SF2524_HW2_group34

Xinyi Ji, Jiacheng Xu

November 2025

0

AI level: 2
We use AI tool to form the code structure and plot. We write code implementation and report answer by ourself.

1

(a) The convergence of norm of relative errors and relative residuals, and theory prediction derived in task 1.b is visualized, as shown in Fig. 1.

The predicted convergence factor is calculated by:

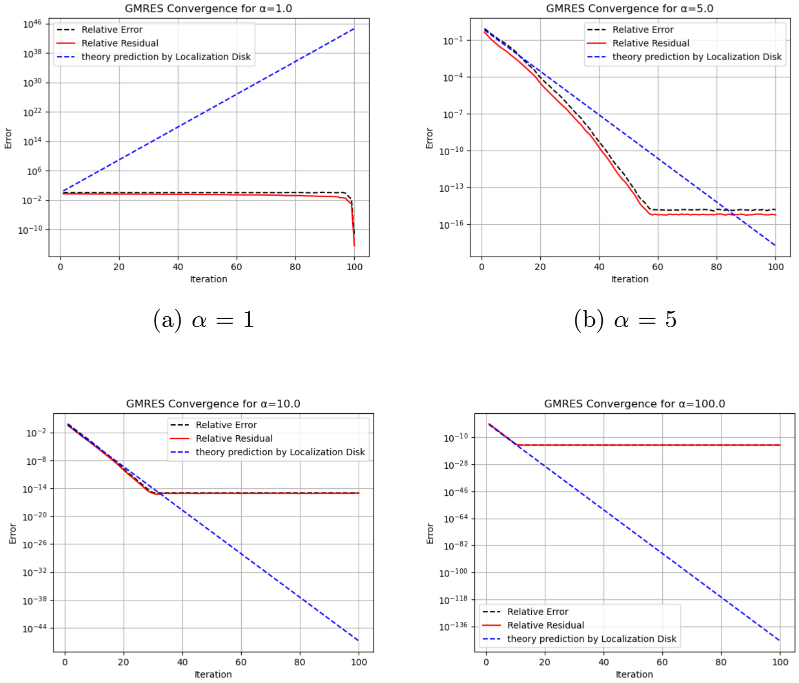
$$\rho = \frac{R}{|c|}$$

For the random sparse matrices used, when α is set to be small, the radius R is close to 1, while the center $|c|$ is relatively small, which leads the explosion growth of theoretical prediction.
With the increasing of α, the prediction becomes more accurate.

(b) The eigenvalues of matrices are plotted in Fig. 2, as well as localization disks.

For $\alpha = 1$, the eigenvalues are widely spread in a large localization disk, which is close to the origin.
As α increases, the spectrum shifts to the right, getting far from the origin. And the associated localization disk shrinks significantly.
These changes explain the GMRES behavior observed in Fig. 1: when α gets bigger, the prediction bound decreases and provides a better description of the actual GMRES convergence.

(c) The residual norms and CPU-time are recorded in the Table 1, 2 and 3.



