Breakdown and near-breakdown control in the CGS algorithm using stochastic arithmetic

Jean-Marie Chesneaux[*] Ana C. Matos[†]

Abstract

In the Conjugate-Gradient-Squared method, a sequence of residuals r_k defined by $r_k = P_k^2(A)r_0$ is computed. Coefficients of the polynomials P_k may be computed as ratio of scalar products from the theory of formal orthogonal polynomials. When a scalar product in a denominator is zero or very affected by round-off errors, situations of breakdown or near-breakdown appear. Using floating point arithmetic on computers, such situations are detected with the use of ε_i in some ordering relations like $|x| \leq \varepsilon_i$. The user has to choose the ε_i himself and these choices conditionate entirely the efficient detection of breakdown or near-breakdown. The subject of this paper is to show how stochastic arithmetic eliminates the problem of the ε_i with the estimation of the accuracy of some intermediate results.

Keywords: Lanczos methods, othogonal polynomials, breakdown, near-breakdown, stochastic arithmetic, accuracy.

Subject Classification: AMS(MOS):.

1 Introduction

Let us consider in $\mathbb{C}^n$ the system of linear equations $Ax = b$ where A is a nonsingular matrix. The Lanczos method [12, 13] for solving this system consists in constructing a sequence of vectors (x_k) in the following way:

- choose two arbitrary nonzero vectors x_0 and y;
- set $r_0 = b - Ax_0$;
- determine x_k such that
  - $x_k - x_0 \in E_k = \text{span}(r_0, Ar_0, \cdots, A^{k-1}r_0)$, that means we can write

$$x_k - x_0 = -b_1^{(k)}r_0 - \cdots - b_k^{(k)}A^{k-1}r_0;$$

[*]Laboratoire MASI-IBP - U.R.A. CNRS 818, Université Pierre et Marie Curie, 4 Place Jussieu, 75252 Paris cedex 05, France, E-mail : chesneaux@masi.ibp.fr

[†]Laboratoire d'Analyse Numérique et d'Optimisation, UFR IEEA - M3, Université des Sciences et Technologies de Lille, 59655 Villeneuve d'Ascq cedex, France, E-mail:matos@ano.univ-lille1.fr

– $r_k = b - Ax_k \perp F_k = \text{span}(y, A^*y, \cdots, A^{*k-1}y)$, where A^* is the conjugate transpose of A, which is equivalent to the following orthogonality conditions

$$(A^{*i}y, r_k) = 0 \quad \text{for } i = 0, \cdots, k-1. \tag{1}$$

If we set

$$P_k(\zeta) = 1 + b_1^{(k)}\zeta + \cdots + b_k^{(k)}\zeta^k$$

then we have

$$r_k = P_k(A)r_0.$$

Moreover, if we define the linear functional c on the space of polynomials by

$$c(\zeta^i) = (y, A^i r_0), \quad i = 0, 1, \cdots,$$

then the orthogonality conditions (1) can be written in the form

$$c(\zeta^i P_k) = 0 \quad \text{for } i = 0, \cdots, k-1.$$

This approach, which was developed in [4, 3], shows that P_k is the orthogonal polynomial of degree at most k belonging to the family of formal orthogonal polynomials with respect to c [1]. This polynomial is defined apart from a multiplying factor which in our case is chosen such that $P_k(0) = 1$. So r_k can be computed if P_k exists and the existence and uniqueness of P_k is determined by the condition

$$H_k^{(1)} = \begin{vmatrix} (y, Ar_0) & (y, A^2r_0) & \cdots & (y, A^k r_0) \\ (y, A^2r_0) & (y, A^3r_0) & \cdots & (y, A^{k+1}r_0) \\ \vdots & \vdots & & \vdots \\ (y, A^k r_0) & (y, A^{k+1}r_0) & \cdots & (y, A^{2k-1}r_0) \end{vmatrix} \neq 0.$$

The variant of Lanczos method proposed by Sonneveld [14] – the *the Conjugate Gradient Squared method* (CGS in short) is the method that we will consider in this paper. It consists in computing

$$r_k = P_k^2(A)r_0,$$

in order to avoid the computation of A^*. Lanczos method, as well as CGS, have the property of finite termination, namely

$$\exists\ k \leq n \text{ such that } r_k = 0 \text{ and } x_k = x = A^{-1}b.$$

The polynomials P_k can be computed recursively in different ways leading to various Lanczos type methods [4]. The coefficients of the different recurrence relations for the computation of the orthogonal polynomials P_k are given by ratios of scalar products. When a scalar product in the denominator is zero, which corresponds to the non existence or non uniqueness of some polynomial, a situation of breakdown occurs in the algorithm and we have to stop. To avoid breakdown, algorithms based on formulas that jump over the non-existing polynomials and compute only the existing ones have been proposed (for the description of these algorithms and theoretical background see [2]). The orthogonal polynomials approach of Lanczos type methods has two main advantages: it makes the presentation of the methods clear and simple and gives formulae for avoiding breakdown.

