MATHEMATICS OF COMPUTATION
VOLUME 000, NUMBER 0
0000, PAGES 000–000

MATRIX INVERSION MADE DIFFICULT

DANIEL J. BERNSTEIN

ABSTRACT. If $ADA = A$ and $DAD = D$ then D is a *quasiinverse* of A. This paper presents *generalized Gaussian elimination*, which builds a quasiinverse of an arbitrary homomorphism out of quasiinverses of two smaller homomorphisms. In conjunction with fast matrix multiplication, generalized Gaussian elimination is a fast, reasonably stable method that can (1) invert any invertible matrix, (2) solve any soluble set of linear equations, and (3) compute a basis for the kernel of any matrix. A fast algorithm for (1) was already obtained by Bunch and Hopcroft, but the construction here is simpler. Fast algorithms for (2) and (3) appear to be new.

1. INTRODUCTION

Let $A : V \to W$ and $D : W \to V$ be module homomorphisms. Then D is a **quasiinverse** of A if $ADA = A$ and $DAD = D$.

If A has a left inverse then its quasiinverses are exactly its left inverses, and if A has a right inverse then its quasiinverses are exactly its right inverses. In particular, when A is invertible, its only quasiinverse is its inverse.

Any quasiinverse D determines a **coimage** of A in V, namely $\operatorname{im} D$, and a **cokernel** of A in W, namely $\ker D$. (It is tempting to say that D is $\operatorname{co} A$; D reverses all exact sequences related to A.) Then V splits as $(\ker A) \oplus (\operatorname{im} D)$ and W splits as $(\ker D) \oplus (\operatorname{im} A)$. Note that A is an isomorphism from $\operatorname{im} D$ to $\operatorname{im} A$, with D as its inverse.

A quasiinverse D **respects** a basis for W if $\ker D$ is generated by a subset of that basis. Similarly, D **respects** a basis for V if $\operatorname{im} D$ is generated by a subset of that basis. Note that the corresponding columns of A then form a basis for $\operatorname{im} A$.

Any matrix over a field has a quasiinverse respecting any given bases for V and W.

Outline of this paper. Section 2: **Generalized Gaussian elimination** constructs a quasiinverse of A given quasiinverses for a *pivot* and a *reduced map*.

Section 3: Given a matrix A over a field, Algorithm Q computes a respectful quasiinverse D of A, along with bases for $\ker D$ and $\operatorname{im} D$. Algorithm N computes a basis for $\ker A$. Algorithm S solves linear equations.

Section 4: The algorithms of section 3 run at essentially the speed of matrix multiplication.

Received by the editor 19951021 (draft 7).
1991 *Mathematics Subject Classification.* Primary 65F05, 15A09.
Thanks in advance to anyone who suffers through the first few drafts of this paper.

Fast matrix operations. Strassen [4] showed that one can multiply two n-by-n matrices in time n^β for a certain $\beta < 3$. (Currently the best known β is a bit below 2.38 [2].) For moderately large matrices Strassen's method is much faster than the usual algorithm.

Strassen then suggested block Gaussian elimination as a fast method to invert an n-by-n matrix. Unfortunately, this method can fail even if the original matrix is invertible. (It is also rather unstable.) Bunch and Hopcroft [1] fixed this problem with a tricky series of nested pivots; they proved that one can invert any invertible n-by-n matrix at roughly multiplication speed.

Generalized Gaussian elimination achieves the same result as the Bunch-Hopcroft method for invertible matrices, in a much less convoluted way. Strassen, Bunch, and Hopcroft chopped up A both horizontally and vertically; generalized Gaussian elimination chops up A in just one direction. This simplification depends crucially on the concept of a quasiinverse.

Generalized Gaussian elimination also produces useful results for non-invertible matrices. Examples: (1) it computes a basis for $\ker A$; (2) it points out a basis for $\operatorname{im} A$; (3) it solves systems of linear equations—if D is any quasiinverse of A, and $w \in \operatorname{im} A$, then $w = A(Dw)$.

Relation to Gaussian elimination. In section 2, S may be chosen to be one-dimensional. This special case is exactly **Gaussian elimination with partial pivoting** [XXX]. The latter algorithm is normally viewed as searching for a pivot element in (let's say) the first row of A, and then using the corresponding column to reduce A. I[1] prefer to view the entire first row as being the pivot. The inverse of the pivot element is really a quasiinverse of the pivot row. In generalized Gaussian elimination, the pivot may be any subset of the rows of A; the quasiinverse of the pivot amounts to the inverse of some maximal nonsingular square submatrix.

