

**Can we Utilize the Cancellation  
of the  
Most Significant Digits?**

Victor Pan

TR-92-061

December 1992



# Can we Utilize the Cancellation of the Most Significant Digits?

Victor Pan\*

Mathematics and Computer Science Departments, Lehman College, CUNY  
Bronx, NY 10468

**Summary.** If the sum of several positive and negative numbers has a small magnitude, relative to the magnitudes of the summands, then we show how to decrease the precision of the computation of this sum (without affecting the output precision). Furthermore, if the magnitude of the inner product of two vectors is small and if one of them is filled with “short” binary numbers, each represented with only a few bits, then we decrease the precision of the computation of such an inner product (without affecting the output precision), and we extend this result to the iterative improvement algorithm for a linear system of equations, whose coefficients are represented by “short” binary numbers. We achieve this by truncating both the least and the most significant digits of the operands, according to our new scheme of *backward binary segmentation*.

**Key words:** precision of computations, data compression, matrix computations, linear system of equations, numerical algorithms.

**1991 Mathematics Subject Classification:** 65G10, 65F10, 65G05

---

\* Supported by NSF Grant CCR 9020690 and PSC CUNY Award # 662478.

**1. Introduction.** Cancellation of the most significant digits of the operands of an addition or a subtraction is highly undesirable in numerical computations; the resulting numerical stability problems usually force the user to look for alternate algorithms or to try to control the errors by means of the expensive increase (doubling) of the precision of the computation.

We will show, however, how one may take advantage of such a cancellation, in particular, in the cases of the subtraction and summation of numbers and of the computation of the inner product of two vectors where the output has a relatively small magnitude. We will then show a further application to iterative improvement of an approximate solution to a linear system of equations, whose coefficients are assumed to be "short" binary numbers, each represented with only a few bits.

The idea of our approach is simple: truncate the most significant bits of the operands, as soon as they are not going to influence the output, and enjoy the benefit of decreasing the precision of computations. For this we need, of course, a computer (such as MASPAR), which performs faster if the computation has a lower precision. Due to recent progress in the data compression area, we should expect that more computers of this kind will be used. We also recall recent specific progress in data compression for basic matrix operations ([P,a], [P91], [P92], [BP]), which implies the acceleration of computations by the factor of the order  $u/b$ , showing the ratio of the single precision,  $u$ , and of the actually needed bit-precision of the operands,  $b$ .

Let us next comment on our approach.

To recognize which of the most significant digits of the operands are not needed for obtaining the output and to recover the output after the deletion of these digits of the operands, we apply our simple but, surprisingly, new techniques of *backward binary segmentation* (b.b.s.), which enable us to bound the magnitudes of (one or) both operands of an arithmetic operation if we have an upper bound on the magnitudes of its output (and, respectively, another operand). We demonstrate the power of these techniques in

the cases of computing the difference of two positive numbers, the sum of several positive and negative numbers, the inner product of two vectors (in section 2), provided in all these cases that a known upper bound on the output magnitudes is substantially less than the magnitudes of some input values.

Is this a realistic assumption? Yes, for instance, in the case of many iterative algorithms of linear algebra, where we compute residual vectors of the form  $\mathbf{r}(p) = \mathbf{f} - A\mathbf{x}(p)$  converging to 0 as  $p \rightarrow \infty$ . (Note that here each component of  $\mathbf{r}(p)$  is the inner product of two vectors.) In particular, we elaborate some details of application of our techniques to iterative refinement of the solutions to a large class of linear systems of equations whose coefficients have “short” binary representation. Formally, in  $p$  iterations of iterative refinement, each performed with the precision of the order of at most  $\log p$  bits, we obtain the order of  $p$  correct bits of each output value. Specifically, we show this in the case of an iterative improvement algorithm, adjusted so as to accentuate the power of our techniques and applied in its generalized version (in section 3) and in its classical version (in section 4). In section 5 we give an example of further extensions, by applying the b.b.s. process to Gauss-Seidel iteration. Probably most effective is the application of the b.b.s. techniques to the solution of PDEs by means of multigrid algorithms (see [PR] and [PR,a]).

It would be most interesting for numerical linear algebra to find applications of the b.b.s. techniques which decrease the needed precision of computations from double to single. We leave this as a challenge to the reader.

**2. Binary segmentation and backward binary segmentation. Application to the summation and subtraction of numbers and to the computation of the inner product of two vectors.** Hereafter, all logarithms are to the base 2, and we will rely on the binary representation of numbers. Our study can easily be extended using  $b$ -ary representation for  $b > 2$ .

**Definition 2.1.** Let  $\infty$  stand for  $+\infty$  and expand a real number  $r$  as follows:

$$r = \text{sign}(r) \sum_{s=-\infty}^{\infty} r_s 2^s, \quad r_s = \begin{cases} 0 \\ 1 \end{cases}, \quad s = 0, \pm 1, \pm 2, \dots, \quad (2.1)$$

$r_s = 0$  for  $s = -\infty$  and for  $s > \log |r|$ . Furthermore, for a pair of integers  $g$  and  $h$ , let

$$\begin{aligned} r(g, h) &= \text{sign}(r) \sum_{s=g}^{h-1} r_s 2^s, \\ r^+(g) &= \text{sign}(r) \sum_{s=g}^{\infty} r_s 2^s, \\ r^-(h) &= \text{sign}(r) \sum_{s=-\infty}^{h-1} r_s 2^s. \end{aligned}$$

Thus,  $r(g, h)$ ,  $r^+(g)$ ,  $r^-(h)$  denote the binary expansion of  $r$  chopped on both sides and on each side, respectively. Let  $S(g, h)$ ,  $S^+(g)$  and  $S^-(h)$  denote the respective operators of *binary segmentation*, such that

$$\begin{aligned} S(g, h)r &= r(g, h), \quad S^+(g)r = r^+(g), \quad S^-(h)r = r^-(h), \\ S(g, h) &= S^+(g)S^-(h) = S^-(h)S^+(g). \end{aligned}$$

The operator  $S(g, h)$  chops all the bits representing a binary number  $b$ , except for ones in the fixed range from  $2^g$  to  $2^{h-1}$ .

