Development of a Generalized Grid Interface for Turbomachinery simulations with OpenFOAM

Martin Beaudoin [a], Hrvoje Jasak [b]

[a] *Institut de recherche Hydro-Québec, 1800 boul. Lionel-Boulet, Varennes, QC, Canada, J3X 1S1*

[b] *Wikki Ltd., 33 Palmerston House, 60 Kensington Place, London W8 7PU, United Kingdom*

Abstract

This paper presents an implementation of the Generalized Grid Interface (GGI) for OpenFOAM. It is used to couple multiple non-conformal regions into a single contiguous domain at matrix level.

The main advantage of the Generalized Grid Interface comes from removing the requirement to adjust the topology of the mesh at the interface between two non-conformal meshes. Instead, a set of weighting factors is evaluated in order to properly balance the flux at the GGI interface.

Evaluation of the GGI weighting factors is based on the precise computation of the patch faces intersection on each side of the coupling interface as follows from discretisation. This is complicated due to the different grid resolution of the mesh on each side of the interface. This paper describes the choice and implementation of robust and quick algorithms for computing the GGI weighting factors.

Furthermore, for turbomachinery simulations in particular, various derived forms of GGI are used regularly. Cyclic GGI, partial GGI and mixing plane GGI are all variants of the basic GGI interface that are needed for various use cases.

Finally, it is very important that the GGI interface can be used for parallel OpenFOAM simulations on clusters of computers; otherwise, this would impose a severe limit on the size of the problems.

This paper will show the current state of development of the Generalized Grid Interface for turbomachinery simulations and steps towards its generalisation and parallelisation.

Key words: CFD, OpenFOAM, Generalized Grid Interface, GGI, cyclic GGI, Turbomachinery

1. Introduction

Implicit coupling interfaces are present in OpenFOAM in order to join multiple mesh regions into a single contiguous domain. The *processor*, *regionCouple* and *cyclic* are examples of such interfaces [1]. They are built to join conformal mesh regions, where the patches nodes on each side of the interface are matching one to one.

The Generalized Grid Interface (GGI) is another example of coupling interface, used for joining multiple non-conformal regions where the patches nodes on each side of the interface do not match.

The GGI is a common interface in the domain of CFD simulation of turbomachines like the water turbine runners used by the electrical power plants to produce electricity. Separate 3D meshes are generally involved in order to simulate the flow of water through a succession of complex geometries like the distributor where the stay vanes are located, the turbine runner and the draft tube. The requirement to fit all meshes with conformal matching interfaces is often very difficult or leads to geometric compromises that would affect the numerical quality of the simulation results. Non-conformal meshes are thus generated separately for each part of the whole geometric model, and joined together using one or many GGIs.

For nonstationary turbomachinery simulations, the relative rotation of mesh parts will necessarily produce non conformal interfaces between the fixed and moving sections. The GGI can replace the OpenFOAM *slidingInterface* for nonstationary simulations by forcing the re-evaluation of the GGI weighting factors at each simulation time step.

Specialized versions of the GGI are also needed in order to simplify the mesh complexity of many turbomachinery simulations and hence reduce the computer time needed to run the simulations. The *Cyclic GGI* is necessary for the simulation of periodic geometries. The *partial GGI* and the *mixing plane* interface are necessary for the simulation of a single blade pair when using a partial coverage of the rotor and stator parts.

Figure 1 shows a close-up view of a mesh for the ERCOFTAC centrifugal pump, a typical example where GGIs are needed for the rotor-stator interface and for the pump passages.

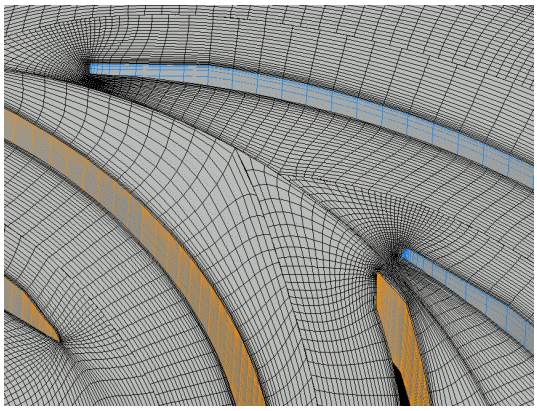


Fig. 1. Close-up view of a ERCOFTAC centrifugal pump non-conformal mesh.

