

Interactive First Turn and Global  
Closed Orbit Correction in the SSC

Vern Paxson, Steve Peggs, and Lindsay Schachinger

*SSC Central Design Group*

*June 1, 1988*

ABSTRACT

We describe an algorithm for correcting the orbit of the SSC using closed bumps; an implementation of the algorithm using a highly graphical user-interface, designed with portability in mind; results from using the algorithm to correct lattices with simulated errors; and methods by which the algorithm can also be used to find erroneous monitors and correctors.

## Introduction

An early step in storing useful beam in the SSC is achieving a first-turn orbit. This is a non-trivial problem since the misalignments of the arc quadrupoles alone are such that the *rms* orbit deviation, without correction, would be 16 mm in the arcs, while the beam pipe is 16.5 mm in radius. The procedure is to first correct the trajectory so that the beam goes around the entire circumference of the ring, then to establish circulating beam, and finally to fine-tune the orbit with a global orbit correction algorithm. The task is complicated by the resolution limits of the beam position monitors (BPMs) and by displacement errors in the monitor positions.

We address four aspects of a program we have developed for simulating orbit correction: the algorithm used, the graphical interface to the program, the results of the simulation, and further applications of the algorithm.

Orbit correction is essentially a minimization problem, in which one computes the set of corrector strengths which will minimize the deviation of the actual trajectory from the desired trajectory, given other constraints. The algorithm described below breaks down the problem of global orbit correction into a series of overlapping local orbit correction problems. Each local problem involves minimizing a function of only one independent variable, which can be done in straight-forward fashion using a simplex method.

Software for exploring and manipulating models, such as studying the behavior of the orbit correction algorithm and its simulated effects on the SSC, benefits greatly from a highly graphical user-interface.<sup>[1]</sup> With a visual way to interact with the model the user can much more readily build intuition and understand how the model operates. Bearing this in mind, we created such a user-interface to the orbit correction algorithm, and used this as an opportunity to explore ways to manageably yet effectively represent a model as complex as the SSC. Furthermore, since graphical interfaces take a lot of time to write from scratch, we took pains to design the user-interface software so that it is reusable, both

by being modular to the extent that many components are generic and can thus be used in other interfaces, and by adhering to emerging standards so that the software can be readily moved to other hardware.

There are several motivations for implementing the orbit correction algorithm: to determine design requirements for the SSC's dipole correctors, to help with the design of corrector configurations for the interaction regions, to provide machines with corrected orbits for aperture studies, and to provide a more realistic machine model for studies of the correction of other errors.<sup>[2][3]</sup> The eventual goal is to realistically simulate commissioning and operation of the SSC with all known errors. We found that the algorithm is able to correct orbits in the presence of expected machine errors<sup>[4]</sup> to a degree satisfactory for further, higher-order correction, and below we summarize the corresponding design requirements.

The final issue we discuss is a method of using the orbit correction algorithm to detect faulty correctors and monitors. The discussion here is theoretical; the method has not yet been implemented in software and simulated.

## Algorithm

We use an orbit-correction algorithm which is based on closed bumps.<sup>[5][6]</sup> Bumps are made using three correctors as shown in Fig. 1. Each bump is parameterized by  $a$ , its height at the central corrector. To correct a region of the accelerator one finds the set of overlapping bump strengths which minimizes the difference between the actual orbit and the desired orbit, including constraints such as physical aperture size and maximum corrector strengths.

In general, the effect on the beam at a point  $s$  downstream of a kick  $\Delta x'_i$  at  $i$  is given by

$$\begin{aligned}\Delta x(s) &= \sqrt{\beta_i \beta_s} (\sin \Delta \phi) \Delta x'_i \\ \Delta x'(s) &= \sqrt{\beta_s / \beta_i} (\cos \Delta \phi - \alpha \sin \Delta \phi) \Delta x'_i\end{aligned}\tag{1}$$

where

$$\Delta\phi = \phi(s) - \phi(i)$$

A three corrector bump is *closed*, that is, the effects of the bump are wholly localized to the area between the first and third correctors, if

$$\sum_{i=1,3} \sqrt{\beta_i} \Delta x'_i \begin{pmatrix} \sin \Delta\phi \\ \cos \Delta\phi \end{pmatrix} = 0$$

for  $s$  outside of the bump. The solution to this vector equation, using the law of sines, is

$$\frac{\sqrt{\beta_i}}{\sin(\phi_j - \phi_k)} \Delta x'_i = \text{constant} \quad (2)$$

where  $i, j, k$  are (1, 2, 3), (2, 3, 1), or (3, 1, 2).

