Multiblock mesh refinement by adding mesh singularities

Cecil G Armstrong[1][0000-0001-8695-5016], Li Tak Sing[2], Christopher Tierney[1][0000-0003-3341-6902] and Trevor T Robinson[1][0000-0002-6595-6308]

[1] School of Mechanical and Aerospace Engineering, Ashby Building, Belfast, BT9 5PY, UK
[2] School of Science and Technology, The Open University of Hong Kong, Homantin, Kowloon, HK
{c.armstrong, christopher.tierney, t.robinson}@qub.ac.uk, tsli@ouhk.edu.hk

Abstract. Several templates for 2D and 3D structured mesh refinement are presented. The templates have the property that the minimum number of irregular points or edges (mesh singularities) are added. For a given set of external division numbers a variety of interior meshes can be generated.

The positions of the internal vertices in the template are calculated explicitly using an extended transfinite mapping scheme, which has previously been shown to be equivalent to iterative iso-parametric smoothing. Since calculating the block vertex positions requires the solution of a small number of linear equations, the optimum mesh in the interior of the template can be evaluated very cheaply before the block structured mesh is generated.

Keywords: Multiblock Quad and Hex Meshing, Quad and Hex Mesh Refinement, Mesh Smoothing.

1 Introduction

Despite huge advances in the state of the art of unstructured mesh generation, e.g. [1], there is still a demand for the generation of structured multi-block meshes. A number of authors [2]–[4] have explored the use of medial axis techniques, since these tend to offer meshes which have close to the minimum number of mesh irregularities or 'singularities'. Other techniques such as frame fields have gained attention, but it was noted in [5] that it is necessary to take into account the global structure of hexahedral meshes. This implies tracking the position and connectivity of singular edges in 3D.

After a multi-block decomposition has been created, adjustment of edge division numbers to achieve an adequate mesh size distribution is required. Frequently mesh sizing requirements in one area leads to the propagation of an overly dense mesh elsewhere, requiring the insertion of 'steerbacks' or additional block topology to create transition meshes and local mesh refinement [6]. Solution errors may indicate a need for adaptive refinement of an existing blocking, which is similarly difficult.

One way to avoid this complexity is to implement a non-conformal refinement strategy, using multi-point constraints at incompatible interfaces. This has the benefit of producing higher quality elements in the refinement regions with no propagation through the remaining mesh. However, this can introduce errors at the mismatching interface and is unsuitable for many applications [7].

Current conformal mesh adaptation strategies for quad and hex elements are generally template-based operations focused on 2-refinement or 3-refinements strategies [8]. Schneider [9] used a refinement strategy to subdivide quad elements and hex elements in the refinement region using a quadtree refinement, maintaining associativity by inserting templates in the transition zone. Ebeida [10] introduced a parallel realization of Schneider's 2-refinement strategy for unstructured meshes, whilst Qian [11] extended this approach for non-manifold conformal mesh generation. Other work [12] incorporated conformal refinement and coarsening strategies by combining template-based operations with localized coarsening and quality improvement in a single workflow. While these techniques provide topological mesh adaptation focusing directly on mesh elements they can also be applied to the 'coarser' block decomposition. The issue is that they are generally implemented using a 2-refinement or 3-refinement strategy, meaning arbitrary element numbers require further refinement and the combination of templates that may introduce large numbers of unnecessary singularities.

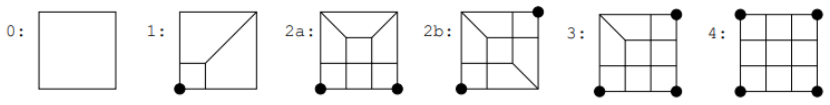


Fig. 1. Two and three refinement templates [9]

The aim of the present work was to identify some generic ways in which an existing block decomposition can be refined to produce meshes which better match mesh density variations. For a given region the minimum necessary number of singular points can be established geometrically, Fogg et al. [13]. Fogg also showed that if additional 2D singularities are to be added, a positive (5-valent) and negative (3-valent) pair must be added simultaneously.

Therefore, in 2D, mesh refinement can be accomplished by incrementally adding pairs of positive-negative singularities to an existing block decomposition. It will be shown that, for a singularity pair, the optimum position of the singular points for a given mesh density can be calculated explicitly. It will also be shown that there are a range of mesh distributions that can satisfy external division number constraints, and it is possible to choose the optimum arrangement at low computational cost.

Other general block arrangements and 3D refinements are also presented.

2 Laplacian smoothing and iso-parametric mapping

The 'midpoint subdivision' technique for decomposing a convex polygon into a series of 2D quadrilateral blocks, or a 3D polyhedron with convex edges and trivalent vertices into hexahedral bricks, has been presented by Li et al. [14]. By joining a face midpoint to all the edge midpoints, one quad is generated at every vertex of the original polygon,

Fig. 2b) and c). By adjusting the internal mesh division numbers, e.g. n_0, n_1, n_2 in **Fig. 2**c), a range of external division numbers can be satisfied.

In a subsequent paper [15], efficient mapping methods were developed for determining nodal positions in the resulting mesh based on an extension to transfinite mapping, and it was shown that the result is identical to that obtained by iterative iso-parametric smoothing.

