Title: Infrastructure in Financial Precision: Theory and Simulation

Authors:

1. Abhilasha Aswal
   International Institute of Information Technology, Bangalore
   Address:
   26/C Electronics City,
   Hosur Road,
   Bangalore – 560100
   Mobile: +91 – 9620058383
   E-mail: abhilasha.aswal@iiitb.ac.in

2. Ganesh Perumal M
   International Institute of Information Technology, Bangalore
   Address:
   26/C Electronics City,
   Hosur Road,
   Bangalore – 560100
   Mobile: +91 – 9986673971
   E-mail: ganesh_perumal@iiitb.ac.in

3. Prof. G. N. Srinivasa Prasanna
   International Institute of Information Technology, Bangalore
   Address:
   26/C Electronics City,
   Hosur Road,
   Bangalore – 560100
   Mobile: +91 – 9844226959
   E-mail: gnsprasanna@iiitb.ac.in

I. INTRODUCTION

Traditionally, calculations in the financial world are specified in decimal arithmetic. Many early computers used decimal arithmetic in hardware level, but binary computing in hardware soon took over.One main reason for this was major issues in hardware and software reliability around the 1950's [6], [7]. Von Neumann et al [7] noted that binary arithmetic gave reasonable precision for scientific calculations, but it may not be sufficient for today's many high precision applications. The use of binary hardware seemed necessary at the time, but now computer hardware is cheaper and far more robust, yet most computers today still use binaryhardware.IEEE standards were developed to provide standard algorithms for developing portable software[8]. It is interesting to check the impact of using binary arithmetic directly for financial transactions. Our study quantifies the resultant impact.

Our work is based on examination of the error which is defined as the *difference* between the answer produced using arbitrary precision decimal arithmetic, and that produced by the IEEE 754 binary arithmetic, afterfinancial rounding rules are applied.

For this study we have a theory that predicts the worst case error and a simulation infrastructure that can be exercised to get error behaviors and to do statistical analysis. The simulator can compare arbitrary precision decimal arithmetic results with those of finite precision binary arithmetic, can generate sequences of transactions including deposits, withdrawals, transfer of funds, interest calculations etc. both in a single currency and multi-currency setting, has currency specific rounding rules implemented and can perform basic statistical tests such asChi-squared, KS, T and F tests.Our theory predicts that if malicious agents have the knowledge of the system, then they may trigger an attack by deliberately choosing transaction amounts such that errors are made to their advantage. Sequence of such transactions will lead to accumulation of large errors which may go unnoticed in basic statistical tests.

A natural question is: if binary arithmetic cannot be used for financial calculations, then what are the financial software using? Contemporarily, software libraries are used for high precision computations such as financial transactions. However, these specialized software libraries suffer from considerable performance penalty over hardware [4]. If for the sake of performance, IEEE binary standard was used for financial computations, then the results may not be same as the exact decimal results. If a malicious agent can determine this difference then they can manipulate the system to their advantage and they may end up gaining significant sums of money illegally.

Figure 1 shows a fairly recent invoice of an airline ticket, which shows the total amount as 4132.639999999999 instead of 4132.64. This indicates that the airlines probably used IEEE 754 double precision for calculation instead of a high precision decimal library and testifies to the importance of the problem.

IndiGo Flight(s)

Flt #	Date	From	To	Depart	Arrive	Seat #
6E415	30Dec10	Mumbai (BOM)	Bangalore (BLR)	21:00	22:40	

IndiGo Fares

Base Fare	1,594.00
Passenger Service Fee	229.00
Airline Fuel Charge	1,850.00
Transaction Charge	50.00
Service Tax	103.00
User Development Fee	100.00
Total Price	3,926.00
Transaction Fee	206.64
Total Payment	4,132.639999999999

FYBKDD

Figure 1: An airline ticket invoice

Our major contribution is the creation of the simulation infrastructure and the theoretical analysis. In summary, our results indicate that the use of binary arithmetic to perform decimal arithmetic is inappropriate from at least a legal standpoint. The errors are typically very small in magnitude, but sequences of transactions exist, whose cumulative error is significant, and we showed that almost any required error process can be generated.

