Math 270C Final Project: Matrix-Free Preconditioners for Symmetric Positive Definite Matrices

YUNUO CHEN, UCLA Department of Mathematics
XUAN LI, UCLA Department of Mathematics

1 INTRODUCTION

Large linear systems are often needed to be solved in large-scale optimizations. For example, in high-resolution simulations using Material Point Method (MPM), there are usually millions of material particles. If implicit optimization time integrators are adopted, a nonlinear optimization with hundreds of thousands of degrees of freedom is needed at each time step [Gast et al. 2015; Jiang et al. 2016]. For example, in the scenario shown in Figure 1, the number of dofs are about 1 million. With the Newton method, a sequence of linear systems with different matrices need to be solved until convergence. The construction of such big matrices may consume more time than solving them, so matrix-free solvers are usually used in implicit MPM simulations, where only matrix-vector products are required. The matrix-vector products can be perfectly paralleled due to the structural grids that time integrations build on. However, when the deformation is large or the material is stiff, the condition numbers will become very large, making linear systems challenging to solve. Usually diagonal preconditioners are used because the contruction of diagonal terms can be pefectly paralleled, which costs roughly the same amount of time with 1 matrix-vector product. However, diagonal preconditioners are not enough for stiff matrices. Good preconditioners that are suitable for matrix-free solvers is urgently needed to accelerate linear solvers to accelerate implicit MPM simulations.

In this project, we would like to explore existing matrix-free preconditioners for matrix-free linear solvers. **We restrict that we only have the following information about linear systems:**

- The matrices are symmetric positive definite.
- The diagonal terms are available.
- Only matrix-vector products are allowed.

We only consider symmetric positive definite matrices because in implicit MPM simulations, the matrices are all Hessians of system energies. And to ensure convergence, the local stencils of hessians are projected to nearby positive definite forms [Teran et al. 2005].

Our achievements in this project includes:

- Derived preconditioned CG and MINRES algorithms.

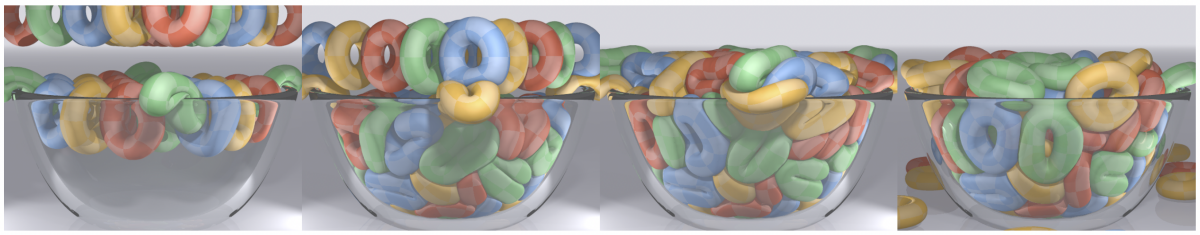

Fig. 1. One simulation example in Gast et al. [2015]. Each time step requires an large-scale optimization with about 1M dofs.

- Implemented the matrix-free conditioner in Bellavia et al. [2013] based on partial Cholesky factorizations and Schur complements.
- Experimented on matrices from stiff MPM simulations and compared with diagonal precondtioners.

2 PRECONDITIONED KRYLOV SOLVERS

In this section, we briefly review the background of the preconditioned Krylov Solvers and include two common-used krylov solvers: Conjugate Gradient (CG) and Minimum Residual (MINRES).

Let's consider the linear system $Hx = b$. For simplicity, we assume the initial guess is a zero vector. Krylov solvers try to represent the solution x^* by the Krylov space generated by H and b:

$$x^* = \sum_{k=0} w_i H^k b.$$

The Krylov spaces $\mathcal{K}^k(H, b) = \text{Span}\{b, Hb, ..., H^{k-1}b\}$ are constructed iteratively and an optimization problem on $\mathcal{K}^k(H, b)$ is solved at each iteration to approximate the true solution. However, the convergence of Krylov subspace methods highly rely on the condition number of the system matrix. In practice, preconditioners will be applied onto the system. The preconditioner matrix P can be viewed as an approximation to the original matrix H so that $P^{-1}H$ has a smaller condition number than H. The application of preconditioner on arbitrary vector v, i.e., $P^{-1}v$ should be able to compute easily. The preconditioned system is,

$$P^{-1}Hx = P^{-1}b.$$

For symmetric positive definite matrices, we also require P are symmetric positive definite to preserve the symmetry of the preconditioned system. Under this assumption, the precondition matrix has a Cholesky factorization:

$$P^{-1} = LL^T.$$

The original preconditioned system can transferred into a symmetric system with the same condition number:

$$\begin{cases} \hat{H}y = L^T HLy = L^T b = \hat{b} \\ L^{-1}x = y \end{cases} \tag{1}$$

For symmetric positive definite systems, two preconditioned Krylov Solvers are often favored: Conjugate Gradient (CG) and Minimum Residual (MINRES). These two methods only require one matrix-vector product at each iteration, along with a fixed number of vectors storing historical information.

