# VTK QUADRATURE POINT SUPPORT DESIGN DOCUMENT

Burlen Loring and Berk Geveci
Kitware Inc., 28 Coporate Drive, Clifton Park, NY 12065 USA

December 8, 2008

## 1 INTRODUCTION

Quadrature points are points at which function values are evaluated for the purpose of numerical quadrature[1]. For each element type and numerical quadrature scheme, there is a given set of quadrature points at fixed parametric locations. The goal of this project is to provide support for quadrature points in the VTK pipeline for the purpose of visualization. To this end we have added the requisite classes and made the relevent structural changes to VTK providing following functionality: programmatic definition of arbitrarily defined quadrature schemes, organization and grouping of individual quadrature scheme definitions into collections called dictionaries, translation of dictionaries to and from an XML representation, interpolation of data fields to quadrature points, and generation of quadrature point point sets.

We implemented this functionality for unstructured grid data sets, where cell types vary across the data set's geometry. In order to process a given unstructured grid data set, one quadrature scheme definition for each cell type found in the data set must be provided. We call such a collection of definitions a dictionary. A given dictionary is "valid" with respect to the data set into which it is embedded when it contains a definition for each cell type found in that data set. We have chosen to implement quadrature point support on a per data field basis, as opposed to a per data set basis. This gives the greatest degree of flexibility so that data fields which have been integrated using different quadrature schemes may be treated in the same dataset. An implication of this choice is that each data field that is desired to be visualized at quadrature points must provide its own dictionary. A data field can only be processed when it has a valid dictionary embedded.

With the new functionality VTK now supports the following operations on quadrature point sets for appropriately defined unstructured grid data sets: scalar and vector warp, clip (geometry subset) via extract geometry, geometry subset via scalar threshold, the computation of basic statistics (i.e. min, max, and mean) over interpolated fields, and histogram generation. This document describes these new features for developers who wish to use them.

[1] The term quadrature and integration are used interchangeably throughout this document.

# 2 NEW FUNCTIONALITY

## 2.1 QUADRATURE SCHEME DEFINITION

The vtkQuadratureSchemeDefinition class is a data type that describes a particular quadrature scheme. A quadrature scheme definition is intended to have a per array scope, and is associated with a single cell type found in the data set in which it is embedded. Quadrature scheme definitions are always stored in quadrature scheme dictionaries, and quadrature scheme dictionaries are always stored in data arrays.

The quadrature scheme definition stores its cell type, the number of nodes of that cell type, and the number of quadrature points. In addition an array of shape function weights,

$$w_{ij} \equiv N_i(\vec{r}_j), \tag{1}$$

are stored. Here $i$ denotes a particular node identifier, and $j$ denotes a particular quadrature point identifier. Note that we are not storing the shape functions directly but rather the shape functions evaluated at the parametric coordinates of the quadrature points, given by $\vec{r}_j$. This provides a high degree of flexibility and extensibility.

## 2.2 QUADRATURE SCHEME DICTIONARIES

The vtkInformationQuadratureSchemeDefinitionVectorKey is a container class designed to map quadrature scheme definitions to cell types. For sake of brevity we refer to this class simply as the "quadrature scheme dictionary". The quadrature scheme dictionary is an implementation of the vtkInformationKey interface. This interface allows specialized meta data to be handled in a general and consistent manner across the VTK and ParaView code base. Classes that implement this interface can be stored in vtkInformation maps and are referenced by a static key. The key used to retrieve a given quadrature scheme dictionary from a given information map, is obtained by calling vtkQuadratureSchemeDefinition::DICTIONARY(). Because the key is static only one dictionary can be embedded in any given information object. Each data array to be processed must have a dictionary mapping cell types, found in the dataset into which it is embedded, to quadrature scheme definitions.

Dictionary generation can be handled in one of two ways. First dictionaries are automatically generated as data is read from disk where they have been encoded in VTK XML data format. This is covered in detail in section 2.2.2. Second dictionaries can be generated pragmatically according to a developers specific needs. An example of pragmatically generating dictionaries can be found in the vtkQuadratureSchemeDictionaryGenerator class. The vtkQuadratureSchemeDictionaryGenerator is a class that is used by our tests to embed a default dictionary inside the data fields of the unstructured grid on its input. Note, it is used by our regression tests so its definitions should not be modified.

