C. Ullrich (editor)
J.C. Baltzer AG, Scientific Publishing Co. © IMACS, 1990
pp. 467-482

THE EFFECTS OF THE ARITHMETIC OF VECTOR COMPUTERS ON BASIC NUMERICAL METHODS

D. RATZ
Institut für Angewandte Mathematik, Universität Karlsruhe, Kaiserstr. 12, D-7500 Karlsruhe 1 Germany

Abstract

A collection of examples is presented which demonstrates the effects of the arithmetic of vector computers on basic numerical methods. It is shown that the results of numerical algorithms which make use of the additional arithmetic operations in vector mode differ from the results computed in scalar mode.

1 Introduction

Today, in a continuous effort to increase the processing speed of numerical software, more and more programs are transferred from general purpose computers to vector computers where fast and extremely pipelined arithmetical operations can be used.

The main problem with these additional elementary operations is that they do not comply with any arithmetical standard. Depending on the differences between scalar mode and vector mode computations even for basic operations (shown in [3]) no conclusion can be drawn concerning the accuracy of the computed results, and, with an algorithm being processed in vector mode, the user may lose control of the executed computations. Nevertheless, there are some books about numerics and scientific computing on vector computers which only use the MFLOPS-rates as a test criterion for algorithms (see [8] for example).

The main effort in using vector computers is to run an implemented algorithm as fast as possible. So the programs transferred from general purpose computers to vector computers have to be tuned via vectorization directives to make extensive use of the additional arithmetic vector operations such as *accumulate* or *multiply and accumulate* [1]. But these additional operations do not work in the way the corresponding DO loops would do in scalar mode. So, even elementary numerical methods may produce different results in vector mode and in scalar mode.

In our tests we studied these effects on dot products, extrapolation, polynomial evaluation, systems of linear equations, vector iterations, Newton's method, systems of nonlinear equations etc.

As a matter of principle, source code, input data, and starting values have been exactly the same for the computations in scalar mode and in vector mode (see [3] for details). In our programs we used two kinds of test data:

- specially constructed (sometimes ill-conditioned) vectors and matrices, for example $S_{n,m}$-vectors, $S'_{n,m}$-vectors, Boothroyd/Dekker-matrices, Hilbert-matrices, HRC-matrices
- random data, using random number functions (NAG, CFT77) or generating "wild" input data by evaluating some arithmetic expressions

In the following sections we will have a look on some of the examples we tested on the vector computers

- SIEMENS/Fujitsu VP 400-EX
- CRAY-2
- CONVEX C120.

For details about operating system, hardware architecture, used number screen, etc. see the tables in [3].

In some cases of the tests it was possible to compare the computed results of scalar and vector mode with the exact results which were obvious or with the verified inclusions computed with ACRITH [4]. The latter was only used to check the results of the VP 400 because ACRITH is not available on machines with the number screens used on the CRAY and on the CONVEX.

2 Notations

In the following sections $S = S(b, l, e_{min}, e_{max})$ denotes the used number screen, where b is the base of the floating-point system, l the length of the mantissa, e_{min} the minimum exponent and e_{max} the maximum exponent.
The test vectors mentioned above are defined in the following way:

Definition 1: Let $C \in S$. The vector $s \in S^{n+m+2}$ with

$$s = (\ C\ , \underbrace{1\ ,\ -1\ ,\ \ldots\ ,\ 1\ ,\ -1}_{n \text{ components}},\ -C, \underbrace{1\ ,\ 1\ ,\ \ldots\ ,\ 1}_{m \text{ components}})^T$$

we call $\underline{S_{n,m}\text{-vector}}$.

Definition 2: Let $C \in S$. The vector $s' \in S^{n+m+2}$ with

$$s' = (\ C\ , \underbrace{1\ ,\ 1\ ,\ \ldots\ ,\ 1\ ,\ 1-n}_{n \text{ components}}, -C, \underbrace{1\ ,\ 1\ ,\ \ldots\ ,\ 1}_{m \text{ components}})^T$$

we call $\underline{S'_{n,m}\text{-vector}}$.

