Chapter 9

Configuration and Execution

In this chapter, configuration, flowchart of data and execution of ***CReSS*** will be described. Two types of ***CReSS*** are present: one is the serial version for a single processor and the other is the parallel version for multiple processors. This chapter describes mostly about the parallel version of ***CReSS***. The configuration and execution of the serial version are essentially the same as those of the parallel version and the differences between the parallel and serial versions will be unspecified.

CReSS has two configuration files: one is `compile.conf` which configures the compile commands and their options, and the other is `user.conf` which controls both compile and run. The former is depend on the machine to be used while the latter is independent of mostly except `\& sysdep`. In this chapter, the latter configuration file will be explained in detail.

The configuration file "user.conf" is written as the format of the Fortran namelist. The user.conf is used for not only run but also the compile. All the dimensions of array arguments are specified by the user.conf. Users, therefore, will never change the source codes of ***CReSS***.

The flowchart of data will also described as well as data formats. There are different data flows which depend on the configuration of calculation.

At the last part of this chapter, an example run of experiment of the Kelvin-Helmholtz waves will be shown: compilation of solver, run the solver and output the result. Compilations and executions of the pre-processors will be also presented.

9.1 User configuration file

9.1.1 Notifications

The user configuration file of ***CReSS***is a namelist of Fortran. The followings are important notices to use the file.

- Since the configuration file is used by a shell script in the compilation of ***CReSS***, a space is necessary to separate between a variable name and "=" and between a value and "=".
- The solver and the pre-processors have dependency of some namelist variables. When a discrepancy of variables between pre-processors and the solver is found, the program will stop execution and output an error.

9.1.2 Details of the user configuration file

& sysdep

The namelist variable is depend on machines. All programs use the variable, while no dependence is present between the programs.

`wlngth` — *integer* $*$ 4
Word length of binary data in the direct access and unformatted. Most machines take `wlngth` as 4, while some others take 1 (eg. Digital UNIX, NEC SX6).

& dmiset

This namelist variables set numbers of grid points of calculation domain and numbers of decomposition in parallel computing. All programs use the variables and have dependency. **(Recompiling is necessary if changed.)**

`xpedim,ypedim` — *integer* $*$ 4
Numbers of decomposition of the calculation domain in x and y directions. The total numbers of processing elements (CPUs) are $x \times y$.

`xdim,ydim` — *integer* $*$ 4
Numbers of grid points of the domain in the x and y directions, respectively. Since the horizontal grid system is the Arakawa C type, the numbers of the scalar grid points are $xdim - 1$ and $ydim - 1$ in the x and y directions, respectively. These numbers must satisfy the following relations in order to each processing element is loaded with the same number of grid points.

$$[\texttt{xdim} + 3(\texttt{xpedim} - 1)]/\,\texttt{xpedim} = integer$$
$$[\texttt{ydim} + 3(\texttt{ypedim} - 1)]/\,\texttt{ypedim} = integer$$

These relations means that numbers of grid points of the physical domain ($\texttt{xdim} - 3, \texttt{ydim} - 3$) must be divided by each number of decomposition.

`zdim` — *integer* $*$ 4
Number of grid point in the z direction. Since the vertical grid system is the Lorenz type, the number of scalar grid points is $zdim - 1$. Using the parameters of `cphopt = 2` or `cphopt = 3`, `zdim` must be larger or equal 12 because of use of temporary array.

& expname

The namelist specifies the name of the experiment, which is used by *solver*, *gridata*, *terrain*. Dependency is present between these programs.

`exprim` *character* $*$ 80
Name of numerical experiment. The number of character used in the variable must be less than or equal to 80 characters. The variable is used in input and output data file names. For example, the sounding data file will be *exprim*`.sounding.txt`. No special characters are allowed to be used.

& project

The namelists define the horizontal coordinate systems of the calculation domain. They are used *solver*, *gridata*, and *terrain*. Dependency is present between these programs.

`mpopt` *integer* $*$ 4
Option for map projection of the domain. The following four projections are available.
- `1`: polar stereo graphic projection
- `2`: Lambert conformal projection
- `3`: Mercator projection
- `4`: No map projection (latitude and longitude)

`nspol` *integer* $*$ 4
Switch to define origin of the coordinate in the northern or southern hemisphere.
- `1`: the origin of coordinate in the northern hemisphere.
- `-1`: the origin of coordinate in the southern hemisphere.

`tlat1,tlat2` *real* $*$ 4
The true latitudes of map projections. No deformation due to projection occurs at the latitudes. The unit is degree [°] and they are negative in the southern hemisphere. If the Lambert conformal projection is used, two true latitude (`tlat1` and `tlat2`) are necessary. If no map projection is used, they are not necessary to be specified.

`tlon` *real* $*$ 4
The true longitude of coordinate. The unit is degree [°] and degree in the West Longitude is negative. The y-axis of the model domain corresponds to the longitude. Even in the case of `mpopt = 4`, `tlon` must be specified.

& gridset

The namelists determine grid intervals and reference longitude and latitude. They are used in *solver*, *gridata*, and *terrain*. Dependency is present between these programs.

`dx,dy,dz` *real* $*$ 4
Grid intervals in the x, y and z directions, respectively. The unit is meter. If non-uniform vertical grid is used (`sthopt = 1,2`) , `dz` is the averaged grid interval.

`ulat,ulon` *real* $*$ 4

Reference longitude and latitude, respectively, which corresponds to a single point within the domain. The unit is degree. Degrees in the southern hemisphere and in the West Longitude are negative. ***CReSS***specifies the region of the domain using the reference longitude and latitude with the following real number of `riu` and verb"rju".

`riu,rju` *real* ∗ 4
The real number of the reference point which corresponds to the point determined by the reference longitude and latitude (`ulat,ulon`).

& gridsth

The namelists determine vertical grid spacings in non-uniform coordinate. They are used in *solver*, *gridata*. Dependency is present between these programs.

`zsfc` *real* ∗ 4
Altitude of the surface. The unit is meter. In most cases, `zsfc = 0.0` m.

`zflat` *real* ∗ 4
The lowest height of the flat level. Unit is meter.

`sthopt` *integer* ∗ 4
Option for vertical stretching.
- `0`: No stretching (uniform grid interval).
- `1`: Stretching using a cubic function.
- `2`: Stretching using a function of tanh.

`dzmin` *real* ∗ 4
Minimum grid interval at the lowest level. Unit is meter.

`layer1,layer2` *real* ∗ 4
In stretching, the grid interval of `dzmin` is used up to a height of `layer1`, the intervals determined by functions between heights of `layer1` and `layer2`, and the grid interval at a height of `layer2` is used above the height of `layer2`. Unit is meter.

& terrain

The namelists specify terrain topography of the model. They are used in the program of *solver* and *gridata*. Dependency is present between these programs.

`trnopt` *integer* ∗ 4
Option for model terrain.
- `0`: the flat ground.
- `1`: the bell-shaped mountain.
- `2`: user specified terrain from the external data file.

`mnthgh` *real* ∗ 4
Height (m) of the summit of the bell-shaped mountain, if `trnopt = 1`

`mntwx,mntwy` *real* ∗ 4
Half width (distance from the center of a half height) of the bell-shaped mountain in the x and y directions, respectively, if `trnopt = 1`. Unit is meter.

`mntcx,mntcy` *real* $*$ 4
The coordinates x, y of the center of the bell-shaped mountain, if `trnopt = 1`. Unit is meter. The relationship between the coordinates and their array number `i,j` is as follows.
$x = (\texttt{i} - 2) \times \texttt{dx}$
$y = (\texttt{j} - 2) \times \texttt{dy}$

& flength

The namelists are with respect to time of calculation. They are used in *solver* and *gridata*. The parameters with $*$ have dependency between these programs.

`sfcast`* *character* $*$ 16
Date and time of the initial time of calculation in the Universal Time Coordinate (UTC). The format is `sfcast = '`*yyyy/mm/dd hh*`:`*mm*`'`. For example, when the initial time is 13:00 UTC, 22 September 2000, `sfcast = '2000/09/22 13:00'`.

`stime` *real* $*$ 4
Initial time of calculation. Unit is second. The time is measured from the time of `sfcast`. In the first calculation, `stime = 0.0`, and In a restart, `stime` is must be the restart time.

`etime` *real* $*$ 4
Time of the termination of calculation. Unit is second. The time is also measured from the time of `sfcast`. For example, `etime` must be 5400.0 to calculate 1800.0 seconds from 3600.0 seconds.

& extpram

The namelists are parameters of external forcing using an external data. They are used in *solver* and *gridata*. The parameters with $*$ have dependency between these programs.