From eqn. (7) of Li et al. [15], the position of a node $\boldsymbol{x}$ at the centre of a k-sided face, **Fig. 2**, can be identified using

$$\sum_0^{k-1} \frac{\boldsymbol{b}_r + \boldsymbol{b}_{r+1} - \boldsymbol{a}_r}{n_r n_{r+1}} - \sum_0^{k-1} \frac{1}{n_r n_{r+1}} \boldsymbol{x} = \boldsymbol{0},$$

where $\boldsymbol{b_r}$ is the point on edge r that the face midpoint is connected to, and $\boldsymbol{a_r}$ is the opposite corner joining edges r and $r+1$. n_r is the number of elements (division number) on the radial edge joining the centre point $\boldsymbol{x}$ to edge r.

An alternative form, shown below, allows the position of the face midpoint $\boldsymbol{x}$ to be found from

$$\sum_0^{k-1} \frac{1}{n_r n_{r+1}} \boldsymbol{a_r} - \sum_0^{k-1} \left(\frac{1}{n_r n_{r+1}} + \frac{1}{n_{r-1} n_r} \right) \boldsymbol{b_r} + \sum_0^{k-1} \frac{1}{n_r n_{r+1}} \boldsymbol{x} = \boldsymbol{0}.$$

Here n_{r-1} is the radial division number of the edge preceding edge r in a clockwise traversal of the boundary, whilst n_{r+1} is the radial division number of the following edge.

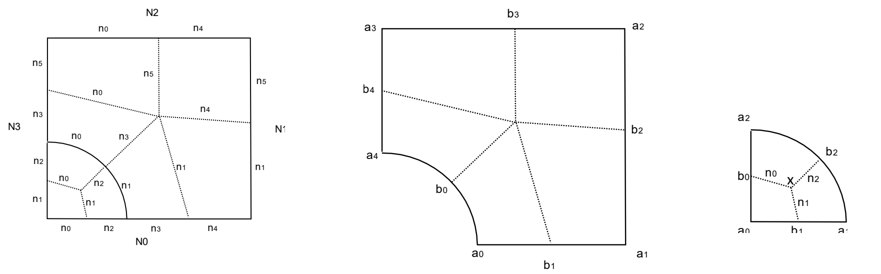
So, for a 3-sided face, **Fig. 2**c) for example,

$$\left[+\frac{1}{n_0 n_1} \quad +\frac{1}{n_1 n_2} \quad +\frac{1}{n_2 n_0} \quad -\left(\frac{1}{n_0 n_1} + \frac{1}{n_2 n_0} \right) \quad -\left(\frac{1}{n_1 n_2} + \frac{1}{n_0 n_1} \right) \quad -\left(\frac{1}{n_2 n_0} + \frac{1}{n_1 n_2} \right) \quad + \sum_{r=0}^{2} \frac{1}{n_r n_{r+1}} \right] \begin{Bmatrix} \boldsymbol{a_0} \\ \boldsymbol{a_1} \\ \boldsymbol{a_2} \\ \boldsymbol{b_0} \\ \boldsymbol{b_1} \\ \boldsymbol{b_2} \\ \boldsymbol{x} \end{Bmatrix} = \boldsymbol{0}$$

This representation is potentially useful when some of the positions are unknown, for example in a network of primitives with common edges.

3 Adding a singularity pair to an existing block

Adding a positive and negative singularity to an existing block can be thought of as partitioning the quad block into a 3-sided and a 5-sided region, **Fig. 2**a). The difficulty is that, even when the singularities are directly connected, the midpoint on the common edge (b_2 in the triangle and b_0 in the pentagon) affects the position of both face midpoints.



a) Singularity pair with internal and external division numbers

b) Positive singularity in a pentagon

c) Negative singularity in a triangle

Fig. 2. A pair of singularities in a quad block

In the equation for the triangular region, **Fig. 2**c), the positions of $\{\boldsymbol{a_0} \quad \boldsymbol{a_1} \quad \boldsymbol{a_2} \quad \boldsymbol{b_0} \quad \boldsymbol{b_1} \qquad\}$ are known and can be moved to the right-hand side, so we end up with two unknown positions for the triangle midpoint $\boldsymbol{x^t}$ and the edge midpoint $\boldsymbol{b_2}$ which is shared with the pentagon, as

$$\left[-\left(\frac{1}{n_2n_0}+\frac{1}{n_1n_2}\right) \quad \sum_{r=0}^{2}\frac{1}{n_rn_{r+1}}\right]\begin{Bmatrix}\boldsymbol{b_2}\\ \boldsymbol{x^t}\end{Bmatrix} = \boldsymbol{rhs^0}.$$

Similarly, the equation describing the face and edge midpoints in the pentagon reduces to

$$\left[-\left(\frac{1}{n_0n_1}+\frac{1}{n_4n_0}\right) \quad \sum_{r=0}^{4}\frac{1}{n_rn_{r+1}}\right]\begin{Bmatrix}\boldsymbol{b_0}\\ \boldsymbol{x^p}\end{Bmatrix} = \boldsymbol{rhs^1}.$$

$\boldsymbol{b_2}$ in the triangle and $\boldsymbol{b_0}$ in the pentagon are of course the same point.

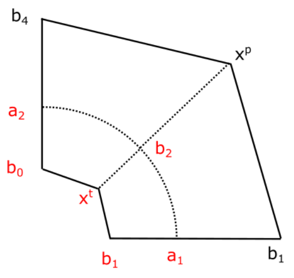


Fig. 3. Quad with common boundary point at the centre. The points are as labelled in Fig. 2b) and c).

