AD/AP-36

Accelerator Development Department
Accelerator Physics Division
BROOKHAVEN NATIONAL LABORATORY
Associated Universities, Inc.
Upton, NY 11973

Accelerator Physics Technical Note No. 36

Computer Precision Effects on Long Term Tracking

D. Maletić, A.G. Ruggiero

January 1992

Computer Precision Effects on Long Term Tracking

D. Maletić, A.G. Ruggiero

Accelerator Development Department
Brookhaven National Laboratory
January 1992

COMPUTER PRECISION EFFECTS ON LONG TERM TRACKING

D. Maletić, A.G. Ruggiero
Accelerator Development Department
Brookhaven National Laboratory
January 1992

1. Introduction

When tracking simulations of particles in accelerators are used, to determine the stability of the motion over very long periods of time, propagation of computer generated errors is of essential importance for the quality of the appropriate results. Present attitude toward solution of this problem is to use the highest computer accuracy at hand. It is common, in this field, to use accuracy of 28 and even more correct digits (ex. DOUBLE PRECISION option in the Cray computers).

But, [REF. 1]: "... we have obviously some philosophical difficulties to justify the need for such accuracy when we know that even in the ideal case of no external perturbation added we have eventually to face Heisenberg uncertainty principle... We do not believe particles would know their position and velocity so accurately as we were pretending during our computer simulations." In [REF. 2] we showed that for the case of the simple FODO beam line, with parameters similar to RHIC ones, quantum treatment of the particle motion lead us to transverse momentum quantum levels of the order of 10^{-11} of the total transverse momentum. That means that only 11 correct digits are physically allowed in such case.

Intention of this paper is to explore the difference in the results of a symplectic model, of analytically proven stability, when it is executed on the computer with different precision levels. Some hypothetical computer with infinite precision would produce definitely stable motion results for this model. Question is, what will happen to the results when only finite number of digits in the precision is possible? Also, how will the results change as one varies the number of correct digits?

2. Model

As for our accelerator model, we took an infinitely long transfer line. We considered a machine which is made only of quadrupole magnets and drifts (regions where particles moves free of any interaction). We assumed that our "accelerator" is infinite in length, straight and composed of FODO cells. This means that it is composed of series of focusing quadrupoles, drifts, defocusing quadrupoles and drifts, which repeats in this order to infinity. Length of the magnets is L and of the drifts is l. Magnets are defined by gradient G and magnetic rigidity $B\rho$. (Earlier mentioned quantum levels for the transverse momentum are obtained for the same model).

If we describe the motion of the particles in this system classically, we have no interaction in the drifts and the equations of motion in the quadrupoles [REF. 2]:

$$\ddot{x} = -Kv\dot{s}x \tag{1}$$

$$\ddot{y} = Kv\dot{s}y \tag{2}$$

$$K = \frac{G}{B\rho} \tag{3}$$

where v is total velocity of the particle and $\dot{s}$ is its component in s direction. Transverse coordinates are x and y.

With approximations common in today's accelerator physics, we approximate $\dot{s}$ with v and replace derivatives with respect to time $(\dot{x}, \dot{y})$ by derivatives with respect to the longitudinal coordinate s itself, (x', y'). After linearization and neglecting nonlinear kicks due to the quadrupole edge fields at the exit and the entrance of the quadrupoles, we can derive and use a matrix formalism, where appropriate transformation matrices for x and x' are:

$$M_1 = \begin{bmatrix} \cos\phi & K^{-1/2}\sin\phi \\ -K^{1/2}\sin\phi & \cos\phi \end{bmatrix} \tag{4}$$

$$M_3 = \begin{bmatrix} \cosh\phi & K^{-1/2}\sinh\phi \\ K^{1/2}\sinh\phi & \cosh\phi \end{bmatrix} \tag{5}$$

$$M_2 = M_4 = \begin{bmatrix} 1 & l \\ 0 & 1 \end{bmatrix} \tag{6}$$

where:

$$\phi = lK^{1/2} \tag{7}$$

Matricies M_2 and M_4 are transformations for the drifts, M_1 is for focusing quadrupoles and M_2 for defocusing quadrupoles. Given the initial conditions:

$$X_o = (x_o, x'_o) \tag{8}$$

simulation is performed by repeated matrix multiplications:

$$X_1 = M_4 (M_3 (M_2 (M_1 X_o))) \tag{9}$$

and by using output (X_1) values of one FODO cell as input (X_o) of next one. This multiplications are done by appropriate computer program, for some large number (N) of FODO cells and results are stored after each n FODO cells.

