

CSALF-Q: A Bricolage Algorithm for Anisotropic Quad Mesh Generation

Nilanjan Mukherjee

Meshing & Abstraction Group
Digital Simulation Solutions
Siemens PLM Software

SIEMENS

2000 Eastman Dr., Milford, Ohio 45150 USA

{mukherjee.nilanjan@siemens.com}

Abstract: “Bricolage”, a multidisciplinary term used extensively in visual arts, anthropology and cultural studies, refers to a creation that borrows elements from a diverse range of existing designs. The term is applied here to describe CSALF-Q, a new automatic 2D quad meshing algorithm that combines the strengths of recursive subdivision, loop-paving and transfinite interpolation to generate surface meshes that are anisotropic yet boundary structured, robust yet sensitive to size fields and competitive in performance. This algorithm is based on the understanding of boundary loops which are initially meshed recursively with a new, boundary based “loop-paving” technique that balances anisotropy and mesh quality. The remaining interior domain is filled out by a symbiotic process that shuttles between recursive subdivision, transfinite interpolation and loop-paving in an efficient manner. The mesh is finally cleaned and smoothed. Loop-fronts are classified and rule sets are defined for each, to aid optimum point placement. Stencils used for loop-closure are presented. Results are presented that compare both local and global element quality of meshes generated by the new algorithm with that of TQM (TriaQua Mesher) which is also known to handle variegated sizemaps.

Keywords: subdivision, tri-qua mesher, paving, transfinite, mapped, flattening, loop, loop-paving, parameterization, quadrilateral.

1 Introduction

Unstructured quadrilateral and hex mesh generation of a large gamut of industrial problems ranging from automotive, aerospace, electronic and appliance structures and thermal and fluid flow problems continue to be a key pre-processing step leading to the complex analyses performed for design validation. In the industry today, we continue to faithfully use just a handful of quad mesh generation algorithms. Added to the existing complexity of engineering demands and interaction problems there is a need to create hybrid meshes that capture quintessential properties of several mesh generation algorithms, thus enabling the application engineer or analyst to produce meshes that largely meet both diverse local and global requirements. This “hybridity”, called for thus, is defined in two ways - first, a mix of elements (quad or hex-dominant meshes) and secondly

quad/hex or quad/hex-dominant meshes confronted with stringent needs of variegated local and global size fields thus requiring challenging anisotropy.

2. Past Research

Owen [1] and Frey and George [2] present detailed overviews on unstructured meshing algorithms both from historical and practical perspectives. The basic work on unstructured mesh generation with quadrilaterals could be classified into the following:

1. Transfinite methods
2. Paving or Advancing front type methods
3. Quadtree methods
4. Subdivision or domain decomposition methods
5. Medial Axis methods
6. Sphere/ball/circle packing methods

This paper will only focus on paving, transfinite and subdivision algorithms.

2.1 Advancing Front Methods

The advancing front approach to domain triangulation was introduced by Marcum [3]. Ted Blacker and Stephenson [4] published their ground breaking work on a quadrilateral advancing front approach they called *Paving*. Paving and advancing front meshes guarantee a boundary conforming mesh with a very high quality. However, they do not guarantee a "controllable layered mesh". There may not exist a completely structured layer of elements around boundary loops - as a result properties of such layers cannot be user-controlled. Staten et al extended the unstructured paving approach to 3D which they called "plastering" [5]. Earlier White and Kinney [6] had attempted to progress a single element row until collision occurs. Although an interesting approach, the method does not help to improve quad element quality and nor guarantee controllable structured layers of high quality quads near interior boundaries. With triangles however, Pirzadeh [7] provided an algorithm to advance the boundary layers in a controlled and structured manner. Peraire et. al [8] made another important contribution in this class of problems by laying down the essential 3D Euler equations for boundary layer advancement. Very recently, Moreno et al [9] reported an improvisation of the paving algorithm that creates continuous and homogeneous curved triangular and quadrilateral meshes directly on NURBS geometry. Another strong limitation of the paving algorithm is its inability to handle complex sizemaps not making it the obvious choice for highly anisotropic quadrilateral meshes.

2.2 Subdivision Methods

Subdivision algorithms dissect a given polygon into either a simultaneous set of convex domains or a recursively split dynamic domain. The main idea here is to generate a best-split-line to subdivide the polygon into smaller areas that are pre-dominantly convex. The polygonal postulate that states that a convex polygon, when linearly truncated, leads to only convex domains, serves as the basis for this approach. Since convex polygon domain yields a dominant convex mesh, the mesh quality produced by these algorithms is

envisably high. It needs to be categorically pointed out that subdivision methods pre-existed advancing-front types and some of these provided the first automatic method of discretizing areas for the purpose of finite element analyses. There are two basic approaches with subdivision - (i) recursive subdivision and (ii) simultaneous subdivision. Each one has its strength and limitations. Sluiter and Hansen [10] and Schoofs et al. [11] made pioneering contributions in this area of recursive subdivision. Sluiter's work was later revisited and revised by many including Talbert and Parkinson [12], Sarrate and Huerta [13] and Cabello [14]. The strength of these algorithms lie in their robustness in handling complex shapes, constraints, sizemaps and efficiency. The meshes produced have reasonable good quality. Mezentsev et al [15] described a generic approach to unstructured mesh generation on subdivision geometry. Berzin et al [16] developed in recent years a subdivision technique based on Modified Butterfly interpolation scheme for triangular mesh generation. Barry Joe [17], in his public domain code GeomPack, proposes a method to proceed from a simultaneous set of convex sub-domains. A similar methodology was proposed by Nowotny [18].

3. CSALF-Q

Automotive and aerospace structural analysts from the engine and transmission areas have long voiced their need for layered boundary conforming meshes in generally unstructured domains. To be able to have suitable control on the parameters of these layered meshes is a key requirement. Another need has been transitioning meshes, well-adapted to surface curvature and local constraints. User control, especially in the boundary layers is also key to these analyses.

