

The Common Land Model (CoLM)

Technical Guide

Duoying Ji, Yongjiu Dai

College of Global Change and Earth System Science
Beijing Normal University
Beijing 100875
China

E-mail:

duoyingji@bnu.edu.cn
yongjiudai@bnu.edu.cn

Oct 28, 2010

Contents

1. Introduction
2. Creating and Running the Executable
 - 2.1 Specification of script environment variables and header file
 - 2.2 Surface data making
 - 2.3 Initial data making
 - 2.4 Time-loop calculation
3. CoLM Surface Dataset
4. CoLM Atmospheric Forcing Dataset
 - 4.1 GSWP2 forcing dataset
 - 4.2 PRINCETON forcing dataset
 - 4.3 Temporal interpolation of the forcing data
5. CoLM Model Structure and Parallel Implementation
 - 5.1 CoLM Model Structure
 - 5.2 CoLM MPI Parallel Design
 - 5.3 CoLM MPI Parallel Implementation
 - 5.4 CoLM Source Code and Subroutines Outline
6. CoLM Parameter and Variables
 - 6.1 Model Parameters
 - 6.2 Time invariant model variables
 - 6.3 TUNABLE constants
 - 6.4 Time-varying state variables
 - 6.5 Forcing
 - 6.6 Fluxes
7. Examples of offline simulation
 - 7.1 Single Point Offline Experiment
 - 7.2 Global Offline Experiment with GSWP2 Dataset
8. Coupling of CoLM with CSM/ESM
 - 8.1 General framework of CSM/ESM
 - 8.2 Coupling with GCCESM

Table 1: Model directory structure
 Table 2: define.h CPP tokens
 Table 3: Namelist variables for initial data making
 Table 4: Namelist variables for Time-loop calculation
 Table 5: The list of raw data available
 Table 6: Description of 24-category (USGS) vegetation categories
 Table 7: Description of 17-category soil categories
 Table 8: The relative amounts of sand, soil, and clay
 Table 9: netCDF File Information of the Processed Atmospheric Forcing Data
 Table 10: Source code and Subroutines Outline
 Table 11: Dimension of model array
 Table 12: Control variables to determine updating on time steps
 Table 13: Model time invariant variables
 Table 14: Model TUNABLE constants
 Table 15: Run calendar
 Table 16: Time-varying Variables for restart run
 Table 17: Atmospheric Forcing
 Table 18: Model output in xy Grid Form
 Table 19: Interfaces added to couple with GCCESM
 Table 20: Fields exchanged between CoLM and coupler in GCCESM

Figure 1: Flow chart of the surface data making
 Figure 2: Flow chart of the initial data making
 Figure 3: Flow chart of the time-looping calculation
 Figure 4: Diagram of the domain partition at surface data making
 Figure 5: Diagram of the domain partition at time-looping calculation
 Figure 6: Diagram of the patches and grids mapping relationship
 Figure 7: General framework of CSM/ESM
 Figure 8: The framework of GCCESM
 Figure 9: Flow Chart of the Time-looping Calculation in Coupled Mode

1. Introduction

This user's guide provides the user with the coding implementation, and operating instructions for the Common Land Model (CoLM) which is the land surface parameterization used in offline mode or with the global climate models and regional climate models.

The development of the Common Land Model ([hereafter we call CLM initial version](#)) can be described as the work of a community effort. Initial software specifications and development focused on evaluating the best features of existing land models. The model performance has been validated in very extensive field data included sites adopted by the Project for Intercomparison of Land-surface Parameterization Schemes (Cabauw, Valdai, Red-Arkansas river basin) and others [FIFE, BOREAS, HAPEX-MOBILHY, ABRACOS, Sonoran Desert, GSWP, LDAS]. The model has been coupled with the NCAR Community Climate Model (CCM3). Documentation for the CLM initial version is provided by Dai et al. (2001) while the coupling with CCM3 is described in Zeng et al. (2002). The model was introduced to the modeling community in Dai et al. (2003).

The [CLM initial version](#) was adopted as the Community Land Model (CLM2.0) for use with the Community Atmosphere Model (CAM2.0) and version 2 of the Community Climate System Model (CCSM2.0). The current version of Community Land Model, CLM3.0, was released in June 2004 as part of the CCSM3.0 release (<http://www.cesm.ucar.edu/models/ccsm3.0/clm3/>). The Community Land Model (CLM3.0) is radically different from [CLM initial version](#), particularly from a software engineering perspective, and the great advancements in the areas of carbon cycling, vegetation dynamics, and river routing. The major differences between CLM 2.0 and [CLM initial version](#) are: 1) the biome-type land cover classification scheme was replaced with a plant functional type (PFT) representation with the specification of PFTs and leaf area index from satellite data; 2) the parameterizations for vegetation albedo and vertical burying of vegetation by snow; 3) canopy scaling, leaf physiology, and soil water limitations on photosynthesis to resolve deficiencies indicated by the coupling to a dynamic vegetation model; 4) vertical heterogeneity in soil texture was implemented to improve coupling with a dust emission model; 5) a river routing model was incorporated to improve the fresh water balance over oceans; 6) numerous modest changes were made to the parameterizations to conform to the strict energy and water balance requirements of CCSM; 7) Further substantial software development was also required to meet coding standards. Besides the changes from a software engineering perspective, the differences between CLM3.0 and CLM2.0 are: 1) several improvements to biogeophysical parameterizations to correct deficiencies; 2) stability terms were added to the formulation

for 2-m air temperature to correct this; 3) the equation was modified to correct a discontinuity in the equation that relates the bulk density of newly fallen snow to atmospheric temperature; 4) a new formulation was implemented that provides for variable aerodynamic resistance with canopy density; 5) the vertical distribution of lake layers was modified to allow for more accurate computation of ground heat flux; 6) a fix was implemented for negative round-off level soil ice caused by sublimation; 7) a fix was implemented to correct roughness lengths for non-vegetated areas. Documentation for the Community Land Model (CLM3.0) was provided by Oleson et al. (2004). The simulations of CLM2.0 coupling with the Community Climate are described in Bonan et al. (2002). The simulations of CLM3.0 with the Community Climate System Model (CCSM3.0) are summarized in the Special Issue of Journal of Climate by Dickinson et al. (2005), Bonan and S. Levis (2005).

Concurrent with the development of the Community Land Model, the CLM initial version was undergoing further development at Georgia Institute of Technology and Beijing Normal University in leaf temperature, photosynthesis and stomatal calculation. Big-leaf treatment by CLM initial version and CLM3.0 that treat a canopy as a single leaf tend to overestimate fluxes of CO₂ and water vapor. Models that differentiate between sunlit and shaded leaves largely overcome these problems. A one-layered, two-big-leaf submodel for photosynthesis, stomatal conductance, leaf temperature, and energy fluxes was necessitated to the CLM initial version, that is not in the CLM3.0. It includes 1) an improved two stream approximation model of radiation transfer of the canopy, with attention to singularities in its solution and with separate integrations of radiation absorption by sunlit and shaded fractions of canopy; 2) a photosynthesis–stomatal conductance model for sunlit and shaded leaves separately, and for the simultaneous transfers of CO₂ and water vapor into and out of the leaf—leaf physiological properties (i.e., leaf nitrogen concentration, maximum potential electron transport rate, and hence photosynthetic capacity) vary throughout the plant canopy in response to the radiation–weight time-mean profile of photosynthetically active radiation (PAR), and the soil water limitation is applied to both maximum rates of leaf carbon uptake by Rubisco and electron transport, and the model scales up from leaf to canopy separately for all sunlit and shaded leaves; 3) a well-built quasi-Newton–Raphson method for simultaneous solution of temperatures of the sunlit and shaded leaves. For avoiding confusion with the Community Land Model (CLM2.0, CLM3.0 versions), we name this improved version of the Common Land Model as CoLM.

This was same as model now supported at NCAR. NCAR made extensive modifications mostly to make more compatible with NCAR CCM but some for better back compatibility with previous work with NCAR LSM. For purpose of using in a variety of other GCMs and mesoscale models, this adds a layer of complexity that may be unnecessary. Thus we have continued testing further developments with CLM initial

version. Some changes suggested by Land Model working groups of CCSM are also implemented, such as, stability terms to the formulation for 2-m air temperature, a new formulation for variable aerodynamic resistance with canopy density. CoLM is radically different from either CLM initial version or CLM2.0 or CLM3.0, the differences could be summarized as follows,

- 1) Two big leaf model for leaf temperatures, photosynthesis-stomatal resistance;
- 2) Two-stream approximation for canopy albedoes calculation with the solution for singularity point, and the calculations for radiation for the separated canopy (sunlit and shaded);
- 3) New numerical scheme of iteration for leaf temperatures calculation; New treatment for canopy interception with the consideration of the fraction of convection and large-scale precipitation;
- 5) Soil thermal and hydrological processes with the consideration of the depth to bedrock;
- 6) Surface runoff and sub-surface runoff;
- 7) Rooting fraction and the water stress on transpiration;
- 8) Use a grass tile 2m height air temperature in place of an area average for matching the routine meteorological observation;
- 9) Perfect energy and water balance within every time-step;
- 10) A slab ocean-sea ice model;
- 11) Totally CoLM coding structure.

The development of CoLM is trying to provide a version for public use and further development, and share the improvement contributed by many groups.

The source code and datasets required to run the CoLM in offline mode can be obtained via the web from:

<http://globalchange.bnu.edu.cn/research/models>

The CoLM distribution consists of three tar files:

CoLM_src.tar.gz
CoLM_src_mpi.tar.gz
CoLM_dat.tar.gz.

The file *CoLM_src.tar.gz* and *CoLM_src_mpi.tar.gz* contain code, scripts, the file *CoLM_src.tar* is the serial version of the CoLM, and the file *CoLM_src_mpi.tar.gz* is the parallel version of the CoLM, the file *CoLM_dat.tar* contains raw data used to make the model surface data. The Table 1 lists the directory structure of the parallel version model.

Table 1: Model Directory Structure

Directory Name	Description
CoLM/rawdata/	"Raw" (highest provided resolution) datasets used by CoLM to generate surface datasets at model resolution. We are currently providing 5 surface datasets with resolution 30 arc second: DEM-USGS.30s LWMASK-USGS.30s (not used) SOILCAT.30s SOILCATB.30s VEG-USGS.30s BEDROCKDEPTH (not available) LAI (not available)
CoLM/data/	Atmospheric forcing variables suitable for running the model in offline mode
CoLM/mksrfddata/	Routines for generating surface datasets
CoLM/mkinidata/	Routines for generating initial datasets
CoLM/main/	Routines for executing the time-loop calculation of soil temperatures, water contents and surface fluxes
CoLM/run/	Script to build and execute the model
CoLM/graph/	GrADs & NCL files for display the history files
CoLM/interp/	Temporal interpolation routines used for GSWP2 & PRINCETON atmospheric forcing dataset
CoLM/tools/	Useful programs related with model running

The scientific description of CoLM is given in

- [1]. Dai, Y., R.E. Dickinson, and Y.-P. Wang, 2004: A two-big-leaf model for canopy temperature, photosynthesis and stomatal conductance. *Journal of Climate*, 17: 2281-2299.
- [2]. Oleson K. W., Y. Dai, G. Bonan, M. Bosilovich, R. E. Dickinson, P. Dirmeyer, F. Hoffman, P. Houser, S. Levis, G. Niu, P. Thornton, M. Vertenstein, Z.-L. Yang, X. Zeng, 2004: Technical Description of the Community Land Model (CLM). NCAR/TN-461+STR.
- [3]. Dai, Y., X. Zeng, R. E. Dickinson, I. Baker, G. Bonan, M. Bosilovich, S. Denning, P. Dirmeyer, P. Houser, G. Niu, K. Oleson, A. Schlosser, and Z.-L. Yang, 2003: The Common Land Model (CLM). *Bull. of Amer. Meter. Soc.*, 84: 1013-1023.

- [4]. Dai, Y., X. Zeng, and R.E. Dickinson, 2002: The Common Land Model: Documentation and User's Guide (<http://climate.eas.gatech.edu/dickinson/>).

We value the responses and experiences of our collaborators in using CoLM and encourage their feedback on problems in the current model formulation and the coding, as well as insight and suggestions for future model refinement and enhancement. It would be particularly helpful if users would communicate such feedback informally and where possible share with us documented model applications including manuscripts, papers, procedures, or individual model development.

2. Creating and Running the Executable

The CoLM model can run as a stand alone executable where atmospheric forcing data is periodically read in. It can also be run as part of the atmosphere model where communication between the atmospheric and land models occurs via subroutine calls or the special coupler. In this technical guide, we'll focus on the parallel version CoLM, most of the scripts and setting of the serial version CoLM are similar to the parallel version, and even more simple.

offline mode

In order to build and run the CoLM on offline mode, two sample scripts: *jobclm.csh*, *jobclm_single.csh*, and the corresponding *Makefile* files are provided in *run* and other source code directories respectively.

The scripts, *jobclm.csh* and *jobclm_single.csh*, create a model executable, determine the necessary input datasets, construct the input model namelist. Users must edit these scripts appropriately in order to build and run the executable for their particular requirements and in their particular environment. These scripts are provided only as an example to aid the novice user in getting the CoLM up and running as quickly as possible. The script *jobclm_single.csh* is used to do a single-point offline simulation experiment, can be run with minimal user modification, assuming the user resets several environment variables at the top of the script. In particular, the user must set **ROOTDIR** to point to the full disk pathname of the model root directory. And the *jobclm.csh* is used to do a global or regional offline simulation experiment, usually should be modified heavily to fulfill different requirements. The following part we'll explain the *jobclm.csh* in detail.

The script *jobclm.csh* can be divided into five sections:

- 1) Specification of script environment variables, creating header file *define.h*;
- 2) Compiling the surface data making, initial data making, time-loop calculation programs respectively.
- 3) Surface data making, including input namelist creating;
- 4) Initial data making: including input namelist creating;
- 5) Time-loop calculation: including input namelist creating.