**Definition 2.2.**  $\{S(g, h)r: \text{all real } r\}$  will be called *the binary segment* and will be denoted  $S[g, h]$ . Equivalently,

$$\begin{aligned} S[g, h] &= \left\{ \pm \sum_{s=g}^{h-1} r_s 2^s, \quad r_s = 0 \text{ or } r_s = 1 \text{ for } s = g, \dots, h-1 \right\}, \\ S[g, h] &= \{r: S(g, h)r = r\}. \end{aligned}$$

Next, we will extend the binary segmentation of a difference, a sum and a product to the binary segmentation of the operands of subtraction, summation and multiplication. We will call such an extension process *backward binary segmentation* (b.b.s.).

We will first apply the b.b.s. process to subtraction, with the predetermined cancellation of certain most significant bits.

**Fact 2.1.** For an integer  $h$  and for two nonnegative numbers  $q$  and  $r$  such that  $|r - q| < 2^h$ , let  $d = d(q, r, h)$  denote  $S^-(h+1)r - S^-(h+1)q$ . Then either

$$|d| < 2^h, \quad r - q = d \quad (2.2)$$

or

$$|d| \geq 2^h, \quad r - q = -2^{h+1} \text{sign}(d) + d. \quad (2.3)$$

**Proof.** Expand  $r$  and  $q$  according to (2.1), so that

$$r = \sum_{s=-\infty}^{\infty} r_s 2^s, \quad q = \sum_{s=-\infty}^{\infty} q_s 2^s, \quad r_s, q_s = \begin{cases} 0 \\ 1 \end{cases} \quad \text{for all } s.$$

Represent  $r - q$  as follows:  $r - q = \sum_{s=-\infty}^{\infty} d_s 2^s$ ,  $d_s = r_s - q_s$ ,  $|d_s| \leq 1$ , for all  $s$ .

Denote  $d_+ = \sum_{s=h}^{\infty} d_s 2^s$ ,  $d_- = \sum_{s=-\infty}^{h-1} d_s 2^s$ , so that  $|d_-| \leq \sum_{s=-\infty}^{h-1} 2^s < 2^h$ ,  $d_+ = r - q - d_-$ ,  $|d_+| \leq |r - q| + |d_-| < 2^{h+1}$ .

Observe that  $(d_+ - d_h 2^h)/2^{h+1} = \sum_{s=0}^{\infty} d_{s+h+1} 2^s$  is an integer, so that the latter inequality,  $|d_+| < 2^{h+1}$ , implies that  $|d_+| \leq 2^h$ . Moreover, either  $d_+ = 0$  or  $|d_+| = 2^h$ , since  $d_+/2^h = \sum_{s=0}^{\infty} d_{s+h} 2^s$  is an integer.

We shall consider two cases. If  $d_s = 0$  for all  $s > h$ , then  $d = S^-(h+1)r - S^-(h+1)q = r - q$ , and both relations of (2.2) hold.

Otherwise,  $|d_+| = 2^h$ ,  $|d_h| = 1$  and  $|d_s| = 1$  for some  $s > h$ . Therefore,  $|d_+ - d_h 2^h| = 2^{h+1}$ , and also  $d_h = \text{sign}(d)$ .

It follows (since  $|d_+| = 2^h$ ) that  $d_+ - d_h 2^h = -d_h 2^{h+1}$ , and since  $r - q = d_+ + d_- = (d_+ - d_h 2^h) + d_- + d_h 2^h = -d_h 2^{h+1} + d = -2^{h+1} \text{sign}(d) + d$ , we arrive at (2.3).  $\square$

**Remark 2.1.** We cannot generally recover  $r - q$  from  $S^-(h)r - S^-(h)q$  under the assumptions of Fact 2.1, as this can be seen from the comparison of the two cases where  $r = 0$ ,  $q = 1$  and  $r = 2$ ,  $q = 1$ , for  $h = 1$ .

Next, we will extend the b.b.s. process to the summation of numbers. Let

$$r = \sum_{i=1}^k p^{(i)}, \quad q = \sum_{i=k+1}^m p^{(i)}, \quad p^{(i)} \geq 0, \quad i = 1, \dots, m, \quad (2.4)$$

$m \geq k \geq 0$ .

If we are given  $p^{(1)}, \dots, p^{(m)}$ , seek  $p = r - q$  and know that

$$|p| < 2^h, \quad (2.5)$$

then Fact 2.1 enables us to reduce the computation essentially to the evaluation of  $S^-(h+1)r - S^-(h+1)q$ . We write that

$$\begin{aligned} d &= S^-(h+1)r - S^-(h+1)q \\ &= S^-(h+1) \sum_{i=1}^k S^-(h+1)p^{(i)} - S^-(h+1) \sum_{i=k+1}^m S^-(h+1)p^{(i)}, \end{aligned}$$

$i = 1, \dots, m$ , so that we may chop all the bits corresponding to  $2^{h+j}$ ,  $j = 1, 2, \dots$ , in the binary representation of  $p^{(i)}$ , for all  $i$ , and of all the partial sums of  $S^-(h+1)p^{(i)}$  that we need to compute at the intermediate stages.

