On the Accuracy of the Solution of Linear Problems on the CELL Processor*

René Alt and Jean-Luc Lamotte
CNRS, UMR 7606, LIP6
University Pierre et Marie Curie
4 place Jussieu, 75252 Paris cedex 05, France
Rene.Alt@lip6.fr, Jean-Luc.Lamotte@lip6.fr

Svetoslav Markov†
Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, "G. Bonchev" st., bl. 8, 1113 Sofia, Bulgaria
smarkov@bio.bas.bg

Abstract

Several super computers have been designed as massively parallel computers using the CELL processor as their main component. Such is for example the IBM Roadrunner which broke the world computing speed record in June 2008. However, even if machines of this kind are absolutely necessary to solve numerical problems that could not be solved otherwise, the question of the accuracy of the solution may become critical when obtained with a monstrous amount of computation. Concerning the question of accuracy, the arithmetic of the eight on chip parallel processors of CELL have two drawbacks: i) rounding is towards zero and not to nearest, ii) division is very inaccurate. The paper deals with the effect of these two particularities on the result of scientific computations. First, it is shown that the classical computation of the inner product of two n-dimensional vectors has an accuracy which is $O(\sqrt{n})$ for rounding to nearest and $O(n)$ for all other rounding modes. Thus the fast rounding to zero mode of the CELL arithmetic is certainly not the best concerning the accuracy of results when solving linear problems. Second, it is shown that in algorithms using divisions, it is necessary to be careful in programming as standard low level functions do not include division but only (multiplicative) inverse with a low precision. The consequence is that solving large linear systems on super computers using the CELL with unsuitable methods may be prone to significant errors and therefore the results must be carefully controlled. Numerical examples are given.

*Submitted: January 10, 2009; Revised: February 8, 2010; Accepted: February 20, 2010.
†S. Markov was partially supported by the Bulgarian NSF Project DO 02-359/2008.

Keywords: CELL processor, linear algebra, inner product, round-off error, CESTAC method, stochastic arithmetic, imprecise data, accuracy of numerical results
AMS subject classifications: 65G20, 65G40, 65G50, 65Fxx

1 Introduction

The CELL parallel processor has been jointly designed by IBM, Sony and Toshiba for the PLAYSTATION 3 with the purpose of very fast processing in game and video. Thus it has been mainly designed for high speed rather than accurate computing. However, the CELL is also used to build super computers such as the IBM Roadrunner. The latter is made up of a blend of 12,960 modified CELL processors and 6948 AMD Opterons and has broken the computing speed record in June 2008 with more than one petaflops. The heart of a CELL is a main processor and eight high speed parallel synergistic processing engines (SPE), each containing a synergistic processing unit (SPU) and communication and hardware connections on a single chip. As speed is the main goal, the floating point arithmetic of the eight parallel processors works with rounding to zero arithmetic and does not posses the four rounding modes of the IEEE 754 standard in single precision arithmetic. Only the main processor has a floating point arithmetic according to this standard. The consequence of this choice is that computing with the eight parallel SPU of the CELL may be much less accurate than with a processor with a rounding to nearest arithmetic, particularly when dealing with linear algebra.

In this paper we are interested in the accuracy of the CELL when used with a programming leading to maximum speed, i.e., using 32 bit single precision and low level functions. It has been shown in [1], [2] that the computational complexity and thus the computation time of an inner product of n components is proportional to n with a rounding to zero arithmetic and to $\sqrt{n}$ with a rounding to nearest arithmetic. Imprecise data with a known Gaussian distribution $N(\mu,\ \sigma)$ can be modelled by so-called stochastic numbers; computation on them uses stochastic arithmetic which is merely defined as operators on Gaussian distributions [4], [11]. The error on the sum of n stochastic numbers is proportional to $\sqrt{n}$. Stochastic arithmetic is a model for the random rounding arithmetic implemented in the CESTAC method and the CADNA Software [14], [15].

The CELL is a hybrid processor with nine cores and two architecture types. It is designed with a PPU processor (A powerPC based processor) and eight SPU processors. On the PPU the division respects the IEEE norm and the performance is equivalent to other processors. The eight SPU provide all the computation power of the CELL processor. They are based on a FMA operator that is executed in six cycles. For division, only a (multiplicative) inverse function is implemented in hardware (six cycles). It only provides twelve exact bits on the result and can be fully pipelined in single precision. A software implementation of an exact division in the 754 IEEE sense is usable but the performances are weaker in comparison with the inverse function.