2. The basic Generalized Grid Interface

The basic Generalized Grid Interface is a new coupling interface for OpenFOAM that is using weighted interpolation to evaluate and transmit flow values across a pair of conformal or non-conformal coupled patches.

The basic GGI is similar to a case of "static" sliding interface, but is also much simpler in the sense that no re-meshing is required for the neighbouring cells of the interface.

2.1. *The GGI base equations*

The equations that control the flow values between the GGI master patch to the GGI shadow patch are derived from basic FVM discretisation reasoning. They state that consistent and conservative discretisation across the interface is achieved using weighted interpolation of the following form:

For the flow values or variables from the master patch to the slave patch:

$$\phi_{Si} = \sum_{n} W_{M_n\_to\_S_i} * \phi_{Mn} \tag{1}$$

For the flow values or variables from the slave patch to the master patch:

$$\phi_{Mj} = \sum_{m} W_{S_m\_to\_M_j} * \phi_{Sm} \tag{2}$$

In order for the interface discretisation to remain conservative, we have the following three constraints:

$$\sum_{n} W_{M_n\_to\_S_i} = 1.0 \tag{3}$$

$$\sum_{m} W_{S_m\_to\_M_j} = 1.0 \tag{4}$$

$$W_{M_n\_to\_S_i} * |\underline{\mathrm{S}}_{M_n}| = W_{S_m\_to\_M_j} * |\underline{\mathrm{S}}_{S_m}| = |\underline{\mathrm{S}}_{\cap_{M\_to\_S}}| \tag{5}$$

with the additional symmetry constraint:

$$if \quad W_{M_n\_to\_S_i} > 0 \quad \Longrightarrow \quad W_{S_i\_to\_M_n} > 0 \tag{6}$$

but in general:

$$W_{M_n\_to\_S_i} \neq W_{S_i\_to\_M_n} \tag{7}$$

where

ϕ_S	:	shadow patch variable
ϕ_M	:	master patch variable
i	:	ith shadow patch face
j	:	jth master patch face
n	:	number of master face neighbours for shadow patch i
m	:	number of shadow face neighbours for master patch j
$W_{M\_to\_S}$	:	master facet to shadow facets weighting factor
$W_{S\_to\_M}$	:	shadow facet to master facets weighting factor
$\lvert\underline{\mathrm{S}}_M\rvert$	:	surface area of master facet
$\lvert\underline{\mathrm{S}}_S\rvert$	:	surface area of shadow facet
$\lvert\underline{\mathrm{S}}_{\cap_{M\_to\_S}}\rvert$	:	intersection surface area between master and shadow facets

We can see that for both the master and shadow GGI patch faces, a list of GGI weighting factors W needs to be evaluate in order to compute the flow values.

The determination of the number of neighbours for each facet and the computation of the GGI weighting factor values require robust and precise algorithms in order to make sure the flow values are correctly distributed across the interface. Experience with simplified interpolation which does not strictly satisfy the above constraints shows that even minimal error in W leads to unacceptable discretisation error. The only reliable weights calculation uses geometrical cutting to determine the facets intersection area.

2.2. *Evaluation of the GGI weighting factors*

The value of the GGI weighting factors can be deduced from equation 5 which is based solely on facet area values and polygonal intersection.