`extvar`* *character* $*$ 80
Specify variables to be inputted to the model from an external data. The first 7 characters of 80 are used. Each character from the first corresponds to the vertical velocity component, water vapor mixing ratio (or relative humidity), mixing ratios of cloud water, rain, cloud ice, snow and graupel. On the other hand, z-coordinate, horizontal velocity components, pressure, potential temperature (or temperature) are indispensable and included without this specification. For example, the vertical velocity component, water vapor mixing ratio (or relative humidity) and mixing ratio of rain are used (when they are available), then `extvar = 'ooxoxxx'`
`o`: the variable is inputted.
`x`: the variable is not inputted.

`extitv`* *real* $*$ 4
Time interval of the external data. Unit is second.

`ndgopt` *integer* $*$ 4
Option for nudging of an analysis data.
`0`: no nudging.
`1`: perform the nudging.

`ndgvar` *character* ∗ 80
Specify variables to be used in nudging. The first 11 characters of 80 are used. Each character from the first corresponds to u and v-components of velocity, vertical velocity component, pressure, potential temperature (or temperature), water vapor mixing ratio (or relative humidity), mixing ratios of cloud water, rain, cloud ice, snow and graupel. For example, when all the three components of velocity, pressure and potential temperature are used for nudging, `ndgvar = 'oooooxxxxxx'`. All variables besides the indispensable variables must be specified by `extvar`.
- `o`: the variable is used in nudging.
- `x`: the variable is not used in nudging.

`ndgcoe` *real* ∗ 4
Coefficient of dumping (e-folding time of dumping) of the nudging of the analysis data.

`ngdelt` *real* ∗ 4
Time interval of nudging. Unit is second. The parameter must be divided by `dtb` and larger than or equal to `dtb`.

`ngtime` *real* ∗ 4
Time to terminate the nudging (measured from the initial). Unit is second.

`exbopt`* *integer* ∗ 4
Option for forcing to external data at the lateral boundary. The forcing is performed with respect to the indispensable variables and variables determined by `extvar`.
- `0`: no forcing at the lateral boundary.
- `1`: forcing to the external data at the lateral boundary is performed.
- `2`: forcing to the external data at the lateral boundary is performed and the terrain is also smoothed to be the external terrain in the lateral sponge layer.
- `3`: forcing to the sounding data at the lateral boundary is performed.

`lspvar` *character* ∗ 80
Specify variables to be forced in the lateral sponge layer. Variables besides the indispensable variables must be determined `extvar`. Specification is the same as `ndgvar`.

`lspspn`* *integer* ∗ 4
Number of the lateral sponge layer. Dependency is present in the case of `refsfc = 1`.

`lspcoe` *real* ∗ 4
Coefficient of the lateral forcing in the lateral sponge layer. The forcing decrease with the distance from the lateral boundary.

`vspopt` *integer* ∗ 4
Option for dumping in the top sponge layer.
- `0`: no dumping in the top sponge layer.
- `1`: dumping in the top sponge layer is performed. The variables are forced to be those at the initial time.

`vspcoe` *real* ∗ 4
Coefficient of the dumping in the top sponge layer. The forcing decreases with the distance from the top of the model.

`zsplow` *real* ∗ 4

The lowest height of the upper dumping. Unit is meter.

& boundry

The namelists determine the lateral boundary condition. They are used in *solver* and *gridata*. The parameters with ∗ have dependency between these programs.

`wbc*,ebc*,` `sbc*,nbc*` — *integer* ∗ 4

Option for lateral boundary conditions at the west, east, south and north boundaries, respectively.

- `1`: periodic boundary condition.
- `2`: rigid wall boundary condition.
- `3`: zero-gradient boundary condition.
- `4`: wave-radiating boundary condition.
- `5`: wave-radiating boundary condition using vertically averaged phase velocity of normal velocity component.

`bbc,tbc` — *integer* ∗ 4

Option for bottom and top boundary condition, respectively. If `impopt = 1`, then these parameters are forced to be `3`.

- `2`: rigid wall boundary condition.
- `3`: zero-gradient boundary condition.
- `4`: wave-radiating boundary condition at the top boundary condition.

& sfcphys

The namelists determine the surface conditions and the ground model. **(Recompiling is necessary if changed.)**

`sfcopt` — *integer* ∗ 4

Option for the surface process.

- `0`: no surface process.
- `1`: surface process is performed.

`sfcdat*` — *character* ∗ 80

Specify the external data of surface process. The first 5 characters of 80 are used. Each character from the first corresponds to land use, land surface temperature, sea surface temperature, surface temperature of sea ice, and surface temperature of snow. For example, when the land use and the land surface temperature are available, then, `sfcdat = 'ooxxx'`.

- `o`: the variable is available and will be used.
- `x`: the variable is not used.

`levpbl` — *integer* ∗ 4

Number of the planetary boundary layer. The thickness of each layer is dz and $1 \leq$ `levpbl` $\leq$ `zdim - 2`.

`levgrd` — *integer* ∗ 4

Number of layers of the ground model.

`dzgrd` — *real* ∗ 4

Thickness of the layers of the ground model. Unit is meter. The total depth of the ground model is desirable to be larger than a few meters.

`cbeta` *real* ∗ 4
A constant evapotranspiration coefficient (moisture availability form the surface) used in the case of no land use data.

`calbe` *real* ∗ 4
A constant albedo used in the case of no land use data.

`cz0m` *real* ∗ 4
A constant roughness over the land used in the case of no land use data. Unit is meter.

`ctdeep` *real* ∗ 4
A constant ground temperature at the deepest layer used in the case of no ground temperature data. Unit is K.

`ctsea` *real* ∗ 4
A constant sea surface temperature used in case of no SST data. Unit is K.

& initype

The namelists determine the initialization of model. Only in *solver*. **(Recompiling is necessary if changed.)**

`iniopt` *integer* ∗ 4
Option for the model initialization.
- `1`: read the sounding data.
- `2`: read the restart file.
- `3`: read 3-dim non-uniform grid data.
- `4`: In addition to `3`, retrieval is performed with respect to the prognostic variables.

`snddim` *integer* ∗ 4
Number of levels of the sounding data if `iniopt = 1`.

`sndhed` *integer* ∗ 4
Number of comments in the header of the sounding data if `iniopt = 1`. The comment lines must be at the beginning of the sounding file.

`sndtyp` *character* ∗ 80
Specify data type of the sounding if `iniopt = 1`. The following combinations are available except the horizontal velocity.
- `ppk`: 1st pressure [Pa] , 2nd θ [K] , 5th q_v [kg kg^{-1}]
- `zpk`: 1st height [m] , 2nd θ [K] , 5th q_v [kg kg^{-1}]
- `ptk`: 1st pressure [Pa] , 2nd temperature [K] , 5th q_v [kg kg^{-1}]
- `ztk`: 1st height [m] , 2nd temperature [K] , 5th q_v [kg kg^{-1}]
- `ppp`: 1st pressure [Pa] , 2nd θ [K] , 5th Rh [%]
- `zpp`: 1st height [m] , 2nd θ [K] , 5th Rh [%]
- `ptp`: 1st pressure [Pa] , 2nd temperature [K] , 5th Rh [%]
- `ztp`: 1st height [m] , 2nd temperature [K] , 5th Rh [%]

`zsnd0,psnd0` *real* ∗ 4
Height [m] and pressure [Pa] of the lowest level of the sounding data, respectively. If `sndtyp(1:1) = 'p'`, then `zsnd0` is used and if `sndtyp(1:1) = 'z'`, then `psnd0` is used.

`retvar` *character* $*$ 80
Specify variables to be performed retrieval. First 3 characters of 80 are used. Each character corresponds to the 3-dim. velocity components, pressure, and potential temperature and mixing ratio of hydrometeors. For example, if the retrieval is performed with respect to the 3-dim. velocity components and pressure, then `retvar = 'oox'` .
`o`: retrieval is performed.
`x`: no retrieval is performed.

`retime` *real* $*$ 4
Integration time of retrieval with respect to the potential temperature and mixing ratios of hydrometeors.

`alpha1,alpha2` *real* $*$ 4
Weighting coefficients of horizontal and vertical directions using the static stability if `iniopt = 4`.

& gridmove

The namelists configure the horizontal displacement of the calculation domain and are used only in *solver*.

`movopt` *integer* $*$ 4
Option for the displacement of the domain. This option is valid when `trnopt = 0`, `iniopt = 1` and `exbopt = 0`.
`0`: No displacement of the domain.
`1`: Displacement of the domain is switched on.

`umove,vmove` *real* $*$ 4
Speeds of the displacement of the domain in the x and y directions, respectively. Unit is [m s^{-1}].

& ptinicon

The namelists set perturbations of the potential temperature at the initial time and are used only in *solver*.

