$\mathcal{H}^2$-MG: A multigrid method for hierarchical rank structured matrices

Daria Sushnikova
daria.sushnikova@kaust.edu.sa

جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

George Turkiyyah, KAUST
Edmond Chow, Georgia Institute of Technology
David Keyes, KAUST

June 17, 2024

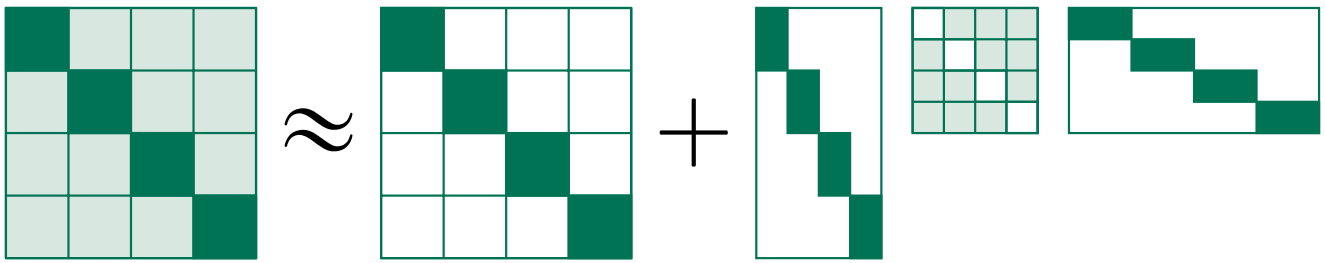
Introduction

Multigrid method
– sparse matrix
– hierarchical grids
– fast convergence

$\mathcal{H}^2$ matrix
– dense matrix approximation
– hierarchical structure
– $\mathcal{O}(N)$ storage and matrix-by-vector product

Idea: MG + $\mathcal{H}^2$ = $\mathcal{H}^2$-MG
– dense matrix
– fast convergence
– fast v-cycle run

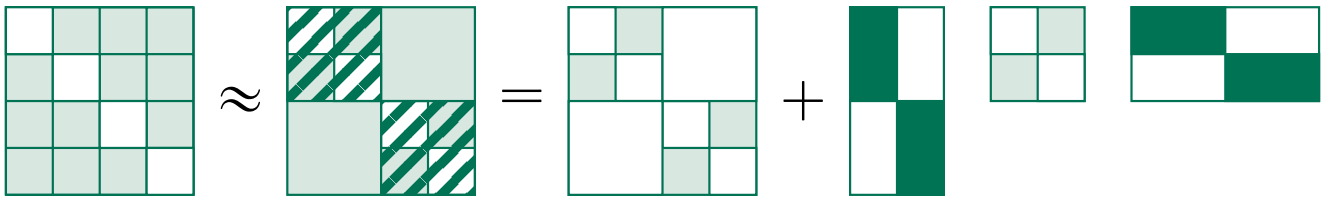
$Ax = b$, where $A \in \mathbb{R}^{N \times N}$ is dense, and $x, b \in \mathbb{R}^N$. The rows and columns of matrix A are partitioned into M blocks of size B.



Matrix D block-sparse, U_1 and V_1 are orthogonal block-diagonal matrices.

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

Next level factorization:

