VMAP Standard

IO-Lib, Wrappers and Converters

www.vmap-standard.org

Andre Oeckerath

Fraunhofer SCAI

VMAP – IO-Lib, Wrappers and Converters

Agenda

- **VMAP IO-Lib 1.3**
  - Configurable floating-point precision
  - Enhanced data compression
- **VMAP - Wrappers and Converters**

VMAP – IO-Lib

Version 1.3

- **Single / double floating-point precision flag added to all dataset**

void writePointsBlock (const std::string &groupName, const sPointsBlock &points, bool useDoublePrecision=true)
Writes **sPointsBlock** to file; Creates DataSet "POINTS" in group "groupName".

void writeVariablesBlock (const std::string &groupName, const std::vector< VMAP::sStateVariable > &variables, bool useDoublePrecision=true)
Writes multiple **sStateVariable** to file; Creates list of variables in group "groupName".

void writeVariable (const std::string &groupName, const VMAP::sStateVariable &variable, bool useDoublePrecision=true)
Writes single **sStateVariable** to group "groupName".

void writeMeasurandValuesBlock (const std::string &groupName, const std::vector< VMAP::sMeasurandValues > &variables, bool useDoublePrecision=true)
Writes a vector measurand values to group "groupName".

void writeMeasurandValues (const std::string &groupName, const VMAP::sMeasurandValues &variable, bool useDoublePrecision=true)
Writes a measurand values as a group "myMeasurandName" from group "groupName".

void writeTable (const std::string &groupName, const sTable &table, bool useDoublePrecision=true)
Writes a 2D table to HDF.

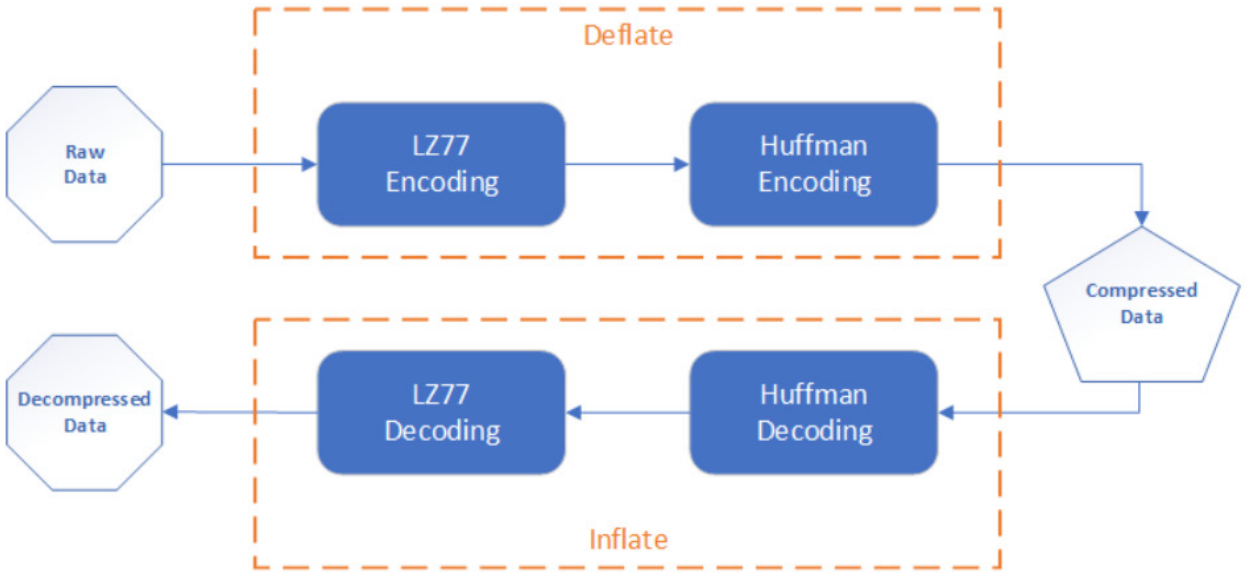
VMAP – IO-Lib

Version 1.3

- **Precision can be different per variable**
  - If precision is critical -> 64-bit double precision
  - In precision is uncritical -> 32-bit float precision
- **Precision can be different per time step**
  - For visualization -> 32-bit float precision
  - For Restart / Data Mapping -> 64-bit double precision for last time increment

VMAP - data size reduction by compression

zip compression – how does it work?



