ABS Algorithms for Linear Systems and Linear Least Squares: Theoretical Results and Computational Performance

Emilio Spedicato[1]

ABS methods have been introduced by Abaffy, Broyden and Spedicato [1] initially for solving linear systems and have been later extended to solving linear least squares, nonlinear systems and nonlinear optimization problems. In this paper, we first review the basic theoretical properties of ABS methods for linear systems and linear least squares. Then we discuss extensive computational experience on sequential, vector and parallel computers. A comparison with codes in the NAG, LINPACK and LAPACK libraries shows that ABS methods are of comparable accuracy (marginally more accurate on ill-conditioned problems) and faster on vector/parallel computers.

INTRODUCTION

The Basic ABS Class

ABS methods have been introduced by Abaffy, Broyden and Spedicato [1] initially for solving a determined or underdetermined linear system $Ax = b$, $A = (a_1, \ldots, a_m)^T \in R^{m,n}$, $x \in R^n$, $b \in R^m$, $m \leq n$, by the so-called basic or unscaled ABS class of algorithms defined by the following procedure, where e_i denotes the ith unit vector:

(A) Give $x_1 \in R^n$, an arbitrary estimate of the solution x^+; give $H_1 \in R^{n,n}$, an arbitrary nonsingular matrix; set $i = 1$.

(B) Compute $s_i = H_i a_i$. If $s_i \neq 0$ go to (C). If $s_i = 0$ and $a_i^T x_i - b^T e_i = 0$, set $x_{i+1} = x_i$, $H_{i+1} = H_i$ and go to (F), the ith equation is a linear combination of the previous equations; otherwise stop, the system has no solution.

(C) Compute the search vector p_i by:

$$p_i = H_i^T z_i \ , \qquad (1)$$

where $z_i \in R^n$ is an arbitrary vector satisfying $z_i^T H_i a_i \neq 0$.

(D) Update the estimate of the solution by:

$$x_{i+1} = x_i - \alpha_i p_i \ , \qquad (2)$$

where the stepsize α_i is given by:

$$\alpha_i = (a_i^T x_i - b^T e_i)/p_i^T a_i \ . \qquad (3)$$

(E) Update the (Abaffian) matrix H_i by:

$$H_{i+1} = H_i - H_i a_i w_i^T H_i / w_i^T H_i a_i \ , \qquad (4)$$

where $w_i \in R^n$ is an arbitrary vector satisfying $w_i^T H_i a_i \neq 0$.

1. Department of Mathematics, University of Bergamo, Piazza Rosate 2, 24100 Bergamo, Italy.

Scientia Iranica, Vol. 1, No. 4, © Sharif University of Technology, January 1995.

(*F*) If $i = m$, stop, x_{m+1} solves the system, otherwise increment i by one and go to (*B*).

Basic properties of the above procedure are the following:

(A1) $H_i a_i = 0$ iff a_i is a linear combination of $a_1, \ldots, a_{i-1}$.

(A2) The search vectors $p_1, \ldots, p_i$ are linearly independent.

(A3) Let $A_i = (a_1, \ldots, a_i)^T$, $P_i = (p_1, \ldots, p_i)$ and define L_i by:

$$L_i = A_i P_i \ , \tag{5}$$

then, if rank$(A_i) = i$, L_i is nonsingular lower triangular.

(A4) The vector x_{i+1} solves the first i equations of the system, i.e., $a_j^T x = b^T e_j, j = 1, \ldots, i$.

(A5) The general solution of the first i equations has the form:

$$x = x_{i+1} + H_{i+1}^T q \ , \tag{6}$$

with $q \in R^n$ arbitrary.

(A6) Null $(H_{i+1}) = $ range(A_i^T) implying $H_{i+1} A_i^T = 0$;
rank$(H_{i+1}) = n - i$ if rank$(A_i) = i$;
null $(A_i) = $ range(H_{i+1}^T).

(A7) $H_i H_1^{-1} H_i = H_i$.

The matrices defined by recursion in Equation 4 have been denominated Abaffian matrices at the First International Conference on ABS Methods, held in Luoyang, September 1991.

THE SCALED ABS CLASS

Let $V = (v_1, \ldots, v_m) \in R^{m,m}$ be an arbitrary nonsingular matrix, called the scaling matrix. Then the scaled system $V^T A x = V^T b$ is equivalent to the system $Ax = b$ in the sense that both systems possess the same set of solutions. Let $r(x) = Ax - b$ be the residual of the original system. Let $r_i = r(x_i)$. Then, by applying the basic ABS class procedure to the above scaled system, we obtain the following recursions, which define the scaled ABS class procedure:

(A') Same as (A).

(B') Compute $s_i = H_i A^T v_i$. If $s_i \neq 0$, go to (C'). If $s_i = 0$ and $r_i^T v_i = 0$, set $x_{i+1} = x_i$, $H_{i+1} = H_i$ and go to (F'), the ith equation of both the original and the scaled system is a linear combination of the previous equations. Otherwise stop, the system has no solution.

(C') Compute the search vector as in (C'), with $z_i \in R^{n,n}$ arbitrary save that $z_i^T H_i A^T v_i \neq 0$.

(D') Update the estimate of the solution by Equation 2 with α_i given by:

$$\alpha_i = r_i^T v_i / p_i^T A^T v_i \ . \tag{7}$$

(E') Update the Abaffian by:

$$H_{i+1} = H_i - H_i A^T v_i w_i^T H_i / w_i^T H_i A^T v_i \ ,$$

where $w_i \in R^n$ is an arbitrary vector satisfying $w_i^T H_i A^T v_i \neq 0$.

(F') Same as (F).