The last step is to calculate the position of triangle point $\boldsymbol{b_2}$ using the same transfinite mapping equations. With reference to **Fig. 3**, and using the convention that red points are on the triangle and black points are on the pentagon, the corner points of the 4-sided region surrounding the common edge midpoint b_2 are

$[\boldsymbol{b_1} \quad \boldsymbol{b_1} \quad \boldsymbol{b_4} \quad \boldsymbol{b_0}]$,
whilst the edge midpoints are
$[\boldsymbol{x^t} \quad \boldsymbol{a_1} \quad \boldsymbol{x^p} \quad \boldsymbol{a_2}]$,
and the triangle and pentagon midpoints are $\boldsymbol{x^t}$ and $\boldsymbol{x^p}$ respectively.

Collecting known terms and moving them to the RHS gives 3 vector equations in 3 unknown positions as

$$\begin{bmatrix} . & . & 0 \\ 0 & 0 & . \\ . & . & . \end{bmatrix} \begin{Bmatrix} \boldsymbol{b_2} \\ \boldsymbol{x^t} \\ \boldsymbol{x^p} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{rhs^0} \\ \boldsymbol{rhs^1} \\ \boldsymbol{rhs^3} \end{Bmatrix}$$

Once these equations are solved to find the face midpoints and the common edge midpoint, all the other nodes can be found using Li's existing algorithm.

4 Division numbers

With the arrangement of the triangle and pentagon shown in **Fig. 2** there are 6 internal division numbers. If target external division numbers on the block are required, this means solving an integer programming problem, but here we will proceed initially by specifying the internal division numbers. On the external edges of the block, the corner and edge midpoint nodes of the triangle and pentagon can be identified. If the division numbers or edge meshes are changed then the position of the face midpoints and the common edge midpoint can be re-calculated as above.

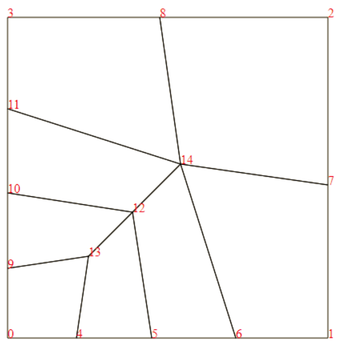


Fig. 4. Labelled corners and midpoints. Internal division numbers n = [1,1,1,1,1,1]

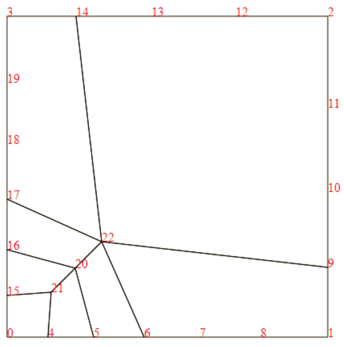


Fig. 5. Internal division numbers = [1,1,1,1,3,3]

Fig. 4 and **Fig. 5** show the block corner points calculated using these equations for two different sets of division numbers. The negative singularity, the positive singularity and the common edge midpoint are at nodes 13, 14 and 12 respectively in **Fig. 4** and nodes 21, 22 and 20 in **Fig. 5**.

5 Common edge nodes

After the division numbers have been set, nodes must be generated on the common edge between the triangle and pentagon.

The nodes on the common edge between nodes 9 and 37 in **Fig. 6** can be derived using the standard transfinite mapping equations. Based on the position of nodes (0, 4, 5, 6), and nodes (3, 29, 28, 27) representing two opposite logical block edges, the position of nodes 50 and 51 can be calculated since (0, 37, 3) and (6, 44, 27) form the other two sides of a logical 4-sided region. For example, the position of node 50 can be calculated using the equation:

$$\begin{pmatrix} \frac{1}{7} & \frac{1}{14} & \frac{1}{16} & \frac{1}{8} & -\left(\frac{1}{7}+\frac{1}{8}\right) & -\left(\frac{1}{7}+\frac{1}{14}\right) & -\left(\frac{1}{14}+\frac{1}{16}\right) & -\left(\frac{1}{16}+\frac{1}{8}\right) \end{pmatrix} \begin{pmatrix} \boldsymbol{p_0} \\ \boldsymbol{p_6} \\ \boldsymbol{p_{27}} \\ \boldsymbol{p_3} \\ \boldsymbol{p_{37}} \\ \boldsymbol{p_4} \\ \boldsymbol{p_{44}} \\ \boldsymbol{p_{29}} \end{pmatrix}$$
$$+\left(\frac{1}{8}+\frac{1}{16}+\frac{1}{7}+\frac{1}{14}\right)\ (\boldsymbol{p_{50}}) = \boldsymbol{0}$$

Similarly, the position of nodes (49, 48, 47) can be found based on the 4-sided region [(0, 9, 1), (1, 17, 18, 19, 20), (20, 44, 40), (40, 41, 42, 43, 0)]. **Fig. 6** shows the position of these nodes derived for a given set of division numbers.

Once the position of all the boundary nodes for both primitives is established, the generation of the radial edge meshes and the quad mesh for each block can proceed as described in Li's original algorithm. **Fig. 7** shows the resulting mesh.

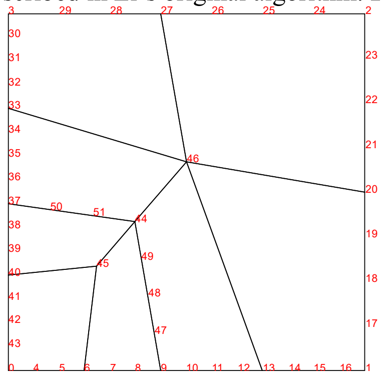


Fig. 6. Nodes on the common edge for the template of **Fig. 2** with internal division numbers [3,4,3,4,4,4].