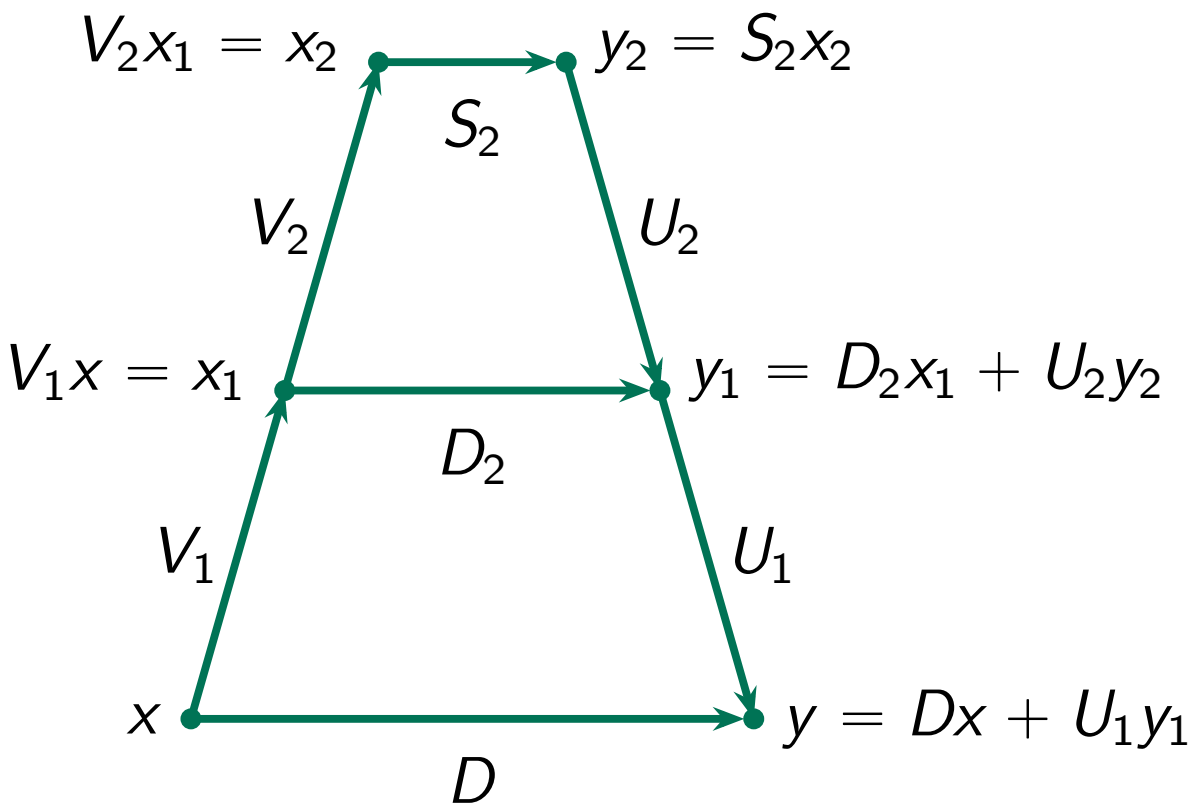


$\mathcal{H}^2$ matrix approximation:

$$A = D + U_1 \left(D_2 + U_2(\ldots(D_l + U_l S_l V_l)\ldots)V_2\right) V_1. \qquad (1)$$

$\mathcal{H}^2$ matrix-by-vector product

$$y = Ax = \left(Dx + U_1\left(D_2 + U_2(\ldots(D_l + U_l S_l V_l)\ldots)V_2\right)V_1\right)x.$$



$\mathcal{H}^2$-MG - fine grid smoothing

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

$Ax = b$, where $A \in \mathbb{R}^{N \times N}$ is an l-level $\mathcal{H}^2$ matrix, SPD, $b \in \mathbb{R}^N$ is the right-hand side, $x_0 \in \mathbb{R}^N$ is the initial guess.

$$r_1 = b - Ax_0,$$

$$Ae_1 = r_1, \quad \text{where } e_1 = x - x_0.$$

Smoothing iterations (CG in our case):

$$\tilde{e}_1 = \textbf{Iter}(A, r_1, 0).$$

$$\hat{r}_1 = r_1 - A\tilde{e}_1.$$

$$A\hat{e}_1 = \hat{r}_1, \quad \text{where } \hat{e}_1 = e_1 - \tilde{e}_1$$

$$(D + U_1 (D_2 + U_2(\dots(D_l + U_l S_l V_l)\dots)V_2)\, V_1)\, \hat{e}_1 = \hat{r}_1.$$

The orthogonal level-to-level transfer matrices U_1 and V_1 are used as interpolation and prolongation operators.

$$U_1^\top (D + U_1 (D_2 + U_2(\dots(D_l + U_l S_l V_l)\dots)V_2)\, V_1) V_1^\top V_1 \hat{e}_1 = U_1^\top \hat{r}_1,$$

$$(U_1^\top D V_1^\top + D_2 + U_2(\dots(D_l + U_l S_l V_l)\dots)V_2) V_1 \hat{e}_1 = U_1^\top \hat{r}_1.$$

$$A_2 = U_1^\top D V_1^\top + D_2 + U_2(\dots(D_l + U_l S_l V_l)\dots)V_2).$$

Note that A_2 is an $\mathcal{H}^2$ matrix. Second level soothing iteration:

$$A_2 e_2 = r_2, \quad \text{where } e_2 = V_1 \hat{e}_1, r_2 = U_1^\top \hat{r}_1.$$