It is expected that some error propagation, due to the finite computing precision, will occur and eventually be big enough to affect stability of the simulated motion. In order to observe this effect we calculate Courant – Snyder invariant [REF. 3] for this motion.

During tracking X coordinates are evaluated at the beginning of the focusing quadrupole for each FODO cell. Now, at the end of the simulation, we are multiplying all results by a matrix for one half of the focusing quadrupole. This is done in order to make easier use of the Courant – Snyder invariant, which in that case is:

$$\frac{x^2}{\beta} + \beta x'^2 = C = \text{const.} \tag{10}$$

where the amplitude function β was found analytically for our model.

When we have results of the simulation for the points at the middle of the focusing quadrupole, Courant – Snyder invariant C is calculated at each of them and then plotted versus the number of the outputs, which represents the time scale. Analytical calculation implies that C should be a constant. However, computer error propagation may change that fact.

In classical mechanics we are allowed to use infinite precision. Now, question arises: is our analytical conclusion – that motion is stable, based on classical mechanics, really correct? What about mentioned quantum effects?

If instability exists for some finite precision and quantum physics does not allow us to use precision higher than that, we can doubt the results of the simulations which are done with higher precision in computing, even if they should conclude that motion is stable. Using classical model without any restrictions is not good enough in that case.

3. Numbers

Values for the parameters in the model described above are picked up in the manner to simulate approximate RHIC behavior, ie. they are close to appropriate parameters for the RHIC.

Next are used [REF. 4]:

Length of the drift: $l = 14\ m$

Length of the quadrupoles: $L = 1\ m$

Magnetic rigidity: $B\rho = 840\ Tm$

Magnetic field gradient: $G = 72\ T/m$

From this parameters we find:

$\beta = 48.50820\ m$

$\phi = 0.29277\ rad$

$K = 0.085714\ m^{-2}$

Initial coordinates X were always picked as: $x_o = 2\ 10^{-2}\ m$, $x'_o = 0$ rad.

At the middle of the focusing quadrupole, where Courant – Snyder invariant will be calculated from our results, its value is:

$C = 0.43455\ 10^{-4}\ m$ rad

The total number of FODO cells N in the simulation was defined by two factors. It needed to be large enough to approach the order of magnitude of 10^9 turns in the RHIC ring, which would happen in practical use of the RHIC when the beam is in it for about 10 hours. On the other side, a program cannot run forever,... One day of the "physical" time was the realistic limit for the running time of such program. We executed this program on the C Cray machine at LNL in Livermore and on Iris 4D computer at the Institute for Theoretical Physics at S.U.N.Y. On both machines limit for N at one day of the "physical" executing time lead to about 10^9 FODO cells in the model (with different CPU times, but with much higher access rate to the CPU at S.U.N.Y.). This limit corresponds only to

about five minutes of beam existence in the RHIC. However, this number of cells was found high enough to observe some effects of computer precision on calculated results.

Acceptable size of the data file led us to the limit of 5 outputs at each 10^6 FODO cells in the simulation.

4. Programs

Simple FORTRAN programs in single and double precision were made to perform the described simulation. Running on the Cray and Iris 4D computers, those programs gave us results for precisions of 32, 64 and 128 bites. On Iris 4D computer all double precision operations are done in 80 bits, which are then rounded to 64 and those in single precision in 40 bits rounded to 32 (IEEE standard). On that way, all digits of the results are significant: precision is equal to accuracy. On the Cray machines situation is similar. Calculations are done with full precision so that all digits of the result are significant.

