Published in : International Journal of Computational Fluid Dynamics (2019), vol. 33, n°04, pp. 171–180
DOI: 10.1080/10618562.2019.1618453
Status : Postprint (Author's version)

SCIENTIFIC *POST HOC* AND *IN SITU* VISUALISATION OF HIGH-ORDER POLYNOMIAL SOLUTIONS FROM MASSIVELY PARALLEL SIMULATIONS

M. Rasquin[a], A. Bauer[b] and K. Hillewaert[a]

[a]*Cenaero, Gosselies, Belgium;*

[b]*Kitware, Inc., Clifton Park, NY, USA*

Keywords:

Parallel scientific visualisation; post-processing; *in situ* analysis and visualisation; high-order finite element; IO

Abstract

This paper presents two recent numerical tools developed respectively to perform traditional post-processing and more advanced *in situ* processing of high-order polynomial data generated by massively parallel finite element codes. For post-processing and visualisation of high-order solutions, we present a new ParaView plugin that integrates Gmsh used as an external library. This plugin therefore combines respectively ParaView's scalability in parallel and Gmsh's ability to apply h-refinement of the initial mesh followed by solution interpolation on the resulting visualization grid, thus enabling parallel visualisation of any arbitrary high-order polynomial solutions in client–server mode. In a second stage, this capacity has been extended to an *in situ* interface based on the Catalyst library which enables *in situ* analysis and visualisation of high-order solutions. These new capacities are demonstrated with the visualisation of high-order solution of the unsteady flow generated by a discontinuous Galerkin method for an unsteady turbomachinery application.

1. Introduction

Recently high-order finite element methods such as discontinuous Galerkin (DG) (Cockburn and Shu 1998; Cockburn 1999; Cockburn, Karniadakis, and Shu 2000) and flux reconstruction (FR) (Huynh 2007; Williams et al. 2013; Witherden, Vincent, and Jameson 2016) methods have gained considerable attention, not only in the research community but also in industry. This rising interest is due to both their high accuracy on unstructured meshes, typically associated with complex geometries, their efficiency and scalability. These properties make these methods extremely well suited for scale resolving simulations of industrial flows.

However, as these high-order methods become more and more popular, new challenges related to the development of efficient and complete data workflows arise. These challenges include high-order

Published in : International Journal of Computational Fluid Dynamics (2019), vol. 33, n°04, pp. 171–180
DOI: 10.1080/10618562.2019.1618453
Status : Postprint (Author's version)

mesh generation, combined geometric (h-) and polynomial (p-) refinement, and visualisation of high-order solutions to name a few. In particular, the lack of general purpose, production quality visualisation tools able to handle a large number of degrees of freedom (dof) has been a major bottleneck so far for the analysis of high-order finite element solutions at large scale. This paper aims to bring a contribution to address this challenge.

In terms of *post hoc* visualisation, efficient opensource parallel visualisation tools such as ParaView (Ahrens, Geveci, and Law 2005) and VisIt (Childs et al. 2012) have been used successfully for the visualisation of large-scale, unstructured data sets up to several billions of dof . Historically, mainly linear interpolation and sporadically quadratic functions have been supported so far in the Visualization Toolkit (VTK) (Schroeder, Lorensen, and Martin 2004) whose data models are extensively used by ParaView and VisIt. Only very recently, arbitrary-order Lagrange polynomials have been introduced in VTK and their support in ParaView (Corona and Thompson 2018) for both high-order meshes and solutions is just gaining traction, which confirms the growing interest and need for such features.

As an alternative, a general method for the post-processing of high-order finite element fields has been proposed in Remacle et al. (2007) and implemented in the open-source mesh and visualisation tool Gmsh (Geuzaine and Remacle 2009). This method relies on recursive h-refinements of the initial mesh, coupled to a projection error estimation. The high-order solution is then interpolated at the new mesh nodes in order to provide an optimal visualisation grid. However, Gmsh is in essence a serial tool and is therefore limited to coarse meshes.

Independently from the order of the simulation, the rise of petascale and upcoming exascale computing systems will require a different paradigm than the one provided by traditional *post hoc* visualisation tools in order to deal with the massive increase in data size produced by these systems. Indeed, it is already well established in the HPC community that traditional *post hoc* visualisation will likely become unsustainable due to the growing gap between IO bandwidth and computational power (Ali et al. 2009; Vishwanath, Hereld, and Papka 2011). Concretely, writing to disk intermediate output data will become more and more expensive and will have to occur at a lower frequency, preventing fine analysis of complex transient or unsteady phenomena. In this context, *in situ* tools allow real-time data analysis and visualisation without involving storage resources as data are generated by massively parallel simulations. A description of various *in situ* systems is provided by the survey papers (Heiland and Baker 1998; Mulder, vanWijk, and vanLiere 1999; Rivi et al. 2012; Bauer et al. 2016). Furthermore, this paradigm allows a finer monitoring and interaction with running simulations since the influence of certain key parameters can be investigated live and not *a posteriori*. Several *in situ* techniques have already been presented in the literature and demonstrated at scale for linear polynomials (see for instance Fabian et al. 2011; Rasquin et al. 2011; Yi et al. 2014). Naturally, these approaches can also apply to high-order simulation codes.

In this paper, two capabilities to perform respectively traditional post-processing and more advanced *in situ* visualisation of high-order polynomial data are presented. In practice, these data are generated by a massively parallel high-order discontinuous Galerkin CFD code named Argo developed at Cenaero but the tools and approaches presented in this paper apply to any high-order polynomial data.