We now can use (1) and (2) to express the corrector strengths in terms of  $a$ , giving

$$\begin{aligned} \Delta x'_1 &= \frac{a}{\sqrt{\beta_1 \beta_2} \sin(\phi_2 - \phi_1)} \\ \Delta x'_2 &= \frac{a \sin(\phi_1 - \phi_3)}{\beta_2 \sin(\phi_3 - \phi_2) \sin(\phi_2 - \phi_1)} \\ \Delta x'_3 &= \frac{a}{\sqrt{\beta_2 \beta_3} \sin(\phi_3 - \phi_2)} \end{aligned} \quad (3)$$

Since (1) gives the additional displacement for a monitor inside a bump due to an additional kick at one of the correctors, and (3) expresses the kicks in terms of  $a$ , by combining them we can introduce a *penalty function*, with  $a$  as the sole independent variable, which quantifies how close the beam trajectory within a bump is to the ideal trajectory:

$$P(a) = \sum_{b=\text{BPM}} w_b (x_b - x_b^*)^2 + \sum_{c=\text{corr.}} w_c P_c(x'_c) \quad (4)$$

where  $w_b$  is a weight associated with each monitor,  $w_c$  a weight associated with

each corrector,  $x_b$  is the present monitor reading plus the change in position due to the bump,  $x_b^*$  is the desired reading at the monitor, and  $P_c$  is a function which penalizes excessive corrector settings.

Since  $P(a)$  is a function of only one independent variable, it is easily minimized using a one-dimensional simplex algorithm.<sup>[7]</sup> With the simplex algorithm comes the strong advantage that  $P(a)$  need not have a restricted form such as being quadratic, or even being particularly well-behaved (for example,  $P(a)$  can be discontinuous). In particular, the penalty function for corrector strength can be

$$P_c(x'_c) = \begin{cases} k|x'_c|, & \text{if } |x'_c| > x'_c{}^{\max} \\ 0, & \text{otherwise.} \end{cases}$$

for some large constant  $k$ . This function is quite cheap to compute, yet will prevent correctors from being set beyond their maximum strengths.

The region of the accelerator to be corrected is divided into overlapping closed bumps, and then the penalty function for each bump is minimized in turn, incorporating the effects of overlapping bumps in the  $x_b$  of (4). This iteration over the set of bumps is repeated until the change in the *global* penalty function,

$$G(a_1, \dots, a_n) = \sum_{i=1, n} P(a_i)$$

is negligible. At this point, since we minimize the  $P(a_i)$  separately, we have

$$\frac{\partial G(a_i)}{\partial a_i} = 0$$

for all  $i$ , so we have arrived at a local minimum of  $G$ .

The flexibility of this algorithm is appealing—it can correct an arbitrary region of the accelerator, both first-turn and closed orbit; since the solution generated is composed entirely of closed bumps, its effects are confined to the particular region of interest; the goal orbit can be any arbitrary trajectory; the

penalty function can have non-quadratic forms, incorporating effects such as maximum corrector strength and monitor readings; both monitors and correctors can be weighted to emphasize or remove particular elements; it is stable in regions which have no BPM's; and the algorithm is straightforward to implement in software.

## Graphics Interface

We implemented a highly graphical user-interface for the orbit correction algorithm, both to facilitate use of the algorithm and to explore ways of effectively displaying information about a machine as large as the SSC, which contains thousands of magnets.

The model we adopted was that of a *bird's eye view*, in which a top-view of the entire ring is shown in one window, and a selected, zoomed-in region of the ring is shown in greater detail in another window (Fig. 2). Navigation (zooming in or out, shifting the region of interest) can be done in either window, with both windows updating to show the effects. The zoomed-in view shows the horizontal and vertical beam positions at each monitor, including, optionally, the positions predicted by the orbit correction algorithm. Along the bottom appear blips indicating the position of individual correctors and monitors. These are selected using a mouse to produce information on a particular element.