(a) $\alpha = 1$

(b) $\alpha = 5$

(c) $\alpha = 10$

(d) $\alpha = 100$

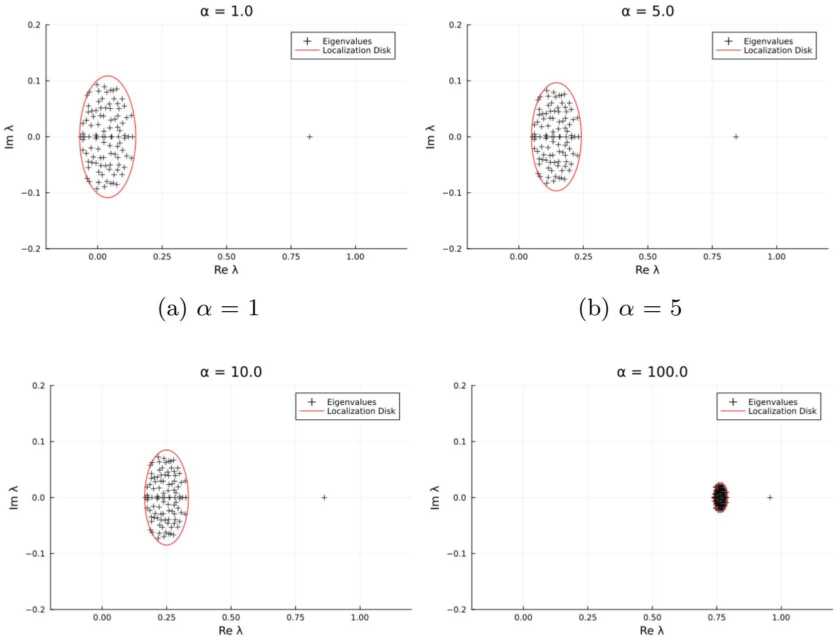
Figure 1: Convergence of GMRES

Table 1: GMRES performance for $\alpha = 1$

	n = 200		n = 500		n = 1000	
	resnorm	time	resnorm	time	resnorm	time
$m = 5$	4.069	0.00018	6.348	0.00051	9.045	0.00206
$m = 10$	4.044	0.00036	6.338	0.00114	9.034	0.00638
$m = 20$	3.868	0.00091	6.305	0.00537	9.010	0.01083
$m = 50$	3.519	0.024	6.134	0.0226	8.867	0.07035
$m = 100$	2.575	0.071	5.798	0.0986	8.627	0.18333

(d)

- When $\alpha = 1$, GMRES performs badly, with the accuracy much larger than 1e-5. In this case, backslash operator is better.
- When $\alpha = 100$, GMRES is the better choice, as it takes less time to reach the tolerance.



(a) $\alpha = 1$

(b) $\alpha = 5$

(c) $\alpha = 10$

(d) $\alpha = 100$

Figure 2: Eigenvalues

Table 2: GMRES performance for $\alpha = 100$

	n = 200		n = 500		n = 1000	
	resnorm	time	resnorm	time	resnorm	time
$m = 5$	5.89e-6	0.00165	9.85e-5	0.000496	0.00073	0.00215
$m = 10$	8.40e-13	0.000299	1.62e-10	0.00136	8.06e-9	0.00454
$m = 20$	4.24e-15	0.000866	1.03e-14	0.00334	2.23e-14	0.0133
$m = 50$	4.25e-15	0.00869	1.03e-14	0.904	2.17e-14	0.0457
$m = 100$	4.25e-15	0.0655	1.02e-14	0.147	2.29e-14	0.193

Table 3: backslash performance

	n = 200		n = 500		n = 1000	
	resnorm	time	resnorm	time	resnorm	time
$\alpha = 1$	1.572e-10	0.0094	3.655e-10	0.0162	3.435e-8	0.148
$\alpha = 100$	5.363e-15	0.0070	1.194e-14	0.0220	2.397e-14	0.0868

2

(a) The residual norms according to iterations of three different methods are shown in Fig. 3 and Fig. 4.

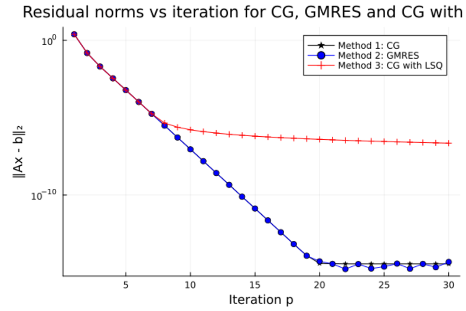


Figure 3: Residual norms of 3 methods

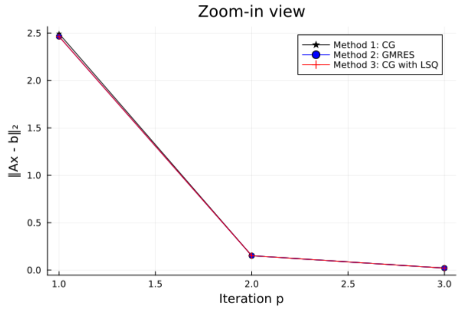


Figure 4: Residual norms(Zoomed)

(b) **In theory**, Method 2) GMRES and Method 3) CG with least squares should be identical. Because both methods minimize the same residual norm(2-norm) over the same Krylov space.

