QUARTERLY OF APPLIED MATHEMATICS
APRIL, 1972

SPECIAL ISSUE: SYMPOSIUM ON
"THE FUTURE OF APPLIED MATHEMATICS"

THE SYNTHESIS OF DYNAMICAL SYSTEMS*

By

R. W. BROCKETT

Harvard University

Abstract. A significant part of contemporary applied mathematics is concerned directly with communication, control and computation. In these fields many of the central problems involve the *synthesis* of algorithms, or dynamical systems, as opposed to the *analysis* of dynamical systems which predominates in mathematical physics. Arithmetic and numerical algorithms, finite-state machines and electrical filters are examples of the types of dynamical systems which are frequently needed to operate on data, in continuous or discrete form, and to produce data on a compatible time scale. In this paper we discuss the scope and success of some of the synthesis procedures currently available to treat these problems.

1. Computation and control. Dynamical phenomena have attracted the attention of mathematicians for centuries; indeed, several important branches of analysis have developed out of studies in this direction. In mathematics the usual idea of a dynamical system is that of a deterministic, autonomous physical system; the goal of a mathematical investigation is to make qualitative and quantitative predictions about the nature of future behavior [1, pp. 1–4]. In this tradition, tremendous emphasis has been placed on problems of an astronomical origin. In particular, from 1773 (Laplace) to 1889 (Poincaré) the question of determining whether or not the solar system is stable was intensively studied, whereas the question of what to do about it should it prove to be unstable was, apparently, left to the theologians. In any case, the descriptive branch of dynamics has a much longer history than the prescriptive one; though there is evidence that the idea of designing and controlling a dynamical system has roots in antiquity [2, pp. 5–10], the mathematical development of this field is mostly a product of the 20th century.

As contrasted with the Newtonian concept of dynamics, the view is held in numerous areas of applied mathematics that a dynamical system is a kind of operator or algorithm which accepts inputs and produces outputs; the goal of a mathematical theory is either to define a dynamical system which corresponds to a given set of input-output pairs or, in the event that the dynamical system is given, to construct an input which will steer the system to a desired goal. This is typical in control theory and in abstract computational theory. On the control side one can trace this circle of ideas back to the English scientists Heaviside and Rayleigh who, around 1885, discussed the response of physical systems (electrical networks and acoustical resonators) to sinusoidal forcing functions of an arbitrary frequency. Heaviside's goal was to extend Kirchoff's penetrating analysis

* This work was supported in part by the U. S. Office of Naval Research under the Joint Services Electronics Program by Contract N00014-67-A-0298-0006.

of linear, resistive electrical networks to all linear time-invariant electrical networks through the introduction of "capacitative operators" and "inductive operators." This leads to the extension of Ohm's law itself and to the characterization of an electrical network through its "impedance" (Heaviside's term [3]) or its "driving point reaction" (Rayleigh's term [4]). As far as computation is concerned, according to Knuth [5, p. 607], algorithms are "precise rules for transforming specified inputs into specified outputs in a finite number of steps." This makes algorithms very much akin to dynamical systems in the above sense of the words. In 1936 Turing [6] set out to formalize computation in terms of a mechanical device which accepts input data and produces outputs. Subsequent developments led to the formalization of the idea of a finite automaton—perhaps the most elementary dynamical input-output model in existance. If further justification for our language is required recall that even elementary arithmetic with its system of carrying and borrowing corresponds to our intuitive idea of a dynamic process.

Thus we regard both linear dynamical systems (filters) and automata as models for algorithms. Both are dynamic and both accept inputs and produce outputs. The technological interest in each case stems from the fact that specific examples of these models can be synthesized readily and the computations which these systems perform can be very useful. Despite the apparent dissimilarities, even the synthesis procedures for these systems proceed along similar lines. One first of all determines a state space, codes the states and then builds the memoryless inter-connection. In both cases the design proceeds smoothly in simple cases, but in the more complex cases combinatorial problems can be severe and may even prevent the successful completion of phase two. We feel that certain problems in numerical analysis are enough like problems arising in linear system theory to justify a deeper investigation. In both fields the synthesis of algorithms is a major problem. Convergence is the prime objective in many naive applications. Optimality is an issue but of limited practical significance.

In this paper we want to discuss, in a general way, some synthesis questions related to dynamical systems of the input-output, or if you like, algorithmic, type. At times it will be necessary to assume that the reader has some familiarity with linear system theory as it is discussed in the recent literature [7, 8].