In a 1962 essay titled "The Savage Mind" [19] legendary French anthropologist Claude Lévi-Strauss used the word **bricolage** to mean "make creative and resourceful use of whatever materials are at hand". Since then, the term "bricolage" has been used extensively in a large range of disciplines including science, visual arts and engineering to suggest construction or creation of a work from a diverse range of existing designs or algorithms. In the present paper, a similar effort is made to create a new hybrid quadrilateral mesher in 2D that combines two mutually complimentary aspects of unstructured meshing. One of them is a recursive subdivision algorithm. The other is a new paving loop-front method. The original recursive subdivision algorithm **TriQuaMesh**, reported more than three decades back by Sluiter and Hansen [10] emphasizes on a recursive contour or loop-splitting algorithm that produces reasonably good quality quad meshes in an automatic mode. An industrial version of the algorithm has been improvised in the **I-DEAS** and **NX** softwares. The algorithm has had more than a quarter-century long industrial mileage and is well known as one of the most general purpose and robust surface meshing codes in the industry. However, this technique does not create boundary-structured quad meshes like paving. Paving, on the other hand, does not adapt to large size transitions very effectively and is known to be susceptible to the presence of a large number of interior point constraints – a dire necessity for body-in-white meshing in the car industry. In this paper, the traditional paving technique of Blacker and Stephenson [4] is improvised to create an advancing loop-paving-loop algorithm that is coupled to the modified recursive subdivision technique. This results in CSALF (Combined Subdivision And Loop-Front) mesher. A triangular version of the same algorithm has been recently reported elsewhere [20]. It is a *bricolage* mesher that can produce anisotropic unstructured meshes that are predominantly boundary structured. The user can optionally control

layer advancement in terms of number and thickness. The mesh, however, has an overall unstructured nature and elegantly adapts to sharply varying size fields and can honor any number of valid interior point constraints.

Given a precisely discretized (isotropic or anisotropic) polygonal boundary, representing the area to be meshed, the proposed hybrid mesher algorithm initiates the meshing process with an advancing loop front method. The initial goal is to generate high-quality boundary conforming layered meshes. As the layer penetrates into the mesh area, new loops are created each time, reducing the mesh area. As the new loops begin overlapping, the advancing loop front algorithm is terminated and the subdivision algorithm is activated as shown in Fig. 1.

At this point, all loops are connected into a single continuous contour following procedures described by Cabello [14]. All interior point constraints are treated as a single-point loop. Next, the contour is split recursively by determining a best-splitting line. The splitting line dissects the compound contour into a left sub-contour and a right sub-contour (Fig 4.) Each left sub-contour is now recursively split into two children sub-contours - one on the right and another one on the left. The split lines are so chosen that they make angles nearing 90 degrees with the contour boundaries.

Recursive splitting finally fills out the mesh area.

The overall algorithmic logic can be expressed in two phases represented by the following twin-algorithms –

ALGORITHM I : Generating a single continuous contour-loop

1. The 3D faceted surface is first flattened into a 2D domain following domain generation procedures as discussed recently by Beatty and myself [21].
2. Face-interior features like holes, scars (loops with repeated edges) and convex-shaped inner loops are identified.
3. A local and global size-map is developed over the 2D domain. Number of paving layers around inner loops are determined based on feature recognition, proximity to other boundaries, global size-map and local user-driven size criteria.
4. Nodes are generated on the boundary of the face accordingly and a triangular background mesh (based on the TQM algorithm, [15]) is created for size-sampling.
5. A mesh area is defined in a 2D domain with **N** meshed loops
6. The **N** meshed loops represent the area boundary and are unique and non-intersecting
7. While loops are non-intersecting and number of layers defined on various loops are not exhausted
 - { 7a. Cycle all **N** loops, inner loops first, the outer loop last
 - {
 - 7aa. Activate the loop-paving-front algorithm on the **i-th** loop
 - 7ab. Check if the new loop self-intersects. If it does, turn off the loop-paver for this loop and go to 7ac.
 - 7ac. Continue the cycle, go to the next loop
 - }
8. Connect all **N** loops into a continuous contour **C**

ALGORITHM II : The Symbiotic Triad

To fill the single continuous contour-loop C with a quad or quad-dominant mesh, three meshing algorithms are used in tandem in a symbiotic manner - Recursive Subdivision, Transfinite Interpolation and Loop-Paving following a stratagem described by the flow-chart in Fig.1. As soon as the contourloop is split, it results in two child contourloops - a left loop C_L and a right loop C_R .

If the left contourloop C_L is convex, meets the sizemap limitations (as explained in section 7) and is map-meshable, the transfinite mesher is called. Otherwise it is loop-paved, provided all apriori and posteriori checks pass. The new loop produced by loop-paving, or the unaffected original loop is returned to the recursive subdivider. This symbiotic handling of the loops continues between the three meshing algorithms until there is no right loop and the left loop is completely filled. The right loop is now split into two loops again and the process is repeated until the complete mesh-area is filled out. Some regions of the 2D mesh generated is topologically cleaned by a mesh cleaner [22] and smoothed using a variational smoother [23].

4. Paving Loop-Front

The paving loop front approach differs from traditional advancing fronts in the sense that a series of “loop fronts” are considered for mesh advancement. When all loop fronts of a loop are advanced successfully, a new loop results that imitates and preserves the profile of the starting loop. This results in a perfectly layered mesh near inner and outer boundaries of the mesh area – a feature which is often preferred by structural and computational fluid dynamics analysts. Each loop front as shown in Fig 2., consists of a real segment BC and a virtual segment AB (represented by the previous element edge on the loop). The virtual segment represents a trailing edge of the real segment of the previous loop-front.

The virtual segment helps pave the loop front. The angle θ , between the real and virtual segments of the loop front is used to characterize the loop front. Table I lists the various loop fronts supported by the algorithm. Q denotes the new node that is created upstream of the loop-front, P denotes the node downstream of the loop-front that mostly pre-exists (created in certain special cases). All additional new nodes R, S, T and U are created between P and Q .

4.1 New Node Placement

When a paving loop-front is advanced, the new or optimal node placement depends on the type of the loop-front. For an flat paving loop-front, the position of the only new node Q can be expressed as

$$\mathbf{r}_Q = \mathbf{r}_B + \mathbf{V}_t \cdot \mathbf{g} \quad (1)$$

where \mathbf{r}_B is the position vector of node B , \mathbf{V}_t is the bisector of angle ABC and grading g is assumed to be the harmonic mean of the sizes at the nodes defining the loop-front.

$$\mathbf{g} = 3\mathbf{g}_A\mathbf{g}_B\mathbf{g}_C / \sin(\theta/2)(\mathbf{g}_A\mathbf{g}_B + \mathbf{g}_B\mathbf{g}_C + \mathbf{g}_C\mathbf{g}_A) \quad (2)$$

$\mathbf{g}_A, \mathbf{g}_B, \mathbf{g}_C$ are the grading values at nodes A, B & C .