The basic operation during establishment of the first turn is to zoom in on the region to be corrected, correct the orbit, reinject, and observe the resulting progress. After the first turn is established, the program can find and correct the closed orbit. Archives of any stage of the orbit correction process, and of statistics summarizing the distribution of corrector strengths that were necessary to correct the orbit, are readily stored. Although this level of functionality is not overwhelming, a key point is that further functionality (e.g., turning off bad monitors or correctors or diagnosing faults as discussed below) is easily added,

due to the modular design. The difficult work—the basic design of the graphics—is in place.

In implementing the orbit correction algorithm and its graphics interface, we followed the tenets of sound, portable program design.<sup>[8]</sup> The algorithm itself is written in standard FORTRAN-77, with the exception of `include` statements. Although the lattice information and machine functions are currently provided by Teapot,<sup>[9]</sup> the implementation of the algorithm is wholly independent of the modeling program — it has no knowledge of any Teapot common blocks, and is isolated from the format of Teapot machine files.

The graphical interface is written using the industry-standard X Window System,<sup>[10]</sup> which runs on a wide variety of computers (DEC, Apollo, Sun, Mass-comp, Apple, IBM PC, etc.; we used Sun workstations). We introduced a further level of graphics-independence by isolating the X-specific code in a package of library routines. The orbit correction user-interface is written solely in terms of these routines, and thus is insulated from changes in the underlying window system.

The only coupling between the graphics and the orbit correction algorithm is a set of interface routines which the graphics code calls to gather information to be displayed (such as element types, positions, and strengths) and to run the orbit correction algorithm. This design keeps the algorithm fully separate from the particular graphics interface used to interact with and animate it, and allows either to be changed without affecting the other.

## Simulation Results

The algorithm was used to correct the SSC 90° injection-optics lattice, which consists of two arcs and two clusters of four straight sections each. Five random number seeds were used to generate different sets of field strength and positional errors, with standard deviations as shown in Table 1. The distribution used was a Gaussian truncated at four standard deviations. The monitors in the simulation had a resolution  $\sigma$  of 100 microns, with errors in the readings not exceeding  $10\sigma$ . The beam was injected on-axis and with no energy errors.

|                   |                |         |
|-------------------|----------------|---------|
| Dipole errors     | $\sigma a_0$   | 5.9     |
|                   | $\sigma b_0$   | 8.5     |
|                   | $\sigma a_1$   | .72     |
|                   | $\sigma b_1$   | .72     |
|                   | $\sigma a_2$   | .64     |
|                   | $\sigma b_2$   | .40     |
|                   | $\sigma\theta$ | .6 mrad |
|                   | $\sigma x, y$  | 1 mm    |
| Quadrupole errors | $\sigma\theta$ | .5 mrad |
|                   | $\sigma x, y$  | .5 mm   |
| BPM misalignments | $\sigma x, y$  | .1 mm   |

Table 1. Errors. The multipole errors are in units of parts per  $10^{-4}$  at 1 cm of the dipole field.

The performance of the algorithm has been extensively studied in the arcs of the SSC, where the corrector and monitor configuration has already been established. The configuration in the straight sections needs further study. In the arcs, there is a biplanar monitor positioned 0.1 meters downstream of each quadrupole, and either a horizontal or a vertical corrector (adjacent to horizontal or verti-

cal focusing quadrupoles, respectively) 3.285 meters downstream of the monitor. Each monitor is aligned with its quadrupole, so the quadrupole misalignments given in Table 1 must be added to the monitor misalignments.

Without any correction, the injected beam was always quickly lost in each of the five versions of the lattice studied. The correction procedure was to steer the beam completely around the ring by progressively correcting the trajectory. This took on the average 3 or 4 applications of the algorithm for each arc and 5 for each cluster of four straight sections. Then the closed orbit was found by tracking. After global correction, the resultant closed orbit was found again, and these two steps repeated until the *rms* orbit error in each plane was less than .4 mm. This never required more than two subsequent iterations.

| Seed | $x_D$ | $y_D$ | $x_B$ | $y_B$ |
|------|-------|-------|-------|-------|
| 1    | .36   | .37   | .28   | .29   |
| 2    | .35   | .36   | .30   | .28   |
| 3    | .35   | .33   | .29   | .30   |
| 4    | .37   | .39   | .29   | .32   |
| 5    | .37   | .36   | .28   | .27   |

Table 2: RMS orbit deviations in mm.

$x_D$  is with respect to the design orbit, while  
 $x_B$  is with respect to the center of the BPM.