The paper is devoted to the effect of rounding to zero arithmetic and imprecise division of the CELL processor on computation of inner products and solution of linear systems. The case of imprecise data in inner products is also considered. Numerical experiments are given to compare results provided by rounding to zero arithmetic of the CELL processor and rounding to nearest of a processor with the IEEE 754 norm. Conclusion is, when computing with the CELL processor, one must be careful concerning the accuracy of the results.

2 Round-off error on a standard inner product

2.1 Statistical estimation of the round-off error

The theory exposed here is a generalization of the one developed in [10] for the estimation of the residuals of linear systems. The main idea is that since the rounding is one-sided it has a bias, so when having n roundings, the errors add up and the average result of round-off error is proportional to n. Usually, with rounding to nearest, the mean is 0, so only variance counts, and for n roundings, the variance grows as n, which leads to $\sqrt{n}$ error.

Let us consider a real inner product of n components

$$p = \sum_{i=1}^{n} x_i \ y_i, \qquad x_i, \ y_i, \ p \in \mathbb{R}. \tag{1}$$

Suppose now that the inner product (1) is computed on a computer with a classical accumulation algorithm such as:

$$\begin{array}{l} P = 0. \\ For \ i = 1 \ to \ n \ do \ P = P \oplus X(i) * Y(i). \end{array} \tag{2}$$

In algorithm (2) the computer data $X(i)$ and $Y(i)$ are provided from the exact data x_i and y_i by the assignment operator; $\oplus$ and $*$ are the addition and multiplication of the computer. The final result P is the computed approximation of the exact result p. Let $\mathbb{F}$ be the set of floating point numbers in use of the computer in consideration, then $X(i), \ Y(i), \ P \in \mathbb{F}$. Our aim is now to evaluate the error $\rho = P - p$.

Let $\mathbb{E}$ be the population of all relative errors of the assignment operator of the computer. We shall see in the following that the mean value $\overline{\alpha}$ and standard deviation σ of $\mathbb{E}$ can be easily computed under some simple assumptions. $X(i)$ and $Y(i)$ being the floating point approximations of x_i and y_i, then:

$$\begin{array}{ll} X(i) = x_i(1 + \lambda_i), & i = 1, ..., n, \\ Y(i) = y_i(1 + \mu_i), & i = 1, ..., n, \end{array} \tag{3}$$

with $\lambda_i, \mu_i \in \mathbb{E}$. Let $\oplus$ and $*$ be the (signed) floating point addition, resp. floating point multiplication of the computer; $\oplus$ and $*$ are accomplished by a rounding when the results are transferred from the register to the memory, thus:

$$\begin{array}{ll} Z(i) = X(i) * Y(i) = (x_i(1 + \lambda_i)y_i(1 + \mu_i))(1 + \beta_i), & i = 1, ..., n, \\ P(i) \oplus Z(i) = (P(i) + Z(i))(1 + \alpha_i), & i = 1, ..., n, \end{array} \tag{4}$$

with $\lambda_i, \ \mu_i, \ \alpha_i, \ \beta_i \ \in \mathbb{E}$. In (4) $P(i)$ is the partial sum up to index i, therefore $P = P(n)$; $\lambda_i, \ \mu_i$ denote the relative round-off errors caused by the assignment operator on x_i and y_i and $\alpha_i, \ \beta_i$ are respectively the relative round-off errors on the results of addition and multiplication. All computer operations are performed in registers and followed by a transfer into memory, so the operations can be considered as followed by an assignment and thus $\lambda_i, \ \mu_i, \ \alpha_i, \ \beta_i$ have same mean values and same standard deviations. Keeping only the first order terms we have:

$$X(i) * Y(i) = x_i y_i(1 + \lambda_i + \ \mu_i \ + \beta_i), \quad i = 1, ..., n, \tag{5}$$

with $\lambda_i, \ \mu_i, \ \alpha_i, \ \beta_i \ \in \mathbb{E}$. Introducing equalities (4) and (5) in algorithm (2) we obtain:

$$\begin{aligned} P = \quad & \sum_{i=1}^{n} x_i \ y_i + \sum_{i=1}^{n} x_i \ y_i(\lambda_i + \mu_i + \beta_i) \\ & + \alpha_1(x_1y_1 + x_2y_2) \\ & + \alpha_2(x_1y_1 + x_2y_2 + x_3y_3) + \ ... \\ & + \alpha_{n-1}(x_1y_1 + x_2y_2 + ... + x_n \ y_n). \end{aligned} \tag{6}$$