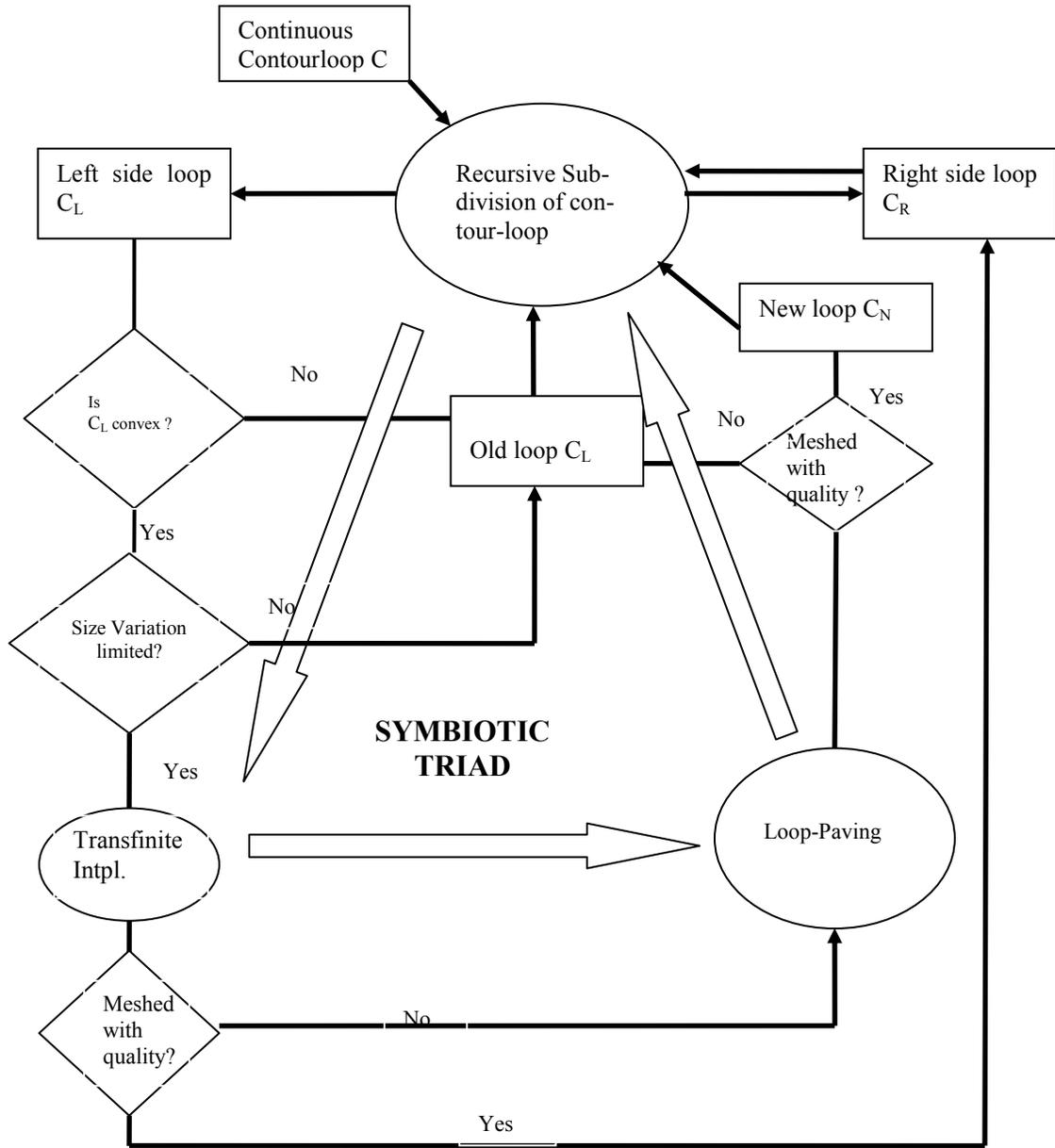


Fig. 1. Flowchart of Symbiotic Triad Algorithm for meshing continuous contourloop C

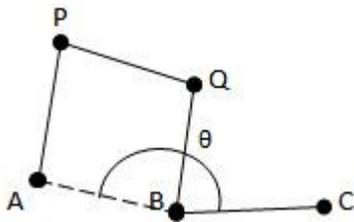


Fig. 2. A typical paving loop-front

Table I shows the formulas for positioning the new nodes during loop-paving. For a Right-Reversal paving loop-front, 3 new nodes (Q,R&S) are created. The angle bisector unit vector is transformed by $\pm \theta/6$ to obtain the translational unit vectors for nodes Q and R. For all nodes that are diagonally opposite to node B, the grading is multiplied by a factor of $\sqrt{2}$ to account for orthogonality. In a similar manner for a Reversal paving loop-front 5 nodes (Q,R,S,T,U) are created. The angle bisector unit vector is transformed by $\pm \theta/4$ and $\pm \theta/8$ to obtain the translational unit vectors for nodes Q,R and U,S respectively. Each new node is tested for proximity with its immediate neighbors in an attempt to avoid creating nearly-squished or hourglass-shaped quads. When such situations arise decision is taken either to merge proximate nodes or completely reject the loop for paving.

4.2 Loop closure

Loop closing is a delicate affair. Exceptions need to be made to the forward creation theorems (as illustrated and explained in Table I) for the various paving loop-fronts in this case. Table II lists the stencils used for each-pair of loop-front types. The terminal loop front is represented by XAB while the first loop-front is ABC. All nodes marked P",Q", R" etc. refer to the original nodes created during the advancement of the first front. Nodes P,Q,R,T etc. denote the last loop-fronts new advancing node positions created at the time of loop closure. To close the loop, 1,2 or 3 terminal elements need to be created based on the pairing type.

After each paving-loop is advanced or paved, the elements and nodes generated in the process are smoothed using an isoparametric smoother as described by Blacker and Stephenson [4] with a minor difference in that it is applied to a closed loop of elements and node rings and thus the Dirichlet and Neumann boundary conditions are placed accordingly.

4.3 Loop Front Evaluators

Loop closing is a delicate affair. Exceptions need to be made to the forward creation theorems (as illustrated and explained in the Appendix (Table I) for the various loop-fronts in this case. Table II lists the various stencils used for each-pair of loop-front types. The terminal loop front is represented by ABC while the first loop-front is BCE. Pt D represents the first new advancing point of the loop. Pt Q denotes the last new advancing point which, at the time of loop closure, has already been created. To close the loop, 1 or 2 terminal elements need to be created based on the pairing type.