2. Discovering interative schemes. Assume that one has a certain function or set of functions to be evaluated. A number of procedures are known for finding recursive schemes to carry out the computation. For example, if the function is a polynomial and if it is to be evaluated in a single point λ then one has Horner's rule,

$$x(k+1) = \lambda x(k) + p_k\,; \qquad x(0) = 0,$$

which evaluates $p_1 s^n + p_2 s^{n-1} + \cdots + p_{n+1}$ at λ with just n additions and n multiplications. This is a special case of a much more general idea which we will go into below.

If U is a set then by U^* we mean the monoid (semigroup with identity) whose elements consist of strings of the elements of U and whose multiplication is concatenation. We include the empty string which is the identity. We consider the problem of evaluating functions defined on U^* mapping into a set Y, and in particular we want to discuss what happens if we give U and Y additional structure corresponding to various types of practical computation.

2.1 *Functions defined on finite sets.* Suppose U is a finite set. If r maps U^* into Y then r defines an equivalence relation on U^* according to which $u_1 \in U^*$ is equivalent to $u_2 \in U^*$ if for each $u_3 \in U^*$ we have $r(u_1u_3) = r(u_2u_3)$. We denote this equivalence

by $u_1 \equiv_r u_2$. Nerode [9] has shown that there exists an iterative scheme to evaluate r which has a finite number of states if and only if $U^*/\equiv_r$ is a finite set. That is to say, if there are only a finite number of equivalence classes in U^* then there exists a finite set X, functions a and c such that $a: X \times U \to X$; $c: X \times U \to Y$, and r is evaluated by

$$x(k+1) = a[x(k), u(k)]; \qquad y(k) = c[x(k), u(k)]; \qquad x(0) = x_0 .$$

The set X is called the set of states. Any two such iterative schemes for evaluating r which are minimal in the sense that every state can be reached from the starting state for some input string, and every two states can be distinguished knowing the input and output data, have the same number of states and this number equals the number of elements in $U^*/\equiv_r$.

2.2 *Functions defined on groups.* Now suppose that U and Y admit a group structure. Say $U = (U, \cdot)$ and $Y = (Y, *)$. Let $r: U^* \to Y$ be a given function. When can we evaluate r by an iterative method of the type

$$x(k+1) = b[u(k)] \cdot a[x(k)]; \qquad y(k) = c[x(k)]; \qquad x(0) = x_0 ,$$

where x takes on values in a group $X = (X, \cdot)$, $a: X \to X$; $b: U \to X$ and $c: X \to Y$ are all group homomorphisms? Since the solution of the above iteration is

$$\begin{aligned} y(0) &= c[x_0] \\ y(1) &= c[b[u(0)] \cdot a[x_0]] \\ y(2) &= c[b[u(1)] \cdot a[b[u(0)]] \cdot a^2[x_0]] \\ &\cdots\cdots \\ y(p) &= c[b[u(p-1)] \cdot a[b[u(p-2)]] \cdot \cdots \cdot a^p[x_0]] \end{aligned}$$

it is clear that in order to be able to evaluate r using a method of this form it is necessary that the value of r corresponding to the sequence $u(0), u(1), \cdots, u(p)$ be expressible as $T_0[u(p-1)]^* T_1[u(p-2)]^* \cdots {}^* T_{p-1}[u(0)]^* \nu_p$. If U is a finite group then in order for X to be finite we require that the Nerode condition be satisfied. Assuming this to be the case, then it is known [10] that there exists a realization of the given form if and only if the sequence $T_0, T_1, T_2 \cdots$ is ultimately periodic in that there exist p and q such that $T_{k+p} = T_k$ for $k = q, q+1, q+2, \cdots$. Moreover, if there are two minimal iterative schemes which compute the same input-output map, say

$$\begin{aligned} x(k+1) &= b[u(k)] \cdot a[x(k)]; \qquad y(k) = c[x(k)], \\ z(k+1) &= g[u(k)] \cdot f[z(k)]; \qquad y(k) = h[z(k)], \end{aligned}$$

then X and Z are homomorphic as groups [10] and there exists a one-to-one and onto homomorphism $\rho: X \to Z$ such that

$$\rho(a(\rho^{-1}(\cdot))) = f(\cdot), \qquad \rho(b(\cdot)) = g(\cdot), \qquad c(\rho^{-1}(\cdot)) = h(\cdot).$$

2.3 *Functions defined on vector spaces.* Perhaps the most intensively studied methods are the linear ones including, for example, Horner's rule as a special case. In this context we give U and Y a vector space structure and consider

$$x(k+1) = Ax(k) + Bu(k); \qquad y = Cx(k)$$

where $x(k) \in \mathfrak{R}^n$; $u(k) \in \mathfrak{R}^m$; $y(k) \in \mathfrak{R}^q$ and A, B, and C are linear maps defined on the appropriate spaces.