If somebody wants precision in between of 32, 64 and 128 bites one must "create" it by himself in the program. This step is hard to overcome. If a program is made to carry all elementary operations and variables in any given precision, it becomes too slow for our purpose. For example, if the modeling is performed in program Mathematica [REF. 5] so that every variable and operation are in given precision, mentioned limit for one day of "physical" execution time reaches only 10^5 FODO cells in the model. As we will see later this is too small number for our goal. In other cases, one can achieve this step partially by allowing operations to be done in the highest precision; after each calculation rounding of the result to the desired precision can be done. This procedure was used and the results suggest that mixture of high precision operations and rounding may not be exactly what we are looking for. In this case there is no additional error spreading due to the low precision in arithmetic operations. On the other hand, there is additional error spreading due to the in perfect rounding procedure. However, some conclusions can be drawn even from results obtained on that way.

Rounding was done always in double precision, exploiting EQUIVALENCE command of FORTRAN language. By EQUIVALENCE (X,IX), where IX is array of dimension 2, every variable in a program is set in a position that we can "round-out" M bites from its

mantissa by performing:

$$IX(2) = \left(\left(IX(2) + 2^M\right)/2^M\right) * 2^M \tag{11}$$

For example, if M is equal to 16 – last 16 bites of X mantissa will be lost and last saved bite will be appropriately rounded. Used binary rounding causes increase in the each mean by "half bite" at the smallest weight place (slightly in excess). One should have this in mind when interpreting computing results obtained with mentioned rounding procedure.

Mentioned rounding is binary. Decimal rounding was also attempted, but it was too much time – consuming and involves much more of "computer interference" out of our control (in that case operations like logarithm must be performed).

The FORTRAN program we have used is very short and simple. For convenience of the reader we have listed it in the Appendix.

5. Results

Differences between results calculated with 32 and 64 bites precision were observed. They are very significant after about 10^7 FODO cells, (see Fig.1). With 32 bites precision, phase space ellipse diffuses in time and emittance grows (see Fig.2). As one can see on Fig.3 and Fig.4, Courant - Snyder invariant, which theoretically remains constant, grows with time. After 10^9 FODO cells results with 32 bites precision are so large that they are meaningless. Motion is unstable.

At the same time, points in the phase space obtained with 64 bites precision remain close to the initial phase space ellipse. There is no emittance growth and the Courant - Snyder invariant remains about constant even after 10^9 FODO cells; exactly as theoretical model predicts. One can observe small oscillations in the value of the Courant - Snyder invariant in this case. They are of the order of 10^{-5} to 10^{-6} of C value and remain confined. This difference shows how computing precision affects results in this case, compared with theoretical predictions. However, motion still looks stable.

In the case of 128 bites precision, results behave almost on the same way as for 64 bites precision. Small oscillations of C value still exists and is of the same order as in 64 bites case.

As for the program by which we explored precision effects in between 32 and 64 bites precision with binary rounding, other kinds of instability arose. As one can see on Fig.5, Fig.6 and Fig.7, mathematical instability appears in this case. Initial phase – space ellipse collapses (see Fig.6). If we approach 32 bites precision this way, emittance growth will not appear! Motion will be unstable, but with decrease of emittance. We cannot explain completely this result. Difference is certainly caused by the used rounding procedure which is not perfectly even. It works slightly in excess. Only conclusion which could be derived from the above group of results is that mathematical instability arises and is growing continuously as precision decreases.

As the given system can be analytically proven to be stable, the observed instabilities are certainly caused by repeated computer calculations at the given precision. Only those cases where both variables and arithmetics are treated at the same precision can describe real physical systems, with the equivalent determination limit of the variables (eg. due to the quantum mechanics). Results with "mixed" precision suggest that observed instabilities do not arise suddenly - stepwise, but gradually with the loss of the precision.

Earlier mentioned quantum levels of the order of 10^{-11} of the total transverse momentum in our model could cause instability of the motion over very long periods of time. In terms of the binary rounding program, this precision would be approximately defined by a loss of 16 bites (see Fig.5 and Fig.7). Some loss of the stability of the motion was observed in that case.

Finally, today's high precision computer modeling is still not "error – free". If instability is generated in our simple model over given time scale and precision, similar effect could be observed with higher precision and more complex models (more errors would arise due to the larger amount of calculations involved) when they are executed over longer time scales.

