myjournal manuscript No.
(will be inserted by the editor)

Representation Deficit Based Optimal Surface Mesh Generation

David O. McLaurin · Suzanne M. Shontz

Received: date / Accepted: date

Abstract This work explores optimal mesh representation of surfaces using triangular surface meshes. Most mesh generation algorithms refine an initial discretization based on *a priori* error estimates or quality metrics. These mesh generation strategies focus on element quality – with the justification being that downstream applications require high quality geometries in order to achieve a desired level of accuracy. The work presented here is contrasted with these traditional methods in that only the degree in which the discretization approximates the geometry is considered. A formal definition of fundamental mesh operations is developed with respect to optimal representation for a continuous, parameterized surface. A method of generating surface grids through constrained, local optimization is then detailed. Further discussion of robustness and mesh quality is also presented. Results are presented showing the developed algorithm to be effective at generating triangular surface meshes which are optimal with respect to *representation deficit*.

Keywords First keyword · Second keyword · More

1 Introduction

This work seeks to explore the concept of optimal mesh representation. Most mesh generation algorithms refine an initial discretization based on *a priori* error estimates or quality metrics. For example, the advancing-front algorithm advances boundaries into space to generate a grid [**?**,**?**]. Other methods generate grids from iterative refinement or enrichment from initial, coarse configurations [**?**,**?**,**?**]. These mesh generation strategies focus on element quality—with the justification being that downstream applications require high quality geometries in order to achieve a desired

The work of the second author is supported in part by NSF CAREER grant ACI-1330056 (formerly ACI-1054459)

David O. McLaurin
10800 Pecan Park Blvd.
Austin, TX 78750
Tel.: +1 512-331-2808
Fax: +1 512-258-5938
E-mail: david.mclaurin@cd-adapco.com

Suzanne M. Shontz
Mississippi State University
Mississippi State, MS 39762
E-mail: sshontz@math.msstate.edu

level of accuracy. These approaches, historically and in a modern sense, has been spectacularly successful at producing high quality meshes for computation simulation.

Traditionally, surface grid generation processes produce good quality grids from the combination of geometric growth rates and smoothing. However, the process requires input and if the input, or starting place, is not appropriate, then the geometry is often under and/or over sampled for the intended use. That fact is not an indictment of the grid generation process, but instead implies that the final grid is heavily dependent on the inputs. In addition, if some way of controlling the point spacing in the middle of the surface (away from the boundaries) is not present, then more points could be wasted/omitted in an attempt to accurately represent the geometry. Efforts have gone into creating a locally or globally optimal surface grid. For example, curvature is a common driver of surface mesh adaptation or refinement [**?**]. Interpolation error, given a desired function, is also ubiquitous [**?**,**?**,**?**,**?**].

The work presented here is contrasted with the above work in that only the degree in which the discretization approximates the geometry is considered. Here a formal definition of fundamental mesh operations (triangle split, edge split, edge collapse, and vertex movement) is developed with respect to optimal representation for a continuous, parameterized surface. The proposed algorithm is a general-use method that can be applied to any surface regardless of its representation. This is due to every step being developed without the use of derivatives. A result of not using derivatives is that each step in the algorithm is robust to large or discontinuous changes in derivatives.

This process can only be automated if some way of judging "how well" a surface grid represents a surface is present. To this end, a method of generating surface grids through constrained, local optimization is detailed below. Results for mesh representation and efficiency are also given and discussed. Further discussion of robustness and mesh quality is also presented.

Element quality is not considered here as a constraint or goal, as there are numerous methods for surface mesh quality optimization [**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**] which could be used in a postprocessing step if greater mesh quality is required. Instead only optimal representation of the underlying geometry is studied.

2 Nomenclature and Definitions

2.1 Parametric Surface

In the field of surface mesh generation there exists an overwhelming precedent for mapping physical space to parametric space and performing the actual mesh generation in parametric space. This strategy has been proven to be efficient and accurate—with the assumption that the parameterization is of reasonable quality. Here, mesh generation in parametric space also enables the development of unambiguous optimization constrains and objective functions. The search space for the local optimization problem is straightforward and intuitive to define—as well as efficient to search. Therefore, this method will be developed for an orientable, parameterized surface, $\mathbf{S}(u, v) : \mathbb{R}^2 \to \mathbb{R}^3$.