`pt0opt` *integer* $*$ 4
Option for perturbations of potential temperature at the initial time. This parameter is valid when `iniopt = 1`.
`0`: No initial perturbation of potential temperature.
`1`: One or more elliptic perturbations are placed in the x direction.
`2`: One or more elliptic perturbations are placed in the y direction.
`3`: Perturbation of trigonometric function in the x direction is set.
`4`: Perturbation of trigonometric function in the y direction is set.
`5`: Random perturbations are generated between the specified two levels.

`pt0num` *integer* $*$ 4
Number of the elliptic perturbation of potential temperature when `pt0opt = 1` or `2`.

`ptp0` *real* $*$ 4
Maximum potential perturbation at the initial time . Unit is Kelvin.

`pt0rx,pt0ry,` *real* $*$ 4

`pt0rz` Radii of the elliptic perturbations, or half wave length of the trigonometric function or half width of the random perturbation layer in z direction (`pt0opt = 5`). Unit is meter.

`pt0cx,pt0cy, pt0cz` *real* $*$ 4
Center of the elliptic perturbations, or the origin of the trigonometric function or middle height of the random perturbation layer in z direction (`pt0opt = 5`). Unit is meter.

`pt0ds` *real* $*$ 4
Distance between bubbles of potential temperature in the case of `pt0opt = 1` or `2`. Unit is meter.

& integrat

The namelists define the parameters of time integration such as time intervals and coefficients of the Asselin's time filter. Only in *solver*.

`dtbig,dtsml` *real* $*$ 4
Time intervals of the large time step and the small time step, respectively. The terms related with the acoustic waves are integrated in the small time step and others in the large time step. If `gwmopt = 1`, then the terms of the gravity waves are integrated in the small time step. Unit is second. These time intervals must satisfy the CFL condition.

`gwmopt` *integer* $*$ 4
Option for the time integration of the terms related with the gravity waves.
- `0`: They are integrated in the large time step.
- `1`: They are integrated in the small time step.

`impopt` *integer* $*$ 4
Option for time integration scheme of the acoustic wave mode in the vertical direction.`impopt = 1` is recommended.
- `0`: Explicit scheme.
- `1`: Implicit scheme using Gaussian elimination.
- `2`: Implicit scheme using Gaussian elimination with pivot option.
- `3`: Implicit scheme using Gauss-Zaidel method.

`weicoe` *real* $*$ 4
Weighting coefficient of the implicit scheme.

`filcoe` *real* $*$ 4
Weighting coefficient of the Asselin's time filter.

& advction

The namelist is the option of the advection scheme. Only in *solver*.

`ad4opt` *integer* $*$ 4
Option for the advection scheme. The center difference is used in the advection scheme. The numerical smoothing is, therefore, necessary to suppress the computational instability.

0: the second order center difference.
1: the fourth order center difference.

& smoother

Configurations of the numerical smoothing are determined in the namelists. Only in *solver*.

`smtopt` *integer* $*$ 4
Option for the numerical smoothing which works as an artificial viscosity to suppress the numerical noise of the advection scheme.
0: No numerical smoothing.
1: The second order numerical smoothing.
2: The fourth order numerical smoothing.

`smndch` *real* $*$ 4
Non-dimensional coefficient of the artificial viscosity in the horizontal direction. Recommended that `smndch`/`dtb`= 0.001. The coefficients of viscosity ν_{2h}, ν_{4h} are defined as
Second order: $\nu_{2h} =$ `smndch` $\times$ (`dxdy`)/ `dtb`
Fourth order: $\nu_{4h} =$ `smndch` $\times$ (`dxdy`)^2/ `dtb`

`smndcv` *real* $*$ 4
Non-dimensional coefficient of the artificial viscosity in the vertical direction. Recommended that `smndcv`/`dtb`= 0.001. The coefficients of viscosity ν_{2v}, ν_{4v} are defined as
Second order: $\nu_{2v} =$ `smndch` $\times$ `dz`^{2}/ `dtb`
Fourth order: $\nu_{4v} =$ `smndch` $\times$ `dz`^{4}/ `dtb`

& mapfcter

The namelist is an option for map factors. Only in *solver*. Map factors are not implemented in the version of 1.3 or earlier.

`mfcopt` *integer* $*$ 4
Option for map factor.
0: No map factor is used.
1: Map factor is used.

& coriolis

Configuration of the Coriolis force terms. Only in *solver*.

`coropt` *integer* $*$ 4
Option for the Coriolis force terms.
0: No Coriolis force terms.
1: Horizontal components of the Coriolis force are calculated.
2: Both horizontal and vertical components of the Coriolis force are calculated.

& buoyancy

Switching the buoyancy term of pressure perturbation in the vertical component of the momentum equation. This term is related to the acoustic waves. Only in *solver*.

`buyopt` *integer* ∗ 4
Option for the buoyancy term.
- `0`: The buoyancy is omitted.
- `1`: The buoyancy is included.

& diabatic

Switching the diabatic term in the pressure equation. Only in *solver*.

`diaopt` *integer* ∗ 4
Option for the diabatic term.
- `0`: The diabatic term is omitted.
- `1`: The diabatic term is included.

& ddamping

Configuration of the divergence dumping of the pressure gradient force in the momentum equations. Only in *solver*.

`divopt` *integer* ∗ 4
Option for the divergence dumping of the pressure gradient force, which is an artificial term to suppress the numerical instability.
- `0`: No divergence dumping is performed.
- `1`: Divergence dumping is performed.

`divndc` *real* ∗ 4
Non-dimensional dumping coefficient. The recommended value is 0.05.

& turbulent

The sub-grid scale turbulence process is configured in the namelist. Only in *solver*.

`tubopt` *integer* $*$ 4
Option for the sub-grid turbulence process.
- `0`: No turbulence process is performed.
- `1`: The Smagorinsky scheme (1 order closure).
- `2`: The 1.5 order closure with TKE.

`isoopt` *integer* $*$ 4
Option for directionality of the grid.
- `1`: Grid is directional in horizontal and vertical directions.
- `2`: Grid is non-directional in horizontal and vertical directions.

`prnum` *real* $*$ 4
Turbulent Prandtl number ($Pr = \nu_\tau / \nu_H$). It is used only for stability check when `tubopt = 2`.

`csnum` *real* $*$ 4
The Smagorinsky constant when `tubopt = 1`.

& cloudphy

Configuration for the cloud microphysics. Only in *solver*. **(Recompiling is necessary if changed.)**

`cphopt` *integer* $*$ 4
Option for the cloud microphysics.
- `0`: No cloud microphysics (the dry model).
- `1`: The bulk warm rain parameterization.
- `2`: The bulk cold rain parameterization.
- `3`: The bulk cold rain parameterization with solving the tendency equation of number densities of ice phase hydrometeors (ice, snow and graupel).

& outfomat

The namelists determine the format and time intervals of outputs. Only in *solver*.

`dmpfmt` *integer* $*$ 4
Option for the output format of the history and geographic files.
- `1`: ASCII text format.
- `2`: Binary data of the unformatted direct access.

`dmpcmp` *integer* $*$ 4
Option of the vertical coordinate of the history file. When `dmpcmp = 2` or 1, A small value of -1.0×10^{35} are outputted below the surface and above the top of the domain.
- `1` The vertical coordinate of output is z^* (ζ).
- `2` The output variables are interpolated at horizontal levels with an interval of `dz`.
- `3` The output variables are interporlated at horizontal levels defined by the stretching functions.

`dmpitv` *real* ∗ 4
Time interval of the history-file outputs. Unit is second. The interval is counted from the initial time even the calculation is started from a restart file. For example, if `dmpitv=300.0` seconds and restarted from 450.0 sec, the first output time will be 600.0 second from the initial time.

`resitv` *real* ∗ 4
Time interval of the restart-file outputs. Unit is second. The time interval is also counted from the initial time even if started from a restart file.

`mxnitv` *real* ∗ 4
Time interval to output maximum and minimum of prognostic variables to the standard output. Unit is second. The time interval is also counted from the initial time even if started from a restart file.

& project0

The namelists describe the coordinate system of an external data. Although they used both *gridata* and *terrain*, no dependency was present between these programs.

`mpopt0` *integer* ∗ 4
Option for the coordinate of the external data.
`1`: latitude and longitude coordinate
`2`: polar stereo graphic projection
`3`: Lambert conformal projection
`4`: Mercator projection
`5`: No map projection

`nspol0` *integer* ∗ 4
Switch to define origin of the coordinate in the northern or southern hemisphere.
`1`: the origin of coordinate in the northern hemisphere.
`-1`: the origin of coordinate in the southern hemisphere.

`tlat10,tlat20` *real* ∗ 4
The true latitudes of map projections. No deformation due to projection occurs at the latitudes. The unit is degree [°] and they are negative in the southern hemisphere. If the Lambert conformal projection is used, two true latitude (`tlat10` and `tlat20`) are necessary. If no map projection is used, they are not necessary to be specified.

