

BEYOND MASSIVE PARALLELISM: NUMERICAL COMPUTATION USING ASSOCIATIVE TABLES*

CRAIG C. DOUGLAS[†] AND WILLARD L. MIRANKER[†]

Abstract. Novel computing devices are exploited for numerical computation. The solution of a numerical problem is sought, which has been solved many times before, but this time with a different set of input data. A table is a classical way to collect the old solutions in order to exploit them to find the new one. This process is extended to more general problems than the usual function value approximation. To do this, a new concept of table is introduced. These tables are addressed associatively. Several problems are treated both theoretically and computationally. These problems include solving linear systems of equations, partial differential equations, nonlinear systems of ordinary differential equations, and Karmarkar's algorithm. Hardware requirements are discussed.

AMS(MOS) subject classifications. 65W05, 65F10, 65N10

Key words. Associative memory, linear systems of equations, ordinary and partial differential equations, Karmarkar's algorithm, table methods

1. Introduction. Among the new parallel computing models are analog devices and related connectionist architectures (neural nets, associative memories, connection machines, \dots , see [19]). Some examples where these models are used include cognitive problems (e.g., pattern matching or language processing), combinatorial problems (e.g., traveling salesman), organ simulations (e.g., retina), etc. (see [15]). The vitality and fecundity of this new parallel computing framework is impressive from all of the obvious points of view (see [1]).

Massive parallelism usually refers to the case when some thousands or a few million digital processors are used in a computation. The new models can be applied when billions (or more) of analog components are available. This type of parallelism goes beyond the usual scope of massive parallelism.

In this paper, we show how numerical analysis and the new models can interact. Ours is not likely to be the only approach (see [3]), but the results presented here point to a rich and varied set of future developments within the framework of this massively parallel computing model.

Our results have a speculative character, since we deal with numerical computation on a connectionist (analog) parallel model and a putative class of hardware implementations. However, the new numerical methods which we develop have a basis outside of the new model, that is, as new methods within the traditional framework of numerical computation. The basis is both conceptual and pragmatic, although not necessarily at the same time in each of our cases. Thus, although framed within the new model concept, our results are not necessarily bound to it as scientific contributions.

The style of the numerical methods of this paper can be motivated by an analogy. Many numerical methods are based on local approximations, such as point interpolation and related expansion methods. We replace these local techniques with nonlocal ones, in particular, least squares. Since many least squares methods exist, a large number of results wait to be adapted, invented, or extended into the new computational frame.

* Based on IBM Research Report RC 14660 (last revised May 10, 1990).

[†] Mathematical Sciences Department, IBM Research Division, Thomas J. Watson Research Center, P. O. Box 218, Yorktown Heights, New York 10598, USA. FAX 914-945-3434.

In order for these methods to be useful in practice, certain hardware must become available in effective scale and performance. The largest associative memory device we are aware of has 10^{10} bits with a cycle time of 10^{-4} seconds [17]. We comment on the hardware environment in §6.

The new model and hardware as applied customarily to higher order (e.g., cognitive) problems are often described as extensions of the traditional construction and use of tables in numerical computations. That is, input-output information which characterizes a problem to be solved is collected, saved, and exploited by making lookups. We use this table picture as motivation for our numerical work.

Tables have long been used as an aid to computation. They are exploited in modern day computers in schemes for efficient function value approximation, e.g., $\sin x$. The use of a table involves three typical steps: (i) preprocessing, such as normalizing the argument to a canonical interval, (ii) the table access which furnishes neighboring (argument, function) value pairs, and (iii) post processing, such as interpolation to approximate the desired value. Of course, these steps require that the function which is being dealt with have some structure, usually differentiability, so that the use of a table is possible and efficient.

Can tables be used for problems more general than function value approximation? Indeed, such use of tables is known. In some floating point computers (e.g., Floating Point Systems machines) the determination of the quotient $q = x/y$, that is the solution of the equation

$$f(q) = 0,$$

where $f(q) = yq - x$ is performed by table lookup. The table is accessed for an approximate value of the reciprocal, say $1/\tilde{y}$, and the product $x(1/\tilde{y})$ is used as a starting guess for q . The post processing consists of a very small number of Newton iterations for solving $f(q) = 0$.

In this paper we show how to prepare and exploit tables to solve a variety of problems. An earlier version of this approach dealt with the problem of sorting[8]. Here, we show how to use tables to solve linear systems of equations in §2 and systems of differential equations in §3 and §5. We also show how Karmarkar's algorithm may be accelerated by our methods in §4. In §6, we summarize the hardware requirements of the methods of this paper and comment on current hardware technology.

Information about a problem is collected in a type of memory (table) called an associative memory [13, 14]. The model of table lookup proceeds by association. An access to this memory by a key produces an output by association. The numerical methods used here are of interest in their own right. For example, in the case of linear systems, we borrow from an existing body of work dealing with the acceleration of convergence of vector sequences[20]. However, when the memory mechanism is implemented in hardware which provides this storage and associative retrieval both rapidly and efficiently, our methods have the potential of being superior to conventional algorithms.

An associative memory is an analog device with the well known advantage of great speed, but limited accuracy. Analog devices can be simulated in digital VLSI hardware[15] with a loss of some speed, but increased accuracy. We stress that even fully digital floating point computations have limited accuracy, usually graded by the problem's condition. The method of residual correction is the standard technique for addressing limited accuracy for digital devices. It is also available for analog devices, including the associative memory we use here. In §2 we show how the relevant

correction process may be done conveniently by a bootstrapping technique; repeating the same inferential techniques with small data changes.

Let us then consider the associative memory paradigm[14].

Associative Memory

Let $y_i, x_i \in \mathfrak{R}^N$ be given column vectors, $i = 1, \dots, k$. The pairs (y_i, x_i) are called (signal, key) pairs. We seek an $N \times N$ matrix M such that

$$y_i = Mx_i, \quad i = 1, \dots, k$$

in the sense of least squares. Compose two $N \times k$ matrices X and Y as follows:

$$X = [x_1, \dots, x_k]$$

$$Y = [y_1, \dots, y_k].$$

Then the least squares problem may be formulated as follows:

$$\min_M \|MX - Y\|^2 \equiv \min_M \text{Tr} (MX - Y)^T (MX - Y) = \min_M \sum_{i=1}^k \|Mx_i - y_i\|^2.$$

Here the norm of a vector is the Euclidean norm.

The solution of this minimization problem follows from the identity

$$(1.1) \quad \begin{aligned} (MX - Y)(MX - Y)^T &= (YX^+ - M)XX^T(YX^+ - M)^T + \\ &Y(I - X^+X)Y^T, \end{aligned}$$

where

$$(1.2) \quad X^+ = X^T(XX^T)^{-1}$$

denotes the pseudoinverse of X (see [14]). Indeed, the M -dependent term in the right member of (1.1) is a square which vanishes for

$$M = YX^+,$$

and this specifies the solution.

We note on a conventional digital computer, the cost of computing (1.2) is normally prohibitive. One way of reducing the cost somewhat is to use Greville's Theorem (see [7]). This is an $O(N^2)$ work per step (maximum N iterations) iterative algorithm for computing the pseudoinverse.

Now imagine that the (signal, key) pairs are stored in a hardware device which, when accessed by a key x , furnishes the output signal y , where

$$y = Mx.$$

y is the same least squares fit to the set of stored signals that x is to the set of stored keys. Any device which provides this function is called an *associative memory*. Indeed y is referred to as an *inference* made about x given the information (the (signal, key) pairs) in the memory.

Here we take the point of view that the memory is a table (an *associative table*) and the table lookup, which is provided automatically by the hardware, is modeled

by the multiplication of the access vector x by the matrix M . Since no confusion will occur, we will use M to denote the memory or table as well as the matrix which models the access process.

In Control Theory, the least squares problem is generalized in a number of ways, such as by adding a small random perturbation to the problem (i.e., by introducing noise). A large class of solution procedures for such generalizations is commonly named Kalman filtering[11]. This theory can have relevance to the approach here when input-output errors or rounding errors are added to the signals and keys. Since Kalman filters are extensively modeled, both synthetically and by special dedicated circuitry, this connection can have a significant pragmatic impact on our work.

2. Inferential Solve.

2.1. The Inferential Method. Let A be an $N \times N$ matrix and let b be an N vector. The linear equation

$$(2.1) \quad s = As + b$$

has a unique solution if 1 is not an eigenvalue of A . We assume this, and we seek to determine s by producing the sequence of iterates using Richardson's method,

$$x_{j+1} = Ax_j + b, \quad j = 0, 1, \dots,$$

where x_0 is prescribed. Note that A and b need not be known. We need the iterates x_j only. Now

$$\lim_{j \rightarrow \infty} x_j = s$$

for any choice of x_0 , if and only if the spectral radius $\rho(A) < 1$. Henceforth, we assume this to be the case. Richardson's method is often very slowly convergent, and is seldom used without modification[23, 24].

We take the view that the successive iterates provide information about the solution sought. We intend to store this information in an associative memory, in such a manner that inquiries addressed to the memory will produce information which accelerates the determination of s . In fact, methods exist which specify s in terms of the first $k + 1$ iterates $\{x_0, x_1, \dots, x_k\}$ where the integer $k \leq N$ is well defined by A and x_0 . Such a method[20], called reduced rank extrapolation (RRE) is a vectorized version of Aitken's Δ^2 -method. Indeed an associative memory provides a method for assembling these iterates (i.e., the information) for implementing the accelerative process and for determining an effective approximation to k .