Before and after a loop is paved a number of checks are performed, each one for a different purpose. These checks are explained below

- 4.3.1 **Loop Area Shrinkage** - With every successful loop-paving, the loop area shrinks. This check is a posteriori check that is performed on the paved loop to ensure the ratio of the loop areas of the new and original loops (A_n, A_o) do not shrink below an area tolerance (ϵ_A) -

$$A_n/A_o > \epsilon_A \quad (3)$$

4.3.2 **Loop Perimeter Growth/Shrinkage** - The loop perimeter can grow and shrink as the loop advances. This check is also posteriori and compares the ratio of the perimeters of the new and original loops (P_n , P_o) against a perimeter tolerance (ϵ_p) as described by eqn. (4).

$$|P_n/P_o| > \epsilon_p \quad (4)$$

4.3.3 **Loop Length Variation:** Since paving is more heuristic than recursive subdivision, the mesh pattern and number of elements tend to vary overtly for small variations of the global size [24]. Also, in traditional paving *wedging* and *tucking* are performed [4] to prevent fronts from shrinking and expanding beyond size limits. With wedges and tucks, paved meshes often produce diamond shaped quads in the middle of isotonic flowlines. Certain types of analyses, especially crash analysis (of vehicles), are sensitive to that topology [25]. Therefore, a check is performed apriori where the rate of variation (dL/ds) of loop lengths (L) along its boundary (s) is kept below a certain limit (ϵ_s).

$$|dL/ds| < \epsilon_s \quad (5)$$

4.3.4 **Loop-Front Relative Growth Rate** - This posteriori check ensures that the ratio of the extremums ($|dL/ds|_{max}$, $|dL/ds|_{min}$) of the loop-front differential growth rates are within a specified limit (ϵ_G). Any irregular and abrupt loop-front size changes are avoided during paving so as to ensure that the paved mesh layers are good quality and regular.

$$|dL/ds|_{max}/|dL/ds|_{min} < \epsilon_G \quad (6)$$

5. Recursive Subdivision

The recursive subdivision algorithm consists of the following steps-

1. join all loops into a single continuous loop
2. recursively split the continuous loop by a best-split-line
3. determine the best split line
4. estimate the number of nodes to be generated on the split line
5. space the nodes on the split line
6. if no more nodes can be generated construct elements if loop has 3 or 4 nodes only
7. goto step 3 and continue until mesh is done

5.1 Connecting loops

In order to connect all loops to a single continuous loop, a cartesian grid of the global element size is constructed in the background. Given a mesh area with n loops, these cells are used to identify a pair of nodes representing the shortest distance between outer loop l_o and any inner loop l_i along a line whose optimum angular deviation ϕ (discussed in sect. 5.2) is minimum. The connecting line is checked for intersection with any other loops. Once this connection is made the problem now is reduced to one connecting $n-1$ loops. The process is repeated until a single continuous loop results.

5.2 Recursion algorithm

The recursive subdivision algorithm takes a single 2D contourloop defined by a sequence of nodes and recursively splits it to fill the region. The input contourloop must not be self-intersecting nor have coincident nodes but can be self-touching. Nodes can also have repeated entry in the loop. The subdivision logic is described by the following flow-logic

ALGORITHM III : The recursive subdivision logic

While the compound contour is not completely filled (refer Fig. 3b)

- ```

{
 1a. Get the best splitting line A that makes an
 angle close to 90 degree with the contour.
 The splitting line divides the contour into
 a sub-contour on the left (CL) and one on
 the right(CR).
 1b. while the left sub-contour CL is unfilled
 {
 Repeat step 1a
 }
 1c. while the right sub-contour CR stays unfilled
 {
 Repeat step 1a
 }
}

```

### 5.3 Selecting the best split line

The split line functional  $\Phi$  for a split line joining boundary nodes  $j$  and  $k$ , is expressed as a linear combination of normalized parameters,  $L$ ,  $\phi$  and  $\varepsilon$  ( where  $A_1$ ,  $A_2$ ,  $A_3$  are constants ).

$$\Phi = A_1 L + A_2 \phi + A_3 \varepsilon \quad (7)$$

The normalized length parameter  $L$  is given by  $L = l_{jk}/l_d$ ;  $l_{jk}$  is the length of the split line  $jk$  (as shown in Fig. 3b),  $l_d$  is the characteristic length, which is the diagonal of the rectangular box bounding the mesh area  $B \equiv [(x_{\min}, y_{\min}), (x_{\max}, y_{\max})]$  and given by

$$l_d^2 = (x_{\max} - x_{\min})^2 + (y_{\max} - y_{\min})^2 \quad (8)$$

The normalized split angle  $\phi$  is expressed as the normalized sum of the deviations of the 4 split angles (shown in Fig. 3b) from the ideal quadrilateral angle  $\pi/2$ .

$$\phi = \frac{\sum_{i=1}^4 |\phi_i - \pi/2|}{2\pi} \quad (9)$$

The percentage of length error  $\varepsilon$  resulting from fitting  $n$  nodes on the split line based on the grading values of sample points is discussed elsewhere [14] in details.

The minimum value of the split line functional gives the best split line. However, many of the split line candidates in a concave loop are invalid as they fall outside the domain (as shown in Fig. 3a). A boundary visibility criterion is set up to eliminate these invalid candidates.



In the natural or parametric coordinates of the split line these  $n$  nodes will be equally spaced. The location of the  $i$ -th node on the split line can be expressed in terms of its grading value  $g_i$ , coordinates of the previous point  $p$  on the same line, and the coordinates of the two end nodes  $j$  and  $k$ . The following pair of equations need to be solved to evaluate the location of node  $i$ . The split line functional  $\Phi$  for a split line joining boundary nodes  $j$  and  $k$ , is expressed as a linear combination of normalized parameters,  $L$ ,  $\phi$  and  $\varepsilon$ .

$$\mathbf{x}_i^2(1+\mathbf{m}^2)+2\mathbf{x}_i[\mathbf{x}_p + \mathbf{m}(\mathbf{c}-\mathbf{y}_p)]+\mathbf{x}_p^2+(\mathbf{c}-\mathbf{y}_p)^2+\mathbf{g}_i^2 = 0 \quad (14)$$

and the equation of the split line

$$\mathbf{y}_i = \mathbf{m}\mathbf{x}_i + \mathbf{c} \quad (15)$$

where the slope of the split line is

$$\mathbf{m} = (\mathbf{y}_k-\mathbf{y}_j)/(\mathbf{x}_k-\mathbf{x}_j) \quad \text{and} \quad (16)$$

$$\mathbf{c} = (\mathbf{x}_k\mathbf{y}_j-\mathbf{x}_j\mathbf{y}_k)/(\mathbf{x}_k-\mathbf{x}_j) \quad (17)$$

## 6. Transfinite Interpolation

Transfinite interpolation (extensively discussed by Armstrong and Tam [26], Mitchell [27,28]) and more recently [25]) is optionally employed in convex sub-contour loops only if the size-map variation within its domain is small. The advantage of TFIs is two fold - a) firstly it is fast and efficient and b) it is insensitive to small local variations in the size-map and is guaranteed to generate a structured quad mesh which neither loop-paving nor recursive subdivision can promise.

