System Design of LSV

Ver. 0.5.0

ISLiM, RIKEN

http://www.csrp.riken.jp/index_e.html

October 2011

1st Edition	Mar.	2011
2nd Edition	Jun.	2011
3rd Edition	Aug.	2011

2-1, Hirosawa, Wako, 351-0198, Japan

Table of Contents

0. Introduction

The LSV is a parallel visualization program built on client-server architecture and capable of handling a large amount of data.

This document describes a detailed system specification that enables parallel streamline visualization and unstructured gird data handling in the LSV. This document also describes the capabilities implemented in the LSV for achieving the streamline visualization of time-series structural-grid data, as well as for loading and visualizing unstructured-grid data.

1. Parallel Streamline Visualization

1.1. Outline

This chapter describes the implementation of functionality to visualize time-series structural-grid data as streamlines in the LSV system. This functionality consists of the following two capabilities:

(1) Streamline visualization map
The maps are implemented for application to SPH or SPX data.
The start point data specified by the client machine is stored in an external file so that the data are interfaced with external programs.
(2) Program for calculating pathline, streakline, and timeline
The LSV implements the external programs for calculating pathline, streakline, and timeline according to the input of time-series data files in SPH or SPX format (Index file) and the start point data files.
The calculated results are output as external discrete files, to which the LSV provides an interface.

To display the output results of (2), the LSV system implements the function for loading discrete data files and the maps for displaying streamline data.

1.2. **Streamline visualization map** (streamlines)

1.2.1. Capabilities

A visualized streamline is drawn from a start point on a specified line segment.

The start point is defined by three parameters: “translate,” “scale,” and “rotate.” The start point in the initial state is a line segment drawn parallel to the X-axis, starting from the coordinate origin of the data bounding box and having length same as that of its side along the X-axis. The final start point is obtained by applying the following geometric transformation to the line:

[M] = [T][Ry][Rx][Rz][S],
where
T: Translation of axis specified by ‘translate’
Ry: Rotation around Y-axis specified by ‘rotate’ parameter
Rx: Rotation around X-axis specified by ‘rotate’ parameter
Rz: Rotation around Z-axis specified by ‘rotate’ parameter
S: Magnification or reduction specified by ‘scale’ parameter

The client program has a function to output the specified geometric transformation parameters and the final start point coordinates to a specified file.

The data to be registered must be the flow velocity vector data.

1.2.2. Build parameter XML format

Writing format

```
<GraphicExtendedBuild>
  <Entity>
    <param name="num_points" value="10" />
    <param name="show_sampler" value="no|yes" />
    <param name="sampler_line_width" value="1.0" />
```

```
    <param name="sampler_point_size" value="2.0" />
    <translate x="0.0" y="0.0" z="0.0" />
    <rotate x="0.0" y="0.0" z="0.0" />
    <scale x="1.0" y="1.0" z="1.0" />
    <param name="num_divs" value="1" />
    <param name="max_pts" value="1000" />
    <param name="colored" value="no|yes" />
    <param name="line_width" value="1.0" />
    <param name="reverse" value="no|yes" />
  </Entity>
</GraphicExtendedBuild>
```

Description of parameters

• num_points (integer: text box)

Specify the number of start points. The value must be greater than 0. The default value when omitted is 10.

• show_sampler (“no” or “yes”: checkbox)

Specify whether or not to display the line and point indicating the start point. If “no” is set, the line and point are not displayed.
The default value when omitted is “yes.”

• sampler_line_width (real number: text box)

Specify the width of the line indicating the start point in pixels. The value must be greater than 0.0.
The default value when omitted is 1.0.

• sampler_point_size (real number: text box)

Specify the size of the point indicating the start point in pixels. The value must be greater than 0.0.
The default value when omitted is 2.0.

• translate

Specify the translation amount of the start point in relation to the initial state as x, y, and z attributes.
The default value when omitted is (0.0, 0.0, 0.0). In the default setting, the start point equals the coordinate origin of data bounding box.

• scale

Specify the zoom ratio of the start point in relation to the initial state as x, y, and z attributes. The default value when omitted is (1.0, 1.0, 1.0). In the default setting, the line length of the start point is equal to the length of data bounding box in x-axis direction. The y and z components are disregarded.

• rotate

Specify in degrees the rotation amount of the start point with respect to the initial state as x, y, z attributes.
Each component represents the rotation amount of each X-, Y-, Z-axis. The default value when omitted is (0.0, 0.0, 0.0). In the default setting, the line at the start point equals the line along X-axis direction.

• num_divs (integer: text box)

Specify the number of time step divisions (i.e. the number of integral steps in relation to virtual time step). The value must be greater than 0.
The time necessary for the maximum velocity to move across the width of one grid is assumed to be a virtual time step. The virtual time step is then divided by the “num divs” value, and the velocity field is integrated assuming the obtained value as integral time.

The default value when omitted is 1.

• max_pts (integer: text box)
Specify the maximum number of points of a single streamline. The value must be greater than 0.
The default value when omitted is 1000.

• colored (“no” or “yes”: checkbox)
Specify whether or not to use separate colors for streamlines.
If “yes” is set, each streamline is displayed in a separate color by referring to the color map. If “no” is set, all streamlines are displayed in base color.
The default value when omitted is “no.”

• line_width (real number: text box)
Specify the width of streamlines in pixels. The value must be greater than 0.0.
The default value when omitted is 1.0.

• reverse (“no” or “yes”: checkbox)
Specify whether or not to perform the reverse calculation of streamlines.
The default value when omitted is “no.”

1.2.3. Response XML format

The <Entity> tag is not used.

1.2.4. Builder function

Prototype

```
LSV_STATUS streamLines (char* pBuildParamXml, char** ppReturnValueXml,
                        LSVSG::Scene* pSceneData)
```

Argument:

pBuildParamXml	Storage area allocated for XML format build parameters
ppReturnValueXml	Storage area allocated for XLM format response
pSceneData	Pointer to scene graph route node

Return value:

LS_TRUE(**success**), LSV_FALSE(**failure**)

Description:

This function builds a map that performs the parallel streamline visualization on the basis of the specified build parameter.

Each RANK performs the integral (in virtual time steps) of particles in its assigned area, and records the trajectories of the particles moving within its own RANK, and adds them to the scene graph.

If the coordinates of a particle are moved to another RANK’s area, the function notifies the coordinates, and repeats the integral until the value reaches a specified maximum number of points (or domain boundary).

The integral of the velocity field is performed using the fourth-order Runge-Kutta Gill technique.

1.2.5. Utility functions

GetIntegrand

Prototype:

```
double GetIntegrand(LSVg gridInterpolate<T>& gus, const int divTime,
                    const LSVG::DVec4 x i, LSVG::DVec3 func,const bool reverse)
```

Argument:

gus	Coordinate data structure for interpolation
divTime	Number of divisions at maximum virtual time step
x i	Coordinate value in grid coordinate space
func	Coefficient for interpolation used in Runge–Kutta Gill method
reverse	Flag indicating opposite direction

Return value:

Virtual time step

Description:

This function calculates the virtual time step used for integration of the velocity field.
It performs calculation assuming the time taken for a virtual particle to move across the width of a grid as the maximum virtual time step, and returns the value divided by “divTime” as a virtual time step.

RKG

Prototype:

```
int RKG(LSVg_gridInterpolate<T>& gus, const int divTime,
        const double t_step, const LSVG::DVec4 x_i, LSVG::DVec3 func,
        const bool reverse)
```

Argument:

gus	Coordinate data structure for interpolation
divTime	Number of divisions at maximum virtual time step
t_step	Virtual time step
x_i	Coordinate value in grid coordinate space
func	Runge–Kutta coefficient for interpolation used in Runge–Kutta Gill method
reverse	Flag indicating opposite direction

Return value:

0(**success**), -1(**out of bounds**), -2(**stagnation point**), -3(**failure**)

Description:

The integral of the velocity field is performed using the fourth-order Runge–Kutta Gill method.
If the return value is negative, the function terminates the calculation.

1.3. Calculation program for pathline, streakline, and timeline (lsvPT)

This program executes MPI parallel calculation of pathlines, streaklines, and timelines according to the input of time-series data files in SPH or SPX format (Index file) and the start point coordinate data file.

The parallel execution is performed based on particles. The fourth-order Runge–Kutta Gill method is employed to integrate the velocity field.

1.3.1. Process outline

1: Reads the index file of the data and the start point coordinate data file.
2: Distributes the particle start point coordinates to each RANK process.
3: Each RANK process loads data of the area including the distributed particles.
The loaded data is cached using the LRU algorithm.
4: Each RANK process stores the trajectories of moved particles by integrating the velocity field.
5: Repeats steps 3 and 4 until the final time step.
6: Outputs the stored particle trajectories as a discrete data file.

1.3.2. Command specification

```
lsvPT -type {streak|ppath|tline|stream} [-rel_skip skip_step] [-rel_max release_max] ＼
    [-dt deltaT] [-reverse] -start start_path -div nx ny nz -data Index_path ＼
    -out out_base [-out_skip out_skip] [-out_mode {master|local}] ＼
    [-cache cache_size] [-cache_hist] [-quiet]
```

-type {streak | ppath | tline | stream}
Specifies the type of calculation (streakline/pathline/timeline/streamline).
-reverse
If specified, traces particles in opposite direction by reversing the velocity components.
-rel_skip skip_steps
Specifies the number of steps to skip for each particle release (integer: If omitted, 1 is assumed).
-rel_max release_max
Specifies the maximum number of the particle release executions (integer: If omitted, unlimited number of times is assumed).
-dt deltaT
Specifies the time interval of a single time step (If omitted, the time value in Index file is referred.)
-start start_path
Specifies the path to the data file of start point coordinates.
-div nx ny nz
Specifies the number of divisions on the domain in input data (in each axis direction).
-data Index_path
Specifies the path to the Index file as input data.
-out out_base
Specifies the base name of output data files.
If '/aaa/bbb/ccc' is specified, '/aaa/bbb/ccc.idx' is created as an Index file, and '/aaa/bbb/ccc_RANK number_step number.par' is created as a particle trajectory file. (If the file output mode is ‘master,’ the RANK number is omitted.)
-out_skip out_skip
Specifies the number of steps to skip for each file output (integer: If omitted, 1 is assumed).
-out_mode {master | localg**}**
Specifies file output mode (output by collecting data to **Rank0** | output per **Rank**)
-cache cache_size

Specifies the maximum cache size of loaded data (bytes: integer: if omitted, 1024^3 = 1GB).
The letters k, m, g, and t attached to the end of the value indicate KB, MB, GB, and TB, respectively.

-cache hist

Outputs cache history logs to files.

-quiet

Restrains output messages during execution.

1.3.3. Program structure

Fig. 1: Package configuration

• **App** (Main application module)

A package including the main class that serves as the foundation of an application.

The App consists of the AppBase, which implements the common functionalities shared by each class, and the functional classes that perform pathline, streakline, and timeline calculations.

• **DSlib** (Data supply library)

This package manages the SPH or SPX time-series data files written in the Index file as a set of divided domain blocks, and it implements the capability to load blocks of specified time step and domain.

The package consists of the IndexFile class that parses XML data in the Index file and the Loader class that loads specified blocks of divided domain. The cache feature of the loaded divided domain block is implemented on the Loader class. This data-loading capability is achieved by using the SPHERE SPL Library.

• **PPlib** (Particle trajectory calculation library)

This package implements the capability to calculate particle trajectories by integrating particles present in a specified position in the given velocity field data provided by the DSlib.

The package consists of the 'Interpolator', which interpolates the flow velocity of a Cartesian grid; the 'Integrator' class, which performs the integral of the fourth-order Runge–Kutta method; and the 'Transport' module, which calculates the trajectories of multiple particles and stores the results.

Configuration of DSlib

• DS_Index class

This class is a structure that parses SPH or SPX data written in the Index file and maintains the data.

Member variables

Name	Type	Description
num_step	size t	Number of time steps
time_list	vector < double >	List of recorded time [num steps]
file_num	size t	Number of divided files
file_formation	V3 < int >	Number of divided files (in each direction)
file_list	vector < vector < string >>	File list [num steps][file num]
file_type	int	File format (SPH=0 | SPX=1)
input_obj	splSpx[ ]	List of file object [file num]
vlen	int	Vector length
rlen	int	Real number data length
dtype	int	Data type
origin	V3 < double >	Global coordinate origin
pitch	V3 < double >	Pitch

whole_size	V3 < size_t >	Total number of grids
divs	V3 < size_t >	Number of domain divisions (in each direction)
divSize	size t	Number of domain divisions
start_list	vector < size_t > [3]	List of start point indices for domain divisions (in each direction)[divs[i]][3]
size_list	vector < size_t > [3]	List of size of divided domain (in each direction)[divs[i]][3]

Public methods

Reset			
Argument	None		
Return value	None		
Description	Initializes all members.		

GetInfo			
Argument	filena	const char*	Path to Index file
Return value	bool	true: success, false: failure	
Description	Reads specified Index file and sets its parameters to member variables		

SetDivs			
Argument	divs	const V3 < size_t >&	Number of domain divisions in each direction
Return value	bool	true: success, false: failure	
Description	Specifies the number of domain divisions		

FindFileIdxByIJK			
Argument	I	const double	Index number in X-axis direction (real number)
	J	const double	Index number in Y-axis direction (real number)
	K	const double	Index number in Z-axis direction (real number)
	idx	V3 < size_t >&	Area allocated to store the index number of divided domain
Return value	bool	true: success, false: failure	
Description	Obtains the index number of divided domain by specifying grid index. Specifies to 'idx' the index number of the divided domain block including the grid specified by (I, J, K).		

FindFileIdxByCRD			
Argument	X	const double	X coordinate
	Y	const double	Y coordinate
	Z	const double	Z coordinate
	idx	V3 < size_t >&	Area allocated to store index number of divided domain
Return value	bool	true: success, false: failure	
Description	Obtains the index number of divided domain by specifying coordinates Specifies to 'idx' the index number of the divided domain block including the grid specified by (X, Y, Z).		

Load			
Argument	idx	const V3 ＜ size t ＞	Index number of divided domain to be loaded (in each direction)
	step	const size_t	Time step number to be loaded
	objLoad	splSpx*	SPL object for storing loaded data
Return value	bool	true: success, false: failure	
Description	Loads divided domain data. Loads to 'splObj' the block of divided domain specified by 'idx' at the time step specified by 'step.'		

• DS_Loader class

This class loads data of each block of a divided domain to SPH or SPX data.

The loaded data are held by the array of the CacheUnit structure (a map using the block index number as key), and are cached by the following LRU algorithms of the 'Load()' method:

1. Search for required block on cache using the block's index number as a key.
2. If a block is found on cache:
   – Set counter value 'n' of the found block to '0'.
   – Out of multiple blocks found on cache, set +1 to those blocks whose counter values are n or smaller.
   – Leave them unchanged if their counter value is greater than n.
   – Return the data of the found block.
3. In case blocks do not exist on cache
   - If the number of cached data reaches the maximum number, delete the block having the largest counter value.
   –Load required block.
   –Set +1 to the counter values of all blocks except new blocks.
   –Return the data of loaded blocks.

Member variables

Name	Type	Description
m_ready	bool	Flag indicating initialized state
m_indexLoader	DS_Index	Information about Index files
m_curStep	size_t	Time step number of loading object
m_memSize	size_t	Maximum cache memory size
m_cacheNum	size_t	Maximum number of cached items
m_cache	map ＜ SV3, CacheUnit	Cache
m_fpCH	FILE*	File identifier for cache history
m_idxSzCH	SV3	Number of digits used for index of cache history
m_rewIdxCH	SV3	Index number of previously loaded block recorded in cache history

Public methods

Init			
Argument	filename	const char*	Index file name
	divs	const SV3 &	Number of domain divisions (in each direction)
	memSize	const size_t	Cache capacity (bytes)
Return value	bool	true: success, false: failure	
Description	Sets Index file specified by 'filename' to 'm_indexLoader,' and allocates the cache pool of 'memSize' for the blocks of divided domain based on 'divs.'		

SetStep			
Argument	step	const size_t	Time step number of object to load
Return value	bool	true: success, false: failure	
Description	Specifies the time step number of object to load. The cache content is cleared.		

Load			
Argument	fileIdx	const SV3 &	Index of divided domain to load (in each axis-direction)
Return value	splSpx*	Pointer to the loaded block data of divided domain	
Description	Loads data of divided domain. If specified divided domain data exists on cache, returns that data. If specified divided domain data does not exist, loads the data, registers to cache, and returns it. If the number of cached items reaches the upper limit, determines the divided domain data to delete from cache according to the LRU algorithm.		

LoadByIJK			
Argument	IJK	const DV3 &	Grid index (in each axis-direction, real number)
Return value	splSpx*	Pointer to the loaded block data of divided domain	
Description	Loads the data of divided domain specified by grid indices.		

LoadByCRD			
Argument	position	const DV3 &	Coordinate value (in each axis-direction)
Return value	splSpx*	Pointer to loaded block data of divided domain	
Description	Loads the data of divided domain specified by coordinates.		

isReady			
Argument	None		
Return value	bool	Value of initialized state	
Description	Returns initialized state.		

GetIndexInfo			
Argument	None		
Return value	const DS_Index*	Maximum number of cached items	
Description	Returns the maximum number of cached items		

GetCurStep			
Argument	None		
Return value	size_t	Current time step number	
Description	Returns the time step number of object to load.		

GetMaxCacheSize			
Argument	None		

Return value	size_t	Maximum cache memory capacity (bytes)
Description	Returns the maximum cache memory capacity.	

GetMaxCacheNum			
Argument	None		
Return value	size_t	Maximum number of cached items.	
Description	Returns the maximum number of cached items.		

CachePurge			
Argument	None		
Return value	None		
Description	Clears cache contents.		

SetCacheHistoryLog			
Argument	filename	const char*	Name of cache history file
Return value	bool	True: normal, False: failure	
Description	Sets up cache history files.		

CacheHistoryLog			
Argument	reqIdx	const SV3 &	Block index number requested to load
	purgeIdx	const SV3*	Pointer to the storage space of block index number deleted from cache
Return value	bool	True: normal, False: failure	
Description	Outputs cache history. Fails if no cache history file is set up by 'SetCacheHistoryLog.' Does not output if 'reqIdx' is identical to the previously loaded block index.		

DS Loader::CacheUnit structure

This is an internal structure of the DS Loader class, and it also serves as the internal data structure to cache the data of each block of a divided domain loaded by the DS Loader.

Member variables

Name	Type	Description
m_hitCount	size_t	Cache hit count
m_step	size_t	Time step number
p_data	splSpx*	Pointer to the loaded block data of divided domain

Public methods

purge			
Argument	None		
Return value	None		
Description	Deletes the cached block data of divided domain. Deletes the divided domain block data pointed by 'p_data' of 'psplFinalizeSpx(),' and initializes each member variable.		

Configuration of PPlib

• PP_PartSet class

This class stores multiple particle trajectories, and implements I/O facility for discrete data files.

Member variables

Name	Type	Description
m_numLines	size_t	Number of particle trajectories (lines)
m_numDatas	size_t	Number of data items held by each particle
m_coordSet	vector < vector < DV3 > >	Array to store particle coordinates [m_numLines][ ]
m_dataSet	vector < vector < double > >	Array to store particle data [m_numLines × m_numDatas][ ]

Public methods

Setup			
Argument	numLines	const size_t	Number of particle trajectories
	numDatas	const size_t	Number of particle data
Return value	bool	true: success, false: failure	
Description	Initializes the store array according to the number of particle trajectories (lines) and number of particle data.		

ExpandNumLines			
Argument	numLines	const size_t	Number of particle trajectories
Return value	bool	true: success, false: failure	
Description	Changes the number of particle trajectories (lines) to 'numLines.'		

SaveParFile			
Argument	path	const char*	Output path to discrete file
	step	const int	Time step number written in discrete file
	time	const float	Time written in discrete file
Return value	bool	true: success, false: failure	
Description	Outputs data to discrete data files.		

LoadParFile			
Argument	path	const char*	Input path to discrete file
	step	int &	Storage space of time step number
	time	float &	Storage space of recorded time
Return value	bool	true: success, false: failure	
Description	Loads discrete data files.		