6. References

1. A.G.Ruggiero, Comparison Between Theoretical Predictions and Tracking, Particle Accelerators, 1986, Vol. 19, pp 157 – 179.
2. D.Maletić, A.Ruggiero, I.Ben-Zvi, Quantum Treatment of the Particle Motion in the Accelerator, in preparation, BNL (Fall 1991).
3. E.Courant, H.Snyder, Annals of Physics 3, 1-48, (1958).
4. Conceptual Design of the RHIC, BNL, Upton, (1989).
5. S.Wolfram, Mathematica, The Advanced Book Program, Addison Wesley, (1991).

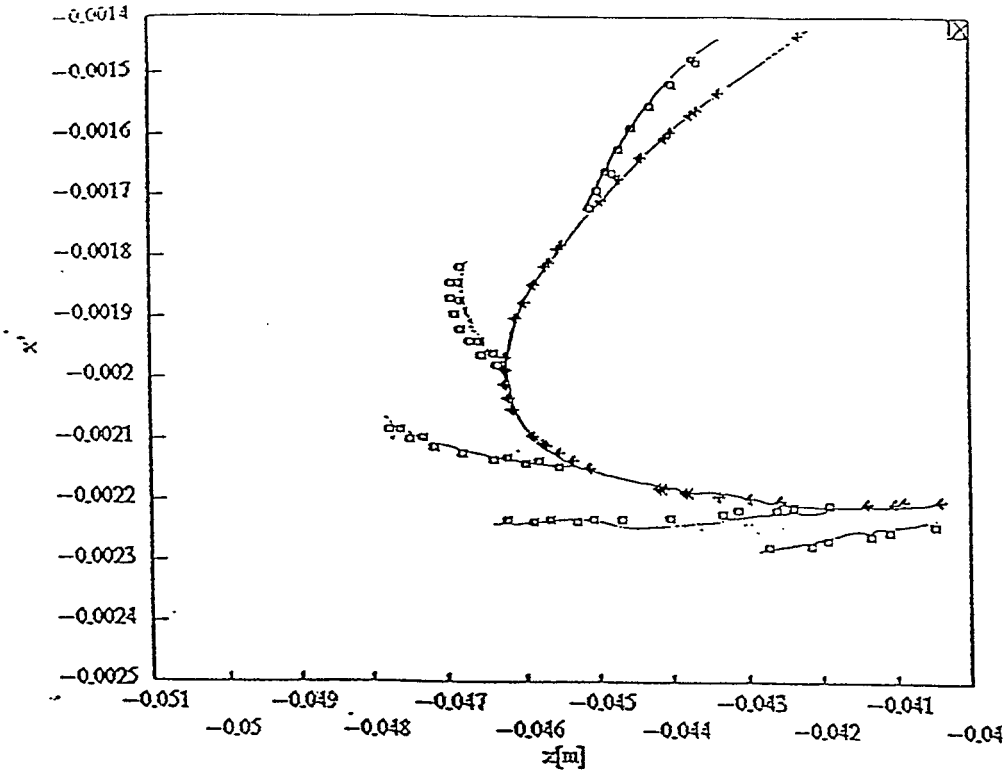

Figure 1: One corner of the phase-space ellipse, calculated for 10^7 FODO cells at the beginning of the focusing quadrupole. Crosses – 64 bites of precision (stable motion); Squares – 32 bites of precision (unstable motion).

