BNL-104893-2014-TECH

AGS/AD/Tech Note No. 487;BNL-104893-2014-IR

Particle Beam Control Design Notes for Neural Models

J. Niederer

June 1999

Collider Accelerator Department

Brookhaven National Laboratory

U.S. Department of Energy

USDOE Office of Science (SC)

DISCLAIMER

For Internal Distribution Only

Accelerator Division
Alternating Gradient Synchrotron Department
BROOKHAVEN NATIONAL LABORATORY
Upton, New York 11973

Accelerator Division
Technical Note

AGS/AD/Tech. Note No. 487

Particle Beam Control Design Notes for Neural Models

J. Niederer, BNL

June 1, 1999

Particle Beam Control

Design Notes for Neural Models

J. Niederer

June 1, 1992

Revised July 13, 1999

1. Summary

These notes discuss simulation schemes for charged particle beam control based on features of neural networks. These studies are carried out in the supporting software framework of the BNL MAD Program environment, which provides a companion model of a working accelerator. Conventional accelerator ideas of sensors, filters, controllers, and control elements readily map into corresponding neural elements. This mapping is particularly convenient when both the conventional and the neural elements are viewed as objects in the MAD object oriented representation of an accelerator or other large connected system.

2. Introduction

As accelerators evolve with ever more complicated beam requirements, beam position control logic can become a very complicated and demanding art. The conditions which disturb lattice apparatus may be changing locally, but effects on a beam are experienced throughout the machine. In circular machines, ordinary beam positioning control schemes are often based upon harmonic analysis of beam displacements. These schemes average perturbations over the whole machine, and can usually stabilize and otherwise improve orbits considerably. However, such global techniques may not always respond well locally to changes in beam transport conditions. For example, when adjusting for a particular intersection of two colliding beams, the tuning for the other intersections is usually disturbed. Similar problems occur in synchrotron light sources, where the radiations to many beam lines have to be maintained in unison.

Neural network technology offers many desirable properties for control systems in general. In nature neural processes are highly adaptive, relatively accurate, fast, and often capable of dealing with great complexity. Neural approaches have been increasingly successful in a variety of pattern recognition studies. Accelerator beam control can be viewed as a kind of pattern recognition problem, in which the patterns of orbit displacements are usually changing somewhat in time. The broader technical field of adaptive control employs features that often closely resemble neural network ones. Signal processing and television compression technologies increasingly apply neural network principles.

There are numerous observations and hints from biological systems which might be expected to influence pattern recognition technology. The neural fields include a vast variety of studies over many specialized topics. Studies range from analyses of perception in the sensory systems of various creatures to properties of individual neurons. In both living subjects and related analog and computer models, the role of connectivity, how a pattern is stored and retrieved, and whatever gross optimizing principles may be involved, are examined.

In a relatively simple perceptory system, such as smell, there are only a handful of kinds of neurons, there are many of them working together, and there are usually only a few steps in a cognition process. While each individual neuron may be noisy, a neural system as a whole is relatively noise immune. Each neuron has a series of features that may include threshold sensitivities, coincidence and correlation mechanisms, and signal limiting. Connected together, neurons combine to deliver meaningful, in phase response patterns. Some sensory systems seem to be clocked - sensing is initiated by a common control signal, which repeats at characteristic intervals. Response appears to happen in a series of steps, with the result of each step being reviewed by successive analyses steps. Responses of natural systems do not usually overshoot, at least not in mechanical ones. There is a neural priming mechanism, in that response is more pronounced as a stimulus is repeated. Neurons appear to be physically grouped in a series of thin layers, over plane areas, corresponding to logical stages. Some experiments suggest that information storage is related to patterns of activity in particular parts of these planes. In some experiments, the number of neurons responding to a given impulse appears to follow a power law, a property seen in self organized critical systems. The overall pattern stored in response to a kind of event itself may adapt in time.