• PP_Interplolator class

This class performs trilinear interpolation on a three-dimensional grid space.

Member variables

Name	Type	Description
m_dims	size_t[3]	Grid size
p_vecd	T* (T = type of template)	Pointer to vector data array
m_vecLen	size_t	Vector length
m_orig	DV3	Grid origin coordinate
m_pitch	DV3	Pitch

Public methods

reset			
Argument	None		
Return value	None		
Description	Initializes member variables.		

setup			
Argument	dims	const size_t[3]	Grid size
	vlen	const size_t	Vector length
	org	const DV3 &	Grid origin coordinate
	pitch	const DV3 &	Pitch
	pdv	T* (T=type of template)	Pointer to vector data array
Return value	bool	true: success, false: failure	
Description	Specifies each parameter.		

InterpolateCoord			
Argument	x I	const LSVG::DVec3	Grid coordinate
	x	DV3 &	Storage space of coordinates
Return value	bool	true: success, false: failure	
Description	Converts grid coordinates to coordinate values.		

InterpolateData			
Argument	x I	const LSVG::DVec3	Grid coordinate
	didx	const int[3]	Index number of data requires interpolation
	dval	DV3 &	Storage space of interpolated vector data
Return value	bool	true: success, false: failure	
Description	Interpolates vector data on specified grid coordinates.		

InterpolateData			
Argument	x I	const LSVG::DVec3	Grid coordinate
	dkind	const size_t	Index number of data requires interpolation
	dval	double &	Storage space of interpolated scalar data
Return value	bool	true: success, false: failure	
Description	Interpolates scalar data on specified grid coordinates.		

• PP_Integrator class

This class performs the integral of the velocity field by the fourth-order Runge–Kutta method. It does not have a local member variable and it implements static methods only.

Public Methods

GetIntegrand			
Argument	gus	PP Interpolator < T >&	Object that performs interpolation
	divTime	const int	Number of divisions redone by Δ t
	x i	const LSVG::DVec4	Grid coordinate
	func	LSVG::DVec3	Coefficient for Runge–Kutta method

	reverse	const bool	Flag for reversed velocity component
Return value	double	Maximum time step interval	
Description	Calculates coefficients for Runge–Kutta method and maximum time step interval.		

RKG			
Argument	gus	PP Interpolator ＜ T ＞&	Object that performs interpolation
	divTime	const int	Number of divisions redone by Δ t
	t step	const double	Time step interval
	x i	const LSVG::DVec4	Grid coordinate
	func	LSVG::DVec3	Coefficient for Runge–Kutta method
	reverse	const bool	Flag for reversed velocity component
	bound	const int	Flag for domain boundary
Return value	int	0: normal, -1: out of bounds, -2: residence point, -3: beyond calculation	
Description	Performs integral of velocity field by fourth-order Runge–Kutta method.		

• PP_Transport module

This functional module calculates the particle movement along the flow-velocity field.

Available subroutine

PP Transport			
Argument	loader	DS Loader &	Data loader
	deltaT	const double	Integral time
	divT	const size_t	Number of divisions redone by integral time
	curPts	const PTarray ＜ DV3 ＞	Position of particle before moved
	newPts	PTarray ＜ DV3 ＞&	Storage area for position of moved particle
	validPts	PTarray ＜ bool ＞&	Storage area for flag indicating the validity of moved particles
	reverse	const bool	Flag indicating reversed velocity
Return value	bool	true: success, false: failure	
Description	A set of functions that calculates particle movement along with velocity field Performs integration over all particles included in 'curPts' with respect to 'deltaT' time, and stores the coordinates of moved particles in 'newPts.' For particles moved out of calculation domain, sets 'validPts = false.' Once this subroutine is executed, the sizes of 'newPts' and 'validPts' are changed to the size of 'curPts.'		

Configuration of App

• lsvPT_AppBase class

This class provides the common functionalities used for pathline, streakline, and timeline calculations.

Member variables

Name	Type	Description
m status	StatusType	Initialized status
m appType	PT_Type	Type of calculation

m particles	PP_PartSet	Storage area of particle trajectory Number of particle trajectories: Pathline: (number of integral start points) Streakline: (number of integral start points) Timeline: (number of particle release processes)
m startPts	PTarray < DV3	Array of managed integral start point
m startIdx	size_t	Index number of managed integral start point out of entire integral start point
m curPts	PTarray < DV3 >	Current array of integral point Number of integral points: Pathline: (number of integral start points) Streakline: (number of integral start points x number of particle release processes) Timeline: (number of integral start points x number of particle release processes)
m validPts	PTarray < bool >	Array of flag indicating validity of integral point
m loader	DS_Loader	Data loader
m reverse	bool	Flag indicating reversed velocity field
m deltaT	double	Time step interval (time between time steps)
m divT	size_t	Number of integral divisions of time step interval
m curStep	size_t	Current time step
m releaseSkip	size_t	Number of steps skipped for particle release
m releaseMax	size_t	Maximum number of particle release processes (0 = unlimited)
m outputSkip	size_t	Number of steps skipped for outputting particle trajectory files
m outputBase	string	Base name for particle trajectory files
m outputPath	string	Naming pattern of particle trajectory files '#nR' in file naming pattern is replaced by rank number, and '#nS' is replaced by time step number. (n = integer ; filled with 0 in n-digit)
m outputMode	OutputType	Output mode of particle trajectory file Out Master: outputs all data collectively to Rank0 node. Out Local: outputs per each rank.
m outputConnectLine	bool	Mode of connecting line segments of each rank at the time of outputting file Specifies whether or not to connect the same numbered line segments of each rank..
m verbosity	bool	Flag indicating verbosity
m perfMonitor	PerfMonitor	Performance monitor

Public Methods

SetupLoader			
Argument	indexPath	const char*	Path to Index file
	divs	const SV3 &	Number of divided domains (in each axis-direction)

	cacheMemSize	const size_t	Cache memory capacity
Return	bool	true: success, false: failure	
Description	Specifies cache for loading data and data of divided domain		

SetupStartPts			
Argument	startPts	const PTarray ＜ DV3 ＞&	List of all integral start points
	startIdx	const size_t	Index number of managed integral start point
	numPts	const int	Number of managed integral start points
Return	bool	true: success, false: failure	
Description	Specifies integral start point to manage. Integral start points to manage are counted as 'numPts' starting from 'startIdx' in the list of all integral start points.		

SetupParams			
Argument	deltaT	const double	Time step interval (time between time steps) If the value is negative, leaves the setting unchanged.
	divT	const size_t	Number of integral divisions of time interval If the value is negative, leaves the setting unchanged.
	releaseSkip	const int	Time step interval for particle release If the value is negative, leaves the setting unchanged.
	releaseMax	const int	Maximum number of particle release (0= unlimited) If the value is negative, leaves the setting unchanged.
	reverse	bool	Flag indicating reversed velocity filed (0: forward direction, 1: opposite direction)
Return	bool	true: success, false: failure	
Description	Specifies the parameters for tracking particles.		

SetupOutputs			
Argument	outputBase	const string &	Base name of output file path
	outputSkip	const int	Number of steps skipped for outputting particle trajectory files If value is 0 or smaller, leaves the setting unchanged.
	outputMode	const OutputType	Output mode of particle trajectory files Out Collect: outputs all data collectively to Rank0 node. Out EachRank: outputs for each rank.
Return value	bool	true: success, false: failure	
Description	Specifies the parameters for output results. Output files are 'outputBase.idx' (Index file), 'outputBase,' rank number, time step number, and '.par' (particle trajectory file).		

GetCurPath			
Argument	rank	const int	Rank number (if value is negative, obtains the actual rank number.)

	stp	const int	Refers to time step number (if value is negative, refers to ‘m_curStep’.)
Return value	string	Name of output file	
Description	Returns the output file name. Creates the output file name based on rank number and current time step according to the specified file naming pattern. Returns a name string by replacing the ‘#nR’ or ‘#nD’ in the file naming pattern with a rank number, and ‘#nS’ with a time step number (n = integer; n-digits are filled with 0). If n is not specified or n-digit is insufficient, n is replaced by the number of required digits.		

SaveParFile			
Argument	connect	const bool	Flag determining whether or not to connect corresponding particle line segments (If true is set, each Rank must have the same number of lines.)
Return value	bool	true: success, false: failure	
Description	Outputs to particle trajectory files.		

WriteIdxFile			
Argument	None		
Return value	bool	true: success, false: failure	
Description	Outputs Index file describing particle trajectory files.		

DispatchStartPts			
Argument	start_path	const std::string &	Path to integral start point files
Return value	bool	true: success, false: failure	
Description	Loads integral start point as well as allocates integral start point to nodes of each rank. Allocates based on the number of nodes and own rank number. An error occurs if the number of all integral start points is less than the number of nodes.		

StepIntegtate			
Argument	curPts	const PTarray < DV3	List of coordinate value of particle before moving
	newPts	PTarray < DV3 >&	Storage area of coordinate value of particle after moving
Return value	bool	true: success, false: failure	
Description	Calculates particle movement Performs integral of velocity field for a single time step over coordinate values of particles stored in ‘curPts,’ and stores the coordinate values of moved particles in ‘newPts.’		

SetCurStep			
Argument	step	const size_t	Current time step number
Return value	bool	true: success, false: failure	
Description	Specifies the current time step.		

StepTransport			
Argument	None		

Return value	bool	true: success, false: failure
Description	Starts moving particle after integrating velocity field of a single time step. Since this method is a pure virtual function, implementation should be performed using derived class.	

MainLoop			
Argument	None		
Return value	bool	true: success, false: failure	
Description	Main loop of application Prior to executing this method, all values must be initialized.		

• lsvPT_ParticlePath class

This class provides the pathline calculation facility.

Public methods

SetCurStep			
Argument	step	const size_t	Current time step number
Return value	bool	true: success, false: failure	
Description	Specifies the current time step and updates coordinates of point to be integrated.		

StepTransport			
Argument	None		
Return value	bool	true: success, false: failure	
Description	Performs integral of velocity field of a single time step, and stores the coordinates of moved particles to pathline data.		

• lsvPT_StreakLine class

This class provides the streakline calculation facility.

Public methods

SetCurStep			
Argument	step	const size_t	Current time step number
Return value	bool	true: success, false: failure	
Description	Specifies the current time step and updates coordinates of point to be integrated		

StepTransport			
Argument	None		
Return value	bool	true: success, false: failure	
Description	Performs integral of velocity field of a single time step, and stores coordinates of moved particle to pathline data.		

• lsvPT_TimeLine class

This class provides the timeline calculation facility.

Public methods

SetCurStep			
Argument	step	const size_t	Current time step number
Return value	bool	true: success, false: failure	
Description	Specifies the current time step and updates coordinates of point to be integrated		

StepTransport			
Argument	None		
Return value	bool	true: success, false: failure	
Description	Performs integral of velocity field of a single time step, and stores coordinates of moved particle to pathline data.		

1.3.4. Discrete data file format

(1) Start point coordinate data

The start point coordinate data file is a text file specifying the integral starting point coordinates for pathline, streakline, and timeline, which contains geometric transformation data indicating initial positions, as well as specific coordinate values.

The geometric transformation data records the 'streamLines' map parameters of the LSV. The data will not be referred in pathline, streakline, and timeline calculations.

```
# TRANSLATE=0.0, 0.0, 0.0
# SCALE=1.0, 1.0, 1.0
# ROTATE=0.0, 0.0, 0.0
10
1.5, 2.5, 3.5
…
10.5, 11.5, 12.5
```

The lines with "#" at the head above are ignored as comments.
The first non-comment line describes the number of start points. The start point coordinates are then written in each of the subsequent lines. The coordinate values are delimited by a comma (,) or space.

Write geometric transformation data in the comment lines in the format of "type = value." Specify the type as "TRANSLATE," "SCALE," or "ROTATE," and the 'value' as the component value in each axis-direction by separating the values by a comma or a space.

(2) Discrete data file

Discrete data files store the binary trajectory data obtained by pathline, streakline, and timeline calculations.

Name	Data type	Size (bytes)	Description
NL	integer	4	Number of lines
ND	integer	4	Number of data item on each vertex
STEP	integer	4	Time step number
TIME	real number	4	Time
np	integer	4	Number of vertexes of the first line
X 1, Y 1, Z 1, V 1 1, … V ND 1	real number	8 × (3+ND)	Coordinate value and data value on the first vertex of the first line
…			
X np, Y np, Z np, V 1 np, … V ND np	real number	8 × (3+ND)	Coordinate value and data value on npth vertex of the first line

Repeatedly write the data items after 'np' as many times as the number of lines (NL).

(3) Description in Index file for discrete data

Specify the layout of divided discrete data file using the Index file in XML format.

```
<?xml version="1.0" encoding="utf-8"?>
<LSV_Index>
  <data type="PARTICLE">
    <info>
      <datas>1</datas>
      <files>2</files>
      </info>
      <step index="1" time="0.0">
        <file>Data_00_000.sca</file>
        <file>Data_01_000.sca</file>
      </step>
      <step index="2" time="0.1">
        <file>Data_00_001.sca</file>
        <file>Data_01_001.sca</file>
      </step>
     </data>
    </LSV_Index>
```

• Specify “PARTICLE” as the type attribute value of < data > tag.
• Write the following data under < info > tag:
datas: Number of data items (ND)
files: Number of file divisions
• Specify discrete data files using the < file > tag under < step > tag.
The number of files specified must be those specified with the < files > of < info > tag. Further, the number of data items in all written files must be equal to the number specified with the < datas > of <info > tag.

1.3.5. Handling discrete data by the LSV

Loading data

(1) Internal Data structure

In the case of loading discrete data, the data is not redivided during the loading process. The internal data structure is therefore defined as shown below, so that a single process can manage multiple domains when the number of divisions of file is greater than the number of processes of the LSV:

- **Discrete data type:** LSV_DATA_SYSTEM_TYPE_PARTICLE
- **Data structure:** LSVs_DATA_SYSTEM_PARTICLE

This structure stores discrete data managed by a single process.
It holds “the number of data blocks” and the corresponding “number of data block structures” internally.

Member	Data type	Description
dataSetCount	int	Number of data blocks
pDataSet	LSVs_DATA_SYSTEM_PARTICLE	Array of data block structure
orig	LSV_VECTOR3D_DOUBLE	Minimum coordinates
giro	LSV_VECTOR3D_DOUBLE	Maximum coordinates
numDatas	int	Number of data
pDids	int*	Data ID [numDatas]
pDrange	LSV RANGE DOUBLE	Data range [numDatas]

- **Data block structure**: LSVs_DATA_SYSTEM_PARTICLE_DATA

This structure stores data held by a single discrete data file.

Member	Data type	Description
numPtsAll	int	Total number of nodes
numLines	int	Number of line segments
pNumPts	int*	Number of vertexes per line segment [numLines]
step	int	Time step number
time	float	Time
orig	LSV_VECTOR3D_DOUBLE	Minimum coordinates of given block
giro	LSV_VECTOR3D_DOUBLE	Maximum coordinates of given block
pxyz	LSV_ VECTOR3D_DOUBLE*	Coordinate value [numPtsAll]
numDatas	int	Number of data
pData	double*	Data [numPtsAll*numDatas]
pDrange	LSV_RANGE_DOUBLE	Data range of given block [numDatas]

(2) Loading function

• LSVs_Particle_Preload

Prototype:

LSV_STATUS LSVs_Particle_Preload (char* pDataFile,
LSVs_DATA_PRELOADED *pPreloadedData)

Argument:

pDataFile File path to calculation result to be visualized

pPreloadedData Metadata of calculation result file

Return value:

LSV_TRUE (success), LSV_FALSE (failure)

Description:

This function loads data from discrete data files, and performs preload process.

• LSVs_Particle_Load

Prototype:

LSV_STATUS LSVs_Particle_Load (char* pDataFile,
LSVs_DATA_LOAD *pRequest, LSV_DATA_LOADED *pResponse,
LSVs_DATA *pData)

Argument:

pDataFile Path to calculation results file used for visualization

pRequest Parameter for loading calculation result file

pResponse Information about the result of loading calculation result file

pData Loaded data to be visualized

Return value:

LSV_TRUE (success), LSV_FALSE (failure)

Description:

This function loads data from discrete files, and converts the data to visualizable data.

Visualization map

1. plotLines

Role:

Parses discrete data as line data, and displays lines.

Build parameter XML format

```
<GraphicExtendedBuild>
  <Entity>
    <param name="use_data" value="dataN | veclen" />
    <param name="use_cmap" value="yes|no" />
    <param name="line_width" value="1.0" />
  </Entity>
</GraphicExtendedBuild>
```

Description of parameters

• **use_data (string: “dataN | veclen”)**

Specify the data to be referred as line color. Specify N an integer from 0 to the number of data items minus 1 (–1). Specifying “veclen” displays lines based on the vector norm if the data is vector data.

The default value when omitted is “data0.”

• **use_cmap (string: “yes | no”)**

Specify whether or not to refer to the color map for line color selection. If "no" is specified, the line is displayed in base color.

The default value when omitted is “yes.”

• **line_width (real number)**

Specify the width of the line in pixels. The value must be greater than 0.0.

The default value when omitted is 1.0.

Response XML format

The < Entity > tag is not used.

Writing format of “lsv_coreconfiguration.xml”

```
<LSVs_CoreConfiguration>
  <Data name="Particle">
    <Graphic name="plotLines">
      <BuildFunction>Part_plotLines</BuildFunction>
      <Library>LSVg_builders.so</Library>
    </Graphic>
  </Data>
</LSVs_CoreConfiguration>
```

2. plotPoints

Role:

Parses discrete data as line data and displays a vertex.

Build parameter XML format

```
<GraphicExtendedBuild>
  <Entity>
    <param name="use_data" value="dataN | veclen" />
    <param name="use_cmap" value="yes|no" />
    <param name="line_width" value="1.0" />
  </Entity>
</GraphicExtendedBuild>
```

Description of parameters

• **use_data (string: “dataN | veclen”)**

Specify the data to be referred as point color. Specify N an integer from 0 to the number of data items minus 1 (–1). Specifying “veclen” displays points based on the vector norm if the data is vector data.

The default value when omitted is “data0”

• use_cmap (string: “yes | no”)

Specify whether or not to refer to the color map for point color selection. If "no" is specified, the point is displayed in base color.

The default value when omitted is “yes.”

• line_width (real number)

Specify the width of the line in pixels. The value must be greater than 0.0.

The default value when omitted is 1.0.

Response XML format

The < Entity > tag is not used.

Writing format of “lsv_coreconfiguration.xml”

```
<LSVs_CoreConfiguration>
  <Data name="Particle">
    <Graphic name="plotPoints">
      <BuildFunction>Part_plotPoints</BuildFunction>
      <Library>LSVg_builders.so</Library>
    </Graphic>
  </Data>
</LSVs_CoreConfiguration>
```

Utility tools

1. ParIndexMaker.py

Role:

Creates Index file for discrete data.

Command line

ParIndexMaker.py --data data_pattern
--div start,end,skip --stp start,end,skip [--nd N][--out file.idx]

Command line options

• --data data pattern

Specifies file naming patterns of data files

File naming pattern can include a domain number pattern such as “%05D” or a time step number pattern such as “%08T.”

• --div start,end,skip

Specifies the start and end division numbers and the number of steps to skip, with the values delimited by a comma (,).

If the number of steps to skip is omitted, “1” is assumed.

• --stp start,end,skip

Specifies the start and end time step numbers and the number of steps to skip, with the values delimited by a comma (,).

If the number of steps to skip is omitted, “1” is assumed.

• --nd N

Specifies the number of data items. The default value when omitted is 0.

• --out file.idx

Specifies the name of the Index file to be created.

If omitted, the file contents are output to the standard output facility.

2. Handling Unstructured-Grid Data

2.1. Outline

This chapter describes the capabilities implemented in the LSV for loading and visualizing time-series unstructured-grid data.

Unstructured-grid data dealt in this chapter consist of tetrahedron and hexahedron components and assumes to be divided into multiple domains and stored in a file.

The work in this chapter involves the format design of unstructured-grid data, data loading by the LSV system, and visualization map implementation.

2.2. File format of unstructured-grid data

This section explains the file format of unstructured-grid data used in the LSV system.