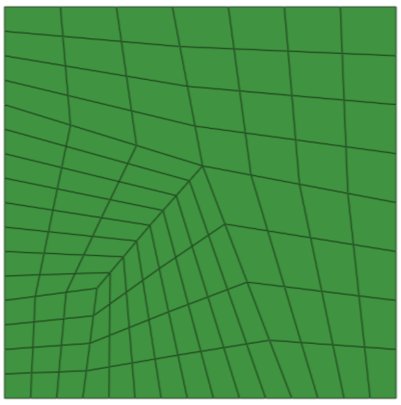


Fig. 7. Final mesh.

For a fixed set of external division numbers, the common edge midpoint is different when n_2 is varied, but if the sum $n_2 + n_3$ is fixed the singular points and the resulting element mesh are the same. **Fig. 8** and **Fig. 9** indicate the different block geometries and division numbers for two variations where $n_2 + n_3 = 5$. Thus n_2 and n_3 are not independent variables.

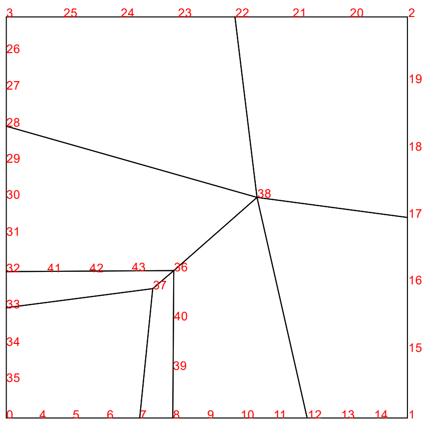


Fig. 8. Blocks for internal divisions n = [4,3,1,4,3,3]. The external numbers are [12, 6, 7, 11].

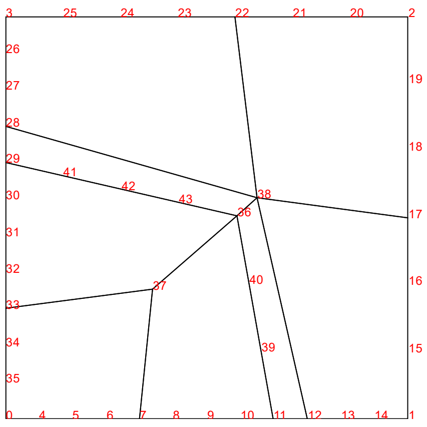


Fig. 9. n = [4,3,4,1,3,3]. Same external division numbers. The singular points 37 and 38 and the final mesh are the same as in **Fig. 8**.

6 Division number constraints

From **Fig. 2**, the internal and external division numbers are related by

$$\begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} \begin{Bmatrix} n_0 \\ n_1 \\ n_2 \\ n_3 \\ n_4 \\ n_5 \end{Bmatrix} = \begin{Bmatrix} N_0 \\ N_1 \\ N_2 \\ N_3 \end{Bmatrix}$$

Summing the rows of the matrix on the left-hand side implies

$$2(n_0 + n_1 + n_2 + n_3 + n_4 + n_5) = N_0 + N_1 + N_2 + N_3$$

which is the expected constraint that the sum of the external division numbers is even. The matrix also implies that

$$n_2 + n_3 = N_0 - N_2 = N_3 - N_1$$

This is a measure of the distance between the singularities. Note that it also requires that the difference in division numbers between opposite sides should be the same. As shown above, the same mesh is generated if the sum $(n_2 + n_3)$ is the same.

The division number constraints also imply

$$N_0 + N_1 - N_3 = n_0 + n_4 = N_2$$

This constraint can also be seen by inspecting **Fig. 2**.

Therefore, the mesh can be parameterized by the position of the edge midpoint on N_2. Varying n_0 whilst keeping the sum $n_0 + n_4$ constant produces the different blocks below for the same external division numbers.

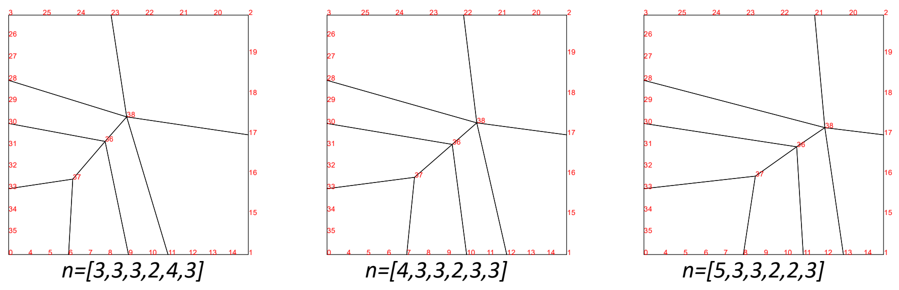


Fig. 10. Range of blocks obtained with the external division numbers [12, 6, 7, 11] whilst varying n_0 for the same $n_0 + n_4$.

Similarly

$$n_1 + n_5 = N_1$$

so the mesh can also be parameterized by where the positive singularity is attached to N_1. **Fig. 11** shows the variation in block geometry for 3 different values of n_1 for $n_1 + n_5 = 6$.