Thus the error $\rho = P - p$ can be written as:

$$\rho = \sum_{i=1}^{n} x_i \ y_i(\lambda_i + \mu_i + \beta_i) + \textstyle\sum_{k=2}^{n} \alpha_{k-1} \ r_k, \ \ r_k = \ \sum_{j=1}^{k} x_j \ y_j. \tag{7}$$

Let us call $\overline{\alpha}$ and σ the mean-value and the standard deviation of the assignment operator and suppose that all relative errors in formula (7) are independent. Then we have for their mean-values: $\bar{\alpha}_i = \bar{\lambda}_i = \bar{\beta}_i = \bar{\mu}_i = \overline{\alpha}$, for their squared mean-values: $\overline{\alpha_i^2} = \overline{\lambda_i^2} = \overline{\beta_i^2} = \overline{\mu_i^2} = \overline{\alpha}^2 + \sigma^2$, and for the mean-values of the terms of the form $\alpha_i \ \lambda_i$: $\overline{\alpha_i \ \lambda_i} = \overline{\alpha}^2$.

Under the hypothesis of independence of errors it is now possible to estimate the mean value of ρ and ρ^2 by replacing each error and each preceding term in the expression of ρ and ρ^2 by their mean-value. A simple computation leads to:

$$\bar{\rho} = \ \overline{\alpha} \ \left(3p + \sum_{k=2}^{n} r_k \right) \tag{8}$$

$$\overline{\rho^2} = \overline{\alpha}^2 \ (9p^2 + 3pu + u^2) \ + \sigma^2(3s^2 + v^2) \tag{9}$$

with $s^2 = \sum_{i=1}^{n}(x_iy_i)^2$, $u = \sum_{k=2}^{n} \ r_k$ and $v^2 = \sum_{k=2}^{n} \ r_k$; p is the exact real result of the inner product defined in (1).

Formulas (8) and (9) are not symmetric because they take into account the order of computations in algorithm (2). They can be easily simplified by considering that all inner products obtained with all possible permutations of terms in the expression of p are computed and that the mean-values of ρ and ρ^2 are now extended to all these results. A rather simple calculation leads then to the following formulas:

$$\overline{\rho} = \overline{\alpha} \ p \ (n^2 + 7n - 2)/(2n), \tag{10}$$

$$\begin{aligned} \overline{\rho^2} = \quad & \overline{\alpha^2} \ \left(\frac{3n^3 + 41n^2 + 134n - 72}{12n} p^2 + \ \frac{(n-2)(n^2 + 3n - 6)}{12n} s^2 \right) \\ & + \sigma^2 \ \left(\frac{n+1}{3} p^2 + \frac{n^2 + 19n - 6}{6n} s^2 \right). \end{aligned} \tag{11}$$

We shall estimate next the mean value and the standard deviation of the assignment operator.

2.2 Mean-value and standard deviation of the assignment operator

2.2.1 Error caused by the assignment operator

Let $x \in \mathbb{R}$ be a real number and $X \in \mathbb{F}$ its floating point approximation provided by the assignment operator. Then we can write using the IEEE 754 radix 2 representation of numbers:

$$X = \ \pm M \ 2^E, \quad 1/2 \ \leq M \ \leq 1 - 2^{-t}, \tag{12}$$

$$x = \pm m\ 2^E, \quad 1/2 \le m < 1, \tag{13}$$

where t is the number of bits of the mantissa in the floating point numbers, M is the limited mantissa of X, m is the mantissa of x and may be unlimited. The exponent E is supposed here to be identical in both expressions (12), (13). The relative assignment error is then $\alpha = (X - x)/x$.