In the next two sections, we briefly review these two Krylov methods.

2.1 Conjugate Gradient Method

Following the notation above, the projection onto Krylov spaces in CG is computed in terms of $\hat{H}$-norm. At each iteration, the following optimization is actually solved

$$y^k = \text{argmin}_{y \in \mathcal{K}^k} \|y - \hat{H}^{-1}\hat{b}\|_{\hat{H}}^2$$

where $\|v\|_{\hat{H}}^2 = v^T \hat{H} v$.

The error bound of CG can be described as [Greenbaum 1997]

$$\frac{\|y_k - y^*\|_{\hat{H}}}{\|y_0 - y^*\|_{\hat{H}}} \leq 2\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^k$$

where κ is the condition number of $P^{-1}H$. Plugging in the relation $y = L^{-1}x$, we have

$$\frac{\|x_k - x^*\|_H}{\|x_0 - x^*\|_H} \leq 2\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^k \tag{2}$$

Although the preconditioned system is written in L, the actual application of CG only requires P itself.

Let's denote $\hat{p}_k = L^{-1}p_k$ and $\hat{r}_k = L^T r_k$. Initially, we have

$$r_0 = L^{-T}\hat{b} = b,$$
$$p_0 = L\hat{b} = P^{-1}b.$$

At iteration k, we have

$$\begin{aligned}
\alpha_k &= \frac{\hat{r}_k^T \hat{r}_k}{\hat{p}_k^T \hat{H} \hat{p}_k} = \frac{r_k P^{-1} r_k}{p_k^T H p_k} \\
x_{k+1} &= L(y_k + \alpha_k \hat{p}_k) = x^k + \alpha_k p_k \\
r_{k+1} &= L^{-T}(\hat{r}_k - \alpha_k \hat{H} \hat{p}_k) = r_k - \alpha_k H p_k \\
\beta_k &= \frac{\hat{r}_{k+1}^T \hat{r}_{k+1}}{\hat{r}_k^T \hat{r}_k} = \frac{r_{k+1} P^{-1} r_{k+1}}{r_k P^{-1} r_k} \\
p_{k+1} &= L(\hat{r}_k + \beta_k \hat{p}_k) = P^{-1} r_k + \beta_k p_k
\end{aligned}$$

At each iteration, only one matrix-vector product (applied on p_k) and one application of preconditioner (applied on r_{k+1}) is required.

2.2 Minimum Residual Method

MINRES is a spectial case of GMRES applied on symmetric matrices. At each step, the following optimization problem is solved

$$y^k = \text{argmin}_{y \in \mathcal{K}^k} \|\hat{H} y - \hat{b}\|_2^2 \tag{3}$$

The error bound for MINRES has the similar form with CG [Greenbaum 1997]:

$$\frac{\|y_k - y^*\|}{\|y_0 - y^*\|} \leq 2\left(\frac{\sqrt{\kappa_{n-l}} - 1}{\sqrt{\kappa_{n-l}} + 1}\right)^{k-l}, \quad \forall\, l = 0, ..., n-1$$

where $\kappa_{n-l} = \frac{\lambda_{n-l}}{\lambda_1}$ and $\lambda_1 < \lambda_2 < \lambda_3 < ... < \lambda_n$ are eigenvalues of $P^{-1}H$ in increasing order. In terms of x, we have

$$\frac{\|x_k - x^*\|_{P^{-1}}}{\|x_0 - x^*\|_{P^{-1}}} \leq 2\left(\frac{\sqrt{\kappa_{n-l}} - 1}{\sqrt{\kappa_{n-l}} + 1}\right)^{k-l}, \quad \forall\, l = 0, ..., n-1$$

