Simulating Rigid Body Fracture with Surface Meshes

(Supplemental Material)

Yufeng Zhu*
University of British Columbia

Robert Bridson†
Autodesk, University of British Columbia

Chen Greif‡
University of British Columbia

1 Boundary Element Method

1.1 Indirect Boundary Integral Formulation

The indirect boundary integral formulation is inspired by the potential theory which is usually applied to Laplace's equation (scalar or vector). Considering the similarity between Laplace and the similarly elliptic linear elastostatic problem, we apply this mathematical tool to stress analysis. This forms a crucial ingredient in our mesh-based fracture evolution algorithm. Here we briefly introduce the potential theory used for Laplace's problem and then describe how we derive our elastostatic physical model.

1.2 Potential Theory

Take a closed codimensional one embedded submanifold $\partial\Omega$, like the contour of a circle in $\mathbb{R}^2$ or a sphere in $\mathbb{R}^3$. Let $\Omega \subseteq \mathbb{R}^d$ be the subdomain enclosed by $\partial\Omega$, either interior or exterior, where d is the dimension of our space. A boundary integral over such a submanifold has the following form:

$$u(\mathbf{x}) = \oint_{\partial\Omega} \rho(\mathbf{y})\Phi(\|\mathbf{x}-\mathbf{y}\|_2)ds, \qquad \mathbf{x}\in\Omega, \mathbf{y}\in\partial\Omega. \tag{1}$$

Here ds is an infinitesimal patch of $\partial\Omega$, $\Phi(\cdot)$ is the kernel function defined in Ω, and $\rho(\cdot)$ is the density function defined on the submanifold. If we choose $\Phi(\cdot)$ to be a fundamental solution of Laplace's equation, then u will be harmonic in Ω. To determine $\rho(\cdot)$, we need to take boundary conditions into account, which in our case are of Neumann type. Given a flux function $f(\cdot)$ on $\partial\Omega$, we require

$$\frac{\partial u}{\partial \mathbf{n}}(\mathbf{y}) = f(\mathbf{y}), \qquad \mathbf{y}\in\partial\Omega. \tag{2}$$

However, to solve for $\rho(\cdot)$, we cannot simply replace the left hand side of Equation 2 with a normal derivative of the boundary integral in Equation 1. The reason is even though the boundary integral in Equation 1 is continuous throughout $\mathbb{R}^d$, its normal derivative is continuous only in Ω or $\bar{\Omega}$ - $\partial\Omega$ and has a jump condition on the submanifold. Such a jump condition can be stated as

$$\begin{aligned}\frac{\partial u}{\partial \mathbf{n}}(\mathbf{x}) &= \alpha(\mathbf{x})\rho(\mathbf{x}) + \oint_{\partial\Omega}\rho(\mathbf{y})\frac{\partial\Phi}{\partial\mathbf{n}}(\|\mathbf{x}-\mathbf{y}\|_2)ds\\ \alpha(\mathbf{x}) &= \mathrm{sgn}(\partial\Omega)\frac{\omega(\mathbf{x})\Gamma(\frac{d}{2})}{2\pi^{\frac{d}{2}}}, \qquad \mathbf{x},\mathbf{y}\in\partial\Omega,\end{aligned} \tag{3}$$

where $\omega(\cdot)$ is the solid angle, $\Gamma(\cdot)$ is the Gamma function. $\alpha(\mathbf{x})$ is $\frac{\omega(\mathbf{x})}{2\pi}$ or $\frac{\omega(\mathbf{x})}{4\pi}$ when d is 2 or 3 respectively. The function $\mathrm{sgn}(\cdot)$ is a sign function, which equals $+1$ if Ω represents an interior domain and -1 otherwise. Notice the normal $\mathbf{n}$ should be consistently pointing out of the subdomain. By combining Equation 2 and Equation 3, we can solve for $\rho(\cdot)$. Once $\rho(\cdot)$ is computed, we can evaluate u and ∇u at any point in the space using Equation 1 or applying the gradient operator to it.

*e-mail:mike323@cs.ubc.ca
†e-mail:rbridson@cs.ubc.ca
‡e-mail:greif@cs.ubc.ca

What is described above is called a *single layer potential* and only handles Laplace's problem, while we are more interested in a more general Poisson problem including the constant body force, due to gravity in our case. Now we introduce a so-called Newton potential which deals with the inhomogeneous term in Poisson's equation. Let's take a look at Equation 1 again. If we apply Laplace's operator to both sides, we will get zero on both sides. Add a domain integral on the right hand side, like

$$u(\mathbf{x}) = \oint_{\partial\Omega}\rho(\mathbf{y})\Phi(\|\mathbf{x}-\mathbf{y}\|_2)ds + g\int_\Omega \Phi(\|\mathbf{x}-\mathbf{z}\|_2)dv \qquad \mathbf{x},\mathbf{z}\in\Omega,\ \ \mathbf{y}\in\partial\Omega, \tag{4}$$

where g is the constant inhomogeneous term. It is easy to verify that after applying Laplace's operator to both sides of Equation 4, g will remain on the right hand side:

$$\begin{aligned}\nabla^2_{\mathbf{x}}u(\mathbf{x}) &= \oint_{\partial\Omega}\rho(\mathbf{y})\nabla^2_{\mathbf{x}}\Phi(\|\mathbf{x}-\mathbf{y}\|_2)ds + g\int_\Omega \nabla^2_{\mathbf{x}}\Phi(\|\mathbf{x}-\mathbf{z}\|_2)dv\\ &= \oint_{\partial\Omega}\rho(\mathbf{y})\delta(\|\mathbf{x}-\mathbf{y}\|_2)ds + g\int_\Omega \delta(\|\mathbf{x}-\mathbf{z}\|_2)dv\\ &= 0 + g = g.\end{aligned}$$

Here $\delta(\cdot)$ is the Dirac delta function (or distribution). The extra volume integral is named the *Newton potential* in Equation 4. To finish the story, we need to take a further step by transforming the volume integral into a boundary integral, by exploiting a higher order fundamental solution Ψ of Laplace's equation,

$$\nabla^2\Psi(r) = \Phi(r).$$

Then the Newton potential can be transformed into a boundary integral:

$$\begin{aligned}g\int_\Omega \Phi(\|\mathbf{x}-\mathbf{z}\|_2)dv &= g\int_\Omega \nabla^2\Psi(\|\mathbf{x}-\mathbf{z}\|_2)dv\\ &= g\oint_{\partial\Omega}\langle\nabla\Psi(\|\mathbf{x}-\mathbf{y}\|_2),\mathbf{n}\rangle ds.\end{aligned} \tag{5}$$

Therefore the final indirect boundary integral formulation can be written as

$$u(\mathbf{x}) = \oint_{\partial\Omega}\rho(\mathbf{y})\Phi(\|\mathbf{x}-\mathbf{y}\|_2)ds + g\oint_{\partial\Omega}\langle\nabla\Psi(\|\mathbf{x}-\mathbf{y}\|_2),\mathbf{n}\rangle ds \qquad \mathbf{x}\in\Omega,\ \ \mathbf{y}\in\partial\Omega.$$