Unstructured-grid data consists of two formats: Mesh file and Data file.

2.2.1. Mesh file

The Mesh file is a binary format file to store mesh information of the unstructured-grid data. It is used to write data of a divided domain of unstructured-grid mesh. This format takes into account the describability of sleeve areas such as the adjacent node and elements when domains are divided.

Block name	Description
Node information	Stores information of nodes included in given domain.
Element information	Stores information of elements included in given domain.
Group information	Stores information about group of nodes, elements and planes.
Communication table information	Stores information related the communication between adjacent domains.
Node coordinates information	Stores coordinate values of nodes included in given domain.

Node information block

Name	Data type	Size (bytes)	Description
nNode	integer	4	Number of nodes inside the entire domain
nnNode	integer	4	Total number of nodes in domain
Nid_1～Nid_nNode	integer	4 × nNode	Global node number of nodes inside the entire domain
Nid_nNode+1～Nid_nnNode	integer	4 × (nnNode-nNode)	Global node number of sleeve nodes within given node

Element information block

Name	Data	Size (bytes)	Description
nETypes	integer	4	Number of element types
EType	integer	4	Element type (4:TETRA,8:HEXA)
nEInner	integer	4	Number of elements per EType in the entire domain
nEOverlap	integer	4	Number of elements per EType in sleeve area
Eid_1～Eid_nEInner	integer	4 × nEInner	Global element number of EType elements in the entire domain

Eid _nEInner+1～ Eid _(nEInner+nEOverlap)	integer	4 × nEOverlap	Global element number of EType elements in sleeve area
E1_Nid1～E1_NidM	integer	4 × M	Local element number row composing the first element of EType (M is TETRA:4 or HEXA:8 depending on EType)
E2_Nid1～E2_NidM	integer	4 × M	Local element number row composing the second element of EType
…			
EnnElem_Nid1～ EnnElem_NidM	integer	4 × M	Local element number row composing 'nnElem' element of EType (nnElem = nEInner+nEOverlap)

Repeatedly write the data items after "EType" as many times as the number of "nETypes."
Note that the "ETypes" value is confined to 1 or 2.

Group information block

(1) Node group information

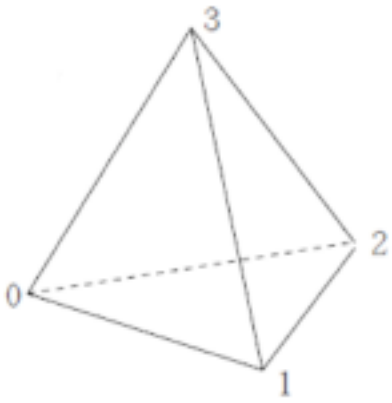
Name	Data	Size (bytes)	Description
nNgrp	integer	4	Number of node groups
NgrpName_1	string	128	Name of the first node group
…			
NgrpName_nNgrp	string	128	Name of the "nNgrp" node group
NgrpMark	integer	4	Storage mark of the number of nodes belonging to node group (fixed to 0)
nNgrpSum_1	integer	4	Number of nodes belonging to the first node group
nNgrpSum_2	integer	4	Number of nodes belonging to the first node group + Number of nodes belonging to the second node group
…			
nNgrpSum_nNgrp	integer	4	Number of nodes belonging to the first node group + Number of nodes belonging to the second node group +… + Number of nodes belonging to the 'nNgrp' node group
Ngrp_1_Nid_1 ～ Ngrp_1_Nid_nNG1	integer	4 × nNG1	Local node number row belonging to the first node group (nNG1= Number of nodes belonging to the first node group)
Ngrp_2_Nid_1 ～ Ngrp_2_Nid_nNG2	integer	4 × nNG2	Local node number row belonging to the second node group (nNG2= Number of nodes belonging to the second node group)
…	integer		

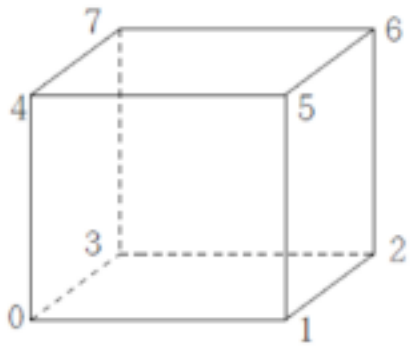
Ngrp_nNgrp_Nid_1 ～ Ngrp_nNgrp_Nid_nNGnNgrp	integer	4 × nNGnNgrp	Local node number row belonging to the 'nNgrp' node group (nNGnNgrp= Number of nodes belonging to the 'nNgrp' node group)

(2) Element group information

Name	Data type	Size (bytes)	Description
nEgrp	integer	4	Number of element groups
EgrpName_1	string	128	Name of the first element group
…			
EgrpName_nEgrp	string	128	Name of the 'nEgrp' element group
EgrpMark	integer	4	Storage mark of the number of elements belonging to element group (fixed to 0)
nEgrpSum_1	integer	4	Number of elements belonging to the first element group
nEgrpSum_2	integer	4	Number of elements belonging to the second element group
…			
nEgrpSum_nEgrp	integer	4	Number of elements belonging to the first element group + Number of elements belonging to the second element group +… + Number of elements belonging to the 'nNgrp' element group
Egrp_1_Eid_1 ～ Egrp_1_Eid_*nEG*1	integer	4 × nEG1	Local element number row belonging to the first element group (nEG1= Number of elements belonging to the first element group)
Egrp_2_Eid_1 ～ Egrp_2_Eid_nEG2	integer	4 × nEG2	Local element number row belonging to the second element group (nEG2= Number of elements belonging to the second element group)
…			
Egrp_nEgrp_Eid_1 ～ Egrp_nEgrp_Eid_nEGnEgrp	integer	4 × nEGnEgrp	Local element number row belonging to the 'nNgrp' element group (nEGnEgrp= Number of elements belonging to the 'nNgrp' element group)

Plane number within element (TETRA) Plane number within element (HEXA)





Plane number	Node composing plane
0	1 2 3
1	0 2 1
2	0 1 3
3	0 3 2

Plane number	Node composing plane
0	3 0 4 7
1	1 2 6 5
2	0 1 5 4
3	2 3 7 6
4	3 2 1 0
5	4 5 6 7

(3) Plane group information

Name	Data type	Size (bytes)	Description
nSgrp	integer	4	Number of plane groups
SgrpName_1	string	128	Name of the first plane group
…			
SgrpName_nNgrp	string	128	Name of the 'nSgrp' plane group
SgrpMark	integer	4	Storage mark of the number of planes belonging to plane group (fixed to 0)
nSgrpSum_1	integer	4	Number of planes belonging to the first plane group
nSgrpSum_2	integer	4	Number of planes belonging to the first plane group + Number of plane belonging to the second plane group
…			
nSgrpSum_nNgrp	integer	4	Number of planes belonging to the first plane group + Number of planes belonging to the second plane group +… + Number of planes belonging to the 'nNgrp' plane group
Sgrp _1_Eid _1, Sgrp _1_Eid _1_S ~ Sgrp_1_Eid_nSG1, Sgrp_1_Eid_nSG1_S	integer	4 × 2 × nSG1	Local element number belonging to the first plane group and plane number row in the element (nSG1= Number of planes belonging to the first plane group)

…			
Sgrp_nSgrp_Eid_1, Sgrp_nSgrp_Eid_1_S ～ Sgrp_nSgrp_Eid_nSGnSgrp, Sgrp_nSgrp_Eid_nSGnSgrp_S	integer	4 × 2 × nSGnSgrp	Local element number belonging to the 'nSgrp' plane group and plane number row in the element (nSGnSgrp= Number of planes belonging to the 'nSgrp' plane group)

Communication table information block

Name	Data type	Size (bytes)	Description
nPE	integer	4	Number of communication PEs
PEid 1 ～ PEid nPE	integer	4 × nPE	PE number
PE1 nN ～ PEnPE nN	integer	4 × nPE	Number of transmitting nodes
PE1 Nid 1 ～ PE1 Nid nN	integer	4 × PE1 nN	Local transmitting node number row of the first PE
…			
PEnPE Nid 1 ～ PEnPE Nid nN	integer	4 × PEnPE nN	Local transmitting node number row of the nPE

Node coordinate block

Name	Data type	Size (bytes)	Description
nnNode	integer	4	Total number of nodes within domain
nD	integer	4	Number of space dimensions
X1 1, X2 1,…, XnD 1	real number	8 × nD	Coordinates of the first node
…			
X1 nnNode, X2 nnNode,…,	real number	8 × nD	Coordinates of the 'nnNode' node

2.2.2. Data file

The Data file is a binary format file to store the physical property value of unstructured-grid data. It is used to describe the data of a divided domain of unstructured-grid mesh. It stores the node and element data described in the corresponding Mesh file in the same order.

Block name	Description
Node data	Stores node data included in given domain.
Element data	Stores element data included in given domain.

Node data block

Name	Data type	Size (bytes)	Description
nnNode	integer	4	Total number of nodes within domain
nND	integer	4	Number of node data items
ND_1_1 ～ ND_1_nND	real number	8 × nND	Data of the first node
ND_2_1 ～ ND_2_nND	real number	8 × nND	Data of the second node
…			
ND_nnNode_1～ ND_nnNode_nND	real number	8 × nND	Data of the 'nnNode' node

Element data block

Name	Data type	Size (bytes)	Description

nnElem	integer	4	Total number of elements within domain
nED	integer	4	Number of element data items
ED_1_1 ～ ED_1_nED	real number	8 × nED	Data of the first element
ED_2_1 ～ ED_2_nED	real number	8 × nED	Data of the second element
…			
ED_nnElem_1 ～ ED_nnElem_nED	real number	8 × nED	Data of the ‘nnElem’ element

2.2.3. Description in Index files

Specify the layout of Mesh file and Data file of each domain created by dividing a domain, using the Index file in XML format.

Writing format

```
<?xml version="1.0" encoding="utf-8"?>
<LSV_Index>
  <data type="UNS">
    <info>
      <files>2</files>
      <ndatas>3</ndatas>
      <edatas>0</edatas>
    </info>
    <mesh>
      <file>Mesh_00.usm</file>
      <file>Mesh_01.usm</file>
    </mesh>
    <step index="1" time="0.0">
      <file>Data_00_000.usd</file>
      <file>Data_01_000.usd</file>
    </step>
    <step index="2" time="0.1">
      <file>Data_00_001.usd</file>
      <file>Data_01_001.usd</file>
    </step>
  </data>
</LSV_Index>
```

Description of tag

• < data >

Specifies “UNS” in ‘type’ attribute

• < info >

Writes the following information:

ndatas Number of node items (nND)
edatas Number of element items (nED)
files Number of file divisions

• < mesh >

Specifies Mesh files by < file > tag under < mesh > tag, and Data file by < file > tag under < step > tag. The number and order of files under < mesh > and < step > tags must be identical to the number specified with the < datas> of < info > tag.

• Writing convention for other tags follows the way described for the Index file of the SPH/SPX format.

2.3. Loading unstructured-grid data

(1) Internal data structure

In the case of loading unstructured-grid data, the data is not redivided during the loading process. The internal data structure is therefore defined as shown below, so that a single process can manage multiple domains when the number of divisions of file is greater than the number of processes of the LSV:

- Data type of unstructured grid data: LSV_DATA_SYSTEM_TYPE_UNSTRUCTUREDMESH
- Data structure: LSVs_DATA_SYSTEM_UNSTRUCTUREDMESH

This structure stores unstructured-grid data managed by a single process.
It holds "the number of data blocks" and the corresponding "number of data block structure" internally.

Member name	Data type	Description
numBlocksTotal	int	Total number of data blocks
numBlocks	int	Number of data blocks
pBlocks	LSVs_DATA_SYSTEM_UNS_BLOCK*	Divided block [numBlocks]
orig	LSV_VECTOR3D_DOUBLE	Minimum coordinates
giro	LSV_VECTOR3D_DOUBLE	Maximum coordinates
numNDatas	int	Number of node data
pNDids	int*	Node data ID[numNDatas]
pNDrange	LSV_RANGE_DOUBLE	Node data range [numNDatas]
numEDatas	int	Number of element data
pEDids	int*	Element data ID[numEDatas]
pEDrange	LSV RANGE DOUBLE	Element data range [numEDatas]

- **Data block structure:** LSVs_DATA_SYSTEM_UNS_BLOCK

This structure stores data held by an unstructured-grid data file.

Member name	Data type	Description
		＜Node information block＞
nNodes	int	Total number of nodes
nInnerNodes	int	Number of nodes inside the entire domain
pNode	LSV_UNS_NODE*	Node [nNodes]
		＜Element information block＞
nElems	int	Total number of elements
numElemBlocks	int	Number of element blocks
pElemBlock	LSV_UNS_ELEM_BLOCK	Element block [numElemBlocks]
		＜Group information block＞
nNgrp	int	Number of node groups
ppNgrpName	char**	Name of node groups [nNgrp][LSV_UNS_GRPNAME_LEN]
pNgrpIndex	int*	Index of node number belonging to node group [nNgrp+1]
pNgrpNids	int*	Local node number belonging to node group [pNgrpIndex[nNgrp]]

nEgrp	int	Number of element groups
ppEgrpName	char**	Name of element groups [nEgrp][LSV_UNS_GRPNAME_LEN]
pEgrpIndex	int*	Index of element number belonging to element group [nEgrp+1]
pEgrpEids	int*	Local element number belonging to element group [pEgrpIndex[nEgrp]]
nSgrp	int	Number of plane groups
ppSgrpName	char**	Name of plane groups [nSgrp][LSV_UNS_GRPNAME_LEN]
pSgrpIndex	int*	Index of node number belonging to plane group [nSgrp+1]
pSgrpESids	int*	Local element/plane number belonging to plane group [pNgrpIndex[nSgrp]*2]
		＜Communication table block＞
nPE	int	Number of communication PEs
pPEids	int*	PE number [nPE]
pPEnN	int*	Number of transmitting nodes per PE [nPE]
ppPEnids	int**	Local transmitting node number per PE [nPE][pPEids[i]]
		＜Node coordinates block＞
orig	LSV_VECTOR3D_DOUBLE	Minimum coordinates
giro	LSV_VECTOR3D_DOUBLE	Muximum coordinates
		＜Node data block＞
nNDatas	int	Number of node data
pNData	double*	Node data [nNodes*nNDatas]
pNDrange	LSV_RANGE_DOUBLE	Node data range [numNDatas]
		＜Element data block＞
nEDatas	int	Number of element data
pEData	double*	Element data [nElems*nEDatas]
pEDrange	LSV_RANGE_DOUBLE	Element data range [numEDatas]

・**Element block structure:** LSV_UNS_ELEM_BLOCK

This structure stores the information of the element block held by each element.

Member name	Data type	Description
elemType	LSV_UNS_ELEM_TYPE	Element type (LSV_UNS_TET｜LSV_UNS_HEX)
nElems	int	Number of elements
nElemsInner	int	Number of elements inside the entire domain
pElem	void*	Element [nElems] Array of LSV_UNS_ELEM_TET or LSV_UNS_ELEM_HEX

・**Tetrahedron element structure:** LSV_UNS_ELEM_TET

This structure stores the information of the tetrahedron element.

Member name	Data type	Description
eid	int	Global element number
pNids	int[4]	Local node number composing element

• **Hexahedron element structure:** LSV_UNS_ELEM_HEX

This structure stores the information of the hexahedron element.

Member name	Data type	Description
eid	int	Global element number
pNids	int[8]	Local node number composing element

• **Node information structure:** LSV_UNS_NODE

This structure stores the node information.

Member name	Data type	Description
nid	int	Global node number
xyz	LSV_VECTOR3D_DOUBLE	Node coordinates

(2) Loading functions

• LSVs_Uns_Preload

Prototype:

```
LSV_STATUS LSVs_Uns_Preload(char* pDataFile,
        LSVs_DATA_PRELOADED *pPreloadedData)
```

Argument:

pDataFile File path to calculation result to be visualized
pPreloadedData Metadata of calculation result file

Return value:

LSV_TRUE (success), LSV_FALSE (failure)

Description:

This function loads data from unstructured-grid data files, and performs preload process.

• LSVs_Uns_Load

Prototype:

```
LSV_STATUS LSVs Uns Load (char* pDataFile,
        LSVs_DATA_LOAD *pRequest, LSV DATA LOADED *pResponse,
        LSVs_DATA *pData)
```

Argument:

pDataFile Path to calculation results file used for visualization
pRequest Parameter for loading calculation result file
pResponse Information about the result of loading calculation result file
pData Loaded data to be visualized

Return value:

LSV_TRUE (success), LSV_FALSE (failure)

Description:

This function loads data from unstructured-grid data files, and converts the data to visualizable data.

2.4. Visualization maps

2.4.1. boundsLine

Role:

This map displays the line of the data bounding box of unstructured-grid data.

Build parameter XML format

```
<GraphicExtendedBuild>
  <Entity>
    <param name="line_width" value="1.0" />
    <param name="each_block" value="yes|no" />
  </Entity>
</GraphicExtendedBuild>
```

Description of parameters

- **line_width (real number)**

  Specify the width of the line in pixels. The value must be greater than 0.0.

  The default value when omitted is 1.0.

- **each_block (string: ”yes｜no”)**

  Specify whether or not to display the data bounding box per data block (divided file).

  If “no” is set only the data bounding box of entire data is displayed.

  The default value when omitted is “no.”

Response XML format

The < Entity > tag is not used.

Writing format of “lsv_coreconfiguration.xml”

```
<LSVs_CoreConfiguration>
  <Data name="UnstructuredMesh">
    <Graphic name="boundsLine">
      <BuildFunction>Uns_boundsLine</BuildFunction>
      <Library>LSVg_builders.so</Library>
    </Graphic>
  </Data>
</LSVs_CoreConfiguration>
```

2.4.2. graphPlot

Role:

This map creates a graph by sampling data on a specified line in unstructured-grid data

It displays colored lines and pointers that indicate sampling points in the color specified in “base color.”

The sampling points are defined by three parameters: “translate,” “scale,” and “rotate.” The sampling points in the initial state are on a line segment drawn parallel to the X-axis, starting from the coordinate origin of the data bounding box and having length same as that of its side along the X-axis. The final sampling points are determined by applying the following geometric transformation to the line:

[M] = [T][Ry][Rx][Rz][S]

where

T: Parallel translation specified by “translate” parameter

Ry: Rotation around Y-axis specified by “rotate” parameter

Rx: Rotation around X-axis specified by “rotate” parameter

Rx: Rotation around Z-axis specified by “rotate” parameter

S: Magnification and reduction specified by “scale” parameter

If the registered data is vector data, the graphPlot samples data on the selected component or vector norm. The client terminal screen displays a graph according to the sampling results returned by “response.”

Build parameter XML format

```
<GraphicExtendedBuild>
  <Entity>
    <param name="use_data" value="dataNn ｜ dataEn ｜ veclenN ｜ veclenE" />
    <param name="show_sampler" value="yes ｜ no" />
    <param name="line_width" value="1.0" />
    <param name="point_size" value="2.0" />
    <param name="translate" value="0.0 0.0 0.0" />
    <param name="scale" value="1.0 1.0 1.0" />
    <param name="rotate" value="0.0 0.0 0.0" />
    <param name="num_points" value="10" />
  </Entity>
</GraphicExtendedBuild>
```

Description of parameters

- **num_points (integer)**
  Specify the number of sampling points.
  The default value when omitted is “10.”
- **use_data (string: “dataNn｜dataEn｜veclenN｜veclenE”)**
  Specify the data format and type for plotting the graph by using “dataNn” format (n: integer, 0 to the number of node data-1) for node data, and “dataEn” format (n: integer, 0 to the number of element data -1) for element data.
  Specifying “veclenN” displays graphs based on the vector norm if node data in vector data.
  Specifying “veclenE” displays graphs based on the vector norm if element data is vector data.
  The default value when omitted is “dataN0” or “dataE0.”
- **show_sampler (string: “yes｜no”)**
  Specify whether or not to display the sampler. If “no” is selected, the sampler is not displayed.
  The default value when omitted is “yes.”
- **line_width (real number)**
  Specify the width of the sampler line.
  The default value when omitted is “1.0.”
- **point_size (real number)**
  Specify the size of the sampler point.
  The default value when omitted is “2.0.”
- **translate (real number real number real number)**
  Specify the translation amount of the sampling point in relation to the data area origin, using real numbers. When (0.0, 0.0, 0.0) is set, the data bounding box origin is regarded as the sampling point origin.
- **scale (real number real number real number)**
  Specify the magnification ratio of the sampling point in relation to the initial state. When (1.0, 1.0, 1.0) is set, the length of the sampling point line is equal to that of the data bounding box in X-direction. Note that the Y and Z components are ignored.
- **rotate (real number real number real number)**
  Specify in degrees the rotation amount of the sampling point with respect to the initial state. When (0.0, 0.0, 0.0) is set, the sampling point line coincides with the X-axis.