The orthogonal basis $\{\hat{q}_0 = \hat{b}, \hat{q}_1, \hat{q}_2, ..., \hat{q}_{n-1}\}$ of $\mathcal{K}^n$ are constructed by Lanczos iteration with the following three-term recursion together with the orthogonality constraint:

$$\hat{H} \hat{q}_i = t_{i+1,i} \hat{q}_{i+1} + t_{i,i} \hat{q}_i + t_{i-1,i} \hat{q}_{i-1} \tag{4}$$

Let $\hat{Q}_i = \{\hat{q}_0, \hat{q}_1, ..., \hat{q}_{i-1}\}$. We have

$$\hat{H} \hat{Q}_i = \hat{Q}_{i+1} T_i$$

where T_i is an $(i+1) \times i$ tridiagonal matrix:

$$T_i = \begin{bmatrix} t_{0,0} & t_{0,1} & & & & \\ t_{1,0} & t_{1,1} & t_{1,2} & & & \\ & t_{2,1} & t_{2,2} & & & \\ & & t_{3,2} & & & \\ & & & \cdots & & \\ & & & & t_{i-2,i-1} & \\ & & & & t_{i-1,i-1} & \\ & & & & t_{i,i-1} & \end{bmatrix}$$

The minimization Equation 3 is equivalent find $w \in \mathbb{R}^k$ such that the following problem is solved

$$\min_w \|\hat{b} - \hat{Q}_{i+1} T_i w\|^2, \quad \hat{y} = \hat{Q}_i w$$

If we set $D_i = \text{Diag}(\|\hat{q}_0\|, \|\hat{q}_1\|, ..., \|\hat{q}_i\|)$,

$$\|\hat{b} - \hat{Q}_{i+1} T_i w\| = \|\hat{b} - \hat{Q}_{i+1} D_{i+1}^{-1} D_{i+1} T_i w\|$$

If we extend $\hat{Q}_{i+1} D_{i+1}^{-1}$ to an orthogonal matrix, and then applied this transformation, we then have

$$\text{argmin}_w \|\hat{b} - \hat{Q}_{i+1} T_i w\| = \text{argmin}_w \|D_{i+1}^{-1} \hat{Q}_{i+1}^T \hat{b} - D_{i+1} T_i w\|$$

The application of MINRES also requires P only. Let $\hat{q}_i = L^T q_i$. Then the Lanczos iteration Equation 4 becomes

$$P^{-1} H P^{-1} q_i = t_{i+1,i} P^{-1} q_{i+1} + t_{i,i} P^{-1} q_i + t_{i-1,i} P^{-1} q_{i-1}.$$

Note that $\langle q_i, q_j \rangle_{P^{-1}} = \langle \hat{q}_i, \hat{q}_j \rangle = 0$.

D_i is computed as $D_i = \text{Diag}(\|q_0\|_{P^{-1}}, \|q_1\|_{P^{-1}}, ..., \|q_i\|_{P^{-1}})$. Then the optimization problem becomes

$$\min_w \|D_{i+1}^{-1} Q_{i+1}^T P^{-1} b - D_{i+1} T_i w\|$$

Furthermore, if we normalize q_i by $\|\hat{q}_i\| = \|q_i\|_{P^{-1}}$ during Lanczos iteration, D_{i+1} can be annihilated, resulting the following optimization problem:

$$\min_w \|Q_{i+1}^T P^{-1} b - T_i w\|$$

At each iteration, we can apply a Givens rotation on the last two rows to eliminate the bottom right element $t_{i,i-1}$ of T_i so as to keep it a bidiagonal matrix (correspondingly, the last two rows of Q_{i+1}^T are permanently transformed as well as each iteration). Then the least square problem becomes a linear system, which can be directly solved.

Overall, MINRES only requires one matrix-vector product and one application of the prconditioner at each iteration.

3 MATRIX-FREE PRECONDITIONERS

3.1 Diagonal Preconditioner

Diagonal preconditioner is the simplest matrix preconditioner which only requires the information of diagonal entries, i.e., we choose precondition matrix $P = diag(H)$. Although it is easy to implement, it can largely reduce the condition number of some ill-conditioned matrices.

3.2 Limited Memory Preconditioner

In contrast to merely using the diagonal information, Limited Memory Preconditioner[Bellavia et al. 2013] performs a more detailed spectral analysis of the preconditioned matrix and cleverly handles the largest eigenvalue(s) of H to improve matrix condition.

