Documentation of Terra Stand Alone
Edited by Israel Meteorological Service

v. 5.3 – 01.06.2016

Yiftach Ziv

Summary

TERRA-ML (shortly TERRA) is the operational land surface scheme of the COSMO model and its climate version COSMO-CLM to supply the lower boundary condition for the atmospheric model. TERRA Stand Alone (TSA) is a decoupled version of TERRA that can be used without an atmospheric model, e.g. for experiments concerning soil parameterizations or for an efficient soil spin-up referred to a given model domain or to a single point (e.g. measurement site). In the Stand Alone Mode, Atmospheric data are required for the model forcing. TSA is now up to date with latest COSMO version (v. 5.3 – 01/06/2016).

Version history

Version	Date	Changes	Contact
Initial version		Initial version of the decoupled soil module TERRA	Felix Ament, MeteoSwiss
v4.13	29/09/2010	- Added an external tool to convert COSMO analysis forcing files to suitable format, see chapter 6.4 - `lpar` can now be used with new I/O (see ch. 6.4 and 7.4) - corrected small bug in computation of lw-net-radiation (see ch. 6.10)	Guy de Morsier, MeteoSwiss
v5.3	01/06/2016	- Updated COSMO modules to latest version - Revised code according to COSMO Coding Standards - Added features from TSA-GUF version: • ability to divide to sub-regions for more efficient resource use • bug fixes	Yiftach Ziv, IMS zivy@ims.gov.il

Contents

1. Background on TERRA Stand Alone

1.1 Role of TERRA

TERRA-ML is the operational land surface scheme of the COSMO model and its climate version COSMO-CLM. It supplies the lower boundary condition for the atmosphere over land points by parameterizing the energy and mass fluxes that are being exchanged at the surface. They depend on both the atmospheric condition and the land surface state. TERRA computes the soil state as expressed by soil temperature (T_SO) and soil moisture (W_SO) at each model time step at various model levels (depths).

1.2 Model characterization

For that purpose, TERRA solves the corresponding prognostic equations (heat diffusion equation and Richards equation) on a one dimensional (vertical) multilayered grid. Each soil column is computed independently. Vegetation is not considered explicitly, instead, its impact is specified by external parameters used in the flux parameterizations.
For more details concerning the physical parameterizations used in TERRA, as well as the techniques to solve the prognostic equations, please see the scientific part of the official COSMO documentation (Doms et al. 2011).

1.3 TERRA in COSMO

TERRA is directly implemented into the COSMO code as module `src_soil_multlay`, where the subroutine `terra_multlay()` performs the adjustment of the soil state on horizontal grid points that are classified as land. In a COSMO model integration, it is called after the atmospheric computations. In particular, the parameterization of surface fluxes is based on the transfer coefficients derived in the TKE scheme.

1.4 TERRA Stand Alone

For allowing soil-specific investigations, TERRA was externalized as an independent program initially by Felix Ament (Uni Hamburg) during his PhD thesis (2006). The work was consolidated by COSMO PP COLOBOC, resulting in the latest TSA version (TSA 4.13) available on the COSMO homepage, http://www.cosmo-model.org/ (in 2010, Guy de Morsier, MeteoSwiss). In 2014, an off-COSMO version was developed mainly by Julian Toedter in University of Frankfurt (GUF), from which some elements were taken to the latest version. In 2016 the main modules of TSA were updated according to COSMO modules and a comprehensive revision of the code according to COSMO coding standards was performed by Yiftach Ziv of Israel Meteorology Service (IMS). It is planned according to Jürgen Helmert (DWD) to develop a new stand alone version within the ICON framework.
It is important to understand that even though TERRA Stand Alone is able to be applied on a whole region, the computations within each soil column are completely independent. Furthermore, it does NOT contain any parallelization (MPI or OpenMP).