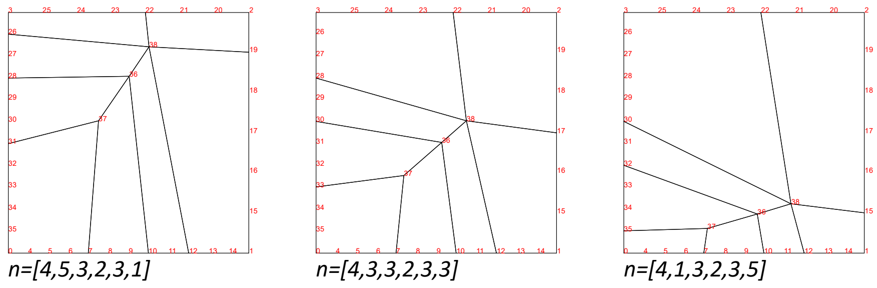


Fig. 11. Range of blocks obtained with the external division numbers [12, 6, 7, 11] whilst varying n_1 for the same $n_1 + n_5$

In summary, for a fixed set of external division numbers, the internal mesh of the template can be varied to create a wide range of meshes with different block shapes, internal element distributions etc.

7 Choosing the optimum internal divisions

The three unknown positions (for the positive singularity, the negative singularity and the common edge midpoint) can be found at low computational cost by solving 3 linear equations for 2D vectors.

For a given set of external division numbers, there are 2 parameters which can be used to vary the internal division numbers, whilst maintaining the same external division numbers:

1. The position where the positive singularity is attached to N_1, i.e. varying n_1 for a given $n_1 + n_5$
2. The position where the positive singularity is attached to N_2, i.e. varying n_0 for a given $n_0 + n_4$

Given that it is possible to compute the position of the singular points cheaply, it is feasible to estimate the mesh distortion, either exhaustively or for a substantial sample of the parametric variations. In this study, the minimum corner angle of all the blocks meeting at the singularities was used as the quality measure. **Fig. 12** show the best and worst quality of internal blocks for a fixed set of external division numbers.

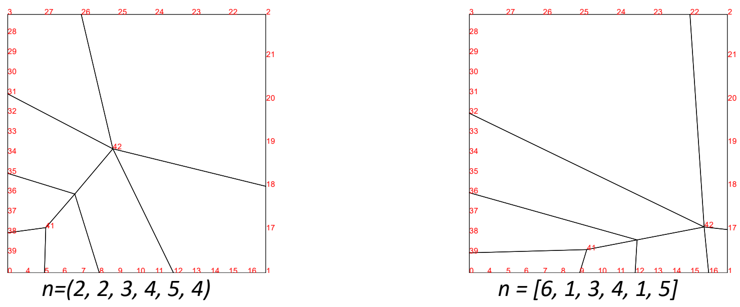


Fig. 12. The best (left) and worst (right) set of blocks obtained with the external division numbers [14,6,7,13] for all parametric variations

Since the division numbers are known before any mesh is generated, the total number of elements can be calculated. For the block arrangement in **Fig. 2**, the total number of elements is

$$n_{transition} = n_0 * n_1 + n_0 * n_2 + n_1 * n_2 + n_0 * n_3 + n_1 * n_3 + n_1 * n_4 + n_4 * n_5 + n_0 * n_5$$

The number of elements required to achieve the same maximum mesh density in a regular mesh would be

$$n_{regular} = \max(N_0, N_2) * \max(N_1, N_3)$$

Fig. 13 shows the minimum corner angle at the singularities for the meshes with this set of external division numbers. The chart also shows the ratio Size= $\frac{n_{transition}}{n_{regular}}$.

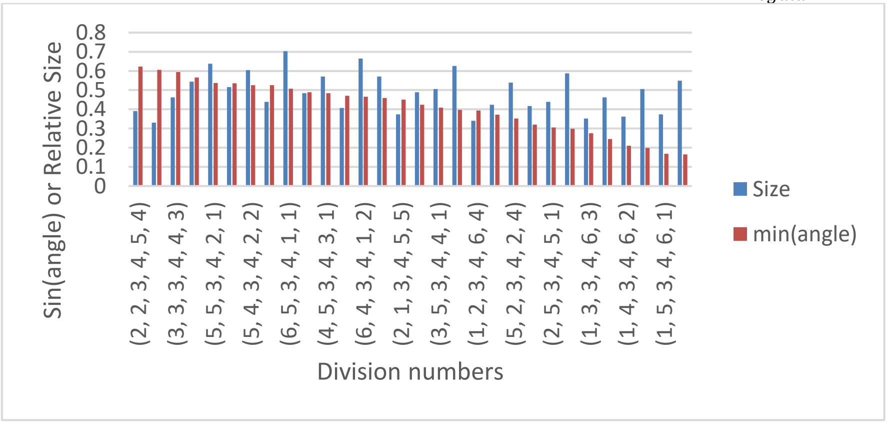


Fig. 13. Variations in mesh quality and size for all parametric variations of external division numbers [14,6,7,13]

8 A pair of singularities which aren't directly connected

Fig. 14 shows a potentially more useful transition which also has 6 internal division numbers, but without the redundant constraints between $N_0 - N_2$ and $N_1 - N_3$. By varying the internal division numbers, transitions like the previous one can be implemented. However, it also allows steerback transitions from N_2 to N_0.