But when those scalar products appearing in denominators are badly computed, then rounding errors can affect seriously all the computations. This is a typical situation of near-breakdown. In scientific computations, due to finite precision, we can't distinguish between breakdown and near breakdown. Recurrence formulas for orthogonal polynomials in order to avoid near-breakdown in the CGS-algorithm have been proposed in [4] as well as an algorithm implementing them - the BSMRZS algorithm. The examples given there show how interesting and important these techniques are in order to obtain good numerical results from the CGS-algorithm. In the BSMRZS-algorithm, some choices have to be made: when to jump and what is the size of the jump. These decisions are made based on the size of some computed quantities and so some ϵ_i's have to be given. The main problem is how to choose these ϵ_i's : a bad choice can lead to bad results. In fact, what is useful to know to decide on the jumps is the accuracy of intermediate computations. But this is impossible with the floating point arithmetic.

The aim of this paper is to use the stochastic arithmetic to try to solve all this problems, avoiding the choice of the different ϵ_i's and deciding on the jump from the precision with which intermediate quantities are computed. The use of this arithmetic can detect some near-breakdowns that can not be controlled by ϵ_i's. We will see that letting the computer decide whether to jump or not and how far, we will obtain the best results for all the examples given in [4] (considering different choices for all the ϵ's given there).

We will begin by recalling the principal recurrence relations on what the BSMRZS-algorithm is based and making a brief description of this algorithm, as well as the near-breakdown tests used there. Then, after a simple description of the stochastic arithmetic, we will describe the BSMRZS-algorithm using the stochastic arithmetic, pointing out all the advantages. Some numerical results will be given to show the good numerical properties.

2 The orthogonal polynomials approach of the CGS algorithm

2.1 Theoretical results

The CGS algorithm consists in computing the quantities

$$r_k = P_k^2(A) r_0, \tag{2}$$

and the BSMRZS algorithm is based on the recursive computation of these quantities using the recurrence relations for the orthogonal polynomials. Let us begin by recalling the recurrence relations that enable us to avoid the near-breakdown situation. See [4] for the details.

As we shall only consider the existing polynomials (called regular), we won't give a name to the other ones and so we shall denote P_k the polynomial of degree at most n_k satisfying the orthogonality conditions

$$c(\zeta^i P_k(\zeta) = 0 \quad \text{for } i = 0, \cdots, n_k,$$

normalized by the condition $P_k(0) = 1$, and $P_k^{(1)}$ the regular monic polynomial of degree n_k belonging to the family of formal orthogonal polynomials with respect to the functional $c^{(1)}$ defined by $c^{(1)}(\zeta^i) = c(\zeta^{i+1})$ (we remark that the existence and uniqueness condition is the same that the one for P_k, that is, $H_{n_k}^{(1)} \neq 0$). So $P_k^{(1)}$ satisfies the orthogonality conditions

$$c^{(1)}(\zeta^i P_k^{(1)}(\zeta)) = 0 \ \text{ for } i = 0, \cdots, n_k - 1.$$

If this quantity is zero for higher values of i and if $P_k^{(1)}$ satisfies

$$c^{(1)}(\zeta^i P_k^{(1)}(\zeta)) = 0 \ \text{ for } i = 0, \cdots, n_k + m_k - 2; c^{(1)}(\zeta^{n_k+m_k-1} P_k^{(1)}(\zeta)) \neq 0,$$

then the next regular orthogonal polynomial $P_{k+1}^{(1)}$ has degree $n_{k+1} = n_k + m_k$. To jump over the non existing polynomials we can use the recurrence relations for the computation of $P_k, P_k^{(1)}$ given by Draux [11].

But if $\left|c^{(1)}(\zeta^{n_k+m_k-1}P_k^{(1)})\right|$ is badly computed, the coefficients in the recurrence relations can also be very badly computed, so that round-off errors could affect drastically the algorithm. The same is verified for the quantities $\left|c^{(1)}(P_k^{(1)})\right|$, $i = n_k, \cdots n_k + m_k - 2$. In this case no breakdown occurs, but we are in presence of numerical instability, a situation called near-breakdown. Recurrence formulas for P_k and $P_k^{(1)}$ that enable to jump over those polynomials that could be badly computed and compute directly the first polynomial following them have been deduced in [4]. It is shown that

$$P_{k+1}(\zeta) = P_k(\zeta) - \zeta w_k(\zeta)P_k^{(1)}(\zeta) - \zeta v_k(\zeta)P_k(\zeta) \tag{3}$$

with

$$\begin{aligned} w_k(\zeta) &= \gamma_0 + \cdots + \gamma_{m_k-1}\zeta^{m_k-1}; \\ v_k(\zeta) &= \gamma_0' + \cdots + \gamma_{l_k}'\zeta^{l_k} \text{ where } l_k = n_k - 1 \text{ if } n_k < m_k - 1 \text{ and } l_k = m_k - 2 \text{ otherwise .} \end{aligned}$$

$$P_{k+1}^{(1)}(\zeta) = q_k(\zeta)P_k^{(1)}(\zeta) + t_k(\zeta)P_k(\zeta) \tag{4}$$

with

$$\begin{aligned} q_k(\zeta) &= \eta_0 + \cdots + \eta_{m_k-1}\zeta^{m_k-1}; \\ v_k(\zeta) &= \eta_0' + \cdots + \eta_{l_k}'\zeta^{l_k} \text{ where } l_k = n_k - 1 \text{ if } n_k < m_k \text{ and } l_k = m_k - 1 \text{ otherwise .} \end{aligned}$$

The coefficients $\gamma_i, \gamma_i', \eta_i, \eta_i'$ are the solution of two linear systems given in [4].