1.3 Elastostatic Model

In our case, we apply the same layer potential idea to the elastostatic problem, representing the displacement field as

$$\begin{aligned}\mathbf{u}(\mathbf{x}) = &\oint_{\partial\Omega}\boldsymbol{\Phi}(\|\mathbf{x}-\mathbf{y}\|_2)\boldsymbol{\rho}(\mathbf{y})ds\\ &+\oint_{\partial\Omega}\boldsymbol{\aleph}(\|\mathbf{x}-\mathbf{y}\|_2)\otimes_2\mathbf{g}\otimes_3\mathbf{n}ds,\end{aligned} \tag{6}$$

where $\mathbf{\Phi}(\cdot)$ is a fundamental solution of the *Navier-Cauchy* equation and $\otimes_n$ is conventional n-mode tensor vector multiplication. The above equation can be viewed as an analogy of Equation 5 in the elastostatic case, with $\aleph(\cdot)$ a third order tensor which corresponds to $\nabla\Psi(\cdot)$. To compute $\boldsymbol{\rho}(\cdot)$, we make use of the jump condition again:

$$\begin{aligned}\frac{\partial \mathbf{u}}{\partial \mathbf{n_x}}(\mathbf{x}) &= \alpha(\mathbf{x})\boldsymbol{\rho}(\mathbf{x}) + \frac{\partial}{\partial \mathbf{n_x}} \oint_{\partial\Omega} \mathbf{\Phi}(\|\mathbf{x}-\mathbf{y}\|_2)\boldsymbol{\rho}(\mathbf{y}) ds \\ &+ \frac{\partial}{\partial \mathbf{n_x}} \oint_{\partial\Omega} \aleph(\|\mathbf{x}-\mathbf{y}\|_2) \otimes_2 \mathbf{g} \otimes_3 \mathbf{n} ds, \qquad \mathbf{x},\mathbf{y} \in \partial\Omega \\ &= \alpha(\mathbf{x})\boldsymbol{\rho}(\mathbf{x}) + \oint_{\partial\Omega} \nabla_{\mathbf{x}} \mathbf{\Phi}(\|\mathbf{x}-\mathbf{y}\|_2)\boldsymbol{\rho}(\mathbf{y}) \cdot \mathbf{n_x} ds \\ &+ \oint_{\partial\Omega} \frac{\partial \aleph(\|\mathbf{x}-\mathbf{y}\|_2)}{\partial \mathbf{n_x}} \otimes_2 \mathbf{g} \otimes_3 \mathbf{n} ds.\end{aligned}$$

The boundary condition we have is an external force $\mathbf{f}(\cdot)$ applied on the boundary surface, which satisfies the following condition,

$$\mathbf{f}(\mathbf{x}) = \boldsymbol{\sigma}(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x}), \qquad \mathbf{x} \in \partial\Omega.$$

Considering the following relations,

$$\begin{aligned}\boldsymbol{\sigma}(\mathbf{x}) &= \mathcal{C} : \boldsymbol{\varepsilon}(\mathbf{x}), \\ \boldsymbol{\varepsilon}(\mathbf{x}) &= \frac{1}{2}(\mathbf{F}(\mathbf{x}) + \mathbf{F}^T(\mathbf{x})) - \mathbf{I}, \\ \mathbf{F}(\mathbf{x}) &= \nabla\mathbf{u}(\mathbf{x}) + \mathbf{I}, \\ \frac{\partial \mathbf{u}}{\partial \mathbf{n_x}} &= \nabla\mathbf{u} \cdot \mathbf{n_x},\end{aligned} \tag{7}$$

we propose the new formula by combining the jump condition and the physical stationary property,

$$\begin{aligned}\mathbf{f}(\mathbf{x}) = \alpha(\mathbf{x})\boldsymbol{\rho}(\mathbf{x}) &+ \oint_{\partial\Omega} \frac{\partial \aleph(\|\mathbf{x}-\mathbf{y}\|_2)}{\partial \mathbf{n_x}} \otimes_2 \mathbf{g} \otimes_3 \mathbf{n} ds \\ &+ \frac{1}{2}\oint_{\partial\Omega} \mathcal{C} : [\nabla_{\mathbf{x}} \mathbf{\Phi}(\|\mathbf{x}-\mathbf{y}\|_2)\boldsymbol{\rho}(\mathbf{y}) \\ &+ (\nabla_{\mathbf{x}} \mathbf{\Phi}(\|\mathbf{x}-\mathbf{y}\|_2)\boldsymbol{\rho}(\mathbf{y}))^T] \cdot \mathbf{n_x} ds.\end{aligned} \tag{8}$$

1.4 Discretization

In order to solve for the layer potential, we discretize Equation 8 using piecewise constant shape functions and adopt a collocation scheme at triangle centroids, which leads to a dense linear system,

$$(\mathbf{D} + \mathbf{A})\rho = \mathbf{f} - \tilde{\mathbf{g}}. \tag{9}$$

In the above equation, $\mathbf{D}$ is a diagonal matrix whose nonzero entries correspond to the jump condition, $\alpha(\mathbf{x})\boldsymbol{\rho}(\mathbf{x})$, and $\mathbf{f}$ collects all the discretized boundary tractions. $\mathbf{A}$ is a dense matrix corresponding to the second boundary integral on the right-hand side of Equation 8, while $\tilde{\mathbf{g}}$ corresponds to the first integral (see paper's Section 4.3). It is trivial to assemble $\mathbf{D}$ and $\mathbf{f}$, and we refer interested readers to [Meßner 2008] for detailed explanation of assembling $\tilde{\mathbf{g}}$. In the sequel, we only provide the implementation details for $\mathbf{A}$.

The matrix $\mathbf{A}$ is composed of k^2 3×3 matrix block $\mathbf{A}_{ij}$, where k is the number of triangles in the boundary mesh. Each block $\mathbf{A}_{ij}$ represents the interaction between triangles $\mathcal{T}_i$ and $\mathcal{T}_j$. Given the centroid $\mathbf{p}$ and surface normal $\mathbf{n}$ of $\mathcal{T}_i$, centroid $\mathbf{q}$ and surface area w of $\mathcal{T}_j$ as well as material properties, including the Poisson ratio ν and the shear modulus G, we can compute $\mathbf{A}_{ij}$ as follows.

The fundamental solution for Navier-Cauchy equation (see paper's Section 4.2) is

$$\mathbf{\Phi}(r)_{ij} = \frac{1}{16\pi G(1-\nu)r}\left[(3-4\nu)\delta_{ij} + r_{,i}r_{,j}\right],$$