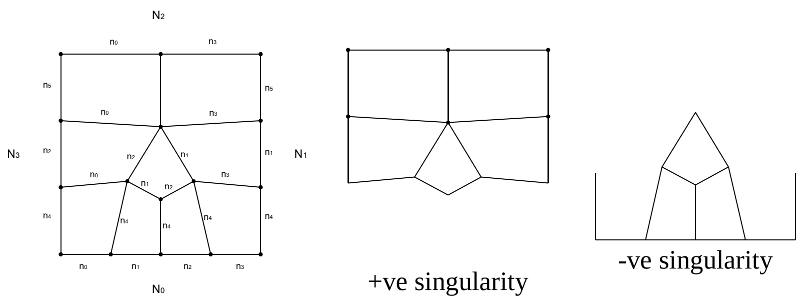


Fig. 14. A more flexible transition

The division number constraints are

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \begin{Bmatrix} n_0 \\ n_1 \\ n_2 \\ n_3 \\ n_4 \\ n_5 \end{Bmatrix} = \begin{Bmatrix} N_0 \\ N_1 \\ N_2 \\ N_3 \end{Bmatrix}$$

Solving this gives

$$n_2 = \frac{N_o - N_1 - N_2 + N_3}{2}$$

$$n_1 = \frac{N_o + N_1 - N_2 - N_3}{2}$$

$$n_4 + n_5 = \frac{-N_o + N_1 + N_2 + N_3}{2}$$

The position of the pentagon edge midpoint on the top edge can be moved by changing n_0 whilst keeping the total $n_0 + n_3 = N_2$.

The value of $n_4 + n_5$ determines how close the two singularities are – if this is large the singularities are close together. Note that $n_4 \geq 0$ and $n_5 \geq 1$. The resulting constraint that

$$-N_o + N_1 + N_2 + N_3 \geq 1$$

tells us that, if for example we have a boundary layer with a dense mesh that feeds into a far field with a much coarser mesh, once N_0 gets too big we need to add another singularity pair. **Fig. 15** show how the internal division numbers can be adjusted to obtain mesh transitions in different directions.

The points which need to be constrained to be the same between the pentagon and the triangle are the two singular points and the two points which act as edge midpoints for both the triangle and the pentagon. This gives 4 vector equations to determine the positions of the positive and negative singularities and the two triangle edge midpoints. As for the first transition described above, the set of internal division numbers to give the best mesh quality can be computed cheaply for a given block shape.

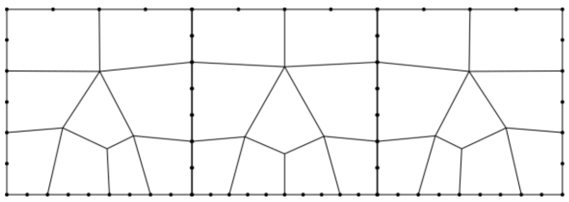


Fig. 15. Three adjacent blocks with singularity pairs adjusted to obtain a mesh transition bottom-right (left), bottom-t4op (middle) and bottom-left (right)

9 3D Transitions

3D transitions can be constructed as 2D extrusions of mesh singularity arrangements such as are shown in **Fig. 2** or **Fig. 14**.

A genuine 3D partitioning is shown in **Fig. 16**, where a single split is used to partition the block into two of the mesh-able primitives identified in [2]. Each can be meshed by midpoint subdivision [14], where each face midpoint is connected to a body midpoint. The result is a positive / negative singularity pair on 3 faces of the block, with a regular mesh on the remaining 3 faces. This can be regarded as the 3D equivalent of **Fig. 2**.

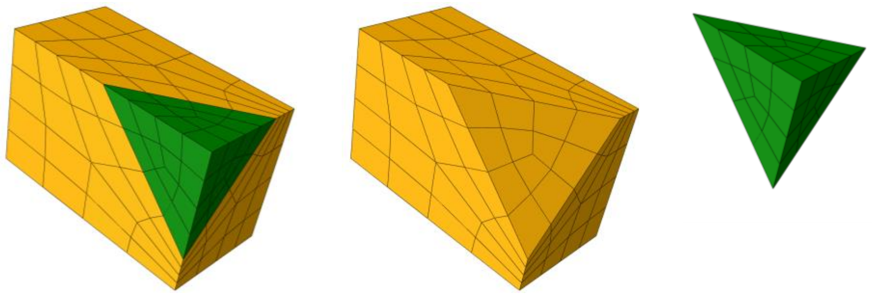


Fig. 16. 3D block refinement

The implementation shown in **Fig. 16** was created using a simple Abaqus Python script This employed the following steps:

1. a hard geometry partition was created from a plane defined by points on 3 edges of the original block. The location of these points is the nodal position which gives the

desired edge division numbers and mesh size in each of the primitives. It can therefore be used to produce different mesh distributions.

2. Both the resulting primitives, the tetrahedron and the cube-with-a chip-off, are meshed using the implementation of midpoint subdivision within Abaqus.

The Abaqus implementation enforces any necessary division number constraints on block edges in the model after midpoint subdivision. In 2D, once the radial division numbers from the singular points at the face centers are chosen, the total edge division numbers are defined and the task of the integer programming is to choose the set of radial division numbers which best fits the target external division numbers.

In 3D [15], once the radial division numbers from the singular point at the center of a primitive volume to the face centers are chosen, the radial division numbers from the singular points on the faces to the face edges and the external edge division numbers are defined. The cube-with-a chip-off has 7 faces so it contributes 7 radial division numbers. The tetrahedron has 4 faces, but with this refinement template all 3 division numbers on the common face in **Fig. 16** must be the same. It is also expected, as was shown in Section 6, that it is the total division number between the singularities which will be significant rather than the two division numbers to the partitioning face. Therefore, there are 7 radial division numbers available to satisfy the 12 edge division numbers of the original cube. Note that if a structured mesh was imposed, only 3 unique edge division numbers are available.