2. Model and code structure

2.1 Workflow

The main program is contained in `terra.f90`. It performs the following tasks:

1. Read namelist values or set standard values (via subroutine `read_namelist()` in `terra_io.f90`)
2. Allocate required fields (via subroutine `allocate_fields()` in terra_lmenv.f90)
3. Read fixed parameters, e.g. soil properties (subroutine `init_variables()` in `terra_lmenv.f90`)
4. Read fixed external parameters (via subroutines in `terra_io.f90`)
5. Read initial conditions (via subroutines in `terra_io.f90`)
6. Time stepping (see below)
7. Finalization and clean-up (via subroutine `clean_up()` in terra_lmenv.f90)

The time stepping computes the soil state at the next time level by solving the underlying equations and paramterizations. It is splitted into the following sub-tasks:

a) Get current date (important for forcing and vegetation adaption)

b) Adapt vegetation-related parameters as `LAI`, `PLCOV` and `ROOTDP` (annual cycles), organized by the routine `organize_extpar()` in `terra_io.f90`

c) Get meteorological forcing at the current time level, organized by `read_metforc()`.

d) Compute transfer coefficients with Louis scheme, `parturs()` in `terra_lmparam.f90`

e) Run the actual soil model (subroutine `terra_multlay()` in `src_soil_multlay.f90`) to compute the soil state at the next time level

f) Write output, if required (via subroutines in `terra_io.f90`)

Note that the state variables such as `T_SO` and `W_SO` have two timelevels (4th dimension of the array): The current and the previous. In the next time step, the previous one is replaced by the next one, and so on.

2.2 Source files

A list of the source files and their tasks:

File	Task
terra.f90	Main program, organization, time stepping
src_soil_multlay.f90	Module src_soil_multlay with subroutine terra_multlay to advance the state variables in time
terra_lmparam.f90	Additional parameterization routines: vegadapt (for vegetation adaption to annual cycles), parturs (Louis transfer scheme), other utility functions e.g. calc_albedo, computation of area indices, tools to convert lon/lat, surface parameterizations
terra_lmenv.f90	Environment routines, e.g. allocate_fields, init_variables, clean_up
terra_interpol.f90	Interpolation routines for input fields.
terra_io.f90	A collection of routines used for I/O (reading and organization).
gribio.f90	Routines to read/write GRIB1.
data_*.f90	Variable definitions. In most cases copied as is from COSMO. Except for data_terra_standalone.f90 containing variables new in TSA and data_soil.f90 containing extra soil data for tests.
utilities.f90 & meteo_utilities.f90	Contains tools for standard meteorological and general computations. (copied as is from COSMO)
support_datetime.f90	Tools for date/time extraction and manipulation
fxtr_definition.f90	Fieldextra code
environment.f90	Sets the environment for the model (copied as is from COSMO)
kind_parameters.f90	Defines precision kind of parameters (copied as is from COSMO)

3. Installation and Running

The code package contains the source code as a set of FORTRAN90 files. The installation depends on the system, and a Makefile is included which should facilitate the installation. The installation is not difficult, yet it may happen that some details turn out tricky depending on your system.

3.1 Package

First, unzip the provided archive (`tar xfzv`), resulting in the following directory structure:
- `/src`: contains all source code files (`*.f90`) and a `Makefile`
- `/DWD-libgrib`: source code for the GRIB1 library (see below)
- `/examples`: test cases to run TSA.
- `/docs`: stuff like this documentation, as well as „old“ documents contained in the TSA 4.13 release.
- `/tools`: Other additional scripts: (1) `run_tsa.sh` to run the model with sub-regions, see chapter 6.7, (2) `merge_gribs_domains.sh` to put together local outputs, see chapter 6.7.

3.2 Basic installation

The program relies on GRIB1 library to deal with I/O data. They should be installed on your system otherwise this has to be done in advance.
The source for the **GRIB1 library (DWD version)** is added in the subdirectory `/DWD-libgrib1`
It can be installed by adjusting the `Makefile`. Usually, only the compilers for FORTRAN90 and C have to be chosen along with the compiler flags. Some examples are included. Then, type „`make`“ to install the library. It should generate "`libgrib1.a`“ (by default in the parent directory, but is can be changed). Typing „`make clean`“ allows to clean up temporary compilation files.

Now, adjust the TSA `Makefile` found in the `/src` directory:
a) Specify the F90 compiler and linker and their flags
b) Link to the DWD-Grib1 library

Finally, type „`make`“ to compile the code. If it works, the executable „terra“ should appear.
Note: Some warnings may occur but they are not important as they concern unused routines. (should be cleaned in future!)

You can use „`make clean`“ to clean up the object and module files. The executable is not deleted.

