Random Butterfly Transformations
with Applications in Computational Linear Algebra

D. Stott Parker
stott@cs.ucla.edu
Computer Science Department
University of California
Los Angeles, CA 90024-1596

July 23, 1995

Abstract

Theory and practice of computational linear algebra differ over the issue of degeneracy. Block matrix decompositions are used heavily in theory, but less in practice, since even when a matrix is nondegenerate (has full rank) its block submatrices can be degenerate. The potential degeneracy of block submatrices can completely prevent practical use of block matrix algorithms.

Gaussian elimination is an important example of an algorithm affected by the possibility of degeneracy. While the basic elimination procedure is simple to state and implement, it becomes more complicated with the addition of a *pivoting* procedure, which handles degenerate matrices having zeros on the diagonal. Pivoting can significantly complicate the algorithm, increase data movement, and reduce speed, particularly on high-performance computers.

We propose a randomization scheme that preconditions an input matrix by multiplying it with random matrices, where this multiplication can be performed efficiently. At the expense of these multiplications, and making the linear system dense if it were not already, this approach makes the system nondegenerate with probability 1. We show that this preconditioning (almost certainly) eliminates the need for pivoting.

In this paper we study the properties of *Random Butterfly Transformations* (RBT), an interesting class of random matrices. RBTs resemble the Fast Fourier Transform (FFT). RBTs have three important benefits. First, they have provable randomization properties. Second, RBTs can be performed efficiently, and in particular should be useful on parallel and vector machines with architectures that support FFT-like computations. Third, RBTs deliver important numerical properties when restricted to unitary matrices, and even when restricted to well-behaved, non-unitary matrices.

The RBT approach also suggests many interesting developments beyond the elimination of pivoting. Randomization offers new possibilities for linear systems solution that are well-suited to high-performance computers.

1 Motivation

Today high-performance computers are used heavily for solving large linear systems. These computers favor algorithms with little data movement, tight loops that run in cache, simple computations that can be performed with independent parallelism over a large set of values, and things like sums that can be formed in pipeline or with fan-in.

Often linear algebra computations that are easy to state are complicated by *degeneracy*. Gaussian Elimination is a good case in point, as it is complicated by *pivoting*, which dynamically permutes rows and columns of the input matrix so that large (nonzero) matrix entries are moved to the diagonal for use in elimination. Pivoting is a necessity in practice, but it interrupts the algorithm's data flow and introduces data movement that can slow high-performance computers.

The communication overhead of pivoting can be significant on a parallel machine. A single pivot can take $O(n)$ steps on machines with limited interconnection schemes; in such a machine, the $O(n^2)$ data movements potentially required by pivoting and the synchronization they require altogether can actually take longer than the $O(n^3)$ arithmetic operations required by Gaussian Elimination. Significant overhead is also generated on both MIMD and SIMD machines. A number of authors have considered scheduling algorithms for Gaussian Elimination on MIMD machines [23]. For example, Bampis et al. [1] point out that MIMD overhead (communication and idle time) can run to $O(n^3)$, even when pivoting is ignored; Veldhorst [28] has considered the effect of partial pivoting. In [22], implementations of Gaussian Elimination with pivoting on SIMD hypercubes are detailed, and experiments show how the communications overhead grows as the number of processors increases.

There is a final very significant reason why pivoting is undesirable. Pivoting strongly encourages an iterative, column-at-a-time style; it obstructs a modern recursive, block-at-a-time style and real parallel computation. Real improvements in parallel software development will result if we can avoid bottlenecks like pivoting. This can yield recursive algorithms that run well on parallel architectures.

There is an alternative to pivoting. The primary role of pivoting is to avoid degeneracies of the input matrix. Pivoting can therefore be replaced by any transformation of the matrix that avoids degeneracy. In this paper we develop recursive randomization algorithms that accomplish this. The algorithms are reminiscent of the Fast Fourier Transform, and are therefore efficient and well-suited to high-performance computers.

2 Randomization as a Method to Avoid Degeneracy

The purpose of this paper is to show how artifacts of degeneracy like pivoting can be avoided with randomization. First, let us sketch the idea of randomizing linear transformations, and how they can be used. The remainder of the paper shows that certain transformations, the Random Butterfly Transforms, actually have the desired properties.

2.1 The Basic Idea

Let A be a nonsingular real $n \times n$ matrix, let U and V be nonsingular random $n \times n$ matrices. (By a random matrix, we mean a matrix whose entries are chosen independently from a suitable distribution. We can guarantee such matrices to be nonsingular by choosing them from a nonsingular class of matrices, such as unitary matrices. More on this shortly.)

Our basic idea is that, given the randomness of U and V, *we can perform Gaussian Elimination on the product matrix UAV without any pivoting.* Intuitively UAV is sufficiently 'random' that, with probability 1, pivoting is not needed.

We will justify this idea formally later. Taking it on faith for the moment, notice that it permits many standard computations:

- To solve $A\mathbf{x} = \mathbf{b}$, we can instead solve $(UAV)\mathbf{y} = U\mathbf{b}$. Afterwards, we compute $\mathbf{x} = V\mathbf{y}$.
- To compute A^{-1}, we can instead compute $V\,(UAV)^{-1}\,U$, or equivalently solve $(UAV)X = U$, $A^{-1} = VX$.
- To compute $det\,A$, we can instead compute $det\,(UAV)/(det\,U\ det\,V)$, assuming the determinants of U and V can be computed easily (e.g., they are unitary).

Each of the computations here is easily implemented if Gaussian Elimination without pivoting successfully factors the preconditioned matrix UAV.

Putting it another way, although it does reduce roundoff error, pivoting is essential for Gaussian Elimination only with **degenerate matrices**: matrices that yield zeroes on the diagonal. In the space of matrices, degenerate matrices have measure zero: almost every matrix is nondegenerate. So, we can transform any problem involving a possibly degenerate matrix A into a problem involving a (randomized, almost certainly) nondegenerate matrix UAV, and then transform back (derandomize) the resulting solution.

2.2 Example: Using Randomization instead of Pivoting

To illustrate the basic idea, consider the 2×2 matrix

$$A = \begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix}.$$

If we arbitrarily select the random matrices

$$U = \begin{pmatrix} +.6483 & +.7614 \\ +.7614 & -.6483 \end{pmatrix}, \quad V = \begin{pmatrix} +.7279 & +.6857 \\ +.6857 & -.7279 \end{pmatrix}$$

we get the random transform

$$UAV \quad = \quad \begin{pmatrix} +.6483 & +.7614 \\ +.7614 & -.6483 \end{pmatrix} \begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} +.7279 & +.6857 \\ +.6857 & -.7279 \end{pmatrix} \quad = \quad \begin{pmatrix} 1.444 & -.4220 \\ .5721 & -1.554 \end{pmatrix}.$$

Gaussian elimination can be applied directly to this matrix without any use of pivoting. For example, with $\mathbf{b} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$,

$$U\,\mathbf{b} \quad = \quad \begin{pmatrix} +.6483 & +.7614 \\ +.7614 & -.6483 \end{pmatrix} \begin{pmatrix} 2 \\ 3 \end{pmatrix} \quad = \quad \begin{pmatrix} 3.581 \\ -.4220 \end{pmatrix}.$$

Naked Gaussian Elimination on $U\,A\,V\,\mathbf{y} = U\,\mathbf{b}$ then gives the solution to $A\mathbf{x} = \mathbf{b}$ as

$$\mathbf{y} \quad = \quad \begin{pmatrix} 2.868 \\ 1.327 \end{pmatrix}, \qquad \mathbf{x} \quad = \quad V\,\mathbf{y} \quad = \quad \begin{pmatrix} 2.998 \\ 1.001 \end{pmatrix}$$

— correct to three digits.

2.3 Historical Perspective

Several approaches have been proposed for addressing the requirement that submatrices be nondegenerate (recursively nonsingular). First, notice that only one diagonal block need be inverted. Second, Bunch and Hopcroft [2] point out that nonsingular input matrices can always be permuted so that the nonsingularity requirement needed for Gaussian Elimination is met. Bunch and Hopcroft also mention (attributing the idea to Schönhage, but it appears in [14, p.1056]) that any nonsingular matrix A can be made recursively block-nonsingular by premultiplying with its Hermitian adjoint (conjugate transpose) A^*. The resulting matrix A^*A is Hermitian and positive definite. With this approach, for example, we can compute the inverse A^{-1} by computing a generalized inverse of A:

$$A^+ \quad = \quad (A^*A)^{-1}\,A^*.$$

When A is nonsingular, $A^+ = A^{-1}$. Unfortunately this approach is relatively expensive, since it requires multiple matrix multiplications. Furthermore, it is less stable than Gaussian Elimination with pivoting, since the condition number of A^*A is always larger than that of A: $\| A^*A \|_2 = \| A \|_2^2$, where $\| \cdot \|_2$ is the spectral matrix norm to be discussed later, so with this norm the condition number of A^*A is the square of that for A. See e.g. [12, p.271].