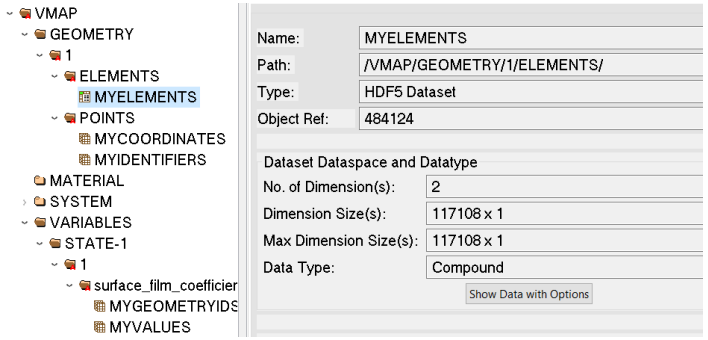
- **LZ77 is a dictionary based lossless compression (basic idea is to replace an occurrence of a particular sequence of bytes in data with a reference to a previous occurrence of that sequence)**
- **Huffman encoding is a statistical compression method (data symbols are decoded by their frequency of appearance). Code for more frequently occurring data symbols are shorter than codes for less frequently occurring data symbols.**

VMAP compression can be improved, if compressed datasets are more homogenic

VMAP – IO-Lib

Consequences to VMAP data structures

- **Avoid Compound data types, in particular the VMAP elements block**



Compound Dataset Members

Name	Type	Array Size	
myIdentifier	32-bit integer	1	← Unique per part
myElementType	32-bit integer	1	← Normally much smaller than #Elements
myCoordinateSystem	32-bit integer	1	← Constant, if not element local coordinate system
myMaterialType	32-bit integer	1	← Normally much smaller than #Elements
mySectionType	32-bit integer	1	← Only needed for certain solvers
myConnectivity	Variable-length of 32-bit integer	1	← Variable-length block

- **'One' dictionary per compound member seems to be more suitable than a dictionary for complete DataSet**

VMAP – IO-Lib

Consequences to VMAP data structures

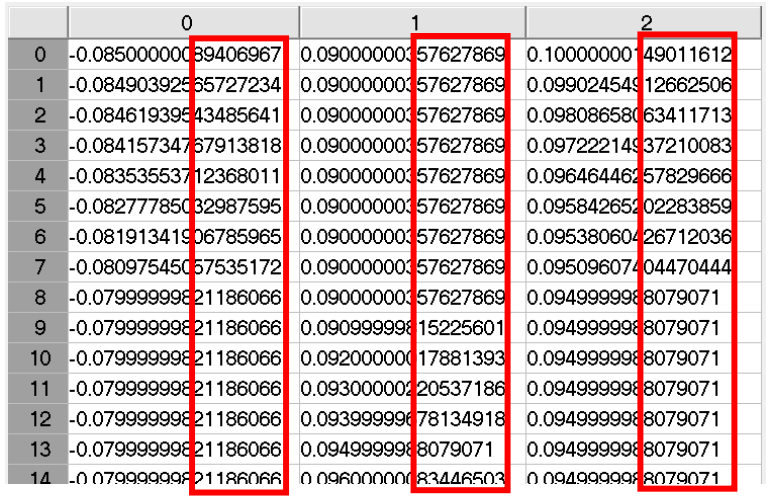
- **Split higher dimensional DataSets into its dimensions**
  - GEOMETRY/POINTS
  - VARIABLES
    - Vectors
    - Tensors
- **E.g. for a sPointsBlock of a model part, it is more likely that X-values are similar than a point has similar xyz values**

VMAP - data size reduction by compression

Lossy compression

- **For certain use of VMAP, e.g. result visualization, we can favor enhanced compression by accepting losing floating point precision**