We note that at the ith step of the scaled ABS class procedure, only the ith column v_i of V is used, implying that V need not be defined initially and that v_i can be considered as a parameter, arbitrary save for linear independence from $v_1, \ldots, v_{i-1}$. We note also that x_{m+1} solves both the scaled and the original systems, hence the scaled ABS class is a generalization, with the extra scaling parameter v_i, of the basic class for solving $Ax = b$. It can be shown that essentially any algorithm of the form $x_{i+1} = x_i - \alpha_i p_i$ which, starting from an arbitrary x_1, solves a linear system in a number of steps no greater than the number of equations, is a member of the scaled ABS class, i.e., it corresponds to some parameter choices in that class.

Properties (A1) to (A7) are easily reformulated for the scaled ABS class. In particular the factorization relation in Equation 5 becomes now, with $V_i = (v_1, \dots, v_i)$:

$$V_i^T A\ P_i = L_i\ . \tag{8}$$

Property (A4) now says that the residual r_{i+1} is orthogonal to the previous scaling vectors, i.e.:

$$r_{i+1}^T v_j = 0\ , \quad j = 1, \dots, i\ . \tag{9}$$

The Scaled Block ABS Methods

ABS methods can also be given in a block formulation, which can be useful in dealing with special structures in the coefficient matrix and which has also turned out to be efficient for improving the speed-up on vector and parallel computers. Assuming for simplicity that A has full rank, the scaled block ABS method is given by the following procedure:

(A'') Give $x_1 \in R^n$ arbitrary. Give $H_1 \in R^{n,n}$ arbitrary nonsingular. Let $m_1, \dots, m_p$ be positive integers such that $m_1 + \cdots + m_p = m$. Let $i = 1$.

(B'') Compute the residual $r_i = Ax_i - b$. Stop if $r_i = 0$.

(C'') Let $V_i \in R^{m,m_i}$ and $Z_i \in R^{n,m_i}$ be arbitrary full rank matrices such that the matrix $V_i^T A H_j^T Z_i$ is nonsingular. Define the matrix $P_i \in R^{n,m_i}$ by:

$$P_i = H_i^T Z_i\ . \tag{10}$$

(D'') Update the estimate of the solution by:

$$x_{i+1} = x_i - P_i d_i\ , \tag{11}$$

where $d_i \in R^{m_i}$ is the unique solution of the system:

$$V_i^T A P_i d_i = V_i^T r_i\ . \tag{12}$$

If $i = p$ stop, x_{p+1} solves the system.

(E'') Update the matrix H_i by:

$$H_{i+1} = H_i - H_i A^T V_i W_i^T H_i\ , \tag{13}$$

where $W_i \in R^{n,m_i}$ is an arbitrary full rank matrix satisfying the condition:

$$W_i^T H_i A^T V_i = I_{m_i}\ . \tag{14}$$

(F'') Increment i by one and go to (B'').

ALTERNATIVE IMPLEMENTATIONS

The basic and the scaled ABS class have been given here in the so-called standard version, based upon the n by n Abaffian matrix H_i. There are several alternative formulations, some discussed in Abaffy and Spedicato [2], others in more recent papers, e.g., Bodon and Spedicato [3], Chen, Deng and Xue [4], Spedicato and Zhu [5], essentially based upon the use of non-square matrices, resulting generally in a reduction of both storage and overhead. For brevity we will consider only some of these versions and only with reference to the unscaled class.

The first version was derived by analogy with the memoryless quasi-Newton method and is particularly convenient if one is dealing with underdetermined systems with $m \ll n$. It requires, at step i, the storage of $2i - 2$ vectors. It uses the following recursions, starting with $s_1 = H_1 a_1$ and $u_1 = H_1^T w_1$:

$$p_i = H_1^T z_i - \sum_{j<i} u_j s_j^T z_i\ , \tag{15}$$

$$s_i = H_1 a_i - \sum_{j<i} s_j u_j^T a_i\ , \tag{16}$$

$$u_i = H_1^T w_i - \sum_{j<i} u_j s_j^T w_i\ . \tag{17}$$

Notice that:

$$H_{i+1} = H_1 - \sum_{j<i} s_j u_j^T\ . \tag{18}$$

Another approach assumes, without loss of generality, that $w_i = z_i / z_i^T H_i a_i$ and that feasible parameters $z_1, \ldots, z_n$ are given initially. Then $p_i = u_i^i$, where $u_j^1 = H_1^T z_j$, $j = 1, \ldots, n$, and the vectors u_j^i are updated, for $i = 1, \ldots, m$, by:

$$u_j^{i+1} = u_j^i - (a_i^T u_j^i / a_i^T u_i^i) u_i^i, \qquad j = i+1, \ldots, n \, . \tag{19}$$

This approach, related to a class of parallel methods considered by Sloboda [6] requires the storage of $n - i$ vectors at step i. Note that the linear variety comprising all solutions of the first i equations consists of the vectors x having the form:

$$x = x_{i+1} + U_i d \, , \tag{20}$$

where $d \in R^{n-i}$ is arbitrary and $U^i = (u_{i+1}^{i+1}, \ldots, u_n^{i+1})$.

The version of Spedicato and Zhu [5] uses formulas formally similar to those in the basic ABS procedure save that the matrix H_i is not n by n but is $n + 1 - i$ by n. At each step, after the update of H_i, a row of H_{i+1} is deleted. The deleted row is dependent on the remaining rows. It is proved that if the kth component of w_i is nonzero, then the kth row of H_i can be deleted. A difference with the version defined by recursions in Equation 19 is that the vectors z_i and w_i have now dimension $n + 1 - i$ and must not be specified in advance.

PARTICULAR ALGORITHMS IN THE BASIC ABS CLASS

We consider now some special choices of the parameters in the ABS class, defining algorithms which are related to well-known methods but whose ABS formulation may differ in computational complexity, in storage requirement, numerical stability and degree of parallelization.