It is useful to write down explicitly the previous relations in the case where $m_k = 1$ because the tests for breakdown and near-breakdown are based on the quantities appearing there. So we have

$$P_{k+1}(\zeta) = P_k(\zeta) - \gamma_0\zeta P_k^{(1)}(\zeta), \text{ where } \gamma_0 = \frac{c(P_k^{(1)}P_k)}{c(\zeta P_k^{(1)^2})} \tag{5}$$

and

$$P_{k+1}^{(1)}(\zeta) = (\zeta + \eta_0)P_k^{(1)}(\zeta) + \eta_0' P_k(\zeta),$$

where

$$\eta_0 = \frac{c(\zeta P_k^{(1)}P_k)}{c(P_k^{(1)}P_k)} - \frac{c^{(1)}(\zeta P_k^{(1)^2})}{c^{(1)}(P_k^{(1)^2})}, \eta_0' = -\frac{c(\zeta P_k^{(1)^2})}{c(P_k^{(1)}P_k)}. \tag{6}$$

2.2 The BSMRZS algorithm in floating point arithmetic

Squaring the recurrence relations (3) and (4) and introducing the auxiliary quantities $z_k = P_k^{(1)^2}(A)r_0$ and $s_k = P_k(A)P_k^{(1)}(A)r_0$, we will obtain the BSMRZS algorithm which consists in computing recursively:

$$\begin{aligned} r_{k+1} &= (I - Av_k(A))^2 r_k - 2(I - Av_k(A))Aw_k(A)s_k + A^2 w_k^2(A)z_k \\ x_{k+1} &= x_k - (Av_k(A) - 2I)v_k(A)r_k + 2(I - Av_k(A))w_k(A)s_k - Aw_k^2(A)z_k \\ z_{k+1} &= q_k^2(A)z_k + 2q_k(A)t_k(A)s_k + t_k^2(A)r_k \\ s_{k+1} &= (q_k(A) - Aq_k(A)v_k(A) - At_k(A)w_k(A))s_k - Aq_k(A)w_k(A)z_k + \\ &\quad +t_k(A)(I - Av_k(A))r_k. \end{aligned}$$

These general formulas require at each step the computation of the coefficients of the polynomials $v_k(\zeta)$, $w_k(\zeta)$, $q_k(\zeta)$ and $t_k(\zeta)$. In the particular case where $m_k = 1$, that is when we don't need a jump, they simplify and only the quantities γ_0, η_0 and $\eta_0^{'}$ defined by (5) and (6) have to be computed. This algorithm is breakdown free (apart from the incurable hard one, which numerically almost never arises).

Let us now consider the breakdown and near-breakdown tests proposed in [4]. There are two kinds of tests.

2.2.1 Tests for the treatment of the theoretical breakdown

At each step, we have to test if we have reached the solution and, if not, what is the length m_k of the jump (eventually $m_k = 1$) to go to the next step. At step k, the value of m_k is determined from two conditions:

1. computation of the coefficients of the recurrence relations: initially $m_k = 1$; if one of the quantities γ_0, η_0 or $\eta_0^{'}$ is not defined because of a division by zero, then we increase the value of m_k until we get two systems for the computation of the coefficients of the polynomials $v_k(\zeta), w_k(\zeta), t_k(\zeta)$ and $q_k(\zeta)$ that are nonsingular;

2. possibility of performing the next step, that is, there will be no division by zero in the beginning of the next step.

There is a quantity defined in [4] by

$$\sigma_{k+1}^{(m_k)} = c(\zeta^{m_k} P_k^{(1)} P_{k+1}) = (y, s_{k+1})$$

which is very useful in verifying **2.**. In fact, if $\sigma_{k+1}^{(m_k)} = 0$ then

- we'll obtain the solution at the end of the current iteration or
- $c(\zeta^{n_{k+1}} P_{k+1}) = 0$, which means that we'll have a jump at the beginning of the next iteration and we'll have to solve a system where $\sigma_{k+1}^{(m_k)}$ is a pivot.

So to ensure **2.** we need to have $\sigma_{k+1}^{(m_k)} \neq 0$.

The practical implementation of these theoretical breakdowns consists in testing if $\left|\sigma_{k+1}^{(m_k)}\right| \leq \epsilon_1$ and if the absolute value of the pivots found when solving the systems for the coefficients is less than a small quantity ϵ_2.

2.2.2 Tests for the treatment of near-breakdown

As we have remarked above, if the previous quantities are badly computed, propagation of round-off errors will affect drastically all the computations. In order to try to avoid this situation other tests are proposed in [4], based on the order of magnitude of some coefficients.

These tests for breakdown and near-breakdown have the disadvantage of the choice of the different ϵ_i's. The efficiency of the algorithm depends on a good choice of the ϵ_i's in order to obtain the correct jumps. Moreover, some near-breakdowns cannot be expressed in terms of ϵ_i's. We have to avoid the introduction of some badly computed quantities whithout knowing their order of magnitude. It is to overcome these drawbacks that we propose to introduce the stochastic arithmetic in the BSMRZS algorithm. We will begin by a brief description of this arithmetic.