Setting $r = m - M$, then $\alpha = -r/m$. The bounds for r depend on the rounding mode. The mean value $\overline{\alpha}$ and variance σ^2 for α are resp.: $\overline{\alpha} = \int\int_D q(r,m)\alpha(r,m)drdm$ and $\sigma^2 = \int\int_D q(r,m)(\alpha(r,m) - \overline{\alpha})^2 drdm$, where D is the domain of variation of r and m and $q(r,m)$ is the density of probability of $\alpha(r,m)$ in this domain. The domain of variation of m is defined in formula (13). The domain of variation of r depends on the rounding mode, i. e. $[-1/2^{t+1},\ 1/2^{t+1}[$ for rounding to nearest and $[0, 1/2^t[$ for rounding to zero.

2.2.2 Hypotheses on the distribution of mantissas

The density of probability $q(r,m)$ requires some hypotheses about the distribution of the error r and of the mantissa m. Here and in the following it is supposed that r is uniformly distributed in its definition domain. This hypothesis on the repartition of lost digits is the simplest and does not appear determinant in what follows in the sense that different but reasonable hypotheses produce very close numerical values for $\overline{\alpha}$ and σ.

Concerning the distribution of mantissas there are two classical hypotheses:

Hypothesis 1 [9]: The mantissas are equally distributed in their definition domain. With this hypotheses the values of $\overline{\alpha}$ and σ^2 can be found [10]:

$$\begin{array}{lll} \overline{\alpha} = -\ln(2)\ 2^{-t}, & \sigma^2 = (2/3 - (\ln(2))^2\ 2^{-2t} & \text{for rounding to zero;} \\ \overline{\alpha} = 0, & \sigma^2 = 2^{-2t}/6 & \text{for rounding to nearest.} \end{array} \tag{14}$$

Hypothesis 2 [8] and [13]: The distribution of the mantissas tends towards the density function $q(m) = 1/(m\ \log b)$, $0 \le m < 1$, b being the base of the number system, here $b = 2$. In this case a simple calculation leads to:

$$\begin{array}{lll} \overline{\alpha} = -\dfrac{1}{2\ln(2)}\ 2^{-t}, & \sigma^2 = \left(\dfrac{2}{3\ln(2)} - \dfrac{1}{(2\ln(2))^2}\right)\ 2^{-2t} & \text{for rounding to zero;} \\ \overline{\alpha} = 0, & \sigma^2 = \dfrac{1}{8\ln(2)} 2^{-2t} & \text{for rounding to nearest.} \end{array} \tag{15}$$

The important result is that in the case of a rounding to nearest arithmetic $\overline{\alpha} = 0$ because of the symmetry of the domain of round-off errors. This is not true for other roundings. The numerical values of $\overline{\alpha}$ and σ for the two hypotheses and rounding to nearest and rounding to zero are summed up in Table 1.

Table 1 shows clearly that both hypotheses lead to very close numerical values. As mentioned above if some different but reasonable hypothesis is done for the repartition of the error r (for example also chose Hamming's hypothesis for r) the numerical values are not much changed.

	rounding to nearest		rounding to zero	
	$\overline{\alpha}$	σ	$\overline{\alpha}$	σ
Constant distribution	0	$0.408\ 2^{-t}$	$-0.693\ 2^{-t}$	$0.431\ 2^{-t}$
Hamming's distribution	0	$0.425\ 2^{-t}$	$-0.718\ 2^{-t}$	$0.454\ 2^{-t}$

Table 1: Mean value and standard deviation of the relative assignment error

3 Numerical experiments for inner products

3.1 Case of exact data, IEEE arithmetic

It follows from the values of Table 1 and keeping the highest order terms in formulas (10), (11) in the previous subsection, that when the computer uses a rounding to nearest arithmetic then $\overline{\alpha} = 0$ and thus the mean error on a computed inner product is 0 and the square root of its quadratic mean error is $O(\sqrt{n})$. On the contrary when the arithmetic has a rounding to zero or to $+\infty$ or to $-\infty$ then $\overline{\alpha} \neq 0$ and consequently the mean error $\bar{\rho}$ is $O(n)$. This theoretical result is summed up in Table 2. Numerical experiments have been performed as follows. On a standard PC with IEEE

rounding to zero	rounding to nearest
$\overline{\rho} \simeq \overline{\alpha}\ p\ n/2$	$\overline{\rho} = 0$
$\overline{\rho^2} \simeq \overline{\alpha^2}\ n^2(3p^2 + s^2)/12$	$\overline{\rho^2} \simeq \sigma\ s^2 n/6$