Our work is very relevant for financial institutions like banks, capital marketing, stock market etc., as well as for applications in retail and high precision machine design as they can utilize our error quantifying techniques.

In the rest of this proposal, we give an outline of these ideas (details available on request). Section II contains a brief overview of the simulation infrastructure. Section IV discusses analysis of error patterns using a theoretical analysis and shows that sequences of transaction volumes exist which can cause errors in each transaction. Section IV concludes.

II. SIMULATION INFRASTRUCTURE AND ITS POWER

We quantified the effects of using ubiquitous IEEE-754 binary arithmetic for financial calculations using our simulator.Figure 2 shows the architecture of our simulator. It has 3 major parts-

1. **Input:** This module generates the input traffic for the simulator and uses recent exchange rate data from the internet.
2. **Simulation engine:** This module can find arbitrage opportunities from currency or stock exchange rate graphs and also performs basic stock and banking transactions.
3. **Error process analysis:** This module performs statistical analysis on the error process and generates reports.

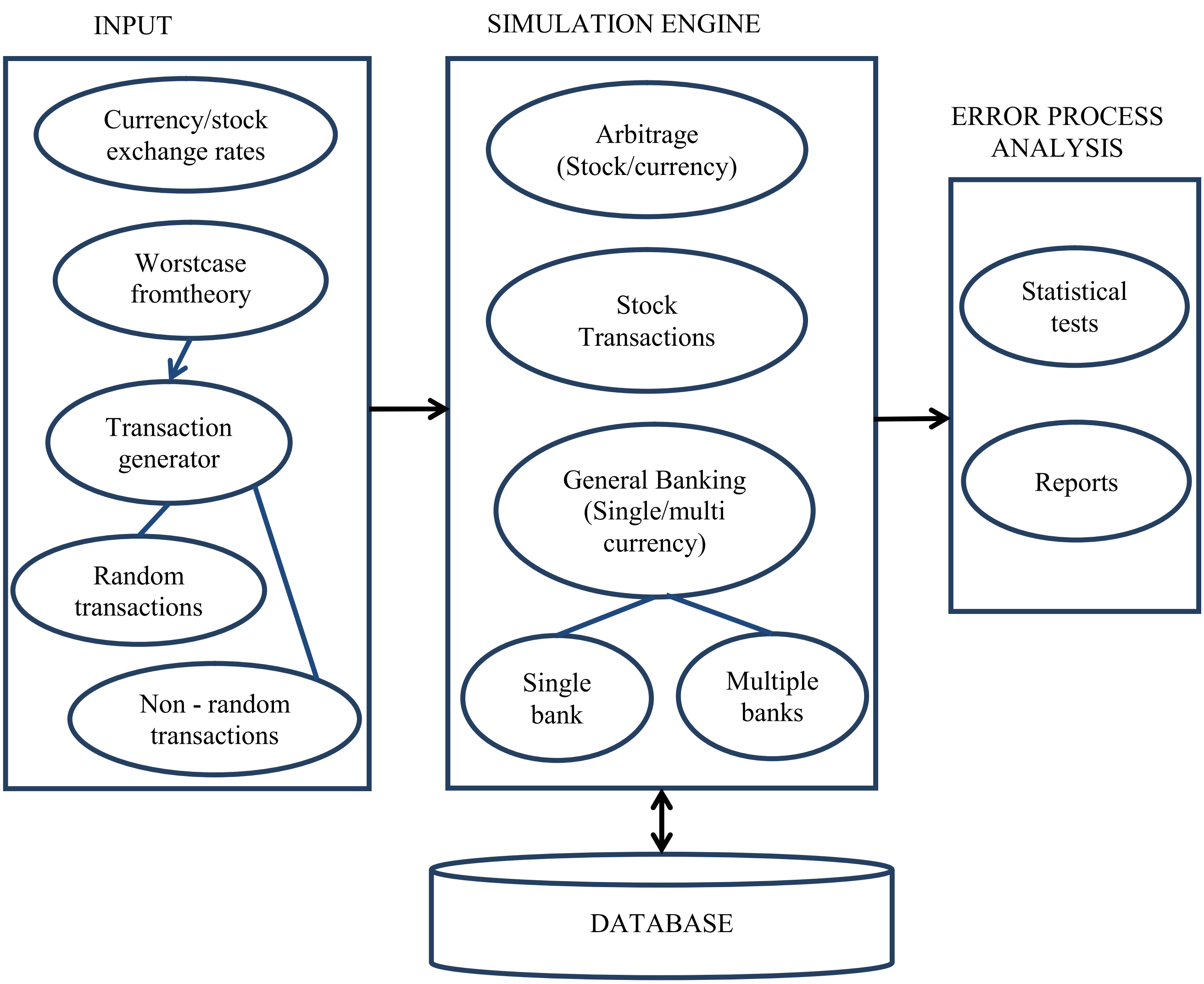

Figure 2: Architecture of the simulator

The simulator can simulate a random sequence of transactions and statistically analyze the simulation results or it can also re-run an earlier simulated sequence with a different precision level in binary arithmetic. Transactions include simple deposits, withdrawals, splits – where money is withdrawn from an account and split up into *n*parts and deposited in *n* different accounts, simple / compound interest calculations, and buying and selling of stocks. In addition, the transactions can be in a single account, between a pair of accounts having same base currency or pair of accounts having different base currency which involves a currency conversion. The system has currency specific rounding rules built in and can be updated to have the latest currency conversion rates. To improve the performance of the simulator a parallel multithreaded version was developed and a speed up of 94% was achieved. The simulator can perform roughly 18000 transactions/second without multithreading and 35000 transactions/second with multithreading on an Intel dual-core 2.26 GHz processor.

Figure 3 and Figure 4 show snap shots of the simulator and Table 2 shows the result of a statistical analysis. Figure 3 shows the simulator with multiple banks using which multi-currency transactions can be simulated. Figure 4 shows the error process resulting from a simulation of single currency transactions using IEEE 754 single precision.