The Huang and the Modified Huang Algorithms

The Huang algorithm, originally considered by Huang [7] in a paper which has been seminal for the development of the ABS class, is based upon the well-defined choices $H_1 = I$, $z_i = w_i = a_i$, which provide:

$$p_i = H_i a_i \, , \tag{21}$$

$$H_{i+1} = H_i - H_i a_i a_i^T H_i / a_i^T H_i a_i \, . \tag{22}$$

The search vectors generated by the Huang algorithm are orthogonal and coincide, in exact arithmetic, with the vectors generated by the Gram-Schmidt orthogonalization procedure applied on the rows of A. If $x_1 = 0$, then x_{i+1} is the solution of least Euclidean norm of the first i equations and moreover the solution x^+ (of least Euclidean norm) is approached monotonically from below (in Euclidean norm).

The modified Huang algorithm is a modification of the previous algorithm which, while generating the same iterates in exact arithmetic, is more accurate in presence of roundoff, as shown for instance in the experiments of Spedicato and Vespucci [8]. It is based upon the formulas:

$$p_i = H_i(H_i a_i) \, , \tag{23}$$

$$H_{i+1} = H_i - p_i p_i^T / p_i^T p_i \, . \tag{24}$$

From Equation 23, we see that the search vector in the original Huang algorithm is reprojected onto the range of H_i, which coincides with the null space of A_{i-1}. This operation tends to annihilate components of the original Huang search vector in the null space of H_i which are not zero due to roundoff. A theoretical analysis of Broyden [9] shows that the error growth of the $i - 1$ small eigenvalues of H_i, which should be zero if there was no roundoff, is two orders lower if formulas in Equations 23 and 24 are used, resulting actually, under simplifying assumptions, in no further error growth. The reprojection technique is applicable to most ABS algorithms. It has been shown numerically to substantially improve the accuracy of several methods in ABS formulation, as the QR, Craig, Hestenes-Stiefel algorithms (see, for instance, Bodon and Spedicato [10,11]).

The Implicit Lu or Gauss-Choleski Algorithm

This algorithm is obtained by the choices $H_1 = I$, $z_i = w_i = e_i$. It is well defined if A is strongly nonsingular (all the principal submatrices are nonsingular), in which case, the scalar product $e_i^T H_i a_i$, appearing at the denominators of Equations 3 and 4, is identical with the pivot at the ith stage of Gaussian elimination. The matrix P_i in Equation 5 is upper triangular, motivating the name. The formulas for the search vector and the Abaffian update are:

$$p_i = H_i^T e_i \ , \tag{25}$$

$$H_{i+1} = H_i - H_i a_i p_i^T / p_i^T a_i \ . \tag{26}$$

From Equation 25, p_i is just the ith row of H_i. From Equation 26, it follows easily that H_{i+1} has the following structure:

$$H_{i+1} = \begin{bmatrix} 0 & 0 \\ S_i & I_{n-i} \end{bmatrix} \ , \tag{27}$$

where $S_i \in R^{n-i,i}$ can be shown to have the following structure,

$$S_i = -\hat{A}^i (A^{(i)})^{-1} \ , \tag{28}$$

where $A^{(i)}$ is the ith principal submatrix of A^T and $\hat{A}^i \in R^{n-i,i}$ is the matrix comprising the last $n-i$ columns of A_i .

It is easy to verify that the number of multiplications required by the implicit LU algorithm is $n^3/3 + O(n^2)$, as for the classical algorithm, while the maximum storage is $n^2/4 + O(n)$. Some other properties are:

- The search vectors are A-semiconjugate, i.e., $p_j^T A p_i = 0$, $j < i$. Hence A-conjugacy holds if A is symmetric but not necessarily positive definite.
- If A is symmetric and positive definite, then $e_i^T H_i a_i > 0$ and, if we set $z_i = e_i/(e_i^T H_i a_i)^{1/2}$, then $P_i^{-1} = L_i^T$ in Equation 5, hence the algorithm implicitly generates the Choleski factorization.
- If A is symmetric and positive definite, then x_{i+1} minimizes the quadratic function $F(x) = (x - x^+)^T A (x - x^+)$ over the linear variety $x_1 + \text{Range}(P_i)$. It also follows that x^+ is approached monotonically from above in the A-weighted Euclidean norm.

If H_1 is not the identity matrix, we obtain the generalized implicit LU algorithm (see Spedicato and Zhu [12]). This algorithm is well defined if the matrix AH_1^T is strongly nonsingular. It can be shown that any algorithm in the basic ABS class, with parameters H_1, $z_i = w_i = s_i$ is equivalent to the generalized implicit LU algorithm with initial Abaffian $H_1' = SH_1$, $S = (s_1, \ldots, s_n)$. For such an algorithm, the number of multiplications is at most n^3.

SUBCLASSES IN THE SCALED ABS CLASS

The Conjugate Direction Subclass

This is a subclass of the scaled ABS class where $v_i = p_i$, a choice which is well-defined if A is square symmetric and positive definite. From the factorization relation in Equation 8, we obtain, with $P = P_n$:

$$P^T A P = D \ , \tag{29}$$

with D diagonal, implying that the search directions are A-conjugate. This subclass contains the Lanczos and the Hestenes-Stiefel algorithms, the last one corresponding to $H_1 = I$, $z_i = w_i = r_i$. It can also be shown that the implicit LU algorithm is a member of this class.