3 Preliminaries

3.1 Randomness and Zeroes of Polynomials

Definition 1 *A* **random real index** θ *is a real-valued integrable function* $\theta : \mathcal{D} \to \mathcal{R}$ *with a strictly increasing (absolutely) continuous probability distribution on a real interval sample space* $\mathcal{D}$ *of nonzero measure.*

A **random value** r *is a 1-to-1 function of a random real index* θ*. Here we use complex-valued functions like* $exp(\mathrm{i}\theta)$*, and insist they be integrable with respect to the probability distribution of* θ*. A* **nonzero random value** *is a random value whose function never takes the value zero.*

If r *is a random value, a property* $P(r)$ **holds with probability 1** *when* $P(r)$ *is true on all of* $\mathcal{D}$ *except possibly a subset of measure 0. (Equivalently,* $P(r)$ **holds almost surely***, or* $P(r)$ **holds almost everywhere***.)* [1]

Lemma 1 *Suppose* $p(x) = ax + b$*, where* a*,* b *are fixed complex values. Let* r *be a nonzero random value. With probability 1, if* $p(r) = 0$ *then* $a = b = 0$*.*

Proof
Assume fixed values a, b, and that $p(r) = 0$ for nonzero r. Thus it must be the case that either $a = b = 0$, or $a \neq 0$, $b \neq 0$, and $r = -b/a$. Because r is a random value, however, $r \neq -b/a$ with probability 1. □

Theorem 1 *Suppose* $p(x_1, \ldots, x_n)$ *is a fixed multivariate polynomial in* n *variables, in which each variable appears with degree at most 1. Let* $r_1, \ldots, r_n$ *be independently chosen nonzero random values. With probability 1, if* $p(r_1, \ldots, r_n) = 0$ *then* $p = 0$*.*

Proof
By induction on n. Lemma 1 establishes the base case, where $n = 1$. For $n > 1$ we can collect terms in x_n and write

$$p(x_1, \ldots, x_n) \quad = \quad a(x_1, \ldots, x_{n-1})\, x_n \;+\; b(x_1, \ldots, x_{n-1})$$

where a and b are polynomials in which variables appear with degree at most one. Suppose $p(r_1, \ldots, r_n) = 0$, for independent nonzero random r_i. Then by the previous lemma, with probability 1 $a(r_1, \ldots, r_{n-1}) = b(r_1, \ldots, r_{n-1}) = 0$, so by induction $a = b = 0$, and $p = 0$. □

When floating point numbers are used in computing these polynomials, the probability in Lemma 1 can decline to a value less than 1. For example, the probability that two randomly chosen t-bit floating-point numbers differ is $1 - 2^{-t}$. In this case Theorem 1 would find $p = 0$ with probability $(1 - 2^{-t})^n = 1 - n\,2^{-t} + O(2^{-2t})$. This is still "with probability $1 - O(2^{-t})$". Since they are distracting and are not likely to be significant (most computers today support $t \geq 56$), we omit these $O(2^{-t})$ terms from now on, but the phrase "with probability 1" should be treated with care.

[1] To support implementation with floating point numbers, we obviously need the requirement that the quantity n of random numbers used be 'small' relative to the number of floating point numbers available. If there are 2^w possible floating point numbers, clearly $n \ll 2^w$ is necessary for the matrices to be considered random in the sense intended here. Furthermore, the phrase "with probability 1" actually must be understood as relative to floating point numbers. If floating point numbers have t-bit precision, then the probability that two randomly-chosen floating point numbers are equal is 2^{-t}, and the phrase must be replaced by "with probability $1 - O(2^{-t})$".

3.2 Submatrices

Definition 2 *The* **submatrix of** A **with rows** $i_1, \ldots, i_p$ **and columns** $j_1, \ldots, j_q$ *(in that order) is denoted* $A[\, i_1, \ldots, i_p \,|\, j_1, \ldots, j_q \,]$*, and consists of the indicated entries of* A*:*

$$A[\, i_1, \ldots, i_p \,|\, j_1, \ldots, j_q \,] \quad = \quad \begin{pmatrix} a_{i_1,j_1} & a_{i_1,j_2} & \cdots & a_{i_1,j_q} \\ a_{i_2,j_1} & a_{i_2,j_2} & \cdots & a_{i_2,j_q} \\ \vdots & \vdots & \ddots & \vdots \\ a_{i_p,j_1} & a_{i_p,j_2} & \cdots & a_{i_p,j_q} \end{pmatrix}.$$

Henceforth, we assume matrix row and column indexing begins with 1.

For example, the submatrix of A *with rows 3, 2 and columns 1, 3, 1, 4 is*

$$A[\, 3, 2 \,|\, 1, 3, 1, 4 \,] \quad = \quad \begin{pmatrix} a_{31} & a_{33} & a_{31} & a_{34} \\ a_{21} & a_{23} & a_{21} & a_{24} \end{pmatrix}.$$

We also follow the convention that, when $p = 0$ *or* $q = 0$*, the submatrix is 1. So,* $A[\;|\;] = 1$*.*

3.3 Unitary Transformations

Definition 3 *Recall that a* **unitary matrix** *is a complex square matrix* U *for which its* **Hermitian adjoint** *(conjugate transpose)* U^* *is its inverse:* $U U^* = U^* U = I$*.*

A **unitary transformation** *of a square matrix* A *is any product*

$$U^* \; A \; V$$

where U *and* V *are unitary matrices.*

Unitary transformations have important computational properties:

1. Unitary transformations are easy to invert.
2. Unitary matrices are closed under a variety of operations:
   - **Product**
     If U and V are unitary matrices, their product UV is unitary.
   - **Direct Sum**
     If U and V are unitary $n \times n$ matrices, their direct sum

     $$U \oplus V \quad = \quad \begin{pmatrix} U & 0 \\ 0 & V \end{pmatrix}$$

     is a unitary $(2n) \times (2n)$ matrix.
   - **Kronecker Product**
     If U and V are $m \times m$ and $n \times n$ unitary matrices, their Kronecker product

     $$U \otimes V \quad = \quad \left(\begin{array}{c|c|c|c} u_{11}V & u_{12}V & \cdots & u_{1m}V \\ \hline u_{21}V & u_{22}V & \cdots & u_{2m}V \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline u_{m1}V & u_{m2}V & \cdots & u_{mm}V \end{array}\right)$$

     is a unitary $(mn) \times (mn)$ matrix: $(U \otimes V)^* = U^* \otimes V^* = U^{-1} \otimes V^{-1} = (U \otimes V)^{-1}$.

3. Unitary transformations affect singular values of a matrix in only limited ways.
   The i-th **singular value** $\sigma_i(A)$ is the i-th largest element of the multiset

   $$\{\!\{\ \sqrt{\lambda}\ \mid\ \lambda \text{ is an eigenvalue of } A^*A\ \}\!\}.$$

   Unitary transformations do not affect singular values: $\sigma_i(U^*\,A\,V) = \sigma_i(A)$, for $1 \le i \le n$. Moreover, if $\widetilde{A}_k$ is a $k \times k$ **principal submatrix** of $\widetilde{A} = U^*\,A\,V$ (i.e., there is some sequence of indices α of length k such that $\widetilde{A}_k = (U^*\,A\,V)[\,\alpha \mid \alpha\,]$), then the singular values of $\widetilde{A}_k$ satisfy the surprising inequalities [10, p.170]:

   $$\sigma_i(\widetilde{A}_k) \quad \le \quad \sigma_i(A) \tag{1}$$

   $$|\,det\ \widetilde{A}_k\,| \quad \le \quad \sigma_1(A)\ \cdots\ \sigma_k(A). \tag{2}$$

   Many numerical methods, including those in this paper, operate on principal submatrices.

4. Unitary transformations do not affect the norm or condition number of a matrix.
   A **unitarily equivalent matrix norm** $\|\cdot\|$ is a matrix norm satisfying $\|\,UAV\,\| = \|\,A\,\|$ for any unitary matrices U, V and any square matrix A [9]. There are a number of such norms, but we will be interested only in the **spectral norm**

   $$\|\,A\,\|_2 \quad = \quad \sigma_1(A) \quad = \quad \max\ \{\ \sqrt{\lambda}\ \mid\ \lambda \text{ is an eigenvalue of } A^*A\ \}.$$

   Using the spectral norm, the **condition number** of a square nonsingular matrix A

   $$\kappa_2(A) \quad = \quad \|\,A\,\|_2\ \|\,A^{-1}\,\|_2 \quad = \quad \frac{\sigma_1(A)}{\sigma_n(A)}$$

   is also invariant under unitary transformations.

5. Unitary transformations are numerically stable.
   Not only do unitary transformations leave condition numbers undisturbed, backward error analysis with them is straightforward. If a matrix A has error E, then a unitary transformation with matrices U, V satisfies

   $$U\,(A+E)\,V \quad = \quad U\,A\,V\ +\ U\,E\,V \quad = \quad U\,A\,V\ +\ F,$$

   and the resulting error F has (unitarily equivalent) norm $\|\,F\,\| = \|\,E\,\|$, of the same magnitude as the input [24, p.278].

The FFT is an example of a unitary transformation that is well-known to be stable, provided that the cosine and sine values are calculated accurately [27, p.22]. We discuss the FFT below.

4 Random Butterfly Transformations

In this section we consider classes of matrices U and V useful for implementing the randomized transformations just described. These classes, similar in ways to the FFT, give computationally efficient transformations with useful numerical properties.

4.1 Recursive Butterfly Matrices

We define a class of matrices that make computationally efficient transformations.

Definition 4 *A* **butterfly matrix** *will be defined here as any $n \times n$ matrix*

$$B^{\langle n \rangle} = \frac{1}{\sqrt{2}} \begin{pmatrix} R_0 & R_1 \\ R_0 & -R_1 \end{pmatrix}$$

where $n \geq 2$ and R_0 and R_1 are diagonal and nonsingular $(n/2) \times (n/2)$ matrices.

When R_0 and R_1 are both unitary (i.e., both have the form $exp(\mathrm{i}\Theta)$ for real diagonal Θ) then $B^{\langle n \rangle}$ is unitary. An important special case is where $\Theta = 0$, in which case $B^{\langle n \rangle}$ is

$$O_n = \frac{1}{\sqrt{2}} \begin{pmatrix} I_{n/2} & I_{n/2} \\ I_{n/2} & -I_{n/2} \end{pmatrix}.$$

In particular, $B^{\langle n \rangle} = O_n R$, where $R = (R_0 \oplus R_1)$ is a diagonal matrix.