To show the next application, assume (2.4) and (2.5) and let  $p^{(i)} = u^{(i)}v^{(i)}$ ,  $i = 1, \dots, m$ , where  $u^{(i)}, v^{(i)}$  are nonnegative input values for all  $i$ , and moreover, for some integers  $a = a(i)$  and  $b = b(i)$ ,  $a < b$ , the  $v^{(i)}$  lie in the binary segment  $S[a, b]$ , that is,

$$v^{(i)} = \sum_{s=a}^{b-1} v_s^{(i)} 2^s, \quad v_s^{(i)} = \begin{cases} 0 \\ 1 \end{cases} \quad \text{for all } i \text{ and } s \quad (2.6)$$

(see Definition 2.2), so that  $2^a \leq v^{(i)} \leq 2^b - 2^a$ .

Then  $p = r - q$  is actually the inner product of two vectors of dimension  $m$ , one of which has its component values bounded according to (2.6). Then, due to (2.6),

$$S^-(h+1)p^{(i)} = S^-(h+1)((S^-(h+1-a)u^{(i)})v^{(i)}),$$

so that we may also truncate the bits corresponding to  $2^{h-a+j}$ ,  $j = 1, 2, \dots$ , in the binary representation of  $u^{(i)}$  for all  $i$ .

Note that the most significant bits of both operands of an addition or a subtraction can be chopped if the magnitude of the output is bounded from above, whereas for a

multiplication, we need to have the magnitudes of both output and one of the two operands bounded in order to be able to safely chop the most significant bits of another operand.

So far, we have been studying the techniques of the truncation of the most significant bits. We will next follow a traditional pattern and will also chop some of the least significant bits of the same auxiliary, output and input values. Specifically, fix an integer  $g$ , denote that

$$\tilde{u}^{(i)} = S(g - b - 1, h + 1 - a)u^{(i)} \text{ for all } i ,$$

and compute

$$\tilde{p}^{(i)} = S(g, h + 1)(\tilde{u}^{(i)} v^{(i)}) ,$$

$i = 1, \dots, m$ . Then recursively sum  $\tilde{p}^{(i)}$  together, separately for  $i = 1, \dots, k$  and for  $i = k + 1, \dots, m$ , applying the operators  $S^-(h + 1)$  to each computed auxiliary value.

Denote  $\hat{q} = S^-(h + 1) \sum_{i=k+1}^m \tilde{p}_i$ ,  $\hat{r} = S^-(h + 1) \sum_{i=1}^k \tilde{p}^{(i)}$ . If  $m \geq 2k$ , let  $\tilde{r} = \hat{r}$  and compute  $\tilde{q} = \hat{q} + (m - 2k)2^{g-1}$ . Otherwise, set  $\tilde{q} = \hat{q}$  and compute  $\tilde{r} = \hat{r} - (m - 2k)2^{g-1}$ . In both cases, subtract  $\tilde{q}$  from  $\tilde{r}$ , so that  $\tilde{r} - \tilde{q} = \hat{r} - \hat{q} - (m - 2k)2^{g-1}$ .

Simple analysis shows that

$$\begin{aligned} 0 &\leq S^-(h + 1 - a)u^{(i)} - \tilde{u}^{(i)} < 2^{g-b-1} , & i = 1, \dots, m, \\ 0 &\leq S^-(h + 1)p^{(i)} - \tilde{p}^{(i)} < 2^g , & i = 1, \dots, m, \\ 0 &\leq S^-(h + 1)r - \tilde{r} \leq k 2^g , & 0 \leq S^-(h + 1)q - \tilde{q} \leq (m - k)2^g , \end{aligned}$$

and therefore,

$$\begin{aligned} (k - m)2^g &\leq S^-(h + 1)r - S^-(h + 1)q - (\tilde{r} - \tilde{q}) \leq k 2^g , \\ |S^-(h + 1)r - S^-(h + 1)q - (\tilde{r} - \tilde{q})| &\leq m 2^{g-1} . \end{aligned}$$

It follows that

$$|\tilde{p}| = |\tilde{r} - \tilde{q}| < 2^{\tilde{h}} - 2^{g-1} , \quad \tilde{h} = \lceil \log(2^h + m 2^{g-1}) \rceil . \quad (2.7)$$

[Note that  $\tilde{r} - \tilde{q} = S^+(g - 1)(\tilde{r} - \tilde{q})$ , and  $|\tilde{r} - \tilde{q}| < 2^{\tilde{h}}$  implies (2.7).]