3 The stochastic arithmetic

The stochastic arithmetic [6, 8, 17] is a modelization of the synchronous implementation of the CESTAC method [15, 17]. The aim of the CESTAC method, based on the probabilistic approach of round-off errors, is to estimate the effect of the propagation of the round-off errors on every computed results obtained with the floating point arithmetic. It consists in making the round-off errors propagate in different ways in order to distinguish between a stable part of the mantissa, considered as the significant one, and an unstable part considered as non-significant.

In the probabilistic approach, round-off errors are considered as independent, identically-distributed random variables. The probabilistic approach takes into account compensation of round-off errors.

The first basic idea of the CESTAC method is to replace the usual floating-point arithmetic by a random arithmetic. The random arithmetic is obtained from the usual floating-point arithmetic by randomly perturbing the lowest-weight bit of the mantissa of the result of each arithmetic operation. The second basic idea is to perform several times a code with this new arithmetic to obtained different results for each run.

In practice, the use of the CESTAC method consists in:

(i) running a same program N times in parallel with the random arithmetic, consequently, for each intermediate result R of any floating-point arithmetic operation, a set of N different computed results R_i, $i = 1, .., N$ is obtained,

(ii) taking the mean value $\overline{R} = \frac{\sum_{i=1}^{N} R_i}{N}$ as the computed result,

(iii) using Student's distribution to estimate a confidence interval for R, and then computing the number $C_{\overline{R}}$ of significant digits of $\overline{R}$ (i. e. the common digits between $\overline{R}$ and the exact result r) defined by $C_{\overline{R}} = log_{10}\left(\frac{\sqrt{N}.|\overline{R}|}{\tau_\beta . s}\right)$ with $s = \sqrt{\frac{\sum_{i=1}^{N}\left(R_i - \overline{R}\right)^2}{N-1}}$, τ_β being the value of Student's variable t for $N - 1$ degrees of freedom and probability β. $N = 2$ or $N = 3$ is sufficient to have a efficient use of this method that makes all its interest.

The validity of this method has been proven under some hypotheses which generally hold in real-life problems [5]. The hypotheses can be controlled during the run.

The first application of the CESTAC method is to compute the number of exact significant digits of the computed results, but the possibility of knowing the accuracy of results leads to a new arithmetic : the **stochastic arithmetic** [6, 8, 17]. The stochastic arithmetic may also be seen as a modelization of floating-point arithmetic with control of accuracy. In stochastic arithmetic, order relations and the notion of equality must be redefined to take into account the accuracy of operands.

More precisely, it has been proved [5] that, under certain regular conditions, every computed result R obtain with the random arithmetic can be modelled by $R = r + \sum_{i=1}^{n} u_i(d) 2^{-p} z_i$ where the $u_i(d)$ are constants depending only on the data d and the z_i are identically distributed and centered random independent variables. The integer p is the number of bits in the mantissa and n the number of operations. This is a first order approximation with respect to 2^{-p}.

The main features of this model are: i) the distribution of R is approximately normal, ii) the mean value of R is the exact result r.

Consequently, the set $\mathcal{S}$ of stochastic numbers is defined as the set of Gaussian random variables. Each stochastic number X is denoted by (m, σ^2) where m is the mean value of X and σ its standard deviation. Therefore, a stochastic number (m, σ^2) represents the mathematical value m.

As with the CESTAC method, we can define a concept of accuracy for stochastic numbers.

If $X = (m, \sigma^2)$ is a stochastic number, there exists λ_β, depending only on β, such that

$$P\left(X \in [m - \lambda_\beta.\sigma, m + \lambda_\beta.\sigma]\right) = 1 - \beta.$$

The number of significant digits of X is defined in $\overline{\mathbb{R}}$ by $C_{\beta,X} = log_{10}\left(\frac{|m|}{\lambda_\beta.\sigma}\right)$ for $X \neq (0,0)$ and by $+\infty$ for $(0,0)$.

Briefly, as for the computer zero introduced in [16] and defined by $C_{\overline{R}} \leq 0$, a new concept of **stochastic zero** is defined by $C_{\beta,X} \leq 0$ or $X = (0,0)$. A stochastic zero is a value which represents the mathematical zero or which is without significance.

The stochastic elementary arithmetic operations are defined as operations between gaussian independent random variables at the first order with respect to σ/m. Stochastic operations are noted $(s+, s-, s*, s/)$. For instance, $X_1 \; s- \; X_2 \; = \; (m_1 - m_2, \sigma_1^2 + \sigma_2^2)$. A stochastic zero is noted @.0 .

We have the following definitions :

Definition 1 : Let X and Y be two elements of $\mathcal{S}$, X is stochasticaly equal to Y, noted $X \; s= \; Y$, if and only if

$$X \; s- \; Y \quad = \quad @.0 \quad \Leftrightarrow \quad |m_X - m_Y| \; \leq \; \lambda_\beta.\sqrt{\sigma_X^2 + \sigma_Y^2}.$$

Definition 2 : X is stochastically strictly greater to Y, noted $X \; s> \; Y$, if and only if

$$m_X - m_Y \; > \; \lambda_\beta.\sqrt{\sigma_X^2 + \sigma_Y^2}.$$

Definition 3 : X is stochastically greater or equal to Y, noted $X \; s\geq \; Y$, if and only if

$$m_X \; \geq \; m_Y \text{ or } |m_X - m_Y| \; \leq \; \lambda_\beta.\sqrt{\sigma_X^2 + \sigma_Y^2}.$$

The main properties of these new concepts are :

1) $s=$ is reflexive, symmetric but not transitive,

2) $X \; s\neq \; Y \quad \Rightarrow \quad m_X \; \neq \; m_Y$,

3) $s>$ is transitive,

4) $s>$ is the opposite of $s\leq$,

5) $s\geq$ is reflexive, anti-symmetric but, as the equality, not transitive.