Reduced Rank Extrapolation

For $j = 0, 1, \dots$, let

$$u_j = \Delta x_j = x_{j+1} - x_j$$

$$v_j = \Delta^2 x_j = \Delta u_j = u_{j+1} - u_j = x_{j+2} - 2x_{j+1} + x_j.$$

Note for further reference, that

$$(2.2) \quad u_j = (A - I)(x_j - s),$$

$$(2.3) \quad u_{j+1} = Au_j,$$

$$(2.4) \quad v_j = (A - I) u_j,$$

for each $j \geq 0$.

For $i = 1, 2, \dots$, let

$$(2.5) \quad U_i = [u_0, u_1, \dots, u_{i-1}],$$

$$(2.6) \quad V_i = [v_0, v_1, \dots, v_{i-1}].$$

That is U_i and V_i are $N \times i$ matrices composed by assembling the indicated u_j and v_j by columns.

DEFINITION 2.1. *The minimal polynomial $P(\lambda)$ of A with respect to a vector z is the unique monic polynomial of least degree such that*

$$P(A)z = 0.$$

Hereafter we take z to be u_0 in this definition.

Let k denote the degree of $P(\lambda)$, $k \leq N$, and write

$$P(\lambda) = \sum_{j=0}^k c_j \lambda^j, \quad c_k = 1.$$

Note that $P(A)$ is a factor of the characteristic polynomial of A . Then since unity is not an eigenvalue of A ($\rho(A) < 1$),

$$P(1) = \sum_{j=0}^k c_j \neq 0.$$

Using (2.3) and (2.5) we have

$$(2.7) \quad U \equiv U_k = [u_0, Au_0, \dots, A^{k-1}u_0].$$

Let $c = (c_0, \dots, c_{k-1})^T$ denote the unknown coefficients of $P(\lambda)$. Then using (2.7), we get

$$(2.8) \quad Uc = -u_k.$$

By definition, k is maximal with respect to the property that the columns of U are linearly independent. Indeed, in this case, the pseudoinverse U^+ of U exists and

$$U^+ = (U^T U)^{-1} U^T.$$

Using (2.4), we see that the pseudoinverse of V exists, and

$$(2.9) \quad V^+ = U^+(A - I)^{-1}.$$

Consider the following theorem whose proof may be found in [20]:

THEOREM 2.1. *For any $k + 1$ consecutive terms of the sequence, say $x_m, x_{m+1}, \dots, x_{m+k}$, we have*

$$\sum_{j=0}^k c_j x_{m+j} = \left(\sum_{j=0}^k c_j \right) s.$$

Now let $U \equiv U_N$ and $V \equiv V_N$. If V is invertible, then using (2.3) and (2.4), we have

$$(I - A)^{-1} = -UV^{-1}.$$

Combining this with (2.2) for $j = 0$, we have

$$(2.10) \quad s = x_0 - UV^{-1}u_0.$$

In the case that U and V in (2.10) are replaced by U_k and V_k , respectively, we have the following reduced rank equivalent of (2.10):

$$(2.11) \quad s = x_0 - UV^+u_0.$$

For a proof of this reduced rank equivalent, (2.5) – (2.6), see [20].

Note for future reference that by combining (2.9) and (2.11), we get

$$(2.12) \quad s = x_0 - UU^+(A - I)^{-1}u_0.$$

From this, in turn,

$$(2.13) \quad (A - I)^{-1}u_0 = UU^+(x_0 - s).$$

The subscript value k is not annotated in (2.11) – (2.13). For smaller values of the subscript, these equations are not expected to be valid. However, by the maximality property of k , the expressions U^+ and V^+ appearing in these equations are well defined. We shall make use of these observations later on.

Aitkens Δ^2 -method

We may note a formal correspondence between (2.11) and Aitken's Δ^2 -method for accelerating convergence. Indeed, writing the first and second difference matrices U and V as ΔX and $\Delta^2 X$, respectively, (2.10) may be written as

$$s = x_0 - \Delta X(\Delta^2 X)^+ \Delta x_0.$$

Associative implementation of Reduced Rank Extrapolation

To implement reduced rank extrapolation by means of an associative memory, we regard the pairs (u_i, v_i) as (signal, key) pairs which are stored successively in that memory as they are produced, $i = 0, 1, \dots$. After each such memory entry, the associative memory paradigm characterizes the memory M_i in terms of a matrix M_i , where

$$M_i = U_i V_i^+, \quad i = 1, 2, \dots$$

By the maximality property of k , note that V_i^+ exists for all $i \leq k$. After each write into memory, the query with u_0 is made with the resulting output

$$M_i u_0, \quad i = 1, 2, \dots$$

The quantity

$$(2.14) \quad s_i = x_0 - M_i u_0$$

is computed, and a tolerance test is applied, e.g., for some prescribed $\epsilon > 0$, we check the inequality

$$\|s_i - s_{i-1}\| \leq \epsilon, \quad i = 1, 2, \dots$$

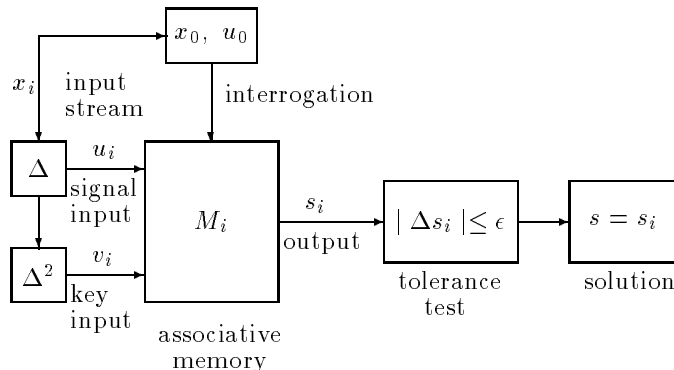


FIG. 1. *Associative memory implementation*

When this test passes, the current value of i is taken to be k , and s_k is taken to be s . This procedure is illustrated in Fig. 2.1. The output s_k of this algorithm should be checked through residual computation:

$$r(s_k) = (A - I)s_k + b,$$

and improved, if necessary, by residual correction. In the numerical experiments of §2.2, we show how the reduced rank extrapolation process may be bootstrapped to provide the corrections.

2.2. Numerical Experiments. In this section, we discuss numerical experiments for inferential solve. We apply this technique to two classes of model problems: discretizations of Poisson's equation on a square and random symmetric, positive definite matrices.

We simulated the associative memory paradigm using a workstation, programmed in Matlab[16]. The inferences, s_i , are calculated using (2.14). We chose Matlab since one of its arithmetic operators is the pseudoinverse of a matrix.

Let

$$r_0 = b - (I - A)x_0,$$

and

$$(2.15) \quad r_i = b - (I - A)s_i, \quad i = 1, 2, \dots$$

Throughout this section, we graph a scaled residual norm,

$$\frac{\|r_i\|}{\|r_0\|} \text{ versus } i,$$

where $\|\cdot\|$ is the Euclidean norm.

In each of our numerical experiments we observe that

$$\|r_i\| \rightarrow r,$$

where r is a positive constant depending on the data in the experiment. This effect, a type of instability, is an artifact of the computational precision used. If this r is not

sufficiently small, we stop the inferencing with some s_j , clear the associative memory, and restart with

$$(2.16) \quad x_0 = s_j.$$

We then apply inferential solve to the residual correction problem

$$c_i = Ac_{i-1} + r_j, \quad i = 1, \dots, j,$$

and set

$$(2.17) \quad s_{j+1} = s_j + c_j.$$

If s_{j+1} is not adequate, then we repeat the residual correction calculation (2.16) – (2.17) substituting $j + 1$ for j to get s_{j+2} (and so on).

First, consider the two dimensional Poisson equation

$$(2.18) \quad \begin{cases} -\Delta u = b & \text{in } \Omega = (0, 1)^2 \\ u = 0 & \text{on } \partial\Omega \end{cases}.$$

We discretize (2.18) using either central differences on an $N \times N$ uniform mesh or finite elements on a uniform triangulation using C^0 piecewise linear polynomials and the usual nodal basis functions. In either case, after discretization, we get an $N^2 \times N^2$ matrix B_P (the subscript P , as in *Poisson*, is used here so that these particular matrices may be referenced later without confusion) which is block tridiagonal (see [22, 23]):

$$B_P = [-I, Q, -I],$$

where $Q = [-1, 4, -1]$ is an $N \times N$ tridiagonal matrix. Then the matrix A_P corresponding to A in (2.1) for the Jacobi method is given by

$$A_P = I - \frac{1}{8}B_P.$$

In our experiments, we used $N = 10$, and both a discrete right hand side b and an initial guess x_0 whose elements were randomly selected from the open interval $(0, 1)$. Figure 2.2(a) contains the scaled residual norm for a typical case. Figure 2.2(b) contains the scaled residual norm for this example when we applied residual correction every 25 inferences.