2.2 Discretization

Consider a discretization, D, defined on S, comprised of n_p points, $P \in \{p_1, ..., p_{n_p}\}$, and a non-overlapping, non-degenerate, consistently oriented, conformal, triangular connectivity $T \in \{t_1, ..., t_{n_t}\}$. Each triangle in T, t_i, is defined by points (p_j, p_k, p_m), and edges (e_n, e_o, e_p). For the purposes of defining an edge in this work, the ordering of the nodes does not matter. However,

the ordering of the triangle connectivity (sometimes referred to as winding) is used to maintain consistent orientation during topological operations and to define constraints in the developed optimization strategy.

Each point, p_i, in D is defined at a parametric coordinate, $(u, v)_i$, and an edge is a straight line in both parametric space and in $\mathbb{R}^3$. This is not the case for a discretization, $D : \mathbb{R}^3$, which is mapped onto a surface, $S : \mathbb{R}^3$. In the absence of a parameterization the mapping of an edge in $\mathbb{R}^3$ to a curve (possibly a straight line, but not guaranteed) on a surface is an ambiguous task which is outside the scope of this work. Additionally, the development of topological constraints for optimization, which will be detailed later, would not be as straightforward as is possible when using a parameterized, planar space.

3 Discretization Error, Discrete Curvature Approximation

The accuracy, or discretization error, of a piecewise, linear representation of an analytical surface in $\mathbb{R}^3$ can be defined in many ways depending on the intended application. The error associated with the discretization is discussed in terms of the deviation from the surface—most often quantified by calculating or approximating the distance from the surface for each linear entity (triangles and edges) in the discretization. Another way of quantifying the error associated with a discretization is to consider how well it approximates the surface area of the surface it represents. In general, the surface area is not known *a priori*; however, for some surfaces, depending on the underlying representation, it can be calculated exactly via computation of a double integral or estimated using numerical cubature. In other cases, such as when the surface is very irregular or rough, it is not always possible to assign a surface area without a generalization of the concept from geometric measure theory [**?**]. Two goals of the developed method is that it be general, i.e., it should be independent of the underlying geometric representation, and practical. Therefore, a method requiring the surface area of the underlying geometry violates the aforementioned concept of generality and restricts the applications for which the proposed method can be applied and hence is impractical. Another way of determining/generating a surface grid based on surface area is needed. This process is detailed later.

The concept of deviation defined above is relatively straightforward and intuitive. However, another related way of describing how well a *discretization* represents a *surface* is the degree to which the discrete representation approximates curvature. First, however, curvature must be defined in such a way that a discrete approximation is meaningful and appropriate. In relevant literature, there are many ways to estimate surface curvature [**?**]. Some of it bears repeating, because it is germane to what is being discussed here: Consider a surface, S, embedded in $\mathbb{R}^3$, at point P. The concept of an osculating circle [**?**] does not generalize well to higher dimensions and therefore cannot be used here as an approximation of curvature. Generally, a (hyper)surface does not have a (hyper)sphere that approximates curvature, but rather a (hyper)ellipsoid. This is because a surface can have multiple values and directions of curvature at a point. Here, an osculating ellipsoid could be constructed by considering the principal curvature directions and surface normal at a given point on a surface. These three vectors are orthogonal and could be used to define the semi-axes of an ellipse. If some way could be found to determine a scale of the semi-axis associated with the surface normal (which is a unit-vector), this ellipse would be a good approximation of the surface in the direction of the principal curvatures. However, this requires the computation or approximation of derivatives—which we have ruled out here.

Instead of determining a three-dimensional analog to an osculating circle for S, an osculating sphere could be defined for a triangle, T. A value of curvature can be calculated for each triangle in the D by considering the corresponding osculating sphere for T. The osculating sphere here

can be approximated by considering the circumscribed sphere (circumsphere) [**?**] defined by the three points of the triangle, P_0, P_1, and P_2, and a point, P on the surface located within the triangle in (u, v) space. The radius of the circumsphere will be referred to as the discrete radius of curvature, R_D. This argument for curvature approximation is a two-dimensional analog to the one in [**?**] for approximating curvature along a curve. In that work, the authors showed that for an edge on a curve, as the radius of curvature of the osculating circle approaches infinity the arc length of the circular segment approaches the arc length of the edge. However, here as R_D approaches infinity, the surface area of the spherical cap does not approach the area of T—it approaches that of the circumscribed circle defined by T.