As explained in [10], we recover with these definitions, specially the stochastic equality (the others only depend on it), the coherence between arithmetic operations and order relations that was lost in floating point arithmetic.

It must be reminded that stochastic numbers are a model for computed results obtained with the CESTAC method. Therefore, the physical meaning of the above definitions are that two values will be stochastically equal if their difference is only due to round-off error propagation. For the order relation, a value will be strictly greater than another value if it is **significantly** greater than the other. On the other hand, a value will be greater or equal to another value if it is greater than the other or if their difference is only due to round-off error propagation.

In pratice, property 2) is very important. It means that when two computed results are stochastically different, we are sure that the mathematical values they represent are different. It also means that the mathematical zero is always represented by a stochastic zero on computer. This will be very important in the Lanczos type algorithms for the number of iterations as shown in the sixth numerical example of section **5**.

On a computer, by using the synchronous implementation of the CESTAC method and by identifying the notions of computed zero and stochastic zero, it is possible to use the stochastic arithmetic with its definitions.

This is done by the CADNA software [7, 17]. CADNA is a library for programs written in FORTRAN 77 or in ADA which allows the computation using stochastic arithmetic concepts by automatically implementing the synchronous CESTAC method. CADNA may also detect numerical instabilities occuring during the run. The numerical debugging is a very important aspect od CADNA which enables us to stabilize algorithms and to obtain more efficient programs.

The use of CADNA multiplies the running time by a factor between 3 and 5. This allows us to use CADNA on large codes (the greatest code on which CADNA has been used had more than 50.000 FORTRAN lines).

4 The BSMRZS algorithm in stochastic arithmetic

The efficiency of the jumps strategy in the BSMRZS algorithm depends on the choice of the ε's. This is not easy for the user to do because it depends on the data of the problem. Our goal is to replace all of them by tests based on the accuracy of some intermediate results and so to eliminate the use of the ε's (i.e. to have exactly the same code whatever the linear system is). This will be very helpfull when using the jumps strategy.

Let us remark that, as we have already said, on computers there only exist near-breakdowns. Of course, some of them come from mathematical breakdown (in this algorithm, it only means divisions by zero) but it is impossible without a background analysis to distinguish the two cases. Apart from making the computations by hand, we cannot know if the denominator of a division is only badly computed or if it effectively represents the mathematical null value. This is why we say that there are only near-breakdowns on computer.

4.1 Near-breakdowns produced by theoretical breakdowns

Mathematically, there are two possible breakdowns in the BSMRZS algorithm : divisions by zero in the computation of γ_0, η_0 and η_0', division by a null pivot when solving the linear systems (in the case of jump).

Classically, on a computer, testing a theoretical breakdown leads to test the magnitude of the denominator. But using the floating point representation, the size of the exponent has no effect on the numerical stability of a division. A division by a *small* value leads to a very stable result if this value is well computed. Only the mantissa is important. For this reason, the right concept on computers is the accuracy of the denominators with respect to the round-off errors propagation and not the magnitude of the denominator. Even in the gaussian elimination, divisions by a small pivot (if someone can define what a small pivot is) are stable with a well computed pivot. One must not confuse the pivoting strategy, which obviously leads to more stable results, and the detection of the numerical singularity of the matrix. Even on a computer, testing the magnitude of the chosen pivot is not related to this detection.

On a computer, the situation corresponding to a theoretical breakdown is always a division by a stochastic zero because a null value always leads to an unsignificant value using the floating point arithmetic.

Therefore, we must consider divisions by stochastic zero on computer as division by zero in mathematics. That is why, using the stochastic arithmetic, the treatment of this kind of near-breakdowns is very simple. The division is not performed if the denominator is a stochastic zero. In the code, the tests become of the form :

$$IF\ (ALPHA\ .EQ.\ 0.D0)\ THEN\ \ .....$$

followed by the code corresponding to the mathematical treatment of the breakdown. It must be pointed out that this stochastic test allows us to return to the mathematical formulation but we must remind that the test means : if $ALPHA$ has no significant digit then ...

For the gaussian elimination, we have used the version of the algorithm described in [9] which is the elimination with total pivoting(to have a more stable algorithm) but only among the values which are not stochastic zeroes(to detect a numerical singularity). The system is detected as singular if and only if there is no more significant value for the choice of the pivot. After choosing the pivot max, the test for the detection of singular systems is $IF\ (PIVOT\ .EQ.\ 0.D0)\ THEN$... as for the mathematical formulation.

4.2 Near-breakdowns produced by numerical instabilities

The difficulty in the treatment of such near-breakdowns is to detect the basic causes of the numerical instabilities. Numerical instabilities mean very badly computed values and so the appearance of stochastic zeroes. With the CADNA software, it is possible, **during the execution**, to stop the process (and to point out the concerned line of the code) at each time a stochastic zero is detected.