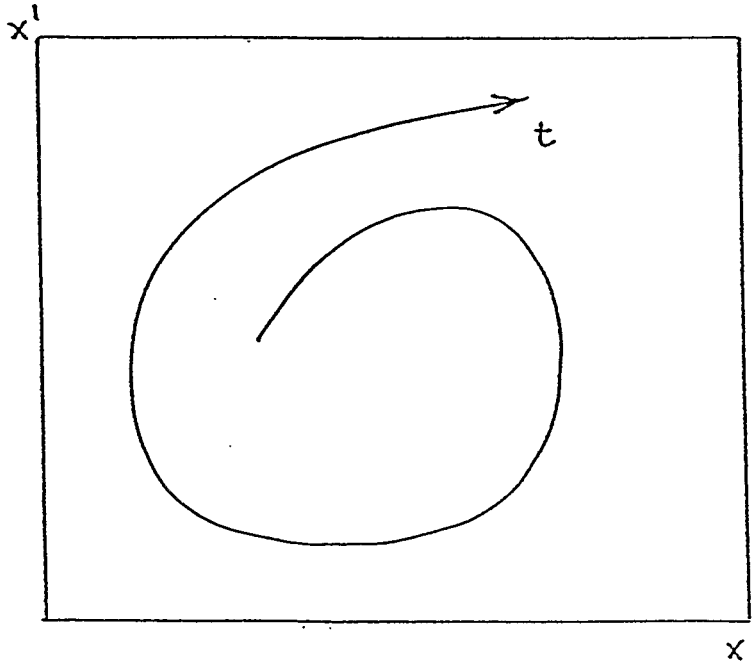

Figure 2: Character of the unstable motion when 32 bites precision was used.

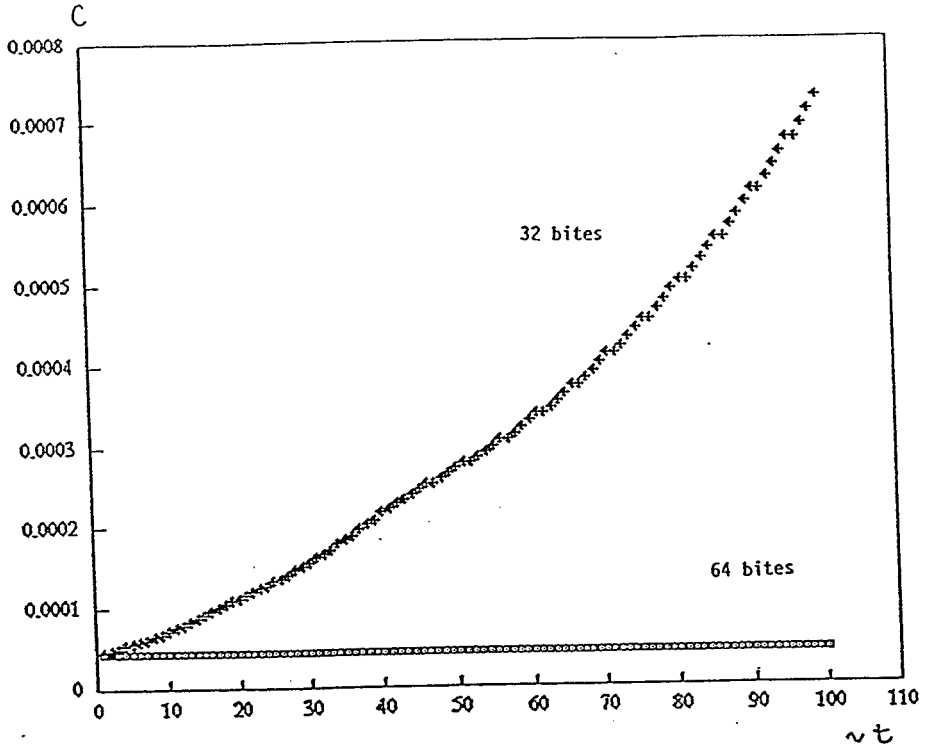

Figure 3: Courant – Snyder invariant vs. number of outputs (proportional to time) for 2 10^7 FODO cells.