2.1 Specification of script environment variables

The user will generally not need to modify the section of *jobclm.csh*, except to:

- 1) set the model domain edges and the basic computer architecture,

- 2) set the model path directory,
- 3) create the subdirectory for output, and
- 4) create the header file *\$CLM_INCDIR/define.h*.

BOX 1: EXAMPLE FOR SPECIFICATION OF SCRIPT ENVIRONMENT VARIABLES

```
# set the basic computer architecture for the model running
setenv ARCH intel

# set the model domain for north, east, south, west edges
setenv EDGE_N      90.
setenv EDGE_E      180.
setenv EDGE_S      -90.
setenv EDGE_W      -180.

# set the number of grids of the CoLM and the forcing dataset at
longitude and latitude directions
setenv NLON_CLM    360
setenv NLAT_CLM    180
setenv NLON_MET    360
setenv NLAT_MET    180

# set the number of processes used to parallel computing, MPI
related.
setenv TASKS 24

# The user has to modify the ROOTDIR to his/her root directory,
for example, /people.
setenv ROOTDIR /people/$LOGNAME

# 1) set clm include directory root
setenv CLM_INCDIR $ROOTDIR/CoLM/include

# 2) set clm raw land data directory root
setenv CLM_RAWDIR $ROOTDIR/CoLM/rawdata

# 3) set clm surface data directory root
setenv CLM_SRFDIR $ROOTDIR/CoLM/mksrfddata

# 4) set clm input data directory root
setenv CLM_DATADIR $ROOTDIR/CoLM/data

# 5) set clm initial directory root
setenv CLM_INIDIR $ROOTDIR/CoLM/mkinidata

# 6) set clm source directory root
setenv CLM_SRCDIR $ROOTDIR/CoLM/main
```

```

# 7) set executable directory
setenv CLM_EXEDIR $ROOTDIR/CoLM/run

# 8) create output directory
setenv CLM_OUTDIR $ROOTDIR/CoLM/output
mkdir -p $CLM_OUTDIR >/dev/null

#-----
# build define.h in ./include directory
#-----
\cat >! .tmp << EOF
#undef COUP_CSM
#undef RDGRID
#undef SOILINI
#define offline
#undef BATS
#undef SIB2
#undef IGBP
#define USGS
#define EcoDynamics
#define LANDONLY
#undef LAND_SEA
#undef SINGLE_POINT
#undef MAPMASK
#define NCDATA
#define PRINCETON
#undef GSWP2
#undef DOWNSCALING
#define WR_MONTHLY
EOF

if ($TASKS > 1) then
    \cat >> .tmp << EOF
#define SPMD
EOF
Endif

\cmp -s .tmp $CLM_INCDIR/define.h || mv -f .tmp
$CLM_INCDIR/define.h

```

The **ARCH** variable is used to set the architecture of the model running, and in the following section of the *jobclm.csh*, the *make* command will use the **ARCH** variable to invoke different *Makefile* to compile the model. The **EDGE_N**, **EDGE_E**, **EDGE_S**, **EDGE_W** four variables are used to locate the model domain edges, especially on the model surface data making. The number of model grids at latitude or longitude direction is set by the **NLAT_CLM** and **NLON_CLM**, these also are used for surface data making. The number of forcing dataset grids at latitude or longitude direction is set by the **NLAT_MET** and **NLON_MET**, these help do some simple forcing data downscaling when the model grids not exactly match the forcing dataset grids. The number of

processors involved in the parallel computing is set by the **TASKS** environment variables, if **TASKS** is great than one, the **SPMD** cpp token will be specified in **define.h** automatically, and the MPI parallel function will be build into the model, users could modify this logic according to your own requirements.

The file **define.h** contains model-dependent C-language cpp tokens. C-preprocessor directives of the form **#include**, **#if defined**, etc., are used in the model source code to enhance code portability and allow for the implementation of distinct blocks of functionality (such as incorporation of different modes) within a single file. Header file, **define.h**, is included with **#include** statements within the source code. When **make** command is invoked, the C preprocessor includes or excludes blocks of code depending on which cpp tokens have been defined in **define.h**.

Table 2: define.h CPP tokens

define.h cpp token	Description
OFFLINE	If defined, offline mode is invoked
RDGRID	If defined, the latitude and longitude of model grids are provided by input data
USGS	If defined, USGS 24 categories land cover legend are used
IGBP	If defined, IGBP 17 categories land cover legend are used
SiB2	If defined, SiB2 11 categories land cover legend are used
BATS	If defined, BATS 19 categories land cover legend are used
EcoDynamics	If defined, dynamic vegetation model is activated
LANDONLY	If defined, only land grid are activated
LAND_SEA	If defined, land and sea grids are activated
MAPMASK	If defined, users should supply the base map file to locate the specific region
NCDATA	If defined, netCDF format atmospheric forcing dataset being read, currently only supporting GSWP2 & PRINCETON datasets.
PRINCETON	If defined, the PRINCETON dataset being used. Depending on the NCDATA token.
GSWP2	If defiend, the GSWP2 dataset being used. Depending on the NCDATA token.
DOWNSCALING	If defined, the simple downscaling method used to re-grid the forcing data, usually used at high resolution

	simulation experiments
SPMD	If defined, the MPI parallel function being build into the model, this token is automatically set by the <i>jobclm.csh</i> according to the <i>TASKS</i> environmental variable
WR_HOURLY	If defined, history file is write at every time step
WR_DAILY	If defined, history file is write in daily average
WR_MONTHLY	If defined, history file is write in monthly average

2.2 Compiling the surface data making, initial data making, time-loop calculation programs

BOX 2: EXAMPLE FOR COMPILING THE MODEL

```

echo 'Compiling mksrfddata...'
cd $CLM_SRFDIR

make -f Makefile.${ARCH} clean
make -f Makefile.${ARCH} >>& $CLM_EXEDIR/compile.log.clm ||
exit 5

cp -f $CLM_SRFDIR/srf.x $CLM_EXEDIR/srf.x

echo 'Compiling mkinidata...'
cd $CLM_INIDIR

make -f Makefile.${ARCH} clean
make -f Makefile.${ARCH} >>& $CLM_EXEDIR/compile.log.clm ||
exit 5

cp -f $CLM_INIDIR/initial.x $CLM_EXEDIR/initial.x

echo 'Compiling main...'
cd $CLM_SRCDIR

make -f Makefile.${ARCH} clean
make -f Makefile.${ARCH} >>& $CLM_EXEDIR/compile.log.clm ||
exit 5

cp -f $CLM_SRCDIR/clm.x $CLM_EXEDIR/clm.x

```

In each source code directory of the model, two *Makfiles* exist: one is *Makefile.intel*, another one is *Makefile.ibm*. The *make* command uses the *ARCH* environment variable to select the right *Makefile* to compile the model, including the

surface making program, initial data making program and the time-loop main program. After the successful compiling procedure, three executable files named *srf.x*, *initial.x* and *clm.x* should occur in the *\$CLM_EXEDIR* directory. If some accident happened, users could refer to the *compile.log.clm* file at the *\$CLM_EXEDIR* directory to figure out the problem.

2.3 Surface data making: input namelist creating and executing

In this part, the *srfdat.stdin* namelist being firstly created, this namelist is used to direct the surface making program how to produce the surface data. The model surface data “*fsurdat*” is created by using the high resolution raw surface dataset, i.e., *fgridname*, *fmaskname*, *flandname*, *fsolaname*, *fsolbname*. If *RDGRID* cpp token defined, the *fgridname* should point to the file which contains the model grid information, including the latitude & longitude of all grids center, else the *fgridname* leaves blank. The *fmaskname* points to the land and ocean mask file, *fsolaname* points to the upper layer soil category dataset (0-30cm), *fsolbname* points to the deeper layer soil category dataset (30-100cm). Currently all these dataset comes from USGS. The *flandname* points to the land cover category classification dataset, currently the CoLM support *USGS*, *IGBP*, *SiB2*, *BATS* four land category legends, and each one could be set by modifying the *define.h* header file. In the default *CoLM_dat.tar.gz* dataset, we only provide the *USGS* land cover category dataset, users could download other land cover category datasets from <http://edcsns17.cr.usgs.gov/glcc> or contact us.

Users want to simulate the limited region (domain) which is not a regular shape, e.g. a city or state, could use the file *fmapmask* to specify a base map file, this file should be a zero/one land mask file, the value one should fill the region interested. And in the surface making process, the program would care about this, and drop the non-interested area. The *fmapmask* file should be at the same resolution as *flandname*, *fsolaname*, *fsolbname* and etc. A similar file is *fmetmask*, which is used to filter some points without atmospheric forcing dataset, it's also a zero/one land mask file, but it has the resolution of the model, the points without forcing dataset are also dropped.

A regular grid surface dataset can be generated for a single gridcell or for gridcells comprising a regional or global domain, *lon_points=1*, *lat_points=1* for a single gridcell simulation or *lon_points=nx*, *lat_points=ny* for a $n_x \times n_y$ model grids simulation. The model resolution are defined by model grid (*lon_points*, *lat_points*) and the domain edges, i.e.,

- edgen*: northern edge of model domain (degrees north)
- edges*: southern edge of model domain(degrees south)
- edgew*: western edge of model domain (degrees west)
- edgee*: eastern edge of model domain (degrees east)

The surface making program is paralleled using MPI, so developers want to add new function should take care of it.

BOX 3: EXAMPLE FOR SURFACE DATA MAKING

```
cd $CLM_EXEDIR

# Create an input parameter namelist file for srf.x

\cat >! $CLM_EXEDIR/srfdat.stdin << EOF
&mksrfexp
fmetmask      = '$CLM_DATADIR/gswp_mask'
fmapmask      = '/c2/data/CN_basemap/chinamap'
fgridname     = ''
fdemname      = '$CLM_RAWDIR/DEM-USGS.30s'
fmaskname     = '$CLM_RAWDIR/LWMASK-USGS.30s'
flandname     = '$CLM_RAWDIR/VEG-USGS.30s'
fsolaname     = '$CLM_RAWDIR/SOILCAT.30s'
fsolbname     = '$CLM_RAWDIR/SOILCATB.30s'
fsurdat       = '$CLM_DATADIR/srfddata.1deg'
lon_points    = $NLON_CLM
lat_points    = $NLAT_CLM
edgen         = $EDGE_N
edgee         = $EDGE_E
edges         = $EDGE_S
edgew         = $EDGE_W
nlon_metdat   = $NLON_MET
nlat_metdat   = $NLAT_MET
/
EOF

echo 'Executing CLM Making Surface Data'

if($TASKS > 1)then
    mpirun -prefix "[%g] " -np $TASKS $CLM_EXEDIR/srf.x <
        $CLM_EXEDIR/srfdat.stdin >& $CLM_EXEDIR/clm.log.srf
    || exit 5
else
    $CLM_EXEDIR/srf.x < $CLM_EXEDIR/srfdat.stdin >&
        $CLM_EXEDIR/clm.log.srf || exit 5
endif

echo 'CLM Making Surface Data Completed'
```

2.4 Initial data making: input namelist creating and executing

Upon successful completion of the surface data making in model grid and patches, surface data file has been generated in **CLM_DATADIR**. This section will make the model time-constant variables and time-varying variables on the model grids and patches.

Table 3: Namelist Variables for Initial data making

<i>Name</i>	<i>Description</i>	<i>Type</i>	<i>Notes</i>
site	case name	character	
greenwich	true: greenwich time, false: local time	logical	required
start_yr	starting date for run in year	integer	required
start_jday	starting date for run in julian day	integer	required
start_sec	starting seconds of the day for run in seconds	integer	required
fsurdat	full pathname of surface dataset (for example, '\$CLM_DATADIR/srfdata.valdai')	character	required
flaidat	full pathname of the leaf and stem area index, dataset	character	
fmetdat	full pathname of the meteorological data (for example, '\$CLM_DATADIR/VAL.DAT.CTRL.INT')	character	required
fhistTimeConst	full pathname of time-invariant dataset (for example, '\$CLM_OUTDIR/VALDAI-rstTimeConst')	character	required
fhistTimeVar	full pathname of time-varying dataset (for example, '\$CLM_OUTDIR/VALDAI-rstTimeVar')	character	required
foutdat	full pathname of output dataset (for example, '\$CLM_OUTDIR/VALDAI')	character	required
finfolist	full pathname of run information (for example, '\$CLM_EXEDIR/clmini.infolist')	character	required
lon_points	number of longitude points on model grid	integer	required
lat_points	number of latitude points on model grid	integer	required
deltim	time step of the run in second	real	required
mstep	total model step for the run	integer	required

BOX 4: EXAMPLE FOR INITIAL DATA MAKING

```
# Create an input parameter namelist file for initial.x

\cat >! $CLM_EXEDIR/inidat.stdin << EOF
&clminiexp
site           = 'GLOBAL'
greenwich      = .true.
```

```

start_yr      = 1948
start_jday    = 1
start_sec     = 1800
fsurdat       = '$CLM_DATADIR/srfddata.1deg'
flaidat       = ' '
fsoildat      = '$CLM_DATADIR/soilini'
fmetdat       = '/disk2/jidy/princeton_30min'
fhistTimeConst = '$CLM_OUTDIR/GLOBAL-rstTimeConst'
fhistTimeVar  = '$CLM_OUTDIR/GLOBAL-rstTimeVar'
foutdat       = '$CLM_OUTDIR/GLOBAL'
finfolist     = '$CLM_EXEDIR/clmini.infolist'
lon_points    = $NLON_CLM
lat_points    = $NLAT_CLM
nlon_metdat   = $NLON_MET
nlat_metdat   = $NLAT_MET
deltim        = 1800
mstep         = 931104
/
EOF

echo 'Executing CLM Initialization'

$CLM_EXEDIR/initial.x <$CLM_EXEDIR/inidat.stdin >&
$CLM_EXEDIR/clm.log.initial || exit 5

echo 'CLM Initialization Completed'