For the master to shadow patch faces:

$$W_{M\_to\_S_i} = \frac{|\underline{S}_{\cap_{M\_to\_S_i}}|}{|\underline{S}_{M_n}|}, \quad with \quad W_{M\_to\_S_i} \quad \in \quad]0.0, 1.0] \tag{8}$$

For the shadow to master patch faces:

$$W_{S\_to\_M_j} = \frac{|\underline{S}_{\cap_{S\_to\_M_j}}|}{|\underline{S}_{S_m}|}, \quad with \quad W_{S\_to\_M_j} \quad \in \quad]0.0, 1.0] \tag{9}$$

where

$\|\underline{S}_{\cap_{M\_to\_S}}\|$ and $\|\underline{S}_{\cap_{S\_to\_M}}\|$	:	Surface intersection area between a master and shadow patch faces
$\|\underline{S}_M\|$ and $\|\underline{S}_S\|$	:	Surface area of a master and shadow patch face
i	:	ith shadow patch face for a given master patch face
j	:	jth master patch face for a given shadow patch face

The GGI weighting factors are basically the percentage of surface intersection between two overlapping faces. These values obviously need to be greater than zero, otherwise this would mean no intersection between two given faces and clearly indicates a failure of the neighbourhood determination algorithm.

In order to compute this simple ratio, the value of the faces surface area can easily be recovered using the standard OpenFOAM API. However, the value of the intersection surface area requires an efficient 2D face-to-face intersection algorithm.

The GGI implementation is using the Sutherland-Hodgman [2] algorithm for computing the master and shadow face intersection surface area. This algorithm is simple, fast and robust; being re-entrant, it allows for a very compact source code implementation. The Sutherland-Hodgman algorithm is also generic enough to handle any convex n-sided polygons.

One important caveat to keep in mind: the Sutherland-Hodgman algorithm is designed to handle convex polygons only, which is generally the case for OpenFOAM meshes. In the unlikely event where non-convex polygons would be present as GGI patch faces, the Sutherland-Hodgman algorithm would need to be replaced with a more general algorithm like the Weiler-Atherton [3] or the Greiner-Hormann [4] clipping algorithms that are designed to handle convex, concave and self-intersecting polygons.

2.3. *Patch faces neighbourhood determination*

Before computing the GGI weighting factors for each patch face using Sutherland-Hodgman, we first need to determine the "neighbourhood" for each of those faces; that is, for any given master patch face, how many faces from the shadow patch are overlapping the master face.

Without using any a priori knowledge about the topology from the master or shadow patches, this search problem has a complexity of $O(n^2)$. This is resolved using optimised search and quick-reject tests. Validation tests have shown that for GGI patches of small to medium sizes, the cost of pre-processing search is quite acceptable using standard computing resources.

2.3.1. *Neighbourhood determination: a first quick reject test*

In order to get acceptable performance for the faces neighbourhood determination algorithm, a first quick reject test based on Axis Aligned Bounding Box [5] (AABB) has been implemented as a first stage filter.

```
1: for all ith faces from GGI master patch do
2:   bb_m ← BoundingBox(masterFace[i])
3:   for all jth faces from GGI shadow patch do
4:     bb_s ← BoundingBox(shadowFace[j])
5:     bb_boosted.xmin ← bb_m.xmin − bb_s.delta_x,   bb_boosted.xmax ← bb_m.xmax + bb_s.delta_x
6:     bb_boosted.ymin ← bb_m.ymin − bb_s.delta_y,   bb_boosted.ymax ← bb_m.ymax + bb_s.delta_y
7:     bb_boosted.zmin ← bb_m.ymin − bb_s.delta_z.   bb_boosted.zmax ← bb_m.ymax + bb_s.delta_z
8:     if bb_s.isInside(bb_boosted) then
9:       masterNeighbours[bb_m].addToList(bb_s)
10:      shadowNeighbours[bb_s].addToList(bb_m)
11:    end if
12:  end for
13: end for
```

Algorithm 1: Quick reject test based on Axis Aligned Bounding Box.

Algorithm 1 describes the simple logic of this first stage quick reject test. This algorithm only uses simple comparisons and additions, so each "inside-outside" test is very quick. No false negatives are generated by this algorithm, which is important in order not to miss any potential face neighbour candidates.

However, in many occasions, this simple algorithm ends up being too greedy because of the usage of AABB; so usually, a reduced number of false positive candidates will be flagged as potential neighbours for a given face.

2.3.2. *Neighbourhood determination: elimination of the false neighbour candidates*

A second filter stage is needed in order to eliminate the false neighbours from the list of potential face neighbours. This filter is implemented as a 2-step algorithm using well-known techniques.