Response XML format

```
<GraphicExtendedBuildResponse>
  <Entity>
  <Message>graphic builder: graphPlot:¥n
```

```
        N ＼n
        T1   D1 ＼n
        T2   D2 ＼n
        …
        TN   DN ＼n
    </Message>
    </Entity>
</GraphicExtendedBuildResponse>
```

Under the <Entity> node, the <Message> node is described where a pair of N (number of sampling points), T (sampling point) and D (sampled physical property value) is written. T is the parameter value (0.0 ~ 1.0) representing a sampled line segment.

Writing format of "lsv_coreconfiguration.xml"

```
<LSVs_CoreConfiguration>
  <Data name="UnstructuredMesh">
    <Graphic name="graphPlot">
      <BuildFunction>Uns_graphPlot</BuildFunction>
      <Library>LSVg_builders.so</Library>
    </Graphic>
  </Data>
</LSVs_CoreConfiguration>
```

2.4.3. isosurf

Role:

This map displays isosurface based on node data of unstructured-grid data.

Isosurfaces are not displayed for element data.

Build parameter XML format

```
<GraphicExtendedBuild>
  <Entity>
    <param name="use_data" value="dataN  |  veclen  |  none" />
    <param name="iso_value" value="0.0" />
    <param name="use_cmap" value="yes  |  no" />
    <param name="use_cmap_alpha" value="yes  |  no" />
    <param name="two_side" value="yes  |  no" />
    <param name="xdr" "0.0 1.0" />
  </Entity>
</GraphicExtendedBuild>
```

Description of parameters

- **use_data (string: "dataN|veclen|none")**

Specify node data used for displaying isosurface in "dataN" format (N: integer, 0 to the number of node data -1).

Specifying "veclen" displays isosurfaces based on the vector norm if the data is vector data. If "none" is specified, no isosurface is displayed.

The default value when omitted is "data0."

- **iso_value (real number)**

Specify the value of isosurface level.

- **use_cmap (string: "yes|no")**

Specify whether or not to use the color map for isosurface colors. If "no" is set, the base color is applied.

The default value when omitted is "yes."

• **use_cmap alpha (string: “yes|no”)**

Specify whether or not to use the color map for isosurface opacity. If “no” is set, the base color opacity is applied.

The default value when omitted is “no.”

• **two_side (string: “yes|no”)**

Specify whether to display both sides (front and back) of isosurface. If “no” is set, the back of isosurface is not displayed.

The default value when omitted is ‘yes.”

• **xdr (real number real number)**

Specify the data range to be displayed by separating with space characters.

Response XML format

The < Entity > tag is not used.

Writing format of “lsv_coreconfiguration.xml”

```
<LSVs_CoreConfiguration>
  <Data name="UnstructuredMesh">
    <Graphic name="isosurf">
      <BuildFunction>Uns_isosurf</BuildFunction>
      <Library>LSVg_builders.so</Library>
    </Graphic>
  </Data>
</LSVs_CoreConfiguration>
```

2.4.4. probe

Role:

This map samples data at the specified points in unstructured-grid data and reports the result.

The symbols indicating the sampling points are displayed as a map. The symbols are displayed in the base color.

The sampling point is specified by “position.” The sampling point in the initial state is the coordinate origin of data bounding box, and the final sampling point is determined by translating the origin as specified in “position.”

The client terminal screen displays the results according to the sampling results returned by “response.”

Build parameter XML format

```
<GraphicExtendedBuild>
  <Entity>
    <param name="position" value="0.0 0.0 0.0" />
    <param name="scale" value="0.0 0.0 0.0" />
    <param name="angle" value="0.0 0.0 0.0" />
  </Entity>
</GraphicExtendedBuild>
```

Description of parameters

• **position (real number real number real number)**

Specify the translation amount of the sampling point in relation to the initial state.

When (0.0, 0.0, 0.0) is set, the data bounding box origin is regarded as the sampling point origin.

• **scale (real number real number real number**

Specify the magnification ratio of the probe symbol in relation to the initial state.

When (1.0 1.0 1.0) is set, the size of the probe symbol is set to 1.0 in length and 0.1 in width.

• **angle (real number real number real number)**

Specify in degrees the rotation amount of the probe symbol with respect to the initial state.

When (1.0 1.0 1.0) is set, the longitudinal direction of the probe symbol coincides with the Y-axis.

Response XML format

```
<GraphicExtendedBuildResponse>
 <Entity>
 <Message>graphic builder: probe:¥n
        Coord:
          x = 0.500000
          y = 0.500000
          z = 0.500000
        Elem:
          id = 444
          type = HEXA
        Compose Node:
          id = 532, 533, 544, 543,
          653, 654, 665, 664
        Node Data:
          # of node data : 3
          dataN0 = 0.000000
          dataN1 = 0.000000
          dataN2 = 0.000000
        Elem Data:
          # of elem data : 0
 </Message>
 </Entity>
</GraphicExtendedBuildResponse>
```

Coordinate values of sampling point, number and type of element that includes the sampling point, node number of the element, and sampled data value are written under the < Entity > node.

Writing format of "lsv_coreconfiguration.xml"

```
<LSVs_CoreConfiguration>
  <Data name="UnstructuredMesh">
    <Graphic name="probe">
      <BuildFunction>Uns_probe</BuildFunction>
      <Library>LSVg_builders.so</Library>
    </Graphic>
  </Data>
</LSVs_CoreConfiguration>
```

2.4.5. shapeMap

Role:

The shapeMap displays geometry surfaces based on unstructured grid data.

Build parameter XML format

```
<GraphicExtendedBuild>
  <Entity>
    <param name="use_data" value="dataNn | dataEn | veclenN | veclenE | none" />
    <param name="two_side" value="yes | no" />
    <param name="shrink" value="1.0" />
    <param name="offset_disp" value="0.0" />
  </Entity>
</GraphicExtendedBuild>
```

Description of parameters

- **use_data (string: “dataNn｜dataEn｜veclenN｜veclenE｜none”)**

Specify the data to be referred as geometry surface color by using “dataNn” format (n: integer, 0 to the number of node data-1) for node data, and “dataEn” format (n: integer, 0 to the number of element data-1) for element data.

Specifying “veclenN” displays geometry surfaces based on the vector norm if node data in vector data.

Specifying “veclenE” displays geometry surfaces based on the vector norm if element data is vector data.

If ”none” is specified, geometry surfaces are displayed in base color.

The default value when omitted is “none.”

- **two_side (string: “yes｜no”)**

Specify whether or not to display both sides (front and back) of geometry surface.

If “no” is set, the backside is not displayed.

The default value when omitted is “no.”

- **shrink (real number: 0.0~1.0)**

Specify the reducing ratio of the elements constructing geometry surface on each face. If 1.0 is set, the element size is not reduced.

The default value when omitted is “1.0.”

- **offset_disp (real number)**

When node data is provided as vector data, specify a coefficient to transform the geometry by regarding the data value as a displacement. If 0.0 is set, the geometry is not transformed.

The default value when omitted is “0.0.”

Response XML format

The < Entity > tag is not used.

Writing format of “lsv_coreconfiguration.xml”

```
<LSVs_CoreConfiguration>
  <Data name="UnstructuredMesh">
    <Graphic name="shapeMap">
      <BuildFunction>Uns_shapeMap</BuildFunction>
      <Library>LSVg_builders.so</Library>
    </Graphic>
  </Data>
</LSVs_CoreConfiguration>
```

2.4.6. sliceContour

Role:

This map displays contour lines on an arbitrary slice of unstructured grid node data.

Build parameter XML format

```
<GraphicExtendedBuild>
  <Entity>
    <param name="use_data" value="dataNn ｜ veclenN" />
    <param name="data_range" value="0.0;1.0" />
    <param name="num_lines" value="5" />
    <param name="use_cmap" value="yes ｜ no" />
    <param name="line_width" value="1.0" />
    <param name="center_pos" value="0.0 0.0 0.0" />
    <param name="rotate" value="0.0 0.0 0.0" />
  </Entity>
</GraphicExtendedBuild>
```

Description of parameters

- **use_data (string: “dataNn | veclenN”)**

  Specify the node data to display as contour lines by using “dataNn” format (n: integer, 0 to the number of node data -1).

  Specifying “veclenN” displays contour lines based on the vector norm if node data in vector data.

  The default value when omitted is “dataN0.”

- **data_range (real number ; real number)**

  Specify the value range to display vector lines by separating with “;.”

  If omitted, the value range of the specified node data is applied.

- **num_lines (integer)**

  Specify the number of contour lines to be displayed.

  The default value when omitted is 5.

- **use_cmap (string: “yes | no”)**

  Specify whether or not to refer to the color map for contours. If “no” is set, contours are displayed in base color.

  The default value when omitted is “yes.”

- **line_width (real number)**

  Specify the contour width in pixels.

  The default value when omitted is “1.0.”

- **center_pos (real number real number real number)**

  Specify the coordinates of a virtual centroid of a slice by separating the values with a space character. The point set here is regarded as the rotation center in the following item “rotate”.

  The default value when omitted is the centroid of data bounding box.

- **rotate (real number real number real number)**

  Specify the rotation amount (in degrees) of the slice around each axis by separating the values with a space character. The slice plane is rotated from the initial state (XY plane) as specified below, with the previous item “center pos” as its center:

  [M] = [Ry][Rx][Rz]; where [Rx]; [Ry]; [Rz] denote the rotations around X, Y, and Z axis, respectively.

  The default value when omitted is (0.0, 0.0, 0.0).

Response XML format

The < Entity > tag is not used.

Writing format of “lsv_coreconfiguration.xml”

```
<LSVs_CoreConfiguration>
  <Data name="UnstructuredMesh">
    <Graphic name="sliceContour">
      <BuildFunction>Uns_sliceContour</BuildFunction>
      <Library>LSVg_builders.so</Library>
    </Graphic>
  </Data>
</LSVs_CoreConfiguration>
```

2.4.7. sliceScalar

Role:

This map displays color fringe on an arbitrary slice of unstructured grid data.

Build parameter XML format

```
<GraphicExtendedBuild>
  <Entity>
    <param name="use_data" value="dataNn | dataEn | veclenN | veclenE | none" />
    <param name="center_pos" value="0.0 0.0 0.0" />
    <param name="rotate" value="0.0 0.0 0.0" />
  </Entity>
```

```
</GraphicExtendedBuild>
```

Description of parameters

• use_data (string: ”dataNn | veclenN”)

Specify the target data to display color fringe by using “dataNn” format (n: integer, 0 to the number of node data-1) for node data, and “dataEn” format (n: integer, 0 to the number of element data-1) for element data.

Specifying “veclenN” displays color fringes based on the vector norm if node data in vector data.

Specifying “veclenE” displays color fringes based on the vector norm if element data is vector data.

If ”none” is specified, color fringes are displayed in base color.

The default value when omitted is “dataN0” or “dataE0.”

• center_pos (real number real number real number)

Specify the coordinates of a virtual centroid of a slice by separating the values with a space character. The point set here is regarded as the rotation center in the following item “rotate”.

The default value when omitted is the centroid of data bounding box.

• rotate (real number real number real number)

Specify the rotation amount (in degrees) of the slice around each axis by separating the values with a space character. The slice plane is rotated from the initial state (XY plane) as specified below, with the previous item “center pos” as its center:

[M] = [Ry][Rx][Rz]; where [Rx]; [Ry]; [Rz] denote the rotations around X, Y, and Z axis, respectively.

The default value when omitted is (0.0, 0.0, 0.0).

Response XML format

The < Entity > tag is not used.

Writing format of “lsv_coreconfiguration.xml”

```
<LSVs_CoreConfiguration>
  <Data name="UnstructuredMesh">
    <Graphic name="sliceScalar">
      <BuildFunction>Uns_sliceScalar</BuildFunction>
      <Library>LSVg_builders.so</Library>
    </Graphic>
  </Data>
</LSVs_CoreConfiguration>
```

2.4.8. sliceVector

Role:

This map displays vectors on an arbitrary slice of non-structure grid data.

Build parameter XML format

```
<GraphicExtendedBuild>
  <Entity>
    <param name="use_data" value="dataNn | dataEn | veclenN | veclenE" />
    <param name="vec_scale" value="1.0" />
    <param name="vec_head" value="yes | no" />
    <param name="use_cmap" value="yes | no" />
    <param name="line_width" value="1.0" />
    <param name="center_pos" value="0.0 0.0 0.0" />
    <param name="rotate" value="0.0 0.0 0.0" />
    <param name="xdr" value="0.0 1.0" />
  </Entity>
</GraphicExtendedBuild>
```

Description of parameters

- **use_data (string:“dataNn | veclenN)**

  Specify the target data to display vector colors by using “dataNn” format (n: integer, 0 to the number of node data-1) for node data, and “dataEn” format (n: integer, 0 to the number of element data-1) for element data.

  Specifying “veclenN” displays color fringes based on the vector norm if node data in vector data.

  Specifying “veclenE” displays color fringes based on the vector norm if element data is vector data.

  The default value when omitted is “dataN0” or “dataE0.”

- **vec scale (real number)**

  Specify the scaling ratio for displaying vectors.

  The default value when omitted is “1.0.”

- **vec_head (string: “yes | no)**

  Specify whether or not to display an arrow at the vector end.

  If “no” is set, vectors are displayed as a line.

  The default value when omitted is “yes.”

- **use_cmap (string:“yes | no)**

  Specify whether or not to use the color map for displaying vectors.

  If “no” is set, vectors are displayed in base color.

  The default value when omitted is “yes.”

- **line_width (real number)**

  Specify the width of vector lines in pixels.

  The default value when omitted is “1.0.”

- **center_pos (real number real number real number)**

  Specify the coordinates of a virtual centroid of a slice by separating the values with a space character. The point set here is regarded as the rotation center in the following item “rotate”.

  The default value when omitted is the centroid of data bounding box.

- **rotate (real number real number real number)**

  Specify the rotation amount (in degrees) of the slice around each axis by separating the values with a space character. The slice plane is rotated from the initial state (XY plane) as specified below, with the previous item “center pos” as its center:

  [M] = [Ry][Rx][Rz]; where [Rx]; [Ry]; [Rz] denote the rotations around X, Y, and Z axis, respectively.

  The default value when omitted is (0.0, 0.0, 0.0).

- **xdr (real number real number)**

  Specify the data range to be displayed by separating with space characters.

Response XML format

The < Entity > tag is not used.

Writing format of “lsv_coreconfiguration.xml”

```
<LSVs_CoreConfiguration>
  <Data name="UnstructuredMesh">
    <Graphic name="sliceVector">
      <BuildFunction>Uns_sliceVector</BuildFunction>
      <Library>LSVg_builders.so</Library>
    </Graphic>
  </Data>
</LSVs_CoreConfiguration>
```

2.5. Utility tool

1. UnsIndexMaker.py

Role:

Creates index files for unstructured-grid data.

Command lines:

UnsIndexMaker.py --mesh mesh_pattern --data data_pattern
--div start,end,skip --stp start,end,skip
[--nd N][--ed E][--out file.idx]

Command line options:

- **--mesh data pattern**
  Specifies the file name pattern of the Mesh files.
  File naming pattern can include a domain number pattern such as "%05D."
- **--data data pattern**
  Specifies the file name pattern of the Data files.
  File naming pattern can include a domain number pattern such as "%05D" or a time step number pattern such as "%08T."
- **--div start,end,skip**
  Specifies the start and end division number as well as the number of steps to skip, with the values delimited by a comma (,).
  If the number of steps to skip is omitted, 1 is assumed.
- **--stp start,end,skip**
  Specifies the start and end time step numbers as well as the number of steps to skip, with the values delimited by a comma (,).
  If the number of steps to skip is omitted, 1 is assumed.
- **--nd N**
  Specifies the number of node data items.
  If omitted, 0 is assumed.
- **--ed E**
  Specifies the number of element data items.
  If omitted, 0 is assumed.
- **--out file.idx**
  Specifies the name of the index file to be created.
  If omitted, the file content is output to the standard output facility.

2.6. Appendix A

List of free software used

The free software used for achieving this work is as follows:

1. **libxml2**
   Version: 2.6
   URL: http://xmlsoft.org/

Licensor: MIT

Use: Parser library for XML document
The software is used in this work for parsing XML used in Index files.

LSV Client User's Guide

Ver. 0.5.0

ISLiM, RIKEN

http://www.csrp.riken.jp/index_e.html

October 2011

1st Edition	Jul.	2010
2nd Edition	Mar.	2011
3rd Edition	Jun.	2011
4th Edition	Oct.	2011

2-1, Hirosawa, Wako, 351-0198, Japan

Table of Contents

1. Introduction

The LSV is a parallel visualization program built on client-server architecture capable of handling large amount of data.

This user's guide describes how to use the LSV's client program (visualization client).

The visualization client serves as a user interface allowing users to operate a variety of functions of server programs over the network, and executed server processes deliver resultant graphical objects to user screens.

This document describes how to install and use the visualization client.

2. Installation

This chapter describes how to install the LSV.

2.1. System requirements

Recommended system requirements are as follows:

Table 2-1 System requirements for the visualization client

Architecture	AMD64, IA32
OS	Linux 2.6 MacOS 10.5/10.6
Software	OpenMPI 1.2/1.3 libxml2 2.6 libjpeg 6b freetype 2.1

2.2. Installing the binary package

The binary packages available for LSV are as follows:

- LSVeoe-0.5.n-centos4-amd64.tgz
- LSVeoe-0.5.n-centos5-amd64.tgz
- LSVeoe-0.5.n-macos10.5-i386.tgz

Follow the procedures below to install the program:

(1) Extracting files from a tar-ball

The LSV binary package can be stored and used at any directory of the user's choice. Move to the directory in which to install LSV and execute the following command:

```
tar xvfz LSVeoe-version-os-arch.tgz
```

A new directory named "lsv" appears in the directory chosen, and the files listed below are found in the new directory.

```
lsv
|– bin
|– etc
|– include
|– lib
|– logs
|– share
|    |– python
|    '– xml
|– LSV.app    (Only for MacOS)
|– tmpl
'– Setup.sh
```

(2) Executing “Setup.sh”

Move to the lsv directory just created, and execute the Setup.sh.

```
cd lsv
./Setup.sh
```

The following files should be created after the completion of the operation above:

- lsv/etc/lsv-proxy.cfg
  Configuration file for the visualization proxy program (lsv-proxy)
  (referred to when the visualization server is executed on the local host)
- lsv/etc/lsv_coreconfiguration.xml
  Configuration file for the visualization server plug-in modules
  (referred to when the visualization server is executed on the local host)
- lsv/share/xml/lsv_layer1_init.xml
  Network configuration file for the visualization client
- lsv/share/xml/lsv_layer2_init.xml
  Network configuration file for the visualization server
  (referred to when the visualization server is executed on the local host)
- $HOME/.lsvc.cfg
  Default configuration file for the visualization client
  Stores the default parameters for connecting to the visualization server.
- $HOME/.ssh/environment
  sshd environment variables file
  (referred to when the visualization server is executed on the local host)

(3) Setting “sshd_config” file (optional)

If the user wishes to execute the visualization server on the local host, it is necessary to set the sshd configuration file so that the user-specified environment variables are set when the user logs into the server via SSH.

Check to be sure that the following code line exists in /etc/ssh/sshdconfig.

```
PermitUserEnvironment yes
```

If the line does not exist, add the line to “/etc/ssh/sshdconfig” and send the HUP signal (kill -HUP sshd process ID) to the sshd process, or restart the system. Note that this operation requires system administrator privilege.