We detected that, without the tests for near-breakdown in the BSMRZS code mentioned in section **2.2.2**, the value η_0 was always a stochastic zero just before the explosion of computations (specially in the fourth example given in the next section). But we also remarked that jumping when η_0 becomes a stochastic zero is not enough. All the linear systems which have to be solved in the sequence are numerically singular. There is always a pivot which is detected as a stochastic zero. Then the process stops after the maximum step for the jump is reached.

This can be explained in the following way.

First, η_0 is one of the parameters which are involved in the recurrence formulae for the computation of the polynomials $P_k(\zeta)$ and $P_k^{(1)}(\zeta)$(the two others are γ_0 and η_0'). If one of these values is very badly computed, the next polynomials will also be. Then we have to test before continuing if all these parameters are well computed.

Secondly, we can remark that if there are numerical instabilities in the recurrence formula and if we decide to jump, they will also appear in the linear system we will have to solve. For instance, the quantity

$$c^{(1)}(\zeta P_k^{(1)^2})c(P_k^{(1)}P_k) - c^{(1)}(P_k^{(1)^2})c(\zeta P_k^{(1)}P_k)$$

(which is nearly the expression of η_0) appears as a subdeterminant of the coefficient matrix and in the right hand side of the system. So, to avoid this situation we have to jump over the polynomials which create instabilities even if they are well computed.

Practically, instead of the tests mentioned in section **2.2.2**, we propose a new strategy of jumping which can be described as follows.

At the beginning of step k, we have computed P_k and $P_k^{(1)}$ and we know that the recurrence formulae of order one (5) and (6) are stable, that means that all the three paramaters γ_0, η_0 and η_0' are not stochastic zeroes. So we can compute the polynomials of degree $n_k + 1$ and take $n_{k+1} = n_k + 1$. We test if for this value of n_{k+1} the order one recurrence relations (5) and (6) are also stable. If not, we begin to jump until the system is regular and the stability conditions are satisfied. We remark that these stability tests include all the cases due to mathematical breakdowns discussed above.

Instead of stochastic zeroes, we may decide to jump when these parameters have only 1 or 2 significant digits. By experience, in the following example using double precision and concerning η_0, a good choice is two significant digits. There is a function which is called CESTAC, in the CADNA software which returns the number of significant digits of every stochastic variable. With this function the test is

$$IF\ (CESTAC(ETA(0))\ .LE.\ 2)\ THEN\ \ ...$$

For an iterative algorithm, we have also to choose a stopping criterion. It is based, in the BSMRZS algorithm, on the vector r_k which is the residue $b - A.x_k$. As explained in [17], the solution will be computationaly available if all the components of r_k are stochastic zeroes. At this time, because of round-off errors propagation, the computer is not able to distinguish the vector r_k from the null vector and continuing has no sense. It is the best result the computer can give. We have chosen this condition for the stopping criteria. As for the divisions and the pivot, the stopping criterion becomes :

$$IF\ (NORM(R(K))\ .EQ.\ 0.D0)\ THEN\ \ ...$$

In the next section we present six examples that were already given in [4]. By applying this strategy to the stochastic version of the BSMRZS algorithm, we have **always** found the results corresponding to the best choice of ϵ_i's.

5 Numerical examples

As all the examples in [4] show, the jump strategy of the BSMRZS algorithms is necessary to prevent from breakowns and near-breakdowns during the run on computers. Using the classical floating point arithmetic, for all the first five following examples, it has been shown in [4] that, at each time, a suitable choice of the ε's gives very good results.

Our goal, in this section, is to show that, using the stochastic arithmetic combined with the new jump strategy, and for the first five examples, we obtain results with the same numerical quality without any use of ε's. Moreover, in the last example, using the stochastic arithmetic allows to eliminate a well-known chaotic behaviour which is only due to the round-off errors propagation.

Computations have been performed on a Sun 4/40 in double precision with the stochastic arithmetic using the CADNA software. The code used is the code presented in the paragraph above.

With the CADNA software, only the exact significant digits of results are printed out, so when it is printed, 0.600 and 0.60000 are different. If the result is a stochastic zero, the symbol @.0 appears. In each example, a table is presented with the number of iterations, the corresponding value of n_k and the residual $Norm\,(r_k)$.

It must be pointed out that if only one jump, in all the following examples, is not performed, the computations explode because of the propagation of the round-off errors and, most of the time, an overflow occurs.

Example 1

The linear system is

$$\begin{pmatrix} 0 & 1 & & & & \\ & 0 & 1 & & & \\ & & \cdot & \cdot & & \\ & & & \cdot & 1 & \\ & & & & 0 & 1 \\ 0.95 & & & & & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ \cdot \\ \cdot \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ \cdot \\ \cdot \\ 1 \\ 0.95 \end{pmatrix}$$

with the dimension of the linear system equal to 40.

The results are :

Iterations	n_k	$Norm\,(r_k)$
1	38	0.400000000
2	40	@.0

As for using floating-point arithmetic, we have a jump from $n_1 = 1$ to $n_2 = 38$ but we have also a jump from 38 to 40 and the solution is reached with about 14 exact significant digits on all the components that is practically the best accuracy of the computer. All the jumps are due to the bad computation of γ_0.