As an aside, we modified our Matlab program for inferential solve to accept vectors x_i , which were generated by three conjugate direction methods: conjugate gradients, conjugate gradients preconditioned by SSOR, and conjugate residuals preconditioned with SSOR (see [24]). While there is more computational work associated with these iterative methods[2], the total number of inferences is reduced enough to offset this extra work. We used the same x_0 and b as used in Figure 2.2. Figure 2.3 contains the scaled residual norms for the same case as used in Figure 2.2. As is readily seen in Figure 2.3(a) (and is expected), the conjugate gradient algorithm does not do as well as the preconditioned conjugate gradient or conjugate residual methods. However, (as is readily seen in Figure 2.3(b)), all three conjugate direction methods produce input vectors $\{x_i\}$ for use by the inferential solve process that are equivalent to each other, and better than the Jacobi iterations used in Figure 2.2(a).

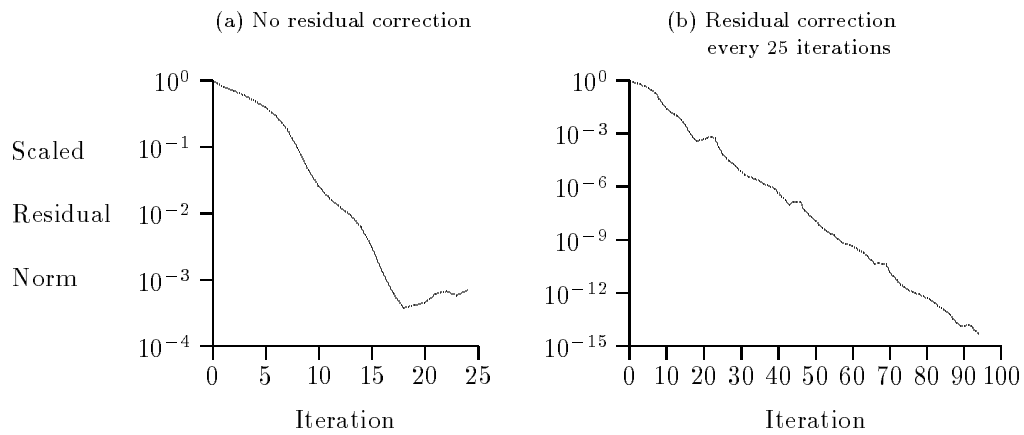
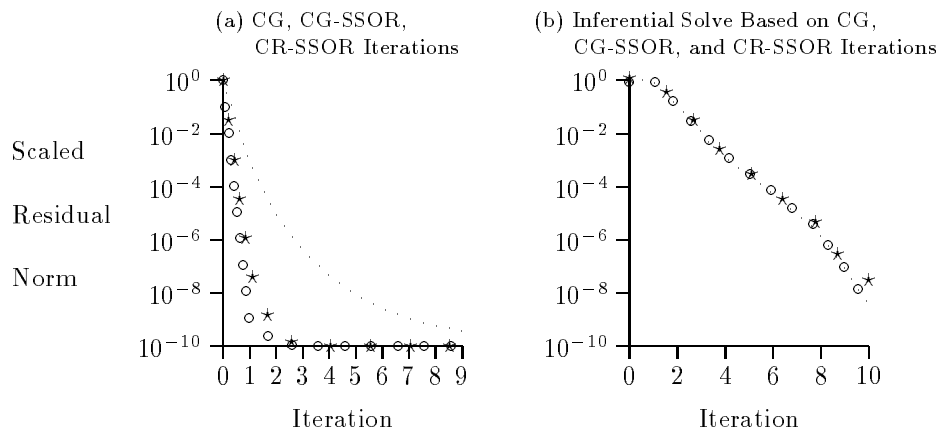


FIG. 2. *Inferential solve Poisson data*



CG: dotted line, CG-SSOR: o line, CR-SSOR: * line

FIG. 3. *Inferential Solve Poisson Data - CG Iterations*

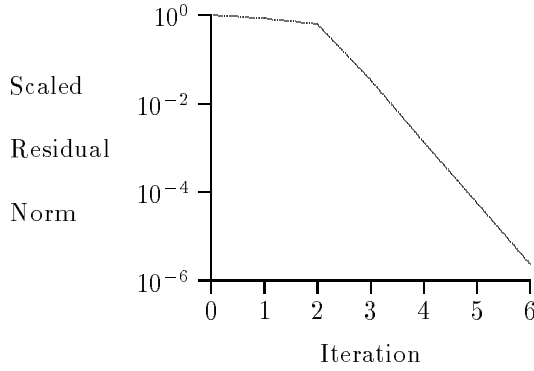


FIG. 4. *Inferential Solve Random Data*

We also experimented (see [4]) with modifying B_P so that its eigenvalue structure was clustered. Results similar to Fig. 2.2 were observed.

Finally, we constructed real, dense, symmetric, positive definite matrices A_r starting from dense matrices with random elements. As before, we used a discrete right hand side b and an initial guess x_0 whose elements were randomly selected from the open interval $(0, 1)$. Figure 2.4 contains the scaled residual norm for a typical case when the largest eigenvalue of A_r is .99. Runs of similar quality for inferential solve were observed when the largest eigenvalue was in the interval $[.9, .99]$.

2.3. Stability Analysis. In this section, we prove that the method is stable. In actual computation, the sequence x_j , $j = 0, 1, \dots$, which is used will be a perturbation of the theoretical sequence. Here we characterize the stability of the table look up solution process to such perturbations.

Let

$$\bar{x}_j = x_j + \epsilon_j, \quad j = 0, 1, \dots$$

denote the actual sequence produced, where in particular

$$\bar{x}_{j+1} = A\bar{x}_j + b + f_{j+1}, \quad j = 0, 1, \dots$$

The vectors f_1, f_2, \dots may be viewed as resulting from roundoff errors. Combining, we have

$$\epsilon_{j+1} = A\epsilon_j + f_{j+1}, \quad j = 0, 1, \dots$$

Define $f_0 \equiv \epsilon_0$. We expect that $\epsilon_0 = 0$ in most cases, but for completeness, we do not set ϵ_0 to zero in the following discussion. The f_0, f_1, \dots are independent perturbations with the ϵ_j , $j = 1, 2, \dots$ being dependent upon them.

Corresponding to the \bar{x}_j , we have for $j = 0, 1, \dots$,

$$\bar{u}_j = \bar{x}_{j+1} - \bar{x}_j = u_j + \epsilon_{j+1} - \epsilon_j$$

and

$$\bar{v}_j = \bar{u}_{j+1} - \bar{u}_j = v_j + \epsilon_{j+2} - 2\epsilon_{j+1} + \epsilon_j.$$

Note that

$$\bar{v}_j = (A - I)\bar{u}_j + f_{j+2} - f_{j+1}, \quad j = 0, 1, \dots$$

Let

$$\bar{U} \equiv [\bar{u}_0, \dots, \bar{u}_{k-1}] \equiv [\dots, \bar{u}_j, \dots]_{j=0}^{k-1}.$$

Then

$$\bar{U} = [\dots, u_j + \epsilon_{j+1} - \epsilon_j, \dots]_{j=0}^{k-1},$$

and correspondingly,

$$\begin{aligned} \bar{V} &= [\dots, (A - I)(u_j + \epsilon_{j+1} - \epsilon_j) + f_{j+2} - f_{j+1}, \dots]_{j=0}^{k-1}, \\ &= (A - I)[\dots, u_j + \epsilon_{j+1} - \epsilon_j + (A - I)^{-1}(f_{j+2} - f_{j+1}), \dots]_{j=0}^{k-1}. \end{aligned}$$

Rewrite \bar{U} and \bar{V} as follows:

$$\bar{U} \equiv U + E,$$

and

$$\bar{V} \equiv (A - I)(U + F).$$

Let \bar{s} be the inferred solution delivered by the perturbation process.

$$\begin{aligned} \bar{s} &\equiv \bar{x}_0 - \bar{U}\bar{V}^+\bar{u}_0 \\ &= x_0 + \epsilon_0 - (U + E)(U + F)^+(A - I)^{-1}(u_0 + \epsilon_1 - \epsilon_0). \end{aligned}$$

Let \bar{k} be the degree of the minimal polynomial in A with respect to \bar{u}_0 . Then there is a tacit assumption here that $\bar{k} = k$. $A\epsilon_0 = 0$ is a sufficient condition for this, since it implies that $\bar{u}_0 = u_0$. Combining this with (2.11), we have

$$\begin{aligned} \bar{s} - s &= \epsilon_0 + U[U^+ - (U + F)^+](A - I)^{-1}u_0 \\ &\quad - E(U + F)^+(A - I)^{-1}u_0 \\ &\quad - (U + E)(U + F)^+(A - I)^{-1}(\epsilon_1 - \epsilon_0) \\ &\equiv \epsilon_0 + T_1 + T_2 + T_3. \end{aligned}$$

Note that

$$(A - I)^{-1}(\epsilon_1 - \epsilon_0) = \epsilon_0 + (A - I)^{-1}f_1,$$

so that

$$T_3 = -(U + E)(U + F)^+(\epsilon_0 + (A - I)^{-1}f_1).$$

To estimate T_1 , we use Theorem 2.2 [21, page 223] for characterizing perturbations in least squares problems with the normal equations

$$Cx = d.$$

In Theorem 2.2, R will be a perturbation of C and the subscripts 1 and 2 denote projections onto the ranges $\mathcal{R}(A)$ and $\mathcal{R}(A)^\perp$, respectively.

