MiNNIE: a **Mi**xed Multigrid Method for Real-Time Simulation of **N**onlinear **N**ear-**I**ncompressible **E**lastics (Supplemental Material)

LIANGWANG RUAN, School of Computer Science, Peking University, China
BIN WANG, State Key Laboratory of General Artificial Intelligence, BIGAI, China
TIANTIAN LIU*, Taichi Graphics, China
BAOQUAN CHEN*, State Key Laboratory of General Artificial Intelligence, Peking University, China

A MIXED FEM

From the paper, elastodynamic simulations can be equated as the following saddle point problem:

$$\begin{aligned}(\boldsymbol{x}^{n+1}, p^{n+1}) &= \arg\min_{\boldsymbol{x}} \max_{p} \mathcal{L}(\boldsymbol{x}, p),\\ \mathcal{L}(\boldsymbol{x}, p) &= \int_\Omega [T(\boldsymbol{x}) + \Psi_d(\boldsymbol{x}) + p\Phi - \frac{1}{2\kappa}p^2] d\Omega.\end{aligned} \tag{1}$$

With P1/P1 elements, the displacement field $\boldsymbol{x}$ and pressure field p are discretized as:

$$\boldsymbol{x} = \sum_i N_i(\boldsymbol{X})\mathbf{x}_i,\ p = \sum_i N_i(\boldsymbol{X})\mathrm{p}_i, \tag{2}$$

in which $N_i(\boldsymbol{X})$ is the linear shape function of vertex i. We use italic letters like $\boldsymbol{x}$, p for continuous field, and roman letters like $\mathbf{x}_i$, p_i for discrete values on vertices. The values on vertices can be further concatenated into a long vector $\mathbf{x} \in \mathbb{R}^{3n}$, $\mathrm{p} \in \mathbb{R}^n$, where n is the number of vertices. We now derive the discrete form of each term in Eq. 1. For the $T(\boldsymbol{x})$ and $\Psi_d(\boldsymbol{x})$ terms, the treatment is the same as classic linear FEM:

$$\begin{aligned}\int_\Omega T(\boldsymbol{x}) d\Omega &= \int_\Omega \frac{\rho}{2h^2}\|\boldsymbol{x} - \boldsymbol{x}^*\|^2\\ &= \frac{1}{2h^2}\sum_{ij}(\mathbf{x}_i - \mathbf{x}_i^*)^T\left(\int_\Omega \rho N_i N_j d\Omega\right)(\mathbf{x}_j - \mathbf{x}_j^*)\\ &= \frac{1}{2h^2}(\mathbf{x} - \mathbf{x}^*)^T\mathbf{M}(\mathbf{x} - \mathbf{x}^*).\end{aligned} \tag{3}$$

The mass matrix $\mathbf{M}$ is approximated by a diagonal matrix:

$$\mathbf{M} \approx \mathrm{Diag}\{m_1, \cdots, m_n\}, m_i = \int_\Omega \rho N_i d\Omega = \sum_{e\in\mathcal{N}_i}\frac{1}{4}\rho V_e. \tag{4}$$

$\mathcal{N}_i$ is the set of neighboring elements of vertex i, V_e is the volume of element e in rest configuration. For the $\Psi_d(\boldsymbol{x})$ term, the deformation gradient $\mathbf{F}$ is constant within each element e, which gives:

$$\int_\Omega \Psi_d(\boldsymbol{x}) d\Omega = \sum_e \Psi_d(\mathbf{F}_e)V_e := U_d(\mathbf{x}). \tag{5}$$

For the mixed volume term $p\Phi - \frac{1}{2\kappa}p^2$, we have:

$$\begin{aligned}\int_\Omega \left(p\Phi - \frac{1}{2\kappa}p^2\right) d\Omega = \sum_e \Phi(\mathbf{F}_e)\int_{\Omega_e}\left(\sum_i N_i(\boldsymbol{X})\mathrm{p}_i\right)d\Omega\\ - \frac{1}{2\kappa}\sum_{ij}\int_\Omega N_i(\boldsymbol{X})N_j(\boldsymbol{X})\mathrm{p}_i\mathrm{p}_j d\Omega,\end{aligned} \tag{6}$$