Published in : International Journal of Computational Fluid Dynamics (2019), vol. 33, n°04, pp. 171–180
DOI: 10.1080/10618562.2019.1618453
Status : Postprint (Author's version)

For offline post-processing and visualisation of high-order solutions, we present a new ParaView plugin that integrates Gmsh as an external library. This plugin therefore combines respectively ParaView's scalability in parallel and Gmsh's ability to apply h-refinement of the initial mesh followed by the interpolation of any arbitrary high-order polynomial solutions on the resulting visualisation grid. This coupling enables parallel visualisation of high-order solution in parallel, in client–server mode.

In a second stage, this capacity has been extended to an interface based on the Catalyst library (Fabian et al. 2011), which enables *in situ* analysis and visualisation of high-order solutions generated by Argo. *In situ* analysis and visualisation with Catalyst can be performed in batch mode and generate for instance images at a high frequency. It can also enable live computational monitoring once it is connected to a ParaView server running concurrently on another visualisation resource.

This paper is organised as follows. Section 2 describes the ParaView plugin developed for high-order solutions. Section 3 presents an interface to the *in situ* library Catalyst dedicated to high-order solutions. Section 4 concludes this paper.

2. Post-processing of high-order solution

2.1. TOOLS AND IMPLEMENTATION

Historically, Gmsh (Geuzaine and Remacle 2009) has been one of the very few open-source tools to provide high-order meshing (including curve boundary layer and mesh optimisation) and visualisation capabilities. A major asset of the 'msh' file format used by Gmsh lies in a flexible representation of any high-order polynomial function associated to usual 2D and 3D elements through the specification of the coefficients and exponents of each of its terms. Lagrange, Legendre or any other polynomials can therefore be represented within the same framework.

The visualisation capabilities of Gmsh are described in Remacle et al. (2007) and rely on the recursive h-refinement of high-order elements based on classical AMR (Automatic Mesh Refinement), followed by the interpolation of the associated high-order polynomial solution on the new nodes of the refined visualisation grid. Once these two steps are complete, linear visualisation of the solution is used through a straightforward piecewise linear interpolation on the refined visualisation grid.

The recursive h-refinement of the initial mesh which generates the visualisation grid can be uniform, that is every edge of the parent element is split in the same way. Besides uniform h-refinement, a second asset of Gmsh consists in a selective h-refinement procedure based on a local visualisation error. This visualisation error controls and applies selectively h-refinements to the parent elements and their child elements until this error is locally minimised. Consequently, this selective h-refinement procedure can reduce significantly the memory requirement compared to uniform h-refinement. In order to accomplish this selective h-refinement, each new child element at a given h-refinement level is treated independently from its neighbours, that is no interior connectivity linking child elements within their parent element is constructed.

Published in : International Journal of Computational Fluid Dynamics (2019), vol. 33, n°04, pp. 171–180
DOI: 10.1080/10618562.2019.1618453
Status : Postprint (Author's version)

Besides these attractive features, the main limitations of Gmsh are its serial implementation and relatively heavy data structure, which makes this tool suitable only for the visualisation of coarse meshes and a limited number of recursive h-refinements.

The second tool considered in this work is ParaView (Ahrens, Geveci, and Law 2005) which has been chosen for the following features: scalability, ability to handle billions of *dof* associated with unstructured meshes, rich library of available data operations (e.g. isosurface, streamlines, etc.), ease to implement plugins and new data readers, and last but not least the Catalyst library which provides live *in situ* visualisation capabilities. Although it has not been tested yet within this work, the support for arbitrary-order Lagrange polynomials recently introduced in ParaView (Corona and Thompson 2018) opens up additional promising perspectives for the visualisation of high-order meshes and solutions. Note that contrary to Gmsh, only Lagrange polynomials are currently supported in VTK and ParaView applies in all cases uniform h-refinement to visualise these polynomials.

The adopted solution to enable efficient *post hoc* visualisation of high-order data generated by massively parallel simulations consists in coupling the assets of both Gmsh and ParaView through the implementation of a new ParaView plugin able to read data saved under the msh format. For that purpose, a new ParaView plugin named GmshReader has been implemented and ported in a first step to the official ParaView source code (version 5.6.0 and above). In a second step, it will be distributed with the official ParaView binary without the need for the user to compile ParaView from scratch. Note that this plugin links to a light version of Gmsh compiled as an external library in order to limit its memory consumption. This light version of Gmsh only includes the internal msh reader along with the h-refinement and interpolation procedures customised for this particular application. Since Gmsh is licensed under GPL, an exception has been added to the Gmsh license so that it can be statically linked to ParaView under BSD without propagating its GPL license.

This plugin uses the same h-refinement and interpolation functions in Gmsh in order to generate a visualisation grid so that it follows exactly the same two-step procedure described above. During this procedure, the plugin also generates a linear VTK data structure that is passed to ParaView for linear visualisation.

The main challenge in the implementation of the coupling between both Gmsh and ParaView comes from the flexible definition of a 'view' in Gmsh. A view in Gmsh is a field associated with a physical variable (e.g. velocity), characterised by a polynomial representation (e.g. third-order Lagrange p3) and potentially located on a given mesh partition. A view can therefore be defined partially over the computational domain if variable order solutions have been generated by the solver. For instance, third-order p3 views can be defined around a wing profile whereas reduced-order p2, p1, and p0 views are defined towards the outlet of the computational domain in order to limit spurious wave reflection induced by the boundary condition. Each view in Gmsh can also be refined independently based on its own local visualisation error if selective h-refinement is requested, leading to a different visualisation grid for each view.