THEOREM 2.2. *Let the columns of C be linearly independent. If $\|C^+\| \|R_1\| < \frac{1}{2}$, then the columns of $C + R$ are linearly independent. Moreover if $x = C^+d$, $\bar{x} = (C + R)^+d$, and κ denotes the generalized condition number of C , $\kappa = \|C\| \|C^+\|$, then*

$$\frac{\|x - \bar{x}\|}{\|x\|} \leq 2\kappa \frac{\|R_1\|}{\|C\|} + 4\kappa^2 \frac{\|R_2\| \|d_2\|}{\|C\| \|d_1\|} + 8\kappa^3 \frac{\|R_2\|^2}{\|C\|^2}.$$

To apply Theorem 2.2 to T_1 , let

$$C = U,$$

$$R = F,$$

$$d = (A - I)^{-1}u_0,$$

$$x = U^+(A - I)^{-1}u_0,$$

and

$$\bar{x} = (U + F)^+(A - I)^{-1}u_0.$$

Theorem 2.2 requires that $\|F_1\|$ be sufficiently small in order that

$$\|U^+\| \|F_1\| < \frac{1}{2}.$$

Note first from (2.13) that $(A - I)^{-1}u_0 \in \mathcal{R}(U)$. That is, $d_2 = 0$, so that the second term in the right member of the estimate of the theorem vanishes. Now applying the theorem, we have

$$\frac{\|T_1\|}{\|U^+(A - I)^{-1}u_0\|} \leq \|U\| \left[2\kappa \frac{\|F_1\|}{\|U\|} + 8\kappa^3 \frac{\|F_2\|^2}{\|U\|^2} \right].$$

Using (2.12), the left member here becomes $\|T_1\|/\|U^+(s - x_0)\|$. Thus,

$$\|T_1\| \leq \kappa \|s - x_0\| \left[2\kappa \frac{\|F_1\|}{\|U\|} + 8\kappa^3 \frac{\|F_2\|^2}{\|U\|^2} \right].$$

The norms of T_2 and T_3 are estimated directly:

$$\|T_2\| \leq \|E\| \|(U + F)^+\| \|(A - I)^{-1}u_0\|,$$

and

$$\|T_3\| \leq \|U + E\| \|(U + F)^+\| (\|\epsilon_0\| + \|(A - I)^{-1}f_1\|).$$

Collecting terms, we have

$$\begin{aligned}
\|\bar{s} - s\| &\leq [1 + \|U + E\| \|(U + F)^+\|] \|\epsilon_0\| \\
&\quad + \kappa^2 \|s - x_0\| \left[2 \frac{\|F_1\|}{\|U\|} + 8\kappa^2 \frac{\|F_2\|^2}{\|U\|^2} \right] \\
&\quad + \|(U + F)^+\| [\|E\| \|(A - I)^{-1}u_0\| + \|U + E\| \|(A - I)^{-1}f_1\|], \\
&= O(\|F\|).
\end{aligned}$$

This estimate completes the stability analysis. The ideal solution s and the delivered one \bar{s} differ by a perturbation. Hence, small $\|F\|$ (i.e., more bits of precision) will result in more accurate solutions.

3. Inferential Extrapolation. There are applications where a system of linear equations is to be solved repeatedly, each time with a different forcing vector b . Associative tables allow us to accumulate the information presented by such a collection of solutions, so that when the new b is available, we may access the table to produce as an inference a good choice for the new solution.

Models for this case arise when initial-boundary value problems are solved for partial differential equations. Consider the heat equation as a prototype:

$$w_t = w_{xx}, \quad t > 0, \quad 0 < x < 1,$$

with prescribed initial condition and boundary conditions, e.g.,

$$w(x, 0) = 0, \quad w(0, t) = a(t), \quad w(1, t) = b(t).$$

Take mesh increments h and k in x and t , respectively, with h^{-1} an integer. Let $x_i = ih$, $t_j = jh$, and consider the mesh (x_j, t_j) , $i = 0, 1, \dots, h^{-1}$, $j = 0, 1, \dots$. Let $W_{ij} = W(ih, jk)$ be a numerical approximation to $w(ih, jk)$, where W_{ij} is defined by a two level implicit difference scheme. For example,

$$\begin{aligned}
\frac{W_{i,j+1} - W_{i,j}}{h} &= \theta \frac{W_{i-1,j+1} - 2W_{i,j+1} + W_{i+1,j+1}}{k^2} \\
&\quad + (1 - \theta) \frac{W_{i-1,j} - 2W_{i,j} + W_{i+1,j}}{k^2},
\end{aligned}$$

$i = 1, 2, \dots, h^{-1} - 1$, and $j = 1, 2, \dots$, where $\theta = 1$ is the Backward Euler scheme and $\theta = .5$ is the Crank-Nicolson scheme (see [18]).

Taking $\theta = .5$ and writing

$$W_j = (W_{1,j}, W_{2,j}, \dots, W_{h^{-1}-1,j})^T,$$

the difference equation at the k th time level leads to the following h^{-1} -dimensional linear system for W_j :

$$(3.1) \quad BW_j = CW_{j-1} + \beta_j,$$

where

$$2k^2\beta_j = (a(t_j) + a(t_{j-1}), 0, \dots, 0, b(t_j) + b(t_{j-1}))^T.$$

This linear system is to be solved for W_j , $j = 1, 2, \dots$ where W_0 is the zero vector. Having derived this model, let us turn to the general problem.

3.1. The Extrapolation Problem. Let x_j , $j = 0, 1, \dots$ be a sequence of N -vectors defined by the recurrence

$$(3.2) \quad Bx_{j+1} = Cx_j + \beta, \quad j = 0, 1, \dots,$$

with x_0 given. B and C are $N \times N$ matrices with B nonsingular. Then setting $A = B^{-1}C$ and $b = B^{-1}\beta$, the recurrence becomes the same considered in Section 2:

$$x_{j+1} = Ax_j + b.$$

We note that A and b are not explicitly known (since B^{-1} is not to be determined). Suppose that the system (3.2) has been solved for a successive number of values of j , say x_1, x_2, \dots, x_{k+1} . We seek to store this information in an associative table and to access the table to produce (an approximation to) x_{k+2} , and then to repeat.

Recall that k is the degree of the minimal polynomial in A with respect to $u_0 = x_1 - x_0$, and that $c = (c_0, \dots, c_{k-1})^T$ is the vector composed of the unknown coefficients of that minimal polynomial. Since the columns of U_k are linearly independent, U_k^+ exists, and (2.8) gives

$$(3.3) \quad c = -U_k^+ u_k.$$

Now if zero is not an eigenvalue of A , which we assume, then $P(\lambda)$ is the minimal polynomial of A with respect to u_j , $j = 1, 2, \dots$ (for a proof, see[20]). Now define the matrices $U_{k,j}$ columnwise as follows:

$$U_{k,j} = [u_j, u_{j+1}, \dots, u_{j+k-1}], \quad j = 0, 1, \dots$$

Then (3.3) may be generalized to yield

$$-u_{k+j} = U_{k,j} c, \quad j = 0, 1, \dots,$$

where we stress that c is j -independent.

Using this relation for $j = 1$, combining it with (3.3), and defining $U_{k,0} \equiv U_k$, we obtain

$$u_{k+1} = U_{k,1} U_{k,0}^+ u_k.$$

Setting $M_{k,j} = U_{k,j} U_{k,j-1}^+$, this relation gives the next iterate as

$$x_{k+2} = x_{k+1} + M_{k,1} u_k.$$

The memory $M_{k,1}$ is composed of the k (signal, key) pairs

$$(u_{j+1}, u_j), \quad j = 0, 1, \dots, k-1.$$

We now delete the oldest pair (u_1, u_0) from the memory, input the current pair (u_{k+1}, u_k) , and repeat. This process is defined by the following recurrence:

$$x_{k+j+1} = x_{k+j} + M_{k,j} u_{k+j-1}, \quad j = 1, 2, \dots$$

The output of this memory reference could be refined by a bootstrapping process (e.g., Greville's Theorem [7]) or possibly a residual correction process before the new signal u_{k+1} is computed for memory update.

3.2. Numerical Experiments. In this section, we discuss numerical experiments for the inferential extrapolation method. We apply the techniques of this section to two parabolic problems (one and two dimensional heat flow problems) and one hyperbolic problem (a two dimensional transport problem).

We define the scaled error by

$$e_i \equiv \frac{\|W_i - x_i\|}{\|W_i\|}, \quad i = 3, 4, 5, \dots,$$

where $\|\cdot\|$ is the Euclidean norm. Throughout this section, we graph the scaled error, e_i , versus i .

Since we do not know k , the degree of the minimal polynomial in A with respect to $u_0 = x_1 - x_0$, we are required to use a two stage approach to computing inferences. First we determine an acceptable approximation $k(\epsilon)$ to k . This is a critical step in the inferential extrapolation technique since a poor approximation to the real value of k (either too large or too small) leads to $e_i \rightarrow \infty$.

There are many simple heuristics to approximate the real value of k . The simplest heuristic is to choose $k(\epsilon)$ a priori. Another heuristic is to choose some tolerance $\epsilon > 0$. Then define $k(\epsilon)$ to be the smallest integer greater than 2 (the minimum number of time steps needed to start the inferential extrapolation process) such that

$$(3.4) \quad e_{k(\epsilon)} \leq \epsilon.$$

This requires actually computing (instead of inferencing) exact solutions $\{W_1, W_2, \dots\}$ using (3.1) and inferences $\{x_3, x_4, \dots\}$. Once $k(\epsilon)$ is determined, we take $W_i \equiv x_i$, $i > k(\epsilon)$. By occasionally computing the exact solution of (3.1), we can determine if (3.4) still holds.

