Particle Accelerators, 1987, Vol. 22, pp. 231–248
Photocopying permitted by license only
© 1987 Gordon and Breach Science Publishers, Inc.
Printed in the United States of America

HARDWARE PROCESSOR FOR TRACKING PARTICLES IN AN ALTERNATING-GRADIENT SYNCHROTRON

M. JOHNSON and C. AVILEZ†
Fermi National Accelerator Laboratory, P.O. BOX 500, Batavia, IL 60510

(Received May 20, 1986; in final form March 4, 1987)

We discuss the design and performance of special-purpose processors for tracking particles through an alternating-gradient synchrotron. We present block diagram designs for two hardware processors. Both processors use algorithms based on the "kick" approximation, i.e., transport matrices are used for dipoles and quadrupoles, and the thin-lens approximation is used for all higher multipoles. The faster processor makes extensive use of memory look-up tables for evaluating functions. For the case of magnets with multipoles up to pole 30 and using one kick per magnet, this processor can track 19 particles through an accelerator at a rate that is only 220 times slower than the time it takes real particles to travel around the machine. For a model consisting of only thin lenses, it is only 150 times slower than real particles. An additional factor of 2 can be obtained with chips now becoming available. The number of magnets in the accelerator is limited only by the amount of memory available for storing magnet parameters.

1. INTRODUCTION

Particle storage rings provide a unique laboratory for studying both theoretically and experimentally the behavior of nonlinear systems for periods of 10^{11} cycles. Their development has created a need for techniques that can verify the stability of proposed designs for new machines.

The tracking method that we will discuss uses linear matrix transformations for dipoles and quadrupoles and transverse momentum kicks for all higher-order terms. This method is commonly called the kick or thin-lens method.

A subset where the thin-lens approximation is used for all elements can be obtained by nulling out the matrix operations. The thin-lens approximation is considerably easier to implement than the full matrix transformation. However, it is not clear to us that these approximations are valid for tracking a very large number of turns. Discussions of other types of tracking algorithms can be found in Refs. 1–3.

The main advantages of the kick method are that there is no restriction on the number of multipoles that can be included and the computer code is quite simple to write and to check. However, it does require a large number of calculations. For a magnet with multipoles up to pole 30 and one kick, an optimized serial algorithm has about 175 multiplications and additions per magnet. Thus, a 10,000-magnet accelerator requires 1,750,000 arithmetic operations per particle per turn.

† On sabbatical leave from Instituto de Fisica, UNAM Apdo. Postal 20–364, Mexico D.F.; Fellow of the John Simon Guggenheim Foundation.

A special processor designed specifically for orbit tracking can be much more powerful than a standard computer. Currently available chips can do 64-bit floating-point calculations at 100 ns per calculation (pipelined).[4] This will soon be decreased to less than 50 ns per multiplication.[5] The processors that we have designed can be completely pipelined so that the number of particles that are simultaneously computed is equal to the number of steps in the calculation. They can also take into account all the inherent parallelism in the calculation and thus get a further increase in speed. Our fastest processor can track 19 particles around the 10,000-magnet accelerator described above in approximately 32 ms (16 ms with new chips). This is for a thin-lens-only subset; a model which uses ideal dipoles and quadrupoles and a single thin lens per magnet for all higher multipoles would take about 50% longer. The same calculation on the CRAY I at 10 million floating-point operations per second would take approximately 2600 ms; we thus realize a speed gain of over 80. This assumes that some form of memory look-up is used for the square root in the CRAY and no synchrotron oscillations are calculated. If this is not true, the CRAY would take longer.

Orbits in a nonlinear system can appear stable for a large number of cycles and then make relatively sudden shifts in phase space. For this reason, we believe that it is more important to calculate a few particles (10 to 20) for a large number of turns, rather than many particles for a few turns. Of course, two processors can do twice as many particles as one, so there is no technical limit to the number of particles that can be tracked.

We also feel that the accelerator physicist must be intimately involved in the calculations. To achieve this, we propose to instrument the dedicated processor with the analog of accelerator instruments. These devices could do such things as plot the phase space or beam position at a point. They could also give a plot of beam position at every quadrupole around the accelerator or the tune of the machine or any other parameter that can be measured. In addition, there can be stimulus devices such as beam dampers or special rf cavities for multibunch coalescing. This approach can certainly be done with current tracking programs. However, these pseudo-instruments, combined with the speed of the processor, should allow someone to sit at a display and do experiments and see the results as if he were working on a somewhat slow-motion accelerator. Of course, we would provide a throttle so that one could move slowly through an interesting region and a reverse so that one could back up and proceed through an interesting region again—perhaps at slower speed.

Section 2 describes algorithms for tracking particles through magnets and drift spaces. Synchrotron oscillations are included. Section 3 discusses the affects of round-off error, and Section 4 describes a method of calculating complex functions using interpolation tables. Finally, Section 5 describes processors to implement the algorithms of Section 2.

2 ALGORITHMS

There are many excellent articles on tracking particles through nonlinear magnetic fields.[6–9] We will list the standard equations in forms suitable for hardware implementation. We will adopt the notation used by Schachinger and Talman for the program TEAPOT.[10]

The kick method calculates the effect of the nonlinear field components as a kick in transverse momentum at a single point. This kick changes the slope of the particles trajectory but not its position. There may, of course, be several kicks per magnet. The linear fields can be treated either by transport matrices or as part of a thin lens.

Synchrotron motion is treated as an instantaneous acceleration at the center of the accelerating gap. We implicitly assume only one accelerating gap per turn, but there could be more. The momentum variation is determined by computing an approximation to the difference in revolution time between the reference particle and the particle being tracked. This is then converted to a phase difference in the applied voltage at the accelerating gap.

2.1. *Linear Matrix Multiplication*

The matrix equation for a focusing quadrupole is[11]

$$\begin{vmatrix} x \\ \dfrac{V_x}{V_s} \end{vmatrix} = \begin{vmatrix} \cos KL & \dfrac{\sin KL}{K} \\ -K\sin KL & \cos KL \end{vmatrix} \begin{vmatrix} x_0 \\ \dfrac{V_{x0}}{V_{s0}} \end{vmatrix}, \tag{1}$$

where $K^2 = G/B\rho$, G is the field gradient, $B\rho$ is the magnetic rigidity of the particle, L is the length of the magnet, x is the horizontal position, and V_x/V_s is the slope of the particles trajectory. V_x is the velocity perpendicular to the magnet axis, and V_s is the velocity along the axis. The equation for a horizontally defocusing quadrupole is the same as Eq. (1), with the trigonometric functions replaced by hyperbolic functions and with no minus sign. K is a function of the particle's momentum, so it must be recomputed every turn if acceleration is present.