## 7. Local and Global Anisotropy

The detailed operation of the symbiotic triad explained in Fig. 1 depends on the driving size-map function defined by the local and global anisotropy requirement. Local anisotropy applies to all face-interior boundaries or constraints and is defined by two parameters - a) Number of layers desired ( $\mathbf{n}_l$ ); and b) A layer thickness function  $\mathbf{L}(\mathbf{n})$ . Figures 4(a)-(b) depict the meshes around a hole for three different sizing functions (number of layer desired  $\mathbf{n}_l = 5$ ). Many applications, especially structural mechanics, require boundary-structured graded local meshes and even if the aspect ratio is higher than usual, two rings of well-shaped quads ( $\mathbf{J}_r < 1.2$ ) minimizes stress solution and smoothening errors. The Fibonacci function [ $\mathbf{L}_i = 1,1,2,3,5; i =0-4;$ ] thus assumes importance. The ramp function, however, is more popular for its simplicity.

Global anisotropy applies to the entire surface except for interior boundaries with local anisotropic definitions and is defined by a sizing function. The size or grading of the mesh at any interior node is evaluated from the size field represented by the background mesh as

$$\mathbf{g} = \Gamma(\mathbf{x},\mathbf{y}) \quad (18)$$

To evaluate the grading at any interior point  $\mathbf{i}(\mathbf{x},\mathbf{y})$ , it's owner triangle  $\mathbf{j}$  in the background mesh is identified by a space hashing mechanism. The grading at point  $\mathbf{i}$  is thus expressed in terms of the field values at the vertices of triangle  $\mathbf{j}$  as

$$\mathbf{g}_i(\mathbf{x},\mathbf{y}) = \sum_{k=1}^3 \mathbf{N}_k \mathbf{g}_{jk}(\mathbf{x},\mathbf{y}) \quad (19)$$

$\mathbf{g}_{jk}(\mathbf{x},\mathbf{y})$  denote the grading at the vertices of triangle  $\mathbf{j}$   
 $\mathbf{N}_k$  represent the shape functions of triangle  $\mathbf{j}$

As the mesher fills out the surface, for the unmeshed area  $U$  with  $N_s$  sampling points, an *Extremum Size Gradient* ( $g_{rU}$ ) is constantly computed as the ratio of the maximum to the minimum grading averaged over the unmeshed area. It can be expressed as -

$$g_{rU} = \left( \int_U (g_{\max}(x,y)/g_{\min}(x,y)) du \right) / N_s \quad (20)$$

For all  $ESG > 1.2$ , transfinite interpolation is never attempted on sub-contour loops as it might render the mesh insensitive to the varying size field.

**Fig.4 Local and global anisotropy near a hole for various local sizing functions**

**(a). Local anisotropy as ramp function**

**(b). Local anisotropy as a parabolic function**



## 8. Mesh Quality

Mesh quality is measured in both local and global domains. It is defined as the harmonic mean of the ratio of the extremum Jacobian's ( $J_r$ ) and can be expressed, for a mesh of  $N$  elements as

$$\tau = N / \left( \sum_{i=1}^N 1.0 / (J_{ri}) \right) \quad (21)$$

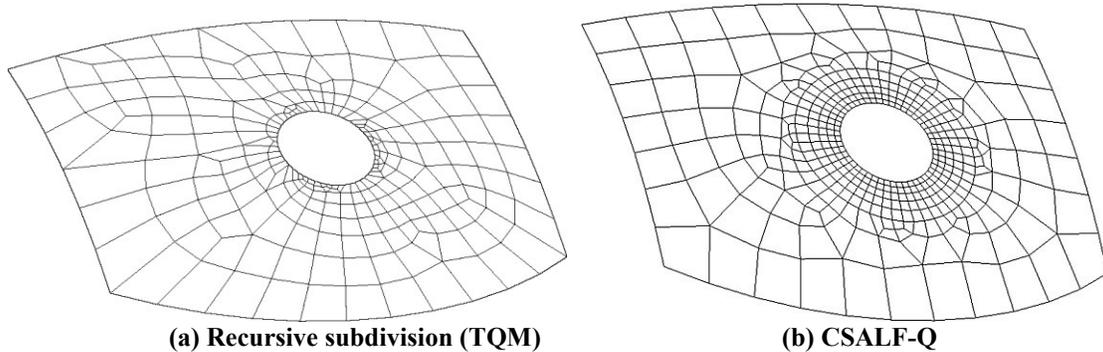
where  $J_{ri} = J_{i\max} / J_{i\min}$  is the Jacobian ratio of the  $i$ -th element

It is measured both in the local paved layers ( $\tau_l$ ) and the entire mesh ( $\tau_g$ ). Both are used to compare meshes generated by the proposed algorithm with that generated by recursive subdivision [14].

## 9. Results & Discussion

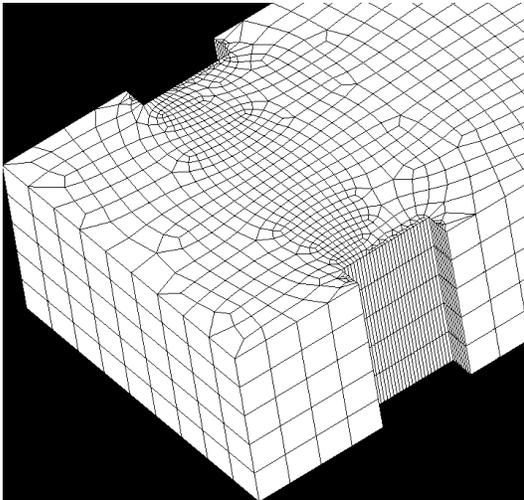
Fig. 5 shows a typical example of a curved face with a central circular cut-out. The analysis demands a tenth of the global element size at the inner loop than the outer. Fig. 5a shows the TQM (recursive subdivision mesh) while Fig. 5b shows the mesh generated by CSALF-Q. While the TQM mesh adapts well to the acutely varying size map between the inner loop and the outer, element quality suffers immensely, especially near the inner boundary - the zone of analytical interest. Neither is the mesh along the outer boundary structured. The local mesh quality  $\tau_l$  over a region defined by a circle with radius  $1.5r$  ( $r$  = radius of the inner loop) is 0.7 while the global mesh quality  $\tau_g$  is 0.78. The CSALF-Q mesh grows the paved ring around the inner loop gradually trying to compromise between the local size map gradient and the mesh quality desired. In comparison to TQM it reports admirably high local and global mesh qualities ( $\tau_l = 0.98$ ,  $\tau_g = 0.83$ ) and one layer of boundary structured elements along the outer loop. The insensitivity of

loop-paving to the local sizemap allows for a good number of boundary structured inner layers followed by a smoother transition leading to an overall enhancement of mesh quality while honoring the sizemaps with reasonable accuracy. For the TQM mesh, the worst quad angle ( $163^\circ$ ) is on the inner boundary while in the CSALF-Q mesh the worst quad ( $152^\circ$ ) is in between the boundaries in the zone of low importance.