```

2.5 Time-loop calculation: input namelist creating and executing

Upon successful completion of the surface data and initial data, files for the time-constant variables, time-varying variables, and the namelist have been generated in '\$CLM_OUTDIR/VALDAI-rstTimeConst', '\$CLM_OUTDIR/VALDAI-rstTimeVar', and the '\$CLM_EXEDIR/clmini.infolist'. These include surface data, initialization files as well as the namelist file for the model time-loop execution. The variables in the namelist file clmini.infolist have been specified as Table 4:

Table 4: Namelist Variables for Time-loop Calculation

<i>Name</i>	<i>Description</i>	<i>Type</i>
site	case name	character
flaidat	full pathname of the leaf and stem area index, dataset	character
fmetdat	full pathname of the meteorological data (for example,	character

	'\$CLM_DATADIR/VAL.DAT.CTRL.INT')	
fhistTimeConst	full pathname of time-invariant dataset (for example, '\$CLM_OUTDIR/VALDAI-rstTimeConst')	character
fhistTimeVar	full pathname of time-varying dataset (for example, '\$CLM_OUTDIR/VALDAI-rstTimeVar')	character
foutdat	full pathname of output dataset (for example, '\$CLM_OUTDIR/VALDAI')	character
lhistTimeConst	logical unit number of restart time-invariant file	integer
lhistTimeVar	logical unit number of restart time-varying file	integer
lulai	logical unit number of LAI data	integer
lumet	logical unit number of meteorological forcing	integer
luout	logical unit number of output	integer
lon_points	number of longitude points on model grid	integer
lat_points	number of latitude points on model grid	integer
numpatch	total number of patches of grids	integer
deltim	time step of the run in second	real
mstep	total model step for the run	integer
spinup_dy	Number of days to spin-up	integer
spinup_yr	Number of years to spin-up	integer
fmetelev	Full pathname of the grid elevation of the atmospheric forcing dataset	character
nlon_metdat	Number of grids of atmospheric forcing data at longitude direction	integer
nlat_metdat	Number of grids of atmospheric forcing data at latitude direction	integer

As the following example showing, the namelist file used to run the time-loop part of the CoLM model is created by initial data making program, according to the patch number and others specified information. Before running the CoLM time-loop program, a *flux.stdin* namelist being created, this namelist is used to direct the CoLM history output. At sometimes, especially with high resolution running case, lots of output data is produced by the model, and most variables in history data are useless, the *flux.stdin* is used to handle this situation, we could use it to filter some useless variables, each variable headed with a “+” sign will be exported as normal, each variable headed with a “-” sign will be dropped. But when using the graph scripts in *graph/* directory, users should modify them to comport with the *flux.stdin*.

Also a *downs.stdin* namelist is created following the *flux.stdin*, which is used to do some simple atmospheric forcing data downscaling.

BOX 5: EXAMPLE FOR TIME-LOOP CALCULATION

```
# Create an input parameter namelist file for clm.x

mv -f $CLM_EXEDIR/clmini.infolist $CLM_EXEDIR/timeloop.stdin

# Create flux export namelist file for clm.x
# Don't change the sequence of the FLUX array elements!!

set FLUX = ( +taux          +tauy          +fsena          +lfevpa
             +fevpa        +fsenl          +fevpl          +etr
             +fseng        +fevpg          +fgrnd          +sabvsun
             +sabvsha      +sabg           +olrg           +rnet
             +xerr         +zerr           +rsur           +rnof
             +assim        +respc          +tss            +wliq
             +wice         +tg            +tlsun          +tlsha
             +ldew         +scv            +snowdp         +fsno
             +sigf         +green          +lai            +sai
             +avsdr        +avsdf          +anidr          +anidf
             +emis         +z0ma          +trad           +ustar
             +tstar        +qstar          +zol            +rib
             +fm           +fh            +fq             +tref
             +qref         +u10m          +v10m            +f10m
             +us           +vs            +tm             +qm
             +prc          +prl           +pbot            +frl
             +solar        )

@ i = 0

set flux_exp = "flux_exp="

foreach str ($FLUX)
  @ i = $i + 1
  if("$str" =~ +*) then
    set flux_exp = "$flux_exp +$i"
  else
    set flux_exp = "$flux_exp -$i"
  endif
end

\cat >! $CLM_EXEDIR/flux.stdin << EOF
&flux_nml
$flux_exp
/
EOF

\cat >! $CLM_EXEDIR/downs.stdin << EOF
&downs_nml
edgen = $EDGE_N
edgee = $EDGE_E
edges = $EDGE_S
```

```

edgew = $EDGE_W
/
EOF

echo 'Executing CLM Time-looping'

setenv FORT9 $CLM_EXEDIR/downs.stdin
setenv FORT7 $CLM_EXEDIR/flux.stdin

if($TASKS > 1)then
    mpirun -prefix "[%g] " -np $TASKS $CLM_EXEDIR/clm.x <
        $CLM_EXEDIR/timeloop.stdin >&
        $CLM_EXEDIR/clm.log.timeloop || exit 5
else
    $CLM_EXEDIR/clm.x < $CLM_EXEDIR/timeloop.stdin >&
        $CLM_EXEDIR/clm.log.timeloop || exit 5
endif

echo 'CLM Running Completed'

```

3. CoLM Surface Dataset

The data available as input to the programs *mksrfdat* include global terrain elevation, landuse/vegetation, land-water mask, soil types, in which the raw datasets are only needed if a surface dataset is to be created at surface data making. All data are available at 30 arc second resolution (**Table 4**). The data arrangement and format in the reformatted data file are as follows,

- Latitude by latitude from north to south in same longitude, the data points are arranged from west to east, starting from 0 degree longitude (or dateline).
- We use 2-character array to store the elevation, and 1-character array to store all other data (values < 100).
- All source data files are direct-access, which makes data reading efficient.
- All data are assumed to be valid at the center of the grid box.

Table 5: The list of raw data available

	<i>Resolution</i>	<i>Data source</i>	<i>Coverage</i>	<i>Size(bytes)</i>
Terrain Height	30 sec. (0.925 km)	USGS	Global	1,866,240,000
Land-Water Mask [#]	30 sec. (0.925 km)	USGS	Global	933,120,000
24-Category Land Cover ^{##}	30 sec. (0.925 km)	USGS	Global	933,120,000
17-Category Soil ^{###}	30 sec. (0.925 km)	FAO+STATSGO	Global	933,120,000

[#] The land-water mask data files are derived from USGS vegetation data files. *At each of lat/lon grid points, there is one number indicating the land (1), water (0), or missing data (-1) at that point.*

^{##} The 24 categories are listed. The 30-sec data are represented by one category-ID number at each of lat/lon grid point..

Table 6: Description of 24-category (USGS) vegetation categories

<i>Land Cover ID</i>	<i>Description</i>
1	Urban and Built-Up Land
2	Dryland Cropland and Pasture
3	Irrigated Cropland and Pasture
4	Mixed Dryland/Irrigated Cropland and Pasture
5	Cropland/Grassland Mosaic
6	Cropland/Woodland Mosaic

7	Grassland
8	Shrubland
9	Mixed Shrubland/Grassland
10	Savanna
11	Deciduous Broadleaf Forest
12	Deciduous Needleleaf Forest
13	Evergreen Broadleaf Forest
14	Evergreen Needleleaf Forest
15	Mixed Forest
16	Water Bodies(Including Ocean)
17	Herbaceous Wetland
18	Wooded Wetland
19	Barren or Sparsely Vegetated
20	Herbaceous Tundra
21	Wooded Tundra
22	Mixed Tundra
23	Bare Ground Tundra
24	Snow or Ice

FAO and STATSGO data are merged together. Both top soil layer (0 - 30 cm) and bottom soil layer (30 - 100 cm) data are provided. The 17 categories are listed. Similar to the vegetation data, the 30-sec data are represented by one category-ID number at each of lat/lon grid point.

Table 7: Description of 17-category Soil categories

<i>Soil Type ID</i>	<i>Soil Description</i>
1	Sand
2	Loamy Sand
3	Sandy Loam
4	Silt Loam
5	Silt
6	Loam
7	Sandy Clay Loam
8	Silty Clay Loam
9	Clay Loam
10	Sandy Clay
11	Silty Clay
12	Clay
13	Organic Materials
14	Water

15	Bedrock
16	Other
17	No data

Table 8: The relative amounts of sand, soil, and clay

<i>Class No.</i>	<i>Soil Texture Class</i>	<i>% Sand</i>	<i>% Silt</i>	<i>% Clay</i>
1	Sand	92	5	3
2	Loamy Sand	82	12	6
3	Sandy Loam	58	32	10
4	Silt Loam	17	70	13
5	Silt	10	85	5
6	Loam	43	39	18
7	Sandy Clay Loam	58	15	27
8	Silty Clay Loam	10	56	34
9	Clay Loam	32	34	34
10	Sandy Clay	52	6	42
11	Silt Clay	6	47	47
12	Clay	22	20	58
13	Organic materials	0	0	0
14	Water	0	0	0
15	Bedrock	0	0	0
16	Other	0	0	0
17	No data	0	0	0

4. CoLM Atmospheric Forcing Dataset

The CoLM needs the atmospheric forcing data when running at offline mode. Currently the CoLM support ASCII & netCDF format atmospheric forcing dataset. The *NCDATA* cpp token is used to distinguish the format being used. When *NCDATA* being set, we could use the *GSWP2* and *PRINCETON* atmospheric dataset of netCDF format, otherwise the ASCII forcing dataset is used. The ASCII data format is relative simple, each line represents a time record, which contains short-wave solar radiation [W/m^2], long-wave radiation [W/m^2], precipitation rate [mm/s], air temperature [K], wind speed [m/s], surface air pressure [Pa], specific humidity [kg/kg]. The ASCII format data is easy to use when doing single-point validating experiments, users could arrange the observed atmospheric variables according the above requirements and then feed them to the model. Some special requirements about the ASCII format forcing dataset could be fulfilled by investigating the source code file GETMET.F90 in *main/* directory.

The following two parts we'll give some details about how to use the *GSWP* and *PRINCETON* dataset in CoLM, the two dataset are widely used in land surface model validation and development.

4.1 GSWP2 Forcing Dataset

The Global Soil Wetness Project (GSWP) is an ongoing environmental modeling research activity of the Global Land-Atmosphere System Study (GLASS) and the International Satellite Land-Surface Climatology Project (ISLSCP), both contributing projects of the Global Energy and Water Cycle Experiment (GEWEX) in the World Climate Research Program(WCRP). GSWP was charged with producing as a community effort global estimates of soil moisture, temperature, snow water equivalent, and surface fluxes by integrating one-way uncoupled land surface schemes (LSSs) using externally specified surface forcings and standardized soil and vegetation distributions. GSWP-2 produced the best model estimates of the land-surface water and energy cycles over a ten year period. This project included an evaluation of the uncertainties linked to the LSSs, their parameters and the forcing variables. One of the main products of the GSWP2 is a state-of-the-art land surface model forcing dataset, which provides a common platform to many land surface models to evaluate their performance.

The GSWP2 dataset contains solar radiation, long-wave radiation, surface air temperature, surface air specific humidity, surface air pressure, total precipitation rate, convective precipitation rate, wind speed. Each variable has a data file for each month, and the date length range from 1982 to 1995, the time interval is 3hours, the spatial resolution is 1degree. Only the land points have data, so to save the storage space, the

GSWP2 dataset is compressed from 2D xy array into 1D vector array, the ocean grids are ignored, and all data files are stored in netCDF format to make it more portable among different computer platforms.

4.2 PRINCETON Forcing Dataset

The PRINCETON dataset is a global, 50-yr, 3-hourly, 1.0° dataset of meteorological forcing that can be used to drive models of land surface hydrology. The dataset is constructed by combining a suite of global observation-based datasets with the National Centers for Environmental Prediction–National Center for Atmospheric Research (NCEP–NCAR) reanalysis. For the known biases in the reanalysis precipitation and near-surface meteorology have been shown to exert an erroneous effect on modeled land surface water and energy budgets, so the PRINCETON dataset corrected these problems by using observation-based datasets of precipitation, air temperature, and radiation. This dataset also made corrections to the rain day statistics of the reanalysis precipitation, which have been found to exhibit a spurious wavelike pattern in high-latitude wintertime. Wind-induced undercatch of solid precipitation was removed using the results from the World Meteorological Organization (WMO) Solid Precipitation Measurement Inter-comparison. The statistical downscaling developed with the Global Precipitation Climatology Project (GPCP) daily product was used to disaggregate the precipitation in space to 1.0° resolution. Also the TRMM 3-hourly real-time dataset was used to disaggregation in time from daily to 3 hourly. Downward radiation, specific humidity, surface air pressure, and wind speed meteorological variables are downscaled in space while accounting for changes in elevation.

The PRINCETON dataset contains download solar radiation, download long-wave radiation, surface air temperature, surface air pressure, surface air specific humidity, wind speed, total precipitation rate. They all are stored in netCDF format, but with the ocean grids, so PRINCETON dataset occupies a huge disk spaces. Its long time series and splendid correction methods made it a good candidate for validating and evaluating the land surface model.

4.3 Temporal Interpolation of the Forcing Data

As stated above, GSWP and PRINCETON datasets all are netCDF format, and of the same spatial resolution, but the PRINCETON dataset has a very long time series. And their time intervals are 3hours, which is not suitable for contemporary land surface models. The CoLM usually uses the time step at 30 minutes, so we have to do temporal interpolation to make the GSWP and PRINCETON dataset suitable for CoLM.

In *interp/src* directory, we provided several temporal interpolation programs to handle different atmospheric variables. These variables includes download solar radiation (*SW_interp.F90*), long-wave radiation (*LW_interp.F90*), precipitation rate (*Rain_interp.F90*, *drv_finterp.F90*), wind speed, air temperature, air specific humidity, air pressure (all share the same temporal interpolation program: *UVTPQ_interp.F90*). The precipitation interpolation program (*drv_finterp.F90*) is a statistical method provided by GSWP2. And other variables' interpolation nearly all based on the Cubic Spline method, except in the solar short wave radiation interpolation, the sun elevation angle being considered.