Quasiinverses in maximum generality. I have stated the results of section 2 for modules over any ring, not just vector spaces over a field, on the theory that unnecessary hypotheses distract attention from the essence of the proofs. For a more thorough application of this philosophy, see [3].

One could go even further, and rephrase section 2 in the language of abelian category theory; note that diagram theorems true for modules are true for all abelian categories [XXX]. I have no intention of carrying out this project.

Quasiinverses in the literature. It seems that there is no standard name for my concept of a quasiinverse. The word "quasiinverse" has been used [XXX] for a map D satisfying merely $ADA = A$. I suggest that such a map be called a "right quasiinverse" or "splitting" of A.

Any matrix over the real numbers has a unique **pseudoinverse**, a particular quasiinverse determined by the **singular value decomposition**. The singular value decomposition supplies orthonormal bases for $\operatorname{im} D$ and $\operatorname{im} A$ upon which A acts diagonally. See [XXX] for further details.

Another example of a quasiinverse is [XXX]; thanks to Henri Gillet and Lawrence Ein for bringing this to my attention.

Note that the construction of the reduced map C from the pivot B in Theorem 2 generalizes the construction of the Schur complement [XXX].

[1] I have abandoned the practice of mechanically replacing "I" with "we" in my papers.

Terminology. I abbreviate module homomorphism as **map**. Following convention and ignoring psychology I write maps on the left. $S \oplus T$ is the direct sum of S and T; it is the same as $S + T$, with the side condition that $S \cap T = \{0\}$. $\#Z$ is the number of elements of Z.

2. Generalized Gaussian elimination

Consider a map $A : V \to S \oplus T$. This section presents **generalized Gaussian elimination**, which builds a quasiinverse D of A out of (1) a quasiinverse E of a **pivot** $B : V \to S$ and (2) a quasiinverse F of a **reduced map** $C : V \to T$.

Theorem 1. *Let $A : V \to W$ and $\pi : W \to S$ be maps. Define $B : V \to S$ by $B = \pi A$. Assume that B has a quasiinverse $E : S \to V$. Define $P : V \to V$ by $P = 1 - EB$ and $Q : W \to W$ by $Q = 1 - AE\pi$. Then $PE = 0$, $BP = 0$, $PP = P$, $QA = AP$, and $\pi QA = 0$.*

Proof. First $PE = E - EBE = 0$; next $BP = B - BEB = 0$ so $PP = P - EBP = P$; next $QA = A - AE\pi A = A - AEB = AP$; finally $\pi QA = \pi AP = BP = 0$. □

Theorem 2. *In the situation of Theorem 1, assume that S is a submodule of W and that $\pi\pi = \pi$. Define $T = \ker \pi$. Define $C : V \to T$ by $C = QA$. Assume that C has a quasiinverse $F : T \to V$. Define $D : W \to V$ by $D = E\pi + PF(1 - \pi)Q$. Then D is a quasiinverse of A. Furthermore $\operatorname{im} D = \operatorname{im} E + \operatorname{im} F$ and $\ker D = \ker E + \ker F$.*

The assumptions on S and W mean that $W = S \oplus T$; π projects W onto S and $1 - \pi$ projects W onto T.

Proof. By Theorem 1, $CE = APE = 0$ and $CP = APP = AP = C$.

Now $DA = E\pi A + PF(1 - \pi)QA = EB + PFC$ so $ADA = AEB + APFC = AEB + CFC = AEB + C = A$.

Furthermore $BD = BE\pi + BPF(1 - \pi)Q = BE\pi$ since $BP = 0$; thus $EBD = EBE\pi = E\pi$. Also $CD = CE\pi + CPF(1 - \pi)Q = CF(1 - \pi)Q$ since $CE = 0$ and $CP = C$; thus $FCD = FCF(1 - \pi)Q = F(1 - \pi)Q$.

Hence $DAD = EBD + PFCD = E\pi + PF(1 - \pi)Q = D$.