where we utilize the fact $\Phi(\mathbf{F}_e)$ is constant within each element e for linear shape functions. By defining $\phi \in \mathbb{R}^n$, $\mathbf{C} \in \mathbb{R}^{n\times n}$ as follows:

$$\begin{aligned}\phi_i &= \sum_e \Phi(\mathbf{F}_e)\int_{\Omega_e} N_i(\boldsymbol{X}) d\Omega = \frac{1}{4}\sum_{e\in\mathcal{N}_i}\Phi(\mathbf{F}_e)V_e,\\ \mathbf{C}_{ij} &= \frac{1}{\kappa}\int_\Omega N_i(\boldsymbol{X})N_j(\boldsymbol{X}) d\Omega,\end{aligned} \tag{7}$$

Eq. 6 can be rewritten into a matrix form:

$$\int_\Omega p\Phi - \frac{1}{2\kappa}p^2 d\Omega = \mathrm{p}^\mathrm{T}\phi - \frac{1}{2}\mathrm{p}^\mathrm{T}\mathrm{C}\mathrm{p}. \tag{8}$$

Here ϕ is the discrete volume constraint at each vertex, $\mathbf{C}$ is the inverse stiffness matrix of volume constraint. For simplicity, we can apply the same diagonal treatment of mass matrix $\mathbf{M}$ in Eq. 4 to $\mathbf{C}$, i.e.:

$$\mathbf{C} \approx \mathrm{Diag}\{\frac{V_1}{\kappa}, \cdots, \frac{V_n}{\kappa}\}, V_i = \frac{1}{4}\sum_{e\in\mathcal{N}_i} V_e. \tag{9}$$

Combining Eq. 3, 5, 8, we get the discrete saddle point problem from Eq. 1:

$$\begin{aligned}(\mathbf{x}^{n+1}, \mathrm{p}^{n+1}) &= \arg\min_{\mathbf{x}}\max_{\mathrm{p}} L(\mathbf{x}, \mathrm{p}),\\ L(\mathbf{x}, \mathrm{p}) &= \frac{1}{2h^2}\|\mathbf{x} - \mathbf{x}^*\|_\mathbf{M}^2 + U_d(\mathbf{x}) + \mathrm{p}^\mathrm{T}\phi - \frac{1}{2}\mathrm{p}^\mathrm{T}\mathrm{C}\mathrm{p}.\end{aligned} \tag{10}$$

The solution of Eq. 10 satisfies the following discrete Karush-Kuhn-Tucker (KKT) condition:

$$\begin{cases}\mathbf{f}(\mathbf{x}, \mathrm{p}) = -\dfrac{\partial L}{\partial \mathbf{x}} = \dfrac{\mathbf{M}}{h^2}(\mathbf{x}^* - \mathbf{x}) + \mathbf{f}_d - \mathbf{G}^\mathrm{T}\mathrm{p} = 0 \in \mathbf{R}^{3n},\\ \mathrm{g}(\mathbf{x}, \mathrm{p}) = -\dfrac{\partial L}{\partial \mathrm{p}} = \mathrm{C}\mathrm{p} - \phi = 0 \in \mathbf{R}^n,\end{cases} \tag{11}$$

where $\mathbf{f}_d = -\frac{\partial U_d}{\partial \mathbf{x}} \in \mathbb{R}^{3n}$ is the distortion force, $\mathbf{G} = \frac{\partial \phi}{\partial \mathbf{x}} \in \mathbb{R}^{n\times 3n}$ is the positional gradient of volume constraint. By the definition of Eq. 7, we have:

$$\mathbf{G}_{ij} = \frac{\partial \phi_i}{\partial \mathbf{x}_j} = \frac{1}{4}\sum_{e\in\mathcal{N}_i}\frac{\partial \Phi(\mathbf{F}_e)}{\partial \mathbf{x}_j}V_e. \tag{12}$$

We use Newton-Raphson iteration to solve this KKT system. Starting from a initial guess $(\mathbf{x}^0, \mathrm{p}^0)$, at each step the following linear system

*corresponding authors

Authors' addresses: Liangwang Ruan, School of Computer Science, Peking University, Beijing, China, ruanliangwang@pku.edu.cn; Bin Wang, State Key Laboratory of General Artificial Intelligence, BIGAI, Beijing, China, binwangbuaa@gmail.com; Tiantian Liu, Taichi Graphics, Beijing, China, ltt1598@gmail.com; Baoquan Chen, State Key Laboratory of General Artificial Intelligence, Peking University, Beijing, China, baoquan@pku.edu.cn.