In *interp/nml* directory some example namelist files for the interpolation are provided, including GSWP2 and PRINCETON data, and the *interp/job_interp.csh* demonstrates how to compile these interpolation programs and execute them. In *job_interp.csh* script, users should select which dataset being used (GSWP2 or PRINCETON), and the interpolation program being compiled dependent on this information to invoke the right data reading procedures. And finally the 3hourly raw forcing data is interpolated into 30minture interval, and all stored in a 1D vector array like GSWP2 dataset format. The ocean grids in PRINCETON data are dropped to save the storage space. The final data is of the same netCDF format, in spite of its source from GSWP2 or PRINCETON, this makes the CoLM time-loop program handle the forcing dataset easier.

The Table 9 lists the netCDF header information of the processed GSWP2 or PRINCETON dataset.

Table 9: netCDF File Information of the Processed Atmospheric Forcing Data

DIMENSIONS:	
<i>lon</i>	The longitude dimension
<i>lat</i>	The latitude dimension
<i>land</i>	The land dimension, used to compress the 2D xy grid data into 1D vector data, by ignoring the ocean grids to reduce the file size.
<i>time</i>	The time dimension
Variables:	
<i>origin_year</i>	The start year of the data
<i>origin_month</i>	The start month of the data

<i>origin_day</i>	The start day of the data
<i>origin_second</i>	The start second of the data
<i>lon</i>	The longitude value of the data grids
<i>lat</i>	The latitude value of the data grids
<i>land</i>	The land index of all on-land grids. A grid's land index value (kland) is calculated from its longitude index (ilon), latitude index (jlat) and the size of the longitude dimension (nlon), using the formula: $kland = (jlat-1)*nlon + ilon$
<i>time</i>	All time records of the data
<i>variable</i>	The variable to store the data of the grids, which is of the dimension (<i>time</i> , <i>land</i>). The <i>time</i> and <i>land</i> variables describe the exact time records and land grids.

The scientific description of GSWP and PRINCETON dataset is given in:

- [1]. Paul Dirmeyer, Xiang Gao and Taikan Oki, 2002: The Second Global Soil Wetness Project – Science and Implementation Plan. *IGPO Publication Series No.37*.
- [2]. Justin Sheffield, Gopi Goteti and Eric F. Wood, 2006: Development of a 50-Year High-Resolution Global Dataset of Meteorological Forcings for Land Surface Modeling. *Journal of Climate*. Vol 19, p3088-3111.

5. CoLM Model Structure and Parallel Implementation

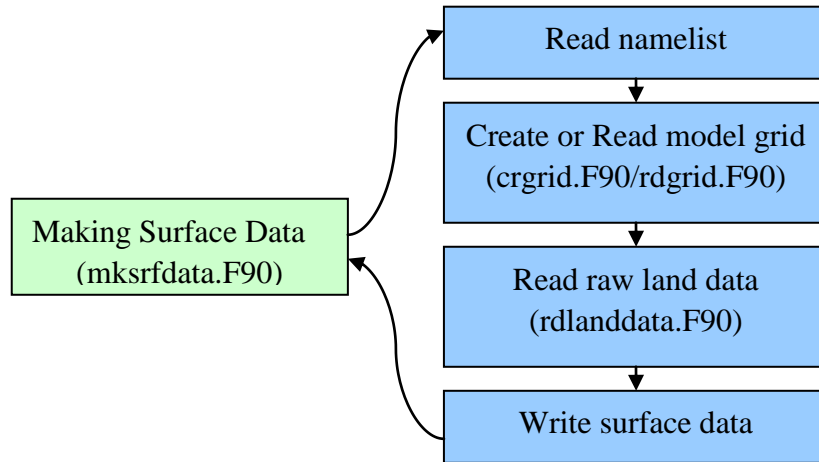
The computing flow of the CoLM could be viewed as doing time-looping calculation for each patch (sub-grid). The whole computing process has no interaction among different patches, except when calculating the grid average fluxes. And currently, the processes of each grid totally have no relation with others. This type computing flow and model structure gives a good agreement with the MPI SPMD (Single Program, Multiple Data) parallel method naturally. Also to archive good portability, we adopted the MPI SPMD as the parallel method.

5.1 CoLM Model Structure

Under the offline condition, the CoLM usually firstly makes the surface dataset, then makes the initial dataset, and finally do the time-loop calculations.

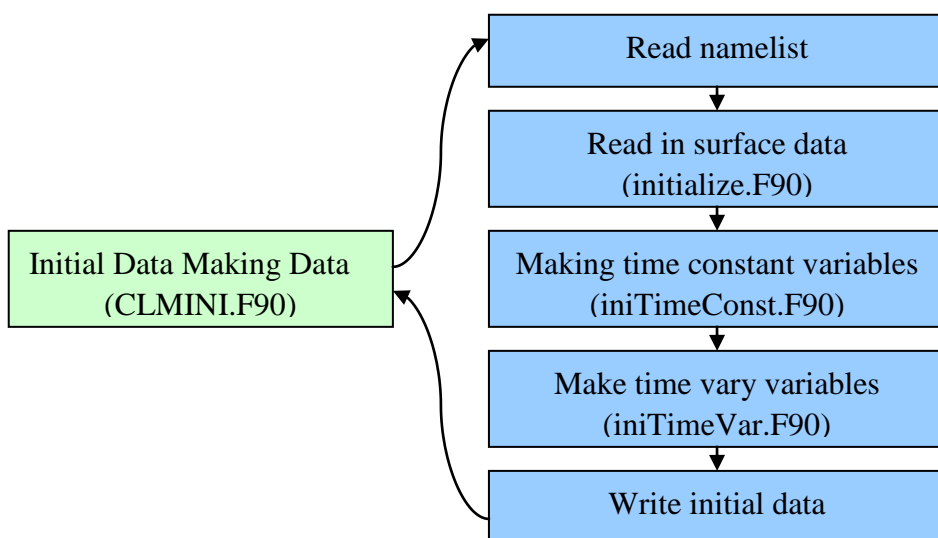
The computing flow and the invoking procedure of the surface data making program are demonstrated in the **Figure 1**:

Figure 1: Flow Chart of the Surface Data Making



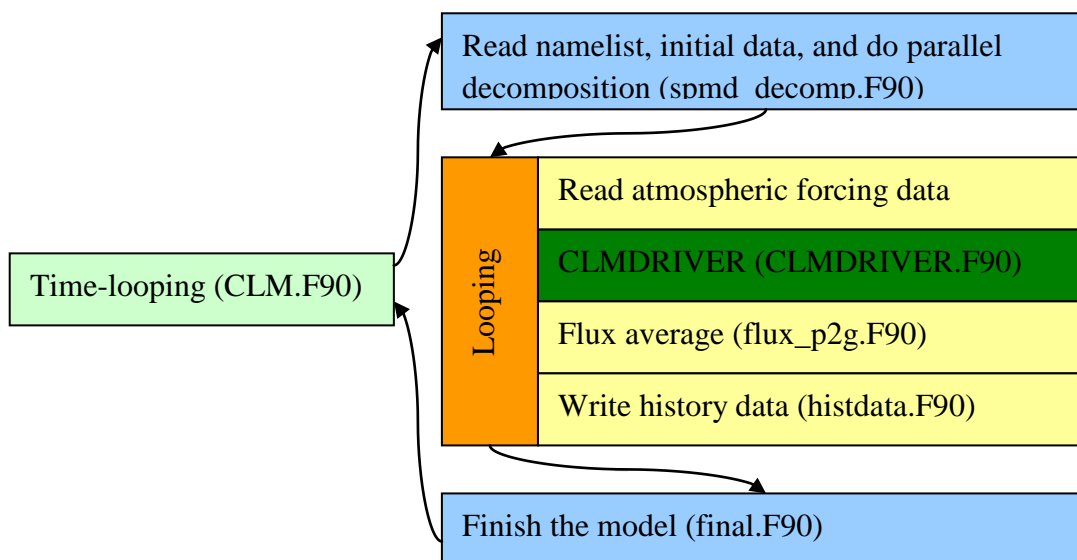
The computing flow and the invoking procedure of the initial data making program are demonstrated in the **Figure 2**:

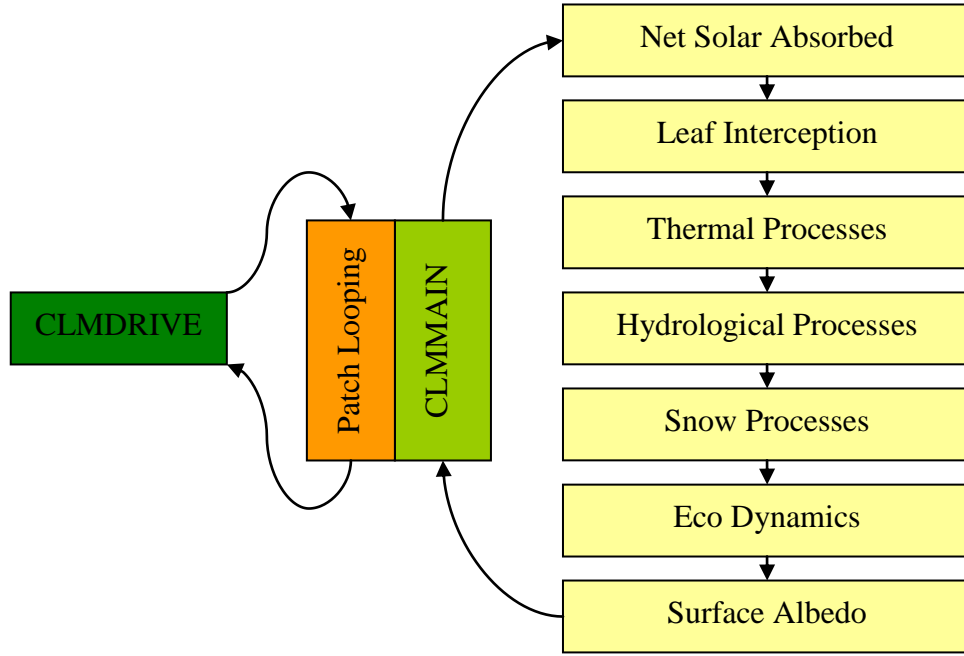
Figure 2: Flow Chart of the Initial Data Making



The computing flow and the invoking procedure of the time-looping program are demonstrated in the **Figure 3**:

Figure 3: Flow Chart of the Time-looping Calculation





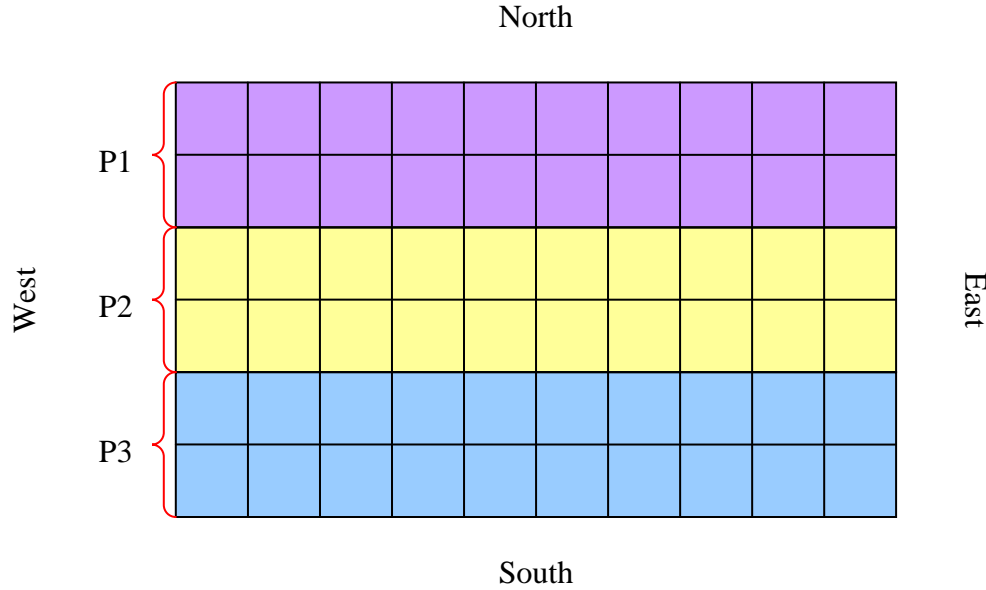
5.2 CoLM MPI Parallel Design

For the surface making and time looping are two time-consuming part of the CoLM, we'll talk their parallel design respectively. And the initial data making program does little computing, we'll leave it as the serial program.

The surface data making program needs read in huge volume of the raw high resolution land surface data (USGS dataset has a spatial resolution at 30arc second), including land cover categories, soil types etc. The characteristic of the land surface making program is 90 percent of the running time occupied by the data reading procedure (*rdlanddata.F90*). So we should emphasize on how to parallel the reading process. In fact, the current USGS dataset being used by CoLM all stored in FORTRAN record format files, which could be accessed randomly. So we could parallel this time consuming part by partitioning the model domain into several sub-domains of the nearly equal area, and each process involved in computing only read the raw data related to its sub-domain. The bottleneck of the method adopted is when the IO bandwidth of the computer system is lower, the parallel efficiency will be not very well. This determined by the characteristic of the surface making program. Some good example is SGI Altix platform, for its large IO bandwidth, a good parallel efficiency is reached.