- **E.g. for visualization "I can accept nodal coordinates to be rounded to 0.01 mm precision"**

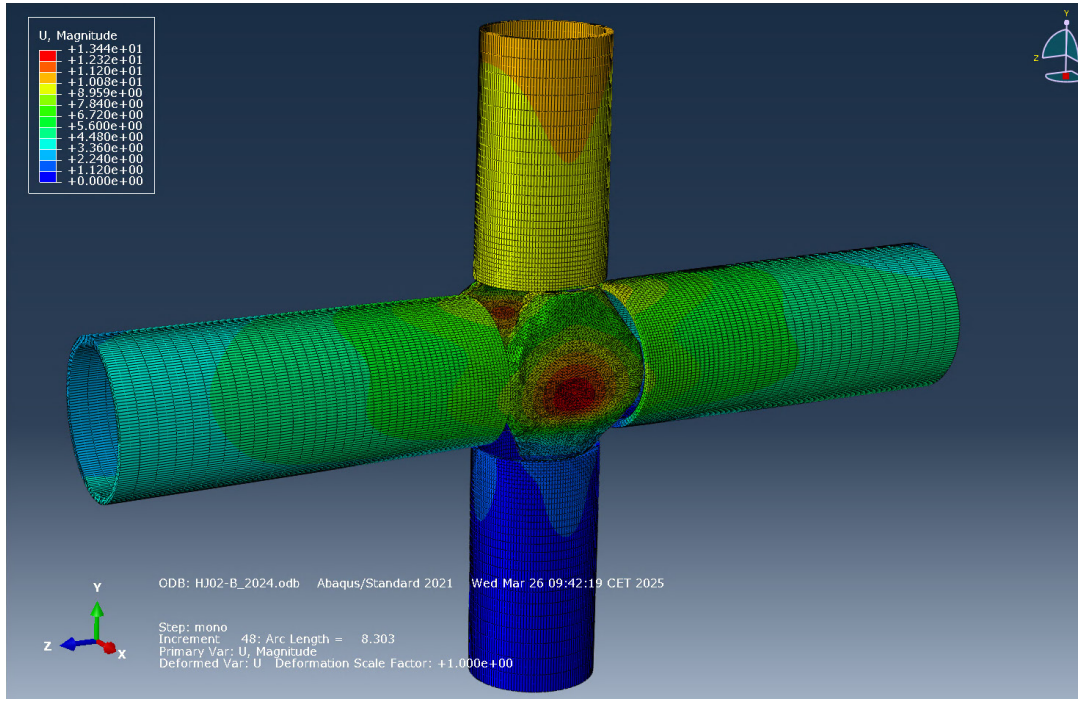


	0	1	2
0	-0.085000000 89406967	0.090000003 57627869	0.100000001 49011612
1	-0.084903925 65727234	0.090000003 57627869	0.099024549 12662506
2	-0.084619395 43485641	0.090000003 57627869	0.098086580 63411713
3	-0.084157347 67913818	0.090000003 57627869	0.097222149 37210083
4	-0.083535537 12368011	0.090000003 57627869	0.096464462 57829666
5	-0.082777850 32987595	0.090000003 57627869	0.095842652 02283859
6	-0.081913419 06785965	0.090000003 57627869	0.095380604 26712036
7	-0.080975450 67535172	0.090000003 57627869	0.095096074 04470444
8	-0.079999998 21186066	0.090000003 57627869	0.094999998 8079071
9	-0.079999998 21186066	0.090999998 15225601	0.094999998 8079071
10	-0.079999998 21186066	0.092000000 17881393	0.094999998 8079071
11	-0.079999998 21186066	0.093000002 20537186	0.094999998 8079071
12	-0.079999998 21186066	0.093999996 78134918	0.094999998 8079071
13	-0.079999998 21186066	0.094999998 8079071	0.094999998 8079071
14	-0.079999998 21186066	0.096000000 83446503	0.094999998 8079071

➢ **Floating points can be transformed to integers to allow better compression ratios**