First, since the last stage of this second filter operates in 2D, a preprocessing step is required in order to project all the potential shadow neighbours faces onto the plane of the master face. We expect all the overlapping neighbours to have a face normal that closely match the face normal of the master face. However, for GGI patches where important local geometric discontinuities are present, it is not uncommon for the greedy AABB algorithm to select potential neighbour faces that are false positives in 3D, but becomes real positive when projected onto the master face 2D plane.

Thus, a feature cosine tolerance factor is required in order to quickly eliminate those false neighbours in 3D before using the 2D algorithms. For all the candidate face neighbours of a given master patch face, we will compare the cosine between the two faces normal, and reject the candidate faces where the cosine value falls outside of a user-defined tolerance zone.

Second, for each master faces, once all the potential neighbour faces are projected onto the master face 2D plane, a final filtering step will use the Separating Axis Theorem (SAT) algorithm [6] in order to remove the last remaining non-overlapping faces. An efficient point-in-polygon technique based on the Hormann-Agathos algorithm [7] has been included into the SAT algorithm implementation in order to rapidly identify and handle some trivial cases, like all the points of a given polygon are inside or outside of a neighbour polygon, etc.

2.4. *Numerical considerations*

As for any numerical algorithm, the evaluation of the GGI weighting factors is prone to classical numerical issues related to the comparison of floating point values and to the internal numerical processor precision.

A specific tolerance parameter has been introduced for the GGI weighting factors calculation in order to deal with this issue; specifically when trying to determine if a pair 2D or 3D point coordinates are close enough to be considered coincident.

This tolerance factor is based on the smallest GGI weighting factor acceptable for a given set of GGI patches. The value of this tolerance factor relates directly to the smallest relative intersection surface area allowed for the GGI patch faces. The square root of this tolerance factor relates directly to the smallest relative circle or sphere radius that would discriminate between coincident or non-coincident 2D or 3D point coordinates.

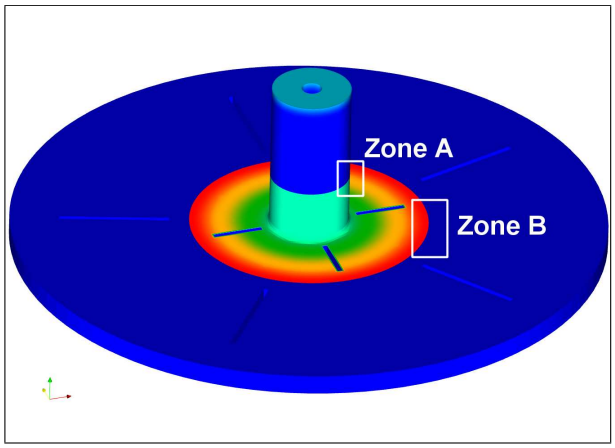
2.5. *Discretisation support*

Another common problem when dealing with discretized geometric models is the effect of the discretisation resolution when dealing with cylindrical or spherical shapes. Cylindrical geometries are very common in turbomachinery simulations.

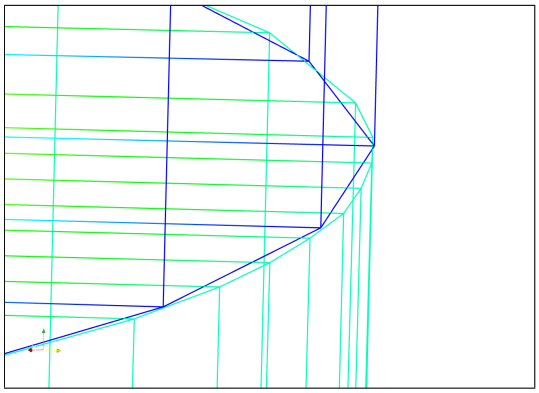
There are three characteristic situations that arise when dealing with circular geometries and GGIs:

1: Coplanar circular patches using different circumferential mesh discretisation
2: Concentric cylindrical patches using different circumferential mesh discretisation
3: Moving meshes with fixed and rotating cylindrical mesh regions.