3.3 Running

In principal, TSA is run by a shell script that calls the executable „`terra`“ together with an appropriate definition of namelist parameters. It typically looks like this:

```
cat > INPUT_TERRA << ****
&RUN_TERRA
ie=114,
je=79,
OTHER NAMELIST SETTINGS…
/END
****
\rm -rf YU*
touch YUCHKDAT
# start terra executable:
./terra
```

* For offline simulation at ONE measurement point it is possible to read the forcing data (measurements) with a direct binary access file.

To run TSA successfully, one further has to provide externally:
- initial conditions
- external parameters
- forcing data

Of course, the files have to correspond to the namelist settings. Please see chapter 6 for a detailed description of I/O handling.

*** Sometimes the model may crash giving an error message like: "*YUCHKDAT no such file or directory*".
This can be avoided by creating an empty file `YUCHKDAT` in the run directory, e.g. by executing „`touch YUCHKDAT`" as in the script fragment shown above.

4. Technical details and changes

This chapter highlights some new aspects of current TSA version as compared to COSMO TERRA and TSA 4.13.

4.1 Calculation Method

While the COSMO model, including TERRA has advanced to Block Computing, TSA remained with the old i,j method Furthermore, COSMO now makes use of tracers, but being a stand alone version, TSA cannot. As a result, qv (specific water vapor content) & qv_bd (qv boundary) in TSA have a 4th dimension of time step (current or next) while in COSMO TERRA they have only 3 (i,j,k).
note: when transferring from COSMO TERRA to TSA, qv & qv_bd must be changed accordingly in data_fields.f90.

4.2 Transfer scheme

The COSMO transfer scheme developed by M. Raschendorfer (DWD) requires information about atmospheric TKE, which is in general not available by measurements or past analysis fields. Therefore TSA uses the Louis scheme which was used by COSMO and GME in former times. The Louis scheme is implemented in the subroutine `parturs()`. However, we found problems in simulations over larger regions which were connected with higher roughness lengths caused by an additional uncommented „wind adaption“, leading to jumps in soil temperature of more than 1 degree within one model time step (computationally not stable!).
Therefore, the implementation was revised such that it always produces realistic transfer coefficients. Now, a new and well-commented subroutine `parturs_new()` is used by default.
In order to switch back to old `parturs()` - change the call in section 2.3 of `terra.f90` to `CALL parturs()`.

4.3 External parallelization

Motivation

Even though TSA is relatively cheap on computing resources, on a large region it can still exhibit considerable computation time. An MPI/OpenMP parallelization was omitted, but an external parallelization procedure, which is possible due to the independency of each vertical soil columns, was realized. It consists of running an individual TERRA program at each core for a certain subdomain such that the full domain is covered. This domain decomposition is similar as done internally in COSMO.

Realization

A convenient usage of this external parallelization was realized by introducing some small namelist variables. Then, the program finds the correct input data itself (given correct preprocessing):
`nsub_x,nsub_y`: Number partitions in longitudinal/latitudinal direction, total number of subregions is `nsub_x*nsub_y` `which_subreg`: An integer specifying which subregion is to be computed by TERRA when starting this runscript. Of course, `which_subreg` has to be smaller or equal to `nsub_x * nsub_y` (otherwise TERRA will notice and abort)

The counting of subregions is done in vertical stripes beginning at the lower-left boundary and ending at the upper-right, e.g. for `nsub_x`=3 and `nsub_y`=4 the large grid is decomposed as:

Subregion 4	*Subregion 8*	*Subregion 12*
Subregion 3	*Subregion 7*	*Subregion 11*
Subregion 2	*Subregion 6*	*Subregion 10*
Subregion 1	*Subregion 5*	*Subregion 9*

Attention: The grid options `startlon,startlat,ie,je` still specify the „large“ grid ! All input files have to be

on the full domain! In case of `nsub_x*nsub_y` > 1 (i.e., a domain decomposition is desired), the code now automatically adapts `startlon,startlat,ie,je` for the computation so that only the sub-region specified by `which_subreg` is computed.
→ To compute all sub-regions, you need to start `nsub_x*nsub_y` runscripts which ONLY differ by the namelist parameter `which_subreg`
Summary
The trick is that on a cluster one can simply run the independent regions in parallel, using almost the same runscript (except the `which_subregion` parameter) and the same input files (external parameters, initial conditions, forcing) as the input routines automatically look which part of the input files they have to extract (using the `startlon, startlat` values of TERRA and comparing them with the corresponding values in the input files). A shell script `run_tsa.sh` is supplied, which creates the runscript for each sub-region.