It holds:

$$\sum_{i=1}^{n+m+2} s_i = \sum_{i=1}^{n+m+2} s'_i = m$$

3 Scalar Products

A fundamental algorithm in regard to basic numerical methods is the one for the computation of the scalar product (dot product) of two vectors $v, w \in S^n$ by

$$v \cdot w = \sum_{i=1}^{n} v_i \cdot w_i \ .$$

The computation of this formula can be made very fast on a vector computer, but the following examples will show that we have to be very careful in using dot product DO loops in vector mode.

Example 3.1

In the first example we tested the computation of dot products using $S'_{n,m}$-vectors. We computed

$$P_{n,m} := \begin{pmatrix} C \\ 1 \\ \vdots \\ 1 \\ 1-n \\ -C \\ 1 \\ \vdots \\ 1 \end{pmatrix} \cdot \begin{pmatrix} D \\ 1 \\ \vdots \\ 1 \\ 1 \\ D \\ 1 \\ \vdots \\ 1 \end{pmatrix}, \quad \begin{array}{l} \\ \left.\begin{array}{l} \\ \\ \\ \\ \end{array}\right\} n \text{ components} \\ \\ \left.\begin{array}{l} \\ \\ \\ \end{array}\right\} m \text{ components} \end{array}$$

so obviously the exact value of the dot product $P_{n,m}$ is m. In the following tables an examplary part of the results we got on the VP 400-EX, on the CRAY-2, and on the CONVEX C120 is listed.

Results: VP 400-EX ($C = 16^7$, $D = 16^8$)

Product	Scalar Mode	Vector Mode
$P_{0,10}$	10	10
$P_{2,8}$	8	4
$P_{4,6}$	6	0
$P_{6,4}$	4	0
$P_{8,2}$	2	8
$P_{10,0}$	0	4

For greater exponents of C and D the results on the VP 400 are the same as those listed in the table above. On the CRAY-2 we noticed a different behaviour. Depending on the exponents of C and D the vector mode yielded the exact results (for small exponents) or wrong results. For large exponents the CRAY-2 results in vector mode are constantly 0.

Results: CRAY-2 ($C = 16^6$, $D = 16^6$)

Product	Scalar Mode	Vector Mode
$P_{0,10}$	10	13.99...9
$P_{2,8}$	8	11.99...9

Results: CRAY-2 ($C = 16^6$, $D = 16^7$)

Product	Scalar Mode	Vector Mode
$P_{4,6}$	6	0
$P_{6,4}$	4	0
$P_{8,2}$	2	0

The effects on the CONVEX are very similar to those on the CRAY, but there is no exponent range where the results are as strange as those in the first table of the CRAY-2.

Results: CONVEX C120 ($C = 16^7$, $D = 16^7$)

Product	Scalar Mode	Vector Mode
$P_{0,10}$	10	0
$P_{2,8}$	8	0
$P_{8,2}$	–6	–8

It has to be emphasized that the range of the used input data covers only a very small part of the allowed number range. So the errors of the presented vector mode results could not be excused due to large exponent ranges.

We made a special experience with the CRAY-2 concerning the bounds of the DO loops. The results in vector mode computed with fixed DO loop bounds differ from the results computed with variables as DO loop bounds. The reason for this is that the CFT77 compiler [2] on the CRAY-2 recognizes so-called *short loops*, i. e. DO loops which are to be executed at least once and at most 64 times.

In such cases the compiler generates special code. For this reason it is not irrelevant to the computed results what kind of DO loop bounds are used.

Example 3.2

In this second example concerning dot products we generated "wild" data for the used vectors by evaluating some arithmetical expressions. Then we used those two vectors x and y as input data for the scalar mode evaluation as well as for the vector mode evaluation of the dot product $P = x \cdot y$, with x and y defined by