`tlon0` *real* ∗ 4
The true longitude of coordinate. The unit is degree [°] and degree in the West Longitude is negative. The y-axis of the model domain corresponds to the longitude.

& gridset0

The namelists describe grid intervals and the reference points. They are used in *gridata* and *terrain* while no dependency is present between the two programs.

`xdim0,ydim0,` *integer* ∗ 4
`zdim0` Dimensions of external data in the x, y and z directions, respectively. The vertical dimension includes the surface. These namelist will be used for two-dimensional variables such as altitude. In this case, `zdim0` is not used.

`dx0,dy0` *real* ∗ 4
Grid intervals of the external data in the x and y directions, respectively. Unit is degree if `mpopt = 0` and otherwise meter. The vertical levels should be included variables of the external data.

`ulat0,ulon0` *real* ∗ 4
Longitude and latitude of a reference point of the external data. Unit is degree and they are negative in the southern hemisphere and/or the West Longitude. The point is corresponded to the real number of the external data (`riu,rju`) to determine the region in the pre-processors

`riu0,rju0` *real* ∗ 4
The real number of the reference point which corresponds to the point determined by the reference longitude and latitude (`ulat0,ulon0`).

& how2intp

The namelist determines the scheme to interpolate the external data to the model grid. It is used in *gridata* and *terrain*, while no dependency is present in these programs.

`biiopt` *integer* ∗ 4
Option for scheme of horizontal interpolation of the external data to the model grid. The vertical interpolation is always the linear interpolation.
- `0`: a linear interpolation.
- `1`: a interpolation using the square function.

& datacf3d

Format of the 3-dimensional external data (such as GPV data) an option of *gridata* are configured in the namelists.

`datype` *character* ∗ 80
Specify data type of variables in the external data. The following combinations are available.
- `tk`: temperature [K] and water vapor mixing ratio [kg kg^{-1}]
- `pk`: potential temperature [K] and water vapor mixing ratio [kg kg^{-1}]
- `tp`: temperature [K] and relative humidity [%]
- `pp`: potential temperature [K] and relative humidity [%]

`refsfc` *integer* ∗ 4
Flag to use or not the height and variables of the external data at the surface in the interpolation of the external data. If `refsfc =1`, the first level of `zdim0` is considered as the surface. This should be consistent with reading data in `rdobj.f` which must be modified by user.
- `0`: The first level is the surface and used in interpolation.
- `1`: The first level is not the surface.

9.2 Data flow in *CReSS*

9.2.1 Outline of data flow

The outline of the data flow in ***CReSS***is shown in Fig.9.1. The figure includes all types of data flows. According to the configuration, some of programs and data depicted in the figure are not necessary. Each case of configuration will be described in the following subsections.

The italic character part of file names in Fig.9.1 are summarized in the following table.

exprim	The experimental name which is defined by user. The beginning part of all input and output files is the name.
xxxxxx	Elapsed time from the initial (second). For example, if the elapsed time is 1200.0 seconds, then *xxxxxx* = `001200`.

These file names are used in the serial version of ***CReSS***. In the parallel version, the number of processing element (CPU or nodes) *yyyy* is appended to the file names as `.pe`*yyyy*.

Details of the data format will be described in Subsection 9.3.

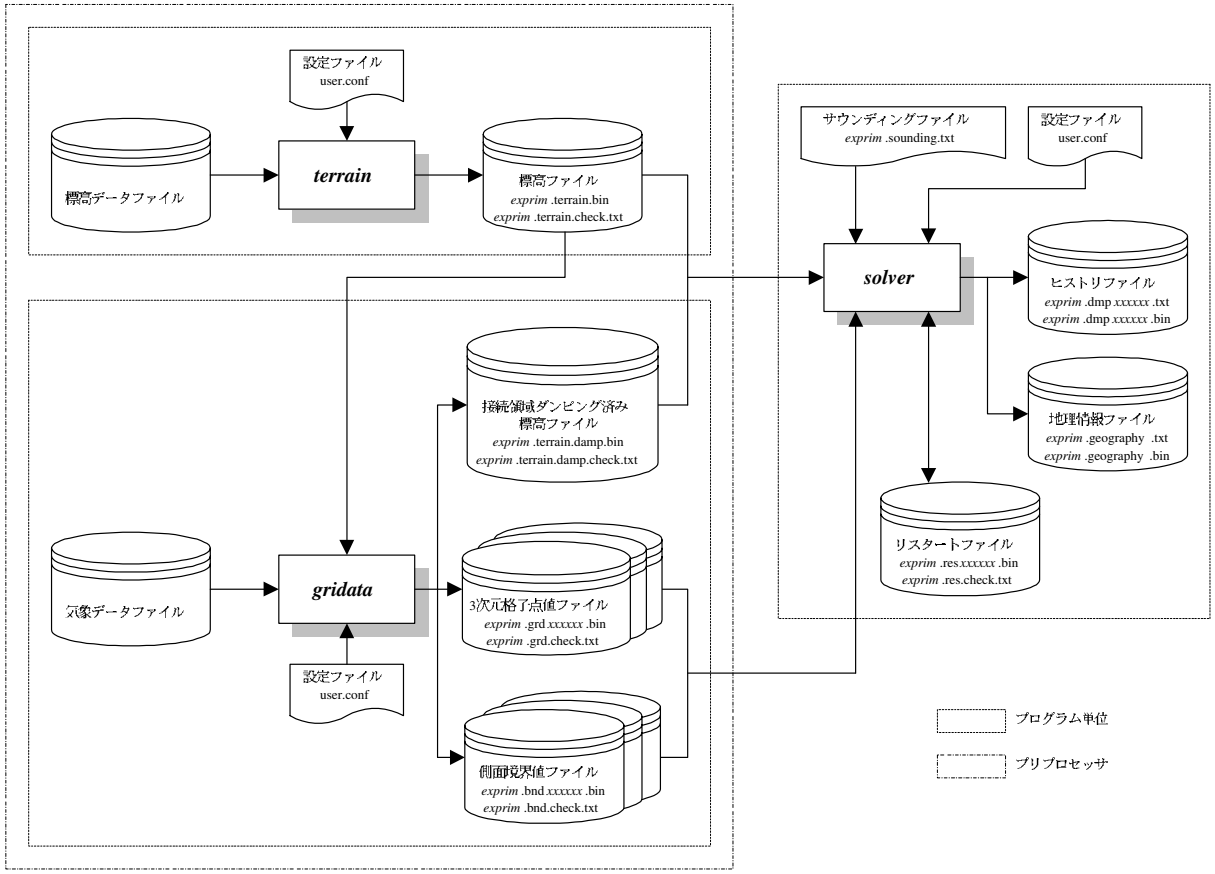


Figure 9.1. The outline of data flow in ***CReSS***.

The data files in Fig.9.1 are summarized in the following table.

file name	description
exprim.terrain.bin	Altitude of the model surface. (unformatted direct access binary file).
exprim.terrain.check.txt	Parameter file to check the configuration of the above *exprim*.terrain.bin. (formatted text file).
exprim.terrain.damp.bin	Terrain data file which lateral boundary is smoothed to connect with the coarse outer terrain when the lateral boundary data is used. (unformatted direct access binary file).
exprim.terrain.damp.check.txt	Parameter file to check the configuration of the above *exprim*.terrain.damp.bin. (formatted text file).
exprim.grd*xxxxxx*.bin	GPV data coordinated at the model grid. Time series of the file is necessary when the nudging is performed. If nudging is not performed, the data file is used at the initial time. (unformatted direct access binary file).
exprim.grd.check.txt	Parameter file to check the configuration of the above *exprim*.grd*xxxxxx*.bin (formatted text file).
exprim.bnd*xxxxxx*.bin	Boundary data coordinated at the model grid. Time series of this data is usually used. (unformatted direct access binary file).
exprim.bnd.check.txt	Parameter file to check the configuration of the above .bnd*xxxxxx*.bin. (formatted text file).
exprim.sounding.txt	One-dimensional sounding data to provide the horizontally uniform initial field. (formatted text file).
exprim.dmp*xxxxxx*.txt or .bin	History dumped file. The format is optional: formatted text file or unformatted direct access binary file.
exprim.geography.txt or .bin	Geographic information of the domain. The format is optional: formatted text file or unformatted direct access binary file.
exprim.res*xxxxxx*.bin	File to restart. (unformatted direct access binary file).
exprim.res.check.txt	Parameter file to check the configuration of the above *exprim*.res*xxxxxx*.bin. (formatted text file).