The coarsest level:

$$A_l e_l = r_l,$$

where A_l is a small dense matrix. The Cholesky factorization:

$$e_l = \textbf{dir\_sol}(A_l, r_l).$$

Prolongation of the error:

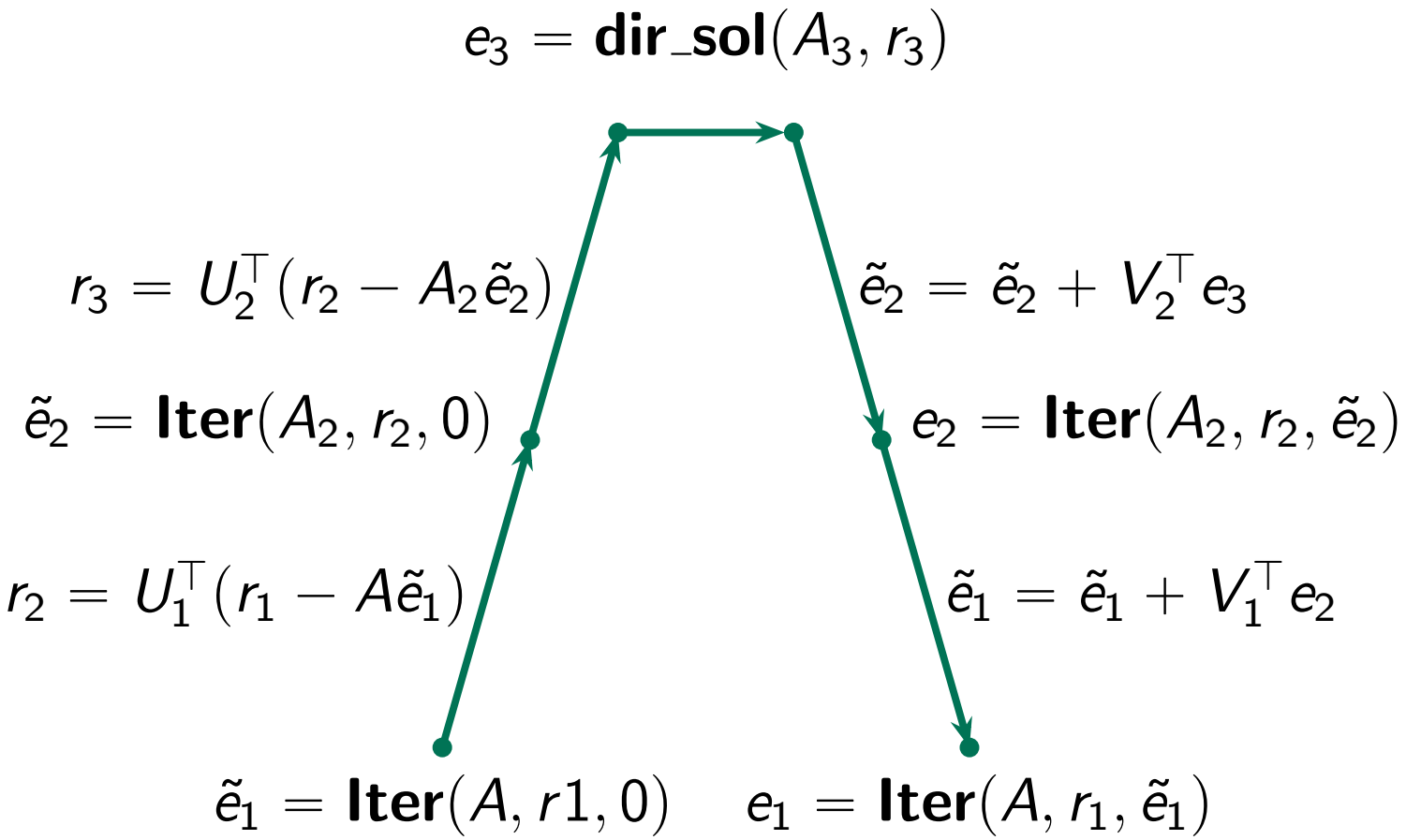
$$\tilde{e}_{l-1} = \tilde{e}_{l-1} + V_l^\top e_l.$$