The idea of LMP is to approximate matrix H based on "partial" Cholesky factorization limited to a small number of columns and approximate the resulting Schur complement. Specifically, let n denote the size of H and given an integer k much less than n, we can rewrite H as block matrix $\begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix}$, where H_{11} is a $k \times k$ square matrix. LMP seeks to exactly reconstruct the $(1,1)$ block H_{11} and approximate the rest with diagonal values. In order to handle the largest eigenvalues of H, one crucial step is to move the k largest diagonal elements of H to the H_{11} block. To reduce computational overhead, this operation is carried out by a static permutation of rows and columns that do not actually modify the entries of H.

After proper permutation, we then extract the first k columns (H_{11} and H_{21}) by multiplying H with e_i. We then perform Cholesky Decomposition on the small dense matrix H_{11} to obtain $H_{11} = L_{11}^T D_1 L_{11}^T$. Now the original matrix can be factorized as

$$H = \begin{bmatrix} L_{11} & \\ L_{21} & I \end{bmatrix} \begin{bmatrix} D_1 & \\ & S \end{bmatrix} \begin{bmatrix} L_{11}^T & L_{21}^T \\ & I \end{bmatrix}. \tag{5}$$

Here L_{21} satisfies $L_{11} D_1 L_{21}^T = H_{21}$, which can be solved trivially by back substitution since both L_{11} and D_1 are known. $S = H_{22} - H_{21} H_{11}^{-1} H_{21}^T$ is the Schur complement of H_{11}.

We then obtain the LMP preconditioning matrix P by approximating S as its diagonal, denoted as D_2:

$$P = LDL^T = \begin{bmatrix} L_{11} & \\ L_{21} & I \end{bmatrix} \begin{bmatrix} D_1 & \\ & D_2 \end{bmatrix} \begin{bmatrix} L_{11}^T & L_{21}^T \\ & I \end{bmatrix} \tag{6}$$

Notice that neither the matrix P nor its inverse P^{-1} is actually computed and stored. We only store all the block matrices listed in the expression above. When applying the preconditioning in Krylov linear solver, one only need to apply matrix-vector multiplication $v \longrightarrow P^{-1}v$, which is carried out by block back substitution.

By the formulation of matrix P, one can see that the degenerating case $k = 0$ gives $P = D_2 = diag(H_{22})$, i.e., Diagonal Precondition, while setting k to be n will result in $P = H$.

We summarize the algorithm as following:

ALGORITHM 1: Limited Memory Preconditioner (LMP)

Require: Matrix-vector operator $u \rightarrow Hu$, $k > 0$

Step 1. Get first k column: $H_{11} \leftarrow H(0:k, 0:k)$, $H_{21} \leftarrow H(k:n, 0:k)$.
Step 2. Compute diagonal entries of H_{22}.
Step 3. Compute LDL^T factorization $H_{11} = L_{11}D_1L_{11}^T$.
Step 4. Compute $L_{21} = H_{21}L_{11}^{-T}D_1^{-1}$.
Step 5. Compute $D_2 = diag(H_{22}) - diag(L_{21}D_1L_{21}^T)$
Step 6. Assemble P as in Eq(6)

4 IMPLEMENTATION

Our code is implemented using C++. The repository is open sourced here: https://github.com/xuan-li/Matrix-Free-Preconditioner.git.

4.1 Permutation of Matrix

The application of permutation matrix is achieved in a matrix-free way as well. We use a permutation vector to encode the application of the permutation, where the ith element of the permutation vector represents its destination position. The application of the permutation and the inverse permutation can be achieved as follows:

```
// Pi * v:
for (int i = 0; i < n; ++i)
    v_out(perm(i)) = v_in(i);
```

```
// Pi^{-1} * v:
for (int i = 0; i < n; ++i)
    v_out(i) = v_in(perm(i));
```

4.2 LDL decomposition

We use Eigen [Guennebaud et al. 2010] C++ Library to do the LDL decomposition of H_{11} in the construction of LMP preconditioner. However, for numerical robustness, the LDL decomposition in Eigen[1] actually solves:

$$PAP^T = LDL^T$$

where P is an permutation matrix. So P introduces an extra permutation on the first $k \times k$ block. After we computed L_{11}, we modify the permutation vector for the largest k diagonal elements to apply this extra permutation and apply the P on the columns of previously computed H_{21} to maintain consistency.