On the other hand, ParaView relies on point (and cell) fields which preferably spread over the whole domain. Consequently, each view in Gmsh follows the two-step procedure described above and applies successively h-refinement followed by an interpolation of the high-order solution to its corresponding visualisation grid. Then the GmshReader plugin merges the views of the same variable

Published in : International Journal of Computational Fluid Dynamics (2019), vol. 33, n°04, pp. 171–180
DOI: 10.1080/10618562.2019.1618453
Status : Postprint (Author's version)

(e.g. velocity) but of different order (e.g. p3, p2, p1 and p0) defined partially over the computational domain to generate single point fields defined over the whole domain. If several variables (e.g. velocity, pressure, etc.) need to be processed, the resulting point fields in the VTK data structure are grouped together and added to the respective mesh points for visualisation purpose in ParaView.

If a single point field is processed by the plugin, both uniform and selective h-refinements based on the local visualisation error implemented in Gmsh are available to the user. Alternatively, if several point fields are processed together, only uniform h-refinement is currently allowed in order to generate a unique visualisation grid for all requested fields. Otherwise, the visualisation error implemented in Gmsh and defined for each field would lead to several distinct visualisation grids which cannot be handled conveniently in ParaView (e.g. isosurface of one field coloured by another field).

Another important feature for the visualisation of high-order data at large scale is the reduction of the memory footprint of the child elements generated during the h-refinement procedure. Indeed, the default h-refinement approach used in Gmsh treats each child element independently, which leads to duplicate mesh nodes on the edges and faces shared by child elements inside their parent element. During the generation of the VTK data structure for ParaView, duplicate mesh nodes are removed and a connectivity internal to each parent element is built. Note that this connectivity does not involve neighbouring parent elements so that all mesh nodes located on edges and faces shared by two neighbouring parent elements remain distinct.

Without this internal connectivity, the number of new mesh nodes would be simply equal to the number of new child elements times the number of nodes per element. This general approach is therefore fully compatible with various high-order methods, including discontinuous Galerkin methods which is the target of this work.

Finally, the capability of the GmshReader plugin is illustrated in **Figure 1** with the 2D flow past a circular cylinder at Reynolds number 100. The p4 vorticity field is shown in the top panels whereas the associated visualisation grids are shown in the bottom panels for several uniform h-refinement levels of the initial mesh. The initial mesh includes 1438 p4 triangles which leads to a total of 21,570 dof with 15 dof per triangle. As shown in Figure 1, two levels of h-refinement are typically required in order to ensure a sufficient level of convergence for the considered field. In this example, two levels of uniform h-refinement result in 23,008 linear triangles and 21,570 nodes in the visualisation grid. In practice, this statement can be extended to more complex 3D visualisation, as illustrated in the next section.

Note that the number of linear triangles and nodes could have been further reduced if selective h-refinement had been applied for the visualisation of this single field. Since the processing of several fields on the same mesh nodes is often desired in our applications, only uniform h-refinement has been illustrated in this section but other examples with selective h-refinement are provided in Remacle et al. (2007).

The linear representation of the cylinder contour is also improved as the number of h-refinement levels increases, which highlights the projection of the new mesh nodes on the boundary surfaces that occurs in Gmsh after each h-refinement.

Published in : International Journal of Computational Fluid Dynamics (2019), vol. 33, n°04, pp. 171–180
DOI: 10.1080/10618562.2019.1618453
Status : Postprint (Author's version)

