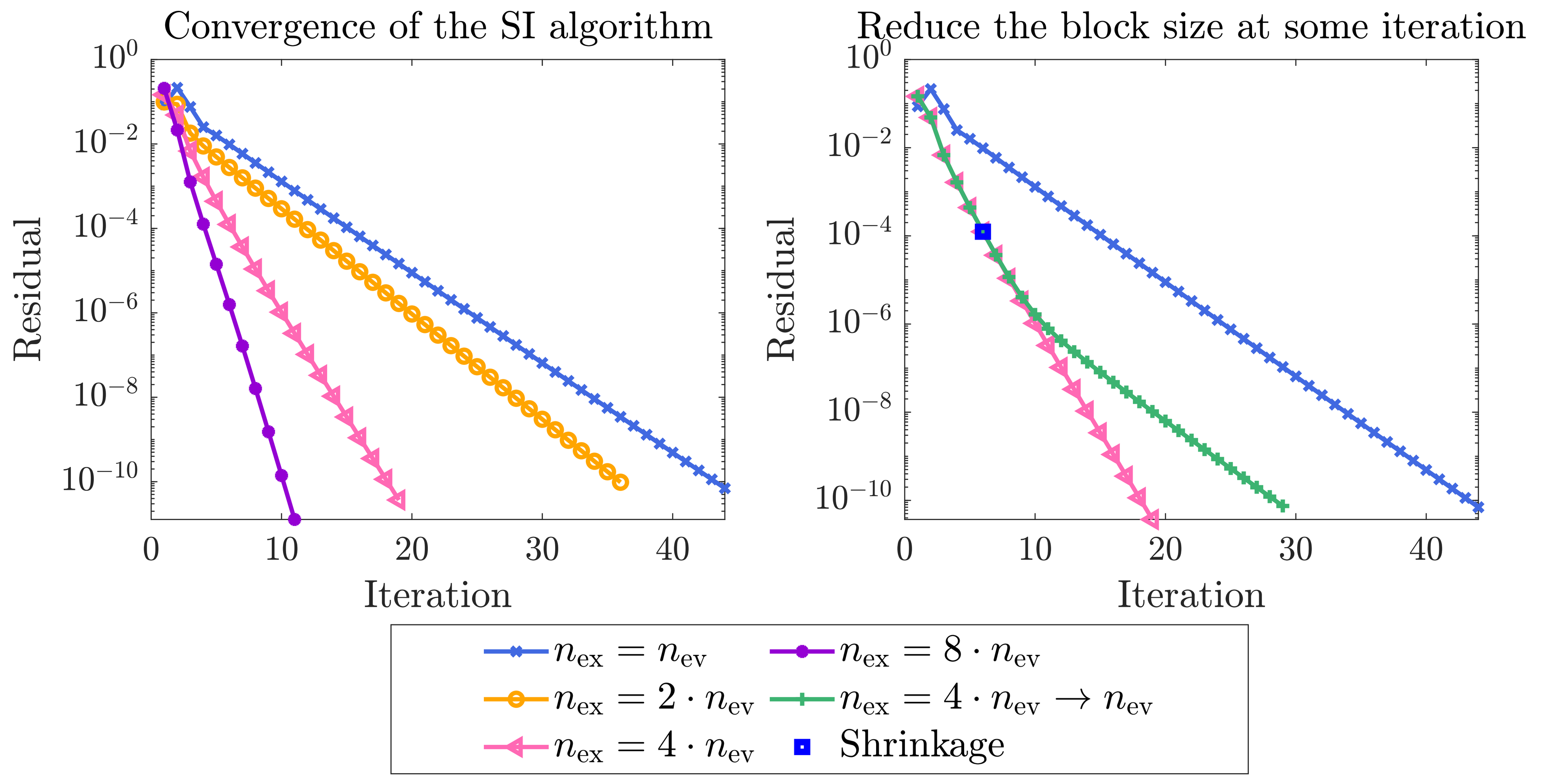
An aggressive shrinkage technique for block eigensolvers

Yuqi Liu[1], Yuxin Ma[2] and Meiyue Shao[3]

1 School of Mathematical Sciences, Fudan University, China. Email: yuqliu21@m.fudan.edu.cn
2 Department of Numerical Mathematics, Faculty of Mathematics and Physics, Charles University, Czechia.
3 School of Data Science and Shanghai Key Laboratory for Contemporary Applied Mathematics, Fudan University, China.

