Fermi National Accelerator Laboratory

FERMILAB-Conf-95/005

Orbit Analysis

Leo Michelotti

Fermi National Accelerator Laboratory
P.O. Box 500, Batavia, Illinois 60510

January 1995

Proceedings of the *Workshop on Nonlinear Dynamics in Particle Accelerators: Theory and Experiments*, Arcidosso, Italy, Sept. 4-9, 1994.

Disclaimer

Orbit Analysis

Leo Michelotti

Fermilab[1]
P.O.Box 500
Batavia, IL 60510

We take an overview of recently developed methods for studying single particle orbits in accelerators and discuss some physics underlying those which involve Lie operators. It will be further argued that object-oriented programming provides the appropriate computing strategy in which to model accelerators and to implement these techniques.

The past fifteen years have witnessed a remarkable development of methods for analyzing single particle orbit dynamics in accelerators. Unlike their more classic counterparts, which act upon differential equations, these methods proceed by manipulating Poincaré maps directly. This attribute makes them well matched for studying accelerators whose physics is most naturally modelled in terms of maps, an observation that has been championed most vigorously by Forest. (16,17) In the following sections we will sketch a little background, explain some of the physics underlying these techniques, and discuss the best computing strategy for implementing them in conjunction with modeling accelerators.

MICRO-HISTORY

Accelerator physics has had a long involvement with perturbative techniques in classical mechanics. By the 1960's reference was made frequently to "Moser transformations" (27) as a technique for constructing normal form Hamiltonians. This referred to the use of generating functions to provide canonical transformations to normal form coordinates in an action-angle representation. Although it seemed the natural approach – largely because textbooks in classical physics emphasized both generating functions and action-angle coordinates – this technique was, in fact, decoying physicists away from another, more profitable path. At the same time that they were barely obtaining second order results for simple models (e.g., see Cole (10)), Deprit and his colleagues (12,20) already had written a general purpose algorithm using Lie brackets that could be used easily to all orders, and they had made it ready for automated application[2] to serious problems in celestial mechanics. Indeed, the fundamental idea of applying Lie brackets to construct normal forms goes back even further. Poincaré himself proved a theorem, one of the many which now bear his name, amounting to a first order normal form calculation using the Lie bracket, and what originally led Sophus Lie to invent Lie algebras and groups was his study of how the form of differential equations change under coordinate transformations. But neither of these geniuses possessed the power of today's computational hardware; whether that actually hindered them will be left to the judgment of the reader.

In 1981, during the first U.S. Particle Accelerator School, Dragt independently introduced a different Lie algebraic method into accelerator theory. (14,13) Unlike Deprit, Dragt based

[1]Operated by the Universities Research Association, Inc. under contract with the U.S. Department of Energy.

[2]In fact, Deprit's work inspired certain developments in the symbolic language MACSYMA, which was being written in the same period. A family of modules involving Poisson brackets and Taylor series were written to support his calculations.

his ideas on discrete maps rather than continuous flows.[3] To some this formalism at first appeared formal and somewhat contrived: a bag of computational tricks that seemed to have little physics motivation behind them.[4] That this misperception arose was a result of the way in which we were taught classical mechanics. Unlike the generating function techniques, which have always possessed a strong academic tradition, key ideas behind these manipulations simply did not appear in the curriculum.

In the next section we will present the central physics concept behind normal form calculations that use Lie operators. While it may not be necessary for everyone to master every detail of these methods, it is important that physicists understand their underlying physical basis, which is so fundamental that the fact that they traditionally have not been an integral part of undergraduate curricula is something of a scandal. In doing this, we do not mean to ignore other activities occuring in this same time frame. Space does not permit anything approaching a comprehensive review, but the following milestones should be noted before moving on to the main topic.[5]

Automatic differentiation and differential algebra. Automatic differentiation and differential algebra (AD/DA) provide a powerful framework upon which to build perturbation theoretic and optimization software. AD is a method of doing *exact* numerical differentiation. The idea itself is old[6] and can be based on the observation that the basic theorems of differential calculus provide rules for propagating the values of a function's derivatives through arithmetic operations. One can also think of this as manipulating polynomials and truncating results at a particular order. In an "object-oriented" computer implementation (see below), one interprets variables appearing in a program not as double precision numbers but as these polynomial objects. Equations appearing in such programs are then viewed as operating on functions rather than numbers. Berz (4), who emphasized the use of DA as well as AD, was the first to use the techniques for solving accelerator problems, but scientists, mathematicians, and engineers in other fields have been developing and using these tools as well, and there are many variations. (For example, see (11). What may distinguish our field is that we tend to be interested in higher order; most others are generally satisfied with taking only one or two derivatives.)