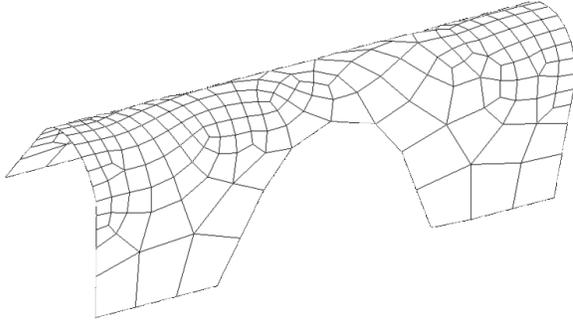


**Fig. 5 Mesh anisotropy on a curved face with a central hole.**

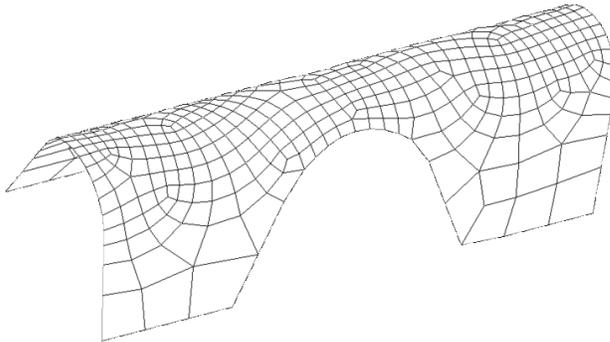
Fig. 6 shows a swept hex mesh on a body with 4 rectangular slots where the wall faces are frozen with a very fine transfinite mesh build to expedite a local contact analysis. As a result, the source face of the body, although absolutely planar, gets a widely varying sizemap. Fig. 7 & 8 depict meshes on a highly curved face for TQM and CSALF-Q respectively. While the TQM and CSALF-Q meshes are close, one can easily notice a more boundary structured mesh and a smoother transition quad mesh [  $\tau_g = 0.79$  (CSALF-Q),  $\tau_g = 0.68$  (TQM)] resulting in relatively better quality mesh for CSALF-Q.



**Fig. 6 Mesh anisotropy on the source face of a swept volume resulting from certain wall faces frozen with a finer transfinite mesh**



**Fig. 7** Anisotropic quad mesh generated by TQM [recursive subdivision, [10,14] ] alone.



**Fig. 8** Anisotropic quad mesh generated by CSALF-Q.

## 10. Conclusion

Paving is known to produce boundary structured quad meshes of admirable quality but fares poorly when challenged by a variegating global sizemap. Recursive subdivision, on the contrary, handles such size variations with elegance and robustness but produces relatively poor quality meshes that are rarely boundary-structured. In this paper, by adopting a *bricolage* technique the contrasting strengths of paving and recursive subdivision are fused and used in conjunction with subarea transfinite meshing to produce CSALF-Q - a new automatic 2D quad meshing algorithm that maintains element quality but is sensitive to both local and global sizemaps. To enable the bricolage algorithm to produce anisotropic yet high-quality boundary structured meshes, the loops are initially meshed recursively with a new “loop-paving” technique. The remaining interior domain is filled out by a symbiotic triad combining recursive subdivision, transfinite interpolation and loop-paving in a balanced and efficient manner. Loop-fronts are classified and rule sets are defined for each, to aid optimum point placement. Stencils used for loop-closure are presented. Results presented compare both local and global element quality of meshes generated by the new algorithm with that of TQM. They clearly indicate that apart from offering user control on the number and thickness of paved layers, CSALF-Q tends to generate meshes that are more boundary structured and smoothens out the quad mesh transition in a manner that strikes a good compromise between quad mesh quality and anisotropy.

## 11. Acknowledgement

The author would like to dedicate the present paper to Thijs Sluiter and Ted Blacker, the founding fathers of TriQuaMesher and Paving respectively. He also extends sincere thanks to Michael Hancock, Jean Cabello and Kirk Beatty of the Meshing and Abstraction Team at Siemens PLM software for their continued support of this research.

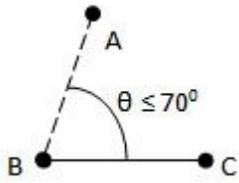
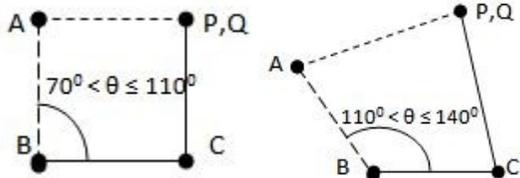
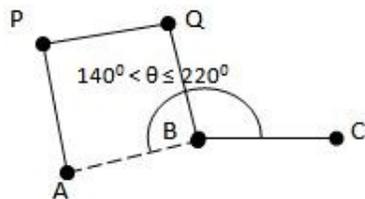
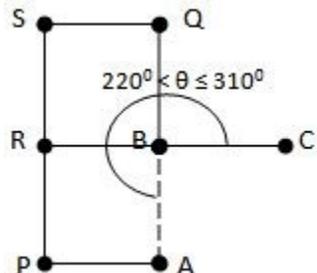
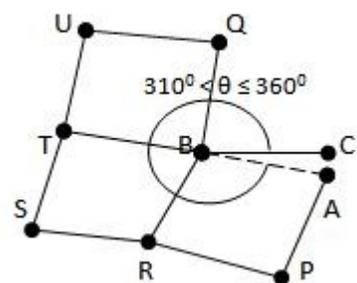
## 12. Reference