Results

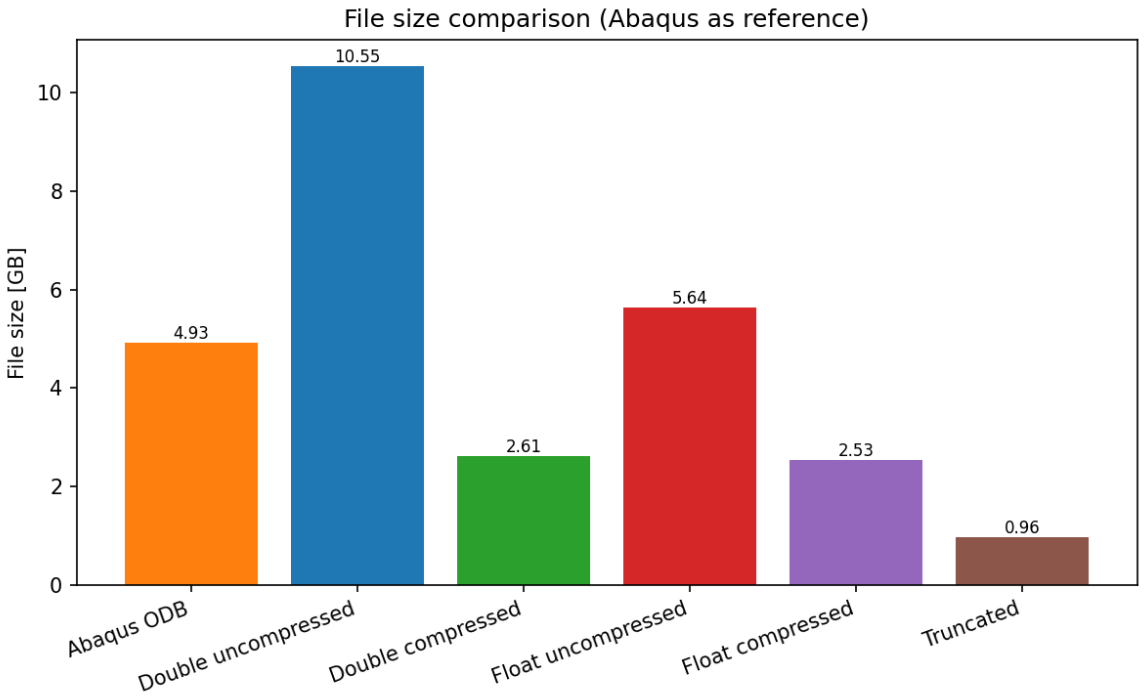
Sample data

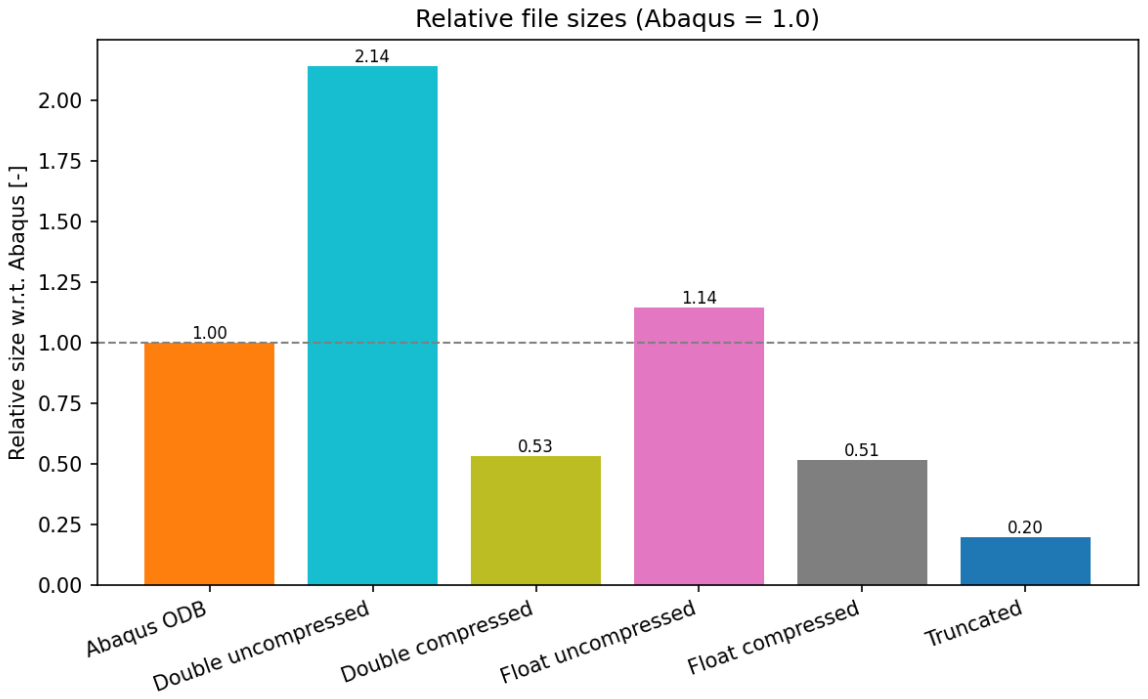


- 2 parts
- ~450.000 nodes
- 150.000 elements (hexahedron 20, tetrahedron 10)
- 48 increments
- Abaqus ODB ~5 GB (single precision)