Symplectic numerical integrators. In 1983, Ruth (28) and Channell (7) independently introduced numerical integration procedures that produced a symplectic map at each step and, therefore, throughout the interval of integration. Two years later, a similar approach was independently developed in China by Kang (21).[7] By 1988, Channell and Scovel (9) were applying such methods more generally to Lie-Poisson systems, and between 1985 and 1990 the procedures were reformulated in terms of Lie operators. (For example, see Yoshida (32). The first realization of the connection with Lie methods is due to Neri around 1985, and by 1986 Forest had rederived Ruth's 4th order algorithm. (18)) The importance of all these developments is such that no one who wants to be taken seriously would now integrate orbits through a magnetic field using, say, a Runge-Kutta formalism.

---

[3]The approach may have been dimly foreshadowed by Stavroudis (30), who earlier had stressed bracket techniques in geometrical optics.

[4]Over a decade ago I argued with an established colleague about the order in which fifth integer resonances should appear in sextupole perturbation theory. Eventually, in frustration, he closed the discussion with, "But what is the physics behind this? Explain it without using the words 'Poisson bracket.' "

[5]Disclaimer: The following paragraphs belong more to the realm of personal impressions and reminiscence than objective history. For the purpose of this essay, I have not put in the time necessary for authoritative historical research.

[6]For example, an early FORTRAN implementation can be found in Chapter 17 of Arden (1), where it is called "formal differentiation."

[7]I am told that René DeVogeleare actually invented this idea in 1956 and that Jack Wisdom also devised his own version independently. Clearly, it was a notion whose time had come.

"Nekhoroshev-like" calculations. By "Nekhoroshev-like" calculations we mean attempts to predict long term stability bounds based on processing short term information.[8] This is accomplished by first constructing near-invariant surfaces in phase space, typically parametrized by some action coordinate. One measures a *very* small variation among the surfaces over small time scales (say, 50,000 turns) and, by converting this into a bound on the growth rate of an action coordinate, extrapolates to larger (but still reasonably small) variations over time scales, say, an order of magnitude greater. The description of the method implies its limitation: it applies to regions of phase space in which the motion is very nearly regular (integrable). Beginning in 1987 and extending to the present, Ruth, Warnock, and collaborators (3,31) have studied properties of maps by Fourier analysis in angle variables. While originally appealing to the Hamilton-Jacobi equation for its foundations, the flavor of their work has changed somewhat recently and can be understood as fitting maps – using the "data" provided by tracking programs – to Fourier series using splines in the action coordinates to model the coefficients. Using such representations, they can construct near invariant surfaces and obtain long term stability estimates, within the constraint of near-integrability already noted (31). Similar surfaces can be obtained using perturbation theoretic methods. Recently Berz and Hoffstätter (6) have added a new facet to this by using intervals[9] to compute their bounds. Of course, none of this has much to do with the rigorous Nekhoroshev theorem, which is more legitimately represented by the work of Dumas and Ellison (15) on crystal channeling.

Lyapunov exponents. Whether Lyapunov exponents provide an efficient way of obtaining information useful for accelerator physics remains a controversial and intriguing question. A nonzero Lyapunov exponent in a region of phase space means that infinitesimally close orbits in that region diverge from each other exponentially rapidly; the exponent is, if you will, the rate of divergence. This diagnosic is best suited for studying attractors of dissipative systems, but it has been applied to Hamiltonian systems. The controversy arises when considering (a) how to calculate the exponent and (b) whether it is correct to use it for defining a dynamic aperture. Regarding (a), the one systematic study I know of was done in 1991 by Schmidt, Willeke, and Zimmerman (29). Regarding (b), the evidence is experiential: while it is true that chaotic orbits do not necessarily escape (8,19), the claim is that, for all practical purposes, in *realistic* models, if you can calculate a non-zero Lyapunov exponent for an orbit, it will eventually escape.

LIE OPERATORS AND EXPONENTIAL MAPS

Perturbation theories are quests for useful coordinates. Their objective is to find charts on which a vector field (dynamical system, differential equation) or its map is represented "as simply as possible." When one has been found, we call the representation the vector field's "normal form." Perturbation theories provide paths through the forest of all possible coordinates using an "order parameter" natural to the problem at hand.

