Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 00 (2017) 000–000

Procedia Engineering

www.elsevier.com/locate/procedia

26th International Meshing Roundtable

A High Order Quad Mesh Generator For Spectral Methods

Bing Yuan[a,*], David A. Kopriva[a]

[a]*Department of Mathematics, Florida State University, Tallahassee, FL 32306, USA*

Abstract

We describe an automatic quad mesh generator designed specifically for spectral methods. The mesh generator features subdivision of the domain chosen by the order of the polynomials to be used in the spectral solver. Curved boundaries are therefore accurately represented to a given tolerance. The algorithm is based on the cross-field method, and allows the direct generation of large, curved elements quadrilateral favored by spectral element and block structured finite difference methods.

Peer-review under responsibility of the organizing committee of IMR 26.

Keywords: quad mesh generation; cross field; spectral method

1. Introduction

A major hurtle in the deployment of spectral element methods [1–4] comes from the generation of acceptable meshes. Spectral element methods can accurately approximate curved boundaries by high order polynomial approximations and their convergence properties favor higher order approximations with fewer elements. At high order, too, the methods are most efficient using a tensor product approximation, so quad meshes are favored, if available. Unfortunately, mesh generators are more available for low order triangular finite element methods, which favor large numbers of elements with straight sides. To date, efforts in the spectral methods community have concentrated on modifying finite element meshes to suit spectral methods by, for example, curving straight sides along physical boundaries and then smoothing to avoid overlapping edges (See, e.g. [5]). However, direct generation of curved element all-quad meshes would be most desirable.

Here we introduce an all-quad mesh generation algorithm designed for spectral element methods. It uses a cross field method to generate an all quad subdivision of a two dimensional domain. It then generates a spectral element mesh by creating quadrilateral elements that match the physical boundaries to a given tolerance for specified polynomial approximation order. We also introduce a way to automatically correct invalid meshes produced by the cross field.

The cross field, also called frame field or 4-way rotational symmetry field, is used in computer graphics [6] and has recently been used to generate quad meshes. The cross field approach has several advantages: It generates a minimum number of irregular nodes, preserves symmetries of the domain, and has boundary alignment. Bunin [7,8] gives a

* Corresponding author.
E-mail address: byuan@math.fsu.edu

Peer-review under responsibility of the organizing committee of IMR 26.

continuous description of the cross field and generates quad meshes by solving an inverse Poisson problem. Kowalski et al. [9] generate a cross field by solving a heat boundary value problem with a nonlinear constraint. Fogg et al. [10] introduce penalty terms to solve for a smooth cross field using a fast marching method. Applications of 3D frame fields for hex mesh generation are studied by Li et al. [11] , Ray and Sokolov [12] and Kowalski et al. [13].

Previous work on cross field methods has targeted the generation of fine quad meshes for finite element methods. Here we show how to use them to generate the types of meshes favored by spectral element methods.

2. The Algorithm

The method is inspired by Bunin [7] and Kowalski et al.'s [9] work. We first define a vector field that is aligned with the boundary of the domain. The cross field is then constructed along the boundary by rotating vectors by multiples of $\pi/2$. It is propagated into the interior of the domain by solving the boundary value problem,

$$\begin{cases} \Delta \boldsymbol{u}(\boldsymbol{x}) = 0 \quad \forall \boldsymbol{x} \in \Omega \\ \boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{u}_0(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in \partial\Omega, \end{cases} \tag{1}$$

where Ω is the two dimensional domain to be meshed and $\partial\Omega$ is its boundary. We solve the Laplace problem using a finite element method on a triangle mesh of the domain generated by DistMesh [14].

