J. Appl. Math. & Computing Vol. **17***(2005), No. 1 - 2, pp. 457 - 473*

NUMERICAL IMPLEMENTATION OF THE QMR ALGORITHM BY USING DISCRETE STOCHASTIC ARITHMETIC

FAEZEH TOUTOUNIAN, DAVOD KHOJASTEH SALKUYEH* AND BAHRAM ASADI

Abstract. In each step of the quasi-minimal residual (QMR) method which uses a look-ahead variant of the nonsymmetric Lanczos process to generate basis vectors for the Krylov subspaces induced by A, it is necessary to decide whether to construct the Lanczos vectors v_{n+1} and w_{n+1} as regular or inner vectors. For a regular step it is necessary that $D_k = W_k^T V_k$ is nonsingular. Therefore, in the floating-point arithmetic, the smallest singular value of matrix D_k, $\sigma_{min}(D_k)$, is computed and an inner step is performed if $\sigma_{min}(D_k) < \epsilon$, where ϵ is a suitably chosen tolerance. In practice it is absolutely impossible to choose correctly the value of the tolerance ϵ. The subject of this paper is to show how discrete stochastic arithmetic remedies the problem of this tolerance, as well as the problem of the other tolerances which are needed in the other checks of the QMR method with the estimation of the accuracy of some intermediate results. Numerical examples are used to show the good numerical properties.

AMS Mathematics Subject Classification : 65F10, 65G50.
Key words and phrases : Iterative methods, QMR method, look-ahead Lanczos algorithm, error propagation, CESTAC method, discrete stochastic arithmetic, CADNA library.

1. Introduction

The classical conjugate gradient method (CG) of Hestenes and Stiefel [11] is one of the most powerful iterative schemes for solving a Hermitian positive definite linear system. For general non-Hermitian matrices, the situation is less satisfactory. The biconjugate gradient (BCG) method is the "natural" generalization of the classical conjugate gradient for Hermitian positive definite matrices

Received September 24, 2003. Revised August 8, 2004. *Corresponding author.

to general non-Hermitian linear system. Unfortunately, the original BCG algorithm is susceptible to possible breakdowns and numerical instabilities. In [9], Freund and Nachtigal have presented a novel BCG-like approach for general nonsingular non-Hermitian linear system, the quasi-minimal residual algorithm (QMR), which overcomes the problems of BCG. The method uses a look-ahead variant of nonsymmetric Lanczos process to generate basis vectors for Krylov subspaces induced by A.

In section 2, we briefly describe the look-ahead Lanczos algorithm and QMR method, and we observe that in each step of the look-ahead Lanczos process, it is necessary to decide whether to construct the Lanczos vectors v_{n+1} and w_{n+1} as regular or inner vectors. As we know [9], for a regular step it is necessary that $D_k = W_k^T V_k$ is nonsingular. Therefore, in the floating-point arithmetic, the smallest singular value of matrix D_k denoted by $\sigma_{min}(D_k)$ is computed and an inner step is performed if $\sigma_{min}(D_k) < \epsilon$, where ϵ is a suitably chosen tolerance. Moreover, in section 2, we observe that there exist the stopping criterions which require the suitable tolerances. As we know, in floating-point arithmetic, it is absolutely impossible to choose correctly the value of such tolerances.

In section 3, we give a brief description of stochastic round-off error analysis, the CESTAC method and the CADNA library [4, 16].

In section 4, we describe the QMR algorithm in the discrete stochastic arithmetic, and we show that with the appropriate tests, it is possible in constructing the Lanczos vectors v_{n+1} and w_{n+1} to decide correctly, to restart the QMR algorithm if it is necessary, to stop the program as soon as a satisfactory solution or an approximate solution with desired accuracy is reached, and to save computer time, because many useless operations and iterations are not performed. Some numerical results are given to show the good numerical properties.

2. The look-ahead Lanczos and the QMR algorithms

In this section, we briefly review the look-ahead Lanczos algorithm proposed in [8], and QMR method [9]. In the following, $A \in \mathbb{C}^{N\times N}$ is always assumed to be a given $N \times N$ matrix. Unless otherwise stated, by $\|x\|$ we will mean $\|x\|_2 = \sqrt{x^H x}$.

2.1. The look-ahead Lanczos algorithm

Let $v_1, w_1 \in \mathbb{C}^N$ be any two vectors different from the zero vector. Starting with v_1, w_1, the look-ahead Lanczos algorithm generate two sequences of vectors $v_1, v_2, \ldots, v_n$ and $w_1, w_2, \ldots, w_n, n = 1, 2, \ldots$, that satisfy

$$\begin{aligned} \text{span}\{v_1, v_2, \ldots, v_n\} &= K_n(v_1, A), \\ \text{span}\{w_1, w_2, \ldots, w_n\} &= K_n(w_1, A^T), \end{aligned} \tag{1}$$

and can be grouped into $k = k(n)$ blocks

$$V_l = [v_{n_l} v_{n_l+1} \dots v_{n_{l+1}-1}], \;\; W_l = [w_{n_l} w_{n_l+1} \dots w_{n_{l+1}-1}], \;\; l = 1, 2, \dots, k-1,$$
$$V_k = [v_{n_k} v_{n_k+1} \dots v_n], \;\; W_k = [w_{n_k} w_{n_k+1} \dots w_n],$$

where

$$1 = n_1 < n_2 < \dots < n_l < \dots < n_k \le n < n_{k+1}.$$

The blocks are constructed such that we have

$$W_j^T V_l = \begin{cases} 0 & \text{if } j \neq l, \\ D_l & \text{if } j = l, \end{cases} \qquad j, l = 1, 2, \dots, k,$$

where D_l is nonsingular, $l = 1, 2, \dots, k-1$, and D_k is nonsingular if $n = n_{k+1} - 1$. The first vectors v_{n_l} and w_{n_l} in each block are called *regular* and the remaining vectors are called *inner*. The kth block is called *complete* if $n = n_{k+1} - 1$; in this case, at the next step $n+1$, a new block is started with the regular vectors $v_{n_{k+1}}$ and $w_{n_{k+1}}$. Otherwise, if $n < n_{k+1} - 1$, the kth block is *incomplete* and at the next step, the Lanczos vectors v_{n+1} and w_{n+1} are added to the kth block as inner vectors.