Let $\underline{z}$ symbolize an array of local coordinates of some phase space and let orbits evolve according to the "equations of motion" ("vector field", "dynamic"), $\dot{\underline{z}} = \underline{v}(\underline{z}, t)$. Let $g : R^n \to R, \underline{z} \mapsto g(\underline{z})$ be (the local representation of) a classical obervable. The Lie derivative of g is its "time" derivative along an orbit and is naturally evaluated by using chain rule. It is associated with a linear, differential operator, $\mathbf{V}$, on the space of observables, in the obvious way.[10]

---

[8] Martin Berz has suggested that these should be called "Lyapunov-like calculations." Actually, they are sufficiently dissimilar to what either Lyapunov or Nekhoroshev did that it probably does not make much difference.

[9] For an introduction to interval analysis, see Moore (26).

[10] A more accepted mathematical notation for "$\mathbf{V}$" is "$L_{\underline{v}}$." I avoid using it here both because

$$\dot{\underline{z}} = \underline{v}(\underline{z},t) \quad \Rightarrow \quad \dot{g} = \dot{\underline{z}} \cdot \frac{\partial g}{\partial \underline{z}} = \underline{v} \cdot \underline{\partial} g = \mathbf{V} g \tag{1}$$

Because d/dt does not depend on the coordinate chart, $\mathbf{V}$ possesses the same property and is itself a geometric object. That is, it is invariant under a transformation, $\underline{z} \mapsto \underline{z}'$:

$$\mathbf{V} \;=\; \underline{v}(\underline{z},t) \cdot \frac{\partial}{\partial \underline{z}} \;=\; \underline{v}'(\underline{z}',t) \cdot \frac{\partial}{\partial \underline{z}'}$$

Because of this, when $\mathbf{V}$ arises from an autonomous, integrable Hamitonian, it can be written immediately in terms of (local) action-angle coordinates as

$$\mathbf{V} = \frac{\partial H}{\partial \underline{I}} \cdot \frac{\partial}{\partial \underline{\varphi}} = \underline{\nu}(\underline{I}) \cdot \frac{\partial}{\partial \underline{\varphi}} \quad , \tag{2}$$

where the array $\underline{\nu}(\underline{I})$ contains the tunes, or winding numbers, associated with the angle variables $\underline{\varphi}$ on the KAM torus labelled with the action coordinates $\underline{I}$. If we take g itself to be a coordinate, then Eq.(1) provides a "covariant" way of writing the original equations of motion.

$$g(\underline{z}) = z_i \quad \Rightarrow \quad (\mathbf{V} g)_{(\underline{z})} = \underline{v} \cdot \underline{\partial} z_i = v_i \quad ,$$

A simple little "proof" that *all* dynamical systems are linear may help to clarify (?) this concept. Consider any two orbits, say $\underline{z}_1(t)$ and $\underline{z}_2(t)$, of an arbitrary vector field, $\mathbf{V} = \underline{v}(\underline{z},t) \cdot \underline{\partial}$, and their sum, $\underline{z}(t) \equiv \underline{z}_1(t) + \underline{z}_2(t)$. To establish linearity, we show that $\underline{z}(t)$ is also an orbit; that is, we verify that it obeys the same equation of motion as $\underline{z}_1(t)$ and $\underline{z}_2(t)$.

$$\begin{aligned} \dot{\underline{z}}(t) &= \frac{d}{dt}(\underline{z}_1(t) + \underline{z}_2(t)) = \dot{\underline{z}}_1(t) + \dot{\underline{z}}_2(t) \\ &= \mathbf{V}\,\underline{z}_1(t) + \mathbf{V}\,\underline{z}_2(t) = \mathbf{V}(\,\underline{z}_1 + \underline{z}_2\,)_{(t)} = \mathbf{V}\,\underline{z}(t) \end{aligned} \tag{3}$$

"Therefore," all dynamical systems are linear, and the subject of nonlinear dynamics does not exist. Anyone requiring more than thirty seconds to spot the fallacy in this argument needs to sit and meditate a little more on the meaning of a Lie derivative.

If $\mathbf{V}$ is autonomous and g is a time-independent observable, then Eq.(1) is solved formally just as Schrödinger's equation is solved in quantum mechanics.

$$\dot{g} = \mathbf{V} g \quad \Rightarrow \quad g_t = e^{t\mathbf{V}}\, g_0 \quad , \tag{4}$$

As a trivial example of this, consider the dynamic on a two-dimensional phase space, $\underline{z} = (x,p)^T$, which expresses a constant force field.

$$\underline{v}(\underline{z}) = \begin{pmatrix} \dot{x} \\ \dot{p} \end{pmatrix} = \begin{pmatrix} p/m \\ ma \end{pmatrix} \quad .$$