Output
Each subdomain produces its own output file. The only post-processing currently necessary is to gather all local output files afterwards. For that purpose, a shell script `merge_gribs_domains.sh` is supplied, which performs this task. This script requires fieldextra and some other modules. For long periods, a script that creates external parallelization for the merging process is available as well (`main_merge.sh`).

4.4 Bug fixes and minor changes

A list of some smaller changes made in the code that are not mentioned before:

- subroutine vegadapt
Here latitute is needed, but the code used `rlat` → changed to `rlat_geo` , i.e., the actual geographical latitude instead of rotated one.

- final simulation time
Instead of specifying ntstep_max (together with `ydate_ini`), one can now also specify the end date `ydate_end` in same format as `ydate_ini`. `ntstep_max` will be computed from it automatically (overridden) and is not needed anymore in the namelist. In order to use `ntstep_max`, set `ydate_end` to "0000000000" (or any date prior to `ydate_ini`).

- forecast of snow temperature
In section II.6 & II.7 of `src_soil_multlay.f90`, the explicit step forward uses (wrongly) zdt*2 (from old leapfrog). Removed this *2 to make it correct.

```
ztsnown(i,j) = ztsnow(i,j) + zdt~~*2._wp~~ *(zfor_snow - zgsb(i,j))  &
               /zrocs(i,j) - ( ztsn(i,j) - zts(i,j) )
```

5. Namelist variables

Numerous variables and options can be read in at run-time via the namelist, usually done in the runscript. Some of them have default values if not supplied, while others are mandatory to be specified in the runscript.
In the following, we go through the different namelist groups and comment on the settings. In particular, recommended settings are given where necessary so that the code can be used properly.

5.1 RUN_TERRA

These variables concern all settings about model setup in space and time as well as the choice between different parameterizations and options within the land surface model.

Variable	Type	Default	Description	Comments
dt	real	60	Model time step in seconds	
ydate_ini	character		Start date/time in format YYYYMMDDHH.	
ydate_end	character		End date/time in format YYYYMMDDHH.	
ntstep_max	integer	0	Number of model time steps to simulate	optional. automatically computed from ydate_int & ydate_end.
ie	integer	1	Number of grid points in x direction	
je	integer	1	Number of grid points in y direction	
pollon	real	-170	Longitude of rotated pole	
pollat	real	32.5	Latitude of rotated pole	
polgam	real	0.0	Angle between north poles	Usually 0.0 with a rotated grid, 90 for a regular grid.
startlon	integer	1	Longitude of lower left corner of domain	
startlat	integer	1	Latitude of lower left corner of domain	
dlon	integer	1	Grid distance in x-direction in degree	
dlat	integer	1	Grid distance in y-direction in degree	
nsub_x	integer	1	Number of grid partitions in x-direction	See chapter 4.3
nsub_y	integer	1	Number of grid partitions in y-direction	See chapter 4.3
which_subreg	integer	1	Which subdomain is to be computed	See chapter 4.3
ke_soil	integer	7	Number of active soil layers	The total number of soil layers is ke_soil+1 !
czhls_const	real		Depths of bottoms of each layer	These have to be ke_soil+1 values!
itype_heatcond	integer	1	Type of soil heat conductivity. 1 = fixed value based on mean soil moisture, 2 = depends on soil moisture	See Schulz et al. 2014
itype_hydbound	integer	1	Type of lower hydrological boundary condition (1 = free drainage; 2 = rigid lid (non-standard); 3 = constant ground water (non-standard))	
itype_hydparam	integer	1	Type of soil moisture drainage and diffusion parameterization(1= standard; 3 = Brooks and Coorey + DWD soil type; 5 = Brooks and Coorey + USDA soil types)	
itype_evsl	integer	2	Type of bare soil evaporation (1 = bucket; 2 = BATS; 3 = Noilhan and Planton (non-standard))	

