

A Modular Design for a Parallel Multifrontal Mesh Generator

Jean-Paul Boufflet¹, Piotr Breitkopf², Alain Rassineux², and Pierre Villon²

¹ UMR 6599 HeuDiaSyc, Université de Technologie de Compiègne,
60205 Compiègne Cedex, France,
`{Jean-Paul.Boufflet}@utc.fr`

² UMR 6066 Roberval, Université de Technologie de Compiègne,
60205 Compiègne Cedex, France,
`{Alain.Rassineux,Pierre.Villon,Piotr.Breitkopf}@utc.fr`

Abstract. The proposed approach consists in extending an existing sequential mesh generator in order to design a parallel mesh generator. A sequential three-dimensional mesh generator builds the internal mesh by applying advancing front techniques from a surface mesh defining the volume of the initial domain. We geometrically decompose the domain by recursively splitting it into subdomains using cutting planes. We run a sequential mesh generator code on different processors, each of them working on a single subdomain. The interface mesh compatibility on the two sides of two subdomains makes it possible to merge the results. We present in this paper our modular approach, an example of a interface mesh generation, and we specify the cutting plane decomposition method.

1 Introduction

The goal of this paper is to present a parallel algorithm of volume mesh generation in 3D. The data is the surface triangular mesh. Two strategies are possible for the generation of internal tetrahedral:

1. parallelize a mesh generation code;
2. decompose the data.

The first strategy is subject to current investigation [1]. In the current work we develop an original technique for the second strategy – the domain decomposition of the surface mesh prior to the volume mesh generation. The obvious benefit of this approach is the re-use of an existing sequential volume mesh generator [5] in parallel over the subdomains. Several issues have however to be addressed. The most important is that splitting of a closed envelope gives open subdomains. We have therefore to generate an interface surface mesh in order to close the subdomains. The second issue is the quality of the resulting volume mesh obtained after merging the subdomains. The special load balancing criteria have also to be defined. The chosen approach may be resumed as follows:

1. we define a “cutting plane” for partitioning the initial surface;
2. we generate the interface surface mesh;
3. we run the sequential volume mesh generator over the subdomains;
4. we merge the subdomain meshes in order to obtain the global volume mesh;
5. we apply the mesh optimization techniques in order to meet the mesh quality criteria in vicinity of the interface.

The actual approach should not be confused with the established mesh partitioning techniques [7,8,9] developed for the computational meshes. In the primary stage of our algorithm the volume mesh does not yet exist and the only information is the triangulation of the envelope. We could obviously use the standard domain decomposition code in order to split the initial data. Two problems would however arise:

- the interface surface mesh has to be generated anyway;
- we have no guarantee that the interface could be easily meshes with surface elements.

Therefore, we propose to simplify the interface geometry by using an explicit dividing surface - in the actual work, a cutting plane. The benefit of this strategy is the possibility of re-use of a standard surface (plane) mesh generator [6] for the interface. The drawback is the impact of the cutting plane on the form of neighbouring volume elements. This issue is treated by usual techniques of tetrahedral mesh optimization [6].

The current work may be related to the technique of Recursive Inertial Bisection (RIB) [2,3]. The fundamental differences between the two techniques is in the definition of the cutting plane by the Moving Least Square (MLS) approximation [10] and in the strategy of updating the cutting plane position and direction. The MLS approximation permits to get a smooth evolution of the cutting plane based on the local information. In fact, only the points “close enough” to the current cutting plane are taken into account and their influence decreases with the distance. This approach permits to treat more complex and multiply connected geometries corresponding to industrial parts. Rather than the domain decomposition, this paper concerns the envelope splitting and interface generation for a standard volume mesh generator. The usual domain decomposition techniques [7,8,9] may be used at the initialisation stage.

2 The Modular Design Strategy

The overall process is presented in figure 1. From left to right clockwise, there is the data flow through the modules.

The first module decomposes a geometric domain into two subdomains by computing a cutting plane as we detail in section 4. The second module generates a triangular mesh of the interface.

