

# ENGINEERING A RANDOM-SAMPLING NUMERICAL LINEAR ALGEBRA ALGORITHM

HAIM AVRON, PETAR MAYMOUNKOV, AND SIVAN TOLEDO

## 1. INTRODUCTION

Several random-sampling algorithms that solve problems in linear algebra have been proposed in the last decade [3, 5]. These techniques have been used successfully in theoretical computer science and in other application areas like machine learning. As for numerical linear algebra their impact has been limited so far. Several new theoretical algorithms have been developed but no practical linear solver has been demonstrated. There is still no evidence that they can replace older algorithms in general-purpose solvers like LAPACK [1], because their theoretical analyses have been asymptotic. In addition, some of these algorithms suffer from poor accuracy/performance tradeoff (their cost rises quickly with increased accuracy). We have engineered a general-purpose random-sampling least-square solver that outperforms LAPACK by large factors on realistic problem sizes while achieving similar accuracy. Our solver scales better than LAPACK's, so the performance difference grows with problem size. These results, shown in this abstract, suggest that random-sampling algorithms should be incorporated into future versions of LAPACK. The full paper (and the CSC09 talk) will also present several variations in the algorithm as well as many more performance results.

## 2. OVERVIEW OF THE ALGORITHM

Our algorithm minimizes large highly overdetermined systems  $x_{\text{opt}} = \arg \min_x \|Ax - b\|_2$  where  $A \in \mathbb{R}^{m \times n}$  and  $b \in \mathbb{R}^m$ . Traditionally,  $A$  is factored using, say, a  $QR$  factorization, at a cost of  $\Theta(mn^2)$ . A simple random-sampling approach is to factor only a randomly-selected subset of  $A$ 's rows. That is, we randomly form an  $r \times m$  sampling matrix  $\mathcal{S}$  and factor  $\mathcal{S}A = QR$ . This factorization cannot be used to minimize the sum of squares, but  $R$  can be used as a preconditioner for an iterative solver like LSQR [4]. Obviously, if  $r = m$  then  $\mathcal{S}A = A$  and LSQR will converge in one iteration, so with enough samples,  $R$  is a good preconditioner. How many samples do we really need?

For matrices with random independent uniform entries  $r = 4n$  usually yields a good preconditioner, but this is not true for all matrices (e.g., one needs  $r = \Omega(m)$  for the  $m$ -by- $n$  identity). It turns out that the number of samples needed is related to the *coherence* of the matrix [2].

**Definition 2.1.** Let  $A$  be an  $m \times n$  matrix and let  $A = U\Sigma V^*$  be it's reduced SVD decomposition. The *coherence* of  $A$  is defined as

$$\mu(A) = \max \|U_{i,*}\|_2^2.$$

The coherence of a matrix is always smaller than 1 and bigger than  $n/m$ . Note that it does not depend on the condition number of  $A$ . Random sampling yields a good preconditioner on incoherent matrices (matrices with small coherence). For example, if  $\mu(A) = n/m$ , then only  $\Theta(n \log n)$  rows need to be sampled to obtain a good preconditioner. Unfortunately, we cannot always guarantee a bound on  $\mu(A)$  in advance.

Drineas et al. [3] suggest to address this difficulty with a row-mixing strategy: they multiply  $A$  from the left by  $\mathcal{H} = HD$  where  $D$  is a random diagonal matrix with  $\pm 1$  on it's diagonal and  $H$  is a Hadamard matrix. Multiplying  $\mathcal{H}$  by  $A$  can be done in  $O(mn \log m)$  operations using the fast Walsh-Hadamard transform. It can be shown that with high probability,  $\mu(\mathcal{H}A) = O((n/m) \log m)$ . At this point, random sampling can be used to form a preconditioner.

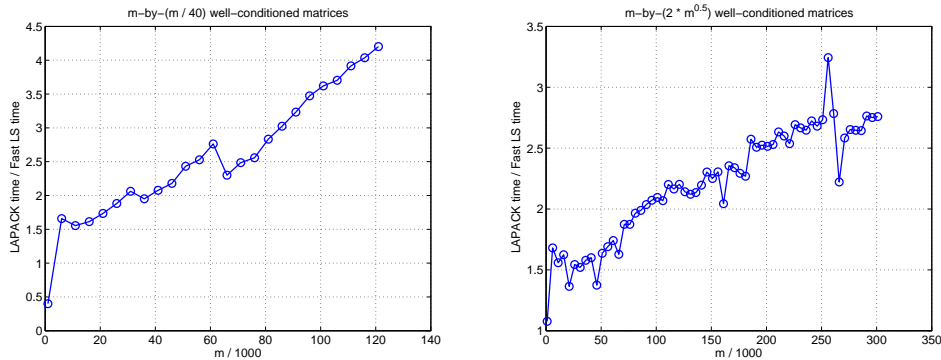


FIGURE 3.1. Comparison between LAPACK and the new solver for increasingly larger matrices. Graphs show the ratio of LAPACK’s running time to the new solver’s running time on random matrices with two kinds of aspect ratios

To summarize, the algorithm proceeds as follows. A random diagonal matrix  $D$  with  $\pm 1$  on its diagonal with equal probability is formed. The fast Walsh-Hadamard transform is applied on  $DA$ . We then sample  $\gamma n$  ( $\gamma$  is a parameter) rows from  $HDA$  to form a new matrix. A reduced  $QR$  factorization of the matrix is found and  $R$  is used as a preconditioner for LSQR. Assuming a constant number of LSQR iterations, the total cost is  $\Theta(mn \log m + n^3)$  operations.

### 3. NUMERICAL EXPERIMENTS

We implemented the least-squares solver and conducted extensive numerical experiments. Here we report only some of the results (the rest are reported in the full version of the paper and will be described in the talk). The benchmark code is LAPACK’s function `DGELS`. The code is implemented in C and uses BLAS routines for basic matrix operations. We set LSQR’s convergence threshold to  $10^{-14}$ , which is close to  $\epsilon_{\text{machine}}$ . We did not set it lower to avoid stagnation of the iterative method close to convergence. We measured the running times on a machine with two AMD Opteron 242 processors (we only used one) running at 1.6 GHz with 8 GB of memory. Matrices were generated using MATLAB’s `rand` function (random independent uniform entries).

In Figure 3.1 we see running time comparison of the new solver to LAPACK for increasingly larger matrices. Graphs show the ratio of LAPACK’s running time to the new solver’s running time. All the matrices are well conditioned. We see that for very small matrices LAPACK is faster, but already for small matrices the new algorithm is faster, and the ratio grows with matrix size. For large matrices the new solver is about four times faster than LAPACK. In experiments not reported here we have found that even on highly incoherent matrices the solver is reliable and faster than LAPACK, although by smaller factors than the results reported in Figure 3.1.

### REFERENCES

- [1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK User’s Guide*. SIAM, Philadelphia, PA, 3rd edition, 1999. Also available online from <http://www.netlib.org>.
- [2] Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *CoRR*, abs/0805.4471, 2008.
- [3] Petros Drineas, Michael W. Mahoney, S. Muthukrishnan, and Tamás Sarlós. Faster least squares approximation. *CoRR*, abs/0710.1435, 2007.
- [4] Christopher C. Paige and Michael A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Softw.*, 8(1):43–71, 1982.
- [5] Vladimir Rokhlin and Mark Tygert. A fast randomized algorithm for overdetermined linear least-squares regression. *Proceedings of the National Academy of Sciences*, 105(36):13212–13217, 2008.