When viewed in more detail, the early logical steps in brains occur in planes of related neurons, a few neurons thick, and a few millimeters in extent. Within a plane, neurons are closely coupled laterally, so incoming stimuli are shared among neighbors, even in the initial sensory stages. Neural signals are typically of the order of a millisecond duration, and within a subsystem, may repeat at uniform intervals when a stimulus is present. Connections radiate around neurons that accept incoming signals, with a radial fall off, a so called center surround layout. Some of these planar neurons react positively to signals on the plane, others produce negative signals. The inhibitory fields have a larger radius than the excitory ones, which leads to local groupings. Members of a local group are usually sensitive to some common feature of a stimulus. The sizes and extents of these groups are related in some optimal way that balances redundancy and the inherent variability among individual neurons. An incoming stimulus is spread over the plane, the neurons of the plane react as a whole, synchronize themselves, and within a step or two, some abstraction of the incoming stimulus is passed on to another stage, or further response is suppressed. Numerous experiments have observed these abstraction processes, but without really understanding them. Various response time constraints indicate that at most about twenty such stages can be involved in any given response, given numbers of a few milliseconds per stage. However the stimuli may be digested by the various stages, there is an evident progression downwards in layer size. For example, in the human eye, about 100,000,000 sensory and processing neurons are involved, in the next stage about 1,000,000, and about 100,000 in the visual cortex. With very imperfect parts, all of this works far better than any of the artificial networks using perfect parts designed to date.

However disappointing our understanding of how the brain may work, imitating even the grossest of these layout features may well lead to better performing technology. In particular, animals make extensive use of feedback systems, with known components and characteristics. While we may not be in the business of modeling cats, when a cat's feedback systems work considerably better than some of our accelerator ones, surely there is something to be learned. It does turn out that this relatively small collection of neural features can be easily represented in computer models, and applied to feedback logic in particular. Experiments can be performed with the models to see which features lead to improvements. In one form or other, some of these features are familiar in accelerator and other technology. To avoid destructive overshoots, power supplies are usually programmed to move to a setting in a series of small steps. The effect of each step is observed before proceeding. Gating techniques and signal threshold criteria are applied to reduce exposure to noise.

The Stanford Linear Collider (SLC) has a number of beam steering systems, of which the one at the downstream end of the linac is a suitable example of the components and connections involved. The early form of this feedback loop went through a rather painful development and debugging stage under severe pressure. Given the luxury of reviewing these SLC approaches afterwards, it appears that they can quite easily be recast into a much simpler neural network point of view. Basic SLC software features can be expressed in more general modular forms resembling neural analogues. Moreover, reasons for very low gain and relative instabilities are now easier to recognize when the rather complicated signal couplings are separated from the original multiply indexed matrix software nightmare. The SLC feedback experience accordingly has been generalized to more elaborate, interacting feedback systems, as an application of a self contained neural module in the BNL version of the MAD Program. Perhaps some of these techniques might also be of use for studying the more demanding beam steering expectations of the newer linear collider proposals as well.

In the jargon of the software for the model explored here, each of the kinds of neuron is pictured as a class of object in an object oriented data base. The connections to a neuron are simply noted by a list of signal sources, such as the output cells of other neurons, together with such weights as may be wanted. At each time step; a neuron combines its source signals according to some rules, and produces a corresponding signal of its own. This output is typically delivered to a signal location in the neuron's object data structure. When viewed as data objects, neurons have rather similar attributes, and differ among themselves mainly in the rules by which responses to input signals are formed. Natural nets seem to work effectively using self clocked time steps, and signals clamped in amplitude. Computer based experiments can be designed to try these very simple kinds of non-linear signal couplings. Additionally, the continuous amplitudes normally seen in conventional fitting and feedback techniques can be included for comparison.

3. Overview of the Model

We have built a general software supporting framework to explore such ideas by means of computer simulations. Stimuli (signals) from measuring devices are passed among a number of interconnected, elementary processing modules, which produce output signals that respond to these measurements. The elementary modules, the neurons, are described by information that includes input signal connections and weights, and keys to the possible ways that the inputs can be combined and processed along with necessary parameters. Additionally there are instructions as to how the output signals are to be formed. Supporting utilities of the framework gather the descriptive information, and make it available in a data base matched to the problem. Configuration services connect the signal paths declared in the input information. Another service operates a beam transit model, and collects measurements as needed. Ideally this collection of supporting services is flexible enough so that the effects of various ideas, features, and operating parameters can be studied.

The model itself is operated by a supervisor which advances a clock, and cycles a series of events, each of which may include communication and execution delays. A sequence of events typically includes:

1. Run the beam(s).
2. Collect orbit data.
3. Transmit orbit data to a feedback controller.
4. Run controller, which operates layers of sensor, filter, adder, and motor neurons in a layered network.
5. Transmit controller responses as signals to correctors.
6. Set correctors.