Note: We assume that ϵ is chosen to be too small. In fact, we choose fixed $k \leq k(\epsilon)$ for use in our computations once $k(\epsilon)$ is determined. A slightly better approach would be to change k slightly during the computation as a function of time. We leave as an open question exactly how to construct heuristics to change k dynamically.

In each of our numerical experiments we observe that

$$e_i \nearrow \eta,$$

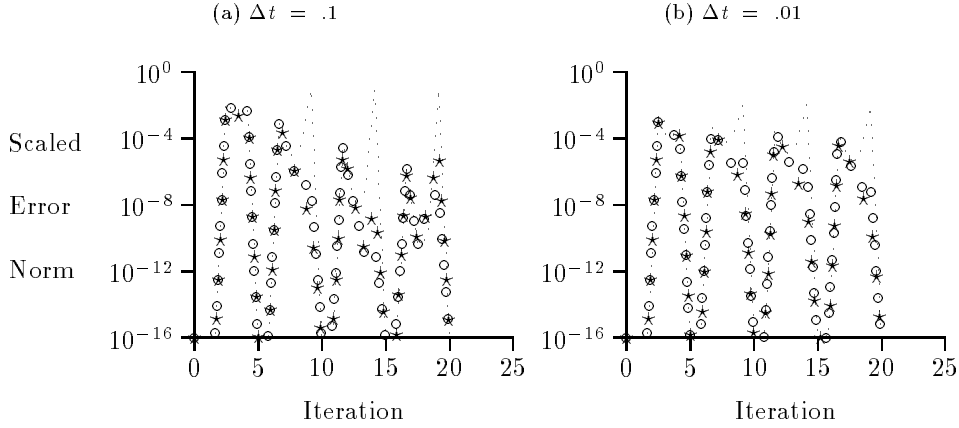
where η is a positive constant depending on the data in the experiment. If this η is not sufficiently small, we stop the inferencing with some x_j , clear the associative memory, and restart the procedure at that time step using a sequence of exact solutions, $\{W_{j+1}, W_{j+2}, \dots, W_{j+\max\{3, k\}}\}$, to (3.1). The second stage of our computation is embodied in this restarting cycle. We found empirically that if the size of ϵ_i was not too great, this restarting step (although based on on inferred values of lower quality) would reduce the error sufficiently so that inferencing could be resumed.

Our first example is a one dimensional heat flow problem on the spatial domain $\Omega = (0, \pi)$:

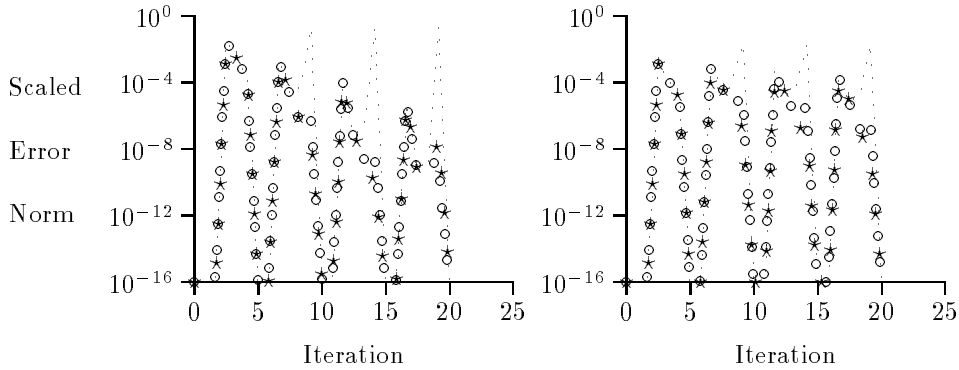
$$(3.5) \quad \begin{cases} w_t &= w_{yy} & \text{for } 0 \leq y \leq \pi, t \geq 0 \\ w(y, 0) &= \phi(y) & \text{for } 0 \leq y \leq \pi \\ w(y, t) &= 0 & \text{for } t > 0 \text{ and } y \in \partial\Omega, \end{cases}$$

where

$$\phi(y) = \begin{cases} y & \text{for } 0 \leq y \leq \pi/2 \\ \pi - y & \text{for } \pi/2 \leq y \leq \pi \end{cases}.$$



$k = 2$: dotted line, $k = 3$: o line, $k = 4$: x line
 (a) $\Delta t = .1$ (b) $\Delta t = .01$
 FIG. 5. *Inferential Extrapolation for One Dimensional Heat Problem Data*



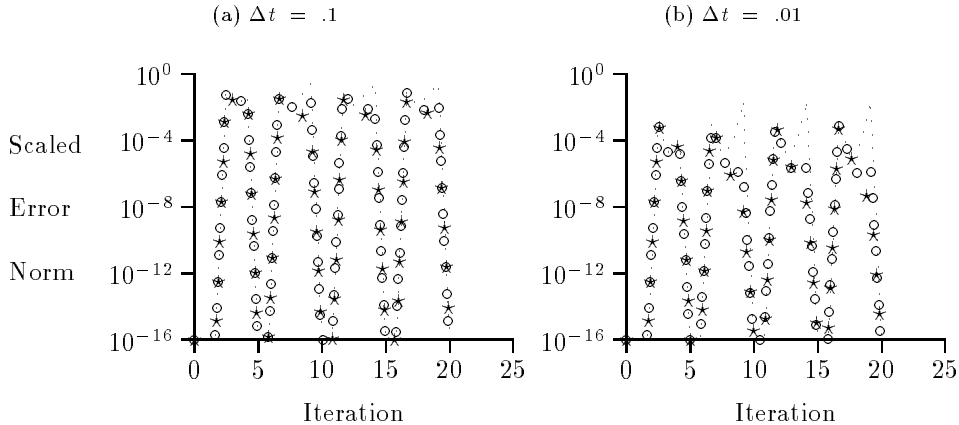
$k = 2$: dotted line, $k = 3$: o line, $k = 4$: x line
 FIG. 6. *Inferential Extrapolation for Two Dimensional Heat Problem Data*

We discretize (3.5) using a Backward Euler differencing scheme on a 10 point uniform spatial mesh. The discrete right hand side $b = 0$. Figure 3.1 contains the scaled errors for this example when $\Delta t \in \{.1, .01\}$. We find that $k(2 \times 10^{-4}) = 6$, and we make runs for $k \in \{2, 3, 4\}$. As can be seen, the right choice of k (e.g., $k = 4$) leads to quite good results, whereas a poor choice of k (e.g., $k = 2$) leads to poor results. When restarting (every 6 iterations), we only calculate $\min\{k, 3\}$ real solutions W_i before inferencing solutions again.

Our second example is a two dimensional heat flow problem on the spatial domain $\Omega = (0, \pi)^2$:

$$(3.6) \quad \begin{cases} w_t = w_{yy} + w_{zz} & \text{for } 0 \leq y, z \leq \pi, t \geq 0 \\ w(y, z, 0) = \phi(y)\phi(z) & \text{for } 0 \leq y, z \leq \pi \\ w(y, z, t) = 0 & \text{for } t > 0 \text{ and } (y, z) \in \partial\Omega. \end{cases}$$

We discretize (3.6) using a Backward Euler differencing scheme on a 10×10 uniform spatial mesh. The discrete right hand side $b = 0$. Figure 3.2 contains the scaled errors



$k = 2$: dotted line, $k = 3$: o line, $k = 4$: x line

FIG. 7. *Inferential Extrapolation for Two Dimensional Transport Problem Data*

for this example when $\Delta t \in \{.1, .01\}$. We find that $k(2 \times 10^{-4}) = 6$, and we make runs for $k \in \{2, 3, 4\}$. As can be seen, the right choice of k (e.g., $k = 4$) leads to quite good results, whereas a poor choice of k (e.g., $k = 2$) leads to poor results. As before we restart every 6 iterations.

Our third example is a two dimensional transport problem on the spatial domain $\Omega = (0, \pi)^2$:

$$(3.7) \quad \begin{cases} w_t = w_y + w_z & \text{for } 0 \leq y, z \leq \pi, t \geq 0 \\ w(y, z, 0) = \sin(2y) \sin(2z) & \text{for } 0 \leq y, z \leq \pi \\ w(y, z, t) = 0 & \text{for } t > 0 \text{ and } (y, z) \in \partial\Omega. \end{cases}$$

We discretize (3.7) using a Crank-Nicolson differencing scheme on a 10×10 uniform spatial mesh. The discrete right hand side $b = 0$. Figure 3.3 contains the scaled error for this example when $\Delta t \in \{.1, .01\}$. Here, we arbitrarily choose $k(\epsilon) = 6$ (the simplest heuristic for choosing $k(\epsilon)$), and we make runs for $k \in \{2, 3, 4\}$. As can be seen, the right choice of k (e.g., $k = 4$) leads to quite good results, whereas a poor choice of k (e.g., $k = 2$) leads to poor results. As before we restart every 6 iterations.

As expected, not all cases give good results. However, for appropriate choices of k , Δt , and how often to restart, the results are quite good. We note that for the three examples in this section, small values of k produce acceptable results.

Note: These experiments illustrate the table lookup aspect of our approach without any corrective postprocessing, e.g., a residual correction process (see §2.2).