Finally, what is most appropriate is to consider the surface area of the tetrahedron, $\mathbb{T}$, formed by P_0, P_1,P_2, and P. This is a meaningful comparison: as R_D approaches infinity (as the distance between P and the plane defined by T approaches P_0) the surface area of $\mathbb{T}$ approaches two times the area of T. Stated another way, as D is refined the surface area of D approaches that of S.

Surface area convergence of a discretization is a sufficient condition for other schemes of surface grid generation/refinement. That is: if the difference between the surface area of the surface and the sum of the surface area of the discrete entities in the discretization approaches zero then that is sufficient to conclude that the distance between the discretization and surface is also approaching zero; also the dihedral angles between segments approaches 0 degrees. However, the converse of that statement is not true. The pathological case of a highly oscillatory, low amplitude surface approximated by two triangles shows that a discretization can have a small deviation or angles between elements but be a poor estimate for surface area.

4 Representation Deficit based Mesh Operations

4.1 Representation Deficit

The concept of representation deficit is discussed in [**?**] in the context of scale-independent measure of curve discretizations. Here, the concept will be extended to two dimensions. It is important that any criteria for refinement maintain scale independence. This is so that the values for representation deficit can be compared between different candidate operations. Below, representation deficit is defined and discussed for four fundamental mesh operations: triangle split, edge split, edge swap, and node movement.

4.2 Triangle Split

A triangle is a planar object, and is representing a possibly non-planar portion of a surface. Any triangle, with area A_T, in the discretization can have at most the same area as the portion of the surface which it represents, A_S. Therefore, the representation deficit for a triangle, RD_T, is defined as $RD_T = \frac{A_S - A_T}{A_T}$. Note that the areas are calculated in $\mathbb{R}^3$. The difference in surface area, $(A_S - A_T)$, is normalized by A_T so that the result is scale independent. Also, this is a representation *deficit* since $A_S \geq A_T$ is always true.

In order to apply the above definition of representation deficit to a mesh generation, a replacement for A_S must be determined since the area of the surface represented by A_S is not always able to be determined—or, most often, the area calculation is impractical. Generally, in order to reduce the representation deficit for a triangle the triangle is split by inserting an interior point. Any point that is inserted into the interior of the triangle would decrease the representation deficit—or at worst it will remain the same. However, the determination of where to split the triangle should be done in such a way that the representation deficit is minimized.

This strategy of refinement, refining each triangle in such a way that the representation deficit is locally minimized, would take advantage of the optimal substructure of the discrete topology.

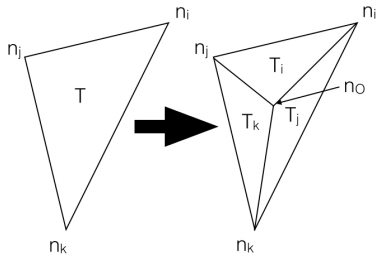


Fig. 1: Triangle Split

The process of determining where to split a triangle is defined here by a locally optimizing an objective function defined for a triangle: Let $S(u, v)$ be a parameterized surface, D be be a discretization of the surface, and T be a triangle (included in D) defined by an ordered tuple of nodes, (n_i, n_j, n_k). These nodes are ordered such that the triangle normal is positive. Specifically, if $\mathbf{V_0} = \{n_j - n_i\}$ and $\mathbf{V_1} = \{n_k - n_i\}$ then the triangle normal, $N_T = \mathbf{V_0} \times \mathbf{V_1}$, is positive. Note that a two-dimensional cross-product (or two-dimensional curl) is a scalar quantity. Additionally, let a node on the interior of T be defined as n_O. The four nodes, n_i, n_j, n_k, and n_O define three triangles, $T_i\left(n_i, n_j, n_0\right), T_j\left(n_j, n_k, n_0\right), T_k\left(n_k, n_i, n_0\right)$. If $A(T)$ is a function that calculates the area of a triangle in (x, y, z) space, then the optimization problem for finding the optimal position for n_0 in T is defined as:

$$\begin{aligned} \underset{n_O}{\text{minimize}}\ O(T) &= -\frac{A(T_i)+A(T_j)+A(T_k)}{A(T)} \\ \text{subject to } N_{T_i} &> 0 \\ N_{T_j} &> 0 \\ N_{T_k} &> 0. \end{aligned}$$

It should be noted that this definition of representation deficit would be difficult to derive or express for a topological entity other than a triangle. This is due to the inherent ambiguity in the definition of not only the surface area, but also the surface representation of (possibly) non-planar elements, e.g. non-planar, or skew, quadrilateral.