The Orthogonally Scaled Subclass

This subclass is obtained by the choice $v_i = Ap_i$ and is well-defined for A with full column rank. From Equation 8 we obtain, with $V = V_n$:

$$V^T V = P^T A^T A P = D \ , \tag{30}$$

hence, the search vectors are $A^T A$-conjugate and the scaling vectors are orthogonal. Two important algorithms in this class are the implicit QR algorithm, given by $H_1 = I$, $z_i = w_i = e_i$, for which P_i in Equation 8 is upper triangular, and the algorithm with $H_1 = I$, $z_i = w_i = A^T r_i$, which generates the same iterates x_i as the minimum residual conjugate gradient method. The vector x_{i+1} in this subclass minimizes the function $F(x) = r(x)^T r(x)$ over the linear variety $x_1 + \text{Range}(P_i)$.

The Optimally Stable Subclass

This subclass, well-defined for A square nonsingular, corresponds to the choice $v_i = A^{-T}p_i$; the inverse appearing in this definition can be removed in the actual recursions, which are also well-defined for underdetermined systems. Taking, without loss of generality, $z_i = w_i$, the Abaffian update can be written in the form:

$$H_{i+1} = H_i - p_i p_i^T / p_i^T p_i \ . \tag{31}$$

Setting $z_i = A^T u_i$, the stepsize can be written as:

$$\alpha_i = r_i^T u_i / a_i^T p_i \ . \tag{32}$$

The search vectors in this subclass are orthogonal. Setting $u_i = e_i$, we get the Huang algorithm. Setting $u_i = r_i$, we obtain an algorithm generating the same iterates x_i as Craig's conjugate gradient algorithm for nonsymmetric systems.

The name of this class comes from the fact that the scaling matrix satisfies the relation:

$$V^T A A^T V = D \ , \tag{33}$$

with D diagonal, which was shown by Broyden [13] to characterize the class of algorithms, in his general class equivalent to the scaled ABS class, such that the error in x_{n+1}, due to the introduction of a single error ε in x_i, is minimized (being actually not greater than ε). It can be shown that the algorithms in the unscaled and in the orthogonally scaled subclass minimize a similar error with respect to the residual r_{n+1}. Hence, the Huang algorithm, being a member of both the unscaled and the optimally stable subclasses, possesses both properties, while the implicit LU and QR algorithms are optimal only with respect to the residual error.

SOLVING LINEAR LEAST SQUARES

Overdetermined linear systems can be solved by several ABS approaches for their generalized solution in the least squares sense, i.e., for the vector $x^\star$ that solves the normal equations of Gauss:

$$A^T A x = A^T b \ . \tag{34}$$

The system in Equation 34 is always compatible. The solution is unique if A has full column rank, otherwise one usually is interested in the solution of least Euclidean norm, formally given by relation:

$$x^\star = A^\star b \ , \tag{35}$$

where $A^\star$ is the Moore-Penrose pseudoinverse.

There are several ABS approaches for the least squares generalized solution. One possibility is obviously to compute a QR or LQ factorization of A by making explicit the implicit factorization associated with the implicit QR or the Huang (or modified Huang) algorithms. Then, such a factorization can be used in the traditional way.

A second possibility is to compute the pseudoinverse $A^\star$ by ABS techniques (see Spedicato and Bodon [14] or Spedicato and Xia [15]). Another approach is based upon the equivalence of the normal equations with the following extended system in the variables x and y:

$$Ax = y \ , \tag{36}$$

$$A^T y = A^T b \ . \tag{37}$$

In order for Equation 36 to be solvable, we must have $y \in \text{Range}\,(A)$, which implies that y must be the unique solution of least Euclidean norm of the underdetermined system in Equation 37. Such a solution can be obtained by the Huang algorithm. Applying any ABS algorithm to Equation 36 removes the $m - q$ dependent equations, in step (B), where $q = \text{rank}\,(A) \leq n$. If $q < n$ and the solution of least Euclidean norm is wanted, then Equation 36 should be solved using the Huang algorithm.

A final approach is based upon the algorithms in the subclass where $v_i = Au_i$, or $V = AU$, $U = (u_1, \ldots, u_n) \in R^{n,n}$ being arbitrary nonsingular and A having full column rank. After n steps of the algorithm, when $H_{n+1} = 0$, if $V^T A$ is nonsingular, it follows from Equation 9 that $V^T r_{n+1} = U^T A^T r_{n+1} = 0$. Since U is nonsingular, then $A^T r_{n+1} = A^T A x_{n+1} - A^T b =$

0, hence, x_{n+1} solves the normal equations and a generalized least squares solution has been found.

If $U = P$, the considered subclass reduces to the orthogonally scaled subclass, whose algorithms, therefore, all have the property of solving overdetermined linear systems for the least squares solution. This is then true in particular for the implicit QR algorithm. If A does not have full column rank, the obtained solution is not one of least Euclidean norm, but, if x_1 is zero, it is a solution of basic type, where the last $n - q$ components are zero, $q = \text{rank}(A)$.

NUMERICAL EXPERIENCE WITH ABS ALGORITHMS FOR GENERAL LINEAR SYSTEMS

ABS algorithms have been extensively tested for general linear systems (see [3,8,10,16]). Important issues to be tested numerically were:

- The numerical stability of the algorithms in comparison with standard formulations.
- The relative performance, in terms of accuracy, of the several alternative formulations of a given algorithm (half a dozen formulations, not counting the possible use of block formulations and reprojections).

The tested ABS algorithms have been the following:

1. The Huang algorithm (see, in particular, [8]).
2. The implicit LU and implicit QR algorithms (see, in particular, [10]).
3. Some algorithms of the conjugate gradient type, including the ABS versions of the Hestenes-Stiefel, the Craig, the STOD and the minimum residual method.