A **Recursive Butterfly** *is a product of butterfly matrices. Specifically, a 1×1 matrix $U^{\langle 1 \rangle}$ is a recursive butterfly, as is any matrix $U^{\langle n \rangle}$ with $n = 2^\nu$ that is the product of the direct sum of two smaller recursive butterflies $U_0^{\langle n/2 \rangle}$, $U_1^{\langle n/2 \rangle}$, with an $n \times n$ butterfly $B^{\langle n \rangle}$:*

$$U^{\langle n \rangle} \quad = \quad (U_0^{\langle n/2 \rangle} \ \oplus \ U_1^{\langle n/2 \rangle}) \ B^{\langle n \rangle} \quad = \quad \begin{pmatrix} U_0^{\langle n/2 \rangle} & 0 \\ 0 & U_1^{\langle n/2 \rangle} \end{pmatrix} \ B^{\langle n \rangle}.$$

When $n = 4$, the recursive butterfly definition can be manipulated as follows:

$$\begin{aligned} U^{\langle 4 \rangle} &= (U_0^{\langle 2 \rangle} \ \oplus \ U_1^{\langle 2 \rangle}) \quad B^{\langle 4 \rangle} \\ &= (((U_{00}^{\langle 1 \rangle} \ \oplus \ U_{01}^{\langle 1 \rangle}) \ B_0^{\langle 2 \rangle}) \ \oplus \ ((U_{10}^{\langle 1 \rangle} \ \oplus \ U_{11}^{\langle 1 \rangle}) \ B_1^{\langle 2 \rangle})) \quad B^{\langle 4 \rangle} \\ &= (U_{00}^{\langle 1 \rangle} \ \oplus \ U_{01}^{\langle 1 \rangle} \ \oplus \ U_{10}^{\langle 1 \rangle} \ \oplus \ U_{11}^{\langle 1 \rangle}) \quad (B_0^{\langle 2 \rangle} \ \oplus \ B_1^{\langle 2 \rangle}) \quad B^{\langle 4 \rangle}. \end{aligned}$$

The definition for $U^{\langle 4 \rangle}$ can be appreciated visually:

$$\begin{aligned} U^{\langle 4 \rangle} &= \frac{1}{2} \quad (U_{00}^{\langle 1 \rangle} \oplus U_{01}^{\langle 1 \rangle} \oplus U_{10}^{\langle 1 \rangle} \oplus U_{11}^{\langle 1 \rangle}) \qquad (B_0^{\langle 2 \rangle} \quad \oplus \quad B_1^{\langle 2 \rangle}) \qquad\qquad B^{\langle 4 \rangle} \\ &= \frac{1}{2} \begin{pmatrix} r_1^{\langle 1 \rangle} & & & \\ & r_2^{\langle 1 \rangle} & & \\ & & r_3^{\langle 1 \rangle} & \\ & & & r_4^{\langle 1 \rangle} \end{pmatrix} \begin{pmatrix} r_1^{\langle 2 \rangle} & r_2^{\langle 2 \rangle} & & \\ r_1^{\langle 2 \rangle} & -r_2^{\langle 2 \rangle} & & \\ & & r_3^{\langle 2 \rangle} & r_4^{\langle 2 \rangle} \\ & & r_3^{\langle 2 \rangle} & -r_4^{\langle 2 \rangle} \end{pmatrix} \begin{pmatrix} r_1^{\langle 4 \rangle} & & r_3^{\langle 4 \rangle} & \\ & r_2^{\langle 4 \rangle} & & r_4^{\langle 4 \rangle} \\ r_1^{\langle 4 \rangle} & & -r_3^{\langle 4 \rangle} & \\ & r_2^{\langle 4 \rangle} & & -r_4^{\langle 4 \rangle} \end{pmatrix} \\ &= \frac{1}{2} \begin{pmatrix} r_1^{\langle 1 \rangle} r_1^{\langle 2 \rangle} r_1^{\langle 4 \rangle} & r_1^{\langle 1 \rangle} r_2^{\langle 2 \rangle} r_2^{\langle 4 \rangle} & r_1^{\langle 1 \rangle} r_1^{\langle 2 \rangle} r_3^{\langle 4 \rangle} & r_1^{\langle 1 \rangle} r_2^{\langle 2 \rangle} r_4^{\langle 4 \rangle} \\ r_2^{\langle 1 \rangle} r_1^{\langle 2 \rangle} r_1^{\langle 4 \rangle} & -\ r_2^{\langle 1 \rangle} r_2^{\langle 2 \rangle} r_2^{\langle 4 \rangle} & r_2^{\langle 1 \rangle} r_1^{\langle 2 \rangle} r_3^{\langle 4 \rangle} & -\ r_2^{\langle 1 \rangle} r_2^{\langle 2 \rangle} r_4^{\langle 4 \rangle} \\ r_3^{\langle 1 \rangle} r_3^{\langle 2 \rangle} r_1^{\langle 4 \rangle} & r_3^{\langle 1 \rangle} r_4^{\langle 2 \rangle} r_2^{\langle 4 \rangle} & -\ r_3^{\langle 1 \rangle} r_3^{\langle 2 \rangle} r_3^{\langle 4 \rangle} & -\ r_3^{\langle 1 \rangle} r_4^{\langle 2 \rangle} r_4^{\langle 4 \rangle} \\ r_4^{\langle 1 \rangle} r_3^{\langle 2 \rangle} r_1^{\langle 4 \rangle} & -\ r_4^{\langle 1 \rangle} r_4^{\langle 2 \rangle} r_2^{\langle 4 \rangle} & -\ r_4^{\langle 1 \rangle} r_3^{\langle 2 \rangle} r_3^{\langle 4 \rangle} & r_4^{\langle 1 \rangle} r_4^{\langle 2 \rangle} r_4^{\langle 4 \rangle} \end{pmatrix}. \end{aligned}$$

This visual presentation suggests the $U^{\langle n \rangle}$ matrix entries obey a formula that will be useful later:

Lemma 2 *In the recursive butterfly $U^{\langle n \rangle} = (u_{ij})$ where $n = 2^\nu$,*

$$u_{ij} \quad = \quad \pm \frac{1}{\sqrt{n}} \prod_{\ell=0}^{\nu} r_{f(i,j,\ell)}^{\langle 2^\ell \rangle}$$

where $f(i,j,\ell)$ is a function of ℓ and the binary representations of $(i-1)$ and $(j-1)$.

Proof As $U^{\langle n\rangle}$ is a product of a diagonal matrix and ν direct sums of butterflies, we can write $u_{ij} = r^1_{ii} \sum_{k_1} \sum_{k_2} \cdots \sum_{k_{\nu-1}} b^1_{i,k_1} b^2_{k_1,k_2} \cdots b^{\nu}_{k_{\nu-1},j}$, where b^{ℓ}_{pq} are the entries of the ℓ-th direct sum of butterflies. The result follows since $b^{\ell}_{pq} \neq 0$ iff $(p-1)$ and $(q-1)$ are equal or differ only in the ℓ-th digit of their binary representations, and in this situation $b^{\ell}_{pq} = \pm(1/\sqrt{2})\, r_q^{\langle 2^\ell\rangle}$ for $\ell \geq 1$. □

If $n = 2^\nu$ and the butterflies $B^{\langle n\rangle}$ are always O_n, we get a recursively-defined **Hadamard matrix**[2]

$$H_n \quad = \quad \left(\frac{1}{\sqrt{2}}\right)^{\nu} \bigotimes^{\nu} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad = \quad \frac{1}{\sqrt{n}} \bigotimes^{\log_2(n)} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

There is a strong connection between recursive butterfly matrices and the FFT. The **Discrete Fourier Transform (DFT)** is just the unitary $n \times n$ matrix

$$F_n \quad = \quad \frac{1}{\sqrt{n}} \left(\omega_n^{(i-1)(j-1)} \right)$$

where typically $\omega_n = exp(-\frac{2\pi}{n}\,\mathrm{i})$. The **Fast Fourier Transform (FFT)** results from factoring F_n into recursive products of butterfly transformations and permutations. Specifically:

$$F_{2n} \quad = \quad B_{2n} \ \ (F_n \oplus F_n) \ \ {\sigma_{2n}}^{-1} \quad = \quad B_{2n} \ \ (I_2 \otimes F_n) \ \ {\sigma_{2n}}^{-1}$$

where I_n is the $n \times n$ identity and B_{2n} is the butterfly

$$B_{2n} \quad = \quad \frac{1}{\sqrt{2}} \begin{pmatrix} I_n & D_n \\ I_n & -D_n \end{pmatrix}, \qquad \text{where} \qquad D_n \quad = \quad diag(\ 1,\ \omega_{2n},\ {\omega_{2n}}^2,\ \ldots,\ {\omega_{2n}}^{n-1}\ )$$

and ${\sigma_{2n}}^{-1}$ is the 'perfect unshuffle' permutation [15, 27]. Thus the adjoint, or inverse DFT, ${F_{2n}}^*$ is a recursive butterfly preceded on the left by a product of unshuffle permutations (a bit-reversal permutation). Van Loan [27] reviews how F_n can be implemented efficiently on high-performance computers. Recursive butterflies can be also.

4.2 Random Butterfly Transformations

Definition 5 *A* **random matrix** *is a matrix whose complex-valued entries are functions of independent random values sharing a common distribution.*[3]

For example, a diagonal matrix Θ whose entries are random real indices θ with a continuous distribution on the interval $[0, 2\pi)$ is a random matrix. Also, a diagonal matrix whose diagonal entries are complex exponentials $exp(\mathrm{i}\,\theta)$ dependent on a random index θ with continuous distribution on the interval $[0, 2\pi)$ are random matrices that are *always* nonsingular and unitary.

Definition 6 *A* **Random Butterfly Transformation (RBT)** *of a square matrix A is a product*

$$\widetilde{A} \quad = \quad U^*\ A\ V$$

where U and V are random recursive butterflies, i.e., recursive butterflies whose individual butterfly factors are are defined with random nonsingular diagonal R_0, R_1.