Table 2 shows the *rms* of the monitor readings in each plane of the final closed, corrected orbit. Two values are given for each plane, one with respect to the design orbit and one with respect to the center of the monitors. This latter reading is what the algorithm endeavors to minimize, since it has no knowledge of monitor errors. The values in the table include the monitor readings in the straight sections, which are not substantially different from those in the arcs.

Table 3 summarizes the *rms* and maximum corrector strengths needed in the arcs for the final corrected orbit. Slightly higher values were needed in the course of correcting the orbit. For the five seeds the greatest corrector strength ever needed was  $26.4 \mu\text{rad}$  in the X plane and  $26.2 \mu\text{rad}$  in the Y plane. The design maximum corrector strength is  $60 \mu\text{rad}$  at 20 TeV.

| Seed | $X_{\text{rms}}$ | $Y_{\text{rms}}$ | $X_{\text{max}}$ | $Y_{\text{max}}$ |
|------|------------------|------------------|------------------|------------------|
| 1    | 8.45             | 8.53             | 20.6             | 19.9             |
| 2    | 8.11             | 8.29             | 19.1             | 18.8             |
| 3    | 7.68             | 8.06             | 20.0             | 18.1             |
| 4    | 7.97             | 8.49             | 19.3             | 19.2             |
| 5    | 8.50             | 8.16             | 19.4             | 21.6             |

Table 3: Corrector strengths needed for final, corrected orbit, in  $\mu\text{rad}$ .

Peggs and Chao<sup>[11]</sup> have calculated the *rms* orbit deviations and corrector strengths for a  $60^\circ$  SSC lattice, for the linear machine with no coupling. A similar calculation for the  $90^\circ$  lattice with the errors used for this study yields an expected *rms* orbit deviation of 0.24 mm in both planes, with respect to the center of the beam position monitors, and expected *rms* horizontal and vertical corrector strengths of  $7.6\mu\text{rad}$ . Both predictions are in good agreement with the simulation results.

When using the algorithm to steer the beam through the first turn we discovered an aspect of its behavior which requires compensation. Since the algorithm uses a set of *closed* bumps, it will set the correctors in the final bump to strengths such that beyond the end of the bump the beam trajectory remains unchanged. During the first turn one corrects the trajectory all the way up to the point where it is lost, so the trajectory beyond the final bump is always very large. Consequently, the algorithm often sets the final corrector to a very high value in an attempt to preserve the closed nature of the bumps. The result was a well-

corrected trajectory up to the end of the final bump, followed by the original large trajectory, which was immediately lost. We solved this problem by simply not setting the final corrector when doing first turn correction. Sometimes the penultimate corrector will also be set to an excessive value. In this case, the beam will still progress forward, but not as far as it would if that corrector were off. Currently we ignore this deficiency as its only effect is to slightly decrease the rate at which the beam is successfully steered around the ring.

## Orbit Correction as an Aid in Fault-Diagnosis

The orbit correction algorithm described above can also be used to detect faulty correctors and monitors. This is done by comparing the measured effect of a set of corrector changes on the orbit with the predicted effect.

One detects faulty correctors by first measuring the orbit and computing the original corrector strength,  $\Delta x_i^{orig}$ , needed to correct the orbit, where  $i$  runs over the set of correctors within the region of interest. These changes are then applied to the correctors and the resultant orbit measured. The orbit correction algorithm is used again to compute the  $\Delta x_i^{new}$  which are now needed to correct the orbit. One then interprets the ratio between each new and original corrector change as follows:

$$\frac{\Delta x_i^{new}}{\Delta x_i^{orig}} \approx \begin{cases} 0, & \text{the corrector is OK;} \\ 1, & \text{the corrector is disconnected;} \\ 2, & \text{the corrector's polarity is reversed.} \end{cases}$$

One also considers the mean of the ratios. If it is not near zero then there is a calibration error in setting the corrector strengths.

After the faulty correctors have been properly taken into account, one can proceed to detecting faulty monitors. The first step is to compute the  $\Delta x_j^{predict}$ , the predicted change in monitor readings in the region of interest, due to a set  $\Delta x_i^{orig}$  of changes in corrector strengths. The correction is then applied and

the new orbit measured, yielding  $\Delta x_j^{actual}$ , the actual change in each monitor reading. The ratio between the actual change and the predicted change can be interpreted as follows:

$$\frac{\Delta x_j^{actual}}{\Delta x_j^{predict}} \approx \begin{cases} 1, & \text{the monitor is OK} \\ 0, & \text{the monitor is disconnected} \\ & \text{(unless both the predicted and actual change} \\ & \text{were quite small);} \\ -1, & \text{the monitor's polarity is reversed.} \end{cases}$$

If the mean of the ratios is not near one, it again indicates that there is a calibration error.