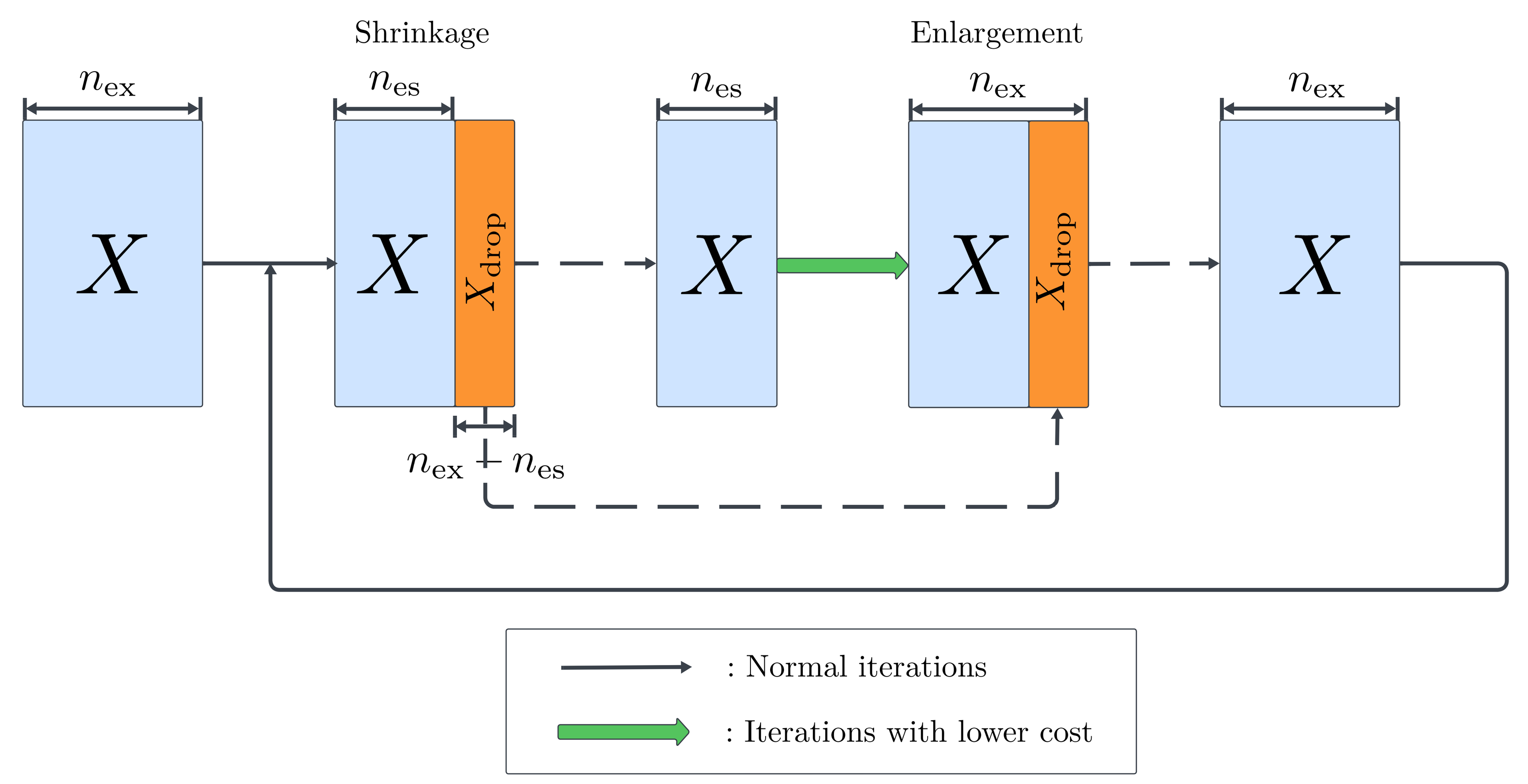
Motivation

- Algorithms where a **fixed-size** subset of vectors is stored and updated in each iteration.
  - The subspace iteration (SI) algorithm.
  - The LOBPCG algorithm.
- A **larger** subspace is used for rapid convergence (left figure).
- Though the **iterations** get fewer, the cost increases at least **quadratically**.
- If we reduce the block size during the process, the convergence rate **will not** decay immediately (right figure).



The shrink-and-enlarge technique

- **Shrinkage**: Reduce the block size for a **lower computational cost**.
- **Enlargement**: Increase the block size for a **better convergence rate**.
- Idea: Alternatively use the shrinkage and the enlargement, reduce computational complexity while keep the convergence rate satisfying.