From the performed experiments the following conclusions can be made:

- The alternative versions of a given ABS algorithm differ not only in overhead and storage, but also in the final accuracy. The most accurate versions are at least as accurate, and often more accurate, on severely ill-conditioned problems, than the codes, based upon standard LU factorization with partial pivoting, of the NAG, LINPACK, ESSL and LAPACK libraries.
- The use of reprojections, usually on the search vector, sometimes on other vectors (for instance on the scaling vector in the implicit QR algorithm), significantly improves the accuracy on the ill-conditioned problems. Use of more than one reprojection is not recommended, its effect being usually negligible.
- While the standard formulation of conjugate gradient type algorithms is unstable and gives very poor results on ill-conditioned problems, the ABS formulation, using reprojections, approximates the solution with accuracy comparable with that obtained by the LU or orthogonalization procedures.

Table 1. Results on ill-conditioned problems.

Method	E1MED	E2MAX	ERES
Huang	2E−1	6E−1	5E−5
Mod Huang	3E−2	2E−1	8E−7
QR	8E−2	1E−1	1E−6
Brent	8E−2	5E−1	1E−6

In Table 1, taken from Spedicato and Vespucci [8], we give the average value, on the 33 tested ill-conditioned problems, of the (geometric) average relative error in the solution E1MED, of the average maximum of such error E2MAX and of the average residual error ERES for the following methods: standard Huang, modified Huang, QR implementation in LINPACK (codes SQRDC, SQRSL) and the code due to Moré and Cosnard [17] implementing the method of Brent. The results are obtained in single precision on an IBM 4361 with unit roundoff error of about 4×10^{-7}.

Very extensive numerical experiments, involving several alternative formulations of the implicit LU, implicit QR and the Huang algorithms, have been done by Bodon and Spedicato [10], showing that the best versions of these methods have accuracy comparable with that attained by the QR code in the NAG library.

Further testing is planned to evaluate the following questions of interest:

- Effect of equations pivoting.
- Effect of iterative refinement (which can be performed in a similar way as with the standard methods by keeping the search vectors).

The quoted experiments have given confidence in the ABS algorithm as a stable and accurate linear solver. In terms of storage or overhead, the ABS algorithms in the sequential formulation are not better than their classical counterparts. Experiments on vector/parallel computers have shown that this judgment may have to be changed, ABS algorithms having the potential for better vectorization and parallelism, associated with their use of compact operations on square or rectangular matrices with better management of the memory access.

The early experiments with vector machines are due to Bertocchi and Spedicato [18-20], who investigated the performance of some vectorized versions of the implicit LU and Huang algorithms in the nonblock and the block formulation. The experiments were performed on the IBM 3090 VF, using the Dongarra matrix and taking as benchmarks the LU codes in the NAG and the ESSL libraries. In order to obtain an efficient vectorization, the following steps were taken:

- Storage of matrices and vectors to allow use of stride one as often as possible.
- Use of BLAS routines as often as possible.
- Following an idea that Robert and Sguazzero [21] found useful in the context of the standard LU factorization, the Abaffian update was modified from a rank-one correction at each step to a rank-k correction every k steps, with corresponding modification of the formula for the search vector. An "optimal" value of k ($k = 8$) was obtained experimentally.
- Loop unrolling was considered and found useful in the rank-k corrections.

The obtained vectorized version of the implicit LU algorithm was found to be about three times faster than the NAG code and only 30% slower than the ESSL code (which was implemented, not in FORTRAN, but in ASSEMBLER).

The modified Huang algorithm was also considered in vector implementation. It was noted that it was better not to take into account the symmetry and to use a rank-k update of the Abaffian (optimal value for k was 8 as for the implicit LU algorithm). The faster version was obtained using formulas in Equations 15–17. In Table 2, we give the total time and the obtained number of megaflops for the standard modified Huang algorithm (MH), the version with rank-8 update (MH8), the version using formulas in Equations 15–17, which can be viewed as a rank-n update (MHn) and the ESSL code.

The experiments in Bertocchi and Spedicato [20] have evaluated the effect of the block formulation. It has been found that the speed-up depends on the block size and increases with the dimension (up to 60% for $n = 700$). The optimal size of the block also appears to increase with the dimension.

Experience with vector and parallel versions of the implicit LU and Huang algorithms has been obtained by Bodon [22] on the Alliant FX/80 with 8 processors. Each algorithm was implemented in several versions depending on:

- Selection from five different formulations of the search vector and the Abaffian, including factorized versions of the algorithms developed by Bodon and Spedicato [3].
- Use of reprojections.
- Use of block formulations.

Table 3 gives the results on some algorithms compared with the LAPACK code SGETF2-SGETRS. The ABS algorithms outperform the

Table 2. Performance of the vectorized MH algorithm.

n	MH	MH8	MHn	ESSL
100	0.2/30	0.15/28.4	0.11/36.4	0.02/33.3
500	21/35.8	14/38	9.4/53.2	1.33/62
900	121/36.8	87.1/35.6	58/50.2	7.24/67

LAPACK code in megaflops, have lower timing and are often more accurate. The improvement appears, however, to degrade with growing n, as was also observed in experiments with only vectorization. This suggests the validity of the following (loosely stated).

Conjecture

For fixed n, there are ABS versions of classical methods that are faster for a sufficiently large number of parallel processors or sufficiently long vector registers. For a given number of parallel processors, the classical version is faster for sufficiently large n.

NUMERICAL EXPERIMENTS WITH ABS ALGORITHMS FOR SPARSE LINEAR SYSTEMS

Large linear systems are often sparse and special methods are developed to take into account the presence of zeros in A to reduce storage and overhead. Sparse problems may be structured or unstructured. Special ABS methods have been developed for the following types of structured matrices:

(a) General band matrices (arising in several PDE discretization processes).

(b) Block angular matrices, i.e., matrices where $A_{ij} = 0$ for $i < j - k_1$ and $j < n - k_2$ (arising inter alia in automatic differentiation processes).

(c) ND (nested dissection) matrices, arising after optimal reordering of positive definite matrices in finite element (FE) methods.