The Abaqus implementation partitions the block volume using a plane, so the adjustment of nodal positions using transfinite mapping as described in Sections 3 - 5 is not available, but this should be a straightforward item of future work.

To prevent propagation of the mesh singularity pairs, the same recipe can be applied to adjacent blocks. A symmetric replica on one adjacent block limits the influence of the mesh singularities to two faces, with the other 4 faces of the double-sized block having a regular mesh. Patterns of 4 blocks can be constructed to limit the mesh refinement to one face, whilst 8 would limit the effect to the interior of a volume, with all external faces having a regular mesh.

Discussion

The illustrations above are for a mesh on a unit cube or block, but the transfinite mapping can be applied to any 2D geometry defined by a sequence of 4 edge meshes, e.g. **Fig. 17**. Rotation and reflection transforms can be used to convert the meshes shown to different geometries where the external division numbers have different constraints e.g. that $N_0 < N_2$ or $N_3 < N_1$ in **Fig. 2**. Graded edge meshes are handled automatically since only the nodal positions are required, and measures of mesh quality can be computed before the mesh is generated.

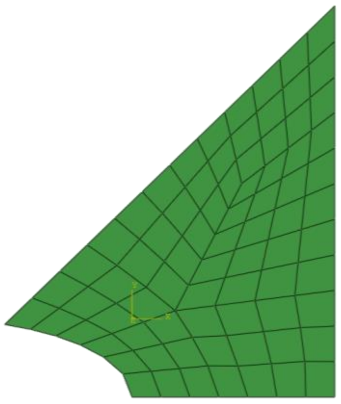


Fig. 17. Mesh in distorted block. Same division numbers as in **Fig. 8**.

Adaptive refinement of block topology and mesh, based on solution error estimates, is an obvious item of future work. There is no fundamental reason why the technique should not be applied to arbitrary block networks – in 2D all that is required is the number of block vertices connected to a given vertex. The same equations as are utilized to compute the face midpoint positions are then applicable.

Multiblock meshes are good at estimating sensitivities to a design change, since a fixed mesh topology is used. The templates here represent the minimal change in mesh topology if a better mesh distribution and solution accuracy is required. The approach could be codified into existing multiblock mesh generators.

The flexible adaptation approach described in this paper enables a wide range of external and internal mesh distributions to be accommodated for given block edge division numbers. This is highlighted in **Fig. 18**, where it is shown that assigning zero divisions to specific block edges enables the 2 and 3 refinement templates from Schneiders to be created. For example, the top left blocking is created by setting n4 as zero.

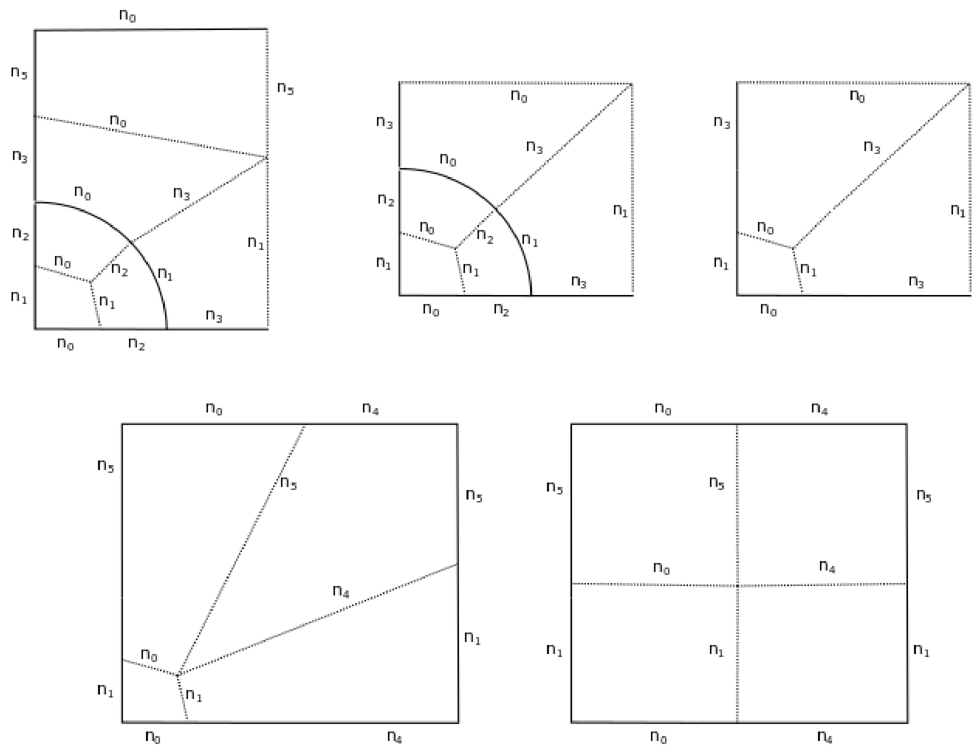


Fig. 18. Reducing block edge to zero division number

The 3-refinement template, top middle blocking, can then be created by assigning n5 as zero, and can be used to limit the transition of mesh refinement. In a next step, if n2 is assigned to be zero then the 2-refinement template is created as shown in the top right blocking. This demonstrates the flexibility of the approach described in this paper, e.g. setting both n2 and n3 to have zero division numbers generates the structured quadtree decomposition shown in the blocking of **Fig. 19**, whose quality can be controlled by altering the division constraints.