In this paper, we focus on recursive butterflies because the resulting transformation has provable randomization properties. Also, recursive butterflies are supported by many high-performance computers [27]; these RBTs are like two-sided FFTs (i.e., like F^*AF).

[2] Hadamard matrices are often not scaled by $1/\sqrt{n}$ as done here. However, this factor makes the matrix unitary.

[3] To support implementation with floating point numbers, we require that the dimension n of any matrix be 'small' relative to the number of floating point numbers available. If there are 2^w possible floating point numbers, $n \ll 2^w$ is necessary for the matrices to be considered random in the sense intended here. Furthermore, the phrase "with probability 1" actually must be understood as relative to floating point numbers. If floating point numbers have t-bit precision, then the probability that two randomly-chosen floating point numbers are equal is 2^{-t}, and the phrase must be replaced by "with probability $1 - O(2^{-t})$". See Section 3.1.

5 Randomization Properties of the RBT

Multiplying by arbitrary random matrices can be expensive, potentially requiring more time than Gaussian Elimination itself. Furthermore, multiplying by random matrices can lose numerical accuracy. We should therefore consider classes of random matrices that, at least on some machines, justify their expense in time and accuracy. We have concentrated on recursive butterfly matrices.

Let $\widetilde{A}$ be a random unitary transformation of A. We show now that we can guarantee that every principal block submatrix of the RBT $\widetilde{A}$ is nonsingular. This is an important randomization property for elimination methods.

In the companion paper [17] we investigate the randomization properties of individual (non-recursive) butterflies. It turns out that recursive block algorithms can be developed using individual butterflies for randomization. (In other words, we can interleave the recursions of the block algorithm and the randomization.) Here, we simply perform recursive randomization in one shot.

5.1 Operators on Sequences

We write **sequences** of integers as $i_1, i_2, \ldots, i_k$. The **length** of this sequence is k.

Following convention, we will use greek letters (α, β, γ, κ, λ, μ) to denote sequences.

Define $(z \operatorname{Mod} m) \stackrel{def}{=} ((z-1) \bmod m) + 1$.

Given two sequences $\alpha = i_1, \ldots, i_k$, $\beta = j_1, \ldots, j_k$, we will use the following notation:

notation	**denotation**
$\alpha \; + \; c$	$(i_1 + c), \ldots, (i_k + c)$ (c an integer)
$\alpha \;\operatorname{Mod}\; m$	$(i_1 \operatorname{Mod} m), \ldots, (i_k \operatorname{Mod} m)$
$\alpha \;\cong\; \beta$	when put into sorted order, α and β are equal
$\alpha \;\equiv\; \beta \;\operatorname{Mod}\; m$	$(\alpha \operatorname{Mod} m) \;\cong\; (\beta \operatorname{Mod} m)$.

S_k^m is the set of **strictly increasing sequences** of length k drawn from $\{1, \ldots, m\}$. For example, $1,2,5 \in S_3^5$, but $2,2,5 \notin S_3^5$, and $1,2,5 \in S_3^6$.

CS_k^m is the set of **(end-around) consecutive strictly increasing sequences** in S_k^m, i.e., sequences that are equivalent to $(i+1), \ldots, (i+k) \operatorname{Mod} m$, for some value i. For example, $1,2,5 \in CS_3^5$, but $1,3,5 \notin CS_3^5$, and $1,2,5 \notin CS_3^6$.

5.2 Determinants of Submatrices of Butterflies

Recall the butterfly matrix $B^{\langle n \rangle} = O_n \, R$, where $O_n = \frac{1}{\sqrt{2}} \begin{pmatrix} I_{n/2} & I_{n/2} \\ I_{n/2} & -I_{n/2} \end{pmatrix}$, and R is the diagonal nonsingular matrix $R = diag(r_1, r_2, \ldots, r_n)$.

Theorem 2 *For $n \geq 2$ and $1 \leq k \leq n$, if α and γ are sequences in S_k^n, then the butterfly $B^{\langle n \rangle}$ satisfies*

$$det \; B^{\langle n \rangle}[\,\gamma \,|\, \alpha\,] \quad = \quad c(n, \gamma, \alpha) \prod_{j \,\in\, \alpha} r_j$$

where $c(n,\gamma,\alpha)$ is a constant (i.e., independent of R). Furthermore, if α is consecutive ($\alpha \in CS_k^n$), then for every sequence $\gamma \in S_k^n$ such that $\gamma \equiv \alpha \operatorname{Mod} (n/2)$, the constant $c(n,\gamma,\alpha)$ is nonzero.

Proof

From $B^{\langle n \rangle} = O_n \, R$ we infer

$$det \; B^{\langle n \rangle}[\,\gamma \,|\, \alpha\,] \quad = \quad det \; O_n[\,\gamma \,|\, \alpha\,] \prod_{j \,\in\, \alpha} r_j .$$

Clearly, the product in this equation is uniquely determined by α. In other words, for any sequence λ in S_k^n with $\lambda \neq \alpha$, $det\, B^{\langle n \rangle}[\,\gamma \,|\, \lambda\,]$ and $det\, B^{\langle n \rangle}[\,\gamma \,|\, \alpha\,]$ will be either both zero, or dependent on different sets of r-variables.

Next, we can show that whenever $\alpha \in CS_k^n$ is consecutive and $\gamma \equiv \alpha$ Mod $(n/2)$, then $c(n, \gamma, \alpha) = det\, O_n[\,\gamma \,|\, \alpha\,]$ will be nonzero. Let $T = \sqrt{2}\, O_n[\,\gamma \,|\, \alpha\,]$, so $det\, O_n[\,\gamma \,|\, \alpha\,] = (\sqrt{2})^{-k} det\, T$. Since $\gamma \equiv \alpha$ Mod $(n/2)$, for any $j \in \alpha$ if we define $i = j \,\mathrm{Mod}\,(n/2)$, then at least one of the two values i and $i + (n/2)$ must be in γ. Because α is consecutive $det\, O_n[\,\gamma \,|\, \alpha\,] \neq 0$, since:

- When $k \leq (n/2)$, T is a $k \times k$ matrix that can be converted by permuting columns into a matrix whose entries are ± 1 on the diagonal and zero otherwise, so $det\, T = \pm 1$ and it follows that $det\, O_n[\,\gamma \,|\, \alpha\,] \;=\; \pm 2^{-k/2} \;\neq\; 0$.

- When $k > (n/2)$, we can reduce k by one by exploiting the fact that there must be some index i such that γ and α contain both i and $i + (n/2)$. With an elementary row operation we can reduce $det\, T$ by one column and row (i):

$$det\; T \;=\; det \begin{pmatrix} A & & B & & C \\ & 1 & & 1 & \\ D & & E & & F \\ & 1 & & -1 & \\ G & & H & & J \end{pmatrix} \;=\; det \begin{pmatrix} A & & B & & C \\ & 2 & & 0 & \\ D & & E & & F \\ & 1 & & -1 & \\ G & & H & & J \end{pmatrix} \;=\; \pm\, 2 \;\; det \begin{pmatrix} A & B & & C \\ D & E & & F \\ & & -1 & \\ G & H & & J \end{pmatrix}.$$

  Repeating this process $k - (n/2)$ times reduces this case to the first, ultimately finding that $det\, O_n[\,\gamma \,|\, \alpha\,] \;=\; (\sqrt{2})^{-k} det\; T \;=\; (\sqrt{2})^{-k}\, (\pm 2)^{k - n/2}\, (\pm 1) \;=\; \pm 2^{k/2 \,-\, n/2} \;\neq\; 0.$

□

5.3 Determinants of Block Submatrices of Recursive Butterflies

Theorem 3 *Let* $U^{\langle n \rangle}$ *be a random recursive butterfly of size* $n = 2^{\nu}$*. Then for* $1 \leq k \leq n$ *and all sequences* α, β *in* S_k^n, *det* $U^{\langle n \rangle}[\,\beta \,|\, \alpha\,]$ *is a polynomial of degree at most one in its random variables. This polynomial is either zero, or is a function of a different set of random variables than det* $U^{\langle n \rangle}[\,\lambda \,|\, \mu\,]$ *for any other* λ, μ *in* S_k^n*. Furthermore, if* $\alpha \in CS_k^n$*, then det* $U^{\langle n \rangle}[\,\beta \,|\, \alpha\,] \neq 0$.

Proof

We prove the theorem by induction on ν. The base case $\nu = 0$ is easy, since then $U^{\langle 1 \rangle}$ is a 1×1 matrix with a single random entry satisfying all the stated properties.

That the induction step holds was suggested by Lemma 2, which observed that the column indices of $U^{\langle n \rangle}$ largely determine which random variables appear in its entries. The induction step can be proved with the **Binet-Cauchy theorem** [13, p.14], which states that for square $n \times n$ matrices X, Y, any integer k such that $1 \leq k \leq n$, and any sequences κ, μ in S_k^n,

$$det\; ((XY)[\,\kappa \,|\, \mu\,]) \;\;=\;\; \sum_{\lambda \in S_k^n} \; det\; X[\,\kappa \,|\, \lambda\,] \;\; det\; Y[\,\lambda \,|\, \mu\,].$$

Now $U^{\langle n \rangle} \;=\; (U_0^{\langle n/2 \rangle} \;\oplus\; U_1^{\langle n/2 \rangle})\, B^{\langle n \rangle}$. Therefore

$$det\; U^{\langle n \rangle}[\,\beta \,|\, \alpha\,] \;\;=\;\; \sum_{\gamma \in S_k^n} \; det\; (U_0^{\langle n/2 \rangle} \;\oplus\; U_1^{\langle n/2 \rangle})[\,\beta \,|\, \gamma\,] \;\;\; det\; B^{\langle n \rangle}[\,\gamma \,|\, \alpha\,].$$

Furthermore the determinant of $(U_0^{\langle n/2 \rangle} \;\oplus\; U_1^{\langle n/2 \rangle})[\,\beta \,|\, \gamma\,]$ is zero unless there are subsequences β_0 and β_1 of $1, \ldots, n/2$ and $(n/2 + 1), \ldots, n$ whose sequence concatenation is β, and likewise for

γ_0, γ_1, and γ, such that the length of β_0 equals that of γ_0 and the length of β_1 equals that of γ_1. Then:

$$det\ (U_0^{\langle n/2\rangle} \oplus U_1^{\langle n/2\rangle})[\,\beta\,|\,\gamma\,] \quad = \quad det\ U_0^{\langle n/2\rangle}[\,\beta_0\,|\,\gamma_0\,] \quad\cdot\quad det\ U_1^{\langle n/2\rangle}[\,\beta_1 - n/2\,|\,\gamma_1 - n/2\,].$$

First, by induction these two smaller determinants are each polynomials of degree at most one in their new (disjoint) sets of random variables. By Theorem 2, $det\ B^{\langle n\rangle}[\,\gamma\,|\,\alpha\,]$ is also a polynomial of degree at most one in its variables, which in a recursive butterfly are disjoint from those of $U_0^{\langle n/2\rangle}$ and $U_1^{\langle n/2\rangle}$. Thus, for every γ, the product of these three polynomials has this property as well, and altogether $det\ U^{\langle n\rangle}[\,\beta\,|\,\alpha\,]$ is a polynomial of degree at most one in its random variables.