For dipoles we use the small-angle approximation for wedge dipoles.[11] The vertical coordinate y is represented by 2×2 matrices:

$$\begin{vmatrix} y \\ \dfrac{V_y}{V_s} \end{vmatrix} = \begin{vmatrix} 1 & L \\ 0 & 1 \end{vmatrix} \begin{vmatrix} y_0 \\ \dfrac{V_{y0}}{V_{s0}} \end{vmatrix}. \tag{2}$$

The horizontal coordinate requires a 3×3 matrix in order to incorporate momentum dispersion:

$$\begin{vmatrix} x \\ \dfrac{V_x}{V_s} \\ \dfrac{\delta P}{P} \end{vmatrix} = \begin{vmatrix} \cos\alpha & \rho\sin\alpha & \rho(1-\cos\alpha) \\ \dfrac{-\sin\alpha}{\rho} & \cos\alpha & \sin\alpha \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} x_0 \\ \dfrac{V_{x0}}{V_{s0}} \\ \dfrac{\delta P_0}{P_0} \end{vmatrix}, \tag{3}$$

where α is the bending angle of the dipole and ρ is the radius of curvature for the reference particle.

2.2 Nonlinear Filed Components

The equation for the thin-lens approximation is given by[10,12,13]

$$\frac{\delta V_x}{V} - i\frac{\delta V_y}{V} = \frac{cLB_0}{(1+\delta)p_0} \sum_{n=1}^{k} (b_n + ia_n)[(x-\delta x) - i(y-\delta y)]^n, \tag{4}$$

where δx and δy are the x and y offset of the magnetic center with respect to the reference particle, b_n and a_n are the normal and skew components of the multipole field, k is the number of multipoles, V is the velocity of the particle, p_0 is the momentum of the reference particle, L is the length of the magnet, B_0 is value of the principal field of the magnet (dipole or quadrupole), c is a constant, and δ is the momentum difference of the particle from the reference particle, such that $(1+\delta)p_0$ equals the particle's momentum. The real part of Eq. (4) is the kick in the x direction, and the imaginary part is the kick in the y direction.

Applying Horner's rule to the complex equations in Eq. (4) gives

$$\begin{aligned}\frac{\delta V_x}{V} - i\frac{\delta V_y}{V} &= \frac{cLB_0}{(1+\delta)p_0} \sum_{n=1}^{k} (b_n + ia_n)(x-iy)^n \\ &= \frac{cLB_0}{(1+\delta)p_0}\{\cdots[(b_k + ia_k)(x-iy) \\ &\quad + (b_{k-1} + ia_{k-1})](x-iy) + \cdots\} \\ &= \frac{cLB_0}{(1+\delta)p_0}\{\cdots[(b_k x + a_k y + b_{k-1}) \\ &\quad + i(a_k x - b_k y + a_{k-1})](x-iy) + \cdots\},\end{aligned} \tag{5}$$

where the δx and δy offsets have been set to zero for clarity.

A second algorithm can be obtained by writing Eq. (4) in polar coordinates. This algorithm involves evaluating trigonometric functions and taking square roots. Thus, on a serial computer, it would be much slower than the first algorithm. However, in a dedicated processor, these functions can be computed very quickly by storing an interpolation table for each function in a large memory and then performing a quadratic interpolation to get the value. In addition, each multipole term can be evaluated in parallel with all the other terms.

In polar coordinates,

$$\begin{aligned}\frac{\delta V_x}{V} &= \frac{cLB_0}{(1+\delta)p_0} \sum_{n=1}^{k} R^n(b_n \cos n\phi - a_n \sin n\phi), \\ \frac{\delta V_y}{V} &= \frac{cLB_0}{(1+\delta)p_0} \sum_{n=1}^{k} R^n(b_n \sin n\phi + a_n \cos n\phi),\end{aligned} \tag{6}$$

where $R = \sqrt{[(x-\delta x)^2 + (y-\delta y)^2]}$, $x = R\ \cos\phi$, and $y = R\sin\phi$. Thus, it is possible to achieve higher computational speeds than by using Horner's rule, but at a substantial cost in memory.

2.3. Coordinate Transformations

The above equations transport a particle through a magnetic element in the local coordinate system of that element. To go on to the next element, one must translate into the local coordinate system of that element. Thin lenses imbedded in the center of a dipole or quadrupole are treated as residing in the local coordinate system of that element. That is, there is no coordinate transformation until the end of the magnet is reached.

The coordinate system that we use is the same one used in TEAPOT. All the elements are described in the absolute Cartesian coordinate system (x, y, z). The y coordinate of all devices is zero and the plane of the reference orbit is also taken as $y = 0$. The local coordinate system is defined as y for vertical x along the radius from the nominal center of the accelerator, and s in the direction of particle motion, such that x, y, and s form a right-handed system.

If there is a drift distance between elements, the particle orbit must be tracked across it. The TEAPOT report describes this rotation and transport in great detail. We use the same method, but we have modified the equations so that they are easier for a dedicated processor to compute.

The ith magnet is located at point P_i. The quantities X_{i+}, S_{i+} are the coordinates of the next element (e.g., beginning of a dipole or location of a thin lens) in the current local coordinate system. The velocity components are v_x, v_y, and v_s. The rotation angle to the next element is ϕ_{i+}, and it is ngeative for clockwise rotations.

The equations for rotation and translation from the TEAPOT report are

$$x_{i+1} = \frac{1}{\cos \phi_{i+}} \frac{x_i - X_{i+} + v_x/v_s S_{i+}}{1 + v_x/v_s \tan \phi_{i+}}, \tag{7}$$

$$y_{i+1} = y_i + \frac{v_y}{v_s} \frac{S_{i+} - (x_i - X_{i+}) \tan \phi_{i+}}{1 + v_x/v_s \tan \phi_{i+}}, \tag{8}$$

$$s_{i+1} = 0, \tag{9}$$

$$v_{x(i+1)} = v_x \cos \phi_{i+} - v_s \sin \phi_{i+}, \tag{10}$$

$$v_{y(i+1)} = v_y, \tag{11}$$

$$v_{s(i+1)} = v_x \sin \phi_{i+} + v_s \cos \phi_{i+}. \tag{12}$$

One must compute v_s from the relation $v_s = \sqrt{(v^2 - v_t^2)}$, where $v_t^2 = v_x^2 + v_y^2$. We have identified 2 ways to calculate this quantity in a hardware processor. The first method relies on the fact that v_x and v_y are much less than v_s. Typical maximum ratios in the Tevatron are less than 0.001.[14] Thus, one can expand $1/v_s$ in a Taylor series to order 6. The error is then $\sim(v_t/v)^8$ or 10^{-24}. The expansion is

$$\frac{1}{v_s} = \frac{1}{\sqrt{v^2 - v_t^2}} \approx \frac{1}{v}\left(1 + \frac{1}{2}\frac{v_t^2}{v^2} \cdots\right). \tag{13}$$

The velocity of the particle, v, is constant between accelerating gaps, so $1/v$ can be calculated once at a gap and then used for all tracking in between.