Repeat.

In the operations mode, a feedback controller organizes the flow of input signals from the measuring elements, through the analysis neurons of each layer, and into the corresponding correction elements. In the training or calibration mode, depending upon the kind of neural model in use, signal weights may be adjusted among the neurons of the layers so that the system responds properly to a known pattern of displacements. Neurons belonging to the same logical stage comprise a layer, which is described in the program by a simple list of the neurons involved. The collection of layers comprises a network. Likewise a simple list of the names of the layers describes the network of a controller. Finally, a number of such feedback loops and their controllers may be operated together.

Details of connectivity are free parameters of this system, so connection schemes that range from the conventional controls theory calibration matrices to the more remarkable biological examples may be considered. A few basic kinds of neural model have been implemented in the rules of neurons to date. The formal design is intended to accept other kinds readily without disturbing the options already in place. Features of the major classes of artificial neural networks can also be accommodated, often by minor changes in the input descriptions of neurons and the network. Noise can be introduced onto any signals or processes, to test performance and stability. Foregoing for the present this wealth of interesting possibilities, a relatively simple controller scheme of a few layers, each populated with a single kind of neuron, will be developed in these notes.

In some of the simpler configurations, such as N correctors being driven by measurements from M monitors, this neural representation at first sight may seem to be just another way of describing a feedback system already well known to accelerator physics. However, the possibilities for non linear couplings among the signal elements leads to a far more powerful and stable controller. Controllers are expected to perform in an environment where both beams and measurements are noisy, and responses are subject to all the inherent instabilities of the system. The generally non linear, dynamic neural approaches have direct ways of dealing with system noise, which are not present in the traditional Micado and similar static orbit adjustment techniques. Even small configurations show remarkable damping behavior. They are also wonderfully flexible by comparison. For example, the familiar Micado algorithm is easily adapted to our quite general ensemble of sensors, filters, and correctors with the various non linear couplings.

4. Kinds of Neurons

This section lists the elementary kinds (classes) of neuron implemented in this network based controls model. These neurons are the building blocks from which controller and feedback networks are configured. Basic neurons can be configured for both circular and linear lattices.

4.1. Sensory Neurons

A sensory neuron delivers a reading on demand, typically directed by a command (gate signal) from some supervisor or central clocking source. Readings, presumably originating in attached hardware (or beam simulators), are expected to be subject to noise. All neurons have rules for forming and shaping output levels or pulses. As represented in a computer, a sensory neuron data structure includes latch cells from which its outputs may be taken. In this controls model, sensory attributes include a reference to the primary source of a signal, a device such as a beam monitor, in a tracking program. Other attributes define rules to verify the input signal, to refine it, and to form the output.

Sensory neurons presumably are noisy, at some level. Various noise patterns can be supplied by a noise generating service. Many kinds of real sensory neurons may deliver a chain of output pulses, a kind of quantized amplitude to pulse train duration conversion, which has virtues in an extremely noisy environment. The use of pulse trains is rather extreme for our purposes, however.

Each of our neurons processes input signals, and delivers a resulting signal to an output stage. The form given to output signals is described by attributes submitted on the definition statements of individual neurons. Each of the kinds of neuron outlined here has the same choice of output options, which cover most of the needs of the principal neural models. The response of the output stage is one of the following functions of the amplitude of its input.

1. **Signum**

   Output is a clamped positive value if a signal threshold is exceed. Simpler artificial neural networks tend to use this form. This is basically a yes / no response to input signal amplitudes.

2. **Sigmoid**

   The input versus output curve has an S shaped characteristic. Input values below a given cut off lead to a lower fixed output. Values above a second limit lead to an upper fixed output. Response in the middle region between these limits is about linear, but approaches either fixed output asymptotically. This form resembles that for real neurons.

3. **Linear, Clamped**

   Output is linear with inputs between two limits, but takes hard clamped values for inputs above the upper limit or below the lower limit. In our models, there is little difference in overall controller behavior between this mode and the sigmoid.

4. **Linear**

   Outputs are proportional to inputs. The linear option is mainly for comparing with certain primitive systems. Systems based on linear responses tend to be unstable, unless run at very low gains, especially relative to the non linear coupled ones presented here.