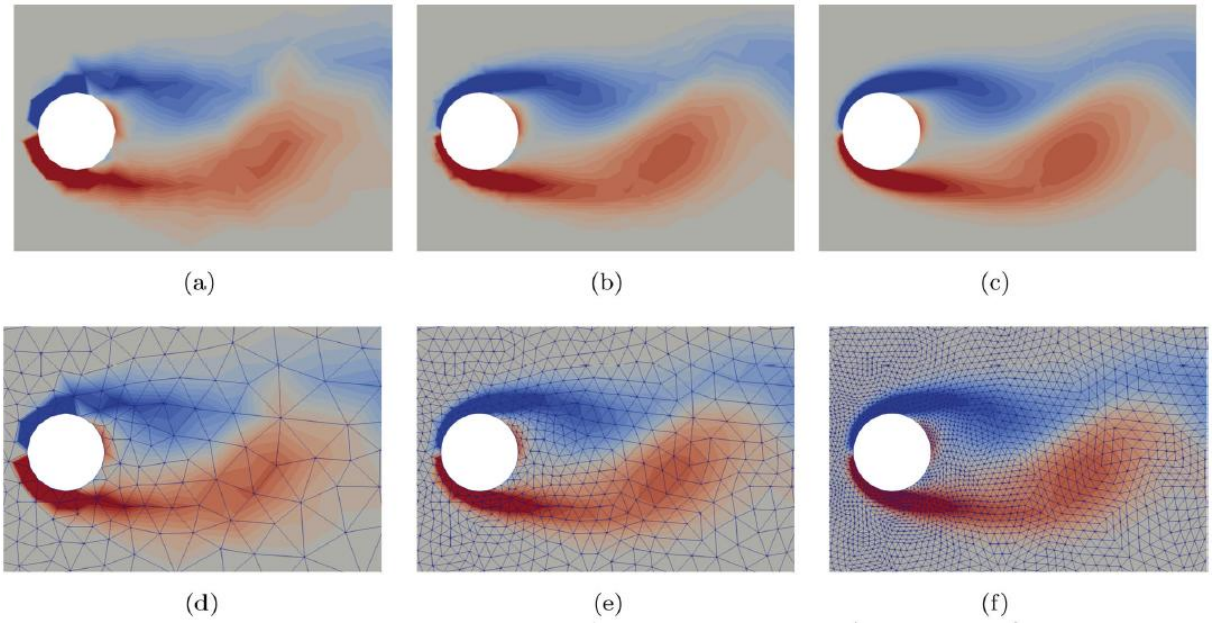

Figure 1. Illustration of a visualisation grid generated by the GmshReader plugin with the p4 velocity field past a circular cylinder at Reynolds number 100: (a) vorticity field for the initial mesh with no refinement, (b) vorticity field for the first h-refinement level, (c) vorticity field for the second h-refinement level, (d) initial visualisation grid with no refinement (1438 triangles and 4314 nodes), (e) visualisation mesh for the first h-refinement level (5752 triangles and 8628 nodes), (f) visualisation mesh for the second h-refinement level (23,008 triangles and 21,570 nodes).

2.2. FILE FORMAT

As described in Childs et al. (2010), *post hoc* visualisation is mainly IO bound. Special care must therefore be brought to the file format used on one hand by the massively parallel solver when it writes its output files, and on the other hand the file format used by the visualisation software when it reads back these output files. Unfortunately, these two constraints can sometimes be in opposition to each other.

Note that a view in Gmsh defined in Section 2.1 is described by a header followed by a binary data block. The header includes information such as the variable name, the polynomial representation, the partition id, etc. As a first level of data parallelism, each physical variable computed by Argo is also stored in a distinct file or set of files.

The legacy file format used in Gmsh is the msh v2 format. Historically, this format follows two modes which apply to both mesh files and solution files. However, since the mesh is only read once, we will focus on the output files which are written on a regular basis from the parallel solver and read back from the visualisation tool. A third mode has been recently added in order to limit the drawbacks of the first two modes.

- The 'single file' mode merges the data divided across all partitions in single views associated to a given polynomial representation and stored in a single file. This mode is only viable for small-scale computations. Although it is possible to use parallel IO strategies (e.g. MPI-IO) to write efficiently this single file from massively parallel simulations, this mode prevents an

Published in : International Journal of Computational Fluid Dynamics (2019), vol. 33, n°04, pp. 171–180
DOI: 10.1080/10618562.2019.1618453
Status : Postprint (Author's version)

efficient usage of a parallel ParaView server since the output data associated to a given partition cannot be extracted solely without reading the whole file.

- The 'separate file' mode writes all views associated to a given physical variable in a separate file per partition. With this mode each process of the simulation creates a file that no other process accesses. However, this mode does not scale well to very large process counts where the number of files created by the solver can quickly saturate the metadata server of the file system. Moreover, managing the resulting large number of files can quickly become another bottleneck. Nevertheless, a visualisation server usually runs with a number of processes typically 2 or 3 orders of magnitude smaller than the simulation. For the visualisation server, this format prevents file or block locking on the file system level since each file is accessed at most by a single process. Moreover, the unstructured data readers in ParaView rarely use parallel IO strategies and rely mainly on independent parallel reads, typically associated to a given partition. This limitation is currently shared by most data readers in ParaView and contributes to the flexibility of the software in terms of arbitrary number of processes.
- A more recent export mode dubbed 'separate views' consists in concatenating all views of the same physical variable divided across all partitions into a single file without merging them. In this case, each view still corresponds to a distinct mesh partition and polynomial representation. As the writing in the file can be coordinated using MPI-IO, this version provides superior writing performance than the 'separate files' mode from the perspective of a massively parallel simulation. Each view header contains also the size of its corresponding data block so that a ParaView process can quickly jump over a data block when searching for a given partition data. In contrast, reading performance from a ParaView server is deteriorated since several servers have to access independently and not collectively the same file. This compromise is nevertheless necessary to write output data from massively parallel simulations as quickly as possible and limit the high CPU time associated with this operation.

Finally, an XML file is used as an interface between the msh files and the plugin (see the manual in the GmshReader repository for more details). In this file are specified the number of partition data, the paths to the mesh and solution data, the number of h-refinement levels and eventually the value of the visualisation error if applicable.

Note that a new format msh v4 is currently being implemented. Contrary to the msh v2 format, the msh v4 format is based from scratch on a block partitioning structure. So far, the M-to-1 (all partition data saved in one file) and M-to-M (each partition data saved in separate files) modes have been considered in the msh v2 format. The new msh v4 format will also offer the possibility to use the M-to-N mode where a subset of partitions are stored per file. An additional improvement includes a proper offset table to limit file access. A high-order extension of the CGNS format based on HDF5 is also currently under development and will facilitate the integration of high-order tools into traditional workflows. One last improvement consists in the implementation of collective IO access to the output files from the visualisation software but this requires a much deeper reorganisation of the visualisation software.