Second, we can prove by induction that recursive butterflies can be factored $U^{\langle n\rangle} = R^{\langle 1\rangle}\,P\,R^{\langle n\rangle}$, where $R^{\langle 1\rangle} = diag(r_j^{\langle 1\rangle})$ and $R^{\langle n\rangle} = diag(r_j^{\langle n\rangle})$ $(1 \le j \le n)$, and P is a product of other matrices. So

$$det\ U^{\langle n\rangle}[\,\beta\,|\,\alpha\,] \quad = \quad \left(\prod_{j\,\in\,\beta} r_j^{\langle 1\rangle}\right) \quad p(\beta,\alpha) \quad \left(\prod_{j\,\in\,\alpha} r_j^{\langle n\rangle}\right)$$

where $p(\beta,\alpha) = det\ P[\,\beta\,|\,\alpha\,]$ is a polynomial not involving the random variables $r_j^{\langle 1\rangle}$ or $r_j^{\langle n\rangle}$. So, $det\ U^{\langle n\rangle}[\,\beta\,|\,\alpha\,]$ is nonzero iff $p(\beta,\alpha)$ is nonzero, and if it is nonzero then substituting any other sequences λ, μ for β, α will yield a function of a different set of random variables.

Third, if α is a *consecutive* sequence, we claim $det\,U^{\langle n\rangle}[\,\beta\,|\,\alpha\,] \neq 0$, for any given β. Continuing the factorization, $(U_0^{\langle n/2\rangle} \oplus U_1^{\langle n/2\rangle}) = R^{\langle 1\rangle}\,Q\,R^{\langle n/2\rangle}$ where $Q\,R^{\langle n/2\rangle}\,O_n = P$, so with Theorem 2

$$det\ U^{\langle n\rangle}[\,\beta\,|\,\alpha\,] \quad = \quad \sum_{\gamma\in S_k^n} \left(\prod_{j\,\in\,\beta} r_j^{\langle 1\rangle}\right) \quad q(\beta,\gamma) \quad \left(\prod_{j\,\in\,\gamma} r_j^{\langle n/2\rangle}\right) \quad c(n,\gamma,\alpha) \quad \left(\prod_{j\,\in\,\gamma} r_j^{\langle n\rangle}\right)$$

with $q(\beta,\gamma) = det\,Q[\,\beta\,|\,\gamma\,]$. The contribution to this sum by any fixed γ is thus either zero, or a function of a different set of random variables than that for any other. All that is needed to complete the proof, then, is to find a value of γ for which both $q(\beta,\gamma)$ and $c(n,\gamma,\alpha)$ are nonzero, since no other value can affect its contribution to the sum.

If β_0 and β_1 are subsequences of $1, \ldots, n/2$ and $(n/2{+}1), \ldots, n$ whose sequence concatenation is β, it is sufficient to show that there are always *consecutive* subsequences γ_0, γ_1 of $1, \ldots, n/2$ and $(n/2+1), \ldots, n$ such that γ is their concatenation, $c(n,\gamma,\alpha) \neq 0$, and also the length of γ_0 equals the length of β_0 and the length of γ_1 equals the length of β_1. Let m and $k-m$ be the respective lengths of β_0 and β_1. If $\alpha \equiv (i+1), \ldots, (i+k)$ (Mod n), let

$$\begin{aligned}
\gamma_0 &\equiv (i+1), (i+2), \ldots, (i+m) \quad (\mathrm{Mod}\,n/2)\\
\gamma_1 &\equiv (i+m+1), (i+m+2), \ldots, (i+k) \quad (\mathrm{Mod}\,n/2) \quad + \quad n/2.
\end{aligned}$$

Since $m \le (n/2)$ and $(k-m) \le (n/2)$, both γ_0 and γ_1 are (end-around) consecutive. Furthermore $\gamma \equiv \alpha$ Mod $(n/2)$. By Theorem 2, $c(n,\gamma,\alpha) \neq 0$. By induction each of the determinants $det\,U_0^{\langle n/2\rangle}[\,\beta_0\,|\,\gamma_0\,]$, $det\,U_1^{\langle n/2\rangle}[\,\beta_1 - n/2\,|\,\gamma_1 - n/2\,]$ is a nonzero polynomial, since γ_0 and γ_1 are each consecutive. Thus $q(\beta,\gamma) \neq 0$ for this particular γ. Altogether, when α is a consecutive sequence, $det\,U^{\langle n\rangle}[\,\beta\,|\,\alpha\,] \neq 0$. □

Example 1 *Consider the matrix $U^{\langle 8\rangle}$, extending the example preceding Lemma 2.*
If α is a consecutive sequence, then always $det\,U^{\langle 8\rangle}[\,\beta\,|\,\alpha\,] \neq 0$. However, if α is an arbitrary sequence in S_k^n, it is possible that $det\ U^{\langle n\rangle}[\,\beta\,|\,\alpha\,] = 0$. As an example, $det\ U^{\langle 8\rangle}[\,1,2\,|\,1,5\,] = 0$. In fact, many such sequences β and α will give a determinant that is identically zero. The table below gives the number of such sequences for $U^{\langle 8\rangle}$:

k	$\lvert\{\, (\beta,\alpha) \mid \beta,\ \alpha \in S_k^8 \,\}\rvert$	$\lvert\{\, (\beta,\alpha) \mid \beta,\ \alpha \in S_k^8,\ \ det\, U^{\langle 8\rangle}[\,\beta \mid \alpha\,] = 0 \,\}\rvert$
1	64	0
2	784	80
3	3136	320
4	4900	468
5	3136	320
6	784	80
7	64	0

5.4 The RBT yields Block Nondegenerate Matrices

Definition 7

An $n \times n$ matrix A is **block nondegenerate** *if for $1 \le k \le n$, and for every sequence of consecutive integers $\alpha \in CS_k^n$, the block principal submatrix $A[\,\alpha \mid \alpha\,]$ is nonsingular.*

Theorem 4 *Let A be an $n \times n$ nonsingular matrix, where n is a power of 2. If U and V are random recursive butterflies, then, with probability 1, the RBT $\widetilde{A} = U^* A V$ is block nondegenerate.*

Proof

Assume $\widetilde{A}$ is block degenerate, so that for some k and some consecutive sequence $\alpha \in CS_k^n$, $det\ \widetilde{A}[\,\alpha \mid \alpha\,] = 0$. However, using the Binet-Cauchy theorem again, the determinant of $\widetilde{A} = U^*AV$ satisfies

$$\begin{aligned} det\ \widetilde{A}[\,\alpha \mid \alpha\,] &= \sum_{\kappa \in S_k^n} \sum_{\lambda \in S_k^n} det\ U^*[\,\alpha \mid \kappa\,]\ \ det\ A[\,\kappa \mid \lambda\,]\ \ det\ V[\,\lambda \mid \alpha\,] \\ &= \sum_{\kappa \in S_k^n} \sum_{\lambda \in S_k^n} \overline{det\ U[\,\kappa \mid \alpha\,]}\ \ det\ A[\,\kappa \mid \lambda\,]\ \ det\ V[\,\lambda \mid \alpha\,] \end{aligned}$$

where the overbar denotes complex conjugation. By Theorem 3, because α is a consecutive sequence, $det\, U[\,\kappa \mid \alpha\,]$ is a nonzero polynomial of degree at most one in the butterfly random variables, and is dependent upon a different set of random variables from that for any other κ in the sum. Similarly $det\, V[\,\lambda \mid \alpha\,]$ is also nonzero, and is dependent upon a different set of random variables from that for any other λ in the sum. Thus $det\ \widetilde{A}[\,\alpha \mid \alpha\,]$ is a polynomial of degree one in these random variables, and for each coefficient $det\ A[\,\kappa \mid \lambda\,]$ the variable polynomial $\overline{det\ U[\,\kappa \mid \alpha\,]}\ det\ V[\,\lambda \mid \alpha\,]$ is nonzero and is dependent upon a different set of random variables from that for any other κ and λ in the sum.