lvegadapt	logical	.TRUE.	Adaption of LAI and PLCOV according to sinusoidal annual cycle of COSMO	Needs LAI_MN and LAI_MX values in external parameters and same for PLCOV
lrootadapt	logical	.TRUE.	Adaption of ROOTDP according to sinusoidal annual cycle of COSMO	Needs only ROOTDP value in external parameters
itype_root	integer	1	2 = non-uniform root depth distribution (experimental)	
lmelt	logical	.FALSE..	soil model with melting process	Makes W_SO_ICE prognostic
lmelt_var	logical	.FALSE..	freezing temperature dependent on water content	Should be .TRUE. if lmelt=.TRUE.
lmulti_snow	logical	. TRUE.	use multi-layer snow model	implemented by E. Machulskaya
ke_snow	integer	2	Number of snow layers in multi-layer snow model	
linfil_revised	logial	.FALSE..	Revised parameterization of infiltration (allows higher infiltration than standard version)	
lconstvegalb	logical	.TRUE.	NO spatially varying plant albedo (vegalb ee=213, tab=201)	implemented by J. Helmert
lstomata	logical	.TRUE.	spatially varying stomata resistance (rsmin2d ee=212, tab=201)	Is currently set to .FALSE. by init_variables() subroutine
lz0local	logical	.FALSE..	GRIB only: reads local roughness length (ee=82, tab=250) without contribution due to subgrid-scale orography	
nrecmax	integer	-1	Special option to allow equilibrium runs by repeating a certain periods many times (experimental)	implemented by Guy de Morsier
lcheck	logical	.TRUE.	print additional output on the screen to check input data	
lcalc	logical	.TRUE.	debug option: lcalc=false ==> terra and parturs are not invoked. if true, computation as usual	Should be true.
itype_hydcond	integer	0	type of soil hydraulic conductivity	0: standard 1: exponential profile of saturated hydraulic conductivity by J. Helmert
kexpdec	real	2.0	1/m Exponential Ksat-profile decay parameter for itype_hydcond=1	See Decharme et al. (2006)
crsmin	real	150.0	Minimum stomatal resistence	

5.2 EXTPARA

These variables are concerned with all setting about external parameters and their processing, in particular, vegetation properties.

Variable	Type	Default	Description	Comments
constfilename	character		Name of GRIB file containing constant surface parameters (external parameters)	
lext_monthly	logical	.FALSE..	Monthly values of LAI,PLCOV,Z0	Requires LAI12, PLCOV12,Z012 in ext.para.file
lgettcl	logical	.FALSE..	Get T_CL for lowest soil temperature layer from external parameters.	Usually it is taken from initial condition file. Not used if itype_heatbound=2!
lhomosoil	logical	.TRUE.	Use homogeneous (same) parameters for all gridpoints, as specified by following namelist variables:	This requires to give all parameters in the namelist! With input, all of these not needed.
soiltyp_const	real		Constant COSMO soil type	
plcov_const	real		Constant plant cover (used only if lvegadapt=.FALSE..)	
rootdp_const	real		Constant root depth (used only if lvegadapt=.FALSE..)	
lai_const	real		Constant leaf area index (used only if lvegadapt=.FALSE..)	
lai_min_const	real		Minimal leaf area index during winter season (used only if lvegadapt=.TRUE.)	
lai_max_const	real		Maximum leaf area index during summer season (used only if lvegadapt=.TRUE.)	
plcov_min_const	real		(Used only if lvegadapt=.TRUE.)	
plcov_max_const	real		(Used only if lvegadapt=.TRUE.)	
rstom_mn_const	real		Minimal stomatal resistance (s/m) (only if lstomata=.FALSE..)	
rstom_mx_const	real		Maximum of stomatal resistance (s/m) (only if lstomata=.FALSE..)	
vegalb_const	real		Albedo of vegetation (only if lconstvegalb=.TRUE.)	
z0_const	real		Constant roughness length (m)	
lat	real	52.0	Latitude (relevant only in case of single grid point integrations)	
hsurface	real	0.0	Height above sea level (relevant only in case of single grid point integrations)	

5.3 SOILINIT

These variables define the reading and treatment of the soil initial conditions.

Variable	Type	Default	Description	Comments
lmulti_in	logical	.TRUE.	Initial conditions on same multi-layer vertical grid? (false corresponds to old 2-layer model)	Strongly recommended to use new version
soilinitdir	character		Directory where initial fields file is	
soilinitprefix	character		Initial fields file name (without .nc)	
lvol_in	logical	.FALSE..	Initial soil moisture as volumetric soil water content (m H2O / m soil)	
lrel_in	logical	.FALSE..	Initial soil moisture contents as given as fraction of pore volume occupied by water	Not really supported anymore
lhomoinit	logical	.TRUE.	Use homogeneous (same) initial conditions for all gridpoints, as specified by following namelist variables:	This requires to give all values in the namelist! With input, all of these not needed.
t_soil0	real		Profile of initial soil temperature of ke_soil layers	ke_soil+1 values
t_cl0	real		Soil temperature of lowest climatological soil layer	Not used if lgettcl=.FALSE.
w_snow0	real		Initial snow water content	Often 0.0
w_g0	real		Profile of initial soil moisture content of all layers	ke_soil+1 values
w_i0	real		Initial water content of the interception store	Usually 0.0
w_cl0	real		Initial soil moisture content of lowest layer	Not used actually