We take x as our observable and apply $\mathbf{V} = \underline{v} \cdot \underline{\partial}$ to it.

$$\mathbf{V}x = \left(p/m \frac{\partial}{\partial x} + ma \frac{\partial}{\partial p} \right) x = p/m \quad , \qquad \mathbf{V}^2 x \;=\; \mathbf{V}(p/m) = a \quad ,$$

and, of course, for all $n > 2$ we have $\mathbf{V}^n x = 0$. Evaluating the exponential,

---

it is a bit cumbersome, and because a symbol like "$\mathbf{V}$" represents more familiarly an object with vector properties.

$$x_t = \exp(t\mathbf{V})x_0 = \left(1 + t\mathbf{V} + \frac{1}{2}t^2\mathbf{V}^2\right)x_0 = x_0 + (p_0/m)t + \frac{1}{2}at^2 \quad . \tag{5}$$

The shift in viewpoint that one adopts in doing this is similar to going between the Schrödinger and Heisenberg pictures in quantum mechanics. One normally thinks of the dynamical variables appearing in Eq.(5) as standing for numbers. Start thinking of them as *standing in for functions,* in particular, coordinate functions.

Using these concepts we can now state the "fundamental problem" solved by Lie derivative normal form techniques: given the one-turn map, $\underline{z}_i \mapsto \underline{z}_f$,[11] of a periodic system, find (local) observables $\underline{I}$ and $\underline{\varphi}$ and a function $\underline{\nu} : R^N \to R^N$ such that

$$\underline{z}_f = e^{2\pi\underline{\nu}(\underline{I})\cdot\partial/\partial\underline{\varphi}}\underline{z}_i + O(\epsilon^n) \quad , \tag{6}$$

where $\underline{z}_i$ and $\underline{z}_f$ are now interpreted as coordinate functions, not points. The extra formal term has been attached to the end as a reminder that there is typically an order parameter, ϵ, and that the equivalence to a normal form vector field is calculated up to some finite order in ϵ. From Eqs.(2) and (4) — and remembering that "time" for a one-turn map is really an azimuthal angle that measures location around the ring, so that it increases by 2π in one turn — we see that, on such a chart, the vector field would be $O(\epsilon^n)$ indistinguishable from that of an integrable autonomous Hamiltonian.

That, in a nutshell, is the objective and the physics that drives it. What should be clear, from the expression in Eq.(6), is that the map itself contains all the information necessary to carry out this programme, if it can be carried out. This was the breakthrough of Dragt, Forest, and their collaborators which, among other things, bypassed one of the fundamental difficulties of flow-based procedures: the need to repeatedly traverse the ring of accelerator elements in order to calculate multiple integrals as one goes to higher and higher order.[12] We cannot go into details here on how one tries to solve Eq.(6). Suffice it to say that *computationally effective* perturbation theories, whether applied to maps or continuous flows, proceed along the following lines: (a) Represent the vector field or map on a chart whose origin is a critical orbit, most typically, a fixed point or periodic orbit. (b) Use a "near-identity" transformation to keep linear dynamics invariant. In multi-dimensional problems it is best to begin with a chart that diagonalizes the linear field; these are then the "linear normal form" coordinates, which will be complex for stable systems. (c) Try to eliminate only the lowest order nonlinear terms from the representation via a transformation whose *reverse specification contains only terms of that order.* (d) Represent the map (or vector field) on the new chart. We then have the identical problem as in Step (c) but at higher order; that is, the lowest order nonlinear terms that appear are now of higher order than before. (e) At each stage of the iterative/recursive procedure we solve a *linear system of equations,* the so-called "homological equations," for both the next transformation and the normal form map (or vector field) at that order. This is the reality lurking behind the fraudulent Eq.(3). Dynamical systems are not linear, but their homological equations are, and they *have the same structure at all orders,* one that results from applying a Lie operator to the linear part of the map (or vector field). (f) The complete transformation from the first chart to the last is obtained by composing the individual, much simpler ones together.

---

[11] Which can be constructed by using AD/DA variables in a tracking program.

[12] Fourteen, or so, years ago, Ferdi Willeke complained strenuously about this feature of perturbation theories, stating, We're doing something wrong; there must be a better way. Since then I have always thought of it as "Willeke's lament."