Note that because this method operates locally, it can be used for both first turn and closed orbit trajectories; also, there are similar techniques for finding sources of closed orbit errors or dispersion errors using localized convolution techniques.<sup>[12]</sup>

## Further Work

The orbit correction program is not a complete or even especially thorough treatment of the possibilities. In particular, it would benefit from: simulation of injection errors; the ability to deactivate or alter the weighting of faulty or suspect monitors and correctors; the ability to move and insert monitors and correctors for interactive exploration of optimal layouts (particularly useful for studying correction of the IR's); manual adjustment of correctors; additional display capabilities such as phase information and graphical display of corrector strengths; and an implementation of the fault-diagnosis algorithm described above.

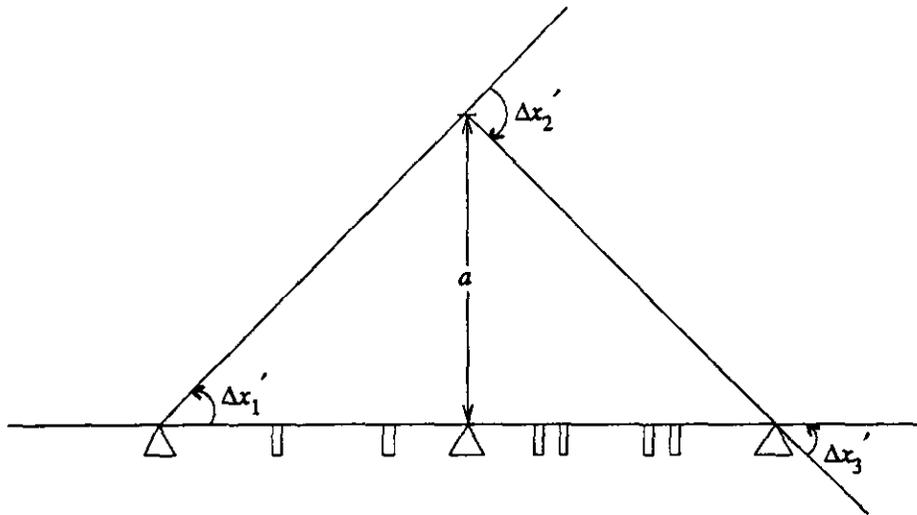
## Acknowledgments

We would like to thank Richard Talman for valuable discussions on orbit correction algorithms and user-interface needs, and Mark Eliot for inspiration and feedback during the development of prototypes of the graphics packages.