which can be rewritten as

$$\begin{aligned}\mathbf{\Phi}(r) &= \frac{1}{16\pi G(1-\nu)}\left[(3-4\nu)\bar{\mathbf{I}} + \bar{\mathbf{M}}\right] \\ \bar{\mathbf{M}} &= \frac{1}{r}\begin{pmatrix} \frac{\partial r}{\partial \mathbf{q}_x}\cdot\frac{\partial r}{\partial \mathbf{q}_x} & \frac{\partial r}{\partial \mathbf{q}_x}\cdot\frac{\partial r}{\partial \mathbf{q}_y} & \frac{\partial r}{\partial \mathbf{q}_x}\cdot\frac{\partial r}{\partial \mathbf{q}_z} \\ \frac{\partial r}{\partial \mathbf{q}_y}\cdot\frac{\partial r}{\partial \mathbf{q}_x} & \frac{\partial r}{\partial \mathbf{q}_y}\cdot\frac{\partial r}{\partial \mathbf{q}_y} & \frac{\partial r}{\partial \mathbf{q}_y}\cdot\frac{\partial r}{\partial \mathbf{q}_z} \\ \frac{\partial r}{\partial \mathbf{q}_z}\cdot\frac{\partial r}{\partial \mathbf{q}_x} & \frac{\partial r}{\partial \mathbf{q}_z}\cdot\frac{\partial r}{\partial \mathbf{q}_y} & \frac{\partial r}{\partial \mathbf{q}_z}\cdot\frac{\partial r}{\partial \mathbf{q}_z} \end{pmatrix} \\ \bar{\mathbf{I}} &= \frac{1}{r}\mathbf{I} \\ r &= \sqrt{(\mathbf{q}_x-\mathbf{p}_x)^2 + (\mathbf{q}_y-\mathbf{p}_y)^2 + (\mathbf{q}_z-\mathbf{p}_z)^2} \\ \nabla_{\mathbf{q}} r &= \begin{pmatrix} \frac{\partial r}{\partial \mathbf{q}_x} \\ \frac{\partial r}{\partial \mathbf{q}_y} \\ \frac{\partial r}{\partial \mathbf{q}_z} \end{pmatrix} = \begin{pmatrix} \frac{\mathbf{q}_x-\mathbf{p}_x}{r} \\ \frac{\mathbf{q}_y-\mathbf{p}_y}{r} \\ \frac{\mathbf{q}_z-\mathbf{p}_z}{r} \end{pmatrix} \\ \mathbf{r} &= \mathbf{q} - \mathbf{p}.\end{aligned}$$

Next, we compute $\nabla_{\mathbf{p}}\mathbf{\Phi}$. Since $\mathbf{\Phi}$ is a second order tensor, its gradient will be third order. For notational clarity, let us present it in standard form, avoiding Einstein notation. By making use of linearity, we apply the gradient operator to each column of $\mathbf{\Phi}$ (viewed as a 3×3 matrix). Then we have

$$\nabla_{\mathbf{p}}\mathbf{\Phi}(r) = \frac{1}{16\pi G(1-\nu)}\left[(3-4\nu)\nabla_{\mathbf{p}}\bar{\mathbf{I}} + \nabla_{\mathbf{p}}\bar{\mathbf{M}}\right].$$

Since the gradient is a third order tensor, it is useful to describe the process by referring to individual columns. We use the first column, denoted by $\mathbf{col}_1(\mathbf{\Phi})$, as an illustration example. We have used the symbolic package *Mathematica* to obtain the values below:

$$\begin{aligned}\nabla_{\mathbf{p}}\mathbf{col}_1(\mathbf{\Phi})(r) &= \frac{(3-4\nu)\nabla_{\mathbf{p}}\mathbf{col}_1(\bar{\mathbf{I}}) + \nabla_{\mathbf{p}}\mathbf{col}_1(\bar{\mathbf{M}})}{16\pi G(1-\nu)} \\ \nabla_{\mathbf{p}}\mathbf{col}_1(\bar{\mathbf{I}}) &= \frac{1}{r^3}\begin{pmatrix} \mathbf{r}_x & \mathbf{r}_y & \mathbf{r}_z \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\ \nabla_{\mathbf{p}}\mathbf{col}_1(\bar{\mathbf{M}}) &= \frac{1}{r^3}\begin{pmatrix} 2\mathbf{r}_x - \frac{3\mathbf{r}_x^3}{r^2} & -\frac{3\mathbf{r}_x^2\mathbf{r}_y}{r^2} & -\frac{3\mathbf{r}_x^2\mathbf{r}_z}{r^2} \\ \mathbf{r}_y - \frac{3\mathbf{r}_x^2\mathbf{r}_y}{r^2} & \mathbf{r}_x - \frac{3\mathbf{r}_y^2\mathbf{r}_x}{r^2} & -\frac{3\mathbf{r}_x\mathbf{r}_y\mathbf{r}_z}{r^2} \\ \mathbf{r}_y - \frac{3\mathbf{r}_x^2\mathbf{r}_z}{r^2} & -\frac{3\mathbf{r}_x\mathbf{r}_y\mathbf{r}_z}{r^2} & \mathbf{r}_x - \frac{3\mathbf{r}_z^2\mathbf{r}_x}{r^2} \end{pmatrix}.\end{aligned}$$

Similarly, one can compute $\nabla_{\mathbf{p}}\mathbf{col}_2(\mathbf{\Phi})$ and $\nabla_{\mathbf{p}}\mathbf{col}_3(\mathbf{\Phi})$ analytically. Here we only provide their corresponding component:

$$\begin{aligned}\nabla_{\mathbf{p}}\mathbf{col}_2(\bar{\mathbf{I}}) &= \frac{1}{r^3}\begin{pmatrix} 0 & 0 & 0 \\ \mathbf{r}_x & \mathbf{r}_y & \mathbf{r}_z \\ 0 & 0 & 0 \end{pmatrix} \\ \nabla_{\mathbf{p}}\mathbf{col}_3(\bar{\mathbf{I}}) &= \frac{1}{r^3}\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \mathbf{r}_x & \mathbf{r}_y & \mathbf{r}_z \end{pmatrix} \\ \nabla_{\mathbf{p}}\mathbf{col}_2(\bar{\mathbf{M}}) &= \frac{1}{r^3}\begin{pmatrix} \mathbf{r}_y - \frac{3\mathbf{r}_x^2\mathbf{r}_y}{r^2} & \mathbf{r}_x - \frac{3\mathbf{r}_y^2\mathbf{r}_x}{r^2} & -\frac{3\mathbf{r}_x\mathbf{r}_y\mathbf{r}_z}{r^2} \\ -\frac{3\mathbf{r}_y^2\mathbf{r}_x}{r^2} & 2\mathbf{r}_y - \frac{3\mathbf{r}_y^3}{r^2} & -\frac{3\mathbf{r}_y^2\mathbf{r}_z}{r^2} \\ -\frac{3\mathbf{r}_x\mathbf{r}_y\mathbf{r}_z}{r^2} & \mathbf{r}_z - \frac{3\mathbf{r}_y^2\mathbf{r}_z}{r^2} & \mathbf{r}_y - \frac{3\mathbf{r}_z^2\mathbf{r}_y}{r^2} \end{pmatrix} \\ \nabla_{\mathbf{p}}\mathbf{col}_3(\bar{\mathbf{M}}) &= \frac{1}{r^3}\begin{pmatrix} \mathbf{r}_z - \frac{3\mathbf{r}_x^2\mathbf{r}_z}{r^2} & -\frac{3\mathbf{r}_x\mathbf{r}_y\mathbf{r}_z}{r^2} & \mathbf{r}_x - \frac{3\mathbf{r}_z^2\mathbf{r}_x}{r^2} \\ -\frac{3\mathbf{r}_x\mathbf{r}_y\mathbf{r}_z}{r^2} & \mathbf{r}_z - \frac{3\mathbf{r}_y^2\mathbf{r}_z}{r^2} & \mathbf{r}_y - \frac{3\mathbf{r}_z^2\mathbf{r}_y}{r^2} \\ -\frac{3\mathbf{r}_z^2\mathbf{r}_x}{r^2} & -\frac{3\mathbf{r}_z^2\mathbf{r}_y}{r^2} & 2\mathbf{r}_z - \frac{3\mathbf{r}_z^3}{r^2} \end{pmatrix}\end{aligned}$$

Finally, we incorporate the constitutive law shown in Equation 7 to assemble $\mathbf{A}_{ij}$:

$$\mathbf{S}_1 = 2G\mathbf{E}_1 + \frac{2G\nu\mathbf{tr}(\mathbf{E}_1)}{1-2\nu}\mathbf{I}$$
$$\mathbf{E}_1 = \frac{1}{2}\left[\nabla_{\mathbf{p}}\mathbf{col}_1(\bar{\mathbf{M}}) + \nabla_{\mathbf{p}}\mathbf{col}_1(\bar{\mathbf{M}})^T\right]$$

After computing $\mathbf{S}_2$ and $\mathbf{S}_3$ in the same way, $\mathbf{A}_{ij}$ is constructed as

$$\mathbf{A}_{ij}(\mathbf{p},\mathbf{q},\mathbf{n},w) = w\left(\mathbf{S}_1\mathbf{n},\mathbf{S}_2\mathbf{n},\mathbf{S}_3\mathbf{n}\right). \quad (10)$$

We may now directly construct the matrix $\mathbf{A}$ by assembling all 3×3 matrix blocks, and then applying an iterative linear system solver for nonsymmetric systems, such as BiCG-Stab or GMRES. After solving for ρ, we can evaluate the displacement and stress information by incorporating Equation 6 and Equation 7.