At this stage, we have generated two subdomains having a complete surface mesh by building a compatible interface mesh on the two sides of the cutting

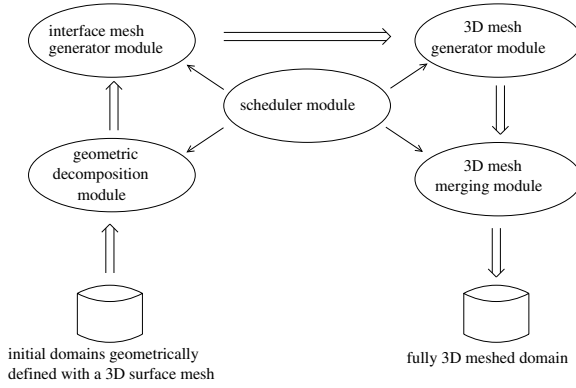


Fig. 1. The modules of the parallel multifrontal mesh generator

plane for the two subdomains that ensures merging the results. Two processors using the sequential mesh generator can generate the internal meshes of the two subdomains in parallel. The sequential mesh generator is the third module. The fourth module merges the meshes to obtain the full mesh.

One can recursively generate the subdomains by applying the first and the second module, and then compute each sub-mesh using the sequential mesh generator. In this case the merging module has to be accordingly invoked. Consequently a scheduling module has to pilot the process.

3 The Interface Mesh Generation

At this stage the goal is to generate the surface mesh in order to close the two open subdomains obtained by applying the cutting plane Π to the domain D . We perform as follows:

1. we define the interface surface nodes that are close to Π ;
2. we project this interface nodes to Π , that defines the geometry needed for the surface mesh generator;
3. we generate a surface mesh using this geometry with a standard 2D mesh generator [6];
4. we fit this surface mesh to the coordinates of the interface nodes.

The step 1 can be achieved by assigning first the triangles of the envelope to three sets:

- S_1 : the triangular finite element on one side of Π ;
- S_2 : the triangular finite element on the other side of Π ;
- S_3 : the triangular finite element intersected by Π .

Then, using the geometric informations, we assign each triangular finite elements of S_3 either to S_1 or S_2 . We therefore obtain the nodes defining the

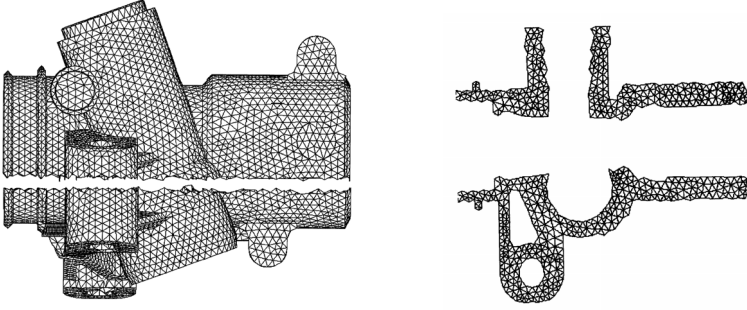


Fig. 2. An example of a interface mesh generation: on the left the two subdomains obtained by applying a cutting plane, on the right the interface mesh generated at the boundary

boundary between the two subdomains. Then, we project the nodes on Π (step 2). These projections define a two-dimensionnal geometry and a two-dimensional mesher can be applied on this classical problem [6]. At this stage we obtain a set of plane two dimensionnal meshes composed of new triangular finite elements. Finally, by applying geometrical transformations, we adapt the new triangles near the interface nodes in order to fit with the three-dimensional shape of the boundary. Consequently the interface mesh generated is not plane and can be composed of several part if there are holes.

By applying this process we build a unique interface mesh that geometrically fits. By supplementing \mathcal{S}_1 and \mathcal{S}_2 with this interface mesh we finally obtain two subdomains with two complete envelopes.

On the left of figure 2 we show an example of two subdomains obtained by applying the process. One can notice the “cutted eggshell” shape aspect on the boundary: we do not split the triangular finite elements of the initial envelope. On this exemple the cutting plane Π applied to the complex geometry of the initial domain leads to several areas with dense parts and holes. However, we show on the right of figure 2 that the interface mesh is correctly generated at the boundary between the subdomains.

Whatever the results of the three-dimensional meshing applied on these two new complete envelopes can be, the unique interface mesh generated for the two subdomains ensures the perfect merging of the two results.

4 Geometric Decomposition

The decomposition method that we present in this section exploits only the geometric information of the nodes of the surface mesh. We choose to partition a geometric domain by splitting it into two parts balancing the number of nodes on both sides of the cutting plane. We have to choose the cutting plane defined by its direction and its position with regard to the domain we want to partition in order to satisfy the quality criteria of the submeshes.