(4) Setting up the command search path (optional)

Set as follows so that the command search path includes the lsv/bin directory:

- When using BSH (sh, bash, etc.)
  Add below to $HOME/.profile or $HOME/.bashrc:
  export PATH=installation directory/lsv/bin:$PATH
- When using CSH (csh, tcsh, etc.)
  Add below to $HOME/.cshrc or $HOME/.tcshrc:
  setenv PATH install directory/lsv/bin:$PATH

3. Operation

This chapter describes how to operate the visualization client.

3.1. Starting up the program

The executable file for the visualization client is stored at the following directory:

(Installation directory)/lsv/bin/lsvc

Execute the file using the command line as follows:

lsvc [-h][-v][-cfg*config-file*][-ol][-tm] [-tmOut timeout]

-h	Displays the command line options.
-v	Displays the version details (0.5.n).
-cfg	Specifies the configuration file to be referred to. (If omitted, the program refers to $HOME/.lsvc.cfg.)
-ol	Outputs process logs to stderr. (If omitted, the logs are output to lsv/logs/lsvc.log)
-tm	Outputs the logs of time measurement results
-tmOut	Specifies the time before timeout (in seconds). (If omitted, 180 sec is set. Specifying “0” sets unlimited time.)

Note that if a value already set to the environment variable http_proxy, the program may not properly connect to the visualization server.

Try executing “echo $http_proxy” and if a value is output, execute the lsvc as follows:

env http_proxy= (installation directory)/lsv/bin/lsvc

Once the lsvc is executed, the main control window of the visualization client appears.

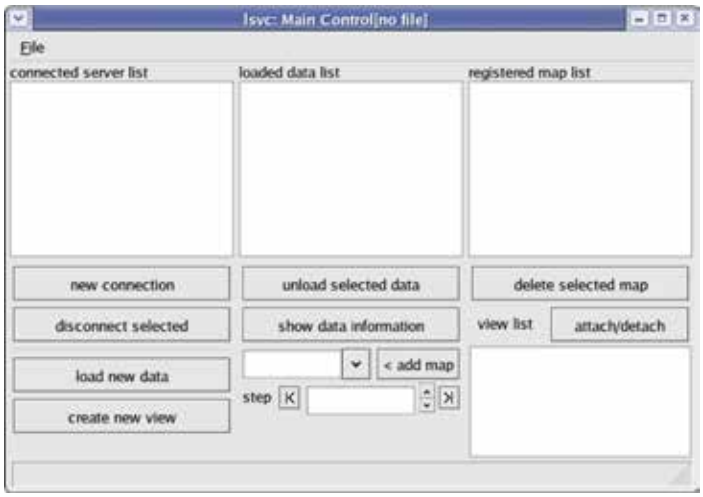


Fig. 3-1 Visualization client main control window

To quit the visualization client, either select Quit from the File menu on the main control window, or close the main control window itself. (For Mac OS, select "Quit Python" from the "Python" menu on the menu bar.)

3.2. Remote connection to the visualization server

To establish the remote connection between the visualization server and the client, the following two methods are available:

- WebService connection
- UserMode connection

The details on each connection method are described below:

(1) WebService connection

In this connection, the visualization client starts and connects to the visualization server via the WebService located on the visualization relay node.
The WebService consists of the resource management service, process boot service and communication relay service.
Note that before a WebService connection is established, the resource management service and process boot service must be started on the visualization relay node, so that the two services are ready to receive a client request.
When the HTTP communication through a user-designated port is unavailable between the visualization client host and the visualization relay node, their communication must be established using the SSH port forwarding function. Note here that in the case of the WebService connection, an SSH port forwarding process must be started beforehand.

¡ Specifying WebService URLs

Selecting "Set Services URL" from the "File" menu in the main control window displays the service URL setup dialog as shown below:

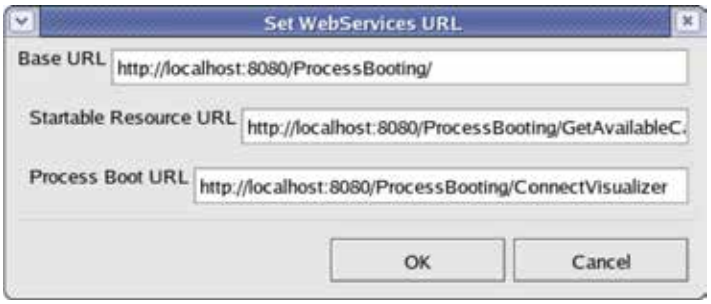


Fig. 3-2 Service URL setup dialog

Specify the “Startable Resource URL” and “Process Boot URL” in this dialog, and click OK. The URLs for both services will be registered. Entering a string in the “Base URL” column and pressing the Enter key will replace the head of the URLs with the entered string.
The registered URLs for both services are stored in the file $HOME/.lsvc.cfg as properties, and referred as the default URLs at the next boot.

¡ Starting and connecting to the visualization server

Clicking the "new connection" button in the main control window displays the connection dialog. Select the "WebService" tab at the top of the window.

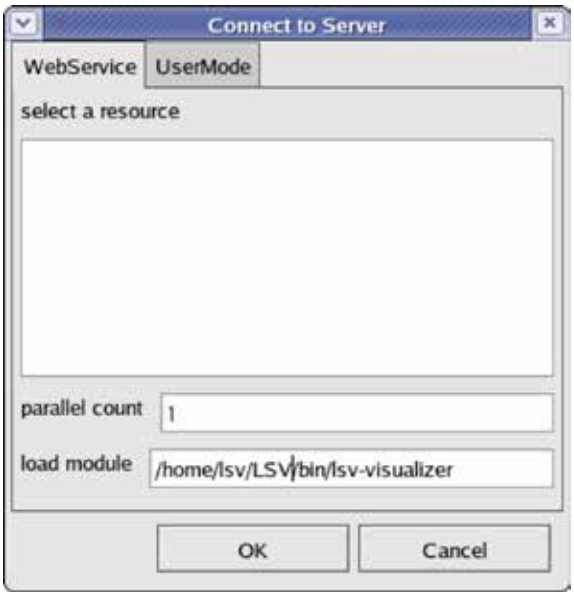


Fig. 3-3 Connection dialog WebService tab

This tab displays the "computer resource list" which was returned from the resource management service. Select from the list the desired computer resource on which to execute the visualization server.

Specify also the degree of parallelism for the visualization server process in "parallel count", and the path name for the visualization server program (lsv-visualizer) in "load module". Clicking OK will submit the request to the process boot service, and connect the client to the started visualization server process.

(2) UserMode connection

In this connection, the visualization client is connected to the visualization server via the visualization proxy program (lsv-proxy), instead of the WebService on the visualization relay node.

The visualization client first starts up the lsv-proxy on the visualization relay node, and then the lsv-proxy starts the visualization server and establishes the connection.

When the HTTP communication through a user-designated port is unavailable between the visualization client host and the visualization relay node, their communication must be established with the SSH port forwarding function. In the case of the UserMode connection, however, an SSH port forwarding process is automatically started.

¡ Starting and connecting to the visualization server

Clicking the "new connection" button in the main control window displays the connection dialog. Select the "UserMode" tab at the top of the window.

Fig. 3-4 Connection dialog UserMode tab

Set the following items in this dialog:

- proxy host
  The host name of the visualization relay node
- proxy user
  User name for connecting to the visualization relay node
- authentication type
  Authentication type to access the visualization relay node via SSH
  (password: password authentication, publickey: public key authentication)
- via ssh port-forwarding
  Check this when using the SSH port forwarding function.
- proxy command
  Command line for the visualization proxy program (lsv-proxy) on the visualization relay node
- parallel count
  Degree of parallelism for the visualization server process
- load module
  Path name for the visualization server program (lsv-visualizer)

Specifying the items above and clicking OK will start the lsv-proxy and visualization server, and connect the client to the started server process.

Once the visualization client successfully starts and connects to the visualization server, the "connected server list" in the main control window displays the name of the computer resource (for WebService connection) or the visualization relay node (for UserMode connection).

To terminate the visualization server selected in “connected server list” and shut down the connection, click "disconnect selected" button in the main control window.

3.3. Starting LSV via on-site connection

Use the lsv command to execute the LSV in the on-site mode. This command is an executable shell script. Execute the command from the command line as follows:

lsv [-h][-cli][-np *num-proc*][*script.py*]

-h	Displays “usage” messages
-cli	Starts LSV in CLI (command line) mode If this option is specified, the LSV does not provide GUIs.
-np *num-proc*	Executes the visualization server with parallelism of the *num-proc* value If this option is specified, the server is executed via mpirun command even when 1 is set to “*num-proc*”.
-tmOut	Specifies the time before timeout (in seconds). (If omitted, 180 sec is set. Specifying “0” sets unlimited time.)
script.py	Specifies the script file to be executed after startup For CLI mode, the LSV terminates after the script is executed.

3.4. Loading data

To load data to the visualization server, first select a server to upload the data from the "connected server list" in the main control window. Then click the "load new data" button in the window to display the file selection dialog, where the user can select the data file to be uploaded.

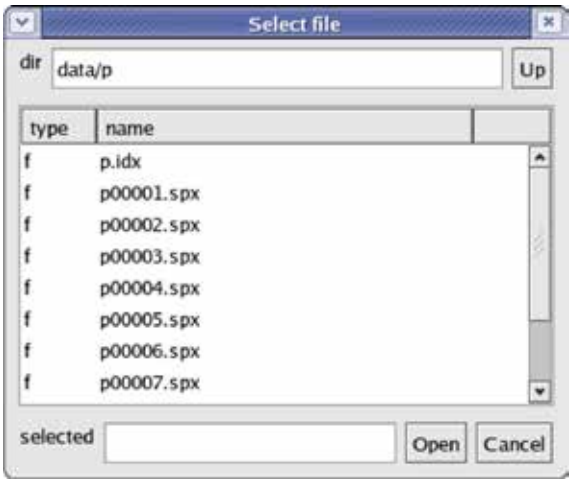


Fig. 3-5 File selection dialog for data upload

The dialog displays the list of the files located on the host computer, which operates the visualization server selected in the "connected server list".

The "dir" column on the upper part of the dialog indicates the directory currently displayed. Enter the desired directory in “dir” and press the Enter key to move to the directory. The default directory for this dialog can be set with the environmental variable LSVCDATADIR.

The alphabet at left in the list indicates the file type (d: directory, f: file). Select (by double-clicking) a directory to move to the directory.

Selecting a file in the list (by double-clicking, or selecting the file and clicking “Open”) will request the visualization server to load the selected file.

The file formats that can be uploaded are as follows:

File format	Extension
Index file	.idx
SPX	.spx
SPH	.sph
STL/Ascii	.stl, .sla, .stla
STL/Binary	.stl, .slb, .stlb
Wavefront OBJ	.obj
PARTICLE	.par

Once the file is loaded, the base name of the file data is displayed in the "loaded data list" in the main control window.

To unload the data file, select the data from the "loaded data list" and click the "unload selected data" button in the main control window.

3.5. Registering visualization maps

The functional unit for the visualization is referred to as a “map” in the LSV system. Registering a map to a data file applies visualization to the data file.

To register a map, select the target data from the "loaded data list" in the visualization client main control window, and click the "<add map" button in the pull-down menu.

Note that available maps vary according to each data type, as follows:

¡ Regular orthogonal grid data (SPX, SPH and Index files)

boundsLine	Displays data area boundaries
graphPlot	Plots a graph of physical quantities sampled on lines
histogram	Plots a histogram of physical quantity
isosurf	Displays isosurfaces
minmaxPoints	Displays the minimum and maximum value points
orthoContour	Displays contour lines on orthogonal slice
orthoGrid	Displays grid lines on orthogonal slice
orthoScalar	Displays color fringes on orthogonal slice
orthoVector	Displays vector arrows on orthogonal slice
probe	Samples physical quantities
staggeredVector	Displays staggered vector arrows on orthogonal slice
streamLines	Displays streamlines
timeStep	Displays time steps
timeStepSync	Synchronizes time steps
volume	Displays volumes

¡ Geometry data (STL and Wavefront OBJ)

triaShape	Displays geometries in polygons

¡ Non-structured grid data (Index files)

boundsLine	Displays the lines of the bounding box
graphPlot	Plots a graph of physical quantities sampled on lines
isosurf	Displays isosurfaces based on node data
probe	Samples physical quantities
shapeMap	Displays geometry surface
sliceContour	Displays contour lines on slice based on node data
sliceScalar	Displays color fringes on slice
sliceVector	Displays vectors on slice
timeStep	Displays time steps
timeStepSync	Synchronizes time steps

¡ Discrete data (PARTICLE and Index files)

plotLines	Displays lines
plotPoints	Displays vertex points
timeStep	Displays time steps
timeStepSync	Synchronizes time steps

For details on each map function, see Chapter 4.

Once the map is registered to the data, the map name is displayed in the "registered map list" in the main control window.

To delete the map, select the map from the "registered map list", and click the "delete selected map" button in the main control window.

3.6. Creating views and displaying maps

The LSV creates visualized images (i.e. rendered images) based on a functional unit called a “view”.

Once the view is created, the LSV creates a window called “view frame”, and there displays the image rendered with registered maps.

To create a view, first click the desired visualization server from the "connected server list". Then click the "create new view" button in the main control window.

When the view is created, an associated view frame window appears, and the name of the view is added to the "view list" in the main control window.

To display a map already registered to the data in the view, it is necessary to register the map to the view, as follows:

1) Select the desired map to be registered from the "registered map list" in the main control window.

The "view list" in the main control window displays the list of the views where the map can be registered. (These views were created on the same visualization server currently selected.)

The mark "*" before view name indicates that the selected map is already registered to the view.

2) Select the view to which to register the selected map from the "view list".

3) Click the "attach/detach" button to register the selected map to the view. (Alternatively, the user can double-click the desired view in the "view list".) To cancel the registration of a map already registered to a view, perform the similar operation to the view.

Once the map is registered, the view frame window displays the visualized image rendered with the map, and the map list in the window displays the name of the registered map. Selecting the name of the map displays the map parameter controls in the window.

The user can register a map to multiple views. In this case, the views with the same map display the same rendered images.

If the user closes the view frame window (by selecting "Close" from the "View" menu, or by clicking the "X" button on the window title bar), the view will be deleted, but not the map registered to the view, and only the registration to the view is cancelled.

If a map registered to a view is deleted (with "delete selected map" button in the main control window), the registration will automatically be cancelled.

3.7. GUI operation

(1) Main control window configuration

The main control window consists of the following components:

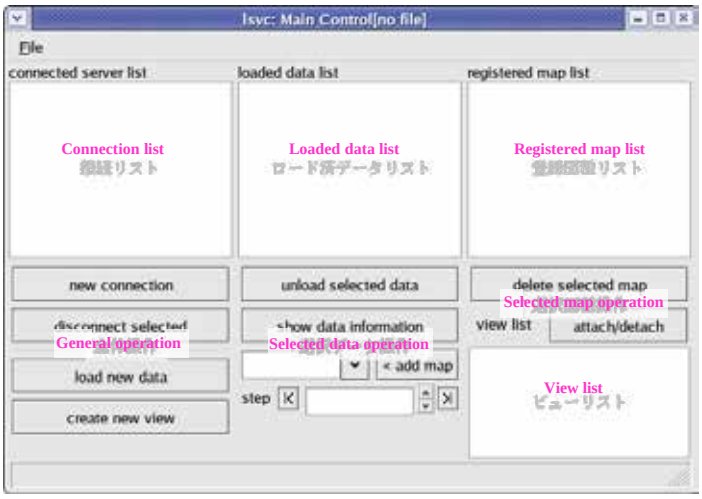


Fig. 3-6 Main control window configuration

The following describes the details on each component:

(a) Connected server list

Displays the list of the visualization servers currently connected.

(b) Loaded data list

Displays the list of the data loaded on the visualization server currently selected in the connected server list.

(c) Registered map list

Displays the list of the maps registered to the data currently selected in the loaded data list.

(d) General operation

This area displays the controls common to the visualization client operation.

- new connection
  Starts and connects to a new visualization server.
- disconnect selected
  Disconnects the client from the visualization server selected in the connected server list.
- load new data
  Loads new data onto the visualization server selected in the connected server list.
- create new view
  Creates a new view frame window on the visualization server selected in the connected server list.

(e) Selected data operation

This area displays the controls for operating the data currently selected in the loaded data list.

- unload selected data
  Unloads the selected data. All maps registered to this data will also be discarded.
- show data information
  Displays the data details.

- add map
  Registers the map selected from the list at left.
- step
  Specifies the time step to upload the data. The beginning of the time step is designated as 1.

(f) Selected map operation

This area displays the controls for operating the map selected in the registered map list.

- delete selected map
  Deletes the selected map.
- attach/detach
  Registers a map to the view selected in the view list, or cancels the registration.

(g) View list

Displays the list of the views on the visualization server currently selected in the connected server list.
An "*" is indicated before the name of the views where the current map is registered.

(2) Menu operation in the main control window

The menus in the main control window allow the following operation:

(a) File menu

- Open script
  Displays the file selection dialog, where the user can select a Python script file to be loaded and executed.
- Save script/Save script for Batch
  Saves the current visualization method as a Python script file.
  For details on the file saving procedure, see "(3) Saving and replaying script files".
- Set WebServices URL
  Displays the service URL setup dialog.
- Import configuration file
  Displays the file selection dialog, where the user can select a configuration file.
- Quit
  Terminates the visualization client program.

(3) Saving and replaying script files

The user can save a series of LSV visualization operations as a Python script file, and also load the saved script file to reproduce the visualization on LSV.

(a) Saving a general script

Select [File] -> [Save script] on the main control frame menu bar. In the displayed dialog, specify the name of the file (.py) in which to save the script. The operations that can be saved are: loading data, creating views, registering maps, display positions of each view, detailed settings on registered maps, and playing animations.

(b) Saving a script for movie files

When the timeStep map is registered, the user can save script files containing the command to create movie files.

Select [File] -> [Save script for Batch...] on the main control frame menu bar. In the displayed dialog, specify the name of the file (.py) in which to save the script. Clicking "Save" will display the following dialog:

Fig. 3-7 Confirm dialog for creating movies

Selecting "No" creates a script file not containing the commands to output movie files.
Selecting "Yes" displays the following dialog:

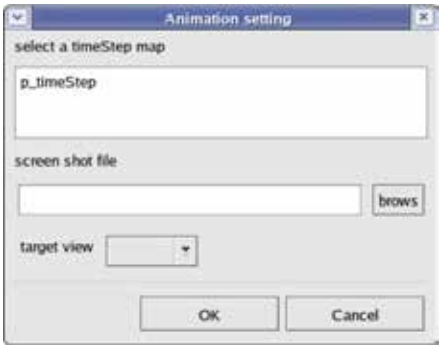


Fig. 3-8 Animation setting dialog

To create script files for outputting movie files, specify the following items:

- select a timeStep map
  Select a timeStep map driven on the script from the list of registered timeStep maps.
- screen shot file
  Specify the path to save the movie file.
  Directly enter the path in the textbox, or click the "browse" button to display the file dialog, where the user can select the file name and format.

  If a movie format (mpg, avi, mov) is set, a movie file is created.
  If an image format (png, jpg, bmp) is set, still-image files per frame are created.

  If "#*n*S" is included in the file name, the part will be replaced with the time step number when the files are output. Specify an integer for “*n*”, which will be treated as the number of zero-filled digits. If "#*n*S" is not included in the file name, the specified file will be overwritten.
  <Example>
  file.mpg (creates an mpeg file)
  file#3S.png (creates file001.png, file002.png, ...)
- target view
  Select the view from which to obtain movies.

Clicking "OK" outputs a script file containing the commands to output movie files.
Clicking "Cancel" cancels saving the file. No script file will be output.

(c) Saving a script for image files

Even when the timeStep map is not registered, the user can save script files to output visualization result as image files.

Select [File] -> [Save script for Batch...] on the main control frame menu bar. In the displayed dialog, specify the name of the file in which to save the images. Clicking "Save" will display the following dialog:

Fig. 3-9 Confirm dialog for saving image files

Selecting "No" creates a script file not containing the commands to output image files.
Selecting "Yes" displays the following dialog:

Fig. 3-10 Screenshot setting dialog

To create script files to output image files, specify the following items:

- screen shot file
  Specify the path to save the image file.
  Directly enter the path in the textbox, or click the "browse" button to display the file dialog, where the user can select the file name and format. The png, jpg, or bmp format can be specified.
- target view
  Select the view in which to capture images.