2 Fast Multipole Method

Solving the dense system described in the last section by applying an iterative solver may be inefficient. Even though the system is well conditioned (and hence the iteration count is very small), matrix-vector products take $O(n^2)$ floating point operations. By replacing the standard matrix-vector multiplication in each iteration with a fast summation method such as fast multipole method (FMM), an asymptotic linear running time could be achieved to solve the system. We refer interested readers to [Greengard and Rokhlin 1987] for a thorough introduction of FMM. Matrix-vector multiplication, **Ax = b**, can be viewed as an *n body* problem, where **x** is a collection of particles' masses that generate a gravity potential field. **b** stores the potential value evaluated at corresponding particle's position. Entry of **A**, $\mathbf{A}_{ij}$, represents a gravity potential interaction between particle *i* and particle *j*. The idea can be generalized by replacing the gravity potential kernel function with other non-oscillatory decaying kernel functions, like the fundamental solution of Laplace's equation, the electro-static equation, the elasto-static equation as well as all the boundary integral kernels used in our paper. Instead of considering Equation 10, we stick with gravity potential point of view to give a clear presentation of the algorithm.

2.1 Algorithm Introduction

Figure 1: *n particles*

Given n massive particles as shown in Figure 1, we want to evaluate the gravity potential generated by these particles. In particular, we'd like to evaluate the gravity potential at the position of these particles. One can definitely sum up all the contributions from each particle when considering the gravity potential at each position. However, the total complexity is $O(n^2)$. A trivial way to reduce the complexity is by introducing the concept of *center of mass*. For example, given two well separated clusters of n particles, evaluating gravity potential generated by one cluster at the other cluster of particles naively will take $O(n^2)$. A more economical approach will approximate the evaluation by first computing the center of mass for each cluster, which costs $O(n)$. Next we take $O(1)$ operations to evaluate the gravity potential generated by one center of mass at the other one. Finally, we spread the computed gravity potentials from the center of mass to all its neighboring particles. The total computational cost for the proposed method is $O(n)$, but the accuracy depends on how well the clusters are separated. The above method

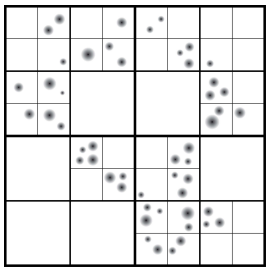


Figure 2: *quad tree*

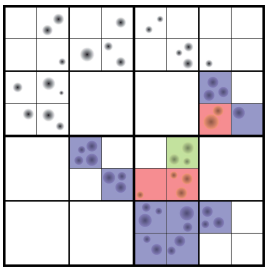


Figure 3: *well separated cell*

is used in Barnes-Hut [Barnes and Hut 1986] hierarchically and the center of mass approximation is called monopole expansion. This simple example provides an essential building block of FMM. A complete description includes also the notion of hierarchical structure, multipole expansion, and near-far field decomposition.

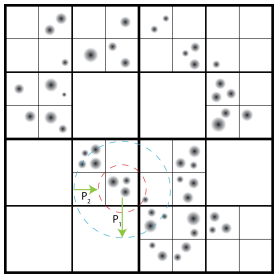


Figure 4: *particle to multipole*

The FMM algorithm starts with a spatial division tree structure, usually quad tree or octree as shown in Figure 2. At each tree level, two cells are regarded as well-separated if their distance is more than *h*, where *h* is the cell size in that level. Otherwise, two cells will be viewed as neighbors. One cell in the quad tree will have at most 8 neighbors while cell in the octree will have 26 neighbors at most. As in Figure 3, if we take the green cell as an example, cells colored in red are its neighbors. Furthermore, we introduce the idea of interaction cells colored in blue, which are well-separated from the green cell, but *not that far*. The formal definition can be stated as follows. The interaction cells $\mathcal{I}(\cdot)$ are a collection of cells within the same tree level as green cell $\mathfrak{g}$, whose parent cells $\mathfrak{P}(\cdot)$ (in the tree structure) are neighbors $\mathcal{N}(\cdot)$, of the $\mathfrak{g}$'s parent cell but they themselves are not neighbors of $\mathfrak{g}$:

$$\mathcal{I}(\mathfrak{g}) = \{\mathfrak{a} : \mathfrak{P}(\mathfrak{a}) \in \mathcal{N}(\mathfrak{P}(\mathfrak{g})) \wedge \mathfrak{a} \notin \mathcal{N}(\mathfrak{g})\}.$$

In a quad tree, one cell will have at most 27 interaction cells while 189 in an octree at most. After the spatial division tree is constructed, FMM will perform the summation in three steps:

- In the first step, we build up a far field approximation for each cell in each tree level, similar to the evaluation of the center of mass in the above toy example. However, in order to control the accuracy, FMM uses a multipole expansion which can be realized in many different ways, such as a spherical harmonic expansion, layer potential or the RBF-style approach as we propose (see paper's Section 5). The multipole expansion computation starts from the tree leaf cells up to the tree root. For each leaf cell, we apply a multipole expansion or a particle-to-multipole step, as shown in Figure 4. As described in Section 5 of the paper, we assign two spheres for each cell where we either place the kernel function or evaluate the potential value. This idea is similar to [Ying et al. 2004], who classified the spheres as a *representing sphere* or a *checking sphere*, depending on how they are used. In Figure 4, the red sphere is representing sphere and the blue sphere is checking sphere. We place *k* particles uniformly on each sphere and first evaluate the gravity potential at samples on the blue sphere generated by the 3 massive particles in the cell:

$$\boldsymbol{\phi}_i^{p2m} = \mathbf{P}_1\boldsymbol{\rho}_i,$$

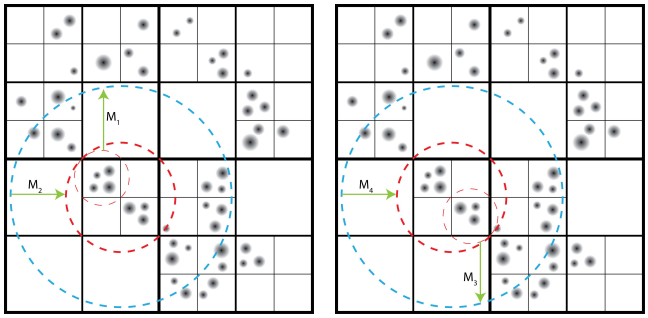


Figure 5: *multipole to multipole*

where $\boldsymbol{\rho}_i$ is the collection of particles' mass. Then we compute the multipole expansion coefficient $\boldsymbol{\sigma}$ evaluated on the red representing sphere,

$$\boldsymbol{\sigma}_i = \mathbf{P}_2^{-1}\boldsymbol{\phi}_i^{p2m}.$$

After computing the multipole expansion in the leaf cell level, we pop up the multipole expansion coefficients to their upper level; this process is called multipole-to-multipole translation, as shown in Figure 5. Similar to the particle-to-multipole step, we use the blue sphere as a checking sphere and the red sphere as a representing sphere. We first evaluate the gravity potential on the parent cell's checking sphere generated by all its children cells' representing spheres,

$$\boldsymbol{\phi}_i^{m2m} = \mathbf{M}_1\boldsymbol{\sigma}_{i_j} + \mathbf{M}_3\boldsymbol{\sigma}_{i_k}, \quad \mathfrak{P}(j) = \mathfrak{P}(k) = i.$$

