REFERENCE JACOBIAN REZONING STRATEGY FOR ARBITRARY LAGRANGIAN-EULERIAN METHODS ON POLYHEDRAL GRIDS

Vadim Dyadechko[1] Rao Garimella[2] Mikhail Shashkov[3]

[1] *MS B284, T-7, Los Alamos National Lab., Los Alamos, NM 87545, U.S.A., vadik@t7.lanl.gov*
[2] *MS B284, T-7, Los Alamos National Lab., Los Alamos, NM 87545, U.S.A., rao@lanl.gov*
[3] *MS B284, T-7, Los Alamos National Lab., Los Alamos, NM 87545, U.S.A., shashkov@lanl.gov*

ABSTRACT

The rezoning step is an important part of the Arbitrary Lagrangian-Eulerian (ALE) simulation cycle. The objective of the rezoning algorithm is to improve geometric quality of the mesh elements by minimalistic repositioning of the mesh nodes. By means of numerical experiment we show that the Reference Jacobian rezoning effectively eliminates ill-shaped elements of 3D polyhedral grid while keeping the optimized grid close to the original one.

Keywords: ALE methods, Reference Jacobian rezoning algorithm, Jacobian-based geometrical quality measure, constrained optimization, polyhedral grid

1. INTRODUCTION

The relationship of the computational grid motion to the motion of a fluid is an important issue in the numerical simulation of multidimensional fluid flow. There are two choices that are typically made: one representing a Lagrangian framework and another representing a Eulerian one. The former assumes that the grid moves along with a fluid, the latter deals with a static grid.

The major advantage of the Lagrangian framework over the Eulerian one is a non-diffusive approximation of the advection term. Among other benefits, this property allows the mesh to follow the interface between different fluid phases. The downside of the Lagrangian approach is that the geometrical quality of the mesh elements may degrade significantly. Moreover, the mesh can eventually become entangled, causing the simulation to halt.

Arbitrary Lagrangian-Eulerian (ALE) methods [1, 2, 3, 4, 5, 6, 7] were introduced to exploit the advantages of the Lagrangian approach without facing mesh folding. The main idea of the ALE methodology is to move (*rezone*) the computational grid using the fluid flow *only as a guide*.

The three constitutive elements of the ALE simulation cycle are: an explicit Lagrangian step, a rezoning step in which nodes of the Lagrangian mesh are repositioned to improve geometric quality of the computational grid, and a remapping step in which the Lagrangian solution is transfered to the rezoned grid. Here we focus only on the rezoning phase of the ALE simulation.

A good rezoning algorithm should satisfy two competing criteria. First, it should *maintain a good geometrical quality of the computational grid* to minimize the approximation error. Second, it should *keep the rezoned grid adapted to the Lagrangian flow* to better resolve regions of rapid variation of the flow variables.

Satisfying these competing criteria in a robust and automatic rezoning algorithm is key to a successful ALE procedure. One can find a comprehensive survey of existing rezoning strategies presented in [8]. Although

the effective solution can be found for some special problems ([9, 1, 10, 11, 12, 13, 7, 14, 15]), very few of these schemes come close to meeting essential rezoning requirements when employed for a more general problem.

In this paper it is shown that the Reference Jacobian (RJ) rezoning algorithm [8] can be successfully used to improve the quality of polyhedral 3D meshes with minimal node movements. The RJ rezoning method can be briefly described as a a two-step smoothing algorithm. First it computes *locally* optimal positions for each Lagrangian node, and then uses knowledge of these *virtual* positions to construct a *global* functional. The minimizer of this global functional gives the coordinates of the rezoned mesh nodes. In the preceding publications on this technique [8, 16, 17, 18], the 2D and 3D surface mesh smoothing issues were addressed and here we present the results of 3D volume mesh optimization.

The rest of the paper is organized as follows. After presenting, in Section 2, an abstract outline of the strategy for the general geometry quality measure of a computational grid, we develop, in Sections 3 and 4, a complete rezoning algorithm for the specific choice of the shape quality measure, e.g. for the Frobenius condition number of the Jacobi matrix describing local distortion of mesh elements. Section 5 gives a detailed description of the optimization algorithm that was used for the arising nonlinear constrained minimization problems. We consider a NLCG-PR method with restarts [19, 20] coupled with a special line search strategy. Numerical examples of this rezoning technique are presented.

2. ABSTRACT OUTLINE OF THE STRATEGY

Let $F(\mathbf{x}_1, \ldots, \mathbf{x}_{N_V})$ be a function of mesh node coordinates $\mathbf{x}_i \in \mathbb{R}^3, i = 1, \ldots, N_V$ that measures quality of the mesh.

The first step of the rezoning algorithm is to find, for each mesh node V_i, $i = 1, \ldots, N_V$, the *virtual reference position* $\mathbf{x}_i^{(r)}$ given by the minimizer

$$\mathbf{x}_i^{(r)} = \arg \min_{\mathbf{x}_i \in \mathbb{R}^3} F_i(\mathbf{x}_i) \qquad (1)$$

of the functional

$$F_i(\mathbf{x}_i) \equiv F(\mathbf{x}_1^{(l)}, \ldots, \mathbf{x}_{i-1}^{(l)}, \mathbf{x}_i, \mathbf{x}_{i+1}^{(l)}, \ldots, \mathbf{x}_{N_V}^{(l)}),$$

where $\mathbf{x}_j^{(l)}$, $j = 1, \ldots, N_V$ are the Lagrangian (original) positions of mesh nodes.

Each objective function $F_i(\mathbf{x}_i)$, $i = 1, \ldots, N_V$ depends only on Lagrangian coordinates of the adjacent to the V_i mesh nodes and therefore is *local*. The optimization problem (1) corresponds to the mesh quality improvement attained by movement of a single mesh node. We have to emphasize that no actual node relocation occurs at this stage of the algorithm.