With these preliminaries, the basic structure of the look-ahead Lanczos algorithm is as follows.

`Algorithm 2.1: A sketch of the look-ahead Lanczos algorithm [8]`

0) `Choose` $v_1, w_1 \in \mathbb{C}^N$ `with` $\|v_1\| = \|w_1\| = 1$;
`Set` $V_1 = v_1, W_1 = w_1, D_1 = W_1^T V_1$;
`Set` $n_1 = 1, k = 1, v_0 = w_0 = 0, V_0 = W_0 = \emptyset, \rho_1 = \xi_1 = 1$;
`For` $n = 1, 2, \dots$ do:

1) `Decide whether to construct` v_{n+1} `and` w_{n+1} `as regular or inner vectors and go to 2) or 3), respectively`;

2) `(Regular step.) Compute`

$$\begin{aligned} \tilde{v}_{n+1} &= A v_n - V_k D_k^{-1} W_k^T A v_n - V_{k-1} D_{k-1}^{-1} W_{k-1}^T A v_n, \\ \tilde{w}_{n+1} &= A^T w_n - W_k D_k^{-T} V_k^T A^T w_n - W_{k-1} D_{k-1}^{-T} V_{k-1}^T A^T w_n, \end{aligned} \tag{2}$$

`set` $n_{k+1} = n+1, k = k+1, V_k = W_k = \emptyset,$ `and go to` 4);

3) `(Inner step.) Compute`

$$\begin{aligned} \tilde{v}_{n+1} &= A v_n - \zeta_{n-n_k} v_n - (\eta_{n-n_k}/\rho_n) v_{n-1} - V_{k-1} D_{k-1}^{-1} W_{k-1}^T A v_n, \\ \tilde{w}_{n+1} &= A^T w_n - \zeta_{n-n_k} w_n - (\eta_{n-n_k}/\xi_n) w_{n-1} - W_{k-1} D_{k-1}^{-T} V_{k-1}^T A^T w_n; \end{aligned} \tag{3}$$

4) `Compute` $\rho_{n+1} = \| \tilde{v}_{n+1} \|$ `and` $\xi_{n+1} = \| \tilde{w}_{n+1} \|$;
`If` $\rho_{n+1} = 0$ `or` $\xi_{n+1} = 0,$ `stop`;
`Otherwise, set`

$$\begin{aligned} &v_{n+1} = \tilde{v}_{n+1}/\rho_{n+1}, \qquad w_{n+1} = \tilde{w}_{n+1}/\xi_{n+1}, \\ &V_k = [V_k \; v_{n+1}], \quad W_k = [W_k \; w_{n+1}], \quad D_k = W_k^T V_k. \end{aligned} \tag{4}$$

Now, we list some properties of Algorithm 2.1 which will be used in the sequel. First, in view of (4), we have

$$\|v_n\| = \|w_n\| = 1, \quad n = 1, 2, \ldots. \tag{5}$$

It is convenient to introduce the notation

$$\begin{array}{ll} V^{(n)} = [v_1\ v_2\ \ldots\ v_n] & (= [V_1\ V_2\ ...\ V_k]), \\ W^{(n)} = [w_1\ w_2\ \ldots\ w_n] & (= [W_1\ W_2\ ...\ W_k]). \end{array} \tag{6}$$

Hence, by (1),

$$\begin{array}{c} K_n(v_1, A) = \{V^{(n)} z \mid z \in \mathbb{C}^N\}, \\ K_n(w_1, A^T) = \{W^{(n)} z \mid z \in \mathbb{C}^N\}. \end{array} \tag{7}$$

Moreover, the recursions for the v's in (2) and (3) can be rewritten in matrix formulation as follows:

$$AV^{(n)} = V^{(n)} H_n + [0\ \ldots\ 0\ \tilde{v}_{n+1}]. \tag{8}$$

Here,

$$H_n := \begin{bmatrix} \alpha_1 & \beta_2 & 0 & \ldots & 0 \\ \gamma_2 & \alpha_2 & \beta_3 & \ddots & \vdots \\ 0 & \gamma_3 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_k \\ 0 & \ldots & 0 & \gamma_k & \alpha_k \end{bmatrix}, \tag{9}$$

is a block-tridiagonal matrix of size $n \times n$ that is also upper Hessenberg. Furthermore, the diagonal blocks $\alpha_1, \alpha_2, \ldots, \alpha_k$ of the matrix H_n are all squares, and their sizes are just the lengths of the look-ahead steps.

2.2. The QMR method

We now consider linear systems

$$Ax = b, \tag{10}$$

with $A \in \mathbb{C}^{N \times N}$ as coefficient matrix and $b \in \mathbb{C}^N$. Let $x_0 \in \mathbb{C}^N$ be an arbitrary initial guess for the solution of (10), and set $r_0 = b - Ax_0$. The goal is to construct an approximate solutions of (10) of the form

$$x_n \in x_0 + K_n(r_0, A), \quad n = 1, 2, \ldots. \tag{11}$$

If we choose $v_1 = r_0/\|\rho_0\|$ and any $w_1 \in \mathbb{C}^N, \|w_1\| = 1$, as the starting vectors for the look-ahead Lanczos algorithm, then, by (7), the right Lanczos vectors $v_1, v_2, \ldots, v_n$ span Krylov subspace $K_n(r_0, A)$ in (11). Therefore, any iterate (11) can be presented in the form

$$x_n = x_0 + V^{(n)} z_n, \quad z_n \in \mathbb{C}^n, \tag{12}$$

and $V^{(n)}$ is the matrix defined in (6). It remains to select the free parameter z_n in (12). Based on the Lanczos matrix (9),H_n, and its extended version, $H_n^{(e)}$, defined by

$$H_n^{(e)} = \begin{bmatrix} H_n \\ \rho_{n+1}(e_n^{(n)})^T \end{bmatrix}, \quad (e_n^{(n)})^T = [0 \ \dots \ 0 \ 1]^T \in \mathbb{R}^n, \tag{13}$$