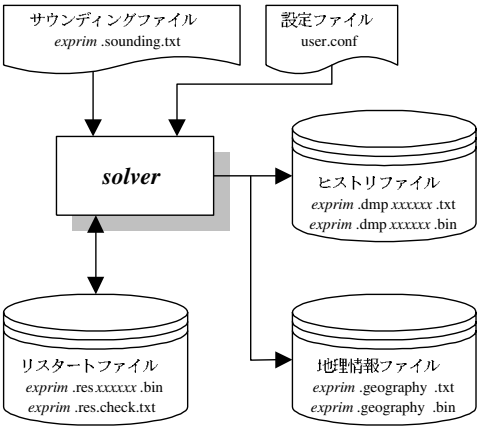
9.2.2 Data flows of each configuration

Using sounding data

(1) In the case of no terrain data

The configuration in `user.conf` is summarized in the following table and the data flow is shown in the figure.

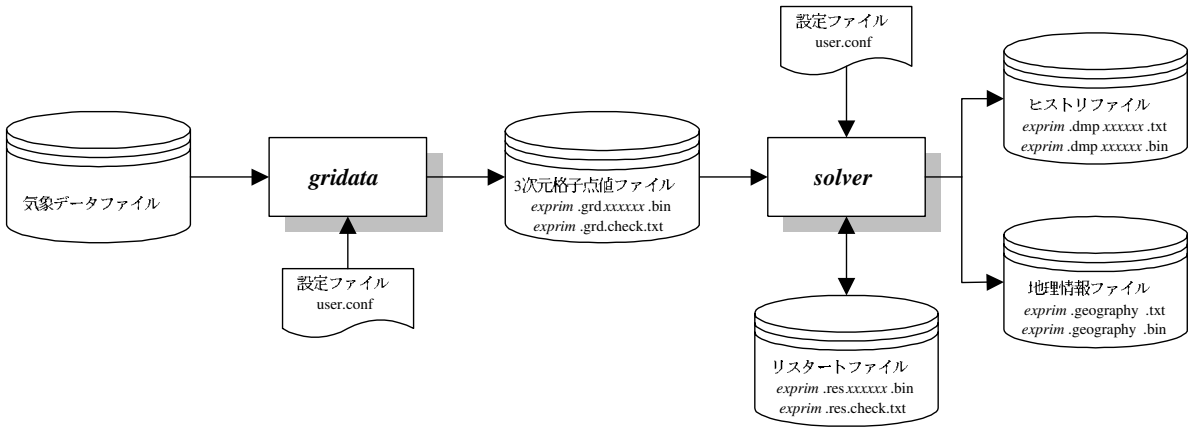
`iniopt = 1`	Initial data is provided by the sounding data.
`trnopt = 0 or 1`	No external terrain data is used.
`exbopt = 0 or 3`	No external boundary forcing is performed.



(2) Using the external terrain data

The Configuration in `user.conf` and the data flow are as follows.

`iniopt = 1`	Initial data is provided by the sounding data.
`trnopt = 0 or 1`	External terrain data is used for model topography.
`exbopt = 0 or 3`	No external boundary forcing is performed.

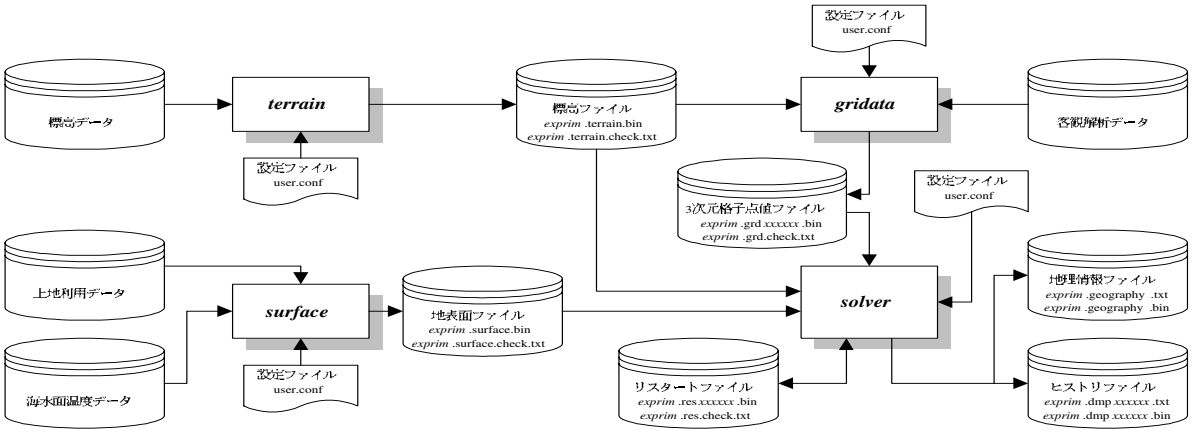


Using the 3-dimensional GPV data

(1) No external lateral boundary data and no terrain data

The Configuration in `user.conf` and the data flow are as follows. (This configuration will not generally used.)

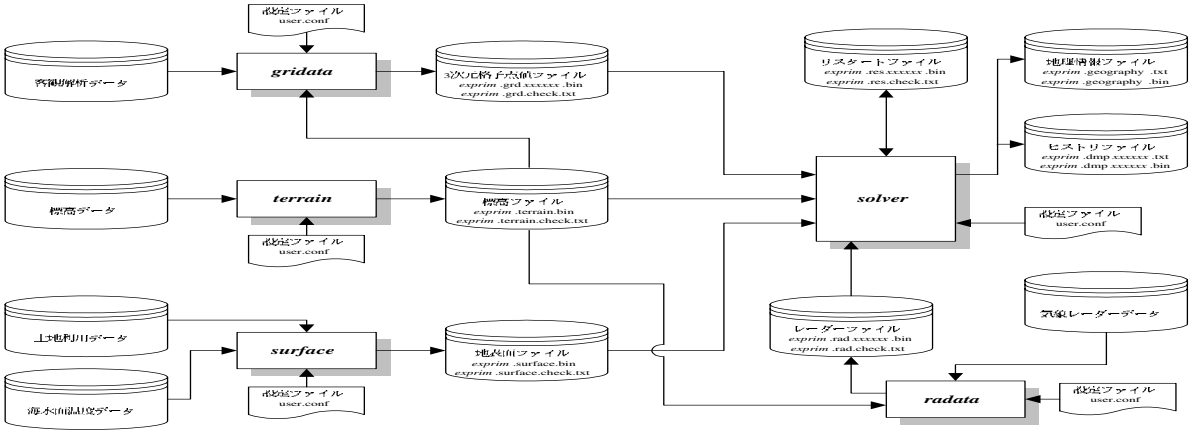
`iniopt = 3` or `4`	The initial data is a 3-dimensional GPV. Time series of the data is necessary if the nudging is performed.
`trnopt = 0` or `1`	No external terrain data is used.
`exbopt = 0`	No external boundary forcing is performed.



(2) A lateral boundary data is used while no external terrain data

The Configuration in `user.conf` and the data flow are as follows. (This configuration will not generally used.)

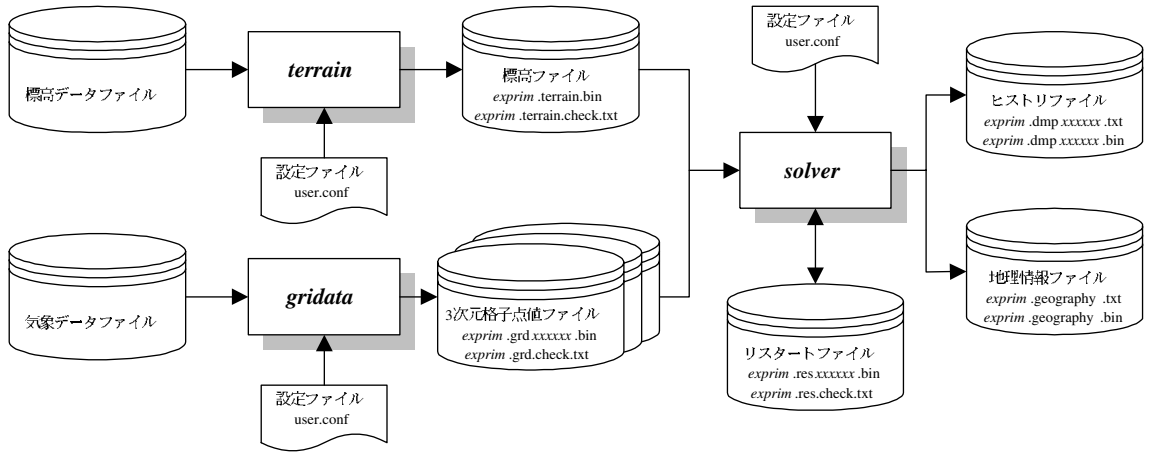
`iniopt = 3` or `4`	The initial data is a 3-dimensional GPV. Time series of the data is necessary if the nudging is performed.
`trnopt = 0` or `1`	No external terrain data is used.
`exbopt = 1` or `2`	External boundary forcing is performed.