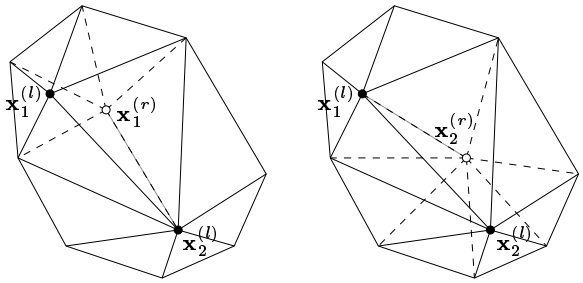


Figure 1: *Lagrangian and reference positions for the pair of adjacent mesh nodes V_1, V_2 in 2D.*

For each reference position $\mathbf{x}_i^{(r)}$ we consider a set of *reference edges – virtual* segments that connect an $\mathbf{x}_i^{(r)}$ with the neighbors of the mesh node V_i at their Lagrangian positions (dashed lines at Figure 1). Every reference edge is assumed to be directed from the reference point to the Lagrangian one. Since either end of a mesh edge has its own reference position, then either orientation of the mesh edge $E_j = V_{j_1} V_{j_2}$, $j = 1, \ldots, N_E$ has a respective reference counterpart:

$$\begin{aligned} \mathbf{e}_{j_1 j_2} = \mathbf{x}_{j_2} - \mathbf{x}_{j_2} \quad &\longleftrightarrow \quad \mathbf{e}^{(r)}_{j_1 j_2} = \mathbf{x}^{(l)}_{j_2} - \mathbf{x}^{(r)}_{j_1}, \\ \mathbf{e}_{j_2 j_1} = \mathbf{x}_{j_1} - \mathbf{x}_{j_2} \quad &\longleftrightarrow \quad \mathbf{e}^{(r)}_{j_2 j_1} = \mathbf{x}^{(l)}_{j_1} - \mathbf{x}^{(r)}_{j_2}. \end{aligned}$$

Here N_E is the total number of mesh edges.

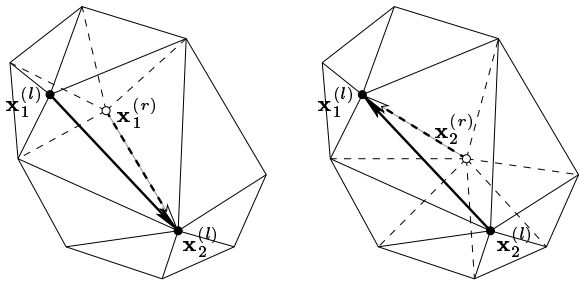


Figure 2: *Two different orientations of the edge $E = V_1 V_2$ and the respective reference counterparts in 2D.*

The second step of the algorithm is to find final nodal positions $(\mathbf{x}_1^*, \ldots, \mathbf{x}_{N_V}^*)$ by minimizing the deviation of mesh edges from their respective reference counterparts:

$$(\mathbf{x}_1^*, \ldots, \mathbf{x}_{N_V}^*) = \arg \min_{\substack{\mathbf{x}_i \in \mathbb{R}^3, \\ i=1,\ldots,N_V}} F_{RJ}(\mathbf{x}_1, \ldots, \mathbf{x}_{N_V}), \quad (2)$$

where

$$F_{RJ}(\mathbf{x}_1, \ldots, \mathbf{x}_{N_V}) \sim \sum_{j=1}^{N_E} \|\mathbf{x}_{j_1} - \mathbf{x}_{j_2} - \mathbf{e}^{(r)}_{j_2 j_1}\|^2 + \|\mathbf{x}_{j_2} - \mathbf{x}_{j_1} - \mathbf{e}^{(r)}_{j_1 j_2}\|^2.$$

The minimization of such an objective function pushes mesh edges toward their respective reference counterparts and they finally settle somewhere in between. Since one of the ends of the reference edge is given by the local minimizer of the the grid quality functional, we can expect that the minimization of the global F_{RJ} objective function also results in improvement of the geometrical quality of the grid.

Both (1) and (2) optimization problems have an important constraint: the computational grid is required to stay unfolded. This issue is related to the implementation of the optimization procedure and will be addressed later.

Below we describe in detail the rezoning framework for the particular choice of the geometrical quality measure that justifies the name of the Reference Jacobian method. We start with a simplicial grid and then show that the algorithm is transparently extensible on a wider class of polyhedral meshes.

3. THE CONDITION NUMBER SHAPE QUALITY MEASURE

Let Ω_h be a conforming partitioning of the computational domain Ω into tetrahedra. With each vertex V_i of the mesh tetrahedron T_k we can associate a 3x3 matrix

$$J_{i,k} \equiv J_{V_i,T_k} \stackrel{def}{=} [\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3],$$

where $\mathbf{e}_1, \mathbf{e}_2$, and $\mathbf{e}_3$ [1)] are vector representations of the edges that connect V_i with the other vertices of T_k.

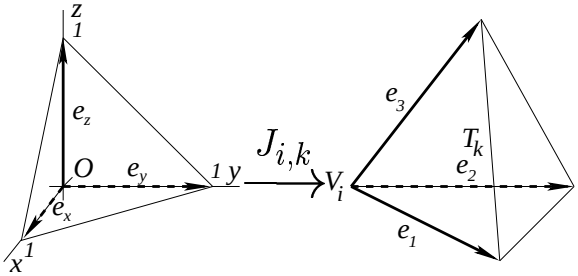


Figure 3: *Mapping between the octant corner of the canonical 3D simplex and the corner V_i of the tetrahedron T_k.*

This matrix $J_{i,k}$ is a Jacobi matrix of the mapping between the octant corner of the canonical 3D simplex and the corner V_i of the tetrahedron T_k (see Figure 3). The Jacobi matrix quantifies the distortion of the parent space and therefore can be used to derive a *shape quality measure* (SQM) of *the trivalent corner* V_i [21, 22]. We are particularly interested in the (Frobenius) condition number of the Jacobi matrix $J_{i,k}$

$$SQM_{V_i,T_k} \stackrel{def}{=} \mathrm{cond}_F(J_{i,k}) = \|J_{i,k}\|_F \cdot \|J_{i,k}^{-1}\|_F =$$

$$= \frac{\sqrt{\|\mathbf{e}_1\|^2 + \|\mathbf{e}_2\|^2 + \|\mathbf{e}_3\|^2}\sqrt{\|\mathbf{e}_1\times\mathbf{e}_2\|^2 + \|\mathbf{e}_2\times\mathbf{e}_3\|^2 + \|\mathbf{e}_3\times\mathbf{e}_1\|^2}}{|\det[\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3]|}.$$

The condition number is a good choice for the SQM of a trivalent corner because:

- it is dimensionless (*invariant to the scaling*);
- it is a convex function of the edge vectors (*the minimization problem is well-posed*);
- it is minimized by the *well-shaped* octant corner of a canonical simplex;
- it tends to infinity as the value of the Jacobian $\det J_{i,k}$ approaches zero (the corner becomes degenerate).