For matrices of the type (a) and (b), the relevant formulas have been developed by Abaffy and Dixon [23] (see also Abaffy [24]) in relation with the implicit LU, implicit QR and Huang algorithms. For ND matrices, the formulas have been obtained by Zhu [25]

Table 3. LAPACK versus ABS algorithms on the Alliant FX/8.

n	Algorithm	EX	ER	Time	M-flops
100	LAPACK	3E–5	1E–6	0.201	3.39
	LU	5E–5	7E–7	0.100	10.35
	LUBLOCK	5E–4	9E–7	0.080	11.40
	MHUANG	7E–7	2E–7	0.240	21.08
	MHUANGBLOCK	4E–6	2E–7	0.155	19.69
500	LAPACK	1E–4	8E–6	6.23	13.43
	LU	8E–5	2E–6	6.749	18.62
	LUBLOCK	2E–4	3E–6	3.701	21.17
	MHUANG	2E–5	4E–7	14.26	35.06
	MHUANGBLOCK	1E–5	3E–7	4.19	66.41

Table 4. Results on tridiagonal matrices.

n	Method	Time	M-flops	EX	ERES
10,000	GAUSS	0.040	2.03	3E–8	3E–8
	ABS	0.022	3.90	5E–8	4E–8
50,000	GAUSS	0.197	2.03	4E–8	3E–8
	ABS	0.092	4.60	6E–8	4E–8
100,000	GAUSS	0.394	2.03	4E–8	3E–8
	ABS	0.181	4.70	6E–8	4E–8

and have been further developed in Yang and Zhu [26]. While the overhead of the special ABS formulas is not usually lower than for the classical procedures in the sequential case (it is higher, for instance, for the implicit LU algorithm in the banded case, see Galantai [27], unless a block formulation is used), the storage may be lower. For instance, the storage is lower by a factor of two for the implicit LU algorithm in the band case and, in the ND case, it is lower by a factor which increases with the dimension.

Preliminary testing for banded matrices on a sequential machine was done by Abaffy [28], who found the ABS methods numerically stable (and successful on a very large problem where the Harwell code failed). Ceribelli [29] found his pilot implementation of the band implicit LU algorithm several times faster than the (admittedly out-of-date) LU factorization code in the IBM SP library (still heavily used by engineers).

Extensive testing has been done by Bodon [30–33] on matrices of the types (a) and (b) and particularly on tridiagonal matrices, on the Alliant VF80. Several versions of the implicit LU algorithm were studied, using, in particular, sophisticated partitioning techniques. The comparison with the LAPACK solver for band matrices shows that the ABS solver, despite its generally higher overhead in the sequential case, is faster on the used vector/parallel machine for sufficiently small bandsize (less than about 20) and of similar accuracy. For tridiagonal matrices, the ABS codes are about two times faster than the LAPACK code (which is based on an algorithm essentially not parallelizable). For this type of problem there is no apparent deterioration with growing n of the relative improvement obtained by the ABS algorithms. Table 4 gives timings, megaflops, relative error in the solution EX and relative error in the residual ERES for GAUSS and the best tried ABS algorithm.

Table 5 gives the best results for GAUSS and the best of the tested algorithms (method A2 in Bodon [33]) for banded matrices with band size $k = 5$ and $k = 21$. Notice that the performance of the ABS methods deteriorates with k much farther than GAUSS despite an improvement in megaflops.

The testing of ABS methods for ND matrices is still at a preliminary stage. A sequential code has been implemented by Deng, Spedicato and Vespucci [34] and has been later vectorized by Marletta and Vespucci [35]. The vectorization on the IBM 3090VF improves the performance when the size of the matrices is at least 100; the improvement reaches 60% for $n = 300$. No comparisons are available with commercial codes.

For unstructured sparse matrices, one can consider reordering techniques that reduce the fill-in in the Abaffian. Preliminary work has been done by Tuma [36] and Spedicato and Tuma [37] on Abaffians generated by the implicit LU algorithm in the standard formulation. Benzi and Meyer [38] have independently done similar work on the version of this algorithm corresponding to the formula in Equation 19. Fill-in growth is controlled

Table 5. Results on banded matrices.

n	k	Method	Time	M-flops	EX	ERES
1000	5	GAUSS	0.213	0.09	2E–5	4E–7
		ABS	0.058	5.70	2E–5	3E–6
	21	GAUSS	0.231	1.08	2E–3	2E–6
		ABS	0.223	10.06	2E–3	1E–5
8000	5	GAUSS	1.707	0.09	5E–6	4E–7
		ABS	0.424	6.30	7E–6	2E–6
	21	GAUSS	1.858	1.08	2E–4	6E–6
		ABS	1.840	9.84	3E–3	8E–6

by a Markovitz type approach coupled with a threshold strategy that sets an element of the Abaffian to zero if its modulus is below a given small positive value τ. A sequential formulation of the algorithm has been tested on several problems from the Harwell-Boeing collection. Spedicato and Tuma have compared the performance with that of the NAG code MA28. The codes give a similar accuracy, except on the very ill-conditioned problems NNC1, NNC2 where MA28 fails while the ABS code computes a solution with residual norm less than 10^{-11}. Fill-in control performance differs remarkably between the two approaches, sometimes being better for the ABS code, sometimes for the NAG code. While, generally, MA28 has much lower timings, Benzi and Meyer have found a class of matrices, related to equilibrium equations in chemistry, where the ABS algorithm is significantly faster. See Table 6 for some results ("Fill" gives the final number of nonzero elements, "Cond" the condition number). Much further work is needed, with an expected improvement of the ABS algorithms on vector/parallel machines.

NUMERICAL EXPERIENCE WITH ABS ALGORITHMS FOR LINEAR LEAST SQUARES

Extensive numerical experiments with several ABS approaches to linear least squares have been performed on both sequential and vector/parallel computers. In Spedicato and

Table 6. Results on sparse ill-conditioned matrices.