(3) An external terrain data is used while no external lateral boundary data

The Configuration in `user.conf` and the data flow are as follows.

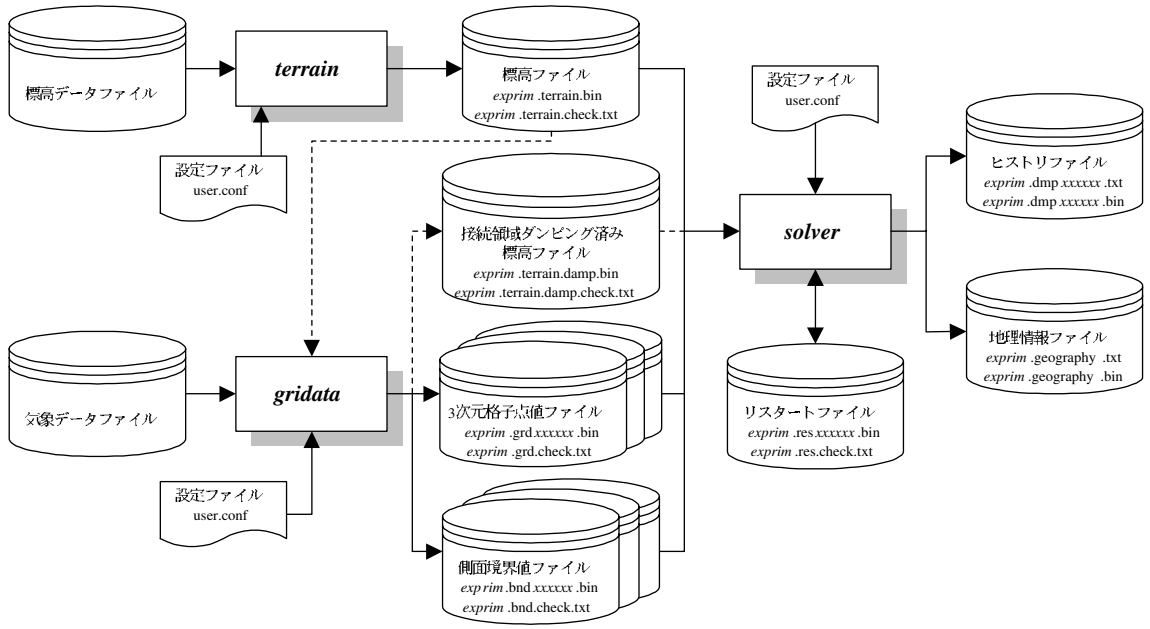
`iniopt = 3` or `4`	The initial data is a 3-dimensional GPV. Time series of the data is necessary if the nudging is performed.
`trnopt = 2`	External terrain data is used.
`exbopt = 0`	No external boundary forcing is performed.



(4) Using both external lateral boundary data and terrain data

The Configuration in `user.conf` and the data flow are as follows.

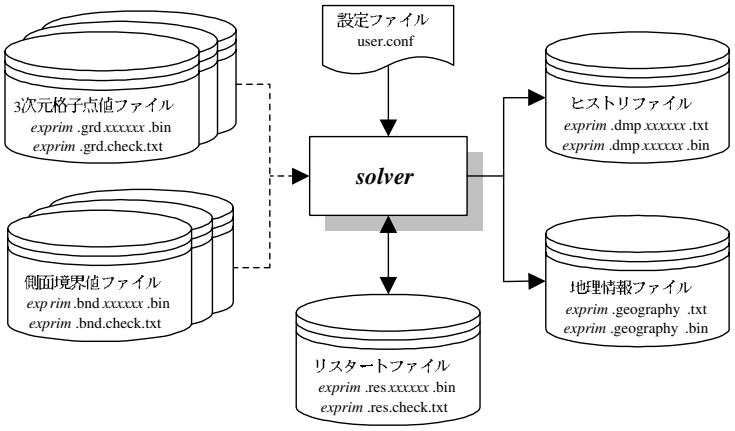
`iniopt = 3` or `4`	The initial data is a 3-dimensional GPV. Time series of the data is necessary if the nudging is performed.
`trnopt = 2`	External terrain data is used.(Boundary smoothing is performed optionally.)
`exbopt = 1` or `2`	External lateral boundary forcing is performed.



Restarting from restart file

The Configuration in `user.conf` and the data flow are as follows.

`iniopt = 2`	Begin from the restart file. Time series of GPV is necessary if nudging is performed.
`trnopt =` *anything*	Terrain condition is already determined in the previous run. (No data is necessary.)
`exbopt =` *anything*	External lateral boundary data is necessary if the external lateral forcing is performed.



9.3 Formats of I/O data

In this section, formats of I/O data which controlled by users will be described: sounding data (*exprim*`.sounding.txt`), history dumped data (*exprim*`.dmp`*xxxxxx*`.txt` or *exprim*`.dmp`*xxxxxx*`.bin`) and geography data (*exprim*`.geography.txt` or *exprim*`.geography.bin`).

9.3.1 Format of the sounding data

The sounding data is a text data composed of five columns of variables. The data types of the first, second and fifth columns are defined by users. All types of combination are allowed. Therefore, there are eight combinations of variables.

First column	height [m] or pressure [Pa]
Second column	temperature [K] or potential temperature [K]
Third column	x-component of horizontal velocity [m s^{-1}]
fourth column	y-component of horizontal velocity [m s^{-1}]
fifth column	water vapor mixing ratio [kg kg^{-1}] or relative humidity [%]

This is an example of the sounding file. It is easy to make a sounding file following the example. Some samples are found in `Form` directory of ***CReSS***.

```
#################################################################################
#                                                                               #
#     One dimensional sounding input file, sounding.txt.cats.eye.form           #
#                                                                               #
#     This is the cats eye simulation data.                                     #
#                                                                               #
#     Author       : SAKAKIBARA Atsushi                                         #
#     Date         : 1999/07/23                                                 #
#     Modification : 1999/07/28                                                 #
#                  : 1999/11/19                                                 #
#                                                                               #
#     First  column: height [m]                                                 #
#     Second column: tempereture [K]                                            #
#     Third  column: x components of velocity [m/s]                             #
#     Fourth column: y components of velocity [m/s]                             #
#     Fifth  column: water vapor relative humidity [%]                          #
#                                                                               #
#################################################################################
 780.e0 300.200e0  8.0000e0 0.e0 0.e0
 720.e0 300.200e0  7.9999e0 0.e0 0.e0
 660.e0 300.200e0  7.9993e0 0.e0 0.e0
                 :
                 :
 120.e0 299.800e0 -7.9946e0 0.e0 0.e0
  60.e0 299.800e0 -7.9993e0 0.e0 0.e0
   0.e0 299.800e0 -7.9999e0 0.e0 0.e0
```

9.3.2 Formats of the history dumped file and the geographic information file

Formatted text data or unformatted binary data is optionally chosen for the history dumped data and the geographic information data.

In the case of the formatted text data (`dmpfmt = 1`), a variable will be outputted as follows.

```
      do xxx k=2,nk-2
        write(ionum,*,err=errnum) (variable(i,j,k),i=2,nx-2,j=2,ny-2)
xxx   continue
```

If the unformatted binary data is selected (`dmpfmt = 2`), a variable will be outputted as follows.

```
      do xxx k=2,nk-2
        recnum=recnum+1
        write(ionum,rec=recnum,err=errnum)
     .       (variable(i,j,k),i=2,nx-2,j=2,ny-2)
xxx   continue
```

(refer to `outdmp3d.f` and `outdmp2d.f`). where `nx,ny,nz` are numbers of the grid points in the x, y and z directions, respectively. The number of `nz` is unity in the geographic information file.

The region of the output data is indicated by the thick lines in Fig.9.2. The data points are indicated by the crosses which is the scalar points. The vector variables defined at the closed circles are interpolated to the scalar points. The dimensions of the x, y and z directions are respectively smaller than those of the model grid by three.