Published in : International Journal of Computational Fluid Dynamics (2019), vol. 33, n°04, pp. 171–180
DOI: 10.1080/10618562.2019.1618453
Status : Postprint (Author's version)

2.3. PERFORMANCE

The performance of the plugin is investigated with the visualisation of the flow field around the full span MTU T161 low pressure turbine blade cascade. This test case was chosen as part of the H2020 European research project Tilda, which aims to develop innovative methods combining advanced and efficient high-order numerical schemes for the aeronautics industry.

This flow is illustrated in **Figure 2** with six views of a vorticity isosurface pseudo-coloured by velocity magnitude. This test case includes 14.9 million hexahedra partitioned in 4800 mesh parts and the solution is represented by p3 polynomials, leading to about 0.96 billion dof. Two levels of h-refinement are applied in **Figure 2**. The visualisation system Cooley located at the Argonne Leadership Computing Facility was used for these tests. Each Cooley node includes two 2.4∼GHz Intel Haswell E5-2620 v3 processors per node (six cores per CPU, 12 cores in total) and one NVIDIA Tesla K80 (with two GPUs) per node.

Table 1 summarises the time spent in the Gmsh Reader plugin for zero, one and two levels of uniform h-refinement, and for both the separate files and separate view modes. These tests used 2, 4 and 8 visualisation nodes with 24, 48 and 96 cores, respectively (one ParaView server per core).

For the separate files mode, the time spent in the plugin is roughly inversely proportional to the number of visualisation nodes. This was expected since each ParaView server accesses independently its respective files without file contention. The time spent in the plugin increases by a factor 1.4 and 3.8 for respectively one and two levels of h-refinement respective to the configuration with no refinement. This cost is due to the recursive implementation of the h-refinement functions in Gmsh which allow a selective h-refinement based on a visualisation error. These functions are therefore not optimised for uniform h-refinement and could be accelerated for this specific usage.

For the separate views mode, the time spent in the plugin is in most cases 1 order of magnitude larger than for the separate files mode, which is a clear result of file or block locking at the file system level. A second observation is linked to the variability of the time spent in the plugin among the processes of the visualisation server, which is difficult to analyse and is thought to be another consequence of file contention. For two levels of h-refinement, the best result is obtained on 4 visualisation nodes with 48 servers, which is 'only' 3.6 times slower respective to the separate views mode. The improvements listed in the previous section are currently investigated in order to reduce the gap between the separate files and separate views modes, which is critical for data saved from massively parallel simulations.

In **Table 2**, the benefit of the construction of an internal connectivity after the h-refinement of the parent elements is highlighted. After one and two levels of h-refinement, the ratio between ParaView points and cells drops respectively to 3.38 and 1.95 for 119 and 956 million hexahedra, whereas this ratio would remain equal to eight if child elements at each refinement level were treated independently.

Published in : International Journal of Computational Fluid Dynamics (2019), vol. 33, n°04, pp. 171–180
DOI: 10.1080/10618562.2019.1618453
Status : Postprint (Author's version)