Once the cross field is generated, we can decide where the singularities, which determine the valence of irregular mesh nodes, are located inside the domain. Singularities, *i.e.* where the cross field is not smooth, are restricted to two types: index $k = -1$, which has three separatrices, and index $k = 1$, which has five separatrices. Here we use the method provided by Kowalski [9] to find interior singularities by checking the orientation of the cross at vertices of each triangle. The singularities on the boundary are decided by the change of curvature. Where the boundary is not smooth, there is a singularity. The number of separatrices of boundary singularities are decided as shown in Table 1.

Table 1. Separatrices of boundary singularities.

angle of boundary corner	number of separatrices
$0 < \theta \leq 3\pi/4$	no separatrix
$3\pi/4 < \theta \leq 5\pi/4$	1
$5\pi/4 < \theta \leq 7\pi/4$	2
$7\pi/4 < \theta < 2\pi$	3

An initial block structured quad mesh is generated by tracing the streamlines of the cross field. To trace a streamline, we use the propagation method suggested by Kowalski in [9]. However, because of errors introduced by numerical integration and the restriction of singularities to $k = \pm 1$, it is possible that the quad mesh is not valid, i.e. the mesh can contain polygons other than quads.

We then subdivide invalid (non-quadrilateral) elements as follows:

1. If the element is a triangle, we connect its centroid with the midpoints of the three edges.
2. If the element is a pentagon, we find the vertex with largest inner angle and connect it with the non-adjacent vertex, which gives a larger smallest angle among newly created angles, then repeat 1.
3. If the element has more than five edges, we stop and ask user to use a finer triangle base mesh. This step could be automated by using an adaptive finite element method for the solution of the Laplace problem.

After subdivision, we propagate newly created streamlines using transfinite interpolation. The streamline will terminate either at a boundary or another singularity. An illustration of this subdivision method can be seen in Fig. 1.

Once the gross quad mesh is successfully generated, we continue to subdivide boundary edges of the mesh so that the element boundaries match a user specified error tolerance for a user specified polynomial approximation order. Here we use a simple recursive method, shown in Alg. 1.

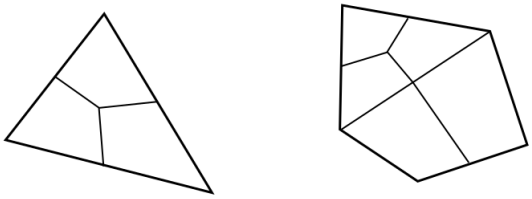

Fig. 1. Subdividing invalid elements. (Left: subdivision of a triangle; Right: subdivision of a pentagon.)

Algorithm 1 Boundary Subdivision

```
for each boundary edge do
    while interpolation error > tolerance do
        divide the boundary edge at middle
    end while
end for
```

The final step is to smooth the quad mesh to avoid small elements or thin blocks. Inspired by Canann et al.'s work [15], we combine the Laplacian smooth with an equal-space algorithm [16] to smooth the quad mesh. After smoothing, it is possible that boundary edges no longer meet the error tolerance. Therefore, we repeat boundary subdivision and smoothing until all boundary edges can be represent accurately.

We summarize the steps of the algorithm as follows:

0. Input: boundary description functions, order of polynomial, error tolerance
1. Generate a triangle base mesh
2. Solve the Laplace equation to generate the cross field
3. Find singularities inside and on the boundaries of the domain
4. Propagate streamlines using the triangle mesh
5. Generate a quad dominant mesh by tracing streamlines
6. Detect and correct invalid elements
7. Subdivide boundary edges by order of polynomial
8. Smooth the quad mesh
9. Repeat 7 and 8 until all boundary edges meet error tolerance

3. Results

The general steps to create a spectral quad mesh are illustrated in Fig. 2. The cross field is generated on the unit circle. Then the initial quad mesh is constructed by tracing cross field streamlines. The boundary is then subdivided for a polynomial of order $N = 5$ to match the exact boundary to an error tolerance of $e = 10^{-5}$ in the L^2-norm. Finally, the mesh is smoothed to get an even distribution of element sizes.

