstat	1
[CLM]		
	problemname	-
	nprocs	0
	update_swc	1
	print_swc	0
[DA]		
	nreal	0
	outdir	-
	da_interval	1
	stat_dumpoffset	0

`obserr_pf` (real) Observation errors which can be different for individual observations (optional). If this variable is not present. the command line option `rms_obs` (see Section 2.6) will be used to define the observation error (equal for all observations).

`ix` (integer) Position of the observation in the ParFlow grid in x-direction.

`iy` (integer) Position of the observation in the ParFlow grid in y-direction.

`iz` (integer) Position of the observation in the ParFlow grid in z-direction.

`idx` (integer) Index of the observation in the ParFlow grid.

The positions are always relative to the south-east corner cell at the lowest model layer. The index `idx` can be calculated as follows:

$$idx = (iz - 1) * nx * ny + (iy - 1) * nx + ix \tag{2.1}$$

where nx and ny are the number of grid cells in x- and y-direction respectively and ix, iy and iz are the positions of the observation in x-, y- and z-direction.

If TerrSysMP-PDAF is only applied with CLM, different variables have to be specified in the observation files:

`obs_clm` (real) Observations for CLM (soil moisture)

`obserr_clm` (real) Observation errors which can be different for individual observations (optional). If this variable is not present. the command line option `rms_obs` (see Section 2.6) will be used to define the observation error (equal for all observations).

`lon` (real) Longitude of the observation.

`lat` (real) Longitude of the observation.

`layer` (integer) CLM layer where the observation is located (counted from uppermost CLM layer).

`dr` (real) Snapping distance for the observation. This variable should have a length of 1 instead of `dim_obs`

Each of the CLM observations is snapped to the nearest CLM grid cell based on the given `lon`, `lat` and the snapping distance `dr` which should be smaller than the minimum grid cell size.

2.6 Command line options

The following command line options must be specified when TerrSysMP-PDAF is executed:

`n_modeltasks` (integer) Number of realisations. Must be consistent with `[DA]nreal`.

`filtertype` (integer) Type of filter used for data assimilation. For more details see the PDAF documentation. Currently, only option `2` (Ensemble Kalman Filter) is supported.

`delt_obs` (integer) Time lag for TerrSysMP stop times (see `[DA]da_interval`) at which assimilation is performed.

`rms_obs` (real) Measurement error.

`obs_filename` (string) Prefix for observation files.

An example of the model execution is given below:

```
mpiexec -np 512 ./tsmp-pdaf -n_modeltasks 64 -filtertype 2 -
    delt_obs 1 -rms_obs 0.1 -obs_filename myobs
```

With this command, the TerrSysMP-PDAF executable `tsmp-pdaf` will be run with 64 realisations (8 processors for each realisation) and EnKF analysis will be performed every assimilation cycle with an observation error of 0.1 and observations read from the files `myobs.xxxxx`.

3 Relevant publications

Papers describing the model physics or the technical implementation of the TerrSysMP-PDAF components are summarized in Table 3.1:

Table 3.1: References for different parts of TerrSysMP-PDAF

Model	References
CLM	Oleson et al. (2004)
	Oleson et al. (2008)
ParFlow	Ashby and Falgout (1996)
	Jones and Woodward (2001)
	Kollet and Maxwell (2006)
	Maxwell (2013)
OASIS-MCT	Valcke (2013)
	Valcke et al. (2013)
PDAF	Nerger and Hiller (2013)

Bibliography

Ashby, S. and Falgout, R. (1996). A parallel multigrid preconditioned conjugate gradient algorithm for groundwater flow simulations. *Nucl. Sci. Eng.*, 124:145–159.

Jones, J. E. and Woodward, C. S. (2001). Newton–krylov-multigrid solvers for large-scale, highly heterogeneous, variably saturated flow problems. *Adv. Water Resour.*, 24(7):763–774.

Kollet, S. J. and Maxwell, R. M. (2006). Integrated surface–groundwater flow modeling: A free-surface overland flow boundary condition in a parallel groundwater flow model. *Adv. Water Resour.*, 29(7):945–958.

Maxwell, R. M. (2013). A terrain-following grid transform and preconditioner for parallel, large-scale, integrated hydrologic modeling. *Adv. Water Resour.*, 53:109–117.

Nerger, L. and Hiller, W. (2013). Software for ensemble-based data assimilation systems—implementation strategies and scalability. *Comput. Geosci.*, 55:110–118.

Oleson, K. W., Dai, Y., Bonan, G. B., Bosilovichm, R., Dickinson, R. E., Dirmeyer, P., Hoffman, F., Houser, P., Levis, S., Niu, G.-Y., Thornton, P. E., Vertenstein, M., Yang, Z.-L., and Zeng, X. (2004). Technical description of the community land model (clm). NCAR Technical Note NCAR/TN-461+STR, National Center for Atmospheric Research, Boulder, Colorado.

Oleson, K. W., Niu, G.-Y., Yang, Z.-L., Lawrence, D. M., Thornton, P. E., Lawrence, P. J., Stöckli, R., Dickinson, R. E., Bonan, G. B., Levis, S., Dai, A., and Qian, T. (2008). Improvements to the community land model and their impact on the hydrological cycle. *J. Geophys. Res.*, 113(G1).

Valcke, S. (2013). The oasis3 coupler: a European climate modelling community software. *Geosci. Model Dev.*, 6(2):373–388.

Valcke, S., Craig, T., and Coquart, L. (2013). Oasis3-mct user guide: Oasis3-mct 2.0. Technical Report TR/CMGC/13/17, CERFACS/CNRS SUC URA No 1875, Toulouse, France.