Freund and Nachtigal [9] suggested to choose z_n as the solution of the least-squares problem

$$\| \Omega_{n+1}(f_{n+1} - H_n^{(e)} z_n) \| = \min_{z \in \mathbb{C}^n} \| \Omega_{n+1}(f_{n+1} - H_n^{(e)} z) \| . \tag{14}$$

Here, $f_{n+1} = [\|r_0\| \quad 0 \quad \dots \quad 0]^T \in \mathbb{R}^{n+1}$, and

$$\Omega_{n+1} = \operatorname{diag}(\omega_1, \omega_2, \dots, \omega_{n+1}), \quad \omega_j > 0, \quad j = 1, 2, \dots, n+1, \tag{15}$$

is a weight matrix. The default choice is $\omega_j = 1$ for all j, but there are also situation where other weights are useful (see [10]). We remark that $H_n^{(e)}$ is an unreduced upper Hessenberg matrix, and hence, in contrast to H_n, the extended matrix $H_n^{(e)}$ is always guaranteed to have full column rank n. Therefore, the least-squares problem (14) always has a unique solution z_n. The motivation for this choice of z_n is as follows. By inserting the scaling matrix Ω_{n+1}, in view of (8) and (12), we can write the residual vector of any iterate (11) as follows:

$$r_n = V^{(n+1)}(\Omega_{n+1})^{-1}(\Omega_{n+1}(f_{n+1} - H_n^{(e)} z_n)). \tag{16}$$

Hence, in view of (14), the iterate x_n is characterized by a minimization of the second factor in the representation (16) of its residual r_n. This is called the *quasi-minimal residual* (QMR) property, and the resulting iterative scheme for solving linear system (10) is the QMR method.

In the QMR algorithm, the least-squares problem (14) is solved by the standard approach based on a QR decomposition of $\Omega_{n+1} H_n^{(e)}$:

$$\Omega_{n+1} H_n^{(e)} = (Q_{n+1})^H \begin{bmatrix} R_n \\ 0 \end{bmatrix}, \tag{17}$$

Here Q_{n+1} is a unitary $(n+1) \times (n+1)$ matrix, and R_n is a nonsingular upper triangular $n \times n$ matrix. By means of (17), the least-squares problem (14) can be written in the form

$$\min_{z \in \mathbb{C}^n} \|\Omega_{n+1}(f_{n+1} - H_n^{(e)} z)\| = \min_{z \in \mathbb{C}^n} \left\| \omega_1 Q_{n+1} f_{n+1} - \begin{bmatrix} R_n \\ 0 \end{bmatrix} z \right\|,$$

and thus z_n is given by

$$z_n = R_n^{-1} t_n, \ \text{ where } t_n = \begin{bmatrix} \tau_1 \\ \vdots \\ \tau_n \end{bmatrix} \in \mathbb{C}^n, \quad \begin{bmatrix} t_n \\ \tilde{\tau}_{n+1} \end{bmatrix} = \omega_1 Q_{n+1} f_{n+1}. \tag{18}$$

Furthermore, we have

$$\| \Omega_{n+1}(f_{n+1} - H_n^{(e)} z_n)\| = \mid \tilde{\tau}_{n+1} \mid . \tag{19}$$

Since $\Omega_{n+1} H_n^{(e)}$ is upper Hessenberg, the unitary matrix Q_{n+1} can be chosen as a product of n Givens rotations. The QR decomposition (17) can then be updated easily from step to step. In particular, Q_{n+1} is obtained from Q_n by a simple multiplication with one Givens rotation that modifies the last two rows only. This implies that the vector t_n in (18) differs from the previous one, t_{n-1}, only by its additional last elements τ_n. Together with (12) and (18), it follows that consecutive QMR iterates are connected by the update formula

$$x_n = x_{n-1} + \tau_n p_n, \quad \text{where } p_n = V^{(n)} R_n^{-1} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}. \tag{20}$$

The basic structure of the resulting QMR algorithm is then as follows.

Algorithm 2.2: A sketch of the QMR algorithm [9]

0) Choose $x_0 \in \mathbb{C}^N$ and set $r_0 = b - Ax_0$ and $\rho_0 = \|r_0\|_2, v_1 = r_0/\rho_0$.
   Choose $w_1 \in \mathbb{C}^N$ with $\|w_1\| = 1$.
   For $n = 1, 2, \ldots,$ do:
1) Perform the nth iteration of the Look-ahead Lanczos Algorithm 2.1; This yields matrices $V^{(n)}$, $V^{(n+1)}$ and $H_n^{(e)}$ which satisfy $AV^{(n)} = V^{(n+1)} H_n^{(e)}$.
2) Update the QR factorization (17) of $\Omega_{n+1} H_n^{(e)}$ and the vector t_n in (18).
3) Update the vector p_n in (20).
4) Set $x_n = x_{n-1} + \tau_n p_n$.
5) If x_n has converged, stop.

The update of the vector p_n in step 3) can be implemented with only short recurrences. This is due to the block tridiagonal structure of H_n; see [9] for details.

3. The CESTAC method

When some numerical algorithm is performed on a computer, each result thus provided always contains an error resulting from round-off error propagation. In this section, we briefly review the CESTAC (Control et Estimation Stochastique des Arrondis de Calcul) method [14, 16] which is able to estimate the accuracy

of the results provided by a computer, to detect the numerical instabilities occurring during the run of a scientific code, and to check the branchings that exist in the code.