The algorithm automatically detects triangles, as seen in the subdivision shown in Fig. 3, and subdivides it into three quads. New streamlines are propagated using the quad mesh, from which an all quad mesh is generated.

The algorithm can also handle complex geometries like the gingerbread man shown in Fig. 4. The gingerbread man has a height of 60 and a width of approximately 45. The boundary subdivision is based on polynomial of order $N = 5$ with error tolerance of $e = 10^{-2}$ in the L^2-norm.

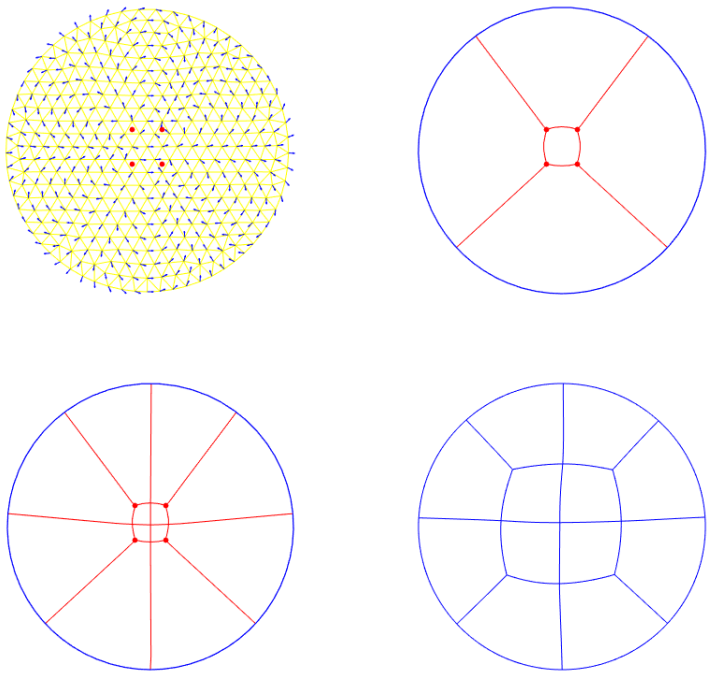

Fig. 2. Steps to generate a quad mesh. Upper left: generate a cross field and find singularities, marked in red; Upper right: create the initial quad mesh by tracing streamlines; Bottom left: subdivide boundary edges; Bottom right: final mesh after smoothing.

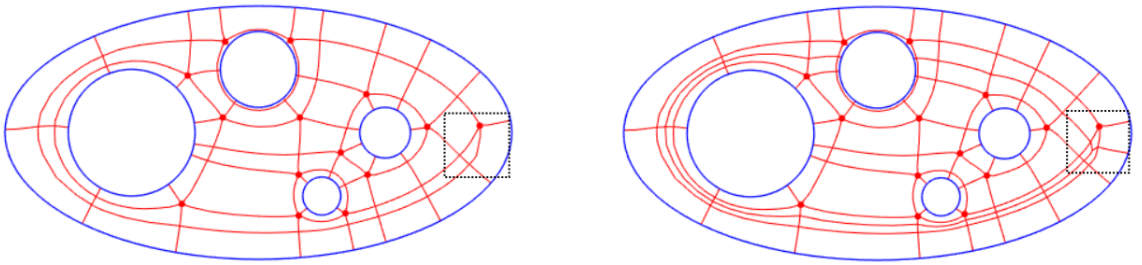

Fig. 3. Subdivision of invalid elements. Left: an invalid triangle in the mesh, marked by dashed box; Right: subdivide the triangle and propagate new streamlines.

4. Summary

The mesh generator described here can adapt a quad mesh according to the order of the polynomial used in high order numerical methods. It can also automatically detect and correct invalid elements. Such a mesh generator can be used to construct meshes for spectral element methods or high order block structured finite difference methods.