is solved:

$$\mathbf{Au} = \mathbf{b}, \quad \mathbf{A} = \begin{pmatrix} \mathbf{K} & \mathbf{G}^{\mathrm{T}} \\ \mathbf{G} & -\mathbf{C} \end{pmatrix}, \mathbf{u} = \begin{pmatrix} \delta\mathbf{x} \\ \delta\mathrm{p} \end{pmatrix}, \mathbf{b} = \begin{pmatrix} \mathbf{f} \\ \mathrm{g} \end{pmatrix}, \tag{13}$$

in which $\mathbf{K} = -\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \frac{\partial^2 L}{\partial \mathbf{x}^2}$.

B SMOOTHERS FOR MIXED FEM

The indefinite system we need to solve is the Eq. 5 in the paper:

$$\mathbf{Au} = \mathbf{b}, \quad \mathbf{A} = \begin{pmatrix} \mathbf{K} & \mathbf{G}^{T} \\ \mathbf{G} & -\mathbf{C} \end{pmatrix}, \mathbf{u} = \begin{pmatrix} \mathbf{x} \\ p \end{pmatrix}, \mathbf{b} = \begin{pmatrix} \mathbf{f} \\ g \end{pmatrix}, \tag{14}$$

where we ignore the k superscript to simplify notations. A general representation of a stationary iteration solver can be expressed in the following form:

$$\mathbf{u}^{m+1} \leftarrow \mathbf{u}^{m} + \hat{\mathbf{A}}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{u}^{m}), \tag{15}$$

where $\hat{\mathbf{A}}$ is an approximation of $\mathbf{A}$ that is designed for easy inversion.

B.1 Kaczmarz Smoother

The Kaczmarz smoother [Stefan 1993] addresses the indefiniteness of the system by solving the normal equation $\mathbf{A}^2\mathbf{y} = \mathbf{b}$, where $\mathbf{u} = \mathbf{A}\mathbf{y}$. Due the poor sparsity of $\mathbf{A}^2$, Jacobi iterations are welcomed in terms of parallelism, resulting in the following form:

$$\begin{aligned} \delta\mathbf{y}^{m} &= \omega \mathrm{Diag}(\mathbf{A}^2)^{-1}(\mathbf{b} - \mathbf{A}\mathbf{u}^{m}), \\ \mathbf{u}^{m+1} &\leftarrow \mathbf{u}^{m} + \mathbf{A}\delta\mathbf{y}^{m}, \end{aligned} \tag{16}$$

where ω is the relaxation parameter. Though its simplicity, the $\mathbf{A}^2$ in Kaczmarz smoother has squared condition number compared to the original system, making it slow to converge. This smoother is used in [Setaluri et al. 2015] for 2D deformations.

B.2 Inexact Uzawa Smoother

The inexact Uzawa smoother [Elman and Golub 1994] belongs to a large group of block smoothers, which use the Schur complement factorization to approximate the system matrix $\mathbf{A}$[Drzisga et al. 2018]:

$$\mathbf{A} = \begin{pmatrix} \mathbf{K} & \mathbf{G}^{\mathrm{T}} \\ \mathbf{G} & -\mathbf{C} \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{G}\mathbf{K}^{-1} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{0} & -\mathbf{S} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{K}^{-1}\mathbf{G}^{\mathrm{T}} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}, \tag{17}$$

where $\mathbf{S} = \mathbf{C} + \mathbf{G}\mathbf{K}^{-1}\mathbf{G}^{\mathrm{T}}$. The inexact Uzawa smoother only use the first two matrices to approximate $\mathbf{A}$:

$$\hat{\mathbf{A}} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{G}\hat{\mathbf{K}}^{-1} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \hat{\mathbf{K}} & \mathbf{0} \\ \mathbf{0} & -\hat{\mathbf{S}} \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{K}} & \mathbf{0} \\ \mathbf{G} & -\hat{\mathbf{S}} \end{pmatrix}, \tag{18}$$