A second method is to use Newton's interation method to compute $1/v_s$. Let A = an initial estimate of $1/v_s$ and $x = 1/v_s$. Then,

$$x_{i+1} = 0.5x_i(3 - Ax_i^2). \tag{14}$$

This algorithm involves no divisions, so it is easy to implement. If the initial guess is close, it converges quadratically, i.e., the error is squared with each iteration.[15] By using two memories, one to look up the exponent and the other to look up the mantissa, one can make a very good initial guess.[16] The exponent for IEEE standard floating point is 11 bits, so a 2K by 11 memory will uniquely determine the exponent. The mantissa approximation can be found to 15 bits by using a 64K by 16 memory. The error in this is then 2^{-16}, and it is reduced quadratically with each iteration. With two iterations, the error is less than the rounding error in IEEE double-precision floating point. This method has the advantage of working for all values of A. However, it takes somewhat longer than the Taylor expansion mentioned above.

ϕ_i is also a small quantity. In a machine such as the Tevatron, ϕ_i is of the order of $2\pi/1000 \simeq 6.28 \times 10^{-3}$. Therefore, we can expand the denominator of Eq. (7) as

$$\frac{1}{1 + \dfrac{v_x}{v_s}\tan\phi_i} \simeq 1 - \frac{v_x}{v_s}\tan\phi_i + 2\frac{v_x^2}{v_s^2}\tan^2\phi_i \ldots, \tag{15}$$

Since v_x/v_s is of order 10^{-3}, the error in dropping the fourth-order term is $\sim 10^{-20}$. $1/v_s$ was calculated above. Thus, all the terms in the above expressions can be calculated with multiplications and additions. Note that if these approximations are not adequate there is little computational penalty for including one additional term in Eq. (15).

2.4. Synchrotron Oscillations

Synchrotron oscillations depend on the difference in revolution time between the reference particle and the particle being tracked. One element in computing this time difference is the difference in path length.

The path length difference for a dipole is developed as follows. From Fig. 1, the angle ϕ is

$$\phi = \alpha + \theta_0 - \theta = \alpha + \Delta\theta, \tag{16}$$

where α is the bend angle of the reference particle, θ_0 is the incoming angle, and θ is the exit angle. The reference path length is $\rho\alpha$, where ρ is the radius of curvature for the reference particle. The particle path length for small θ is then $(\rho + \delta\rho)\phi$.

We have

$$\frac{\Delta\rho}{\rho} = \frac{\Delta p}{p} = \delta, \tag{17}$$

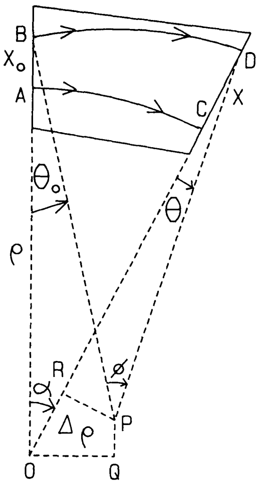


FIGURE 1 Diagram showing particle paths through a wedge dipole. AC is the path of the reference particle, and BD is the path of the particle of interest. x_0 is the input position and x is the output position.

where p is the particle's momentum and δ is the fractional difference in momentum. This is the same definition as in TEAPOT. Therefore,

$$\Delta l = \rho\alpha - (\rho - \Delta\rho)\phi = -(\rho\Delta\theta - \Delta\rho\alpha - \Delta\rho\Delta\theta) = -\rho(\Delta\theta - \delta(\alpha + \Delta\theta). \quad (18)$$

The path length difference for a quadrupole is approximated by a drift distance of equal length.

The formula for Δl for a drift distance is given in Refs. 17 and 18 as

$$\Delta l = \sqrt{\Delta x^2 + \Delta y^2 + \Delta s^2} - \sqrt{X_i^2 - S_i^2}. \quad (19)$$

Equation (19) involves taking the difference of two large numbers. Reference 17 describes a formula which avoids this problem. However, this formula is considerably harder to implement in a processor. Our approach is as follows.

We first factor out Δs^2 from the first term in Eq. (19). $(\Delta x^2 + \Delta y^2)/\Delta s^2$ is the square of the tangent of the angle that the particle makes with respect to the reference path. It is equal to v_t^2/v_s^2. $1/v_s$ has been calculated in the drift section [eq. (4)], so v_t^2/v_s^2 can be calculated with no divisions. It was also mentioned above that v_t/v_s is of order 0.001. Therefore, we expand this term in a Taylor series:

$$\Delta s\left[1 + \frac{1}{2}\frac{v_t^2}{v_s^2} - \frac{1}{8}\left(\frac{v_t^2}{v_s^2}\right)^2 \cdots\right] - \sqrt{X_i^2 + S_i^2}. \quad (20)$$

Next, we use an expression for Δs from Ref. 17:

$$\Delta s = s_{i+} = S_{i+} - \tan\phi_{i+}(x_{i+} - X_{i+}). \quad (21)$$

Substituting Eq. (21) into the leading term in Eq. (19) gives

$$\Delta s\left[\frac{1}{2}\frac{v_t^2}{v_s^2} - \frac{1}{8}\left(\frac{v_t^2}{v_s^2}\right)^2 + \cdots\right] - \tan\phi_{i+}(x_{i+} - X_{i+}) + (S_{i+} - \sqrt{S_{i+}^2 + X_{i+}^2}. \quad (22)$$

The last term is a constant for each element and can be computed to arbitrary precision before starting the tracking processor. Thus, the problem of subtracting large numbers has been removed without the need for divisions and square roots. The error in the above expansion is equal to the magnitude of the last term [not the first neglected term since a common term in the first part of Eq. (22) can be factored out]. Assuming a maximum ratio of $\Delta t/\Delta s = 0.01$, an expansion to 5 terms is all that is required. Expansion to 8 terms incurs little penalty in time but does require more hardware.