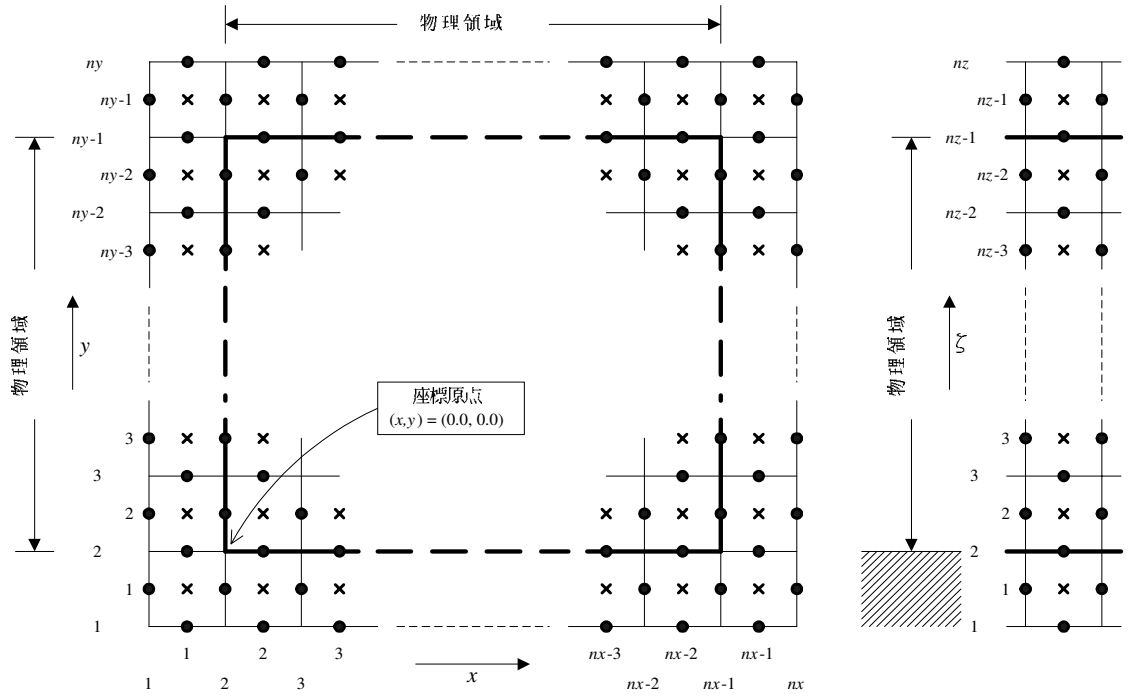


Figure 9.2. Setting of variables in the calculation domain. The closed circles are points of vector variables and the crosses are of scalar variables.

Options for vertical grid of the output data are shown in Fig.9.3.

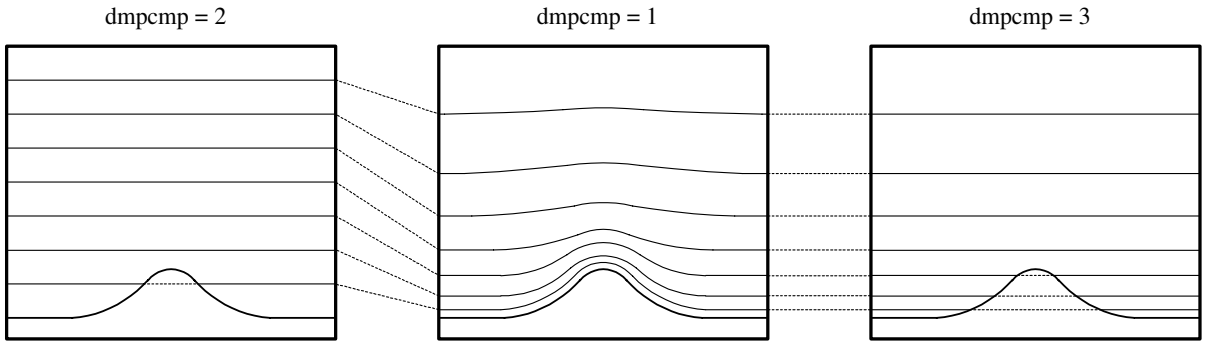


Figure 9.3. Vertical coordinates of the history dumped data.

Output variables in the geographic information data and the history dumped data are summarized in the following tables. The types of output variables of the history dumped data are depend on the configurations.

Geographic information file		
`ht`	altitude of terrain	m
`lat`	latitude	°
`lon`	longitude	°
`fs`	Coriolis parameter	
`fc`	Coriolis parameter	
`mf`	map factor	

History dumped file		
`u`	x-component of the horizontal velocity	m s^{-1}
`v`	y-component of the horizontal velocity	m s^{-1}
`w`	vertical component of velocity	m s^{-1}
`pbar`	basic state of pressure	Pa
`pp`	perturbation pressure	Pa
`ptbar`	basic state of potential temperature	K
`ptp`	perturbation potential temperature	K
`qv`	mixing ratio of water vapor	kg kg^{-1}
`qc`	mixing ratio of cloud water	kg kg^{-1}
`qr`	mixing ratio of rain water	kg kg^{-1}
`qi`	mixing ratio of cloud ice	kg kg^{-1}
`qs`	mixing ratio of snow	kg kg^{-1}
`qg`	mixing ratio of graupel	kg kg^{-1}
`nci`	number density of cloud ice	m^{-3}
`ncs`	number density of snow	m^{-3}
`ncg`	number density of graupel	m^{-3}
`zph`	height of grid points	m
`prr`	rainfall intensity,total rainfall	m s^{-1}, m
`prs`	snow intensity, total snowfall	m s^{-1}, m
`prg`	graupel intensity, total graupel fall	m s^{-1}, m

9.4 Execution of *CReSS*

9.4.1 Execution of *solver*

Using an example experiment of the Kelvin-Helmholtz waves, the procedures to execute the solver program *solver* will be explained.

First, the user uncompress and expand the compressed archive file of ***CReSS***. (This example is ***CReSS***ver.1.1 while the essence is the same in other version.)

```
% ls
cress1.1m.tar.Z
% uncompress -c cress1.1m.tar.Z | tar xvf -
  messages
     :
% ls
CReSS1.1m        cress1.1m.tar.Z
%
```

The directory structure will be as follows.

```
% cd CReSS1.1m
% ls
Doc           Src           compile.conf
Form          Tmp           compile.csh
%
```

Directory begin with a capital letter while files with a small letter. These are summarized in the following table.

`Doc`	Documents and `Readme` are included.
`Form`	Examples of configuration and the related sounding data are included.
`Src`	All source codes are archived in the directory.
`Tmp`	A temporary directory used in compilation of ***CReSS***.
`compile.conf`	Configuration file of compilation.
`compile.csh`	Shell script to control the compilation.

Some configuration files and the related sounding files are archived in the directory of `Form`. Copy the configuration file and the sounding file of the Kelvin-Helmholtz waves to the directory of `CReSS1.1m` as the following names. We will use `test` as the name of experiment. All I/O data must begin with `test.`.

```
% cp Form/user.conf.cats.eye.form.2 user.conf
% cp Form/sounding.txt.cats.eye.form test.sounding.txt
% ls
Doc               Src               compile.conf       test.sounding.txt
Form              Tmp               compile.csh        user.conf
%
```

Second, the configuration file of compilation `user.conf` will be edited if necessary. If the compiler is different from `f90` or `mpif90` or special options of compilation are used, the file should be edited. Otherwise, the file is not changed. The word length of the real number is depend on the machine. If it is not 4, the parameter `wlngth` in the user configuration file `user.conf` must be changed to the appropriate value (usually 1).

Third, compile the solver as follows. The dimension is defined in the user configuration file `user.conf` and include files are produced automatically. It is, therefore, not necessary to change the source codes of the solver.

```
% compile.csh solver user.conf
```

```
cd Src;  messages
         :
%
```

When the compilation is completed, the executable file `solver.exe` is generated. (Actually, `solver.exe` is a symbolic link to that in the directory of `Src`.)

```
% ls
Doc                Tmp                solver.exe
Form               compile.conf       test.sounding.txt
Src                compile.csh        user.conf
%
```

Forth, execute the solver with reading the user configuration file `user.conf` from the standard input and outputting `test.out` to the standard output. In order to use main frame computer, job script and NQS could be necessary. In that case, ask the system manager how to use NQS.

```
% solver.exe < user.conf > test.out &
%
```

If the calculation is terminated normally, ***CReSS***leaves the message "`This program stopped normally.`" at the end of the standard output `test.out` and the history dumped files (the names include `dmp`), the geographic information file (the names include `geography`) and restart files (the names include `res`) are produced. The standard output includes history of calculation and maximum and minimum of prognostic variables.

```
% ls
Doc                          test.dmp000120.pe0003.bin
Form                         test.dmp000160.pe0000.bin
Src                          test.dmp000160.pe0001.bin
Tmp                          test.dmp000160.pe0002.bin
compile.conf                 test.dmp000160.pe0003.bin
compile.csh                  test.dmp000200.pe0000.bin
solver.exe                   test.dmp000200.pe0001.bin
test.dmp000000.pe0000.bin    test.dmp000200.pe0002.bin
test.dmp000000.pe0001.bin    test.dmp000200.pe0003.bin
test.dmp000000.pe0002.bin    test.dmp000240.pe0000.bin
test.dmp000000.pe0003.bin    test.dmp000240.pe0001.bin
test.dmp000040.pe0000.bin    test.dmp000240.pe0002.bin
test.dmp000040.pe0001.bin    test.dmp000240.pe0003.bin
test.dmp000040.pe0002.bin    test.geography.bin
test.dmp000040.pe0003.bin    test.out
test.dmp000080.pe0000.bin    test.res000240.check.txt
test.dmp000080.pe0001.bin    test.res000240.pe0000.bin
test.dmp000080.pe0002.bin    user.res000240.pe0001.bin
test.dmp000080.pe0003.bin    test.res000240.pe0002.bin
test.dmp000120.pe0000.bin    test.res000240.pe0003.bin
test.dmp000120.pe0001.bin    test.sounding.txt
test.dmp000120.pe0002.bin    user.conf
%
```

9.4.2 Execution of *gather*

In parallel calculation, the history dumped files are outputted separately by each precessing element. To combine the files, the post-processor *gather* is used.