4.3 Edge Split

An edge in parametric space represents a (possible) curve on the surface. Any edge, with length L_E, can have at most the same length as the portion of the surface which it represents, L_S. Therefore, the representation deficit for an edge, RD_E, is defined as $RD_E = \frac{L_S - L_E}{L_E}$. Note that the lengths are calculated in $\mathbb{R}^3$. The difference in length, $(L_S - L_E)$, is normalized by L_E so that the result is scale independent. Also, this is a representation *deficit* since $L_S \geq L_E$ is always true;

In order to apply the above definition of representation deficit to mesh generation, a replacement for L_S must be determined since the length along the surface represented by L_S is not

always able to be determined – or, most often, the arc-length calculation is impractical. Generally, in order to reduce the representation deficit for an edge the edge is split by inserting an interior point. Any point that is inserted into the interior of the edge would decrease the representation deficit — or at worst it will remain the same. However, the determination of where to split the edge should be done in such a way that the representation deficit is minimized. This strategy of refinement, refining each edge in such a way that the representation deficit is locally minimized, would take advantage of the optimal substructure of the discrete topology. A method for generating a locally optimal edge split is detailed in [**?**,**?**].

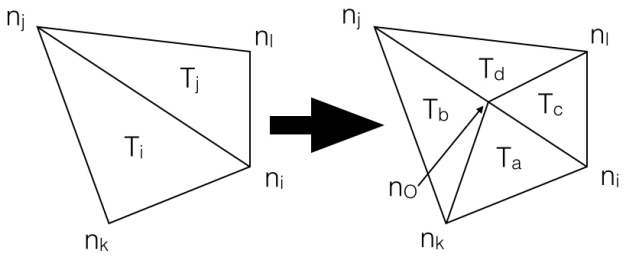


Fig. 2: Edge Split

In addition, the fact that an edge split will change the surface area of the discretization should be considered. Since the overall justification of this work is reduce the *area* representation deficit of a discretization, the representation deficit will not be defined for an edge but for an edge-split. Therefore, for a given edge, only the edge-split that minimizes *area* representation deficit is considered. The process of determining where to split a triangle is defined here by locally optimizing an objective function defined for an edge-split. Let $E\left(n_i, n_j\right)$ be an edge in D that is topologically adjacent to two triangles, $T_i\left(n_i, n_j, n_k\right)$, and $T_j\left(n_i, n_l, n_j\right)$. Additionally, let a node on the interior of the edge be defined as n_O. The five nodes, $\{n_i, n_j, n_k, n_l, n_O\}$ define four triangles, $T_a\left(n_i, n_O, n_k\right)$, $T_b\left(n_O, n_j, n_k\right)$, $T_c\left(n_i, n_l, n_O\right)$, $T_d\left(n_O, n_l, n_j\right)$. The optimization problem for finding the optimal position for n_O on E is defined as:

$$\begin{aligned} \underset{n_O}{\text{minimize}}\ O(E) &= -\frac{A(T_a)+A(T_b)+A(T_c)+A(T_d)}{A(T_i)+A(T_j)} \\ \text{subject to } N_{T_a} &> 0 \\ N_{T_b} &> 0 \\ N_{T_c} &> 0 \\ N_{T_d} &> 0. \end{aligned}$$

4.4 Nodal Movement

In this paper, the focus is on using nodal movement in order to locally minimize the representation deficit in the surface mesh. It should be noted that, although a locally optimal mesh topology could also be obtained, computing such a topology would involve the solution of a discrete optimization problem (via integer programming) which would specify the relevant sequence of edge flips. The discrete and continuous optimization problems could then be solved in an iterative, interleaving fashion. However, such an approach would require significant additional computational expense.