This type of first-order iteration is appropriate for evaluating the product

$$\begin{bmatrix} CB & 0 & 0 & \cdots \\ CAB & CB & 0 & \cdots \\ CA^2B & CAB & CB & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix} \begin{bmatrix} u(0) \\ u(1) \\ u(2) \\ \vdots \end{bmatrix} = \begin{bmatrix} y(1) \\ y(2) \\ y(3) \\ \vdots \end{bmatrix} \tag{*}$$

It is well known [7] that given a block Toeplitz matrix it is possible to evaluate its product with a vector, using a first-order linear recurrence with x n-dimensional, if and only if the family of Hankel matrices

$$H_r = \begin{bmatrix} CB & CAB & \cdots & CA^{r-1}B \\ CAB & CA^2B & \cdots & CA^rB \\ \cdots & \cdots & \cdots & \cdots \\ CA^{r-1}B & CA^rB & \cdots & CA^{2r-1}B \end{bmatrix}; \qquad r = 1, 2, \cdots$$

are each of rank n or less. Moreover, if there is one H_r which is of rank n then no lower-dimensional recursive scheme exists and any two recursive schemes of this form, and this minimal order, are related in a simple way. That is, if

$$x(k+1) = Ax(k) + Bu(k); \qquad y(k) = Cx(k)$$

$$z(k+1) = Fx(k) + Gu(k); \qquad y(k) = Hz(k)$$

both compute the same function and if both are of minimal dimension, then there exists a nonsingular constant matrix P such that

$$PAP^{-1} = F, \qquad PB = G, \qquad CP^{-1} = H.$$

The minimum dimension of a realization of the map (*) is called its McMillan degree.

Considering, for a moment, the case where u and y are scalars, there are two canonical realizations available over any field (not necessarily algebraically closed). The first corresponds to the choice

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ -p_0 & -p_1 & -p_2 & \cdots & -p_{n-1} \end{bmatrix}; \qquad b = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}; \qquad c = [q_{n-1}, q_{n-2}, \cdots, q_0]$$

and second corresponds to the choice

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ -p_0 & -p_1 & -p_2 & \cdots & -p_{n-1} \end{bmatrix}; \qquad b = \begin{bmatrix} l_0 \\ l_1 \\ \vdots \\ l_n \end{bmatrix}; \qquad c = [1, 0, \cdots 0]$$

The relationship between the two is most conveniently expressed in terms of a generating function relationship

$$\frac{q_{n-1}z^{n-1} + q_{n-2}z^{n-2} + \cdots + q_0}{z^n + p_{n-1}z^{n-1} + \cdots + p_0} = l_0z^{-1} + l_1z^{-2} + \cdots .$$

2.4 *Functions defined on modules.* To indicate further the scope of these methods we consider an example where the iteration takes place in a module. Starting with a one-dimensional diffusion equation with constant coefficients and periodic boundary conditions, say

$$\frac{\partial x(t, z)}{\partial t} - \alpha^2 \frac{\partial^2 x(t, z)}{\partial z^2} = u(t, z); \qquad x(t, 0) = x(t, \pi), \qquad \frac{\partial x(t, 0)}{\partial z} = \frac{\partial x(t, \pi)}{\partial z},$$

a standard discretization with respect to time and space gives

$$x(k + 1) = Ax(k) + u(k)$$

where A is a circulant matrix. Let $R_p[z]$ denote the ring of polynomials of degree less than p with real coefficients, where all computations are done modulo $z^p - 1$. As is well known, the set of p by p circulant matrices form a ring which is isomorphic to $R_p[z]$. If $R_p^n[z]$ is the n-dimensional module over $R_p[z]$ then we can study

$$x(k + 1) = Ax(k) + Bu(k); \qquad y(k) = Cx(k)$$

where u, x and y take on values in $\mathfrak{R}_p^m[z]$, $\mathfrak{R}_p^n[z]$ and $\mathfrak{R}_p^q[z]$ and A, B, and C are all module homomorphisms. See Kalman, Falb, and Arbib [7] for a general discussion. This example comes from [11] where the above isomorphism is used to good advantage.