Then, we apply the smoothing operator starting with initial guess $\tilde{e}_{l-1}$:

$$e_{l-1} = \textbf{Iter}(A_{l-1}, r_{l-1}, \tilde{e}l - 1).$$

We continue until we reach level 1. From the estimated error e_1, we obtain the approximation of the solution x^*:

$$x^* = x_0 + e_1.$$


$e_3 = \mathbf{dir\_sol}(A_3, r_3)$
$r_3 = U_2^\top (r_2 - A_2 \tilde{e}_2)$
$\tilde{e}_2 = \tilde{e}_2 + V_2^\top e_3$
$\tilde{e}_2 = \mathbf{Iter}(A_2, r_2, 0)$
$e_2 = \mathbf{Iter}(A_2, r_2, \tilde{e}_2)$
$r_2 = U_1^\top (r_1 - A \tilde{e}_1)$
$\tilde{e}_1 = \tilde{e}_1 + V_1^\top e_2$
$\tilde{e}_1 = \mathbf{Iter}(A, r1, 0)$
$e_1 = \mathbf{Iter}(A, r_1, \tilde{e}_1)$

$\mathcal{H}^2$-MG complexity

$\mathcal{H}^2$ matrix-by-vector product has complexity $\mathcal{O}(N)$, with constant c_{H2}.

$$n_{\text{op}} = 2n_f c_{H2} N + 2n_c \sum_{i=2}^{l-1} c_{H2} N_i + c_d N_l^3.$$

Assume $N = MB$, $N_i = \frac{Mr}{d^{i-2}}$, for $i = 2, \ldots, l$, rank r.

$$n_{\text{op}} = 2n_f c_{H2} MB + 2n_c \sum_{i=2}^{l-1} \frac{Mrc_{H2}}{d^{i-2}} + c_d N_l^3.$$

Using the sum of geometric progression, obtain:

$$n_{\text{op}} = 2n_f c_{H2} MB + 2n_c \left(\frac{d - \frac{1}{d^{l-3}}}{d-1} \right) c_{H2} Mr + c_d N_l^3.$$

By the construction of $\mathcal{H}^2$, the size of the coarsest level N_l is a constant; thus,

$$n_{\text{op}} = \mathcal{O}(N),$$

Numerical results (preliminary)

- Python implementation
- The tests were done on a MacBook Pro with Apple M1 Max Chip
- Points: uniform tensor grid on a unit square $P \subset \mathbb{R}^2$
- $\mathcal{H}^2$ matrix:
  - Number of levels: adaptive
  - Approximation accuracy: $\epsilon = 10^{-5}$
  - Block size: $B = 256$