Since by assumption $det\ \widetilde{A}[\,\alpha \mid \alpha\,] = 0$, Theorem 1 now allows us to infer that, with probability 1, $det\ A[\,\kappa \mid \lambda\,] = 0$ for every choice of κ and λ in S_k^n. However, for any sequence $\kappa \in S_k^n$ the **general Laplace expansion theorem for determinants** [13, p.14] states that

$$det\ A \;=\; (-1)^{sum(\kappa)} \sum_{\lambda \in S_k^n} (-1)^{sum(\lambda)}\ \ det\ A[\,\kappa \mid \lambda\,]\ \ det\ A[\,\kappa' \mid \lambda'\,]$$

where κ' is the (strictly increasing) complement of κ among indices of A, and $sum(\kappa) = \sum_i \kappa_i$. Therefore the assumption implies, with probability 1, that the right side of this equation is zero. This gives a contradiction since A is nonsingular. So, with probability 1, the assumption that $\widetilde{A}$ is not block nondegenerate is false — i.e., with probability 1, $\widetilde{A}$ is block nondegenerate. □

6 Gaussian Elimination with the RBT

We can now show that the RBT can be used to provide Gaussian Elimination without pivoting. It is well-known that Gaussian Elimination without pivoting is unstable. It is natural to ask whether this instability will persist with the randomization methods proposed in this paper.

6.1 The RBT yields Gauss Eliminable Matrices

Gaussian elimination without pivoting succeeds on a matrix only when the computed diagonal entries are all nonzero. It is known (e.g., [16]) that this is equivalent to the following:

Definition 8 *An $n \times n$ matrix A is* **Gauss eliminable** *if* $det\, A[1,\ldots,k \,|\, 1,\ldots,k] \neq 0$, *for* $1 \leq k \leq n$.

Theorem 5 *The RBT yields Gauss eliminable matrices, i.e.: with probability 1, Gaussian elimination can operate without any pivoting on the result $\widetilde{A} = U^* A V$ of the RBT.*

Proof
Block nondegeneracy implies Gauss eliminability, so by Theorem 4 the result $\widetilde{A}$ of an RBT is always Gauss eliminable. This idea is explored in somewhat more detail in [18], and with block algorithms specifically in mind in [19]. □

6.2 Error Analysis of Gaussian Elimination

Classical error analysis as in [14] and [5] determines that Gaussian Elimination computes triangular matrices $\widehat{L}$, $\widehat{U}$ such that $A = \widehat{L}\widehat{U} + E$, where $E = \sum_{k=1}^{n-1} E^{(k)}$ is a matrix of error values such that $E^{(k)} = (\epsilon_{ij}^{(k)})$ and

$$|\epsilon_{ij}^{(k)}| \quad = \quad \begin{cases} |\widehat{a}_{ij}^{(k)}|\,\mathbf{u} & i \geq k+1,\ j = k \\ |\widehat{a}_{ij}^{(k+1)}|\ 2\mathbf{u}\ +\ |\widehat{a}_{ij}^{(k)}|\ \mathbf{u} & i \geq k+1,\ j \geq k+1 \\ 0 & \text{otherwise} \end{cases}$$

where $\mathbf{u}$ is the unit roundoff. The Gaussian elimination algorithm itself repeatedly transforms the input matrix $A = A^{(1)}$ into iterates $A^{(k+1)}$, $1 \leq k \leq n-1$, ultimately giving $U = A^{(n)}$. Any bound on $|\widehat{a}_{ij}^{(k+1)}|$ will therefore give a bound on the error $\| E \|$. Such a bound is usually swept under a carpet called the *growth factor*

$$\rho(A) \quad = \quad \frac{\max_{ijk}\ |\widehat{a}_{ij}^{(k)}|}{\| A \|}.$$

Weak bounds are known for the growth factor for both partial pivoting and complete pivoting, and for partial pivoting these grow exponentially in the rank of A. In practice, nevertheless, pivoting usually keeps Gaussian elimination stable, and Gaussian elimination without pivoting is unstable.

Typically we can estimate $\widehat{a}_{ij}^{(k)}$ with $a_{ij}^{(k)}$, and for this an explicit formula is possible:

Theorem 6 *If A is a Gauss eliminable $n \times n$ matrix, then:*
For $1 \leq k < i \leq n$, $1 \leq k < j \leq n$,

$$a_{ij}^{(k+1)} \quad = \quad \frac{det\ A[1,\ldots,k,\,i \,|\, 1,\ldots,k,\,j]}{det\ A[1,\ldots,k \,|\, 1,\ldots,k]}.$$

Proof
These underappreciated formulas are rederived in the recent note [16]. □

These formulas are illuminating since they expose the effect on errors of the principal leading $k \times k$ submatrices of A. They lead to the following tight bound [16]:

Theorem 7 *For $1 \le k < i \le n$, $1 \le k < j \le n$,*

$$a_{ij}^{(k+1)} \;=\; a_{ij} \;-\; A[\,i \,|\, 1,\ldots,k\,]\,(A[\,1,\ldots,k \,|\, 1,\ldots,k\,])^{-1}\,A[\,1,\ldots,k \,|\, j\,].$$

Therefore

$$|\,a_{ij}^{(k+1)}\,| \;\le\; |\,a_{ij}\,| \;+\; \kappa_2(A[\,1,\ldots,k \,|\, 1,\ldots,k\,])\,\frac{\|\,A[\,i \,|\, 1,\ldots,k\,]\,\|_2\;\|\,A[\,1,\ldots,k \,|\, j\,]\,\|_2}{\|\,A[\,1,\ldots,k \,|\, 1,\ldots,k\,]\,\|_2}.$$

These formulas give error bounds for Gaussian elimination in terms $\kappa_2(\widetilde{A}[\,1,\ldots,k \,|\, 1,\ldots,k\,])$. Expected values for these condition numbers are available for random matrices by Edelman [3]. However, these condition numbers are usually not known for a given problem. For more, see [16].

6.3 Errors with the RBT

The formulas above do give a way to estimate the effect of the RBT on the submatrix condition numbers $\kappa_2(\widetilde{A}[\,1,\ldots,k \,|\, 1,\ldots,k\,])$, however. Gaussian Elimination on $\widetilde{A}$ computes

$$\widetilde{a}_{ij}^{(k+1)} \;=\; \frac{det\;\widetilde{A}[\,1,\ldots,k,\,i \,|\, 1,\ldots,k,\,j\,]}{det\;\widetilde{A}[\,1,\ldots,k \,|\, 1,\ldots,k\,]}.$$

Recall the *Binet-Cauchy theorem* used in Theorem 3 above states that for square $n \times n$ matrices X, Y, any integer k such that $1 \le k \le n$, and any sequences κ, λ in S_k^n,

$$det\;((XY)[\,\kappa \,|\, \lambda\,]) \;=\; \sum_{\mu \in S_k^n} det\;X[\,\kappa \,|\, \mu\,]\;\;det\;Y[\,\mu \,|\, \lambda\,]$$

where S_k^n is the set of ascending sequences of k numbers in $\{1,\ldots,n\}$. Thus, for $\widetilde{A} = U^*AV$,

$$\widetilde{a}_{ij}^{(k+1)} \;=\; \frac{\sum_{\kappa,\lambda \,\in\, S_{k+1}^n} det\;U^*[\,1,\ldots,k,\,i \,|\, \kappa\,]\;det\;A[\,\kappa \,|\, \lambda\,]\;det\;V[\,\lambda \,|\, 1,\ldots,k,\,j\,]}{\sum_{\mu,\nu \,\in\, S_k^n} det\;U^*[\,1,\ldots,k \,|\, \mu\,]\;det\;A[\,\mu \,|\, \nu\,]\;det\;V[\,\nu \,|\, 1,\ldots,k\,].}$$

This gives sort of an 'average of growth factors' that would be obtained over all possible pivoting sequences (good and bad) with the original matrix. Each sum involves determinants of submatrices of A multiplied by a product of entries in the random matrices U^* and V. Assuming that both:

- the random entries in U, V are distributed so that the products have nonzero expected value
- the determinants of submatrices of a matrix follow a Gaussian distribution (as is true in other contexts [6])

then the expected value for $\widetilde{a}_{ij}^{(k+1)}$ can be estimated. This analysis shows that the RBT does modify the submatrices of the input matrix, and can modify its growth factor, but in general more information about the entries is needed to determine whether the growth factor will increase.

The formula for $\widetilde{a}_{ij}^{(k+1)}$ can be made more explicit using results about determinants of submatrices of U^* and V such as Theorems 2 and 3, or the explicit formulas for determinants in [16]. With these we can obtain a precise formula for the 'averaging' of growth factors.

7 Implementation

It is easy to test whether $\widetilde{A}$ is Gauss eliminable (Theorem 5) with an implementation of Gaussian Elimination that uses no pivoting. Many implementations are possible. As a result the effort here probably raises more questions than it answers, but it does validate the basic approach.

7.1 Example

For simplicity, let us consider an example in which we use the RBT. In this example we have used four-digit arithmetic in base 10. We have also used real-valued RBTs (with nonsingular random diagonal matrices with expected determinant 1) rather than more general unitary RBTs since the behavior of the algorithm is much clearer. Consider the matrix and vector

$$A \;=\; \begin{pmatrix} 0 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \qquad \mathbf{b} \;=\; \begin{pmatrix} 1 \\ 2 \\ 3 \\ 2 \end{pmatrix}.$$