A general framework

- A framework for applying the shrink-and-enlarge technique in block eigensolvers.

Input: A matrix A, an initial guess $X^{(0)}$.
Output: The approximate eigenpairs (Λ, X).

```
Obtain an initial approximation (Λ, X) by the Rayleigh–Ritz process on span{X^(0)}.
for k = 1, 2, ... until convergence do
    Check convergence.
    Update X (e.g., X ← A^{-1}X for the SI algorithm).
    if ifenlarge() then
        X ← [X, X_drop].
    end if
    Construct the search space S by X (possibly, also by other information).
    Obtain the approximate eigenpairs (Λ, X) by the Rayleigh–Ritz process on S.
    if ifshrink() then
        [X, X_drop] ← X.
    end if
end for
```

My homepage

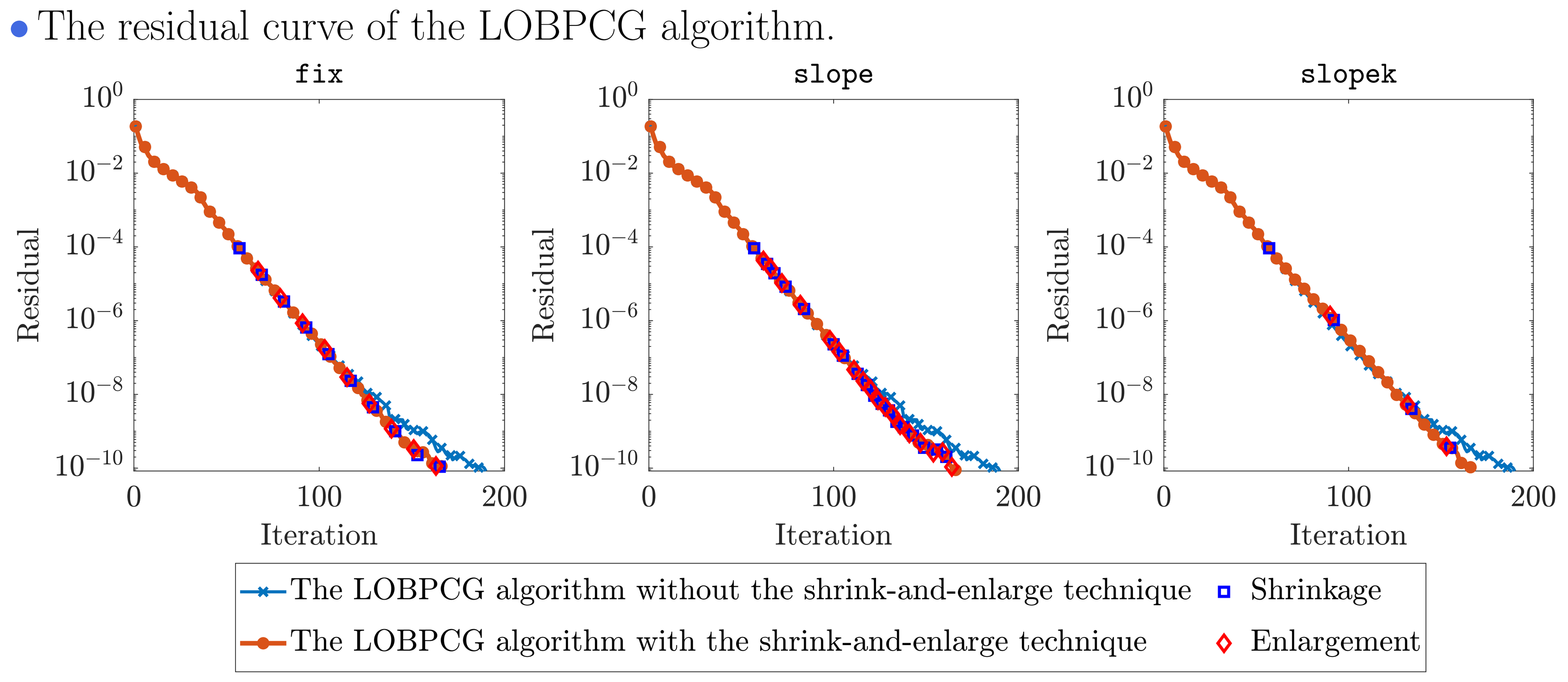
(for a lager one, at the back)