4. Karmarkar's Algorithm. Karmarkar's algorithm[12] for solving the linear programming problem may be formulated in terms of an associative table. In particular, the time intensive part of the algorithm, which corresponds to solving a least squares problem, may be replaced by a single memory reference to an associative memory in which the table is stored.

Let the vector $a = (a_1, \dots, a_n)^T$ be a feasible solution to the linear programming problem

$$\min_x c^T x,$$

$$A x = b, \quad x \geq 0.$$

Let $D = \text{diag}(a_1, \dots, a_n)$. Karmarkar introduces the following additional constraint to the linear programming problem:

$$(x - a)^T D^{-2} (x - a) = 1.$$

Let b_1, \dots, b_n denote the columns of the matrix B , where $B = (AD)^T$. Then one step of the algorithm replaces a by \bar{a} where

$$(4.1) \quad \bar{a} - a = -D \frac{I - BB^+}{\|(I - BB^+)Dc\|} Dc.$$

Existence of the pseudo-inverse B^+ as well as the feasibility of \bar{a} is an intrinsic property of the algorithm. The time intensive part of this computation, namely, actually computing BB^+Dc , may be accelerated through use of an associative table. As the components of a are produced, compute the rows of AD . These rows are the columns of B , namely, $b_j, j = 1, \dots, n$. As the latter are produced, load the (signal, key) pairs $(b_j, b_j), j = 1, \dots, n$ into an associative memory. When the memory is loaded, the matrix

$$M = BB^+.$$

Now perform a table lookup. In particular, interrogate the memory with Dc . For the output, MDc , we have

$$MDc = BB^+Dc.$$

We conclude this section by attempting to quantify the savings by using the inferential technique instead of a standard implementation based on a least squares solution to (4.1). Let

$$NZ(A) \equiv \text{number of nonzeros of } A.$$

The following table summarizes the time for each of the operations required by (4.1):

Operation \ Implementation	Inferential	Standard
$B = AD$	$O(NZ(A) \cdot n)$	$O(NZ(A) \cdot n)$
$w = Dc$	$O(n)$	$O(n)$
$s = Mw$	$O(n)$	$O(n^q)$
$y = w - s, \quad \bar{a} = a - Dy/\ y\ $	$O(n)$	$O(n)$

Typically, $q \approx 3$. Hence, the inferential implementation reduces the cost from $O(n^q) + O(NZ(A) \cdot n) + O(n)$ to $O(NZ(A) \cdot n) + O(n)$. When A is dense, $NZ(A) = n^2$, so the two implementations require time proportional to each other. When A is sparse, the inferential implementation results in a very substantial reduction in the time. In fact, the following complexity result is immediate:

THEOREM 4.1. *Suppose the number of nonzeros in any row or column of A is bounded by a constant. Then, the inferential implementation only requires $O(n)$ time per iteration on a serial computer with an associative memory device attached.*

5. Inferential Multistep Methods. Here we show how associative tables can be used to furnish numerical approximations to solutions of systems of ordinary differential equations. The new methods are generalizations of linear multistep methods, one of the commonly used classes of numerical methods for solving differential equations.

The data furnished by a solution procedure is Hermite data, that is, the values and derivative values of the solution being sought at the successive steps of a mesh. The Hermite interpolation polynomial is fitted to this data, and the successor approximate value is obtained by extrapolation using the polynomial. The coefficients of the linear multistep methods are independent of the values and derivative values in question, and this accounts for the utility of this class of numerical methods.

Here we propose to replace the Hermite interpolation polynomial by a polynomial fitted to the data by least squares. The expected advantage is that more data can be used without increasing the degree of the polynomial, a property which suggests improved stability and accuracy. A second advantage is that the dependence of the approximating polynomial on the data causes the method to be data dependent; i.e., the coefficients change at each new point, a property which suggests improved accuracy also.

Adaptive fitting of a method to data is not a new idea. However, while in principle, adaptivity is advantageous, the advantage is bought at computational cost. It is here where associative tables come into play to virtually eliminate this cost. The Hermite data comprises a signal and the interpolatory basis a key. These (signal, key) pairs are loaded into an associative memory as the solution is developed. A single appropriate memory interrogation produces the next value of the numerical approximation, the memory is updated, and the process is repeated.

5.1. The Numerical Method. We consider the initial value problem for the p -dimensional system

$$u' = f(u), \quad 0 \leq t \leq 1.$$

We use the mesh $\{t_j = jh, j = 0, 1, \dots\}$ with mesh increment $h \leq 1$. Let

$$u_j \equiv u(jh) \text{ and } f_j \equiv f(u_j), \quad j = 0, 1, \dots$$

Let

$$U_n \equiv (u_{n-\ell+1}, \dots, u_n),$$

and

$$F(U_n) \equiv (f_{n-\ell+1}, \dots, f_n), \quad n = \ell - 1, \ell, \dots$$

Here ℓ is the method step number. The numerical approximation to u_n is denoted by $y_n, n = 0, 1, \dots$. Let

$$Y_n \equiv (y_{n-\ell+1}, \dots, y_n), \quad n = \ell - 1, \ell, \dots$$

U, F , and Y are $p \times \ell$ arrays. Let

$$Z \equiv Z_n = (Y_n, F(Y_n)), \quad n = \ell - 1, \ell, \dots,$$

where the notation here means that Y_n and $F(Y_n)$ are concatenated blockwise to form a $p \times 2\ell$ array.

Now we introduce a set of $(d + 1)$ -times differentiable functions which comprise an interpolating basis

$$\{\varphi_j(t), \quad j = 0, \dots, d\},$$

and the generalized Vandermonde matrix, a $(d + 1) \times \ell$ array,

$$\Phi \equiv \Phi_n^{\ell, d} \equiv [\varphi_{d-i}(t_j)], \quad 0 \leq i \leq d, \quad n - \ell + 1 \leq j \leq \ell.$$

An augmented generalized Vandermonde matrix is defined as the $(d + 1) \times 2\ell$ array,

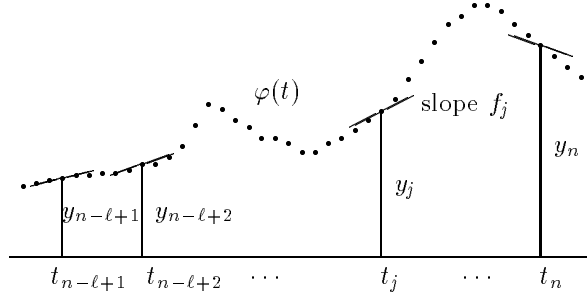
$$W \equiv W_n \equiv W_n^{\ell, d} \equiv [\Phi, \Phi'],$$

where Φ' denotes the derivative of Φ . Finally, we introduce the generalized moment vector

$$m_d \equiv m_n^d = (\varphi_d(t_n), \varphi_{d-1}(t_n), \dots, \varphi_0(t_n))^T, \quad n = 1, 2, \dots$$

The least squares problem

The Hermite least squares problem consists of fitting a function in the basis $\{\varphi_{d-i}(t)\}_{i=0}^d$ to the data Z_n at the points $\{t_j\}_{j=n-\ell+1}^n$. Let the coefficients of the interpolant be $\{a_j\}_{j=0}^d$, where each a_j is a p -vector,



Let $a = (a_0, \dots, a_d)$. Denote the interpolant by

$$\varphi(t) = \sum_{j=0}^d a_j \varphi_j(t).$$

$\varphi(t)$ is a p -vector valued function of t . Then the $p \times (d + 1)$ array a is defined as the solution of the least squares problem

$$\min_a \|Z - aW\|.$$

The solution of this problem is

$$a = ZW^+.$$

The extrapolation which defines the next approximation, namely, y at t_{n+1} is

$$y(t_{n+1}) \doteq \varphi(t_{n+1}) = ZW^+ m_{n+1}.$$

a is the memory matrix heretofore called M . Indeed setting

$$M \equiv M_n \equiv M_n^{\ell, d} = a,$$

we have

$$y(t_{n+1}) = M_n m_{n+1},$$

so that $y(t_{n+1})$ is specified by accessing the memory with m_{n+1} . With y_{n+1} determined, $f_{n+1} = f(y_{n+1})$ is computed and the memory is updated by inserting this new data and deleting the oldest data.

(Signal, key) pairs

The signals are the p -vectors comprising the columns of Z while the corresponding keys are the $(d + 1)$ -vectors comprising the columns of W . In particular, at the stage of the process where y_{n+1} is to be determined, these pairs are

$$(y_{n-j}, (\varphi_d(t_{n-j}), \dots, \varphi_0(t_{n-j}))^T)$$

and

$$(f_{n-j}, (\varphi'_d(t_{n-j}), \dots, \varphi'_0(t_{n-j}))^T), \quad j = \ell, \ell - 1, \dots, 0.$$

5.2. Stability and Error Analysis. We impose certain regularity properties on the initial value problem.

HYPOTHESIS 5.1. *We assume that there exists a constant \bar{u} so that*

$$\max_{0 \leq t \leq 1} \|u(t)\| \leq \bar{u}.$$

We also assume that the gradient, ∇f is bounded. Namely, that

$$\|\nabla f(\xi)\| \leq 1 \quad \text{for} \quad \|\xi\| \leq \bar{x}.$$

Here we normalized (by scaling t) so that this bound is unity, as indicated. This normalization along with \bar{x} is specified in the proof to follow. Note that the inequality involving \bar{x} is a mesh restriction.