Problem	n	Cond.	Method	Fill	ERES	Time
NNC1	1374	1E+15	MA28	80585	3E+1	208
			ABS	157637	5E–12	1466
NNC2	261	9E+14	MA28	7053	1E+1	3.7
			ABS	4851	3E–13	6.5
WEST	156	1E+31	MA28	410	2E–8	0.08
			ABS	156	9E–10	0.34

Table 7. Results on linear least squares.

m	n	**Method**	**EX**	**ERES**	**Time**	**M-flops**
110	100	NAG	9E–3	1E–8	0.268	9.6
		ABS	1E–4	3E–7	0.268	9.8
500	100	NAG	7E–3	2E–8	0.880	11.9
		ABS	9E–4	2E–9	0.616	18.6
1000	100	NAG	1E–3	2E–8	1.67	12.4
		ABS	2E–3	2E–8	1.10	20.5
1500	300	NAG	6E–3	1E–7	26.00	10.8
		ABS	1E–3	4E–9	12.5	22.5

Bodon [39] algorithms based upon the extended system in Equations 36 and 37, algorithms in the subclass with $v_i = Au_i(u_i = a_i$ and $u_i = p_i)$ and algorithms using the explicit QR and LQ factorization computed via ABS approach were tested on about 600 problems and compared with the NAG and LINPACK solvers using the QR factorization via Householder rotations or the singular value factorization. The results show that several ABS methods are more accurate than the NAG or LINPACK codes, especially on ill-conditioned or rank deficient problems. Similar good performance is shown by the methods using the ABS computation of the pseudoinverse (see Spedicato and Bodon [40]). The best performance, in terms of accuracy, is given by some implementations of the implicit QR algorithm with reprojection in both the search and the scaling vectors (see Spedicato and Bodon [41]). In all these experiments, only non-block versions were considered.

Further work on the implicit QR algorithm, also using block formulations, has been done by Spedicato and Bodon [42] on the Alliant FX 80 with 8 processors. Overall, 31 versions of the implicit QR algorithm were implemented, differing in the formulas used for the search and the scaling vector and in the block size. Compared with a vectorized and parallelized version of the NAG codes F01BKF and F04AUF, the ABS algorithms appear to be generally more accurate and, in most cases, faster, up to a factor of three (the factor depends on n and m). The use of the block formulation improves the performance of the ABS methods up to a factor of two, the optimal block size depending on n and m. Some results are given in Table 7.

CONCLUSION

In this paper we have presented the main theoretical properties of ABS methods for linear equations and linear least squares. The numerical results indicate that these methods are numerically stable, are competitive with classical methods on ill-conditioned problems and can perform faster on vector/parallel machines than their classical counterparts. Further work in this area should look more deeply at ABS formulations of conjugate gradient type methods and at iterative versions of the ABS methods via truncation of the procedure. ABS packages for general distribution should also be made available.

ACKNOWLEDGEMENT

This work was supported by MURST 40%

Programma Analisi Numerica e Matematica Computazionale.

REFERENCES

1. Abaffy, J., Broyden, C.G. and Spedicato, E. "A class of direct methods for linear equations", *Numerische Mathematik*, **45**, pp 361–376 (1984).

2. Abaffy, J. and Spedicato, E. *ABS Projection Algorithms: Mathematical Techniques for Linear and Nonlinear Equations*, Ellis Horwood, Chichester, UK (1989).

3. Bodon, E. and Spedicato, E. "Factorized ABS algorithms for linear systems: derivation and numerical results", Report DMSIA 14/91, University of Bergamo, Italy (1991).

4. Chen, Z., Deng, N. and Xue, Y. "A general algorithm for underdetermined linear systems", *Proceedings of the First International Conference on ABS Algorithms*, Luoyang, University of Bergamo, Italy, pp 1–13 (1992).

5. Spedicato, E. and Zhu, M. "A reduced ABS-type algorithm I: basic properties", Report DMSIA 10/94, University of Bergamo, Italy (1994).

6. Sloboda, F. "A parallel projection method for linear algebraic systems", *Apl. Mat. Ceskosl. Akad. Ved.*, **23**, pp 185–198 (1978).

7. Huang, H.Y. "A direct method for the general solution of a system of linear equations", *J. Optim. Meth. Appl.*, **16**, pp 429–445 (1975).

8. Spedicato, E. and Vespucci, M.T. "Variations on the Gram-Schmidt and the Huang algorithms for linear systems: a numerical study", *Aplikace Mathematiky*, **2**, pp 81–100 (1993).

9. Broyden, C.G. "On the numerical stability of Huang's update", *Calcolo*, **28**, pp 303–311 (1991).

10. Bodon, E. and Spedicato, E. "Numerical evaluation of the implicit LU, LQ and QU algorithm in the ABS class", Report DMSIA 20/90, University of Bergamo, Italy (1990).

11. Bodon, E. and Spedicato, E. "On some STOD-ABS algorithms for large linear systems", Report DMSIA 15/92, University of Bergamo, Italy (1992).

12. Spedicato, E. and Zhu, M. "The generalized implicit LU algorithm of the ABS class", Report DMSIA 3/94, University of Bergamo, Italy (1994).

13. Broyden, C.G. "On the numerical stability of Huang and related methods", *J. Optim. Meth. Appl.*, **47**, pp 7–16 (1985).

14. Spedicato, E. and Bodon, E. "Solving linear least squares by orthogonal factorization and pseudoinverse computation via the modified Huang algorithm in the ABS class", *Computing*, **42**, pp 195–205 (1989).

15. Spedicato, E. and Xia, Z. "On some ABS methods for the computation of the pseudoinverse", *Ricerca Operativa*, **22**, pp 35–41 (1992).