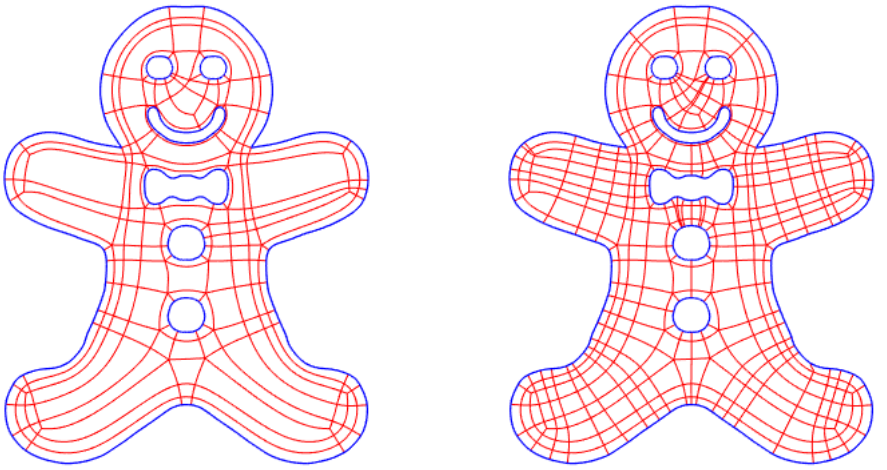

Fig. 4. Gingerbread man. Left: The initial streamlines generated from cross field; Right: Boundary edges subdivided to match the error tolerance.

References

[1] M. Deville, P. Fischer, E. Mund, High Order Methods for Incompressible Fluid Flow, Cambridge University Press, 2002.
[2] G. E. Karniadakis, S. J. Sherwin, Spectral/hp Element Methods for Computational Fluid Dynamics, Oxford University Press, 2005.
[3] J. Hesthaven, T. Warburton, Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications, Springer, 2008.
[4] D. Kopriva, Implementing spectral methods for partial differential equations: Algorithms for scientists and engineers, Springer Science & Business Media, 2009.
[5] F. Hindenlang, Mesh curving techniques for high order parallel simulations on unstructured meshes, Ph.D. thesis, University of Stuttgart, 2004.
[6] J. Palacios, E. Zhang, Rotational symmetry field design on surfaces, ACM Transactions on Graphics (TOG) 26 (2007) 55.
[7] G. Bunin, A continuum theory for unstructured mesh generation in two dimensions, Computer Aided Geometric Design 25 (2008) 14–40.
[8] G. Bunin, Towards unstructured mesh generation using the inverse poisson problem, arXiv preprint arXiv:0802.2399 (2008).
[9] N. Kowalski, F. Ledoux, P. Frey, Automatic domain partitioning for quadrilateral meshing with line constraints, Engineering with Computers 31 (2015) 405–421.
[10] H. J. Fogg, C. G. Armstrong, T. T. Robinson, Automatic generation of multiblock decompositions of surfaces, International Journal for Numerical Methods in Engineering 101 (2015) 965–991.
[11] Y. Li, Y. Liu, W. Xu, W. Wang, B. Guo, All-hex meshing using singularity-restricted field, ACM Transactions on Graphics (TOG) 31 (2012) 177.
[12] N. Ray, D. Sokolov, On smooth 3d frame field design, arXiv preprint arXiv:1507.03351 (2015).
[13] N. Kowalski, F. Ledoux, P. Frey, Block-structured hexahedral meshes for cad models using 3d frame fields, Procedia Engineering 82 (2014) 59–71.
[14] P.-O. Persson, G. Strang, A simple mesh generator in matlab, SIAM review 46 (2004) 329–345.
[15] S. A. Canann, J. R. Tristano, M. L. Staten, et al., An approach to combined laplacian and optimization-based smoothing for triangular, quadrilateral, and quad-dominant meshes., in: IMR, 1998, pp. 479–494.
[16] J. Yao, D. Stillman, An equal-space algorithm for block-mesh improvement, Procedia Engineering 163 (2016) 199–211.