1. S. Owen, "A Survey of Unstructured Mesh Generation Technology", *Proceedings, 7th International Meshing Roundtable*, Sandia National Lab, pp.239-267, (1998).
2. P.L. George and P.J. Frey, *Mesh Generation*. Hermes Science. UK. (2000)
3. D. L. Marcum, URL: <http://www.erc.msstate.edu/thrusts/grid/>
4. T. Blacker and M. Stephenson, "Paving: A new approach to automated quadrilateral mesh generation", *Int. Journal Num. Meth. Engg.*, Vol. 32, pp.811-847, (1991).
5. M.L.Staten, R. A. Kerr, S.J. Owen and T. D. Blacker, "Unconstrained Paving and Plastering: Progress Update", *Proceedings, 15th International Meshing Roundtable*, Springer-Verlag, pp.469-486, (2006).
6. D.R. White, and P. Kinney, "Redesign of the Paving Algorithm: Robustness Enhancements through Element by Element Meshing", *Proceedings, 6th International Meshing Roundtable*, Sandia National Laboratories, pp.323-335, (1997).
7. S. Pirzadeh, "Unstructured viscous grid generation by advancing-layers method", AIAA 93-3453-CP, AIAA, pp. 420-434, (1993).
8. J. Peraire, J. Peiro, L. Formaggia, K.Morgan and O.C. Zienkiewicz, "Finite Element Euler Computations in Three Dimensions", *Int. Journal Num. Meth. Engg.*, Vol. 26, pp.2135-2159, (1988).
9. J. Moreno, M.J. Algar, I.G. Diego and F. Catedra, "A new mesh generator optimized for electromagnetic analysis" *Proceedings, 5<sup>th</sup> European Conference on Antennas and Propagation (EUCAP)*, pp. 1734-1738, Rome, (2011).
10. M. L. Sluiter, D. L. Hansen, "A general purpose automatic mesh generator for shell and solid finite elements", *Computers in Engineering*, ASME pp29-34, (1982).
11. Schoofs, L.H.Th.M. Van Buekering and M.L.C. Sluiter, *TRIQUAMESH User's Guide, Report WE 78-01*, Eindhoven University of Technology, Netherlands, (1978).
12. J.A. Talbert, and A.R. Parkinson, "Development of an automatic, two-dimensional finite element mesh generator using quadrilateral elements and bezier curve boundary definitions." *Int. J. Num.Meth.Engg.* Vol. 29, pp1551-1567, (1991).
13. J. Sarrate and A. Huerta, "Efficient unstructured quadrilateral mesh generation self-organizing mesh generation program", *Int. Journal Num. Meth. Engg.*, Vol. 49, pp.1327-1350, (2000).
14. J. Cabello, "Towards Quality Surface Meshing", *Proceedings, 12th International Meshing Roundtable*, pp 201-213, (2003).
15. A.A. Mezentsev, A. Munjiza and J.P. Latham, "Unstructured computational meshes for subdivision geometry of scanned geological objects", *Proceedings. 14<sup>th</sup> International Meshing Roundtable, Sandia National Lab*, pp.73-89. Springer, (2005).
16. D. Berzin, N. Kojekine and I. Hagiwara, "Finite element mesh generation using subdivision technique" *Nihon Kikai Gakkai Sekkei Kogaku*, Vol. 11, pp. 207-210, Japan (2001).
17. B. Joe, "Quadrilateral mesh generation in polygonal regions", *Computer Aided Design*, Vol. 27, pp. 209-222, (1995).

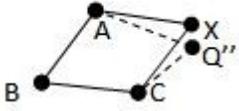
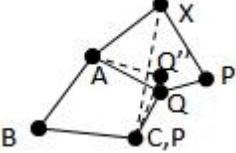
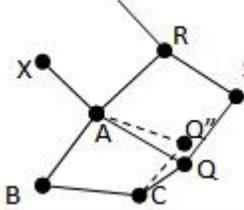
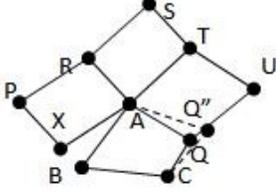
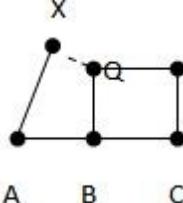
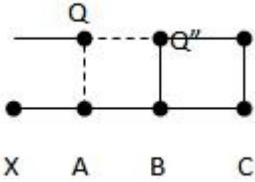
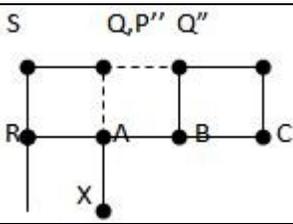
18. D. Nowotny, "Quadrilateral mesh generation via geometrically optimized domain decomposition" *Proceedings, 6<sup>th</sup> Int. Meshing Roundtable*, pp. 309-320, (1997).
19. Claude Lévi-Strauss, *La Pensée sauvage* (Paris, 1962). English translation as *The Savage Mind* (Chicago, 1966). ISBN 0-226-47484-4.
20. N. Mukherjee, "A Combined Subdivision and Advancing Loop-Front Surface Mesher (Triangular) for Automotive Structures", *Int. J. Vehicle Structures & Systems*, 2(1), pp. 28-37, (2010).
21. K. Beatty and N. Mukherjee, "Flattening 3D Triangulations for Quality Surface Mesh Generation", *Proceedings, 17<sup>th</sup> International Meshing Roundtable*, pp 125-139, (2008).
22. P. Kinney, "CleanUp: Improving Quadrilateral Finite Element Meshes", *Proceedings, 6th International Meshing Roundtable*, pp.437-447, (1997).
23. N. Mukherjee, "A hybrid, variational 3D smoother for orphaned shell meshes", *Proceedings., 11<sup>th</sup> Int. Meshing Roundtable*, pp.379-390, (2002).
24. P. Knupp, "Applications of mesh smoothing: Copy, morph, and sweep on unstructured quadrilateral meshes", *International Journal for Numerical Methods in Engineering*, (1999).
25. Nilanjan Mukherjee, "High Quality Bi-Linear Transfinite Meshing with Interior Point Constraints", *Proceedings, 15th International Meshing Roundtable*, pp. 309-323,(2006).
26. T. K. H. Tam and C. G. Armstrong, "Finite element mesh control by integer programming", *International Journal for Numerical Methods in Engineering*, vol. 36, pp. 2581-2605, (1993).
27. S. Mitchell, "Choosing corners of rectangles for mapped meshing", *Proceedings, 13th annual symposium on Computational Geometry*, pp 87-93, (1993).
28. S. Mitchell, "High Fidelity Interval Assignment", *Proceedings, 6th International Meshing Roundtable*, pp 33-44, (1997).
29. Paving algorithm, UG/Scenario, (1999-2002); NX3, Siemens PLM Software (2004).

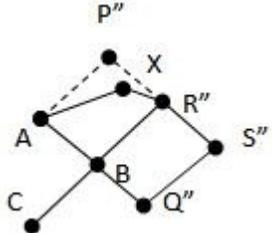
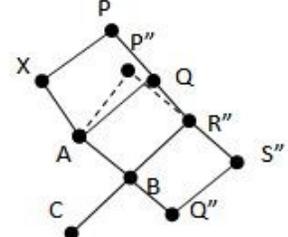
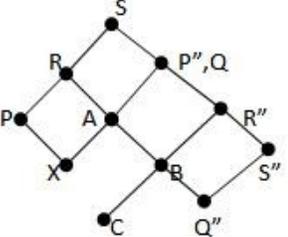
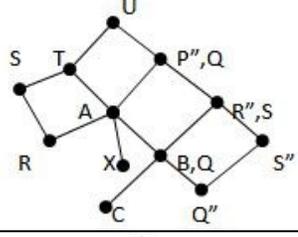
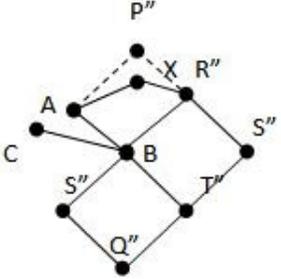
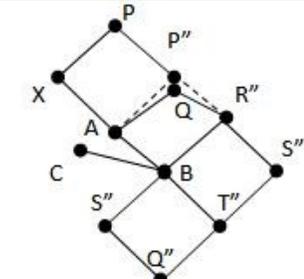
## Appendix