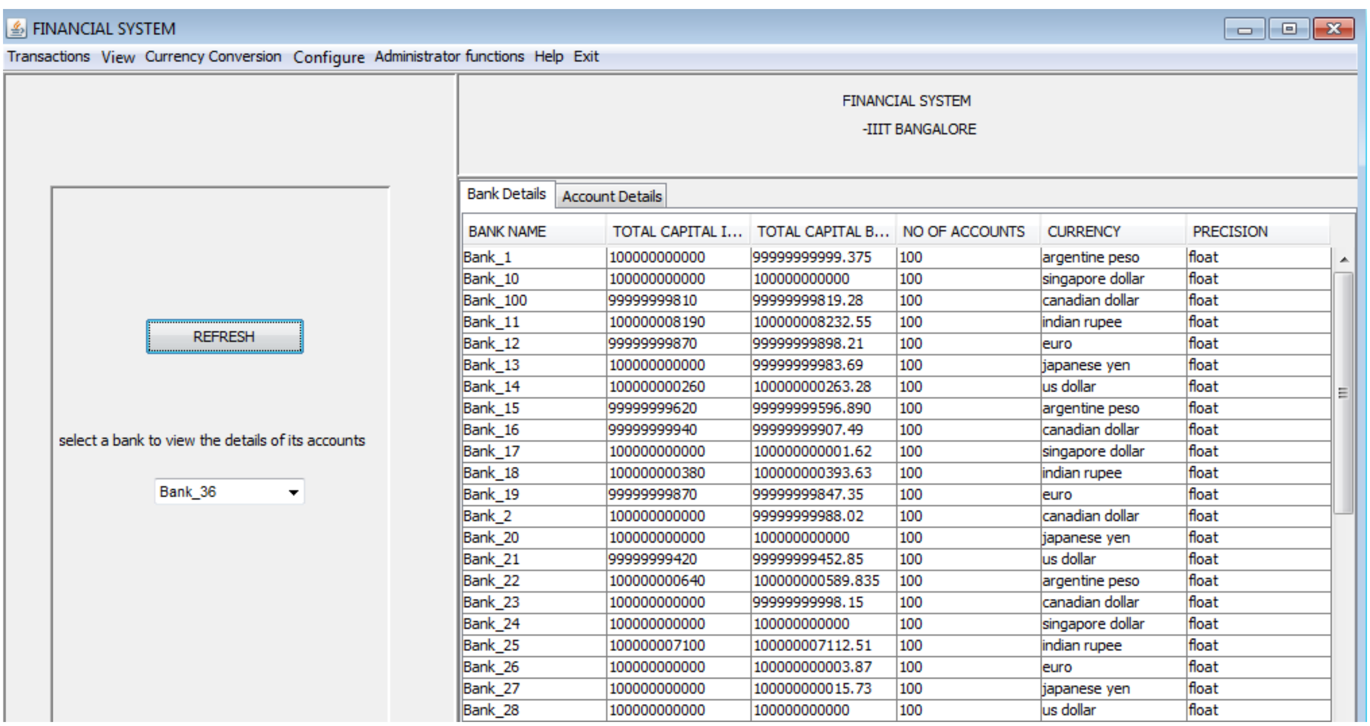

Figure 3: Binary (IEEE 754 Single Precision) vs. Decimal balances

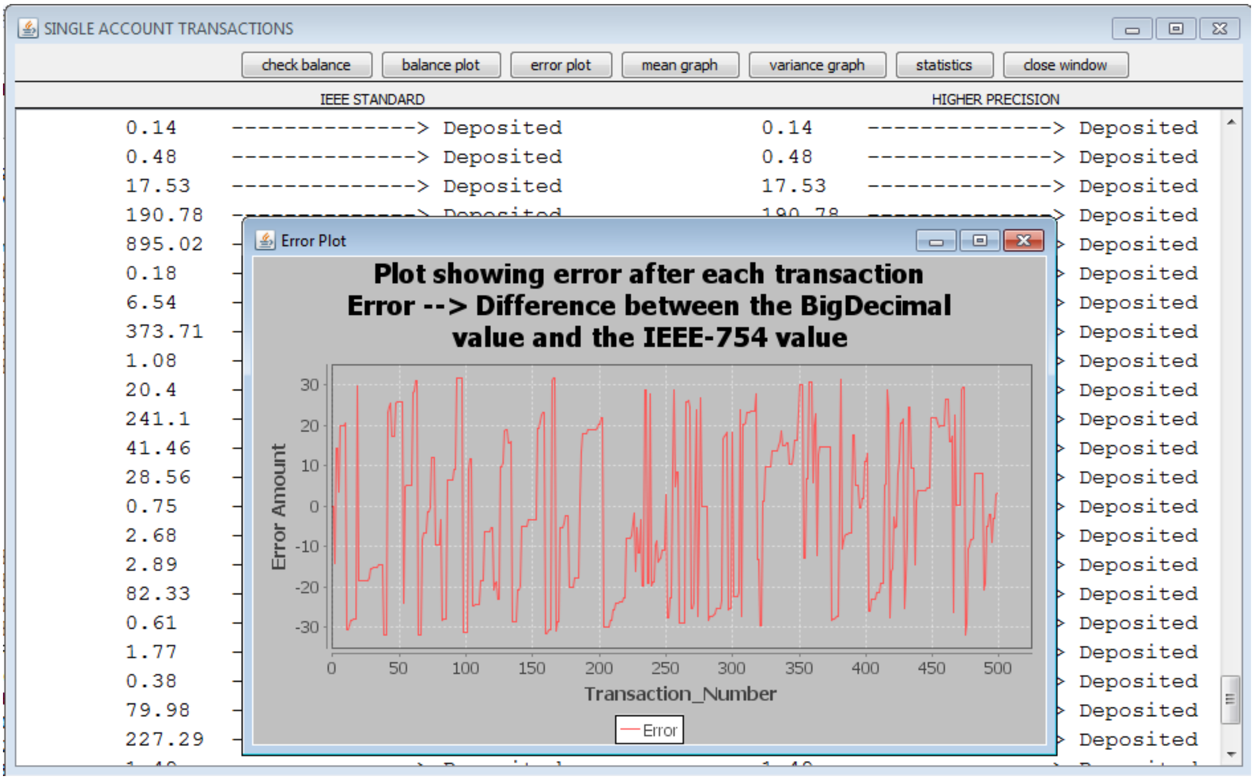

Figure 4: Error pattern when using IEEE 754 single precision

In addition, our simulator can process real-time currency/stock exchange rate data and search for arbitrage opportunities. Figure 5 shows an example of 2 arbitrage cycles in 5 currencies (currency exchange rates as applicable on Sept 29, 2009), one is shown in blue and another in red. The gain in 100 cyclic traversals in blue cycle is just over 4%and in red cycle is 