Coimage: First $DAE = EBE + PFCE = E$ since $CE = 0$; thus $\operatorname{im} E \subseteq \operatorname{im} D$. Similarly $DAF = EBF + PFCF = EBF + PF = F$; thus $\operatorname{im} F \subseteq \operatorname{im} D$. Conversely $D = E\pi + PF(1-\pi)Q = E\pi + F(1-\pi)Q - EBF(1-\pi)Q$ so $\operatorname{im} D \subseteq \operatorname{im} E + \operatorname{im} F$.

Cokernel: If $s = \pi s$ and $Es = 0$ then $E\pi s = 0$ so $Qs = s$ so $Ds = E\pi s + PF(1 - \pi)Qs = PF(1 - \pi)s = 0$; thus $\ker E \subseteq \ker D$. If $\pi t = 0$ and $Ft = 0$ then $Qt = t$ so $Dt = E\pi t + PF(1 - \pi)Qt = PF(1 - \pi)t = PFt = 0$; thus $\ker F \subseteq \ker D$. Conversely, say $Dw = 0$. Then $E\pi w = EBDw = 0$, so $Qw = w$, so $F(1-\pi)w = F(1-\pi)Qw = FCDw = 0$. Thus $w = \pi w + (1-\pi)w \in \ker E + \ker F$. □

Theorem 3. *In the situation of Theorem 2, let M be a submodule of $\ker A$. Assume that $\ker C = (M + \operatorname{im} E) \oplus N$ for some module N. Then $\ker A = M \oplus PN$, and P is an isomorphism between N and PN.*

Proof. If $v \in N$ and $Pv \in M$ then $v = Pv + EBv \in M + \operatorname{im} E$ so $v = 0$. Thus (1) P is injective on N and (2) $M \cap PN = \{0\}$.

If $v \in N$ then $APv = Cv = 0$ so $Pv \in \ker A$. Hence $PN \subseteq \ker A$.

If $Av = 0$ then $Cv = QAv = 0$ so $v = m + Es + n$ for some $m \in M$, $s \in S$, and $n \in N$. Next $A(Es + n) = Av - Am = 0$, so $Es + n = P(Es + n) = Pn$, so $v = m + Pn \in M + PN$. Hence $\ker A \subseteq M + PN$. □

3. Computation

The following algorithms use generalized Gaussian elimination, recursively, to solve various problems in linear algebra over a field.

A vector space of dimension n is normally represented inside a computer by the integer n. Its basis vectors are represented by the integers $1, 2, \ldots, n$ (or $0, 1, \ldots, n-1$). A respectful subspace is represented as a list of basis vectors. Subspace sum and intersection then correspond to list union and intersection.

Say X is a basis for V, and Y is a basis for W. Recall that a **matrix** is a function a that assigns a field element to each pair $(y, x) \in Y \times X$. This matrix represents the map $A : V \to W$ defined by $A(x) = \sum_y a(y, x)y$. If $I \subseteq X$ and $J \subseteq Y$, the notation A on $J \times I$ means the submatrix a on $J \times I$, i.e., the function a restricted to $J \times I$. Note that a on $J \times I$ determines A, provided that (1) the span of $Y - J$ is contained in $\ker A$ and (2) $\operatorname{im} A$ is contained in the span of I.

In these algorithms, V and W are respectful finite-dimensional subspaces of some (irrelevant) larger space. A is a map from V to W. M is a respectful subspace of V, with $M \subseteq \ker A$. X is a basis for V minus a basis for M, Y is a basis for W, and $a = A$ on $Y \times X$; note that a determines A. X and Y are represented as finite lists of integers.

Computing a quasiinverse. The following algorithm computes a quasiinverse for A. It avoids storing matrix rows and columns that are guaranteed by construction to be entirely zero.

Algorithm Q. Given (X, Y, a), compute $Q(X, Y, a) = (I, J, D$ on $I \times J)$. Here $D : W \to V$ is a quasiinverse of A; I is a basis for $\operatorname{im} D$, with $I \subseteq X$; and $Y - J$ is a basis for $\ker D$, with $J \subseteq Y$.