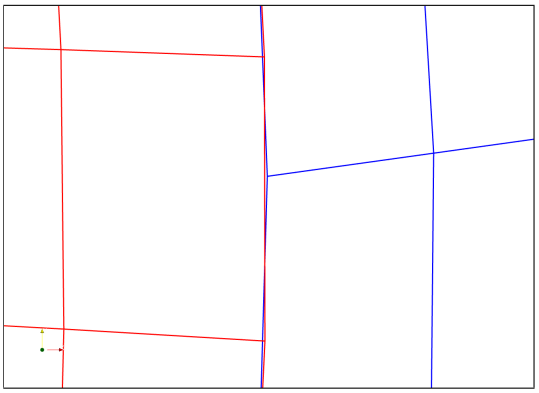
For the first situation, small sections from the GGI patch faces located at the outer edge of the coplanar circular regions will be hanging "outside", with no counterpart region from the neighbour patch to transmit the face flow values. For all the faces with such non-overlapping regions, the values of the GGI weights will not sum up to 1.0. Figure 2b, taken from a moving mesh simulation of a simplified turbine using two GGIs, illustrates this situation.



(a) Simplified turbine: rotor and stator.



(b) Zone A: Discretisation effects for coplanar circular patches.



(c) Zone B: Discretisation effects for concentric cylindrical patches.

Fig. 2. Example of discretization effects for cylindrial geometries.

For the second situation, the projection of the shadow neighbours faces onto a given master face will distort slightly the resulting projected polygon. Since the surface area of this distorted polygon is being used for the computation of the GGI weighting factors, the resulting GGI weight will be slightly underestimated, and for this master face the value of the GGI weights will not sum up to 1.0. Figure 2c illustrates this situation.

The third situation is basically a combination of the first two cases, as shown in Figure 2.

In all those situations, we will end up by default with a violation of the requirement for equations 3, 4 and 5; the GGI will not be conservative.

The current implementation of the GGI corrects this problem by rescaling the face weighting factors so they will sum up to 1.0. For each neighbouring face involved, the rescaling will be proportional to the initial value of the neighbour face weighting factor, so the overall correction for each face ends up being weight interpolated. Validation test cases have shown that this correction strategy gives adequate results and a typical scaling factor is of the order of 0.01%

2.6. *Dictionary entries for the GGI*

As for any type of boundary conditions in OpenFOAM, the GGI needs to be properly defined in the case *boundary* file, and in the case *time* directories for the different field values.

Listing 1 shows an example of a GGI dictionary entry for constant/polymesh/boundary. This is taken from one of the validation test cases presented in section 4.

```
                    :
upstreamPatch_GGI       // Master GGI patch
{
    type                ggi;
    nFaces              1600;
    startFace           174240;
    shadowPatch         downstreamPatch_GGI;
}

downstreamPatch_GGI     // Shadow GGI patch
{
    type                ggi;
    nFaces              2654;
    startFace           175840;
    shadowPatch         upstreamPatch_GGI;
}
                    :
```

Listing 1: Example of a GGI entry for the dictionary constant/polymesh/boundary.

Listing 2 shows an example of a GGI dictionary entry for the times directories, in this case the dictionary *0/U*

2.7. *Handling of non-overlapped faces*

The presence of non-overlapped facets in the GGI patches will introduce weighting factors with zero value that are easy to detect and need to be handled properly. The proposed solution is to modify the topoplogy of the GGI patches in order to dynamically reassign the non-overlapped faces to a user-defined boundary condition, usually a wall-type boundary condition.

Using this feature, it is possible to assemble complex meshes using dissimilar patch geometries by simply coupling the neighbour patches through a GGI.