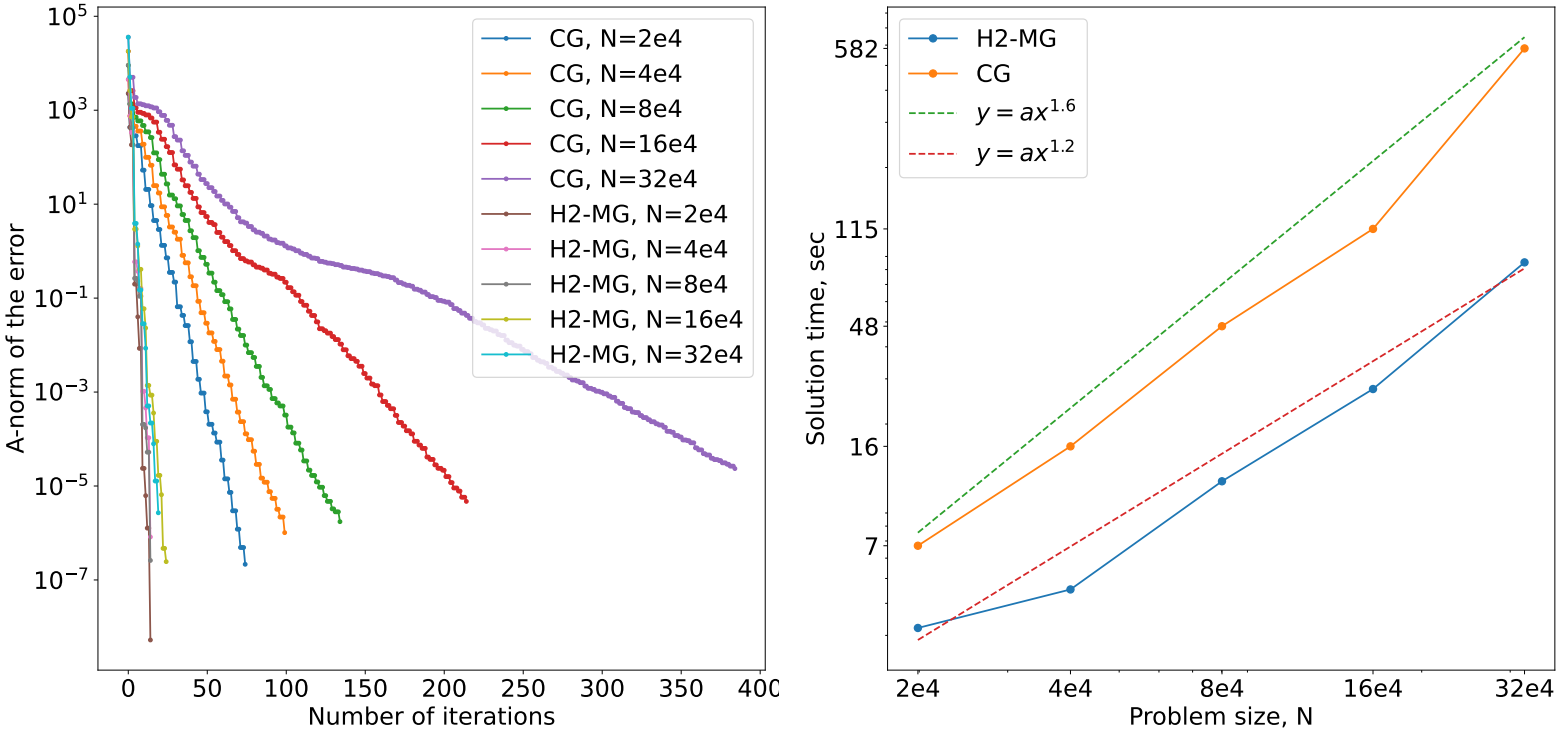
Exponential kernel

$Ax = b$, where $b \in \mathbb{R}^N$ is random, $x \in \mathbb{R}^N$ is an unknown, and $A \in \mathbb{R}^{N \times N}$ is a kernel matrix:

$$a_{ij} = \begin{cases} \exp\left(-\frac{|p_i - p_j|}{\sigma}\right), & \text{if } i \neq j \\ c, & \text{if } i = j \end{cases},$$

where points $p_i \in P$, $i \in 1 \ldots N$, $\sigma = 0.1$ is the dispersion parameter of the matrix, $c = 64$ is a constant.

Exponential kernel



(a) Convergence for various problem sizes.

(b) Solution time.

Figure 2: Convergence comparison of the methods $\mathcal{H}^2$-MG and CG for different number of coarse iterations.

Exponential kernel

Number of V-cycles required for the convergence to the residual $\varepsilon = 10^{-5}$:

N	2e4	4e4	8e4	16e4	32e4
Number V-cycles	3	3	3	5	4

Table 1: Number of V-cycles to solve the system with exponential matrix.