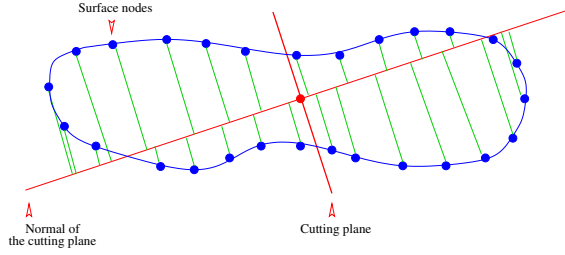


Fig. 3. An example of a domain D with the cutting plane principle

Let D be a domain having a surface mesh (its envelope) such that its volume is defined by the n nodes of the surface (cf. figure 3). Each node X_i is defined by its Cartesian coordinates x_i , y_i and z_i (for $i = 1, \dots, n$). The purpose is to determine the mean plane that ensures a balanced partition of the domain. First, we calculate the centre of gravity \bar{X} :

$$\bar{X} = (\bar{x}, \bar{y}, \bar{z}) = \left(\frac{1}{n} \sum_i x_i, \frac{1}{n} \sum_i y_i, \frac{1}{n} \sum_i z_i \right) \quad (1)$$

Then, we compute the nodal coordinates centered around \bar{X} . For each node X_i :

$$(\xi_i, \eta_i, \nu_i) = (x_i - \bar{x}, y_i - \bar{y}, z_i - \bar{z}) \quad \text{for } i = 1, \dots, n \quad (2)$$

We now determine the plane passing through the point \bar{X} and as near as possible to the nodes X_i . The equation of this plane is *a priori*:

$$a(x - \bar{x}) + b(y - \bar{y}) + c(z - \bar{z}) = 0 \quad (3)$$

We have to compute the triplet (a, b, c) minimizing the following cost:

$$J(a, b, c) = \frac{1}{2} \sum_i w_i (a \xi_i + b \eta_i + c \nu_i)^2 \quad (4)$$

with respect to the constraint:

$$a^2 + b^2 + c^2 = 1 \quad (5)$$

where the weights are $w_i = w_{ref}(dist(h(X_i), \bar{X})/r)$. The term $h(X_i)$ is the projection of each node X_i to the normal of the cutting plane passing through the point \bar{X} (cf. figure 3). The quantity r is a reference radius that permits the selection of the nodes contributing to the calculus of the weights w_i . The function w_{ref} is the reference attenuation function. For instance, one can choose $w_{ref}(d) = 0.5(1 + \cos(\pi d))$ for $d \in [0, 1]$, and 0 otherwise. This function permits to assign higher weights to the nodes near \bar{X} for building the plane.

In order to compute the plane we set $\alpha^T = (a, b, c)$ and we built the matrices P and W as follows:

$$P = \begin{bmatrix} \xi_1 & \eta_1 & \nu_1 \\ \xi_2 & \eta_2 & \nu_2 \\ \vdots & \vdots & \vdots \\ \xi_n & \eta_n & \nu_n \end{bmatrix} \quad W = \begin{bmatrix} w_1 & 0 & \cdots & 0 \\ 0 & \ddots & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & w_n \end{bmatrix}$$

Considering these notations the equations 4 and 5 can be formulated as: $\min_{\alpha} J(\alpha)$ under the constraint $\alpha^T \alpha = 1$ and where $J(\alpha) = \frac{1}{2} \alpha^T P^T W P \alpha$. The expression of the Lagrangian of this problem is:

$$\mathcal{L}(\alpha, \lambda) = \frac{1}{2} \alpha^T P^T W P \alpha - \lambda \alpha^T \alpha \quad (6)$$

and the associated optimality system is:

$$P^T W P \alpha - \lambda \alpha = 0 \quad (7)$$

$$\alpha^T \alpha = 1 \quad (8)$$

The equation 7 is equivalent to $(\lambda I - P^T W P) \alpha = 0$. That is to say (λ, α) are respectively eigenvalues and eigenvectors of the matrix $P^T W P$. This matrix is positive definite symmetric, therefore there are 3 real eigenvalues $\lambda_1, \lambda_2, \lambda_3$ such that $0 < \lambda_1 \leq \lambda_2 \leq \lambda_3$ and 3 associated orthonormal eigenvectors. Consequently the solution of the system of equation (7-8) is either (λ_1, α_1) or (λ_2, α_2) or (λ_3, α_3) . These three solutions represent the *extrema* of the cost function $J(\alpha)$. In order to find the solution that minimizes, we evaluate $J(\alpha_j)$ for $(j = 1, 2, 3)$.

