Testing and Applications of Stochastic Rounding

Vladislav Yotkov

February 24, 2023

Abstract. *This report investigates the effect of Stochastic Rounding (SR) in software applied in Floating-Point (FP) arithmetic. To achieve this, we carry out three experiments that demonstrate the performance of SR in repeated rounding, harmonic series summation, and vector inner product simulation. Our observations are that in binary 32 arithmetic, the absolute error of SR is significantly lower than the one in RN due to the former "curing stagnation" [2] by maintaining zero-mean rounding errors ([1], [3]). Numerical results are provided to illustrate the effectiveness of SR incorporated into computationally-intensive tasks.*

1. Introduction

Rounding is the operation of finding the next representable floating-point (FP) number in an FP system (as defined in [2]) implemented either in hardware or software. The IEEE standard 754 defines four rounding modes in FP arithmetic: Round to Nearest (RN) - default, Round to Zero (RZ), Round to Infinity (RU), and Round to Negative Infinity (RD) [7]. Hence, if we take the example of rounding the number *0.3*, applying the default operation will result in *RN(0.3) = 0*. However, performing following mathematical operations and functions (*e.g.*, summation, multiplication, etc.) with this rounded number would cause the propagation of errors and, ultimately, significantly reduce the accuracy. A way to tackle this problem is to implement a probabilistic method - Stochastic Rounding (SR), that will round up or down with probabilities *p* and *1-p*, respectively. In this way, we can minimise the propagation of the errors by allowing the error terms to cancel out throughout the calculation of the result. It is essential in various problems such as Neural Network Training ([6], [9]), ODE Solvers [5], and Weather Forecasting [8].

Formally, we can define SR as:

$$P \sim U(0,1) \tag{1}$$

$$p = \frac{x - RZ(x)}{|RZ(x) - RA(x)|} \tag{2}$$

$$SR(x) = \begin{cases} RZ(x) & \text{if } P \leq 1 - p \\ RA(x) & \text{otherwise} \end{cases} \tag{3}$$

where *P* is an FP-representable probability sampled from the uniform distribution, *p* is the normalized distance to one of the IEEE 754-defined rounding methods (*i.e.*, RZ), and hence *1-p* being the probability of applying it (intuition is that the closer *RZ(x)* to *x* is, the more likely should it be *SR(x)* to be *RZ(x)*).

2. Repeated Rounding

After having mathematically defined Stochastic Rounding (SR) in *Equation 3*, we will implement it in C and analyse the behaviour under different *p*-probability values. We will discuss the key procedural steps of the implementation below:

1. Calculate *RZ(x)* - amounts to finding the next representable binary32 FP-value no larger in magnitude than the argument *x* as defined in [2].

2. Calculate *RA(x)* - amounts to finding the next representable binary32 FP-value towards $\pm\infty$ depending on the sign of the argument *x*.

3. Calculate *p*-probability as the normalized distance, described in *Equation 2*.

4. Sample from the uniform distribution - Generating a random probability could be practically achieved either through *C*s *Arc4RandomUniform* or through normalizing the *rand* by the $(RANDMAX+1)$ to guarantee specified bounds $[0, 1)$. However, neither of the approaches guarantees absolute randomness.

5. Compute *Equation 3* with above 4 values.

For the repeated rounding experiment two SR methods were compared with binary64 input - one which is defined by the elaborated procedure above, and the other - applying a bit mask to the argument value that truncates the least significant bits of the significand, thus simulating rounding to binary32. Our hypothesis is that both implementations should behave similarly and produce similar statistics regarding the probabilities of rounding up/down.

In fact, results (Tables 1, 2) confirm our expectations for rounding of $ $\pi, astheyshowthat$*SR averag* the true probabilities of both SR methods will be the same.

Value being rounded	3.141592653589793115997963468544185161590576171875000
SR average value	3.141592653628730857917616958729922771453857421875000
SR alt. average value	3.141592653506755983272569210384972393512725830078125
Binary32 value before	3.141592502593994140625000000000000000000000000000000
Binary32 value after	3.141592741012573242187500000000000000000000000000000
Closest binary32	3.141592741012573242187500000000000000000000000000000

Table 1. Repeated Rounding Results of π for SR Provided and Alternative

SR Alt Approximate Probability RU	0.632974
SR Alt Approximate Probability RD	0.367026
SR Expected Probability RU	0.633322
SR Expected Probability RD	0.366678
SR Provided Probability RU	0.633486
SR Provided Probability RD	0.366514