The Figure 4 shows how to partition the whole model domain into several sub-domains in surface making program. In this example, three processes are involved in surface making, which being represented as P1, P2 and P3 in Figure 4:

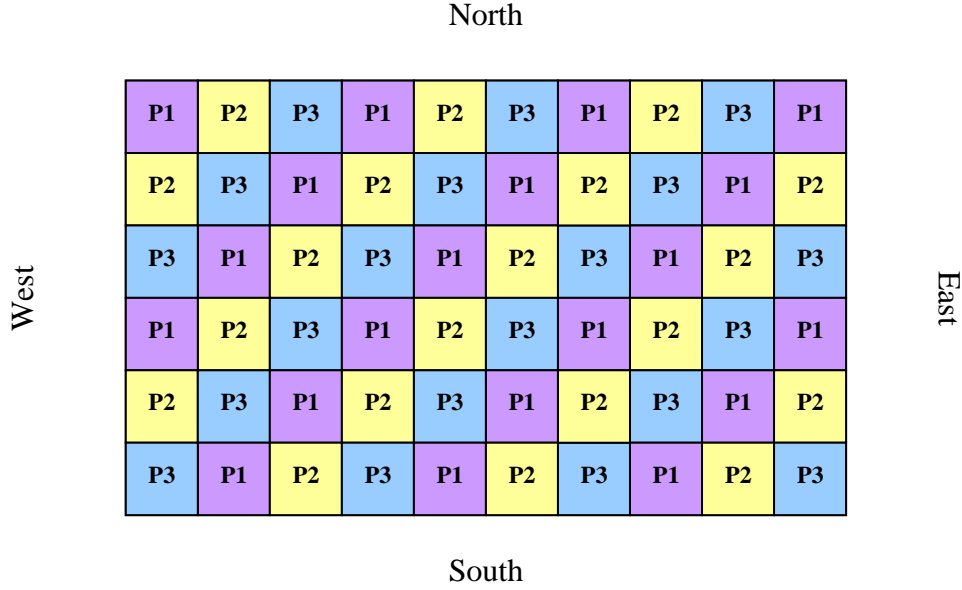
Figure 4: Diagram of the domain partition at surface data making



The time-looping calculation program's most running time occupied on reading the atmospheric forcing dataset and integrating over each patch. The forcing data reading time varies with different dataset, mostly related with its format, and its size. Parallel the forcing data read procedure will bring into many difficult when new data format being introduced, some format file not easy to work under parallel environments. So we could migrate the time costing in reading forcing data into preparation stages of the forcing dataset. And make the forcing data more easily to handle in time-looping calculation program. This is also helpful to introduce new dataset, and we don't need to touch too much model code. So the main problem left is how to parallel the integration over each patch. As we know, different patch with different land cover category, and thus involving different physics, biogeophysics, biogeochemistry processes, so with different running time. How to balance these differences is the critical point to parallel CoLM model. But on a large spatial scale, the adjacent grids usually have same or similar vegetation cover and other surface characters, so assigning adjacent grids on geographical locations to each involving computing processes could eliminate the difference stated above. So we assigned all model grids from north to south, from west to east, using the Round-Robin method, to each computing process. Finally each computing process get a nearly equal share of grids of certain land surface characters.

In Figure 5, a diagram demonstrates how to decompose domain grids when time-looping calculation. This example also contains three processes to calculate, each being represented as P1, P2 and P3.

Figure 5: Diagram of the domain partition at time-looping calculation



5.3 CoLM MPI Parallel Implementation

We adopted the most common parallel mode, master & slave model, to parallel the CoLM. It means that the N processes involved computing, one of them will handle some extra work, such as assigning workset, taking charge of reading or writing data etc.

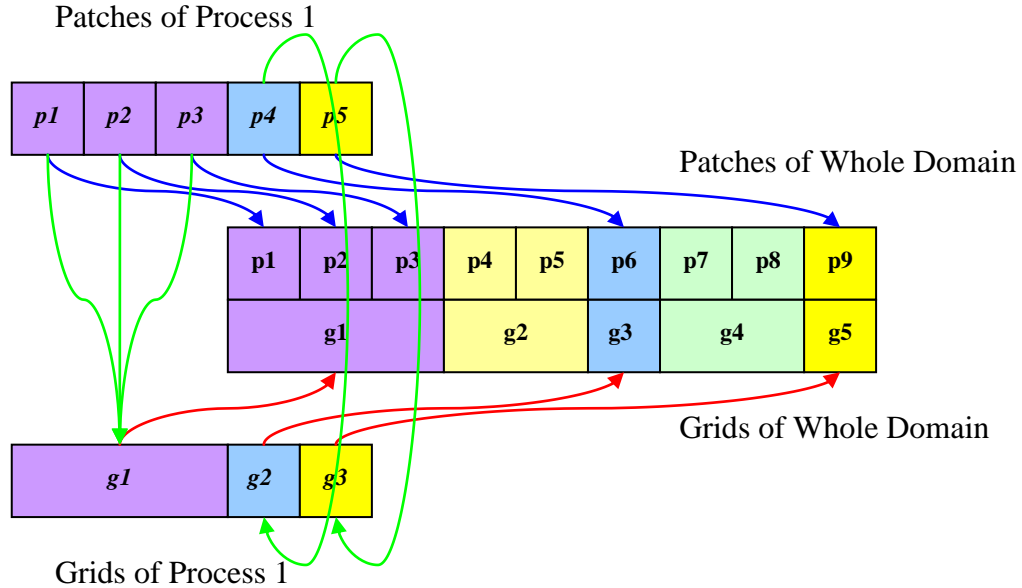
To make the surface making program working with a sub-domain partition fashion, we partition the whole domain before the program reading the high resolution land dataset (*rdlanddata.F90*), and use two variables (js , je) to indicate the indices of the begin and end point at latitude direction. The *rdlanddata.F90* subroutine will use these two variables to calculate the exact dataset it reads. After all processes processed their sub-domains, the master will gather all data of sub-domains and finally write them out.

The parallel of the time-looping part is a little complex than the surface making program, most of the difficulty comes from the Round-Robin fashion assigning workset, which makes the grids calculated by each process not adjacent on spatial, thus the data collection procedure becomes complex. To locate every patch or grid of all processes, we add a variable (*pgmap*) for each process to establish the mapping relationship between its own patches and its own grids, a variable (*pmap*) for each process to establish the mapping relationship of its own patches with the whole domain's total patches, a variable (*gmap*) for each process to establish the mapping relationship of its own grids with the whole domain's total grids. These three variables help do data scattering and gathering in

parallel model. And they are built after the model invoked `spmd_decomp.F90`. In `spmd_decomp.F90`, the model will build the *pgmap*, *pmap*, *gmap* according to the variables *ixy_patch* and *jxy_patch*, which are created in initial data making. After these procedures, all related model time constant and time varying variables will be dispatched respectively according to patches mapping relationship and grids mapping relationship. When forcing data being dispatched (in *forcedata.F90*), and fluxes data being gathered (in *histdata.F90*), these three variables also are used to build the exact grid-patch mapping relationship.

In Figure 6, a diagram demonstrates the mapping relationship between patches and grids. In this example, two processes, five model grids (*g1*, *g2*, *g3*, *g4*, *g5*), nine model patches (*p1*, *p2*, *p3* ... *p9*) are involved in calculation. The grid *g1* of the whole domain has three patches: *p1*, *p2* and *p3*, the grid *g2* of the whole domain has two patches: *p4* and *p5*, and etc. Only the first process's patches and grids mapping relationship illustrated on the figure. The green lines represent the mapping relationship between process's own patches and grids. The blue lines represent the mapping relationship between process's own patches and the whole domain's total patches. The red lines represent the mapping relationship between the process's own grids and the whole domain's grids.

Figure 6: Diagram of the patches and grids mapping relationship



5.4 CoLM Source Code and Subroutines Outline

Table 10: Source Code and Subroutines Outline

File in include/ directory	
include/define.h	CPP tokens used to distinguish the function blocks of the CoLM
Files (subroutines) in mksrfddata/ directory	
mksrfddata.F90	The main program to do the surface data making
crgrid.F90 (crgrid)	Create the model grids based on the domain edges and the number of grids specified
rdgrid.F90 (rdgrid)	Read the model grids information form a file, depending on the CPP token RDGRID
celledge.F90 (celledge)	Calculate the edges of each model grid
cellarea.F90 (cellarea)	Calculate the area of each model grid based on the its edges
rdlanddata.F90 (rdlanddata)	Read the USGS high resolution raw dataset, including land cover categories, soil categories etc
spmd.F90 (p_init, p_exit)	The MPI common subroutines, shared with the time-looping source code
Files (subroutines) in mkinidata/ directory	
CLMINI.F90	The main program to do the initial data making
initialize.F90 (initialize)	The subroutine to do further preparation work and calls all initial data making related subroutines
vegpara.h	A header file contains all vegetation related parameters for USGS, IGBP, SiB2, BATS land cover category legends
iniTimeConst.F90 (iniTimeConst)	The subroutine to set the model parameters not varying with time
iniTimeVar.F90 (iniTimeVar)	The subroutine to set the model parameters varying with time
lai_empirical.F90 (lai_empirical)	A empirical method to calculate the leaf area index, stem area index and etc.
orb_coszen.F90 (orb_coszen)	Calculate the cosine of the solar zenith angle

rstFileMod.F90 (rstTimeConstRead, rstTimeConstWrite, rstTimeVarRead, rstTimeVarWrite)	Subroutines to handle the model restart files
snowfraction.F90 (snowfraction)	Calculate the snow fraction relative to the whole model grid
twostream.F90 (twostream)	Calculate the canopy albedos via two stream approximation (direct and diffuse) and partition of incident solar
Files (subroutines) in main/ directory	
CLM.F90	The main program to do the time-looping calculation
CLMDRIVER.F90 (CLMDRIVER)	CoLM driver which to do further preparation work before calculate each model patch
CLMMAIN.F90 (CLMMAIN)	The core subroutine which invokes all land surface processes for each patch
GETMET.F90 (getmet)	Read the ASCII format atmospheric forcing dataset
LAKE.F90 (lake)	A lake sub-model invoked when the land cover category is Lake
SOCEAN.F90 (socean)	A simple ocean sub-model
THERMAL.F90 (THERMAL)	Calculate the thermal processes and surface fluxes
WATER.F90 (WATER)	Calculate the hydrological processes
albland.F90 (albland)	Calculate fragmented albedos (direct and diffuse) in wavelength regions split at 0.7um.
albocean.F90 (albocean)	Calculate the ocean surface albedoes for direct/diffuse for two spectral intervals
dewfraction.F90 (dewfraction)	Determine fraction of foliage covered by water and fraction of foliage that is dry and transpiring
eroot.F90 (eroot)	Calculate the effective root fraction and maximum possible transpiration rate
final.F90 (final)	Subroutine to do some final work when the model finished
flux_p2g.F90 (flux_p2g)	To do the grid fluxes averaging from its patches
forcedata.F90 (read_forcedata)	Read the atmospheric forcing dataset, also could read the LAI & SAI forcing dataset
groudfluxes.F90 (groudfluxes)	Calculate the surface fluxes
groupdtem.F90 (groundtem)	Calculate the soil and snow temperature
hCapacity.F90 (hCapacity)	Calculate the soil and snow heat capacities

hConductivity.F90 (hConductivity)	Calculate the soil and snow thermal conductivities
histdata.F90 (write_histdata)	Write out the model history data
spmd_decomp.F90 (mpi_decomp)	Initial work before doing time-looping calculation, including reading time constant and varying data, also the parallel decomposition
lai_empirical.F90 (lai_empirical)	A empirical method to calculate the leaf area index, stem area index and etc.
leafinterception.F90 (leafinterception)	Calculate the interception and drainage of the precipitation
leaftemone.F90 (leaftemone)	One-Big-Leaf canopy model
leaftemt看.F90 (leaftemt看)	Two-Big-Leaf canopy model
lpwrite.F90 (lpwrite)	Determine when to write the history data
meltf.F90 (meltf)	Calculate the phase change within snow and soil layers
Moninobuk.F90 (moninobuk)	Calculation of friction velocity, relation for potential temperature and humidity profiles of surface boundary layer
ncdata.F90 (ncdata_init, ncdata_read, ncdata_close)	netCDF dataset interfaces to handle the GSWP & PRINCETON atmospheric forcing data
netsolar.F90 (netsolar)	Net solar absorbed by surface
newsnow.F90 (newsnow)	Add new snow nodes
orb_coszen.F90 (orb_coszen)	Calculate the cosine of the solar zenith angle
paramodel.h	Model const parameters
phycon_module.F90	Physical constants
qsadv.F90 (qsadv)	Compute saturation mixing ratio and change in saturation mixing ratio with respect to temperature
rstFileMod.F90 (rstTimeConstRead, rstTimeConstWrite, rstTimeVarRead, rstTimeVarWrite)	Subroutines to handle the model restart files
snowage.F90	Update snow cover and snow age
snowcompaction.F90 (snowcompaction)	Compute the metamorphisms of changing snow characteristics caused by destructive, overburden, and melt.
snowfraction.F90 (snowfraction)	Calculate the snow fraction relative to the whole model grid
snowlayerscombine.F90 (snowlayerscombine)	Combine the snow layers according to the prescribed minimum thickness
snowlayersdivide.F90 (snowlayersdivide)	Subdivides snow layer when its thickness exceed the prescribed maximum
snowwater.F90 (snowwater)	Calculate the melted and infiltrated water

soilwater.F90 (soilwater)	Calculate the soil water contents based on the Richard Equation
spmd.F90 (p_init, p_exit)	The MPI common subroutines
stomata.F90 (stomata)	Calculation of canopy photosynthetic rate using the integrated model relating assimilation and stomatal conductance.
subsurfacerrunoff.F90 (subsurfacerrunoff)	Calculate the subsurface runoff
surfacerrunoff.F90 (surfacerrunoff)	Calculate the surface runoff
timemgr.F90 (ticktime)	Step up the model time and control the model spin-up
twostream.F90 (twostream)	Calculate the canopy albedos via two stream approximation (direct and diffuse) and partition of incident solar

6. CoLM Parameter and Variables

CoLM contains many model parameters and variables, which control the model behavior, store the model states, diagnose the model performance and etc. Most of them can be categorized into six categories: 1) model parameters; 2) time invariant model variables; 3) tunable model constants; 4) time-varying state variables; 5) atmospheric forcing variables; 6) fluxes variables. In the following Tables, most of model parameters and variables will be explained.