By averaging the condition number over the respective set of mesh corners we can consistently define

- a SQM of a single mesh element T_k:

$$SQM_{T_k} \stackrel{def}{=} \frac{1}{C} \cdot \frac{1}{N_V(T_k)} \sum_{V_i \in T_k} \mathrm{cond}_F(J_{i,k}),$$

  where $N_V(T_k)$ is the number of vertices per element (=4 for a tetrahedron);

- a SQM associated with a single mesh node V_i:

$$SQM_{V_i} \stackrel{def}{=} \frac{1}{C} \cdot \frac{1}{4N_T(V_i)} \sum_{T_k \ni V_i} \sum_{V_j \in Adj(V_i)} \mathrm{cond}_F(J_{j,k}),$$

  where $N_T(V_i)$ is the number of mesh elements that share the vertex V_i, $Adj(V_i)$ is a set that includes V_i along with all adjecent mesh nodes;

- a SQM of the whole computational grid Ω_h:

$$SQM_{\Omega_h} \stackrel{def}{=} \frac{1}{C} \cdot \frac{1}{N_C} \sum_{T_k} \sum_{V_i \in T_k} \mathrm{cond}_F(J_{i,k}),$$

  where N_C is the total number of mesh corners ($= 4\times$ *total number of mesh elements* for a tetrahedral grid).

[1)] To be rigorous, we have to equip each edge symbol with two extra subscripts i and k which denote the context of the V_i corner of the T_k tetrahedron. Nevertheless, to maintain the readability of the text, we prefer to drop these indices as long as the context is unambiguous.

The scaling factor

$$C \stackrel{def}{=} \min_{\substack{J\in\mathbb{R}^{3\times 3}\\ \det J\neq 0}} \mathrm{cond}_F(J) = 3$$

guarantees that $1 \leqslant SQM$.

If we were *only* interested in improving the shape quality of the grid with respect to the given SQM *as much as possible*, then we could have just minimized the global *condition number* (CN) objective function

$$\begin{aligned} F(\mathbf{x}) \equiv F_{CN}(\mathbf{x}) &= SQM_{\Omega_h}(\mathbf{x}) \\ &= \tfrac{1}{C}\cdot\tfrac{1}{N_C}\sum_{T_k}\sum_{V_i\in\overline{T}_k}\mathrm{cond}_F(J_{i,k}(\mathbf{x})), \end{aligned} \tag{3}$$

where $\mathbf{x} = [\mathbf{x}_1^T,\ldots,\mathbf{x}_{N_V}^T]^T \in \mathbb{R}^{3N_V}$ is a vector of all node coordinates.

But improvement of the geometrical quality of the grid is only one of the two objectives of the rezoning procedure. That is why a more complex smoothing scheme has to be employed.

4. REFERENCE JACOBIAN REZONING ALGORITHM

Following the abstract outline, first we find the *virtual reference position* $\mathbf{x}_i^{(r)}$ for each mesh node V_i, $i = 1,\ldots,N_V$. For this we have to solve a set of N_V minimization problems with local objective functions

$$\begin{aligned} F_i(\mathbf{x}_i) &= SQM_{V_i}(\mathbf{x}_i) \\ &= \tfrac{1}{C}\cdot\tfrac{1}{4N_T(V_i)}\sum_{\overline{T}_k\ni V_i}\sum_{V_j\in Adj(V_i)}\mathrm{cond}_F(J_{j,k}(\mathbf{x}_i)), \\ &\quad \mathbf{x}_i\in\mathbb{R}^3, \quad i=1,\ldots,N_V. \end{aligned}$$

Once the reference nodes are found, we can define *reference edges — virtual* segments that connect reference nodes to their neighbors fixed at the original (Lagrangian) positions (see Figure 2).

At this point for each vertex V_i of the tetrahedron T_k we can introduce a *reference Jacobi matrix*

$$J_{i,k}^{(r)} \equiv J_{V_i,T_k} \stackrel{def}{=} [\mathbf{e}_1^{(r)},\mathbf{e}_2^{(r)},\mathbf{e}_3^{(r)}],$$

where $\mathbf{e}_1^{(r)},\mathbf{e}_2^{(r)}$, and $\mathbf{e}_3^{(r)}$ are the reference counterparts of the respective edges $\mathbf{e}_1,\mathbf{e}_2$, and $\mathbf{e}_3$ associated with the V_i (see Figure 4).

The global Reference Jacobian objective function is now given by

$$F_{RJ}(\mathbf{x}) = \frac{1}{N_C}\sum_{T_k}\sum_{V_i\in\overline{T}_k}\frac{\det J_{i,k}^{(r)}}{\det J_{i,k}(\mathbf{x})}\frac{\|J_{i,k}(\mathbf{x})-J_{i,k}^{(r)}\|_F^2}{\|J_{i,k}^{(r)}\|_F^2}. \tag{4}$$

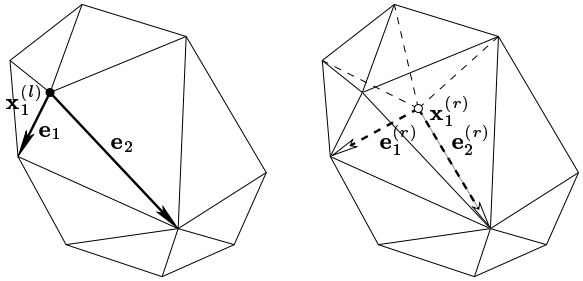


Figure 4: *The bases for the Jacobi matrices* $J_{1,k} = [\mathbf{e}_1,\mathbf{e}_2]$ *and* $J_{1,k}^{(r)} = [\mathbf{e}_1^{(r)},\mathbf{e}_2^{(r)}]$ *in 2D.*

This objective function penalizes deviation of mesh edges from the reference counterparts and takes some efforts to prevent mesh degeneration by introducing barrier factors of $1/\det J_{i,k}(\mathbf{x})$ for each term. The reciprocal barrier, though, has a discontinuity and doesn't guarantee that the grid stays unfolded.