Then multipole expansion coefficient of the parent cell can be computed as

$$\boldsymbol{\sigma}_i = \mathbf{M}_2^{-1}\boldsymbol{\phi}_i^{m2m}, \quad \mathbf{M}_2 = \mathbf{M}_4$$

We follow the above description and construct the multipole to multipole translation level by level.

- In the second step, we compute the influence between cells falling in the same interaction list for each tree level, which is called multipole to local expansion as shown in Figure 6. For each cell at any tree level, we aggregate the influence from all its interaction cells,

$$\boldsymbol{\phi}_i^{m2l} = \mathbf{T}_1\boldsymbol{\sigma}_j + \mathbf{T}_3\boldsymbol{\sigma}_k + \cdots.$$

Then the local expansion coefficient $\boldsymbol{\tau}$, which represents far field influence, can be computed as

$$\boldsymbol{\tau}_i = \mathbf{T}_2^{-1}\boldsymbol{\phi}_i^{m2l}, \quad \mathbf{T}_2 = \mathbf{T}_4.$$

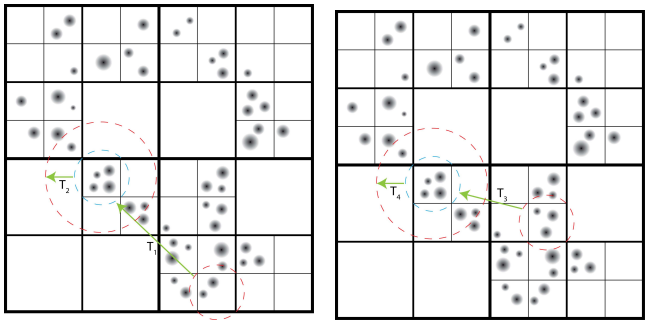


Figure 6: *multipole to local*

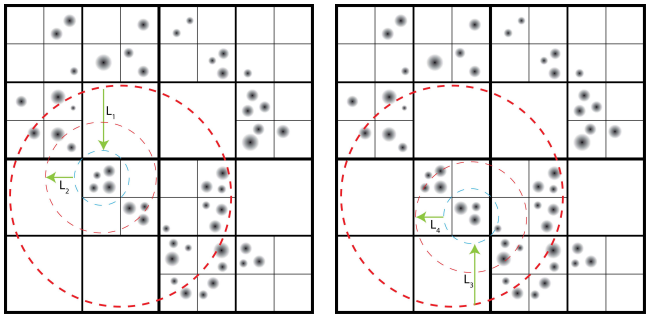


Figure 7: *local to local*

Similarly to multipole to multipole translation, we apply this step to each level.

- After the second step, we have the influence from interaction cells at each level. In the third step, we pop down such influence from the tree root to the tree leaf level by level as shown in Figure 7. Similarly, we start from evaluating the gravity potential from the parent cell's representing sphere onto its childrens' checking sphere,

$$\boldsymbol{\phi}_{i_j}^{l2l} = \mathbf{L}_1\boldsymbol{\tau}_i$$
$$\boldsymbol{\phi}_{i_k}^{l2l} = \mathbf{L}_3\boldsymbol{\tau}_i.$$

The popped down local expansion coefficients can be computed for each children cell as

$$\boldsymbol{\tau}_{i_j} = \mathbf{L}_2^{-1}\boldsymbol{\phi}_{i_j}^{l2l}$$
$$\boldsymbol{\tau}_{i_k} = \mathbf{L}_4^{-1}\boldsymbol{\phi}_{i_k}^{l2l}$$
$$\mathbf{L}_2 = \mathbf{L}_4.$$

The new coefficients will be added to the children's local expansion coefficients computed in the second step and popped down to the next level as above. Once the local expansion is popped down to the leaf level, we apply the local expansion to particles within each leaf cell as shown in Figure 8,

$$\boldsymbol{\phi}_i^{l2p} = \mathbf{Q}_1\boldsymbol{\tau}_i.$$

Notice $\boldsymbol{\phi}_i^{l2p}$ already contains influence from cells well separated from cell *i* but the influence from neighbor cells and cell *i* itself is still not accounted for, so we finally add the near field

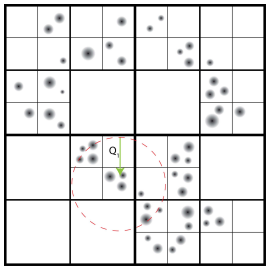


Figure 8: *local to particle*

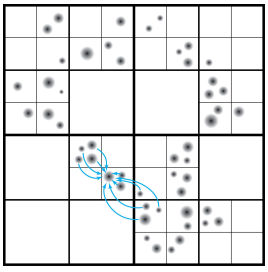


Figure 9: *near field summation*

contribution by brute froce computation for each leaf cell as shown in Figure 9.

We release an implementation of the above kernel independent FMM at `http://www.cs.ubc.ca/~mike323/fsm.html`.

3 Discussion

In this section, we provide a brief discussion on potential extension of both boundary element method and fast multipole method.

3.1 On BEM

To evolve fracture inside the material, various criteria can be applied as mentioned in Section 1 and 6.2 of the paper. In the paper, we use the *Rankine* condition which requires the principal eigenvalue of the stress tensor. Once we obtain ρ by solving Equation 9, one can evaluate stress information analytically at any point inside the material by incorporating Equations 6 and 7. The derivation is similar to the previous section, so we are not going to restate it here. Instead, we give a brief description on how the indirect BEM can be combined with another fracture criteria, *Griffith* energy minimization, which is used in [Hegemann et al. 2013]. We show how to analytically compute energy gradient which is the essential building block in shape optimization. We use the following definition for elastic energy measure,

$$\psi = \boldsymbol{\sigma} : \boldsymbol{\varepsilon}.$$

Since we use a linearized strain and linear material constitution law, the energy will be a function of deformation gradient $\mathbf{F}$, by which we mean there will be *quadratic* terms like

$$\mathbf{F} : \mathbf{F}, \qquad \mathbf{tr}(\mathbf{F})\mathbf{I} : \mathbf{F}$$

and a *linear* term $\mathbf{F} : \mathbf{I}$. Thus when we compute $\nabla\psi$, we need to evaluate

$$\nabla(\mathbf{F} : \mathbf{F}), \qquad \nabla(\mathbf{tr}(\mathbf{F})\mathbf{I} : \mathbf{F}), \qquad \nabla(\mathbf{F} : \mathbf{I})$$