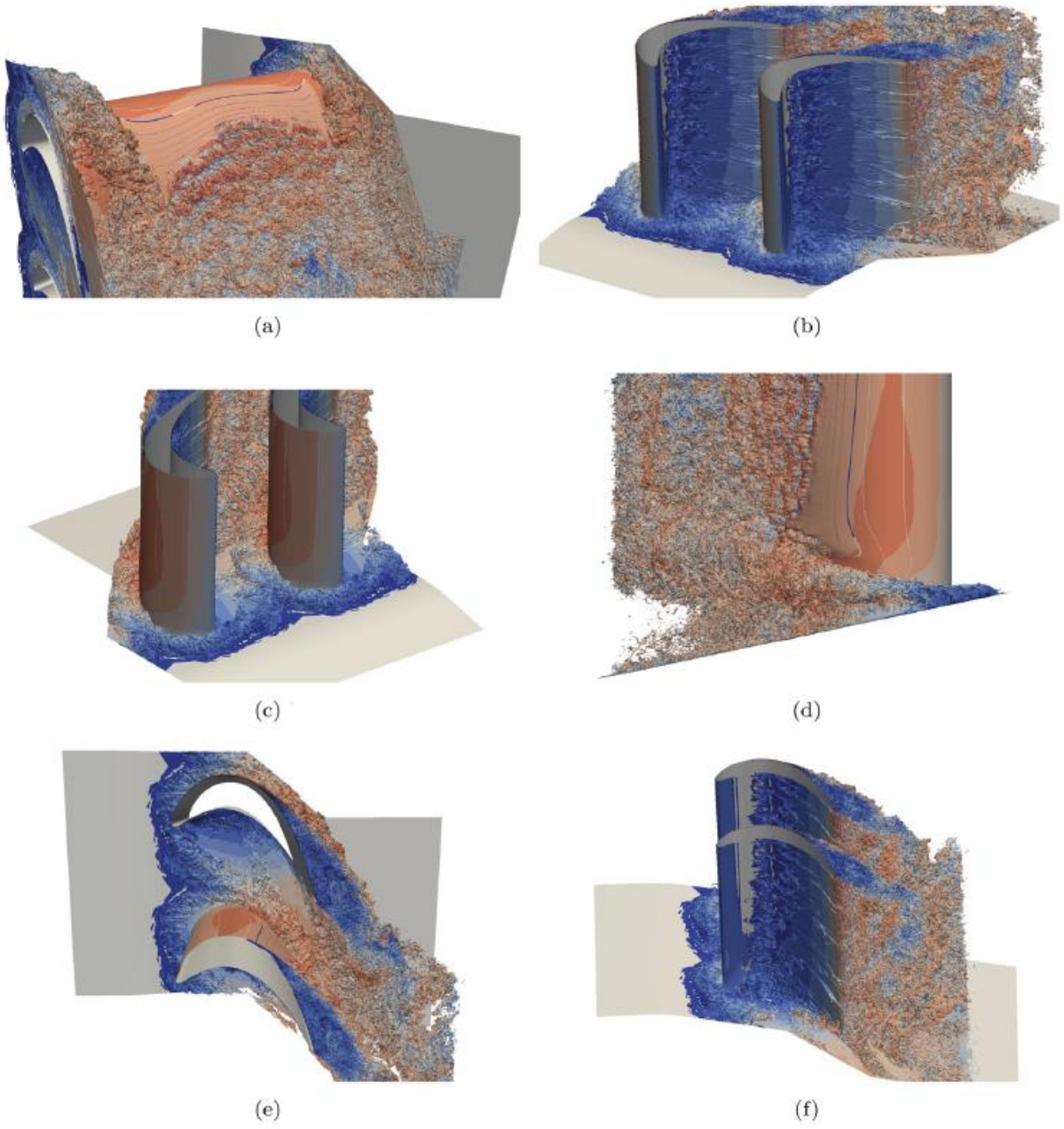

Figure 2. Illustration of secondary flows around the full span MTU T161 low pressure turbine blade cascade with an isosurface of the p3 vorticity field pseudo-coloured by velocity magnitude: (a) View 1, (b) View 2, (c) View 3, (d) View 4, € View 5, (f) View 6.

Published in : International Journal of Computational Fluid Dynamics (2019), vol. 33, n°04, pp. 171–180
DOI: 10.1080/10618562.2019.1618453
Status : Postprint (Author's version)

Table 1. Performance of the GmshReader plugin for the visualisation of the MTU T161 turbine blade with 14.9 million hexahedra and p3 representation of the solution fields. The minimum and maximum time among all visualisation servers are reported in seconds between square brackets.

		Refinement level		
	Vis. nodes	0	1	2
Separate files	2	[158.1 173.7]	[220.5 225.5]	[518.6 536.2]
	4	[65.8 75.4]	[90.6 100.5]	[261.3 274.5]
	8	[32.9 37.5]	[45.0 50.8]	[130.0 142.9]
Separate views	2	[1251.9 1276.5]	[3728.6 5156.2]	[2111.9 3003.5]
	4	[789.8 1008.4]	[2245.3 2878.2]	[909.3 996.9]
	8	[274.4 1063.2]	[355.2 1160.1]	[469.1 4023.5]

Table 2. Characteristics of the visualisation grid for different levels of h-refinement applied to the T161 application with 14.9 million hexahedra included in the initial mesh.

	Level 0	Level 1	Level 2
ParaView cells	14,945,700	119,565,600	956,524,800
ParaView points	119,565,600	403,533,900	1,868,212,500
Ratio points/cells	8	3.38	1.95

3. *Insitu* analysis and visualisation

3.1. IMPLEMENTATION

Although *post hoc* visualisation tools able to display high-order polynomials solutions have become a requirement, *in situ* analysis and visualisation tools are the only viable solution able to extract continuous insights from high-fidelity massively parallel simulations. For that purpose, an interface between Argo and the Catalyst library has been implemented. This interface uses the same concepts of h-refinement of the initial mesh followed by the interpolation of high-order solutions on the visualisation grid. The main difference with the implementation of the ParaView plugin is that only uniform h-refinement of the parent element is implemented because several fields often need to be processed at the same time. Since all elements present in the compute mesh are grouped according to their topology in Argo, the parent element of a given topology is refined only once in its parametric space and the values of the shape functions at the nodes of the resulting visualisation grid are computed only once for all elements of the same topology. This strategy allows the use of a constant projection matrix for the interpolation of high-order solutions on the visualisation grid pattern, which can be performed efficiently with optimised BLAS functions. This feature could also be ported back to

Published in : International Journal of Computational Fluid Dynamics (2019), vol. 33, n°04, pp. 171–180
DOI: 10.1080/10618562.2019.1618453
Status : Postprint (Author's version)

the ParaView plugin in order to reduce significantly the current cost associated with a uniform h-refinement of the initial mesh in Gmsh.

3.2. PERFORMANCE

This Catalyst interface was tested in production mode on the MareNostrum supercomputer located at Barcelona Supercomputing Center as part of a PRACE allocation granted during the 15th call. Similar to the ParaView plugin tests, the MTU T161 test case with the same resolution was again considered. For these tests, 19,200 Intel Platinium cores were used with a total of 4800 MPI processes and 4 threads per MPI. One and six views already illustrated in Figure 2 were considered for these tests.

The performance of the Argo solver coupled to Catalyst is summarised in **Table 3**. The resolution of an implicit time step without Catalyst requires on average 20.8 s. The time spent to perform two levels of uniform h-refinement and the corresponding high-order interpolation is equal to 0.23 s. The Catalyst *in situ* time is equal to 1.36 s and 4.1 s for one and six views, respectively. This *in situ* time includes data processing, image rendering and compositing, and image saving. Note that the resolution for each picture was set to 1920 × 1080 pixels, which also plays a significant role in the performance of the *in situ* workflow. Indeed, more requested pixels require more *in situ* processing time. The visualisation overhead per time step is therefore 7.6% and 21.2% for respectively one and six views if *in situ* processing occurs every time step. In practice for this case, *in situ* processing was applied every four time steps which still provides enough temporal resolution to generate smooth movies. In this case, the visualisation overhead drops to respectively 1.9% and 5.3% for one and six views, which is deemed an acceptable cost with respect to the benefit of extracting continuous insights from the simulation.