16. Abaffy, J. and Spedicato, E. "Numerical experiments with the symmetric algorithm in the ABS class for linear systems", *Optimization*, **18**, pp 197–212 (1987).

17. Moré, J.J. and Cosnard, M.Y. "Numerical solution of nonlinear equations", *ACM Trans.*, **5**, pp 64–85 (1979).

18. Bertocchi, M. and Spedicato, E. "Vectorizing the implicit Gauss-Cholesky algorithm of the ABS class on the IBM3090 VF", *Quaderno* DMSIA 22/88, University of Bergamo, also Technical Report 1/37 (1989), Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo, Consiglio Nazionale delle Ricerche, Roma, Italy (1988).

19. Bertocchi, M. and Spedicato, E. "Vectorizing the modified Huang algorithm of the ABS class on the IBM3090 VF", *Quaderno* DMSIA 23/88, University of Bergamo (1988), also in *Proceedings of the International Meeting on Parallel Computing*, Verona, Italy, pp 83–90 (October 1988).

20. Bertocchi, M. and Spedicato, E. "Block ABS algorithms for dense linear systems in a vector processor environment", *Quaderno* DMSIA 2/90, University of Bergamo (1990), also in *Proceedings of the Conference on Supercomputing Tools for Science and Engineering*, D. Laforenza and R. Perego, Eds., Franco Angeli, Milano, Italy, pp 39–46 (December 1989).

21. Robert, Y. and Sguazzero, P. "The LU decomposition algorithm and its efficient FORTRAN implementation on the IBM 3090 vector multiprocessor", Report ICE - 006, IBM ECSEC Center, Rome (1987).

22. Bodon, E. "Numerical results on the ABS algorithms for linear systems of equations", Report DMSIA 9/93, University of Bergamo, Italy (1993).

23. Abaffy, J. and Dixon, L.C.W. "On solving sparse band systems with three algorithms of the ABS family", Technical Report 191, Numerical Optimisation Centre, Hatfield Polytechnic, Hatfield, UK (1987).

24. Abaffy, J. "ABS algorithms for sparse linear systems", in *Computer Algorithms for Solving Linear Algebraic Equations: The State of the Art*, E. Spedicato, Ed., NATO ASI Series, **F77**, pp 111–132, Springer-Verlag, Berlin (1991).

25. Zhu, M. "The implicit LL^T algorithm for sparse nested dissection linear systems", Technical Report 196, Numerical Optimisation Centre, Hatfield Polytechnic, UK (1987).

26. Yang, Z.H. and Zhu, M. "The practical ABS algorithm for large scale nested dissection linear system", Report DMSIA 11/94, University of Bergamo, Italy (1994).

27. Galantai, A. "Testing of implicit LU ABS method on large nonlinear systems with banded Jacobians", Report DMSIA 19/93, University of Bergamo, Italy (1993).

28. Abaffy, J. "Preliminary test results with some algorithms of the ABS class", Technical Report 193, Numerical Optimisation Centre, Hatfield Polytechnic, Hatfield, UK (1987).

29. Ceribelli, C. "Implementazione dell'algoritmo di Huang per sistemi lineari a bande", Dissertation, University of Bergamo, Italy (1989).

30. Bodon, E. "Numerical experiments with ABS algorithms on upper banded systems of linear equations", Report DMSIA 17/92, University of Bergamo, Italy (1992).

31. Bodon, E. "Numerical experiments with ABS algorithms on banded systems of linear equations", Report DMSIA 18/92, University of Bergamo, Italy (1992).

32. Bodon, E. "Numerical experiments with Gauss-ABS algorithms on tridiagonal systems of linear equations", Report DMSIA 31/92, University of Bergamo, Italy (1992).

33. Bodon, E. "Numerical performance of the ABS implicit LU algorithm for banded type systems of linear equations", Report DMSIA 8/93, University of Bergamo, Italy (1993).

34. Deng, N., Spedicato, E. and Vespucci, M.T. "Experiments with the ABS implicit Gauss-Cholesky algorithm on nested dissection matrices", Technical Report 1/69, Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo, Consiglio Nazionale delle Ricerche, Roma, Italy (1991).

35. Marletta, C. and Vespucci, M.T. "Vettorizzazione di condici per la soluzione di sistemi lineari con matrici ND via l'algoritmo di Gauss-Choleski implicito", Report DMSIA 3/93, University of Bergamo, Italy (1993).

36. Tuma, M. "The implicit Gauss algorithm for solving sparse unsymmetric sets of linear equations", Technical Report 1/85, Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo, Consiglio Nazionale delle Ricerche, Roma, Italy (1992).

37. Spedicato, E. and Tuma, M. "Solving sparse unsymmetric linear systems by implicit Gauss algorithm: stability", Report DMSIA 4/93, University of Bergamo, Italy (1993).

38. Benzi, M. and Meyer, C.D. "A direct projection method and its application to sparse linear systems", Report NCSU NA-01051593, North Carolina State University, Raleigh, USA (1993).

39. Spedicato, E. and Bodon, E. "Solution of linear least squares via the ABS algorithms", *Mathem. Progr.*, **58**, pp 111–136 (1993).

40. Spedicato, E. and Bodon, E. "Biconjugate algorithms in the ABS class II: numerical evaluation", *Quaderno* DMSIA 4/89, University of Bergamo, Italy (1989).

41. Spedicato, E. and Bodon, E. "Numerical behaviour of the implicit QR algorithm in the ABS class for linear least squares", *Ricerca Operativa*, **22**, pp 43–55 (1992).

42. Spedicato, E. and Bodon, E. "Computational performance of the implicit QR algorithm for linear systems on the Alliant FX80", Report DMSIA 20/93, University of Bergamo, Italy (1993).