Result

Sample data file sizes

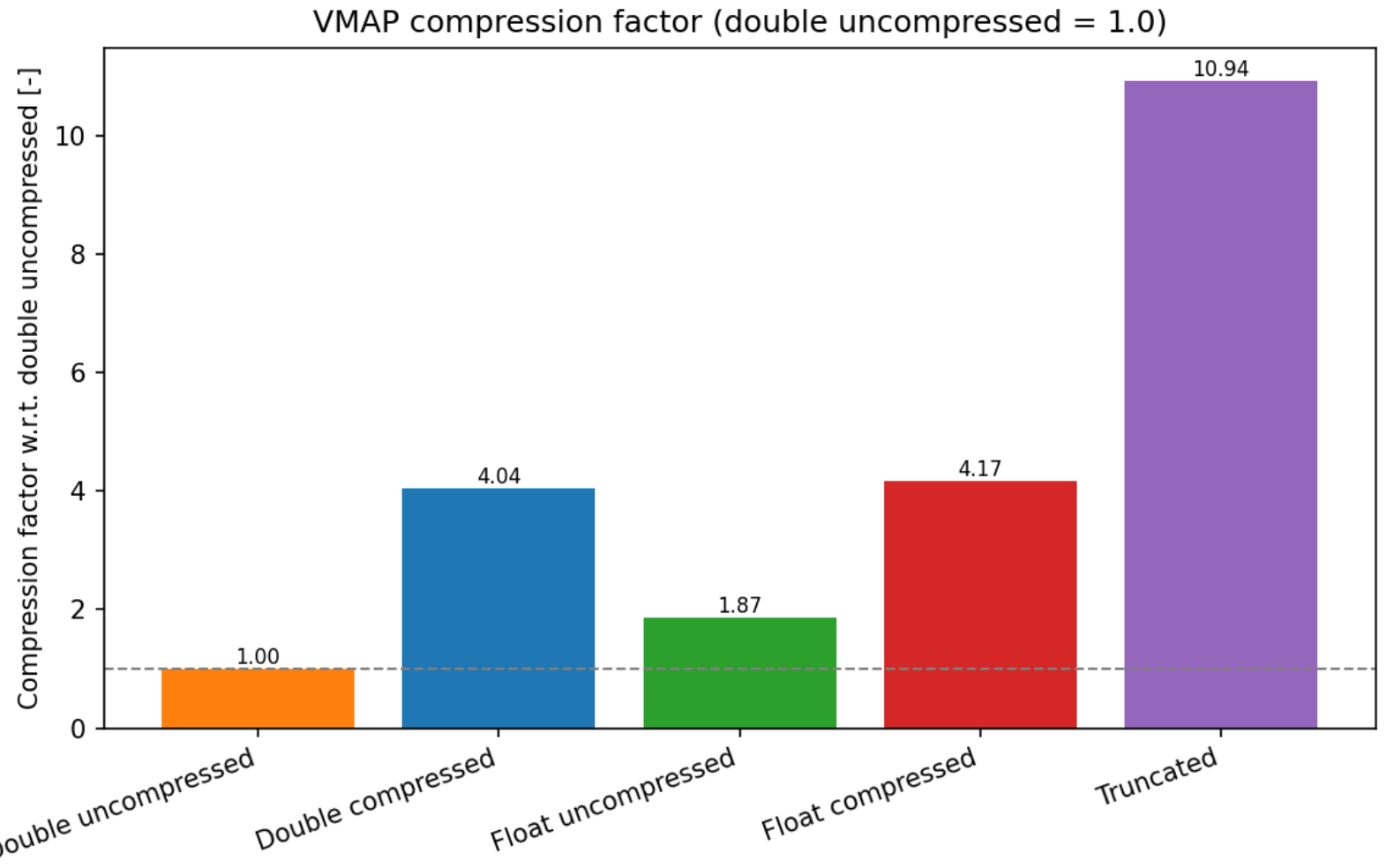




Fraunhofer SCAI

Results

VMAP



VMAP – IO-Lib

Summary

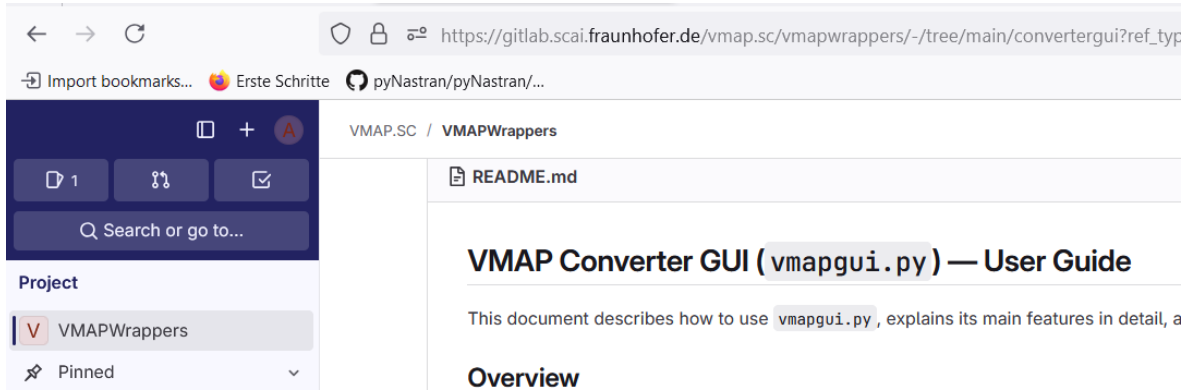
- **Single / double floating-point precision can be assigned individually to variables or time steps**
- **Data split has been applied to VMAP-IO Lib internally, so no changes to present use of lib is needed**
- **VMAP-IO Lib supports both, the compression optimized layout as well as the ‘Standard’ layout**

VMAP

Wrappers and Converters

VMAP Converter GUI

- **Lightweight Python GUI on top of the vmapwrapper package**
- **Unified front-end to convert CAE, CFD, measurement and robot data into VMAP files**
- **Discovers converter modules via formats.ini and exposes them as selectable formats**
- **Build-in PyVMAP handler for Python 3.9-3.14**



Fraunhofer SCAI

VMAP – Converter GUI

Supported formats

Finite Element

- **Abaqus (ODB)**
  Abaqus Python
- **CalculiX (FRD)**
  ccx2paraview + VTK
- **Ansys Mechanical APDL (RST)**
  pyANSYS
- **LS-DYNA (D3Plot)**
  VTK
- **Nastran (OP2)**
  pyNastran
- **Marc (HDF5)**
  h5py

CFD

- **OpenFOAM (foam)**
  VTK
- **Fluent CFF (HDF5)**
  VTK
- **EnSight Gold**
  VTK