As we already mentioned, both reference position recovery procedure and the RJ-minimization fall into the category of constrained optimization problems: a computational grid should always be kept unfolded. As long as the initial Lagrangian mesh is unfolded, one can easily detect the mesh entanglement by the alteration of the sign of some Jacobian $\det J_{i,k}$. Without loss of generality, we can assume that all Jacobians $\det J_{i,k}$ of the Lagrangian (original) grid are positive. Then the feasible regions $D(F_i)$ and $D(F_{RJ})$ can be formally described as

$$D(F_i) = \{\mathbf{x}_i\in\mathbb{R}^3 : 0 < \det J_{j,k}(\mathbf{x})\ \forall\ \overline{T}_k\ni V_i,\ V_j\in Adj(V_i)\},\quad i=1,\ldots,N_V,$$

$$D(F_{RJ}) = \{\mathbf{x}\in\mathbb{R}^{3N_V} : 0 < \det J_{i,k}(\mathbf{x})\ \forall\ T_k,\ V_i\in\overline{T}_k\}.$$

Since the evaluation of both local and global objective functions requires the calculation of the respective Jacobians anyway, the validation procedure does not produce any computational overhead.

All functions $F_i(\mathbf{x}_i)$, $i = 1,\ldots,N_V$ and $F_{RJ}(\mathbf{x})$ are defined by means of the transformation matrices for the respective trivalent mesh corners. Therefore this algorithm is transparently applicable to all types of polyhedral meshes with trivalent corners such as:

- simplicial,
- prismatic,
- hexagonal,
- and non-degenerate Voronoi meshes.

5. OPTIMIZATION PROCEDURE

Here we give a description of the multivariate optimization procedure employed. The algorithms are presented in informal algorithmic language and accompanied by short annotations.

5.1 Nonlinear conjugate gradient method

A smooth multivariate objective function $f : \mathbb{R} \longrightarrow \mathbb{R}^n$ with the convex feasible region $D(f) \subset \mathbb{R}^n$ can be effectively minimized by means of the nonlinear conjugate gradient (NLCG) method.

Given current iterate $\mathbf{x}_k$, $k = 0, 1, 2, \ldots$, an NLCG method

1) calculates the *search direction*

$$\mathbf{p}_k = \begin{cases} -\mathbf{g}_k, & k = 0, \\ -\mathbf{g}_k + \beta_k \mathbf{p}_{k-1}, & k = 1, 2, 3, \ldots, \end{cases}$$

where

$$\mathbf{g}_k \equiv \nabla f(\mathbf{x}_k),$$

2) by means of the *line search* algorithm identifies a *step length* α_k along the search direction $\mathbf{p}_k$ that gives a substantial reduction of the objective function:

$$f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) < f(\mathbf{x}_k),$$

3) and then advances to the next iterate

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k.$$

Following the Polak-Ribière method [19] with restarts [20], which is known to be the most robust one in the class of NLCG optimization routines, we select

$$\beta_k = \max\{\frac{\mathbf{g}_k^T (\mathbf{g}_k - \mathbf{g}_{k-1})}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}}; 0\}, \quad k = 1, 2, 3, \ldots$$

Note that this choice of β_k guarantees that $\mathbf{p}_k$ is always a descent direction.

The iteration process is terminated either when the the objective function gradient becomes sufficiently small

$$\|\mathbf{g}_k\|_2 \leqslant tol_g$$

or when the argument increment falls below the given threshold

$$\|\alpha_k \mathbf{p}_k\|_2 \leqslant tol_x,$$

whatever happens first.

Algorithm 1: $\mathbf{x} \longleftarrow conj\_grad(f,\ \mathbf{x}_0)$

input:

1) a smooth objective function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ with a convex feasible region $D(f) \subset \mathbb{R}^n$
2) an initial guess $\mathbf{x}_0 \in D(f)$

output:

1) an approximation of some local minimizer of f

begin

evaluate $\mathbf{g}_0 \longleftarrow \nabla f(\mathbf{x}_0)$;
$\mathbf{p}_0 \longleftarrow -\mathbf{g}_0$;
for $k \longleftarrow 0,\ 1,\ 2,\ \ldots$ **do**
  if $\|\mathbf{g}_k\|_2 \leqslant tol_g$ **then**
    return $\mathbf{x}_{k+1}$;
  define $\phi_k(\alpha) \equiv f(\mathbf{x}_k + \alpha \mathbf{p}_k)$;
  if $k = 0$ **then**
    pick some $\Delta\alpha > tol_x$ based on an a priori knowledge about ϕ_0 ;
  else
    $\Delta\alpha \longleftarrow \alpha_{k-1}$;
  $\alpha_k \longleftarrow line\_search(\phi_k,\ \Delta\alpha)$;
  if $\alpha_k \|p_k\|_2 \leqslant tol_x$ **then**
    return $\mathbf{x}_k$;
  $\mathbf{x}_{k+1} \longleftarrow \mathbf{x}_k + \alpha_k \mathbf{p}_k$;
  evaluate $\nabla f(\mathbf{x}_{k+1})$;
  $\mathbf{g}_{k+1} \longleftarrow \nabla f(\mathbf{x}_{k+1})$;
  $\beta_{k+1} \longleftarrow \frac{\mathbf{g}_{k+1}^T (\mathbf{g}_{k+1} - \mathbf{g}_k)}{\mathbf{g}_k^T \mathbf{g}_k}$;
  if $\beta_{k+1} < 0$ **then**
    $\beta_{k+1} \longleftarrow 0$;
  $\mathbf{p}_{k+1} \longleftarrow -\mathbf{g}_{k+1} + \beta_{k+1} \mathbf{p}_k$;

end

5.2 Line search algorithm

The ideal choice for the step length α_k would be the global minimizer of the univariate function

$$\phi_k(\alpha) \equiv f(\mathbf{x}_k + \alpha \mathbf{p}_k)$$

over the feasible region

$$D(\phi_k) = \{\alpha \geqslant 0 \mid \mathbf{x}_k + \alpha \mathbf{p}_k \in D(f)\},$$

but the exact solution of this problem is too expensive. It is more practical to perform an *inexact line search* to identify a step length $\alpha_k \in D(\phi_k)$ that provides an adequate reduction in f at minimal cost.