3.1. Brief recall of the CESTAC method and its implementation

The basic idea of the method is defined in [13, 14] and consists in:

- performing the same code N times with a different round-off error propagation for each run.
- estimating the common part of these results and to consider that this part is representative of the exact result.

In practice, these different round-off error propagations are obtained in using random rounding mode.

Indeed, each result r of a FP operation which is not an exact floating-point value is always bounded by two floating values R^- and R^+, each of them being so representative of the exact result.

The random rounding consists at the level of each FP operation or assignment to choose as result randomly with an equal probability either R^- or R^+. Then when the same code is executed N times with a computer using this random rounding, N results $R_k, k = 1, \ldots, N$ are obtained. It has been proved in [2, 6] that, under some hypotheses, these N results belong to a quasi-Gaussian distribution centered on the exact result r. So, in practice, by considering the mean value $\overline{R}$ of the R_k as the computed result, and using Student's test, it is possible to obtain a confidence interval of $\overline{R}$ with a probability $(1-\beta)$ and then to estimate the number of exact significant digits of $\overline{R}$ by the formula (21)

$$C_{\overline{R}} = \log_{10}(\sqrt{N}|\overline{R}|/\tau_\beta \sigma), \tag{21}$$

with $\overline{R} = (1/N)\Sigma_{i=1}^{N} R_i$ and $\sigma^2 = \frac{1}{N-1}\Sigma_{i=1}^{N}(R_i - \overline{R})^2$. τ_β is the value of the Student distribution for $N-1$ degrees of freedom and a probability level $1-\beta$. In practice $N = 3, \beta = 0.05$ and then $\tau_\beta = 4.4303$.

The result provided by equation (21) is reliable if the hypotheses underlying the method hold in practice. It has been proved that [2, 6, 17], these hypotheses hold when:

1) The operands of any multiplication are both significant.

2) The divisor of any division is significant.

It is then absolutely necessary during the run of a code to control the points 1) and 2). This control is done with the concept of the informatical zero also named computational zero or computed zero [15].

Definition 1. Each result provided by CESTAC method is an informatical zero denoted by @.0 iff one of the two conditions holds:

1) $\forall i, i = 1, \ldots, N, R_i = 0$.
2) $C_{\overline{R}} \leq 0$, ($C_{\overline{R}}$ obtained with equation (21)).

When $C_{\overline{R}} \leq 0$, then $\overline{R}$ is an unsignificant value.

From the concept of @.0, discrete stochastic relations (DSR) have been defined (equality and order relations).

Definition 2. Let X and Y be N-samples provided by CESTAC method, discrete stochastic equality denoted by $s =$ is defined as:

$$X s = Y \text{ if } X - Y = @.0.$$

Definition 3. Let X and Y be N-samples provided by CESTAC method, discrete stochastic inequalities denoted by $s >$ and $s \geq$ are defined as:

$$X s > Y \text{ if } \overline{X} > \overline{Y} \text{ and } X - Y \neq @.0.$$

$$X s \geq Y \text{ if } \overline{X} \geq \overline{Y} \text{ or } X - Y = @.0.$$

The Discrete Stochastic Arithmetic (DSA) is the association of the CESTAC method, the concept of informatical zero and the discrete stochastic relations (see [5, 7, 17]). With this DSA it is possible to control the run of a scientific code, to detect the numerical instabilities and the violation of the hypotheses underlying the method. But in practice how to implement this?

As we observed, the two main specificities of the CESTAC method are:

- The random rounding, which consists in creating R^- and R^+ and in choosing randomly one or the other.
- The manner to perform the N runs of a code.

With IEEE arithmetic and the possibilities of ADA, C++, and Fortran to create new structures and to overload the operators it is easy to implement the CESTAC method.

The random rounding uses the IEEE rounding toward $+\infty$ and toward $-\infty$. These roundings occur whenever an arithmetic operation has a result that is not exact. Then none artificial round-off error is introduced in the computation. The choice of the rounding is at random with an equal probability for the $(N-1)$ first samples and the choice of the last one is the opposite of the choice of the $(N-1)$th sample.

We have seen previously that it is absolutely necessary to detect, during the run of a code, the emergence of @.0 for controlling the validity of the CESTAC method. To achieve this it suffices to use the synchronous implementation which consists in performing each arithmetic operation N times with the random rounding before performing the next. Thus for each numerical result we

have N samples, from which with equation (21) the number of significant digits of mean value, considered as the computed result, is estimated.

With this implementation the stochastic order relations defined above may also be easily created. Then during the run of a code a dynamic control may be done.

3.2. The CADNA library

The CADNA software [3, 4] is a library which implements automatically the DSA in any code written in Fortran. Using the CADNA library, each standard floating-point types have their corresponding stochastic types. Every intrinsic function and operator are overloaded for those types. When a stochastic variable is printed, only its significant digits are displayed to point out its accuracy. If a number has no significant digit (i.e., a computed zero), the symbol @.0 is displayed.

The modifications that the user has to do in his Fortran source are mainly to change the declaration statements of real type by stochastic type, and the input-output statement (see [4]). Thus, when a modified Fortran source combined with the CADNA library is run, it is as ($N = 3$) identical codes were simultaneously run on N synchronized computers each of them using the random rounding mode. So round-off error propagation can be analyzed step by step and then any numerical anomaly can be dynamically detected. This leads to the self validation of the method and a numerical debugging scientific codes.

We shall see in the numerical study how the use of CADNA library has allowed us to stabilize the code of QMR method and to reduce numerical instabilities.