37%.Figure 6 shows another example with 19 currencies (currency exchange rates as applicable on 16th Nov 2009). The gain in this case was just over 7% in 1 cycle. For simplicityspreads are not accounted for in these examples.

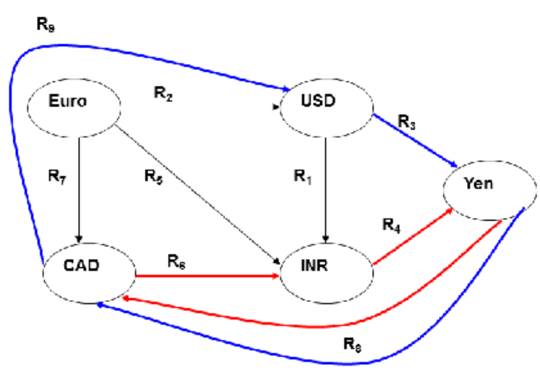

Figure 5: Arbitrage example with 5 currencies

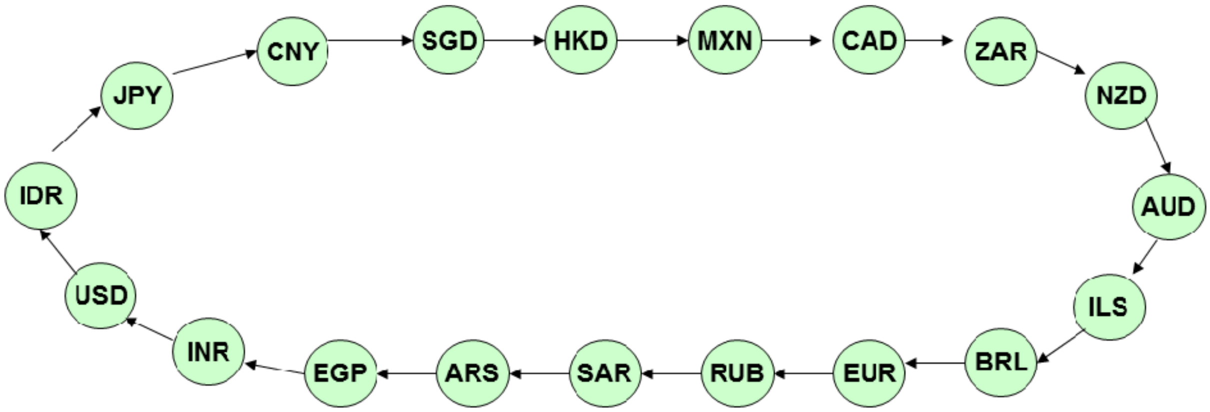

Figure 6: Arbitrage example with a cycle having 19 currencies

III. THEORETICAL ANALYSIS

In addition to the simulator we have a theory to analyze the error process. Our theoretical analysis, compared to previous work [3, 4, 5], systematically categorizes financial transactions, and shows the existence of arbitrary error patterns by a suitable choice of transaction amounts.A novel tabular approach is used to examine the worst case errors, as outlined below. For simplicity of exposition, we have considered a single payer or payee (single currency transactions).With capitalization amounts exceeding 10^{13} (2^{43}), errors are possible in additions using IEEE 754 double precision. Here, using a tabular approach, we demonstrate a sequence of transaction amounts, such that an error is made in every transaction.