Table 2: The mean error and quadratic mean error in an inner product

754 arithmetic n pairs (X_i, Y_i) have been randomly generated between two arbitrary bounds a and b in double precision for various size n. The inner products $P = \sum_{i=1}^{n} X_i Y_i$ have been computed in single and in double precision with rounding to zero and with rounding to nearest arithmetic. The double precision result is supposed to be exact compared to the one with single precision. In the case of rounding to zero the relative error being theoretically proportional to n with a coefficient $\overline{\alpha} = 2^{-24}\log(2)$ the experimental value $\beta = 2^{24}|relat.error|/n$ is reported in Fig 1. In the case of a rounding to nearest arithmetic it is the value of $\beta = 2^{24}|relat.error|/\sqrt{n}$ which is reported in Fig 2. In both cases the exact result, the relative error, the number of exact significant digits on the inner product and the coefficient β are given.

Results are reported in Table 3 for rounding to nearest and in Table 4 for rounding to zero, for $a = -100$, $b = 100$.

It can be seen that there is some dispersion in the values of β. In particular, from the formulas in Table 2 this value should be close to $0.7/2 = 0.35$ in the case of a rounding to zero arithmetic. Table 5 shows the results obtained for the error in inner products when the X_i and Y_i are generated in $[0,\ 100]$ instead of $[-100,\ 100]$. In this case all numbers are positive and there is no error compensation. The experimental values of β are very close to theoretical ones.

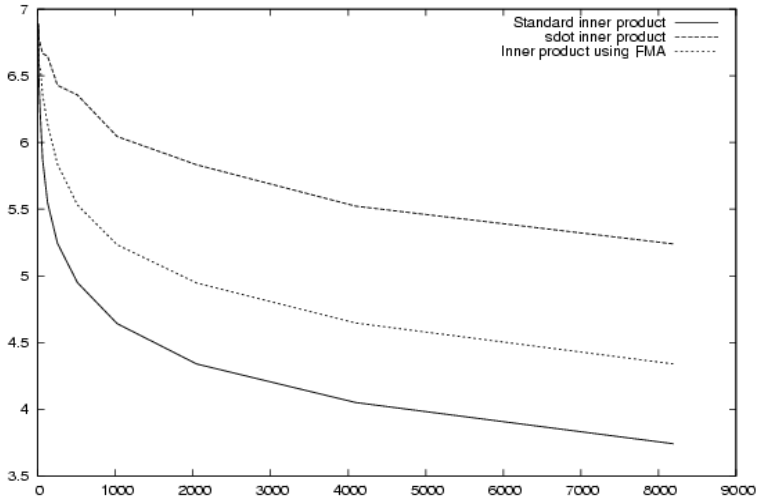

Figure 1: CELL: Inner products of dimension n, Nb exact signif. digits

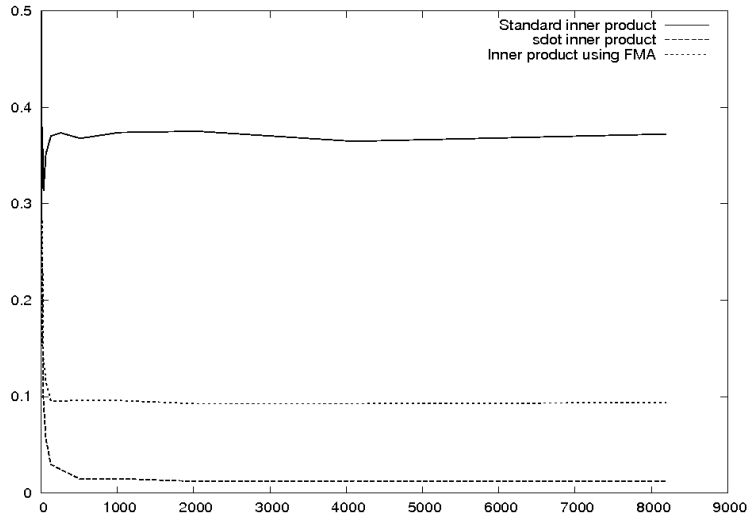

Figure 2: CELL: Inner products of dimension n, coeff $\beta = 2^{24}|relat.error|/n$

n	*result*	*relative error*	*nb sign. dig.*	β
10	$0.50807393E+04$	$0.53093494E-07$	7.00	0.282
100	$0.21252789E+05$	$0.33788535E-07$	7.00	0.057
1000	$0.11471347E+06$	$0.19676749E-06$	6.71	0.104
10000	$-0.12076502E+06$	$0.10804285E-05$	5.97	0.181
100000	$-0.61547312E+06$	$0.59440595E-05$	5.23	0.315