1. If $a = 0$: Set d to the empty matrix. Return $(\{\}, \{\}, d)$.
2. If Y has just one element y: Select $v \in X$ such that $a(v, y) \neq 0$. Set d to the 1-by-1 matrix $1/a(v, y)$. Return $(\{v\}, \{y\}, d)$.
3. Select $Z \subset Y$, $Z \neq \{\}$. (In the language of section 2: Z is a basis for S, and $Y - Z$ is a basis for T.)
4. Extract $b = a$ on $Z \times X$. (b represents $B = \pi A$.)
5. Recursively compute $(I, J, e) = Q(X, Z, b)$. (e represents E.)
6. Extract $t_1 = a$ on $(Y - Z) \times I$.
7. Compute the product $t_2 = t_1 e$, a matrix on $(Y - Z) \times J$. (t_2 represents $(1 - \pi)AE$.)
8. Extract $t_3 = b$ on $J \times (X - I)$. (t_3 represents B off $\operatorname{im} E$.)
9. Compute the product $t_4 = t_2 t_3$, a matrix on $(Y - Z) \times (X - I)$. (t_4 represents $(1 - \pi)AEB$ off $\operatorname{im} E$.)
10. Extract $t_5 = a$ on $(Y - Z) \times (X - I)$. (t_5 represents $(1 - \pi)A$ off $\operatorname{im} E$.)
11. Compute $c = t_5 - t_4$, a matrix on $(Y - Z) \times (X - I)$. (c represents $C = (1 - \pi)A - (1 - \pi)AEB$.)
12. Recursively compute $(G, H, f) = Q(X - I, Y - Z, c)$. ($f$ represents F.)
13. Extract $t_6 = t_3$ on $J \times G$.
14. Compute the product $t_7 = t_6 f$, a matrix on $J \times H$. (t_7 represents BF.)
15. Compute the product $t_8 = e t_7$, a matrix on $I \times H$.
16. Compute $t_9 = t_8 - f$, a matrix on $(G \cup I) \times H$. (t_9 represents $-PF = EBF - F$.)
17. Extract $t_{10} = t_2$ on $H \times J$. (t_{10} represents the projection of $(1 - \pi)AE$ to the

coimage of F.)

18. Compute the product $t_{11} = t_9 t_{10}$, a matrix on $(G \cup I) \times J$. (t_{11} represents $-PF(1-\pi)AE$.)
19. Compute $d = e + t_{11} - t_9$, a matrix on $(G \cup I) \times (H \cup J)$. ($d$ represents $D = E\pi - PF(1-\pi)AE\pi + PF(1-\pi)$.)
20. Return $(G \cup I, H \cup J, d)$.

Note that the quasiinverse is square: if r is the rank of A, then $\#I = \#J = r$. Note also that it is easy to adapt Algorithm Q to compute the determinant of a maximal nonsingular submatrix of A.

Solving a system of linear equations. Fix $w \in W$. The following algorithm finds a solution z to the equation $Az = w$, if any solution exists. It is a straightforward optimization of the following procedure: first find a quasiinverse D of A by Algorithm Q; then compute $z = Dw$; then throw away D.

Algorithm S. Given (X, Y, a, w), compute $S(X, Y, a) = (I, Dw \text{ on } I)$, where D is some respectful quasiinverse of A and I is a basis for $\operatorname{im} D$.

1. If $a = 0$: Halt if $w \neq 0$; there is no solution. Otherwise set z to the empty vector and return $(\{\}, z)$.
2. If Y has just one element y: Select $v \in X$ such that $a(v, y) \neq 0$. Set z to the length-1 vector $(u/a(v, y))$, where $w = uy$. Return $(\{v\}, z)$.
3. Select $Z \subset Y$, $Z \neq \{\}$.
4. Extract $b = a$ on $Z \times X$.
5. Compute $(I, J, e) = Q(X, Z, b)$.
6. Compute $t_1, t_2, t_3, t_4, t_5, c$ as in Algorithm Q.
7. Compute $w_2 = w$ on J.
8. Compute $w_3 = w$ on $(Y - Z)$.
9. Compute $w_4 = t_2 w_2$, a vector on $Y - Z$.
10. Recursively compute $(G, z_1) = S(X - I, Y - Z, c, w_3 - w_4)$.
11. Extract $t_{12} = b$ on $J \times G$.
12. Compute $w_5 = t_{12} z_1$, a vector on J.
13. Compute $z_2 = e(w_2 - w_5)$, a vector on I.
14. Compute $z = z_1 + z_2$, a vector on $G \cup I$.
15. Return $(G \cup I, z)$.