At the next accelerating gap, this length difference is converted to a time difference by the following relation:[18]

$$\Delta t = \frac{\Delta l}{v_i/c} + \Delta C\left(\frac{1}{v_i/c} - \frac{1}{v_0/c}\right), \tag{23}$$

where v_i is the velocity of the particle, v_0 is the velocity of the reference particle, and ΔC is the path length of the reference particle in this element.

The change in energy is given by converting the time difference to a phase difference in the accelerating voltage. δ and the quadrupole constant K are then computed for the next orbit.

3 ERROR ANALYSIS

Error analysis for such a large computational effort is very important. Each magnet requires about 175 multiplications and additions to track a particle through it. For a machine such as the Tevatron there are about 1.7×10^{12} calculations for 10 million turns. There have been some recent publications[19,20] which analyze the errors for large-scale tracking of particles using kick codes. They show that the error is proportional to n^2 where n is the number of turns. The primary source of error comes from the fact that the finite precision of the computer forces the quadrupole transport matrix to be non unitary. For the case where there is no acceleration, the same incorrect matrix multiplies the phase-space vector at every quadrupole. This causes the phase space to grow or shrink, depending on whether the determinant of the matrix is greater or less than unity. For the accelerating case, this should be less of a problem since at every turn the determinant is recalculated and should have an equal probability of being greater or less than unity. Reference 20 shows that going to 120-bit precision for the quadrupole matrix increases the precision of the calculation by several orders of magnitude.

Round-off errors from the nonlinear part of the calculation should go as the square root of the number of turns, so less precision is needed. Preliminary studies indicate that 32-bit floating point is not adequate.

4. INTERPOLATION

The rapid increase in size of semiconductor memories makes it practical to evaluate complex functions by interpolating a table stored in memory. A common example of this is evaluating log (x) by linearly interpolating a table of logarithms. This is done by fitting a straight line through a pair of adjacent values in the table and then using this line as an approximation to the function in this interval. Most logarithm tables contain only the endpoints of the intervals. However, the endpoints are not the points which minimize the maximum error on the interval. This is done exactly by finding the minmax polynomial on the interval or, to a very good approximation, by using the zeros of the Chebychev polynomial. The minmax polynomial of a given order is the polynomial which minimizes the maximum error on an interval. Thus, if one wanted the best straight line fit to the logarithm function between 0.1 and 0.2, one would not use the endpoints of the interval for the line fit, but rather determine the coefficients of the line which would minimize the maximum error. This would then be the minmax straight line.

Table I shows the maximum error for quadratic fits for four different functions, using Chebychev points as a function of the size of the interpolation table for a specified range. The maximum error was determined by finding the difference between the function and the quadratic fit in the specified interval. This difference has a maximum in the interval since the two functions are equal at the Chebechev points. The maximum of this equation was found using standard calculus and numerically solving the equations using 120-bit precision on a Cyber 875 computer. The results were checked by a Monte Carlo method which chose random points in random intervals and computed the difference between the fit and the function. Again, 120-bit precision was used. These results agreed quite well for the larger intervals (typically less than 2^{14}). For the small intervals, the Monte Carlo method gave errors that became constant, independent of the interval size. This did not occur for the analytical method. Figure 2 shows a plot of $\log_2$ (number of intervals) versus $\log_e$ (maximum error). A straight line fit to the data is also shown. Finding the true minmax polynomial changes the maximum error by less than 0.1%. Use of the Chebechev points instead of the endpoints of the interval reduced the maximum error by about 50% for all four functions.

TABLE I

Maximum error in the specified interval for four different functions, as a function of the number of bits in the look-up table. The results are for a quadratic fit to the Chebychev points

No. of bits	$\sin(15 \arctan \phi)$ $(0<\phi<1)$	e^x $(0.5<x<1)$	$x^{7.5}$ $(0<x<1)$	$\sin x$ $(0.5<x<1)$
14	$4.0\ 10^{-12}$	$3.2\ 10^{-15}$	$3.2\ 10^{-13}$	$1.0\ 10^{-15}$
15	$5.0\ 10^{-13}$	$4.0\ 10^{-16}$	$4.0\ 10^{-14}$	$1.3\ 10^{-16}$
16	$6.2\ 10^{-14}$	$5.0\ 10^{-17}$	$5.0\ 10^{-15}$	$1.6\ 10^{-17}$
17	$7.8\ 10^{-15}$	$6.3\ 10^{-18}$	$6.2\ 10^{-16}$	$2.0\ 10^{-18}$
18	$9.7\ 10^{-16}$	$7.9\ 10^{-19}$	$7.8\ 10^{-17}$	$2.5\ 10^{-19}$
19	$1.2\ 10^{-16}$	$9.8\ 10^{-20}$	$9.7\ 10^{-18}$	$3.2\ 10^{-20}$
20	$1.5\ 10^{-17}$	$1.2\ 10^{-20}$	$1.2\ 10^{-18}$	$4.0\ 10^{-21}$

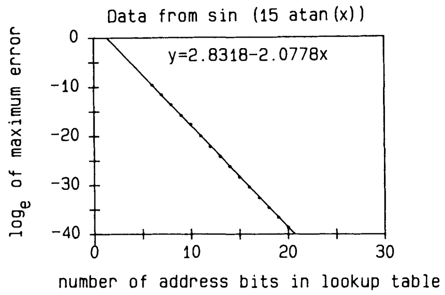


FIGURE 2 Plot of the number of address bits in versus the $\log_e$ of the maximum error from Table I. The equation is that of the best fit straight line.