All output signal forms can have an effective gain given. Forms #2 and #3 have an adjustable input / output curve slope. The first three forms have adjustable limiting output levels. A dead zone, that is a small part cut out of a central region of input signals, may be specified, so no output signal will result unless the input level is beyond the dead zone. Dead zones are normally set at relatively low levels suitable for filtering noise. They effectively dampen oscillations and ringing of the system, and permit higher gains.

Excessive dead zones and inadequate clamping cut offs can contribute to system response delays, but performance is otherwise rather insensitive to these variables over a considerable range of values.

4.2. Filter Neurons

A filter neuron combines one or more connected input signals, applies rules for filtering the combined signal, and supplies an output signal accordingly. The forms encoded here are based on SLC practice about 1989. Filters are intended to reduce the effects of short term noise and transients on the combined input, which is typically derived from one or more sensory neurons. Filtering rules include averaging over recent history, some criteria for enthusiasm in responding, and threshold criteria. Selection, median, and exponential behaviors are included. Ideally filters can also have some adaptive characteristics. Filters also can be told to include a response based upon predicting the signal expected a cycle or two ahead, using most recent derivatives of the filtered signal.

Filters introduce a delay in system response to changing sensory signals, and typically are set for about two or three cycles for the median, and at least one for the exponential. Together with signal clamping, properly adjusted filters effectively eliminate overshoots.

Filter attributes include a list of input signals, along with weights, averaging and history criteria, and output pulse forming information.

4.3. Mixing Neurons

Mixing or adder neurons combine a set of connected input signals according to weights, and supply an output signal. A typical rule for firing might be based upon the weighted adding of all inputs. Other choices might be to add the absolute values, or the squares of inputs, as in finding the analogue of a penalty function. Mixers differ from filter neurons mainly in that they do not try to smooth input noise. A main application is building a corrector signal from a collection of filtered input signals. In one simple kind of system, the input weights would correspond to the coefficients of a calibration matrix linking filtered sensor inputs to correction outputs. Ideally, these weights should be adjusted according to the tracking experience of the system. Both exciting and inhibiting inputs are accommodated. Mixers are suitable for position controllers in single pass lattices, such as linacs and beam lines.

Mixer neurons resemble the basic middle neurons of most artificial neural nets. In our model they may also be given various roles in calibrating their own input weights. Overall system gain might be specified at this stage. For stability, system gains should be set less than the inverse of the mean filter delay.

Mixer attributes include a list of input signals and weights, and output pulse forming information.

A neural based controller with the set of simple properties given here has been used to model the SLC beam steering feedback at Stanford with considerable success. Using plain linear coupling, and typical SLC signal characteristics, observed stability, gain, and timing can be matched quite well. Using non linear couplings, the model version of the feedback can be operated at much higher gain and hence better response time to transients. Filter properties could be somewhat improved over those originally used, and were conveniently optimized in the program. Similar changes were indeed introduced into later feedback improvements on the actual linac.

4.4. Micado Based Controller

For circular machines, the simpler mixer schemes can need a lot of work to operate reliably without additional constraints. There are a number of complications that can lead to instabilities such as opposing correctors running off scale if not properly damped and constrained. The orbit has to be well sampled, which leads to relatively large networks, which in turn tend to need further guidance beyond simple mixing of weighted signals. As an alternative, the conventional Micado correction technique has been adapted to serve as a controller, functioning as a complete mixer layer. It receives beam position signals from a front end filter layer, computes a single iteration, and feeds the resulting error signals into output pulse handlers, and then to the correctors. These steps repeat for each beam transit or measurement cycle. This scheme appears very stable under BNL AGS and RHIC based simulations. Corrector strengths can be constrained.

The inner workings of the Micado resemble somewhat those of our mixer layer. Micado makes use of a matrix correlating monitor readings with corrector settings. The coefficients of the inverse of this matrix correspond to the mixer weights for the case of a centered orbit. Micado forms a penalty function from among these weights and current monitor readings, and then selects the corrector with the most effect on this function. The corrector is adjusted to maximize its effects by means of a Householder transformation. This procedure is iterated among the correctors.

4.5. Motor Neurons

In a simple system, motor (relay) neurons carry the network response as computed from the incoming sensory signals back to the device hardware. For example, a motor neuron might attach to a corrector driver, delivering signals to adjust a kicker magnet one step up or down. This kind of neuron can provide an added layer of non linear coupling between the controller and its correctors, giving further noise suppression. We have found that the Micado kind of controller works quite well without this additional layer. In another case, a motor neuron might accept an inhibit feedback pulse if the corrector is at the end of its adjustment range. Output noise and delays in transmitting the signal to the device might be included.