in which $\hat{\mathbf{K}}$ and $\hat{\mathbf{S}}$ are the approximation of $\mathbf{K}$ and $\mathbf{S}$ using standard smoothers like Jacobi or Gauss-Seidel. According to Eq. 18, the inexact Uzawa smoother updates $\mathbf{x}$ and p sequentially at each step:

$$\begin{aligned} \mathbf{x}^{m+1} &\leftarrow \mathbf{x}^{m} + \omega\hat{\mathbf{K}}^{-1}(\mathbf{f}^{m} - \mathbf{K}\mathbf{x}^{m} - \mathbf{G}^{\mathrm{T}}\mathrm{p}^{m}), \\ \mathrm{p}^{m+1} &\leftarrow \mathrm{p}^{m} + \omega\hat{\mathbf{S}}^{-1}(\mathrm{g}^{m} - \mathbf{G}\mathbf{x}^{m+1} + \mathbf{C}\mathrm{p}^{m}). \end{aligned} \tag{19}$$

in which ω is the relaxation parameter. For positional DoFs, Jacobi or Gauss-Seidel iterations can be used because $\mathbf{K}$ is SPD. For pressure DoF, since $\mathbf{S}$ holds poor sparsity and $\mathbf{K}^{-1}$ in it, the most efficient way is to use Jacobi iterations for an approximated $\mathbf{S}$: $\mathbf{C}+\mathbf{G}\mathrm{Diag}(\mathbf{K})^{-1}\mathbf{G}^{\mathrm{T}}$.

The inexact Uzawa smoother in Eq. 19 might be confused with the nonlinear Uzawa method in [Frâncu et al. 2021]:

$$\begin{aligned} \mathbf{x}^{m+1} &= \arg\min_{\mathbf{x}} L(\mathbf{x}, \mathrm{p}^{m}), \\ \mathrm{p}^{m+1} &= \mathrm{p}^{m} + \omega\phi(\mathbf{x}^{k+1}). \end{aligned} \tag{20}$$

To clear the difference, the inexact Uzawa smoother only smooth the equation and serves as a part of a Newton iteration, while the nonlinear Uzawa method solves the optimization problem separately, which is no longer a Newton method. Also the inexact Uzawa smoother uses Jacobi iterations for pressure DoFs, while the nonlinear Uzawa method uses Richardson iteration.

B.3 Vanka Smoother

The Vanka-type smoother [Vanka 1986] partitions the entire simulation domain Ω into subdomains $\{\Omega_m\}$, and directly solves the small indefinite KKT system for each subdomain Ω_m successively:

$$(\mathbf{R}_m\mathbf{A}\mathbf{R}_m^{\mathrm{T}})\mathbf{u}_m = \mathbf{R}_m(\mathbf{b} - \mathbf{A}\mathbf{u}), \mathbf{u} \leftarrow \mathbf{u} + \omega\mathbf{R}_m^{\mathrm{T}}\mathbf{u}_m. \tag{21}$$

Here $\mathbf{R}_m$ is the DoF restriction operator of subdomain Ω_m, $\mathbf{A}_m = \mathbf{R}_m\mathbf{A}\mathbf{R}_m^{\mathrm{T}}$ is the restricted KKT matrix, ω is the relaxation parameter. The $\hat{\mathbf{A}}$ of the Vanka smoother has the following form:

$$\mathbf{I} - \mathbf{A}\hat{\mathbf{A}}^{-1} = \prod_{\Omega}(\mathbf{I} - \mathbf{A}\mathbf{R}_m^{\mathrm{T}}\omega\mathbf{A}_m^{-1}\mathbf{R}_m), \tag{22}$$

In previous papers[Emami 2013; John and Tobiska 2000; Larin and Reusken 2008], there are generally two type of Vanka smoothers: the cell-oriented Vanka that defines Ω_m to have all the pressure DoFs and displacement DoFs in a cell, the patch-based Vanka that defines Ω_m to have one pressure DoF and all the displacement DoFs connected to it.