$$x = \begin{pmatrix} 0.140445440000000000\text{E}{+}10 \\ -0.819377966713735131\text{E}{-}05 \\ -0.258123638852471902\text{E}{-}04 \\ 0.351113600000000000\text{E}{+}09 \\ 0.231498901015390023\text{E}{-}01 \\ 0.171760842031043567\text{E}{+}00 \\ 0.200636342857142854\text{E}{+}09 \\ -0.570377732683359966\text{E}{+}02 \\ -0.852656035829167934\text{E}{+}03 \\ 0.140445440000000000\text{E}{+}09 \\ 0.799350588825554769\text{E}{+}04 \\ -0.129982276418151855\text{E}{+}10 \end{pmatrix}, \quad y = \begin{pmatrix} -0.379611791241210937\text{E}{+}09 \\ 0.328135475199999988\text{E}{+}09 \\ -0.164067737600000000\text{E}{+}10 \\ -0.181776074907448264\text{E}{-}02 \\ 0.410169344000000000\text{E}{+}09 \\ -0.820338688000000000\text{E}{+}09 \\ 0.104345990084875078\text{E}{+}02 \\ 0.546892458666666627\text{E}{+}09 \\ -0.546892458666666627\text{E}{+}09 \\ -0.498019822870722965\text{E}{+}05 \\ 0.820338688000000000\text{E}{+}09 \\ -0.410169344000000000\text{E}{+}09 \end{pmatrix}$$

To compare the results computed on the VP 400 in scalar and vector mode with a verified inclusion we used ACRITH on the IBM4381.

Results:

VP 400-EX (Scalar Mode):	0.640000000000000000E+02
VP 400-EX (Vector Mode):	−0.199133434295654297E+02
IBM4381 (ACRITH):	$0.4254811937828514\,^{88}_{34}\,\text{E}{-}03$

The result computed with ACRITH shows that the scalar mode as well as the vector mode evaluation deliver a wrong value for P. Moreover, the two results of the VP 400 are totally different.

Example 3.3

As a last example, we will have a look on dot products $P_i = x^{(i)} \cdot y^{(i)}$, $i = 1, 2$ with

$$x^{(1)} = \begin{pmatrix} 27182818280 \\ -31415926540 \\ 14142135620 \\ 5772156649 \\ 0 \\ 3010299957 \end{pmatrix}, \quad y^{(1)} = \begin{pmatrix} 14862497000000 \\ 8783669879000000 \\ -223749200000 \\ 47737146470000000 \\ 0 \\ 1850490 \end{pmatrix}$$

$$x^{2} = \begin{pmatrix} 5772156649 \\ 3010299957 \\ 0 \\ 27182818280 \\ -31415926540 \\ 14142135620 \end{pmatrix}, \quad y^{2} = \begin{pmatrix} 47737146470000000 \\ 1850490 \\ 0 \\ 14862497000000 \\ 8783669879000000 \\ -223749200000 \end{pmatrix}.$$

We took these dot products from [7] using some modifications:

- an additional vector component with value 0
- all components integers and therefore screen numbers

- interchanged vector components

We computed the dot products on the VP 400-EX and on the CRAY-2 in scalar and vector mode and on the IBM4381 with ACRITH.

Results:

	P_1
VP 400-EX (Scalar Mode):	0.462915718600000000E+10
VP 400-EX (Vector Mode):	0.334189890000000000E+09
CRAY-2 (Scalar Mode):	–0.179496213989999999E+13
CRAY-2 (Vector Mode):	–0.110380659507200000E+13
IBM4381 (ACRITH):	–0.100657107000000000E+10

	P_2
VP 400-EX (Scalar Mode):	–0.115474432000000000E+10
VP 400-EX (Vector Mode):	0.000000000000000000E+00
CRAY-2 (Scalar Mode):	–0.170625964441600000E+13
CRAY-2 (Vector Mode):	–0.110380659507200000E+13
IBM4381 (ACRITH):	–0.100657107000000000E+10

As expected the results of the scalar mode evaluation on the VP 400 and on the CRAY-2 are wrong. Moreover, the results of vector mode evaluation differ from the scalar results. For P_2 the unexpected result computed by the VP 400 even is 0.

Considering the results of the CRAY-2 we notice that the results computed in vector mode do not depend on the special order of the vector components. So the result is the same for all tested orders of the vector components.