It follows:

$$J(\alpha_j) = \frac{1}{2} \alpha_j^T P^T W P \alpha_j = \frac{1}{2} \alpha_j^T \lambda_j \alpha_j = \frac{1}{2} \lambda_j \alpha_j^T \alpha_j = \frac{1}{2} \lambda_j \quad (9)$$

To summarize, the vector $\alpha = (a, b, c)$ minimizing $J(\alpha)$ corresponds to the normalized eigenvector associated with the smallest eigenvalue of the matrix $P^T W P$. From a practical point of view, we use the inverse iterate power method to compute an approximation of the eigenvector associated with the smallest modulus eigenvalue.

5 The Partitioning Algorithm

The outline of the algorithm consists in bringing up the cutting plane Π from an initial position until we obtain a decomposition of the domain into two equal parts subdomains.

The cutting plane is characterized by the point \bar{Y} and a normal direction α (cf. former section). The initialization step includes the load of the coordinates of the nodes of the surface mesh of the domain (the envelope) from data files and the determination of the initial cutting plane. This initial cutting plane passes

through the centre of gravity \bar{X} (cf. equation 1) and the associated normal vector is the normalized eigenvector associated with the smallest eigenvalue of the matrix $P^T P$. At this initialization step we take $W = I$.

The main loop that updates the cutting plane is as follows:

- select the nodes X_i that are the nearest from the intersection of the cutting plane with the envelope;
- compute their centre of gravity \bar{Y} ;
- compute the projections Y_i of all the nodes X_i to the straight line Δ in normal direction α from the cutting plane;
- sort the points Y_i in an increasing order of distance from \bar{Y} ;
- compute the weights $w_i = w_{ref}(dist(\bar{Y} - Y_i)/r)$;
- build the matrix $P^T W P$;
- compute the new normal vector α , (normalized eigenvector associated with the smallest eigenvalue of $P^T W P$);
- actualize the cutting plane.

The actualization of the cutting plane is performed as follows: the new cutting plane passes through the point X_k and its new normal vector is the new normal vector α . The point X_k has the point Y_k as its projection to the straight line Δ such that Y_k is the nearest point from \bar{Y} decreasing the balancing criterion. We define the balancing criterion as $C(\Pi) = \frac{abs(n_1 - n_2)}{n_1 + n_2}$ where n_1 is the number of nodes of the surface mesh (the envelope) located at the side with direction $+\alpha$ from the cutting plane (while n_2 is the number of nodes located at the other side with direction $-\alpha$ from the cutting plane).

The main loop of the algorithm stops when the criterion $C(\Pi)$ is lower or equal than a chosen threshold. The parameter r we defined for computing the weights w_i can evolve during the process in order to escape from local minima or to avoid cycling.

We tested our algorithm on three examples having complex geometric shapes. On the right part of figure 4, we use two level of grey to identify the subdomains. On the left part, for each mesh, we report the graph of the percentage of unbalance over the two subdomains as a function of the number of iterations of the main loop.

We observe a rapid convergence to a cutting plane that balances the number of surface nodes over the two subdomains. We also notice that the stopping criterion we proposed for the algorithm depends on the detection of the first local minimum encountered before stabilizing. We reach a relative gap roughly of 2 % between the number of nodes by spending a computing time between one and two minutes on a SUN Solaris workstation, that is satisfying with respect to the time spent by the three-dimensional mesh generator on the domain.

The cutting plane Π may geometrically split the surface mesh into more than two subdomains. The figure 5 show an example of such situation. We observe on the left part of figure 5, more precisely top left near the cutting plane, that a small part of the surface mesh has been assigned to the dark grey subdomain. There is no connection with the other part of the dark grey subdomain. However, the two separate dark grey parts belong to the same side of Π . We associate a graph