Adaptive strategies

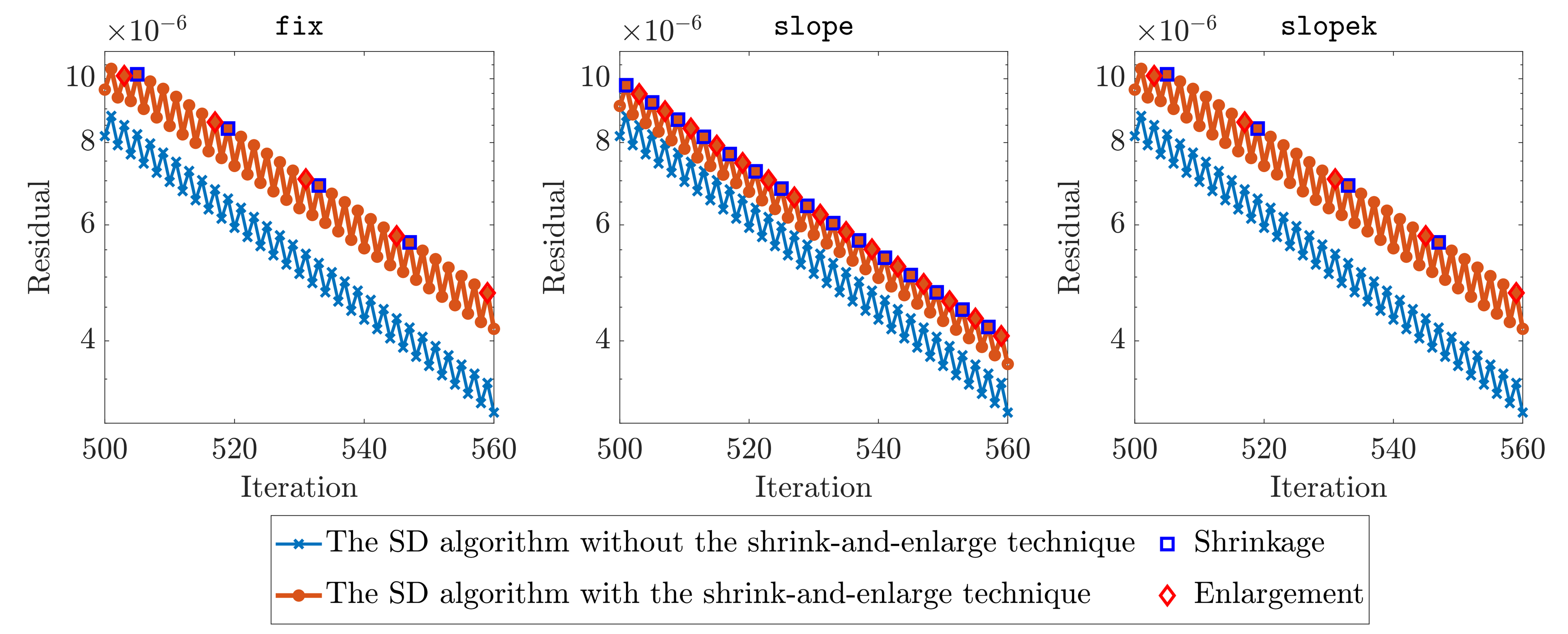
- We provide three adaptive strategies to determine the timing of employing the shrinkage and the enlargement.
  - `fix`: Alternatively employ the shrinkage and the enlargement at **fixed intervals**.
  - `slope`: Employ the shrinkage when **the slopes of the residual curves** are steep and employ the enlargement when the slopes are shallow.
  - `slopek`: Use the average slope of **several** iterations in the `slope`.