4 Extrapolation

In this section we will have a look on a very simple method for linear extrapolation of three given pairs of values (x_1, y_1), (x_2, y_2) and (x_3, y_3) fitting on a line. A best linear approximation for $y(x)$ is $L(x) = mx + b$ where m and b must be computed via the formulas

$$m = \frac{x_1y_1 + x_2y_2 + x_3y_3 - \frac{1}{3}(x_1 + x_2 + x_3)(y_1 + y_2 + y_3)}{x_1^2 + x_2^2 + x_3^2 - \frac{1}{3}(x_1 + x_2 + x_3)^2},$$

and

$$b = \frac{1}{3}(y_1 + y_2 + y_3) - \frac{m}{3}(x_1 + x_2 + x_3).$$

Obviously, the numerator as well as the denominater of the expression m can be evaluated via a dot product. That was the way we did it in our test series on the VP 400 in which we proceeded in three steps:

1. Computation of $L(x)$ using REAL arithmetic

2. Computation of $L(x)$ using DOUBLE PRECISION arithmetic
3. Computation of $L(x)$ using DOUBLE PRECISION arithmetic avoiding the division by 3 in the formula for m.

In step 1 and 2 we computed m via

$$m = \frac{a \cdot b}{a \cdot c}$$

with

$$a = \left(x_1, x_2, x_3, -\frac{x_1}{3}, -\frac{x_1}{3}, -\frac{x_1}{3}, -\frac{x_2}{3}, -\frac{x_2}{3}, -\frac{x_2}{3}, -\frac{x_3}{3}, -\frac{x_3}{3}, -\frac{x_3}{3}\right),$$

$$b = (y_1, y_2, y_3, y_1, y_2, y_3, y_1, y_2, y_3, y_1, y_2, y_3), \quad \text{and}$$

$$c = (x_1, x_2, x_3, x_1, x_2, x_3, x_1, x_2, x_3, x_1, x_2, x_3).$$

The vectors a, b, and c served as input data for the scalar mode as well as for the vector mode. Caused by the divison by three in the vector components of a the dot product has to handle with "full" mantissas. In step 3 we avoided this division and we used the vector

$$a = (3x_1, 3x_2, 3x_3, -x_1, -x_1, -x_1, -x_2, -x_2, -x_2, -x_3, -x_3, -x_3).$$

So, in the case of integer data for the x_i and the y_i, the algorithm has to handle only "short" mantissas.

Example 4.1

As a first example we computed $L(x) = mx + b$ with given

x_i	5201477	5201478	5201479
y_i	99999	100000	100001

, $x = 5201480.$

This example we took from [6].

Results: VP 400-EX

Step / Mode	m	b	$L(x)$
1 / Scalar	0.0221354142	–15136.812	100000.06
1 / Vector	0.0192254297	–0.625	100000.06
2 / Scalar	1.0020718654	–5112254.7625	100002.00414
2 / Vector	0.9981115984	–5091655.5208	100001.99622
3 / Scalar	1.0	–5101478.0	100002.0
3 / Vector	1.0	–5101478.0	100002.0

Astonishingly, the computed values for $L(x)$ in step 1 happened to be the same for the scalar mode and the vector mode although the values of m and b totally differed. In step 2 the results got better, but there was still a difference between scalar mode and vector mode. Only in step 3 the right results could be calculated.

Example 4.2

In this second example we computed $L(x) = mx + b$ with some other given pairs of values

x_i	5789134	5789146	5789158
y_i	56789	113578	170367

, $x = 5789170.$

The results presented below demonstrate the same effects.

Results: VP 400-EX

Step / Mode	m	b	$L(x)$
1 / Scalar	0.00520833209	+83426.25	113578.12
1 / Vector	0.09809404610	–454302.0	113581.06
2 / Scalar	4732.69750284	–27398163239.76	227162.74007
2 / Vector	4732.28829561	–27395794279.36	227152.91909
3 / Scalar	4732.41666666	–27396537438.17	227156.00001
3 / Vector	4732.41666666	–27396537438.17	227156.00001

5 Polynomial Evaluation

Usually the evaluation of a polynomial

$$p(x) = a_n x^n + \cdots + a_1 x + a_0$$

is done via Horner's scheme. This leads to a linear first order recurrence for which a vectorization is only possible with the help of an expansion method like *recursive doubling* or *cyclic reduction*.