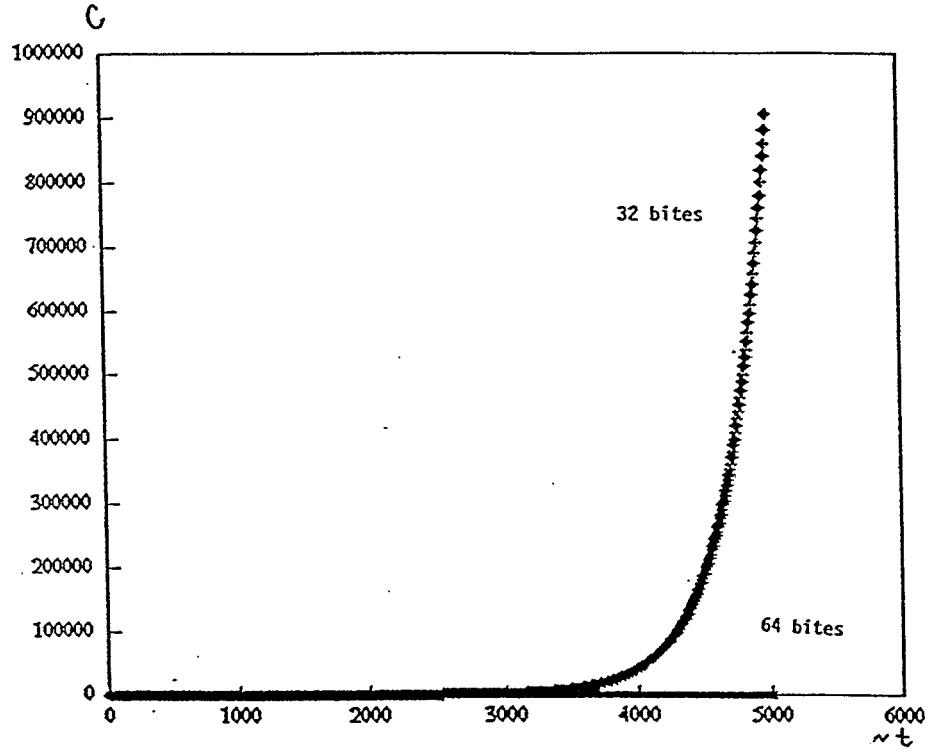

Figure 4: Courant – Snyder invariant vs. number of outputs (proportional to time) for 10^9 FODO cells.