The convergence history

- Solving 100 smallest eigenpairs of matrix `Muu` from [3].
- The residual curve of the LOBPCG algorithm.



- The residual curve of the steepest descent (SD) algorithm.



Overall performance

- The performance of the proposed technique on four popular eigensolvers.
  - The test matrices are 1: `Muu`, 2: `shuttle_eddy`, 3: `mario001` and 4: `Andrews` from [3].
  - The sizes of the four test matrices range from 7,102 to 60,000.
  - Solving 1% smallest eigenpairs of the test matrices, at most 500, at least 100.
  - Time limit is set as 3600 seconds.

Algorithm	No.	w/o		fix			slope			slopek		
		time	iter	time	iter	save	time	iter	save	time	iter	save
(Inverse) Subspace iteration	1	6.395	76	5.395	80	16%	5.104	81	20%	5.103	84	20%
	2	30.57	196	23.04	212	25%	22.03	222	28%	22.08	224	28%
	3	287.9	113	207.1	121	28%	198.1	124	31%	198.5	128	31%
	4	2227	79	1608	83	28%	1586	85	29%	1581	86	29%
Steepest descent	1	75.28	990	59.21	1090	21%	74.63	1077	1%	53.28	1102	29%
	2	205.6	507	115.9	592	44%	155.7	577	24%	114.1	596	45%
	3	3566	1144	2268	1229	36%	2909	1211	18%	2111	1248	41%
	4	1622	272	1164	292	28%	1393	288	14%	1072	301	34%
LOBPCG	1	18.51	184	13.19	163	29%	14.13	163	24%	13.31	178	28%
	2	40.25	131	29.23	135	27%	29.17	133	28%	27.24	132	32%
	3	808.8	197	552.6	181	32%	604	183	25%	537.5	185	34%
	4	550.8	88	432.5	94	21%	476.3	89	14%	434.8	95	21%
TraceMIN	1	45.25	82	34.12	87	25%	33.94	87	25%	34.52	90	24%
	2	128.1	197	92.64	215	28%	91.56	226	29%	92.93	229	27%
	3	∞	-	2334	125	100%	2147	129	100%	2149	133	100%
	4	∞	-	∞	-	-	∞	-	-	∞	-	-

References

[1] Yuqi Liu, Yuxin Ma, and Meiyue Shao. An aggressive shrinkage technique for block eigensolvers.
[2] Yuqi Liu, Xinyu Shan, and Meiyue Shao. A contour integral-based algorithm for computing generalized singular values, 2023. arXiv:2401.00121
[3] Timothy A. Davis and Yifan Hu. The university of Florida sparse matrix collection. *ACM Trans. Math. Software*, 38(1):1-25, 2011.