Other normal forms

The normal form expressed in Eq.(6), containing only shear terms, is what is traditionally meant by the term "normal form," but others are possible. One of great importance for accelerator physics is the "resonance normal form," typically associated with the Hamiltonian of an isolated resonance model.

$$H_{\rm res}(\underline{\varphi},\underline{I};\theta) = \underline{\nu}\cdot\underline{I} + H_s(\underline{I}) + H_r^{(c)}(\underline{I})\cos(\,\underline{m}\cdot\underline{\varphi} + n\theta\,) + H_r^{(s)}(\underline{I})\sin(\,\underline{m}\cdot\underline{\varphi} + n\theta\,) \quad .$$

H_s symbolizes the "shear" terms, while $H_r^{(c)}$ and $H_r^{(s)}$ are the "resonance" terms, which produce separatrices. Normally, one untwists the separatrices via transformations, $(\underline{\varphi},\underline{I}) \mapsto (\underline{\xi},\underline{J})$, linear in the action-angle coordinates. On this chart, the resonance normal form is autonomous,

$$K(\underline{\xi},\underline{J}) = K_l(\underline{J}) + K_s(\underline{J}) + K_r^{(c)}(\underline{J})\cos\xi_1 + K_r^{(s)}(\underline{J})\sin\xi_1 \quad .$$

where the K_l is a linear functional of $\underline{J}$, and the coefficient of J_1 is $\partial K_l/\partial J_1 = \underline{m}\cdot\underline{\nu} + n$. Automated perturbation theories can compute resonance normal forms at whatever order they appear. Indeed, these can be extended to multiple resonance models, (e.g., see (22)) in which case the "normal form" is no longer integrable.

And still other forms are possible, such as linear models with complex eigenvalues off the unit circle, which provide normal forms for linearly unstable motion (e.g., see Berz (5), which uses a different procedure, based solely on composition rather than exponential maps). All of these are obtainable by applying automated perturbation theories to either maps or flows, but the map algorithms are generally better for application to realistic, "hopelessly complicated" (16) accelerator problems.

COMPUTATIONAL CONSIDERATIONS

In the original email message announcing the Arcidosso Workshop, the Theoretical and Computational Techniques Group was charged with the question, "What can we and what cannot we calculate well?" To this I would add, "*How* can we calculate well?" This is no small consideration. Models of what it means to compute have undergone revolutionary changes in the last ten to fifteen years, and yet many (most?) physicists still program as they did in 1965. It is rather embarassing, but musicians and artists have done a much better job of keeping up than we. Many (most?) of us have become, by today's standards, computationally illiterate. It may be stretching an analogy, but it almost seems as though late twentieth century physicists think of programming in a way similar to that in which their late nineteenth century predecessors thought of group theory. Perhaps developments in the twenty-first century will be able to shake off this complacency. If nothing else, it is certainly true that the next generation will view this activity very differently from the present one.

The particular computing model that I strongly commend to your attention is object-oriented programming (OOP). It began in the late 1960's, at places like Xerox PARC, but did not become dominant until the early 1980's, when the infamous "Macintosh interface" made its advantages manifestly obvious even to the most obtuse. Over five years ago (25) I became convinced that OOP provided our best hope for writing accelerator modeling and analysis software that might survive into the 21st century and guessed that C++ would be the appropriate language for creating scientific objects (24,23). This idea is shared by a small but growing number of people writing software for accelerator applications[13] from

[13] Among whom are included Christoph Iselin, Chris Saltmarsh, Hiroshi Nishimura, Jim Niederer, Jim Holt, Johan Bengtsson, Nick Walker, Jane Richards, and others.

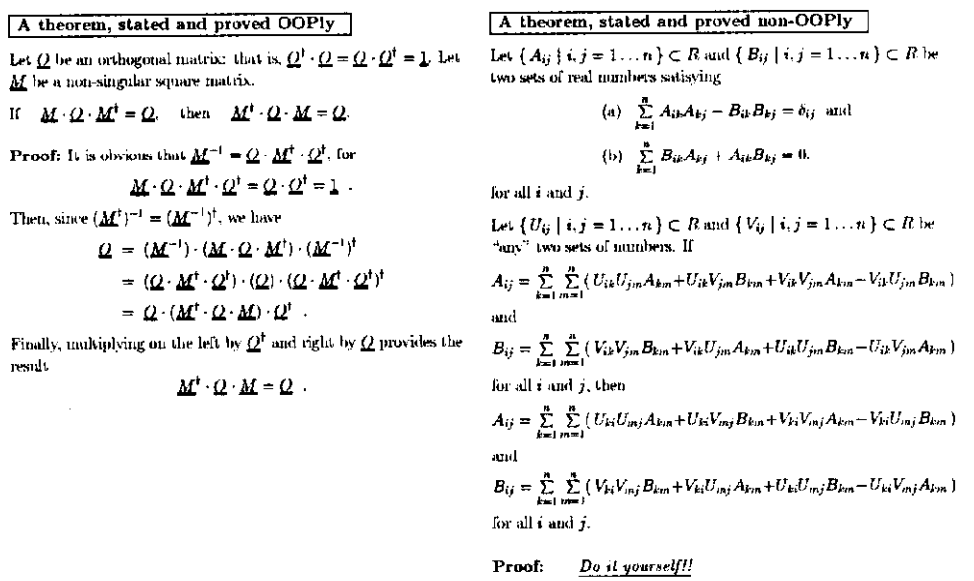

FIG. 1. A theorem stated with and without useful mathematical objects.