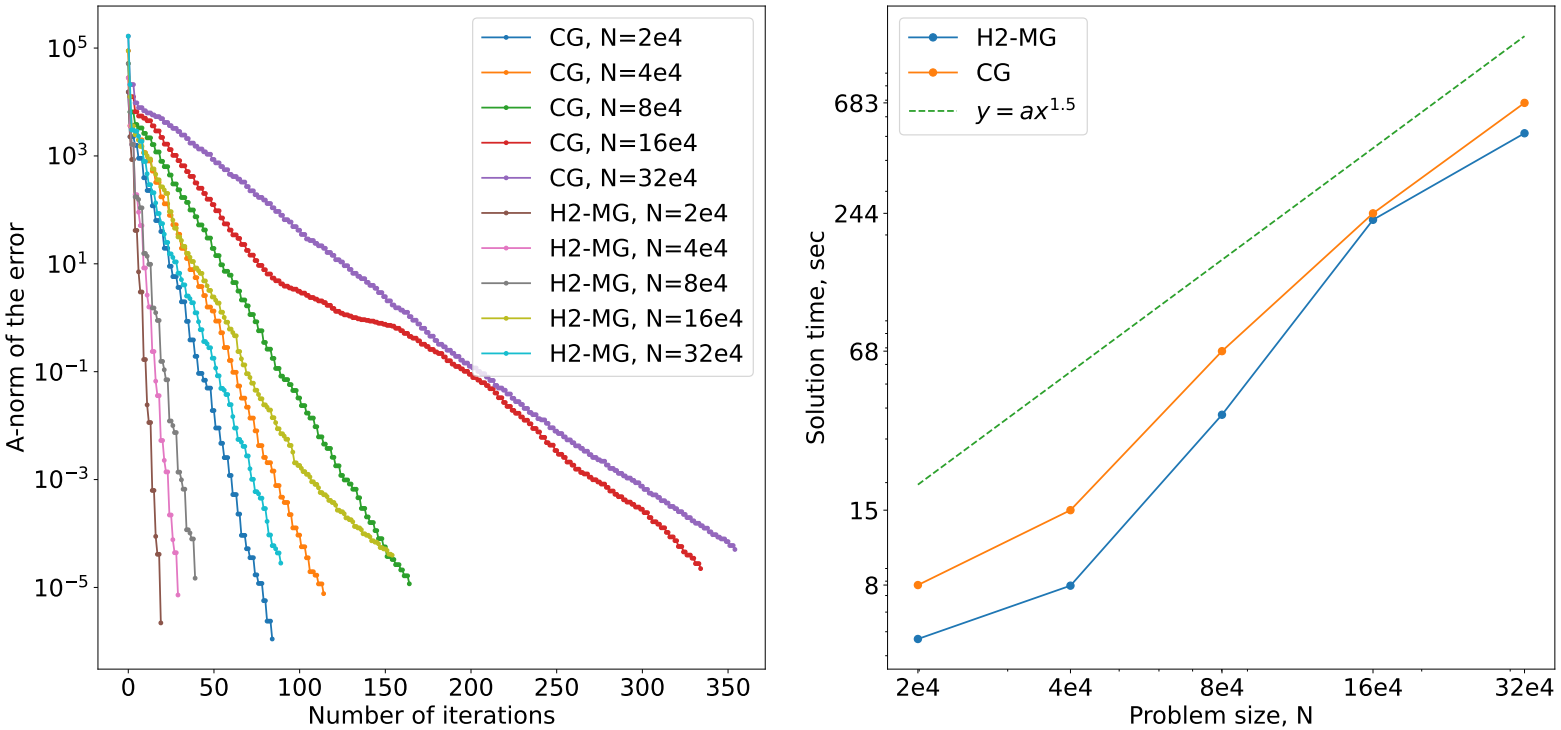
Inverse distance kernel

$Ax = b$, where $b \in \mathbb{R}^N$ is random, $x \in \mathbb{R}^N$ is an unknown, and $A \in \mathbb{R}^{N \times N}$ is a kernel matrix:

$$a_{ij} = \begin{cases} \frac{1}{|p_i - p_j|}, & \text{if } i \neq j \\ c, & \text{if } i = j \end{cases},$$

where points $p_i \in P$, $c = 2000$.

Inverse distance kernel



(a) Convergence for various problem sizes.

(b) Solution time.

Figure 3: Convergence comparison of the methods $\mathcal{H}^2$-MG and CG for different number of coarse iterations.

Inverse distance kernel

Number of V-cycles required for the convergence to the residual $\varepsilon = 10^{-5}$:

N	2e4	4e4	8e4	16e4	32e4
Number V-cycles	6	17	7	17	10

Table 2: Number of V-cycles to solve the system with inverse distance matrix.