10 Conclusions

Structured multiblock decompositions are very expensive to create in terms of engineering time. Adapting the mesh density to improve solution quality in a multiblock decomposition usually requires a change in block division numbers, which has a global effect on the mesh and the solution problem size.

Local templates for block refinement can be constructed which can be implemented cheaply and have only a local effect on mesh density and block structure. For a given set of external division numbers on a block, there is a parametric family of solutions for the internal division numbers in the refinement template.

Using an extended transfinite mapping scheme, a mesh can be generated is equivalent to that which would be generated by iterative iso-parametric smoothing. Whilst the quality may be inferior to that generated using more sophisticated smoothing algorithms, the optimum set of internal division numbers and the validity and quality of the final mesh can be guaranteed before the mesh is generated. It could therefore serve as an initialization step before other smoothing algorithms are applied.

The template refinement procedure adds the minimum number of mesh singularities e.g. a positive and negative singularity pair in 2D. Despite this, a wide range of mesh distributions can be accommodated.

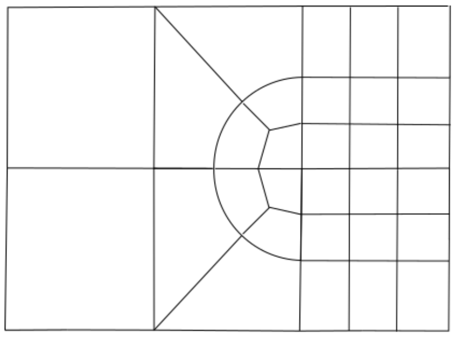


Fig. 19. 3-refinement adaptation

References

[1] A. Loseille, "Recent Improvements on Cavity-Based Operators for RANS Mesh Adaptation," in *2018 AIAA Aerospace Sciences Meeting*, American Institute of Aeronautics and Astronautics, 2018.

[2] M. A. Price, C. G. Armstrong, and M. A. Sabin, "Hexahedral mesh generation by medial surface subdivision: Part I. Solids with convex edges," *Int. J. Numer. Methods Eng.*, vol. 38, no. 19, pp. 3335–3359, 1995.

[3] Z. Ali, P. C. Dhanasekaran, P. G. Tucker, R. Watson, and S. Shahpar, "Optimal multi-block mesh generation for CFD," *Int. J. Comput. Fluid Dyn.*, vol. 31, no 4-5, pp 195-213, 2017.

[4] W. R. Quadros, "LayTracks3D: A new approach for meshing general solids using medial axis transform," *CAD Comput. Aided Des.*, vol. 72, pp. 102–117, 2016.

[5] N. Kowalski, F. Ledoux, and P. Frey, "Smoothness driven frame field generation for hexahedral meshing,"*CAD Comput. Aided Des.*, vol. 7772, pp 65-77, Mar. 2016.

[6] Gridpro, "Multi-Scale Tools," *GridPro website*. [Online]. Available: https://www.gridpro.com/gridpro-advantages. [Accessed: 27-May-2018].

[7] A. Keskin *et al.*, "On the quantification of errors of a pre-processing effort reducing contact meshing approach," in *53rd AIAA Aerospace Sciences*

Meeting, American Institute of Aeronautics and Astronautics, 2015.

[8] J. S. Sandhu, F. C. M. Menandro, H. Liebowitz, and E. T. Moyer, “Hierarchical mesh adaptation of 2D quadrilateral elements,” *Eng. Fract. Mech.*,vol. 50, no. 5/6, pp. 727-736, Mar.-Apr. 1995.

[9] R. Schneiders, “Refining Quadrilateral and Hexahedral Element Meshes,” in *5th International Conference on Grid Generation in Computational Field Simulations*, 1996, pp. 679–688.

[10] M. S. Ebeida, A. Patney, J. D. Owens, and E. Mestreau, “Isotropic conforming refinement of quadrilateral and hexahedral meshes using two-refinement templates,” *Int. J. Numer. Methods Eng.*, vol. 88, no. 10, pp 974-985, Dec. 2011.

[11] J. Qian and Y. Zhang, “Automatic unstructured all-hexahedral mesh generation from B-Reps for non-manifold CAD assemblies,” *Eng. Comput.*, vol. 28, no. 4, pp. 345–359, Oct. 2012.

[12] B. D. Anderson, S. E. Benzley, and S. J. Owen, “Automatic all quadrilateral mesh adaption through refinement and coarsening,” in *Proceedings of the 18th International Meshing Roundtable, IMR 2009*, 2009.

[13] H. J. Fogg, L. Sun, J. E. Makem, C. G. Armstrong, and T. T. Robinson, “Singularities in structured meshes and cross-fields,” *CAD Comp. Aided Des.*, vol.105, pp 11-25, Dec. 2018.

[14] T. S. Li, R. M. McKeag, and C. G. Armstrong, “Hexahedral meshing using midpoint subdivision and integer programming,” *Comput. Methods Appl. Mech. Eng.*, vol. 124, no. 1–2, pp. 171–193, Jun. 1995.

[15] T. S. Li, C. G. Armstrong, and R. M. McKeag, “Quad mesh generation for k-sided faces and hex mesh generation for trivalent polyhedra,” *Finite Elem. Anal. Des.*, vol. 26, no. 4, pp. 279–301, Aug. 1997.