### 2.2.1 VTK PIPELINE SUPPORT FOR QUADRATURE SCHEME DICTIONARIES

It is advantageous to be able to leverage VTK filters that do have knowledge of quadrature scheme dictionaries. These filters must pass dictionaries from input

to output, so that downstream quadrature point aware filters can function[2]. We have implemented the requisite changes to vtkAbstractArray and its subclasses, and to vtkFieldData and its subclasses to make this possible. Deep copying vtkAbstractArray and its subclasses, and vtkFieldData and its subclasses results in a deep copy of the information objects embedded in associated data arrays. On a lower level the CopyStructure, CopyAllocate and InterpolateAllocate methods also deep copy information objects embedded in arrays as well.

### 2.2.2 XML REPRESENTATION OF QUADRATURE SCHEME DICTIONARIES

We have added support for reading and writing quadrature scheme dictionaries and definitions to and from XML data files. The XML structure for a dictionary is as follows.

```
<DataArray ... >
    .
    .
    .
  <InformationKey name="DICTIONARY" location="vtkQuadratureSchemeDefinition">
    <vtkQuadratureSchemeDefinition>
      <CellType value="T1"/>
      <NumberOfNodes value="N1"/>
      <NumberOfQuadraturePoints value="Q1"/>
      <ShapeFunctionWeights>
         ...
      </ShapeFunctionWeights>
      <QuadratureWeights>
          ...
      </QuadratureWeights>
    </vtkQuadratureSchemeDefinition>
       .
       .
       .
    <vtkQuadratureSchemeDefinition>
      <CellType value="Tn"/>
      <NumberOfNodes value="Nn"/>
      <NumberOfQuadraturePoints value="Qn"/>
      <ShapeFunctionWeights>
         ...
      </ShapeFunctionWeights>
      <QuadratureWeights>
          ...
      </QuadratureWeights>
    </vtkQuadratureSchemeDefinition>
  </InformationKey>
    .
    .
    .
```

[2]A second requirement is that the filter does not split any cells.

```
</DataArray>
```

Here ellipses have been used to show data we have left out, and we have shown the structure for $n$ definitions. $T$ coresponds to a vtkCell type, $N$ to its number of nodes, and $Q$ to the number of quadrature points represented. Weights are written as space delimitted ASCII data. In our design the quadrature scheme dictionaries and definitions are responsible for representing and restoring their states to and from an XML representation on request. The vtkXML-Reader/Writer is responsible for extracting/placing the XML representation out of/into its associated stream.

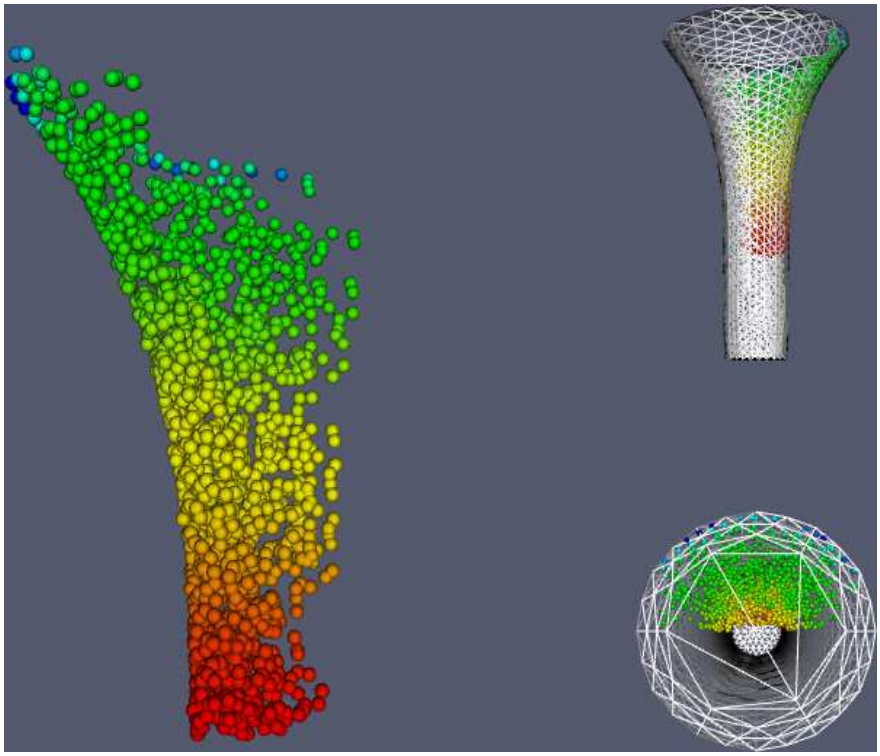

Figure 1: Results of the TestQuadraturePoints regression test. An unstructured grid with tetrahedral cells arranged in a cylinder is warped, clipped, and threshold-ed, before interpolating data fields to quadrature points and generating the quadrature point point set.