Under the assumptions (2.6) and (2.7), we may numerically compute the inner product  $p = \sum_{i=1}^k u^{(i)}v^{(i)} - \sum_{i=k+1}^m u^{(i)}v^{(i)}$  within the error bound  $m2^{g-1}$ , even if we chop some of the most and least significant bits of the operands and outputs of the arithmetic operations involved in this computation, specifically, if we chop:

- a) the most and the least significant bits of the input binary values of  $u^{(i)}$ , for all  $i$ , leaving in only  $h+2-g+b-a$  bits corresponding to  $2^{g-b-1+j}$ ,  $j = 0, 1, \dots, h-g+b-a+1$ ;
- b) the most significant bits of the auxiliary values of  $p^{(i)}$  for all  $i$  and of their partial sums, as well as of the “final” sums  $\tilde{r}$  and  $\tilde{q}$ , as long as these bits correspond to  $2^{h+j}$ ,  $j > 0$ , thus leaving in  $h-g+1$  bits, for every  $p^{(i)}$  and for each of the “final” sums  $S^-(h+1)\tilde{r}$ ,  $S^-(h+1)\tilde{q}$  and the partial sums, and
- c) the most significant bits of  $\tilde{r}$  (if  $m < 2k$ ),  $\tilde{q}$  (otherwise) and  $\tilde{r} - \tilde{q}$ , leaving in the values  $S^-(\tilde{h}+1)\tilde{r}$  (if  $m < 2k$ ),  $S^-(\tilde{h}+1)\tilde{q}$  (otherwise), and  $S^-(\tilde{h}+1)(\tilde{r} - \tilde{q})$  [represented with  $\tilde{h}-g+1$  bits (if  $m$  is even) or  $\tilde{h}-g+2$  bits otherwise].

Note that part c) deals with chopping the operands and the output of the subtraction  $\tilde{r} - \tilde{q}$ .

The next fact summarizes our estimates for the precision of computing the inner product, which includes the summation and the subtraction of numbers as special cases (where  $b = a + 1$ ,  $v^{(i)} = 1$  for all  $i$  and, in the case of subtraction,  $m = 2k = 2$ ).

**Fact 2.2.** *The inner product  $p = r - q = \sum_{i=1}^k u^{(i)}v^{(i)} - \sum_{i=k+1}^m u^{(i)}v^{(i)}$  can be computed within the error bound  $m2^{g-1}$ , for any fixed integers  $g, h, k, m$  such that  $g \leq h$ ,  $0 \leq k \leq m$ , by performing  $m-2$  additions, each with the precision of  $h-g+1$  bits, one addition and one subtraction, with the precision of  $\tilde{h}-g+2$  (for odd  $m$ ) or (otherwise) of  $\tilde{h}-g+1$  bits, and  $m$  multiplications of  $u^{(i)}$  by  $v^{(i)}$ ,  $i = 1, \dots, m$ , with the precision of  $h-g+b-a+2$  bits in the fractions (mantissas) of the binary representation of  $u^{(i)}$  and  $b-a$  bits in the fractions (mantissas) of the binary representation of  $v^{(i)}$ , for all  $i$ , provided that (2.6) and (2.7) hold, and  $u^{(i)} \geq 0$ , for all  $i$ .*

**Remark 2.2.** We may ensure the output error bound  $2^{g-1}$ , rather than  $m2^{g-1}$ , thus

decreasing  $\tilde{h}$  of (2.7) to  $\lceil \log(2^h + 2^{g-1}) \rceil$ , if we chop by  $\lceil \log m \rceil$  fewer of the least significant bits of  $u^{(i)}$ , for each  $i$ . Then (2.7) will actually turn into

$$|\tilde{p}| = |\tilde{r} - \tilde{q}| < 2^h . \quad (2.8)$$

**Remark 2.3.** The round-off errors of the terms of  $p = \sum_{i=1}^k p^{(i)} - \sum_{i=k+1}^m p^{(i)}$  may cancel each other; then the overall error of computing  $p$  may decrease substantially below  $m 2^{g-1}$ , and then again in such a case  $\tilde{h}$  of (2.7) would decrease, and, similarly, if we extend the above b.b.s. process and replace chopping by rounding.

**Remark 2.4.** Without the relations (2.6), we cannot extend the bounds on the precision of  $p^{(i)}$  to the bounds on the precision of  $u^{(i)}$ ,  $i = 1, \dots, m$ , but surely, we still may slightly simplify the multiplications of  $u^{(i)}$  by  $v^{(i)}$  if the precision of  $p^{(i)}$  is bounded.

### 3. Application of b.b.s. to generalized iterative improvement algorithm.

Let us apply our b.b.s. tools in order to bound the precision of computations in the well-known algorithm  $([A], [W])$  for iterative improvement of a solution to a nonsingular linear system of  $n$  equations,

$$Ax = f .$$

Hereafter, we will assume the matrix and vector norm  $\|\cdot\| = \|\cdot\|_\infty$ . For an input vector  $f$ , for a pair of  $n \times n$  matrices  $A$  and  $C$  (the latter approximating  $A^{-1}$ , so that

$$\|I - CA\|_\infty \leq 2^{-b} < 1 , \quad (3.1)$$

for some fixed positive scalar  $b$ ) and for some initial vector  $x(0)$  (say, for  $x(0) = 0$ ), the algorithm successively computes the vectors

$$r(p) = f - Ax(p-1) , \quad (3.2)$$

$$e(p) = Cr(p) , \quad (3.3)$$

$$x(p) = x(p-1) + e(p) , \quad (3.4)$$

for  $p = 1, 2, \dots$ . Then, it can be easily shown that, for  $p = 1, 2, \dots$ ,

$$\mathbf{x} - \mathbf{x}(p) = (I - CA)(\mathbf{x} - \mathbf{x}(p-1)) = (I - CA)^p(\mathbf{x} - \mathbf{x}(0)) ,$$

$$\mathbf{r}(p) = A(\mathbf{x} - \mathbf{x}(p-1)) ,$$

$$\mathbf{e}(p) = CA(\mathbf{x} - \mathbf{x}(p-1)) .$$

Therefore, if (3.1) holds, then  $\mathbf{r}(p)$  and  $\mathbf{e}(p)$  converge to 0 with the speed of a geometric progression, as  $p \rightarrow \infty$ . Furthermore, this analysis can be extended to the case where  $\mathbf{r}(p)$  in (3.3) and  $\mathbf{e}(p)$  in (3.4) are replaced by their numerical approximations,