The price to pay for the vectorization is that there are additional operations to be executed. Therefore, the operation count for the scalar coding is $O(n)$ while the vectorized coding has an operation count of $O(n \log_2 n)$ using recursive doubling (see [8] for details).

Another very simple and fast method to evaluate a polynomial is to compute $p(x) = a \cdot v_x$, the dot product of

$$a = \begin{pmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_1 \\ a_0 \end{pmatrix} \quad \text{and} \quad v_x = \begin{pmatrix} x^n \\ x^{n-1} \\ \vdots \\ x \\ 1 \end{pmatrix}.$$

In the following example we will compare the results of this method in scalar mode and vector mode with the results of vectorized Horner's scheme and of ACRITH.

Furthermore, we will have a look on a very special effect of the VP 400-EX concerning the difference between the results in scalar mode and vector mode.

Example 5.1

We computed

$$p(x) = \sum_{i=0}^{n} a_i \cdot x^i$$

with the coefficients

$$a_i = \left\{ \begin{array}{ll} -16^7 & \text{for } \; i = 8 \\ 16^4 & \text{for } \; i = 11 \\ 16^{-i} & \text{for } \; i \neq 8, 11 \end{array} \right\} \quad i = 0, 1, \ldots, n$$

by a dot product in scalar mode and vector mode, with (vectorized) Horner's scheme, and with DPEVAL (ACRITH), a routine for the exact evaluation of polynomials.

Results: VP 400-EX ($x_j = 16 + j \cdot 16^{-7}$, $p_j = p(x_j)$)

j	p_j (scalar)	p_j (vector)	p_j (Horner)	p_j (ACRITH)
–6	–4831838152	–4831838140	–4831838162.0	–4831838139.25
–5	–4026531800	–4026531788	–4026531810.5	–4026531789.81
–4	–3221225448	–3221225436	–3221225456.0	–3221225437.00
–3	–2415919096	–2415919084	–2415919098.5	–2415919080.81
–2	–1610612728	–1610612716	–1610612738.0	–1610612721.25
–1	–805306360	–805306348	–805306374.5	–805306358.31
0	8	4	8.0	8.00
1	805306376	805306372	805306377.5	805306377.68
2	1610612744	1610612740	1610612750.0	1610612750.75
3	2415919112	2415919108	2415919125.5	2415919127.18
4	3221225480	3221225476	3221225504.0	3221225507.00

For all evaluation points x_j the values of p_j computed in scalar and vector mode differ from each other as well as from the value computed with Horner's scheme and the exact value delivered by ACRITH. At most cases the results of Horner are worse than the dot product results.

Comparing the listed results of the table above, the attentive reader may notice a very strange behaviour of the difference between scalar mode and vector mode which is constantly -12 for the negative values of p_j and 4 for the negative values.

Using smaller and larger steps for the values of x_j we examined this effect more precisely. In the following picture this "special effect" concerning the difference between scalar mode and vector mode evaluation is demonstrated.

`Ps(x)` and `Pv(x)` denote the value computed in scalar mode and in vector mode, respectively. The graph shows `Ps(x)-Pv(x)` being constant in special areas of `x` and switching from positive values, to negative values, back to positive values, and to zero. So, according to this unexpected and unexplainable behaviour, there was no possibility to predict the vector mode results being greater than or lower than the results of scalar mode.

6 Systems of Linear Equations

In our test series concerning systems of linear equations $Ax = b$ with $x, b \in S^n$ and $A \in S^{n \times n}$ we used several methods.

Method 1: Computation of $R \approx A^{-1}$ with an arbitrary inversion routine and then computation of $x = R \cdot b$.

Method 2: Solving $Ax = b$ using the Gaussian elimination method

Method 3: Solving $Ax = b$ using the Gaussian elimination method with partial pivot selection.

Method 4: Solving $Ax = b$ using the Gaussian elimination method with complete pivot selection.

In method 1 we made no requirements on the accuracy of the used inversion routine. The inverse R was computed only once and then used as input data for the scalar mode and vector mode multiplication with the vector b.

We tested these four methods with Boothroyd/Dekker matrices, Hilbert matrices, HRC matrices (self-designed), HRC$'$ matrices (self-designed), and random matrices.