Since the interpolatory basis functions are differentiable, the extrapolation operator has bounded norm. In particular

$$\|W^+ m_{n+1}\| \leq 1 + \gamma h,$$

where the constant $\gamma > 0$, depends on the basis.

Let

$$e_n = y_n - u_n, \quad n = 0, 1, \dots$$

Now for $n \geq \ell - 1$, we have

$$e_{n+1} = T_1 + T_2,$$

where

$$T_1 = [(Y_n, F(Y_n)) - (U_n, F(U_n))] W_n^+ m_{n+1},$$

and

$$T_2 = (U_n, F(U_n)) W_m^+ m_{n+1} - u_{n+1}.$$

T_2 is the local truncation error, and we turn now to estimating it. To begin, consider the scalar case and least squares approximation with point data (i.e., Lagrangian data in lieu of Hermite data).

Let $g(x)$ be defined on the interval $[0, \ell]$ and have $d + 1$ derivatives there. Apply Taylor's theorem with remainder to $g(x)$. We get

$$g(x) = t_d(x) + R_d(x),$$

where $t_d(x)$ is the Taylor polynomial of degree d , say, with respect to the point $x = 0$.

Let $p_d(x)$ be the least squares polynomial of degree d corresponding to the data $g(k)$, $k = 0, 1, \dots, \ell - 1$. We require $d \leq \ell$ for uniqueness of $p_d(x)$. Let \mathcal{P}_d be the operator mapping $g(x)$ into $p_d(x)$:

$$p_d(x) = \mathcal{P}_d[g(x)].$$

Note that \mathcal{P}_d is a linear operator, corresponding to solving the linear system of normal equations. Then

$$\begin{aligned} p_d(x) &= \mathcal{P}_d[t_d(x) + R_d(x)] \\ &= \mathcal{P}_d[t_d(x)] + \mathcal{P}_d[R_d(x)] \\ &= t_d(x) + \mathcal{P}_d[R_d(x)], \end{aligned}$$

since \mathcal{P}_d is the identity operator on polynomials of degree at most d . Then

$$\begin{aligned} p_d(x) - g(x) &= \mathcal{P}_d[g(x)] - t_d(x) - R_d(x) \\ &= \mathcal{P}_d[t_d(x) + R_d(x)] - t_d(x) - R_d(x) \\ &= \mathcal{P}_d[R_d(x)] - R_d(x). \end{aligned}$$

Now for the remainder we have

$$R_d[g](x) = \int_0^\ell K(x, \theta) g^{(d+1)}(\theta) d\theta,$$

where K is the Peano kernel. Then

$$\mathcal{P}_d[R_d] - R_d = \int_0^\ell \{\mathcal{P}_d[K(x, \theta)] - K(x, \theta)\} g^{(d+1)}(\theta) d\theta.$$

Then

$$\|\mathcal{P}_d[R_d] - R_d\|_\infty \leq k_d \max_{0 \leq \theta \leq \ell} |g^{(d+1)}(\theta)|,$$

where

$$k_d = \max_{0 \leq x, \theta \leq \ell} |\mathcal{P}_d[K(x, \theta)] - K(x, \theta)|.$$

Now scale the problem so that $[0, \ell]$ becomes $[0, \ell h]$. Then $g^{(d)}(x)$ becomes $h^d g^{(d)}(x)$, and we have derived the following extrapolation estimate.

$$\|\mathcal{P}_d[R_d] - R_d\|_\infty \leq k_d h^{d+1} \max_{0 \leq \theta \leq \ell h} |g^{(d+1)}(\theta)|.$$

We note that doubling the quantity of data, as in the Hermite case will increase the bound on d from $d \leq \ell$ to $d \leq 2\ell$. We also note that the order of the estimates is not affected by vectorizing this framework, that is by letting $g(x)$, $t_d(x)$, \mathcal{P}_d , and R_d be vector valued.

Thus we have the following bound for T_2 :

$$\|T_2\| \leq c_2 h^r,$$

where $r \leq \min(d+1, 2\ell+1)$. Note that c_2 depends on

$$\max_{0 \leq \theta \leq \ell h} |g^{(d+1)}(\theta)|.$$

We continue the error analysis by induction. We suppose that there exists a positive constant α , and positive constants c_1 and \bar{x} to be specified so that for $j \leq n$,

- (i) $\|y_j - u_j\| \leq c_1(1 + \alpha h)^j h^{r-1}$
- (ii) $\|y_j\| \leq \frac{1}{2}\bar{x}$

Here $n \geq \ell$, so we assume that the starting values, $y_0, \dots, y_{\ell-1}$, which are determined by a separate process, satisfy these hypotheses. Write T_1 as

$$T_1 = (Y_n - U_n, F(Y_n) - F(U_n)) W_n^+ m_{n+1}.$$

Then

$$\begin{aligned} (Y_n - U_n, F(Y_n) - F(U_n)) &= (y_{n-\ell+1} - u_{n-\ell+1}, \dots, y_n - u_n, \\ &\quad \nabla f(\xi_{n-\ell+1})(y_{n-\ell+1} - u_{n-\ell+1}), \dots, \nabla f(\xi_n)(y_n - u_n)). \end{aligned}$$

By the mean value theorem and the induction hypothesis,

$$\begin{aligned} \|\xi_j\| &\leq \|y_j\| + \|y_j - u_j\| \\ &\leq \frac{1}{2}\bar{x} + c_1(1 + \alpha h)^j h^{r-1}, \\ &\leq \frac{1}{2}\bar{x} + c_1 e^\alpha h^{r-1}, \end{aligned}$$

since $j \leq n$ and $jh \leq 1$. By taking $h < 1$ sufficiently small, we may arrange that $c_1 e^\alpha h^{r-1} < \frac{1}{2}\bar{x}$. Then

$$\|\xi_j\| \leq \bar{x}, \quad j \leq n.$$

Then

$$\begin{aligned} \|T_1\| &\leq c_1(1 + \alpha h)^n h^{r-1} \|W_n^+ m_{n+1}\| \\ &\leq c_1(1 + \alpha h)^n (1 + \gamma h) h^{r-1}. \end{aligned}$$

Combining the estimates for $\|T_1\|$ and $\|T_2\|$, we have

$$\begin{aligned}\|e_{n+1}\| &\leq c_1(1+\alpha h)^n(1+\gamma h)h^{r-1} + c_2h^r \\ &= c_1(1+\alpha h)^nh^{r-1} \left[1 + h \left(\gamma + \frac{1}{(1+\alpha h)^n} \frac{c_2}{c_1} \right) \right].\end{aligned}$$

In order to complete the induction, we seek to make

$$\gamma + \frac{1}{(1+\gamma h)^n} \frac{c_2}{c_1} \leq \alpha,$$

or

$$\frac{c_2}{c_1} \leq (\alpha - \gamma)(1 + \alpha h)^n.$$

This is the case when both

$$\alpha > \gamma \text{ and } \frac{c_2}{c_1} \leq (\alpha - \gamma),$$

which is assured when

$$\alpha > \gamma + c_2/c_1.$$

(Note that both α and c_2 depend on the d -th derivative on u .) In this case

$$\|e_{n+1}\| \leq c_1(1+\alpha h)^{n+1}h^{r-1},$$

completing the induction (i). Using this estimate, we have

$$\begin{aligned}\|y_{n+1}\| &\leq \|u_{n+1}\| + c_1(1+\gamma h)^{n+1}h^{r-1} \\ &\leq \bar{u} + c_1e^\alpha h^{r-1}.\end{aligned}$$

This is less than $\frac{1}{2}\bar{x}$ if \bar{x} is chosen so that

$$\bar{x} > 2(\bar{u} + c_1e^\alpha h^{r-1}),$$

completing the induction (ii).

With the induction completed we note that for all $jh \leq 1$,

$$\begin{aligned}\|e_j\| &\leq c_1(1+\gamma h)^jh^{r-1} \\ &\leq c_1e^\alpha h^{r-1}.\end{aligned}$$

We summarize these considerations in the following theorem:

THEOREM 5.1. *Under the regularity conditions stated in Hypothesis 5.1 and the condition $d < 2l$, the class of inferential multistep methods are stable and convergent to order $r \leq d$.*

5.3. Numerical Experiments. We apply the numerical methods in two cases, $p = 1$ and $p=2$. For $p = 1$, we treat the initial value problem

$$\dot{u} = \lambda u + \beta(1 - \lambda)e^t, \quad u(0) = \alpha + \beta.$$

For $p = 2$, we treat the equation $\ddot{x} + \lambda^2 x = \lambda^2 \sin t$ which corresponds to the system

$$\dot{u} = \begin{pmatrix} 0 & 1 \\ \lambda^2 & 0 \end{pmatrix} u + \lambda^2 \begin{pmatrix} 0 \\ \sin t \end{pmatrix},$$

with the initial condition

$$u(0) = (\beta, \alpha\lambda + \lambda^2/(\lambda^2 - 1))^T.$$

We employ the monomial basis $\varphi_j(t) = t^j$, $j = 0, 1, \dots$. The generalized Vandermonde matrix becomes the conventional Vandermonde matrix in this case:

$$\Phi = (t_j^{d-i}), \quad 0 \leq i \leq d, \quad 1 \leq j \leq \ell.$$

Introduce the matrix

$$D = \begin{pmatrix} 0 & d & & & \\ 0 & 0 & d-1 & & \\ & & & \ddots & \\ & & & & 1 \\ & & & & 0 \end{pmatrix}.$$

Then the augmented Vandermonde matrix is $[\Phi, D\Phi]$ and the generalized moment vector is

$$m_n = (t_n^d, t_n^{d-1}, \dots, t_n, 1)^T.$$

At each mesh point we compute the Euclidean norms $\|e_n\|$ and $\|u_n\|$, and we tabulate the normalized global error

$$\epsilon = \left[\frac{\sum_{n=0}^{h-1} \|e_n\|}{\sum_{n=0}^{h-1} \|u_n\|} \right]^{\frac{1}{2}}$$

in Table 5.1. The quality of the results varies in the expected manner with respect to changes in λ , h , and d . As expected, not all cases give good results. However, for h small compared to λ , the results are quite good. If h is too small, instabilities entering into our simulation of the computation of W^+ accumulate to mask the asymptotic behavior of the error. It is likely that more care in computing the least squares fit in the simulation would lead to improved results. Some divergent cases are shown for completeness.