We adapted the line search algorithm described in [23] to fit the constrained minimization problem we solve. It exploits popular inexact line search criteria known as *the strong Wolfe conditions*:

$$\phi_k(\alpha_k) \leqslant \phi_k(0) + c_1 \alpha_k \phi_k'(0) \quad (\textit{sufficient decrease condition})$$

$$|\phi_k'(\alpha_k)| \leqslant -c_2 \phi_k'(0) \quad (\textit{curvature condition})$$

where

$$0 < c_1 < c_2 \text{ [2)]} < 1.$$

From now on we will drop the subscript k that identifies the NLCG iteration count unless this will introduce confusion.

The line search algorithm consists of two successive steps:

1) first it generates a monotonically increasing sequence of trial steps

   $$0 = \alpha_0 < \ldots < \alpha_j < \ldots < \alpha_i, \quad \alpha_j \in D(\phi),\ j = \overline{0,i}$$

   until either

   - the sufficient decrease condition is violated:

     $$\phi(\alpha_i) > \phi(0) + c_1 \alpha_i \phi'(0),$$

   - the function goes up:

     $$\phi(\alpha_i) > \phi(\alpha_{i-1}),$$

   - or merely the point of increase of ϕ is found:

     $$\phi'(\alpha_i) \geqslant 0;$$

   the interval $[\alpha_{i-1}, \alpha_i]$ is known to contain a step length that satisfies the strong Wolfe conditions;

2) then it applies a *zoom* procedure to the interval $[\alpha_{i-1}, \alpha_i]$ to identify the required step length.

The performance of the line search algorithms depends highly on the *step length selection* strategy that governs the process of the trial step sequence generation. In order to maintain good performance of the algorithm on a wide range of input data, the *line_search* takes an additional argument $\Delta\alpha$ that is used as an initial value for the step length increment.

[2)] The common values for these constants are $c_1 = 10^{-4}$ and $c_2 = 10^{-1}$.

[3)] $tol_\alpha \leqslant tol_x / \|\mathbf{p}_k\|_2$

Algorithm 2: $\alpha \longleftarrow line\_search(\phi,\ \Delta\alpha)$

input:

1) a smooth objective function $\phi : \mathbb{R} \longrightarrow \mathbb{R}$ such that $\phi'(0) < 0$ and the feasible region $D(\phi)$ is known to include a continuous interval $[0, \alpha_{max}[$,
2) suggested step length increment $\Delta\alpha > 0$

output:

1) the step length α_i that satisfies the strong Wolfe conditions

begin

```
α_max ⟵ ∞ ;
α_0 ⟵ 0 ;
for i ⟵ 1, 2, 3, ... do
    Δα, α_max ⟵
        step_length(α_{i-1}, Δα, α_max) ;
    if Δα ⩽ tol_α [3)] then
        return α_i ;
    α_i ⟵ α_{i-1} + Δα ;
    evaluate φ(α_i) ;
    if φ(α_i) > φ(0) + c_1 α_i φ'(0) or
        (φ(α_i) ⩾ φ(α_{i-1}) and i > 1) then
        return zoom(i, α_{i-1}, α_i) ;
    evaluate φ'(α_i) ;
    if |φ'(α_i)| ⩽ −c_2 φ'(0) then
        return α_i ;
    if φ'(α_i) ⩾ 0 then
        return zoom(i, α_i, α_{i-1}) ;
```

end

5.3 Step length selection strategy

Given the last trial step α_i and the suggested increment $\Delta\alpha$, the *step_length* function returns the next trial step $\alpha_{i+1} \in D(\phi)$ and the actual step length increment $\Delta\alpha_i$.

If $\alpha_i + \Delta\alpha$ happens to be out of $D(\phi)$, then the algorithm gradually decreases $\Delta\alpha$ by the factor

$1/\rho,\ 0 < \rho^{4)} < 1$ until $\alpha_i + \Delta\alpha \in D(\phi)$.

An initial value of $\Delta\alpha$ is picked by the upper level routine (*conj_grad* in our case) based on the some a priori information about the ϕ. Each next iteration the *line_search* feeds the *step_length* routine with the last $\Delta\alpha_i$ accepted so far.

In order to avoid unnecessary computations, the algorithm also utilizes information about all unsuccessful trials to keep a track of $\overline{\alpha_{max}}$ — the minimal known $\alpha \notin D(\phi)$. This helps to identify some $\Delta\alpha$ to be invalid without even evaluating the characteristic function at $\alpha = \alpha_i + \Delta\alpha$.

Algorithm 3: $\alpha_{i+1},\ \Delta\alpha,\ \overline{\alpha_{max}} \longleftarrow$ *step_length*$(\alpha_i,\ \Delta\alpha,\ \overline{\alpha_{max}})$

---

input:

1) the last trial step $\alpha_i \in D(\phi)$ generated so far
2) suggested step length increment $\Delta\alpha > 0$
3) the estimate of the α_{max} from above $\overline{\alpha_{max}}$

output:

1) $\max\{\rho^n \Delta\alpha\},\ n \in \mathbb{Z}^+$ subject $\alpha + \rho^n \Delta\alpha \in D(\phi)$
2) $\min\{\alpha + \rho^n \Delta\alpha\},\ n \in \mathbb{Z}^+$ subject $\alpha + \rho^n \Delta\alpha \notin D(\phi)$
3) updated value of $\overline{\alpha_{max}}$

begin

- **while** $\Delta\alpha > tol_\alpha$ **and** $\alpha_i + \Delta\alpha \geqslant \overline{\alpha_{max}}$ **do**
  - $\Delta\alpha \longleftarrow \rho\Delta\alpha$;
- **while** $\Delta\alpha > tol_\alpha$ **and** $(\alpha_i + \Delta\alpha) \notin D(\phi)$ **do**
  - **if** $\alpha + \Delta\alpha < \overline{\alpha_{max}}$ **then**
    - $\overline{\alpha_{max}} \longleftarrow \alpha_i + \Delta\alpha$;
  - $\Delta\alpha \longleftarrow \rho\Delta\alpha$;
- **return** $\alpha_{i+1},\ \Delta\alpha,\ \overline{\alpha_{max}}$;

end

5.4 Zoom algorithm

The order of the input arguments α_{lo} and α_{hi} of the *zoom* procedure is chosen such that

1) the interval bounded by α_{lo} and α_{hi} contains step lengths that satisfy the strong Wolfe conditions;
2) α_{lo} is, among all step lengths generated so far and satisfying the sufficient decrease condition, the one giving the smallest function value; and
3) α_{hi} is chosen so that $\phi'(\alpha_{lo})(\alpha_{hi} - \alpha_{lo}) < 0$.