Example 6.1

Our first example delt with method one, so we computed $x = R \cdot b$ with $R \approx A^{-1}$ delivered by a Gauss-Jordan method where A is a 12×12 HRC$'$ matrix with the following structure

$$\begin{pmatrix}
-\frac{17}{60C} & -\frac{17C+77}{60C} & \frac{77}{120C} & \frac{77C-43}{120C} & \frac{1}{4C} & \frac{C-3}{4C} & \frac{1}{6C} & \frac{C-5}{6C} & \frac{1}{8C} & \frac{C-7}{8C} & \frac{1}{10C} & \frac{C-9}{10C} \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & C+1 & -1 & -(C+1) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & \frac{C+1}{2} & -\frac{1}{4} & -\frac{C+1}{4} & -\frac{1}{4} & -\frac{C+1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{3} & \frac{C+1}{3} & -\frac{1}{6} & -\frac{C+1}{6} & 0 & 0 & -\frac{1}{6} & -\frac{C+1}{6} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
\frac{1}{4} & \frac{C+1}{4} & -\frac{1}{8} & -\frac{C+1}{8} & 0 & 0 & 0 & 0 & -\frac{1}{8} & -\frac{C+1}{8} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
\frac{1}{5} & \frac{C+1}{5} & -\frac{1}{10} & -\frac{C+1}{10} & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{10} & -\frac{C+1}{10} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

and $b \in S^{12}$ with $b_i = 1,\ i = 1, \ldots, 12$.

Results: VP 400-EX ($C = 16^{15}$)

x (Scalar Mode)	x (Vector Mode)
0.268436029600708008E+09	0.268436029600706100E+09
0.999999999767169204E+00	0.999999999767169329E+00
-0.656728087816238403E+05	-0.656588087987899780E+05
0.999999999999999958E+00	0.999999999999999889E+00
-0.107935622215270996E+03	-0.800000000000000000E+02
0.100000000000000000E+01	0.100000000000000000E+01
0.225999990463256836E+03	0.256000000000000000E+03
0.100000000000000000E+01	0.100000000000000000E+01
-0.126999999046325684E+03	-0.962659921646118164E+02
0.100000000000000000E+01	0.100000000000000000E+01
0.256000000000000000E+03	0.581210824966430644E+03
0.100000000000000000E+01	0.100000000000000000E+01

With the exception of the vector components with the value 1 all components of x differ. In the worst cases not even one digit of the corresponding components is the same in scalar mode and vector mode.

Example 6.2

Using the three different Gaussian elimination methods (methods 2 - 4) we computed the solution x of $Ax = b$ where $b \in S^n$ with $b_i = 1,\ i = 1, \ldots, n$ and A is an $n \times n$ Boothroyd/Dekker matrix, random matrix, Hilbert matrix or a HRC matrix. The latter we defined by

$$A = \begin{pmatrix}
\frac{1}{2C} & -\frac{C-1}{2C} & 0 & -\frac{1}{C} & 0 & -\frac{1}{C} & 0 & -\frac{1}{C} & 0 & -\frac{1}{C} & \frac{1}{2C} & \frac{C-1}{2C} \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & \frac{C+1}{2} & -\frac{1}{2} & -\frac{C+1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2} & \frac{C+1}{2} & -\frac{1}{2} & -\frac{C+1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{2} & \frac{C+1}{2} & -\frac{1}{2} & -\frac{C+1}{2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{C+1}{2} & -\frac{1}{2} & -\frac{C+1}{2} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{C+1}{2} & -\frac{1}{2} & -\frac{C+1}{2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

for $n = 12$.

After solving the systems of linear equations we compared the solutions delivered by scalar mode and by vector mode. In the following table we present the results for method 3 only, due to the fact that the results of the other methods are very similar. For each tested matrix the maximum number of differing hexadecimal digits is listed. The index of the HRC matrices represents the exponent P of $C = 16^P$ used to build up the matrix.