Motor neuron attributes include a list of signals such as mixer neuron outputs, weights, and output signal descriptors.

4.6. Others

While an example controller using the above collection of parts already offers a big improvement over an elementary linear feedback system, there are further ideas to be tried. Most other feedback features and functions that come to mind can either be added as rules in one or more members of the above set, or if too specialized, can be expressed in another kind of neuron. One major problem is that calibration schemes in artificial nets seldom even barely resemble the learning process in living systems. Some learning algorithms need excessive amounts of computation, and tend to forget when another kind of event is commanding attention. Some kind of artificial logic has to decide whether a previous step leads to a better or a worse orbit situation. In overdetermined cases, this decision might be based on rules using a traditional penalty function or some weighted comparison of before and after displacements. There are unexplored trade offs between rigid learning schemes and more adaptable ones.

The neural paradigm in principle should be able to provide logic for handling misbehavior of various automated or even manual systems. Situations that come to mind are dynamic reconfiguring when monitors or correctors drop in and out of service, and smoothing operations that avoid excessive power in correctors. Two or more nearby correctors should not be pushing against each other to cure a small beam displacement. Recasting the Micado algorithm along more neural lines could be helpful for steering orbits which are deliberately not centered in their machines.

Another kind of neuron, perhaps related, on a longer term averaging basis, could be concerned with adjusting weights among the mixing and filter neurons, essentially implementing second order feedback in the network. A wider choice of training and response rules might be helpful here, including the by now classical ones. (Back propagation, feed forward, etc.)

5. Device Drivers

Device drivers are classes of logical control system elements that attach directly to hardware. They are distinguished in this model framework from the neurons of the controller networks. Purely hardware related duties, such as conversions, calibrations, response time features, and the like, are treated in these drivers, so these tasks do not spill out into parts of the controllers. For configuration purposes, these drivers are considered as the external layers of the network. However, their duties are performed as separate steps under the control of the accelerator model supervisor.

The simulation framework provides a communications model so devices and neurons can be configured among a number of controllers, possibly located in different servers. A crude slotted buffer scheme is supplied which is adequate for exploring timing effects. In accelerator control systems, devices are usually distributed along the machine, conceivably complicating event timing because of network delays. Those signals which must be shared among neurons in different servers are assigned network slots, which look logically like cells in a buffer which is repeatedly copied to each active node. An initial configuration process handles the details of linking the signal paths among the neurons and devices. In practice most complications caused by signal delays appear to be handled readily by reducing system gain.

5.1. Beam Position Monitors

Monitor drivers simulate the gathering of signals from beam monitor hardware registers. They apply conversions and calibrations, and supply refined coordinate values to local buffers and optionally to network buffers. In a real machine, registers are filled by hardware processing analog signals from beam transits. During the monitor reading step in a beam transit model, an event supervisor sends messages to a utility that gathers readings at requested lattice positions saved during the last transit(s). Optionally the supervisor can reformat this data to values that resemble those produced by hardware, and puts it into assigned registers. Then instrumental noise and least count effects can be introduced as appropriate to the real monitor hardware. The supervisor commands the device monitors to process their raw readings, and distributes these readings to other parts of the control system. Thus at regular intervals register values are converted to beam position data, noise is included, and the data may be passed to the controllers.

5.2. Corrector Drivers

Corrector drivers accept signals from motor neurons, perform any necessary conversions, do validity checks, and operate the necessary hardware, real or pretended. In the models, a simple linear delayed corrector magnet response mechanism is also included so simulated corrector readings will better reflect the way magnets respond to new settings. Long term signal and setting averaging is performed, which can be used for dynamic weight updating. Corrector drivers have an option for the gain applied to their processed input signals, and system gain as a whole can be adjusted by applying a common value to all corrector gains.

6. Calibration

Calibration procedures find the weights which link input signals to the kinds of response expected of the controller. In effect they train the controller. Ideally these efforts achieve trustworthy controller behavior in a relatively short training period. The major kinds of neural networks differ strongly in the way that they train, and these differences tend to require specialized controller sections to implement for modeling. In our model, the various calibration schemes outlined here are applied to a network which uses mixers for the middle layer.