$$\mathbf{r}^*(p) = \mathbf{r}(p) + \Delta \mathbf{r}(p) , \quad \mathbf{e}^*(p) = \mathbf{e}(p) + \Delta \mathbf{e}(p) , \quad (3.5)$$

respectively, for  $p = 1, 2, \dots$ , where

$$\|\Delta \mathbf{r}(p)\| \leq 2^{g^*(p)} , \quad \|\Delta \mathbf{e}(p)\| \leq 2^{g(p)} , \quad (3.6)$$

$$g^*(p) = g^* - bp , \quad g(p) = g - bp , \quad (3.7)$$

for  $b$  of (3.1),  $p = 1, 2, \dots$  and for some fixed scalars  $g$  and  $g^*$ . It is not hard to show that we may choose these scalars so as to preserve rapid convergence of  $\|\mathbf{x} - \mathbf{x}(p)\|$  to 0 (with the speed of a geometric progression) and that we may fix two scalars  $h$  and  $h^*$  such that

$$\|\mathbf{r}^*(p)\| \leq 2^{h^*(p)} , \quad \|\mathbf{e}^*(p)\| \leq 2^{h(p)} , \quad (3.8)$$

$$h^*(p) = h^* + \log p - bp , \quad h(p) = h + \log p - bp , \quad (3.9)$$

$p = 1, 2, \dots$  (see Appendix B). We will assume that  $b, g, g^*, h$  and  $h^*$  have been precomputed, and we will also assume that all the entries of the input matrix  $A$  lie in a fixed binary segment  $S[g(A), h(A)]$  of a moderately small length  $h(A) - g(A)$ .

**Remark 3.1.** The latter assumption is needed to bound the precision of computing the product of  $A$  by  $\mathbf{x}(p-1)$  in (3.2). This assumption holds, for instance, for many linear systems obtained by discretization of linear PDE's with constant coefficients. Generally, if  $A$  is a well-conditioned matrix, we may decrease  $h(A) - g(A)$  by chopping the entries

of  $A$  and/or by applying the routine technique of algebraic segmentation, described in Appendix A.

The above assumptions enable us to apply the b.b.s. process [based on Fact 2.2, Remark 2.2 and the relations (2.8)] in order to bound the precision of computations performed according to (3.2) and (3.3) with  $\mathbf{r}^*(p)$  replacing  $\mathbf{r}(p)$  in (3.3) and with  $\mathbf{e}(p)$  and  $\mathbf{r}(p)$  evaluated with errors represented by the vectors  $\Delta\mathbf{e}(p)$  and  $\Delta\mathbf{r}(p)$ , which satisfy (3.5)–(3.9).

Let hereafter  $c_i(W)$  denote the number of nonzero entries of row  $i$  of a matrix  $W$ , and for simplicity, assume that  $c_i(A) + 1$  and  $c_i(C)$  are even for all  $i$  [otherwise, we would just need an extra bit of the precision to handle each term, corresponding to  $(m - 2k)2^{g-1}$  with odd  $c_i(A) + 1$  or  $c_i(C)$ , playing the role of  $m$ ]. Then we arrive at the following bounds on the precision of the operands (for  $p = 1, 2, \dots$ ):

(a)  $d_i^*(p) = h^*(p) - g^*(p) + \lceil \log(c_i(A) + 1) \rceil + 1 = h^* - g^* + \log p + \lceil \log(c_i(A) + 1) \rceil + 1$  bits suffice for the representation of each operand of any addition or subtraction involved in the evaluation of the  $i$ -th component of  $\mathbf{r}(p)$  [according to (3.2)];

(b)  $d_i^*(p) + h(A) - g(A) + 1$  bits suffice for the representation of any component of  $\mathbf{x}(p - 1)$  when this component is multiplied by an entry of row  $i$  of  $A$  [according to (3.2)];

(c)  $d_i(p) = h(p) - g(p) + \lceil \log c_i(C) \rceil + 1 = h - g + \log p + \lceil \log c_i(C) \rceil + 1$  bits suffice for the representation of each operand of any addition or subtraction involved in the evaluation of the  $i$ -th component of  $\mathbf{e}^*(p)$  [according to (3.3)];

(d)  $d_i(p) + h^*(p) - g^*(p) + 1 = d_i(p) + h^* - g^* + 1 + \log p$  bits suffice for the representation of any entry of row  $i$  of  $C$  when this entry is multiplied by a component of  $\mathbf{r}^*(p)$  [according to (3.3)].

Thus, the b.b.s. process enables us to compute the solution values with the precision of the order of  $p$  in  $p$  calls to the loop (3.2)–(3.4), even though only the order of  $(\log i)$ -bit precision is needed in the computations of the  $i$ -th call to this loop, for  $i = 1, 2, \dots$

In yet another comparison, if we do not use the b.b.s. process, then we generally must

increase, at least to  $H^*(p) - g^*(p)$  and  $H(p) - g(p)$ , the precision of the computations performed according to (3.2) and (3.3), respectively, where

$$H^*(p) = \log \max\{\|\mathbf{b}\|, \max_{i,j} |a_{ij}x_j(p-1)|\},$$

$$A = [a_{ij}], \quad \mathbf{x}(p-1) = [x_j(p-1)],$$

$$H(p) = \log \max_{i,j} |c_{ij}r_j^*(p)|,$$

$$C = [c_{ij}], \quad \mathbf{r}^*(p) = [r_j^*(p)], \quad i, j = 1, \dots, n; p = 1, 2, \dots,$$

so that  $H^*(p) - h^*(p)$  and  $H(p) - h(p)$  are bounded from below by  $H^* + bp - \log p$  and by  $\log \max_{i,j} |c_{ij}| + H$ , respectively, for two constants  $H^*$  and  $H$ .