5.4 METFORCING

These variables specifies properties about the external forcing and its treatment, e.g., interpolation.

Variable	Type	Default	Description	Comments
ntype_atminput	integer	1	Data source of atmospheric forcing	
ntype_raininput	integer	1	Data source of precipitation forcing	
ntype_radinput	integer	1	Data source of radiative forcing	
metfiledir	character		Directory of files containing atmospheric forcing	
metfileprefix	character		Prefix of files containing atmospheric forcing (without .nc)	
lpar	logical	.FALSE.	Read also PAR (photosynthetic active radiation) from forcing data. If false (default!), PAR is set to 50% of the net sw radiation at surface.	
radofiledir			Directory containing RADOLAN rain data	
rain_fac			Factor to scale precipitation data to m/s	
dz	real	2.0	Height of T and qv observations (m)	
dz_u	real	10.0	Height of wind measurements (m)	
ke_model	integer	-1	Specify lowest model level (useful, if input files contain several levels and are unordered)	
lhourly_data	logical	.FALSE..	GRIB / NetCDF input contains hourly data (precip, radiation)	
tincr_max	integer	0	Max. gap in input data, expressed in hours	
ldestaggeruv	logical	.FALSE..	Destagger velocities on input from external GRIB / NetCDF file	

5.5 OUTPUT

These variables control how the model output is to be performed.

Variable	Type	Default	Description	Comments
ntype_output	Integer	1	Type of output	
outdir	Character		Directory to store the output	
outprefix	Character		Prefix of outputfiles (in case of ntype_output=4: filename)	
nout_interval	Integer	60	Output interval in units of time steps(!). e.g., with dt=60sec, nout_interval=360 is output every 6 hours.	Take care to adapt it in case dt changes
lconstout	Logical	.TRUE.	Include (time constant) surface parameters in the output file	Old, not possible currently

6. References

Ament, F. (2006). Energy and moisture exchange processes over heterogeneous land-surfaces in a weather prediction model. PhD thesis, Bonn.

Baldauf, M., Seifert, A., Förster, J., Majewski, D., Raschendorfer, M. (2011). Operational Convective-Scale Numerical Weather Prediction with the COSMO Model: Description and Sensitivities. Mon.Wea.Rev. 139, 3887-3905.

G. Doms, J. Förstner, E. Heise, H.-J. Herzog, D. Mironov, M. Raschendorfer, T. Reinhardt, B. Ritter, R. Schrodin, J.-P. Schulz and G. Vogel (2011). A Description of the Nonhydrostatic Regional COSMO Model. Part II: Physical Parameterization. Technical Report. Available from http://www.cosmo-model.org/content/model/documentation/core/cosmoPhysParamtr.pdf

Kalinka, F., B. Ahrens (2011). A modication of the mixed form of Richards equation and its application in vertically inhomogeneous soils. Adv. in Sci. and Res. 6, 123-127. doi:10.5194/asr-6-123-2011

Schulz, J.-P., G. Vogel,C. Becker, S. Kothe, U. Rummel, and B. Ahrens (2016). Evaluation oft he ground heat flux simulated by a multi-layernland surface scheme using high-quality observations at grassland and bare soil. Met.Z. *In revision in press.*

SRNWP Data Exchange Programme. Cosmo hosted Observation Data Exchange Among European Meteorological Services. http://www.cosmo-model.org/srnwp/content/default.htm

Weedon, G.P., Balsamo, G., Bellouin, N., Gomes, S., Best, M.J. and Viterbo, P., 2014. The WFDEI meteorological forcing data set: WATCH Forcing Data methodology applied to ERA-Interim reanalysis data. Water Resources Research, 50, doi:10.1002/2014WR015638.
See also http://www.eu-watch.org/gfx_content/documents/README-WFDEI(1).pdf