[1]https://eigen.tuxfamily.org/dox/classEigen_1_1LDLT.html

Table 1. **Numerical Results.**

(1)"None" means without preconditioning; (2)Max iter = 10^5

Example	Parameter k	Minres Iteration	CG Iteration	Largest Eigenvalue
H_A	None	60187	Max iter reached	11139.3
H_A	0	12091	12503	51.0728
H_A	1	10290	10370	46.023
H_A	100	1546	1641	28.0157
H_B	None	Max iter reached	Max iter reached	3.11727e+08
H_B	0	6576	6694	29.4704
H_B	1	1203	1329	47.0134
H_B	100	775	1032	14.41

4.3 Krylov Solvers

Preconditioned Matrix-free version of CG and MINRES implementations are taken from the Ziran codebase[2] for MPM simulations.

4.4 Eigenvalue Solver

To test our implemented matrix-free conditioners, we use matrix-free eigenvalue solvers from Spectra[3] to compute the largest/smallest eigenvalue of the preconditioned matrices $P^{-1}H$.

5 EXPERIMENT

The matrices in our experiments are hessians from an 3D MPM simulation on an elastic cube. The optimization needs to solve at each time step is [Jiang et al. 2016]

$$\min_{\Delta v} \frac{1}{2}\|\Delta v\|_{M^n} + \Delta t^2 \sum_q \Psi((I + \nabla(v^n + v)_q)F_q^n)$$

where the first term is for inertia and the second term is for elasticity. F_q^n denotes the deformation gradient of particle q at the last time step. The condition number of the hessian matrix will increase if Young's modulus of the matrial is larger, the time step size is larger, or F_q is further from the identity matrix. We randomly sample $F_q^0 \sim I + \text{Unif}([-0.5, 0.5]^{3\times3})$ for each particle p, meaning that each particle of the cube is randomly deformed before the simulation, and take the hessian matrix at the beginning of the first time step. The matrices data we used in our experiments can be found here: https://drive.google.com/drive/folders/1OHcjV29MGlddUY2DfFa1um68hr65ssMh?usp=sharing.

[2]https://github.com/penn-graphics-research/ziran2020/tree/master/Lib/Ziran/Math/Linear
[3]https://github.com/yixuan/spectra/

We design the following experiments to demonstrate the efficacy of the matrix-free preconditioner. We test our algorithm on two randomly generated matrices with different size to showcase the impact of the matrix size on the preconditioner. One large matrix H_A (238521×238521) and one ill-conditioned small matrix H_B (6591×6591) were adopted.

We run the LMP preconditioned CG/MINRES algorithm with various choices of parameter k and record the number of iterations required to achieve a certain precision ($\epsilon = 10^{-7}$ for H_A and 10^{-4} for H_B). We record some representative values in Table 1, and plot the number of iterations for both matrices (see Fig. 2 and 3).

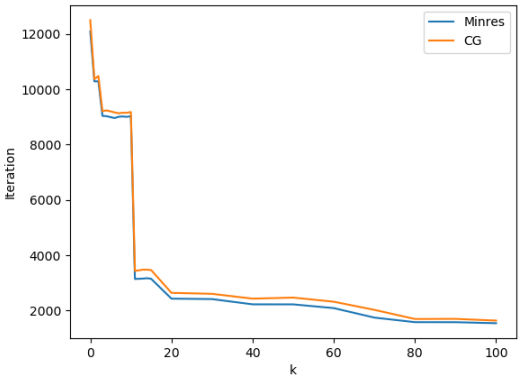

Fig. 2. Iteration numbers when different k applied to H_A

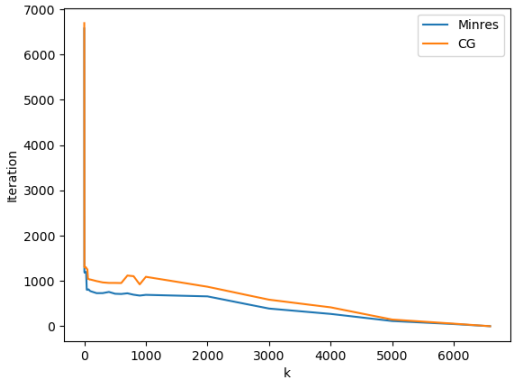

Fig. 3. Iteration numbers when different k applied to H_B

From the results above, we observe that for a large / ill-conditioned sparse linear system, only a small integer k is needed to resolve the large number of iterations required. After the

number reaches some certain level, further increasing k will not lead to significant drop in iteration numbers. Instead, increasing k will increase the burden of applying the precondition matrix (i.e., solving $P^{-1}v$ becomes more time consuming as k increases).