Table 3: Rounding to nearest, $-100 \leq X_i, Y_i \leq 100$, relative error is $O(\sqrt{n})$

n	*result*	*relative error*	*nb sign. dig.*	β
10	$0.50807378E+04$	$0.23521963E-06$	6.63	0.395
100	$0.21252777E+05$	$0.58518668E-06$	6.23	0.098
1000	$0.11471073E+06$	$0.23639801E-04$	4.63	0.397
10000	$-0.12072600E+06$	$0.32415066E-03$	3.49	0.544
100000	$-0.61404406E+06$	$0.23159622E-02$	2.64	0.389

Table 4: Rounding to zero, $-100 \leq X_i, Y_i \leq 100$, relative error is $O(n)$

3.2 Case of exact data, CELL processor

The same inner product has been computed in single precision on the parallel (SPU) processors of the CELL with three different methods: i) the standard accumulation algorithm (2); ii) the special function provided by the IBM BLAS library routines; and iii) the special operation fused multiply and add (FMA). The FMA function requires six cycles but pipelining provides one result every cycle. The results for the standard algorithm are reported in Table 6 showing clearly an error in order of $O(n)$.

Concerning the special function `sdot_spu` and the FMA the experiments prove that in both cases the error is still in $O(n)$ and that only the coefficient β differs. Figures (1) and (2) reporting the number of exact significant digits on the result and the coefficient β for the three methods illustrate this assertion.

3.3 Case of uncertain data

It is supposed here that the data are uncertain but are distributed with a Gaussian law with known mean value and a known standard deviation in some interval. These data can then be modelled by so-called stochastic numbers working with stochastic arithmetic [3], [4], [5], [11], [12]. In particular it has been shown that the sum of n stochastic numbers has a standard deviation in $O(\sqrt{n})$. This phenomenon has an explanation which is close to the one of the computation of inner products with a rounding to nearest arithmetic. In both cases the errors have a symmetric repartition around their mean value. This is also an illustration of Wilkinson's theory in which approximate computation on exact data can be viewed as exact computation on some perturbed data and vice-versa [17]. The following experiment illustrates the theory. In this experiment the X_i are n uniform random numbers generated in $[0, 100]$ with a relative error $\varepsilon = 0.001$. Then for each X_i, 30 samples have been generated in its range of uncertainty with a Gaussian distribution such that their mean-value is X_i

n	*result*	*relative error*	*nb sign. dig.*	β
10	$0.26858420E+05$	$0.36228360E-06$	6.44	0.608
100	$0.27473297E+06$	$0.20813466E-05$	5.68	0.349
1000	$0.24908610E+07$	$0.23635033E-04$	4.63	0.397
10000	$0.24885198E+08$	$0.22302596E-03$	3.65	0.374
100000	$0.24798467E+09$	$0.20583759E-02$	2.69	0.345

Table 5: Rounding to zero, $0 \leq X_i, Y_i \leq 100$, relative error in $O(n)$

n	*result*	*relative error*	*nb sign. dig.*	β
4	$+5.835503e+01$	$+1.29e-07$	6.89	0.539
16	$+3.127193e+02$	$+3.62e-07$	6.44	0.379
64	$+1.624892e+03$	$+1.34e-06$	5.87	0.352
256	$+5.865698e+03$	$+5.70e-06$	5.24	0.373
1024	$+2.485485e+04$	$+2.28e-05$	4.64	0.374
4096	$+9.967466e+04$	$+8.91e-05$	4.05	0.365
8192	$+1.994063e+05$	$+1.82e-04$	3.74	0.372