4. Numerical implementation of the QMR algorithm by using CADNA library

In section 2, we observed that in each step of the look-ahead Lanczos process, it is necessary to decide whether to construct the Lanczos vectors v_{n+1} and w_{n+1} as regular or inner vectors. As we know [9], for a regular step it is necessary that $D_k = W_k^T V_k$ is nonsingular. Therefore, in implementation of QMR algorithm, in the floating-point arithmetic the smallest singular value of matrix D_k, $\sigma_{min}(D_k)$, is computed and the criterion

$$\text{if } (\sigma_{min}(D_k) < Tol) \text{ then go to step 3)}$$

is used to check whether this matrix is singular or close to singular, and to decide whether to construct the Lanczos vectors v_{n+1} and w_{n+1} as regular or inner vectors. Here Tol is a suitably chosen tolerance. The efficiency of the algorithm depends on a good choice of the Tol and to construct correctly the

Lanczos vectors v_{n+1} and w_{n+1}. In addition, if the quantity $\sigma_{min}(D_k)$ is badly computed, propagation of round-off errors will affect drastically all the computations. In order to overcome these drawbacks, we propose to introduce the discrete stochastic arithmetic in the QMR algorithm, which is able to estimate the round-off error propagation, and to detect the informatical singularity of the matrix D_k by means the following simple test,

$$\text{if } (C_{\sigma_{min}(D_k)} < 1 \text{ or } \sigma_{min}(D_k) = 0) \text{ then go to step 3).} \tag{22}$$

In this section C_X represents the number of significant digits of computed result X which is furnished by the CADNA library.

In addition, for step 5) of Algorithm 2.2 a convergence criterion is needed. In the floating-point arithmetic, for stopping the process at iteration n, we can use the following termination criterion,

$$\text{if } (\|r_n\| \leq \epsilon_1) \text{ then stop,} \tag{23}$$

where ϵ_1 is a suitably chosen tolerance. As explained in [15], in floating-point arithmetic, it is absolutely impossible to choose correctly the value of the convergence tolerance ϵ_1. It is possible, due to numerical instabilities or/and stationarity, that this stopping criterion is never satisfied. So, we need to use a termination criterion for stopping the process as soon as the desired approximate solution is reached, or the numerical instabilities or/and stationarity occur during the run of the program and the computer is not able to improve the computed solution, because of round-off error propagation. With CADNA library and using the stopping criterion

$$\text{if } (\|r_n\| \leq \epsilon_1 \text{ or } C_{\|r_n\|} < 1) \text{ then stop,} \tag{24}$$

it is possible to stop the iterative process in the above cases.

Finally, it has been shown in [9] that, in exact arithmetic, the stopping criterion in step 4) of algorithm 2.1 will be satisfied after at most N step - except in a very special situation. If $\rho_{n+1} = 0$ or $\xi_{n+1} = 0$ then $K_n(v_1, A)$ is A-invariant subspace or $K_n(w_1, A^T)$ is A^T-invariant subspace. In the first case x_n is the exact solution and the QMR algorithm must be stopped. In the second case, by restarting the QMR method and using the last available QMR iterate x_{n-1} (which is a good choice [9]) as the new initial guess, it is possible to improve the approximate solution. In floating-point arithmetic, the experiments show that when ρ_{n+1} or ξ_{n+1} have small values the method has slow convergence. So, in floating-point arithmetic for checking the sizes of ρ_{n+1} and ξ_{n+1} and deciding to restart QMR algorithm, we must use the following criteria after the stopping criterion (23),

$$\text{if } (\rho_{n+1} \leq \epsilon_2) \text{ then restart,} \tag{25}$$

$$\text{if } (\xi_{n+1} \leq \epsilon_2) \text{ then restart.} \tag{26}$$

In addition, it is necessary to check the size of $\tilde{\tau}_{n+1}$ which is defined by the relations (18) and (19). The following similar criterion allows us to check this coefficient,

$$\text{if } (\tilde{\tau}_{n+1} \leq \epsilon_2) \text{ then restart.} \tag{27}$$

The experiments show that when ϵ_2 has a small or large value the method has slow convergence. We observed that, for double precision, $\epsilon_2 = 10^{-8}$ is a good value. But, there are the cases (examples 1 and 2), in which, due to round-off errors propagation, the coefficients ρ_{n+1}, ξ_{n+1}, and $\tilde{\tau}_{n+1}$ become nonsignificant and in the same time they have a large magnitude. In this situation, it is clear that the iterations of QMR method are not able to improve the approximate solution. In order to avoid the performance of many useless operations and to prevent the instabilities which may occur, we can define with CADNA library the tests

$$\text{if } (\rho_{n+1} \leq \epsilon_2 \text{ or } C_{\rho_{n+1}} < 1) \text{ then restart,} \tag{28}$$

$$\text{if } (\xi_{n+1} \leq \epsilon_2 \text{ or } C_{\xi_{n+1}} < 1) \text{ then restart,} \tag{29}$$

$$\text{if } (\tilde{\tau}_{n+1} \leq \epsilon_2 \text{ or } C_{\tilde{\tau}_{n+1}} < 1) \text{ then restart,} \tag{30}$$

which allow us, by checking the value of the coefficients ρ_{n+1}, ξ_{n+1}, and $\tilde{\tau}_{n+1}$ to prevent the slow convergence and the instabilities which may occur and to restart QMR algorithm.

Remark. It should be mentioned here that we also used, in floating-point arithmetic, the stopping criterion

$$\text{if } (\omega_2 \leq \acute{\epsilon_1}) \text{ then stop,} \tag{31}$$

with $\omega_2 = \|r_n\|_\infty / (\|A\|_\infty \|x_n\|_1 + \|b\|_\infty)$, which has been proposed in [1], and we observed that it is absolutely impossible to choose correctly the value of the convergence tolerance $\acute{\epsilon}_1$. The following examples show that when $\acute{\epsilon}_1$ is chosen too large, the QMR process is stopped too soon, and the solution furnished is not the best that the computer may provide. When $\acute{\epsilon}_1$ is chosen too small many useless iterations are performed without improving the accuracy of the solution.