The interval that satisfies all these three properties we call a *target interval*.

Algorithm 4: $\alpha \longleftarrow$ *zoom*$(i,\ \alpha_{lo},\ \alpha_{hi})$

---

input:

1) number of the trial steps generated so far i
2) bounds of the target interval α_{lo}, α_{hi}

output:

1) the step length α between α_{lo} and α_{hi} that satisfies the strong Wolfe condition

begin

- **if** $|\alpha_{hi} - \alpha_{lo}| \leqslant tol_\alpha$ **then**
  - **return** α_{lo} ;
- $i \longleftarrow i + 1$;
- $\alpha_i \longleftarrow$ *interpolate*$(\alpha_{lo},\ \alpha_{hi})$;
- evaluate $\phi(\alpha)$;
- **if** $\phi(\alpha_i) > \phi(0) + c_1\alpha\phi'(0)$ **or** $\phi(\alpha_i) \geqslant \phi(\alpha_{lo})$ **then**
  - **return** *zoom*$(i,\ \alpha_{lo},\ \alpha_i)$;
- evaluate $\phi'(\alpha)$;
- **if** $|\phi'(\alpha_i)| \leqslant -c_2\phi'(0)$ **then**
  - **return** α ;
- **if** $\phi'(\alpha_i)(\alpha_{hi} - \alpha_{lo}) \geqslant 0$ **then**
  - **return** *zoom*$(i,\ \alpha_i, \alpha_{lo})$;
- **else**
  - **return** *zoom*$(i,\ \alpha_i, \alpha_{hi})$;

end

The logics of the *zoom* algorithm follows the *divide and conquer* principle. Each call of *zoom* generates an iterate α_i between α_{lo} and α_{hi} and then replaces one of the endpoints by α_i in such a way that all three properties continue to hold. If the new iterate α_i happens to satisfy the strong Wolfe conditions, the *zoom* terminates and returns α_i. Otherwise, if α_i satisfies the sufficient decrease condition and has a lower function value than α_{lo}, then we set $\alpha_{lo} \longleftarrow \alpha_i$ to maintain

[4)] We used the value $\rho = 1/2$.

condition 2). If this results in violation of condition 3), we remedy the situation by setting α_{hi} to the old value of α_{lo}.

5.5 Interpolation algorithm

The interpolation algorithm generates a new iterate α_i for the *zoom* procedure based on the values of ϕ and ϕ' at the endpoints of the given interval.

First the α_i is set to the local minimizer of the cubic polynomial that interpolates $\phi(\alpha_{lo}), \phi'(\alpha_{lo}), \phi(\alpha_{hi})$, and $\phi'(\alpha_{hi})$. For any target interval there exists a unique α between α_{lo} and α_{hi} that is a local minimizer of such a polynomial.

In order to maintain a *persistent minimal* rate of convergence of the *zoom* algorithm, we have to ensure that the new step length is not too close to the endpoints. With the target interval, it suffices to shift α_i toward α_{hi} if it happens to be too close to the α_{lo} ($|\alpha_i - \alpha_{lo}| < tol_{\alpha_{lo}}$[5])$|\alpha_{hi} - \alpha_{lo}|$).

Algorithm 5: $\alpha \longleftarrow interpolate(\alpha_{lo},\ \alpha_{hi})$

---

input:

1) bounds of the target interval α_{lo}, α_{hi}

output:

1) the new iterate α_i for the *zoom* procedure

begin

$d_1 \longleftarrow \phi'(\alpha_{lo}) + \phi'(\alpha_{hi}) - 3\,\dfrac{\phi(\alpha_{lo}) - \phi(\alpha_{hi})}{\alpha_{lo} - \alpha_{hi}}$;

$d_2 \longleftarrow \sqrt{d_1^2 - \phi'(\alpha_{lo})\phi'(\alpha_{hi})}$;

if $\alpha_{hi} < \alpha_{lo}$ **then**

$d_2 \longleftarrow -d_2$;

$\alpha_i \longleftarrow \alpha_{hi} - (\alpha_{hi} - \alpha_{lo})\dfrac{\phi'(\alpha_{hi}) + d_2 - d_1}{\phi(\alpha_{hi}) - \phi(\alpha_{lo}) + 2d_2}$;

if $|\alpha_i - \alpha_{lo}| < tol_{\alpha_{lo}}|\alpha_{hi} - \alpha_{lo}|$ **then**

$\alpha_i \longleftarrow \alpha_{lo} + tol_{\alpha_{lo}}(\alpha_{hi} - \alpha_{lo})$;

return α_i ;

end

5.6 Numerical differentiation

Each step of the NLCG procedure requires knowledge of the value of the multivariate objective function $f(\mathbf{x})$ and the gradient $\nabla f(\mathbf{x})$, $\mathbf{x} \in D(f)$. The linear search subroutine, in its turn, relies on the information about derivatives $\frac{\partial f}{\partial \mathbf{p}}$ along the search direction $\mathbf{p} \in \mathbb{R}^3$. Whenever analytical partial derivatives of $f(\mathbf{x})$ are not available or the straightforward calculation according to the formulae is expensive, the numerical differentiation is employed. Since the ultimate precision is not the priority, the backward difference formula seems to be a reasonable choice (it takes only one extra evaluation of the objective function):

$$\frac{\partial f}{\partial \mathbf{p}}(\mathbf{x}) = \frac{f(\mathbf{x}) - f(\mathbf{x} - \delta\mathbf{p})}{\delta} + O(\delta),$$

where $\mathbf{p} \in \mathbb{R}^n$ is a unit vector that defines the direction of the differentiation and $\delta \in \mathbb{R}$ is an argument increment. Simple analysis shows that, in order to minimize the round-off error, one has to pick $\delta \sim \sqrt{\varepsilon_{mach}}$, where ε_{mach} is a precision of the floating point arithmetic for the particular computer architecture.