whose combined activities will undoubtedly emerge even better objects for orbit analysis. And in the larger realm of high energy physics can be found the GISMO project (2), a collaborative exploitation of OOP for the purpose of modeling detectors. And yet these are exceptions: three decades have gone by without seeing OOP enter the mainstream of scientific programming. In a final, forlorn attempt to influence a few people to look into *what may well be the most significant breakthrough in computing since FORTRAN,* we will touch on a few of its basic ideas.

Object oriented programming

Early introductory books on object-oriented programming employed the metaphor of "passing messages to objects," which was more appropriate for graphical user interfaces than scientific programming. To illustrate better what OOP means in the context of physics, consider the two foils shown in Figure 1. On the left a simple matrix theorem is stated and proved; on the right the same theorem is only stated *but without using the mathematical objects Matrix and Complex Number.* This is how the theorem would have to be written if mathematicians had not invented these useful objects. While both the content and proof of the theorem are simple to understand on the left, the statement on the right is essentially impossible to comprehend, and the obscurity of a proof can only be imagined. High level mathematical objects allow one to focus on essential algebraic features without tripping over details of implementation. In mathematics, *objects hide inessential detail* to make the structure of calculations and proofs more transparent.

"Object-oriented programming" is an extension to programming of this same fundamental strategy that has been used in mathematics for centuries, as seen in statements like:

$$z = x + iy \;=\; r \cdot e^{i\theta} \;, \quad \underline{y} = \underline{\underline{A}} \cdot \underline{x} + \underline{\varepsilon} \;, \quad \text{or} \quad \underline{\underline{B}} = \left(\underline{\underline{A}}^\dagger \cdot \underline{\underline{A}} \right)^{-1} \cdot \underline{\underline{A}}^\dagger \;. \tag{7}$$

By using objects, these simple expressions hide details of manipulations on what could be large structures of data, freeing the mind to focus on *what* is happening rather than *how.* OOP is, in fact, a *natural* way to organize complex problems.

In the context of both mathematics and programming, objects comprise (a) structures of data and (b) functions and operators that manipulate these data; to these, programming adds (c) rules for specifying what happens when objects come into or go out of scope. OOP is a technology for realizing and fully utilizing this abstract concept. Its defining attributes are:

Data Abstraction/Encapsulation: Structures of data and the functions and operations which manipulate them are joined to define new data types, thereby extending the programming language. As far as the language is concerned, these new, abstract data types are as valid as its original, primitive ones. Data hiding builds a wall around a type's "private" data so that it can be accessed and modified only by the object's member functions.

Inheritance: One class (or object) can "inherit" the data, functions, and properties of another by being "derived" from it. For example, if a Matrix object is available, a new object, SquareMatrix, can inherit all of its properties and then extend them by including new member functions, such as *determinant* or *inversion*. In this way, we can reuse already existing functionality and build on it incrementally.

Polymorphism: Polymorphism means using different implementations of the same interface. For example, in Eq.(7) operator symbols like "+," "=," and "·" have been overloaded so that their meaning depends on context, yet the human reading these equations automatically understands what is going on, because the symbols, in fact, have similar properties in different contexts. The principle is that things that behave similarly should look the same, so that application programmers can learn more quickly how to use what is new.

Sample: C++ code fragment

Despite popular belief to the contrary, the language in which one programs *is* important. A computing language is more than a method for communicating instructions to a machine, or we would all be programming in assembly language; it is also the medium in which people formulate solutions to problems. In the last section we suggested that object-oriented programming provided a model of extensible computing languages that was flexible and powerful enough to be used in scientific work. The objective in using OOP for programming is the same as that in mathematics: to use a syntax that makes it easier to understand what is being done. For example, let the columns of $\underline{\underline{B}}$ be the eigenvectors of the Jacobian of a symplectic one-turn map. One step in writing a linear normal form normalizes the eigenvectors according to the condition

$$\underline{\underline{B}}^T \underline{\underline{J}}\,\underline{\underline{B}}\,\underline{\underline{J}} = \eta\,\underline{\underline{1}}, \quad \text{where } |\eta| = 1 \text{ and } \underline{\underline{J}} = \begin{pmatrix} \underline{\underline{0}} & \underline{\underline{1}} \\ -\underline{\underline{1}} & \underline{\underline{0}} \end{pmatrix} . \tag{8}$$