Computing a basis for the nullspace. By assumption $\ker A$ contains a respectful subspace M of X. The following algorithm computes a (generally nonrespectful) subspace N of X such that $\ker A = M \oplus N$. It represents N as a matrix whose columns form a basis for N. Presumably we know a basis for M—for example, we could start with $M = \{0\}$—so we can compute a basis for $\ker A$.

Algorithm N. Given (X, Y, a, w), compute $N(X, Y, a)$.

1. If $a = 0$: Return the identity matrix on $X \times X$.
2. If Y has just one element y: Select $v \in X$ such that $a(v, y) \neq 0$. For each $x \in X$ with $x \neq v$, form the vector $x - (a(x, y)/a(v, y))v$. Return the matrix of these vectors.
3. Select $Z \subset Y$, $Z \neq \{\}$.
4. Extract $b = a$ on $Z \times X$.
5. Compute $(I, J, e) = Q(X, Z, b)$.
6. Compute $t_1, t_2, t_3, t_4, t_5, c$ as in Algorithm Q.

7. Recursively compute $N_1 = N(X - I, Y - Z, c)$.
8. Compute the product $N_2 = t_3 N_1$.
9. Compute the product $N_3 = eN_2$.
10. Return $N_1 - N_3$.

Numerical analysis. The following observations are well known for the special case of Gaussian elimination with partial pivoting.

Say A is a matrix over the real numbers. Each of the above algorithms appears to be numerically stable, provided that, in step 2, one selects the element largest in absolute value.

Inexact computations may lead to spurious failures when Algorithm S is applied to a singular system of linear equations. To avoid this one should treat sufficiently small numbers as zero in step 1. Another possibility is to skip the failure case entirely; w is thus projected (in some direction) onto $\operatorname{im} A$.

4. Speed

In this section I'll consider the speed of Algorithms Q, S, and N, from three points of view: (1) asymptotically, if we use fast matrix multiplication; (2) in the special case $\#Z = 1$; (3) in practice.

Preliminaries. Write $n = \#X$ and $m = \#Y$, so that a is an m-by-n matrix. In step 2 of each algorithm we split Y into two subsets, Z and $Y - Z$; write $p = \#Z$, with $0 < p < m$. Define r and s as the ranks of A and B respectively.

Algorithm Q computes (for $m \geq 2$) five matrix products, of sizes $m - p$ by s by s, $m - p$ by s by $n - s$, s by $r - s$ by $r - s$, s by s by $r - s$, and r by $r - s$ by s. It also calls itself recursively with a p-by-n matrix of rank s and an $(m-p)$-by-$(n-s)$ matrix of rank $r - s$.

Algorithm S computes matrix products of sizes $m - p$ by s by s, $m - p$ by s by $n - s$, $m - p$ by s by 1, s by $r - s$ by 1, and s by s by 1. It calls Algorithm Q with a p-by-n matrix of rank s; it calls itself recursively with an $(m-p)$-by-$(n-s)$ matrix of rank $r - s$.

Algorithm N computes matrix products of sizes $m - p$ by s by s, $m - p$ by s by $n - s$, s by $n - s$ by $n - r$, and s by s by $n - r$. It calls Algorithm Q with a p-by-n matrix of rank s; it calls itself recursively with an $(m-p)$-by-$(n-s)$ matrix of rank $r - s$.

Fast quasiinversion. Say we can compute the product of two k-by-k matrices with at most αk^β field multiplications. Here α and β are real numbers with $\beta > 2$. Then one can find a quasiinverse of a k-by-k matrix, together with a basis for the nullspace of the matrix, in $O(k^\beta)$ field multiplications.

Theorem 4. *Assume that m is a power of 2, that n is a positive multiple of m, and that p is always chosen as $m/2$ when $m \geq 2$ in Algorithm Q. Set $\gamma = 7/(2^\beta - 4)$. Then Algorithm Q uses at most $\alpha\gamma n m^{\beta-1}$ field multiplications.*

Proof. For $m = 1$ Algorithm Q performs no multiplications.

For $m \geq 2$ the matrix products in Algorithm Q may be broken up into, at worst, $n/p + 5$ products of p-by-p matrices, using a total of $\alpha(n + 5p)p^{\beta-1}$ field multiplications. The two recursive calls are each of size at worst p by n, so by induction they take at most $2\alpha\gamma n p^{\beta-1}$ field multiplications.