Figure 3 illustrates a typical situation where the outlet of an ERCOFTAC conical diffuser is connected to the inlet of a large cylindrical dump. Both outlet and inlet can be coupled using a GGI, without any

```
                    :
dimensions          [0 1 -1 0 0 0 0];
internalField       uniform (0 0 0);

boundaryField
{
    upstreamPatch_GGI
    {
        type            ggi;
        value           uniform (0 0 0);
    }
    downstreamPatch_GGI
    {
        type            ggi;
        value           uniform (0 0 0);
    }
                    :
```

Listing 2: Example of a GGI entry for the dictionary 0/U.

prior modification to the topology of the large dump inlet. The overlapping section of the dump inlet will form a GGI with the diffuser outlet. For this specific case, the non-overlapping faces from the dump inlet patch would be reassigned to a standard wall boundary condition.

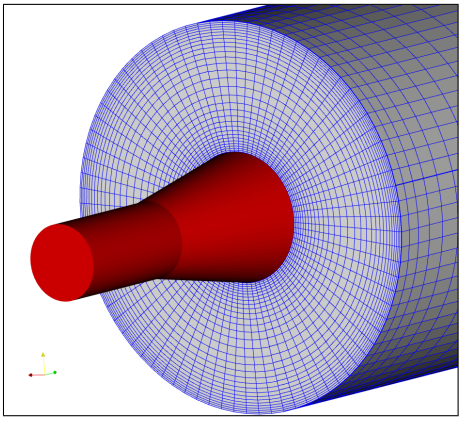


Fig. 3. GGI with non-overlapped faces: the ERCOFTAC conical diffuser, Case2: Dump case.

Experimentation with the OpenFOAM class *repatchPolyTopoChanger* shows that this solution is sound and gives adequate results for static meshes. Future work will address the case for moving meshes where GGI facets can switch between overlapped and non-overlapped states at different time steps.

2.8. *Using the GGI with conformal meshes*

For conformal coupled patches, the GGI weighting factors are obviously equal to 1.0. Applying a GGI for such trivial case can still be useful as a last resort solution for coupling multi-parts meshes when OpenFOAM utilities like *mergeMesh* and *stitchMesh* fail to produce usable meshes. The cyclic GGI is also extremely useful for handling mis-ordered cyclics.

3. The cyclic GGI

The cyclic GGI is a variation of the basic GGI developed for handling periodic non-conformal mesh. It uses the same internal algorithms as the basic GGI.

The cyclic GGI adds an internal transform to be applied to the shadow patch data in order to internally superpose them on top of the master patch data. This transformation is required in order to determine the cyclic GGI patch faces neighbourhood, to compute the cyclic GGI weighting factors, and also to transform any shadow patch vector field values before computing the GGI weighted interpolation across the interface.

```
                    :
ggiPerio1
{
    type              cyclicGgi;
    nFaces            243;
    startFace         8991;
    shadowPatch       ggiPerio2;
    rotationAxis      (0 0 1);
    rotationAngle     50;
    separationOffset  (0 0 0);
}

ggiPerio2
{
    type              cyclicGgi;
    nFaces            243;
    startFace         9234;
    shadowPatch       ggiPerio1;
    rotationAxis      (0 0 1);
    rotationAngle     50;
    separationOffset  (0 0 0);
}
                    :
```

Listing 3: Example of a cyclic GGI entry for the dictionary constant/polymesh/boundary.

Listing 3 shows an example of a cyclic GGI dictionary entry for a case *boundary* file. In this specific example, both cyclic GGI patches are separated by a rotation of 50^o about the z axis.

For rotation periodicity, both the *rotationAxis* and *rotationAngle* parameters define the rotation transform. For translational periodicity, the *separationOffset* parameter defines the translational transform.

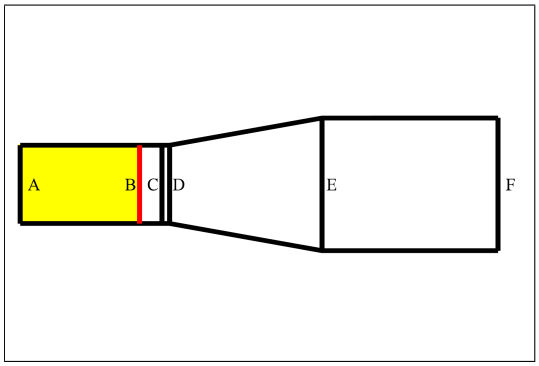
4. Validation test cases

Numerous validation test cases have been utilized all along the development process of the GGI and cyclic GGI. For every type of test cases, both conformal and non-conformal versions of the same case have been generated and compared.