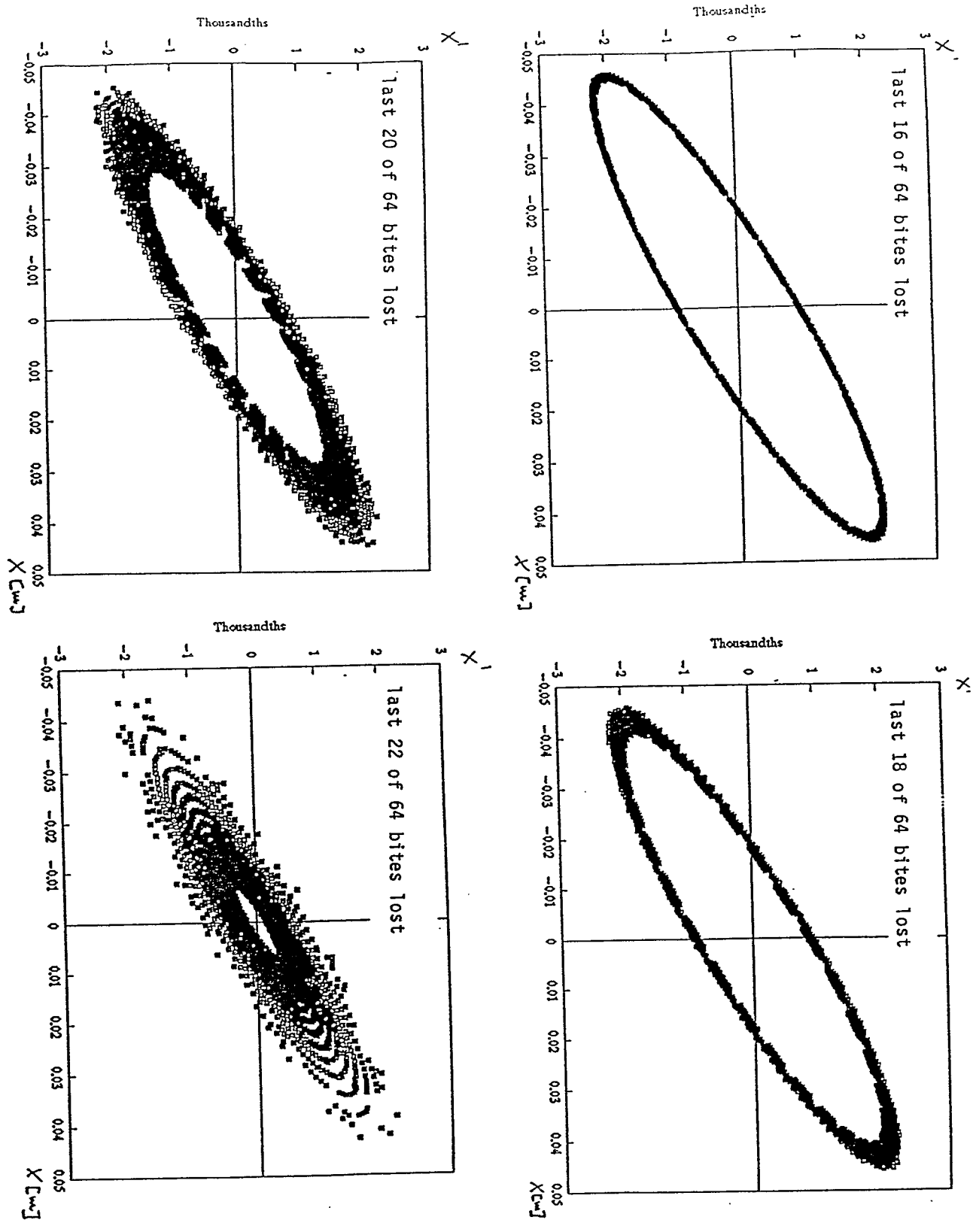

Figure 5: Phase space ellipse for the points at beginning of the quadrupole. Four different cases of results from binary rounding program. See Fig. 6 for the character of the unstable motion in this case.

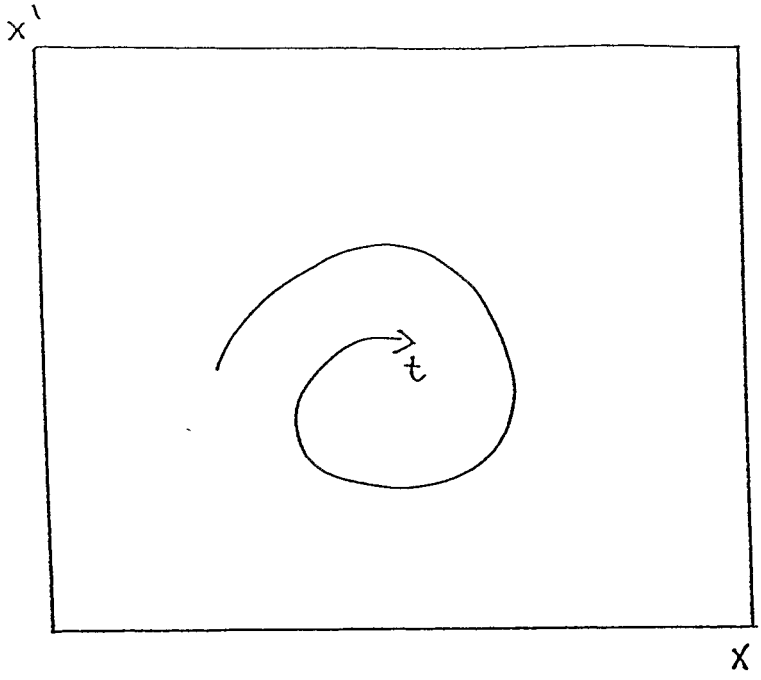

Figure 6: Character of the unstable motion calculated by the binary rounding program when some number of bites were lost in precision.