For a node, N_i, the representation deficit is defined only by comparing it to the node when located at another point in space. Here the comparison is bound by limiting the range of comparison within the edge-hull topologically adjacent to N_i. Formally, let N_i be a node in D that is shared topologically by n triangles, where n is the face-valence of N_i. The optimization for finding the optimal position for N_i is defined as:

$$\begin{aligned} \underset{N_O}{\text{minimize}}\ & O(N) = -\sum_{j=1}^{n_t-1} A\left(T_j\right) \\ \text{subject to}\ & N_{T_1} > 0 \\ & N_{T_2} > 0 \\ & N_{T_3} > 0 \\ & \vdots \\ & N_{T_{n-1}} > 0 \\ & N_{T_n} > 0. \end{aligned}$$

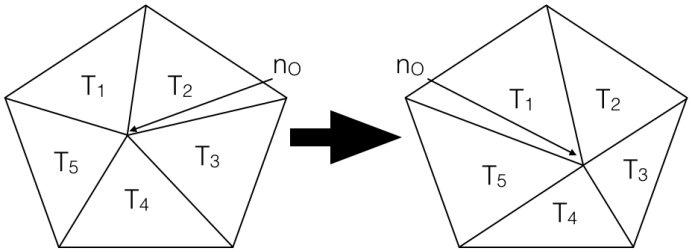


Fig. 3: Nodal Movement

5 Mesh Refinement Algorithm

In order to formulate an optimization problem which, when solved, yields an optimal mesh triangulation based on representation deficit, a local optimization approach [**?**] is used. Such a formulation comes much more naturally than does a formulation based on global optimization [**?**] when the mesh operations defined in Section 4 are employed. One reason for this is the variety of mesh operations that are performed at different stages in the mesh generation process; they naturally lend themselves to the formulation of different (but related) optimization problems as opposed to one unified optimization problem.

It should also be noted that global optimization algorithms are typically not used to solve optimization problems with a large number of variables – even if the problem is formulated as a global optimization problem. Since there are typically millions to billions of nodes and elements in meshes from application problems of interest, a global optimization problem would be impractical solve based on the huge number of variables that it would necessarily contain. In addition, any global optimization problem which would be formulated as a generalization of the local optimization problems in this work would be a continuous optimization problem. Typically global optimization methods are designed for use on discrete optimization problems or on contious optimization problems with convex objective functions with guaranteed global minima.

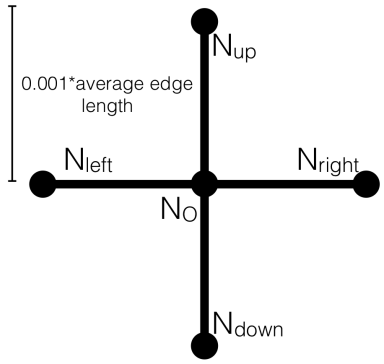


Fig. 4: Cross Pattern used for Pattern Search Optimization

For the above reasons, iterative refinement (a local optimization approach) is used to find optimal substructures at each stage of the mesh generation process. The local optimization technique employed is motivated by a class of optimization methods called pattern search techniques [**?**]. Pattern search and multidirectional search techniques were developed by Park and Shontz for mesh quality improvement [**?**]; their techniques were based on the underlying classes of optimization methods in the literature. Pattern search methods do not use derivatives and hence can be used on problems for the most general problems for which derivatives either do not exist or are impractical to compute.

The optimization employed here uses a simple "cross" pattern (up down left right center, [**?**]) which is sized based on the local geometry so that it could be moved inside of the triangles/hulls many times before encountering the boundary. In the case of triangle splitting, the cross was sized as 0.001 multiplied by the average edge length of the triangle. In the case of node movement, the cross was sized as 0.001 multiplied by the average edge length of the topologically attached edges. The cross pattern was not resized during optimization. Additionally, the search was not allowed to create inverted elements. This was enforced by halting a search whose next step would create any overlapping edges, tangled elements, etc... Care was taken to ensure that convergence of the local optimization method was obtained; this is important since criteria must be met in order for the heuristic pattern search methods to converge [**?**,**?**].