**4. An application of b.b.s. to the classical version of the iterative improvement algorithm.** Let us comment on some specifics of the application of the b.b.s. process in the classical case of the iterative improvement algorithm, where

$$C = (PLU)^{-1} = U^{-1}L^{-1}P^{-1},$$

$P$  is a permutation matrix,  $L = I + L^*$ ,  $U = D + U^*$ ,  $L^*$  and  $(U^*)^T$  are proper lower triangular matrices,  $D$  is a nonsingular diagonal matrix,  $|(L)_{ij}| \leq 1$ , for every entry  $(L)_{ij}$  of  $L$ .

In this specific case, the evaluation of  $\mathbf{r}^*(p)$  based on (3.2) does not change but the evaluation of  $\mathbf{e}(p)$  from (3.3) is replaced by the solution of two triangular linear systems of equations:

$$L\mathbf{y}(p) = P^{-1}\mathbf{r}(p), \tag{4.1}$$

$$U\mathbf{e}(p) = \mathbf{y}(p). \tag{4.2}$$

Applications of (3.8) [with  $\mathbf{e}(p)$  replacing  $\mathbf{e}^*(p)$  where  $\|\Delta\mathbf{e}(p)\|$  is small] and of (4.2) implies the bound

$$\|\mathbf{y}(p)\| = \|U\mathbf{e}(p)\| \leq 2^{h(p)}\|U\|,$$

and since we seek  $\mathbf{e}(p)$  within the bound  $2^{g(p)}$  on the error vector norm, we only need to compute the components of  $\mathbf{y}(p)$  within the error bound  $\|U^{-1}\|2^{g(p)}$ .

Let  $\tilde{h}$  and  $\tilde{g}$  be two fixed integers such that

$$\|U\| \leq 2^{\tilde{h}}, \quad \|U^{-1}\| \leq 2^{\tilde{g}}.$$

Then

$$\|y(p)\| \leq 2^{\tilde{h}+h(p)}.$$

We now compute a desired approximation to  $y(p)$  [by solving (4.1)], substitute it into (4.2) and solve for  $e(p)$ . Both linear systems (4.1) and (4.2) are triangular and are solved by means of substitution, which amounts to successive evaluation of the components of the vectors  $L^*y(p)$  and  $D^{-1}U^*e(p)$  and to subtracting them from the respective components of the vectors  $P^{-1}r(p)$  and  $D^{-1}y(p)$ .

Suppose that we apply the b.b.s. process [based on Fact 2.2, Remark 2.2 and the relations (2.8)] to the evaluation of every inner product in this computation, and again, for simplicity, assume that  $c_i(L)$  and  $1 + c_i(U)$  are even numbers. Then we bound the precision of the operands as follows:

a)  $f_i^*(p) = h(p) - g(p) + \tilde{h} - \tilde{g} + \lceil \log c_i(L) \rceil + 1 = h - g + \tilde{h} - \tilde{g} + \log p + \lceil \log c_i(L) \rceil + 1$  bits suffice for the representation of each operand of any addition or subtraction involved in the evaluation of the  $i$ -th component of the vector  $y(p)$  from (4.1), by means of substitution;

b)  $f_i^*(p) + h(p) - g(p) + \tilde{h} - \tilde{g} + 1 = 2(h - g + \tilde{h} - \tilde{g} + 1 + \log p) + \lceil \log c_i(L) \rceil$  bits suffice for the representation of every entry of row  $i$  of  $L$  when this entry is multiplied by a component of  $y(p)$  in the substitution process for solving (4.1);

c)  $f_i(p) = h(p) - g(p) + \lceil \log c_i(U) \rceil + 1 = h - g + \log p + \lceil \log c_i(U) \rceil + 1$  bits suffice for the representation of each operand of any addition or subtraction involved in the evaluation of the  $i$ -th component of the vector  $e(p)$  from (4.2), by means of substitution;

d)  $f_i(p) + h(p) - g(p) + 1 = 2(h - g + 1 + \log p) + \lceil \log(1 + c_i(U)) \rceil$  bits suffice for the representation of every entry of row  $i$  of the matrix  $D^{-1}U^*$  when this entry is multiplied by the components of  $e(p)$  in the substitution process for solving (4.2).

This way, the precision of these computations may substantially decrease against the case of the usual solution [with using no b.b.s. process], except, possibly, for the case of smaller values of  $p$ .

**5. An example of further extensions of the b.b.s. process.** To exemplify various possible extensions, assume that  $A$  denotes a matrix filled with short binary numbers and having 1's on its diagonal,  $A = L^* + I + U^*$ , with  $L^*$ ,  $(U^*)^T$  being proper lower triangular matrices.