Clicking "OK" outputs a script file containing the commands to output image files.
Clicking "Cancel" cancels saving the file. No script file will be output.

<Note>
The Python script in the files saved via "Save script..." or "Save script for Batch..." assumes that the LSV client and server are already connected before the script execution. For the remote visualization, be sure to connect the client to the server before replaying the script.

(d) Replaying the script

The user can load an already-saved script file to reproduce the visualization on LSV, as follows.

First make sure that the client is connected to the server. Check to be sure that the "connected server list" in the main control window displays the servers where the script data can be loaded and executed.
Then, select [File] -> [Save script for Batch...] on the main control frame menu bar. In the displayed dialog, select the desired script file (.py) to be replayed.
The script file will then be loaded and automatically replayed.

(4) View frame window configuration

The configuration of the view frame window is as follows:

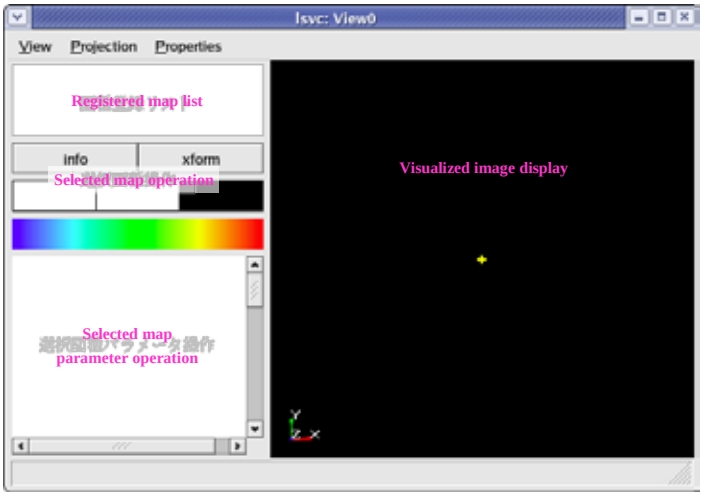


Fig. 3-11 View frame window configuration

The following describes the details on each component:

(a) Registered map list

Displays the list of the maps registered to the current view.

(b) Selected map operation

This area displays the controls for operating the map currently selected in the registered map list.

- info

  Displays the details on the selected map.

- xform

  Displays the geometric transformation dialog for the selected map.

Fig. 3-12 Geometric transformation dialog

  Entering a value for each column and pressing the Enter key will apply the geometric transformation to the map.

- Surface properties

  The selected map's color, opacity and highlight coefficient values are displayed in the three rectangles. Double-clicking this area displays the surface properties dialog.

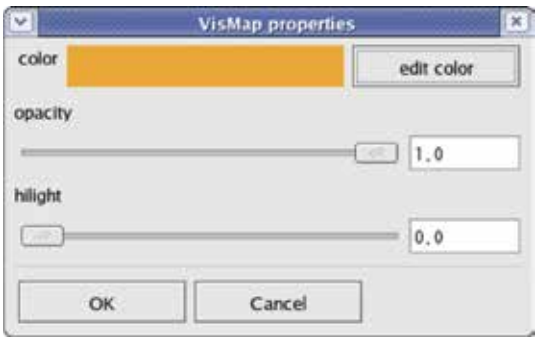


Fig. 3-13 Surface properties dialog

Changing the color, opacity and highlight values and clicking OK will apply the settings to the current map.
The color set here is used as the default color for the map.

- Color map display
  Displays the color map data for the selected map in the color bar. Double-clicking this area displays the color map dialog. For details on the dialog operation, see "(7) Defining color maps".

(c) Selected map parameter operation

This area displays the parameters and the controls for the map currently selected in the registered map list. The controls displayed vary according to the selected map. For details on each map's parameters, see Chapter 4.

(d) Visualized image display

This area displays the visualized image for the current view, as well as the mouse/keyboard controls for the images. For details on the image controls, see (6) Visualized image display controls

(5) Menu operation in the view frame window

The menus in the view frame window allow the following operation:

(a) View menu

- Show UI panel
  Specifies whether to display the view UI panel.
  The default is On.
- Normalize
  Normalizes the view. The program rescales and translates the registered map image to fit into the view.
- Set Center
  Defines the view center as the rotation and scaling center.
- BBox Manipulation
  Switches On/Off of the bounding box operation mode.
  When the mode is On, the bounding box (cube expressing the data area) is displayed as a wire frame during the mouse control of geometric transformation. The rendering request is then submitted to the server when the user completes the control.
  The default is On.
- Show BBox always
  Specify whether or not to always display the bounding box. The default is Off (Does not display the bounding box).

- Enable Mouse Wheel
  Switches the view zooming function On/Off with mouse wheel. The default is Off for the remote visualization server views, and On for the local visualization server views.
- Close
  Deletes the view and closes the view frame window.

(b) Projection menu

- Perspective
  Specifies whether to draw the object in perspective projection or horizontal projection. The default is On (Perspective projection).
- Near/Far Clip
  Displays the clipping plane distance dialog, where the user can set the near and far clipping plane distances from the view point

(c) Properties menu

- Show Center
  Specifies whether to display a yellow crosshair to indicate the rotation and scaling center. The default is On.
- Show Front Axis
  Specifies whether to display the coordinate axis at the left bottom corner of the view. The default is On.
- Background Color
  Displays the color selection dialog, where the user can select the background color.

(6) Operation for the visualized image display

In the visualized image display area, the user can operate the displayed images using the mouse and keyboard.

(a) Mouse operation

- Rotating a view
  Press and hold the Shift key, and slide the mouse to draw a circle while holding left mouse button down and still keeping the Shift key held down.
- Scrolling a view horizontally
  Press and hold the Shift key, and slide the mouse horizontally while holding left mouse button down and still keeping the Shift key held down.
- Zooming a view
  Press and hold the Ctrl or Apple keys, and move the mouse up and down while holding left button down.
- Scrolling a view horizontally
  Press and hold the Shift key, and slide the mouse horizontally while holding left mouse button down and still keeping the Shift key held down.
- Zooming with special effects
  Press and hold both Ctrl and Shift keys, and slide the mouse up and down while the left button is kept held down. (In the case of MacOS X, use Apple or Command key instead of Ctrl key.)

(b) Key operation

- Space key
  Normalizes the scene by rescaling and translating the registered map to fit into the view.
- Z and X keys
  Zooms in (Z) and out (X) of the scene.
- Arrow keys
  Rotates the scene by 90 degrees with respect to the screen per each press (180 degrees while the Shift key is held down, and 45 degrees while the Ctrl key (Apple key for MacOS X) is held down).
- C key
  Defines the view center as the rotation and scaling center.
- Home key
  Resets the scene rotation to the home position, where the view point views from +Z direction to –Z direction at right angle to the XY plane.

(7) Defining color maps

Double-clicking the color bar in the “selected map operation area” in the view frame window displays the color map dialog for the selected map, as shown below:

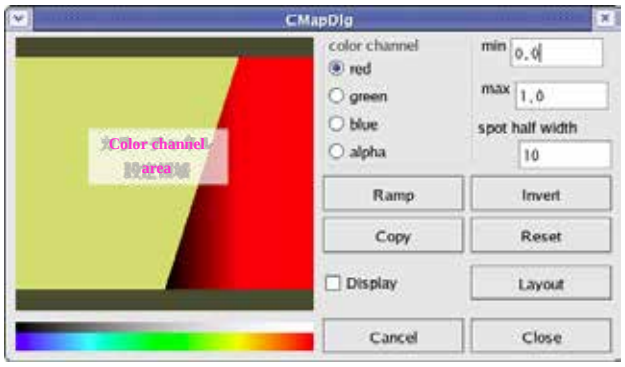


Fig. 3-14 Color map dialog

The color map is a table used for converting data values to colors during visualization processes. Each visualization map has its own color map.
The color map consists of an array of four color components of R (Red), G (Green), B (Blue), and A (Alpha: Opacity) (referred to as “color channels”), as well as the value ranges (the minimum and maximum values). The array size of each color component is usually 256. The value range is divided into 256 levels with a color assigned to each level.
The value range can be set in the "min" and "max" columns in the color map dialog. Note that these values may be automatically changed depending on the map type.

The user can set each color channel value in the color map dialog, as follows:

(a) Color channel settings

First select the channel to set from the "color channel" column.
Dragging the mouse within the "color channel area" then changes the values of the selected channel.
The figure below shows the color channel defined by dragging the mouse from A to B in the color channel area:

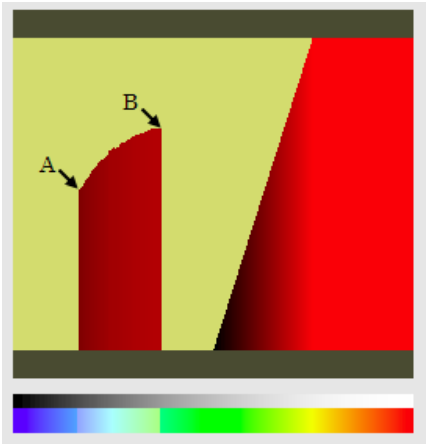


Fig. 3-15 Defining color channel

The user can also add a Gaussian distribution spot by clicking the right/middle mouse button while the mouse cursor is in the color channel area. The click position is set as the spot center. The spot width (half width) can be set in "spot half width".
The figure below shows the color channel defined by clicking on the position C in the color channel area:

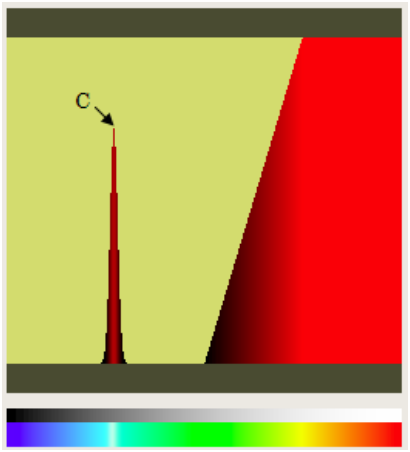


Fig. 3-16 Adding a spot to the color channel

Dragging the mouse on the dark colored areas at the top and bottom of the “color channel area” sets the value to 1.0 and 0.0, respectively.
Clicking the “Ramp" button sets the current channel so that the minimum and maximum values are connected in a linear fashion.
Clicking the “Invert" button horizontally flips the current channel values.
Clicking the “Reset" button resets all channel values to the standard color map values.
The "Copy" button does not function in the current version.
Clicking the “Cancel" button cancels all the changes made on the dialog, and returns the color map to the status before the dialog was opened.
Clicking the “Close" button closes the dialog with all changes applied.

(b) Color bar settings

In the color map dialog, the user can also set the color bar to be superimposed on visualized images. Checking the "Display" checkbox in the dialog will superimpose the current map's color bar on the visualized image in the view frame window.

Further, clicking the "Layout" button in the dialog will display the "color bar layout dialog".

Fig. 3-17 Color bar layout dialog

In this dialog, the user can set the following color bar parameters:

- direction
  Select the display direction of the color bar from "vertical" or "horizontal". The default is "vertical".
- show alpha
  Specify whether or not to display the Alpha (opacity) data in the color bar. The default is Off.
- Title label
  Set a title string for the color bar. Pressing the Enter key in the input field will update the title. The default value is a blank.
- #of labels
  Specify the number of numeric labels described on the color bar. A value larger than 3 cannot be specified when the direction is "horizontal". The default value is 2.
- #of decimals
  Specify the number of figures after the decimal point indicated on the color bar numeric label. The default value is 2.
- Position
  Specify the color bar position by entering X and Y coordinates. The default value is (0.7, 0.0).
- Size
  Set the sizes of the color bar in X and Y directions. When the bar direction is switched, these X and Y values will also be switched. The default values are (0.2, 1.0) and (1.0, 0.2) when the direction setting is "vertical" and "horizontal", respectively.
  Note that the values entered in "Position" and "Size" are based on the screen coordinate system according to the visualized image area in the view frame window. Therefore, when the view frame window size is changed, the color bar size will also be changed accordingly.
  The user cannot define the display order of multiple color bars when they overlap each other. When displaying multiple color bars, adjust the Position and Size values so that they do not overlap.

4. Map parameter settings

4.1. boundsLine

The boundsLine displays the line representing the data area, and also displays the boundaries of each process on the visualization server.

Selecting the boundsLine in the view frame will display the following parameters in the selected map parameter area:

(1) line width

Specify the line width. The value must be a real number larger than 0.0. The default value is 1.0.

(2) show each block

When visualization servers are in a parallel execution, specify whether or not to display boundaries for each server process. The default is Off (Does not display).

4.2. graphPlot

When the graphPlot is selected in the view frame, the selected map parameter area draws a graph by sampling the data on the line specified below.

The graph is displayed in the graphPlot dialog, with the lines and pointers representing the data sampling points. The lines and pointers are displayed in the base color.

The sampling points are obtained by geometrically transforming the default line. The default line is a line parallel to the bounding box X-axis, having the same length as that of the X-axis, and passing through the box origin. If any vector data is registered, the data is sampled from the selected component or vector norm.

Selecting the graphPlot in the view frame will display the following parameters in the selected map parameter area:

(1) select data

Select the target data of the graphPlot.

For regular orthogonal grid data, available data are as follows:

The default value is “data0".

data0 (0th data)

...

dataN (Nth data)

veclen (Absolute vector length. Only for vector data)

For unstructured grid data, available data are as follows:

The default value is “dataN0" or "dataE0".

For node data

dataN0 (0th data)

...

dataNn (Nth data)

veclenN (Absolute vector length. Only for vector data)

For element data

dataE0 (0th data)

…

dataEn (Nth data)

veclenE (Absolute vector length. Only for vector data)

(2) show graph window

Specify whether or not to display the graphPlot dialog.
The default is On (Displays the dialog).
The details on the graphPlot dialog will be explained separately.

(3) edit sampler

Specify whether or not to display the dialog for controlling the sampler position. The default is Off (Does not display the dialog). The details on the sampler control dialog will be explained separately.

(4) show sampler

Specify whether or not to display the sampler. The default is On (Displays the sampler).

(5) # of points

Specify the number of sampling points. Enter an integer larger than zero. The default value is 10.

(6) line width

Specify the width of the sampling line in pixels. Enter a real number larger than 0.0. The default value is 1.0.

(7) point size

Specify the size of the sampling point in pixels. Enter a real number larger than 0.0. The default value is 2.0.

Set the position of the sampler in the following dialog:

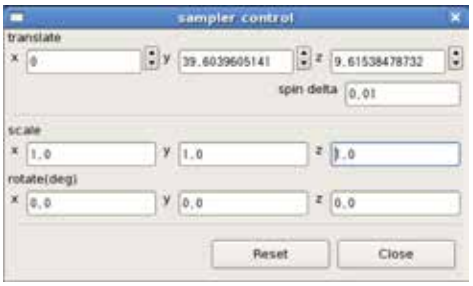


Fig. 4-2-1 Sampler control dialog

- translate (x, y, z)
  Specify the translation amount of the sampling point in relation to the data area origin, using real numbers. When (0.0, 0.0, 0.0) is set, the data bounding box origin is regarded as the sampling point origin.
- spin delta
  Specify the increment of the “translate” spin buttons with real numbers.
- scale (x, y, z)
  Specify the magnification ratio of the sampling point in relation to the initial state, using real numbers. The default value is (1.0, 1.0, 1.0), where the length of the sampling point line is equal to that of the data bounding box in X-direction. Note that the Y and Z components are ignored.
- rotate (x, y, z)
  Specify in degrees the rotation amount of the sampling point with respect to the initial state. The default value is (0.0, 0.0, 0.0), where the sampling point line coincides with the X-axis.
- reset
  Resets the other items to the default.
- close
  Closes the dialog.

Graphs are displayed in the dialog as shown in the figure below:

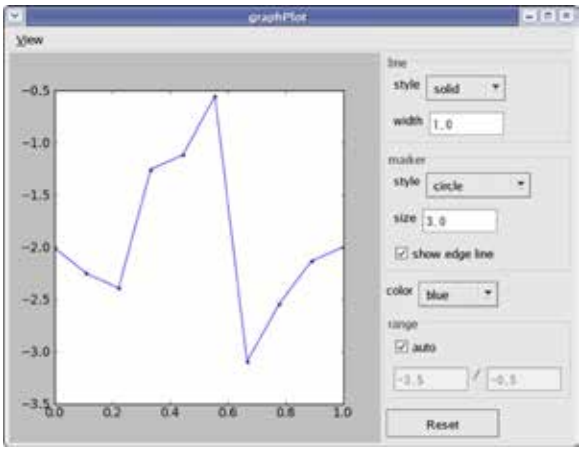


Fig. 4-2-2 graphPlot dialog

- line style
  Specify the line style from the following. The default value is “solid”.
  - solid
  - dashed
  - dash-dot
  - dotted
  - (None)
- line width
  Specify the line width. Enter a real number larger than 0.0. The default value is 1.0.
- marker
  Select the marker type from below. The default value is “circle”.
  - circle
  - point
  - pixel
  - plus
  - x
  - diamond
  - thin-diamond
  - triangle-down
  - triangle-up
  - triangle-left
  - triangle-right
  - square
  - pentagon
  - hexagon1
  - hexagon2
  - tri-down (⋁)
  - tri-up (⋀)
  - tri-left (<)
  - tri-right (>)
  - hline (|)
  - vline (−)
  - (None) (No marker)
- marker size
  Set the size of the marker. Enter a real number larger than 0.0. The default value is 3.0.

- color
  Specify the bar color from the following. The default value is “blue”.
  - blue
  - green
  - red
  - cyan
  - magenta
  - yellow
  - black
  - white
- show marker edge line
  Specify whether or not to display the marker edge line. The default is On (Displays the line).
- reset
  Resets the other items to the default.

4.3. histogram

This map creates histograms from the results of counting physical quantities at all grid points for each class. The histograms are displayed in the dialog, and nothing is displayed as a map. If any vector data is registered, physical quantities are counted from the selected component or vector norm.
Selecting the histogram map in the view frame will display the following parameters in the selected map parameter area:

(1) select data

Select either of the following data as the histogram target:
The default value is “data0”.

data0	(0th data)
…	
dataN	(Nth data)
veclen	(Absolute vector length. Only for vector data)

(2) show graph window

Specify whether or not to display the histogram dialog. The default is On (Displays the dialog).
The details on the histogram dialog will be explained separately.

(3) # of classes

Specify the number of classes for histograms. Enter an integer larger than 0. The default value is 100.

Histograms are displayed in the dialog as shown in the figure below:

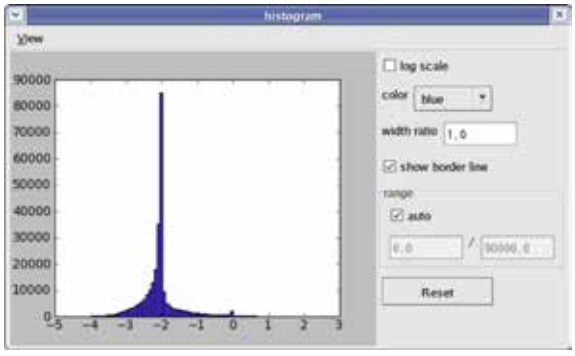


Fig. 4-3-1 histogram dialog

- log scale
  Specify whether or not to display histograms in log scale. The default is Off (Does not display in log scale).
- color
  Select the bar color from the following. The default value is “blue”.
  blue
  green
  red
  cyan
  magenta
  yellow
  black
  white
- width ratio
  Set the ratio of the bar length. Enter a real number from 0.0 to 1.0. The default value is 1.0.
- show border line
  Specify whether or not to display bar borderlines. The default is On (Displays the borderlines).
- reset
  Resets the other items to the default.

4.4. isosurf

The isosurf map displays isosurfaces with marching cubes.
The base color or color map setting is applied to the surface color and its opacity.
If any vector data is registered, the isosurface of the selected component or vector norm is displayed.
Selecting the isosurf in the view frame will display the following parameters in the selected map parameter area:

(1) value

Specify the data values to be displayed in real numbers.
Nothing will be displayed when the specified value is out of the minimum and maximum range.