The additive form of the SQM allows to calculate any partial derivative $\frac{\partial f}{\partial x_i}$, $i = 1, \ldots, n$ with $O(1)$ operations. A gradient ∇f or a derivative along an arbitrary direction $\frac{\partial f}{\partial \mathbf{p}}$ can be evaluated with $O(n)$ operations. Proper dynamic caching of intermediate results may save a lot of CPU time and significantly boost the performance of the entire optimization routine.

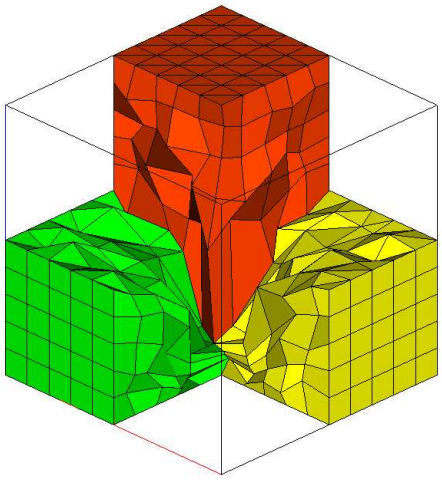
6. NUMERICAL EXAMPLES

Here we present some numerical examples of this optimization technique. An initial grid configuration for each experiment was prepared by introducing some long-wave-length distortion mixed with a short-wave-length noise to the respective smooth quasi-uniform mesh.

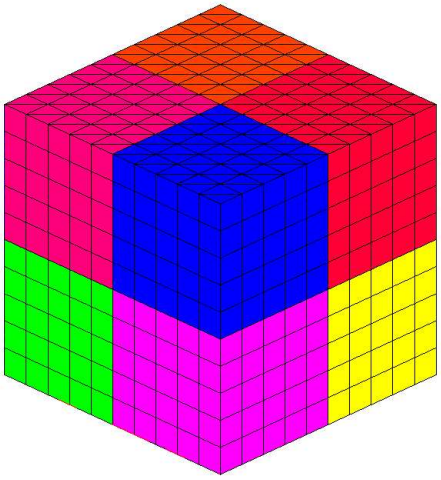
In order to show the inherent difference between the rezoning algorithm and a generic smoothing technique, we performed both RJ (4) and CN (3) optimizations on the grids. The CN optimization gives an idea of the best possible quality of the grid that can be attained with a smoothing procedure. One can see that CN-optimized grids exhibit no memory of an initial state. The RJ optimization, on the contrary, preserves a long-wave-length distortion pattern of the grid, efficiently eliminating ill-shaped elements. The improvement of mesh geometry quality in this case is compared to that given by the CN optimization, while the average node movement is much less.

[5] We used the value $tol_{\alpha_{lo}} = 0.1$

6.1 Example 1: prismatic mesh

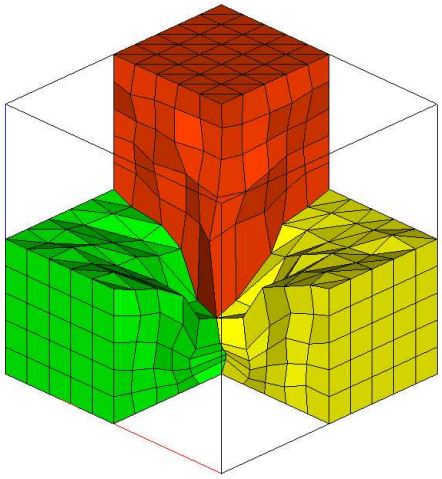


Original mesh (cut)



Original mesh

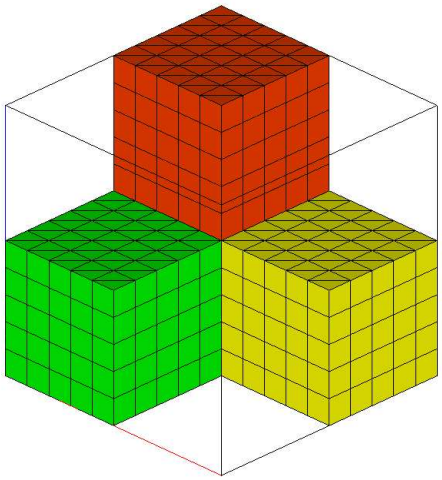
Number of nodes = 1331
Number of elements = 2000



RJ-optimized mesh (cut)

Node displacement, % of the average local spacing		
	RJ-optimized	*CN-optimized*
min	1.0	3.2
avg	15.9	47.5
max	67.7	297.8

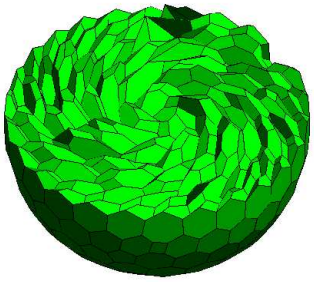
Element shape quality			
	Original mesh	*RJ-optimized*	*CN-optimized*
min	1.1	1.1	1.2
avg	2.1	1.4	1.2
max	377.8	3.7	1.2



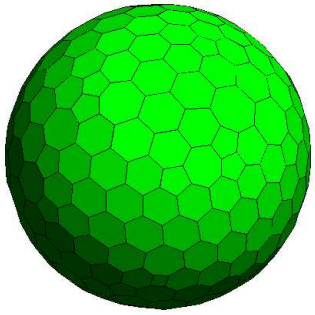
CN-optimized mesh (cut)

Shape quality	**Fraction of elements, %**		
	Original mesh	*RJ-optimized*	*CN-optimized*
1.00 – 1.25	12.3	32.6	100.0
1.25 – 1.50	40.5	53.7	0.0
1.50 – 2.00	27.1	10.6	0.0
2.00 – 3.00	12.6	2.8	0.0
3.00 – 5.00	5.2	0.3	0.0
5.00 – 9.00	1.6	0.0	0.0
9.00 –17.00	0.5	0.0	0.0
17.00 – ∞	0.2	0.0	0.0

6.2 Example 2: Polyhedral Voronoi mesh

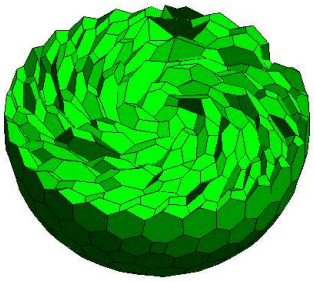


Original mesh (cut)