6. Current Technology and Parameter Requirements. Associative memories now under construction limit the number of (signal, key) pairs that are stored to $k \leq N$, where N is the dimension of these pairs [9].

Associative memories are typically used only for pattern matching. Kohonen (see [13]) suggests a holographic implementation and includes a picture of such a device.

TABLE 1
Inferential Multistep Numerical Experiments

case $p = 1, \alpha = \beta = 1, \ell = 5$

		$d = 2$		
$\lambda \setminus h$.1	.01	.001
1		2.36×10^{-3}	7.35×10^{-4}	3.52×10^{-7}
-1		7.20×10^{-4}	7.74×10^{-6}	7.78×10^{-8}
-10		3.25×10^{-2}	3.41×10^{-4}	4.13×10^{-6}
-100		5.24×10^4	1.36×10^{-2}	1.13×10^{-4}
		$d = 3$		
$\lambda \setminus h$.1	.01	.001
1		1.71×10^{-4}	2.50×10^{-7}	3.06×10^{-4}
-1		1.07×10^{-4}	1.38×10^{-7}	1.55×10^{-4}
-10		6.59×10^{-2}	2.58×10^{-5}	1.16×10^{-4}
-100		6.35×10^5	1.32×10^6	2.94×10^{-5}

case $p = 2, \alpha = \beta = 1, \ell = 5$

		$d = 2$		
$\lambda \setminus h$.1	.01	.001
2		1.76×10^{-1}	2.16×10^{-3}	2.19×10^{-5}
10		1.38×10^{-1}	1.80×10^{-3}	3.75×10^{-4}
		$d = 3$		
$\lambda \setminus h$.1	.01	.001
2		2.70×10^1	6.20×10^{-2}	6.63×10^{-4}
10		3.61×10^1	3.91×10^{-2}	1.62×10^{-3}

TABLE 2
Summary of Parameter Values and Restrictions

Method	(signal,key) pairs	Dimension of pairs	Theoretical restrictions	Parameter values used in experiments
Inferential solve	(u_j, v_j)	(N, N)	$k \leq N$	$(k, N)=(15-20,100)$
Inferential extrapolation	(u_{j+1}, v_j)	(N, N)	$k \leq N$	$(k, N)=(2-4,100)$
Karmarkar	(b_j, b_j)	(n, n)	$k = n$	none
Inferential multistep	$((y_j, (\varphi_d(t_j), \dots, \varphi_0(t_j))^T),$ $(f_j, (\varphi'_d(t_j), \dots, \varphi'_0(t_j))^T))$	$(p, d + 1)$	$d \leq 2\ell$	$(p, d, \ell) = (1-2, 2-3, 5)$
Noisy sort	$(\tau(x_j), x_j)$	(N, N)	$N!$	$10 \leq N \leq 100$

A current trend is to implement them using a neural network. Unfortunately, neural networks are notoriously slow. A typical neural network is not real hardware, but a software approximation on a conventional digital computer. Connection machines and Intel hypercubes appear to be the current favorite simulators. These machines are much too slow to be taken seriously in this context because of their inadequate communication speeds (in this context).

At Caltech, Psaltis has constructed a three dimensional holographic memory with 10^{10} bits and a cycle time of 10^{-4} seconds. This device can be modified so that the cycle time is lower, but at the expense of the memory size.

A description of recent digital-analog devices is described in [6] and [10]. It is interesting to note that in 1986, several hundred neurons could be placed on a single CMOS chip. Yet in 1989, 100,000 neurons could be placed on a similarly sized chip. In 1990, another five to ten fold increase in neurons per chip is expected [9]. The density of these chips follows behind standard memory chips by roughly one generation.

A description of an associative memory using high density neural network chips is contained in [5]. Loading the memory is equivalent to writing data into conventional VLSI memory. The query time is negligible, and getting data back is equivalent to reading data from conventional memory. Due to the high density of neurons, and their analog nature, cycle times for the network is in the low nanosecond range, which is many orders of magnitude faster than a typical software simulator on a parallel digital computer.

Table 6.1 contains a summary of the requirements for inferential solve, inferential extrapolation, Karmarkar's algorithm, inferential multistep, and noisy sort (see [8]). Except for the latter two methods, the theoretical bound for k is N . For the inferential multistep method, the number of pairs must be larger than half of the problem dimension, but may be less than N . For noisy sort, the number of pairs is far greater than N . However, theory enables us to compute the memory matrix in closed form.

Acknowledgements. We are grateful to Daniel Miranker for bringing the parallel computational possibilities of associative processing to our attention. We are also grateful to Charles Micchelli and Martin Schultz for helpful suggestions.

REFERENCES

- [1] J. A. ANDERSON AND E. ROSENFELD, *Neurocomputing, Foundations of Research*, The MIT Press, Cambridge, MA, 1985.
- [2] R. E. BANK AND C. C. DOUGLAS, *An efficient implementation of the SSOR and ILU preconditionings*, *Appl. Numer. Math.*, 1 (1985), pp. 489–492.
- [3] L. BLUM, M. SHUB, AND S. SMALE, *On a theory of computation and complexity over the real numbers, NP-completeness, recursive functions and universal machines*, *Bulletin AMS*, 21 (1989), pp. 1–46.
- [4] C. C. DOUGLAS AND W. L. MIRANKER, *Numerical computation using associative tables*, Tech. Report 14660, IBM Research Division, Yorktown Heights, New York, 1989.
- [5] H. P. GRAF AND P. DEVEGVAR, *A CMOS implementation of a neural network model*, in *Advanced Research in VLSI*, P. Lasleben, ed., The MIT Press, Cambridge, MA, 1987.
- [6] H. P. GRAF, L. D. JACKEL, R. E. HOWARD, B. STRAUGHN, J. S. DENKER, W. HUBBARD, D. M. TENNANT, AND D. SCHWARTZ, *VLSI implementation of a neural network memory with several hundreds of neurons*, in *Proceedings of Conference on Neural Networks for Computing*, Snowbird, Utah, J. S. Denker, ed., American Institute of Physics, 1986, pp. 182–187.
- [7] T. N. E. GREVILLE, *Some applications of the pseudoinverse of a matrix*, *SIAM Review*, 2 (1960), pp. 15–22.
- [8] S. HORIGUCHI AND W. L. MIRANKER, *Noisy sort, a memory-intensive sorting algorithm*, *Linear Algebra & Its Applications*, (1989), pp. 641–658.
- [9] L. D. JACKEL, 1989. private communication.
- [10] L. D. JACKEL, H. P. GRAF, AND R. E. HOWARD, *Electronic neural network chips*, *Applied Optics*, 26 (1987), pp. 5077–5080.
- [11] R. E. KALMAN, *Fundamental study of adaptive control systems*, ASTIA AD 8 28 73, (1962).
- [12] N. KARMARKAR, *A new polynomial time algorithm for linear programming*, *Combinatorica*, 4 (1984), pp. 373–395.
- [13] T. KOHONEN, *Associative Memory: A System-Theoretical Approach*, Springer-Verlag, New York, 1977.
- [14] ———, *Self-Organization and Associative Memory*, Springer-Verlag, New York, 1987.
- [15] C. MEAD, *Analog VLSI and Neural Systems*, Addison-Wesley, Reading, MA, 1989.
- [16] C. MOLER, J. LITTLE, S. BANGERT, AND S. KLEIMAN, *MATLAB User's Guide*, The MathWorks, Inc., Sherborn, MA, 1987.
- [17] D. PSALTIS, 1990. private communication.
- [18] R. D. RICHTMYER AND K. W. MORTON, *Difference Methods for Initial-Value Problems*, Interscience Publishers (J. Wiley & Sons), New York, 1967.
- [19] D. E. RUMMELHARDT AND J. L. MCCLELLAND, *Parallel Distributed Processing*, The MIT Press, Cambridge, MA, 1986.
- [20] D. A. SMITH, W. F. FORD, AND A. SIDI, *Extrapolation methods for vector sequences*, *SIAM Review*, 29 (1987), pp. 199–233.
- [21] G. W. STEWART, *Introduction to Matrix Computation*, Academic Press, New York, 1973.
- [22] G. STRANG AND G. J. FIX, *An Analysis of the Finite Element Method*, Prentice-Hall, New York, 1973.
- [23] R. S. VARGA, *Matrix Iterative Analysis*, Prentice-Hall, New York, 1962.
- [24] D. M. YOUNG, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.