Gauss-Seidel's iteration for  $Ax = f$  takes the form

$$x(p+1) = f - L^*x(p+1) - U^*x(p), \quad p = 0, 1, \dots$$

Rewrite this as follows:

$$\Delta x(p+1) = -L^*\Delta x(p+1) - U^*\Delta x(p), \quad p = 0, 1, \dots \quad (5.1)$$

$$x(p) = \sum_{i=0}^p \Delta x(i), \quad p = 0, 1, \dots$$

If the iteration converges, then

$$\|\Delta x(p)\|_{\infty} < 2^{g-bp}, \quad p = 0, 1, \dots,$$

for some constants  $b > 0$  and  $g$ , and an application of the b.b.s. process enables us to decrease the precision of the computations.

Note that in this case we do *not* require that the input coefficients of the linear system be short binary numbers.

The reader may easily check that the analysis and the results are similar for several other well-known iterative techniques, such as Jacobi's, SOR, SSOR.

## 6. Extension to the Solution of Piecewise-Linear PDE's.

The results of the previous sections enable us to apply the b.b.s. techniques in order to decrease the precision required in the computation of the solution of piecewise-linear partial

differential equations (PDE's) by means of multigrid methods (compare [PR], [PRa]). First let us briefly recall the multigrid approach. For a fixed sequence of  $d$ -dimensional grids  $G_0 \subset G_1 \subset \dots \subset G_n = G$ , we assume that, for  $i = 0, 1, \dots, n$ , a solution of a given PDE on  $G_i$  has been approximated by a  $N_i = |G_i|$ -dimensional vector  $\mathbf{u}_i$  that satisfies a linear system of difference equations,

$$D_i \mathbf{u}_i = \mathbf{b}_i, \quad (6.1)$$

generated by means of the discretization of the PDE over the grid  $G_i$ . Let  $u_i(\mathbf{x})$  for  $\mathbf{x} \in G_i$  denote the respective component of the vector  $\mathbf{u}_i$ , define operators  $P_i$  of prolongation of  $u_{i-1}(\mathbf{x})$  from  $G_{i-1}$  to  $G_i$  (such operators usually amount to interpolation by averaging) and let

$$e_i(\mathbf{x}) = u_i(\mathbf{x}) - P_i u_{i-1}(\mathbf{x}), \quad \mathbf{x} \in G_i, \quad (6.2)$$

denote the prolongation errors of these operators for  $i = 1, 2, \dots, n$ . Furthermore, we define the vectors  $P_i \mathbf{u}_{i-1}$  and  $\mathbf{e}_i$  with the components  $P_i u_{i-1}(\mathbf{x})$  and  $e_i(\mathbf{x})$ , respectively, for  $\mathbf{x}$  ranging on  $G_i$ , and then we define the residual vectors

$$D_i \mathbf{e}_i = \mathbf{r}_i, \quad (6.3)$$

or equivalently,

$$\mathbf{r}_i = \mathbf{b}_i - D_i P_i \mathbf{u}_{i-1}, \quad (6.4)$$

$i = 1, \dots, n$ .

Now, we recall the customary loop ( $V$ -cycle) of the multigrid algorithms for solving the system (6.1), for  $i = n$ , by means of successive evaluation of the following values, defined at stage  $i$  ( $i = 1, \dots, n$ ) for all  $\mathbf{x} \in G_i$  [where, say, we fix  $u_0(\mathbf{x}) = 0$  for  $\mathbf{x} \in G_0$ ]:

- a)  $\hat{u}_{i-1}(\mathbf{x})$  (by the prolongation of  $u_{i-1}(\mathbf{x})$  from the grid  $G_{i-1}$ ),
- b)  $r_i(\mathbf{x})$  [from (6.4)],
- c)  $e_i(\mathbf{x})$  [from (6.3)],
- d)  $u_i(\mathbf{x})$  [by using (6.2)].

The vector equations (6.3) and (6.4) define the computational pattern of the iterative improvement of the solution to a linear system (6.1). Furthermore, in the case of a piecewise-linear PDE with constant coefficients, the entries of the matrix  $D_i$  are “short” binary values, each represented with  $O(1)$  bits.

The only difference with the usual application of the iterative improvement scheme is the stage of the solution of the linear system (6.3). In the iterative improvement algorithms, this system is usually solved by direct methods, whereas the multigrid algorithms solve it by means of iterative methods (say, of Gauss-Seidel’s or SSOR type in the symmetric case).

Furthermore, the number of iterations for solving the systems (6.3) arising in the multigrid algorithms is typically bounded from above by a fixed constant, which corresponds to setting  $p = O(1)$  in section 3. Thus, by applying the techniques of section 5 for solving the system (6.3) and the techniques of section 3 for computing  $\mathbf{r}_i$  from the system (6.4), we decrease the precision of these computations to  $O(1)$  bits, without affecting the accuracy of the output.

## References

- [AHU] A. V. Aho, J. E. Hopcroft, J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Mass., 1976.
- [A] K. E. Atkinson, *Introduction to Numerical Algorithms*, Wiley, 1978.
- [BP] D. Bini, V. Pan, *Numerical and Algebraic Computations with Matrices and Polynomials*, Birkhauser, Boston, 1993.
- [D] P. Duhamel, Implementations of "Split-Radis" FFT Algorithms for Complex, Real and Real Symmetric Data, *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. 34, pp. 285-295, 1986.
- [GL] G. H. Golub, C. F. van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, Maryland, 1989.
- [N] H. J. Nusbaumer, Fast Polynomial Transform Algorithms for Digital Convolution, *IEEE Trans. on ASSP*, ASSP-28, 2, pp. 205-215, 1980.
- [P91] V. Pan, Complexity of Algorithms for Linear Systems of Equations, in *Computer Algorithms for Solving Linear Algebraic Equations, The State of the Art*, edited by E. Spedicato, NATO ASI Series, Series F: Computer and System Sciences: vol. 77, pp. 27-56, Springer, 1991.
- [P92] V. Pan., Complexity of Computations with Matrices and Polynomials, *SIAM Review*, 34, 2, pp. 225-262, 1992.
- [P,a] V. Pan, On Binary Segmentation for Matrix and Vector Operations, to appear in *Computer and Math. (with Applications)*.
- [PR] V. Pan, J. Reif, Compact Multigrid, *SIAM J. on Sci. and Statist. Comp.* 13, 1, 119-127, 1992.
- [PR,a] V. Pan, J. Reif, Generalized Compact Multigrid, to appear in *Computers and Math. (with Applications)*.
- [W] J. M. Wilkinson, *Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.
- [Win] S. Winograd, *Arithmetic Complexity of Computations*, SIAM, Philadelphia, 1980.