The following algorithm is used to generate an optimal triangulation based on representation deficit refinement:

5.1 Algorithm Specification

Algorithm 1 Find optimal point for triangle splitting

procedure FINDOPTIMALPOINT(t_i) ▷ for Triangle
  $pattern \leftarrow sizePattern\,(t_i)$
  $pattern \leftarrow calculateTrianglePatternValues\,(t_i)$
  $minValue \leftarrow determineMinPatternValue\,(pattern)$
  while $newTriangulationIsValid\,(minValue)$ **do**
    $pattern \leftarrow shiftPattern\,(minValue)$
    $pattern \leftarrow calculateTrianglePatternValues\,(t_i)$
    $minValue \leftarrow determineMinPatternValue\,(pattern)$
  end while
  $p_i \leftarrow currentPatternValue\,(pattern)$
  $return \leftarrow (p_i, minValue)$
end procedure

Algorithm 2 Find optimal point for edge splitting

procedure FINDOPTIMALPOINT(e_i) ▷ for Edge
  $pattern \leftarrow sizePattern\,(e_i)$
  $pattern \leftarrow calculateEdgePatternValues\,(e_i)$
  $minValue \leftarrow determineMinPatternValue\,(pattern)$
  while $newTriangulationIsValid\,(minValue)$ **do**
    $pattern \leftarrow shiftPattern\,(minValue)$
    $pattern \leftarrow calculateEdgePatternValues\,(e_i)$
    $minValue \leftarrow determineMinPatternValue\,(pattern)$
  end while
  $p_i \leftarrow currentPatternValue\,(pattern)$
  $return \leftarrow (p_i, minValue)$
end procedure

Algorithm 3 Iterative Refinement

procedure TRIANGLEREFINE(T, tol_{RD}) ▷ Refine the triangulation, T
  for each triangle, t_i **do**
    $(p_i, minValue) \leftarrow findOptimalPoint\,(t_i)$
    if $tooCloseToEdge\,(p_i, t_i)$ **then**
      $e_i \leftarrow findNearEdge\,(p_i, t_i)$
      $(p_i, minValue) \leftarrow findOptimalPoint\,(e_i)$
      $operation \leftarrow EdgeSplit\,(e_i, p_i)$
    else
      $operation \leftarrow triangleSplit\,(t_i, p_i)$
    end if
    if $minValue < tol_{RD}$ **then**
      perform *operation*
    end if
  end for
end procedure

Algorithm 4 Find the optimal point for node relocation

```
procedure findOptimalPoint(n_i)                                   ▷ for Node
    pattern ← sizePattern (n_i)
    pattern ← calculateNodePatternValues (n_i)
    minValue ← determineMinPatternValue (pattern)
    while newTriangulationIsValid (minValue) do
        pattern ← shiftPattern (minValue)
        pattern ← calculateNodePatternValues (n_i)
        minValue ← determineMinPatternValue (pattern)
    end while
    p_i ← currentPatternValue (pattern)
    return ← (p_i, minValue)
end procedure
```

Algorithm 5 Nodal Movement

```
procedure NodalSmooth(T, tol_RD)                                  ▷ Smooth Node Positions
    for each node, n_i do
        pattern ← sizePattern (e_i)
        pattern ← calculateNodePatternValues (n_i)
        (p_i, minValue) ← findOptimalPoint (e_i)
    end for
    if minValue < tol_RD then
        relocate n_i to p_i
    end if
end procedure
```

5.2 Observations

5.2.1 Nodal Movement

In practice it was observed that nodes will often move toward each other when there is a "local minimum" nearby (*race-condition*). However, once the two nodes get sufficiently near to each other one will inevitably get nearer to the local minimum first and the other node will be "pushed" away.

5.2.2 Edge Flipping

Another phenomenon that was noticed was that during triangle splitting and node movement a node would often be repositioned near edges. This usually meant that the nearby minimum was either on the edge or on the other topological side of the edge. Without some sort of intervention, this would lead to nearly degenerate geometry being formed. In order to prevent geometry which could cause numerical robustness issues, when a node neared an edge during node movement the edge was flipped. This allowed the node to move unobstructed by the newly-reconnected edge.

The criteria for determining if a node is "too close" to an edge is based on the size of the pattern[CLARIFY]. If the minimum value for the objective function is on or across an edge, then the edge is flipped and the process is repeated with the new topology.