Notice the second *quadratic* term can be rewritten as,

$$\mathbf{tr}(\mathbf{F})\mathbf{I} : \mathbf{F} = \mathbf{tr}(\mathbf{F})^2.$$

Its gradient turns out to be

$$\nabla(\mathbf{tr}(\mathbf{F})\mathbf{I} : \mathbf{F}) = 2\mathbf{tr}(\mathbf{F})\nabla\mathbf{tr}(\mathbf{F}) = 2\mathbf{tr}(\mathbf{F})\nabla(\mathbf{F} : \mathbf{I}).$$

As the derivation of $\nabla(\mathbf{F} : \mathbf{I})$ is similar to $\nabla(\mathbf{F} : \mathbf{F})$, we will just focus on the latter case,

$$\nabla(\mathbf{F} : \mathbf{F}) = \begin{pmatrix} \frac{\partial(\mathbf{F}:\mathbf{F})}{\partial x} \\ \frac{\partial(\mathbf{F}:\mathbf{F})}{\partial y} \\ \frac{\partial(\mathbf{F}:\mathbf{F})}{\partial z} \end{pmatrix}$$

$$\frac{\partial(\mathbf{F} : \mathbf{F})}{\partial\chi} = 2\frac{\partial\mathbf{F}}{\partial\chi} : \mathbf{F} = 2\mathbf{tr}[\frac{\partial\mathbf{F}}{\partial\chi}\mathbf{F}^T].$$

where χ represents a coordinate parameter. As we have the analytical representation of $\mathbf{F}$ based on layer potential, we can also compute $\frac{\partial\mathbf{F}}{\partial\chi}$ with little additional effort. Thus computing the energy gradient is trivial.

Interestingly, it is also possible to compute the gradient of $\mathbf{F}$'s singular values, which might be related to work on stress-aware fabrication, such as [Zhou et al. 2013]. By applying the gradient to the SVD decomposition of $\mathbf{F}$, we have

$$\mathbf{F} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$$

$$\frac{\partial\mathbf{F}}{\partial\chi} = \frac{\partial\mathbf{U}}{\partial\chi}\boldsymbol{\Sigma}\mathbf{V}^T + \mathbf{U}\frac{\partial\boldsymbol{\Sigma}}{\partial\chi}\mathbf{V}^T + \mathbf{U}\boldsymbol{\Sigma}\frac{\partial\mathbf{V}^T}{\partial\chi}$$

$$\mathbf{U}^T\frac{\partial\mathbf{F}}{\partial\chi}\mathbf{V} = \underbrace{\mathbf{U}^T\frac{\partial\mathbf{U}}{\partial\chi}}_{\hat{\omega}_\mathbf{U}}\boldsymbol{\Sigma} + \frac{\partial\boldsymbol{\Sigma}}{\partial\chi} + \boldsymbol{\Sigma}\underbrace{\frac{\partial\mathbf{V}^T}{\partial\chi}\mathbf{V}}_{\hat{\omega}_{\mathbf{V}^T}},$$

where both $\hat{\omega}_\mathbf{U}$ and $\hat{\omega}_{\mathbf{V}^T}$ are skew-symmetric matrices. Since diagonal entries of matrix product between skew-symmetric matrix and diagonal matrix will be zero, we have

$$\frac{\partial\sigma_i}{\partial\chi} = \mathbf{u}_i^T\frac{\partial\mathbf{F}}{\partial\chi}\mathbf{v}_i.$$

3.2 On FMM

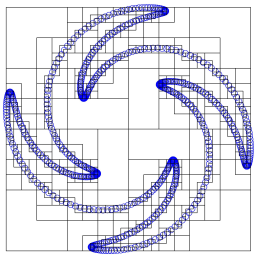


Figure 10: *particles with depth-6 quad tree*

As described in the previous section, FMM provides an asymptotically linear complexity approach for matrix-vector multiplication, **Ax = b**, based on *n body* fast summation, especially when the summation kernel is non-oscillatory and decaying. Below we provide a brief description of FMM from an algebraic point of view, which relies on matrix low rank approximations.

Given *n* particles as well as spatial division tree structure as shown in Figure 10, we can reorder the particles based on *Morton order* such that particles that are close in the tree structure will stay close in the algebraic order. We adopt the definition of neighbor cell, interaction cell and well-separated cell shown in Figure 3. In particular, we regard interaction between particles lying in neighbor cells as *full rank* and those lying in well-separated cells as low rank. We show the sparse full rank (colored in black) patterns of Figure 10 for tree levels from *2* to *6* in Figure 12. The low rank setting is

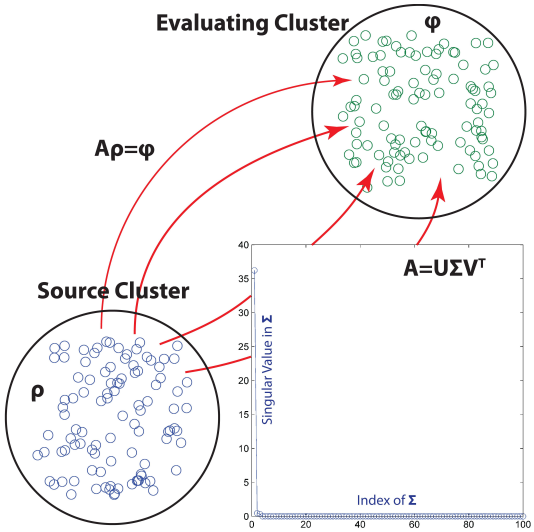
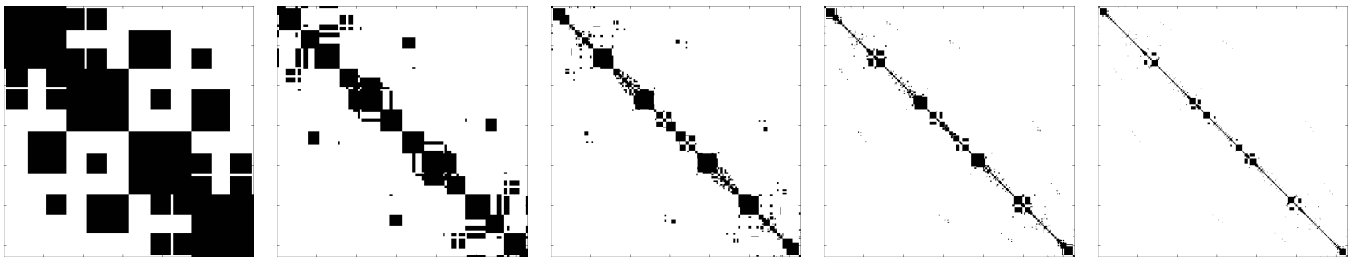


Figure 11: *rank deficient interaction between well-separated clusters*



(a) *level 2* **(b)** *level 3* **(c)** *level 4* **(d)** *level 5* **(e)** *level 6*

Figure 12: *full rank pattern (colored in black) of Figure 10 from tree level 2 to 6*

due to the usage of non-oscillatory decaying kernel function. If we take two clusters of particles and build their interaction matrix using such kind of kernel function, we will see such low rank property by plotting the matrix's singular value distribution as shown in Figure 11. If a matrix is rank deficient, the best known rank k approximation is by truncated SVD. Such approaches minimize the error between the original matrix and the approximation in either L_2 or $Frobenius$ norm. However, obtaining such optimal approximation not only requires $O(n^2)$ matrix construction but also the time consuming SVD decomposition. If we take a look at how FMM works again, the three step algorithm actually builds up a hierarchical low rank approximation tree structure for well-separated particles. Such an algebraic tree structure is similar to hierarchical semi-separable (HSS) or hierarchical block separable (HBS) matrix tree [Corona et al. 2015]. Nevertheless, we view neighbor cell interactions as full rank and want to handle them directly instead of making approximations as HSS or HBS approaches.