Since $p = m/2 \leq n/2$ we have $n + 5p + 2\gamma n \leq (7/2 + 2\gamma)n = \gamma 2^{\beta-1} n$. □

Under the same assumptions, Algorithm S uses at most $\alpha(\delta_1 nm^{\beta-1}+\delta_2 m^\beta)+m^2$ field multiplications, and Algorithm N uses at most $\alpha(\delta_3 n^2 m^{\beta-2}+(4\gamma/7)nm^{\beta-1}+\delta_2 m^\beta)$ field multiplications. Here $\delta_1=(2+2\gamma)/(2^\beta-2)$, $\delta_2=1/(2^\beta-1)$, and $\delta_3=(4+2\gamma)/(2^\beta-2)$.

One-dimensional pivots. If $p=1$, my algorithms reduce to Gaussian elimination with partial pivoting.

Theorem 5. *Assume that p is always chosen as* 1 *in Algorithm Q. Then Algorithm Q takes at most $rb(m,n,r)$ field multiplications, where $b(m,n,r)=mn+r^2-1-m(r-1)/2-n(r+1)/2$.*

For $r=m$ this bound factors as $m(m-1)(m+n+2)/2$. For $r=m=n$ the bound is roughly m^3; for $r=m\ll n$ the bound is roughly $m^2n/2$.

Proof. For $r=0$ there are no multiplications. For $m=1$ and $r=1$ there are again no multiplications; and $b(1,n,1)=0$.

For $m\geq 2$ we have two cases. If $s=0$, Algorithm Q ends up doing nothing except calling itself recursively with an $(m-1)$-by-n matrix of rank r; by induction this takes at most $rb(m-1,n,r)\leq rb(m,n,r)$ multiplications.

If $s=1$, Algorithm Q does various matrix products totalling $(m-1)n+2r(r-1)$ field multiplications. It also calls itself recursively with an $(m-1)$-by-$(n-1)$ matrix of rank $r-1$. Total: $(r-1)b(m-1,n-1,r-1)+(m-1)n+2r(r-1)=rb(m,n,r)$ multiplications. □

Similarly, Algorithm S uses at most $r(mn-m(r-3)/2-n(r+1)/2+(r^2-1)/3)$ multiplications. For $r=m=n$ this is roughly $m^3/3$. For $r=m\ll n$ this is roughly $m^2n/2$.

Algorithm N uses at most XXX. For $r=m=n$ this is roughly $m^3/3$. For $r=m\ll n$ this is roughly mn^2.

Practical concerns. XXX Reader beware: I have not yet tested these algorithms; I have merely proven them correct.

How should we choose p in practice? One sensible possibility is $p=\lfloor m/2\rfloor$. This takes maximum advantage of fast multiplication, for large m. Even without fast multiplication, it is a good idea to break the original matrix into solid chunks rather than peeling off one row at a time, because this uses the cache very effectively.

On the other hand, there is a mild source of inefficiency in Algorithm Q as p grows. The reason is that Algorithm Q always computes all of C in Theorem 2, even though in principle only a portion of C should be necessary. Imagine, for example, that C is a matrix with 100 rows and 10000 columns. Its quasiinverse F is determined by at most 100 of those columns. If we somehow knew which columns to look at, or if we guessed the right columns and verified them against some magically acquired knowledge of C's rank, we could avoid computing the other columns of C. Note that Strassen's (failure-prone) method avoids this slight slowdown, essentially by assuming that the first 100 columns of C form a nonsingular matrix. One could perhaps modify Algorithm Q to optimistically check for this case, and to back off if those columns turn out to be singular.

I recommend experimenting with various choices of p between 1 and $m/2$, for whatever type of matrix is at hand.

Note that there is a dual form of generalized Gaussian elimination, in which V, rather than W, is split up. Perhaps one could gain some speed in practice by combining the original method with its dual.

References

1. J. R. Bunch and J. E. Hopcroft, *XXX*, Mathematics of Computation **28** (1974), 231–236.
2. Donald Coppersmith and S. Winograd, *XXX*, Journal of Symbolic Computation **9** (1990), 251–280.
3. Carl Linderholm, *Mathematics made difficult*, XXX, XXX.
4. Volker Strassen, *Gaussian elimination is not optimal*, Numerische Mathematik **13** (1969), 354.

X. XXX, *XXX*, to appear, XXX **XXX** (XXX), XXX–XXX.

5 Brewster Lane, Bellport, NY 11713