## 2.3 NODAL DATA FIELD INTERPOLATION

The vtkQuadraturePointInterpolator class is a filter that takes in an unstructured grid data set with quadrature scheme dictionaries embedded into its data arrays and puts out an unstructured grid data set with data fields interpolated to the quadrature points. The filter will only process data arrays of type float or double with valid dictionaries embedded, all other arrays are ignored. The filter interpolates data fields to the quadrature points of each cell. The interpolation

for a scalar data fields is made as follows:

$$\phi_j = \sum_{i=1}^{n} w_{ij}\phi_i, \tag{2}$$

where $i$ is the node identifier, $j$ is the quadrature point identifier. Data fields of higher dimension (vectors,tensors etc...) are treated in an analogous manner.

The results of the interpolation are stored in a vtkFieldData array with "_QP_interpolated" appended to its original name. A vtkIDType array is also generated with "_QP_Indexes" appended to its name. This array maps cell id to the index into the interpolated data of the points associated with that cell. Having the indexes allows random access to the data for the quadrature points of a specific cell. Note that the interpolated data and index arrays are directly accessible through ParaView's spreadsheet view. All computations are made with double precision.

## 2.4 QUADRATURE POINT STATISTICS

The vtkQuadraturePointStatistics class is a filter that takes an unstructured grid data set containing scalar and vector fields that have been interpolated to the quadrature points (as described in section 2.3), and computes the minimum, maximum and mean over each of these. The results are placed into a vtkTable data set on its output. One column in the table is produced for each interpolated scalar field and four columns are produced for each interpolated vector field. For the vector fields the four columns produced are: the statistics computed on the norm of the vector, followed by the statistics computed on the first, second and third components of the vector. All computations are made with double precision.

## 2.5 QUADRATURE POINT POINT SET GENERATION

The vtkQuadraturePointsGenerator class is a filter taking an unstructured grid with quadrature scheme dictionaries embedded into its data arrays and produces a poly data data set containing vertices placed at the quadrature points for each cell. The filter will only process data arrays of type float or double, other types of data are ignored. The filter will only process a single array at a time. For the selected array, a set of quadrature points will be generated by interpolating each cell's nodal coordinates to the quadrature points as defined by its corresponding cell type's quadrature scheme definition. The interpolation is made as follows:

$$\vec{x}_j = \sum_{i=1}^{n} w_{ij}\vec{x}_i, \tag{3}$$

where $i$ is the node identifier, $j$ is the quadrature point identifier. After generating the points, we generate a set of vertices and insert them into the filter's output and if an interpolated array is present, it is set as point data. All computations are made with double precision.