6 Results

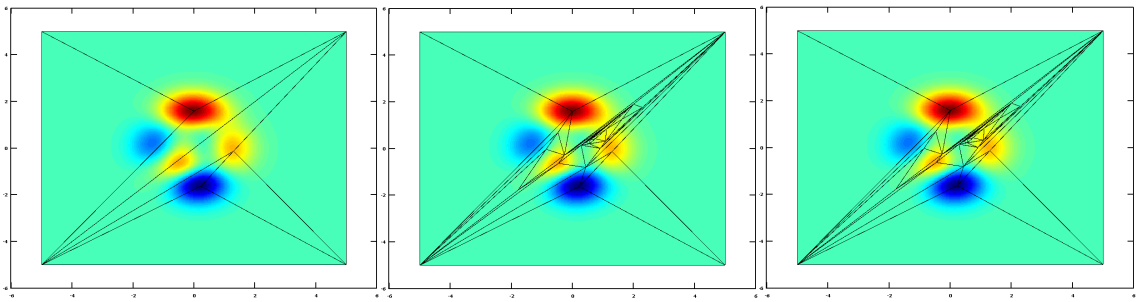
In this explorative work, results will be shown for the how each individual mesh operations (triangle split, edge split, node movement) affects the representation deficit. In addition, an

effective combination of the aforementioned mesh operations was developed and the results will be shown. The geometry which is being discretized is the "peaks" function which has a well-known parametrization [**?**]. The domain was chosen so that the boundary is sufficiently far away from the local minima/maxima present near the origin to show that the developed scheme is efficient at reducing representation deficit. The x and y values are each confined between -5.0 and 5.0 and the initial triangulation is two triangles that fill the quadrilateral defined by the four corner points.

The main driver of representation deficit, RD, based mesh refinement is the value for representation deficit. This is a ratio of the "before" area to the "after" area for each mesh operation. The surface area of the peaks function in the given domain was calculated by uniformly triangulating the surface with a very fine resolution. The result was found to converge to 177.944 with a u and v resolution of 301 points. For ease of presentation, the peaks functions is shown as a contour plot where red represents local maxima and blue represents local minima/maxima.

6.1 Triangle and Edge Splitting

Restricting the mesh refinement to only triangle splits or only edges splits is not useful, as discussed above. Therefore, the combination of the two was performed (as described in 3) to produce the following results. The following figure shows the results from $tol_{RD} = (0.5, 0.25, 0.125)$ from left to right. The mesh was refined by splitting triangles and edges where appropriate as described above.



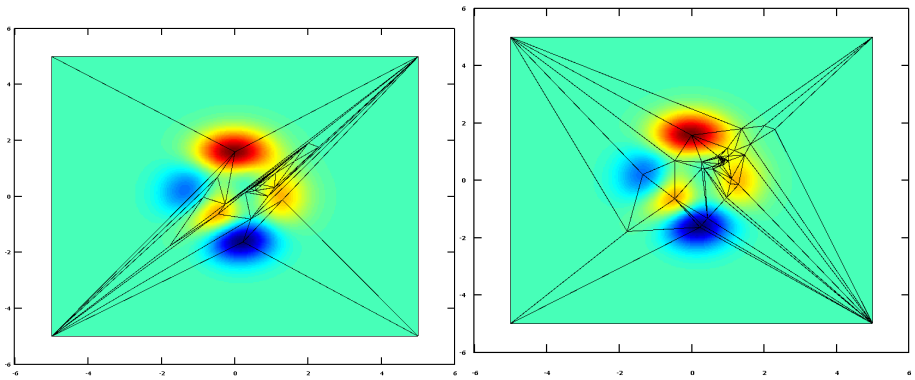
(a) $tol_{RD} = 0.5$ (b) $tol_{RD} = 0.25$ (c) $tol_{RD} = 0.125$

Fig. 5: Mesh Refinement Only

In 5a it can be seen that only three nodes were inserted. This is due to the very high value of tol_{RD} for this example. However, the nodes that were inserted were inserted very near to the local minima/maxima of the function. If tol_{RD} were halved to 0.25, 5b then 92 nodes are inserted. Again, the nodes are inserted very near to local minima/maxima and additionally near saddle points. If tol_{RD} is halved again to 0.125, 5c, no more nodes are inserted. This is due to the very poor quality triangles. The sliver triangles that can be seen in 5b, 5c represent nearly planar section of the mesh and therefore not refined further. This is the major shortcoming of the developed algorithm: element quality is not considered and this limitation leads to poor quality meshes that cannot be refined further.