This is implemented in the C++ code fragment shown in Figure 2. Lines 1-5 both instantiate variables via declaration of their types (`Map`, `MatrixC`, `MatrixD`, and so forth) and perform computations using member functions **Jacobian** and **eigenVectors** of the Map and Matrix classes. In Line 2 eigenvectors of the Jacobian of **theMap** are calculated and stored in the complex matrix variable `B`. The operations are run in tandem (piped): **Jacobian** is a member function of the Map class which returns a double precision matrix. Upon this matrix is invoked **eigenVectors**, a member function of the matrix class, which then returns the complex matrix that is stored in `B`. Line 3 is a declaration of the 6×6 matrix $\underline{\underline{J}}$,[14] Line 5, which implements Eq.(8), begins constructing a normalizing matrix, `Nx`, and if **theMap** is symplectic, `Nx` is diagonal, so that Lines 6-8 perform the required normalization.

[14] Matrix constructors allow simple declarations for special matrices, such as $\underline{\underline{J}}$ or $\underline{\underline{1}}$.

```
/* Line  1 */    Map theMap;

                 ... << theMap is obtained here >> ...

/* Line  2 */    MatrixC B = theMap.Jacobian().eigenVectors();
/* Line  3 */    MatrixD J( "J", 6 );
/* Line  4 */    complex mi( 0., -1. );

/* Line  5 */    MatrixC Nx = mi * B.transpose() * J * B * J;

/* Line  6 */    for( int i = 0; i < 6; i++ )
/* Line  7 */     Nx( i, i ) = 1.0 / sqrt( abs( Nx(i,i) ) );

/* Line  8 */    B = B*Nx;
```

FIG. 2. C++ code fragment that normalizes eigenvectors of a Jacobian.

The important point here is that even those not familiar with the C++ language can follow the flow of the program and understand what is going on fairly easily.[15] High level programming languages were devised for the convenience of humans, not machines. The ideal is: *If you understand the theory, the program itself should be transparent.* The syntax of C++ , and other OOP languages, lends itself to this ideal, although it is difficult to overcome completely the ability of bad programmers to write obscure code.

REFERENCES

1. Bruce W. Arden. *An Introduction to Digital Computing.* Addison-Wesley, Reading, Massachusetts, 1963.
2. W. B. Atwood and T. H. Burnett. Gismo: A set of C++ classes for HEP. In *Proceedings of the International Conference on Computing in High Energy Physics*, 1992.
3. J. S. Berg and R. L. Warnock. Symplectic full-turn maps in a Fourier representation. In *Proceedings of the 1991 IEEE Particle Accelerator Conference.* IEEE, 1991. IEEE Catalog Number 91CH3038-7.
4. M. Berz. *COSY INFINITY: Version 6 Reference Manual.* National Superconducting Cyclotron Laboratory. Department of Physics and Astronomy. Michigan State University. East Lansing, MI 48824. 1993. MSUCL-869;, *Nuclear Instruments and Methods*, A258:431, 1987; Differential algebraic description of beam dynamics to very high orders. *Particle Accelerators*, 24(2):109, March 1989.
5. — Differential algebraic formulation of normal form theory. In *Proceedings of the International Workshop on Nonlinear Problems in Accelerator Physics. Berlin.*, pages 77–86. IOP Publishing, Ltd, 1992.
6. — and G. Hoffstätter. Exact estimates of the long term stability of weakly nonlinear systems applied to the design of large storage rings. *Interval Computations*, In press.
7. Paul J. Channell. Symplectic integration algorithms. AT-6:ATN-83-9, April 1983.
8. — Stable symplectic maps. *J. Math. Phys.*, 32:408, February, 1991.
9. — and C. Scovel. *Physica D*, 50:80, 1991; Symplectic integration of Hamiltonian systems. *Nonlinearity*, 3(2):231–59, May 1990.
10. F. T. Cole. Nonlinear transformations in action-angle variables. Technical report, Fermilab, June 13, 1969. TM-179.
11. G. Corliss and A. Griewank, editors. *Automatic Differentiation of Algorithms: Theory, Implementation, and Application.* SIAM, 1991. Philadelphia, PA.

[15] If you are asking yourself why the complex variable `mi`, set to $-i$, was needed, you are helping to prove this point. In fact, it isn't.