	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95
0.05	-0.03	-0.04	0.01	0	-0.01	-0.03	-0.04	0.01	0	-0.01	-0.03	-0.04	0.01	0	-0.01	-0.03	-0.04	0.01	0
0.1	-0.04	-0.05	0	-0.01	-0.03	-0.04	-0.05	0	-0.01	-0.03	-0.04	-0.05	0	-0.01	-0.03	-0.04	-0.05	0	-0.01
0.15	0.01	0	0.05	0.04	0.03	0.01	0	0.05	0.04	0.03	0.01	0	0.05	0.04	0.03	0.01	0	0.05	0.04
0.2	0	-0.01	0.04	0.03	0.01	0	-0.01	0.04	0.03	0.01	0	-0.01	0.04	0.03	0.01	0	-0.01	0.04	0.03
0.25	-0.01	-0.03	0.03	0.01	0	-0.01	-0.03	0.03	0.01	0	-0.01	-0.03	0.03	0.01	0	-0.01	-0.03	0.03	0.01
0.3	-0.03	-0.04	0.01	0	-0.01	-0.03	-0.04	0.01	0	-0.01	-0.03	-0.04	0.01	0	-0.01	-0.03	-0.04	0.01	0
0.35	-0.04	-0.05	0	-0.01	-0.03	-0.04	-0.05	0	-0.01	-0.03	-0.04	-0.05	0	-0.01	-0.03	-0.04	-0.05	0	-0.01
0.4	0.01	0	0.05	0.04	0.03	0.01	0	0.05	0.04	0.03	0.01	0	0.05	0.04	0.03	0.01	0	0.05	0.04
0.45	0	-0.01	0.04	0.03	0.01	0	-0.01	0.04	0.03	0.01	0	-0.01	0.04	0.03	0.01	0	-0.01	0.04	0.03
0.5	-0.01	-0.03	0.03	0.01	0	-0.01	-0.03	0.03	0.01	0	-0.01	-0.03	0.03	0.01	0	-0.01	-0.03	0.03	0.01
0.55	-0.03	-0.04	0.01	0	-0.01	-0.03	-0.04	0.01	0	-0.01	-0.03	-0.04	0.01	0	-0.01	-0.03	-0.04	0.01	0
0.6	-0.04	-0.05	0	-0.01	-0.03	-0.04	-0.05	0	-0.01	-0.03	-0.04	-0.05	0	-0.01	-0.03	-0.04	-0.05	0	-0.01
0.65	0.01	0	0.05	0.04	0.03	0.01	0	0.05	0.04	0.03	0.01	0	0.05	0.04	0.03	0.01	0	0.05	0.04
0.7	0	-0.01	0.04	0.03	0.01	0	-0.01	0.04	0.03	0.01	0	-0.01	0.04	0.03	0.01	0	-0.01	0.04	0.03
0.75	-0.01	-0.03	0.03	0.01	0	-0.01	-0.03	0.03	0.01	0	-0.01	-0.03	0.03	0.01	0	-0.01	-0.03	0.03	0.01
0.8	-0.03	-0.04	0.01	0	-0.01	-0.03	-0.04	0.01	0	-0.01	-0.03	-0.04	0.01	0	-0.01	-0.03	-0.04	0.01	0
0.85	-0.04	-0.05	0	-0.01	-0.03	-0.04	-0.05	0	-0.01	-0.03	-0.04	-0.05	0	-0.01	-0.03	-0.04	-0.05	0	-0.01
0.9	0.01	0	0.05	0.04	0.03	0.01	0	0.05	0.04	0.03	0.01	0	0.05	0.04	0.03	0.01	0	0.05	0.04
0.95	0	-0.01	0.04	0.03	0.01	0	-0.01	0.04	0.03	0.01	0	-0.01	0.04	0.03	0.01	0	-0.01	0.04	0.03

Table 1: Transaction error matrix (TEM)

First, we define the capitalization-transaction error matrix (TEM)$T(D,B)$, where D is the number of decimal digits in the fraction and B is the number of binary bits in the binary approximation, as an $n \times m$matrix, with entries

$$T_{ij} = \text{Error in adding Transaction amount } \Delta_j \text{ to Capital } C_i,$$

where n is the number of possible capitalization values and m is the number of possible transaction amounts. This error is with respect to exact decimal arithmetic.