To see how FMM can be related to such low rank approximation, let's take two leaf cells of particles as an example. Given the mass $\boldsymbol{\rho}_s$ of one cell $\mathcal{C}_s$ and its interaction matrix $\mathbf{A}$ with cell $\mathcal{C}_e$, we can evaluate the potential $\boldsymbol{\phi}_e$ at particles in $\mathcal{C}_e$ by matrix-vector multiplication, $\mathbf{A}\boldsymbol{\rho}_s = \boldsymbol{\phi}_e$. If these two cells are well-separated, the interaction can be instead represented as follows based on what we have described in Section 2.1:

$$\boldsymbol{\sigma}_s = \mathbf{P}_2^{-1}\boldsymbol{\phi}_s^{p2m} = \mathbf{P}_2^{-1}\mathbf{P}_1\boldsymbol{\rho}_s$$
(*particle to multipole*)
$$\Downarrow$$
$$\boldsymbol{\sigma}_{\mathfrak{p}(s)} = \mathbf{M}_2^{-1}\boldsymbol{\phi}_{\mathfrak{p}(s)}^{m2m} = \mathbf{M}_2^{-1}\mathbf{M}_1\boldsymbol{\sigma}_s$$
(*multipole to multipole*)
$$\Downarrow$$
$$\vdots$$
$$\Downarrow$$
$$\boldsymbol{\tau}_{\mathfrak{p}\circ\cdots\circ\mathfrak{p}(e)} = \mathbf{T}_2^{-1}\boldsymbol{\phi}_{\mathfrak{p}\circ\cdots\circ\mathfrak{p}(e)}^{m2l} = \mathbf{T}_2^{-1}\mathbf{T}_1\boldsymbol{\sigma}_{\mathfrak{p}\circ\cdots\circ\mathfrak{p}(s)}$$
(*multipole to local*)
$$\Downarrow$$
$$\vdots$$
$$\Downarrow$$
$$\boldsymbol{\tau}_e = \mathbf{L}_2^{-1}\boldsymbol{\phi}_e^{l2l} = \mathbf{L}_2^{-1}\mathbf{L}_1\boldsymbol{\tau}_{\mathfrak{p}(e)}$$
(*local to local*)
$$\Downarrow$$
$$\boldsymbol{\phi}_e^{l2p} = \mathbf{Q}_1\boldsymbol{\tau}_e$$
(*local to particle*).

If we collect the above expansions together, we have the following relation,

$$\tilde{\mathbf{A}} = \underbrace{\mathbf{Q}_1\mathbf{L}_2^{-1}}_{\mathbf{L}_1^*}\underbrace{\mathbf{L}_1}_{\mathbf{L}_2^*}\cdot\cdots\cdot\underbrace{\mathbf{T}_2^{-1}}_{\mathbf{L}_k^*}\underbrace{\mathbf{T}_1}_{\mathbf{T}_1^*}\cdots\underbrace{\mathbf{M}_2^{-1}\mathbf{M}_1}_{\mathbf{M}_2^*}\underbrace{\mathbf{P}_2^{-1}\mathbf{P}_1}_{\mathbf{M}_1^*} \quad (11)$$
$$\boldsymbol{\phi}_e \approx \boldsymbol{\phi}_e^{l2p} = \tilde{\mathbf{A}}\boldsymbol{\rho}_s, \qquad \mathbf{A} \approx \tilde{\mathbf{A}},$$

where k is the depth of the tree structure and $\tilde{\mathbf{A}}$ is a low rank approximation as shown in Figure 13. If we extend this simple two

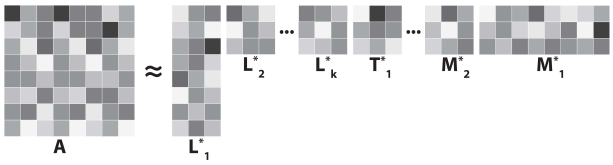


Figure 13: *low rank approximation of* $\mathbf{A}$ *represented in FMM*

well-separated clusters interaction example to general n body problem where full rank interaction and hierarchical structure need to be taken care of, we obtain a hierarchical matrix algebraic expansion similar to the HSS or HBS structure as shown in Figure 14.

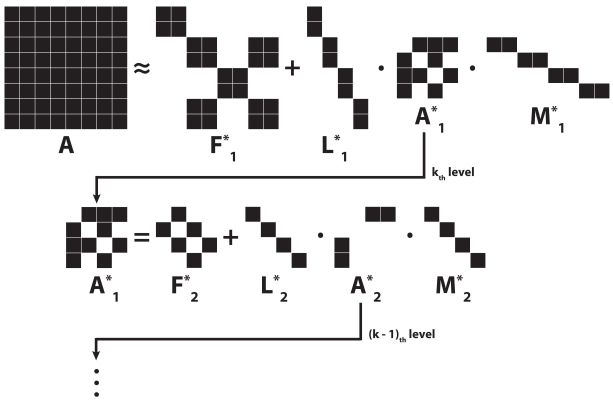


Figure 14: *algebraic expansion of FMM*

In the above figure, $\mathbf{F}_i^*$ stands for full rank block matrix which is composed of i_{th} level neighbor cells interactions. More specifically, $\mathbf{F}_1^*$'s matrix blocks represent brute force near field interaction in the bottom level of FMM tree while $\mathbf{F}_{i+1}^*$'s (i $\geq$ 1) matrix blocks are $(k-i+1)_{th}$ level $\mathbf{T}_1^*$ matrix as shown in Equation 11. By viewing FMM in this way, we can also observe linear time matrix-vector multiplication from an algebraic aspect.