6.2 Node Movement Effect

In order to show the effect of node movement, the resultant mesh shown in 5b is used as input and the same value for $tol_{RD} = 0.25$ is used with 4.



(a) $tol_{RD} = 0.25$ (b) $tol_{RD} = 0.25$

The node movement was very effective at reducing representation deficit. For this particular case, it reduced the representation deficit of the input mesh by 23.36%. It should also be noted that the local minima/maxima were accurately maintained by this procedure. It can also be seen that moved nodes that were nearby but not in local minima/maxim were moved to the local minima/maxima. Additionally, since the node movement was allowed to perform local reconnections where needed (discussed above), the element quality was also improved as a side-effect.

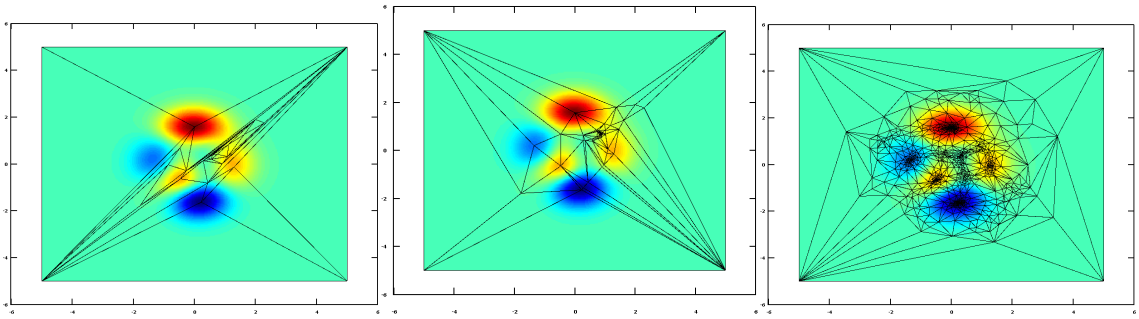
6.3 Effective Combination

It easily noted that while the aforementioned mesh operations are effective at reducing representation deficit they do not create high quality triangles. However, with the addition of a simple min-max edge flip pass in between refinement and node movement passes the element quality and representation deficit can be improved significantly.

In 6c the marked difference in mesh quality and mesh density caused by local reconnection can be seen. The addition of the local reconnection enabled the quality of triangle to be improved every step therefore allowed more nodes to be inserted. In this case, instead of 92 nodes being inserted (6a), 713 nodes were inserted. The surface area of the final mesh was 179.234. This is a representation deficit of 0.725%.

7 Conclusions

In this work, the concept of optimal mesh refinement was explored. This was done through the development of a formal definition for three fundamental mesh operations, triangle split, edge split, and node movement; each with respect to the concept of *representation deficit*. Results were given showing this methodology to be an effective method for mesh refinement. The most



(a) $tol_{RD} = 0.25$, no movement (b) $tol_{RD} = 0.25$, movement post (c) $tol_{RD} = 0.25$, reconnection

effective method was found to be a combination of the above three fundamental mesh operations and delaunay local reconnection. The representation is also efficient in that the interior of the domain was not overly refined in order to achieve the desired surface representation.

7.1 Future Work

There are several possible directions for future work. One avenue for future work is to further improve the optimization algorithm in the following two ways. First, optimization could be employed in the determination of which mesh operation to perform at each stage of the optimization procedure. Second, the objective function could be modified in order to generate surface meshes that simultaneously minimize the representation deficit and optimize the mesh quality. Another possibility for future research would be to extend the surface mesh generation algorithm to handle the volume mesh generation case so that a hierarchical approach for mesh generation (i.e., edge grid [**?**] $\mapsto$ surface mesh [**?**] $\mapsto$ volume mesh) could be employed. This would allow for optimization of each stage of the mesh generation procedure. Finally, parallelization of the algorithm would allow for optimal generation of very large surface meshes such as those required by **Dave: Cite a "killer" engineering application here.** applications.

The work of the second author is supported in part by NSF CAREER grant ACI-1330056 (formerly ACI-1054459).