

# RANDOMIZED AND ASYNCHRONOUS ALGORITHMS FOR LARGE LINEAR SYSTEMS

ERIK G. BOMAN  
SANDIA NATIONAL LABORATORIES

Randomization is a useful algorithmic tool, but is often considered a method of last resort. As we enter the age of exascale computing with ever-growing problem sizes, we believe randomized algorithms will become increasingly important. They may even outperform deterministic algorithms for standard problems such as solving linear systems. Randomized algorithms have several advantages:

- (1) They are often simple to implement.
- (2) They often require little synchronization, and some versions may run completely asynchronously.

As synchronization will become a bottleneck on exascale systems, we believe these types of methods will become important. Randomized algorithms often raise concerns, such as:

- Are they sufficiently accurate? Or too slow (for a given tolerance)?
- What if they fail? Success “with high probability” is not acceptable in many cases.

We think that although these are valid concerns, the objections are most relevant when randomized algorithms are used as a complete solver. Instead, we argue they should be used as a preconditioner in an iterative method. For preconditioners, it is acceptable to have low accuracy and occasionally fail to return a correct answer, since there is an outer iteration to guarantee convergence. Preconditioners try to accelerate convergence.

Randomized and asynchronous algorithms have been successful in several areas of computer science, and have received a growing amount of interest in recent years in linear algebra. We focus on two specific problems: Linear least squares (dense) and general sparse linear systems.

## 1. LINEAR LEAST SQUARES

The classic least squares problem is

$$\min_x \|Ax - b\|_2,$$

where we assume  $A \in R^{m \times n}$  is overdetermined so  $m \geq n$ . This problem is typically solved either by the normal equations ( $A^T Ax = A^T b$ ) or by computing the QR factorization of  $A$  (preferred, as it is more stable). Either way, good software exist and the complexity is  $O(mn^2)$ . A different approach is to sample the rows of  $A$  and solve a (much) smaller least-squares problem. A naive approach requires a sample size of  $O(\epsilon^{-1})$  to reach an accuracy of  $\epsilon$ , so is not very practical. However, a random sample may be a highly effective preconditioner for a Krylov method, such as conjugate gradients or LSQR. Rokhlin and Tygert [5] proposed a randomized algorithm (based on sampling and preconditioning) that requires  $O(\log(1/\epsilon)mn + n^3)$  work, which is asymptotically less than the classic methods. Avron and Toledo [1] proposed a similar randomized algorithm and demonstrated that it not only has a good asymptotic bound, but that an implementation actually outperforms LAPACK on real problem instances by up to 5X.

All these results were for dense problems. Sparse problems are harder because sampling by itself is not sufficient. In the works above, row mixing is used to reduce the “coherence”. Unfortunately, this step also destroys sparsity, so the sparse version is still an open problem.

---

Sandia is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energys National Nuclear Security Administration under contract DE-AC04-94AL85000.

## 2. SPARSE LINEAR SYSTEMS

Solving  $Ax = b$  where  $A$  is large and sparse is a key problem in scientific computing and has been well studied. Although asymptotically optimal multigrid solvers exist for certain types of problems, there is no optimal solver for general systems. In practice, preconditioning is crucial and is often application-specific. We believe randomized methods have a role to play. Methods that have long fallen out of favor might come back. One example is the *Kaczmarz* row projection method. The idea is simple: Pick a row in  $A$  and project the current iterate onto that constraint; repeat until converged. Traditionally, one cycles through the rows from first to last. This method is generally not competitive with Krylov methods and is only used in a few application areas. However, recently a randomized version [6] was proposed where the constraints are randomly picked according to some probability distribution. It was shown that convergence is exponential. The constant factor appears to be large, so for small moderate problem sizes this is still not a competitive method. For very large problem sizes though, the randomized Kaczmarz method will eventually outperform most other iterative solvers. The randomized Kaczmarz method has recently been used in a to solve symmetric diagonally dominant systems in near linear time [4], using graph theory and a novel reformulation of the original problem.

Another line of work is *asynchronous* iterative methods. Iterative methods usually have several synchronization points per iteration, for example, inner products. Synchronizations will become increasingly costly as the number of processors (cores) grow, and should be avoided as much as possible. An important question is thus if iterative methods can still converge with less synchronization. Chaotic relaxation [2] first demonstrated that asynchronous methods indeed converge, and inspired much other work [3]. A typical approach is that processors use the most recent information available to them, and different processors may proceed at different speeds. This greatly reduces the processor idle time. The analysis has mostly focused on correctness, and less on the convergence rate. Randomization is a useful tool, though not always required. However, the combination of asynchronous and random is powerful. This is still a young research topic compared to the mainstream numerical linear algebra. Currently, only quite simple iterative methods, such as Kaczmarz and Gauss-Seidel have been analyzed. The trade-offs for exascale computing are not well understood. In particular, Krylov methods may improve convergence at the expense of some synchronization. The question is how long to run an asynchronous method as a preconditioner. Note that if the preconditioner is randomized, a flexible Krylov method is necessary since the preconditioner changes in each iteration. Much research remains to be done.

## REFERENCES

1. Haim Avron, Petar Maymounkov, and Sivan Toledo, *Blendenpik: Supercharging lapack's least-squares solver*, SIAM J. Scientific Computing **32** (2010), no. 3, 1217–1236.
2. D. Chazan and W. Miranker, *Chaotic relaxation*, Linear Algebra and its Applications **2** (1969), no. 2, 199 – 222.
3. Andreas Frommer and Daniel B. Szyld, *On asynchronous iterations*, Journal of Computational and Applied Mathematics **123** (2000), 201–216.
4. Jonathan A. Kelner, Lorenzo Orecchia, Aaron Sidford, and Zeyuan Allen Zhu, *A simple, combinatorial algorithm for solving sdd systems in nearly-linear time*, CoRR **abs/1301.6628** (2013).
5. Vladimir Rokhlin and Mark Tygert, *A fast randomized algorithm for overdetermined linear least-squares regression*, Proceedings of the National Academy of Sciences **105** (2008), no. 36, 13212–13217.
6. T. Strohmer and R. Vershynin, *A randomized Kaczmarz algorithm with exponential convergence*, Journal of Fourier Analysis and Applications **15** (2009), 262–278.