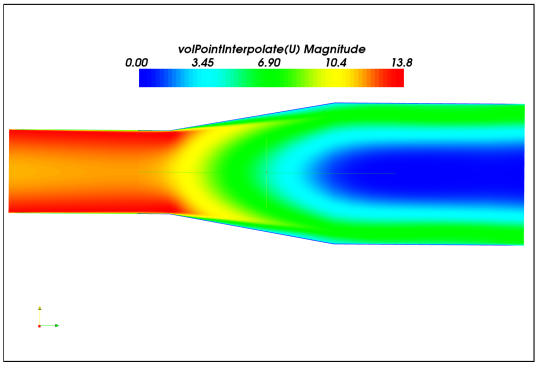
Two of the most useful test cases for developing the standard GGI were the ERCOFTAC conical diffuser test cases called Case0 and Case1 [8].

Figures 4a to 4f illustrate a typical mesh topology and a series of results for a modified version of Case1 where a section of the stationary part of the swirl generator was rotated by 15^o about the diffuser axis. The relative position of the two O-grid patches provides for a complex non-conformal geometry, well suited for testing the GGI.

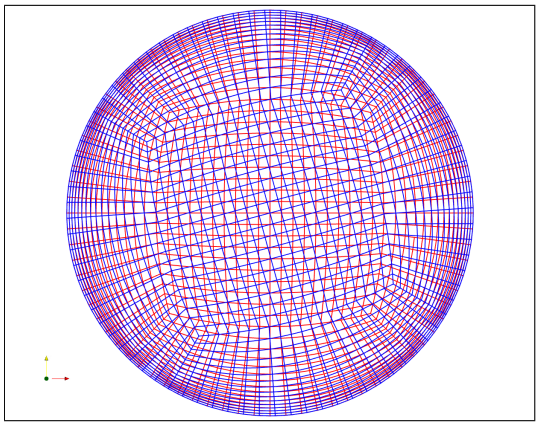
From the comparison of the simulation results, we observe that the velocity results computed with a non-conformal mesh closely match the results coming from the same case using a conformal mesh. We can see a slight discrepancy for the peak values of the *turbulence kinetic energy (k)*, but it was observed that this difference becomes smaller as we refine the circumferential mesh resolution on each side of the GGI. This sensitivity of k to the mesh resolution in the presence of a GGI would be worth investigating in future work. Looking at the residual values of the same simulation, we can see that the convergence rate and residual values are almost identical for both the conformal and non-conformal case.



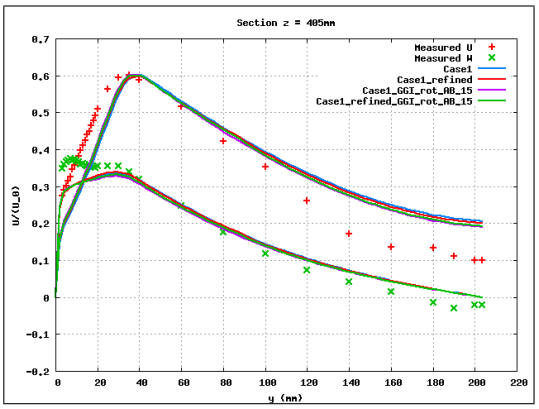
(a) Case1 with location of section A-B.



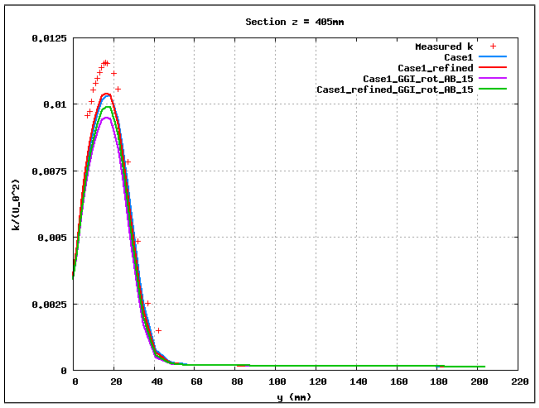
(b) Velocity component U from a longitudinal cutting plane.