Table 6: Inner products on the CELL, $0 \leq X_i, Y_i \leq 10$, relative error in $O(n)$

and their standard deviation is $\delta = \varepsilon/2$. Then the 30 sums have been computed from these samples leading to the mean value and standard deviation of the results. These are reported in Table 7. The exact digits in the results are in bold characters. This table shows a non evident property which is: if all terms are known with the same precision, the longest the sum, the more accurate the result.

n	*result*	*relative error*	*nb sign. dig.*	*Std. dev*$/\sqrt{n}$
10	$\mathbf{0.5649}7847D+03$	$0.186E-04$	4.21	$0.581E-01$
100	$\mathbf{0.4911}4757D+04$	$0.184E-04$	4.71	$0.540E-01$
1000	$\mathbf{0.51429}722D+05$	$0.687E-05$	5.55	$0.528E-01$
10000	$\mathbf{0.49883}679D+06$	$0.147E-05$	5.57	$0.525E-01$
100000	$\mathbf{0.498633}30D+07$	$0.437E-06$	6.05	$0.491E-01$

Table 7: Sum of n imprecise numbers. Relative error in $O(\sqrt{n})$

4 Solving linear systems with the CELL

As said in the introduction the CELL processor has on chip parallel processors with a very inaccurate division. Namely the only low level function is not a division but an inversion called `spu_re` and this inversion provides only twelve exact bits on the result. Thus $y = a/b$ must be computed as $y = a * (1/b)$ and is generally very inaccurate. This is illustrated by the following example.

Solve Wilson's system $AX = B$, see [16] using a classical Gaussian elimination method with standard IEEE754 rounding to nearest and rounding to zero arithmetic and with the CELL parallel SPU using standard division or SIMD instructions. The system is:

$$\begin{pmatrix} 5 & 7 & 6 & 5 \\ 7 & 10 & 8 & 7 \\ 6 & 8 & 10 & 9 \\ 5 & 7 & 9 & 10 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 23 \\ 32 \\ 33 \\ 31 \end{pmatrix}$$

The exact result is $x_i = 1, i = 1, ..., 4$. The computed solutions are given in Table 8.

IEEE 754		CELL Processor	
rounding to nearest	rounding to zero	standard div.	SIMD instr.
$0.99999970E+00$	$0.10000181E+01$	$1.0000135E+00$	$1.0407186E+00$
$0.10000004E+01$	$0.99998879E+00$	$9.9999154E-01$	$9.7517973E-01$
$0.99999982E+00$	$0.99999547E+00$	$9.9999678E-01$	$9.9008989E-01$
$0.10000001E+01$	$0.10000027E+01$	$1.0000020E+00$	$1.0059216E+00$

Table 8: Solution of Wilson's system with IEEE754 and CELL arithmetic

It is clear that the standard division provides results close to the IEEE 754 rounding to zero arithmetic whereas the fast SIMD instructions give much poorer results.

5 Is CELL optimal for scientific computation?

The CELL is a very fast and very powerful processor for single precision numbers, thus allowing the design of extremely powerful super computers. Anyhow as mentioned in the beginning of this study it has been designed for multimedia applications which generally do not require very high precision. In fact the first version has not been designed for accurate floating point computation.

First, the rounding to zero mode implemented in the parallel SPU is clearly faster than any other rounding mode, but as seen here it is statistically much less accurate than rounding to nearest. In particular, in the case of inner products rounding to zero leads to an error in $O(n)$ whereas rounding to nearest produces an error in $O(\sqrt{n})$. This is certainly why the default IEEE 754 rounding mode is actually rounding to nearest. It is true that multimedia applications do not generally require the use of very long inner products, but what about large linear problems coming from optimization, meteorology, fluid and solid mechanics etc.? It must also be noted that BLAS are more efficient than the standard computation algorithm but still are with an error in $O(n)$.

Second, the division implemented on the SPU as a very imprecise multiplication by inverse may also lead to very inaccurate or even totally erroneous results. These two features encourage the scientific programmer to make an extensive use of double precision but this has produced on our tests an enormous decrease of performance. In the tested version double precision arithmetic was a 15 Gigaflops one. But it must be noted that the last version of the CELL which is used in the Roadrunner has a 100 Gflops double precision. Anyhow, a mix of fast single precision and few well chosen double precision operations can provide very fast and still accurate results, see for

example [6], but this supposes as a preamble a keen analysis of the problem. Therefore, the conclusion is that when using the CELL for solving large (or unstable) scientific problems, it is necessary to be aware that this processor is fast and powerful but has an inaccurate single precision arithmetic. Then inaccurate computations may produce false results. Consequently it is necessary to chose adapted algorithms, to make an extensive use of the special functions of the CELL (Blas, FMA), or of interval methods adapted to the CELL, see [7], and above all to control the solution.