Results: (With Partial Pivot Selection)

Matrix	VP 400-EX	CRAY-2	CONVEX C120
HRC_{15}	all	-	-
HRC_{14}	0	0	-
HRC_{13}	-	all	-
HRC_{12}	-	0	-
B/D	7	4	0
Hilbert	5	7	-
Random	4	3	-

A user who is familiar with numerical problems in solving systems of linear equations would probably expect bad results, but he would never expect such large discrepancies between scalar mode and vector mode. Above all, the ill-conditioned problems deliver results with large differences. For the HRC-matrices there are values of C which cause the results to differ in all digits. Surprisingly the results for random data also differ, although random data is to be known as well-conditioned.

7 Vector Iteration

Another well-known method for solving a system of linear equations

$$Ax = b$$

is the iterative method

$$x^{k+1} = Dx^k + b$$

with

$$D = I - A.$$

Comparing the scalar mode iteration with vector mode iteration we wanted to get information about the convergence of the algorithms. Our special intention was to see whether the effects described in the last sections could be neglected during an iteration or not.

We chose two kinds of input data.

- $A \in S^{12\times 12}$ with

$$A = \begin{pmatrix} \frac{15}{16} & & & & \\ & \frac{15}{16} & & & -\frac{1}{16} \\ & & \frac{15}{16} & & \\ -\frac{1}{16} & & & \ddots & \\ & & & & \frac{15}{16} \end{pmatrix},$$

  so $D = (d_{ij})$ with $d_{ij} = \frac{1}{16}$ for $i, j = 1, 2, \ldots, 12$. The vector b was an $S'_{n,m}$-vector

- A and b random matrix and random vector, respectivly.

Example 7.1

We tested the iteration $x^{k+1} = Dx^k + b$ where $x^k, b \in S^{12}$, $D \in S^{12\times 12}$ with $d_{ij} = \frac{1}{16}$, for $i, j = 1, \ldots, 12$, and b an $S'_{n,m}$-vector with the starting vector $x^0 = b$.

Considering the results we noticed at first, that the scalar mode iteration and the vector mode iteration stopped after different numbers of iteration steps. Furthermore, the two resulting vectors were different with the exception of two components.

Results: VP 400-EX ($C = 16^{15}$)

$b = S'_{2,8}$

Scalar Mode (54 iterations)	Vector Mode (28 iterations)
0.115292150460684698E+19	0.115292150460684698E+19
0.199999999999999978E+01	0.133333333333333326E+01
−0.111022302462515654E−15	−0.666666666666666685E+00
−0.115292150460684698E+19	−0.115292150460684698E+19
0.199999999999999978E+01	0.133333333333333326E+01
0.199999999999999978E+01	0.133333333333333326E+01
0.199999999999999978E+01	0.133333333333333326E+01
0.199999999999999978E+01	0.133333333333333326E+01
0.199999999999999978E+01	0.133333333333333326E+01
0.199999999999999978E+01	0.133333333333333326E+01
0.199999999999999978E+01	0.133333333333333326E+01
0.199999999999999978E+01	0.133333333333333326E+01

Results: CRAY-2 ($C = 16^{12}$)

$b = S'_{8,2}$

Scalar Mode (26 iterations)	Vector Mode (4 iterations)
0.281474976710655999E+15	0.281474976710655999E+15
0.171428571428571530E+01	0.137500000000000000E+01
0.171428571428571530E+01	0.137500000000000000E+01
0.171428571428571530E+01	0.137500000000000000E+01
0.171428571428571530E+01	0.137500000000000000E+01
0.171428571428571530E+01	0.137500000000000000E+01
0.171428571428571530E+01	0.137500000000000000E+01
0.171428571428571530E+01	0.137500000000000000E+01
−0.628571428571427759E+01	−0.662500000000000000E+01
−0.281474976710655999E+15	−0.281474976710655999E+15
0.171428571428571530E+01	0.137500000000000000E+01
0.171428571428571530E+01	0.137500000000000000E+01

In addition to the tests with $S'_{n,m}$-vectors we wanted to see what would happen with the iteration using random data.

Example 7.2

We tested the iteration $x^{k+1} = Dx^k + b$ where $x^k, b \in S^{12},\ D \in S^{12\times 12}$ with random numbers d_{ij} and b_i ($-\frac{1}{16} \leq d_{ij} \leq \frac{1}{16},\ -1 \leq b_i \leq 1,\ \ i,j = 1,\ldots,12$).