12. A. Deprit. Canonical transformations depending on a small parameter. *Celestial Mechanics*, 1:12–30, 1969.
13. A. J. Dragt. Lectures on nonlinear orbit dynamics. In *Physics of High Energy Particle Accelerators*, pages 147–313. American Institute of Physics, New York, 1982.
14. —, Filippo Neri, Govindan Rangarajan, David R. Douglas, Liam M. Healy, and Robert D. Ryne. Lie algebraic treatment of linear and nonlinear beam dynamics. In *Annual Review of Nuclear and Particle Science, Vol. 38*, pages 455–496. Annual Reviews, Inc., 1988.
15. H. Scott Dumas and James A. Ellison. Nekhoroshev's theorem, ergodicity, and the motion of energetic charged particles in crystals. In James Ellison and Herbert Überall, editors, *Essays in Classical and Quantum Dynamics*. Gordon & Breach, 1991. Chapter 2.
16. Etienne Forest. A Hamiltonian-free description of single particle dynamics for hopelessly complex periodic systems. *Journal of Mathematical Physics*, 31:1133, 1990.
17. —, Leo Michelotti, Alex J. Dragt, Scott J. Berg, and Johan Bengtsson. The modern approach to single particle dynamics for circular rings. In Melvin Month, Alessandro G. Ruggiero, and Wu-Tseng Weng, editors, *Stability of Particle Motion in Storage Rings*. American Institute of Physics, 1994. Particle and Fields Series 54, No. 292.
18. — Private communication.
19. R. Gerig, Y. Chao, and L. Michelotti. Modelling nonlinear behavior in the Fermilab Main Ring. In *Proceedings of the 1989 IEEE Particle Accelerator Conference. Chicago, IL.*, volume 2, pages 1277–1279. IEEE, March 20-23, 1989.
20. A. A. Kamel. Expansion formulae in canonical transformations depending on a small parameter. *Celestial Mechanics*, 1:190–199, 1969.
21. Feng Kang. On difference schemes and symplectic geometry. In *Proceedings of the 1984 Beijing Symposium on Differential Geometry and Differential Equations*. Science Press, Beijing, China, 1985; Difference schemes for Hamiltonian formalism and symplectic geometry. *Journal of Computational Mathematics*, 4(3):279–289, July 1986.
22. Leo Michelotti. Perturbation theory and the single sextupole. In Y. S. Kim and W. W. Zachary, editors, *The Physics of Phase Space*. Springer-Verlag, 1987.
23. — MXYZPTLK: A C++ hacker's implementation of automatic differentiation. In G. Corliss and A. Griewank, *op cit.*
24. — MXYZPTLK and BEAMLINE: C++ objects for beam physics. In *Advanced Beam Dynamics Workshop on Effects of Errors in Accelerators, their Diagnosis and Correction. (Corpus Christi, Texas. October 3-8, 1991)*. American Institute of Physics, 1992. Conference Proceedings No.255.
25. — Exploratory orbit analysis. In Floyd Bennett and Joyce Kopta, editors, *Proceedings of the 1989 IEEE Particle Accelerator Conference*. IEEE, March 20-23, 1989. IEEE Catalog Number 89CH2669-0.
26. Ramon E. Moore. *Methods and Applications of Interval Analysis*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, Pennsylvania, 1979. SIAM Studies in Applied Mathematics.
27. J. Moser. *Nach. Akad. Wiss. Gottingen*, IIA, No.6:87, 1955; *Nachr. Akad. Wiss. Gottingen Math. Phys.*, 2, 1962.
28. Ronald D. Ruth. A canonical integration technique. *IEEE Transactions on Nuclear Science*, NS-30(4):2669–71, August 1983.
29. F. Schmidt, F. Willeke, and F. Zimmerman. Comparison of methods to determine long-term stability in proton storage rings. *Particle Accelerators*, 35(4):249–56, 1991.
30. O. N. Stavroudis. *The Optics of Rays, Wavefronts, and Caustics*. Academic Press, New York, NY, 1972.
31. Robert L. Warnock. Close approximations to invariant tori in nonlinear mechanics. *Physical Review Letters*, 66(14):1803–1806, 8 April 1991; – and R. D. Ruth. Invariant tori through direct solution of the Hamilton-Jacobi equation. *Physica*, 26D:1–36, 1987; – and Ron D. Ruth. Long-term bounds on nonlinear Hamiltonian motion. *Physica D*, 56:188–215, 1992.
32. H. Yoshida. *Physics Letters A*, 47:190, 1990.