Using the butterfly matrices

$$U^{\langle 1\rangle} \;=\; \begin{pmatrix} .8872 & 0 & 0 & 0 \\ 0 & .7445 & 0 & 0 \\ 0 & 0 & .7727 & 0 \\ 0 & 0 & 0 & .9585 \end{pmatrix} \qquad V^{\langle 1\rangle} \;=\; \begin{pmatrix} 1.632 & 0 & 0 & 0 \\ 0 & .4677 & 0 & 0 \\ 0 & 0 & .5073 & 0 \\ 0 & 0 & 0 & 2.137 \end{pmatrix}$$

$$U^{\langle 2\rangle} \;=\; \begin{pmatrix} 1.101 & 1.502 & 0 & 0 \\ 1.101 & -1.502 & 0 & 0 \\ 0 & 0 & .4623 & 1.444 \\ 0 & 0 & .4623 & -1.444 \end{pmatrix} \qquad V^{\langle 2\rangle} \;=\; \begin{pmatrix} 1.675 & 0.9265 & 0 & 0 \\ 1.675 & -0.9265 & 0 & 0 \\ 0 & 0 & 1.268 & 2.001 \\ 0 & 0 & 1.268 & -2.001 \end{pmatrix}$$

$$U^{\langle 4\rangle} \;=\; \begin{pmatrix} 1.188 & 0 & 0.6729 & 0 \\ 0 & 1.499 & 0 & 0.7312 \\ 1.188 & 0 & -0.6729 & 0 \\ 0 & 1.499 & 0 & -0.7312 \end{pmatrix} \qquad V^{\langle 4\rangle} \;=\; \begin{pmatrix} 2.104 & 0 & 0.5667 & 0 \\ 0 & 0.5583 & 0 & 0.8901 \\ 2.104 & 0 & -0.5667 & 0 \\ 0 & 0.5583 & 0 & -0.8901 \end{pmatrix}$$

we can define the recursive butterflies $U = U^{\langle 1\rangle}U^{\langle 2\rangle}U^{\langle 4\rangle}$, $V = V^{\langle 1\rangle}V^{\langle 2\rangle}V^{\langle 4\rangle}$. We can then use the RBT to form the revised system of equations $U^*\,A\,V\,\mathbf{y} \;=\; U^*\,\mathbf{b}$. This gives the Gauss eliminable transformed system

$$\begin{pmatrix} 20.68 & 0.085 & 1.869 & 3.887 \\ 0.77 & -13.83 & -7.333 & 10.19 \\ 4.632 & 2.122 & 2.541 & -0.735 \\ -18.19 & 2.588 & -0.027 & -8.649 \end{pmatrix} \mathbf{y} \;=\; \begin{pmatrix} 5.434 \\ -.485 \\ .4426 \\ -1.084 \end{pmatrix}.$$

Naked Gaussian Elimination (without any pivoting) finds that this linear system has solution

$$\mathbf{y} \;=\; \begin{pmatrix} .5653 \\ -.2578 \\ -.9704 \\ -1.138 \end{pmatrix}, \qquad \mathbf{x} \;=\; V\,\mathbf{y} \;=\; \begin{pmatrix} -0.001 \\ 1.002 \\ 2.001 \\ 0.998 \end{pmatrix},$$

which is correct to three digits.

This matrix A has the low condition number $\kappa_2(A) \approx 2.618$, and the results are fairly accurate. On poorly-conditioned matrices, the results will be less accurate. For example, if we have instead

$$A \;=\; \begin{pmatrix} 0 & 1 & 0 & 0 \\ 100 & 0 & 1 & 0 \\ 0 & 1 & 0 & 100 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

then $\kappa_2(A) = 100$, and four-digit arithmetic using the same U and V above finds

$$\mathbf{y} \quad = \quad \begin{pmatrix} .5132 \\ -.1404 \\ -.8063 \\ -1.177 \end{pmatrix}, \qquad \mathbf{x} \quad = \quad V\,\mathbf{y} \quad = \quad \begin{pmatrix} 0.001 \\ .9758 \\ 1.972 \\ 0.019 \end{pmatrix}, \qquad \text{exact solution} \quad = \quad \begin{pmatrix} 0 \\ 1 \\ 2 \\ .02 \end{pmatrix}$$

which is correct only to one digit. We have had some success with matrix scaling in situations like this. Also, intelligent choice of random matrices can make a difference. However, the RBT can give sizeable roundoff erorrs with an ill-conditioned input matrix.

7.2 Experiments

To test the concepts presented above, we implemented the RBT in FORTRAN 77, and compared a randomized version of Gaussian Elimination using it with the standard LINPACK `dgefa/dgesl` routines on a Sun Sparcstation 20 at UCLA. Results are tabulated in Figure 1 for 11 different classes of matrices, including five random classes in [25], and six classes that have known exact solutions (the first 3 of which appear in [11]):

Matrix class	Problem: $A\mathbf{x} = \mathbf{b}$	Exact solution $\mathbf{x}$
`normal`	normal random a_{ij} and b_i ($\mu = 0,\ \sigma = 1$)	
`[-1,1]`	uniform random a_{ij} and b_i, drawn from $[-1, 1]$	
`[0,1]`	uniform random a_{ij} and b_i, drawn from $[0, 1]$	
`{-1,1}`	discrete random a_{ij} and b_i ($pr(-1) = pr(1) = 0.5$)	
`{0,1}`	discrete random a_{ij} and b_i ($pr(0) = pr(1) = 0.5$)	
`|i-j|`	$a_{ij} = \|i - j\|, \quad b_i = 1$	$\mathbf{x} = (\frac{1}{n-1}, 0, \ldots, 0, \frac{1}{n-1})$
`max(i,j)`	$a_{ij} = max(i, j), \quad b_i = i$	$\mathbf{x} = (1, 0, \ldots, 0)$
`C(i+j,j)`	$a_{ij} = \binom{i+j\ -2}{j\ -1}, \quad b_i = 1$	$\mathbf{x} = (1, 0, \ldots, 0)$
`Hadamard`	$A = \bigotimes^{\log_2(n)} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad b_i = 1$	$\mathbf{x} = (1, 0, \ldots, 0)$
`Permute`	A is a random permutation, $\quad (A^{-1}b)_i = i$	$\mathbf{x} = (1, 2, 3, \ldots, n)$
`Turing`	$a_{ij} = \left\{ \begin{array}{cc} 0 & i < j \\ 1 & i = j \\ -1 & i > j \end{array} \right\}, \quad b_i = 1 \quad$ (Wilkinson's test)	$\mathbf{x} = (1, 2, 4, \ldots, 2^{n-1})$

The current implementation is straightforward. It is a sequential program, and does not exploit the parallelism of the algorithm. Also, it is not optimized. For example, Brad Pierce has noticed a clever way to reduce the time required by the RBT by a factor of 2, and we will report on this soon. The overhead of random number generation was significant, and dominated the execution times, so we suspect it will be a factor in any future implementation. The current implementation generates an initial set of RBT random values for each value of n, and reuses these in all runs for n, as a simple-minded way of avoiding the overhead. The RBT uses random diagonal matrices with diagonal values $exp(\ r/10\ )$ where the uniform random variables r range over $[-1/2, 1/2]$. The determinant of these butterflies thus has expected value 1. The width of this interval was made a parameter that could be varied, but moderate variations in the width have not affected the roundoff error greatly.

Matrix Class	Matrix Rank	Good Runs	RBT avg sec	LINPACK avg sec	avg Error RBT-LINPACK	max Error RBT-LINPACK	avg Error RBT-exact	max Error RBT-exact
normal	32	256	0.0754	0.0621	1.66D-12	6.97D-10		
[-1,1]	32	256	0.0708	0.0617	1.64D-12	2.71D-10		
[0,1]	32	256	0.0709	0.0613	3.51D-12	7.46D-10		
{-1,1}	32	256	0.0708	0.0538	1.73D-12	1.93D-10		
{0,1}	32	256	0.0707	0.0532	2.30D-12	2.67D-10		
|i-j|	32	256	0.0707	0.0625	5.84D-15	2.47D-14	5.45D-15	2.58D-14
max(i,j)	32	256	0.0709	0.0589	8.73D-15	3.56D-14	8.73D-15	3.56D-14
C(i+j,j)	32	256	0.0708	0.0581	1.76D-13	5.07D-13	2.73D-02	8.75D-01
Hadamard	32	256	0.0709	0.0303	4.13D-16	1.16D-15	4.13D-16	1.16D-15
Permute	32	256	0.0708	0.0171	4.24D-13	1.13D-10	4.24D-13	1.13D-10
Turing	32	256	0.0708	0.0141	6.30D-09	1.01D-07	6.30D-09	1.01D-07
normal	64	128	0.5040	0.2862	3.01D-12	8.83D-10		
[-1,1]	64	128	0.4971	0.2867	3.89D-12	4.79D-10		
[0,1]	64	128	0.4965	0.2864	3.23D-12	3.44D-10		
{-1,1}	64	128	0.4966	0.2489	1.77D-11	5.65D-09		
{0,1}	64	128	0.4970	0.2516	2.63D-12	2.78D-10		
|i-j|	64	128	0.4950	0.2848	2.87D-14	1.08D-13	2.70D-14	9.45D-14
max(i,j)	64	128	0.4965	0.2764	3.36D-15	1.03D-14	3.36D-15	1.03D-14
C(i+j,j)	64	128	0.4975	0.2766	2.27D-12	2.64D-11	1.37D-02	8.75D-01
Hadamard	64	128	0.4966	0.1102	2.46D-16	1.73D-15	2.46D-16	1.73D-15
Permute	64	128	0.4973	0.0503	5.21D-13	3.60D-11	5.21D-13	3.60D-11
Turing	64	128	0.4958	0.0437	4.39D+01	1.41D+03	4.39D+01	1.41D+03
normal	128	64	2.9080	1.8112	4.74D-11	1.09D-08		
[-1,1]	128	64	2.8932	1.8122	5.54D-11	1.08D-08		
[0,1]	128	64	2.8891	1.8098	7.54D-12	8.26D-10		
{-1,1}	128	64	2.8935	1.6784	4.25D-12	1.38D-10		
{0,1}	128	64	2.8845	1.6769	7.84D-12	3.85D-10		
|i-j|	128	64	2.8944	1.8115	2.06D-13	9.48D-13	2.07D-13	9.90D-13
max(i,j)	128	64	2.8942	1.7758	5.25D-13	3.20D-12	5.25D-13	3.20D-12
C(i+j,j)	128	64	2.8902	1.7779	2.04D-07	2.38D-06	6.84D-03	8.75D-01
Hadamard	128	64	2.8863	0.5464	3.69D-15	3.62D-14	3.69D-15	3.62D-14
Permute	128	64	2.8897	0.1885	1.83D-12	1.16D-10	1.83D-12	1.16D-10
Turing	128	64	2.8891	0.1609	2.79D+21	1.79D+23	2.79D+21	1.79D+23
normal	256	32	26.1787	15.8590	2.81D-11	1.07D-09		
[-1,1]	256	32	26.1590	15.8459	3.97D-10	1.97D-08		
[0,1]	256	32	26.1903	15.8391	9.78D-11	7.80D-09		
{-1,1}	256	32	26.1119	15.1486	4.17D-11	1.38D-09		
{0,1}	256	32	26.2068	15.2045	3.33D-11	1.96D-09		
|i-j|	256	32	26.1261	15.8444	2.70D-12	1.83D-11	2.70D-12	1.83D-11
max(i,j)	256	32	26.0597	15.6487	3.11D-13	1.65D-12	3.11D-13	1.65D-12
C(i+j,j)	256	32	26.1756	15.6780	1.34D-01	2.05D+01	2.05D-01	3.85D+01
Hadamard	256	32	26.1069	3.5442	1.01D-13	1.80D-12	1.01D-13	1.80D-12
Permute	256	32	26.2584	0.9486	1.50D-11	2.76D-09	1.50D-11	2.76D-09
Turing	256	32	26.0673	0.7355	6.92D+59	8.85D+61	6.92D+59	8.85D+61
normal	512	20	248.0461	173.0853	4.75D-10	1.67D-08		
[-1,1]	512	20	248.1719	173.0854	2.14D-10	7.56D-09		
[0,1]	512	20	248.1133	173.0481	7.41D-10	2.96D-08		
{-1,1}	512	20	248.1001	169.9761	3.29D-10	7.82D-09		
{0,1}	512	20	248.0032	169.6792	7.32D-10	4.15D-08		
|i-j|	512	20	248.0747	173.4006	3.84D-13	2.29D-12	2.27D-13	2.14D-12
max(i,j)	512	20	248.3359	172.6650	4.67D-14	4.76D-13	4.67D-14	4.76D-13
C(i+j,j)	512	20	247.9565	172.8687	1.30D+17	6.66D+19	2.44D+17	1.25D+20
Hadamard	512	20	248.7100	24.2266	8.06D-14	4.42D-12	8.06D-14	4.42D-12
Permute	512	20	248.4258	5.5688	1.90D-10	1.53D-08	1.90D-10	1.53D-08
Turing	512	20	248.5327	3.6685	2.20+137	5.64+139	2.20+137	5.64+139