Table 3. Performance of the Catalyst library coupled with Argo on the turbine MTU T161 test case.

	one view	six views
Argo solver (1 implicit time step)	20.8 s	
Mesh refinement + HO interpolation	0.23 s	
Catalyst *in situ* processing	1.36 s	4.1 s
Visualisation overhead per time step	7.6%	21.2%
Visualisation overhead per every 4 time steps	**1.9%**	**5.3%**

4. Conclusion

An open-source ParaView plugin coupled to the Gmsh library named GmshReader has been implemented for the parallel visualisation of any arbitrary high-order polynomial solution saved under the msh v2 format. The approach considered in this work relies on the recursive h-refinement of the initial mesh to generate a visualisation grid, followed by the interpolation of the high-order solution on the visualisation grid. The h-refinement in Gmsh can be uniform or selective based on a local visualisation error. Selective h-refinement can reduce significantly the memory requirements of the visualisation tool compared to uniform h-refinement. However, it is currently limited to the processing

Published in : International Journal of Computational Fluid Dynamics (2019), vol. 33, n°04, pp. 171–180
DOI: 10.1080/10618562.2019.1618453
Status : Postprint (Author's version)

of a single field whereas multiple fields can be handled with uniform h-refinement. The resulting visualisation grid and associated interpolated solution can then be efficiently visualised with traditional linear visualisation tools.

This plugin has been tested with the visualisation of large-scale data including about 0.95 billion degrees of freedom generated from a massively parallel high-order discontinuous Galerkin CFD solver named Argo developed at Cenaero. For such data partitioned in several thousands of mesh parts, the msh v2 format can be derived in two modes, namely the separate files mode with one file per mesh part, and the separate views mode where partitioned data are saved in separate blocks in a single file. While the separate files mode does not scale well for the massively parallel solver due to the saturation of the metadata server of the file system, this mode still offers the best IO performance for the visualisation software. On the contrary, the separate views mode offers the possibility to write efficiently partitioned data to disks from massively parallel simulations using collective IO functions (e.g. MPI-IO). However, this single file approach leads to file and/or block locking when multiple instances of the visualisation software access the same output file independently to load their respective data. Increasing the number of files saved under the separate views mode with a subset of partitioned data per file is one solution to reduce file contention, which is currently investigated as part of the development of the msh v4 format. A high-order extension of the CGNS format based on HDF5 is another solution currently under development which will facilitate the integration of high-order tools into traditional workflow. Ideally, the best IO performance would be obtained if the visualisation software could also rely on collective IO access to load back the data.

An interface to the *in situ* library Catalyst has also been implemented for Argo. This interface enables *in situ* analysis and visualisation of high-order solutions. This interface relies on the same approach as the one implemented for the ParaView plugin, although only uniform h-refinement is available for this application through efficient projection of the high-order solution on the visualisation grid using BLAS functions. Tests at scale on 19,200 cores of the MareNostrum supercomputer revealed an overhead of 1.9% and 5.3% for the generation of respectively one and six images, which is an acceptable cost with respect to the benefit of extracting continuous insights from a massively parallel simulation.

Furthermore, the recent introduction of arbitrary high-order Lagrange polynomials in VTK and their support in ParaView pave the way to further CPU time and memory requirements reduction. Although ParaView currently relies on uniform h-refinement of high-order meshes, only the elements directly involved in a given data operation could be selectively refined in the future (e.g. elements intersected by a slice or an isosurface), reducing significantly the memory consumption. Finally, for *post hoc* visualisation with the GmshReader plugin, uniform h-refinement specifically applied to Lagrange elements could be further accelerated in ParaView compared to the current flexible and recursive algorithm based on a local visualisation error in Gmsh.

Acknowledgements

The authors acknowledge PRACE (Partnership for Advanced Computing in Europe) for awarding us access to MareNostrum hosted at the Barcelona Supercomputing Center (BSC), Spain. Another award

Published in : International Journal of Computational Fluid Dynamics (2019), vol. 33, n°04, pp. 171–180
DOI: 10.1080/10618562.2019.1618453
Status : Postprint (Author's version)

of computer time used for this work was provided by the Innovative and Novel Computational Impact on Theory and Experiment (INCITE) program. This research used the visualisation resource Cooley of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357. Last but not least, Mathieu Westphal and Joachim Pouderoux from Kitware are gratefully acknowledged for their help to integrate the plugin in the ParaView sources.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

The first author was supported by a BEWARE fellowship (industry 2014_1) from the Walloon Region in Belgium under grant agreement no. 1410131 (project Atalanta). This project has received funding from the European Union's Horizon 2020 research and innovation programme (H2020-EU.3.4 - Societal Challenges - Smart, Green And Integrated Transport) under grant agreement no. 635962 (project Tilda).

References

Ahrens James, Geveci Berk, and Law Charles. 2005. "Paraview: An End-User Tool for Large Data Visualization." *The Visualization Handbook* 717.