**Table I: Paving Loop Front Models and Front Advancement Rules**

| Paving Loop front Type                                                                                           | No. of new nodes                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | New Elements                                                                   |
|------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| Acute Loop Front<br>            | None                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | None                                                                           |
| Right/Obtuse Loop Front<br>     | Pt P or Q if this is the first front.<br>Else, none. Node P comes from the previous front, Q is same as P. The obtuse loop-front is differentiated from right, because sometimes based on quality criteria, the obtuse front may be treated as a flat-front.                                                                                                                                                                                                                                                                                                                            | □PABC<br>or<br>□QABC<br><br>If element is created, next front must be skipped. |
| Flat Loop Front<br>            | 1 Node Q<br>Position vector of pt Q<br>$r_Q = r_B + Tg_Q$<br>T = unit bisector vector of angle $\theta$<br>Grading at Q<br>$g_Q = 3g_{AgB}g_{C} / \sin(\theta/2) (g_{AgB} + g_{B}g_{C} + g_{C}g_A)$                                                                                                                                                                                                                                                                                                                                                                                     | □QPAB                                                                          |
| Right Reversal Loop Front<br> | 3-Nodes, Q, R & S<br>$r_Q = r_B + T_Qg_Q$ $r_R = r_B + T_Rg_R$<br>$r_S = r_B + T_Sg_S$<br>T = unit bisector vector of angle $\theta$<br>T <sub>Q</sub> = T transformed by (- $\theta$ /6)<br>T <sub>R</sub> = T transformed by ( $\theta$ /6)<br>$g_Q = g_R = 3g_{AgB}g_{C} / \sin(\theta/3) (g_{AgB} + g_{B}g_{C} + g_{C}g_A)$<br>$g_S = 3\sqrt{2}g_{AgB}g_{C} / \sin(\theta/3) (g_{AgB} + g_{B}g_{C} + g_{C}g_A)$                                                                                                                                                                     | □BRPA<br>□QSRB                                                                 |
| Reversal Loop Front<br>       | 5- Nodes, Q, R, S, T & U<br>$r_Q = r_B + T_Qg_Q$ $r_R = r_B + T_Rg_R$<br>$r_S = r_B + T_Sg_S$ $r_T = r_B + T_Tg_T$<br>$r_U = r_B + T_Ug_U$<br>T = bisector vector of angle $\theta$<br>T <sub>Q</sub> = T transformed by (- $\theta$ /4)<br>T <sub>U</sub> = T transformed by (- $\theta$ /8)<br>T <sub>S</sub> = T transformed by ( $\theta$ /8)<br>T <sub>R</sub> = T transformed by ( $\theta$ /4)<br>$g_Q = g_T = g_R = 3g_{AgB}g_{C} / \sin(\theta/4) (g_{AgB} + g_{B}g_{C} + g_{C}g_A)$<br>$g_S = g_U = 3\sqrt{2}g_{AgB}g_{C} / \sin(\theta/4) (g_{AgB} + g_{B}g_{C} + g_{C}g_A)$ | □ABRP<br>□BTSR<br>□BQUT                                                        |

**Table II: Paving Loop Front Closure Stencils**

| First Paving Front (ABC)  | Last Paving Front (XAB)   | Elements to create                                                                                                   | Description                                                                           |
|---------------------------|---------------------------|----------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| <b>Acute/Obtuse/Right</b> | <b>Acute</b>              | None                                                                                                                 | Not applicable                                                                        |
|                           | <b>Obtuse/Right</b>       | □ XYAC<br><br>Node Q'' is replaced by paving front node X                                                            |    |
|                           | <b>Flat</b>               | □ PXAQ □ Q''ABC changes to QABC<br>Node Q'' of the first loop-front is merged with Q of the last front.              |    |
|                           | <b>Right Reversal</b>     | □ S PXAR, SRAQ<br>□ Q''ABC changes to QABC<br><br>Pt Q'' of the first loop-front is merged with Q of the last front. |    |
|                           | <b>Reversal</b>           | □ S PBAR, TSRA, UTAQ<br>□ Q''ABC changes to QABC<br><br>Node Q'' ≡ Node Q (merged)                                   |   |
| <b>Flat</b>               | <b>Acute/Obtuse/Right</b> | □ XABQ                                                                                                               |  |
|                           | <b>Flat</b>               | □s Q''QAB                                                                                                            |  |
|                           | <b>Right Reversal</b>     | □s PXAR, AQSR, Q''QAB<br><br>Node Q ≡ Node P'' (merged)                                                              |  |
|                           | <b>Reversal</b>           |                                                                                                                      | Similar to previous                                                                   |

|                       |                           |                                                                                                                                                                                          |                                                                                       |
|-----------------------|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| <b>Right Reversal</b> | <b>Acute/Obtuse/Right</b> | <p><math>\square R''P''AB</math> is changed to <math>R''XAB</math></p> <p>Node <math>P''</math> is replaced by Node <math>X</math></p>                                                   |    |
|                       | <b>Flat</b>               | <p><math>\square QPXA</math></p> <p><math>\square R''P''AB</math> is changed to <math>R''QAB</math> as</p> <p>Node <math>Q \equiv</math> Node <math>P''</math> (merged)</p>              |    |
|                       | <b>Right Reversal</b>     | <p><math>\square sRPXA, SRAQ</math></p> <p><math>\square R''P''AB</math> is changed to <math>R''QAB</math> as</p> <p>Node <math>P'' \equiv</math> Node <math>Q</math> (merged)</p>       |    |
|                       | <b>Reversal</b>           | <p><math>\square sRPXA, TSRA, UTAQ</math></p> <p><math>\square R''P''AB</math> is changed to <math>R''QAB</math> as</p> <p>Node <math>P'' \equiv</math> Node <math>Q</math> (merged)</p> |   |
| <b>Reversal</b>       | <b>Acute/Obtuse/Right</b> | <p><math>\square R''P''AB</math> is changed to <math>R''XAB</math> as</p> <p>Node <math>P''</math> is replaced by Node <math>X</math></p>                                                |  |
|                       | <b>Flat</b>               | <p><math>\square sQPXA, R''QAB</math></p> <p><math>\square R''P''AB</math> is changed to <math>R''QAB</math> as</p> <p>Node <math>P'' \equiv</math> Node <math>Q</math> (merged)</p>     |  |
|                       | <b>Right Reversal</b>     | None                                                                                                                                                                                     | Not Applicable                                                                        |
|                       | <b>Reversal</b>           | None                                                                                                                                                                                     | Not Applicable                                                                        |