Table 2. Rounding Probabilities of π for SR Provided and Alternative

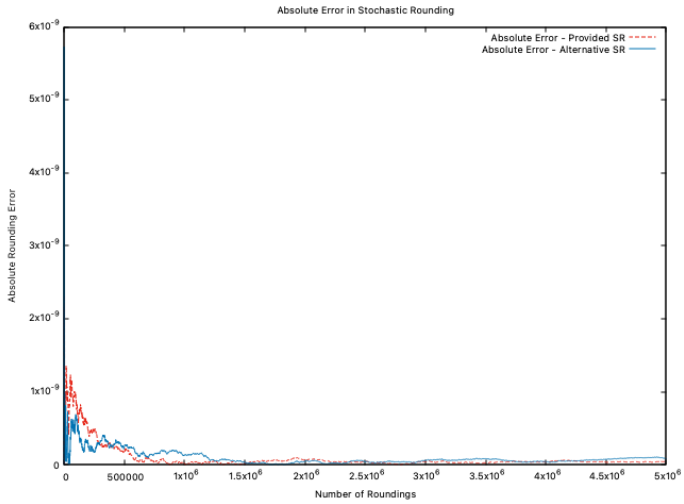

Figure 1. Absolute Error in Stochastic Rounding of π: Provided vs Alternative

3. Stagnation and the Harmonic series

The next experiment we carry out is computing a truncated version of the Taylor series based on: RN rounding (default), SR Alternative rounding (from *Equation 3*), and Compensated Summation [4].

Results (Tables 3, 4, and Figure 2) suggest that the Compensated Summation has a slightly lower absolute error compared with the SR Summation, possibly because of the nature of the algorithm - adding the error back to the sum, while the SR method is also reliant on the random function and hence the difference between the two errors (Figure 2). Nevertheless, both methods significantly outperform the RN Summation, which we believe encounters the phenomenon of stagnation - occurring when small values get lost during rounding [2]. Therefore, in single precision arithmetic, the RN summation fails to accumulate the increasingly smaller terms, and the Harmonic series value considerably diverges from the double precision, hence the absolute error of $5.20...$ (Table 4).

Furthermore, we find out that the point of stagnation of the RN summation happens on the $699,050$th iteration. Our methodology is that we assess whether the difference between two following summation values, a, b is within a relative tolerance of $FloatEpsilon * max(a, b)$. We use relative instead of absolute tolerance to make the comparison less sensitive to the magnitudes of the arguments.

Recursive summation, binary32	15.4036827087402343750000000000000
Recursive summation with SR, binary32	20.6337852478027343750000000000000
Compensated summation, binary32	20.6073341369628906250000000000000
Recursive summation, binary64	20.6073343222888425430028291884 81

Table 3. Approximating value of Harmonic Series after 500,000,000 iterations

Recursive summation error, binary32	5.203651613548608168002829188481
Recursive summation with SR error, binary32	0.026450925513891831997170811519
Compensated summation error, binary32	0.000000185325951918002829188481

Table 4. Absolute errors of the Harmonic Series summation after 500,000,000 iterations

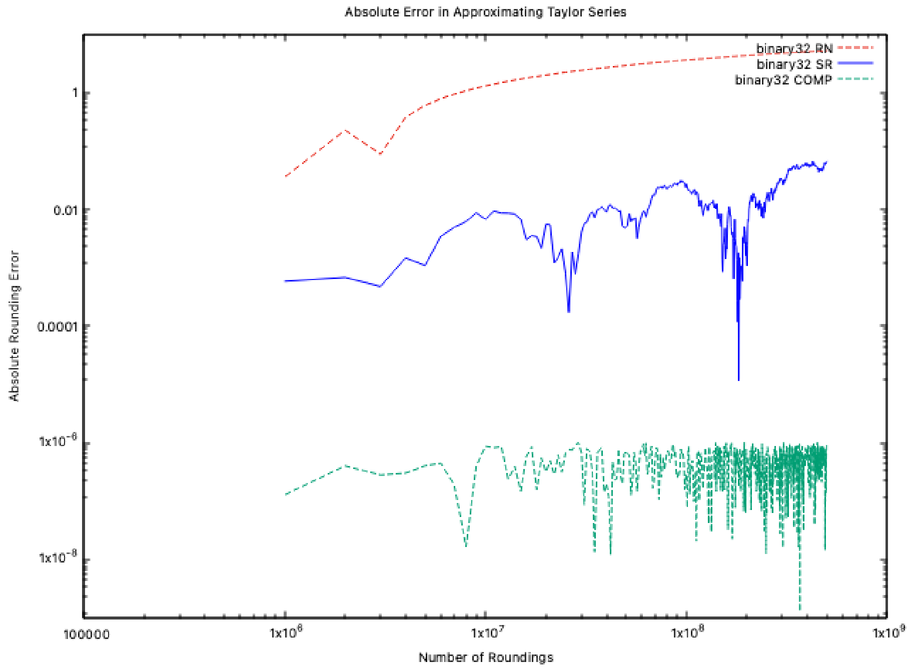

Figure 2. Stagnation and the Harmonic series

4. Vector Inner Product

For the third experiment, we chose to implement vector inner product ([1], [2]) and thus demonstrate the superiority of Stochastic rounding over the default Round-to-Nearest one. To achieve this, we decide on the vector size and assign random uniform values in $[0, 1)$ to the elements. However, instead of allocating the memory for the vector, we simply simulate the dot product by directly generating pairs of vector elements and accumulating their product. By choosing a significantly large vector size of $5,000,000$, we then force the RN into stagnation and, ultimately, show the effectiveness of the SR (Table 5, Figure 3).