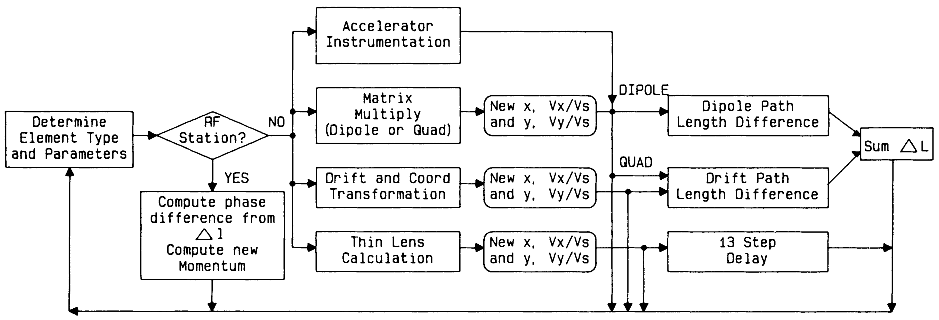
Interpolation can be used to compute more complicated functions. To calculate the contribution from pole 30, one must evaluate R^{15}, $\cos(15\phi)$, and $\sin(15\phi)$, where $R = \sqrt{(x^2 + y^2)}$ and $\phi = \arctan(y/x)$. Interpolation can be used to compute each of these five functions independently, but it can also be used to compute $(R^2)^{15/2}$, $\sin(15 \arctan(u)]$, and $\cos[15 \arctan(u)]$ directly, where $u = y/x$. R is the radius of the good-field region of the magnet, so it has a restricted range of values. For the purposes of error analysis, we have chosen $0 < \mathrm{R} < 1$. For the best accuracy, the argument of the arctan function should be less than unity. So, if $y > x$, we interchange x and y before dividing. This also removes any possible division by zero and allows the table to cover the entire range. Interchanging x and y changes the sine into the cosine and vice versa. Since both are being calculated anyway, we just set a flag identifying which term is the sine. Table I shows that these complex functions can be calculated to IEEE double precision with 2^{19}-bit tables.

5. PROCESSOR ARCHITECTURE

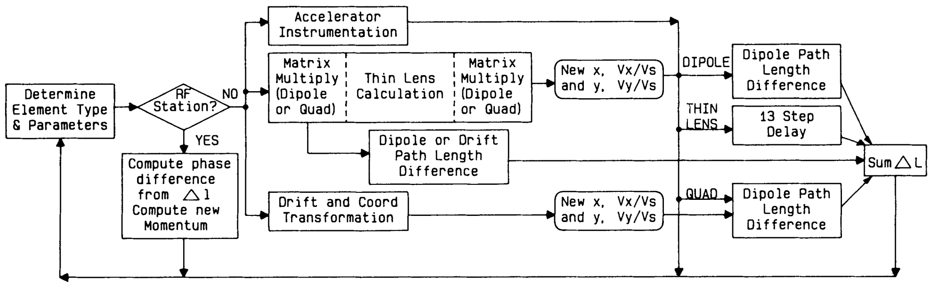
Figure 3a shows a block diagram of the processor. The processor calculates one element in the accelerator at a time. If a real dipole is represented as two ideal dipoles with a thin lens in the middle for the higher-order multipoles, it would have three elements.

The operation of the processor is quite straightforward. The element type is used to select one of four calculation routes. If the type is an rf station, the new momentum and such things as the quadrupole constant K are calculated. Since there is usually only one rf station in an accelerator, these calculations are not performed very frequently. Thus, we plan to use a fast conventional computer. If quadrupole matrices are used, this involves sine functions. We would use interpolation to compute these functions in order to keep the calculation fast.

The other three calculations are done with dedicated hardware and are described below. The matrix multiply section is used for both quadrupoles and dipoles. If one is using only thin lenses, this section is never used. The drift space calculation is used for all drift spaces.


Accelerator Instrumentation
Determine Element Type and Parameters
RF Station?
NO
YES
Matrix Multiply (Dipole or Quad)
Drift and Coord Transformation
Thin Lens Calculation
New x, Vx/Vs and y, Vy/Vs
Compute phase difference from △l Compute new Momentum
DIPOLE
QUAD
Dipole Path Length Difference
Drift Path Length Difference
13 Step Delay
Sum △L

(a)


Accelerator Instrumentation
Determine Element Type & Parameters
RF Station?
NO
YES
Matrix Multiply (Dipole or Quad)
Thin Lens Calculation
Matrix Multiply (Dipole or Quad)
Dipole or Drift Path Length Difference
Drift and Coord Transformation
New x, Vx/Vs and y, Vy/Vs
Compute phase difference from △l Compute new Momentum
DIPOLE
THIN LENS
QUAD
Dipole Path Length Difference
13 Step Delay
Sum△L

(b)

FIGURE 3 (a) Overall processor flow diagram, where the matrix operations are performed in parallel with the thin-lens ones. The accelerator instrumentation section is just a part on the processor for adding specialized functions. (b) Overall processor flow diagram, where matrix operations are appended to the beginning and end of a thin-lens calculation.

After the new coordinates are determined, the processor loops around to start the next element. At the same time, the new coordinates are passed to the path length difference calculator. Thus, the path length for element $n-1$ is being computed in parallel with the transport through element n. This doubles the speed of the calculations.

For the case where most of the magnets are represented as two ideal dipoles or quadrupoles with a thin lens in the center, the architecture shown in Fig. 3b is more than a factor of two faster. Here the matrix multiplies have been put at the beginning and end of the thin-lens calculation. This increases the time of the thin-lens calculation by 33% but reduces the number of elements to calculate from three to one. It does require a length difference calculation to be added in parallel with the thin-lens calculation to pick up the length difference from the first half of the magnet.

The matrix multiplication part for a 2×2 matrix is shown in Fig. 4. The calculations in vertical columns are done at the same clock step (and thus in parallel). It is a pipeline design so that at every clock cycle all elements are active. Each column is working on a different particle. That is, on a given clock cycle, column 1 computes $a_{11} * y_0 \cdots$ for particle 1. On the next cycle, it computes the same quantity for particle 2 while column 2 computes $a_{11}y_0 + a_{12} * vy_0$ for particle 1. When the pipeline is full, each hardware unit is engaged in processing a different particle. The total time to process one particle is the same as with one

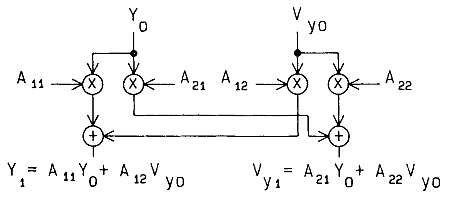
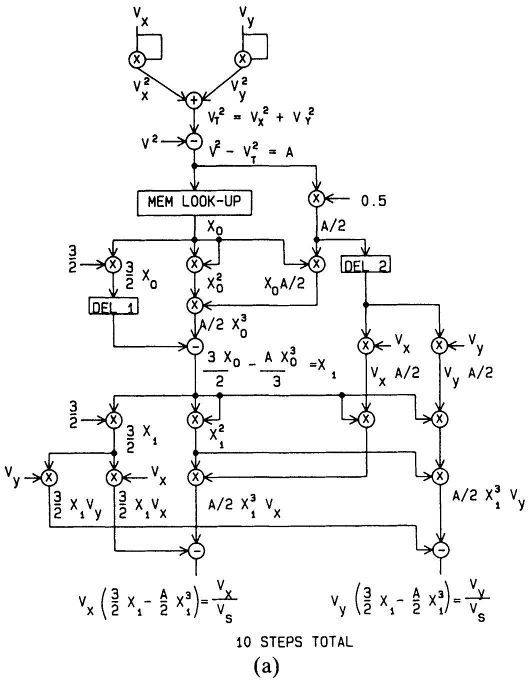