Analysis of this matrix $\mathrm{T(D,B)}$ enables us to generate worst case, and in general arbitrary transaction sequences.The following lemma can be proved about the TEM-

Lemma:

1. The TEM $T(D,B)$ is completely specified by using only all possible values of the fractional (decimal) portion of transactions and amounts. The integer portions add exactly, and do not cause errors.
2. $T(D,B)$is a *symmetric* matrix for deposits and *skew symmetric* for withdrawals and the rows and columns correspond to the smallest currency unit.
3. Since the row corresponds to using all possible decimal fractional values <= unity, the error period is <= the row length.

Based on 3, we can show (proof omitted) that sequences of transaction amounts exist which are biased while still passing basic statistical tests for randomness. Our algorithm works by randomly choosing one of the many choices available. More details will be given in the presentation.

An example of a TEM is shown in Table 1 for 2 decimal digit values approximated using 4 bits. We have shown only every 5^{th} row and 5^{th} column of the TEM for brevity.

Every transaction will either produce an exact answer or an approximate answer, depending on the numbers involved. We classify these results as zero error, positive error and negative error, where positive error is encountered when the approximated binary result is greater than the exact decimal result and negative error is encountered when the approximated binary result is less than the exact decimal result. We have devised a randomized algorithm to find a sequence of transaction amounts for this such that either always a negative error is made or always a positive error is made.

For the TEM in Table **1**, we used our algorithm to get an exemplary sequenceof 100 transactions which has errors that are positively biased and still passes basic statistical tests. This is summarized in Table 2. The table lists the total gain and the chi-squared values for 6 degrees of freedom which signify randomness for 100 transactions.

Total Gain (in currency units)	Chi-Square value
1.15	46.18

Table 2: Total gain and chi-squared value for example sequencefor TEM in Table 1

Figure 7 shows a TEM created for 2 decimal digit values approximated using IEEE 754 double precision. We show positive and negative errors in color-coded form where the positive errors are represented by dark cells and negative errors are represented by lighter cells and zero errors by white cells. The figure also shows a sequence with all positive errors and another sequence with all negative errors.

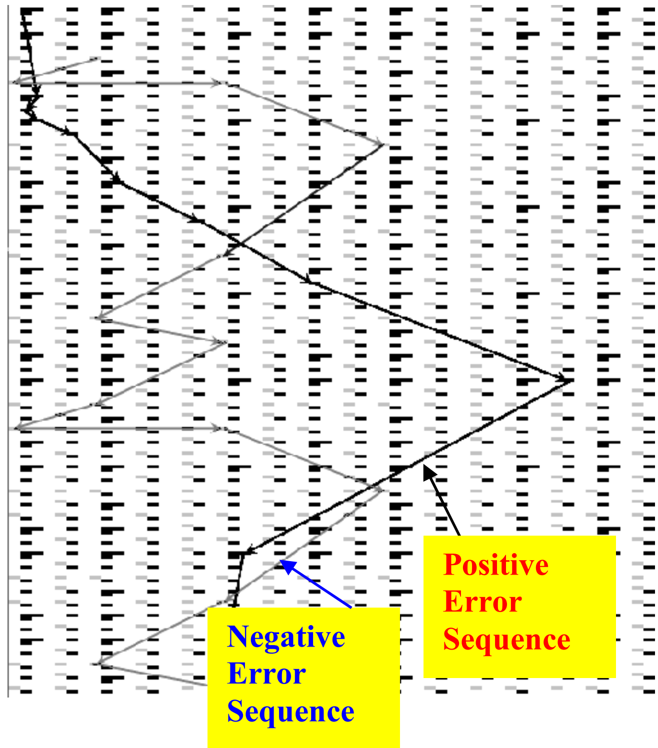

Figure 7: Sample error sequences in a TEM matrix

The graph in Figure 8 illustrates accumulation of error in 60,000transactions cycling over the 2 transaction sequences shown in Figure 7. These were obtained using our simulatorbased on our theoretical analysis.