References

[1] R. Alt, *Etude statistique de l'erreur numérique d'affectation sur un ordinateur d'arithmétique en base quelconque. Application à l'erreur commise dans le calcul d'une somme de produits de nombres*, Techn. Rep. IP 76.5., Inst. de Programmation, Univ. Pierre et Marie Curie, Paris, 1976 (in French).

[2] R. Alt, "Error propagation in Fourier Transforms", *Math. Comp. in Sim.*, vol. 20, pp. 37–43, 1978.

[3] R. Alt, J.-L. Lamotte, and S. Markov, "On the numerical solution to linear problems using stochastic arithmetic", In: H. M. Haddad et al. (eds.), *Applied Computing 2006* (Proc. 2006 ACM Symposium on Applied Computing, SAC'06, Dijon, France, April, 23–27, 2006), ACM, pp. 1635–1639, 2006.

[4] R. Alt, J.-L. Lamotte, and S. Markov, "Abstract structures in stochastic arithmetic", In: B. Bouchon-Meunier and R. R. Yager (Eds.), *Proc. 11-th Conference on Information Processing and Management of Uncertainties in Knowledge-based Systems IPMU'2006*, Editions EDK, Paris, pp. 794–801, 2006.

[5] R. Alt, J.-L. Lamotte, and S. Markov, "Numerical study of algebraic problems using stochastic arithmetic", In: I. Lirkov, S. Margenov, and J. Wasniewski (Eds.), *Large-Scale Scientific Computing*, Springer Lecture Notes in Computer Science, vol. 4818, pp. 123–130, 2008.

[6] M. Baboulin, A. Buttari, J. Dongarra, J. Kurzak, J. Langou, J. Langou, P. Luszczek, and S. Tomov, "Accelerating scientific computations with mixed precision algorithms", *Computer Physics Commun.*, vol. 180, no. 12, pp. 2526–2533, 2009.

[7] S. Graillat, J.-L. Lamotte, and Diep Nguyen Hong, "Extended precision with a rounding mode toward zero environment. Application on the Cell processor", *Int. J. Reliability and Safety*, vol. 3, no. 1/2/3, pp. 153–173, 2009.

[8] R. W. Hamming, "On the distribution of numbers", *The Bell Syst. Tech. J.*, vol. 40, pp. 1609–1625, 1970.

[9] M. La Porte and J. Vignes, "Evaluation statistique des erreurs dans les calculs sur ordinateur", *Proc. Canadian Comp. Conf.*, pp. 414201–414213, 1972.

[10] M. La Porte, and J. Vignes, *Algorithmes numériques, analyse et mise en oeuvre*, vol. 1, Technip Eds., Paris, 1974.

[11] S. Markov, R. Alt, and J.-L. Lamotte, "Stochastic arithmetic: S-spaces and some applications", *Numer. Algo.*, vol. 37, no. 1–4, pp. 275–284, 2004.

[12] S. Markov and R. Alt, "Stochastic arithmetic, addition and multiplication by scalars", *Appl. Numer. Math*, vol. 50, pp. 475–488, 2004.

[13] N. Tsao, "On the distribution of significant digits and round-off errors", *Communications of the ACM*, vol. 17, no. 5, pp. 269–271, 1974.

[14] J. Vignes, "A stochastic arithmetic for reliable scientific computation", *Math. and Comp. in Sim.*, vol. 35, pp. 233–261, 1993.

[15] J. Vignes, "Discrete stochastic arithmetic for validating results of numerical software", *Numer. Algo.*, vol. 37, pp. 377–390, 2004.

[16] J. R. Westlake, *A handbook of numerical matrix inversion and solution of linear equations*, Wiley, 1968, pp. 136–157.

[17] J. H. Wilkinson, *Rounding errors in algebraic processes*, Prentice Hall, Englewood Cliffs, N. J., 1963.