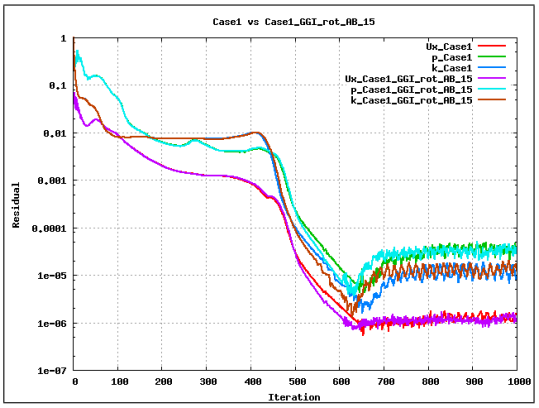
(c) Illustration of the GGI patch topology.



(d) Velocity comparison.



(e) Turbulence Kinetic Energy comparison.



(f) Convergence comparison.

Fig. 4. ERCOFTAC conical diffuser: Case1 with GGI: rotation of section AB by 15^o.

5. Summary and Future work

This paper presents an implementation of the Generalized Grid Interface for OpenFOAM. The design is based on robust algorithms for evaluating the GGI weighting factors necessary to accurately interpolate the flow values across the coupling interface. A cyclic GGI is also available for coupling periodic non-conformal mesh regions. A special algorithm was experimented with in order to easily couple non-conformal patches where regions of non-overlapping faces are present. The current design has been validated on numerous test cases, and provides satisfactory results.

Future work will focus on improving the GGI patch faces neighbourhood determination algorithm using multi-resolution algorithms based on octrees. This should significantly help in lowering the computation cost of this pre-processing stage which is quite significant when very large number of patch faces are involved.

Handling of non-overlapped faces will be revisited for the case of moving meshes where GGI facets can switch between overlapped and non-overlapped states at different time steps.

Parallelisation of the GGI will also be addressed in order to run large meshes using the GGI on cluster of computing machines.

Finally, new specialized versions of the GGI will be developed for turbomachinery applications, namely the partial GGI and mixing plane interfaces.

6. Acknowledgements

The authors would like to thank Dr. Håkan Nilsson and the department of Applied Mechanics, Fluid Dynamics at Chalmers University of Technology, Göteborg, Sweden for helping make this work progress in a very productive and stimulating environment while MB visited Chalmers University on a 6 month research leave.

References

[1] OpenFOAM: The OpenSource CFD Toolbox, User Guide, Version 1.4.1 (August 1st 2007).
[2] I. E. Sutherland, G. W. Hodgman, Reentrant polygon clipping, Commun. ACM 17 (1) (1974) 32–42.
[3] K. Weiler, P. Atherton, Hidden surface removal using polygon area sorting, in: SIGGRAPH '77: Proceedings of the 4th annual conference on Computer graphics and interactive techniques, ACM, New York, NY, USA, 1977, pp. 214–222.
[4] G. Greiner, K. Hormann, Efficient clipping of arbitrary polygons, ACM Trans. Graph. 17 (2) (1998) 71–83.
[5] J. V. Verth, L. Bishop, Essential Mathematics for Games and Interactive Applications: A Programmer's Guide, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.
[6] S. Gottschalk, Separating axis theorem, Tech. Rep. TR96-024, Dept. of Computer Science, UNC Chapel Hill (1996).
[7] K. Hormann, A. Agathos, The point in polygon problem for arbitrary polygons, Computational Geometry 20 (3) (2001) 131–144.
[8] H. Nilsson, M. Page, M. Beaudoin, B. Gschaider, H. Jasak, The OpenFOAM Turbomachinery working-group, and conclusions from the Turbomachinery session of the third OpenFOAM workshop, IAHR: 24th Symposium on Hydraulic Machinery and Systems, Foz do Iguassu, Brazil (2008).