(2) select data

Select the target data to display isosurfaces.
The default value is “data0".

data0	(0th data)
...	
dataN	(Nth data)
veclen	(Absolute vector length. Only for vector data)
None	(Nothing is displayed.)

(3) use cmap

Specify whether or not to use the color map for isosurface colors. If Off (no) is set, the base color is applied. The default is On (yes).

(4) use cmap alpha

Specify whether or not to use the color map for isosurface opacity. If Off (no) is set, the base color component A is applied. The default is Off.

(5) update minmax

Specify whether or not to update the minimum and maximum values. The default is On (yes).

(6) show two side

Specify whether or not to draw images on both sides. The default is On (yes).

4.5. minMaxPoints

The minMaxPoints displays points to the grid points where the maximum or minimum value is obtained.

The maximum value position is displayed in the maximum value color in the color map (the default is red), and the minimum value position is displayed in the minimum value color (the default is blue).

If any vector data is registered, the points for the selected component or vector norm are displayed.

Selecting the minMaxPoints in the view frame will display the following parameters in the selected map parameter area:

(1) select data

Select the target data from the following:

The default value is “data0".

data0	(0th data)
...	
dataN	(Nth data)
veclen	(Absolute vector length. Only for vector data)
None	(Nothing is displayed.)

(2) tolerance

Specify how far a value can be off from the maximum or minimum value to display a point in percentage to (max. value-min. value). The default value is 0.1.

(3) point size

Specify the point size in pixels.

Enter a real number larger than 0.0. The default value is 3.0.

(4) show min point

Specify whether or not to plot the minimum point. The default is On (yes).

(5) show max point

Specify whether or not to plot the maximum point. The default is On (yes).

(6) use cmap

Specify whether or not to use the color map for point colors. If Off (no) is set, the base color is applied. The default is On (yes).

4.6. orthoContour

The orthoContour displays contour lines on a grid slice. Contour lines are displayed in the base color, or the color referred from the color map. If any vector data is registered, contours for the selected component or vector norm are displayed.

Selecting the orthoContour in the view frame will display the following parameters in the selected map parameter area:

(1) slice axis

Select the slicing direction of the slice from below:

I (I axis (JK surface))
J (J axis (KI surface))
K (K axis (IJ surface))

(2) slice plane

Specify the slicing position of the slice.
Select an integer from 0 to (the number of grids in the slicing direction-1). The default value is (number of grids in the slicing direction/2).

(3) select data

Select the target data from the following:
The default value is “data0".

data0 (0th data)
...
dataN (Nth data)
veclen (Absolute vector length. Only for vector data)
None (Nothing is displayed.)

(4) # of lines

Set the number of contour lines to be displayed. Enter an integer larger than 0. The default value is 5.

(5) range

Specify the maximum and minimum values for drawing contour lines. The default values are the minimum and maximum values of the data. A minimum value larger than the maximum value cannot be specified, and vice versa.

(6) use cmap

Specify whether or not to use the color map for contour colors. If Off (no) is set, the base color is applied. The default is On (yes).

(7) update minmax

Specify whether or not to update the minimum and maximum values. The default is On (yes).

(8) line width

Specify the line width.
Enter a real number larger than 0.0. The default value is 1.0.

4.7. orthoGrid

The orthoGrid displays grid lines on a grid slice.
Grid lines are displayed in the base color.
Selecting the orthoGrid in the view frame will display the following parameters in the selected map parameter area:

(1) slice axis
Select the slicing direction of the slice from below:
I (I axis (JK surface))
J (J axis (KI surface))
K (K axis (IJ surface))

(2) slice plane
Specify the slicing position of the slice.
Select an integer from 0 to (the number of grids in the slicing direction-1). The default value is (number of grids in the slicing direction/2).

(3) line width
Specify the line width.
Enter a real number larger than 0.0. The default value is 1.0.

4.8. orthoScalar

The orthoScalar displays color fringes on a grid slice.
Fringes are displayed in the base color, or the color referred from the color map.
If any vector data is registered, color fringes are displayed for the selected component or vector norm.
Selecting the orthoScalar in the view frame will display the following parameters in the selected map parameter area:

(1) slice axis
Select the slicing direction of the slice from below:
I (I axis (JK surface))
J (J axis (KI surface))
K (K axis (IJ surface))

(2) slice plane
Specify the slicing position of the slice.
Select an integer from 0 to (the number of grids in the slicing direction-1). The default value is (number of grids in the slicing direction/2).

(3) select data
Select the target data from the following:
The default value is “data0".
data0 (0th data)
...
dataN (Nth data)
veclen (Absolute vector length. Only for vector data)
None (Nothing is displayed.)

(4) update minmax

Specify whether or not to update the minimum and maximum values. The default is On (yes).

4.9. orthoVector

The orthoVector displays vector arrows on a grid slice. The arrows are displayed in the base color, or the color referred from the color map. If scalar data is registered, the arrows will not be displayed.

Selecting the orthoVector in the view frame will display the following parameters in the selected map parameter area:

(1) slice axis

Select the slicing direction of the slice from below:

I (I axis (JK surface))
J (J axis (KI surface))
K (K axis (IJ surface))

(2) slice plane

Specify the slicing position of the slice.
Select an integer from 0 to (the number of grids in the slicing direction-1). The default value is (number of grids in the slicing direction/2).

(3) vector scale

Specify the factor in real numbers to multiply to the actual vector value to obtain the arrow length. The default value is 1.0.

(4) line width

Specify the line width.
Enter a real number larger than 0.0. The default value is 1.0.

(5) arrow head

Specify whether or not to display arrowheads. The default is On (yes).

(6) use cmap

Specify whether or not to use the color map for arrow colors. If Off (no) is set, the base color is applied. The default is On (yes).

(7) select data

Select the target data from the following:
The default value is “data0".

data0 (0th data)
...
dataN (Nth data)
veclen (Absolute vector length. Only for vector data)

(8) update minmax

Specify whether or not to update the min/max values. The default is On (yes).

4.10. plotLines

The plotLines map interprets discrete data as line data and displays the lines.

Selecting the plotLines in the view frame will display the following parameters in the selected map parameter area.

(1) select data

Select the data to display as line color.

The default value is “data0”.

data0 (0th data)

...

dataN (Nth data)

veclen (Absolute vector length. Only for vector data)

None

(2) use cmap

Specify whether or not to use the color map for line colors. If Off (no) is set, the base color is applied. The default is On (yes).

(3) update minmax

Specify whether or not to update the minimum and maximum values. The default is On (yes).

(4) line width

Specify the line width in pixels. The default value is 1.0. Enter a value larger than 0.0.

4.11. plotPoints

The plotPoints map interprets discrete data as line data to display the vertexes.

Selecting the plotPoints in the view frame will display the following parameters in the selected map parameter area.

(1) select data

Select the data to display as point color.

The default value is “data0”.

data0 (0th data)

...

dataN (Nth data)

veclen (Absolute vector length. Only for vector data)

None

(2) use cmap

Specify whether or not to use the color map for point colors. If Off (no) is set, the base color is applied. The default is On (yes).

(3) update minmax

Specify whether or not to update the minimum and maximum values. The default is On (yes).

(4) point size

Specify the point size in pixels. The default value is 1.0. Enter a value larger than 0.0.

4.12. probe

The probe map samples data at the specified points and reports the result.

The symbols indicating the sampling points are displayed as a map. The symbols are displayed in the base color.

Define the sampling point by specifying translation amount from the data bounding box origin, in “position”.

Selecting the probe map in the view frame will display the following parameters in the selected map parameter area:

(1) probe control

Specify whether or not to display the probe control dialog. The default is On (Displays the dialog).

(2) Result display column

Displays the sampling results.

The sampling points can be defined in the dialog as shown below:

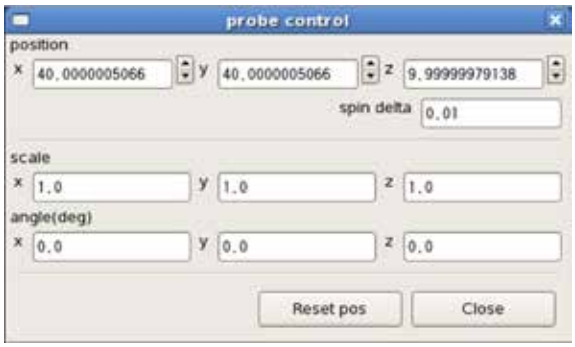


Fig. 4-10-1 probe control dialog

- position (x, y, z)
  Specify the translation amount of the sampling point in relation to the initial state, using real numbers. When (0.0, 0.0, 0.0) is set, the data bounding box origin is regarded as the sampling point origin.
- spin delta
  Specify the increment of the “position” spin buttons with real numbers.
- scale (x, y, z)
  Specify the magnification ratio of the probe symbol in relation to the initial state, using real numbers. The default value is (1.0, 1.0, 1.0), where the size of the probe symbol is set to 1.0 in length and 0.1 in width.
- angle(deg) (x, y, z)
  Specify in degrees the rotation amount of the probe symbol with respect to the initial state. The default value is (0.0, 0.0, 0.0), where the longitudinal direction of the probe symbol coincides with the Y-axis.
- Reset pos
  Resets the position values to the default.
- close
  Closes the dialog.

4.13. shapeMap

The shapeMap displays geometry surfaces based on unstructured grid data.

Selecting the shapeMap in the view frame displays the following parameters in the selected parameter control area:

(1) select data

Select the data to display as geometry surface color. If “none” is specified, the base color is used. The default value is “none”.

For node data

dataN0	(0th data)
...	
dataNn	(Nth data)
veclenN	(Absolute vector length. Only for vector data)
none	(base color)

For element data

dataE0	(0th data)
…	
dataEn	(Nth data)
veclenE	(Absolute vector length. Only for vector data)
none	(base color)

(2) update minmax

Specify whether or not to update the minimum and maximum values. The default is On (yes).

(3) show two side

Specify whether or not to display both sides (front and back) of geometry surface. If “Off” is set, the backside is not displayed. The default is Off (no).

(4) elem shrink

Specify the reducing ratio of the elements constructing geometry surface on each face. If 1.0 is set, the element size is not reduced. The default value is 1.0.

(5) Offset disp

When node data is provided as vector data, specify a coefficient to transform the geometry by regarding the data value as a displacement. If 0.0 is set, the geometry is not transformed. The default value is 0.0.

4.14. sliceContour

The sliceContour displays contour lines on an arbitrary slice of the unstructured grid node data. Selecting the sliceContour in the view frame will display the following parameters in the selected map parameter area:

(1) select data

Specify the node data to display contour lines.
The default value is “dataN0”.

dataN0 (0th data)
…
dataNn (nth data)
veclenN (Absolute vector length. Only for vector data)

(2) data range

Specify the value range to display contours by separating with “;”. The default is the value range of the specified node data.

(3) num lines

Specify the number of contours to be displayed. The default value is 5.

(4) use cmap

Specify whether or not to refer to the color map for contours. If “Off (no)” is set, the base color is applied. The default is On (yes).

(5) line width

Specify the contour width in pixels. The default value is 1.0.

(6) center pos

Specify the coordinates of the virtual center point of the slice by separating the values with space characters. The point set here is regarded as the rotation center in the following item “rotate”.
The default value is “the data bounding box center”.

(7) rotate

Specify the rotation amount (deg) of the slice around each axis by separating the values with space characters. The slice plane is rotated from the initial state (XY plane) as specified below, with the previous item “center pos” as its center:

[M]=[Ry][Rx][Rz]; where [Rx]; [Ry]; [Rz] denote the rotations around X, Y, and Z axis, respectively. The default value is (0.0, 0.0, 0.0).

4.15. sliceScalar

The sliceScalar displays color fringe on an arbitrary slice of unstructured grid data.

Selecting the sliceScalar in the view frame will display the following parameters in the selected map parameter area:

(1) slice control

Displays the slice plane dialog, where the user can set the center point coordinates and rotation amount (deg) of the slice plane.

The details on the slice plane dialog are as follows:

- position
  Specify the coordinates of the virtual center point of the slice. The point specified here is regarded as the rotation center of “angle”. The default value is “the data bounding box center”.
- spin delta
  Specify the increment of the “position” spin buttons with real numbers.
- scale
  This item is not used.
- angle(deg)
  Specify the rotation amount (deg) of the slice around each axis.
  The slice plane is rotated from the initial state (XY plane) as specified below, with the previous item “position” as its center:
  [M]=[Ry][Rx][Rz]; where [Rx]; [Ry]; [Rz] denote the rotations around X, Y, and Z axis, respectively. The default value is (0.0, 0.0, 0.0).
- Reset pos
  Resets the “position” settings
- close
  Closes the dialog.

(2) select data

Select the target data to display color fringe.

The default value is “dataN0” or “dataE0”.

For node data

dataN0	(0th data)
…	
dataNn	(Nth data)
veclenN	(Absolute vector length. Only for vector data)
None	(base color)

For element data

dataE0	(0th data)
…	
dataEn	(Nth data)
veclenE	(Absolute vector length. Only for vector data)
None	(base color)

(3) update minmax

Specify whether or not to update the minimum/maximum values. The default is On (yes).

4.16. sliceVector

The sliceVector displays vectors on an arbitrary slice of non-structure grid data.
Selecting the sliceVector will display the following parameters in the selected map parameter area:

(1) select data

Specify the target data to display vector colors.
The default value is “dataN0” or “dataE0”.

For node data

dataN0	(0th data)
…	
dataNn	(Nth data)
veclenN	(Absolute vector length. Only for vector data)
None	(base color)

For element data

dataE0	(the 0th data)
…	
dataEn	(Nth data)
veclenE	(Absolute vector length. Only for vector data)

(2) vec scale

Specify the scaling ratio for displaying vectors. The default value is 1.0.

(3) vec head

Specify whether or not to display an arrow at the vector end. If Off (no) is set, vectors are displayed as a line. The default is On (yes).

(4) use cmap

Specify whether or not to use the color map for displaying vectors. If Off (no) is set, the base color is applied. The default is On (yes).

(5) line width

Specify the width of vector lines in pixels. The default value is 1.0.

(6) center pos

Specify the coordinates of the virtual center point of the slice by separating the values with space characters. The point set here is regarded as the rotation center in the following item “rotate”.
The default value is “the data bounding box center”.

(7) rotate

Specify the rotation amount (deg) of the slice around each axis by separating the values with space characters. The slice plane is rotated from the initial state (XY plane) as specified below, with the previous item “center pos” as its center:

[M]=[Ry][Rx][Rz]; where [Rx]; [Ry]; [Rz] denote the rotations around X, Y, and Z axis, respectively. The default value is (0.0, 0.0, 0.0).

(8) xdr

Specify the data range to be displayed by separating with space characters.

4.17. staggeredVector

The staggeredVector displays vector arrows assuming staggered grids on grid slice.

Assuming the data grid point to be staggered grid cell center, this map displays an arrow indicating velocity component at the velocity definition point.

Nothing will be displayed if scalar data is registered.

Selecting the staggeredVector in the view frame will display the following parameters in the selected map parameter area.

(1) slice axis

Select the slicing direction of the slice from below:

I (I axis (JK surface))
J (J axis (KI surface))
K (K axis (IJ surface))

(2) slice plane

Specify the slicing position of the slice.

Select an integer from 0 to (the number of grids in the slicing direction-1). The default value is (number of grids in the slicing direction/2).

(3) vector layout

Specify the type of the velocity definition point. Select from the following options:

regular	(regular grid: assuming that all data is defined on the original grid points)
collocated	(collocate grid: assuming that all data is defined at the centroid of grid cell)
staggered1	(staggered grid: assuming that the velocity component is defined at the negative side of the cell wall)
staggered2	(staggered grid: assuming that the velocity component was defined at the plus side of the cell wall)

(4) vector scale

Specify the factor in real numbers to multiply to the actual vector value to obtain the arrow length. The default value is 1.0.

(5) line width

Specify the line width.

Enter a real number larger than 0.0. The default value is 1.0.

(6) arrow head

Specify whether or not to display arrowheads. The default is On (yes).

(7) use cmap

Specify whether or not to refer to the color map. If Off (no) is set, the base color is applied. The default is On (yes).

(8) select data for color

Select the target data from the following:

The default value is “data0".

data0	(0th data)
...	
dataN	(Nth data)
veclen	(Absolute vector length. Only for vector data)

(9) update minmax

Specify whether or not to update the min/max values. The default is On (yes).

(10) show X component

Specify whether or not to display arrows for the X component of velocity. The default is On (yes).

(11) show Y component

Specify whether or not to display arrows for the Y component of velocity. The default is On (yes).

(12) show Z component

Specify whether or not to display arrows for the Z component of velocity. The default is On (yes).

4.18. streamLines

The streamLines map displays streamlines from the start point specified on lines.

Selecting the streamlines in the view frame will display the following parameters in the selected map parameter area:

(1) edit sampler

Select whether to specify the start point. If On (yes) is set, the start point dialog is displayed. The default is Off (no).

The details on the start point dialog will be explained separately.

(2) export

Exports the start point parameter as a file. The file dialog for the settings will be displayed.

(3) show sampler

Specify whether or not to display a line and point representing the start point position. If Off (no) is set, they are not displayed. The default is On (yes).

(4) # of points

Specify the number of start points. The default value is 1.0. Enter a value larger than 0.

(5) sampler line width

Specify the width of the line indicating the start point position in pixels.

The default value is 1.0. Enter a value larger than 0.

(6) sampler point size

Specify the size of the point indicating the start point position in pixels.

The default value is 2.0. Enter a value larger than 0.

(7) div time

Specify the number of time step divisions (i.e. the number of integral steps in relation to virtual time step).

The time necessary for the maximum velocity to move across the width of one grid is assumed to be a virtual time step. The virtual time step is then divided by the “num divs” value, and the velocity field is integrated assuming the obtained value as integral time.

The default value is 1.0. Enter a value larger than 0.

(8) max points

Specify the maximum number of points for a streamline.
The default value is 1000. Enter a value larger than 0.

(9) colored line

Specify whether or not to use separate colors for streamlines. If On (yes) is set, each streamline is displayed in separate color by referring to the color map. If Off (no) is set, all streamlines are displayed in the base color. The default is Off (no).

(10) line width

Specify the width of streamlines in pixels.
The default value is 1.0. Enter a value larger than 0.

(11) reverse vector

Specify whether or not to perform the reverse calculation for streamlines. The default is Off (no).

The following describes the details on the start point dialog. The available controls are as follows:

- translate
  Specify the translation amount of the start point in relation to the initial position for x, y, and z attributes. When (0.0, 0.0, 0.0) is set, the DATA bounding box origin is regarded as the start point.
- scale
  Specify the zoom ratio of the start point in relation to the initial state for x, y, and z attributes.
  When (1.0, 1.0, 1.0) is set, the length of the line at the start point is equal to that of the DATA bounding box in X-direction.
  Note that the Y and Z components are ignored.
- rotate
  Specify the rotation amount of the start point with respect to the initial state for x, y, and z attributes. Each component represents the rotation amount around X, Y, and Z-axis. When (0.0, 0.0, 0.0) is set, the start point line coincides with the X-axis.
- reset
  Resets the other items to the default.
- close
  Closes the dialog.

4.19. timeStep

The timeStep displays a time step animation of time series data, creates movies, and displays a text string with timeStep numbers on the visualized image in the registered view.

Selecting the timeStep in the view frame will display the following parameters in the selected map parameter area:

(1) Control buttons

Controls the time steps of the displayed time series data.

- |◃ Returns to the first time step.
- □ Stops the time step animation.
- ▹ Starts the time step animation.
- ▹| Moves to the last time step.

(2) loop

Specify whether or not to loop the time step animation.
If Off (no) is set, the animation stops after displaying the last frame. If On (yes) is set, the animation jumps from the last frame to the first frame, and repeats endlessly. The default is Off (no).

(3) show text label

Specify whether or not to show a text label in the view. If On (yes) is set, a text label with the time step number is displayed. The default is Off (no).

