VLSV2 File Format Specification

Sandroos, A.

May, 2011.

Part I
VLSV2 File Format

1 Introduction

VLSV2 file format reads and writes arrays that consist of vector elements. Additional metadata can be supplemented in the form of XML tags. A VLSV2 file contains one or more arrays (in binary) and a footer, which contains the necessary information to interpret and read the file contents. The arrays can be written and read in parallel with collective MPI functions.

VLSV2 file format does not specify the kind of data in the file, i.e. the file format is not specific to Vlasov simulations as such - it just contains quite general arrays. A creation of yet another file format was necessary, however, because of difficulties in writing Vlasov data to files using existing libraries, e.g. Adios.

2 VLSV2 File Structure

A VLSV2 file contains arrays whose elements are vectors, which consist of one or more components. An array specification is static in the sense that each vector in a given array is of fixed size. Vector sizes can, of course, vary between different arrays. Furthermore, each component of a vector has a byte size corresponding to the byte size of a native C++ datatype. Array and vector sizes are given in footer XML tags. The footer always contains an XML tag per array (Fig. 1).

2.1 XML Tags

The footer contains one XML tag (in human-readable form) per an array in the file. The XML tag tree begins with a tag called `<VLSV>` as a root, which does not have any attributes nor a value. The rest of the tags are children of the root and are user-specific.

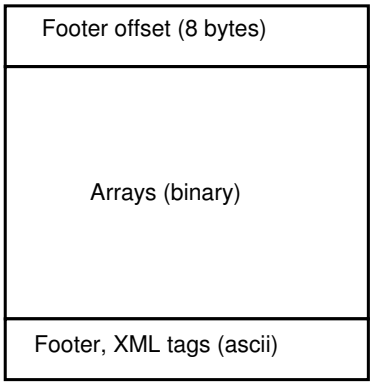

Figure 1: General structure of a VLSV2 file. The file always begins with an unsigned int of size eight bytes giving the offset to the start of the footer. The footer contains XML tags describing the contents of arrays, written in binary, as well as offsets to the start of each array.

Algorithm 1 Mandatory XML tag attributes in a VLSV2 file footer.

arrayname	string	Name of the array
arraysize	uint	Size of array
vectorsize	uint	Size of vector
datatype	string	int/uint/float
datasize	uint	Byte size of vector comp.

Each XML tag has a name and value as well as several attributes, some of which are mandatory (Algorithm 1). The (XML tag name, array name) pair is used as a unique identifier. The value of the tag is always the offset (in ascii) into the beginning of the array data. Note that attributes `datatype` and `datasize` together uniquely define the C++ datatype written in each array. For example, if `datatype=float` and `datasize=8`, then the array contains `doubles`. In practice the reading and writing of XML tags and arrays is handled through `VlsvReader2` and `VlsvWriter2` class interfaces.

Part II
Vlasov VLSV2 Files

3 Vlasov Data

A Vlasov output file contains at minimum spatial cell data, and values computed by data reduction operators (DROs). At least spatial cell global IDs,

coordinates, and cell sizes have to be written in order to be able to reconstruct the grid from file. Spatial cell IDs are written to an array with `tagname=MESH`, `name=SpatialGrid`. The coordinates of the bottom lower left corner of each cell, and size of cell in xyz-directions, are written into an array with `tagname=COORDS`, `name=SpatialGrid`. The coordinate array contains values x, y, z, dx, dy, dz. Thus, `vectorsize=6`.

DRO variables are written into arrays with `tagname=VARIABLE`, `name=<var name>`, where `<var name>` is a variable name issued by each DRO. More information is required to associate the variables with a spatial mesh, and this is handled by adding an attribute `mesh=SpatialGrid` into the corresponding XML tags. `vectorsize` for DRO variables is given by the DROs themselves. For example, 3D-vector variables have a `vectorsize=3`.

The spatial cell ID array defines the ordering of data in arrays `(COORDS, SpatialGrid)` and `(VARIABLE, <var name>)`. In other words, the data in the abovementioned arrays is in the same order as in array `(MESH, SpatialGrid)`.

Velocity block data is handled in a similar manner, although more (meta)data is required because every spatial cell may not have written its velocity grid, and the velocity grids may be of different sizes. Velocity grid blocks, however, have the same size over the lifetime of the simulation (typically $4^3 = 64$), which is used as the `vectorsize` XML attribute in the array containing volume averages.

Global ID numbers of spatial cells whose velocity grids are stored are written into an array `(CELLSWITHBLOCKS, SpatialGrid)`. This array also defines the order in which data is written into subsequent arrays described below.

Each velocity grid may contain a different amount of velocity grid blocks. The number of blocks in each stored velocity grid is written into an array `(NBLOCKS, SpatialGrid)`. The coordinates of the bottom lower left corner of each velocity grid block, and the sizes of cells in each coordinate direction in each block, are written into an array `(BLOCKCOORDINATES, SpatialGrid)`. Block coordinate array contains values $v_x, v_y, v_z, dv_x, dv_y, dv_z$, thus `vectorsize=6`. Finally, the volume averages are written into an array `(BLOCKVARIABLE, avgs)`, which also has an attribute `mesh=SpatialGrid` to associate the block variable with the spatial mesh.

Part III
Sample VLSV2 XML Tree

Below is a sample XML tree from a VLSV2 Vlasov file. The file contains the spatial cells (IDs and coordinates), data written by DROs (E, B, rho, rho_v, MPI_Rank), and velocity blocks. Note that there is only one spatial cell in the file, thus the array sizes are equal to unity.

```
<VLSV>
        <BLOCKCOORDINATES
                arraysize="640"
                datasize="4"
```

```
          datatype="float"
          name="SpatialGrid"
          vectorsize="6">
          96
</BLOCKCOORDINATES>
<BLOCKVARIABLE
          arraysize="640"
          datasize="4"
          datatype="float"
          mesh="SpatialGrid"
          name="avgs"
          vectorsize="64">
          15456
</BLOCKVARIABLE>
<CELLSWITHBLOCKS
          arraysize="1"
          datasize="4"
          datatype="uint"
          name="SpatialGrid"
          vectorsize="1">
          88
</CELLSWITHBLOCKS>
<COORDS arraysize="1"
          datasize="4"
          datatype="float"
          name="SpatialGrid"
          vectorsize="6">
          20
</COORDS>
<MESH arraysize="1"
          datasize="4"
          datatype="uint"
          name="SpatialGrid"
          vectorsize="1">
          16
</MESH>
<NBLOCKS
          arraysize="1"
          datasize="4"
          datatype="uint"
          name="SpatialGrid"
          vectorsize="1">
          92
</NBLOCKS>
<VARIABLE
          arraysize="1"
          datasize="4"
          datatype="float"
          mesh="SpatialGrid"
          name="B"
          vectorsize="3">
          44
</VARIABLE>
<VARIABLE
          arraysize="1"
          datasize="4"
          datatype="float"
```

```
                mesh="SpatialGrid"
                name="E"
                vectorsize="3">
                56
        </VARIABLE>
        <VARIABLE
                arraysize="1"
                datasize="4"
                datatype="float"
                mesh="SpatialGrid"
                name="rho"
                vectorsize="1">
                68
        </VARIABLE>
        <VARIABLE
                arraysize="1"
                datasize="4"
                datatype="float"
                mesh="SpatialGrid"
                name="rho_v"
                vectorsize="3">
                72
        </VARIABLE>
        <VARIABLE
                arraysize="1"
                datasize="4"
                datatype="int"
                mesh="SpatialGrid"
                name="MPI_rank"
                vectorsize="1">
                84
        </VARIABLE>
</VLSV>
```