Original mesh

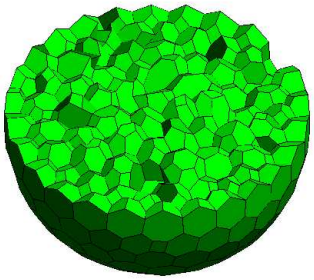
Number of nodes = 3566
Number of elements = 663



RJ-optimized mesh (cut)

Node displacement, % of the average local spacing		
	RJ-optimized	*CN-optimized*
min	0.1	11.5
avg	7.1	183.3
max	138.4	591.9

Element shape quality			
	Original mesh	*RJ-optimized*	*CN-optimized*
min	1.1	1.2	1.0
avg	12.4	2.1	1.3
max	6662.7	5.9	1.8



CN-optimized mesh (cut)

Shape quality	**Fraction of elements, %**		
	Original mesh	*RJ-optimized*	*CN-optimized*
1.00 – 1.25	0.9	0.8	26.2
1.25 – 1.50	5.1	4.5	71.0
1.50 – 2.00	36.2	38.2	2.7
2.00 – 3.00	41.0	51.4	0.0
3.00 – 5.00	14.3	5.0	0.0
5.00 – 9.00	2.3	0.2	0.0
9.00 –17.00	0.0	0.0	0.0
17.00 – ∞	0.2	0.0	0.0

7. ACKNOWLEDGMENTS

This work was performed at the Los Alamos National Laboratory operated by the University of California for the US Department of Energy under the contract W-7405-ENG-36. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of publication or guarantee its technical correctness.

The authors also acknowledge a partial support of the DOE/ASCR Program in the Applied Mathematical Sciences, the Laboratory Directed Research Development (LDRD) program, and the DOE Accelerated Strategic Computing Initiative (ASCI).

References

[1] Benson D.J. "An Efficient, Accurate, Simple ALE Method for Nonlinear Finite Element Programs." *Computer Methods in Applied mathematics and Engeneering*, vol. 72, 305–350, 1989

[2] Benson D.J. "Computational Methods in Lagrangian and Eulerian Hydrocodes." *Computer Methods in Applied Mechanics and Engineering*, vol. 72, 305–350, 1989

[3] Hirt C.W., Amsden A.A., Cook J.L. "An Arbitrary Lagrangian-Eulerian Finite Element Methods." *Computational Mechanics*, vol. 21, 203–216, 1998

[4] Kershaw D.S., Pradas M.K., Shaw M.J., Milovich J.L. "3D Unstructured Mesh ALE Hydrodynamics with the Upwind Discontinuous Finite Element Method." *Computer Methods in Applied Mathematics and Engineering*, vol. 158, 81–116, 1998

[5] Kjellgren P., Hyvarien J. "An Arbitrary Lagrangian-Eulerian Finite Element Method." *Computational Mechanics*, vol. 21, 81–90, 1998

[6] Margolin L.G. "Introduction to an Arbitrary Lagrangian-Eulerian Computing Method for All Flow Speeds." *Journal of Computational Physics*, vol. 135, 198–202, 1997

[7] Perry J.S., Carroll D.E. "Multi-Material ALE in Unstructured Grids." *Computer Methods in Applied Mathematics and Engineering*, vol. 187, 591–619, 2000

[8] Knupp P.M., Margolin L.G., Shashkov M. "Reference Jacobian Optimization-Based Rezone Strategies for Arbitrary Lagrangian-Eulerian Methods." *Journal of Computational Physics*, vol. 176, no. 1, 93–128, Feb. 2002

[9] Baines M.J. *Moving Finite Elements*. Oxford Science Publications. Claredon Press, Oxford, 1994

[10] Brackbill J.U., Saltzman J.S. "Adaptive for Singular Problems in Two Dimensions." *Journal of Computational Physics*, vol. 46, 342–368, 1982

[11] Charakhchyan A., Ivanenko S. "A Variational Form of the Winslow Grid Generator." *Journal of Computational Physics*, vol. 136, 385–398, 1997

[12] Dukowicz J.K. "A Simplified Adaptive Mesh Technique Derived from the Moving Finite Element Method." *Journal of Computational Physics*, vol. 56, 689–723, 1982

[13] Dukowicz J.K., Cline M.C., Addessio F.L. "A General Topology Godunov Method." *Journal of Computational Physics*, vol. 82, 29–63, 1989

[14] Stoker C., Gay C., Bay F., Chenot J.L. *A Velocity Approach for the ALE-method Applied for 2D and 3D Problems*. Huetink and Baanijens, Balkema, Rotterdam, 1998

[15] Winslow A. "Numerical Solution of the Quasilinear Poisson Equations in a Nonuniform Triangle Mesh." *Journal of Computational Physics*, vol. 1, 149–172, 1966

[16] Garimella R., Shashkov M., Knupp P. "Optimization of Surface Mesh Quality Using Local Parametrization." *11th International Meshing Roundtable*. Sandia National Laboratories, 2002

[17] Garimella R., Knupp P., Shashkov M. "Triangular and Quadrilateral Surface Mesh Quality Optimization Using Local Parametrization." *Computer Methods in Applied Mechanics and Engineering*, vol. 193, no. 9–11, 913–928, Mar. 2004

[18] Garimella R., Shashkov M. "Polygonal Surface Mesh Optimization." 2004. URL `http://math.lanl.gov/Research/Publications/Docs/garimella-2004-polygonal.pdf`. To be published in *Engineering with Computers*

[19] Polak E., Rebière G. "Note sur la convergence de méthodes de directions conjuguées." *Revue Françise d'Informatique et de Recherche Opérationalle*, vol. 16, 35–43, 1969. In French

[20] Powell M.J.D. "Restart Procedures for the Conjugate Gradient Method." *Mathematical Programming*, vol. 12, 241–254, 1977

[21] Knupp P.M. "Achieving Finite Element Mesh Quality via Optimization of the Jacobian Matrix Norm and Associated Quantitites. Part II: A Framework for Volume Mesh Optimization and the Condition Number of the Jacobian Matrix." *IJNME*, vol. 48, 401–420, 2000

[22] Jacquotte O.P., Cabello J. "A new variational method for the generation of two- and three-dimensional adapted grids in computational fluid dynamics." Tech. Rep. TP 1989-31, ONERA, France, 1989

[23] Nocedal J., Wright S.J. *Numerical Optimization.* Springer–Verlag, New York, 1999