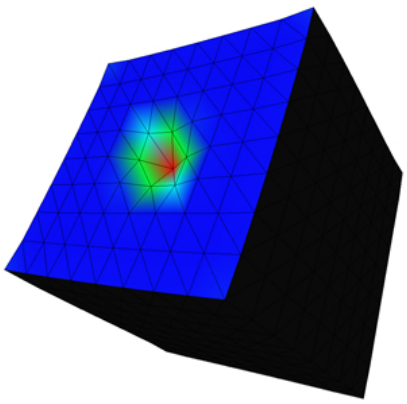
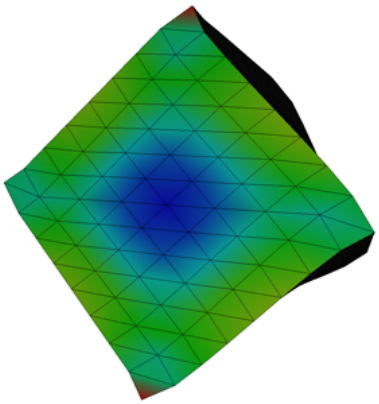
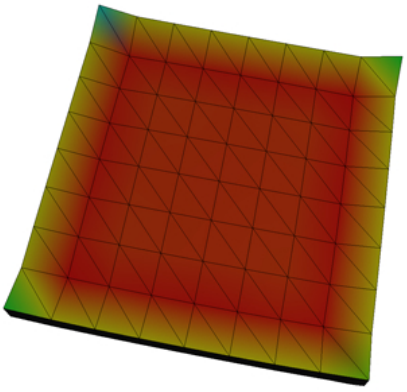
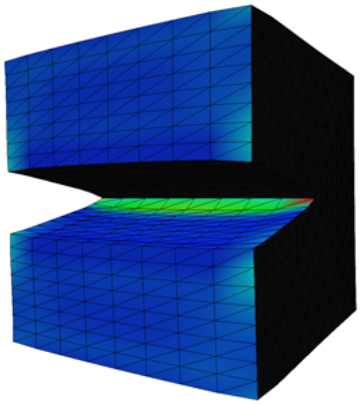
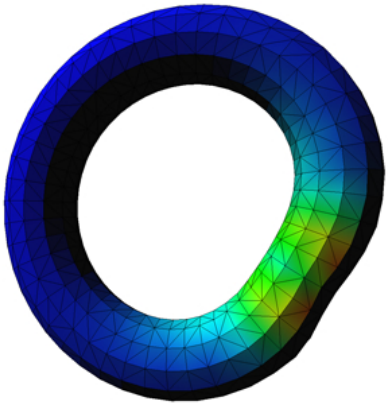
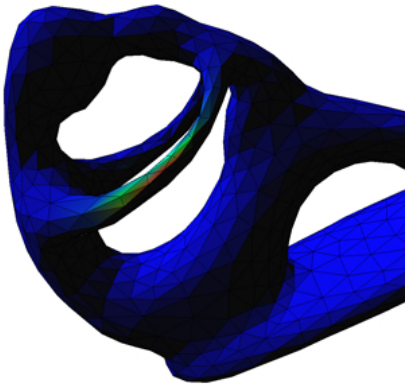
punch cube	twist cube	compress cube
		
stretch cut	**punch torus**	**push model**
		

Table 1: *Elastic deformation deformation solved by indirect boundary integral formulation. Color on the surface represents elastic energy.*

4 Numerical Results

Here we provide some numerical results computed using the above methods. In Table 1, we apply various surface tractions on different surface models and solve for displacement and elastic energy on the surface only, whereas in Table 2, we solve for displacement on the surface as well as elastic energy inside the material.

References

BARNES, J., AND HUT, P. 1986. A hierarchical $o(nlogn)$ force-calculation algorithm. *Nature 324*, 446–449.

CORONA, E., MARTINSSON, P.-G., AND ZORIN, D. 2015. An direct solver for integral equations on the plane. *Applied and Computational Harmonic Analysis 38*, 284 – 317.

GREENGARD, L., AND ROKHLIN, V. 1987. A fast algorithm for particle simulations. *J. Comput. Phys. 73*, 2 (Dec.), 325–348.

HEGEMANN, J., JIANG, C., SCHROEDER, C., AND TERAN, J. M. 2013. A level set method for ductile fracture. In *Proc. ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*, 193–201.

MESSNER, M. 2008. *Time-dependent body forces within a boundary element formulation*. PhD thesis, Technische Universität Graz.

YING, L., BIROS, G., AND ZORIN, D. 2004. A kernel-independent adaptive fast multipole algorithm in two and three dimensions. *J. Comput. Phys. 196*, 2 (May), 591–626.

ZHOU, Q., PANETTA, J., AND ZORIN, D. 2013. Worst-case structural analysis. *ACM Trans. Graph.*, 137:1–137:12.

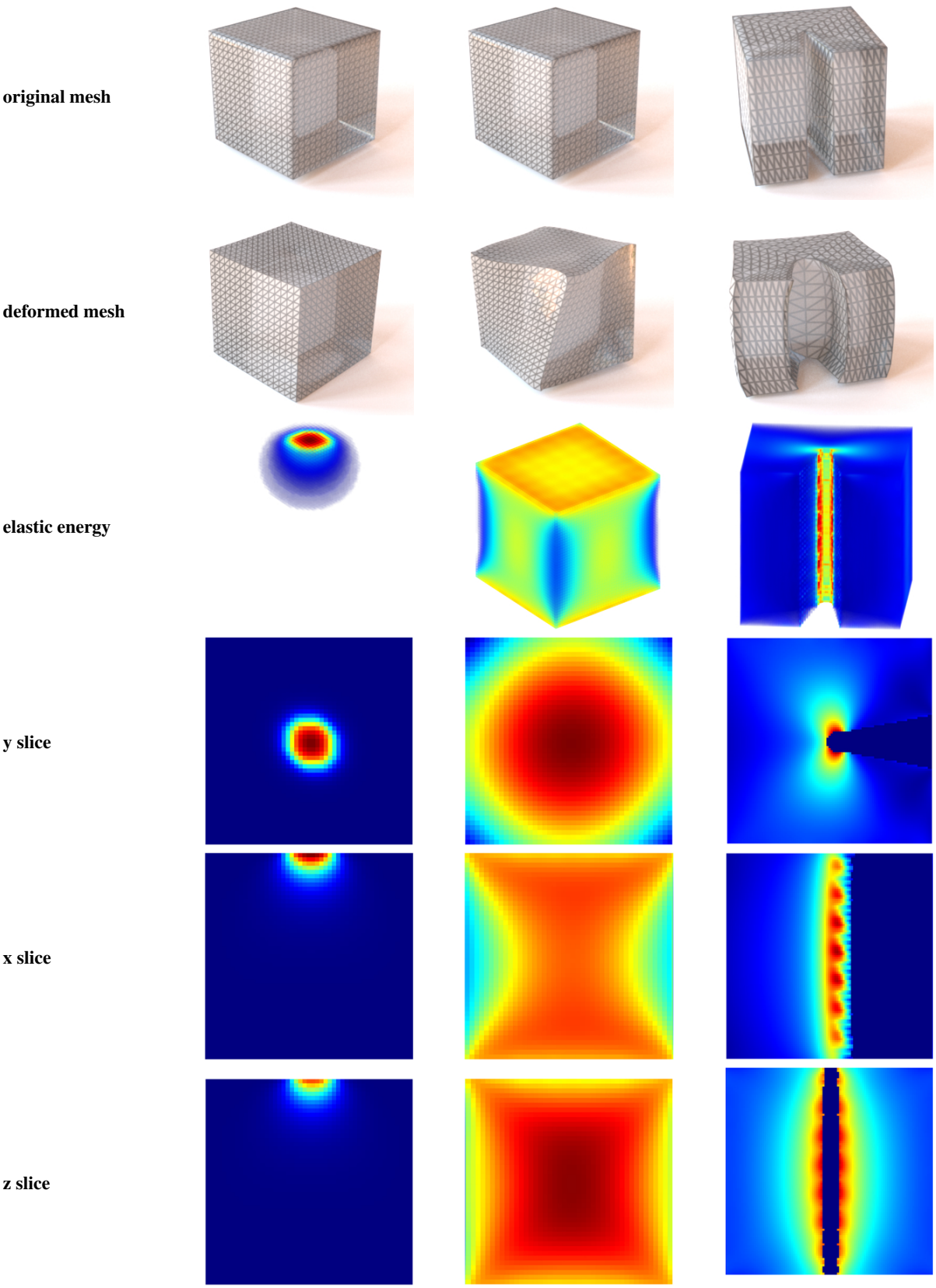


Table 2: *Elastic deformation solved by indirect boundary integral formulation.* ***first row:*** *applying point force load on the top face;* ***second row:*** *twisting the cube;* ***third row:*** *stretching the object.*