Let us now, to present the examples and the results which we obtained by the Fortran code of the QMR algorithm, with floating-point arithmetic for $Tol = 10^{-4}$ which is suggested by Parlett [12], and this code with CADNA library and the tests (22),(24), (28)-(30). $\epsilon_2 = 10^{-8}$ has been taken for all cases. Computations have been performed on a PC computer in double precision and the maximum number of iterations allowed set to 200000. For all the examples the initial guess has been $x_0 = [0,\ 0, \ldots,\ 0]^T$, and the values of $\acute{\epsilon}_1$, for stopping criterion (31), have been chosen so that the computed solutions have about the same accuracy as those corresponding to the values of ϵ_1, for stopping criterion (23).

Example 1. We consider the ill-conditioned linear system $Hx = b$ with the Hilbert matrix H ($h_{ij} = 1/(i + j - 1)$), and dimension equal to 100. The right hand side is determined so that the exact solution x is 1 everywhere. This allows an easy verification of the results. With $\epsilon_1 = 10^{-6}, 10^{-7}, 10^{-8}$, by using floating-point arithmetic and CADNA library the results obtained are presented in Tables 1 and 2, respectively. The results obtained, by using floating-point arithmetic and stopping criterion (31) with $\acute{\epsilon}_1 = 0.56 \times 10^{-9}, 0.41 \times 10^{-10}, 0.44 \times 10^{-11}$, are also presented in columns 3, 5, 7 of the Table 1, respectively.

Table 1. The results of the example 1, using floating-point arithmetic

	$\epsilon_1 = 10^{-6}$	$\acute{\epsilon}_1 = 0.56 \times 10^{-9}$	$\epsilon_1 = 10^{-7}$	$\acute{\epsilon}_1 = 0.41 \times 10^{-10}$	$\epsilon_1 = 10^{-8}$	$\acute{\epsilon}_1 = 0.44 \times 10^{-11}$
$x(1)$	1.0001660	1.0001635	0.9999542	0.9999539	1.0000064	1.0000063
$x(2)$	0.9982230	0.9982621	1.0007666	1.0007826	0.9998362	0.9998376
$x(3)$	1.0024256	1.0023395	0.9977080	0.9976219	1.0008133	1.0008104
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$x(98)$	0.9935930	0.9936288	0.9968849	0.9968086	0.9986009	0.9985950
$x(99)$	0.9928798	0.9929182	0.9964407	0.9963570	0.9983446	0.9983380
$x(100)$	0.9921452	0.9921863	0.9959814	0.9958868	0.9980741	0.9980667
RN	1385	1403	6405	6068	59332	58525
TN	4519	4574	19579	18568	138312	136700
$\|r_n\|$	0.999×10^{-6}	0.998×10^{-6}	1.0×10^{-7}	1.06×10^{-7}	0.996×10^{-8}	1.04×10^{-8}

Table 2. The results of the example 1, using CADNA library

	$\epsilon_1 = 10^{-6}$	$\epsilon_1 = 10^{-7}$	$\epsilon_1 = 10^{-8}$
$x(1)$	1.0002	1.0000	1.0000
$x(2)$	0.9978	0.999	1.00
$x(3)$	1.003	0.999	0.999
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$x(98)$	0.9932	0.995	0.995
$x(99)$	0.9924	0.994	0.995
$x(100)$	0.9917	0.994	0.994
RN	1342	3336	3886
TN	3808	6036	6669
$\|r_n\|$	0.1×10^{-5}	0.6×10^{-6}	0.4×10^{-6}

At last lines of these Tables we can find the corresponding number of restart of QMR Algorithm, the total number of iterations, and residual norm; which are denoted by RN, TN, and $\|r_n\|$, respectively. As we observe, for different values of ϵ_1 (and the corresponding values of $\acute{\epsilon}_1$) the results corresponding to the floating-point arithmetic and CADNA library have not significantly difference. But with

CADNA library the solution was reached with $TN = 3808, 6036, 6669$ iterations which are very less than those needed with floating-point arithmetic, $TN = 4519, 19579, 138312$ (and $TN = 4574, 18568, 136700$ for the corresponding values of $\acute{\epsilon}_1$), respectively. It must be noted that with $\epsilon_1 = 10^{-9}$ (and $\acute{\epsilon}_1 = 0.5 \times 10^{-12}$) and floating-point arithmetic, no solution has been obtained after 200000 allowed iterations. On the contrary, with $\epsilon_1 = 10^{-9}, 10^{-10}, \ldots$ and CADNA library, the solution which is similar to that of $\epsilon_1 = 10^{-8}$, has been obtained at the iteration 6841st with $\|r_n\| = @.0$. The $\|r_n\| = @.0$ shows that, after the iteration 6841st, the computer is not able to distinguish the vector r_n from the null vector and to improve the computer solution, because of the round-off error propagation. So for this example, the algorithm using the CADNA library and appropriate tests is more efficient than that using floating-point and we observe that the CADNA library is able to stabilize the algorithm and to save computer time, because many useless operations are not performed.

Table 3. The results of the example 2, using CADNA library

$x(1)$	1.0	$x(51)$	1.0
$x(2)$	1.0	$x(52)$	1.0
$x(3)$	1.0	$x(53)$	1.0
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$x(25)$	1.01	$x(75)$	1.01
$x(26)$	1.01	$x(76)$	1.01
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$x(48)$	1.0	$x(98)$	1.0
$x(49)$	1.0	$x(99)$	1.0
$x(50)$	1.0	$x(100)$	1.1

Table 4. The results of the example 2, using floating – point arithmetic

$x(1)$	1.11	$x(51)$	1.03
$x(2)$	1.09	$x(52)$	1.03
$x(3)$	1.07	$x(53)$	1.03
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$x(25)$	0.964	$x(75)$	0.967
$x(26)$	0.967	$x(76)$	0.964
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$x(48)$	1.03	$x(98)$	1.07
$x(49)$	1.03	$x(99)$	1.09
$x(50)$	1.03	$x(100)$	1.11