3. The optimization of algorithms. The procedures of the previous section can be used to minimize the memory associated with a given computation. Recently, however, there has been a great deal of work devoted to the development of algorithms which minimize other criteria such as the number of multiplications necessary to compute a given function. The problem of evaluating a polynomial at one or more points and the problem of multiplying two polynomials together are examples of problems which have been looked at from this point of view. There has also been some investigation of optimal iteration, as we will discuss later.

3.1 *Arithmetic optimality.* A number of frequently occurring computations, including the two just mentioned, center around evaluating the product of a Toeplitz matrix and vector. That is, to evaluate $p(s) = p_1s^n + p_2s^{n-1} + \cdots p_{n+1}$ at λ using Horner's rule one must compute the product

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ \lambda & 1 & 0 & \cdots & 0 \\ \lambda^2 & \lambda & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \lambda^n & \lambda^{n-1} & \lambda^{n-2} & \cdots & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ \vdots \\ p_{n-1} \end{bmatrix} = \begin{bmatrix} p_1 \\ \lambda p_1 + p_2 \\ \lambda^2 p_1 + \lambda p_2 + p_3 \\ \vdots \\ p(\lambda) \end{bmatrix}$$

To compute the product of $p_1s^n + p_2s^{n-1} + \cdots p_{n+1}$ and $q_1s^m + q_2s^{m-1} + \cdots q_{m+1}$ one must compute

$$\begin{bmatrix} p_1 & 0 & 0 & \cdots & 0 \\ p_2 & p_1 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ p_{n+1} & p_n & p_{n-1} & \cdots & p_{n-m} \\ 0 & p_{n+1} & p_n & \cdots & p_{n-m-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & p_{n+1} \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_{m+1} \end{bmatrix} = \begin{bmatrix} p_1 q_1 \\ p_1 q_2 + p_2 q_1 \\ \vdots \\ p_{n+1} q_{m+1} \end{bmatrix}$$

From the results of the previous section one sees that there are several general statements which can be made giving upper bounds on the number of operations needed to evaluate certain collections of functions.

Case 1 (abstract Horner's rule—I): Let U and Y be sets, let $X = (X, \cdot)$ be a semigroup and let $a: X \to X$ be a semigroup homomorphism. Given any $b: U \to X$ and $c: X \to Y$, the collection of functions

$$\begin{aligned} y(1) &= c(b(u_1)) \\ y(2) &= c(b(u_2) \cdot a(b(u_1))) \\ y(3) &= c(b(u_3) \cdot a(b(u_2)) \cdot a^2(b(u_1))) \\ &\cdots\cdots \\ y(p) &= c(b(u_p) \cdot a(b(u_{p-1})) \cdot \cdots \cdot a^{p-1}(b(u_1))) \end{aligned}$$

can be evaluated with p evaluations of a, b and c and p X-multiplications using the rule

$$x(k+1) = b(u(k)) \cdot a(x(k)); \qquad y(k) = c(x(k)).$$

This specializes to Horner's rule by letting b and c be trivial, letting $X = (R, +)$ and letting $a: R \to R$ be given by $a(x) = \lambda x$. Recall that through the recent efforts of Ostrowki, Motzkin, Pan, Garsia, etc., we now know that Horner's rule is optimal with respect to both the number of multiplications and additions required to evaluate a polynomial at a single point λ (see Knuth [5] for references).

We can also capture Horner's rule as a special case of a vector space recursion.

Case 2 (abstract Horner's rule—II): Let $u(0)$, $u(1)$, $u(2)$, $\cdots$ and p_0, p_1, p_2, $\cdots$ be two sets of real numbers. The product

$$\begin{bmatrix} p_0 & 0 & 0 & \cdots & 0 \\ p_1 & p_0 & 0 & \cdots & 0 \\ p_2 & p_1 & p_0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ p_p & p_{p-1} & p_{p-2} & \cdots & p_0 \end{bmatrix} \begin{bmatrix} u(0) \\ u(1) \\ u(2) \\ \vdots \\ u(p) \end{bmatrix} = \begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ \vdots \\ y(p) \end{bmatrix}$$

can be evaluated in $2\delta p - \delta^2 + \delta$ multiplications and $2\delta p - \delta^2 + \delta$ additions, where δ is the McMillan degree (see Sec. 2.3) of the sequence $\{p_i\}$, by using the first of the recursions given at the end of Sec. 2.3.