We further evaluate the effects of the LMP preconditioner on matrix conditions. For both linear system, we record its largest eigenvalue after preconditioning. Table1 shows that even with Diagonal preconditioning ($k = 0$), the largest eigenvalue is largely shrunken (See Fig. 4 and 5).

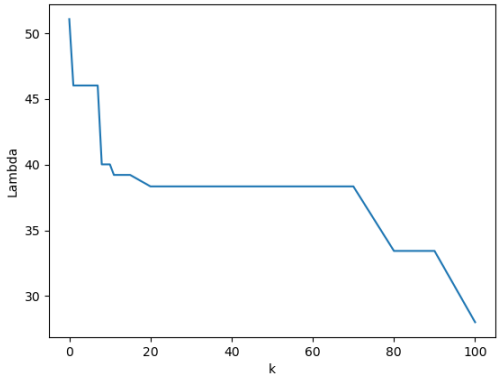

Fig. 4. Largest eigenvalue when different k applied to H_A

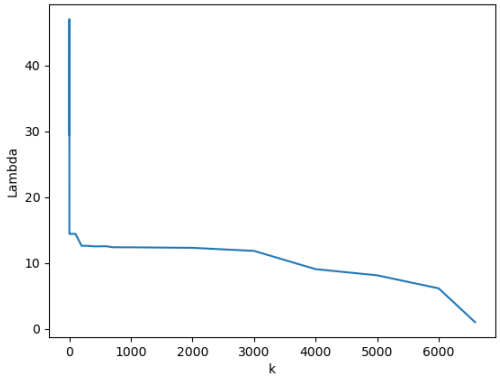

Fig. 5. Largest eigenvalue when different k applied to H_B

Combing the above figures one can see that the largest eigenvalue is a pretty good indicator of the efficacy of the LMP preconditioner. As the largest eigenvalue shrunk, the required number of iteration decrease. Since computing the largest eigenvalue is relatively easier comparing

to solving the system directly, this can be used as a method to find a suitable parameter k. In practice, one can test on different choices of k and find the largest eigenvalue after preconditioning, then choose the number k that produces an eigenvalue that is small enough. This can be very useful since sometimes increase k by 1 can significantly affect the efficiency of the method.

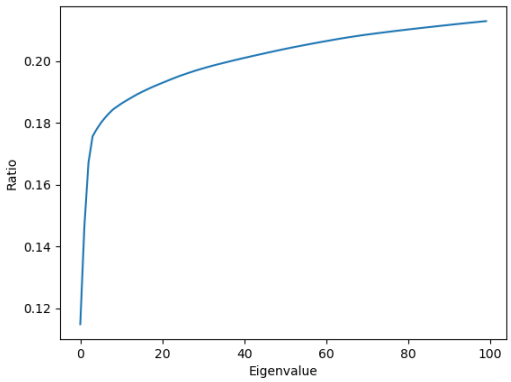

Fig. 6. Percentage of largest k eigenvalues in H_A

To further illustrate the impact of largest eigenvalues on the preconditioner, we compute the cumulative sum of the largest k eigenvalues and plot its ratio with respect to the sum of all eigenvalues (i.e., trace of H_A) (See Fig. 6). We can see that although the size of H_A is around a quarter million, the largest 20 (0.008%) eigenvalues counts for almost 20% of the total sum. This explains the sudden drop in iteration numbers when the LMP preconditioner with a parameter k less than 20 is applied.

REFERENCES

Stefania Bellavia, Jacek Gondzio, and Benedetta Morini. 2013. A matrix-free preconditioner for sparse symmetric positive definite systems and least-squares problems. *SIAM Journal on Scientific Computing* 35, 1 (2013), A192–A211.

Theodore F Gast, Craig Schroeder, Alexey Stomakhin, Chenfanfu Jiang, and Joseph M Teran. 2015. Optimization integrator for large time steps. *IEEE transactions on visualization and computer graphics* 21, 10 (2015), 1103–1115.

Anne Greenbaum. 1997. *Iterative methods for solving linear systems*. SIAM.

Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org.

Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. 2016. The material point method for simulating continuum materials. In *ACM SIGGRAPH 2016 Courses*. 1–52.

Joseph Teran, Eftychios Sifakis, Geoffrey Irving, and Ronald Fedkiw. 2005. Robust quasistatic finite elements and flesh simulation. In *Proceedings of the 2005 Symp. Computer animation*. 181–190.