Y_0
V_{y0}
A_{11}
A_{21}
A_{12}
A_{22}
$Y_1 = A_{11}Y_0 + A_{12}V_{y0}$
$V_{y1} = A_{21}Y_0 + A_{22}V_{y0}$

FIGURE 4 Overall block diagram of the processor for performing the 2 × 2 matrix multiply to calculate y_1 and V_{y1}.

cell running in a loop. What is gained is the ability to compute several particles at no increase in time. This is an important advantage of a dedicated processor over a supercomputer. Multiple array processors or microprocessors can also compute several particles at once, but they do it by replicating processors and are limited by the speed at which they can compute one particle.

Figure 5a shows the evaluation of Eq. (14) for $1/v_s$. Two Newton–Raphson iterations are used. Figure 5b shows the evaluation of Eq. (15), and Fig. 5c shows the complete calculation of x_{i+1} and y_{i+1} [Eqs. (7) and (8)]. Sixteen steps are required. This means that 16 particles can be computed simultaneously with no loss in speed. It also means that each section except the rf in Fig. 3 must take at least 16 steps.



(a)

FIGURE 5 (a) Section of processor to evaluate Eq. (14). Delay blocks are used to synchronize the calculation. (b) Section of processor to evaluate Eq. (15). (c) Complete processor section to compute the coordinate transformation [Eq. (7) and (8)]. The numbers in the upper left index indicates the number of delay cycles referred for the evaluation of the sections shown in Figs. 5a and 5b.

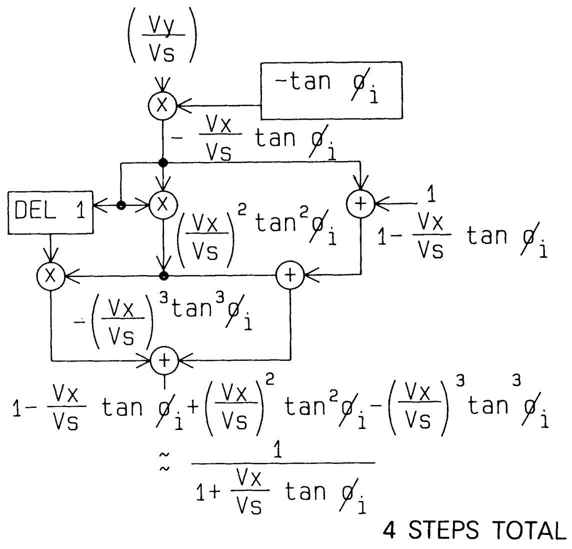
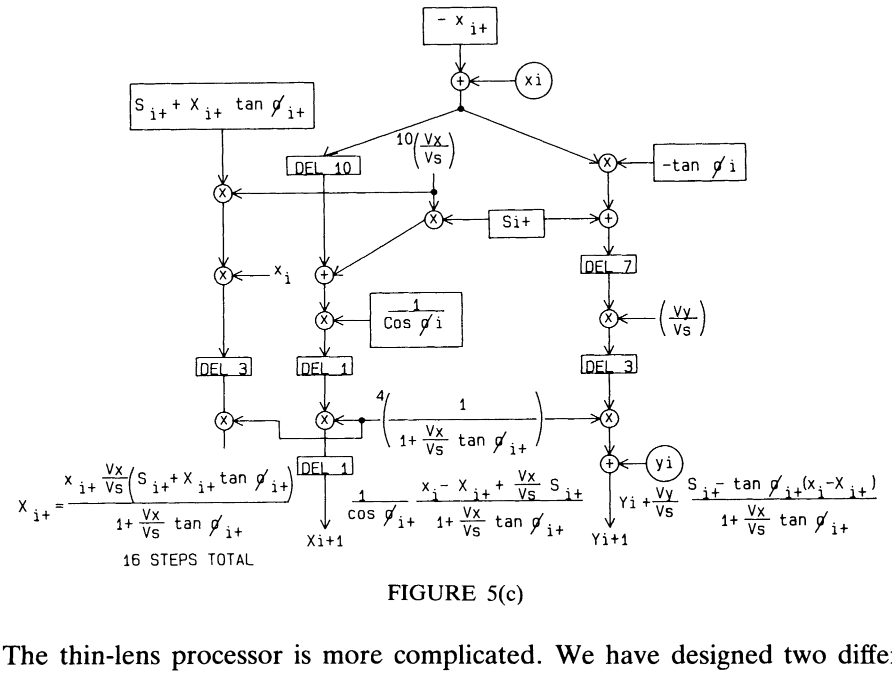


FIGURE 5(b)

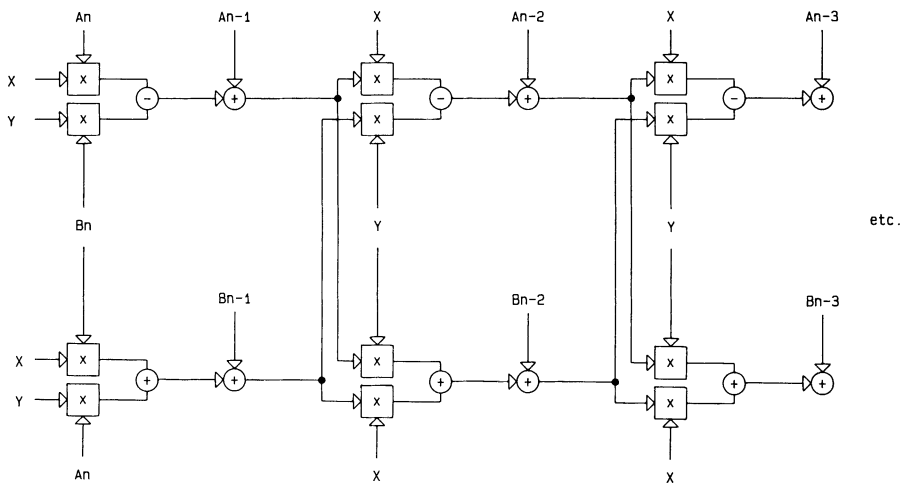


- X i+
xi
S i+ + X i+ tan ∅ i+
10(Vx/Vs)
DEL 10
-tan ∅i
Si+
xi
DEL 7
1/Cos ∅i
(Vy/Vs)
DEL 3
DEL 1
DEL 3
4(1 / (1+ Vx/Vs tan ∅ i+))
yi
DEL 1
X i+ = (x i+ Vx/Vs(S i+ + X i+ tan ∅ i+)) / (1+ Vx/Vs tan ∅ i+)
16 STEPS TOTAL
1/cos ∅ i+ · (xi - X i+ + Vx/Vs S i+) / (1+ Vx/Vs tan ∅ i+)
Xi+1
Yi + Vy/Vs · (S i+ - tan ∅ i+(xi - X i+)) / (1+ Vx/Vs tan ∅ i+)
Yi+1