(c) **In practice**, CG with least squares has a higher y-value after several iterations. This might be because this method forms a highly ill-conditioned basis and consequently, it stagnates at a large residual. While the residuals of CG and GMRES continue to decrease, because of the orthonormal basis.

(d) For this problem with a corresponding larger sparse SPD matrix, CG Method is the best. Because it only needs 3 vectors, which saves memory storage, and more stable.

3

(a) The local heat distribution of each finite element unit in the CPU socket provides key data for optimizing the CPU heat design. The direct method ensures the accuracy of the solution but the computational cost is high, while the iterative method is preferred because of its efficient handling of large- scale problems.

(b) GMRES convergence in a scenario where singe eigenvalue outside the disk. The convergence rate between with and without single outlier are similar, because single eigenvalue is independent of m (Exponential term).

(c) GMRES's implementations often encounter unfavorable distribution of eigenvalues. Preconditioning by multiplying the system by the inverse of a matrix M, aims to enhance GMRES's performance without explicitly computing M.

(d) Apply precondition to the conjugate gradient (CG) method. Employing both left (M1) and right preconditioning (M2), to make B SPD and approximately equal to the identity matrix for rapid CG convergence.

(e) To apply CG with preconditioning, we use change variable of x_m, p_m, r_m. Update $\hat{x_m}$ by $R^{-1}x_m$, $\hat{r_m}$ by $R^T r_m$, $\hat{P_m}$ by $R^{-1}P_m$.

(f) We derive Bi-CG from PCG. When A and M are not SPD, Bi-CG then can be applied. Bi-CG is equivalent to system that PCG about to adjoint system of Ax=b.

CGNE convergence theory: CGNE means CG applied to $A^T AX = A^T b$. The error bound is min max of $A^T A$, which is sigular value of A. Since $\lambda(A^T A)$ more spread out than $\lambda(A)$. CGN convergence per iteration slower than CG or GMRES.

(4)

The convergence of the CG method is bounded by

$$\frac{\|Ax_m - b\|_A}{\|b\|_A} \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^m, \tag{1}$$

where

$$\kappa = \frac{\lambda_{\max}}{\lambda_{\min}}. \tag{2}$$

Maximum eigenvalue: $\lambda_{\max} = 10.5$.
Minimum eigenvalue: $\lambda_{\min} = 2$.

Thus, the condition number κ is

$$\kappa = \frac{\lambda_{\max}}{\lambda_{\min}} = \frac{10.5}{2} = 5.25. \tag{3}$$

Compute the convergence factor θ:

$$\theta = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} = \frac{\sqrt{5.25}-1}{\sqrt{5.25}+1}. \tag{4}$$

Using the convergence bound,

$$2\theta^m \leq 10^{-7}. \tag{5}$$

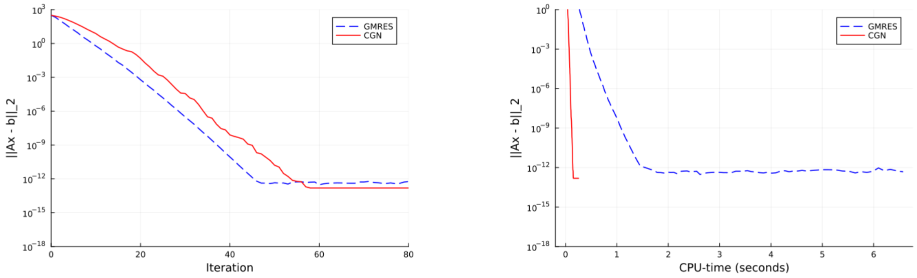
Taking logarithms,

$$\ln(2) + m\ln(\theta) \leq -7\ln(10). \tag{6}$$

Hence,

$$m = 18. \tag{7}$$

5



(a) (b)

Figure 5: Compare GMRES and CGN

(b) GMRES has slightly less iteration steps compare to CGN, but CGN is way more faster on CPU-time.