For these tests we chose six different sets of random numbers, the starting vector $x^0 = b$, and the stopping criterion $x^{k+1} = x^k$.

Results: VP 400-EX

Set	Scalar Mode	Vector Mode	Difference
1	stop after 24 iterations	stop after 24 iterations	1-2 hex. digits
2	stop after 24 iterations	alternating	
3	stop after 22 iterations	stop after 22 iterations	1-2 hex. digits
4	stop after 22 iterations	stop after 22 iterations	1-2 hex. digits
5	stop after 22 iterations	stop after 22 iterations	1-2 hex. digits
6	stop after 22 iterations	stop after 21 iterations	2-3 hex. digits

Results: CRAY-2

Set	Scalar Mode	Vector Mode	Difference
1	stop after 16 iterations	stop after 17 iterations	1 hex. digits
2	stop after 18 iterations	stop after 18 iterations	0 hex. digits
3	stop after 17 iterations	stop after 17 iterations	0 hex. digits
4	stop after 18 iterations	stop after 20 iterations	0 hex. digits
5	alternating	alternating	
6	stop after 19 iterations	alternating	

Although using only random data the resulting vectors of scalar mode and vector mode iteration differed in about 2-3 hexadecimal digits on the VP 400. Even worse is the fact that there were special cases in which the scalar mode iteration converged but the vector mode iteration, however, alternated.

8 Concluding Remarks

For many users of vector computers the presented results have been very surprising because they have never thought of such fatal errors appearing in very simple numerical algorithms. Therefore, we hope that some of those users will lose their lack of concern in using vectorizing compilers.

At the moment we should be very careful in transferring numerical programs to vector processors. New programs should be tested in vector mode *and* in scalar mode with the question of numerical correctness being at least as important as the MFLOPS-rate.

Moreover, the user of vector computers should get more information about the macros used for vectorizing DO loops. With this information he might prevent some of the resulting errors by modifying the implementation of an algorithm. A better way to prevent these errors would be a modification of the summation macros of vector computers by the manufacturers themselves in such a way that there is no difference between the result of scalar mode and vector mode.

For the future, vector computer manufacturers should comply with the requirements of the "IMACS-GAMM Resolution on Computer Arithmetic" [5]. These prerequisites will serve as a basis for the implementation of numerical methods with automatic result verification on vector computers (see [9], [10]). The necessity of the computer doing the error control computations by itself should be obviously for every user of vector computers thinking of future peak peformances of more than 100 GFLOPS.

References

[1] Buchholz, W.: *The IBM System/370 Vector Architecture.* IBM Systems Journal 25/1, 1986.

[2] CRAY Research, Inc.: CFT77 Reference Manual, SR-0018 C, 1988.

[3] Hammer, R.: *How Reliable is the Arithmetic of Vector computers?* This Volume.

[4] IBM *High-Accuracy Arithmetic Subroutine Library* (ACRITH). Program Description and User's Guide, SC 33-6164-02, 3rd Edition, 1986.

[5] IMACS-GAMM *Resolution on Computer Arithmetic.* Preface of this Volume.

[6] Kulisch, U. and Miranker, W. L.: *The Arithmetic of the Digital Computer: A New Approach.* SIAM Review, Vol. 28, No. 1, 1986.

[7] Rump, S. M.: *How Reliable are Results of Computers.* In: Jahrbuch Überblicke Mathematik, Bibliographisches Institut, Mannheim, 1983.

[8] Schönauer, W.: *Scientific Computing on Vector Computers.* North-Holland, Amsterdam, 1987.

[9] Schumacher, G.: *Verified Computations on Supercomputers - Error Control in Matrix Problems.* This Volume.

[10] Schumacher, G.: *Einschließung der Lösung von linearen Gleichungssystemen auf Vektorrechnern.* In: Kulisch, U. (Ed.): *Wissenschaftliches Rechnen mit Ergebnisverifikation - Eine Einführung.* Akademie Verlag, Ost-Berlin, Vieweg, Wiesbaden, 1989.