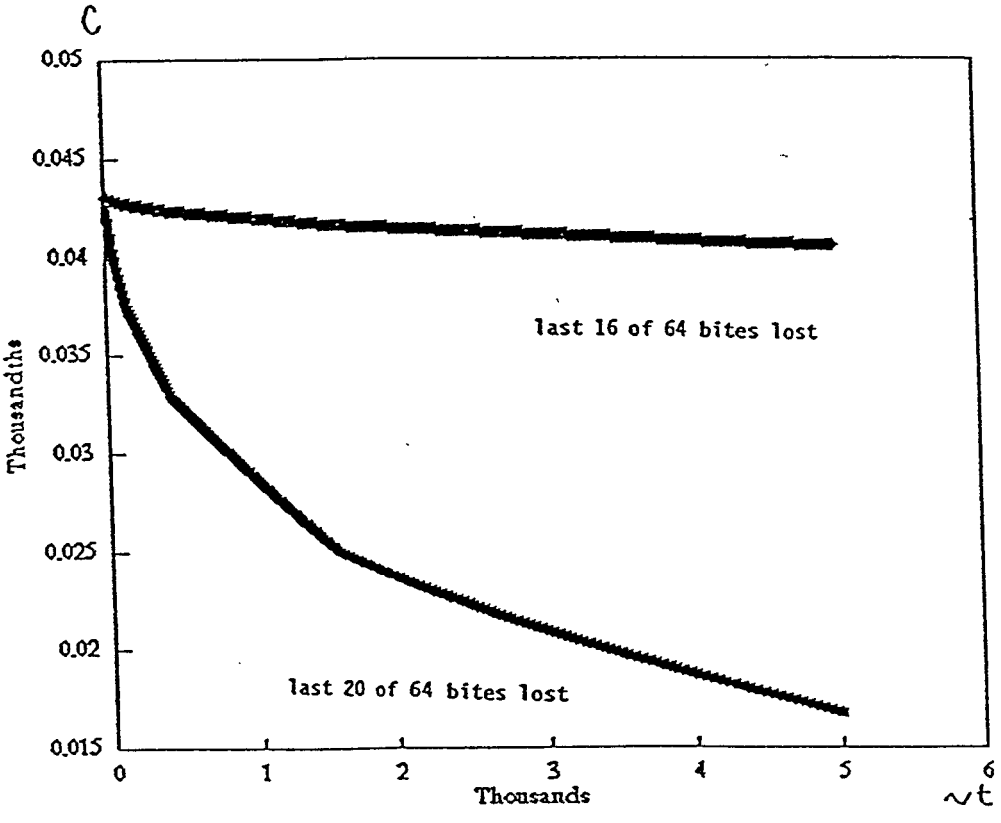

Figure 7: Courant – Snyder invariant vs. number of outputs (proportional to time) as result of the binary rounding program, for 10^9 FODO cells.

APPENDIX: PROGRAM USED TO PERFORME CALCULATIONS IN THIS PAPER

```
C COMPUTER PRECISION EFFECTS ON THE LONG TERM TRACKING
      DOUBLE PRECISION A,B,C,D,TETA,OSK,DL,X,XP,TX,TXP,POMOC,POMOC1
      DIMENSION A(4),B(4),C(4),D(4),IX(2),IXP(2)
      EQUIVALENCE (X,IX),(XP,IXP)
      OPEN(UNIT=7,FILE='CINP.DAT')
      OPEN(UNIT=8,FILE='CREZX.DAT')
      READ(7,*) TETA,OSK,DL
      READ(7,*) X,XP
      READ(7,*) MAX,NWORK,NPRINT
      READ(7,*) IBIT
      RNWORK=NWORK
      A(1)=DCOS(TETA)
      A(3)=DCOSH(TETA)
      A(2)=1.D0
      A(4)=1.D0
      B(1)=OSK*DSIN(TETA)
      B(3)=OSK*DSINH(TETA)
      B(2)=DL
      B(4)=DL
      C(1)=-DSIN(TETA)/OSK
      C(3)=DSINH(TETA)/OSK
      C(2)=0.D0
      C(4)=0.D0
      D(1)=A(1)
      D(3)=A(3)
      D(2)=1.D0
      D(4)=1.D0
      DO 1 I=1,MAX
      DO 2 J=1,4
      TX=X
      TXP=XP
      X=A(J)*TX+B(J)*TXP
      XP=C(J)*TX+D(J)*TXP
C
      IX(2)=((IX(2)+IBIT)/IBIT)*IBIT
      IXP(2)=((IXP(2)+IBIT)/IBIT)*IBIT
C
 2    CONTINUE
      RI=I
      POMOC=RI/RNWORK
      IPOMOC=I/NWORK
      POMOC=POMOC-IPOMOC
      POMOC1=POMOC*NWORK-NPRINT
      IF(POMOC) 1,4,5
 5    IF(POMOC1) 4,4,1
 4    WRITE(8,100) X,XP
 100  FORMAT( E22.16,' , ',E22.16)
 1    CONTINUE
      STOP
      END
```

Example of the input file CINP.DAT:

```
0.29277D0
3.4156503D0
14.0D0
2.D-2
0.0D0
1000000000
1000000
5
1048576
```

Note: Number used for the IBIT in this example

$$1048576 = 2^{20}$$

caused rounding of the last 20 bites.