<u>Example 2</u>

The matrix of the system is $\begin{pmatrix} M_1 & & & \\ & M_2 & & \\ & & \cdot & \\ & & & M_{n/2} \end{pmatrix}$ with $M_j = \begin{pmatrix} 1 & j-1+a \\ 0 & -1 \end{pmatrix}$. As explained in [4], the solution is reached at the first iteration if there is a jump of length 2 at the first step. Using floating point arithmetic, it depends on the choice of ε's.

The second member is $b = (5, -3, 4, -4, 0, ..., 0)^T$. Then the solution is given by $x_{2j-1} = b_{2j-1} + (j - 1 + a).b_{2j}$ and $x_{2j} = -b_{2j}$.

With $a = 10^{-4}$, the results are

Iterations	n_k	$Norm\,(r_k)$
1	2	@.0

The vector x_2 is

$x_2(1) = 0.499970000000000E + 001$
$x_2(2) = 0.300000000000000E + 001$
$x_2(3) = -0.40000000000E - 003$
$x_2(4) = 0.400000000000000E + 001$
$x_2(5) = 0.000000000000000E + 000$
....
$x_2(40) = 0.000000000000000E + 000$

The solution is reached with the best accuracy on all the components except the third which have "only" 11 exact significant digits.

Moreover, whatever is the value of a, the good jump is always performed. The jump is due to the bad computation of γ_0.

<u>Example 3</u>

It is the same as above with $M_j = \begin{pmatrix} 0 & 1 + j.a \\ -1 & 0 \end{pmatrix}$, $b = (1, 2, ..., n)^T$ and $a = 10^{-9}$. The dimension of the linear system is 40.

The results are

Iterations	n_k	$Norm\,(r_k)$
1	2	@.0

The vector x_2 is

$x_2(1) = -0.200000000000000E + 001$
$x_2(2) = 0.999999999000000E + 000$
$x_2(3) = -0.400000000000000E + 001$
$x_2(4) = 0.299999999400000E + 001$
$x_2(5) = -0.600000000000000E + 001$
....
$x_2(39) = -0.400000000000000E + 002$
$x_2(40) = 0.389999992200000E + 002$

The solution is reached with the best accuracy on all the components. The jump is due to the bad computation of γ_0.

Example 4

The linear system is

$$\begin{pmatrix} a & 1 & & & & \\ -1 & a & . & & & \\ & -1 & . & . & & \\ & & . & . & 1 & \\ & & & . & a & 1 \\ & & & & -1 & a \end{pmatrix} . \begin{pmatrix} 1 \\ 1 \\ . \\ . \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} a+1 \\ a \\ . \\ . \\ a \\ a-1 \end{pmatrix} .$$

It has been shown that a good solution is obtained if there are jumps of lenght 2 from the beginning until the dimension of the sytem is reached.

For the numerical example, $a = 10^{-8}$ and the dimension is 200.

The results are

Iterations	n_k	$Norm\,(r_k)$
1	2	$0.60000019200002E+001$
2	4	$0.10000001860000E+002$
3	6	$0.14000001800001E+002$
4	8	$0.1800000174000E+002$
...	...	....
98	196	$0.80000001E+001$
99	198	$0.40000000E+001$
100	200	@.0

The vector x_{200} is

$x_{200}(1) = 0.99999999999E+000$
$x_{200}(2) = 0.10000000000E+001$
$x_{200}(3) = 0.10000000000E+001$
$x_{200}(4) = 0.1000000000E+001$
....
$x_{200}(198) = 0.9999999999E+000$
$x_{200}(199) = 0.1000000000E+001$
$x_{200}(200) = 0.1000000000E+001$

The solution is reached at x_{200}. Because of the instability of the system, all the components have only about 10 exact significant digits.

All the jumps are due to the bad computation of η_0 of the next iteration.

Example 5

The linear system is

$$\begin{pmatrix} 2 & 1 & & & & & \\ 0 & 2 & 1 & & & & \\ 1 & 0 & 2 & . & & & \\ & 1 & . & . & . & & \\ & & . & . & . & 1 & \\ & & & 1 & 0 & 2 & 1 \\ & & & & 1 & 0 & 2 \end{pmatrix} . \begin{pmatrix} 1 \\ 1 \\ 1 \\ . \\ . \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 3 \\ 3 \\ 4 \\ . \\ . \\ 4 \\ 3 \end{pmatrix}$$

The dimension is 400.

The results are

Iterations	n_k	$Norm\,(r_k)$
1	2	$0.159700025716005E+004$
2	3	$0.178219984620E+003$
3	4	$0.54406861448E+002$
4	5	$0.3792866771E+004$
5	6	$0.1824035830E+000$
...	...	....
30	31	$0.150E-008$
31	32	$0.1070E-008$
32	33	$0.391E-009$
33	34	@.0

We have only one jump of lenght 2 at the first iteration due to the bad computation of η_0 at the next iteration. All the components of x_{34} have a least 14 exact significant digits.

If the jump is not performed, an overflow occurs. It may also be pointed out that a good solution is obtained very far from the dimension of the system that is, of course, the interest of the Lanczos type algorithms.

<u>Example 6</u>

The linear system is

$$\begin{pmatrix} 0 & & & & -1 \\ 1 & 0 & & & \\ & 1 & . & & \\ & & . & . & \\ & & & 1 & O \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 2 \\ . \\ . \\ N \end{pmatrix} = \begin{pmatrix} -N \\ 1 \\ 2 \\ . \\ N-1 \end{pmatrix}$$

The dimension of the system is $N = 12$.