For the cell-oriented Vanka solves a $\mathbf{A}_m \in \mathbb{R}^{16x16}$ for each element, where the MINRES [Paige and Saunders 1975] iteration or direct solver can be applied. To parallelize it, we need to colorize neighboring elements to different colors, process parallelly within each color groups and sequentially between groups. As we mentioned in our paper, this might requires over 200 colors for a large mesh, in contrast to only a dozen of colors are needed to colorize vertices. For patch-based Vanka, the dimension of $\mathbf{A}_i$ relates to the number of neighboring vertices of vertex i, which might be over 20 in common meshes, thus $\mathbf{A}_i$ might has a dimension over 60. To parallelize it, we need to colorize vertices that don't share a common neighbor to different colors, i.e. to colorize the square of the original connectivity graph of vertices. Again, hundreds of colors are needed.

The additive Vanka method [Saberi et al. 2022] on the other hand is more parallel-friendly, which has a much simpler form of $\hat{\mathbf{A}}$:

$$\hat{\mathbf{A}}^{-1} = \sum_{\Omega} \mathbf{R}_m^{T}\omega\mathbf{A}_m^{-1}\mathbf{R}_m. \tag{23}$$

All the subdomains can be processed in parallel without colorization, accumulating updates to all vertices at the same time. The times of updates on one vertex equals the number of subdomains overlapping with it, which can be easily over hundred. This strongly limits the relaxation parameter ω, making additive Vanka smoother hard to tune and slow to converge.

REFERENCES

Daniel Drzisga, Lorenz John, Ulrich Rüde, Barbara Wohlmuth, and Walter Zulehner. 2018. On the Analysis of Block Smoothers for Saddle Point Problems. *SIAM J. Matrix Anal. Appl.* 39, 2 (2018), 932–960. https://doi.org/10.1137/16M1106304 arXiv:https://doi.org/10.1137/16M1106304

Howard C. Elman and Gene H. Golub. 1994. Inexact and Preconditioned Uzawa Algorithms for Saddle Point Problems. *SIAM J. Numer. Anal.* 31, 6 (1994), 1645–1661. https://doi.org/10.1137/0731085 arXiv:https://doi.org/10.1137/0731085

M Emami. 2013. Efficient Multigrid Solvers for the Stokes Equations using FiniteElements. *Lehrstuhl für Informatik* 10, Systemsimulation (2013).

Mihai Frâncu, Arni Asgeirsson, Kenny Erleben, and Mads J. L. Rønnow. 2021. Locking-Proof Tetrahedra. *ACM Trans. Graph.* 40, 2, Article 12 (apr 2021), 17 pages. https://doi.org/10.1145/3444949

Volker John and Lutz Tobiska. 2000. Numerical performance of smoothers in coupled multigrid methods for the parallel solution of the incompressible Navier–Stokes equations. *International Journal for Numerical Methods in Fluids* 33, 4 (2000), 453–473. https://doi.org/10.1002/1097-0363(20000630)33:4<453::AID-FLD15>3.0.CO;2-0

Maxim Larin and Arnold Reusken. 2008. A comparative study of efficient iterative solvers for generalized Stokes equations. *Numerical Linear Algebra with Applications* 15, 1 (2008), 13–34. https://doi.org/10.1002/nla.561 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/nla.561

C. C. Paige and M. A. Saunders. 1975. Solution of Sparse Indefinite Systems of Linear Equations. *SIAM J. Numer. Anal.* 12, 4 (1975), 617–629. https://doi.org/10.1137/0712047 arXiv:https://doi.org/10.1137/0712047

S. Saberi, G. Meschke, and A. Vogel. 2022. A restricted additive Vanka smoother for geometric multigrid. *J. Comput. Phys.* 459 (2022), 111123. https://doi.org/10.1016/j.jcp.2022.111123

Rajsekhar Setaluri, Yu Wang, Nathan Mitchell, Ladislav Kavan, and Eftychios Sifakis. 2015. Fast Grid-Based Nonlinear Elasticity for 2D Deformations. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Copenhagen, Denmark) *(SCA '14)*. Eurographics Association, Goslar, DEU, 67–76.

Kaczmarz Stefan. 1993. Approximate solution of systems of linear equations†. *Internat. J. Control* 57, 6 (1993), 1269–1271. https://doi.org/10.1080/00207179308934446 arXiv:https://doi.org/10.1080/00207179308934446

S.P Vanka. 1986. Block-implicit multigrid solution of Navier-Stokes equations in primitive variables. *J. Comput. Phys.* 65, 1 (1986), 138–158. https://doi.org/10.1016/0021-9991(86)90008-2