Measurement

- **Excel Spreadsheets**
  pandas + matplotlib
- **STL Files**
  stl
- **MX3D Files**
  pandas + json
- **KUKA Robots Src**
  pandas + json

General-purpose

- **VTK Unstructured Grid (VTU)**
  VTK

VMAP – Converter GUI

FE code wrappers - features

Abaqus (ODB)

- **Mesh**
  - Nodes
  - Elements
  - Sets
  - Parts
- **Variables**
  - Displacement
  - Stress
  - Strain
  - Temperature
  - Forces
  - Others
- **Analysis**
  - Static
  - Transient

Ansys MAPDL (RST)

- **Mesh**
  - Nodes
  - ~30 solid and shell elements
- **Variables**
  - Displacement
  - Forces
  - Stress
  - Strain
  - Temperature
- **Analysis**
  - Static
  - Transient

LS-DYNA (D3Plot)

- **Mesh**
  - Nodes
  - Shell elements
  - Parts
- **Variables**
  - Displacement
  - Velocity
  - Acceleration
  - Stress/Strain
  - Energy
  - Thickness
  - History Variables
  - Element death
- **Analysis**
  - Transient

Nastran (OP2)

- **Mesh**
  - Nodes
  - Solid and Shell elements
- **Variables**
  - Displacement
  - Forces
  - Stress
  - Strain
- **Analysis**
  - Static
  - Transient

Marc (HDF5)

- **Mesh**
  - Nodes
  - Shell elements
- **Variables**
  - Displacement
  - Reaction forces
  - External forces
  - Stress
  - Strain
- **Analysis**
  - Static
  - Transient

CalculiX (FRD)

- **Mesh**
  - Nodes
  - Solid and Shell elements
  - Parts
- **Variables**
  - Displacement
  - Temperature
  - Reaction forces
  - Stress
- **Analysis**
  - Static
  - Transient

VMAP – Converter GUI

CFD code wrappers - features

OpenFOAM (foam)

- **Mesh**
  - Nodes
  - Elements
  - Parts
- **Variables**
  - Temperature
  - Pressure
  - Other scalar or vector fields
- **Analysis**
  - Transient
  - No-remeshing support

Fluent CFF (HDF5)

- **Mesh**
  - Nodes
  - Elements
  - Parts
- **Variables**
  - Any
- **Analysis**
  - single time step result

EnSight Gold

- **Mesh**
  - Nodes
  - Elements
  - Parts
- **Variables**
  - Displacement
  - Velocity
  - Acceleration
  - Stress/Strain
  - Energy
  - Thickness
  - History Variables
  - Element death
- **Analysis**
  - Transient

VMAP – Converter GUI

Measurement and general purpose wrappers - features

Excel Spreadsheets (.xls)

- **Sheet to VMAP Table**
- **Optional images / plots**

STL (stl)

- **Mesh**
  - Nodes
  - Elements
  -
- **Variables**
  - Normals

MX3D

- **Mesh**
  - Nodes
  - Elements
- **Variables**
  - Any
- **Analysis**
  - Transient

KUKA Robot

- **Trajectory path mesh**
  - Nodes
  - Beam elements
- **Variables**
  - Position
  - Orientation
  - Power
  - Velocity
- **Analysis**
  - Transient

VTK Unstructured Grid

- **Mesh**
  - Nodes
  - Elements
- **Variables**
  - Point data (scalar / vector / tensor)
  - Cell data (scalar / vector / tensor)
- **Analysis**
  - Transient

Converter GUI

Configurable and extensible design

- **Central configuration via formats.ini (one section per format)**
  - module → subfolder under vmapwrapper (e.g. abaqus, ansys, vtk, measurement, robots)
  - function → converter function or script entry point
  - extensions → file types offered in the file dialog

- **New formats can be added without changing the GUI core:**

- **Implement a converter function (recommended signature: myformat_to_vmap(input_path, output_path, compress, precision, timesteps, interval))**

VMAP – IO-Lib, Wrappers and Converters

Thank you very much.

Any questions?