Ali Nawab, Carns Philip, Iskra Kamil, Kimpe Dries, Lang Samuel, Latham Robert, Ross Robert, Ward Lee, and Sadayappan Ponnuswamy. 2009. "Scalable I/O Forwarding Framework for High-Performance Computing Systems." In *IEEE International Conference on Cluster Computing and Workshops, 2009. CLUSTER'09* , 1–10. IEEE.

Bauer A. C., Abbasi H., Ahrens J., Childs H., Geveci B., Klasky S., and Moreland K., et al. 2016. "*In Situ Methods, Infrastructures, and Applications on High Performance Computing Platforms*." Computer Graphics Forum 35 (3): 577–597. https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12930.

Childs Hank, Brugger Eric, Whitlock Brad, Meredith Jeremy, Ahern Sean, Pugmire David, and Biagas Kathleen, et al. 2012. "VisIt: An End-User Tool for Visualizing and Analyzing Very Large Data." *High Performance Visualization–Enabling Extreme-Scale Scientific Insight*, 357–372. Childs Hank, Pugmire David, Ahern Sean, Whitlock Brad, Howison Mark, Weber Gunther H, and Bethel E. Wes, et al. 2010. "*Extreme Scaling of Production Visualization Software on Diverse Architectures*." IEEE Computer Graphics and Applications 30 (3): 22–31.

Cockburn Bernardo. 1999. "Discontinuous Galerkin Methods for Convection-Dominated Problems." In *High-Order Methods for Computational Physics*, 69–224. Springer.

Cockburn Bernardo, Karniadakis George E, and Shu ChiWang. 2000. "The Development of Discontinuous Galerkin Methods." In *Discontinuous Galerkin Methods*, 3–50. Springer.

Cockburn Bernardo, and Shu Chi-Wang. 1998. "*The Local Discontinuous Galerkin Method for Time-Dependent Convection - Diffusion Systems*." SIAM Journal on Numerical Analysis 35 (6): 2440–2463.

Corona Thomas J, and Thompson David. 2018. "*Arbitrary-Order Lagrange Finite Elements in the Visualization Toolkit*." Kitware Source Quarterly Magazine 43: 6–9.

Published in : International Journal of Computational Fluid Dynamics (2019), vol. 33, n°04, pp. 171–180
DOI: 10.1080/10618562.2019.1618453
Status : Postprint (Author's version)

Fabian Nathan, Moreland Kenneth, Thompson David, Bauer Andrew C, Marion Pat, Gevecik Berk, Rasquin Michel, and Jansen Kenneth E. 2011. "The Paraview Coprocessing Library: A Scalable, General Purpose In Situ Visualization Library." In *2011 IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, 89–96. IEEE.

Geuzaine Christophe, and Remacle Jean-François. 2009."*Gmsh: A 3-D Finite Element Mesh Generator with Built-in Pre-and Post-Processing Facilities*." International Journal for Numerical Methods in Engineering 79 (11): 1309–1331.

Heiland Randy, and Baker M. Pauline. 1998. *A Survey of Co-Processing Systems*. Technical Report. NCSA University of Illinois (USA). http://sda.iu.edu/docs/CoprocSurvey.pdf.

Huynh Hung T. 2007. "A Flux Reconstruction Approach to High-Order Schemes Including Discontinuous Galerkin Methods." In *18th AIAA CFD Conference*, 4079.

Mulder Jurriaan D., van Wijk Jarke J., and van Liere Robert. 1999. "*A Survey of Computational Steering Environments*." Future Generation Computer Systems 15 (1): 119–129. http://www.sciencedirect.com/science/article/pii/ S0167739X98000478.

Rasquin Michel, Marion Patrick, Vishwanath Venkatram, Matthews Benjamin, Hereld Mark, Jansen Kenneth, Loy Raymond, et al. 2011. "Electronic Poster: Co-Visualization of Full Data and in Situ Data Extracts from Unstructured Grid cfd at 160k Cores." In *Proceedings of the 2011 Companion on High Performance Computing Networking, Storage and Analysis Companion*, 103–104, ACM.

Remacle Jean-François, Chevaugeon Nicolas, Marchandise Emilie, and Geuzaine Christophe. 2007. "*Efficient Visualization of High-order Finite Elements*." International Journal for Numerical Methods in Engineering 69 (4): 750–771.

Rivi Marzia, Calori Luigi, Muscianisi Giuseppa, and Slavnic Vladimir. 2012. "In-Situ Visualization: State-of-the-Art and Some Use Cases." *PRACE White Paper* 1–18.

Schroeder Will J, Lorensen Bill, and Martin Ken. 2004. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Kitware.

Vishwanath Venkatram, Hereld Mark, and Papka Michael E. 2011. "Toward Simulation-Time Data Analysis and I/O Acceleration on Leadership-Class Systems." In *2011 IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, 9–14. IEEE.

Williams David M, Castonguay Patrice, Vincent Peter E, and Jameson Antony. 2013. "*Energy Stable Flux Reconstruction Schemes for Advection–diffusion Problems on Triangles*." Journal of Computational Physics 250: 53–76.

Witherden F. D., Vincent P. E., and Jameson A. 2016. "HighOrder Flux Reconstruction Schemes." In *Handbook of Numerical Analysis*. Vol. 17, 227–263. Elsevier.

Yi Hong, Rasquin Michel, Fang Jun, and Bolotnov Igor A. 2014. "In-Situ Visualization and Computational Steering for Large-Scale Simulation of Turbulent Flows in Complex Geometries." In *2014 IEEE International Conference on Big Data*, 567–572. IEEE.