Two similar calibration schemes for a feed forward layered geometry are considered here. In both, mixer layer signal weights are adjusted by measuring the effects of a series of corrector settings on the beams. Purely linear output signals are considered, so weights can be compared with the results from lattice optics. Other elements can be exchanged easily in techniques that look for improperly measured monitor locations.

Case 1. With the controller off, a corrector is set to deflect the beams, and corresponding monitor measurements made. The weights of each adder are adjusted so that the resulting adder signals match the corrector values, using the Adaline rule of Widrow [ ]. A pattern of corrector changes is iterated, "randomly", through a few cycles, until the weight changes settle down. Trials on a linac example show good convergence to reasonable values, from sets of starting weights within an order of magnitude or so of the results. The weights and conditions are also effective for tracking, leading to reasonable response without ringing. While there is no general guarantee of convergence for this technique, the model can readily be operated to explore plausible ranges of critical parameters.

Case 2. This case matches the original SLC scheme, and so allows the behavior of real and simulated systems to be compared. Correctors are suitably varied in turn, and averages of dX / dC among correctors and filtered X signals are obtained. These responses are gathered into a calibration matrix, which is inverted. Each row of the resulting matrix corresponds to the weights of an adder.

This method delivers reasonable fits when the rank of the system matches the number of degrees of freedom. Given the involvement of the inversion step, it misbehaves for overdetermined configurations.

The Micado option currently uses computed phase advances among the monitor - corrector ensemble for a somewhat equivalent internal weighting scheme.

7. Linac Example

Having been strongly influenced by an early SLC beam steering feedback problem, naturally our feedback controller modeling software was applied to that particular system. By joining a controller to a model of the SLC, the feedback strategies were evaluated and parameters adjusted, comparing with the behavior of the original system when helpful. These remarks refer to SLC operation during late 1989.

During each SLC cycle, a pulse of positrons was accelerated. A pulse of electrons closely followed, using the same beam optics, and spaced in time so the two bunches met at a collision point in the arcs. These cycles repeated at 60 or 120 hertz. Because most beam disturbing conditions change slowly, an elementary system of sensors feeding back to local beam steering correctors proved adequate to guide the Linac beams into the SLC Arcs. In each transverse plane four sensors recorded beam positions for a positive and for a negative beam transit. These beam position signals were routed into four filters, which combined pairs of readings with previous filtered ones and applied certain rules to avoid grossly bad readings. These filter signals were then passed into a signal mixer, which combined them into a set of corrector settings, using a previously prepared calibration table (Case 2 above). The computed changes in output values were reduced to some small fractions, the gains, and passed on as signals to the corrector hardware controllers. Finally, corrector settings moved to the requested values, subject to appropriate rules about limits, etc. When a beam drift was detected, its effect made its way through the filters, and a partial correction signal was applied to succeeding pulses.

Originally, the SLC feedback system was run at very low gain to avoid instabilities, and had no overall optimizing criteria other than reducing individual beam displacements. The low gain led to an effective response time of the order of a second. Judicious placement of monitors and correctors in principle allowed separate two monitors by two correctors circuits to control the two beams independently. In practice there was substantial cross talk between these two circuits, which this model has shown. Including the otherwise neglected correlations in four by four configured models improved transient response significantly and further reduced overshoots.

In a more general scheme other feedback systems may be working at different places along a beam, so the coupling among the systems has to be considered. One simple strategy is to emphasize one such local feedback system at a time, progressing along the machine. An upstream controller gates off, or reduces the gain of the ones below it off until the first one has resolved its own problem. Then the emphasis shifts to the next controller. Invariably there are tradeoffs among response time and average system performance, and simulations can help with the needed tuning among the various controllers.

8. Environment

Beam controller studies and implementations are practical on modern workstations, which have effective compilers, impressive computing speed, and few memory constraints. The slowest part of models is still the simulating of particle tracking, even with factor of ten improvements in the BNL MAD versions used here. Attached to an accelerator, the correction schemes seem capable of millisecond scale sampling cycles. The schemes have been easily incorporated into the MAD language and BNL data base servers. The nastier parts of the object definitions and structures are readily handled by the structure generating tools and file management techniques developed for the MAD Program at BNL. Hardware costs of minimal workstations are low enough for them to be used as dedicated controllers.