

UNSTRUCTURED MOVING MESH BY A LEAST-SQUARES FINITE ELEMENT METHOD

Xian-Xin Cai ⁽¹⁾, Dionisio Fleitas ⁽²⁾, Bonan Jiang ⁽³⁾ and Guojun Liao ⁽²⁾

⁽¹⁾ Nanhua Power Research Institute,
Zhuzhou, Hunan 412002, PR China

⁽²⁾ Department of Mathematics, University of Texas,
Arlington, Texas 76019

⁽³⁾ Department of Mathematics and Statistics, Oakland University,
Rochester, Michigan

A method for generating adaptive unstructured meshes is described. In the adaptive method, we use Least-Squares Finite Element Method to compute the node velocity according to a monitor function measuring the error of the solution. The node connectivity is unchanged. Various optimization procedures can be applied after adaptation if necessary. The deformed grid becomes refined in regions where the solution error is large.

For a user-defined monitor function f , the method controls the cell size by making the Jacobian determinant $J = f$, and thus, it generates desirable cell sizes. Moreover, since $f > 0$, the adaptive grid is non-folding. Boundary conditions are easily enforced with LSFEM and it always leads to symmetric positive-definite matrices which can be efficiently solved. Slippery wall condition ensures that boundary nodes move along the boundary. Inflow conditions can be imposed for free surface and moving boundary problems. The residual in LSFEM is an ideal error indicator which can be used to construct the monitor function.

The method determines the node velocity by solving a div-curl system as follows.

Assuming that $f(\mathbf{x}, t) > 0$ for $\mathbf{x} \in \Omega = \Omega(t)$ (physical domain) and $t > 0$, and such that $\int_{\Omega} \frac{1}{f(\omega, t)} d\omega = |\Omega|$. We look for ϕ on $\Omega = \Omega(t)$ such that $\det \nabla \phi(\mathbf{x}, t) = f(\phi(\mathbf{x}, t), t)$ at any time t . Two steps to obtain ϕ :

First, find a vector field \mathbf{u} , by LSFEM, that satisfies, for $t_{k-1} \leq t \leq t_k$ where \mathbf{n} is the outward normal to $\partial\Omega$,

$$\begin{aligned} \operatorname{div} \mathbf{u}(\mathbf{x}, t) &= -\frac{\partial}{\partial t} \left(\frac{1}{f(\mathbf{x}, t)} \right), & \mathbf{x} \in \Omega, \\ \operatorname{curl} \mathbf{u}(\mathbf{x}, t) &= 0, & \mathbf{x} \in \Omega, \\ \mathbf{u}(\mathbf{x}, t) \cdot \mathbf{n} &= 0 \quad \text{or} \quad \mathbf{u} = g & \mathbf{x} \in \partial\Omega. \end{aligned}$$

Second, compute ϕ by solving the system for a node \mathbf{x} :

$$\frac{\partial}{\partial t} \phi(\mathbf{x}, t) = f(\phi(\mathbf{x}, t), t) \mathbf{u}(\mathbf{x}, t), \quad \mathbf{x} \in \Omega, \text{ for } t_{k-1} \leq t \leq t_k$$

It can be proved that ϕ satisfies

$$\left(J(\phi) = \right) \det \nabla \phi(\mathbf{x}, t) = f(\phi(\mathbf{x}, t), t)$$

by showing that

$$\frac{d}{dt} \left(\frac{J(\phi)}{f(\phi, t)} \right) = 0 \quad \text{if} \quad \frac{J(\phi)}{f(\phi, t)} \Big|_{t=t_0} = 1$$

In addition to adapting a mesh on a fixed domain, this method can be used to deform a mesh on a domain with moving boundary. 2D and 3D examples will be presented to demonstrate the method.