binary32 with RN	4288.02552
binary32 with SR	146.65052

Table 5. Absolute error in vector inner product calculation

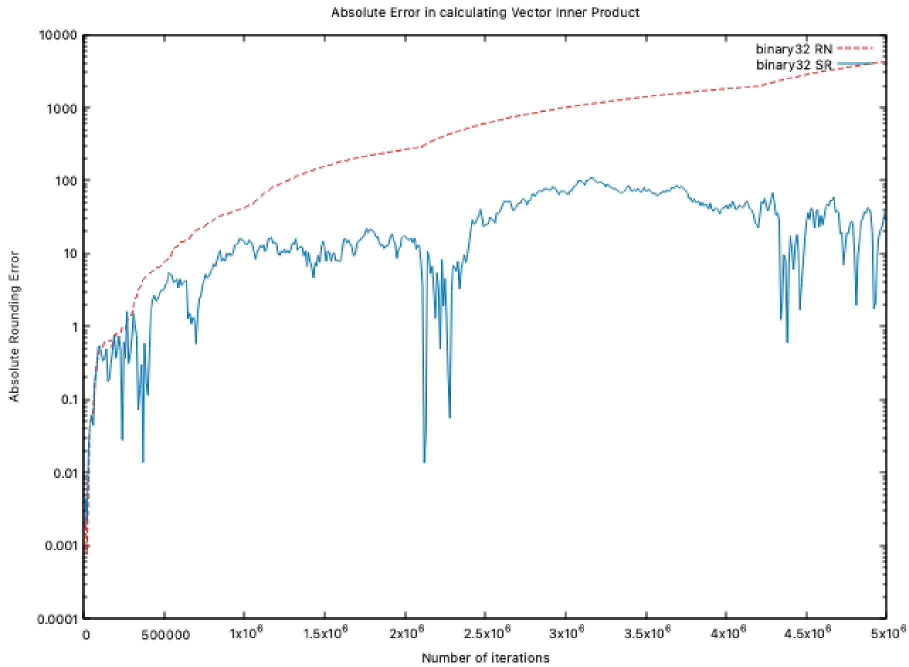

Figure 3. Absolute error in vector inner product calculation

Copmiler Details

Hardware and software specifications:

- Processor: 1.7 GHz Quad-Core Intel Core i7

- C version supported: 201710

References

[1] Connolly, M. P., Higham, N. J., and Mary, T. (2020). Stochastic rounding and its probabilistic backward error analysis. *SIAM J. Sci. Comput.*, 43:A566–A585.

[2] Croci, M., Fasi, M., Higham, N. J., Mary, T., and Mikaitis, M. (2022). Stochastic rounding: implementation, error analysis and applications. *Royal Society Open Science*, 9(3):211631.

[3] Fasi, M. and Mikaitis, M. (2021). Algorithms for stochastically rounded elementary arithmetic operations in ieee 754 floating-point arithmetic. *IEEE Transactions on Emerging Topics in Computing*, 9(3):1451–1466.

[4] Higham, N. J. (1993). The accuracy of floating point summation. *SIAM Journal on Scientific Computing*, 14(4):783–799.

[5] Hopkins, M., Mikaitis, M., Lester, D. R., and Furber, S. (2020). Stochastic rounding and reduced-precision fixed-point arithmetic for solving neural ordinary differential equations. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 378(2166):20190052.

[6] Lin, D. D. and Talathi, S. S. (2016). Overcoming challenges in fixed point training of deep convolutional networks.

[7] Muller, J.-M., Brunie, N., de Dinechin, F., Jeannerod, C.-P., Joldes, M., Lefvre, V., Melquiond, G., Revol, N., and Torres, S. (2018). *Handbook of Floating-Point Arithmetic*. Birkhäuser Basel, 2nd edition.

[8] Paxton, E. A., Chantry, M., Klöwer, M., Saffin, L., and Palmer, T. (2022). Climate modeling in low precision: Effects of both deterministic and stochastic rounding. *Journal of Climate*, 35(4):1215 – 1229.

[9] Qian Zhang, S., McDanel, B., and Kung, H. T. (2022). Fast: Dnn training under variable precision block floating point with stochastic rounding. pages 846–860.