The user can set the following label parameters:

- format
  Specify the label format. The string “#S” is replaced with the time step number in the view. The default is “step=#S”.
- pos X
  Specify the position of the label in the X axis direction. Assuming the horizontal size of the view to be 100%, specify the display start point using the percentage value. The default value is 60.0.
- pos Y
  Specifies the position of the label in the Y axis direction. Assuming the vertical size of the view to be 100%, specify the display start point using the percentage value. The default value is 90.0.
- size
  Specify the font size. Assuming the vertical size of the view to be 100%, specify the font size using the percentage value. The default value is 5.0.

(4) screen shot

Specify whether or not to save images.
If On (yes) is set, screen shots or movies are saved when the replay button is pressed for the time step animation.
The default is Off (no).

If the animation is started from the time step in the middle, the movie from the time step to the last time step will be saved. If the animation is stopped in the middle, the movie until the time step will be saved. If the loop setting is On, only the first loop will be saved.
To save images, it is also necessary to set the following items:

- file
  Specify the name of the file in which to save images.
  Clicking the "browse" button will display the file dialog, where the user can specify the file name and format. Alternatively, enter the desired file name directly in the textbox and press the Enter key.

  If a movie format (mpg, avi, mov) is set, a movie file is created.
  If an image format (png, jpg, bmp) is set, image files per frame are created.

  If "#" is included in the file name, the part will be replaced with the time step number when the files are output. If “#” is entered repeatedly, the number of “#” is treated as the number of digits. If "#" is not included in the file name, the specified file will be overwritten.
  <Example>
  file.mpg (creates an mpeg file)
  file###.png (creates file001.png, file002.png, ...)

- target view
  Select the view from which to obtain the images among the views where the timeStep map is registered.

4.20. timeStepSync

The timeStepSync allows the user to create a time step animation based on multiple time series data.

To do so, first load multiple time series data, and then register the timeStep map to one data, and the timeStepSync map to the other data. This operation will synchronize the other data to the timeStep map animation.

Selecting the timeStepSync in the view frame will display the following parameters in the selected map parameter area:

(1) selected

Displays the data registered as a reference target. The user cannot directly input this item.

(2) data list

Displays the list of the data that can be synchronized. The user can register the selected data by using the “set data to sync time-step” button.

(3) set data to sync time-step

Registers the selected data as a reference target. After the data is registered, the data name is displayed in “selected”, and the (synchronized) current time step of the original data is displayed in “current step”.

(4) current step

Displays the current time step of the original data synchronized to the reference target data. The user cannot directly input this item.

(5) pause

Specify whether or not to pause the animation. When this item is On (yes), the time step change in the target data will not be reflected on the time step on the original data. When this item is Off (no), the time step change in the target will be reflected on the time step on the original. The default is Off (no).

(6) show text label

Specify whether or not to show a text label in the view. If On (yes) is set, a text label with the time step number is displayed. The default is Off (no).

The user can set the following label parameters:

- format
  Specify the label format. The string “#S” is replaced with the time step number in the view. The default is “step=#S”.
- pos X
  Specify the position of the label in the X axis direction. Assuming the horizontal size of the view to be 100%, specify the display start point using the percentage value. The default value is 60.0.
- pos Y
  Specify the position of the label in the Y axis direction. Assuming the vertical size of the view to be 100%, specify the display start point using the percentage value. The default value is 90.0.
- size
  Specifies the font size. Assuming the vertical size of the view to be 100%, specify the font size using the percentage value. The default value is 5.0.

4.21. triaShape

The triaShape displays geometry data using sets of triangles.

Selecting the triaShape in the view frame will display the following parameters in the selected map parameter area:

(1) render type

Specify the method of rendering geometries. Select from the following:

face (polygons)
wire (wire frames)
point (vertex points)

(2) show two side

Specify whether or not to render both sides. The default is Off (no).

(3) line width

Specify the line width in pixels.
Enter a real number larger than 0.0. The default value is 1.0.
This setting will apply only to the render type “wire”.

(4) point size

Specify the point size in pixels.
Enter a value larger than 0.0. The default value is 3.0.
This setting will apply only to the render type “wire”.

(5) show colored

When a color is registered to a data face, specify whether or not to draw the face with the color. The default is On (yes).

(6) smooth normal

When a plane-normal is defined to a data piece, specify whether or not to calculate the vertex normal to implement smooth shading. If Off (no) is set, the data is displayed with flat shading. The default is Off (no).

4.22. volume

The volume map applies volume rendering to orthogonal grid data.

When “Lighting” is enabled, the map applies shadings referring to normal vectors generated from gradient data and “Hilight” value.

In addition, the map combines the alpha map according to the gradient vector size and the alpha channel of the data’s color map, and reflects the result in opacity values

Selecting the volume map in the view frame will display the following parameters in the selected map parameter area.

(1) select data

Select the data to which to apply volume rendering.
The default value is “data0”.

data0 (0th data)

...
dataN (Nth data)
veclen (Absolute vector length. Only for vector data)
None (Nothing is displayed.)

(2) show shade

Specify whether or not to enable “Lighting”. The default is On (yes).

(3) num slices

Specify the number of sampling slices for rendering process. The default value is 256.

(4) update minmax

Specify, when the target data (in “select data”) is changed, whether to change the color map range (max/min) according to the minimum and maximum values of the new data.
The default is On (yes).

(5) crop region

Specify the area to be rendered. Assuming the size of the entire region to be 1, set a value between 0.0 and 1.0 for each of the minimum and maximum positions in X, Y and Z directions.

(6) reset region

Resets the crop region values to the default.

(7) Gradient map

Define how the alpha map (opacity value) corresponds to gradient vector size.
Selecting this item displays the color map dialog where only the alpha channel can be edited.
The opacity based on the alpha map defined here is combined with the opacity set in the alpha channel of the color map applied to the current data values during rendering process.
The default values are all 1.0 (Opaque regardless of the gradient size).

The following controls are available in the color map dialog:

- Color map
  Change opacity values by moving the mouse cursor.
- Ramp
  Sets the opacity values to the default.
- Invert
  Inverts the values.
- Reset
  Clears all values.
- Cancel
  Closes the dialog without saving any changes.
- Close
  Closes the dialog after saving changes.

5. Visualization Proxy Program Settings

The visualization proxy (lsv-proxy) is a program that connects the client to the server, when executing a remote visualization process (including localhost) in the UserMode connection.

In the UserMode connection, users can construct their own remote visualization environment by controlling the server-side parameters.

This chapter describes the procedures for setting the lsv-proxy program for the UserMode connection.

5.1. Program startup options

The user can start the lsv-proxy via the visualization client through SSH. To do so, specify the command line for the lsv-proxy in the “proxy command” column on the UserMode connection tab (see Chapter 3.2 (2)) on the visualization client.

The command line specification for the lsv-proxy is as follows:

(lsv/bin/)**lsv-proxy**[-t timeout] [-p start] setupfile

-t timeout	Specify the time out value (in seconds). If omitted, 30 sec is assumed.
-p startP	Specify the first number of the communication port candidate. Ports are searched from “startP” to (“startP”+100). If omitted, 2000 is assumed.
setupfile	Specify the setup file path. This cannot be omitted.

Once the lsv-proxy is started, it executes the visualization server process via the execution host on the server through SSH according to the specified setup file. The program then continues to relay the communication between the visualization server and client, until the server process completes.

5.2. Setup file description

The setup file is a text file specifying how the lsv-proxy executes the visualization server. The description format is as follows:

- Comments
  Empty lines and those starting with “#” are ignored.
- Description format
  A line from the head to the line feed is considered to be a record. The record should be described in the following format:
  keyword=value

  Note that spaces and tabs are interpreted as a part of keyword or value.
  There are no priorities in writing each record. If the same keyword record is specified more than once, the last specification takes effect.

The available keywords are as follows:

- RHOST
  Specify the host name of the computer (execution host) that starts the visualization server. If omitted, the localhost is assumed.
- RUSER
  Specify the user name for the execution host. If omitted, the current user name is assumed.

- RPSWD
  Specify the password when password authentication is required for the SSH connection to the execution host.
- RSA_PASS
  Specify the private key passphrase when public key authentication is required for the SSH connection to the execution host.
- HOSTBASED
  Specify whether or not to use hostbased authentication for the SSH connection to the execution host, by “yes” or “no”.
- QSUB
  Specify the qsub command line for the execution host.
  The “#NP” in the string is replaced with the degree of parallelism for the visualization server process. (e.g. qsub –proc #NP)
  If omitted, the host will not be executed with the qsub command, but directly with MPIRUN command.
- MPIRUN
  Specify the mpirun command line for the execution host.
  The "-np (number of parallel processes)" is added to the string specified here, and the host is executed with the command line.
  If omitted, the MPI will not be used, and the visualization server is directly started. In this case, no parallel executions will be executed regardless of the degree of parallelism specified.
- SCRIPTDIR
  Specify the execution host directory to create the script file for starting the visualization server. If omitted, “/tmp” is assumed.
- LOGPATH
  Specify the path for the visualization server’s log output. If omitted, “/dev/null” is assumed.
- MSGPATH
  Specify the file path where the standard output and standard error output are redirected from the visualization server or mpirun.
  If omitted, “/dev/null” is assumed.
- PROXYLOGPATH
  Specify the file path where the standard output and standard error output are redirected from the lsv-proxy. If omitted, “/dev/null” is assumed.
- INITFILEPATH
  Specify the path of the initialization file used for the visualization server layer 2 access. If omitted, “$HOME/lsvlayer2init.xml” is assumed.
- LSVPATH
  Specify the visualization server path. An error will return if omitted.
- PORTFORWARDNODES
  When the lsv-proxy is executed on load-balanced nodes, specify the host names of the nodes by separating the names with spaces or tabs.

[Sample description]

```
# lsv-proxy setup file
RHOST=visnode
RUSER=visuser
RPSWD=passwordofvisuser
QSUB=/opt/nqs/bin/qsub -proc #NP -time 00:30:00 -mem 2048mb
MPIRUN=/opt/openmpi/bin/mpirun -machinefile/home/visuser/LSV/machines
LOGPATH=/tmp/lsv.log
MSGPATH=/tmp/lsv.msg
INITFILEPATH=/home/visuser/LSV/etc/lsvlayer2init.xml
LSVPATH=/home/visuser/LSV/bin/lsv-visualizer
```

5.3. Other notes

When the host executing the visualization server during remote visualization is running on Linux, the visualization server process must either be ready for starting the X server, or have an access to the started X server.

Note that if any X login service such as XDM (X Display Manager) is running on the execution host, the visualization server cannot operate. In this case, stop the X login service by switching the Linux run level to 3 (init 3) or by other means.

Change History

1st edition	July 2010
2nd edition	March 2011
3rd edition	June 2011

Table of Contents

1. Introduction

This text is the installation guide for the large-scale data visualization system (LSV). The LSV is a parallel visualization program built on client-server architecture capable of handling a large amount of data.

The visualization server is a visualization program that runs on a PC cluster supporting MPI parallel execution. The system serves either interactive- or batch-mode operations.

The visualization client serves as a user interface allowing users to operate a variety of functions of server programs over the network, and the executed server processes deliver resultant graphical objects to user screens

This document describes the requirements for building an operational environment, and the instructions for installing the LSV system.

2. Installation

2.1. System requirements

Recommended system requirements are as follows;

Table 2-1. System requirements for the LSV system

Architecture	AMD64, IA32
OS	Linux 2.6 MacOS 10.5/10.6
Software	OpenMPI 1.2/1.3 libxml2 2.6 libjpeg 6b freetype 2.1 glib2 2.4 curl 7.19 (*1) libssh2 1.2 Python 2.5 PIL 1.1 wxPython 2.8 PyOpenGL 3.0 Numpy 1.4 Matplotlib 0.99

If any of the software above does not exist on the current system, be sure to install the software before installing the LSV.

When installing the above software under a directory other than the system default path, change the environment variables PATH and LD_LIBRARY_PATH(DYLD_LIBRARY_PATH) accordingly so that the system can correctly refer to the software.

(*1)

When compiling the curl from its source code, configure the system as follows so that the system does not create any links to unnecessary libraries:

```
configure --prefix=installing directory--without-libssh2 --without-gnutls ¥
        --without-nss --without-ca-bundle --without-ca-path --without-libidn ¥
        --disable-ldap --disable-sspi --without-ssl
```

3. Compilation

3.1. Compiling software from source code

This section describes how to compile the LSV from its source code.

The following description assumes that the LSV operation environment is built under the $HOME/LSV directory. The user can change the path according to the user's choice of directory.

Follow the procedure below at the directory where the LSV source was extracted:

(1) Compiling the Sphere_SPL

- Modify the Sphere_SPL/Config.make according to the user's environment.
  Specify the compiler path (CXX, CC).
  Set the MPI (OpenMPI) environment:
  Set the OpenMPI directory under OMPI_DIR, and set MPICH_DIR value as a blank space.
- Execute "make" under the Sphere_SPL
  (cd Sphere_SPL; make)
- Copy the library to the LSV_Server/lib.
  cp Sphere_SPL/lib/libspl.a LSV_Server/lib

(2) Compiling the LSV_SceneGraph

- Execute "make" under the LSV_SceneGraph.
  (cd LSV_SceneGraph; make)
- Copy the library to the LSV_Server/lib.
  cp LSV_SceneGraph/libLSV_SceneGraph.so LSV_Server/lib

(3) Compiling the LSV_Server

- Execute "configure" under the LSV_Server/src.
  (cd LSV_Server/src; ./configure)
- Modify the LSV_Server/src/config.make according to the user's environment.
- Execute "make" under the LSV_Server/src.
  (cd LSV_Server/src; make)

(4) Compiling the LSV_HWRenderer

- Execute "make" under the LSV_HWRenderer/LSV_HWRenderer.
  (cd LSV_HWRenderer/LSV_HWRenderer; make)

(5) Compiling the LSVG

- Modify the LSVG/src/Makefile according to the user's environment.
  Set CXX the C++ compiler path of the MPI in use.
- Execute "make" under the LSVG/src.
  (cd LSVG/src; make)

(6) Compiling the PROXY

- Modify PROXY/src/Makefile according to the user's environment.
  Specify the path of LIBSSH2_DIR.
- Execute "make" under PROXY/src.
  (cd PROXY/src; make)

(7) Compiling the LSVC

- Modify LSVC/Makefile according to the user's environment.
  Specify the paths for LIBXML2_DIR and CURL_DIR.
- Execute "make" under LSVC
  (cd LSVC; make)

(8) Compiling Tools

- Modify Tools/IndexFileMaker/Makefile according to the user's environment.
  Set CXX the C++ compiler path of the MPI in use.
- Execute "make" under Tools/IndexFileMaker.
  (cd Tools/IndexFileMaker; make)
- Modify Tools/SphereFileDivider/Makefile according to the user's environment.
  Set CXX the C++ compiler path of the MPI in use.
- Execute "make" under the Tools/SphereFileDivider.
  (cd Tools/SphereFileDivider; make)
- Modify Tools/ParticleTracer/Makefile according to the user's environment.
  Set CXX the C++ compiler path of the MPI in use.
- Execute "make" under the Tools/ParticleTracer.
  (cd Tools/ParticleTracer; make)

(9) Executing the INSTALL_LSV.py

```
INSTALL_LSV.py --prefix=$HOME/LSV
```

The following directories are created under $HOME/LSV/.

```
|-- bin
|-- etc
|-- lib
|-- logs
`-- share
    |-- python
    `-- xml
```

3.2. Configuration

After completing the compilation and installation of LSV, configure the LSV operation environment as follows:

(1) Configuring environment variables

The LSV system requires the following environment variables to be set:

- PATH
- LD_LIBRARY_PATH
- LSV_CORE_CONFIG_FILEPATH

Sample description for $HOME/.bashrc

```
export PATH =$HOME/LSV/bin:$PATH
export LD_LIBRARY_PATH =$HOME/LSV/lib:$LD_LIBRARY_PATH
export LSV_CORE_CONFIG_FILEPATH =$HOME/LSV/etc/lsv_coreconfiguration.xml
```

Sample description for $HOME/.ssh/environment

```
PATH=/home/user_name/LSV/bin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
LD_LIBRARY_PATH =/home/user_name/LSV/lib:/usr/local/lib:/usr/lib
LSV_CORE_CONFIG_FILEPATH =/home/user_name/LSV/etc/lsv_coreconfiguration.xml
```

Be sure not to refer to shell variables in the definitions of the $HOME/.ssh/environment. Further, it is necessary to set the sshd configuration file so that the user-specified environment variables are set at SSH login.

Check to be sure that the following code line exists in the /etc/ssh/sshd_config.

```
PermitUserEnvironment yes
```

If the line does not exist, add the line above to the “/etc/ssh/sshd_config” and send the HUP signal (kill -HUP sshd process ID) to the sshd process, or restart the system.

Note that this operation requires system administrator privilege.

(2) Modifying PROXY configuration file

If necessary, change the descriptions in the PROXY configuration file ($HOME/LSV/etc/lsv-proxy.cfg) appropriately.

- When using password authentication, set a password in the RPSWD.
- When using public key authentication, set the passphrase in the RSA_PASS. (Specify a blank when no passphrase is used.)
- When using hostbased authentication, set HOSTBASED to “yes”.
- To start jobs using qsub, specify the qsub command line in the QSUB.
- Specify the ”mpirun” command line in MPIRUN.

4. Installation of Binary Package

The LSV binary package is available for some computer environments. The user can use the LSV in these environments by extracting and installing the binary package.

The binary packages available for the LSV are as follows:

- LSVeoe-0.5.n-centos4-amd64.tgz : Linux(CentOS 4.x) x86_64
- LSVeoe-0.5.n-centos5-amd64.tgz : Linux(CentOS 5.x) x86_64
- LSVeoe-0.5.n-macos10.5-i386.tgz : MacOS 10.5(Leopard)/10.6(Snow Leopard) i386/x86_64

Follow the procedures below to install the program:

(1) Extracting the binary package from the tar file

The LSV binary package can be stored and used at any directory of the user's choice. Move to the directory in which to install the LSV and execute the following command:

tar xvfz LSVeoe-*version-os-arch*.tgz

A new directory named "lsv" appears in the directory chosen, and the files listed below are found in the new directory:

```
lsv
|-- bin
|-- etc
|-- include
|-- lib
|-- logs
|-- share
|     |-- python
|     `-- xml
|-- LSV.app (Only for MacOS)
|-- tmpl
`-- Setup.sh
```

(2) Executing the Setup_lsv.sh

Move to the lsv directory just created, and execute the Setup_lsv.sh.

```
cd lsv
./Setup.sh
```

The following files should be created after the completion of the operation above:

- lsv/etc/lsv-proxy.cfg
  Configuration file for the visualization proxy program (lsv-proxy)
  (referred to when the visualization server is executed on the local host)
- lsv/etc/lsv_coreconfiguration.xml
  Configuration file for the visualization server plug-in modules
  (referred to when the visualization server is executed on the local host)
- lsv/share/xml/lsv_layer1_init.xml
  Network configuration file for the visualization client
- lsv/share/xml/lsv_layer2_init.xml
  Network configuration file for the visualization server
  (referred to when the visualization server is executed on the local host)
- $HOME/.lsvc.cfg
  Default configuration file for the visualization client
  Stores the default parameters for connecting to the visualization server.

- $HOME/.ssh/environment
  sshd environment variables file
  (referred to when the visualization server is executed on the local host)

(3) Setting the sshd_config (optional)

If the user wishes to execute the visualization server on the local host, it is necessary to set the sshd configuration file so that the user-specified environment variables are set at SSH login.
Check to be sure that the following code line exists in the /etc/ssh/sshd_config.

```
PermitUserEnvironment yes
```

If the line does not exist, add the line to the "/etc/ssh/sshd_config" and send the HUP signal (kill -HUP sshd process ID) to the sshd process, or restart the system.
Note that this operation requires system administrator privilege.

(4) Setting up the command search path (optional)

Set as follows so that the command search path includes the lsv/bin directory:

- When using BSH (sh, bash, etc.)
  Add below to the $HOME/.profile or $HOME/.bashrc:

```
export PATH=installing directory/lsv/bin:$PATH
```

- When using CSH (csh, tcsh, etc.)
  Add below to the $HOME/.cshrc or $HOME/.tcshrc:

```
setenv PATH installing directory/lsv/bin:$PATH
```