Example 2. We consider the ill-conditioned linear system $Ax = b$ with the coefficients matrix A with $a_{ij} =\mid i-j \mid$, and different dimensions. The right hand side is determined so that the exact solution x is 1 everywhere. With $\epsilon_1 = 10^{-1}$ (and $\epsilon_1 = 10^{-2}$), and dimension $N \leq 8$ (and $N \leq 5$, respectively), by using floating-point arithmetic, we obtained the approximate solution with desired accuracy. For these cases the results obtained with CADNA library are similar to those of floating-point arithmetic. For smaller ϵ_1 and larger dimension no solution has been obtained with floating-point arithmetic, because an arithmetic exception occurred during the run of code. On the contrary, with CADNA

library which detects the numerical instabilities the code has been run without any difficulty. For example, for dimension equal to 100 and $\epsilon_1 = 10^{-1}$ the result obtained, which has $\|x - x_n\|_\infty = 0.1$, is presented in Table 3.

Table 5. The results of the example 3, using floating − point arithmetic

	$\epsilon_1 = 10^{-6}$ and $\acute{\epsilon}_1 = 10^{-10}$	$\epsilon_1 = 10^{-12}$ and $\acute{\epsilon}_1 = 0.9 \times 10^{-16}$	$\epsilon_1 = 10^{-16}, 10^{-17}$ and $\acute{\epsilon}_1 = 10^{-19}, 10^{-20}$
$x(1)$	0.99999999602	1.0000000000000000000	1.0000000000000000000
$x(2)$	0.99999999934	0.99999999999999966693	1.0000000000000000000
$x(3)$	0.99999999994	1.0000000000000004441	1.0000000000000000000
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$x(798)$	1.00000000208	1.0000000000000000000	1.0000000000000000000
$x(799)$	0.99999999340	1.0000000000000002220	1.0000000000000002220
$x(800)$	1.00000000260	0.99999999999999988898	0.99999999999999988898
RN	13	67	113
TN	202	273	319
$\|r_n\|$	0.805×10^{-6}	0.937×10^{-12}	0.0

Table 6. The results of the example 3, using CADNA library

	$\epsilon_1 = 10^{-6}$	$\epsilon_1 = 10^{-12}$	$\epsilon_1 = 10^{-16}, 10^{-17}$
$x(1)$	0.99999999602	1.00000000000000	1.00000000000000
$x(2)$	0.99999999934	0.999999999999999	1.00000000000000
$x(3)$	0.99999999994	1.00000000000000	1.00000000000000
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$x(798)$	1.0000000021	1.00000000000000	1.00000000000000
$x(799)$	0.99999999340	1.00000000000000	1.00000000000000
$x(800)$	1.0000000026	0.999999999999999	1.00000000000000
RN	13	67	105
TN	203	274	312
$\|r_n\|$	0.805×10^{-6}	0.938×10^{-12}	@.0

It is necessary to mention that for all values of $\epsilon_1 \leq 10^{-1}$ the same results have been obtained, because, in these cases, the process was stopped by the stopping criterion (24) at iteration 44th with $\|r_n\| = @.0$. By using the **old-type** function which exists in the CADNA library we obtained the classical type value of residual norm $\|r_n\| = 44.9561$ with $C_{\|r_n\|} = 0$ which shows that this residual norm has no significant digit. When the process runs with floating-point

arithmetic, the computer is not able to detect this numerical instability and the value of residual norm becomes larger and larger. In this case, the stopping criterion (23) was never satisfied, and an arithmetic exception occurred during the run of code.

For this example, we also used the stopping criterion (31). The computed solution obtained with $\acute{\epsilon}_1 = 0.16 \times 10^{-3}$ is presented in Table 4. This solution has been obtained at second iteration with $\|r_n\| = 42.409$. As we observe, this computed solution is less accurate than that obtained with CADNA library. With $\acute{\epsilon}_1 = 0.15 \times 10^{-3}$ (and $\acute{\epsilon}_1 = 0.15 \times 10^{-4}$) we could only obtain the solution for $N \leq 6$ (and $N \leq 4$, respectively).

As we observe, for this example, the algorithm using the CADNA library is also more efficient than that using floating-point arithmetic.

Example 3. We consider

$$A = \begin{bmatrix} 2 & 1 & & & & \\ 0 & 2 & 1 & & & \\ 1 & 0 & 2 & 1 & & \\ & \ddots & \ddots & \ddots & \ddots & \\ & & 1 & 0 & 2 & 1 \\ & & & 1 & 0 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} 3 \\ 3 \\ 4 \\ \vdots \\ 4 \\ 3 \end{bmatrix},$$

with dimension equal to 800. The exact solution is given by $[1, 1, \ldots, 1]^T$. The results obtained by using the floating-point arithmetic and CADNA library with different values of ϵ_1 (and corresponding values of $\acute{\epsilon}_1$) are presented in Tables 5 and 6, respectively.

As we observe, for this example which is a well-conditioned system, the similar solutions have been obtained. In addition, with $\epsilon_1 = 10^{-12}, 10^{-16}, 10^{-17}$, all the digits of the solutions obtained by using CADNA library are significant. The residual norm $\|r_n\| = @.0$ shows that the solution obtained with $\epsilon_1 = 10^{-16}, 10^{-17}$, is a satisfactory informatical solution, and this solution was reached with 312 iterations which is less than that needed with floating-point arithmetic, $NT = 319$. As we observe, for this example with a well conditioned system, the algorithm using the CADNA library is as efficient as that using floating-point arithmetic.

5. Conclusion

In this paper we have seen, in the floating-point arithmetic, that the QMR method has to face the two inherent difficulties:

- How to determine the singularity or near singularity of matrix D_k?

- How can the iterative process be restarted or stopped correctly?