6.1 CoLM Model Parameters

Table 11: Dimension of model array (paramodel.h)

<i>Parameter</i>	<i>Description</i>	<i>Value</i>
nl_soil	Number of soil layers	10
maxsnl	maximum number of snow layers	-5
nfcon	number of time constant variables	119
nftune	number of tunable constants	14
nfvar	number of time varying variables	126
nforc	number of forcing variables	18
nfldv	number of output fluxes	92
nflai	number of leaf-area-index time varying variables	4
maxpatch	maximum number of patches in model grid	25
nlandcateg	number of land cover categories	25
nsoilcateg	number of soil texture categories	17

Table 12: Control Variables to Determine Updating on Time Steps

<i>Variables</i>	<i>Description</i>
dolai	True if time for time-varying vegetation parameters updating
doalb	True if time for surface albedo calculation
dosst	True if time for update sst/ice/snow

6.2 CoLM Time invariant model variables

Table 13: Model Time invariant variables - fcon (numpatch,1:nfcon)

<i>Internal name</i>	<i>Description</i>	<i>Unit</i>	<i>Code No.</i>

<u>Assigned or Derived by Using above Indices</u>			
dlat	Latitude in radians	radians	1
dlon	Longitude in radians	radians	2
itypwat	Land water type	index	3
ivt	Land cover type of classification	index	4
 <u>Soil physical parameters</u> <u>Derived from soil sand and clay percentages, and soil color type</u>			
albsol	Soil albedo for different coloured soils	-	5
csol (nl_soil)	Heat capacity of soil solids	J/(m ³ K)	6:15
porsl (nl_soil)	Fraction of soil that is voids	-	16:25
phi0 (nl_soil)	minimum soil suction	mm	26:35
bsw (nl_soil)	Clapp and hornbereger "b" parameter	-	36:45
dkmg (nl_soil)	Thermal conductivity of soil minerals	W/(m K)	46:55
dksatu(nl_soil)	Thermal conductivity of saturated soil	W/(m K)	56:65
dkdry (nl_soil)	Thermal conductivity for dry soil	W/(m K)	66:75
hksati(nl_soil)	Hydraulic conductivity at saturation	mm /s	76:85
 <u>Vegetation static parameters</u> <u>Derived from vegetation type</u>			
z0m	Aerodynamic roughness length	m	86
displa	Displacement height	m	87
sqrtdi	Inverse sqrt of leaf dimension	m ^{-0.5}	88
effcon	Quantum efficiency of RuBP regeneration	molCO ₂ /molquanta	89
vmax25	Maximum carboxylation rate at 25°C at canopy top		90
slti	s ₃ : slope of low temperature inhibition function		91
hlti	s ₄ : 1/2 point of low temperature inhibition function		92
shti	s ₁ : slope of high temperature inhibition function		93
hhti	s ₂ : 1/2 point of high temperature inhibition function		94
trda	s ₅ : temperature coefficient in g _s -a model		95
trdm	s ₆ : temperature coefficient in g _s -A model		96
trop	Temperature coefficient in g _s -A model		97
gradm	Conductance-photosynthesis slope parameter		98
binter	Conductance-photosynthesis intercep		99
extkn	Coefficient of leaf nitrogen allocation		100
chil	Leaf angle distribution factor		101
ref (2,2)	Leaf reflectance (iw=iband, il=life and dead)		102:105

tran (2,2)	Leaf transmittance (iw=iband, il=life and dead)	106:109
rootfr(nl_soil)	Fraction of roots in each soil layer	111:119

6.3 CoLM TUNABLE constants

Table 14: Model TUNABLE constants - ftune(1:14)

<i>Internal Name</i>	<i>Description</i>	<i>Unit</i>	<i>Code No.</i>
zld	Roughness length for soil	m	1
zsno	Roughness length for snow	m	2
csoilc	Drag coefficient for soil under canopy	-	3
dewmx	Maximum dew		4
wtfact	Fraction of model area with high water table		5
capr	Tuning factor to turn first layer T into surface T		6
cnfac	Crank Nicholson factor between 0 and 1		7
ssi	Irreducible water saturation of snow		8
wimp	Water impermeable if porosity less than wimp		9
pondmx	Ponding depth	mm	10
smpmax	Wilting point potential in mm	mm	11
smpmin	Restriction for min of soil poten.	mm	12
trsmx0	Max transpiration for moist soil+100% veg.	mm/s	13
tcrit	Critical temp. to determine rain or snow		14

6.4 CoLM Time-varying state variables

Table 15: Run Calendar - idate(3)

<i>Internal Name</i>	<i>Description</i>	<i>Unit</i>	<i>Code No.</i>
year	Current year of model run		1
jday	Current julian day of model run		2
msec	Current seconds of model run (0 - 86400)		3

Table 16: Time-varying Variables for restart run - fvar(numpatch, nfvar)

<i>Internal name</i>	<i>Description</i>	<i>Unit</i>	<i>Code No.</i>
<u>Main land surface variables</u>			

z (maxsnl+1:nl_soil)	Node depth	m	1:15
dz (maxsnl+1:nl_soil)	Interface depth	m	16:30
tss (maxsnl+1:nl_soil)	Soil temperature	K	31:45
wliq(maxsnl+1:nl_soil)	Liquid water in layers	kg/m ²	46:60
wice(maxsnl+1:nl_soil)	Ice lens in layers	kg/m ²	61:75
tg	Ground surface temperature	K	76
tlsun	Sunlit leaf temperature	K	77
tlsha	Shaded leaf temperature	K	78
ldew	Depth of water on foliage	mm	79
sag	Non dimensional snow age	-	80
scv	Snow cover, water equivalent	mm	81
snowdp	Snow depth	m	82
 <i><u>Vegetation dynamic parameters</u></i>			
fveg	Fraction of vegetation cover	-	83
fsno	Fraction of snow cover on ground	-	84
sigf	Fraction of veg cover, excluding snow-covered veg	-	85
green	Leaf greenness	-	86
lai	Leaf area index	m ² /m ²	87
sai	Stem area index	m ² /m ²	88
 <i><u>Radiation related (albedoes)</u></i>			
coszen	Cosine of solar zenith angle		89
albg (2,2)	Albedo, ground	-	90:93
albv (2,2)	Albedo, vegetation	-	94:97
alb (2,2)	Averaged albedo	-	98:101
ssun (2,2)	Sunlit canopy absorption for solar radiation (0-1)	-	102:105
ssha (2,2)	Shaded canopy absorption for solar radiation (0-1)	-	106:109
thermk	Canopy gap fraction for tir radiation	-	110
extkb	(k, g(mu)/mu) direct solar extinction coefficient	-	111
extkd	Diffuse and scattered diffuse PAR extinction coefficient	-	112
 <i><u>Additional variables required by regional model (WRF & RSM)</u></i>			

tad	Radiative temperature of surface	K	113
tref	2 m height air temperature	K	114
qref	2 m height air specific humidity		115
rst	Canopy stomatal resistance	s/m	116
emis	Averaged bulk surface emissivity	-	117
z0ma	Effective roughness	m	118
zol	Dimensionless height (z/L) used in Monin-Obukhov theory	-	119
rib	Bulk Richardson number in surface layer	-	120
ustar	u* in similarity theory	m/s	121
qstar	q* in similarity theory	kg/kg	122
tstar	t* in similarity theory	K	123
fm	Integral of profile function for momentum		124
fh	Integral of profile function for heat		125
fq	Integral of profile function for moisture		126

6.5 Atmospheric Forcing

Table 17: Atmospheric Forcing - forcxy(lon_points,lat_points,nforc)

<i>Internal Name</i>	<i>Description</i>	<i>Unit</i>	<i>Code No.</i>
pco2m	CO ₂ concentration in atmos. (35pa)	pa	1
po2m	O ₂ concentration in atmos. (20900pa)	pa	2
us	Wind in eastward direction	m/s	3
vs	Wind in northward direction	m/s	4
tm	Temperature at reference height	K	5
qm	Specific humidity at reference height	kg/kg	6
prc	Convective precipitation	mm/s	7
prl	Large scale precipitation	mm/s	8
psrf	Atmospheric pressure at the surface	pa	9
pbot	Atm bottom level pressure (or reference height)	pa	10
sols	Atm vis direct beam solar rad onto srf	W/m ²	11
soll	Atm nir direct beam solar rad onto srf	W/m ²	12
solsd	Atm vis diffuse solar rad onto srf	W/m ²	13
sollid	Atm nir diffuse solar rad onto srf	W/m ²	14
frl	Atmospheric infrared (longwave) radiation	W/m ²	15

hu	Observational height of wind	m	16
ht	Observational height of temperature	m	17
hq	Observational height of humidity	m	18

6.6 Fluxes Required by Atmospheric Model or Model Output

Table 18: Model Output in xy Grid Form - fldxy (lon_points,lat_points,nforc,nfldv)

<i>Internal Name</i>	<i>Description</i>	<i>Unit</i>	<i>Code No.</i>
<u><i>Fluxes required by atmospheric models</i></u>			
taux	Wind stress: E-W	kg/m/s ²	1
tauy	Wind stress: N-S	kg/m/s ²	2
fsena	Sensible heat from canopy height to atmosphere	W/m ²	3
lfevpa	Latent heat flux from canopy height to atmosphere	W/m ²	4
fevpa	Eevapotranspiration from canopy to atmosphere	mm/s	5
fsenl	Sensible heat from leaves	W/m ²	6
fevpl	Evaporation+transpiration from leaves	mm/s	7
etr	Transpiration rate	mm/s	8
fseng	Sensible heat flux from ground	W/m ²	9
fevpg	Evaporation heat flux from ground	mm/s	10
fgrnd	Ground heat flux	W/m ²	11
sabvsun	Solar absorbed by sunlit canopy	W/m ²	12
sabvsha	Solar absorbed by shaded	W/m ²	13
sabg	Solar absorbed by ground	W/m ²	14
olrg	Outgoing long-wave radiation from ground+canopy	W/m ²	15
rnet	Net radiation	W/m ²	16
xerr	Error of water balance	mm/s	17
zerr	Error of energy balance	W/m ²	18
rsur	Surface runoff	mm/s	19
rnof	Total runoff	mm/s	20
assim	Canopy assimilation rate	mol m ⁻² s ⁻¹	21
respc	Respiration (plant+soil)	mol m ⁻² s ⁻¹	22
<u><i>Model state variables</i></u>			
tss	Soil temperature	K	23-32
wliq	Liquid water in soil layers	kg/m ²	33-42
wice	Ice lens in soil layers	kg/m ²	43-52
tg	Ground surface temperature	K	53
tlsun	Sunlit leaf temperature	K	54
tlsha	Shaded leaf temperature	K	55
ldew	Depth of water on foliage	mm	56
scv	Snow cover, water equivalent	mm	57
snowdp	Snow depth	m	58

fsno	Fraction of snow cover on ground		59
sigf	Fraction of veg cover, excluding snow-covered veg		60
green	Leaf greenness		61
lai	Leaf area index		62
sai	Stem area index		63
<i>Variables required by coupling with regional models</i>			
albmdir	Averaged albedo [visible, direct]		64
albndir	Averaged albedo [near-infrared, direct]		65
albvdif	Averaged albedo [visible, diffuse]		66
albndif	Averaged albedo [near-infrared, diffuse]		67
emis	Averaged bulk surface emissivity		68
z0ma	Effective roughness	m	69
trad	Radiative temperature of surface	K	70
ustar	U* in similarity theory	m/s	71
tstar	t* in similarity theory		72
qstar	Q* in similarity theory	kg/kg	73
zol	Dimensionless height (z/L) used in Monin-Obukhov theory		74
rib	Bulk Richardson number in surface layer		75
fm	Integral of profile function for momentum		76
fh	Integral of profile function for heat		77
fq	Integral of profile function for moisture		78
tref	2 m height air temperature	K	79
qref	2 m height air specific humidity	kg/kg	80
u10m	10m u-velocity	m/s	81
v10m	10m v-velocity	m/s	82
f10m	Integral of profile function for momentum at 10m		83
<i>Atmospheric Forcing</i>			
us	Wind in eastward direction	m/s	84
vs	Wind in northward direction	m/s	85
tm	Temperature at reference height	K	86
qm	Specific humidity at reference height	kg/kg	87
prc	Convective precipitation	mm/s	88
prl	Large scale precipitation	mm/s	89
pbot	Atmospheric pressure at the surface	pa	90
frl	Atmospheric infrared (longwave) radiation	W/m ²	91
solar	Downward solar radiation at surface	W/m ²	92

7. Examples of offline simulation

In Section 2, we explained the main scripts related with the model running, most of them based on the parallel version CoLM, and the serial version running is similar to this. In fact, most of steps to run the CoLM are summarized in the script files *jobclm.csh* and *jobclm_single.csh*. But in this section, we'll give two examples on how to run the CoLM step by step without using the existing scripts, to deepen our understanding of the model running flow. And in the experiments of this section, we assume the top directory of the CoLM source code is at */home/CoLM*, and running the model on Linux or Unix system based on Intel IA32 or IA64 platform, with an Intel Fortran compiler (*ifort*) and GNU make tool (*gmake* or *make*) installed. If the netCDF format dataset is used, we also assume the netCDF software package is installed, including its header files and library files. And users should be familiar with the Linux/Unix shell environments (such as C Shell, GNU Bourne-Again SHell), also at least one editor tool (such as nano, vim, emacs).

7.1 Single Point Offline Experiment

The single point offline experiment often is very useful to examine the model performance, for example to validate some improved or newly added land surface processes. The atmospheric forcing dataset used in single point experiments usually not very huge, most of them maybe come from the observation. The land surface properties maybe also have observation values, such as the sand/clay percentage of the soil and land cover type, which are more accurate and could replace the USGS dataset. Even the initial soil temperature and soil moisture have the observed values, using these values we could set the model at a reasonable initial state, and reduce the spin-up period.