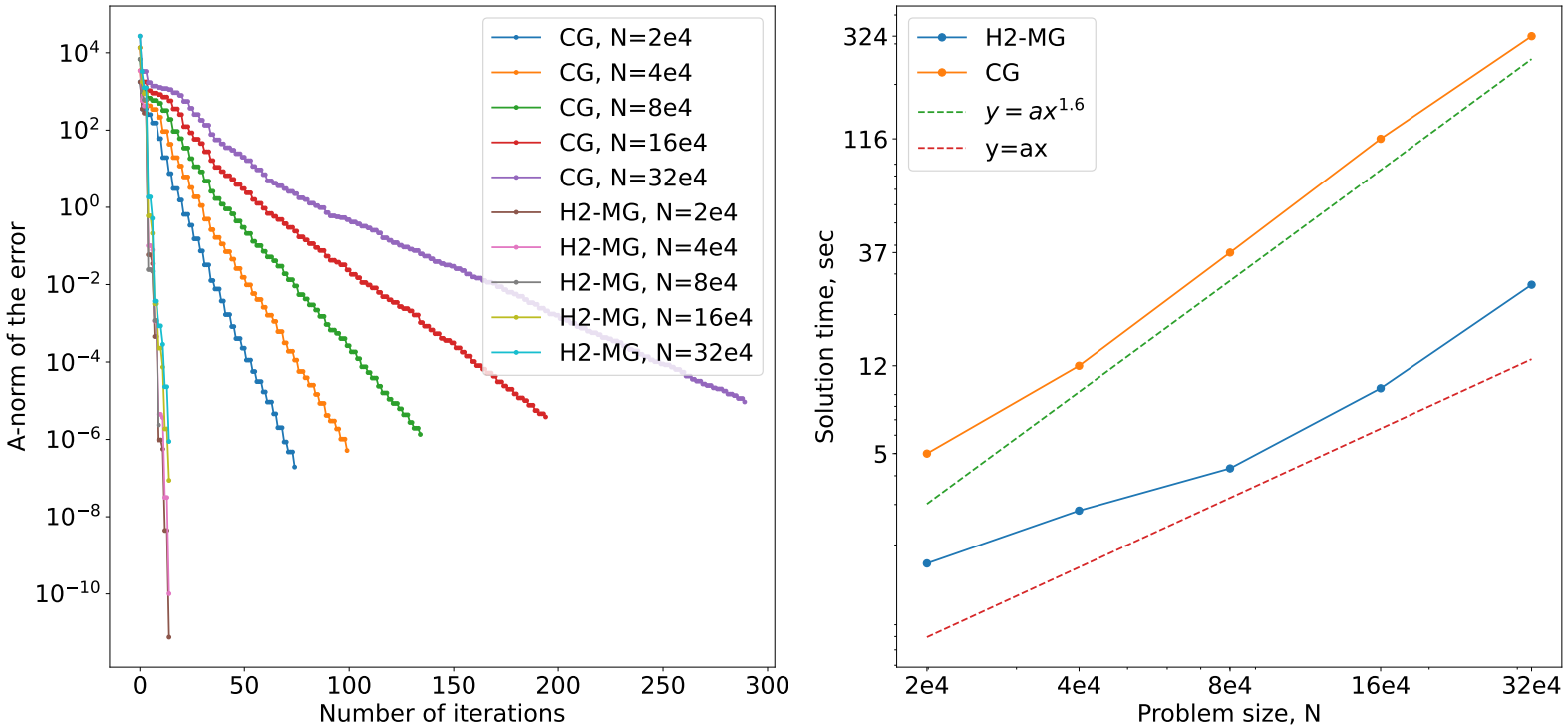
Gaussian kernel

$Ax = b$, where $b \in \mathbb{R}^N$ is random, $x \in \mathbb{R}^N$ is an unknown, and $A \in \mathbb{R}^{N \times N}$ is a kernel matrix:

$$a_{ij} = \begin{cases} \exp\left\{-\frac{|p_i - p_j|^2}{\sigma}\right\}, & \text{if } i \neq j \\ c, & \text{if } i = j \end{cases},$$

where points $p_i \in P$, $i \in 1 \ldots N$, $\sigma = 0.01$ is the dispersion parameter of the matrix, $c = 64$.

Gaussian kernel



(a) Convergence for various problem sizes.

(b) Solution time.

Figure 4: Convergence comparison of the methods $\mathcal{H}^2$-MG and CG for different number of coarse iterations.

Gaussian kernel

Number of V-cycles required for the convergence to the residual $\varepsilon = 10^{-5}$:

N	2e4	4e4	8e4	16e4	32e4
Number V-cycles	3	3	2	3	3

Table 3: Number of V-cycles to solve the system.

Conclusion

We successfully applied the multigrid method to the system with $\mathcal{H}^2$ matrix receiving the $\mathcal{H}^2$-MG solver:

- The rapid convergence (from the multigrid method)
- Time and memory efficiency (from $\mathcal{H}^2$)

Future work:

- Expand its applicability beyond the SPD matrices
- High-performance implementation
- Integration into other computational frameworks