Through the use of nonrecursive techniques one can multiply a finite Toeplitz matrix of size p by p by a vector in $6p - 1$ multiplications (see corollary 15 of Fiduccia [12] and use the imbedding of a finite Toeplitz matrix into a circulant matrix of twice the size). The recursive scheme is better if the McMillan degree is low. However, there is room

for improvement in the recursive scheme if one uses more memory, as the following example shows.

To illustrate, consider the question of evaluating the first n terms in the product of a Toeplitz matrix and a vector:

$$\left[\begin{array}{cc:cc:c} a & 0 & 0 & 0 & \vdots \\ b & a & 0 & 0 & \\ \hdashline 0 & b & a & 0 & \vdots \\ 0 & 0 & b & a & \\ \hdashline \cdots & & \cdots & & \ddots \end{array}\right]\left[\begin{array}{c} u_1 \\ u_2 \\ \hdashline u_3 \\ u_4 \\ \hdashline \vdots \end{array}\right] = \left[\begin{array}{c} y_1 \\ y_2 \\ \hdashline y_3 \\ y_4 \\ \hdashline \vdots \end{array}\right].$$

The obvious iterative scheme to evaluate this requires $2n - 1$ multiplications to compute the first n terms, which is no better than naive multiplication. However, if this computation is "blocked" as indicated, then an appropriate iterative scheme is

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}$$

$$\begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} = \begin{bmatrix} a & 0 & b \\ b & a & 0 \end{bmatrix}\begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix}.$$

This requires four multiplications per step but if we want to reduce these we can observe that the transformation

$$\begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}$$

gives for CP^{-1},

$$\begin{bmatrix} a & 0 & b \\ b & a & 0 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a+b & 0 & b \\ a+b & a & 0 \end{bmatrix}.$$

Thus we can construct a realization which requires only three multiplications per step,

$$\begin{bmatrix} z_1(k+1) \\ z_2(k+1) \\ z_3(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}\begin{bmatrix} z_1(k) \\ z_2(k) \\ z_3(k) \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ -1 & 1 \\ -1 & 0 \end{bmatrix}\begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}$$

$$\begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} = \begin{bmatrix} a+b & 0 & b \\ a+b & a & 0 \end{bmatrix}\begin{bmatrix} z_1(k) \\ z_2(k) \\ z_3(k) \end{bmatrix}.$$

Even further reduction is possible by working with bigger blocks.

3.2 *Optimal root-finding.* We also mention that there is a growing body of literature which assesses the performance of comparable methods for root-finding. If r is the root of $f(x) = 0$ then the performance measure of an algorithm whose ith approximation is x_i might be a function of the power p where p is the largest number such that

$$\limsup_{i\to\infty} |x_{i+1} - r|/|x_i - r|^p < \infty .$$

Ostrowski [13] showed, for example, that for the secant method

$$x_{k+1} = -\left[\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}\right] f(x_k) + x_k$$

the power p is the larger root of $p^2 - p - 1 = 0$, which is about 1.62. Moreover, subsequently the secant method has been shown to be optimal with respect to a large class of algorithms which use the same data. For a survey of this literature see the paper by Traub [14].

4. Two examples. Two instances in which the kind of analysis being proposed here has been carried a little further along are the areas of numerical integration and iterative root-finding. We will briefly discuss some results.

4.1 *Numerical integration.* The wide variety of methods available for the numerical integration of the initial value problem

$$\dot{x}(t) = f[x(t)]; \qquad x(0) = x \tag{1}$$

are usually introduced by describing their implementation. Butcher [15] has proposed a scheme which gives a degree of unification to many diverse methods. An alternative was developed by Evans in his thesis [16–17].