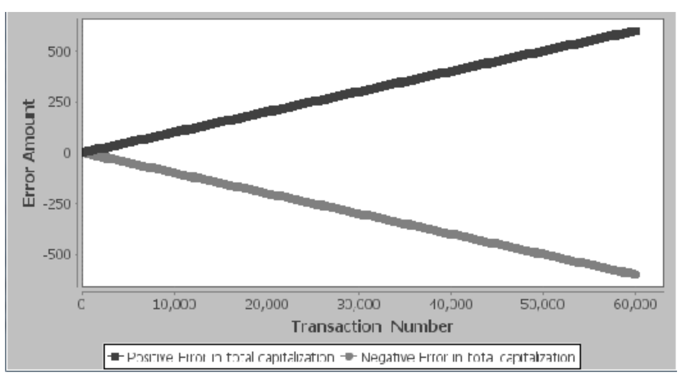

Figure 8: Error accumulation over large number of transactions

TEM enables us to design transaction volumes which do not always cause errors, but whose error process appears random and passes statistical tests but gains/loses money over a long enough time horizon.

Multi-currency transactions

While the single-currency examples used very large transaction amounts, this is not required when multi-currency transactions are used. When we have a transacting pair with different base currencies, it is still possible, with relatively small capitalizations, to find a sequence of transactions that will always give an error. This can also be done by finding a path through the TEM as explained earlier. Also, there may be multiple TEMs in this case, depending on the currency rounding rules.

IV. CONCLUSIONS

If financial software neglect the effects of IEEE-754 finite precision, then the results could be disastrous and huge monetary losses may occur.Our work quantifies these losses/gains using a simulator and a theory.Using the simulator, we have done statistical analysis of the error process. Using the theory, we showed that binary arithmetic can be exploited to create arbitrary error sequences, with considerable possibilities of financial arbitrage. These sequences can be chosen so as to pass many common statistical tests, and thus avoid detection. We note that good financial software should use exact decimal arithmetic andalso the importance of statistical tests in analyzing the errors.All these ideas will benefit applications in financial, retail and design sectors.

REFERENCES

[1]. David Goldberg, "What every computer scientist should know about floating-point arithmetic", Computing Surveys, 1991

[2]. Nicholas J. Higham, "The accuracy of floating point summation", SIAM J. Sci. Comput., July 1993

[3]. Michael F. Cowlishaw, "General decimal arithmetic", July 2008

[4]. Michael F. Cowlishaw, "Decimal floating-point: Algorism for computers", Proceedings of 16th IEEE Symposium on computer arithmetic, 2003

[5]. Michael F. Cowlishaw's page at IBM, "Decimal Arithmetic FAQ", http://speleotrove.com/decimal/decifaq.html

[6]. Paul M. Cohen, "Reflections on Early Computers", available at - http://www.paulcweb.com/reflect/Chap04.html

[7]. John von Neumann, "First Draft of a Report on the EDVAC", IEEE Annals of the History of Computing, Vol. 15, No. 4, 1993

[8]. W. Kahan, "IEEE Standard 754 for Binary Floating-Point Arithmetic", Lecture notes, University of California, Berkeley, 1996

[9]. William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery, "Numerical Recipes in C", Second edition, Cambridge university press, 1992

[10]. "The ASTRÉE Static Analyzer", available at - http://www.astree.ens.fr/

[11]. Donald Knuth, "The art of computer programming: Seminumerical Algorithms", Second edition, Addison-Wesley, 1981

[12]. W. Kahan, "Floating-Point Arithmetic Besieged by "Business Decisions"", Keynote address, ARITH17.