The results are

Iterations	n_k	$Norm\,(r_k)$
1	1	$0.606328125000000E+002$
2	9	$0.416000000000E+003$
3	10	$0.12028386845E+003$
4	12	@.0

We have a first jump (1 to 9) due to the bad computation of γ_0 and a second jump (10 to 12) due to the bad computation of the next η_0.

The vector x_{12} is

$x_{12}(1) = 0.10000000E+001$
$x_{12}(2) = 0.20000000E+001$
$x_{12}(3) = 0.30000000E+001$
$x_{12}(4) = 0.400000000E+001$
$x_{12}(5) = 0.50000000E+001$
$x_{12}(6) = 0.60000000E+001$
$x_{12}(7) = 0.7000000E+001$
$x_{12}(8) = 0.8000000E+001$
$x_{12}(9) = 0.90000000E+001$

$x_{12}(10) = 0.10000000E + 002$
$x_{12}(11) = 0.1100000000E + 002$
$x_{12}(12) = 0.1200000000E + 002$
The solution is reached with between 8 and 10 exact significant digits on the components.

In this example, using floating point arithmetic, most of the time because of the propagation of round-off errors, the number of iterations is greater than the dimension of the system. This is a well-known behaviour in the Lanczos type algorithms, for instance in the famous conjugate gradient method.

If there is no incurable breakdown, **this never happens using stochastic arithmetic**. As in mathematics, such processes always stop at least when the number of iterations reaches the dimension of the system. At this time, we are sure that the mathematical residual is zero. Then, because the mathematical zero is always represented by a stochastic zero on computer, we are sure that the last computed residual wil be a stochastic zero and so that the process will stop.

6 Conclusion

This paper shows some advantages of the application of the stochastic arithmetic to the BSMRZS algorithm. The use of different ϵ_i in stopping criteria and in testing "small" quantities has been eliminated. The user doesn't need to choose them anymore. We completely let the computer decide by itself for the jumps, the determination of singular systems and the stopping criterion from the accuracy of the intermediate results during the code execution. Moreover, it can also detect some instabilities that are very difficult to express by ϵ_i.

These techniques can be applied to different algorithms involving computations with recurrence relations for orthogonal polynomials and can be useful in detecting and controlling numerical instabilities.

Acknowledgements: we would like to thank C. Brezinski and M. Redivo-Zaglia for their help with the code of BSMRZS.

References

[1] C. Brezinski, *Padé-type Approximation and General Orthogonal Polynomials*, I.S.N.M. vol.50, Birkhauser-Verlag, Basel, 1980.

[2] C. Brezinski, M. Redivo-Zaglia, H. Sadok, Avoiding breakdown and near-breakdown in Lanczos type algorithms, Numerical Algorithms, 1 (1991) 207-221.

[3] C. Brezinski, H. Sadok, Lanczos type methods for solving systems of linear equations, App. Numer. Math., to appear

[4] C.Brezinski, M. Redivo-Zaglia, Treatment of near-breakdown in the CGS algorithm, Num. Algo.,to appear.

[5] J.-M. Chesneaux, Study of the computing accuracy by using probabilistic approach, *Contribution to Computer Arithmetic and Self Validiting Numerical Methods*, ed. C. Ulrich (J.C. Baltzer) 1990, pp. 19-30.

[6] J.-M. Chesneaux, Stochastic arithmetic properties, *Computational and Applied Mathematics, I-Algorithms and Theory*, ed. C. Brezinski (North-Holland) 1992, pp. 81-91.

[7] J.-M. Chesneaux, Descriptif d'utilisation du logiciel CADNA-F, MASI Report, n 92-31, 1992.

[8] J.-M. Chesneaux and J. Vignes, Les fondements de l'arithmétique stochastique, C.R. Acad. Sci., Paris, série I, 315 (1992) 1435-1440.

[9] J.-M. Chesneaux and J. Vignes, L'algorithme de Gauss en arithmétique stochastique, C.R. Acad. Sci., Paris, série II, 316 (1993) 171-176.

[10] J.-M. Chesneaux, The equality relations in scientific computing, Num. Algo. 7 (1994) 129-143.

[11] A. Draux, *Polynomes Orthogonaux Formels. Applications*, LNM 974, Springer Verlag, Berlin, 1983.

[12] C. Lanczos, An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, J. Res. Natl. Bur. Stand., 45 51950) 255–282.

[13] C. Lanczos, Solution of systems of linear equations by minimized iterations, J. Res. Natl. Bur. Stand., 49 (1952) 33–53.

[14] P. Sonneveld, CGS, a fst Lanczos-type solver for nonsymmetric linear systems, SIAM J. Sci. Stat. Comp., 10 (1989) 36–52.

[15] J. Vignes, New methods for evaluating the validity of the results of mathematical computation, Math. Comp. Simul. 20 (1978) 221-249.

[16] J. Vignes, Zéro mathématique et zéro informatique, La vie des sciences, Comptes Rendus, série générale, 4 (1987) 1-13.

[17] J. Vignes, A stochastic arithmetic for reliable scientific computation, Math. Comp. Simul. 35 (1993) 233-261.