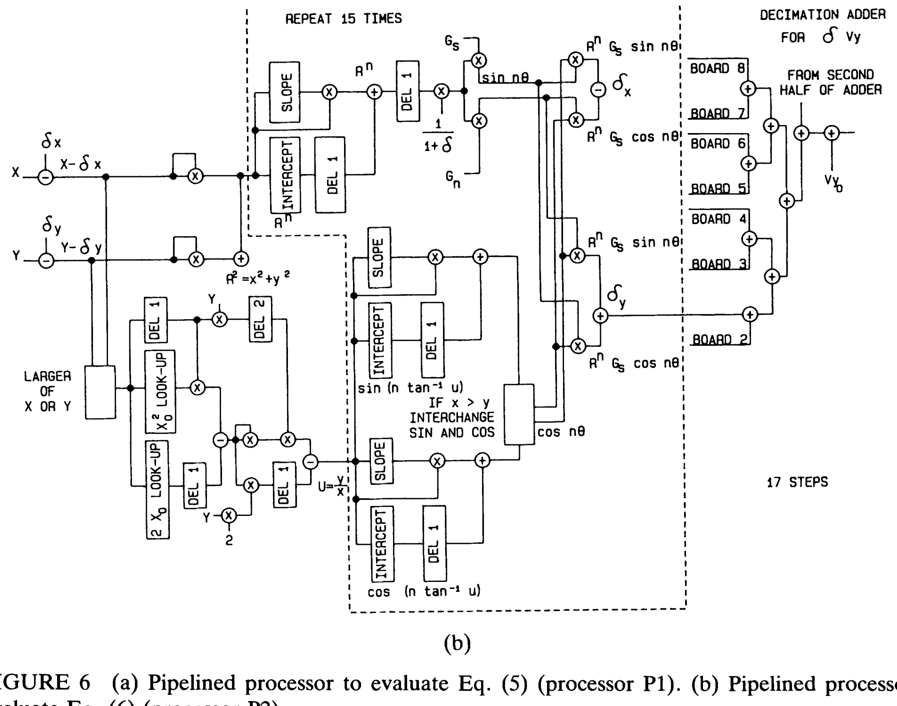
FIGURE 5(c)

The thin-lens processor is more complicated. We have designed two different processors (labeled P1 and P2) for calculating the nonlinear term. Processor P1 is a straightforward implementation of Horner's rule [Eq. (5)]. This is shown in Fig. 6a. The relationship between the number of multipoles and the number of steps needed to compute them is:

$$n_s = 3\left(\frac{n}{2} - 1\right) + 1, \tag{24}$$



(a)



(b)

FIGURE 6 (a) Pipelined processor to evaluate Eq. (5) (processor P1). (b) Pipelined processor to evaluate Eq. (6) (processor P2).

where n_s is the number of steps and n is the pole number ($n = 2$ for dipole, 4 for quadrupole, etc.). The constant 1 at the end is for adding the Δv to the input v. If we assume that 16 steps are available, this processor could compute only up to pole 12. Computing through pole 20 would require 28 steps.

Processor P2 is based on the expansion given by Eq. (9). It makes extensive use of interpolation, pipelining, and parallel processing. Figure 6b shows a block diagram of the kick section of the processor. All of the multipole terms are evaluated in parallel with each other and then summed by a decimation adder at the end. For simplicity only a linear interpolation is shown. An actual processor would require a quadratic interpolation which adds two steps to the pipeline. For each of the multipoles, the input x and y coordinates are converted to polar coordinates, and then Eq. (9) is calculated. The coordinate transformation is combined with calculating the sine, cosine, and r^n terms into one look-up table for each function, as was described in Section 4. The output from each multipole calculator is the δv_x and δv_y contribution from that multipole. These are then added together by a decimation adder to give the total δv_x and δv_y for this kick. These results are finally added to the input v_x and v_y to get the output values. For clarity only linear interpolation is shown; an actual processor would need quadratic interpolation to get the required accuracy. The number of steps needed to compute through a given multipole is given by

$$n_s = 15 + \log_2 (n/2). \tag{25}$$

The result from the log function must always be rounded up to the nearest integer. The processor shown in Fig. 6b has 14 steps in it. One employing quadratic interpolation would require 19 steps. Assuming floating-point calculation times of 100 ns, the total time would be about 7.4 μs per magnet (assuming an ideal magnet, thin lens, ideal magnet, and a drift space for each real magnet) or about 7 ms per turn for the Tevatron. Since there are 19 steps in the pipeline, 19 particles would be computed in this time.

Figure 7 shows the time difference calculation for synchrotron oscillations as given in Eq. (19).

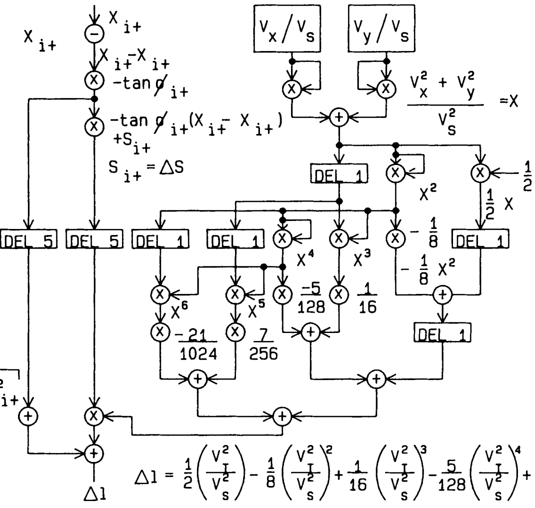


FIGURE 7 Calculation of Eq. (22) to evaluate synchrotron oscillations. Delays are shown as "del n," where n is the number of cycles to wait.