About iteration steps: GMRES's m step iteration, is to minimize $Ax - b$ in Krylov subspace $K_m(A, b)$

$$\frac{\|Ax_m - b\|}{\|b\|} \leq \|V\|\,\|V^{-1}\| \min_{p\in\mathcal{P}_m^0} \max_{i=1,\ldots,n} |p(\lambda_i)|. \tag{8}$$

polynomial $p(\lambda_i)$ corresponds to Krylov subspace. CGNE also minimize in Krylov subspace, but with $A^TAx = A^Tb$. The singular values, are often more spread out than the eigenvalues.

About CPU time: The GMRES based on Arnoldi, every iterates need to apply orthogonolization to all previous basis in Krylov subspace. CGNE is a short-term recurrence method, every iterates only contains two matrix vector products. So every iterates of CGNE is cheap.

6

(a) The average computation time for one matrix vector product. The vector x using random vector.

- Model $X = 1:$ $128.271 \mu s$
- Model $X = 2:$ $1042.096 \mu s$

(c) The average computation time for backslash:

- Model $X = 1:$ $46.659 ms$
- Model $X = 2:$ $841.16 ms$

(d) Using given criteria to measure accuracy of D^{-1} and M^{-1}. Lower means higher accurate. With Model X = 1:

- Average accuracy (M^{-1}): 0.0120198
- Average accuracy (D^{-1}): 0.0230292
- precondition average time(M^{-1}): 430.879 μs
- precondition average time(D^{-1}): 184.929 μs

For the preconditioner, M^{-1} is more accurate, D^{-1} is faster.

(e) Improved pcg_v1 implementation by 1 matrix-vector product with A and 1 preconditioner application per iteration.

GMRES with preconditioner M is the best method here. Compare within iterations, GMRES with M got lowest relative residual. Backslash is better when high accuracy is required ($\leq 10^{-10}$).

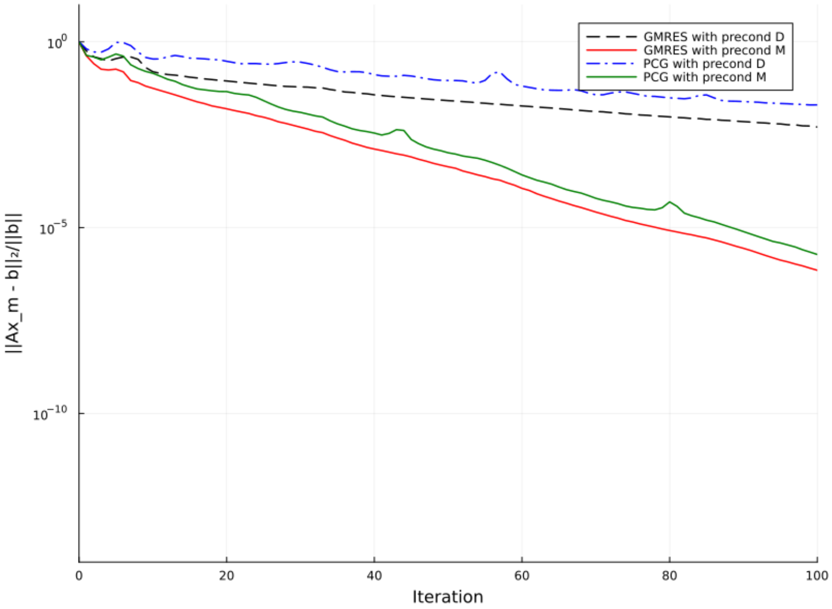


Figure 6: Relative residual per iteration with pre-condition