We observed that the use of CADNA library allows us to solve these problems. It has been shown that it is possible, on the one hand, by using the number of significant digits of $\sigma_{min}(D_k)$ which is furnished by CADNA library, to determine correctly the informatical singularity of matrix D_k, if it is, and to prevent the numerical instabilities which may occur, and, on the other hand, by using the number of significant digits of some intermediate coefficients and residual norm in the appropriate tests, to continue the iterations, to restart the QMR method if it is necessary, to stop correctly the iterative process, to detect numerical instabilities, to prevent an arithmetic exception which may occur, and to save computer time, because many useless iterations are not performed. The numerical experiments show that, the total number of iterations in the run of iterative process with CADNA library is a reasonable number versus that needed with floating-point arithmetic, and QMR algorithm with CADNA library is a stable algorithm for ill-conditioned systems as well as for well conditioned systems. Consequently, QMR algorithm with CADNA library is a stable and efficient algorithm and can be used without any difficulty for solving large nonsymmetric systems of linear equations.

6. Acknowledgements

We would like to thank Professor J. Vignes, Professor J. M. Chesneaux, and the anonymous referee for advise on many aspects of this work.

References

1. M. Arioli, I. S. Duff and D. Ruiz, *Stopping criteria for iterative solvers,* SIAM J. Matrix Anal. Appl. **13** (1992), 138-144.
2. J. M. Chesneaux, *Study of the computing accuracy by using probabilistic approach,* Contribution to Computer Arithmetic and Self Validating Numerical methods, IMACS, New Brunswick, NJ, (1990), 19-30.
3. J. M. Chesneaux, *CADNA: An ADA tool for round-off errors analysis and for numerical debugging,* Congress on ADA in Aerospace, Barcelon, (1990), 390-396
4. J. M. Chesneaux, *Descriptif d'utilisation du logiciel CADNA-F,* MASI Report, No. 92-32(1992), 855-860.
5. J. M. Chesneaux, *Stochastic arithmetic properties,* Computational and Applied Mathematics, I-Algorithms and Theory, edited by C. Brezinski (North Holland, Amsterdam, 1992), 81-91.
6. J. M. Chesneaux and J. Vignes, *Sur la robustesse de la méthode CESTAC,* C. R. Acad. Sci. Paris, Sér. I, Math. **307** (1988), 855-860.
7. J. M. Chesneaux and J. Vignes, *Les fondements de l'arithmétique stochastique,* C. R. Acad. Sci. Paris, Sér. I, Math. **315** (1992), 1435-1440.

8. R. W. Freund, M. H. Gutknecht and N. M. Nachtigal, *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices,* SIAM J. Sci. Comput. **14** (1993), 137-158.
9. R. W. Freund and N. M. Nachtigal, QMR:*a quasi-minimal residual method for non-Heritian linear systems,* Numer. Math. **60** (1991), 315-339.
10. R. W. Freund and T. Szeto, *A transpose-free quasi-minimal residual squared algorithm for non-Heritian linear systems.* In Advances in Computer Methods for Partial Differential Equations-VII, R. Vichnevetsky, D. Knight, and G. Richter, Eds. IMACS (1992), 258-264.
11. M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems,* J. Res. Nat1. Bur. Stand **49** (1952), 409-436.
12. B. N. Parlett, *Reduction to tridiagonal form and minimal realizations,* Preprint, Berkeley, January (1990), 409-436.
13. J. Vignes, *Error analysis in computing,* International Federation for Information Processing congress. Proceedings, Stockholm, (1974), 610-614
14. J. Vignes, *New methods for evaluating the validity of the results of mathematical computations,* Math. Comp. Simul, Vol. **20**(1978), 227-249.
15. J. Vignes, *Zéro mathématique et zéro informatique,* C. R . Acad. Sci., Paris, sér I Math. **303** (1986), 997-1000; also La Vie des Sciences **4** (1987), 1-13.
16. J. Vignes, *A stochastic arithmetic for reliable scientific computation,* Math. Comp. Simul. **35** (1993), 233-261.
17. J. Vignes, *A stochastic approach to the analysis of round-off error propagation,* A Survey of the CESTAC Method. Proceeding of Real Numbers and Computer Conference. Marseille, (1996), 233-251.

Faezeh Toutounian received her B. Sc in Mathematics from Ferdowsi University of Mashhad, Iran, two degree of M. Sc in Mathematical statistics and applied computer and her Ph. D in Mathematics from Paris VI University, France. She spent two sabbatical years in 1985 and 1996 at Paris VI University. She is currently a professor of Mathematics at Ferdowsi University of mashhad. Her research interests are mainly numerical linear algebra, iterative methods and error analysis.

Department of Mathematics, School of Mathematical Sciences, Ferdowsi University of Mashhad, P.O. Box 1159-91775, Mashhad, Iran
e-mail: toutouni@math.um.ac.ir

Davod Khojasteh Salkuyeh received his B. Sc from Sharif University of Technology, Tehran, Iran and his M. Sc from Ferdowsi University of Mashhad, Mashhad, Iran. He received his Ph. D degree under supervision of professor Faezeh Toutounian at Ferdowsi University of Mashhad in 2003. He is currently an assistant professor of Mathematics at Mohaghegh Ardabili University of Ardabil, Iran. His research interests are mainly iterative methods for sparse linear systems and finite element method.

Department of Mathematics, Mohaghegh Ardabili University, P. O. Box. 56199-11367, Ardabil, Iran
e-mail: khojaste@uma.ac.ir & khojaste@math.um.ac.ir

Bahram Asadi received his M. Sc degree under supervision of professor Faezeh Toutounian at Ferdowsi University of Mashhad, Iran. He is currently a lecturer in Islamic Azad Univrsity of Hamadan, Iran. His research interest is mainly iterative methods for sparse linear systems.

Department of Mathematics, Islamic Azad University of Hamadan, Hamadan, Iran
e-mail: brasadi@yahoo.com