6. DISCUSSION AND REMARKS

Table I shows that the interpolation tables for sin [15 arctan (x)] require 18 or 19 bits in the memory address. One-megabit static memories are just now coming to market. These are typically organized as 128 K (17 address bits) by 8 bits. Quadratic interpolation requires three 64-bit constants per function. Twenty-four of the above chips are needed for a 17-address bit table and 96 for a 19-address bit table. The sin and cos tables together require 192 chips, and the $R^{7.5}$ table requires 48 chips for a total of 240 chips. This is required for each multipole.

Such a large number of chips is difficult to work with. Since the error in Table I is the maximum error in the entire interval, it may be possible to go to the next larger interval which will halve the number of memory chips. Another solution is to switch to cubic interpolation. This adds more adders and multipliers but only one more step in the pipeline. cubic interpolation requires only a 15-address bit table, so one can use 32 K by 8 memories. Cubic interpolation requires four constants, which means 32 chips per table or 96 memory chips for all three tables.

Each multipole also requires 23 arithmetic units (32 for cubic interpolation) and the memory for the multipole constants. The latter is quite small. Assuming 32,000 magnets in the accelerator and one real and one imaginary double-precision constant per multipole, only sixteen 32 K by 8–bit chips would be required. The above results are summarized in Table II.

TABLE II

Number of multipliers, adders, and memories required for the different processors; n is the number of multipoles. The quadratic interpolation tables are based on 18-address bit interpolation for sin [15 arctan (x)] and 17-address bit tables for $R^{7.5}$. The drift space and Δl calculations have been optimized for speed, assuming processor P2. If P1 is used, more time is available, so less hardware is required

	Number of Multipliers	Number of Adders	Number of Memories
Thin Lens Horner's rule	$4n+1$	$4n+3$	magnet const $16n$ 32 K $\times$ 8
Thin Lens Interpolation	$15n+8$	$9n+5$	magnet const $16n$ 32 K $\times$ 8 quad interp $120n$ 128 K $\times$ 8 cubic interp $96n$ 32 K $\times$ 8
Δl calculation	18	12	
Drift Space	34	13	
Matrix Multiplies 3 $\times$ 3	6	4	

One multipole from P2 takes about 200 chips, including bus interfacing, etc. This should easily fit on a single FASTBUS card. The drift space and Δl calculations should also each fit on a card. The acceleration system, system control, and accelerator instrument interface will also take about a card each, so a complete system will fit easily into a single crate.

P1 takes about 400 chips using $32\,\mathrm{K} \times 8$ memories. Switching to $128\,\mathrm{K} \times 8$ reduces the count to around 200, which again would fit on a single card. Thus, P2 takes about 15 times as much hardware as P1 for the kick calculation.

So which processor is best? It depends on the number of particles to be tracked. For multipoles up to 30 and 18 particles, P2 is more than twice as fast. It requires more hardware, but it is not at all impractical. On the other hand, tracking 1000 particles would require over 50 crates for P2 and only 5 crates for P1. This assumes a 40-particle pipeline and five processors per crate (note that P1 still requires acceleration, drift spaces, etc.). All of the other calculations, such as the Δl calculation, were optimized for speed under the assumption that they would be in parallel with P2. For P1 there would be more than twice as much time, so fewer operations would have to be done in parallel, and less hardware would be required.

Verifying the correct operation of these processors will be a difficult task. Initial commissioning can be done by checking the results against programs coded on a standard computer. For long runs, this will be impossible. One way of verifying the correct operation is to run the same problem on two different processors and to use specialized hardware to compare the output after every element calculation. This can be done by using the accelerator instrumentation port. The results from the last 20 or so elements would be stored in a circular buffer so that if a difference is found, the correct result can be determined by a standard computer. The part of the problem that caused the failure can be run repeatedly on the broken processor until the fault is found.

ACKNOWLEDGMENT

One of us, Clicerio Avilez, thanks the hospitality of Fermilab in general, and in particular, Marvin Johnson and the Accelerator Division Instrumentation Group.

REFERENCES

1. A. J. Dragt, *AIP Conf. Proc.* **87,** (1982), p. 147.
2. D. R. Douglas and A. J. Dragt, in *12th International Conf. on High Energy Accelerators,* F. T. Cole and R. Donaldson, Eds. (1983), p. 369.
3. D. R. Douglas and A. J. Dragt, *IEEE Trans. Nucl. Sci.* **NS-30,** 2442 (1983).
4. Documentation on WTL1064/1065, Weitek Corporation, 501 Mercury Drive, Sunnyvale, CA.
5. *Electronics,* Feb. 19, 1977, p. 88.
6. D. A. Edwards, *AIP Conf. Proc.* **127,** (1985), p. 1.
7. A. A. Kolomensky and A. N. Lebedev, *Theory of Cyclic Accelerators* (John Wiley and Sons, New York, 1966).
8. "Theoretical Aspects of the Behavior of Beams in Accelerators and Storage Rings," in Proc. First

Course of International School of Particle Accelerators of "Etore Majorana" Centre for Scientific Culture, M. H. Blewett, Ed., CERN 77-13 (1977).
9. E. D. Courant and H. S. Snyder, *Ann. Phys.* **3,** 1 (1958).
10. L. Schachinger and R. Talman, *TEAPOT: A Thin Element Accelerator Program for Optics and Tracking,* SSC Central Design Group report SSC-52 (1986).
11. S. Penner, *Rev. Sci. Instrum.* **32,** 150 (1961).
12. K. Halbach, *Nucl. Instrum. Methods* **78,** 185 (1970).
13. A. Asner and C. Iselin, "Some Analytical Methods for Winding Configurations of Ironless Beam Transport Magnets and Lenses," in *Proceedings of the 2nd International Conference on Magnet Technology,* (Oxford, 1967), p. 32.
14. R. Johnson, private communication.
15. F. Scheid, *Numerical Analysis,* (McGraw-Hill, New York, 1968), p. 316.
16. Application note on floating point division, Weitek Corp, 501 Mercury Drive, Sunnyvale, CA.
17. R. Hinkins, L. Schachinger, and R. Talman, *Synchrotron Oscillations in the SSC with TEAPOT,* SSC Central Design Group report SSC-N-217 (1986).
18. L. Schachinger, private communication.
19. P. Wilhelm and E. Lohrmann, *Particle Accelerators* **19,** 99 (1986).
20. M. Johnson and A. J. Slaughter, *Particle Accelerators,* **19,** 93 (1986).