In Evans' work one associates with each linear numerical integration method and each step size h an infinite lower triangular matrix W. If w_{ij} are the entries in this matrix then the approximating values to the solution of (1) are obtained by solving the system

$$\begin{bmatrix} g(0) \\ g(1) \\ g(2) \\ \vdots \end{bmatrix} + \begin{bmatrix} w_{11} & 0 & 0 & \cdots \\ w_{21} & w_{22} & 0 & \cdots \\ w_{31} & w_{32} & w_{33} & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix} \begin{bmatrix} f[y(0)] \\ f[y(1)] \\ f[y(2)] \\ \vdots \end{bmatrix} = \begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ \vdots \end{bmatrix} \tag{*}$$

for $y(k)$. In all interesting cases the approximate value of $x(hk)$ is $y(kp)$ where p is a positive integer. No significance is attached to the points which are not multiples of p. These are, for example, the intermediate points which appear in a Runge–Kutta method and which are ultimately discarded. Think of $1/p$ as the "rate" of the method, since p is the number of evaluations of f required to go one step forward. The vector g depends on the initial data.

If we are discussing a linear multistep method then W is a lower triangular Toeplitz matrix and p is one. The method is explicit if and only if W is strictly lower triangular. The Runge–Kutta and midpoint rules, as well as linear multistep methods, give rise to W matrices which are periodic in the sense that $w_{i+p,j+p} = w_{i,j}$. Necessary and sufficient conditions for consistency, stability, and convergence, in the sense that these terms are used in numerical analysis, can be found in terms of the matrix W. This is also done in Evans' thesis.

Since the lower triangular matrix W is generally full, the actual implementation of the algorithm requires one to solve the equation (*) by a recursive scheme. This means finding a blocking for W, say $W = (W^{ij})$, and then finding $A(k)$, $B(k)$, and $C(k)$ such that

$$W^{ij} = C(i)A(i) \cdots A(j)B(j), \qquad i \geq j.$$

Having this decomposition we can implement the algorithm as

$$x(k+1) = A(k)x(k) + B(k)\hat{f}(y(k)), \qquad y(k+1) = C(k)x(k) + D\hat{f}(y(k)).$$

System theory gives a procedure for finding the lowest-dimensional vector x to do this job. Moreover, one has procedures for finding all the realizations (triples A, B, C) which will work. Examples can be found in [16] and [17].

4.2 *Root-finding.* Picard [18] proposed a very general method for finding the solution of $f(x) = x$. This method is very close in spirit to feedback control theory in the following sense. Picard suggested that if one makes a guess x_0 and if $f(x_0) \neq x_0$ then a better guess is to let x_1 equal $f(x_0)$. Repeating the process gives $x_{k+1} = f(x_k)$. This is a feedback scheme in that the adjustment in x_k to get x_{k+1} is dependent on the error in the kth approximation $x_{k+1} - x_k = f(x_k) - x_k$. The availability of an easy sufficient condition for convergence, $||f(x) - f(y)|| < k\,||x - y||$ where $||\cdot||$ indicates a suitable norm and $k < 1$, makes this a very useful theoretical tool. However the speed of convergence is another matter, and in this respect Picard's method is far from ideal.

Adopting the point of view of the control engineer one might try to speed up the convergence rate by replacing Picard's algorithm by a higher-order one. That is, we design a dynamical system which will seek the null point more rapidly without becoming unstable. A possible generalization is to write

$$p(E)x(k) = f[q(E)x(k)]$$

where p and q are polynomials and E is the shift operator $E^n x(k) = x(n+k)$. Suppose f is continuously differentiable and that $f(x_s) = x_s$. For this general method consistency demands that $p(1) = q(1)$. This, plus the local stability condition asking that $p(E) + f'(x_s)q(E)$ has all its zeros in the open disk $\{z\colon |z| < 1\}$, insures that there exists $\epsilon > 0$ such that if we initialize the algorithms with

$$|x(-n) - x_s| + |x(-n+1) - x_s| + \cdots |x(-1) - x_s| + |x(0) - x_s| < \epsilon,$$

then the algorithm will converge.

What about convergence for arbitrary initializations? Thanks to some recent developments this question can also be answered at the expense of introducing additional machinery. Let Γ be defined by

$$\Gamma = \{z \mid z = q(e^{i\theta})/p(e^{i\theta});\ 0 \leq \theta \leq 2\pi\}.$$

This is the image of the unit circle in the complex plane under the mapping defined by $g = q/p$. It is possible to show that (i) if the local stability condition is satisfied; (ii) if $|f(x) - x_s| < |x - x_s|$ and if $\Gamma \subset \{z\colon |z| < 1\}$ then the convergence is global. There are many types of proofs and many extensions of this result. Some indication of the scope of the literature can be gained from the recent book of Willems [19]. Examples treating what these refinements yield in the way of improved convergence are given in the thesis of Gorille [20].