Before we compile the model source code and start the simulation experiment, we could adjust the default model configure file at */home/CoLM/include/define.h* according to the requirements by setting the different cpp tokens, for example which land cover category legend is used, whether the initial soil temperature and soil moisture values are used, or of which format atmospheric forcing is used. In this experiment, we'll use the atmospheric forcing data coming along with the default distribution of CoLM, the data is at the */home/CoLM/data/VAL.DAT.CTRL.INT*, which comes from the Valdai Grassland Site (57.6 N 33.1 E) in Russia, with a length of 18 years starting from 1962. For the format of this atmospheric forcing data is ASCII format, we'll undefine the *NCDATA* cpp token in */home/CoLM/include/define.h*. For we don't use the initial soil state values for this site, we could undefine *SOILINI* cpp token, also we use the USGS land cover category legend, and write the history data at every model step. The parallel function is also disabled for only one model grid involved in single point experiment. Finally we could get the following *define.h* (BOX 6):

BOX 6: EXAMPLE FOR COLM/INCLUDE/DEFINE.H

```
#undef COUP_CSM
#undef RDGRID
#undef SOILINI
#define offline
#undef BATS
#undef SIB2
#undef IGBP
#define USGS
#define EcoDynamics
#define LANDONLY
#undef LAND_SEA
#undef MAPMASK
#undef NCDATA
#undef PRINCETON
#undef GSWP2
#define WR_HOURLY
#undef SPMD
```

Users could refer to Section 2 of this technical guide to check the exact meaning of each cpp token. Finished the model configuration file, we could go to each model source directory and compile the model, executing the following commands (BOX 7) in Shell environment to compile the surface making program, initial data making program and the time-looping calculation program, also we copy the produced executable files into */home/CoLM/run* directory.

BOX 7: COMMANDS TO COMPILE THE MODEL

```
cd /home/CoLM/mksrfddata
make -f Makefile.intel clean
make -f Makefile.intel
cp srf.x /home/CoLM/run

cd /home/CoLM/mkinidata
make -f Makefile.intel clean
make -f Makefile.intel
cp initial.x /home/CoLM/run

cd /home/CoLM/main
make -f Makefile.intel clean
make -f Makefile.intel
cp clm.x /home/CoLM/run
```

If all goes smoothly, we'll get three executable files *srf.x*, *initial.x* and *clm.x* in */home/CoLM/run* directory, which are used to make surface data, initial data and simulate the land surface processes respectively. If something broken, firstly checking the model configuration file *define.h*, whether we set some unreasonable cpp tokens or not; Then checking *Makefile*, and making sure we have some useable Fortran 9x compiler; Finally checking the model source code if users modified some of them.

Now we should create the surface data for the point/site to be simulated. For a simple running case, we could use the provided USGS land surface data, including the land cover category, soil category. In the namelist required by the surface making program (*srf.x*), we should specify a very small region to cover the point/site to be simulated, also the exact path name which points to the USGS dataset. Using any editor tool you like to create the following namelist file (*srfdat.stdin*) at */home/CoLM/run* directory:

BOX 8: EXAMPLE NAMELIST FILE FOR CREATING SURFACE DATA

```
&mksrfexp
fmetmask      = ''
fmapmask      = ''
fgridname     = ''
fdemname      = '/home/CoLM/rawdata/DEM-USGS.30s'
fmaskname     = '/home/CoLM/rawdata/LWMASK-USGS.30s'
flandname     = '/home/CoLM/rawdata/VEG-USGS.30s'
fsolaname     = '/home/CoLM/rawdata/SOILCAT.30s'
fsolbname     = '/home/CoLM/rawdata/SOILCATB.30s'
fsurdat       = '/home/CoLM/data/srfdat.valdai'
lon_points    = 1
lat_points    = 1
edgen         = 57.625
edgee         = 33.125
edges         = 57.575
edgew         = 33.075
nlon_metdat   = 1
nlat_metdat   = 1
/
```

For single point running, we specify the *lon_points=1* and *lat_points=1*, also the boundaries surround the experiment site: *edgen*, *edgee*, *edges*, *edgew*, these variables direct the surface making program to retrieve the exact land cover category and soil category data of this site from the raw USGS dataset. The meanings of the other variables could refer to the Section 2 of this User's Guide.

At this stage, we could execute the following commands to make the surface data:

BOX 9: EXAMPLE COMMANDS TO CREATE SURFACE DATA

```
cd /home/CoLM/run  
./srf.x < srfdat.stdin >& log.srf
```

The successful running gives a prompt “*Successful in surface data making*” at the end of the **log.srf** file and a binary surface data at the **/home/CoLM/data/srfdat.valdai**. The file **log.srf** stores all information related with the surface making process.

The second major step to run the CoLM is to make the initial data, which creates two files: one storing time-constant variables (**fhistTimeConst**), such as soil physical attributes; another one storing time-varying state variables (**fhistTimeVar**), such as soil temperature and soil moisture. Firstly we create a namelist file required by the initial data making program. In this namelist file, we should specify the surface data being created in the above step. Also we should set initial date to start the time-looping calculation, which must conform to the date of the atmospheric forcing data. Most of information specified in this namelist will be copied into **finfolist** file, which later will be used as the input namelist for the time-looping program. Finally a namelist file named **inidat.stdin** is located at **/home/CoLM/run**, which contains the following clauses:

BOX 10: EXAMPLE NAMELIST FILE FOR CREATING INITIAL DATA

```
&clminiexp  
site           = Valdai  
greenwich      = .true.  
start_yr       = 1962  
start_jday     = 1  
start_sec      = 1800  
fsurdat        = '/home/CoLM/data/srfddata.valdai'  
flaidat        = ''  
fsoildat       = ''  
fmetdat        = '/home/CoLM/data/VAL.DAT.CTRL.INT '  
fhistTimeConst = '/home/CoLM/output/Valdai-rstTimeConst'  
fhistTimeVar   = '/home/CoLM/output/Valdai-rstTimeVar'  
foutdat        = '/home/CoLM/output/Valdai'  
finfolist      = '/home/CoLM/run/clmini.infolist'  
lon_points     = 1  
lat_points     = 1  
nlon_metdat    = 1  
nlat_metdat    = 1  
deltim         = 1800  
mstep          = 931104
```

/

Then we could execute the following command to make the initial data (BOX 11):

BOX 11: EXAMPLE COMMANDS TO CREATE INITIAL DATA

```
cd /home/CoLM/run  
./initial.x < inidat.stdin >& log.ini
```

A successful running of the initial data making program gives a prompt “*CLM Initialization Execution Completed*” at the end of the *log.ini*, also other three files: */home/CoLM/output/Valdai-rstTimeConst*, */home/CoLM/output/Valdai-rstTimeVar* and */home/CoLM/run/clmini.infolist*. Users could check *log.ini* to watch the process of the initial data making. The file */home/CoLM/run/clmini.infolist* contains the namelist used to run the time-looping program. In this case, it looks like the following example (BOX 12):

BOX 12: EXAMPLE NAMELIST FILE FOR TIME-LOOPING

```
&clmexp  
site           = Valdai  
flaidat        = ''  
fmetdat        = '/home/CoLM/data/VAL.DAT.CTRL.INT '  
fhistTimeConst = '/home/CoLM/output/Valdai-rstTimeConst'  
fhistTimeVar   = '/home/CoLM/output/Valdai-rstTimeVar-1962-001-  
01800'  
foutdat        = '/home/CoLM/output/Valdai'  
lhistTimeConst = 150  
lhistTimeVar   = 160  
lulai          = 120  
lumet          = 140  
luout          = 170  
lon_points     = 1  
lat_points     = 1  
nlon_metdat    = 1  
nlat_metdat    = 1  
numpatch       = 2  
deltim         = 1800  
mstep          = 931104  
/  

```

Also we should create a *flux.stdin* file to control the flux variables to export as history files, the following example will export all flux variables (BOX 13):

BOX 13: EXAMPLE NAMELIST FILE FOR FLUX-FILTER

```
&flux_nml
flux_exp= +1 +2 +3 +4 +5 +6 +7 +8 +9 +10 +11 +12 +13 +14 +15 +16
+17 +18 +19 +20 +21 +22 +23 +24 +25 +26 +27 +28 +29 +30 +31 +32
+33 +34 +35 +36 +37 +38 +39 +40 +41 +42 +43 +44 +45 +46 +47 +48
+49 +50 +51 +52 +53 +54 +55 +56 +57 +58 +59 +60 +61 +62 +63 +64
+65 +66 +67 +68 +69 +70 +71 +72 +73 +74 +75 +76 +77 +78 +79 +80
+81 +82 +83 +84 +85 +86 +87 +88 +89 +90 +91 +92
/
```

Now we could run the time-looping program to do the final single-point simulation. The commands in BOX 14 show the example:

BOX 14: EXAMPLE COMMANDS TO DO TIME-LOOPING CALCULATION

```
cd /home/CoLM/run
mv clmini.infolist timeloop.stdin
ln -sf flux.stdin fort.7
./clm.x < timeloop.stdin >& log.clm
```

And the command “*ln -sf flux.stdin fort.7*” is used to redirect the Fortran logical unit. The running of the time-looping program maybe need some time, after the model finished, we could check *log.clm* to watch if some problems occurred. In this case, the model results is saved at */home/CoLM/output*, the model restart files have the form like “*Valdai-rstTimeVar-YEAR-DAY-SECOND*”, history files have the form like “*Valdai-YEAR-DAY-SECOND*”, which contain the simulation results. Users could refer to the *GrADS* description file at */home/CoLM/graph/flx.ctl* to plot the results according to your requirements.

In the single point experiment, users could also replace the land surface data derived from USGS raw dataset with the observation values, such as sand/clay percentage, land cover category, bedrock depth. The easiest way to complete this is to modify the value of the corresponding variables before the surface making program writes the surface data, and users could refer to the relevant code fragment in the source file

/home/CoLM/mksrfddata/mksrfddata.F90. The initial value of the soil temperature and soil moisture also could be changed, users could refer to the code fragment covered by the cpp token *SOILINI* in the file */home/CoLM/mkinidata/initialize.F90*.

7.2 Global Offline Experiment with GSWP2 Dataset

The global offline experiment is similar with the single point offline experiment, most of the namelist files are also similar, only the number of the model grids and the atmospheric forcing data has some difference, and the running flow is same. In this experiment, we'll skip those similar steps, and only focus on how to prepare the forcing data for a global offline experiment, using the GSWP2 dataset as an example.

In Section 7.1, the single point offline experiment uses the atmospheric forcing data of the ASCII format, the time-looping program uses the subroutine *GETMET* in */home/CoLM/main/GETMET.F90* source file to handle this type forcing data. But when using GSWP2 dataset, which is of netCDF format, the subroutine *ncdata_read* in */home/CoLM/main/ncdata.F90* is used. Currently this code only support pre-processed *GSWP2* and *PRINCETON* dataset.

As stated in Section 4, CoLM usually uses a model time step of 30 minutes, and most of the re-analysis data products have a time interval of 3 hours. To eliminate this gap, we could do a temporal interpolation for the raw re-analysis data. And in the default distribution of the parallel version CoLM, some temporal interpolation subroutines based on Cubic Spline method are provided, these subroutines are not perfect, users are encouraged to improve them. In this section we'll explain how to use these subroutines to per-process the GSWP2 dataset.

Most of the information about the GSWP2 dataset has been stated in Section 4, here we'll demonstrate how to interpolate the GSWP2 data and feed them to the CoLM. We'll use the short wave solar radiation dataset as an example. Assuming the original GSWP2 solar radiation dataset is at */home/gswp2/SWdown_srb*. We could use the command *ncdump* provided with the netCDF software package to check the file information of these GSWP2 dataset. Now we'll interpolate the 3-hour interval original GSWP2 solar radiation data into 30-minute interval, using the interpolation program provided in the default CoLM distribution. Firstly we compile the interpolation program, the netCDF package is assumed being install at */usr/local/netCDF* directory, the BOX 15 shows the example commands.

BOX 15: EXAMPLE COMMANDS TO COMPILE THE INTERPOLATION PROGRAM

```
cd /home/colm/interp/src

ifort -c spline_interp.F90
ifort -c -fpp -DGSWP2 -I/usr/local/netCDF/include
data_io.F90
ifort -c -fpp -DGSWP2 -I/usr/local/netCDF/include
SW_interp.F90
ifort -o SW_interp.x spline_interp.o data_io.o
SW_interp.o -L/usr/local/netCDF/lib -lnetcdf
```

The successful compilation produces the *SW_interp.x* program, which is used to interpolate the GSWP2 solar radiation dataset. Now we create an input file to list the number of files and the variable to interpolate, also the original files' name and the output files' name. In original GSWP2 dataset, *SWdown* is the variable to store the solar radiation dataset. BOX 16 gives a simple example:

BOX 16: EXAMPLE INPUT FILE TO CONTROL THE INTERPOLATION

```
1
'SWdown'
'/home/gswp2/SWdown_srb/SWdown_srb198207.nc'
'/home/gswp2/SWdown_srb/SWdown_srb198207_30min.nc'
```

Saving the above content into the file *gswp_sw.stdin*, and using it as the input file to the *SW_interp.x* program, with the executing the command “*./SW_interp.x < gswp_sw.stdin*”, we'll get the solar radiation dataset of 30-minute interval. The procedures to interpolate other GSWP2 atmospheric forcing dataset are similar, and we'll skip them here.

The netCDF files produced by the interpolation program are of the format required by the *ncdata.F90* in time-looping calculation program. Introducing any new netCDF format data, users should pre-process them according to the requirements stated in Section 4, the detailed information about the file format, users could refer to the *ncdata.F90* source code file.

With these processed netCDF format GSWP2 atmospheric forcing dataset, we could repeat the steps in the single point offline experiment, with some little modification to the namelist files, to run a global offline experiment.