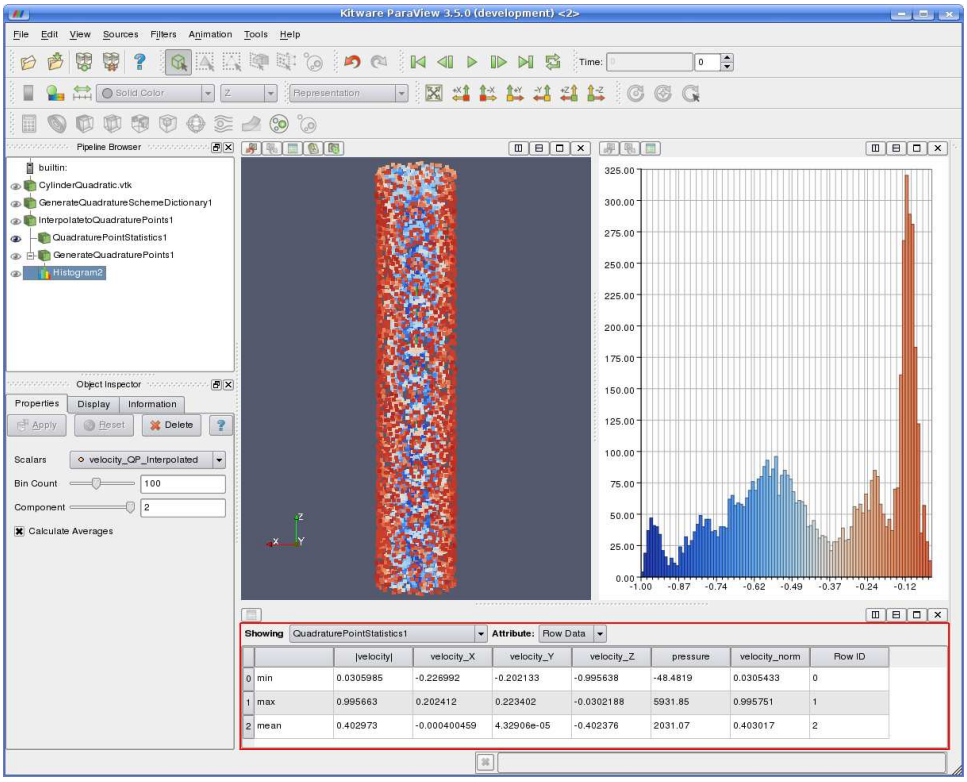


Figure 2: The new filters have been exposed in ParaView. This screen captures shows vector data interpolated to a fictional set of quadrature points, a table of basic statistics that has been computed on the interpolated fields, and a histogram generated using an interpolated data field.

# 3 DEMONSTRATION

## 3.1 REGRESSION TEST

We successfully demonstrate the following algorithms,

- Warp by Vector or Scalar.
- Geometry subset (clip) via extract geometry.
- Geometry subset via scalar threshold.

in the regression test named TestQuadraturePoints. This test loads an unstructured grid data set from the VTKData repository, creates a default set of dictionaries, writes the modified data to disk using the XML writer, reads the file back into memory using the XML reader, warps the data by a vector field, extracts resulting geometry which lies above a plane passing through its axis, thresholds the resulting geometry by a scalar value, then interpolates the data fields to quadrature points and finally generates the quadrature point point set, rendering glyphs and a representation of the warped surface. The source code for the test can be found in *VTK/Graphics/Testing/Cxx/TestQuadraturePoints.cxx* .

The test can be ran interactively with the following command: */path/to/vtk-build/bin/GraphicsCxxTests TestQuadraturePoints -D /path/to/VTKData -T ./ -I*. The results of the pipeline are shown in figure 1.

We successfully demonstrate the computation of basic statistics on scalar and vector fields interpolated to quadrature points with the regression test named TestQuadraturePointStatistics. This test loads an unstructured grid data set from the VTKData repository, creates a default set of dictionaries, then interpolates the data fields to quadrature points, and finally computes statics on the interpolated fields. Comparisons are made between computed results and known values. The source code for the test can be found in *VTK/Graphics/Testing/-Cxx/TestQuadraturePointStatistics.cxx*.

### 3.2 PARAVIEW

The new functionality has been exposed inside ParaView. The four filters introduced can be accessed via the filters menu under the alphabetical section. The filter vtkQuadraturePointInterpolator is exposed under the name "Interpolate to Quadrature Points", the filter vtkQuadraturePointsGenerator is exposed under the name "Generate Quadrature Points", the filter vtkQuadratureSchemeDictionaryGenerator is exposed under the name "Generate Quadrature Scheme Dictionary", and the filter vtkQuadraturePointStatistics is exposed under the name "Quadrature Point Statistics". In order to view the results of the vtkQuadraturePointStatistics filter, after connecting it to the pipeline, one must split ParaView's view panel, by adding a "Spreadsheet View". The table produced will then be displayed when its visibility is toggled on. Figure 2 shows the four new filters in action as vector data interpolated to a fictional set of quadrature points.

## 4 CONCLUSION

We have successfully added support for quadrature point visualization into VTK and have demonstrated the following algorithms:

- Geometry warp by Vector or Scalar.
- Geometry subset (clip) via extract geometry.
- Geometry subset via scalar threshold.
- Computation of basic statistics over interpolated fields.

In addition we have added quadrature scheme dictionary support to the VTK pipeline and XML data readers and writers. The new functionality has been successfully demonstrated via a set of regression tests.