Figure 1: Results of simple implementation: RBT vs. LINPACK's `dgefa/dgesl`

Rounding errors do grow rapidly with the problem size n. The final columns in the table give averages and maxima of 'relative errors'

$$\frac{| b_i^{RBT} - b_i |}{\| \mathbf{b}^{RBT} \|_\infty}$$

where the b_i values are either the computed solution from LINPACK, or the exact solution. For poorly-conditioned problems the RBT has large roundoff errors, which are somewhat worse than those of LINPACK. The ill-conditioned problems `C(i+j,j)` and `Turing` have respective condition numbers 2^{4n} (spectral norm) and $n\,2^{n-1}$ (row sum norm). Edelman [3] has shown that normally distributed random matrices are well-behaved, with expected spectral norm essentially $\sqrt{2n}$ (actually this has been known for quite some time [26, p.299]). Although the errors were not as bad as we feared, they do suggest the RBT approach as implemented here should be limited to well-behaved problems.

In spite of the fact that the code is not optimized or refined, the results show that the approach is already usable, and for the test matrices used often gives numerical results comparable in quality to that of LINPACK's `dgefa/dgesl`.

8 Concluding Remarks

The Random Butterfly Transformation (RBT) pre- and post-multiplies a given matrix with nonsingular random matrices. In this paper we have studied RBTs based on butterfly and recursive butterfly matrices that resemble the Fast Fourier Transform (FFT). We noted that RBTs have three important benefits. First, RBTs have provable randomization properties. Second, they can be performed efficiently, and in particular should be useful on many parallel and vector machines with architectures that support FFT-like computations. Third, RBTs have useful numerical properties, if restricted to unitary matrices: many important matrix norms are invariant under unitary transformations, and therefore condition numbers with respect to these norms are also invariant. Non-unitary RBTs can be constructed that also have similar numerical properties.

The emphasis of the paper has been on describing how RBTs can be used to transform nonsingular matrices into nondegenerate matrices (with probability 1). This transformation has various applications in computational linear algebra. A noteworthy possibility is the development of new Gaussian elimination algorithms that operate without pivoting. The RBT appears to offer new strategies for linear systems solution that are well-suited to high-performance computers.

Putting the results here in perspective, we have shown that by paying a bit in preprocessing with certain random matrices, we can buy out of the need for pivoting during Gaussian elimination. Furthermore, although the randomized approach certainly is adversely affected by ill-conditioned input matrices, and is apparently less stable than partial pivoting, it performs well for random test matrices in our test suite. A complete error analysis and more implementation experience are still needed, but these preliminary results are encouraging. We are optimistic that with further experience, mature variations on the RBT will find useful places in numerical libraries.

Acknowledgement

I am indebted to Brad Pierce for several very careful readings of the paper, and many very clever suggestions for improving it. Brad also mentioned it is intriguing that, according to some researchers (e.g., [8]), Gauss should be credited with inventing the FFT algorithm.

References

[1] E. Bampis, J.C. Konig, D. Trystram, "Impact of communications on the complexity of the Parallel Gaussian Elimination", *Parallel Computing* **17**:1, 55–61, 1991.

[2] J.R. Bunch, J. Hopcroft, "Triangular Factorization and Inversion by Fast Matrix Multiplication", *Math. of Computation* **28**:125, 231–236, 1974.

[3] A. Edelman, "Eigenvalues and Condition Numbers of Random Matrices", *SIAM J. Matrix Anal. Appl.* **9**:4, 543–560, 1988.

[4] D.K. Fadeev, V.N. Fadeeva, *Computational Methods of Linear Algebra*, San Francisco, W.H. Freeman, 1963.

[5] G.E. Forsythe, C.B. Moler, *Computer Solution of Linear Algebraic Systems*, Prentice-Hall, 1967.

[6] V.L. Girko, *Theory of Random Determinants*, Boston: Kluwer Academic Publishers, 1990.

[7] G.H. Golub, C.F. Van Loan, *Matrix Computations: Second Edition*, Baltimore: Johns Hopkins University Press, 1989.

[8] M.T. Heideman, D.H. Johnson, C.S. Burrus, "Gauss and the history of the Fast Fourier Transform" *IEEE ASSP Magazine* **1**:4, 14–21, October 1984.

[9] R.A. Horn, C.R. Johnson, *Matrix Analysis*, NY: Cambridge University Press, 1985; reprinted 1992.

[10] R.A. Horn, C.R. Johnson, *Topics in Matrix Analysis*, NY: Cambridge University Press, 1991.

[11] R.T. Gregory, D.L. Karney, *A Collection of Matrices for Testing Computational Algorithms*, Wiley-Interscience, 1969.

[12] A.W. Marshall, I. Olkin, *Inequalities: Theory of Majorization and Its Applications*, NY: Academic Press, 1979.

[13] M. Marcus, H. Minc, *A Survey of Matrix Theory and Matrix Inequalities*, NY: Dover, 1964.

[14] J. von Neumann, H.H. Goldstine, "Numerical inverting of matrices of high order", *Bull. Amer. Math. Soc.* **53**, 1021–1099, 1947.

[15] D.S. Parker, "Notes on Shuffle/Exchange-Type Switching Networks", *IEEE Trans. Comput.* **C-29:3**, 213-222 (March 1980).

[16] D.S. Parker, "Explicit Formulas for the Results of Gaussian Elimination", Technical Report CSD-950025, UCLA Computer Science Department, 1995.

[17] D.S. Parker, "A Randomizing Butterfly Transformation Useful in Block Matrix Computations", Technical Report CSD-950024, UCLA Computer Science Department, 1995.

[18] D.S. Parker, D. Lê, "How to Eliminate Pivoting from Gaussian Elimination — by Randomizing Instead", Technical Report CSD-950022, UCLA Computer Science Department, 1995.

[19] D.S. Parker, D. Lê, "Quadtree Matrix Algorithms Revisited: Basic Issues and their Resolution", Technical Report CSD-950028, UCLA Computer Science Department, 1995.

[20] G. Poole, L. Neal, "A Geometric Analysis of Gaussian Elimination. I", *Linear Algebra Appl.* **149**, 249–272, 1991.

[21] G. Poole, L. Neal, "Gaussian Elimination: When is Scaling Beneficial?", *Linear Algebra Appl.* **162**, 309–324, 1992.

[22] F.F. Rivera, R. Doallo, J.D. Bruguera, E.L. Zapata, R. Peskin, "Gaussian Elimination with Pivoting on Hypercubes", *Parallel Computing* **14**:1, 51–60, 1990.

[23] Y. Robert, *The Impact of Vector and Parallel Architectures on the Gaussian Elimination Algorithm*, NY: Halsted Press, 1990.

[24] G.W. Stewart, *Introduction to Matrix Computations*, Academic Press, 1973.

[25] L.N. Trefethen, R.S. Schreiber, "Average-Case Stability of Gaussian Elimination", *SIAM J. Matrix Anal. Appl.* **11**:3, 335–360, July 1990.

[26] A.M. Turing, "Rounding-Off Errors in Matrix Processes", *Quart. J. Mech.* **1**, 287–308, 1948.

[27] C. Van Loan, *Computational Frameworks for the Fast Fourier Transform*, Philadelphia, PA: SIAM Press, 1992.

[28] M. Veldhorst, "Gaussian Elimination with Partial Pivoting on an MIMD computer", *J. Parallel Distr. Computing* **6**, 62–68, 1989.

[29] J.H. Wilkinson, "Error analysis of direct methods of matrix inversion", *J. ACM* **8**, 281–330, 1961.