## Appendix A. Algebraic segmentation.

In this appendix we recall the routine techniques of algebraic segmentation, cited in Remark 3.1. For convenience, assume a floating point representation of the nonzero entries of a matrix (or a vector)  $W$  taking the form

$$w_{i,j} = \pm b^e \sum_{k=0}^{M-1} a_{k,i,j} b^k$$

where  $b$ ,  $e$  and  $M$  denote three fixed integers (the same for all the entries of  $W$ ),  $M \geq 1$ ,  $b > 1$ , and all  $a_{k,i,j}$  are integers varying with the entries,  $0 \leq a_{k,i,j} < b$ ,  $k = 0, \dots, M-1$ .

For a fixed integer  $d > 1$ , let us denote  $m = \lceil M/d \rceil$ ,  $x = b^m$  and rewrite the nonzero entries of  $W$  as polynomials in  $x$ , that is,  $\pm b^e \sum_{g=0}^{d-1} \sum_{h=0}^{m-1} (a_{h+gm,i,j} b^h) x^g$ , whose coefficients have been represented with the  $b$ -base precision  $m$ . The matrix turns into the matrix polynomial of degree  $d-1$ ,

$$W(x) = \sum_{g=0}^{d-1} W_g x^g,$$

where  $W = W(b^m)$  and where the entries of  $W_g$  have form  $\pm b^e \sum_{h=0}^{m-1} a_{h+gm,i,j} b^h$ . An addition or a subtraction of two such matrix polynomials amounts to  $d$  additions or subtractions of matrices whose entries are represented with the ( $b$ -based) precision  $m$ , rather than  $M$ ; a multiplication amounts to  $d^2$  matrix multiplications and  $d^2 - 2d - 1$  matrix additions or subtractions, all performed with a lower precision ranging from  $m$  to  $2m$  (more advanced techniques require  $O(d \log d)$  matrix multiplications and  $O(d \log d \log \log d)$  matrix additions/subtractions, see [AHU], [D], [N], [Win]). In spite of a certain increase of the number of matrix operations, the operations themselves are simplified because of the precision decrease, and this trade-off may be accepted as beneficial in many cases.

## Appendix B. Bounding the error and residual norms.

Let us prove (3.8), (3.9). Replace  $\mathbf{e}(p)$  by  $\mathbf{e}^*(p)$  in (3.4) and  $\mathbf{r}(p)$  by  $\mathbf{r}^*(p)$  in (3.3) and obtain

$$\mathbf{x} - \mathbf{x}(p) = \mathbf{x} - \mathbf{x}(p-1) - \mathbf{e}(p) - \Delta \mathbf{e}(p) = \mathbf{x} - \mathbf{x}(p-1) - C\mathbf{r}(p) - C\Delta \mathbf{r}(p) - \Delta \mathbf{e}(p) .$$

Substitute

$$\mathbf{r}(p) = \mathbf{b} - A\mathbf{x}(p-1) = A(\mathbf{x} - \mathbf{x}(p-1))$$

and obtain that

$$\begin{aligned} \mathbf{x} - \mathbf{x}(p) &= (I - CA)(\mathbf{x} - \mathbf{x}(p-1)) - C\Delta \mathbf{r}(p) - \Delta \mathbf{e}(p) \\ &= (I - CA)^p(\mathbf{x} - \mathbf{x}(0)) - \sum_{i=1}^p (I - CA)^{p-i}(C\Delta \mathbf{r}(i) + \Delta \mathbf{e}(i)) \end{aligned} \quad (B.1)$$

Now we recall (3.6) and (3.7) and deduce from (B.1) that

$$\|\mathbf{x} - \mathbf{x}(p)\| \leq \|I - CA\|^p \|\mathbf{x} - \mathbf{x}(0)\| + \sum_{i=1}^p \|I - CA\|^{p-i} (\|C\|2^{g^*} + 2^g)2^{-bi} .$$

Denote that  $N = \|C\|2^{g^*} + 2^g$ ,  $E_0 = \|\mathbf{x} - \mathbf{x}(0)\|$ , substitute (3.1) and obtain that

$$\|\mathbf{x} - \mathbf{x}(p)\| \leq 2^{-bp}(E_0 + pN) . \quad (B.2)$$

Since

$$\mathbf{r}(p) = \mathbf{f} - A\mathbf{x}(p-1) = A(\mathbf{x} - \mathbf{x}(p)) ,$$

$$\mathbf{e}(p) = C\mathbf{r}(p) + \Delta \mathbf{r}(p) ,$$

we now immediately deduce (3.8) and (3.9) from (B.2) and (3.6).