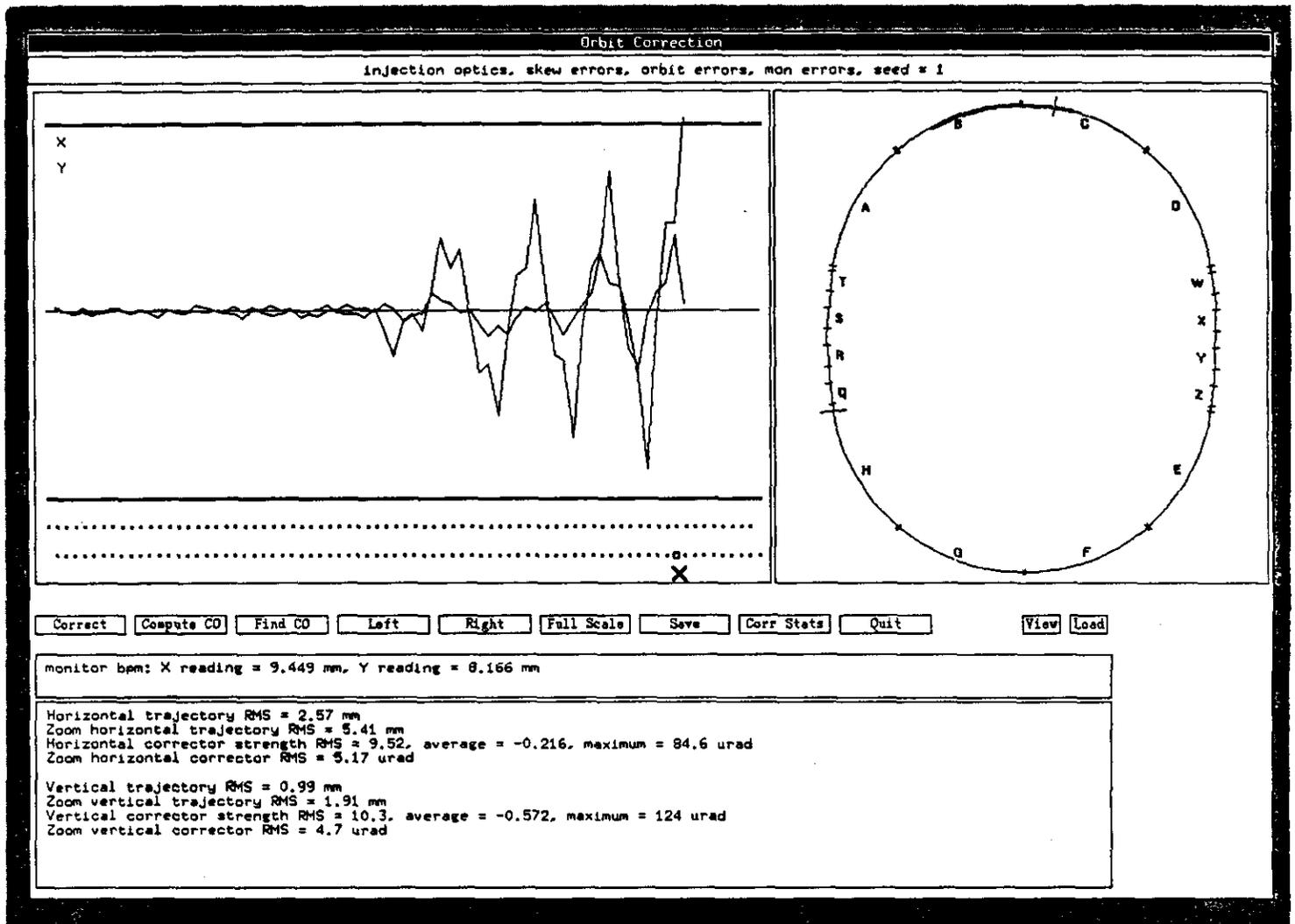
## REFERENCES

1. V. Paxson, V. Jacobson, E. Theil, M. Lee, and S. Clearwater, "A Scientific Workstation Operator Interface for Accelerator Control," 1987 IEEE PAC, p. 556, Washington, D. C.
2. L. Schachinger, "Interactive Global Decoupling of the SSC Injection Lattice," SSC-N-433, December 1987.
3. Lindsay Schachinger and Richard Talman, "Simulation of Chromaticity Control in the SSC," SSC-167, March 1988.
4. SSC Central Design Group, "Superconducting Super Collider Conceptual Design," SSC-SR-2020, March 1986.
5. S. G. Peggs, "Some Aspects of Machine Physics in the Cornell Electron Storage Ring," Ph.D. Thesis, Cornell University, 1981.
6. K. Steffen and J. Kewisch, "Effect of Magnet Distortion and Closed Orbit Correction in PETRA," DESY PET-77/01, 1977.
7. W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling, **Numerical Recipes**, Cambridge University Press, 1986.
8. V. Paxson and E. Theil, "Towards Portability in Model-Based Control Software," LBL-24723, November 1987.
9. L. Schachinger and R. Talman, "Teapot: A Thin-element Accelerator Program for Optics and Tracking," **Particle Accelerators**, Vol. 22, 1987.
10. R. Scheifler, J. Gettys, "The X Window System," **ACM Transactions on Graphics**, No. 63, 1986.

11. S. Peggs and A. Chao, "An Updated Look at Long Arc Orbit Correction," SSC-N-96, 1985.
12. S. G. Peggs, "Automatic Dipole Hunting," CESR CBN-84-11, 1984.



1. Structure of a closed bump. The triangles indicate correctors, the thin rectangles are monitors.



2. Graphical interface to the orbit correction algorithm. The bird's-eye view showing the entire ring is on the right; the window on the left shows the area currently zoomed-in on.