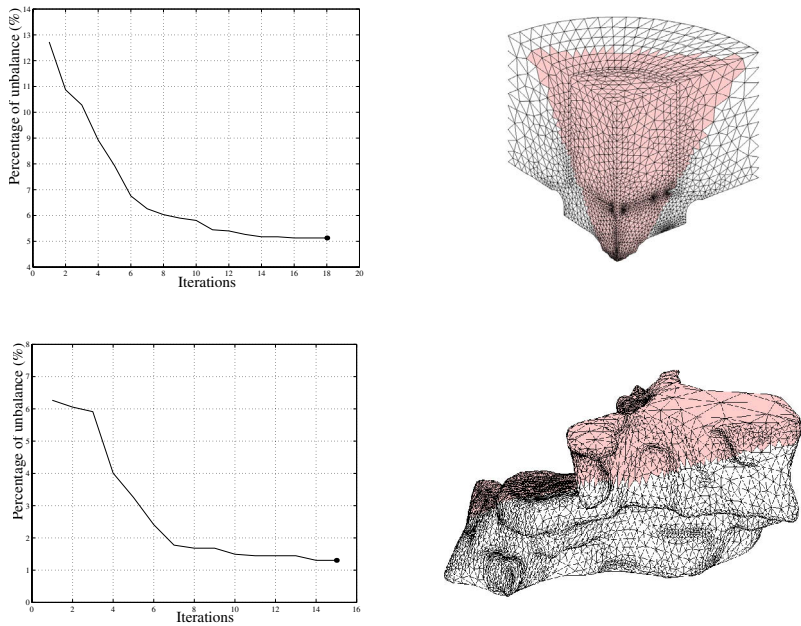


Fig. 4. Two examples of geometric partitioning (light grey and grey parts)

$G(D)$ with the surface mesh of D . The post-processing consists in detecting first all the connected components issued from Π in $G(D)$. Considering the exemple of figure 5 we obtain three connected components. Next, we merge the smallest connected component with its biggest neighbour. The right part of figure 5 shows the repaired mesh. By assigning correctly the parts we finally obtain two connected components. This can be done by applying basic graph algorithms on the parts of the surface mesh issued from the cutting plane.

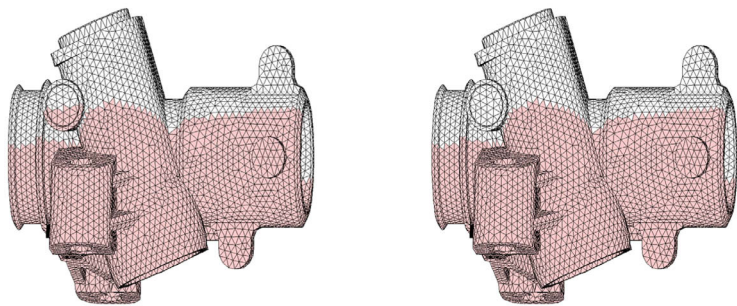


Fig. 5. On the left, the initial geometric domain decomposition, on the right after detection of the connected composant and re-assigning (Bench PSA)

6 Conclusion

The three main modules needed to design the parallel multifrontal mesh generator are: the sequential multifrontal mesh generator, the geometric decomposition and the interface mesh generator. The preliminary results have to be confirmed on other benchmarks and have to be compared with the meshes computed using the sequential mesh generator alone. The behavior of our approach has to be studied concerning the load balancing in order to design the scheduling module.

References

1. Chen M-B, Chuang T-R, Wu J-J.: Experience in parallelizing mesh generation code with High Performance Fortran. In 9th SIAM Conference on Parallel Processing for Scientific Computing. San Antonio, Texas, USA, SIAM Press. March (1999).
2. Hendrickson B., Devine K.: Dynamic Load Balancing in Computational Mechanics. *Comp. Meth. Applied Mechanics & Engineering*. 184(2-4):485-500, (2000).
3. Simon H. D. Partitioning of unstructured problems for parallel processing. *Computing Systems in Engineering*, 2(2/3):135-148, 1991.
4. Bouattoura D., Boufflet J.P., Rassineux A., Villon P. : Mailleur Parallèle Multifrontal. *Proceedings of Quatrième Colloque National en Calcul des Structures*, (1999) 297-302
5. Rassineux A.: 3D Mesh Adaptation - Optimization of Tetrahedral Meshes by Advancing front Technique. *Compt. Methd. Appli. Mech. Eng.*, Vol. 141. (1997)
6. Frey P. J., George P-L.: Maillages, applications aux éléments finis. HERMES Science Publication (1999).
7. Karypis G. and Kumar V. METIS : A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices. Tech. Report University of Minnesota, Department of Computer Science, sept, (1998).
8. Pellegrini F. SCOTCH 3.0 User's guide. Tech. Report LaBRI, URA CNRS 1304, Université Bordeaux I,(1996).
9. Hendrickson B. and Leland R. The *Chaco* user's guide, version 2.0. Tech. Report Sandia National Laboratories, Albuquerque SAND95-2344, jul, (1995).
10. Lancaster, P., Salkauskas, K., Surfaces Generated by Moving Least Squares Methods, *Math. of Comp.* Vol, 155, pp. 141-158, (1981).