The compilation of *gather* is as follows.

```
% compile.csh gather user.conf
cd Src;  messages
         :
%
```

Then the post-processor `gather.exe` is generated. `gather.exe` is an interactive executable. The following example combines the history dumped file at 240 seconds. The name of experiment is `test`.

```
% gather.exe
Now the program, gather start.

  Input the history file name of PE #0000. readfl(character) = ?
    >> test.dmp000240.pe0000.bin
  Input the word length for your machine. wlngth(integer) = 4

  Input the selected cloud micro physics option. cphopt(integer) = 0

  Input the selected history file z coordinates option. dmpcmp(integer) = 1

  Can I delete history files after processing? "yes" or "no", fdelet = no

  Your setting are:
    readfl = "test.dmp000240.pe0000.bin"
    wlngth = 4,  cphopt = 0,  dmpcmp = 1,  fdelet = "no"

  Are these setting correct? "yes" or "no" >> yes
    messages
       :
This program stopped normally.
%
```

The combined file is `test.dmp000240.gather.bin`. The format of the file is described in Subsection 9.3.2. Types and order of output are also found in the standard output `test.out`.

The file is used in a graphic application such as Grads, then the result will be displayed as in Fig.9.4 and Fig.9.5.

9.4.3 Execution of *terrain*

The pre-processor *terrain* is a program to interpolate a terrain data into the model grids. This is necessary in a prediction experiment as well as a numerical experiment using a non-analytic topography.

While an external terrain data is necessary to produce the model terrain using *terrain*, the format of the external terrain data is not specified in ***CReSS***. In order to read the data, it is necessary to modify the subroutine `rdtrn.f` which will be found in the directory `Src`. User will modify the parts of the `open` statement and `read` statement in `rdtrn.f` which is indicated by `#` to read the external terrain data.

```
* ##### You need to modify the following lines. #####

      siz=nid*njd*wlngth

      write(trnfl(1:16),'(a16)') 'data.terrain.bin'
```

```
      open(iotrn,iostat=stat,file=trnfl(1:16),status='old',
     .     access='direct',form='unformatted',recl=siz)

* ####################

* ##### You need to modify the following lines. #####

      read(iotrn,rec=1,err=100) ((htdat(id,jd),id=1,nid),jd=1,njd)

* ####################
```

The unit of the terrain data is meter. If the unit is not meter, it must be corrected here.

After the namelists of the user configuration file are specified (refer to Section 9.1), compile as follows.

```
% compile.csh terrain user.conf
cd Src; messages
        :
%
```

When the compilation is completed, the executable program `terrain.exe` is generated. Actually, `terrain.exe` is a symbolic link to that in the directory of `Src`.

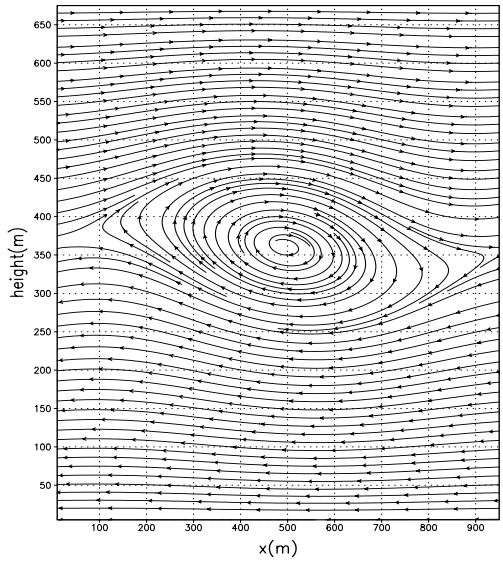


Figure 9.4. Stream function of $(u - w)$ at 240 seconds from the initial.

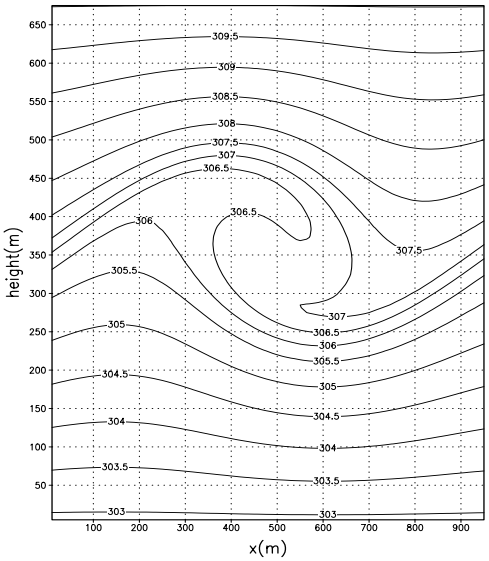


Figure 9.5. Potential temperature [K] at 240 seconds from the initial.

The execution of `terrain.exe` is as follows. The input data file must be in the current directory.

```
% terrain.exe < user.conf > test.out.terrain &
%
```

If the execution succeeded, the terrain data file with the experimental name at the beginning (in the case, it is `test`) will be generated. (For example`test.terrain.bin`). The program `terrain.exe` of the serial and parallel versions is performed in a single processing element. A single file will be produced by the serial version while multiple files by the parallel version with an extention of `.pe`*yyyy*. Where *yyyy* is the unit number of the processing element. This is necessary to execute the parallel version of `solver.exe`.

9.4.4 Execution of *gridata*

The pre-processor *gridata* produces 3-dimensional grid point data files and the lateral boundary data files.

The format of the external GPV data such as an objective analysis data or a coarse model output is not specified in ***CReSS***. In order to read the data, it is necessary to modify the subroutine `rdobj.f` which will be found in `Src`. User will change the `open` statement and `read` statement of the subroutine `rdobj.f`, which are indicated by `#`. The file names of the input data must include date and time of the Gregorian calendar *yyyymmddhhmm* (UTC).

```
* ##### You need to modify the following lines. #####

      write(objfl(1:24),'(a8,a12,a4)') 'data.obj',cdate(1:12),'.bin'

      open(iobj,iostat=stat,file=objfl(1:24),status='old',
     .     access='sequential',form='unformatted')

* ####################

* ##### You need to modify the following lines. #####

      read(iobj,end=100,err=100) zdat

      read(iobj,end=100,err=100) udat
      read(iobj,end=100,err=100) vdat
```

`qrdat`	mixing ratio of rain water	kg kg^{-1}
`qidat`	mixing ratio of cloud ice	kg kg^{-1}
`qsdat`	mixing ratio of snow	kg kg^{-1}
`qgdat`	mixing ratio of graupel	kg kg^{-1}

If the units of the variable are different from the above, correction is necessary in the subroutine.

If data at the surface is present (`refsfc = 1`), the data is allocated at the first level (`kd = 1`). If the altitude of the surface is allocated at the first level of `zdat` and the corresponded data is allocated at the first level of each data, the level which is lower than the altitude will be ignored in the vertical interpolation when `refsfc = 1`. If `refsfc = 0`, the interpolation or extrapolation will be performed in the model grid even if the data is below the ground.

Compilation and execution of `gridata.exe` are essentially the same as `terrain.exe`. After the namelists are configured (refer to Section 9.1), compile the program as follows.

```
% compile.csh gridata user.conf
cd Src;  messages
         :
%
```

If the compilation is succeeded, the executable file `gridata.exe` is generated. Actually `gridata.exe` is also a symbolic link to that in the directory `Src`.

The execution of `terrain.exe` is as follows. The input data file must be in the current directory.

```
% gridata.exe < user.conf > test.out.gridata &
%
```

If the execution succeeded, 3-dimensional grid data files (in this case `test.grd`*xxxxxx*`.bin`), the lateral boundary data files (`test.bnd`*xxxxxx*`.bin`), and optionally the model terrain data which is connected with the external terrain smoothly (`test.terrain.damp.bin`) are produced.

The program `gridata.exe` of the serial and parallel versions is performed in a single processing element. A single file will be produced by the serial version while multiple files by the parallel version with an extension of `.pe`*yyyy*. Where *yyyy* is the unit number of the processing element. This is necessary to execute the parallel version of `solver.exe`.