5. Closing remarks. Because of the occasion, I will conclude with a few nontechnical remarks.

The growth of any field depends on its ability to attract bright young people into the fold. Labels are useful only insofar as they help characterize the activity to those who are outside the field. The label "applied mathematics" is not always an asset. For example, in section 1 of [21] Edward Nelson defends his intent to discuss Brownian Motion as a physical concept as follows:

> The only legitimate justification is a mathematical one. Now 'applied mathematics' contributes nothing to mathematics. On the other hand, the sciences and technology do make vital contributions to mathematics. The ideas in analysis which had their origin in physics are so numerous and so central that analysis would be unrecognizable without them.

As far as the second sentence is concerned, the point is a good one. However there is a play on words here. One should not judge applied mathematicians by their contribution to mathematics any more than one judges historians by their contribution to history. Much creative effort goes into interpretation. To try to use mathematical ideas on problems which at first sight seem too complex or obscurely stated to fit any pattern is, for many people, an exciting challenge and it is this which I feel to be the essence of applied mathematics.

References

[1] R. Abraham, *Foundations of mechanics*, W. A. Benjamin, Inc., N. Y., 1967

[2] G. C. Newton, Jr., L. A. Gould and J. F. Kaiser, *Analytical design of linear feedback controls*, John Wiley and Sons, Inc., N. Y. 1957

[3] O. Heaviside, *Electrical Papers*, Vol. I and II, Macmillan and Co. 1892 (see Vol. II, pp. 353–374)

[4] Lord Rayleigh, *The reaction upon the driving-point of a system executing forced harmonic oscillations of various periods, with applications to electricity*, Phil. Mag. **21**, 369–381 (1886); *Scientific papers*, Vol. 2, pp. 475–485

[5] D. E. Knuth, *The Art of Computer Programming*, Vol. 2, Addison-Wesley, 1969

[6] A. M. Turing, *On computable numbers with application to the entscheidungs-problem*, Proc. London Math. Soc. **42**, 230–265 (1936–37)

[7] R. E. Kalman, P. Falb, and M. Arbib, *Topics in Mathematical System Theory*, McGraw-Hill, 1969

[8] R. W. Brockett, *Finite Dimensional Linear Systems*, John Wiley and Sons, N. Y., 1970

[9] A. Nerode, *Linear automaton transformation*, Proc. Amer. Math. Soc. **9**, 541–544 (1958)

[10] R. W. Brockett and A. Willisky, *Finite group-homomorphic sequential machines*, IEEE Trans. on Automatic Control (to appear)

[11] R. W. Brockett and Jacques Willems, *Least squares optimization for stationary linear partial difference equations*, in *Proc. IFAC Symposium on The Control of Distributed Parameter Systems*, Banff, Canada, 1971

[12] C. M. Fiduccia, *Fast matrix multiplication*, in *Proc. Third Annual ACM Symposium on Theory of Computing*, Shaker Heights, Ohio, May 3–5, 1971

[13] A. M. Ostrowski, *Solutions of equations and systems of equations*, Academic Press, N. Y., 1960

[14] J. F. Traub, *Optimal iterative processes: theorems and conjectures*, IFIP Congress Booklet TA-1 (1971), 29–32

[15] J. C. Butcher, *On the convergence of numerical solutions to ordinary differential equations*, *Math. Comp.* **20**, no. 93, (1966)

[16] D. S. Evans, *Finite-dimensional realizations of discrete-time weighting patterns*, SIAM J. App. Math. **22**, (1972)

[17] D. S. Evans, *Generalized linear multistep methods: a weighting pattern approach to numerical integration*, Ph.D. Thesis, M.I.T. 1969

[18] E. Picard, *Memoire sur la théorie des equations aux derivées partielles et la méthode des approximations successives*, J. Math. Pures Appl. (5)**6**, 423–441 (II 1) (1890)

[19] Jan Willems, *The analysis of feedback systems*, M.I.T. Press, Research Monograph No. 62, 1971

[20] I. J. Gorille, *On the application of discrete time stability criteria to numerical analysis*, M.S. Thesis, Dept. of Electrical Engineering, M.I.T., 1966

[21] E. Nelson, *Dynamical theories of Brownian motion*, Mathematical Notes, Princeton University Press, 1967