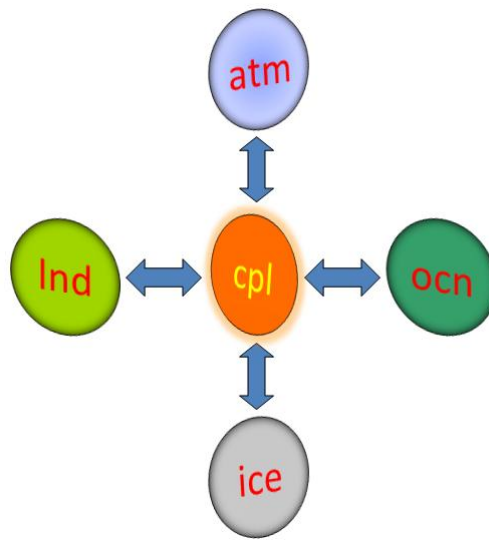
8. Coupling of CoLM with CSM/ESM

In above several sections, we explained the procedures to make surface data and initial data, also basic steps to carry out single point or regional/global offline simulations. When carrying out offline simulations, we need near surface meteorology fields as upper boundary data to drive land surface model. On the other hand, land surface as a part of earth system, its surface albedo, evapotranspiration, latent and sensible heat fluxes all affect the evolution of the upper atmosphere at many different time scales, so the land surface model becomes an important component in contemporary climate system models (CSMs) or earth system models (ESMs). In this section, we'll discuss the basic principles on coupling CoLM with CSM or ESM, also some modules helping to build an integrated CSM or ESM. Especially, we'll use the coupling between CoLM and GCESSM (Global Change Consortium - Earth System Model) as an example. Before explain the detailed coupling procedures, we'll give a brief introduction on the general framework of contemporary CSM and ESM.

8.1 General framework of CSM/ESM

As the performance of super computer advances rapidly, it becomes possible to consider more and more physics or chemistry processes in CSM, also increase the spatial resolution of CSM. With explicit consideration of biogeochemistry cycles in traditional CSM, especially focusing on terrestrial and marine carbon cycle, even nitrogen, phosphorus, ecosystems and human earth interactions, CSM evolves to ESM, which describes the earth climate system more comprehensively and more accurately. At the same time, CSM and ESM grow into super complex software systems, which push a big burden on model development and maintenance. In contemporary CSM or ESM, to simplify the model development and decrease the model complex, model communities adopt a modular framework to define the whole structure and interactions among different components. In this framework, different model or component represents a different part of earth climate system, then all components interact with each other through a central component to exchange fluxes at interfaces, this central component is usually called coupler. With this new modular framework, CSM and ESM could also maintain a good computing scalability on contemporary super computer architecture. The Figure 7 is a general framework being widely used in contemporary CSM or ESM, around the central coupler, there're atmosphere (atm), ocean (ocn), land (lnd), sea ice (ice) components, each component represents a single model to simulate its part of earth climate system, and they interact with each other via coupler (cpl) by sending or receiving its boundary flux to the coupler.

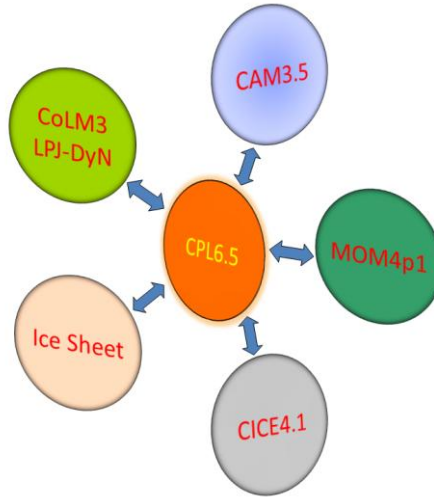
Figure 7: General framework of CSM/ESM



8.2 Coupling with GCCESM

The GCCESM is an Earth System Model which was built up to improve our understanding of global changes and human-earth interactions (Figure 8). Besides one central component, GCCESM currently contains four separate models simultaneously simulating the earth's atmosphere, ocean, land surface and sea-ice, an ice-sheet component based on GLIMMER is under coupling. The initial framework of GCCESM is based on the atmosphere model CAM3.5 from National Center for Atmospheric Research (NCAR), the ocean model MOM4p1 (2009 version) from Geophysical Fluid Dynamics Laboratory (GFDL), the land surface model CoLM3 from Beijing Normal University (BNU), the sea ice model CICE4.1 from Los Alamos National Laboratory (LANL), the coupler and software framework are based on CCSM3.5 from National Center for Atmospheric Research (NCAR). Lots of further work was integrated into GCCESM by Global Change Consortium of China after the initial framework was built up. The College of Global Change and Earth System Science (GCESS) at Beijing Normal University (BNU), as the founder of the Global Change Consortium of China, contributed much important work to GCCESM, especially on MOM4p1, CICE4.1, CoLM3 components coupling and biogeochemistry cycle modeling, such as carbon-nitrogen coupled terrestrial biogeochemistry scheme based on Lund-Postdam-Jena (LPJ) dynamic vegetation model.

Figure 8: The framework of GCCESM



As we all know, in offline mode, land model reads near surface atmospheric forcing data from files provided by model users. But in coupled mode, the near surface atmospheric forcing fields are simulated by atmosphere model, and the bottom boundary conditions required by atmosphere model are simulated by land model. To run two models continuously, they have to exchange fluxes at the interface. Under the framework of contemporary CSM/ESM, such as GCCESM, land model and atmosphere model don't exchange fluxes directly, but send them to coupler, coupler will pass necessary fields to each component. In this course, coupler could do further work like regridding, mapping, fluxes checking and so on in a more general manner. The communications among different components and coupler usually use Messages Passage Interface (MPI) based library to establish, send and receive. In the following section, we'll focus on the interaction between land component and coupler, and use GCCESM and its land component CoLM as an example to explain the basic principle on the coupling procedure.

To establish communication with coupler in CoLM, we should add some interfaces in land model to interact with coupler, such as sending or receiving fluxes or status variables. Except normal surface temperature, albedo, sensible and latent heat fluxes, land model should send the river runoff to ocean model to maintain the water mass balance of the whole earth climate system. The river runoff to ocean was calculated by and River Transport Module (RTM) in CoLM. The RTM enables the hydrologic cycle to be closed in global models, and helps to improve ocean convection and circulation simulations, which is affected by freshwater input. The RTM in CoLM uses a linear transport scheme at 0.5° resolution to route water from each grid cell to its downstream neighboring grid cell. In ESM, for maintaining the carbon cycle, land model also should send the net ecosystem exchange (NEE) flux to atmosphere model via coupler to calculate the CO₂ concentration, and receive CO₂ concentration of bottom atmosphere layer. All these sending and receiving functions use same procedures provided by coupler, here we'll only give a table

showing which files and interfaces are added in CoLM to wrap complex coupler procedures.

Table 19: Interfaces added to couple with GCCESM

Files (interfaces) added in main/ directory to couple with GCCESM		
colm_cplMod.F90	colm_cpl_init	Initialize variables used to pass fluxes to coupler
	colm_cpl_l2a	Extract fluxes simulated by land model, prepare to send them to coupler
	colm_cpl_a2l	Extract atmospheric forcing fluxes received by coupler
	colm_cpl_exit	Free resources used by coupling
colm_csmMod.F90	csm_setup	Setup communication partially
	csm_shutdown	Shutdown the communication with coupler
	csm_initialize	Initialize communication between land model and coupler
	csm_dosndrcv	Check whether to send or receive fluxes
	csm_rcv	Receiving atmospheric status and fluxes from coupler
	csm_send	Sending fluxes of land model to coupler
	csm_sendalb	Sending 4 bands land albedo to coupler at the first time step.
	csm_flxave	Average land model fluxes sending to coupler
	csm_restart	Read or write restart information about coupling between land & coupler.

The following table shows the variables exchanged between CoLM and coupler in GCCESM, all sending or receiving communication use the procedures listed in the Table 20.

Table 20: Fields exchanged between CoLM and coupler in GCCESM

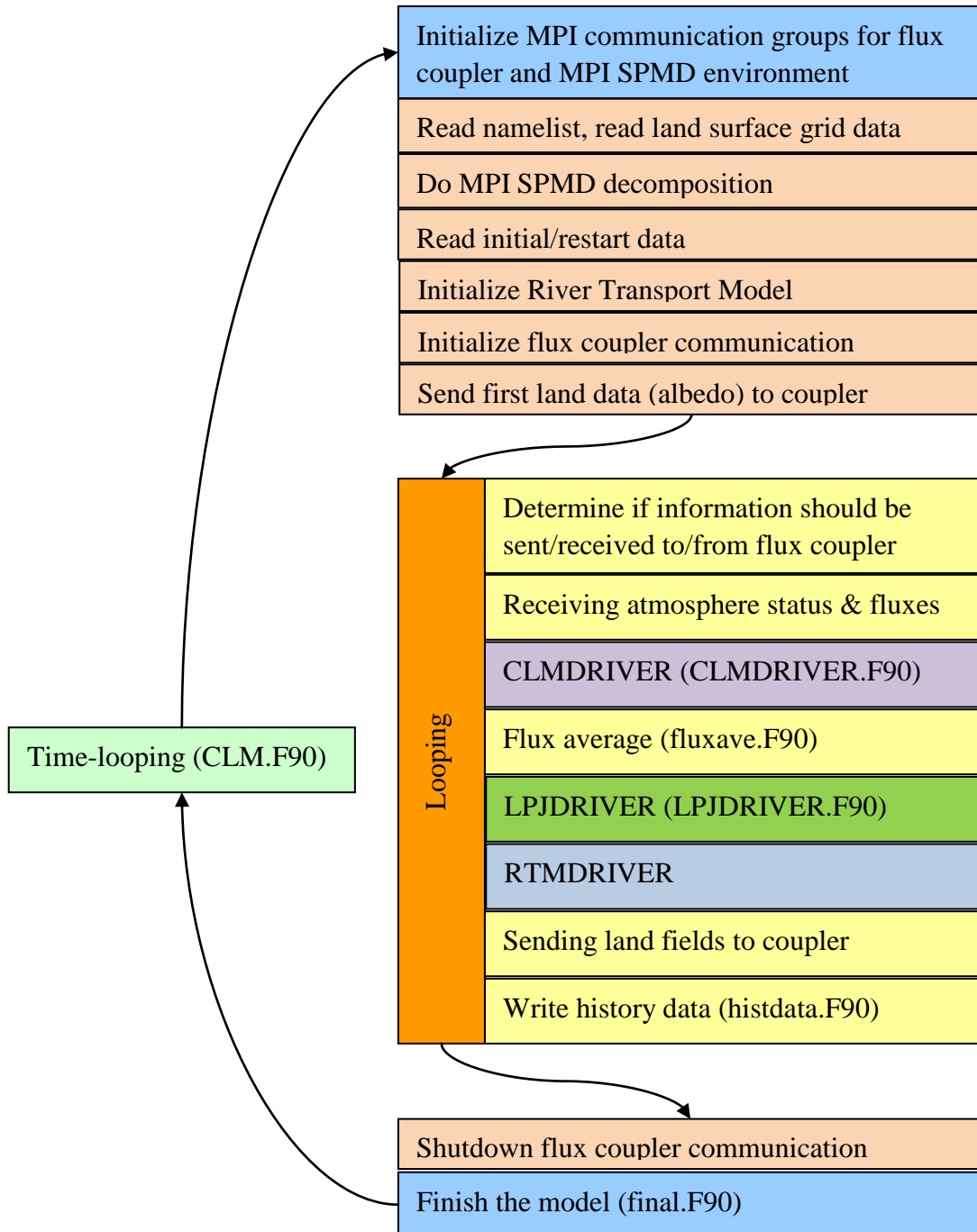
Fields exchanged between CoLM and coupler in GCCESM		
Sending fields from land model	taux	wind stress: E-W [kg/m/s^2]
	tauy	wind stress: N-S [kg/m/s^2]
	fsena	sensible heat from canopy height to atmosphere [W/m^2]

	lfevpa	latent heat flux from canopy height to atmosphere [W/m ²]
	fevpa	evapotranspiration from canopy to atmosphere [mm/s]
	swabs	net absorbed solar radiation [W/m ²]
	olrg	outgoing long-wave radiation from ground+canopy [W/m ²]
	avsdrr	averaged albedo [visible, direct]
	avsdff	averaged albedo [visible, diffuse]
	anidrr	averaged albedo [near-infrared, direct]
	anidff	averaged albedo [near-infrared,diffuse]
	Trad	radiative temperature of surface [K]
	tref	2 m height air temperature [K]
	qref	2 m height air specific humidity [kg/kg]
	scv	snow cover, water equivalent [mm]
	nee	net ecosystem exchange flux [mol C/m ² /s]
	roff	river flux to the ocean (m ³ /s)
Receiving from coupler	co2_ppmv	CO2 concentration of the bottom atmosphere layer
	pbot	pressure of the bottom atmosphere layer [Pa]
	u	zonal wind of the bottom atmosphere layer [m/s]
	v	meridional wind of the bottom atmosphere layer [m/s]
	tbot	air temperature of the bottom atmosphere layer [K]
	shum	air humidity of the bottom atmosphere layer [kg/kg]
	rainc	liquid Convective precipitation rate [kg/m ² /s]
	rainl	liquid Large scale precipitation rate [kg/m ² /s]
	snowc	convective snow rate [kg/m ² /s]
	snowl	large scale snow rate [kg/m ² /s]
	swvdr	downward visible direct shortwave radiation flux [W/m ²]
	swvdf	downward visible diffuse shortwave radiation flux [W/m ²]
	swndr	downward near-infrared direct shortwave radiation flux [W/m ²]
	swndf	Downward near-infrared diffuse shortwave radiation flux [W/m ²]
	lwdn	downward long wave heat flux [W/m ²]
	dens	air density of the bottom atmosphere layer [kg/m ³]
	z	height of the bottom atmosphere layer [m]

To adapt for the communication mechanisms required by coupler, some initial parts in CoLM time-looping part have to be modifies. The following figure demonstrates the updated computing flow of the main part of CoLM in coupled mode. In this figure,

we also included the River Transport Module and LPJ based DGVM scheme, to reflect the whole structure of CoLM in GCCESM.

Figure 9: Flow Chart of the Time-looping Calculation in Coupled Mode



[End of Document]